



PACKET ARCHITECTS AB

**Ethernet Switch/Router
Datacenter ToR 32x100G
User Guide**

Core Revision unknown
Datasheet Revision unknown
March 29, 2024 © Packet Architects AB.

Contents

1	Overview	19
1.1	Feature Overview	20
1.2	Port Numbering Table	24
2	Packet Decoder	25
2.1	Decoding Sequence	25
3	Packet Processing	31
3.1	Ingress Packet Processing	31
3.2	Egress Packet Processing	34
4	Latency and Jitter	35
4.1	Latency	35
4.2	Jitter	35
5	VLAN Processing	37
5.1	Assignment of Ingress VID	37
5.1.1	VID Assignment from Packet Fields	37
5.1.2	Force Ingress VID from Ingress Configurable ACL	38
5.2	VLAN membership	38
5.3	VLAN operations	38
5.3.1	Default VLAN Header	39
5.3.2	Source Port VLAN Operation	39
5.3.3	Operation Based On Incoming Packets Number of VLANs	40
5.3.4	Configurable ACL VLAN Swap Operation	40
5.3.5	VLAN Table Operation	40
5.3.6	VLAN Table VID Operation Based On the Packets Number of VLANs	40
5.3.7	Egress Port VLAN Operation	40
5.3.8	Egress Port VID Operation	40
5.3.9	Egress Vlan Translation	40
5.3.10	Priority Tagged Packets	41
5.3.11	Router VLAN Operations	41
5.3.12	VLAN Operation Order	41
5.3.13	VLAN Operation Examples	41
5.3.14	VLAN Reassembly	42
6	Switching	45
6.1	L2 Destination Lookup	45
6.2	Software Interaction	46
6.3	L2 Action Table	46
6.3.1	Learning Unicast and Learning Multicast	47
6.3.2	Drop and Learning	47
6.3.3	Priorities Between Actions	47
6.3.4	Using L2 Action Table for 802.1X	48
7	Routing	49
7.1	Order of Operation	49

8	Tunneling	53
8.1	Packet Decoder For Tunnel Exit	53
8.2	Tunnel Exit	54
8.2.1	To Not To Use Second Lookup	55
8.2.2	Use Second Lookup With Packet Data	55
8.2.3	How To Remove Data From Packet In A Tunnel Exit	55
8.2.4	Packet Insertion and Removal Limits	56
8.2.5	Tunnel Exit Options	56
8.2.6	Tunnel Exit from Tables	56
8.3	Tunnel Entry	56
8.3.1	Tunnel Length Insertion	57
8.3.2	Tunnel Entry Tables	58
8.3.3	Priority between Tunnel Exit and Tunnel Entry in Tables	58
8.3.4	Tunnel Entry and Routing with MTU check	59
9	MPLS	61
9.1	MPLS Header Operations	61
9.2	MPLS Penultimate Pop	61
9.3	MPLS Header Insertion To Reach Next Hop	61
10	NAT - Network Address Translation	63
10.1	Ingress Packet Processing Option	63
10.2	NAT Action Table Check	63
11	Mirroring	65
11.1	Input Mirroring	65
11.2	Output Mirroring	65
11.2.1	Requeueing FIFO	66
12	Link Aggregation	67
12.0.1	One-to-one Port Mapping	67
12.1	Example	67
12.2	Hash Calculation	69
13	IEEE 1588/PTP Support	71
13.1	Timestamp from RX MAC	71
13.1.1	Timestamp to the CPU	71
13.2	PTP Frame Decoding	71
13.2.1	PTP over 802.3 Ethernet	72
13.2.2	PTP over UDP	72
13.3	Software Control of TX MAC PTP Actions	72
13.3.1	Packet Updates by the Transmit MAC	73
13.4	Support for Ordinary Clock	73
13.4.1	Master sending Sync	73
13.4.2	Slave receiving Sync	73
13.4.3	Slave sending DelayReq	74
13.4.4	Master receiving DelayReq	74
13.4.5	Master sending DelayReply	74
13.4.6	Slave receiving DelayReply	74
13.5	Support for 1-step Peer to Peer	74
13.5.1	Initiator sending PDelayReq	74
13.5.2	Peer receiving PDelayReq	74
13.5.3	Peer sending PDelayResp	74
13.5.4	Initiator receiving PDelayResp	74
14	Classification	75
14.1	L2 Classification	75
14.2	Configurable Ingress ACL Engine	75



14.2.1	Field Selection	75
14.2.2	Example Of Selecting Fields For Configurable Ingress ACL Table 0	79
14.2.3	Example Of Selecting Fields For Configurable Ingress ACL Table 1	82
14.2.4	Example Of Selecting Fields For Configurable Ingress ACL Table 2	86
14.2.5	Example Of Selecting Fields For Configurable Ingress ACL Table 3	89
14.2.6	ACL Search	89
14.2.7	ACL Actions	89
14.3	Multiple ACL Lookups	89
14.3.1	Multiple Actions	90
14.3.2	Default Port ACL action	91
14.4	Configurable Egress ACL Engine	91
14.4.1	Field Selection	92
14.4.2	Example Of Selecting Fields For Configurable Egress ACL Table 0	94
14.4.3	Example Of Selecting Fields For Configurable Egress ACL Table 1	97
14.4.4	ACL Search	98
14.4.5	ACL Actions	99
14.5	Multiple ACL Lookups	99
14.5.1	Multiple Actions	99
15	VLAN and Packet Type Filtering	101
16	Hashing	103
16.1	Hashing Functions	103
16.1.1	MAC Table Hashing	103
16.1.2	IP Table Hashing	104
16.1.3	MPLS Table Hashing	106
16.1.4	Hash function for Ingress Configurable ACL 0	107
16.1.5	Hash function for Ingress Configurable ACL 1	111
16.1.6	Hash function for Ingress Configurable ACL 2	114
16.1.7	Hash function for Ingress Configurable ACL 3	114
16.1.8	Hash function for Egress Configurable ACL 0	114
16.1.9	Hash function for Egress Configurable ACL 1	116
16.1.10	Hash function for Tunneling	116
17	D-left Lookup	119
17.1	Functions using D-left	119
17.1.1	Egress VLAN Translation	119
17.1.2	Ingress Configurable ACL	119
17.1.3	Egress Configurable ACL	121
17.1.4	Tunnel Exit	121
18	Learning and Aging	123
18.1	L2 Forwarding Information Base (FIB)	123
18.1.1	Tables for MAC DA lookup	123
18.1.2	Tables for MAC SA lookup	124
18.1.3	Status Tables	124
18.1.4	Hash Collision Accommodation	125
18.2	Hardware Learning and Aging	125
18.2.1	Learning Unit	125
18.2.2	Hardware Learning Exceptions	126
18.2.3	Aging Unit	127
18.2.4	MAC DA Hit Update Unit	127
18.3	Software Learning and Aging	127
18.3.1	Injection of Learning Packets	128
18.3.2	Direct Access to FIB	129
18.3.3	Software Reserved Entry	129
18.3.4	Software Aging	129



18.4	Software And Hardware Interaction	129
18.4.1	Data FIFO Interrupts	130
18.4.2	Writeback Bus Control	130
19	Spanning Tree	131
19.1	Spanning Tree	131
19.2	Multiple Spanning Tree	131
19.3	Spanning Tree Drop Counters	132
20	Token Bucket	133
21	Egress Queues and Scheduling	135
21.1	Determine Egress Queue	135
21.2	Determine a packets outgoing QoS headers PCP, DEI and TOS fields	137
21.2.1	Remap Egress Queue to Packet Headers	137
21.2.2	Using Packet Type, Destination Port and Switching/Routing to do QoS Mappings	138
21.3	Priority Mapping	138
21.4	Shapers	138
21.4.1	Queue Shaper	139
21.4.2	Prio Shaper	139
21.5	Scheduling	141
21.6	DWRR Scheduler	141
21.7	Queue Management	141
21.8	How To Make Sure A Port Is Empty	142
22	Packet Coloring	143
22.1	Ingress Packet Initial Coloring	143
22.2	Remap Packet Color to Packet Headers	145
23	Admission Control	147
23.1	Ingress Admission Control	147
23.1.1	Traffic Groups	147
23.2	Meter-Marker-Policer	148
24	Tick	151
25	Multicast Broadcast Storm Control	153
25.1	Inspected Traffic	153
25.2	Rate Configuration	154
26	Egress Resource Manager	157
26.1	Yellow Zone	157
26.2	Red Zone	158
26.3	Green Zone	158
26.4	Configuration Example	158
26.5	Restrictions	158
27	Flow Control	159
27.1	Pausing	159
27.2	Tail-Drop	159
27.2.1	Tail-drop as police for Pausing	159
27.3	Buffer partitioning	160
27.3.1	Reserves	160
27.4	Non-PFC mode	160
27.5	PFC-mode	160
27.5.1	Pausing Thresholds	161
27.5.2	Tail-drop Thresholds	162
27.6	Enabling Tail-Drop	162



27.7	Enabling Pausing	162
27.8	Multiple Switch Slices	162
27.9	Dropped packets	162
27.10	Reconfiguration	163
27.11	Debug Features	163
28	Egress Port Shaper	165
29	Statistics	167
29.1	Packet Processing Pipeline Drops	169
29.2	ACL Statistics	169
29.3	SMON Statistics	169
29.4	Routing Statistics	170
29.5	Ingress Port Receive Statistics	170
29.6	Packet Datapath Statistics	170
29.7	Miscellaneous Statistics	170
29.8	Debug Statistics	171
29.8.1	Debug Statistics Accuracy	171
30	Packets To And From The CPU	173
30.1	Packets From the CPU	173
30.1.1	Identify the From CPU Tag	174
30.1.2	From CPU Header and Packet Modification and Operations	174
30.2	Packets To the CPU	175
30.3	To CPU Header format	176
30.3.1	To CPU Header in IETF format	178
30.3.2	Packet Type Table	179
30.3.3	Reason Table	180
30.3.4	Reason Code Operations	181
31	Core Interface Description	183
31.1	Clock, Reset and Initialization interface	183
31.1.1	Assert Reset	184
31.2	Packet Interface	184
31.3	Configuration Interface	187
31.4	Interrupt Interface	188
31.5	Pause Interfaces	189
31.5.1	PFC Status	189
31.5.2	External Pause	189
31.6	Debug Read Interface	189
31.7	Debug Write Interface	210
32	Configuration Interface	211
32.1	Response time	211
32.2	Out of range accesses	211
32.3	Atomic Wide Access	211
32.4	Accumulator Accesses	212
33	Debugging the Design	213
33.1	Debug Counters in Ingress Packet Processing	213
33.2	Debug Counters in Egress Packet Processing	216
34	Implementation	219
34.1	Floorplanning	219
34.1.1	Pipelining	219
34.1.2	Configuration and debug	220
34.2	Clock crossings	220
34.2.1	IPP and EPP Structure	220



34.3	Memory wrappers	220
34.4	Dual ported memories	223
34.5	Memory timing	223
34.6	Lint set up	224
34.6.1	Waivers	224
35	Registers and Tables	225
35.1	Address Space For Tables and Registers	233
35.2	Byte Order	233
35.3	Register Banks	234
35.4	Registers and Tables in Alphabetical Order	268
35.5	Active Queue Manager	298
35.5.1	ERM Red Configuration	298
35.5.2	ERM Yellow Configuration	299
35.5.3	Egress Resource Manager Pointer	300
35.5.4	Resource Limiter Set	300
35.6	Core Information	301
35.6.1	Core Version	301
35.7	Egress Packet Processing	301
35.7.1	Beginning of Packet Tunnel Entry Instruction Table	301
35.7.2	Color Remap From Egress Port	302
35.7.3	Color Remap From Ingress Admission Control	303
35.7.4	Debug Counter debugMatchEPP0 Setup	303
35.7.5	Debug Counter fromPort Setup	304
35.7.6	Debug Counter reQueuePortId Setup	304
35.7.7	Disable CPU tag on CPU Port	305
35.7.8	Drain Port	305
35.7.9	EPP Debug addNewMpls	306
35.7.10	EPP Debug debugMatchEPP0	306
35.7.11	EPP Debug delSpecificVlan	306
35.7.12	EPP Debug fromPort	307
35.7.13	EPP Debug imActive	307
35.7.14	EPP Debug imExtra	307
35.7.15	EPP Debug isIPv4	308
35.7.16	EPP Debug isIPv6	308
35.7.17	EPP Debug isPPPoE	308
35.7.18	EPP Debug omEnabled	309
35.7.19	EPP Debug omImActive	309
35.7.20	EPP Debug reQueue	309
35.7.21	EPP Debug reQueuePkt	310
35.7.22	EPP Debug reQueuePortId	310
35.7.23	EPP Debug updateTosExp	310
35.7.24	Egress Ethernet Type for VLAN tag	311
35.7.25	Egress MPLS Decoding Options	311
35.7.26	Egress MPLS TTL Table	312
35.7.27	Egress Multiple Spanning Tree State	312
35.7.28	Egress NAT Operation	312
35.7.29	Egress Port Configuration	313
35.7.30	Egress Port VID Operation	316
35.7.31	Egress Queue To MPLS EXP Mapping Table	317
35.7.32	Egress Queue To PCP And CFI/DEI Mapping Table	318
35.7.33	Egress Router Table	318
35.7.34	Egress Tunnel Exit Table	318
35.7.35	Egress VLAN Translation TCAM	319
35.7.36	Egress VLAN Translation TCAM Answer	320
35.7.37	IP QoS Mapping Table	320
35.7.38	Ingress NAT Operation	321



35.7.39	L2 QoS Mapping Table	321
35.7.40	L2 Tunnel Entry Instruction Table	322
35.7.41	L3 Tunnel Entry Instruction Table	323
35.7.42	MPLS QoS Mapping Table	323
35.7.43	NAT Add Egress Port for NAT Calculation	324
35.7.44	Next Hop DA MAC	324
35.7.45	Next Hop MPLS Table	324
35.7.46	Next Hop Packet Insert MPLS Header	325
35.7.47	Output Mirroring Table	327
35.7.48	Router Port Egress SA MAC Address	327
35.7.49	Select Which Egress QoS Mapping Table To Use	328
35.7.50	TOS QoS Mapping Table	328
35.7.51	Tunnel Entry Header Data	329
35.7.52	Tunnel Entry Instruction Table	330
35.8	Flow Control	330
35.8.1	FFA Used PFC	330
35.8.2	FFA Used non-PFC	331
35.8.3	PFC Dec Counters for ingress ports 0 to 7	331
35.8.4	PFC Dec Counters for ingress ports 16 to 23	332
35.8.5	PFC Dec Counters for ingress ports 24 to 31	332
35.8.6	PFC Dec Counters for ingress ports 8 to 15	332
35.8.7	PFC Inc Counters for ingress ports 0 to 7	333
35.8.8	PFC Inc Counters for ingress ports 16 to 23	333
35.8.9	PFC Inc Counters for ingress ports 24 to 31	334
35.8.10	PFC Inc Counters for ingress ports 8 to 15	334
35.8.11	Port FFA Used	334
35.8.12	Port Pause Settings	335
35.8.13	Port Reserved	335
35.8.14	Port Tail-Drop FFA Threshold	336
35.8.15	Port Tail-Drop Settings	336
35.8.16	Port Used	337
35.8.17	Port Xoff FFA Threshold	337
35.8.18	Port Xon FFA Threshold	338
35.8.19	Port/TC Reserved	338
35.8.20	Port/TC Tail-Drop Total Threshold	338
35.8.21	Port/TC Xoff Total Threshold	339
35.8.22	Port/TC Xon Total Threshold	339
35.8.23	TC FFA Used	340
35.8.24	TC Tail-Drop FFA Threshold	340
35.8.25	TC Xoff FFA Threshold	341
35.8.26	TC Xon FFA Threshold	341
35.8.27	Tail-Drop FFA Threshold	342
35.8.28	Xoff FFA Threshold	342
35.8.29	Xon FFA Threshold	342
35.9	Global Configuration	343
35.9.1	Core Tick Configuration	343
35.9.2	Core Tick Select	343
35.9.3	MAC RX Maximum Packet Length	344
35.9.4	Scratch	344
35.10	Ingress Packet Processing	344
35.10.1	AH Header Packet Decoder Options	344
35.10.2	ARP Packet Decoder Options	345
35.10.3	Aging Data FIFO	345
35.10.4	Aging Data FIFO High Watermark Level	346
35.10.5	Allow Special Frame Check For L2 Action Table	346
35.10.6	BOOTP and DHCP Packet Decoder Options	348
35.10.7	CAPWAP Packet Decoder Options	349



35.10.8	CPU Reason Code Operation	349
35.10.9	Check IPv4 Header Checksum	350
35.10.10	DNS Packet Decoder Options	351
35.10.11	Debug Counter debugMatchIPPO Setup	351
35.10.12	Debug Counter dstPortmask Setup	352
35.10.13	Debug Counter finalVid Setup	352
35.10.14	Debug Counter l2DaHash Setup	352
35.10.15	Debug Counter l2DaHashHitAndBucket Setup	353
35.10.16	Debug Counter l2DaHashKey Setup	353
35.10.17	Debug Counter l2DaTcamHitsAndCast Setup	354
35.10.18	Debug Counter nextHopPtrFinal Setup	354
35.10.19	Debug Counter nextHopPtrHash Setup	355
35.10.20	Debug Counter nextHopPtrLpm Setup	355
35.10.21	Debug Counter nrVlans Setup	356
35.10.22	Debug Counter spVidOp Setup	356
35.10.23	Debug Counter srcPort Setup	357
35.10.24	Debug Counter vlanVidOp Setup	357
35.10.25	Default Packet To CPU Modification	357
35.10.26	ESP Header Packet Decoder Options	358
35.10.27	Egress ACL Rule Pointer TCAM	358
35.10.28	Egress ACL Rule Pointer TCAM Answer	359
35.10.29	Egress Configurable ACL 0 Large Table	360
35.10.30	Egress Configurable ACL 0 Rules Setup	361
35.10.31	Egress Configurable ACL 0 Search Mask	362
35.10.32	Egress Configurable ACL 0 Selection	362
35.10.33	Egress Configurable ACL 0 Small Table	363
35.10.34	Egress Configurable ACL 0 TCAM	364
35.10.35	Egress Configurable ACL 0 TCAM Answer	365
35.10.36	Egress Configurable ACL 1 Rules Setup	366
35.10.37	Egress Configurable ACL 1 TCAM	366
35.10.38	Egress Configurable ACL 1 TCAM Answer	367
35.10.39	Egress Port NAT State	368
35.10.40	Egress Spanning Tree State	368
35.10.41	Enable Enqueue To Ports And Queues	368
35.10.42	Flooding Action Send to Port	369
35.10.43	Force Non VLAN Packet To Specific Color	369
35.10.44	Force Non VLAN Packet To Specific Queue	370
35.10.45	Force Unknown L3 Packet To Specific Color	370
35.10.46	Force Unknown L3 Packet To Specific Egress Queue	371
35.10.47	Forward From CPU	371
35.10.48	GRE Packet Decoder Options	371
35.10.49	Hairpin Enable	372
35.10.50	Hardware Learning Configuration	372
35.10.51	Hardware Learning Counter	373
35.10.52	Hash Based L3 Routing Table	373
35.10.53	Hit Update Data FIFO	374
35.10.54	Hit Update Data FIFO High Watermark Level	375
35.10.55	IEEE 1588 L2 Packet Decoder Options	375
35.10.56	IEEE 1588 L4 Packet Decoder Options	376
35.10.57	IEEE 802.1X and EAPOL Packet Decoder Options	377
35.10.58	IKE Packet Decoder Options	377
35.10.59	IPP Debug debugMatchIPPO	378
35.10.60	IPP Debug doL2Lookup	378
35.10.61	IPP Debug dropPktAfterL2Decode	379
35.10.62	IPP Debug dropPktAfterL3Decode	379
35.10.63	IPP Debug dstPortmask	379
35.10.64	IPP Debug finalVid	380



35.10.65	IPP Debug isBroadcast	380
35.10.66	IPP Debug isFlooding	380
35.10.67	IPP Debug l2DaHash	381
35.10.68	IPP Debug l2DaHashHitAndBucket	381
35.10.69	IPP Debug l2DaHashKey	381
35.10.70	IPP Debug l2DaTcamHitsAndCast	382
35.10.71	IPP Debug nextHopPtrFinal	382
35.10.72	IPP Debug nextHopPtrHash	382
35.10.73	IPP Debug nextHopPtrHashHit	383
35.10.74	IPP Debug nextHopPtrLpm	383
35.10.75	IPP Debug nextHopPtrLpmHit	383
35.10.76	IPP Debug nrVlans	384
35.10.77	IPP Debug routed	384
35.10.78	IPP Debug routerHit	384
35.10.79	IPP Debug spVidOp	385
35.10.80	IPP Debug srcPort	385
35.10.81	IPP Debug vlanVidOp	385
35.10.82	IPv4 TOS Field To Egress Queue Mapping Table	386
35.10.83	IPv4 TOS Field To Packet Color Mapping Table	386
35.10.84	IPv6 Class of Service Field To Egress Queue Mapping Table	386
35.10.85	IPv6 Class of Service Field To Packet Color Mapping Table	387
35.10.86	Ingress Admission Control Current Status	387
35.10.87	Ingress Admission Control Initial Pointer	387
35.10.88	Ingress Admission Control Mark All Red	388
35.10.89	Ingress Admission Control Mark All Red Enable	388
35.10.90	Ingress Admission Control Reset	389
35.10.91	Ingress Admission Control Token Bucket Configuration	389
35.10.92	Ingress Configurable ACL 0 Large Table	390
35.10.93	Ingress Configurable ACL 0 Pre Lookup	393
35.10.94	Ingress Configurable ACL 0 Rules Setup	393
35.10.95	Ingress Configurable ACL 0 Search Mask	394
35.10.96	Ingress Configurable ACL 0 Selection	394
35.10.97	Ingress Configurable ACL 0 Small Table	395
35.10.98	Ingress Configurable ACL 0 TCAM	397
35.10.99	Ingress Configurable ACL 0 TCAM Answer	398
35.10.100	Ingress Configurable ACL 1 Large Table	400
35.10.101	Ingress Configurable ACL 1 Pre Lookup	404
35.10.102	Ingress Configurable ACL 1 Rules Setup	404
35.10.103	Ingress Configurable ACL 1 Search Mask	405
35.10.104	Ingress Configurable ACL 1 Selection	405
35.10.105	Ingress Configurable ACL 1 Small Table	406
35.10.106	Ingress Configurable ACL 1 TCAM	410
35.10.107	Ingress Configurable ACL 1 TCAM Answer	410
35.10.108	Ingress Configurable ACL 2 Pre Lookup	413
35.10.109	Ingress Configurable ACL 2 Rules Setup	414
35.10.110	Ingress Configurable ACL 2 TCAM	414
35.10.111	Ingress Configurable ACL 2 TCAM Answer	415
35.10.112	Ingress Configurable ACL 3 Rules Setup	418
35.10.113	Ingress Configurable ACL 3 TCAM	418
35.10.114	Ingress Configurable ACL 3 TCAM Answer	418
35.10.115	Ingress Drop Options	419
35.10.116	Ingress Egress Port Packet Type Filter	420
35.10.117	Ingress Ethernet Type for VLAN tag	422
35.10.118	Ingress MMP Drop Mask	423
35.10.119	Ingress Multiple Spanning Tree State	423
35.10.120	Ingress Port Packet Type Filter	424
35.10.121	Ingress Router Table	425



35.10.122	Ingress VID Ethernet Type Range Assignment Answer	427
35.10.123	Ingress VID Ethernet Type Range Search Data	427
35.10.124	Ingress VID Inner VID Range Assignment Answer	428
35.10.125	Ingress VID Inner VID Range Search Data	428
35.10.126	Ingress VID MAC Range Assignment Answer	429
35.10.127	Ingress VID MAC Range Search Data	429
35.10.128	Ingress VID Outer VID Range Assignment Answer	430
35.10.129	Ingress VID Outer VID Range Search Data	430
35.10.130	L2 Action Table	430
35.10.131	L2 Action Table Egress Port State	432
35.10.132	L2 Action Table Source Port	432
35.10.133	L2 Aging Collision Shadow Table	434
35.10.134	L2 Aging Collision Table	434
35.10.135	L2 Aging Status Shadow Table	435
35.10.136	L2 Aging Status Shadow Table - Replica	435
35.10.137	L2 Aging Table	435
35.10.138	L2 DA Hash Lookup Table	436
35.10.139	L2 Destination Table	436
35.10.140	L2 Destination Table - Replica	437
35.10.141	L2 Lookup Collision Table	438
35.10.142	L2 Lookup Collision Table Masks	438
35.10.143	L2 Multicast Handling	439
35.10.144	L2 Multicast Table	439
35.10.145	L2 Reserved Multicast Address Action	440
35.10.146	L2 Reserved Multicast Address Base	440
35.10.147	L2 SA Hash Lookup Table	441
35.10.148	L2 Tunnel Decoder Setup	441
35.10.149	L3 LPM Result	442
35.10.150	L3 Routing Default	442
35.10.151	L3 Routing TCAM	443
35.10.152	LACP Packet Decoder Options	444
35.10.153	LLDP Configuration	445
35.10.154	Learning And Aging Enable	445
35.10.155	Learning And Aging Writeback Control	446
35.10.156	Learning Conflict	447
35.10.157	Learning DA MAC	447
35.10.158	Learning Data FIFO	447
35.10.159	Learning Data FIFO High Watermark Level	448
35.10.160	Learning Overflow	448
35.10.161	Link Aggregate Weight	449
35.10.162	Link Aggregation Ctrl	449
35.10.163	Link Aggregation Membership	450
35.10.164	Link Aggregation To Physical Ports Members	450
35.10.165	MPLS EXP Field To Egress Queue Mapping Table	451
35.10.166	MPLS EXP Field To Packet Color Mapping Table	451
35.10.167	NAT Action Table	452
35.10.168	NAT Action Table Force Original Packet	452
35.10.169	Next Hop Packet Modifications	453
35.10.170	Next Hop Table	454
35.10.171	Port Move Options	455
35.10.172	RARP Packet Decoder Options	456
35.10.173	Reserved Destination MAC Address Range	456
35.10.174	Reserved Source MAC Address Range	457
35.10.175	Router Egress Queue To VLAN Data	458
35.10.176	Router MTU Table	459
35.10.177	Router Port MAC Address	459
35.10.178	SCTP Packet Decoder Options	460



35.10.179	SMON Set Search	460
35.10.180	SNAP LLC Decoding Options	461
35.10.181	Second Tunnel Exit Lookup TCAM	461
35.10.182	Second Tunnel Exit Lookup TCAM Answer	462
35.10.183	Second Tunnel Exit Miss Action	463
35.10.184	Send to CPU	463
35.10.185	Software Aging Enable	464
35.10.186	Software Aging Start Latch	464
35.10.187	Source Port Default ACL Action	465
35.10.188	Source Port Table	467
35.10.189	Time to Age	474
35.10.190	Tunnel Entry MTU Length Check	474
35.10.191	Tunnel Exit Lookup TCAM	475
35.10.192	Tunnel Exit Lookup TCAM Answer	476
35.10.193	VLAN PCP And DEI To Color Mapping Table	477
35.10.194	VLAN PCP To Queue Mapping Table	477
35.10.195	VLAN Table	478
35.11	MBSC	482
35.11.1	L2 Broadcast Storm Control Bucket Capacity Configuration	482
35.11.2	L2 Broadcast Storm Control Bucket Threshold Configuration	482
35.11.3	L2 Broadcast Storm Control Enable	482
35.11.4	L2 Broadcast Storm Control Rate Configuration	483
35.11.5	L2 Flooding Storm Control Bucket Capacity Configuration	483
35.11.6	L2 Flooding Storm Control Bucket Threshold Configuration	483
35.11.7	L2 Flooding Storm Control Enable	484
35.11.8	L2 Flooding Storm Control Rate Configuration	484
35.11.9	L2 Multicast Storm Control Bucket Capacity Configuration	484
35.11.10	L2 Multicast Storm Control Bucket Threshold Configuration	485
35.11.11	L2 Multicast Storm Control Enable	485
35.11.12	L2 Multicast Storm Control Rate Configuration	485
35.12	Scheduling	486
35.12.1	DWRR Bucket Capacity Configuration	486
35.12.2	DWRR Bucket Misc Configuration	486
35.12.3	DWRR Weight Configuration	487
35.12.4	Map Queue to Priority	487
35.12.5	Output Disable	488
35.13	Shapers	488
35.13.1	Port Shaper Bucket Capacity Configuration	488
35.13.2	Port Shaper Bucket Threshold Configuration	489
35.13.3	Port Shaper Enable	489
35.13.4	Port Shaper Rate Configuration	489
35.13.5	Prio Shaper Bucket Capacity Configuration	490
35.13.6	Prio Shaper Bucket Threshold Configuration	490
35.13.7	Prio Shaper Enable	491
35.13.8	Prio Shaper Rate Configuration	491
35.13.9	Queue Shaper Bucket Capacity Configuration	492
35.13.10	Queue Shaper Bucket Threshold Configuration	492
35.13.11	Queue Shaper Enable	492
35.13.12	Queue Shaper Rate Configuration	493
35.14	Shared Buffer Memory	493
35.14.1	Buffer Free	493
35.14.2	Egress Port Depth	494
35.14.3	Egress Queue Depth	494
35.14.4	Minimum Buffer Free	495
35.14.5	Packet Buffer Status	495
35.15	Statistics: ACL	495
35.15.1	Egress Configurable ACL Match Counter	495



35.15.2	Ingress Configurable ACL Match Counter	496
35.16	Statistics: Debug	496
35.16.1	Debug EPP Counter	496
35.16.2	Debug IPP Counter	497
35.16.3	EPP PM Drop	497
35.16.4	IPP PM Drop	497
35.16.5	PS Error Counter	498
35.16.6	SP Overflow Drop	498
35.17	Statistics: EPP Egress Port Drop	499
35.17.1	Egress Port Disabled Drop	499
35.17.2	Egress Port Filtering Drop	499
35.17.3	Tunnel Exit Too Small Packet Modification To Small Drop	500
35.17.4	Unknown Egress Drop	500
35.18	Statistics: IPP Egress Port Drop	500
35.18.1	Egress Spanning Tree Drop	500
35.18.2	Ingress-Egress Packet Filtering Drop	501
35.18.3	L2 Action Table Per Port Drop	501
35.18.4	MBSC Drop	502
35.18.5	Queue Off Drop	502
35.19	Statistics: IPP Ingress Port Drop	502
35.19.1	AH Decoder Drop	502
35.19.2	ARP Decoder Drop	503
35.19.3	BOOTP and DHCP Decoder Drop	503
35.19.4	CAPWAP Decoder Drop	504
35.19.5	DNS Decoder Drop	504
35.19.6	ESP Decoder Drop	504
35.19.7	Egress Configurable ACL Drop	505
35.19.8	Empty Mask Drop	505
35.19.9	Expired TTL Drop	505
35.19.10	GRE Decoder Drop	506
35.19.11	IEEE 802.1X and EAPOL Decoder Drop	506
35.19.12	IKE Decoder Drop	507
35.19.13	IP Checksum Drop	507
35.19.14	Ingress Configurable ACL Drop	507
35.19.15	Ingress Packet Filtering Drop	508
35.19.16	Ingress Spanning Tree Drop: Blocking	508
35.19.17	Ingress Spanning Tree Drop: Learning	508
35.19.18	Ingress Spanning Tree Drop: Listen	509
35.19.19	Invalid Routing Protocol Drop	509
35.19.20	L2 Action Table Drop	510
35.19.21	L2 Action Table Port Move Drop	510
35.19.22	L2 Action Table Special Packet Type Drop	510
35.19.23	L2 IEEE 1588 Decoder Drop	511
35.19.24	L2 Lookup Drop	511
35.19.25	L2 Reserved Multicast Address Drop	512
35.19.26	L3 Lookup Drop	512
35.19.27	L4 IEEE 1588 Decoder Drop	512
35.19.28	LACP Decoder Drop	513
35.19.29	Learning Packet Drop	513
35.19.30	Maximum Allowed VLAN Drop	513
35.19.31	Minimum Allowed VLAN Drop	514
35.19.32	NAT Action Table Drop	514
35.19.33	RARP Decoder Drop	515
35.19.34	Reserved MAC DA Drop	515
35.19.35	Reserved MAC SA Drop	515
35.19.36	SCTP Decoder Drop	516
35.19.37	Second Tunnel Exit Drop	516



35.19.38	Source Port Default ACL Action Drop	516
35.19.39	Tunnel Exit Miss Action Drop	517
35.19.40	Tunnel Exit Too Small Packet Modification Drop	517
35.19.41	Unknown Ingress Drop	518
35.19.42	VLAN Member Drop	518
35.20	Statistics: IPP Ingress Port Receive	518
35.20.1	IP Multicast ACL Drop Counter	518
35.20.2	IP Multicast Received Counter	519
35.20.3	IP Multicast Routed Counter	519
35.20.4	IP Unicast Received Counter	520
35.20.5	IP Unicast Routed Counter	520
35.21	Statistics: Misc	521
35.21.1	Buffer Overflow Drop	521
35.21.2	Drain Port Drop	521
35.21.3	Egress Resource Manager Drop	521
35.21.4	Flow Classification And Metering Drop	522
35.21.5	IPP Empty Destination Drop	522
35.21.6	Ingress Resource Manager Drop	523
35.21.7	MAC RX Broken Packets	523
35.21.8	MAC RX Long Packet Drop	523
35.21.9	MAC RX Short Packet Drop	524
35.21.10	Re-queue Overflow Drop	524
35.22	Statistics: NAT	524
35.22.1	Egress NAT Hit Status	524
35.22.2	Ingress NAT Hit Status	525
35.23	Statistics: Packet Datapath	525
35.23.1	EPP Packet Head Counter	525
35.23.2	EPP Packet Tail Counter	526
35.23.3	IPP Packet Head Counter	526
35.23.4	IPP Packet Tail Counter	526
35.23.5	MAC Interface Counters For RX	527
35.23.6	MAC Interface Counters For TX	527
35.23.7	PB Packet Head Counter	527
35.23.8	PB Packet Tail Counter	528
35.23.9	PS Packet Head Counter	528
35.23.10	PS Packet Tail Counter	529
35.24	Statistics: Routing	529
35.24.1	Next Hop Hit Status	529
35.24.2	Received Packets on Ingress VRF	529
35.24.3	Transmitted Packets on Egress VRF	530
35.25	Statistics: SMON	530
35.25.1	SMON Set 0 Byte Counter	530
35.25.2	SMON Set 0 Packet Counter	531
35.25.3	SMON Set 1 Byte Counter	531
35.25.4	SMON Set 1 Packet Counter	532
35.25.5	SMON Set 2 Byte Counter	532
35.25.6	SMON Set 2 Packet Counter	532
35.25.7	SMON Set 3 Byte Counter	533
35.25.8	SMON Set 3 Packet Counter	533

List of Figures

1.1 Switch Core Overview	19
4.1 Jitter Overview	36
5.1 VLAN Packet Operations	39
6.1 L2 Lookup Overview	47
17.1 D-left Function	120
18.1 Learning and Aging Engine	125
18.2 Learning Frame	128
20.1 General Token Bucket Illustration	133
21.1 Egress Queue Selection Diagram	136
21.2 Egress Queue Scheduling example. Here using half the priorities, with two queues mapped to each.	140
22.1 Packet Initial Color Selection Diagram	144
23.1 MMP pointer Selection Diagram	148
26.1 Buffer memory congestion zones	157
27.1 The buffer memory is partitioned into Reserved and FFA areas. The unallocated area is the space set aside for the currently incoming packets.	161
29.1 Location of Statistics Counters	169
30.1 Packet from CPU with CPU tag	174
30.2 Packet to CPU with CPU tag	175
31.1 Core Initialization	184
34.1 Timing diagram for a single ported memory used in the dual ported memory wrapper. In this case a concurrent read and write to the same address of a memory wrapper set for one cycle latency and with the write through attribute set.	223
35.1 Address space usage by tables	234

List of Tables

1.1 Port Numbering Table	24
------------------------------------	----

8.1 Tunnel Entry Unicast or Multicast	59
13.1 PTP Header Format	71
13.2 PTP over 802.3 Ethernet	72
13.3 PTP over UDP/IPv4	72
13.4 PTP over UDP/IPv6	72
14.1 Ingress ACL Engine Settings	77
14.4 Hash Key Example for Ethernet Type	79
14.5 Hash Key Example for Simple L2 ACL	79
14.6 Hash Key Example for L3 IPv4 ACL	79
14.7 Hash Key Example for L4 ACL	80
14.8 Hash Key Example for Ingress NAT Entry	80
14.11 Hash Key Example for IPv4 DA	82
14.12 Hash Key Example for Destination MAC Address and Outer LAN VID	83
14.13 Hash Key Example for Complex L2 ACL	83
14.14 Hash Key Example for L3 IPv4 ACL	83
14.15 Hash Key Example for L4 ACL	83
14.16 Hash Key Example for Openflow Entry	84
14.17 Hash Key Example for Ingress NAT Entry	84
14.20 Hash Key Example for TOS Byte	86
14.21 Hash Key Example for Destination MAC Address and Outer LAN VID	86
14.22 Hash Key Example for Complex L2 ACL	87
14.23 Hash Key Example for L3 IPv6 ACL	87
14.24 Hash Key Example for L4 ACL	87
14.25 Hash Key Example for Openflow Entry	87
14.26 Hash Key Example for Ingress NAT Entry	88
14.28 Hash Key Example for TOS Byte	89
14.29 Hash Key Example for Exception ACL	89
14.30 Actions that will take effect if one or more is set.	90
14.31 The lowest numbered takes effect if no priority else the highest numbered with priority set.	91
14.32 Egress ACL Engine Settings	91
14.33 Fields used in the rule search.	92
14.35 Hash Key Example for Ethernet Type	94
14.36 Hash Key Example for Simple L2 ACL	94
14.37 Hash Key Example for L3 IPv4 ACL	94
14.38 Hash Key Example for L4 ACL	95
14.39 Hash Key Example for Egress NAT Entry	95
14.40 Hash Key Example for IPsec Encryption Entry	95
14.41 Hash Key Example for MACsec Encryption Entry	95
14.43 Hash Key Example for TOS Byte	97
14.44 Hash Key Example for Simple L2 ACL	97
14.45 Hash Key Example for L3 IPv6 ACL	97
14.46 Hash Key Example for L4 ACL	98
14.47 Hash Key Example for Egress NAT Entry	98
14.48 Hash Key Example for IPsec Encryption Entry	98
14.49 Hash Key Example for MACsec Encryption Entry	98
14.50 Actions that will take effect if one or more is set.	99
14.51 The lowest numbered takes effect if no priority else the highest numbered with priority set.	100
18.1 Hardware Aging Operations	127
18.2 Learning Header	128
22.1 Code for Colors	143
23.1 Rate Configuration Example (Assume tickFreqList = [1MHz, 100KHz, 10KHz, 1KHz, 100Hz])	149
29.1 Sequence of Statistics Counters	168



30.1 From CPU tag format	173
30.2 To CPU Header	177
30.3 Packet Type Table	179
30.4 Reason for packet sent to CPU	181
31.1 Clock and Reset interfaces	184
31.2 Packet RX interface. N is the ingress interface number.	185
31.3 Packet TX interface. N is the egress interface number.	186
31.4 The APB interface signals	187
31.5 Interrupt interface	188
31.6 ThePFC status and External Pause interfaces, where N is the packet interface number	189
31.7 The Debug Read interface	189
31.8 Debug Selection Map	209
31.9 The Debug Write interface	210
33.1 IPP Debug List	215
33.2 EPP Debug List	217
34.1 The settings for pipeline flops between floorplan blocks	219
34.2 The settings for input and output flops for the floorplan blocks	219
34.3 The memory macros needed for this core.Types: dp=two ports, one read and one write, running on the same clock. dc=two ports, one read and one write, with separate clocks for read and write.	222



Chapter 1

Overview

This L2/L3 Ethernet Switching/Routing Core offers full wire-speed on all 32 ports, with up to 100 Gbit of bandwidth per port. Each port has 8 egress queues which are controlled by a multi-level scheduler.

The core is built around a shared buffer memory architecture capable of simultaneous wire-speed switching on all ports without head of line blocking. Packets are stored in the shared buffer memory as fixed size cells of 256 bytes. In total the buffer memory has a capacity of 16400 cells. The packet processing is done in parallel in four switch slices.

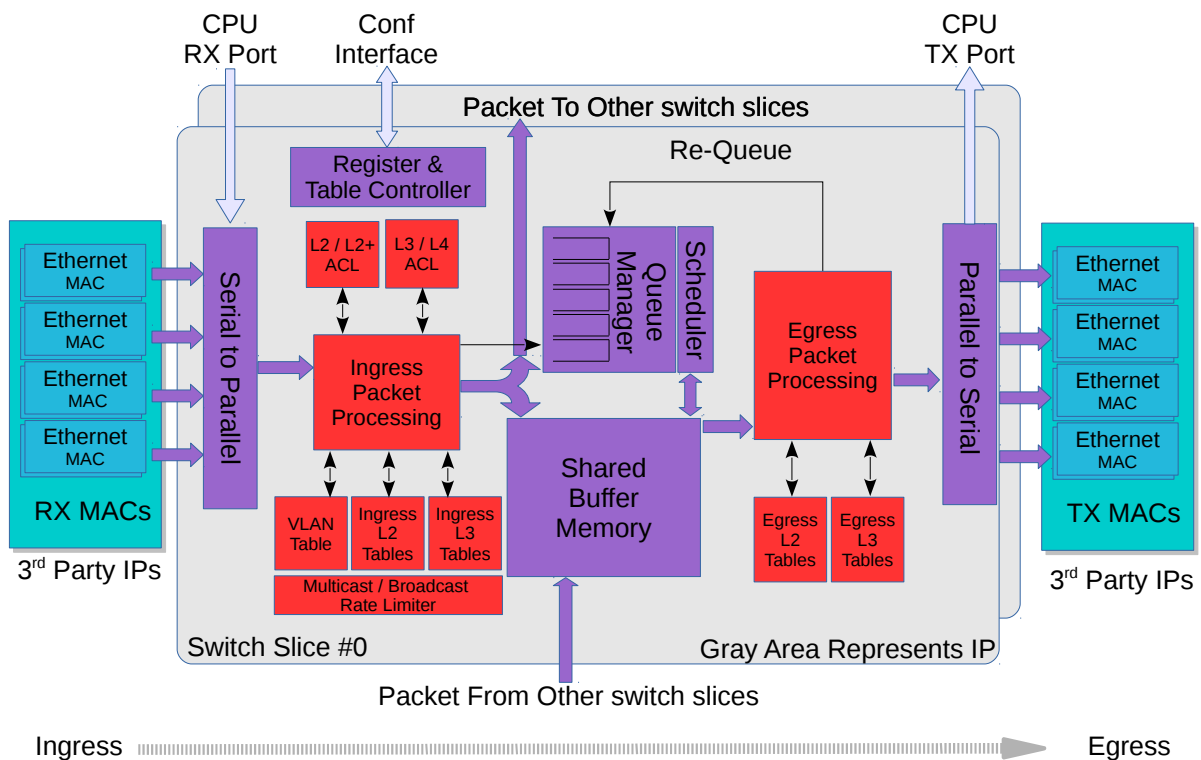


Figure 1.1: Switch Core Overview

Configuring tables and registers are done through a Configuration interface. However it is not required to perform any configuration. The core is ready to receive and forward Ethernet frames once the reset sequence has been completed.

1.1 Feature Overview

- 32 x 100 Gigabit Ethernet ports.
- Full wire-speed on all ports and all Ethernet frame sizes.
- Store and forward shared memory architecture.
- Support for jumbo packets up to 32738 bytes.
- Passes maximum overlap mesh test (RFC2899) using all ports for all packet sizes up to 1601 bytes.
- Queue management operations:
 - Disable scheduling of packets on a port.
 - Disable queuing new packets to a port.
 - Allow a port to be drained without sending out packets.
 - Allow checking if a port is empty or not.
- Input and output mirroring.
- 4 source MAC address ranges with a number of different actions.
- 4 destination MAC address ranges with a number of different actions.
- 4,096 entry L2 MAC table, hash based 4-way.
- 4,096 entry VLAN table.
- 32 entry synthesized CAM to solve hash collisions.
- 4 entries of the synthesized CAM are fully maskable.
- 64 entry L2 multicast table.
- Automatic aging and wire-speed learning of L2 addresses. Does not require any CPU/software intervention.
- Spanning tree support, ingress and egress checks.
- 16 multiple spanning trees, ingress and egress checks.
- Allows software to inject special packets which are used to write into MAC tables while hardware learning engine is running.
- Allows software to track which L2 MAC entries are being learned and port moved.
- Allows software to track which L2 MAC entries are being aged out.
- Egress VLAN translation table allowing unique VID-to-VID translation per egress port.
- VLAN priority tag can bypass VLAN processing and be popped on egress.
- MPLS forwarding with support for swap,push,pop and penultimate pop operations.
- 4 entry VRF table.
- 512 * 4 hash based L3 routing table.
- 16 entry L3 routing TCAM.
- 1,024 entry next hop table. Pointed to from the routing entries.
- 1,024 entry packet modification table used by the next hop table to determine how build l2 fields in a packet to find the next hop.
- Configurable ECMP support based on L3 protocol field,L3 Tos, and L4 SP/DP.
- ECMP supports with up to 64 paths.



- 2,048 number of Ingress Network Address Translation (NAT) entries.
- 1,024 number of Egress Network Address Translation (NAT) entries.
- 2504 entries of ingress classification / ACL Lookups. The classification / ACL keys are configurable for each source port and the fields are selected from a incoming packets L2, L3 or L4 fields. The selection is described in 14.2 The classification / ACL key can be up to 540 bits long. The classification / ACL lookup is based on a combination of hash and TCAM. The actions which can be done is listed below:
 - Multiple actions can be assigned to each result. All results can be done in parallel if the user so wishes.
 - Result action can be to drop a packet.
 - Result action can be to send a packet to the CPU port.
 - Result action can be to send a packet to a specific port.
 - Result action can be to update a counter. There are 64 counters which can be used by the classification / ACL engine.
 - Result action can be to force packet to a specific queue on a egress port.
 - Result action can be to assign a meter/market/policer to measure the packet bandwidth.
 - Result action can be to assign a color to the packet which is used by the meter/marker/policer.
 - Result action can be to force the packet to use a specific VID when doing the VLAN table lookup.
 - Result action can be to do a input mirror on a packet.
 - Result action can be to not allow the packet to be learned in L2 MAC table.
- The ingress configurable classification / ACL engine can use the type and code fields from ICMP frames.
- The ingress configurable classification / ACL engine can use the fields, including the group address, from IGMP frames.
- 1312 entries of egress classification / ACL rules. The classification / ACL keys are configurable based on what forwarding actions has been done and the fields are selected from the incoming packets L2, L3 or L4 fields and from forwarding results. The selection is described in 14.4 The ACL key can be up to 540 bits long. For each field there are options to only select part of the bits in a field. The ACL lookup is based on a combination of hash and TCAM. The actions are listed below:
 - Multiple actions can be assigned to each result. All results can be done in parallel if the user so wishes.
 - Result action can be to drop a packet.
 - Result action can be to send a packet to the CPU port.
 - Result action can be to send a packet to a different port than ingress forwarding has decided.
 - Result action can be to update a counter. There are 64 counters which can be used by the classification / ACL engine.
- The egress configurable classification / ACL engine can use the type and code fields from ICMP frames.
- The egress configurable classification / ACL engine can use the fields, including the group address, from IGMP frames.
- 33587200 bits shared packet buffer memory divided between 4 switch slices. Each slice holds 4100 cells with a cell size of 256 bytes.
- 8 priority queues per egress port.



- Configurable mapping of egress queue from IP TOS, MPLS exp/tc or VLAN PCP bits.
- 32 ingress admission control entries.
- Deficit Weighted Round Robin Scheduler.
- Bandwidth shapers per port.
- Individual bandwidth shapers for each priority on each port.
- Individual bandwidth shapers for each queue on each port.
- Egress queue resource limiter/guarantee with four sets of configurations.
- Configuration interface for accessing configuration and status registers/tables.
- Multicast/Broadcast storm control with separate token buckets for flooding, broadcast and multicast packets.
- Multicast/Broadcast storm control is either packet or byte-based, configurable per egress port.
- LLDP frames can optionally be sent to the CPU.
- IEEE 1588 / PTP support for 1-step and 2-step Ordinary Clock mode. The switch supports transfer of 8 byte timestamp from receive MAC to software and from software to transmit MAC.
- The packets which are sent to the CPU can contain extra sw-defined "meta-data" which software sets up. Meta-data is 2 bytes and can come from a number of different tables.
- Wirespeed tunnel exit and tunnel entry. No looping of packets is needed.
- Tunnel unit for both tunnel entry and tunnel exit. Tunnel exit can be done in the beginning of the packet processing or after normal L2, L3, ACL lookups. The tunnel exit can be done on known fields or by looking up bytes anywhere in the first cell of the packet. Tunnel entry can be done as a result from the normal L2,L3, ACL processing.
- The tunnel exit allows packet headers/bytes to be removed and certain information to be copied from the original packet to new tunnel exited packet. Once a tunnel exit has been done the new tunnel exited packet will be processed as normal packet at wirespeed.
- The tunnel entry allows packet headers/bytes to be added and certain information from the previous packet to be copied to the new tunnel headers. The tunnel entry is reached from normal L2,L3 and ACL processing and happens just before the packet is sent out allowing the inner packet to do full switching and routing.

A Packets Way Through The Core

This section describes the path of a packet through the core from reception to transmission, i.e from the RX MAC bus to the TX MAC bus. See Figure 1.1.

1. A packet is received on the RX MAC bus with a *start of packet* signal.
2. Ingress port counters are updated.
3. The asynchronous ingress FIFO synchronizes the incoming data from the data rate of the MAC clock to the data rate of the core clock.
4. The serial to parallel converter accumulates 256 bytes to build a cell, and the cell is sent to ingress processing, if a packet consists of more than 256 bytes then a new cell is built. This is repeated until the *end of packet* signal is asserted.
5. Ingress processing (see chapter 3.1) determines the destination port (or ports) and egress queue of the packet. It then decides whether the packet shall be queued or dropped. Many different tables and registers are used in the process to determine the final portmask and final egress queue for the packet.
6. If the packet matches a certain traffic type whose bandwidth is monitored by the core, it will be pointed to one of the 32 meter-marker-droppers to do the rate measurement. The result may drop the packet or change the packet color.
7. Packets are never modified before they are written into the buffer memory. Rather an ingress to egress header (I2E header) is appended to the packet. Any modifications are done in the egress packet processing pipeline, based on the I2E header.
8. Unless the packet is dropped, the packet is written cell-by-cell into the buffer memory with the I2E header appended.
9. A packet which shall be written to multiple switch slices is written to all destination switch slices in parallel. The buffer memory on each switch slice has enough write accesses to every clock cycle accept one cell from each switch slice without write contention. At the same time as writes are performed to the buffer memory a readout can be performed without any contention.
10. Once the entire packet is written to buffer memory, it is placed in one or more egress queues and made available to the egress scheduler.
11. Each queue is a linked list of pointers to the first cell in each packet linked to the queue. Each egress queue can link all the packets in the buffer memory even if the buffer memory is filled with only minimum size packets.
12. Counters of the number of cells per ingress port, per ingress port priority, per egress port and egress port queue are updated according to where the packet is sent.
13. A port with packets available for transmission, will only transmit a new packet if the port shaper allows it to.
14. When an instance of the packet is selected for output by the egress scheduler, the queue manager will read the packet from the buffer memory and send it, cell-by-cell to the egress packet processing.
15. Egress processing (see chapter 3.2) determines how and if the packet shall be sent out and does the final modifications of the packet. A packet can be re-queued again if it shall be sent out multiple times, which could be the case if input/output mirroring is used. L3 multicast may also re-queue a packet multiple times to the same port.
16. Once the packet is no longer part of any egress queue, the cells it occupied in the buffer memory are deallocated so they can be used by other packets.
17. The parallel to serial converter divides the cell into MAC-bus sized chunks.
18. One asynchronous FIFO per egress port synchronizes the outgoing data from the core clock to the MAC clock.



19. Data is transmitted on the output port.
20. Egress port counters are updated.

1.2 Port Numbering Table

Table 1.1 shows the port numbering. Port 31 can serve as a CPU port.

Interface Number	BW	Slice	Clock	Clock Frequency	Sync With Core Clock	Port Number & Multicast Table Bit	CPU Port
0	100.0Gbit/s	0	clk_mac_rx/tx_0	781.25MHz	No	0	No
1	100.0Gbit/s	0	clk_mac_rx/tx_1	781.25MHz	No	1	No
2	100.0Gbit/s	0	clk_mac_rx/tx_2	781.25MHz	No	2	No
3	100.0Gbit/s	0	clk_mac_rx/tx_3	781.25MHz	No	3	No
4	100.0Gbit/s	0	clk_mac_rx/tx_4	781.25MHz	No	4	No
5	100.0Gbit/s	0	clk_mac_rx/tx_5	781.25MHz	No	5	No
6	100.0Gbit/s	0	clk_mac_rx/tx_6	781.25MHz	No	6	No
7	100.0Gbit/s	0	clk_mac_rx/tx_7	781.25MHz	No	7	No
8	100.0Gbit/s	1	clk_mac_rx/tx_8	781.25MHz	No	8	No
9	100.0Gbit/s	1	clk_mac_rx/tx_9	781.25MHz	No	9	No
10	100.0Gbit/s	1	clk_mac_rx/tx_10	781.25MHz	No	10	No
11	100.0Gbit/s	1	clk_mac_rx/tx_11	781.25MHz	No	11	No
12	100.0Gbit/s	1	clk_mac_rx/tx_12	781.25MHz	No	12	No
13	100.0Gbit/s	1	clk_mac_rx/tx_13	781.25MHz	No	13	No
14	100.0Gbit/s	1	clk_mac_rx/tx_14	781.25MHz	No	14	No
15	100.0Gbit/s	1	clk_mac_rx/tx_15	781.25MHz	No	15	No
16	100.0Gbit/s	2	clk_mac_rx/tx_16	781.25MHz	No	16	No
17	100.0Gbit/s	2	clk_mac_rx/tx_17	781.25MHz	No	17	No
18	100.0Gbit/s	2	clk_mac_rx/tx_18	781.25MHz	No	18	No
19	100.0Gbit/s	2	clk_mac_rx/tx_19	781.25MHz	No	19	No
20	100.0Gbit/s	2	clk_mac_rx/tx_20	781.25MHz	No	20	No
21	100.0Gbit/s	2	clk_mac_rx/tx_21	781.25MHz	No	21	No
22	100.0Gbit/s	2	clk_mac_rx/tx_22	781.25MHz	No	22	No
23	100.0Gbit/s	2	clk_mac_rx/tx_23	781.25MHz	No	23	No
24	100.0Gbit/s	3	clk_mac_rx/tx_24	781.25MHz	No	24	No
25	100.0Gbit/s	3	clk_mac_rx/tx_25	781.25MHz	No	25	No
26	100.0Gbit/s	3	clk_mac_rx/tx_26	781.25MHz	No	26	No
27	100.0Gbit/s	3	clk_mac_rx/tx_27	781.25MHz	No	27	No
28	100.0Gbit/s	3	clk_mac_rx/tx_28	781.25MHz	No	28	No
29	100.0Gbit/s	3	clk_mac_rx/tx_29	781.25MHz	No	29	No
30	100.0Gbit/s	3	clk_mac_rx/tx_30	781.25MHz	No	30	No
31	100.0Gbit/s	3	clk_mac_rx/tx_31	781.25MHz	No	31	Yes

Table 1.1: Port Numbering Table

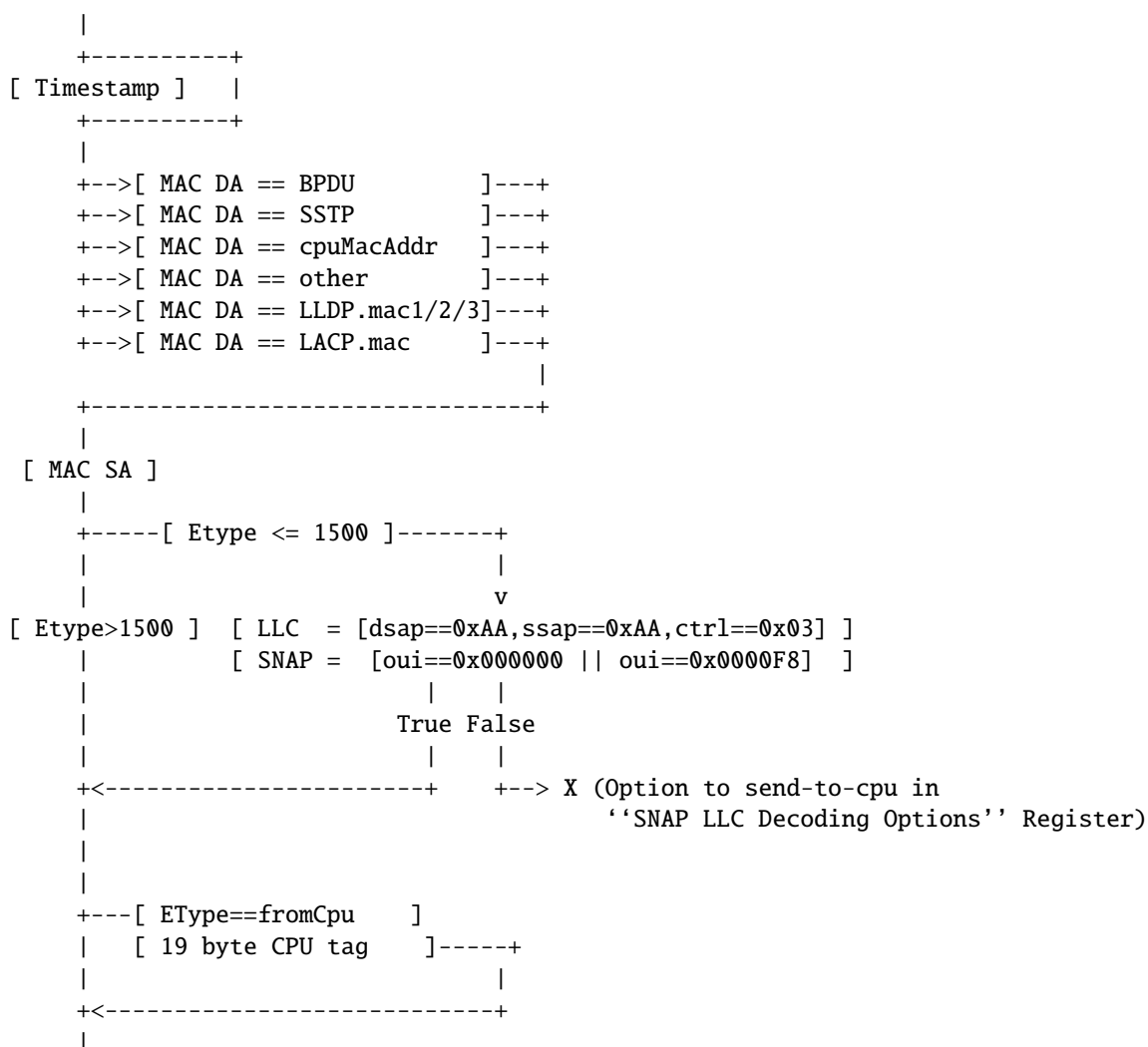
Chapter 2

Packet Decoder

The packet decoder identifies protocols and extracts information to be used in the packet processing.

2.1 Decoding Sequence

In the following diagram the decoding of the incoming packet header is described. The comparison used to determine protocol types are described as well as the order they are decoded. The end of decoding process is denote by an X.



oui==0x0000F8) then there exists a option to send the packet to the CPU in register **SNAP LLC Decoding Options**. If not sent to the CPU the decoding will stop here.

- (b) LLDP

If the MAC DA address is equal to any of the **LLDP Configuration** mac1/mac2/mac3 addresses and the Ethernet Type is equal to the register **LLDP Configuration** field **eth** then the field **portmask** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. Default is to forward LLDP frames to the CPU port. A packet that matches the LLDP criteria will not be considered a BPDU packet even if it matches the BPDU multicast address.
- (c) ARP

If the Ethernet Type field is equal to the **ARP Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
- (d) RARP

If the Ethernet Type field is equal to the register **RARP Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
- (e) 802.1X and EAPOL Packets

If the Ethernet Type field is equal to register **IEEE 802.1X and EAPOL Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped. The drop counter is located in **IEEE 802.1X and EAPOL Decoder Drop**.
- (f) IEEE 1588 L2 Ethernet Type

If the Ethernet Type field is equal to register **IEEE 1588 L2 Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
- (g) PTP

When identified as a PTP/1588 packet by the EtherType and if the packet is sent to the CPU with a To CPU Tag then the *ptp* bit will be set.
- (h) VLAN Tags

There are a number of fixed VLAN types that are identified as well as configurable types. The VLAN processing will use the VLAN tags that decoding has identified and ignore intermediate tags of other types.

 - i. Customer VLAN Type - 0x8100
 - ii. Service VLAN Tag - 0x88A8
 - iii. Configurable VLAN Type setup **Ingress Ethernet Type for VLAN tag**.

When using the Configurable Customer/Service VLAN Type the egress pipeline needs to be setup with the same values if there are actions configured that pushes new VLAN tags to the packet. This is setup in register **Egress Ethernet Type for VLAN tag**.
- (i) MPLS.

One MPLS tag is decoded. No other L3 decoding will be done after this.
- (j) From CPU Tags

Packets from CPU will use a Ethernet type value of 0x9988. The From CPU Tag is further described in Chapter 30.
- (k) IPv4 or IPv6.

If the type identifies these protocols (potentially also after a PPPoE header) the following IPv4



or IPv6 headers are decoded. IPv4 packet with wrong header checksum can be accepted or dropped according to the **Check IPv4 Header Checksum** register. If the L4 protocol is TCP or UDP these headers are also decoded.

- (l) Routing Header.

If a routing header is identified in IPv6, the L4 protocol is decoded from the routing header. The core supports further process for the segment routing header, for other routing types the core will skip the routing header if the segments left field is 0, otherwise the packet will be treated as unrecognized and sent to the CPU.
- (m) L4 Protocol.

If the packet is either a IPv4 or IPv6 and if the L4 protocol is either UDP or TCP then the source port and destination port fields will be extracted.

 - i. ICMP header

The ICMP type along with the code extracted.
 - ii. IGMP header

The IGMP type along with the code and IPv4 group address is extracted.
 - iii. AH Header

If the next protocol field in IPv4 or IPv6 is equal to the register **AH Header Packet Decoder Options** field **I4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
 - iv. ESP Header

If the next protocol field in IPv4 or IPv6 is equal to the register **ESP Header Packet Decoder Options** field **I4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
 - v. GRE

If the next protocol field in IPv4 or IPv6 is equal to the register **GRE Packet Decoder Options** field **I4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
 - vi. SCTP

If the next protocol field in IPv4 or IPv6 is equal to the register **SCTP Packet Decoder Options** field **I4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
- (n) UDP or TCP Source or Destination Port Checks
 - i. GRE

If the Destination Port in UDP is equal to the **GRE Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
 - ii. DNS

If the Destination Port in UDP or TCP is equal to the **DNS Packet Decoder Options** field **I4Port** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
 - iii. BOOTP or DHCP

If the Destination Port in UDP is equal to the register **BOOTP and DHCP Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.



- iv. CAPWAP
If the Destination Port in UDP is equal to the register **CAPWAP Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
 - v. IKE
If the Destination Port in UDP is equal to the register **IKE Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
 - vi. IEEE 1588 L4
If the Destination Port, and IPv4 or IPv6 and the UDP is equal to the register **IEEE 1588 L4 Packet Decoder Options** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
- (o) Unknown.
After an unknown Ethernet type no further decoding is done.



Chapter 3

Packet Processing

3.1 Ingress Packet Processing

The ingress packet processing is done as soon as the packet enters the switch. The packet is not sent to the buffer memory until the ingress packet processing is done.

1. Source Port to Link Aggregate
Source port is mapped to a link aggregate through the [Link Aggregation Membership](#) table. From this point all references to source ports are actually link aggregate numbers. For details see the [Link Aggregation](#) chapter.
2. Packet Decoding for Tunnel Exit Lookup
The packet headers are decoded and data extracted. For details see the [Packet Decoder For Tunnel Exit](#) section in the tunneling chapter.
3. Tunnel Exit Lookup
The packet is subjected to a tunnel exit lookup which if found true can remove a part of the packets headers and/or payload of the packet. Certain fields from the original packet can also be copied to the inner packet. Once this has been done the packet processing will be only done on the inner packet. For details see the [Tunnel Exit](#) section.
4. Packet Decoding
The packet headers are decoded and data extracted. For details see the [Packet Decoding](#) chapter.
5. Destination MAC Address Range Classification
The destination MAC address is compared with [Reserved Destination MAC Address Range](#) table to determine if it should be dropped, sent to CPU or if priority should be forced.
6. Source MAC Address Range Classification
The destination MAC address is compared with [Reserved Source MAC Address Range](#) table to determine if it should be dropped, sent to CPU or if priority should be forced.
7. SMON
If the packets source port and the VID for the outermost VLAN matches an SMON counter then that counter will be updated (see the [Statistics](#) chapter).
8. Ingress Port Packet Type Filter
The ingress packet type filter, setup through [Ingress Port Packet Type Filter](#) per source port, determines if the packet will be dropped or be processed further. This is based on protocol type and type of VLAN. See the [VLAN and Packet Type Filtering](#) chapter.
9. Configurable ACL
The incoming packet is classified on a configurable selection of L2, L3 and L4 fields. The ACL lookup is a d-left hash search, described in [Dleft Lookup](#). There are numerous actions that can be applied when a packet matches an ACL entry. For details see the [Configurable ACL Engine](#) section.

10. Ingress Spanning Tree

The ingress spanning tree state of the source port (from the **Source Port Table**) is checked to determine if packet processing should continue. STP is further described in the **Spanning Tree** chapter.
11. Ingress VLAN Processing

VLAN processing consists of two parts. Determining the VLAN membership and performing VLAN header modifications.

The VLAN membership is determined from the assigned ingress VID. See the **Assignment of Ingress VID** section. This will then be used to index into the **VLAN Table** to determine, among other things, VLAN port membership, MSTP and Global ID used in L2 lookups.
12. Ingress MSTP

The VLAN membership determines which MSTP the packet belongs to by pointing into the **Ingress Multiple Spanning Tree State** table. The state of the source port within this MSTP is checked to determine if packet processing should continue. MSTP is further described in the **Spanning Tree** chapter.
13. IP Routing

The routing function figures out where to forward the packet by determining the Next Hop. For details on the routing function see the **Routing** chapter.

 - (a) Determine Next Hop

The routing function is entered if an IP packet matches the router ports MAC address (**Router Port MAC Address**) and routing is allowed on the packets VLAN. L2 lookup, learning and aging will not be performed on routed packets. The router will search for the IP destination address in the routing tables to determine the packets Next Hop, i.e. which port to send the packet to.
 - (b) VLAN Operations

The Next Hop will also determine up to two VLAN operations to perform on the routed packet.
14. IPv4 checksum check and drop.

For IPv4 packets calculate the checksum value and optionally drop the packet with wrong checksum value. For a routed IPv4 packet the check and drop is always performed.
15. L2 Switching

If the packet is not routed the destination MAC address is searched for in the **L2 DA Hash Lookup Table**. If the address is found the corresponding entry in the **L2 Destination Table** will return a single destination port or multiple egress ports (if the destination address points to a multicast entry). The status in the **L2 Aging Table** is also updated. If the destination address is not found then the packet will be flooded to all ports that are members of the packets VLAN. See chapter **L2 Switching** for details.
16. L2 Action Table Lookup

The L2 Action Table Lookups provides a extra level of controll over what shall be done with the L2 packets. It can be used to archive 802.1X compliance and be used to secure the switch. The functionality has a enable bit in the **Source Port Table** field **enableL2ActionTable**. Depending on the result from both the L2 SA Lookup, L2 DA Lookup and status on source port (**I2ActionTablePortState**) and destination port(s) **L2 Action Table Egress Port State** a address is formed to read out L2 Action Tables. The **L2 Action Table** is based on the packets destiantion ports, while **L2 Action Table Source Port** is based on the packets incoming source port. If the packet is going to no egress port (portmask==0) then none of the **L2 Action Table** actions will be done while the **L2 Action Table Source Port** is always carried out (When function is enabled).
17. Egress Spanning Tree

When the destination port(s) are known, the spanning tree state for the destination ports are checked in **Egress Spanning Tree State** register.
18. Egress MSTP

The MSPT state for the destination ports are checked in the **Egress Multiple Spanning Tree State**



register. The MSTP id, determined above, is used to index the table.

19. Learning Lookup
If the packet is not routed the source MAC address is searched in the **L2 SA Hash Lookup Table**. If the address is not found or it has moved to a different port then the Learning Engine will update the tables unless the packet was marked to be dropped. See the **Learning and Aging** chapter for details.
20. IP Statistics
Statistics of IP unicast, multicast and routed packets are updated.
21. Configurable Egress ACL
The Egress ACL can classify incoming packet based on a configurable selection of L2, L3 and L4 fields but also based on the result from switching and routing. The ACL lookup is a D-left hash search, described in **Dleft Lookup**. There are numerous actions that can be applied when a packet matches an ACL entry. For details see the **Configurable Egress ACL Engine** section.
22. Ingress/Egress Port Packet Type Filter
As the packet is ready to be queued, the **Ingress Egress Port Packet Type Filter** is applied for each egress port where the the packet is to be queued. See chapter **VLAN and Packet Type Filtering**.
23. Link Aggregation
The destination ports are now mapped to physical ports using a hash function on the packet headers. The hash index selects which of the physical member ports of this link aggregate that the packet should be sent to. See the **Link Aggregation** chapter.
24. Multicast Broadcast Storm Control
Multicast packets that are destined for physical ports that have exceeded the MBSC limits will be dropped at this point. See chapter **Multicast Broadcast Storm Control**.
25. Input Mirroring
If the source port is setup to be input mirrored the mirror port is now added to the list of destination ports. A copy of the input packet, without modifications, will be transmitted on the selected mirror port.
26. Determine Egress Queue Priority
Egress queues are assigned to packets based on their L2/L3 protocols or classification results. See the **Determine Egress Queue Priority** section.
27. Packet Initial Coloring
Initial colors are assigned to packets based on their L2/L3 protocols or classification results to represent the drop precedence. See the **Ingress Packet Initial Coloring** section.
28. NAT Action Table Check
Certain processing bits, if the packet was routed, if the packet was switch, if the packet was flooded along with bits from ingress and egress ACL plus status bits from ports are looked up in the table **Egress Port NAT State**. This table can redirect packets to the CPU, Drop the packet or do nothing.
29. Queue Management
If queue management has turned off queuing to a port the packet will be dropped at this point. See section **Queue Management** for details.
30. Drop Statistics
If the preceding processing has not set any destination ports then the packet is dropped and the **Empty Mask Drop** counter is incremented.
31. Ingress Admission Control
Packets are grouped into traffic groups based on source port numbers and packet headers, and the bandwidth of each traffic group is measured. If a traffic group exceeds the configured bandwidth or burst size, the initial packet color can be remarked or the packet can be dropped. See the **Ingress Admission Control** section. While the grouping process is through sequence of ingress packet



processing steps, the metering process is after all other ingress packet processing are done and before the enqueueing of the packet.

3.2 Egress Packet Processing

After ingress packet processing the packet is stored in the packet buffer memory. The egress packet processing is done when the packet is scheduled for transmission. A single packet can be sent out in multiple copies, for example due to broadcast or mirroring. If the copies are not identical, or multiple copies should be transmitted on the same port, then the packet will be re-queued. This means that it will be re-inserted into the queue engine, where it will again be selected for output and passed once more through the egress packet processing.

1. Output Mirroring
If output mirroring is enabled for the egress port then the packet is re-queued, so that a copy of the outgoing packet will be transmitted on the output mirror destination port. See the [Mirroring](#) chapter.
2. IP Header Update
For routed packets the IP checksum is updated after TTL update, as setup in [Egress Router Table](#).
3. Routed DA/SA MAC Update
For routed packets update the MAC addresses based on the Next Hop.
4. Egress Port VLAN
A VLAN header operation can be performed based on the physical output port. See the [VLAN Processing](#) chapter.
5. Egress Port Packet Type Filter
The egress packet type filter, setup through [Egress Port Configuration](#) per egress port, determines if the packet will be dropped or be allowed to be transmitted. See the [VLAN and Packet Type Filtering](#) chapter.
6. VRF Statistics
If the packet is routed it will be counted in [Transmitted Packets on Egress VRF](#) counter for the VRF it belongs to.
7. Egress VLAN Translation
Potentially replace the outgoing VID and Ethernet Type on a specific port with a specific VID. Uses a TCAM located in register [Egress VLAN Translation TCAM](#).
8. Reassemble Packet Headers
Depending on if the packet shall enter a tunnel or not this can be the final step in the egress processing which is to reassembly the outgoing (potentially inner) packet header.
9. Tunnel Entry
Result from packet processing, both ingress and egress, can result in that a packet shall enter a tunnel. This tunnel is described as a number of bytes to be added to the packet at certain points. There also exists options which allows the outer packet to copy certain data from the inner packet (such as TOS byte, next header).



Chapter 4

Latency and Jitter

This chapter is meant as an introduction to the causes of latency and jitter in the core. It gives some numbers, but mostly points out the general principles.

The switch has a fixed minimal latency, the bulk of which comes from the ingress and egress packet processing, the store-and-forward operation, and the dataflow registers between design units.

4.1 Latency

The major contributors to latency:

1. The Serial to Parallel converter (SP) gathers the data chunks from the MAC into wider cells.
2. The IPP has a fixed latency of 159 core clock cycles.
3. The queue engine stores the entire packet in buffer memory before adding it to the queues.
4. The EPP has a fixed latency of 76 core clock cycles.
5. Packet modifications that decrease the packet size (for example removing a VLAN) will cause a packet to be delayed one scheduling slot for certain packet sizes.

4.2 Jitter

There are three places (t_1 - t_3) in the core where latency jitter can be introduced. See Figure 4.1 on page 36.

- t1** In the SP the ports are visited in a fixed order, thus introducing a jitter the size of the port visitation period. There is also an asynchronous FIFO between the port and the core clock regions, adding one clock period (of the slowest clock) of jitter.
- t2** The egress scheduler visits the ports in a fixed order, introducing a jitter the size of the port visitation period.
- t3** The asynchronous FIFO between the core and port clock regions adds one core clock period (of the slowest clock) of jitter.

Note, though, that the core is dimensioned to handle even the worst case jitter without causing packet drops or increased IFG.

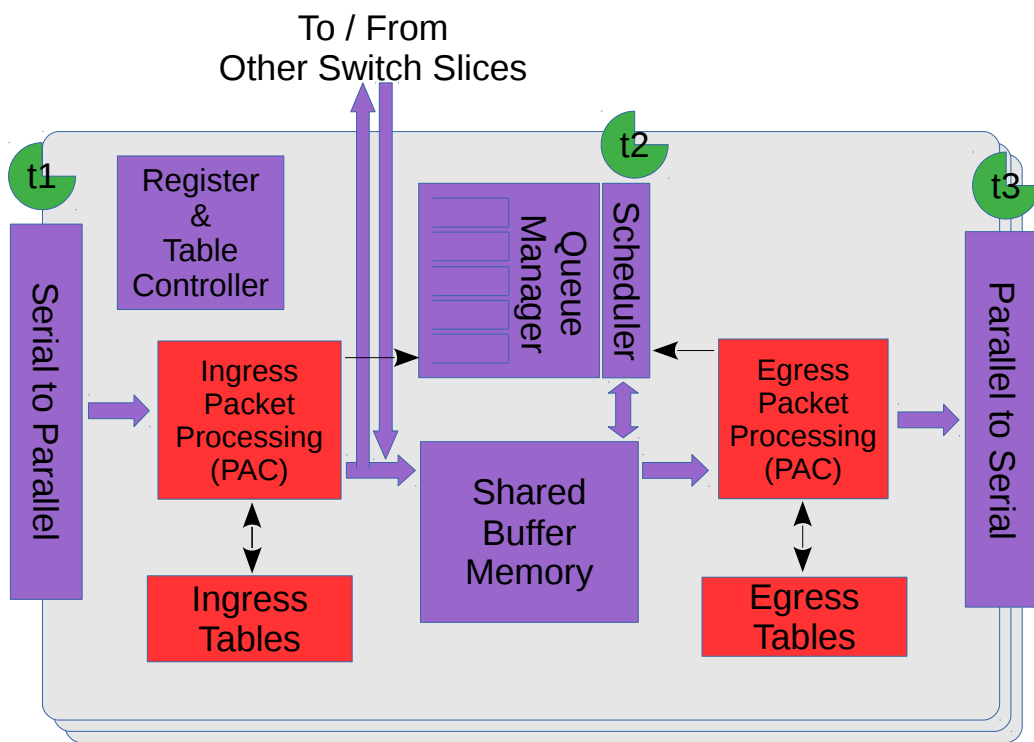


Figure 4.1: Jitter Overview

Chapter 5

VLAN Processing

5.1 Assignment of Ingress VID

All packets entering the switch will be assigned an ingress VID even if the incoming packet doesn't have a VLAN header. This is the VID used to lookup in the [VLAN Table](#).

The ingress VID assignment is processed in several steps. The initial assignment is controlled per source port by the [vlanAssignment](#) in the [Source Port Table](#) and then it can be updated in a number of ways ranging from L2 to L4 protocols.

5.1.1 VID Assignment from Packet Fields

Ingress VID can be assigned from certain packet fields, other than the packets incoming VID.

There exists a number of these field tables listed below:

- On the L2 MAC layer in [Ingress VID MAC Range Search Data](#) and its result table [Ingress VID MAC Range Assignment Answer](#), the search data can be either on source MAC or destination MAC ranges.
- On the Outer VID in [Ingress VID Outer VID Range Search Data](#) and its result table [Ingress VID Outer VID Range Assignment Answer](#). If the packet has no outer VID then this is skipped. There exists options if the packets VID shall be matched depending on if this is a S-tag or C-tag.
- On the Inner VID in [Ingress VID Inner VID Range Search Data](#) and its result table [Ingress VID Inner VID Range Assignment Answer](#). If the packet has no inner VID then this is skipped. There exists options if the packets VID shall be matched depending on if this is a S-tag or C-tag.
- On the Ethernet Type which is following the innermost VLAN tag. The setup is in [Ingress VID Ethernet Type Range Search Data](#) and its result table [Ingress VID Ethernet Type Range Assignment Answer](#).

VID Assignment Search Order

If there are matches in multiple tables then the "order" field determines which result to use. The result with the highest order value will be used. The search order within a table is not affected by the order field.

The search is carried out as follows:

1. The MAC ranges, defined in [Ingress VID MAC Range Search Data](#)
2. The Outer VID ranges, defined in [Ingress VID Outer VID Range Search Data](#)
3. The Inner VID ranges, defined in [Ingress VID Inner VID Range Search Data](#)

4. The Ethernet Type ranges, defined in [Ingress VID Ethernet Type Range Search Data](#)

5.1.2 Force Ingress VID from Ingress Configurable ACL

The ACL engine has an option to override the ingress VID assigned above. If the forceVidValid field in the [Ingress Configurable ACL N Small Table](#) is set to 1, the corresponding forceVid field will be used as the new ingress VID value. The same applies to the [Ingress Configurable ACL N Large Table](#) and [Ingress Configurable ACL N TCAM Answer](#) tables. The detailed L2 ACL match and action are described in the [Configurable ACL Engine](#) section.

5.2 VLAN membership

All packets entering the switch will be member of a VLAN, either assigned from the incoming VLAN headers or through a default configuration described below.

The VLAN membership defines which ports that are part of a VLAN. Packets belonging to a VLAN can only enter on the ports that are member of the VLAN.

The L2 switching can only send out packet on the ports that are members of the VLAN, including broadcast, multicast and flooding. This limitation does not apply to routed packets.

The VLAN membership also assigns a global identifier (GID) to a packet which is used during L2 lookup to allow multiple VLANs to share the same L2 tables.

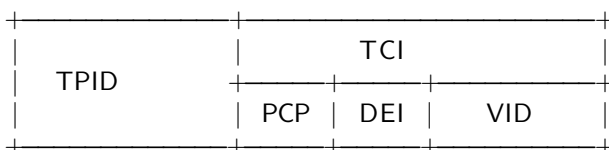
The VLAN membership also determines which multiple spanning tree (MSTP) a packet is part.

The egress queue priority can also be assigned from the VLAN membership (see chapter [21.1](#)).

5.3 VLAN operations

There are a number of operations that can be performed on the packet's VLAN headers such as push/pop etc. Multiple operations can be performed in sequence such that the resulting VLAN header stack from one operation becomes the input to the following operation. However the content of the VLAN headers do not come from previous VLAN operations, they are always created from the original incoming packet or from tables.

For reference here is the 802.1Q VLAN header:



When referring to outermost and innermost VLAN header, outermost means the first VLAN header that the packet decoding has identified as a VLAN header. Innermost means the second VLAN header as identified by the packet decoder.

The VLAN operations that can be performed are:

- Pop - The outermost VLAN header in the packet is removed.
- Push - A new VLAN header is added to the packet before any previous VLANs. It will become the new outer VLAN. The selection of each of the VLAN fields such as TPID, VID, PCP and DEI/CFI are configurable. These fields can either come from existing VLAN headers in the original incoming packet or from tables.
- Swap/Replace - The outermost VLAN header in the packet is replaced. The selection of each of the VLAN fields such as TPID, VID, PCP and DEI/CFI are configurable. These fields can either come from existing VLAN headers in the original incoming packet or from tables.



- Penultimate Pop - All VLAN headers (up to as many as supported by the packet decoder) are removed from the packet.

Figure 5.1 shows the effect of one of these operations on a packet.

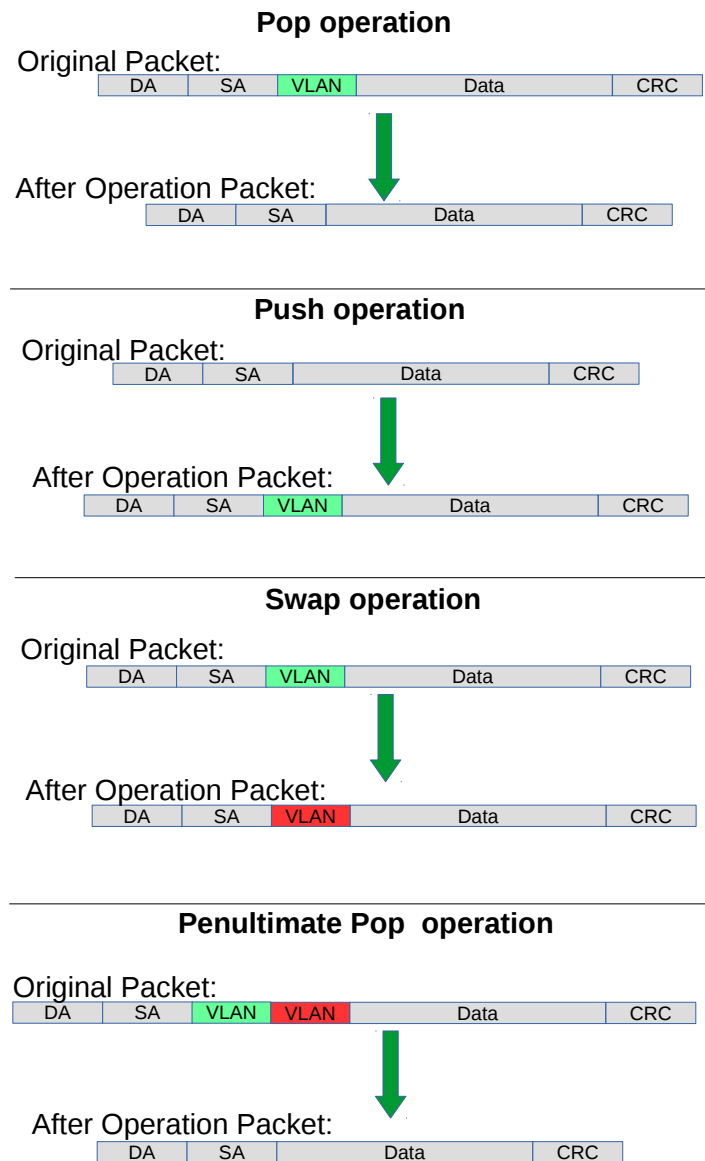


Figure 5.1: VLAN Packet Operations

5.3.1 Default VLAN Header

When a packet enters without a VLAN header an internal default VLAN header will be created. The internal header will have VID, CFI and PCP from [Source Port Table](#) fields [defaultVid](#), [defaultCfiDei](#), [defaultPcp](#).

The default VLAN header is only used in VLAN operations that selects data from the VLAN packet header.

5.3.2 Source Port VLAN Operation

A VLAN operation to be performed (e.g. push, pop, swap) can be selected by the [vlanSingleOp](#) field in [Source Port Table](#).



If the packet is routed this VLAN operation will not be performed.

5.3.3 Operation Based On Incoming Packets Number of VLANs

There exists a option which overrides the default [vlanSingleOp](#) field depending on the number of VLANs the packet has. This operation allows a user to set a specific operation depending on the number of VLANs the incoming packet has. This VID operation then overrides the default VID operation. This operation is setup in field [nrVlansVidOperationIf](#).

5.3.4 Configurable ACL VLAN Swap Operation

The [Ingress Configurable ACL N Small Table](#) , [Ingress Configurable ACL N Large Table](#) and [Ingress Configurable ACL N TCAM Answer](#) tables provides three fields `updateVid`, `updatePcp` and `updateCfiDei` to perform a VLAN swap operation. The VLAN type can also be changed using the `updateEType`. VLAN push and pop operations are not supported in this ACL.

If the packet is routed then the VLAN swap operation in the ACL will not be performed.

5.3.5 VLAN Table Operation

The [VLAN Table](#) defines the VLAN port membership, which GID (Global Identifier) to use in L2 lookups, the MSPT to use , if routing is allowed and a VLAN operation to be performed (e.g. push, pop or swap).

If the packet is routed then the VLAN operation from [VLAN Table](#) will not be performed.

5.3.6 VLAN Table VID Operation Based On the Packets Number of VLANs

There exists a option which overrides the default [vlanSingleOp](#) field depending on the number of VLANs the packet has. This operation allows a user to set a specific operation depending on the number of VLANs the incoming packet has after the source port operation push/pop/swap/penultimate pop has been done. The VID operation then overrides the default VID operation specified in field [vlanSingleOp](#) and all its data fields. This operation is setup in field [nrVlansVidOperationIf](#). This setting is done on a per port basis allowing each source port to have its own setting. Source port 0 is represented in bits [1:0] , Source port 1 is represented in bits [3:2] and so on.

5.3.7 Egress Port VLAN Operation

A VLAN operation to be performed (e.g. push, pop, swap) can be selected by the [vlanSingleOp](#) field in [Egress Port Configuration](#).

A pop operation is done on packets that match a specific VID if [enablePriorityTag](#) is set in [Source Port Table](#).

5.3.8 Egress Port VID Operation

[Egress Port VID Operation](#) provides an option to override the default [vlanSingleOp](#) depending on the number of VLANs the packet has and the ingress VID of the packet. Each entry of the [Egress Port VID Operation](#) register compares the egress port, ingress VID and VLAN tagging conditions and activate the corresponding VLAN operation from the first hit.

5.3.9 Egress Vlan Translation

This operation which is located in the egress path allows a replacement of the outermost VLAN Identifier in the packet. The egress port, the outermost VID of the packet after all VLAN operations and the outermost VID type (C or S tag) creates a lookup key to be used in a TCAM lookup located in [Egress VLAN Translation TCAM](#) which allows maskable bits for each entry..



5.3.10 Priority Tagged Packets

Priority tagged packets are packets that have a VLAN tag with VLAN ID equal to 0. The purpose of these are to extract the PCP bits and use as priority.

The priority extraction can be done as described in [21.1 Determine Egress Queue](#) section.

The priority tag can be ignored in all VLAN processing and finally removed on the egress if `enablePriorityTag` is set in [Source Port Table](#). Which VLAN ID that triggers this is configured in `priorityVid`

The priority extraction is not dependent on the `enablePriorityTag` setting.

5.3.11 Router VLAN Operations

- If a packet is routed then any VLAN headers in the incoming packet detected by the packet decoder will be removed on the egress.
- All other VLAN operations during ingress packet processing will not be done on routed packets.
- The routers next hop will point to the [Next Hop Packet Modifications](#) table which can specify up to two push VLAN operations to perform.
- The [Egress Port Configuration](#) VLAN operation is performed on routed packets after the VLAN operations specified in [Next Hop Packet Modifications](#).

5.3.12 VLAN Operation Order

All VLAN operations are performed in sequence on a packet. They follow the order as:

1. One of the four VLAN operations from:
 - [Source Port Table](#) VLAN operation.
 - Inner VLAN push operation from routers [Next Hop Packet Modifications](#).
2. One VLAN swap operation from:
 - `updateVid`, `updatePcp`, `updateCfiDei` or `updateEType` in the [Configurable ACL Engine](#).
3. One of the four VLAN operations from:
 - [VLAN Table](#) VLAN operation.
 - Outer VLAN push operation from routers [Next Hop Packet Modifications](#).
4. One of the four VLAN operations from:
 - [Egress Port Configuration](#) VLAN operation.

The input to the first VLAN operation is the incoming packet. The packet decoder identifies the position of the VLAN headers in the packet and this information is used for the subsequent VLAN operations.

The output from one VLAN operation is input to the next VLAN operation. For example if the first VLAN operation is a push and the second is a swap then the effect will be that the pushed header is replaced by the swap.

If a VLAN operation needs a VLAN header in the packet, i.e. a swap or a pop, and there is no VLAN header in the packet then the operation will not be performed.

5.3.13 VLAN Operation Examples

This process is first described informally with a few examples but to fully specify the behavior it is also described as pseudo code.

Here are examples of sequences of VLAN operations performed on packets with mixed VLANs and custom tags. The incoming packet headers, sequence of VLAN operations and outgoing packet header are briefly described.



'V1'..'V2' are VLAN tags in original packet
 'new V1'..'new V2' are VLAN tags that have been created by the VLAN operations

Example 1)

incoming packet:
 [DA][SA][V1]

VLAN operations: 1. swap new V1
 outgoing packet:
 [DA/SA][new V1]

Example 2)

incoming packet:
 [DA][SA][V1]

VLAN operations: 1. push new V1

outgoing packet:
 [DA/SA][new V1][V1]

Example 3)

incoming packet:
 [DA][SA][V1][V2]

VLAN operations: 1. push new V1

outgoing packet:
 [DA/SA][new V1][V1][V2]

Example 4)

incoming packet:
 [DA][SA][V1][V2]

VLAN operations: 1. pop

outgoing packet:
 [DA/SA][V2]

Example 5)

incoming packet:
 [DA][SA][V1][V2]

VLAN operations: 1. pop
 VLAN operations: 2. swap new V1
 VLAN operations: 3. push new V2

outgoing packet:
 [DA/SA][new V2][new V1]

5.3.14 VLAN Reassembly

The reassembly of the VLAN headers uses data from the packet decoding together with data from the VLAN operations to create the new packet headers.



The following is Python code that exactly models the reassembly operation. The process starts when the L3 and payload in the outgoing packet has been reassembled but before any VLAN or other L2 tags have been added.

The code uses the same incoming packet and VLAN operations as **Example 5)** in the previous section to illustrate the data structure.

```
# The design supports this number of VLAN tags in the ingress packet.
nr_of_ingress_vlans = 2

# Packet decoding results in a list of all VLAN tags from the ingress packet.
pkt_vlan_tags = [ 'V2', 'V1' ]

# Number of VLAN tags that will be used from the original packet. Before any
# VLAN operations this equals number of incoming VLANs, it could be decreased by
# swap or pop but can't be increased. When nr_of_new_vlans==0, pop or swap will
# decrement it. At any time popAll will set it to 0.
nr_of_pkt_vlans = 2

# Number of new VLAN tags to be used in the reassembly. Push and swap operations
# will increment this and at the same time the new VLAN to the end of new_vlans.
# popAll will set it to 0.
nr_of_new_vlans = 0

# New VLAN tags to be used in the reassembly.
new_vlans = []

# After all VLAN operation sequences: pop, swap new V1, push new V2, VLAN
# reassembly collects needed information to get started.
nr_of_pkt_vlans = 0
nr_of_new_vlans = 2
pkt_vlan_tags = [ 'V2', 'V1' ]
new_vlan_tags = [ 'new V1', 'new V2' ]

# At the starting point of re-assembling the VLAN tags the egress packet contains the
# updated packet after the original tags, i.e. L3/L4/payload.
egress_pkt = ['payload']

# Reassemble the tags with updated VLANs.
while nr_of_pkt_vlans > 0: # Egress packet has VLAN tags from ingress
    # Pop inner most tag from pkt_vlan_tags and insert it first in the egress_pkt
    egress_pkt.insert(0,pkt_vlan_tags[0])
    pkt_vlan_tags = pkt_vlan_tags[1:]
    nr_of_pkt_vlans -= 1

while nr_of_new_vlans > 0: # Egress packet has new VLAN tags
    # Insert a new VLAN first in the egress_pkt from internal VLAN stack.
    egress_pkt.insert(0,new_vlan_tags[0])
    new_vlan_tags = new_vlan_tags[1:]
    nr_of_new_vlans -= 1

# Now egress_pkt contains all updated VLAN headers and tags. After this new DA/SA
# and other new tags like to_cpu_tag is added to get the final egress packet.
```





Chapter 6

Switching

Most packets will be subjected to a L2 MAC destination address lookup to determine the destination egress port (or ports). These are the exceptions:

- Packet decoder determines that this protocol should be send to the CPU. See [Packet Decoder](#) chapter.
- A classification unit action dropped the packet, sent the packet to the CPU, or sent the packet to a specific egress port. See [Classification](#) chapter.
- The packet has a From CPU tag which allows the normal packet forwarding process to be bypassed. See [Packet From CPU Port](#) section.
- The packet is routed. See the [Routing](#) chapter.
- The packet is dropped earlier in the packet processing chain. See chapter [Ingress Packet Processing](#) for details.

6.1 L2 Destination Lookup

If none of the above applies a L2 MAC address destination lookup will be performed in the following manner:

- The GID is given by the [gid](#) field from the [VLAN Table](#) lookup. See the [VLAN Processing](#) chapter.
- The hash is calculated with {GID,DA MAC} as key (see [MAC Table Hashing](#)).
- The hash is used as index into the [L2 DA Hash Lookup Table](#). 4 entries are read out in parallel, each corresponding to a hash bucket.
- The bucket entries are all compared with the {GID,DA MAC} key and if one entry is equal to the key that entry is considered a match.
- The {GID, DA MAC} key is also compared with all the entries in the [L2 Lookup Collision Table](#) CAM. The CAM is searched starting from entry 0 and the first matching entry is treated as a match. Any following matching entries are ignored.
- Some entries in [L2 Lookup Collision Table](#) has per-bit masks. These are set up in the [L2 Lookup Collision Table Masks](#) registers. Using the mask an entry can define with single-bit granularity what shall be included in the comparison. A zero in the mask means that the corresponding bit shall be ignored, while a one means that the bit shall be compared.
- An entry in the [L2 DA Hash Lookup Table](#) is only compared if the corresponding valid bits are set. The valid bits are located in the [L2 Aging Table](#) , the [L2 Aging Status Shadow Table](#) and the [L2 Aging Status Shadow Table - Replica](#) . If all the valid bits are not set then this will result in a non-match even if the {destination MAC , GID} in the [L2 DA Hash Lookup Table](#) entry matches. For the collision CAM the valid bits are located in the [L2 Aging Collision Table](#) and [L2 Aging Collision Shadow Table](#). See figure 6.1.

- If both CAM and L2 hash tables return a match, the result from the CAM table will take precedence.
- Once the final entry has been determined, the result is read out from the **L2 Destination Table**. It has enough entries to fit the destinations for both the L2 hash table and the L2 CAM table. The L2 CAM table entries are located after the L2 hash table entries.
- If the **pktDrop** field in the **L2 Destination Table** is set the packet will be dropped.
- If the destination shall be a single port (i.e. it is not to be multicasted) then the **uc** field shall be set to one and the **destPort or mcAddr** field shall contain the egress port number.
- If a packet shall be sent to multiple output ports then the **uc** field shall be set to zero and the **destPort or mcAddr** field shall contain a pointer to an entry in the **L2 Multicast Table**. The entry in the **L2 Multicast Table** contains a portmask where bit 0 represents port 0, bit 1 port 1, and so on. A bit set to one results in the corresponding port receiving a packet.
- The DA MAC address ff:ff:ff:ff:ff:ff is the broadcast address, meaning that all the member ports in the VLAN (configured in the **VLAN Table vlanPortMask** field) will receive a packet.
- Normally the source port is excluded from the destination portmask. If that results in an empty destination port mask then the packet is dropped and counted in the **L2 Lookup Drop** register.
This behaviour can be changed using the **Hairpin Enable** register, allowing a packet to be switched to the same port it came in.
- Ports that are not members of the VLAN will be removed from the portmask. If there are no ports left in the port mask then the packet is dropped and counted in the **L2 Lookup Drop** register.
- If there is no hit in either the **L2 DA Hash Lookup Table** or the **L2 Lookup Collision Table**, then the packet will be flooded, i.e. sent out to all ports in the VLAN. This means that the port mask for the outgoing packet will be taken from the **vlanPortMask** field in the **VLAN Table**.
- If the **Flooding Action Send to Port** is enabled on this source port (using **enable** set to one) and the packet is flooded then the packet is sent to the destination port pointed to by the field **destPort** instead of being flooded to all ports part of the packets VLAN. The destination port does not need to be part of the packets VLAN group membership.
- If there is a hit then the hit bit in the **L2 Aging Table** is set to one.
- The final physical port is determined by the link aggregation. See chapter [Link Aggregation](#) for more information.
- Learning new unknown SA MAC addresses is described in chapter [Learning and Aging](#).

6.2 Software Interaction

Observe that L2 tables can not be directly written by software if learning engine is turned on. Doing so can cause packets to be dropped and/or flooded and the learning engine may stop working. See chapter [Learning and Aging](#) for information how to safely update the L2 tables.

6.3 L2 Action Table

There are two tables which allow detailed control for each packet depending on the source L2 MAC table result, the destination L2 MAC table result and the ingress and egress port which each has a configurable state. This is the L2 Action Table used for each egress port which the packet shall be sent to is defined in **L2 Action Table** and secondly the **L2 Action Table Source Port**. Both tables use a number of bits from the source port table, egress port state, SA and DA MAC lookups to form an address into the tables which is then read out and acted on. Each source port enables if the L2 Action tables shall be used or not using the field **enableL2ActionTable**. The L2 Action Tables can be used to permit specific frames from certain source ports to other destination ports using a filter defined in **Allow Special Frame Check For L2 Action Table**. There are 4 rules which are shared among all ports and pointed from the L2 Action Tables as a result by setting **useSpecialAllow** to one and then pointing to the rule using field **allowPtr**.



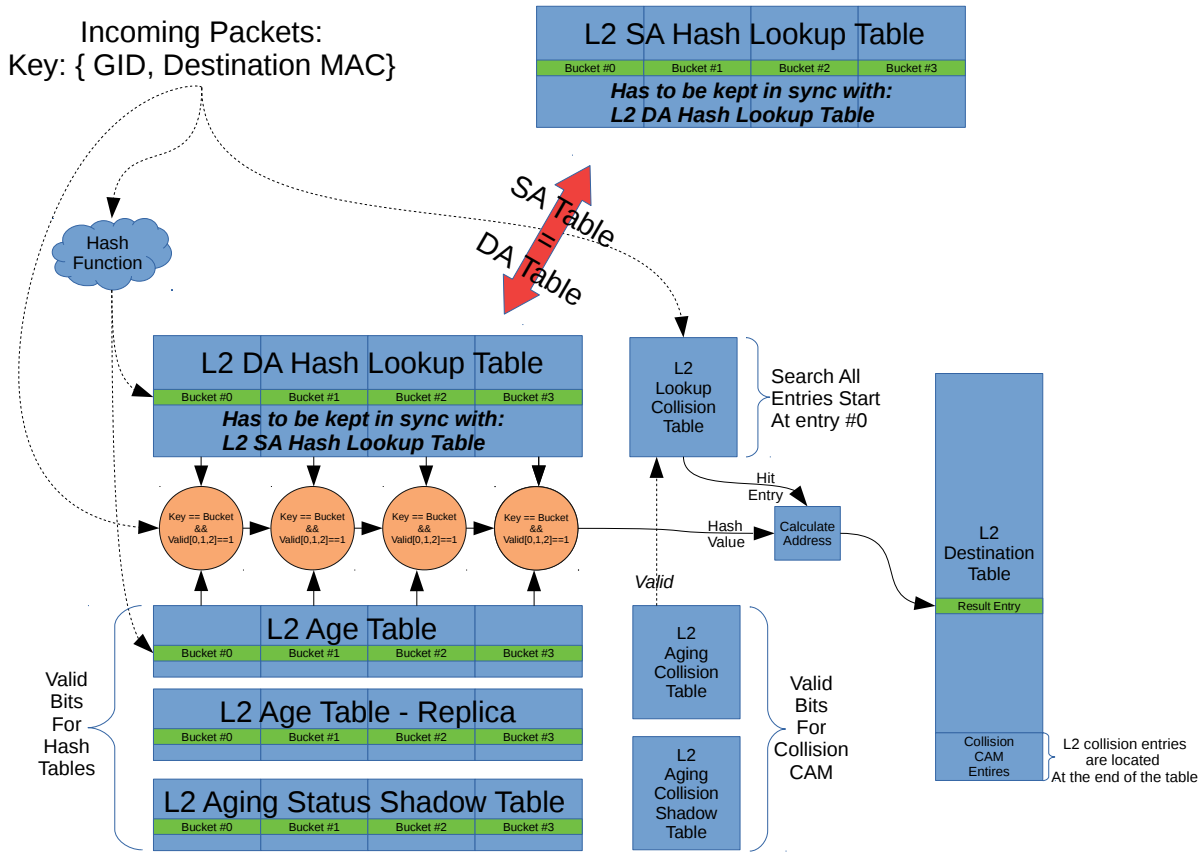


Figure 6.1: L2 Lookup Overview

If a packet is going to no egress ports (portmask==0) then none of the actions in the **L2 Action Table** will be carried out, while the **L2 Action Table Source Port** will always be carried out since a packet always comes in on a source port. Because of this the addressing is slightly different for these two table lookups.

The use cases for the tables is described below. Both tables have the same result actions.

6.3.1 Learning Unicast and Learning Multicast

As stated before the L2 Action Table can be used to stop learning on certain frames. There is an additional setting allowing the user to define if the learning is not to be allowed for unicast or multicast packets. Since a learning lookup is based on the Source MAC address this is also what is compared against. If the SA MAC is a multicast address then the **noLearningMc** field will be used to determine if the packet shall be learned or if SA MAC address is a unicast then the **noLearningUc** will determine if the packet shall be learned or not.

6.3.2 Drop and Learning

If a packet is dropped by the L2 Action Table the packet will still be learned. If you want the packets not to be learned then both **dropAll** and **noLearningUc** and **noLearningMc** should be turned on (set to one).

6.3.3 Priorities Between Actions

There are multiple actions from the L2 action table this section explains the order between them.

1. The drop special packet is first carried out and drops all instances of the packet



2. The drop port move then takes priority and drops all instance of the packet
3. The drop-all drops all instances of a packet however special type packets can still be accepted if they are setup to do so.
4. After the drops the send-to-CPU is carried out. Only a single copy will be sent to the CPU.

6.3.4 Using L2 Action Table for 802.1X

Simple Port Authentication

By using the source port bit **I2ActionTablePortState** and the egress port state bit in register **L2 Action Table Egress Port State** to indicate if a port is authenticated or not packets can be limited to communicate with other ports. This is done by setting up the different addresses in the L2 Action Table to do drop operations when a packet comes in from a non-authenticated port going to a authenticated port.

Port Authentication with MAC addresses

In order to allow already existing computers (MAC address) allow to pass through the switch without any problems the SA lookup result bit **I2ActionTableSaStatus** can be used indicate if this source MAC address (i.e. computer/end-station) has been authenticated or not on this port. A non-authenticated computer shall still be able to communicate with other ports which are not authenticated. Since the three bits partly forms the address into the L2 Action Table it is possible to form rules which when a packet is allowed to access other ports depending on what the state of these ports are and if the computer it wants to communicate with is known to the switch or not. The field **I2ActionTableDaStatus** can be used to further enhance the security wheather or not two computers shall be able to communicate.

Port Authentication Enhancements with Learning and Port-Move

As the network security needs to be enhanced further the L2 Action Table allows setting up rules if a packet coming in and going to different ports shall be able be able to be learned or if a already existing MAC address shall be able to be port moved.

Port Authentication Enhancements only allow certain traffic types

As the last enhancement there can be special rules formed which allows only certain packet types to pass on a port combination using the result options **useSpecialAllow** and **allowPtr**. This allowPtr points to general rules of which packet types to drop or to allow. This rules are setup in **Allow Special Frame Check For L2 Action Table**.



Chapter 7

Routing

This core supports IPv4 and IPv6 routing as well as MPLS switching.

The routing is disabled by default and needs to be setup from the configuration interface before it can be used. This core supports virtual router ports/functions (VRFs). The VRFs allow the core to handle multiple virtual routers sharing the same set of tables and register. A VRF identifier is used to determine which virtual router each table entry belongs to.

The routing is done separately from the L2 switching. There is no switching done before or after the router. The router is entered when a packets destination MAC address equals the routers MAC address. The packet exits the router directly to an egress port.

MPLS follows the same order of operations as IP routing and uses the same tables. The MPLS processing is therefore described here.

7.1 Order of Operation

Routing function is done after the L2 ACLs. The routing engine performs the following steps:

1. Check if the VLAN allows packets to be routed. If this is not the case normal L2 lookups will be done. This is specified by the **allowRouting** field in **VLAN Table**.
2. Compare the incoming packets MAC destination address with all the entries in the **Router Port MAC Address**. There are per source port option in field **selectMacEntryPortMask** which allows the compared MAC address to be different based on which source port the packet comes in. The alternative MAC address to compare is located in field **altMacAddress**. If no match then the routing function is skipped. If the router port search found a match then the packet enters the router with an assigned VRF from the table.
3. The carried packet type (IPv4, IPv6 or MPLS) is checked against the allowed type that are setup in **Ingress Router Table**. If the type is not allowed the packet will be dropped. There is a alternative to dropping the packets and instead send them to the CPU. This can be archived by setting the **sendToCpuOrDrop** bit to one.
4. If the incoming packets TTL is below the allowed TTL, as specified in **Ingress Router Table** then the packet is dropped.
5. To determine the packets destination/next hop the destination address combined with the assigned VRF is searched for in the **Hash Based L3 Routing Table** and in the **L3 Routing TCAM**. If there is a match in both the TCAM and the hash table then the hash entry is selected since the hash table always contains the longest prefix. For the hash based search the next hop result is setup in the **Hash Based L3 Routing Table** and for the LPM search it is setup in **L3 LPM Result**.

The difference between MPLS and IP search is that in MPLS the 20-bit MPLS label from the outermost MPLS header is used as destination address.

6. If there is a match in the routing tables and the ECMP is enabled in the matched entry (either the **useECMP** in the **Hash Based L3 Routing Table** or **useECMP** in the **L3 LPM Result** table) then ECMP next hop calculation is performed.

ECMP calculates a hash based on the IP source and destination addresses, the IP proto field, IP TOS and the TCP/UDP source port and destination port.

For MPLS the ECMP hash key consists of the outermost header and does not include embedded IP headers. The hash value is added as an offset to the **nextHopPointer** after masking (**ecmpMask**) and shifting (**ecmpShift**).

7. If there is no hit in the destination address search then the default next hop is used. The default is defined in **L3 Routing Default** per VRF. There are also options to drop the packet or send to CPU port.
8. IP statistics is updated in the **IP Unicast Received Counter**, **IP Unicast Routed Counter** and **Received Packets on Ingress VRF** registers. MPLS forwarded packets are only counted in **Received Packets on Ingress VRF**
9. The next hop from the previous steps is used as index into the **Next Hop Table**. The entries determine where to route the packet, which is either a single destination port or a pointer to a L2 multicast entry. There are also options to drop the packet or send to CPU port.

Each entry also contains a packet modification pointer which points to several tables that determines what header modification that should be done when the packet exits the router.

- The **Next Hop Packet Modifications** table determines what VLAN operations to perform when exiting the router. If the entry's valid bit is not set the packet will be send to the CPU.
- The **Next Hop DA MAC** which determines the destination MAC address to use in the outgoing packet.
- For MPLS the **Next Hop MPLS Table** determines what MPLS header modifications that should be done on the outgoing packets. These are described in detail in the register description and in the **MPLS** chapter.

The **srv6Sid** flag is the local instantiated SRv6 segment identifier that enables the packet modification on egress to update the IPv6 destination address to the next segment. When hitting the SRv6 segment identifier, a legal segment routing header needs to be provided, otherwise the packet will be send to the CPU instead.

10. An MTU check, as specified in the **Router MTU Table**, is performed on incoming routed packets. This check is executed by comparing the IPv4 Total Length field with the limit configured in field **maxIPv4MTU**, separately for each destination port and VRF. Similarly, the IPv6 Payload Length field is compared with field **maxIPv6MTU**. If either length field exceeds its respective limit, the packet will be forwarded to the CPU for further processing. Notably, the MTU check is not applied to MPLS packets.
11. When next hop hit status updates are enabled in the **Ingress Router Table** then each time a packet is routed using a **Next Hop Table** entry the corresponding status bit is set in the **Next Hop Hit Status**.
12. The ingress part of routing is now completed. This is followed by other ingress functions such as L3 ACL etc. Finally the packet is queued to one or multiple egress ports.
13. The egress processing of the routed packet performs the packet header modifications. First step is update of the TTL field which is controlled by the **Egress Router Table**.
14. There exists an option called **Next Hop Packet Insert MPLS Header** which enables a outgoing routed packet to add MPLS labels after the L2 / VLAN headers. This allows the router to enter a MPLS tunnel in order to reach the next hop though a MPLS network. If a packet is already a MPLS packet this option offers a way to insert extra MPLS headers on top of the MPLS label stack. NOTE: It is not possible to insert MPLS headers if the packet has a PPPoE header, If the packet is a PPPoE then no MPLS insertion is then carried out.



15. A new L2 header is constructed with a DA MAC from the **Next Hop DA MAC** table. The SA MAC will be the incoming DA MAC. The **Router Port Egress SA MAC Address** allows the user to insert an alternative SA MAC address instead of the normal which should have been the packets DA MAC address. This setting is done per egress port.
16. The routers VLAN operations are performed. See the **VLAN Processing** chapter.
17. The segment routing operations are performed if needed.
 - Decrement Segments Left by 1.
 - Copy Segment List[Segments Left] from the SRH to the destination address of the IPv6 header.
18. The IPv4 header checksum is recalculated.
19. Egress router statistics is updated in **Transmitted Packets on Egress VRF**.
20. Egress VLAN Translation is done using the Dleft lookup **17**, on the newly assigned outermost egress VID of the packet.
21. If the result from the **Next Hop Table** points to a tunnel entry using fields **tunnelEntry** then the tunnel entry is carried out after all the packet modifications has been done according to the router exit.
22. If the result from the next **Next Hop Table** points to a tunnel exit using fields **tunnelExit** then the tunnel exit is carried out before the packet is modified by the router. Please note that if the tunnel exit packet modifications are modifying the same fields as the router (SA/DA MAC and VLANs and TTL fields) then these fields will be overwritten by the router.
23. The egress ports VLAN operations are performed. See the **VLAN Processing** chapter.





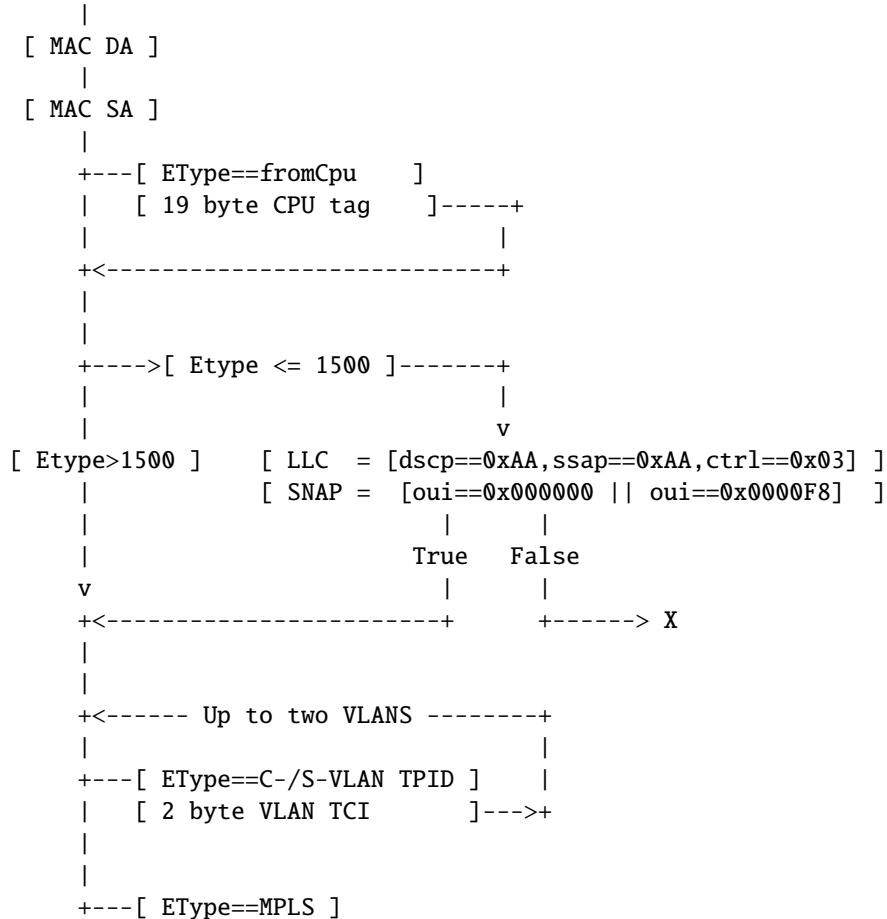
Chapter 8

Tunneling

The tunneling has two functions, first the tunnel exit, which enables the user to remove a number of bytes from a incoming packet, then process the packet as the inner layer packet. Secondly enter a tunnel in which a outgoing packet is encapsulated with a number of bytes somewhere in the packet.

8.1 Packet Decoder For Tunnel Exit

In the following diagram the decoding of the incoming packet header is described. The comparison used to determine protocol types are described as well as the order they are decoded. The end of decoding process is denote by an X.



```

| [ MPLS tag 1 ]--+
| [ MPLS tag 2 ]--+
| [ MPLS tag 3 ]--+
| [ MPLS tag 4 ]--+
| +-----+
| |
| +-----> X
|
+-->[ EType==unknown ]--> X
|
+-->[ EType==IPv4 ]-----+
+-->[ EType==IPv6 ]--+ | |
| | |
| v v v
| [ IPv6 Header ] [ IPv4 Header ]
| | |
+-----+-----+
|
+-->[ TCP Header ]--> X
+-->[ UDP Header ]--> X

```

There are options for the tunnel exit when it comes to recognizing the Ethernet Types for VLANs. The settings are located in register [L2 Tunnel Decoder Setup](#) which allows the user to setup custom types for C-tagged and S-tagged VLAN packets. If a packet originates from the CPU port and bears the from-CPU-Tag, with the Force Original Bit enabled within this header, there will be no execution of tunnel exit or tunnel entry.

8.2 Tunnel Exit

The tunnel exit can be done in multiple ways. In order for a packet to be enabled do a tunnel exit the field in register [Source Port Table](#) field [disableTunnelExit](#) in the incoming source port needs to be set to zero. The packet decoder described in [8.1](#) extracts the relevant fields from the incoming packet:

1. Packet is a SNAP/LLC Packet
2. Ethernet Type Field after possible VLAN headers
3. Ethernet Type for L3
4. If MPLS: Up to 4 MPLS headers.
5. If IPv4 then IPv4 Destination Address
6. If IPv4 then IPv4 Source Address
7. If IPv6 then IPv6 Destination Address
8. If IPv6 then IPv6 Source Address
9. L4 source port, if TCP or UDP packet
10. L4 destination port, if TCP or UDP packet
11. One bit to indicate that the incoming packet had a from CPU tag.

All of these fields are then looked up in the Tunnel Exit Table using the Dleft function described in [Dleft Tunnel Exit](#). If the first tunnel exit lookup has a hit then the packet will do a second tunnel lookup which can result in a tunnel exit. The second lookup is needed because some protocols require a second field to be looked up before a tunnel exit can be determined, example of these types of protocols are VxLAN and GRE-over-UDP. There also exists options which enables the user to not use the packet data for the second lookup, instead use data from the first lookup answer fields, thereby allowing the first lookup to be the only lookup which matters (second lookup will still be performed but data is controlled from first lookup).



Packets with From CPU Tag

When a packet matches the criteria for tunnel exit and is tagged with a 'from CPU' label, and if the 'force-original-packet' bit is set within this tag, the packet will not undergo tunnel exit. Consequently, any rules configured to forward such packets to the CPU upon a hit in the initial lookup but not in the second tunnel lookup will also not be executed.

Tunnel Exit Places in the packet

The tunnel exit operations can remove configurable number of bytes, a maximum of 256 bytes can be removed, at the following places:

1. At the beginning of the packet.
2. After the DA and SA MAC and the VLAN headers.
3. After the DA and SA MAC, the VLAN headers and IPv4,IPv6 headers .

8.2.1 To Not To Use Second Lookup

To only use the first lookup and not select any new data for the second lookup this is done by setting the direct bit to one (**direct**) and setting up the field `tblIndex`, This `tblIndex` field is then used to do the search in the **Second Tunnel Exit Lookup TCAM**. The result in the **Second Tunnel Exit Lookup TCAM Answer** tells the tunnel unit how to remove the data from the packet.

8.2.2 Use Second Lookup With Packet Data

If the user does not want to use the direct method but rather extract new data from the packet then the second lookup field is extracted on a byte boundry which is pointed out by the field **secondShift**, this second shift value can also take into account if the incoming packet has zero,one or two vlans by setting the field **secondIncludeVlan** to one (which will increase the value of **secondShift** with 4 for each VLAN.). For the value used in the second lookup (either from packet data or from result from first table lookup) there exists options if each bit should be used or not in the lookup (using a mask located in field **lookupMask**). The second tunnel exit lookup has a type field which comes from the first Dleft tunnel exit lookup result thereby allowing different tunnel exit types not to get a false positive.

8.2.3 How To Remove Data From Packet In A Tunnel Exit

Once the first and second tunnel exit lookups are done a tunnel exit is performed. How the tunnel exit is done is described by the result from the second tunnel exit lookup. The results are located in tables **Second Tunnel Exit Lookup TCAM** and **Second Tunnel Exit Lookup TCAM Answer**

There are important fields are **howManyBytesToRemove** and **removeVlan** which specifies which bytes to remove in the incoming packet starting from the position after the L2 DA/SA + VLANs. The **removeVlan** removes 1 or 2 VLANs in the coming packet.

There exists options if the second lookup should fail which allows the user to drop the packet (while the first was tunnel exit lookup was a hit). This is located in **Second Tunnel Exit Miss Action**. This action has a setting for each of the different packet keys which comes from the first tunnel exit lookup result.

The same operations done at ingress during a tunnel exit must be mirrored in the egress register **Egress Tunnel Exit Table** otherwise the packet will look different once its sent out. Which entry to use in the **Egress Tunnel Exit Table** is pointed to by the field **tunnelExitEgressPtr**.

Example 1) Remove IPv4 Header

Incoming packet:

```

+-----+-----+-----+-----+-----+
| DA | SA | VLAN | IPv4 Header | Rest of packet |
+-----+-----+-----+-----+-----+
|

```



```

      |
      v
Outgoing packet:
+-----+-----+-----+-----+
| DA | SA | Rest of packet |
+-----+-----+-----+-----+

In tunnel exit table: SA+DA+VLAN ID+IPv4 SA+IPv4 DA.
Remove from start byte:12 to end byte:72.
The removal is done setting the register howManyBytesToRemove = 20
and then setting the removeVlan = 1

```

Example 2) Remove Dual VLANs.

```

Incoming packet:
+-----+-----+-----+-----+-----+
| DA | SA | first VLAN | Second VLAN | Rest of packet |
+-----+-----+-----+-----+-----+
      |
      |
      v
Outgoing packet:
+-----+-----+-----+-----+
| DA | SA | Rest of packet |
+-----+-----+-----+-----+

In tunnel exit table: SA+DA+2 * VLAN Headers
Remove from start byte:12 to end byte:20.
The removal is done setting the register howManyBytesToRemove = 0
and then setting the removeVlan = 1

```

8.2.4 Packet Insertion and Removal Limits

For the core to operate correctly it needs enough bytes in the first part of the packet. The packet processing gets the first 256 bytes of the whole packet. Once a packet is passed to the egress processing pipeline 29 bytes of the total 256 bytes is consumed by an internal header. For tunnel exit this means that if the inner packet headers (L2+L3+L4) after a tunnel exit goes beyond 256 - 29 bytes minus the tunnel exit removed bytes then this inner packet will be dropped due to insufficient bytes to decode the packet.

8.2.5 Tunnel Exit Options

Besides the above tunnel exit operations there are also a number of other operations which can be done.

- Drop the packet. Using field `dontExit`.
- Set the VLAN table VID which shall be used. Using field `replaceVidset` to one and then setting the VID to be used in field `newVid`
- Do not do the tunnel exit. Using field `dontExit`.

8.2.6 Tunnel Exit from Tables

Tables such as VLAN, L2, L2 Multicastouting, L3 tables and ACLs have option to do a tunnel exit. If this packet already did a tunnel exit then the packet will be sent to CPU since the hardware can not process two tunnel exits after each other, the packet will be sent to the CPU with the reason code 3,889.

8.3 Tunnel Entry

Entering a tunnel allows adding protocol headers at the beginning of the packet, after the L2 headers (After Ethernet MAC DA/SA and VLANs) and finally after L3 (After IPv4/IPv6/MPLS headers, before L4 headers). The After L2



tunnel headers starts with an IP header (IPv4 or IPv6) followed by an optional UDP header. After IP/UDP headers additional headers can be inserted but the content of those headers are not modified by the switch.

The same table address is read out in all of the tunnel entry instructions [Tunnel Entry Instruction Table](#), [Beginning of Packet Tunnel Entry Instruction Table](#), [L2 Tunnel Entry Instruction Table](#) and [L3 Tunnel Entry Instruction Table](#)) pointed to by the tunnel entry pointers from L2,L3,ACL tables.

Original egress packet before tunnel header insertion:

```

+-----+-----+-----+-----+-----+-----+
| MAC DA | MAC SA | VLAN* | Orig EType | Original L3 | Original L4 |
+-----+-----+-----+-----+-----+-----+

```

After tunnel insertion at beginning of packet:

```

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| (NEW) At Beginnig Tunnel Header | orig. | orig. | | Original |Original|Original|
| MAC DA/SA | VLAN* | IP/MPLS Hdr | UDP | X | MAC DA | MAC SA | VLAN* | EType | L3 | L4 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

After tunnel insertion After L2 Headers:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
+ | | | (NEW)After L2 Tunnel Header | |Original|
| MAC DA | MAC SA | VLAN* | New EType | IP | UDP | X | Original L3 | L4 |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

After tunnel insertion After L3 Headers:

```

+-----+-----+-----+-----+-----+-----+-----+-----+
| | | | Updates in L3 Header: | (NEW) After L3 Tunnel Header | Original|
| MAC DA | MAC SA | VLAN* | -L4 Type and IP Length| X | L4 |
| | | | (original L3) | | |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The content of the inserted protocol headers is configured in the [Tunnel Entry Header Data](#) table.

The length of the IPv4 header is fixed at 20 bytes. The IPv6 header is 40 bytes. These can be followed by 8 bytes of UDP header.

The tunnel entry is done in the egress processing after all other packet modifications. For example any VLAN operations are done before tunnel headers are inserted. If the packet was routed the Next Hop packet modifications (such as IP header TTL update and MAC DA/SA update) will be done before the tunnel header insertion.

For tunnel entry after L2 the insertion point after Ethernet and VLAN headers is automatically identified and for tunnel entry after L3 headers the insertion point after the L2, L3 headers is automatically identified.

The tunnel inserted header can be updated with correct Payload Length/ Total Length fields if the fields [Beginning of Packet Tunnel Entry Instruction Table I3Type](#) is IPv6 or IPv4 or in [L2 Tunnel Entry Instruction Table](#) field [I3Type](#) is set to IPv4 or IPv6. For IPv4 header the Header Checksum is calculated based on the configured header but after updating the Total Length field. For these length fields and the checksum field the value stored in the [Tunnel Entry Header Data](#) is not used.

All other fields in the IP header are unchanged and taken directly from the [Tunnel Entry Header Data](#). It is up to the software configuration to create a valid IP header. This includes setting the Protocol/Next Header field if an UDP header follows the IP header.

After the inserted IP/UDP headers can follow additional headers up to the maximum width of the [Tunnel Entry Header Data](#) (80 bytes).

The tunnel insertion process will always perform the tunnel header insertion if instructed to by table actions in the ingress processing. There is no check at all of the content of the original protocol headers at this point.

For tunnel entry after L3 there exists options in which the preceeding IPv4 or IPv6 headers protocol type / next header byte will be updated. This is controlled by the [L3 Tunnel Entry Instruction Table](#) field [updateL4Type](#). Besides this the IPv4 or IPv6 length field is updated with the header added.

8.3.1 Tunnel Length Insertion

There exists a option to insert a length into the header data. This length field is first inserted ,by overwriting 2 bytes in the insertion data, defined in the [Tunnel Entry Header Data](#) which is then inserted into the packet.



Software needs to make room for the insertion data. There is no extra length added to the insertion data.

Example as follows: (Inserting at beginning of packet, same applies to all other insertions.)

Original egress packet before tunnel header insertion:

```
+-----+-----+-----+-----+-----+-----+
| MAC DA | MAC SA | VLAN* | Orig EType | Original L3 | Original L4 |
+-----+-----+-----+-----+-----+-----+-----+
```

Tunnel Header to be inserted.

```
+-----+
| Data |
+-----+
```

First insertion of length field in Tunnel Header is carried out:
(Data is written over in the Tunnel Header)

```
+-----+-----+-----+
| Data | Length |
+-----+-----+-----+
                                <-- 2 bytes-->
```

After tunnel insertion at beginning of packet:

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| <----- Data Inserted ----->| orig. | orig. |      | Original |Original|Original|
|                               | Length | MAC DA | MAC SA | VLAN* | EType  | L3     | L4     |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

8.3.2 Tunnel Entry Tables

A packet can enter the tunnel from a number of tables. Each table has a tunnel entry action bit and a pointer into the [Tunnel Entry Instruction Table](#), this table is the master table which then determines which of the tunnel entry tables [Beginning of Packet Tunnel Entry Instruction Table](#), [L2 Tunnel Entry Instruction Table](#) or [L3 Tunnel Entry Instruction Table](#) to use. This is determined by the field `tunnelEntryType`.

In the [Tunnel Entry Instruction Table](#) is a pointer to the tunnel header to be inserted, the length of the tunnel header. This table also contains a instruction which enables a 2-byte length field to be inserted into the tunnelHeader at any byte position. If this is used the bytes in this position will be overwritten.

1. Next Hop Table
The field `tunnelEntry` and field `tunnelEntryPtr` points to a tunnel entry instruction.
2. Result from a Ingress Configurable ACL
Result can point to a tunnel entry or a tunnel exit. The user can define if this should be done as a unicast or multicast tunnel entry. In a unicast entry all the packets use the same [Tunnel Entry Instruction Table](#) entry independent of the outgoing port while the multicast entries means that each destination port is used as a offset to the base pointer.
3. Result from a Egress Configurable ACL
Result can point to a tunnel entry or a tunnel exit. The user can define if this should be done as a unicast or multicast tunnel entry. In a unicast entry all the packets use the same [Tunnel Entry Instruction Table](#) entry independent of the outgoing port while the multicast entries means that each destination port is used as a offset to the base pointer.

The tunnel pointers from these tables can be used as unicast or multicast pointers. Multicast means that the tunnel entry can be configured differently for each egress port. When a pointer is of unicast type the pointer value is used to directly index the [Tunnel Entry Instruction Table](#).

If a pointer is of multicast type then the destination port number will be added to the pointer before index the [Tunnel Entry Instruction Table](#). This allows for using different tunnel headers for different ports.

8.3.3 Priority between Tunnel Exit and Tunnel Entry in Tables

Since the tunnel entry and tunnel exit can be pointed to by several tables what is the priority between them.



Table	Unicast or Multicast	Comment
Next Hop Table	Unicast or Multicast	A Next Hop Entry is unicast, however it can point to a L2 Multicast Entry.
Ingress ACL Result Tables	Unicast or Multicast	Enables the user to freely select if unicast or multicast.
Egress ACL Result Tables	Unicast or Multicast	Enables the user to freely select if unicast or multicast.

Table 8.1: Tunnel Entry Unicast or Multicast

- Egress Port Configuration Priority
If a port has set the [Egress Port Configuration](#) with tunnel entry or tunnel exit this will take priority over previous set tunnel exit or tunnel entry.
- Egress Port Configuration and Tunnel Exit Lookup
Between tunnel exit unit and tunnel exit from egress port configuration table then the egress port configuration table takes precedence. This means that what the processing done on ingress can alter from how the packet will actually look when it is sent out.
- Tunnel Exit Lookup and Tables Tunnel Exit
If both the tunnel exit lookup and tables tunnel exit says to do a tunnel exit then the packet will be sent to the CPU port for further checks by software.

8.3.4 Tunnel Entry and Routing with MTU check

Since a ACL or IP entry might call upon a packet to enter a tunnel this might mean that the outgoing IPv4 or IPv6 packet might be too long for the next hops MTU. This check can be turned on for each tunnel entry and it will only be checked if a packet is routed. If the packet is over the MTU then it will be removed from the output ports destination masks and a copy will be sent to the CPU. In order for this to work the table [Tunnel Entry MTU Length Check](#) must be setup to reflect the additional bytes being added to the IP packet headers.





Chapter 9

MPLS

This core is equipped with MPLS forwarding. The processing of MPLS packets follows the same pattern as IP routing, with the major difference that an MPLS header operation (such as push, pop, swap and penultimate pop) can be carried out. Since the order of operation for MPLS is almost identical to IP routing it is described in the [Routing](#) chapter.

9.1 MPLS Header Operations

In addition to the processing that is done for IP routed packets the MPLS router can perform operations on the MPLS header stack.

The [Next Hop MPLS Table](#) determines which operation to perform.

- Pop - The outermost MPLS header in the packet is removed.
- Push - A new MPLS header is added to the packet before any previous MPLS headers. The label for the new header and the source for the EXP bits are specified in the table entry.
- Swap/Replace - The outermost MPLS header in the packet is replaced. The label for the new header and the source for the EXP bits are specified in the table entry.
- Penultimate Pop - All MPLS headers (up to as many as supported by the packet decoder, see [Packet Decoding](#) chapter) are removed from the packet. In addition the Ethernet Type is set to IPv4 or IPv6, see the following section.
- Remapping of EXP bits in the outermost MPLS header. Either use the existing value, use from the table or use a remapping table [Egress Queue To MPLS EXP Mapping Table](#).

The [Egress MPLS TTL Table](#) determines which operation on the TTL field to perform when exiting the VRP, either decrement the TTL or set a new TTL. Each VRP can have their own setting.

9.2 MPLS Penultimate Pop

A normal Pop operation removes one MPLS header but leaves the Ethernet Type unmodified (identifying the packet as still being a MPLS packet).

The Penultimate Pop operation removes all MPLS headers and also updates the packets Ethernet Type. This assumes that the payload in the MPLS packet is an IP packet. The first nibble in the payload is then decoded (see [Packet Decoding](#) chapter) to determine if the packet is IPv4 or IPv6 and then the Ethernet Type is updated accordingly.

9.3 MPLS Header Insertion To Reach Next Hop

There exists an option called [Next Hop Packet Insert MPLS Header](#) which enables a outgoing routed packet to add up to MPLS labels after the L2 / VLAN headers. The operation is pointed out by the field [nextHopPacketMod](#) in table [Next Hop Table](#). If a packet is already a MPLS packet this option offers a way to insert extra MPLS headers on top of the MPLS label stack.

NOTE: It is not possible to insert MPLS headers if the packet has a PPPoE header. If the packet is a PPPoE then no MPLS insertion is then carried out.



Chapter 10

NAT - Network Address Translation

There are two functions that can determine if a NAT operation shall be performed, the Configurable Ingress ACL and the Configurable Egress ACL. Each of these point to a NAT operation table that will be performed in egress, [Ingress NAT Operation](#) and [Egress NAT Operation](#).

The ACL pointers points to a base index and the egress port number can be added to this base index. The register [NAT Add Egress Port for NAT Calculation](#) determines if this shall be done or not, there is one setting for ingress NAT and one setting for egress NAT. This is a global setting.

The Ingress ACL and Egress ACL has independent NAT operation tables and corresponding NAT actions.

An action is one of the following.

- Replace source IP address.
- Replace destination IP address.
- Replace TCP/UDP source port number.
- Replace TCP/UDP destination port number.

The two NAT operations are performed in the order ingress operation first followed by egress operation (in the case where both operations would modify the same packet field).

If the layer 4 type is TCP and IP address or TCP port number is changed then the TCP checksum is recalculated.

If the layer 4 type is UDP and IP address or UDP port number is changed then the UDP checksum is recalculated.

If an IP address is changed then the IP header checksum is recalculated.

When a NAT operation is perform the status registers [Egress NAT Hit Status](#) and [Ingress NAT Hit Status](#) are updated.

10.1 Ingress Packet Processing Option

Since the packet operations for NAT is carried out just before the packet is sent out there are cases where the user want the ingress routing and other processes to use the private or public IP address (and/or L4 address). This can be done by setting the [enableUpdateIp](#) or [enableUpdateL4](#) fields in one of [Ingress Configurable ACL N Small Table](#) , [Ingress Configurable ACL N Large Table](#) and [Ingress Configurable ACL N TCAM Answer](#) .

10.2 NAT Action Table Check

At the end of the ingress packet processing a NAT port operation check is done. This involves checking all the egress ports NAT state (in register [Egress Port NAT State](#)) and comparing them to the ingress port NAT state (in field [natPortState](#)) together with the NAT operations from ingress and egress ACL and if the packet was routed or switched. These five bits are used as a address into the table [NAT Action Table](#). For all the egress ports the

packet is going out on the table is checked and if any of the actions are send to cpu or drop this takes precedence and is carried out instead of sending out the packet on the already looked up ports. When a packet is sent to the CPU from the NAT Action table there are options if the packet should be the original packet or the modified packet, this is setup in **NAT Action Table Force Original Packet**, there is separate setting for each reason code enabling options when using the two different packets to CPU.

The priority of the **NAT Action Table** is as follows: (Only a single action is carried out.)

1. If all actions are No actions the packet is sent to egress.
2. If any action has the Sent to CPU then the packet will be sent to the CPU
3. If any action has drop then all instances are dropped and a counter is updated



Chapter 11

Mirroring

This core supports both input and output mirroring.

11.1 Input Mirroring

Input mirroring allows all packets received by an ingress port to be copied to an egress port without packet modifications.

- For each port, one input mirroring port can be configured through the [Source Port Table](#). The `inputMirrorEnabled` field enables a input mirror copy and send it to the port configured in the `destInputMirror` field.
- Packets hit in the [Configurable ACL Engine](#) can send an input mirror copy to the port configured in ACL's `destInputMirror` field if there is an enabled `inputMirror` action.

By default the input mirror copy will bypass any packet modification or drop decisions during the ingress or egress packet processing. Extra options are given in the [Source Port Table](#) to limit the range of the mirroring destination. `imUnderVlanMembership` only allows the input mirror copy to be sent to the members of the VLAN. `imUnderPortIsolation` only allows the input mirror copy to be sent to the destination that does not block the current source port from the [Ingress Egress Port Packet Type Filter](#). If a packet has an input mirror action from the ACL and its source port also enables input mirroring, the destination port of that copy is determined by the ACL result.

11.2 Output Mirroring

Output mirroring allows the user to select an egress port to be mirrored so that packet that is transmitted to that egress port can have a copy sent to an egress port. For each port, one output mirroring port can be configured through the [Output Mirroring Table](#):

1. The output mirroring functionality can be enabled per port using the `outputMirrorEnabled` field from the [Output Mirroring Table](#).
2. The port to which the mirror copy is sent is setup by the `outputMirrorPort` field in the [Output Mirroring Table](#). Multiple input ports can use the same output mirroring destination port.

With input mirroring, a port can be used to observe the traffic received by any port. With output mirroring, a port can be used to observe the traffic transmitted from any port. When there are multiple mirror copies requested or the CPU port is involved, the switch works as follows:

- An input mirrored packet can be output mirrored again.
- An output mirrored packet will not be mirrored again even if the destination port has output mirroring turned on.
- When a packet is mirrored to the CPU port, it will not carry an extra to-CPU tag since it is the copy of another packet.

It is possible that a packet is sent out in multiple copies on the same port when mirroring is turned on. In this case at most four instances of the same received packet can appear on an egress port. The order of the packet instances will be:

1. Normal switched/routed packet
2. Input mirror copy
3. Output mirror copy of the switched/routed packet
4. Output mirror copy of the input mirror copy

11.2.1 Requeueing FIFO

Output mirroring (and input mirroring to oneself) is accomplished by requeueing the packets in separate requeueing FIFOs after External Packet Processing. There is one requeue FIFO per egress port.

The egress scheduling will only see the packet at the head of each FIFO, but this packet will be selected before the packets belonging to the same queue in the normal egress queues.

This method of output mirroring means that:

1. The requeueing FIFOs are truly FIFOs per port, so there will be head-of-line blocking between packets of different egress queues mirrored to the same port.
2. The (up to three) mirroring copies for a single input packet are created in series. The first one is not created until the original packet has been scheduled and gone through Egress Packet Processing, the second one not until the first copy has been scheduled and gone through Egress Packet Processing and so on...
3. When several ports output mirror to the same port, or a higher speed port mirrors to a lower speed port (physical or shaped port speed) the requeueing FIFO for the mirroring destination port may fill up and cause packet drops.

The depth of the requeueing FIFOs is ten packets per egress port.

Drops due to the requeueing FIFOs overflowing are counted in the [Re-queue Overflow Drop](#) register.

Chapter 12

Link Aggregation

Link aggregation is a solution to bundle multiple ports into a higher bandwidth link. Each link aggregate is setup using the [Link Aggregation Membership](#) and [Link Aggregation To Physical Ports Members](#).

The [Link Aggregation Membership](#) register maps the incoming packets source port number to a link aggregate number. The link aggregate number is then used during ingress packet processing instead of source port/destination port numbers.

When a destination port (destination link aggregate number) has been determined by ingress packet processing the [Link Aggregation To Physical Ports Members](#) table maps the link aggregate number to which physical ports that are part of the link aggregate, i.e. the physical ports the packet shall be transmitted to.

Note that once link aggregation is enabled all ports needs to be setup as link aggregates, even if a port only has a single port part of its link aggregate. These ports are usually setup as having a one-to-one mapping, i.e. source port number, link aggregate number and physical port number are all the same.

The [Link Aggregation Membership](#) register and the [Link Aggregation To Physical Ports Members](#) register must be kept in sync by software.

To distribute the packets over the ports that are part of a link aggregate, a hash is calculated over some of the packets fields which is configured by register [Link Aggregation Ctrl](#). The hash value calculated is used to index the [Link Aggregate Weight](#) table which results in a port mask of the ports that will be used for this specific hash.

The ratio that each port in a link aggregate is used is determined by the number of times the port is set in the [Link Aggregate Weight](#) table divided by the number of entries in the table.

It is important to setup all entries in the [Link Aggregate Weight](#) table with one port set for each link aggregate, otherwise a certain hash value will have no port set thereby causing the packet to be dropped.

12.0.1 One-to-one Port Mapping

To setup a one-to-one mapping, then the bit which corresponds to the port number shall be set in the [members](#). This maps each link aggregate number to a physical port with the same number.

The [la](#) should then be set so that each source port number maps to the link aggregate with the same number, i.e. table entry 0 should hold a value of 0, table address 1 should hold a value 1, etc.

12.1 Example

Lets say that a link aggregate shall use physical ports 0,1,2 and each port shall have equal amount of traffic. Another link aggregate will use ports 6,7 also with equal load between the ports. The remaining ports are setup to be one-to-one. In this example these are ports 3,4 and 5, on a switch with 8 ports.

To setup the [Link Aggregation Membership](#) register we associate the source port with the link aggregate number that it belongs to. Ports 0,1,2 are part of link aggregate 0 and port 6,7 are part or link aggregate 1. The remaining ports are setup to use the same link aggregate number as the port number.

```

for port in [0,1,2]:
    rg_sp2la[port] = 0

for port in [6,7]:
    rg_sp2la[port] = 1

for port in [3,4,5]:
    rg_sp2la[port] = port

```

In [Link Aggregation To Physical Ports Members](#) we need to setup the relation from link aggregate number to physical port members.

```

rg_la2Phy[0] = 0b00000111 # la #0 = ports 0,1,2
rg_la2Phy[1] = 0b11000000 # la #1 = ports 6,7
rg_la2Phy[3] = 0b00001000 # la #3 = port 3
rg_la2Phy[4] = 0b00010000 # la #4 = port 4
rg_la2Phy[5] = 0b00100000 # la #5 = port 5

```

To setup how the traffic is distributed between the link aggregate member ports we first select which packet headers that will be used in the hash calculation. In this example we chose to select source MAC, destination MAC, IP address, L4, TOS value and vlan header as calculation base for the hash value.

```

rg_linkAggCtrl.useSaMacInHash = 1
rg_linkAggCtrl.useDaMacInHash = 1
rg_linkAggCtrl.useIpInHash = 1
rg_linkAggCtrl.useL4InHash = 1
rg_linkAggCtrl.useTosInHash = 1
rg_linkAggCtrl.useVlanInHash = 1

```

The table [Link Aggregate Weight](#) shall then be setup so that ports 0,1,2 have equal weight. This is accomplished by configuring so that the number of bits set for port 0 in all hash entries are equal to number of bits for port 1 and port 2. Which bits are set are not important as long as only one bit per entry are set and the total number of bits per port are equal.

If the hash of the packets fields are distributed evenly then 1/3 of the packets will be distributed to each of the three ports part of the link aggregate.

Similarly to setup a link aggregate on ports 6,7 with equal load between the ports then each entry in the [Link Aggregate Weight](#) table must have bit 6 or 7 set and with equal number of bits for the two ports.

The ratio for link aggregation 0, is 34% on port 0, 33% on port 1 and 33% on port 2. For link aggregation 1, it is 50% on each port.

```

for hash_index in range(0,85): # 34%
    r_hash2LA[hash_index] = 0b00000001 # port 0
for hash_index in range(86,170): # 33%
    r_hash2LA[hash_index] = 0b00000010 # port 1
for hash_index in range(171,256): # 33%
    f_hash2LA[hash_index] = 0b00000100 # port 2

for hash_index in range(128): # 50%
    r_hash2LA[hash_index] |= 0b01000000 # port 6
for hash_index in range(128,256): # 50%
    r_hash2LA[hash_index] |= 0b10000000 # port 7

for hash_index in range(256): # 100%
    r_hash2LA[hash_index] |= 0b00001000 # port 3
    r_hash2LA[hash_index] |= 0b00010000 # port 4
    r_hash2LA[hash_index] |= 0b00100000 # port 5

```



Finally when all the registers have been configured the link aggregation function is enabled in the [Link Aggregation Ctrl](#) register.

```
rg_linkAggCtrl.enable = 1
```

12.2 Hash Calculation

The hash key consists of the following fields in the order listed starting with the msb.

- MAC DA, 48 bits
- MAC SA, 48 bits
- VLAN ID, 12 bits
- IP TOS, 8 bits
- TCP/UDP Source Port, 16 bits
- TCP/UDP Destination Port, 16 bits
- IP Proto, 8 bits
- IPv4/IPv6 Source Address, 128 bits
- IPv4/IPv6 Destination Address, 128 bits
- Source Port, 5 bits

If a field is disabled in the [Link Aggregation Ctrl](#) register then the field in the hash key will be 0.

If a packet is routed then the MAC DA field will contain the next hop pointer instead of the MAC address and the VLAN ID will be 0.

The hashing is done in two steps, first the key is build, and the fields used in the key depends on the [Link Aggregation Ctrl](#) register, once the key is build then hash function is used to determine the address used ot lookup the [Link Aggregation To Physical Ports Members](#).

```
def build_key(daMac, useDaMacInHash,
             saMac, useSaMacInHash,
             vlanId, useVlanIdInHash,
             tos, useTosInHash,
             sp, useL4InHash,
             dp,
             proto,
             salp, useLpInHash,
             dalp,
             nextHop, useNextHopInHash,
             srcPort, routed):
    # This function builds the key to be
    # used for calculating the hash.
    final_data = 0
    if useDaMacInHash==0:
        daMac = 0
    if useNextHopInHash==0:
        nextHop = 0
    if routed==1:
        daMac = nextHop
        vlanId = 0

    final_data = final_data <<48
    final_data = final_data | daMac
    final_data = final_data <<48
    if useSaMacInHash==1:
        final_data = final_data | saMac
    final_data = final_data <<12
    if useVlanIdInHash==1:
```



```

    final_data = final_data | vlanId
final_data = final_data <<8
if useTosInHash==1:
    final_data = final_data | tos
final_data = final_data <<16
if useL4InHash==1:
    final_data = final_data | sp
final_data = final_data <<16
if useL4InHash==1:
    final_data = final_data | dp
final_data = final_data <<8
if useL4InHash==1:
    final_data = final_data | proto
final_data = final_data <<128
if useIpInHash==1:
    final_data = final_data | salp
final_data = final_data <<128
if useIpInHash==1:
    final_data = final_data | dalp
final_data = final_data <<5
final_data = final_data | srcPort
return final_data

```

```

def calcLaHash( key ):
    mask = (1 << 8) - 1
    _hash = 0
    for j in range(53):
        _hash = _hash ^ (key & mask)
        key = key >> 8
    return _hash & mask

```



Chapter 13

IEEE 1588/PTP Support

The core has support for IEEE 1588 / PTP with a number of features.

- Transfer of timestamp from RX MAC to CPU in the **To CPU Tag**.
- Identify PTP packets and send to CPU.
- Control of TX MAC action from settings in the **From CPU Tag**.
- Transfer of timestamp in the **From CPU Tag** to the TX MAC.
- Provide position of packet fields to the TX MAC needed for timestamp operation.

13.1 Timestamp from RX MAC

Each ingress port can receive a timestamp at the end of the packet. When the ingress port receives the end of the packet from the MAC, a timestamp valid flag indicates whether the packet is timestamped.

The timestamp size is 8 bytes.

13.1.1 Timestamp to the CPU

The RX MAC timestamp will be transferred to the CPU in the **Timestamp** field of the **To CPU Tag**. This will only be done when the packet is identified as a PTP packet by setting the ptp bit and the packet is sent to the CPU port with a **To CPU Tag**. For all other packets the timestamp will be discarded.

If redirecting to the CPU with ptp bit set without having a timestamp header on the source port will result in an invalid timestamp field in the **To CPU Tag** header.

13.2 PTP Frame Decoding

The switch supports PTP packets embedded in an 802.3 Ethernet frame, in an UDP/IPv4 frame or in an UDP/IPv6 frame.

PTP Header Field		byte position
transportSpecific	messageType	byte 0
reserved	versionPTP	byte 1
...	...	byte 2-6
correctionField		byte 8-15
...	...	byte 16-33
originTimestamp		byte 34-43

Table 13.1: PTP Header Format

MAC DA	MAC SA	EtherType=0x88F7	PTP
--------	--------	------------------	-----

Table 13.2: PTP over 802.3 Ethernet

13.2.1 PTP over 802.3 Ethernet

The packet decoder identifies PTP packets embedded in 802.3 Ethernet frames by the Ethernet Type. There is no comparison of the Ethernet destination address.

In order to be sent to the CPU any function (except input mirroring) that sends to the CPU port can be used. For example the 1588 standard multicast group addresses (01-1B-19-00-00-00, 01-80-C2-00-00-0E) can be set in the [L2 Destination Table](#) and point to entries in the [L2 Multicast Table](#). For the link local multicast (01-80-C2-00-00-0E) that should be dropped by bridges, only the CPU port should be set in the [mcPortMask](#). For the general multicast group address (01-1B-19-00-00-00) that should be broadcasted, then set all ports including the CPU port in the mask.

The [ptp](#) bit in the [To CPU Tag](#) will be set when the Ethernet Type matches the PTP type.

13.2.2 PTP over UDP

MAC DA	MAC SA	EtherType	IPv4	UDP	PTP
--------	--------	-----------	------	-----	-----

Table 13.3: PTP over UDP/IPv4

MAC DA	MAC SA	EtherType	IPv6	UDP	PTP	Checksum Correction
--------	--------	-----------	------	-----	-----	---------------------

Table 13.4: PTP over UDP/IPv6

PTP embedded in IPv4/IPv6 UDP can be identified with an L3 ACL rule and sent to the CPU using the [sendToCpu](#) action. The [ptp](#) action must also be set in order for the [ptp](#) bit in the [To CPU Tag](#) to be set together with a valid [Timestamp](#) field.

13.3 Software Control of TX MAC PTP Actions

The TX MAC needs to perform the following PTP actions.

- TX MAC updates timestamp in outgoing packet.
- TX MAC produces timestamp to be read by software.
- TX MAC updates correction field in outgoing packet with current time minus software time from the timestamp in the [From CPU Tag](#).

These actions are controlled by software by sending PTP packets from the CPU port with a [From CPU Tag](#). In the [From CPU Tag](#) header there are fields that will be transferred directly to the transmit MAC on dedicated signals (see [Packet Interface](#)).

- *oupd_ts_ps_N* - this signals will be set when the From CPU Tag field [upd_ts](#) is set. This is used to tell the transmit MAC that it should update the packets originTimestamp field.
- *oupd_cf_ps_N* - this signals will be set when the From CPU Tag field [upd_cf](#) is set. This is used to tell the transmit MAC that it should update the correctionField.
- *ots_ps_N* - this signal will have the value of the [From CPU Tag ptp_ts](#) field and should be used by the transmit MAC when updating the correctionField.
- *ots_to_sw_ps_N* - this signal will have the value of the [From CPU Tag ts_to_sw](#) field. This is used to tell the transmit MAC that it should create a timestamp of the current packet and transfer the timestamp to software. The switch is not involved in the transfer of the timestamp to software.



13.3.1 Packet Updates by the Transmit MAC

When the transmit MAC updates a PTP packet it needs to know position of the fields in the packet. This information is decoded by the switch and passed to the transmit MAC on dedicated ports.

- IPv4/UDP checksum field.
- IPv6/UDP checksum correction field (last 2 bytes in IPv6/UDP packet).
- PTP originTimestamp field.
- PTP correctionField.

When the transmit MAC updates a PTP packets and PTP is embedded in UDP/IP then the UDP checksum needs to be updated.

- For IPv4/UDP packets the UDP checksum field is zeroed by the MAC and therefore needs the position of the UDP checksum field.
- For IPv6/UDP it is forbidden to use zero checksum. Instead the last two bytes of the PTP packet is used to correct the checksum. The MAC therefore needs position of the UDP checksum field and the position of the second-to-last byte of the packet. (see IETF RFC 7821 - UDP Checksum Complement)

The transmit MAC also needs the position of the originTimestamp and correctionField. The position of the originTimestamp is provided to the MAC and from that position the MAC can calculate the position of the correctionField since that is always in the same relative position.

All this information is transferred to the MAC on dedicated signals (see [Packet Interface](#)).

- *oudp4_ps_N* - when this is set the packet is a UDP packet encapsulated in IPv4.
- *oudp6_ps_N* - when this is set the packet is a UDP packet encapsulated in IPv6.
- *oudp_csum_ps_N* - this is the first byte of the UDP Checksum field relative to the first byte of the packet.
- *ots_pos_ps_N* - this is the first byte of the originTimestamp field in a PTP packet relative to the first byte of the packet. This position is correct for all three encapsulation types.
- *oudp_corr_ps_N* - this is the first byte of the UDP checksum correction field. This field is always the last two bytes of the packet.

13.4 Support for Ordinary Clock

In this section is described how to implement the PTP packet handling for Ordinary Clock mode.

13.4.1 Master sending Sync

Software sends a PTP Sync packet to the CPU port with a [From CPU Tag](#). In the [From CPU Tag](#) the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port are included.

In 1-step mode the outgoing frames timestamp field is updated by the MAC with the timestamp. The timestamp is not used by software.

The TX MAC will get the position of the timestamp field from the switch.

If the packet is an IP/UDP packet then the checksum needs to be update by the MAC since the PTP header is changed. The MAC will get the position of the checksum field from the switch.

If PTP is embedded in IPv4/UDP then the UDP checksum field is cleared by the MAC. If it's IPv6/UDP then UDP checksum is not allowed to be cleared and instead the last two bytes in the frame is padding used for checksum adjustment. The MAC will get the position of the checksum adjustment field from the switch.

In 2-step mode the timestamp from the TX MAC is read out by software and the outgoing frame is not modified by the MAC. The [From CPU Tag](#) must control the MAC to produce a timestamp for software.

13.4.2 Slave receiving Sync

The RX MAC timestamps all packets. The timestamp must be prepended to the frames before they enter the switch. The switch port must be configured to receive the prepended timestamp.

Software needs to configure the switch to direct the Sync frame to the CPU port with a [To CPU Tag](#). The ptp bit must be set so that the timestamp that was prepended to the frame is sent to the CPU in the [To CPU Tag](#).



13.4.3 Slave sending DelayReq

Software sends a PTP DelayReq packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port.

The TX MAC must produce a timestamp of this packet. The timestamp from the TX MAC is read out by software and the outgoing frame is not modified by the MAC.

13.4.4 Master receiving DelayReq

The hardware mechanisms used are exactly as in Slave receiving Sync.

13.4.5 Master sending DelayReply

Software sends a PTP DelayReply packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set.

There is no timestamp needed for this frame so the TX MAC is not directed to produce any timestamp.

13.4.6 Slave receiving DelayReply

Software needs to configure the switch to direct the DelayReply frame to the CPU port. The timestamp produced by the RX MAC is not used and the **To CPU Tag** therefore does not need to include the timestamp.

13.5 Support for 1-step Peer to Peer

13.5.1 Initiator sending PDelayReq

Software sends a PTP PDelayReq packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port.

The TX MAC must produce a timestamp of this packet. The timestamp from the MAC is read out by software and the outgoing frame is not modified by the MAC.

13.5.2 Peer receiving PDelayReq

The hardware mechanisms used are exactly as in Slave receiving Sync.

13.5.3 Peer sending PDelayResp

Software sends a PTP PDelayReq packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port.

The TX MAC must produce a timestamp of this packet.

In 1-step mode the outgoing frames correction field is updated by the MAC with the difference between the produced timestamp and software supplied timestamp (from a received PDelayReq). The produced timestamp is not used by software. The TX MAC will get the position of the correction field from the switch.

13.5.4 Initiator receiving PDelayResp

Software needs to configure the switch to direct the PDelayResp frame to the CPU port. The ptp bit must be set so that the timestamp that was prepended to the frame is sent to the CPU in the **To CPU Tag**.

Chapter 14

Classification

14.1 L2 Classification

- L2 Destination MAC range classification is setup in table [Reserved Destination MAC Address Range](#).
 - The table is searched starting from entry 0.
 - When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated.
 - If multiple ranges are matched, any matching range that sets drop will cause a drop.
 - Any match that sets sendToCpu will cause send to CPU (this has priority over drop).
 - When multiple ranges that match has set the forceQueue then the highest numbered entry will determine the value.
- L2 Source MAC range classification is setup in table [Reserved Source MAC Address Range](#).
 - The table is searched starting from entry 0.
 - When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated.
 - If multiple ranges are matched, any matching range that sets drop will cause a drop.
 - Any match that sets sendToCpu will cause send to CPU (this has priority over drop).
 - When multiple ranges that match has set the forceQueue then the highest numbered entry will determine the value.
- If the destination MAC address bits [47:8] matches the [L2 Reserved Multicast Address Base](#) then bits [7:0] of the destination MAC address is used as a index in the table [L2 Reserved Multicast Address Action](#) which determines what action to take on the packet. Actions are set per source port and can either be to drop the packet or to send it to the CPU.

14.2 Configurable Ingress ACL Engine

The ingress ACL engine uses a configurable selection of fields from the incoming packet headers, from L2 fields to L4 fields. From the selected fields a hash table lookup is then done using [D-left hashing](#). The hashing is combined with a TCAM to resolve hash collisions and to enable per entry masking of data. Each of the hash tables can also be masked, but only a single mask can be applied for all data in a hash table.

There are 4 parallel ACL engines that each can perform one lookup per packet. All lookups are done in parallel and then there is a post processing of all the matching results to determine what actions to perform. There can be multiple actions taken for a single packet. How the actions are determined when there are multiple matches are described below.

14.2.1 Field Selection

For each source port the [useAcl/N](#) field in the [Source Port Table](#) configures if the incoming packets shall be subjected to an ACL lookup. By default the ACL is turned off.

If the ACL is turned on then the field [aclRule/N](#) is used as a pointer into [Ingress Configurable ACL N Rules Setup](#). This determines which fields that are used in the ACL lookup for this source port.

Each ACL engine has its own unique fields which can be selected. These are listed below. A field is selected by setting the corresponding bit in the fieldSelectBitMask.



ACL Engine	Width of Search Data	Fields to select from	Nr of Rules (Fields) to maximum use	Number of Parallel Hash Tables	Small Table Entries	Large Table Entries	TCAM Entries
0	330	14	7	4	256	2048	16
1	135	33	7	2	8	128	8
2	540	28	20	2	0	0	24
3	80	10	10	2	0	0	16

Table 14.1: Ingress ACL Engine Settings

Pre Lookup for Configurable Ingress ACL Table 0

This ACL engine has a pre-lookup. This is done to enable a different rule on how to build the ACL fields to be selected. If this lookup does not result in a valid rule pointer then the rule pointer from the source port table will be selected. The prelookup is setup in [Ingress Configurable ACL 0 Pre Lookup](#)

Packet Field	Size in Bits	Description
Source Port Bits	2 bits	The source port bits from source port table preLookupACLBits .
Type of L3 Packet	2 bits	The packets L3 Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Others.

Fields for Configurable Ingress ACL Table 0

The following fields can be selected for Configurable Ingress ACL Table 0, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	MAC DA	48	Always valid	The packets destination MAC address.
1	MAC SA	48	Always valid	The packets source MAC address
2	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
3	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
4	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
5	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
6	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
7	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
8	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
9	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
10	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6



Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
11	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
12	Source Port	5	Always valid	The source port of the packet.
13	Rule Pointer	3	Always valid	The rule pointer (index in the Ingress Configurable ACL N Rules Setup).

14.2.2 Example Of Selecting Fields For Configurable Ingress ACL Table 0

Since this ACL engine can select up to 7 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 330 bits in search key looks like when different fields are selected.

Example ACL with Ethernet Type

In this example we only want to create a rule with one field which is the Ethernet Type. This means that the fieldSelectBitmask, which is 14 bits , will be set as follows 1000000000 in binary format (Hex value of 0x200) and the lookup data will be located as follows:

0	Ethernet Type	Valid
-	Width : 16	1
17	16 1	0 0

Table 14.4: Hash Key Example for Ethernet Type

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destination MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 14 bits , will be set as follows 1000000011 in binary format (Hex value of 0x203) and the lookup data will be located as follows:

0	Ethernet Type	MAC DA	MAC SA	Valid
-	Width : 16	Width : 48	Width : 48	3
115	114 99	98 51	50 3	2 0

Table 14.5: Hash Key Example for Simple L2 ACL

Example of L3 IPv4 ACL

In this example we want to create a rule which with four L3 fields which are Destination IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 14 bits , will be set as follows 100100001100 in binary format (Hex value of 0x90c) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	IPv4 SA	L4 Protocol	Valid
-	Width : 2	Width : 32	Width : 32	Width : 8	4
78	77 76	75 44	43 12	11 4	3 0

Table 14.6: Hash Key Example for L3 IPv4 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destination Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask,



which is 14 bits , will be set as follows 1100111000000 in binary format (Hex value of 0x19c0) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 5	Width : 2	Width : 8	Width : 16	Width : 16	5
52	51 47	46 45	44 37	36 21	20 5	4 0

Table 14.7: Hash Key Example for L4 ACL

Example of Ingress NAT Entry

In this example we want to create a rule where the result would be used to change source IP address and/or source L4 Address. This means that the fieldSelectBitmask, which is 14 bits , will be set as follows 1110001000100 in binary format (Hex value of 0x1c44) and the lookup data will be located as follows:

0	Source Port	L3 Type	IPv4 SA	L4 Type	L4 Source Port	Valid
-	Width : 5	Width : 2	Width : 32	Width : 3	Width : 16	5
63	62 58	57 56	55 24	23 21	20 5	4 0

Table 14.8: Hash Key Example for Ingress NAT Entry

Pre Lookup for Configurable Ingress ACL Table 1

This ACL engine has a pre-lookup. This is done to enable a different rule on how to build the ACL fields to be selected. If this lookup does not result in a valid rule pointer then the rule pointer from the source port table will be selected. The prelookup is setup in [Ingress Configurable ACL 1 Pre Lookup](#)

Packet Field	Size in Bits	Description
Source Port Bits	2 bits	The source port bits from source port table preLookupACLBits .
Number of VLANS	2 bits	The packets number of incoming VLANs.
Type of L3 Packet	2 bits	The packets L3 Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Others.
Type of L4 Packet	3 bits	The packets L4 Type 0 = Not known. 1 = Is IPv4 or IPv6 but type is not any L4 type in this list. 2 = UDP 3 = TCP 4 = IGMP 5 = ICMP 6 = ICMPv6 7 = MLD

Fields for Configurable Ingress ACL Table 1

The following fields can be selected for Configurable Ingress ACL Table 1, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	TCP Flags	9	When packet has a L4 TCP protocol and is not a fragment.	The tcp flags for the packet. Bit 0 : ns, Bit 1: cwr, Bit 2: ece, Bit 3: urg, Bit 4: ack, Bit 5: psh, Bit 6: rst, Bit 7:syn, Bit 8: fin
1	MAC DA	48	Always valid	The packets destination MAC address.
2	MAC SA	48	Always valid	The packets source MAC address



Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
3	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
4	Has VLANs	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANs in packet
5	Outer VLAN Tag Type	1	When packet has an outer VLANs.	When the packet has an outer VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
6	Inner VLAN Tag Type	1	When packet has an inner VLAN.	When the packet has an inner VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
7	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
8	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.
9	Inner VID	12	When packet has a two VLANs.	The packets innermost VLAN Identifier (VID).
10	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.
11	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
12	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
13	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
14	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
15	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
16	Outer MPLS	20	When L2 frame holds a MPLS packet.	Outermost MPLS label.
17	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
18	TTL	8	When packet is a IPv4,IPv6 or MPLS	IPv4, IPv6 or MPLS Time-To-Live (TTL) byte.
19	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
20	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
21	MLD Address	128	When packet is a IPv6 and the ICMPv6 type is equal to 130,131,132	The MLD headers Multicast Address field.



Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
22	ICMP Type	8	When L4 packet is a ICMP packet	ICMP Type.
23	ICMP Code	8	When L4 packet is a ICMP packet	ICMP Code.
24	IGMP Type	8	When L4 packet is a IGMP	IGMP Type.
25	IGMP Group Address	32	When L4 packet is a IGMP	IGMP Group Address.
26	IPv6 Flow Label	20	When a packet is a IPv6.	IPv6 Flow Label.
27	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
28	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
29	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6
30	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
31	Source Port	5	Always valid	The source port of the packet.
32	Rule Pointer	3	Always valid	The rule pointer (index in the Ingress Configurable ACL N Rules Setup).

14.2.3 Example Of Selecting Fields For Configurable Ingress ACL Table 1

Since this ACL engine can select up to 7 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 135 bits in search key looks like when different fields are selected.

Example ACL with IPv4 DA

In this example we only want to create a rule with one field which is the IP DA. This means that the fieldSelectBitmask, which is 33 bits , will be set as follows 10000000000000 in binary format (Hex value of 0x2000) and the lookup data will be located as follows:

0	IPv4 DA	Valid
-	Width : 32	1
33	32 1	0 0

Table 14.11: Hash Key Example for IPv4 DA

Example with Destination MAC Address and Outer VLAN VID

In this example we want to create a rule which with two fields which are destination MAC address and outermost VLAN Identifier. This means that the fieldSelectBitmask, which is 33 bits , will be set as follows 1010 in binary format (Hex value of 0xa) and the lookup data will be located as follows:



0	MAC DA	Outer VID	Valid
-	Width : 48	Width : 12	2
62	61 14	13 2	1 0

Table 14.12: Hash Key Example for Destination MAC Address and Outer LAN VID

Example of Complex L2 ACL

In this example we want to create a rule which with six L2 fields which are Destination MAC address, source MAC address and Ethernet Type, inner and outer VLANs. The rule pointer would be used to enable different number of VLANs. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 33 bits, will be set as follows 1000100000000000000000001000001110 in binary format (Hex value of 0x11000020e) and the lookup data will be located as follows:

0	MAC DA	MAC SA	Ethernet Type	Outer VID	Inner VID	Rule Pointer	Valid
-	Width : 48	Width : 48	Width : 16	Width : 12	Width : 12	Width : 3	6
145	144 97	96 49	48 33	32 21	20 9	8 6	5 0

Table 14.13: Hash Key Example for Complex L2 ACL

Example of L3 IPv4 ACL

In this example we want to create a rule which with four L3 fields which are Destination IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 33 bits, will be set as follows 10010000000000000011000000000000 in binary format (Hex value of 0x48003000) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	IPv4 SA	L4 Protocol	Valid
-	Width : 2	Width : 32	Width : 32	Width : 8	4
78	77 76	75 44	43 12	11 4	3 0

Table 14.14: Hash Key Example for L3 IPv4 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destination Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 33 bits, will be set as follows 11001000000110000000000000000000 in binary format (Hex value of 0xc8180000) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 5	Width : 2	Width : 8	Width : 16	Width : 16	5
52	51 47	46 45	44 37	36 21	20 5	4 0

Table 14.15: Hash Key Example for L4 ACL

Example of Openflow Entry

In this example we want to create a rule which looks like an Openflow entry. This can be done by selecting source port, destination MAC, source MAC, Ethernet Type, inner VLAN, outer VLAN, L3 Type, IPv4 SA, IPv4 DA, L4 protocol, L4 Source port and L4 Destination port and finally the rule pointer. All in all 13 fields are selected. This means that the fieldSelectBitmask, which is 33 bits, will be set as follows 111011000000110000011001000001110 in binary format (Hex value of 0x1d818320e) and the lookup data will be located as follows:



0 - 263	Source Port Width : 5 262 258	MAC DA Width : 48 257 210	MAC SA Width : 48 209 162	Outer VID Width : 12 161 150	Inner VID Width : 12 149 138	Ethernet Type Width : 16 137 122	L3 Type Width : 2 121 120
IPv4 SA Width : 32 119 88	IPv4 DA Width : 32 87 56	L4 Protocol Width : 8 55 48	L4 Destination Port Width : 16 47 32	L4 Source Port Width : 16 31 16	Rule Pointer Width : 3 15 13	Valid 13 12 0	

Table 14.16: Hash Key Example for Openflow Entry

Example of Ingress NAT Entry

In this example we want to create a rule where the result would be used to change source IP address and/or source L4 Address. This means that the fieldSelectBitmask, which is 33 bits, will be set as follows 11100000000010000001000000000000 in binary format (Hex value of 0xe0081000) and the lookup data will be located as follows:

0 - 63	Source Port Width : 5 62 58	L3 Type Width : 2 57 56	IPv4 SA Width : 32 55 24	L4 Type Width : 3 23 21	L4 Source Port Width : 16 20 5	Valid 5 4 0
--------------	-----------------------------------	-------------------------------	--------------------------------	-------------------------------	--------------------------------------	-------------------

Table 14.17: Hash Key Example for Ingress NAT Entry

Pre Lookup for Configurable Ingress ACL Table 2

This ACL engine has a pre-lookup. This is done to enable a different rule on how to build the ACL fields to be selected. If this lookup does not result in a valid rule pointer then the rule pointer from the source port table will be selected. The prelookup is setup in [Ingress Configurable ACL 2 Pre Lookup](#)

Packet Field	Size in Bits	Description
Source Port Bits	2 bits	The source port bits from source port table preLookupACLBits .
Number of VLANS	2 bits	The packets number of incoming VLANS.
Type of L3 Packet	2 bits	The packets L3 Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Others.

Fields for Configurable Ingress ACL Table 2

The following fields can be selected for Configurable Ingress ACL Table 2, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	TCP Flags	9	When packet has a L4 TCP protocol and is not a fragment.	The tcp flags for the packet. Bit 0 : ns, Bit 1: cwr, Bit 2: ece, Bit 3: urg, Bit 4: ack, Bit 5: psh, Bit 6: rst, Bit 7:syn, Bit 8: fin
1	MAC DA	48	Always valid	The packets destination MAC address.
2	MAC SA	48	Always valid	The packets source MAC address
3	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
4	Has VLANS	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANS in packet



Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
5	Outer VLAN Tag Type	1	When packet has an outer VLANs.	When the packet has an outer VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
6	Inner VLAN Tag Type	1	When packet has an inner VLAN.	When the packet has an inner VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
7	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
8	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.
9	Inner VID	12	When packet has a two VLANs.	The packets innermost VLAN Identifier (VID).
10	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.
11	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
12	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
13	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
14	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
15	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
16	Outer MPLS	20	When L2 frame holds a MPLS packet.	Outermost MPLS label.
17	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
18	TTL	8	When packet is a IPv4,IPv6 or MPLS	IPv4, IPv6 or MPLS Time-To-Live (TTL) byte.
19	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
20	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
21	IPv6 Flow Label	20	When a packet is a IPv6.	IPv6 Flow Label.
22	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
23	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
24	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6
25	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS.
26	Source Port	5	Always valid	The source port of the packet.
27	Rule Pointer	2	Always valid	The rule pointer (index in the Ingress Configurable ACL N Rules Setup).

14.2.4 Example Of Selecting Fields For Configurable Ingress ACL Table 2

Since this ACL engine can select up to 20 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 540 bits in search key looks like when different fields are selected.

Example ACL with TOS Byte

In this example we only want to create a rule with one field which is the TOS. This means that the fieldSelectBitmask, which is 28 bits , will be set as follows 100000000000000000 in binary format (Hex value of 0x20000) and the lookup data will be located as follows:

0	TOS	Valid
-	Width : 8	1
9	8 1	0 0

Table 14.20: Hash Key Example for TOS Byte

Example with Destiantion MAC Address and Outer VLAN VID

In this example we want to create a rule which with two fields which are destiantion MAC address and outermost VLAN Identifier. This means that the fieldSelectBitmask, which is 28 bits , will be set as follows 1010 in binary format (Hex value of 0xa) and the lookup data will be located as follows:

0	MAC DA	Outer VID	Valid
-	Width : 48	Width : 12	2
62	61 14	13 2	1 0

Table 14.21: Hash Key Example for Destiantion MAC Address and Outer LAN VID

Example of Complex L2 ACL

In this example we want to create a rule which with six L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type, inner and outer VLANs. The rule pointer would be used to enable different number of VLANs. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 28 bits , will be set as follows 10001000000000000001000001110 in binary format (Hex value of 0x880020e) and the lookup data will be located as follows:



0	MAC DA	MAC SA	Ethernet Type	Outer VID	Inner VID	Rule Pointer	Valid
-	Width : 48	Width : 48	Width : 16	Width : 12	Width : 12	Width : 2	6
144	143 96	95 48	47 32	31 20	19 8	7 6	5 0

Table 14.22: Hash Key Example for Complex L2 ACL

Example of L3 IPv6 ACL

In this example we want to create a rule which with four L3 fields which are Destination IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 28 bits , will be set as follows 10010000001100000000000000 in binary format (Hex value of 0x240c000) and the lookup data will be located as follows:

0	L3 Type	IPv6 DA	IPv6 SA	L4 Protocol	Valid
-	Width : 2	Width : 128	Width : 128	Width : 8	4
270	269 268	267 140	139 12	11 4	3 0

Table 14.23: Hash Key Example for L3 IPv6 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destination Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 28 bits , will be set as follows 11001011000000000000000000 in binary format (Hex value of 0x6580000) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 5	Width : 2	Width : 8	Width : 16	Width : 16	5
52	51 47	46 45	44 37	36 21	20 5	4 0

Table 14.24: Hash Key Example for L4 ACL

Example of Openflow Entry

In this example we want to create a rule which looks like an Openflow entry. This can be done by selecting source port, destination MAC, source MAC, Ethernet Type, inner VLAN, outer VLAN, L3 Type, IPv4 SA, IPv4 DA, L4 protocol, L4 Source port and L4 Destination port and finally the rule pointer. All in all 13 fields are selected. This means that the fieldSelectBitmask, which is 28 bits , will be set as follows 1110110110000011001000001110 in binary format (Hex value of 0xed8320e) and the lookup data will be located as follows:

0	Source Port	MAC DA	MAC SA	Outer VID	Inner VID	Ethernet Type	L3 Type
-	Width : 5	Width : 48	Width : 48	Width : 12	Width : 12	Width : 16	Width : 2
262	261 257	256 209	208 161	160 149	148 137	136 121	120 119
IPv4 SA	IPv4 DA	L4 Protocol	L4 Destination Port	L4 Source Port	Rule Pointer	Valid	
Width : 32	Width : 32	Width : 8	Width : 16	Width : 16	Width : 2	13	
118 87	86 55	54 47	46 31	30 15	14 13	12 0	

Table 14.25: Hash Key Example for Openflow Entry

Example of Ingress NAT Entry

In this example we want to create a rule where the result would be used to change source IP address and/or source L4 Address. This means that the fieldSelectBitmask, which is 28 bits , will be set as follows 11100001000000100000000000 in binary format (Hex value of 0x7081000) and the lookup data will be located as follows:



0	Source Port	L3 Type	IPv4 SA	L4 Type	L4 Source Port	Valid
-	Width : 5	Width : 2	Width : 32	Width : 3	Width : 16	5
63	62 58	57 56	55 24	23 21	20 5	4 0

Table 14.26: Hash Key Example for Ingress NAT Entry

Fields for Configurable Ingress ACL Table 3

The following fields can be selected for Configurable Ingress ACL Table 3, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	L2 Packet Flags	4	Always valid	Type of L2 Packet . Bit [0] Is if the packet has a SNAP header. Bit [1] Is If the L2 packet has a PPPoE header with Ethernet Type==0x8863. Bit [2] Is If the L2 packet has a PPPoE header with Ethernet Type==0x8864. Bit [3] Is if the packet is a PPPoE but the carried type is not IPv4 or IPv6.
1	IPv4 Options	5	Always valid	Bit [0] is if the IPv4 header fragment offset field is not zero. Bit [1] is if the IPv4 header length != 20 bytes. Bit [4:2] IPv4 Header Flags in the header (the three bit flags for fragmentation).
2	TCP Flags	9	When packet has a L4 TCP protocol and is not a fragment.	The tcp flags for the packet. Bit 0 : ns, Bit 1: cwr, Bit 2: ece, Bit 3: urg, Bit 4: ack, Bit 5: psh, Bit 6: rst, Bit 7:syn, Bit 8: fin
3	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
4	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
5	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
6	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
7	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
8	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6
9	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.



14.2.5 Example Of Selecting Fields For Configurable Ingress ACL Table 3

Since this ACL engine can select up to 10 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 80 bits in search key looks like when different fields are selected.

Example ACL with TOS Byte

In this example we only want to create a rule with one field which is the TOS. This means that the fieldSelectBitmask, which is 10 bits , will be set as follows 1000 in binary format (Hex value of 0x8) and the lookup data will be located as follows:

0	TOS	Valid
-	Width : 8	1
9	8 1	0 0

Table 14.28: Hash Key Example for TOS Byte

Example of Exception ACL

In this example we want to create an rule where exception packets are selected they can then be send to CPU or dropped. This means that the fieldSelectBitmask, which is 10 bits , will be set as follows 1100000111 in binary format (Hex value of 0x307) and the lookup data will be located as follows:

0	L2 Packet Flags	IPv4 Options	TCP Flags	L4 Type	L3 Type	Valid
-	Width : 4	Width : 5	Width : 9	Width : 3	Width : 2	5
28	27 24	23 19	18 10	9 7	6 5	4 0

Table 14.29: Hash Key Example for Exception ACL

14.2.6 ACL Search

The hash key is used to perform a lookup using the D-left hashing function described in detail in chapter [D-left Lookup](#).

Before the hash key is used the mask in [Ingress Configurable ACL N Search Mask](#) is applied.

D-left calculates two hash values from the hash key. These hash values are then used to index the [Ingress Configurable ACL N Small Table](#) and [Ingress Configurable ACL N Large Table](#) . The hash calculations are described in section [Hash function for Configurable ACL](#).

In addition to the D-left search the hash key is also used to search in the [Ingress Configurable ACL N TCAM](#) .

14.2.7 ACL Actions

Once a hit has been determined by any of the searches above, the answer is read out from the corresponding answer entry. If it was a D-left hash hit then the answer actions is part of the hash memories ([Ingress Configurable ACL N Small Table](#) , [Ingress Configurable ACL N Large Table](#)). If it was a hit in the TCAM then the [Ingress Configurable ACL N TCAM Answer](#) is used.

The behavior for multiple hits is configured in [Ingress Configurable ACL N Selection](#) .

The statistics counter which can be updated are located in the [Ingress Configurable ACL Match Counter](#)

14.3 Multiple ACL Lookups

The section above describes a single ACL Lookup. There are however 4parallel ACL lookups. The functionality in the different lookup engines is the same with the exception that ACL engine 0 has seperate keys for IGMP, ICMP or MLD packets which are not available in the other engines.



Each of the ACL engines has its own rule configuration as well as its own hash and TCAM tables. The hash and TCAM table sizes and search data width for the different engines are as follows.

By using the same rules for multiple engines the table space for a rule can be extended.

14.3.1 Multiple Actions

If the parallel ACL engines have multiple matches the result actions from each search engine can take effect. How multiple actions are handled depends on the type of action.

Any Match

If one or more ACL engines matches and has this action set then the action will take effect.

Action Field	Ingress Acl 0 Has Ac- tion	Ingress Acl 1 Has Ac- tion	Ingress Acl 2 Has Ac- tion	Ingress Acl 3 Has Ac- tion
ptp	No	Yes	Yes	No
noLearning	No	Yes	Yes	No
dropEnable	Yes	Yes	Yes	Yes
sendToCpu	Yes	Yes	Yes	Yes

Table 14.30: Actions that will take effect if one or more is set.

First Match or Priority

If multiple ACL engines matches and has this action set then the value from the lowest numbered engine will be used. If an entry has the priority field set this value will be used and the values which do not have priority set will be ignored. If multiple matches have the priority field set then value from the highest numbered engine will be used.

Counter Update

All matches that have counter update action, [updateCounter](#) set will take effect. Each counter pointed to will be updated. If multiple actions point to the same counter then the counter value will only be incremented by one.

Send To Port

All matches that have an action [sendToPort](#) will take effect by setting the port number in the packet destination port mask, possibly resulting in a multicast.

Send To CPU

If any match has the [sendToCpu](#) action set it will take effect. When the To CPU Tag is used the reason code will indicate table index in the lowest numbered engine.

Ingress Admission Control Pointer

If there are multiple matches with actions to set the MMP pointer, `mmpPointer` then the selection will be done based on the `mmpOrder` field. This selection is described in [Ingress Admission Control](#).

Update IP Action

In some engines there can also be actions to update the IP fields. Since these actions are only available in one ACL engine there is no need to resolve multiple hits. If an action is enabled and the entry is hit it will take effect.



Enable Field	Priority Field	Value Field	Ingress Acl 0 Has Action	Ingress Acl 1 Has Action	Ingress Acl 2 Has Action	Ingress Acl 3 Has Action
forceVidValid	forceVidPrio	forceVid	No	Yes	Yes	No
forceQueue	forceQueuePrio	eQueue	Yes	Yes	Yes	Yes
forceColor	forceColorPrio	color	Yes	Yes	Yes	Yes
mmpValid	mmpOrder	mmpPtr	Yes	Yes	Yes	Yes
updateCfiDei	cfiDeiPrio	newCfiDeiValue	No	Yes	Yes	No
updatePcp	pcpPrio	newPcpValue	No	Yes	Yes	No
updateVid	vidPrio	newVidValue	No	Yes	Yes	No
updateEType	ethPrio	newEthType	No	Yes	Yes	No
imPrio	inputMirror	destInputMirror	Yes	Yes	Yes	No
natOpValid	natOpPrio	natOpPtr	Yes	Yes	Yes	No
tunnelEntry	tunnelEntryPrio	tunnelEntryPtr	No	Yes	Yes	No
sendToPort	N/A	tunnelEntryUcMc	Yes	Yes	Yes	Yes
metaDataValid	metaDataPrio	destPort	Yes	Yes	Yes	Yes
updateCounter	N/A	metaData	Yes	Yes	Yes	No
enableUpdateIp	N/A	counter	Yes	Yes	Yes	No
enableUpdateL4	N/A	updateSaOrDa	Yes	Yes	Yes	No
updateTosExp	N/A	newIpValue	Yes	Yes	Yes	No
		updateL4SpOrDp	Yes	Yes	Yes	No
		newL4Value	Yes	Yes	Yes	No
		newTosExp	Yes	Yes	Yes	No

Table 14.31: The lowest numbered takes effect if no priority else the highest numbered with priority set.

14.3.2 Default Port ACL action

When a port has the field [enableDefaultPortAcl](#) set then once a packet misses the ingress ACL lookup, on this source port, this action will be carried out. The action to be carried out is specified in the register [Source Port Default ACL Action](#). The actions are the same which can be done for the ACL Lookup. If the bit is set in field [forcePortAclAction](#) then all packets coming in on this source port are subjected to the actions specified in [Source Port Default ACL Action](#). This force ACL default action overrides all other ingress ACL actions/decisions.

14.4 Configurable Egress ACL Engine

The egress ACL engine uses a configurable selection of fields from the incoming packet headers, from L2 fields to L4 fields. From the selected fields a hash table lookup is then done using [D-left hashing](#). The hashing is combined with a TCAM to resolve hash collisions and to enable per entry masking of data. Each of the hash tables can also be masked, but only a single mask can be applied for all data in a hash table.

There are 2 parallel ACL engines that each can perform one lookup per packet. All lookups are done in parallel and then there is a post processing of all the matching results to determine what actions to perform. There can be multiple actions taken for a single packet. How the actions are determined when there are multiple matches are described below.

ACL Engine	Width of Search Data	Fields to select from	Nr of Rules (Fields) to maximum use	Number of Parallel Hash Tables	Small Table Entries	Large Table Entries	TCAM Entries
0	135	18	7	4	256	1024	16
1	540	20	20	2	0	0	16

Table 14.32: Egress ACL Engine Settings



14.4.1 Field Selection

Which fields that will be used in the ACL search is configured in the [Egress Configurable ACL N Rules Setup](#) table. To determine which rule in the table to use the forwarding result from routing and switching is input to a search in [Egress ACL Rule Pointer TCAM](#).

The rule pointer determined through this search is then index into [Egress Configurable ACL N Rules Setup](#) table. This table determines which fields that will be part of the hash key in the ACL search.

The possible fields to select are shown below for each ACL engine.

Determining Rule

The forwarding result fields that are used to search in the [Egress ACL Rule Pointer TCAM](#) are listed below. There is also a mask field for each of the search data fields allowing a selection of which bits in a field that should be compared.

Field	Description
<i>destPortMask</i>	The packets egress ports, one bit per port.
<i>routed</i>	The packet was routed.
<i>vrf</i>	The VRF used when routed.
<i>flooded</i>	The packet was flooded due to L2 table miss.
<i>ucSwitched</i>	The packet was L2 switched to a unicast destination port.
<i>mcSwitched</i>	The packet was L2 switched to a multicast group.
<i>vid</i>	The index used in the VLAN table lookup.
<i>L3 Type</i>	The incoming packets L3 type. IPv4, IPv6 , MPLS or other.
<i>L4 Type</i>	The incoming packets L4 type. TCP,UDP,IGMP,ICMP,ICMPv6,MLD etc.
<i>srcPort</i>	The packets source port.

Table 14.33: Fields used in the rule search.

The TCAM is searched starting at entry 0 and the first matching entry is used. The result is then taken from the [Egress ACL Rule Pointer TCAM Answer](#) table at the corresponding position. The result is a rule pointer into the [Egress Configurable ACL N Rules Setup](#) tables.

If there is no match in the TCAM the rule pointer 0 will be used. The rule setup can thus not be used to disable the ACL search.

Each Egress ACL engine has a separate rule table and separate pointers to each acl rule table.

Creating the hash key

All the bits from the fields selected in a rule are concatenated into a hash key. The hash key is used in several places.

- From the hash key two hash indexes are calculated which points into the [Egress Configurable ACL N Small Table](#) and [Egress Configurable ACL N Large Table](#) .
- Secondly the hash key is stored in the compareData field of the hash table entries.
- If a [Egress Configurable ACL N TCAM](#) entry is used the packet data keys are stored in the compareData field of that entry.
- When searching the tables a hash key is constructed from the incoming packets decoded packet fields. The appropriate valid bits are set.

Following the valid bits are the field data in the order that the fields are selected in [Egress Configurable ACL N Rules Setup](#) .

Selectable Packet Fields

The table below lists which fields that are possible to select along with a description on when the fields are valid.



A selected field will only result in a match if the incoming packet has the correct protocol type for the selected field, as determined by the [Packet Decoder](#). For example to match an IPv4 source address does therefore not require that the rule contains a field that checks that the protocol type is a IPv4 packet.

Fields for Configurable Egress ACL Table 0

The following fields can be selected for Configurable Egress ACL Table 0, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	MAC DA	48	Always valid	The packets destination MAC address.
1	MAC SA	48	Always valid	The packets source MAC address
2	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
3	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
4	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
5	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
6	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
7	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
8	GID	12	Always valid	GID Pointer from VLAN Table.
9	VID	12	Always valid	The internal VID assigned to the packet.
10	L2 Multicast Pointer	6	When a L2 or L3 router points to a L2 Multicast entry.	If a packet uses a multicast pointer (To the L2 Multicast table) then this is the pointer value.
11	Destination Port-mask	32	Always valid	The destination portmask for the packet.
12	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
13	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
14	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6
15	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
16	Source Port	5	Always valid	The source port of the packet.



Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
17	Rule Pointer	3	Always valid	The rule pointer (index in the Ingress Configurable ACL N Rules Setup).

14.4.2 Example Of Selecting Fields For Configurable Egress ACL Table 0

Since this ACL engine can select up to 7 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 135 bits in search key looks like when different fields are selected.

Example ACL with Ethernet Type

In this example we only want to create a rule with one field which is the Ethernet Type. This means that the fieldSelectBitmask, which is 18 bits , will be set as follows 1000000000000000 in binary format (Hex value of 0x2000) and the lookup data will be located as follows:

0	Ethernet Type	Valid
-	Width : 16	1
17	16	0 0

Table 14.35: Hash Key Example for Ethernet Type

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destination MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 18 bits , will be set as follows 1000000000000011 in binary format (Hex value of 0x2003) and the lookup data will be located as follows:

0	Ethernet Type	MAC DA	MAC SA	Valid
-	Width : 16	Width : 48	Width : 48	3
115	114 99	98 51	50 3	2 0

Table 14.36: Hash Key Example for Simple L2 ACL

Example of L3 IPv4 ACL

In this example we want to create a rule which with four L3 fields which are Destination IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 18 bits , will be set as follows 1001000000001100 in binary format (Hex value of 0x900c) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	IPv4 SA	L4 Protocol	Valid
-	Width : 2	Width : 32	Width : 32	Width : 8	4
78	77 76	75 44	43 12	11 4	3 0

Table 14.37: Hash Key Example for L3 IPv4 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destination Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 18 bits , will be set as follows 11001000011000000 in binary format (Hex value of 0x190c0) and the lookup data will be located as follows:



0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 5	Width : 2	Width : 8	Width : 16	Width : 16	5
52	51 47	46 45	44 37	36 21	20 5	4 0

Table 14.38: Hash Key Example for L4 ACL

Example of Egress NAT Entry

In this example we want to create a rule where the result would be used to change destination IP address and/or destination L4 Address. This means that the fieldSelectBitmask, which is 18 bits , will be set as follows 1100000010001000 in binary format (Hex value of 0xc088) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	L4 Type	L4 Destination Port	Valid
-	Width : 2	Width : 32	Width : 3	Width : 16	4
57	56 55	54 23	22 20	19 4	3 0

Table 14.39: Hash Key Example for Egress NAT Entry

Example of IPsec Encryption Entry

In this example we want to create a rule where the result would be used to send the packet to the crypto engine to be encrypted before it should be sent out. This means that the fieldSelectBitmask, which is 18 bits , will be set as follows 1000000000001000 in binary format (Hex value of 0x8008) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	Valid
-	Width : 2	Width : 32	2
36	35 34	33 2	1 0

Table 14.40: Hash Key Example for IPsec Encryption Entry

Example of MACsec Encryption Entry

In this example we want to create a rule where the result would be used to send the packet to the crypto engine to be encrypted before it should be sent out. This means that the fieldSelectBitmask, which is 18 bits , will be set as follows 100000000000 in binary format (Hex value of 0x800) and the lookup data will be located as follows:

0	Destination Portmask	Valid
-	Width : 32	1
33	32 1	0 0

Table 14.41: Hash Key Example for MACsec Encryption Entry

Fields for Configurable Egress ACL Table 1

The following fields can be selected for Configurable Egress ACL Table 1, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.



Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	L2 Packet Processing Flags	2	Always valid	Type of L2 packet processing which is to be done with this packet. Bit[0] If the packet was going to result to a port Move. Note that if the L2 Action Table forced the port move to not happen this field data follow this action. Bit[1] Packet is to be learned.
1	MAC DA	48	Always valid	The packets destination MAC address.
2	MAC SA	48	Always valid	The packets source MAC address
3	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
4	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
5	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
6	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
7	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
8	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
9	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
10	GID	12	Always valid	GID Pointer from VLAN Table.
11	VID	12	Always valid	The internal VID assigned to the packet.
12	L2 Multicast Pointer	6	When a L2 or L3 router points to a L2 Multicast entry.	If a packet uses a multicast pointer (To the L2 Multicast table) then this is the pointer value.
13	Destination Port-mask	32	Always valid	The destination portmask for the packet.
14	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
15	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
16	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6
17	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS.
18	Source Port	5	Always valid	The source port of the packet.



Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
19	Rule Pointer	2	Always valid	The rule pointer (index in the Ingress Configurable ACL N Rules Setup).

14.4.3 Example Of Selecting Fields For Configurable Egress ACL Table 1

Since this ACL engine can select up to 20 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 540 bits in search key looks like when different fields are selected.

Example ACL with TOS Byte

In this example we only want to create a rule with one field which is the TOS. This means that the fieldSelectBitmask, which is 20 bits , will be set as follows 10000000 in binary format (Hex value of 0x80) and the lookup data will be located as follows:

0	TOS	Valid
-	Width : 8	1
9	8 1	0 0

Table 14.43: Hash Key Example for TOS Byte

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destination MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 20 bits , will be set as follows 1000000000000110 in binary format (Hex value of 0x8006) and the lookup data will be located as follows:

0	Ethernet Type	MAC DA	MAC SA	Valid
-	Width : 16	Width : 48	Width : 48	3
115	114 99	98 51	50 3	2 0

Table 14.44: Hash Key Example for Simple L2 ACL

Example of L3 IPv6 ACL

In this example we want to create a rule which with four L3 fields which are Destination IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 20 bits , will be set as follows 100100000001100000 in binary format (Hex value of 0x24060) and the lookup data will be located as follows:

0	L3 Type	IPv6 DA	IPv6 SA	L4 Protocol	Valid
-	Width : 2	Width : 128	Width : 128	Width : 8	4
270	269 268	267 140	139 12	11 4	3 0

Table 14.45: Hash Key Example for L3 IPv6 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destination Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 20 bits , will be set as follows 1100100001100000000 in binary format (Hex value of 0x64300) and the lookup data will be located as follows:



0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 5	Width : 2	Width : 8	Width : 16	Width : 16	5
52	51 47	46 45	44 37	36 21	20 5	4 0

Table 14.46: Hash Key Example for L4 ACL

Example of Egress NAT Entry

In this example we want to create a rule where the result would be used to change destination IP address and/or destination L4 Address. This means that the fieldSelectBitmask, which is 20 bits , will be set as follows 110000001000010000 in binary format (Hex value of 0x30210) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	L4 Type	L4 Destination Port	Valid
-	Width : 2	Width : 32	Width : 3	Width : 16	4
57	56 55	54 23	22 20	19 4	3 0

Table 14.47: Hash Key Example for Egress NAT Entry

Example of IPsec Encryption Entry

In this example we want to create a rule where the result would be used to send the packet to the crypto engine to be encrypted before it should be sent out. This means that the fieldSelectBitmask, which is 20 bits , will be set as follows 100000000000010000 in binary format (Hex value of 0x20010) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	Valid
-	Width : 2	Width : 32	2
36	35 34	33 2	1 0

Table 14.48: Hash Key Example for IPsec Encryption Entry

Example of MACsec Encryption Entry

In this example we want to create a rule where the result would be used to send the packet to the crypto engine to be encrypted before it should be sent out. This means that the fieldSelectBitmask, which is 20 bits , will be set as follows 10000000000000 in binary format (Hex value of 0x2000) and the lookup data will be located as follows:

0	Destination Portmask	Valid
-	Width : 32	1
33	32 1	0 0

Table 14.49: Hash Key Example for MACsec Encryption Entry

14.4.4 ACL Search

The hash key is used to perform a lookup using the D-left hashing function described in detail in chapter [D-left Lookup](#).

Before the hash key is used the mask in [Egress Configurable ACL N Search Mask](#) is applied.

D-left calculates two hash values from the hash key. These hash values are then used to index the [Egress Configurable ACL N Small Table](#) and [Egress Configurable ACL N Large Table](#) . The hash calculations are described in section [Hash function for Configurable ACL](#).

In addition to the D-left search the hash key is also used to search in the [Egress Configurable ACL N TCAM](#)



14.4.5 ACL Actions

Once a hit has been determined by any of the searches above, the answer is read out from the corresponding answer entry. If it was a D-left hash hit then the answer actions is part of the hash memories ([Egress Configurable ACL N Small Table](#) , [Egress Configurable ACL N Large Table](#)). If it was a hit in the TCAM then the [Egress Configurable ACL N TCAM Answer](#) is used.

The behavior for multiple hits is configured in [Egress Configurable ACL N Selection](#) .

The statistics counter which can be updated are located in the [Egress Configurable ACL Match Counter](#)

14.5 Multiple ACL Lookups

The section above describes a single ACL Lookup. There are however 2 parallel ACL lookups.

Each of the ACL engines has its own rule configuration as well as its own hash and TCAM tables. By using the same rules for multiple engines the table space for a rule can be extended.

14.5.1 Multiple Actions

If the parallel ACL engines have multiple matches the result actions from each search engine can take effect. How multiple actions are handled depends on the type of action.

Any Match

If one or more ACL engines matches and has this action set then the action will take effect.

Action Field	Egress Acl 0 Has Ac- tion	Egress Acl 1 Has Ac- tion
dropEnable	Yes	Yes
sendToCpu	Yes	Yes

Table 14.50: Actions that will take effect if one or more is set.

First Match or Priority

If multiple ACL engines matches and has this action set then the value from the lowest numbered engine will be used.

If an entry has the priority field set this value will be used and the values which do not have priority set will be ignored.

If multiple matches have the priority field set then value from the highest numbered engine will be used.

Counter Update

All matches that have counter update action, [updateCounter](#), set will take effect. Each counter pointed to will be updated. If multiple actions point to the same counter then the counter value will only be incremented by one.

Send To Port

All matches that have an action [sendToPort](#) will take effect by setting the port number in the packet destination port mask. Any previous destination ports set will be cleared.

Send To CPU

If any match has the [sendToCpu](#) action set it will take effect. When the To CPU Tag is used the reason code will indicate table index in the lowest numbered engine. Any previous destination ports set will be cleared.



Enable Field	Priority Field	Value Field	Egress Acl 0 Has Action	Egress Acl 1 Has Action
natOpValid tunnelEntry	natOpPrio tunnelEntryPrio	natOpPtr tunnelEntryPtr tunnelEntryUcMc	Yes Yes	No Yes
sendToPort metaDataValid updateCounter	N/A metaDataPrio N/A	destPort metaData counter	Yes Yes Yes	Yes Yes Yes

Table 14.51: The lowest numbered takes effect if no priority else the highest numbered with priority set.



Chapter 15

VLAN and Packet Type Filtering

This chapter gives an overview of the filtering options available on ingress and egress. Filtering allows different types of packets to be accepted or dropped.

A filter is applied at the source port as packets enter the switch core. This is set up in the [Ingress Port Packet Type Filter](#) register.

When the packet is ready to be queued, the [Ingress Egress Port Packet Type Filter](#) is applied for each egress port the packet is to be queued onto. If the packet is dropped then a drop counter is updated for each packet which is dropped.

Before a packet is to be sent out, the egress port it is checked in the [Egress Port Configuration](#) to see if the packet is allowed to be sent out.

The settings are unique for each port.

A packet of a certain type may be allowed to enter on a certain ingress port. But this does not mean the frame is ultimately allowed to be transmit, since ingress and egress port filters are setup independently.

In addition to the egress port packet type filter, there is also a source port filter on the egress port. This is found in [srcPortFilter](#). The source port filter on the egress port allows a user to decide whether packets from a certain source port are allowed to be sent out on an egress port. The outcome of the filtering options are either to drop a packet, or to allow it.

Since the source port table, vlan table and egress port configuration can all have VLAN operations which changes the packet, it is important to understand on which packet the filtering is actually done.

- The source port filtering is done on the packet as it enters the switch without any packet modifications.
- The ingress egress port filtering is done on the packet after the source port and VLAN table VLAN operations. The L2 Multicast is calculated in the same way as MBSC register [L2 Multicast Handling](#).
- The egress port filtering is done after all the VLAN operations has been carried out including the egress ports own VLAN operations.

Note that if a user defined VLAN tag is pushed, it will always be regarded as a C-VLAN tag by the filtering.



Chapter 16

Hashing

Hashing is used to enable the use of SRAM memories instead of using CAMs for lookups.

16.1 Hashing Functions

This section describes the hash functions used in this core.

16.1.1 MAC Table Hashing

The hash function receives the destination MAC address and GID as an input and it returns a hash with the same bit width as the address for the [L2 DA Hash Lookup Table](#) divided by number of buckets (4). The table is divided into equal sized parts/buckets which are readout in parallel.

Hash Function for MAC Table

The XOR hash function splits the key into 6 parts, each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When learning random MAC addresses the hash function results in an average utilization of the L2 table of 40% (including/excluding multicast addresses does not change this). When learning sequential MAC addresses (such as in the RFC2889) the utilization is 100%.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_l2_hash( key ):
    """ key: 60 bits hash key
        key[59:48] = GID
        key[47:0] = MAC
        fold count = 6
        returns: 10 bits hash value
    """
    hashval = key & 0b111111111
    hashval = hashval ^ (key>>10)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>20)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>30)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>40)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>50)
    hashval = hashval & 0b111111111
    return hashval

def mac_str2int( mac_adr ):
    """ Convert Ethernet MAC address from string format , e.g. '46:61:62:bc:84:dd'

```

```

    to integer. """
    hx = ''.join(mac_adr.split(':'))
    return int(hx,16)

def l2_hash( gid , mac ):
    """ Calculate index into L2 hash table from GID and MAC address.
        Both parameters must be integers """
    key = (gid & 0xfff) << 48
    key |= mac & 0xffffffffffff
    return calc_l2_hash( key )

def l2_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # MAC: 46:61:62:bc:84:dd (leftmost byte is first byte received)
    # GID:996
    key = (996)<< 48 | 0x466162bc84dd
    hashval = calc_l2_hash(key) # the hash value is used as index into the L2 DA Hash Table
    assert hashval == 237

```

16.1.2 IP Table Hashing

The hash function receives the destination IP address and VRF as key and returns a hash with the same number of bits as the address for the [Hash Based L3 Routing Table](#) .

Hash Function for IPv4

The XOR hash function splits the key into parts, each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When learning random IPv4 addresses the hash function results in an average utilization of the hash table of 20% .

Python code for the IPv4 hashing function is shown below as well as a test case to clarify how the key is calculated.

```

def calc_l3_ipv4_hash( key ):
    """ key: 34 bits hash key
        key[33:32] = VRF
        key[33:0] = IP address
        fold count = 4
        returns: 9 bits hash value
    """
    hashval = key & 0b11111111
    hashval = hashval ^ (key>>9)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>27)
    hashval = hashval & 0b11111111
    return hashval

def ipv4_str2int( ip_addr ):
    """ Convert IPv4 address from string format, e.g. 192.168.0.123,
        to integer """
    parts = ip_addr.split('.')
    res = 0
    for p in parts:
        res <<= 8
        res |= int(p)
    return res

```




```

def l3_ipv4_hash( vrf, ip_addr ):
    """ Calculate index into L3 hash table from VRF and IP address.
        Both parameters must be integers. """
    key = (vrf & 0x3) << 32
    key |= ip_addr
    return calc_l3_ipv4_hash( key )

def ipv4_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # IP: 70.119.98.188 (leftmost byte is first byte received)
    # VRF:3
    vrf = 3
    ip = 0x467762bc
    key = ( vrf << 32 ) | ip
    # the hash value is used as index into the Hash Based L3 Routing Table
    hashval = calc_l3_ipv4_hash(key)
    assert hashval == 248

```

Hash Function for IPv6

The XOR hash function splits the key into parts, each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When learning random IPv6 addresses the hash function results in an average utilization of the hash table of 20% .

Python code for the IPv6 hashing function is shown below as well as a test case to clarify how the key is calculated.

```

def calc_l3_ipv6_hash( key ):
    """ key: 130 bits hash key
        key[129:128] = VRF
        key[129:0] = IP address
        fold count = 15
        returns: 9 bits hash value
    """
    hashval = key & 0b11111111
    hashval = hashval ^ (key>>9)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>27)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>36)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>45)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>54)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>63)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>72)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>81)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>90)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>99)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>108)
    hashval = hashval & 0b11111111

```



```

hashval = hashval ^ (key>>117)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>126)
hashval = hashval & 0b11111111
return hashval

def l3_ipv6_hash( vrf , ip_addr ):
    """ Calculate index into L3 hash table from VRF and IP address.
        Both parameters must be integers. """
    key = (vrf & 0x3) << 128
    key |= ip_addr
    return calc_l3_ipv6_hash( key )

def ipv6_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # IP: d8a7:da8b:: (leftmost byte is first byte received)
    # VRF:3
    vrf = 3
    ip = 0xd8a7da8b000000000000000000000000
    key = ( vrf << 128 ) | ip
    hashval = calc_l3_ipv6_hash(key)
    # the hash value is used as index into the Hash Based L3 Routing Table
    assert hashval == 294

```

16.1.3 MPLS Table Hashing

The hash function receives the outermost MPLS label, source port number and VRF as key and returns a hash with the same number of bits as the address for the [Hash Based L3 Routing Table](#)

Hash Function for MPLS

The XOR hash function splits the key into parts , each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When storing random MPLS labels the hash function results in an average utilization of the hash table of 19% .

Python code for the MPLS hashing function is shown below as well as a test case to clarify how the key is calculated.

```

def calc_l3_mpls_hash( key ):
    """ key: 27 bits hash key
        key[26:25] = VRF
        key[24:5] = MPLS label
        key[4:0] = source port
        fold count = 3
        returns: 9 bits hash value
    """
    hashval = key & 0b11111111
    hashval = hashval ^ (key>>9)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b11111111
    return hashval

def l3_mpls_hash( vrf , source_port , label ):
    key = (vrf & 0xfff) << 25
    key |= label & 0xffff << 5
    key |= ( source_port & 0x1f )
    return calc_l3_mpls_hash( key )

```



```
def mpls_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # MPLS label: 839243 (leftmost byte is first byte received)
    # VRF:1
    # source port:23
    mpls_label = 839243
    vrf = 1
    srcport = 23
    key = (vrf << (5 + 20) |
          srcport << 20 |
          mpls_label)
    hashval = calc_l3_mpls_hash(key)
    # the hash value is used as index into the Hash Based L3 Routing Table
    assert hashval == 243
```

16.1.4 Hash function for Ingress Configurable ACL 0

The hash function receives the lookup key created by selecting the fields from the packet determined by the [Ingress Configurable ACL 0 Rules Setup](#). The lookup key is up to 330 bits wide. The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_confAcl_small0_hash( key ):
    """ key: 330 bits hash key
        fold count = 55
        returns: 6 bits hash value
    """
    hashval = key & 0b111111
    hashval = hashval ^ (key>>6)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>12)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>24)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>30)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>36)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>42)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>48)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>54)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>60)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>66)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>72)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>78)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>84)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>90)
```



```
hashval = hashval & 0b111111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>102)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>108)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>114)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>126)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>132)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>138)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>144)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>150)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>156)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>162)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>168)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>174)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>180)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>186)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>192)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>198)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>204)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>210)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>216)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>222)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>228)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>234)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>240)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>246)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>252)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>258)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>264)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>270)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>276)
```



```

hashval = hashval & 0b111111
hashval = hashval ^ (key>>282)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>288)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>294)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>300)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>306)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>312)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>318)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>324)
hashval = hashval & 0b111111
return hashval

def confAcl_small0_hash( destination_address ):
    """ Calculate index into confAcl_small0 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_small0_hash( key )

def calc_confAcl_large0_hash( key ):
    """ key: 330 bits hash key
        fold count = 37
        returns: 9 bits hash value
    """
    hashval = key & 0b111111111
    hashval = hashval ^ (key>>9)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>27)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>36)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>45)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>54)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>63)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>72)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>81)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>90)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>99)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>108)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>117)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>126)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>135)
    hashval = hashval & 0b111111111

```



```

hashval = hashval ^ (key>>144)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>153)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>162)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>171)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>180)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>189)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>198)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>207)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>216)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>225)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>234)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>243)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>252)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>261)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>270)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>279)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>288)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>297)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>306)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>315)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>324)
hashval = hashval & 0b11111111
return hashval

```

```

def confAcl_large0_hash( destination_address ):
    """ Calculate index into confAcl_large0 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_large0_hash( key )

```

```

def confAcl0_hash_test():
    key = 6663141436383049711421858719482107758268636509801792817586951760462394549009632021713
    hashval = confAcl_small0_hash(key)
    assert hashval == 46

    hashval = confAcl_large0_hash(key)
    assert hashval == 479

```



16.1.5 Hash function for Ingress Configurable ACL 1

The hash function receives the lookup key created by selecting the fields from the packet determined by the [Ingress Configurable ACL 1 Rules Setup](#). The lookup key is up to 135 bits wide. The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_confAcl_small1_hash( key ):
    """ key: 135 bits hash key
        fold count = 68
        returns: 2 bits hash value
    """
    hashval = key & 0b11
    hashval = hashval ^ (key>>2)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>4)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>6)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>8)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>10)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>12)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>14)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>16)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>20)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>22)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>24)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>26)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>28)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>30)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>32)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>34)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>36)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>38)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>40)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>42)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>44)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>46)
    hashval = hashval & 0b11
    hashval = hashval ^ (key>>48)
    hashval = hashval & 0b11
```



```
hashval = hashval ^ (key>>50)
hashval = hashval & 0b11
hashval = hashval ^ (key>>52)
hashval = hashval & 0b11
hashval = hashval ^ (key>>54)
hashval = hashval & 0b11
hashval = hashval ^ (key>>56)
hashval = hashval & 0b11
hashval = hashval ^ (key>>58)
hashval = hashval & 0b11
hashval = hashval ^ (key>>60)
hashval = hashval & 0b11
hashval = hashval ^ (key>>62)
hashval = hashval & 0b11
hashval = hashval ^ (key>>64)
hashval = hashval & 0b11
hashval = hashval ^ (key>>66)
hashval = hashval & 0b11
hashval = hashval ^ (key>>68)
hashval = hashval & 0b11
hashval = hashval ^ (key>>70)
hashval = hashval & 0b11
hashval = hashval ^ (key>>72)
hashval = hashval & 0b11
hashval = hashval ^ (key>>74)
hashval = hashval & 0b11
hashval = hashval ^ (key>>76)
hashval = hashval & 0b11
hashval = hashval ^ (key>>78)
hashval = hashval & 0b11
hashval = hashval ^ (key>>80)
hashval = hashval & 0b11
hashval = hashval ^ (key>>82)
hashval = hashval & 0b11
hashval = hashval ^ (key>>84)
hashval = hashval & 0b11
hashval = hashval ^ (key>>86)
hashval = hashval & 0b11
hashval = hashval ^ (key>>88)
hashval = hashval & 0b11
hashval = hashval ^ (key>>90)
hashval = hashval & 0b11
hashval = hashval ^ (key>>92)
hashval = hashval & 0b11
hashval = hashval ^ (key>>94)
hashval = hashval & 0b11
hashval = hashval ^ (key>>96)
hashval = hashval & 0b11
hashval = hashval ^ (key>>98)
hashval = hashval & 0b11
hashval = hashval ^ (key>>100)
hashval = hashval & 0b11
hashval = hashval ^ (key>>102)
hashval = hashval & 0b11
hashval = hashval ^ (key>>104)
hashval = hashval & 0b11
hashval = hashval ^ (key>>106)
hashval = hashval & 0b11
hashval = hashval ^ (key>>108)
hashval = hashval & 0b11
hashval = hashval ^ (key>>110)
hashval = hashval & 0b11
```




```

hashval = hashval ^ (key>>112)
hashval = hashval & 0b11
hashval = hashval ^ (key>>114)
hashval = hashval & 0b11
hashval = hashval ^ (key>>116)
hashval = hashval & 0b11
hashval = hashval ^ (key>>118)
hashval = hashval & 0b11
hashval = hashval ^ (key>>120)
hashval = hashval & 0b11
hashval = hashval ^ (key>>122)
hashval = hashval & 0b11
hashval = hashval ^ (key>>124)
hashval = hashval & 0b11
hashval = hashval ^ (key>>126)
hashval = hashval & 0b11
hashval = hashval ^ (key>>128)
hashval = hashval & 0b11
hashval = hashval ^ (key>>130)
hashval = hashval & 0b11
hashval = hashval ^ (key>>132)
hashval = hashval & 0b11
hashval = hashval ^ (key>>134)
hashval = hashval & 0b11
return hashval

def confAcl_small1_hash( destination_address ):
    """ Calculate index into confAcl_small1 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x7fffffffffffffffffffffffffffffff
    return calc_confAcl_small1_hash( key )

def calc_confAcl_large1_hash( key ):
    """ key: 135 bits hash key
        fold count = 23
        returns: 6 bits hash value
    """
    hashval = key & 0b111111
    hashval = hashval ^ (key>>6)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>12)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>24)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>30)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>36)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>42)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>48)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>54)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>60)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>66)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>72)

```



```

hashval = hashval & 0b111111
hashval = hashval ^ (key>>78)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>84)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>90)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>102)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>108)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>114)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>126)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>132)
hashval = hashval & 0b111111
return hashval

```

```

def confAcl_large1_hash( destination_address ):
    """ Calculate index into confAcl_large1 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x7fffffffffffffffffffffffffffffffffffff
    return calc_confAcl_large1_hash( key )

```

```

def confAcl1_hash_test():
    key = 37026369368194631942986752982012943599094
    hashval = confAcl_small11_hash(key)
    assert hashval == 1

    hashval = confAcl_large1_hash(key)
    assert hashval == 11

```

16.1.6 Hash function for Ingress Configurable ACL 2

This ACL engine only has TCAM.

16.1.7 Hash function for Ingress Configurable ACL 3

This ACL engine only has TCAM.

16.1.8 Hash function for Egress Configurable ACL 0

The hash function receives the lookup key created by selecting the fields from the packet determined by the [Egress Configurable ACL 0 Rules Setup](#). The lookup key is up to 135 bits wide. The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```

def calc_confAcl_small0_hash( key ):
    """ key: 135 bits hash key
        fold count = 23
        returns: 6 bits hash value
    """
    hashval = key & 0b111111

```



```

hashval = hashval ^ (key>>6)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>12)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>18)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>24)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>30)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>36)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>42)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>48)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>54)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>60)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>66)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>72)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>78)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>84)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>90)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>102)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>108)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>114)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>126)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>132)
hashval = hashval & 0b111111
return hashval

```

```

def confAcl_small0_hash( destination_address ):
    """ Calculate index into confAcl_small0 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x7fffffffffffffffffffffffffffffffffff
    return calc_confAcl_small0_hash( key )

```

```

def calc_confAcl_large0_hash( key ):
    """ key: 135 bits hash key
        fold count = 17
        returns: 8 bits hash value
    """
    hashval = key & 0b11111111
    hashval = hashval ^ (key>>8)
    hashval = hashval & 0b11111111
    hashval = hashval ^ (key>>16)

```



```

hashval = hashval & 0b11111111
hashval = hashval ^ (key>>24)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>32)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>40)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>48)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>56)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>64)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>72)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>80)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>88)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>104)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>112)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>128)
hashval = hashval & 0b11111111
return hashval

```

```

def confAcl_large0_hash( destination_address ):
    """ Calculate index into confAcl_large0 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x7fffffffffffffffffffffffffffffff
    return calc_confAcl_large0_hash( key )

```

```

def confEgressAcl0_hash_test():
    key = 1368970965343244166845021771927454247000
    hashval = confEgressAcl_small0_hash(key)
    assert hashval == 7

    hashval = confEgressAcl_large0_hash(key)
    assert hashval == 70

```

16.1.9 Hash function for Egress Configurable ACL 1

This ACL engine only has TCAM.

16.1.10 Hash function for Tunneling

The tunneling exit lookups consists of two lookups. First the tunnel exit lookup and secondly the second tunnel exit lookup.

First Tunnel Exit Hash

Uses only TCAM in this design. Located in table [Tunnel Exit Lookup TCAM](#).



Second Tunnel Exit Hash

Uses only TCAM in this design. Located in table [Second Tunnel Exit Lookup TCAM](#).





Chapter 17

D-left Lookup

D-left is a hash table search algorithm that reduces the risk of hash collisions by using two hash tables each indexed by a separate hash key.

This implementation uses two hash tables, one smaller and one larger, combined with a synthesized TCAM to resolve hash collisions. This is shown in figure [17.1](#).

The hash search is done by taking a hash key and calculating two hashes from that. The two hash values are used as index into the small and large hash tables.

Each table has a number of buckets for each hash index. All buckets for the selected index are read out in parallel. The hash key is then compared with the compareData from each bucket. There is a hit if one of the buckets compareData matches the hash key. If multiple buckets matches then the highest numbered bucket is used.

This is done in parallel for both the small and the large table.

In addition the hash key is also searched in the TCAM. In the TCAM search all entries are compared with the hash and if there are multiple matches then the lowest numbered entry is used.

Since a single search can result in multiple hits in all three tables there is configuration that selects which table shall be used in this case.

The two hash tables have separate masks which allows some bits to be masked away. For the TCAM there is a mask per entry.

17.1 Functions using D-left

The following functions use D-left Lookup.

17.1.1 Egress VLAN Translation

In this design the Egress VLAN translation only uses TCAM located in register [Egress VLAN Translation TCAM Answer](#).

17.1.2 Ingress Configurable ACL

The ingress configurable ACL is setup by using the following registers and tables.

- The search data/hash key is the selected packet header fields (see [Selectable Packet Fields](#)).
- Hash tables
 - The hash functions used to index the hash tables are described in section [Hash function for Configurable ACL](#).
 - [Ingress Configurable ACL 0 Small Table](#)
 - [Ingress Configurable ACL 0 Large Table](#)
 - [Ingress Configurable ACL 1 Small Table](#)
 - [Ingress Configurable ACL 1 Large Table](#)

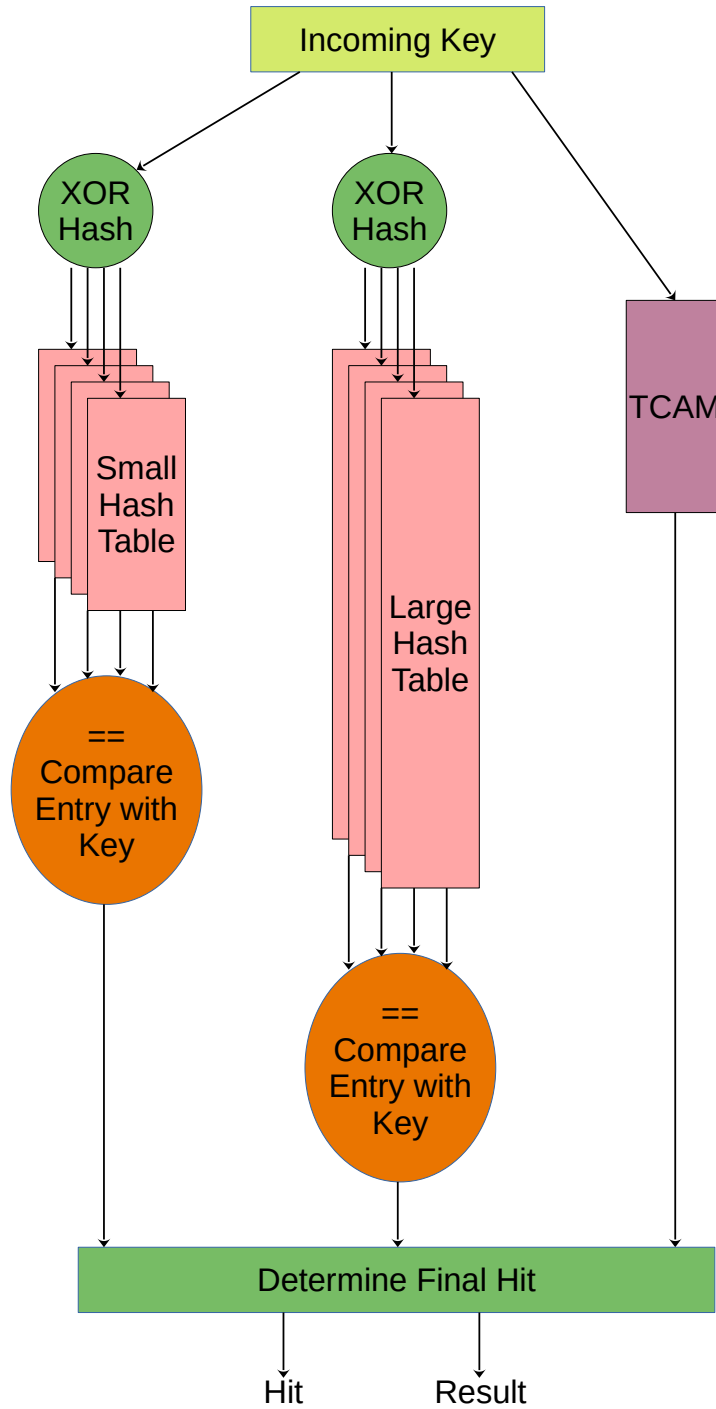


Figure 17.1: D-left Function

- TCAM
 - [Ingress Configurable ACL 0 TCAM](#)
 - [Ingress Configurable ACL 1 TCAM](#)
 - [Ingress Configurable ACL 2 TCAM](#)
 - [Ingress Configurable ACL 3 TCAM](#)
- Masks for the hash tables
 - [Ingress Configurable ACL 0 Search Mask](#)
 - [Ingress Configurable ACL 1 Search Mask](#)



- Configuration for resolving multiple hits
 - [Ingress Configurable ACL 0 Selection](#)
 - [Ingress Configurable ACL 1 Selection](#)
- The ACL actions are stored in the hash tables but the actions for TCAM hits are stored in a separate tables
 - [Ingress Configurable ACL 0 TCAM Answer](#)
 - [Ingress Configurable ACL 1 TCAM Answer](#)
 - [Ingress Configurable ACL 2 TCAM Answer](#)
 - [Ingress Configurable ACL 3 TCAM Answer](#)

17.1.3 Egress Configurable ACL

The ingress configurable ACL is setup by using the following registers and tables.

- The search data/hash key is the selected packet header fields (see [Selectable Packet Fields](#)).
- Hash tables
 - The hash functions used to index the hash tables are described in section [Hash function for Configurable ACL](#).
 - [Egress Configurable ACL 0 Small Table](#)
 - [Egress Configurable ACL 0 Large Table](#)
- TCAM
 - [Egress Configurable ACL 0 TCAM](#)
 - [Egress Configurable ACL 1 TCAM](#)
- Masks for the hash tables
 - [Egress Configurable ACL 0 Search Mask](#)
- Configuration for resolving multiple hits
 - [Egress Configurable ACL 0 Selection](#)
- The ACL actions are stored in the hash tables but the actions for TCAM hits are stored in a separate tables
 - [Egress Configurable ACL 0 TCAM Answer](#)
 - [Egress Configurable ACL 1 TCAM Answer](#)

17.1.4 Tunnel Exit

The first tunnel exit lookup uses only TCAM. Located in table [Tunnel Exit Lookup TCAM](#).

The second tunnel exit lookup uses only TCAM. Located in table [Second Tunnel Exit Lookup TCAM](#).





Chapter 18

Learning and Aging

The switch supports automatic hardware learning and aging as well as software controlled learning and aging.

- With hardware learning the switch can be functional after reset without any software setup. The hardware learning engine saves the source port number, the source MAC address with a Global Identifier (GID) from the **VLAN Table** in the forwarding information base.
- If the destination MAC address and the GID of a packet is in the L2 forwarding information base, the L2 forwarding process will know the destination port of this packet.
- If a learned {GID, MAC} has not been hit by a source or destination MAC address for a while, the hardware aging engine will remove this entry from the table.
- When a learned MAC address is received as MAC SA on a different port than it was setup in the **L2 Destination Table**, it is considered a port move.
- When the hardware aging is enabled, all non-static entries will be aged out after a certain silent period. **Hardware Learning Configuration** configures the initial status of the newly learned entries.
- The software learning and aging feature allows users to fully control the L2 forwarding information base.
- The hardware learning and aging functions are by default turned on and can be turned off through the **Learning And Aging Enable** register.
- When the hardware learning is enabled, all source ports are allowed to get their unknown source MAC address learned. By setting **learningEn** field in the **Source Port Table** to 0 the learning process can be disabled on the corresponding source port.
- For an unknown MAC DA, **dropUnknownDa** field in the **Source Port Table** determines either to drop the packet or allow it to be flooded.

18.1 L2 Forwarding Information Base (FIB)

Multiple tables in groups are involved in the learning and aging functions when making L2 forwarding decisions:

18.1.1 Tables for MAC DA lookup

1. L2 Hash tables.
 - (a) **L2 DA Hash Lookup Table**
 - (b) **L2 Aging Status Shadow Table**
2. L2 Collision tables.
 - (a) **L2 Lookup Collision Table**
 - (b) **L2 Aging Collision Shadow Table**
3. **L2 Destination Table**.
4. **L2 Multicast Table**.

MAC DA lookups are used to find L2 forwarding destinations and the related tables are written as results from learning or aging functions. The forwarding function relies on a hash algorithm described in Section [MAC Table Hashing](#) and a search algorithm described in Section [L2 Destination Lookup](#). In this core, destination MAC addresses and GIDs are combined together to create a 60-bit hash key and the hash function returns a 10-bit hash value.

18.1.2 Tables for MAC SA lookup

1. [L2 SA Hash Lookup Table](#). Holding the same contents as [L2 DA Hash Lookup Table](#).
2. [L2 Aging Status Shadow Table - Replica](#). Holding the same contents as [L2 Aging Status Shadow Table](#).
3. [L2 Destination Table - Replica](#). Holding the same contents as [L2 Destination Table](#).

The MAC SA lookups are used to create new learning requests and requiring the same tables as MAC DA lookups. Due to the fact that the core mostly uses tables with single read port towards the ingress processing pipeline, there are three MAC DA tables duplicated to MAC SA tables listed above to support one read per cycle from the ingress processing pipeline (one MAC DA lookup and one MAC SA lookup at every clock cycle). No matter when the MAC DA/MAC SA lookup tables are updated, the corresponding SA/DA lookup tables need to be filled with the same updates. The L2 collision tables are built to support parallel read by both DA and MAC SA lookups and therefore are not duplicated.

The MAC SA lookups form a key-hash pair by {GID,MAC SA} and do a two step check:

1. Hit or not. Hit is given in two cases:
 - (a) The key-hash pair is found in the [L2 SA Hash Lookup Table](#) and the related entry in [L2 Aging Status Shadow Table - Replica](#) is valid.
 - (b) The key is found in the [L2 Lookup Collision Table](#) and the related entry in [L2 Aging Collision Table](#) is valid.
2. The source port number matches the port number in the L2 destination table.

Based on the lookup result there are three possible learning decisions:

1. Learn a new entry: Not hit.
2. Port move request: Hit with port number mismatching.
3. SA hit update operation: Hit with port number matching.

Figure 6.1 demonstrates how the FIB addressing looks like.

18.1.3 Status Tables

1. [L2 Aging Table](#)
2. [L2 Aging Collision Table](#)

The status tables are located inside the learning and aging engine to monitor and maintain the status of all entries in the FIB. An FIB entry has three status bits:

1. **valid**: Indicate if a hit in the FIB is valid.
2. **stat**: Indicate if an entry is static. Static entries cannot be modified by hardware.
3. **hit**: Indicate either MAC SA or DA has successfully hit this entry since the last aging scan.

When the hardware learning or aging updates the status table, the **valid** bit will be copied to the shadow tables in the ingress processing pipeline.

As in Figure 18.1 the FIB can be accessed from three units:

1. From software through the configuration interface: read and write.
2. Learning and aging unit: read and write.
3. Ingress processing pipeline: read only.

Notice that shadow tables in the FIB have to be updated simultaneously with status tables. MAC SA lookup tables have to be updated simultaneously with MAC DA lookup tables. Unexpected behavior will occur if the tables do not have the same content.



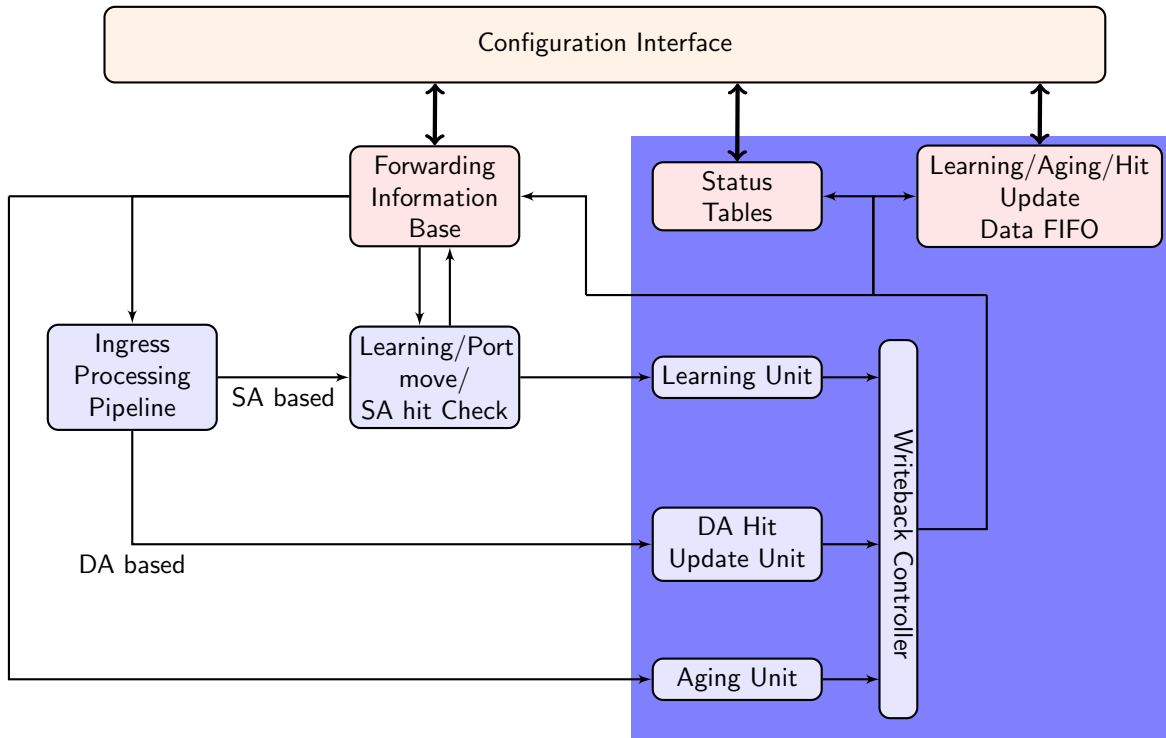


Figure 18.1: Learning and Aging Engine

18.1.4 Hash Collision Accommodation

In order to solve hash collisions, the **L2 DA Hash Lookup Table** has 4 buckets each with 1,024 entries. A given key-hash pair can search in the 4 buckets in parallel by reading from the address that equals the hash value. The 4 buckets entries are all compared with the {GID,MAC DA} key and if one entry is equal to the key that entry is considered a match.

Besides the **L2 DA Hash Lookup Table**, there is an extra **L2 Lookup Collision Table** in case the number of hash collisions is more than the **L2 DA Hash Lookup Table** can handle. For instance, if the hash function calculated the same hash value for more than 4 keys, the first 4 keys can be accommodated in the 4 buckets of **L2 DA Hash Lookup Table** while the rest are stored in the **L2 Lookup Collision Table**. Searching in the **L2 Lookup Collision Table** will return the first entry index that holds the corresponding key.

Addressing into the **L2 Destination Table** is based on the hit index from either the **L2 DA Hash Lookup Table** or the **L2 Lookup Collision Table**.

- Hit in the **L2 DA Hash Lookup Table**: get a 12-bit hit index with the hash value in the lower 10 bits and the bucket number in the higher 2 bits. The corresponding **L2 Destination Table** address equals the hit index.
- Hit in the **L2 Lookup Collision Table**: get a 5-bit hit index from the hit entry address. The corresponding **L2 Destination Table** address is (hit index + 4,096).

18.2 Hardware Learning and Aging

18.2.1 Learning Unit

The core has a dedicated learning unit in hardware, which is tasked with learning L2 MAC addresses combined with GIDs as entries to do L2 destination port lookups. A new learning request is created and processed in several steps:

1. For every packet a learning check is performed based on its MAC SA and GID and issues learning requests to the learning unit.



2. If it is a known entry but the **hit** bit in the status table is 0, the **hit** bit will be refreshed to 1.
3. If the learning request is to learn a new entry, **Hardware Learning Counter** will be checked against the **learnLimit** in **Hardware Learning Configuration**. **learnLimit** limits the maximum number of entries can be learned on a port.
4. If the maximum learning limit is not reached on a port, the status table lookup will try to provide an available entry in a certain order:
 - (a) Find a free entry.
 - i. Select a free bucket for this hash value.
 - ii. If all hash buckets are used, select a free collision table entry.
 - (b) If there is no free entry and **lru** in the **Learning And Aging Enable** register is 0, the learning unit will search in the collision table and overwrite the non-static entries in a round robin order.
 - (c) If there is no free entry and **lru** in the **Learning And Aging Enable** register is 1, the learning unit will overwrite a least recently used non-static entry as follows:
 - i. Search in hash buckets for a bucket with **hit**=0 and **stat**=0. Return the last match.
 - ii. If all buckets have **hit**=1 or **stat**=1, search in the collision table for an entry with **hit**=0 and **stat**=0. Return the first match.
 - (d) If all entries are static or have been hit since the last aging scan, overwrite a non-static entry.
 - i. Search in hash buckets for a bucket with **stat**=0. Return the last match.
 - ii. If all buckets are static, search in the collision table for an entry with **stat**=0 in a round robin order.
5. If the learning unit failed to accomodate the unknown MAC SA and GID combination, or the learning limit on a port is reached, the learning request will be ignored and the corresponding MAC SA, GID and port number will be updated to the **Learning Overflow** register.
6. If a valid entry is found, the learning unit will link it to the port number from the learning request as a L2 unicast entry.
7. If the learning request is for a port move, the process will operate on existing non-static entries directly. For static entries, the **Port Move Options** register gives optional operations for each previously learned port.
8. If the learning unit failed to execute port move due to immutable static entry or the learning limit is reached, the learning request will be ignored and the corresponding MAC SA, GID and port number will be updated to the **Learning Conflict** register.
9. A valid learning decision is sent to a writeback bus which manages all decisions from different learning and aging units. The learning decisions have the highest priority to use the writeback bus.
10. The writeback bus pushes the learning decision to the **Learning Data FIFO**. By default the writeback bus is allowed to send decisions to the **FIB**, but there is also an option to block the table updates from the configuration interface.
11. By setting the **hwLearningWriteBack** field in the **Learning And Aging Writeback Control** to 0, table updates from the hardware learning unit is blocked. In this case the software shall maintain the hardware learning decision from the **Learning Data FIFO**, and updates the **FIB** as described in Section **Software Learning and Aging**.

18.2.2 Hardware Learning Exceptions

The switch support fine granular control to allow certain packets with unknown MAC SA address to not be learned. These settings described below enables a variety of different ways to turn it off on a per packet basis.

- Source port exceptions.
 - If **uniqueCpuMac** is set to 1, the CPU port cannot be learned.
 - If the packet from the CPU port has a from CPU tag, it will bypass L2 lookup hence bypass the learning process.
 - For any source port if its **learningEn** is set to 0 the learning process is disabled.
- To CPU packet. If the packet is sent to the CPU port with a non-zero reason code. ¹
- Classification.
 - If the packet hit in a classification rule that override L2 lookup (i.e. force the destination port), it will not be learned.

¹Check all reason codes in Table 30.2



- If the packet hit in the **Configurable ACL Engine** with **noLearning** enabled.
- Routed. A routed packet will not be learned.
- Dropped. If the ingress processing drops the packet (post-ingress processing is not counted), the packet will not be learned unless it is due to the ingress spanning tree drop and the state says **Learning**.²
- Multicast MAC SA. In the switch core a MAC address with the least-significant bit of the first octet equals 1 (e.g. 01:80:c2:00:00:00) but not equals to ff:ff:ff:ff:ff:ff is marked as Ethernet multicast address. By default a MAC SA that matches an Ethernet multicast address will not be learned. This can be configured per port through the **learnMulticastSaMac** field in the **Source Port Table**.

18.2.3 Aging Unit

When a new L2 entry is learned by the hardware learning unit, the initial entry status is from the **Hardware Learning Configuration** register. A valid non-static entry will be aged out if no L2 MAC SA/DA lookup hit it within a certain time and static entries must have software interactions to get aged/changed. By default a non-static entry will be learned with both **hit** and **valid** set to 1 to prevent it from being aged out immediately. Static entries can be established on a per source port basis by setting the **stat** field in **Hardware Learning Configuration** to 1.

The hardware aging function does a periodic check of the L2 entry status in the **L2 Aging Table** and the **L2 Aging Collision Table**. The waiting period between two checks is tick based³ and configurable via the **Time to Age** register. During an aging check period, the aging unit loops through all entries in the **L2 Aging Table** and **L2 Aging Collision Table** to get the current status. The possible updates are listed in Table 18.1. If the **valid** bit (bit 0) is turned to 0 the entry is aged out. An aged out entry can be learned again.

If the **Time to Age** register is reconfigured during runtime, the updated **tickCnt** will not be available to aging unit until the current aging period is complete. In order to load new values immediately, the aging unit needs to be restarted via the **agingEnable** field in the **Learning And Aging Enable** register. However, changes to the **tick** selection are always applied immediately.

Current Status	Update Status
0b101	0b001
0b001	0b000(entry cleared)
Other values	No update

Table 18.1: Hardware Aging Operations

18.2.4 MAC DA Hit Update Unit

The learning unit has a built-in MAC SA hit update unit to refresh the **hit** bit while another MAC DA hit update unit can operate in parallel. The MAC DA hit update unit can be turned on or off by the **daHitEnable** field in the **Learning And Aging Enable** register and works as such:

1. A packet with L2 MAC DA lookup returns a valid and non-static entry issues a hit update request for the corresponding MAC DA.
2. A hit update FIFO is prepared to buffer the update requests.
3. A hit update request is popped from the FIFO when the writeback bus is free.
4. If the writeback bus keeps busy with learning decisions and causes a buildup in the hit update FIFO, new hit update requests will be ignored when the FIFO is full.
5. The writeback bus forwards the hit update request to both the **Hit Update Data FIFO** and the **FIB**, optionally the FIB updates could be turned off by the **hwHitWriteBack** field in the **Learning And Aging Writeback Control** register.

According to Table 18.1, the automatic **hit** bit update for a non-static L2 entry will keep the hardware aging unit away from setting the **valid** bit to 0, hence avoid aging out the entry.

18.3 Software Learning and Aging

Instead of automatic learning and aging, the switch provides two options for software to manipulate learning and aging behaviors.

²See more in Chapter **Spanning Tree**.

³The system ticks are described in Chapter **Tick**.



18.3.1 Injection of Learning Packets

The switch features a learning protocol to let all ports accept special learning packets to fully control the **FIB**. The learning packet format is shown in Figure 18.2. The MAC DA of a learning packet must match the address configured in the **Learning DA MAC** register. With a compliant MAC DA the packet is dropped inside the switch but the carried learning tag will be decoded and sent to the learning unit.

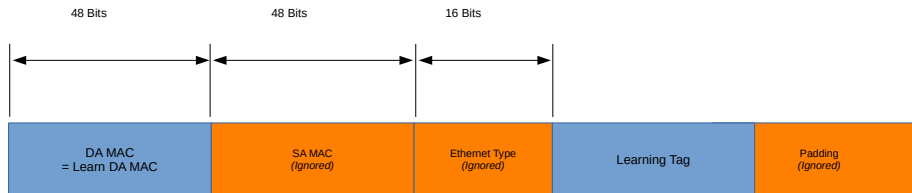


Figure 18.2: Learning Frame

The following table describes the fields which will be in the learning tag.

Name	Short Name	Field Size	Bits	Description
MAC	mac	48	[47:0]	MAC to learn.
GID	gid	16	[63:48]	Bit [15:12] Reserved. Bit [11:0] Global identifier to learn.
Unicast Port or Multicast Pointer	portOrPtr	32	[95:64]	Bit [31:10] Reserved. Bit [9:0] destPort or mcAddr field in the L2 Destination Table .
Unicast	uc	8	[103:96]	Bit [7:1] Reserved. Bit [0] uc field in the L2 Destination Table .
Drop	drop	8	[111:104]	Bit [7:1] Reserved. Bit [0] pktDrop field in the L2 Destination Table .
L2 Destination Table Address	daDestAddr	32	[143:112]	When the value is within the range 0 to 4127, the learning request will be updated to the corresponding entry in the L2 Destination Table . An out-of-range value will ask the learning unit to provide an available entry, and fill it with the data from the learning tag.
Valid	valid	8	[151:144]	Bit [7:1] Reserved. Bit [0] valid field in the L2 Aging Table .
Static	stat	8	[159:152]	Bit [7:1] Reserved. Bit [0] stat field in the L2 Aging Table .
Hit	hit	8	[167:160]	Bit [7:1] Reserved. Bit [0] hit field in the L2 Aging Table .
L2 Action Table DA Status	l2ActDa	8	[175:168]	Bit [7:1] Reserved. Bit [0] l2ActionTableDaStatus field in the L2 Destination Table .
L2 Action Table SA Status	l2ActSa	8	[183:176]	Bit [7:1] Reserved. Bit [0] l2ActionTableSaStatus field in the L2 Destination Table .
Meta Data	meta	16	[199:184]	metaData field in the L2 Destination Table .

Table 18.2: Learning Header



The fields in the learning tag consist of [FIB](#) fields and one 32-bit field for the L2 destination table address. Based on the different values, the learning tag can either directly update an entry in the FIB or ask the learning unit for an available entry. When the value is less than 4,128, the corresponding table entry will be updated directly, regardless of its current state. According to Section [Hash Collision Accommodation](#), the first 4,096 entries in the [L2 Destination Table](#) are reserved for [L2 DA Hash Lookup Table](#) hits and the rest are for [L2 Lookup Collision Table](#) hits, hence the L2 destination table address implies the address to the two search tables.

When the field value exceeds the range of the [L2 Destination Table](#), the learning unit will execute the task to find an available FIB entry, and update it with the corresponding information from the learning tag fields.

18.3.2 Direct Access to FIB

All tables in the [FIB](#) allow direct software writes through a configuration interface. However, the learning and aging engine may constantly update the FIB. Before updating the FIB from the configuration interface the learning and aging engine needs to be turned off through the [Learning And Aging Enable](#) register to avoid hazards. An alternative approach is to use reserved static entries as described in Section [Software Reserved Entry](#).

If the hardware learning unit needs to be turned on again after software setups, it is important to write to both L2 aging tables and the corresponding shadow tables while setting valid entries. Partial validation will cause inconsistencies between the L2 forwarding process and the learning and aging engine. Since the FIB consists of multiple tables it is recommended that the shadow tables are updated in the last step, to ensure the data consistency.

18.3.3 Software Reserved Entry

If the [stat](#) field in the [L2 Aging Table](#) is set to 1 and the [valid](#) field is set to 0, the corresponding entry in the FIB is considered as a reserved static entry and can be used for future software configuration. A reserved static entry is not used for L2 forwarding and is not available as a hardware learning entry.

A typical use case is to pre-allocate entries for L2 multicast. The hardware learning unit can automatically learn L2 unicast but not L2 multicast. One way to reserve entries for L2 multicast is to create a reserved static bucket, i.e. choose one bucket from the L2 hash table and make all entries reserved static. This approach allows the software to update entries in the reserved bucket during traffic without checking hash collisions, and without turning off the hardware learning and aging engine.

18.3.4 Software Aging

The aging unit has a software aging mode which can take over the automatic aging turned on in the [Software Aging Enable](#) register. Under software aging mode the aging steps will then be:

1. Software determines the time to age and responsible to periodically trigger the aging process.
2. Software writes 1 to the [Software Aging Start Latch](#) register to trigger an aging check period.
3. The same procedure as the automatic aging is done, [hash_aging](#) and [cam_aging](#) interrupts listed in Table [31.5](#) are raised.

18.4 Software And Hardware Interaction

The three units in the learning and aging engine (learning unit, aging unit, hit update unit) share the same writeback bus to the [FIB](#) as in Figure [18.1](#), the learning unit has the highest priority, followed by the hit update unit and then the aging unit. In order to let software keep track of FIB updates from the learning and aging engine, the writeback bus is snooped and transactions are made available in three FIFOs. The FIFOs are accessible from the configuration interface.

- [Learning Data FIFO](#) (LDF)
- [Aging Data FIFO](#) (ADF)
- [Hit Update Data FIFO](#) (HDF)



18.4.1 Data FIFO Interrupts

For each of the three FIFOs there are two interrupts:

- High watermark interrupt: `ldf_level/adf_level/hdf_level` interrupt in Table 31.5. The threshold is configurable through:
 - [Learning Data FIFO High Watermark Level](#)
 - [Aging Data FIFO High Watermark Level](#)
 - [Hit Update Data FIFO High Watermark Level](#)
- Overflow interrupt: `ldf_full/adf_full/hdf_full` interrupt in Table 31.5

The LDF/ADF/HDF are all tail drop FIFOs, if new entries are to be pushed to a full LDF/ADF/HDF they will not be written but ignored and cause `ldf_full/adf_full/hdf_full` interrupt. The HDF holds the hit update result which does not change L2 forwarding behaviors, but if software is unable to keep up reading out the LDF/ADF and cause `ldf_full/adf_full` interrupt, then software is no longer in sync with hardware tables. A way to recover from this would be:

1. Turn off the learning and aging engine.
2. Read out all the entries in the LDF/ADF/HDF to make sure they are empty.
3. Read out all tables in the [FIB](#) to compare between software tables. Update whatever the difference is to make tables become synchronized again.
4. Turn on the learning and aging engine.

18.4.2 Writeback Bus Control

As mentioned in Section [Hardware Learning and Aging](#), the writeback bus can be configured through the [Learning And Aging Writeback Control](#) register to block the hardware learning/aging/hit-update decisions to the [FIB](#). By doing so the automatic hardware learning/aging/hit-update units cannot do any changes to the FIB. If needed, the software is able to inspect the hardware decisions from LDF/ADF/HDF and update the FIB either through learning packets or direct table accesses.



Chapter 19

Spanning Tree

Spanning-Tree Protocol (STP) and Multiple Spanning-Tree Protocol (MSTP) support is provided in order to create loop-free logical topology when several ethernet switches are connected. Through registers the STP state of the ports can be controlled by the host SW. The default behavior at power up is that spanning tree is not enabled and spanning tree functionality must therefore be configured by SW before it can be used. A switch running the spanning-tree protocols utilizes BPDU (Bridge Protocol Data Unit) frames to exchange information with other switches in order to decide how to configure it's ports to get a loop-free (tree) logical network topology.

BPDUs are forwarded to the CPU based on the used destination address. By default the MAC multicast addresses 01:80:C2:00:00:00 and 01:00:0C:CC:CC:CD are forwarded to the CPU. Modifications of this is possible through the register [Send to CPU](#).

In order to be able to forward BPDU frames from the CPU to other switches on egress ports where general forwarding is currently not allowed, the bit [enable](#) in register [Forward From CPU](#) shall be set.

More information on the forwarding features to and from the CPU port is available in [Chapter 30](#)

19.1 Spanning Tree

The Spanning-Tree Protocol (STP) state for a port can be independently configured for source and egress behaviors to allow precise management. For ingress in the [spt](#) field of [Source Port Table](#). Similarly for egress, the STP state can be configured in the [sptState](#) in the [Egress Spanning Tree State](#). When STP is used on a port, all the port's associated MSTP instance states (ingress and egress) shall be **Forwarding**, i.e. MSTP is not enabled for this port. The behavior of the different STP states. The difference between Ingress and Egress STP state is only that learning is not affected by the Egress state.

- **Blocking and Listening**
Learning is disabled and no frames are forwarded except BPDU which will be forwarded to the CPU. Frames that are not forwarded is counted in a drop counter.
- **Learning**
Learning is enabled but no frames are forwarded except BPDU which will be forwarded to the CPU. Frames that are not forwarded is counted in a drop counter.
- **Forwarding and Disabled**
Normal operation, learning is enabled and normal switching. BPDU frames will be forwarded to the CPU.

19.2 Multiple Spanning Tree

When VLANs are used in a network there is a need for the Multiple Spanning Tree Protocol (MSTP) to manage the individual spanning-tree instances for the different VLANs. If an incoming frame doesn't have an assigned VLAN membership it will get a default VLAN membership automatically as described in [Chapter 5](#). VLAN membership decides which MSTP instance (MSTI) the frame belongs to. Hence, all frames will belong to an MSTI. The [msptPtr](#) in the register [VLAN Table](#) is an index to the MSTI tables which the packet shall be assigned to. The port's states of this MSTI are available in the tables [Ingress Multiple Spanning Tree State](#) and [Egress Multiple Spanning Tree State](#) for ingress and egress respectively. When a port uses MSTP it's STP states (source and egress) shall be set to **Disabled**, i.e. STP is not enabled for this port.

19.3 Spanning Tree Drop Counters

When a port's ingress or egress spanning tree states causes a frame to be dropped on a switch slice, the frames direction and spanning-tree state are used to select which drop counter on this slice to increase with one. The available drop counter registers are:

- **Ingress Spanning Tree Drop: Listen**
- **Ingress Spanning Tree Drop: Learning**
- **Ingress Spanning Tree Drop: Blocking**
- **Egress Spanning Tree Drop**

The ingress registers are common for all ports in to a switch slice. There is one egress register per port on each switch slice.

The registers above are also used to count MSTI-state caused frame drops. A port's ingress-MSTI drop-causing state is mapped as follows: The state **Learning** is mapped to the register **Ingress Spanning Tree Drop: Learning** and **Discarding** to **Ingress Spanning Tree Drop: Blocking**. For a port's egress MSTI, both the states **Learning** and **Discarding** are mapped to the port's generic egress drop counter **Egress Spanning Tree Drop**.



Chapter 20

Token Bucket

This core provides a rich set of QoS functions, and when a function needs to compare the internal packet or byte rate to a configurable rate, we use token bucket as the basic measurement component. A token bucket is usually combined with packet classifications, packet colorings or the shared buffer memory to achieve metering, marking, policing or shaping with different granularities.

A token bucket has four key parameters:

- bucket capacity
- bucket threshold
- initial tokens in the bucket
- token fill in rate

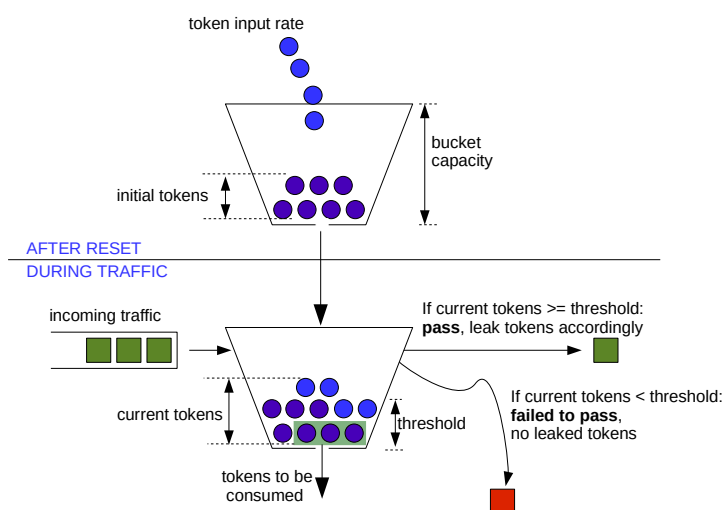


Figure 20.1: General Token Bucket Illustration

Figure 20.1 shows a token bucket with adjustable bucket threshold, the remaining tokens below the threshold can be used to handle the burst. This type of token bucket is used by:

- [multicast broadcast storm control](#)
- [queue shaper](#)
- [prio shaper](#)
- [egress port shaper](#)

In different QoS functions, tokens are represented as packets or bytes. The token fill in rate is achieved by periodically adding a certain number of tokens to the bucket and the fill in frequency is determined by one of the eight core ticks.



Chapter 21

Egress Queues and Scheduling

The order of packet output on each egress port is decided by a complex interaction of back-pressure and different QoS functions, but at the heart of the matter is the egress queue. The egress queues are the lists of packet pointers created by the queue manager when packets have been written to the packet buffer. Each egress port has eight such queues.

When a packet has been written in full to the packet buffer, the queue manager will add pointers to the packet to the end of at least one egress queue¹.

More than one egress port may get the packet linked (due to multicast), but on any single port the same packet may only be linked once. You cannot have the same packet in more than one egress queue on any single egress port.

The order in each egress queue is fixed. Once the packets are linked, the order cannot be changed. What QoS functions and back-pressure can affect is the order in which the packets in different queues are output.

Each egress queue has a *priority* (or *prio*) attribute, ranging from zero to seven. There are no limitations to how the priorities are assigned. All egress queues may have the same priority, or they may all have different priorities (if there are enough priorities to go around). If at all possible, an egress queue with a higher² priority will always get to output a packet before a queue with a lower priority. Egress queues with the same priority will be selected in a round robin manner by the DWRR scheduler.

The egress queue is determined by the ingress packet processing. If a packet is forwarded to multiple egress ports, each packet instance will have the same egress queue assigned.

21.1 Determine Egress Queue

Figure 21.1 describes how the egress queue is determined. If a configuration in the diagram includes a reference number in the end, the related field or register to setup can be found in the list below:

1. **Configurable ACL Engine** has a `forceQueue` action enabled.
2. `forceQueue` in **Reserved Source MAC Address Range**
3. `forceQueue` in **Reserved Destination MAC Address Range**
4. `prioFromL3` in **Source Port Table**
5. **IPv4 TOS Field To Egress Queue Mapping Table**
6. **IPv6 Class of Service Field To Egress Queue Mapping Table**
7. **MPLS EXP Field To Egress Queue Mapping Table**
8. `eQueue` in **Force Unknown L3 Packet To Specific Egress Queue**
9. `forceQueue` in **Force Non VLAN Packet To Specific Queue**

This process is completed only once per packet, and the result is applied to all destination ports for the packet. The input to the process can come from:

¹That is unless the packet is to be dropped, because then the pointer is instead added to the end of the throw queue.

²Priorities are numbered backward, so zero is the highest priority

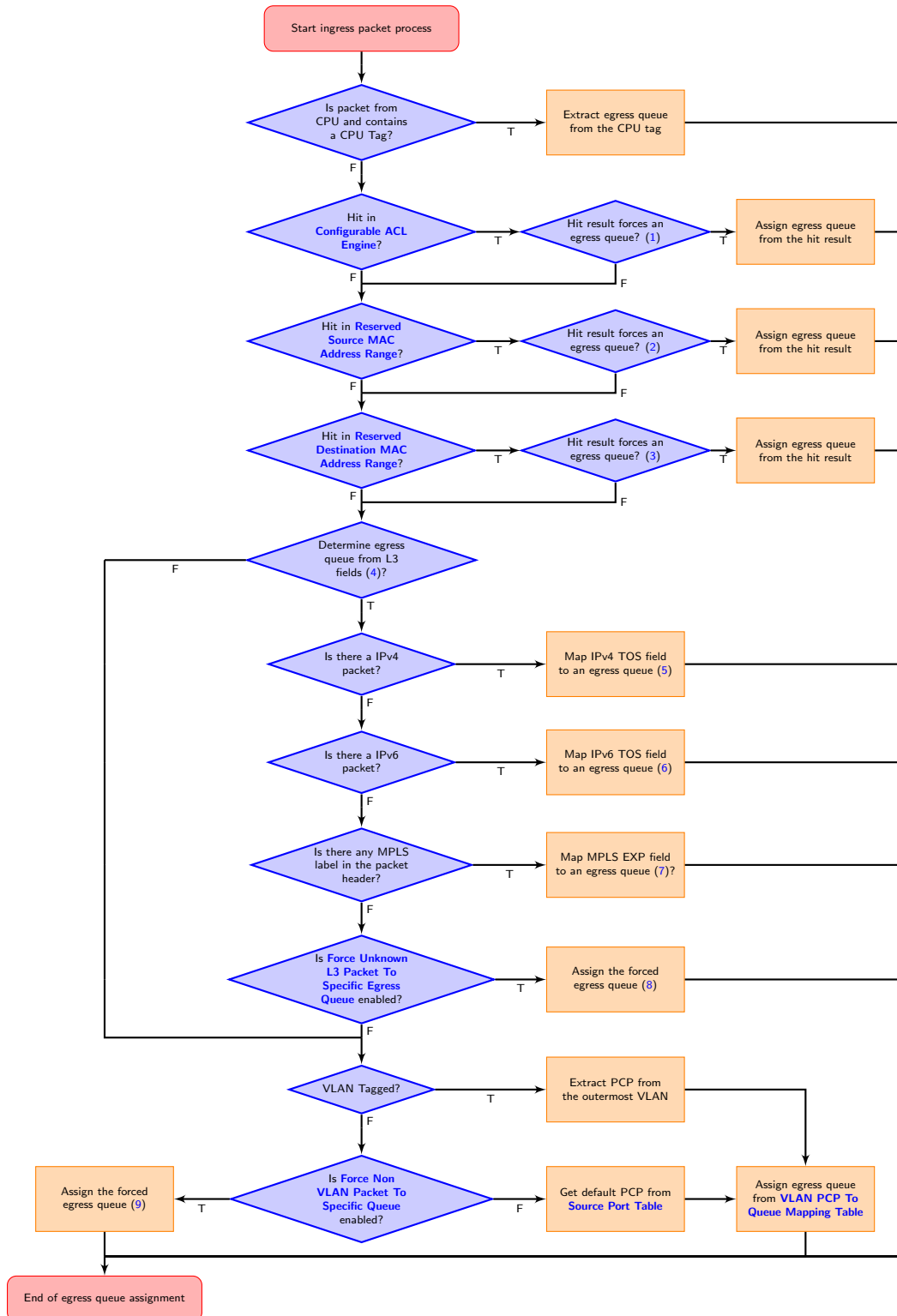


Figure 21.1: Egress Queue Selection Diagram



- Packet L2 headers
- Packet L3 headers
- Packet L4 ports
- Classification results

The available classification engines are described in the [Classification](#) chapter.

Egress queue from packet headers is operated under either trust L2 mode, to map egress queues from L2 headers, or trust L3 mode, to map egress queues from both L2 and L3 headers. In trust L2 mode, the egress queue can be mapped from:

- Priority code point(PCP) field from the outermost VLAN tag
- Source port default PCP when packet is non-VLAN tagged
- Optionally force non-VLAN tagged packets to the same egress queue, ignores source port based default mapping.

In trust L3 mode, a packet first tries to get its egress queue by mapping from:

- Type of Service (TOS)/DiffServ field from IPv4
- Traffic Class(TC) field from IPv6
- Traffic Class(TC)/EXP field from MPLS
- When none of the above are executed, the egress queue mapping under trust L3 mode will fall back on the trust L2 mode and get the egress queue from L2 headers of the packet.

21.2 Determine a packets outgoing QoS headers PCP, DEI and TOS fields

21.2.1 Remap Egress Queue to Packet Headers

This core supports remapping determined egress queues to outgoing packets' headers. These remappings are done first then if field [useEgressQueueRemapping](#) is set to one the remapping described in [21.2.2](#).

- Egress queue to next hop router VLAN PCP remapping:
For routed packets, packets' original VLAN tags are removed and at most two next hop router VLANs are added. Egress queue can be mapped to the PCP field in these VLAN tags through the [Router Egress Queue To VLAN Data](#) table when:
 1. [innerVlanAppend](#) is set and its PCP field selection([innerPcpSel](#)) chooses mapping from egress queue.
 2. [outerVlanAppend](#) is set and its PCP field selection([outerPcpSel](#)) chooses mapping from egress queue.
- Egress queue to next hop router VLAN CFI/DEI remapping:
Similar with next hop router VLAN PCP mapping, egress queue can be mapped to the CFI/DEI field in next hop router VLANs through the [Router Egress Queue To VLAN Data](#) table when:
 1. [innerVlanAppend](#) is set and its CFI/DEI field selection([innerCfiDeiSel](#)) chooses mapping from egress queue.
 2. [outerVlanAppend](#) is set and its CFI/DEI field selection([outerCfiDeiSel](#)) chooses mapping from egress queue.
- Egress queue to outgoing outermost VLAN PCP remapping:
Egress port VLAN push or swap operation provides an option to map egress queue to the outgoing outermost VLAN PCP field. The mapping table is [Egress Queue To PCP And CFI/DEI Mapping Table](#) and the required configurations are:
 1. [vlanSingleOp](#) in [Egress Port Configuration](#) is *push* or *swap*.
 2. [pcpSel](#) in [Egress Port Configuration](#) selects mapping from egress queue.
- Egress queue to outgoing outermost VLAN CFI/DEI remapping:
Similar with outgoing outermost VLAN PCP mapping, egress port VLAN push or swap operation provides an option to map egress queue to the outgoing outermost VLAN CEI/DEI field. The mapping table is [Egress Queue To PCP And CFI/DEI Mapping Table](#) and the required configurations are:
 1. [vlanSingleOp](#) in [Egress Port Configuration](#) is *push* or *swap*.
 2. [cfiDeiSel](#) in [Egress Port Configuration](#) selects mapping from egress queue.



- Egress queue to MPLS TC/EXP remapping:
Packets with MPLS labels have an option to map their egress queues to MPLS TC/EXP field when egressing the core. The mapping table is [Egress Queue To MPLS EXP Mapping Table](#) and the required configurations are:
 1. [mplsOperation](#) is *push* or *swap*.
 2. [expSel](#) in [Next Hop MPLS Table](#) selects mapping from egress queue.

21.2.2 Using Packet Type, Destination Port and Switching/Routing to do QoS Mappings

This core supports remapping determined by egress queues to outgoing packets' headers using the information if the packet was switched, routed, forwarded by classification rules, if the packet type was IP or MPLS and packets outgoing PCP, DEI, TOS and EXP fields. The steps to remap the packet are described below. The input values for PCP, DEI comes from the remapping tables described in [21.2.1](#). The TOS values comes from the [Color Remap From Ingress Admission Control](#) or [Color Remap From Egress Port](#).

1. Determine Which Mapping Table To Use
The mapping table to use to map the internal state to a the outgoing packet is determined by the table [Select Which Egress QoS Mapping Table To Use](#). The packets destination port, packet type and packet forwarding type is used to calculate which entry to read out from the table. This table then points to the one of the QoS remapping tables which remapps the internal state to the outgoing packets PCP,DEI and potentially L3 fields such as TOS field . Since the address takes egress port, forwarding type and packet type into consideration there can be separate rules setup for how to remap the fields in the outgoing packet.
2. Mapping Tables
Use the Mapping tables to map into outgoing packets PCP,DEI , TOS and EXP values.
 - (a) [L2 QoS Mapping Table](#)
This table can be used for all packets being sent out. There exists 2which the field [whichTablePtr](#) points to which to use.
 - (b) [IP QoS Mapping Table](#)
This table can be used for IPv6 and IPv4 packets. There exists 2L3 mapping tables. This remaps part of the TOS byte which has to do with ECN and uses the higher TOS bits [7:2] from the coloring tables ([Color Remap From Ingress Admission Control](#) or [Color Remap From Egress Port](#)).
 - (c) [TOS QoS Mapping Table](#)
This table can be used for IPv6 and IPv4 packets. There exists 2TOS mapping tables. This remaps the whole of the TOS byte from [Color Remap From Ingress Admission Control](#) or [Color Remap From Egress Port](#) to a new TOS bytes along with PCP and DEI information. There is a support to remap to EXP values which can be used if the packet enters a MPLS tunnel in the Next Hop Tables
 - (d) [MPLS QoS Mapping Table](#)
This table can be used for MPLS packets. This remaps the outgoing packets PCP, DEI and EXP values. There exists 2TOS mapping tables.

21.3 Priority Mapping

Each queue is mapped to one of eight egress priorities in the [Map Queue to Priority](#) register. Thus each priority will have between none and all queues as members. The priority mapping affects the scheduling of the packets. See Section [21.6](#), below for the details.

The priorities are ranked in descending order, from the highest priority (zero), to the lowest (seven).

Note that the priority mapping must not be changed for any queue that has packets queued. Doing so would make the ERM counters irrevocably corrupted, necessitating a reset for the core to continue normal operation.

21.4 Shapers

For a queue to be eligible for sending a packet there has to be a packet available in the queue and the average bandwidth for the queue, as measured by the token buckets in the queue shaper, has to be below the threshold set up in the [Queue Shaper Rate Configuration](#) registers.

Additionally the average bandwidth of the priority to which the queue is mapped has to be below the threshold set up in the [Prio Shaper Rate Configuration](#) registers.



21.4.1 Queue Shaper

The egress queue rates are shaped by token buckets configured in the [Queue Shaper Rate Configuration](#) registers. While the token bucket level is below the threshold configured in the [Queue Shaper Bucket Threshold Configuration](#) register, no new packets are scheduled for the corresponding egress queue. Ongoing packets are not affected by the shaping bucket status.

The queue shapers are enabled using the [Queue Shaper Enable](#) register, and the saturation level of the queue shaper buckets is controlled by the [Queue Shaper Bucket Capacity Configuration](#) register.

21.4.2 Prio Shaper

The egress prio rates are shaped by token buckets configured in the [Prio Shaper Rate Configuration](#) registers. While the token bucket level is below the threshold configured in the [Prio Shaper Bucket Threshold Configuration](#) register, no new packets are scheduled for the corresponding egress prio. Ongoing packets are not affected by the shaping bucket status.

The prio shapers are enabled using the [Prio Shaper Enable](#) register, and the saturation level of the prio shaper buckets is controlled by the [Prio Shaper Bucket Capacity Configuration](#) register.

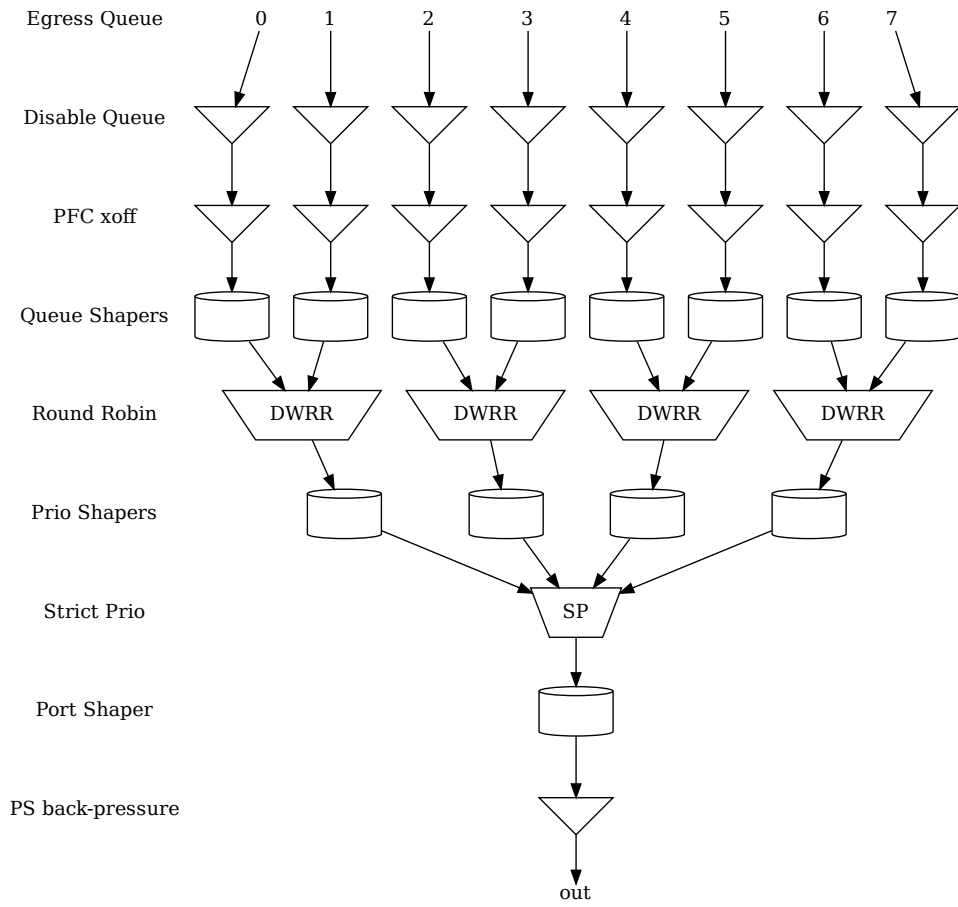


Figure 21.2: Egress Queue Scheduling example. Here using half the priorities, with two queues mapped to each.

21.5 Scheduling

The egress queue scheduling is accomplished by a combination of strict priority schedulers for the priorities and round robin queue schedulers for the queues mapped to the same priority. A visual representation of this is can be found in Figure 21.2. This figure is an example where half the priorities are used and two queues map to each priority³.

For a priority to be allowed to output a packet it must have mapped queues with available packets. It must also:

- be allowed to send by the prio shaper
- not be paused
- not be halted

From the priorities getting through the above needle's eye the highest priority is selected, and then the available queues mapped to that priority are selected by a byte-based deficit weighted round robin scheduler (described below).

21.6 DWRR Scheduler

The DWRR scheduler only acts on queues mapped to the same priority. Within each group of such queues it selects the most appropriate queue to output by comparing the number of bytes output for each queue with the weights set up for the queues.

This is accomplished using one byte counting bucket per queue and port. The non-empty queue with the highest bucket count in the group is selected. Bytes are subtracted from the corresponding bucket when a packet is sent out. Whenever the value in a bucket goes below the value configured in the **threshold** field of the **DWRR Bucket Misc Configuration** register, the buckets for all the queues belonging to the same priority will be replenished. The number of bytes added to each bucket is **weight** \ll X , where weight is taken from the **DWRR Weight Configuration** register, and X is a multiplier (for all queues) that is calculated to make sure that at least one cell worth of bytes is added to the queue that went below the threshold.

$$X = \max(0, \text{highestSetBit}(\text{cellBytes}) - \text{highestSetBit}(\text{weight}))$$

If a queue has no data to send, its bucket will eventually saturate at the cap set in the **DWRR Bucket Capacity Configuration** register.

The value in the **ifg** field of the **DWRR Bucket Misc Configuration** is additionally subtracted from the buckets for each packet.

21.7 Queue Management

This core features a set of queue management operations which can be used by the CPU to monitor, redirect and disable queues and ports. The current size of the queues can be readout by using the **Egress Port Depth** and **Egress Queue Depth** registers, while the current total number of cells left available can be seen in the **Buffer Free** register. The minimum level reached since core was initialized is available in **Minimum Buffer Free**. From this status the CPU can take active actions to determine what the core shall do with the packets on the ports. The optional operations are listed below.

- **Disable scheduling to port:** Disable the core from scheduling a new packet for transmission on a specific port and queue. This is setup in the **Output Disable** register. This allows per-queue granularity of what packets gets scheduled on a specific port. The packets are still kept in the queues until the port or queue is enabled again.
- **Disable queueing to port:** Disable the enqueueing of packets to a specific port and queue. Once the corresponding bit in the **Enable Enqueue To Ports And Queues** register is cleared, no new packets will be queued to that egress queue. New packets destined to that specific queue will be dropped and the **Queue Off Drop** counter for the egress port will be incremented.
- **Drain port:** Drop all packets in all queues on one specific port. This allows the user to clear all packets which have been queued on a port. The register **Drain Port** is used to control this functionality. Statistics for this operation is collected in the **Drain Port Drop** counter.

³So other similar diagrams would result with different settings in the **Map Queue to Priority** register.



21.8 How To Make Sure A Port Is Empty

First, so that no new packets are queued to the port, use the [Enable Enqueue To Ports And Queues](#) to disable all the queues on the port. If the already queued packets should not be sent out, then use the [Drain Port](#) functionality. Once this is done start to read out the [Packet Buffer Status](#) and check the bit which corresponds to the port. When the port bit is high, this means that all the queues on this port are empty.

Now, there may still be a few cells of data being processed in the egress packet processing pipeline, or stored in the parallel-to-serial memories. This data will be drained at the speed of the port, so the time from the port-bit going high in the [Packet Buffer Status](#) register to the port being truly empty will depend on the port speed.



Chapter 22

Packet Coloring

22.1 Ingress Packet Initial Coloring

This core marks packets with 3 colors internally to represent packet drop precedences. The three colors are coded as in Table 22.1.

Color	Code
Green	0
Yellow	1
Red	2

Table 22.1: Code for Colors

A packet's initial color is assigned according to L2/L3 protocols or classification results. It follows similar process steps as the egress queue assignment described in Section 21.1.

1. **Configurable ACL Engine** has a forceColor action enabled.
2. **forceColor** in **Reserved Source MAC Address Range**
3. **forceColor** in **Reserved Destination MAC Address Range**
4. **colorFromL3** in **Source Port Table**
5. **IPv4 TOS Field To Packet Color Mapping Table**
6. **IPv6 Class of Service Field To Packet Color Mapping Table**
7. **MPLS EXP Field To Packet Color Mapping Table**
8. **forceColor** in **Force Unknown L3 Packet To Specific Color**
9. **forceColor** in **Force Non VLAN Packet To Specific Color**

A diagram in Figure 22.1 describes how initial colors are determined. All classification engines which can force egress queues also have an option to force packet initial colors. If none of the engines force the color and the initial color marking is operating under trust L2 mode, the color is mapped from:

- Priority Code Point(PCP) field with Drop Eligible Indicator(DEI) field from the ingress outermost VLAN tag.
- Source port default PCP with default DEI when packet is non-VLAN tagged.
- Optionally force non-VLAN tagged packets to the same specific initial color, ignores source port based default marking.

Otherwise, the initial color marking will be working under trust L3 mode and the color is mapped from:

- Type of Service(TOS)/DiffServ field from IPv4
- Traffic Class(TC) field from IPv6
- Optionally force non-IP packets to the same initial color.

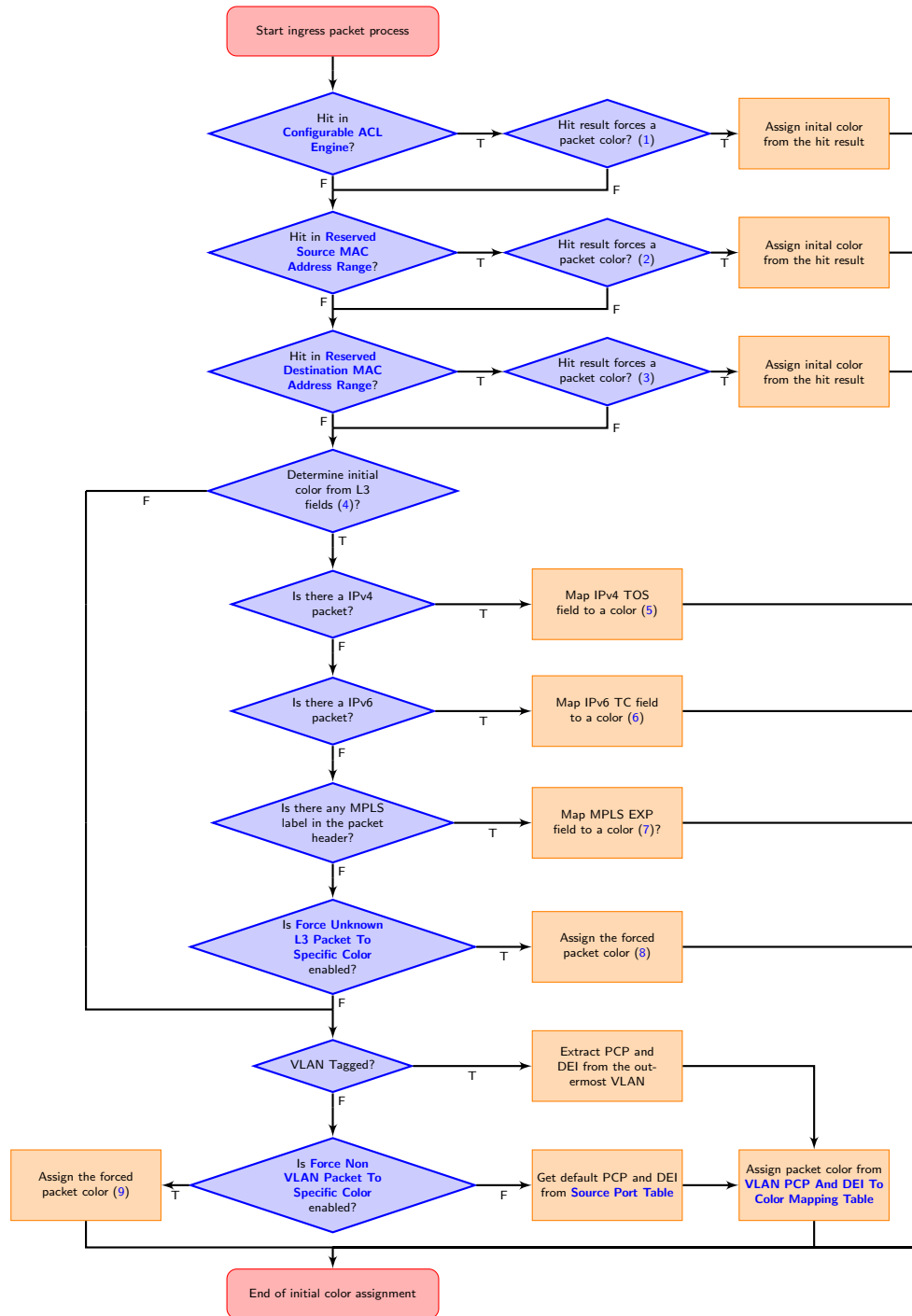


Figure 22.1: Packet Initial Color Selection Diagram

- When none of the above markings are executed, the initial color marking under trust L3 mode falls into processes in trust L2 mode.

By default, green marked packets have low drop probability, yellow marked packets have medium drop probability and red marked packets have high drop probability. But the remarking process has its own configurable settings to decide if packets with a certain remarked color shall be dropped.



22.2 Remap Packet Color to Packet Headers

During egress packet processing, each egress port can be set as color aware or color blind through the **colorRemap** field in the **Egress Port Configuration** table. If an egress port is color blind, packets to that port will not have its color represented in packet headers. If an egress port is color aware, a color remap process is executed to optionally remap the egress packet color to outgoing packet headers.

When an egress port is color aware, the default remap options for that port are configured in the **Color Remap From Egress Port** table. If a packet to a color aware egress port has ingress admission control applied, its meter-marker-policer pointer can also provide color remap options from the **Color Remap From Ingress Admission Control** table. The **enable** field in the table determines whether to perform a color remap operation for each pointer.

The color remap has four modes:

- Skip/Disable:
Color is not remapped to packet headers. This includes overriding previous color remap decisions.
- Remap to L3 only:
Color is remapped to IPv4 TOS field or IPv6 TC field with an AND mask (tosMask). For each bit in the TOS/TC field, the update requires the corresponding bit in the mask set to one. i.e.

$$\text{tos}[i] = (\text{color2Tos}[i] \& \text{tosMask}[i]) \mid (\text{tos}[i] \& (\sim \text{tosMask}[i]))$$
- Remap to L2 only:
A valid color remap updates the DEI bit in the VLAN tag of the outgoing packet. The updated DEI bit will not be changed during further egress packet processes. If there are more than one VLAN tag in the transmitted packet, the color to DEI mapping will be operated on the outermost VLAN.
- Remap to L2 and L3:
Color is remapped to both L2 and L3 fields as listed above.





Chapter 23

Admission Control

23.1 Ingress Admission Control

This core features an ingress admission control unit to control the bandwidth of certain traffic types. If the traffic flow in a group exceeds the configured bandwidth it may get the packet color changed or get denied to be enqueued in the buffer memory.

Ingress admission control includes two main functions. The first function creates admission control groups to classify packets based on source information in packet headers or ACL matches. The second function measures the classified traffic rate against a certain policy to make permit/deny decisions. The decision may take the given packet color into account.

23.1.1 Traffic Groups

The traffic group is classified based on source port number and L2 or L3 packet headers. Initially packets are grouped by their source port numbers and L2 priorities, but during the subsequent admission control processes they may fall into other traffic groups. For each potential traffic group, three configurations are given to validate a policy:

1. mmpValid: Determine if there is a valid Meter-Marker-Policer(MMP) pointer. If there is no valid pointer through the entire process, the packet will not be classified to any traffic group.
2. mmpOrder: Order of the pointer. If a valid pointer exists, its order needs to be higher than the order of previously assigned pointers to override them.
3. mmpPtr: MMP pointer for this traffic group.

The process to set the MMP pointer is illustrated in Figure 23.1. A packet can only belong to one traffic group so hierarchical traffic groups are not possible.

The order of the classification sequence is:

1. Source port number and L2 priority:
First assignment for traffic groups and MMP pointers. For VLAN tagged packet, L2 priority is from its outermost VLAN PCP field. For non-VLAN tagged packet, L2 priority is the default PCP based on the source port number (**defaultPcp** in the **Source Port Table**). Lookup in the **Ingress Admission Control Initial Pointer** table gives a base pointer and its order, also indicates if it is a valid pointer.
2. Source MAC:
Source MAC hit an entry in the **Reserved Source MAC Address Range**.
3. Destination MAC:
Destination MAC hit an entry in the **Reserved Destination MAC Address Range**.
4. ACL rules:
Hit in the **Configurable ACL Engine**.
5. Ingress VID:
Lookup in **VLAN Table** based on the **ingress VID**.
6. VRF:
For a routed packet, lookup in **Ingress Router Table** based on its VRF.

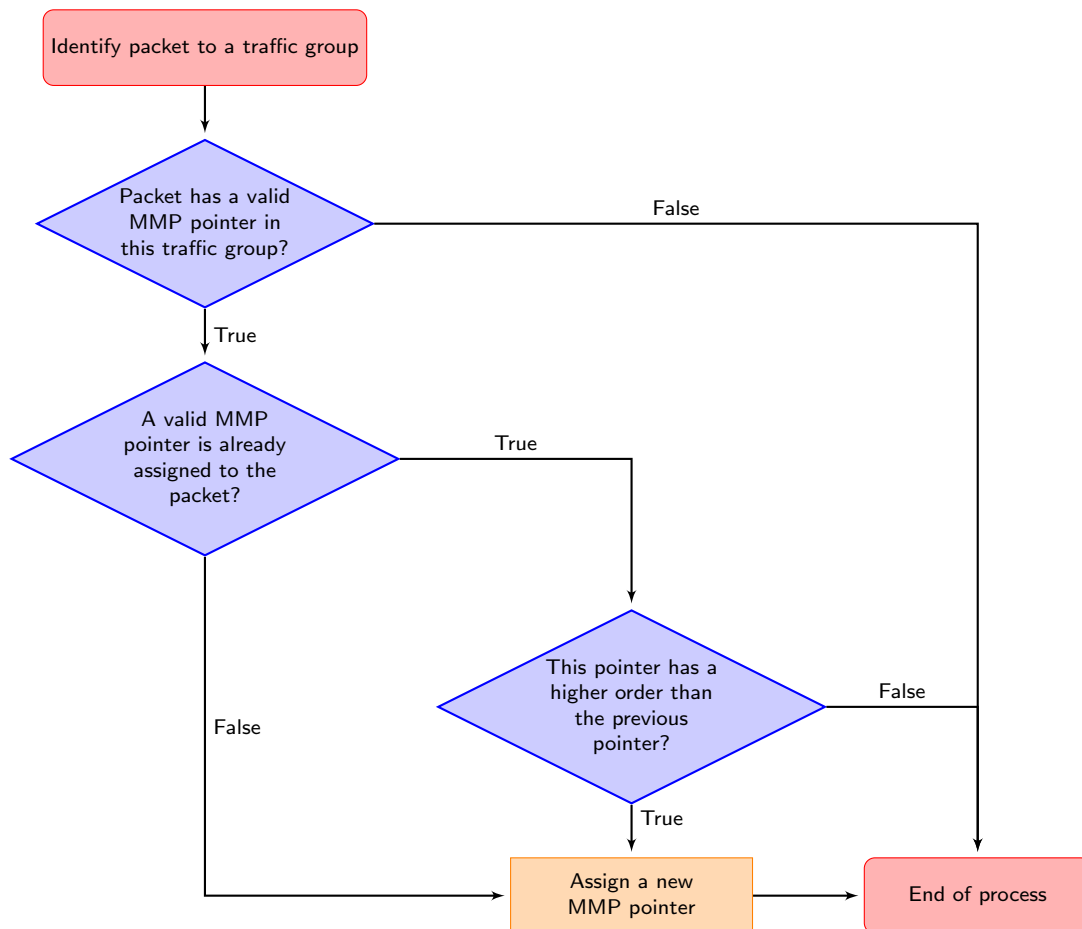


Figure 23.1: MMP pointer Selection Diagram

When a packet arrives to ingress packet processing, it walks through ingress admission control classifications in the order above. A hit in one of the above groups will result in a pointer and a matching order. The pointer is linked to a policy/entry in a meter-marker-policer engine, which will measure the byte rate belonging to this entry. Although a packet can have multiple hits in traffic groups, it will finally fall into one pointer according to the order of the pointers. Later matches only win when they have a higher order than the previous ones.

23.2 Meter-Marker-Policer

An admission control unit contains a meter-marker-policer (MMP) bank where each MMP refers to one admission control policy. An MMP is based on token buckets, and each entry includes two configurable buckets.

The MMP bank used by ingress admission control consists of 32 policies/entries with three related tables.

1. [Ingress Admission Control Token Bucket Configuration](#)
2. [Ingress Admission Control Reset](#)
3. [Ingress Admission Control Current Status](#)

While only one ingress admission control policy is applied to any single packet, the same policy/entry can be pointed to from several different traffic types.

In the Ingress Admission Control, an MMP entry is configured through the [Ingress Admission Control Token Bucket Configuration](#) register to perform either a single rate three color marker (RFC2697: srTCM) or a two rate three color marker (RFC2698: trTCM). The selected marker is operated in either color-aware or color-blind mode, and the packet is marked with a new color when the rate exceeds a certain bandwidth. Based on the updated



packet color, **dropMask** from register **Ingress Admission Control Token Bucket Configuration** decides whether the packet is allowed to be enqueued in the buffer memory.

An MMP entry has a **Ingress Admission Control Mark All Red Enable** option to permanently block the metering process and drop all packets with the corresponding MMP pointer. When **Ingress Admission Control Mark All Red Enable** is set to one, a packet drop on this entry will raise the **Ingress Admission Control Mark All Red** to one, then further packets to that entry will be dropped before metering. The blocking status can be cleared by writing zero to one of the two registers.

When an MMP is selected to be either srTCM or trTCM, it still requires configurations of the two token buckets to make it work properly.

- srTCM: Only the length, not the peak rate of the burst determines service eligibility.
 - Committed Information Rate (CIR): Combining **tokens 0** and **tick 0** to achieve the target rate. Details for tick is described in the **Tick** chapter. Configuration examples are shown in Table 23.1. Under srTCM mode, rate settings for the second token bucket (**tokens 1** and **tick 1**) will not take effect.
 - Committed Burst Size (CBS): **bucketCapacity 0**.
 - Excess Burst Size (EBS): **bucketCapacity 1**.
- trTCM: Enforce peak rate separately from the committed rate.
 - Committed Information Rate (CIR): **tokens 0** and **tick 0**.
 - Committed Burst Size (CBS): **bucketCapacity 0**.
 - Peak Information Rate (PIR): **tokens 1** and **tick 1**.
 - Peak Burst Size (PBS): **bucketCapacity 1**.
- Runtime configuration update:

Any update to register **Ingress Admission Control Token Bucket Configuration** requires writing 1 to register **Ingress Admission Control Reset**. This will reset the buckets to the initial state.
- Status update from hardware:

Besides **Ingress Admission Control Reset**, MMP has a another status register: **Ingress Admission Control Current Status**. It shows the number of tokens in each bucket. Hardware updates these two registers only when a metering process is done, hence **Ingress Admission Control Current Status** shows the number of tokens left in the bucket since the last token consumption in this bucket. **Ingress Admission Control Reset** is always changed back to 0 again after token consumptions.

Bandwidth	Token Bucket Update Frequency	Tick Index	Added Tokens Per Tick (bytes)
8000 bit/s	1KHz	3	1
16000 bit/s	1KHz	3	2
N*64000 bit/s	1KHz	3	N*8
N*1544000 bit/s	1KHz	3	N*193
N*56000 bit/s	1KHz	3	N*7
10M bit/s	10KHz	2	125
250M bit/s	10KHz	2	3125
N*1G bit/s	1Mhz	0	N*125

Table 23.1: Rate Configuration Example (Assume tickFreqList = [1MHz, 100KHz, 10KHz, 1KHz, 100Hz])





Chapter 24

Tick

All token buckets - and all other functions dependent on measuring time - in the core are basing their time measurements on the system ticks.

Tick number zero is the master tick. It is created by dividing the core clock by the number configured in the `clkDivider` field of the **Core Tick Configuration** register. The following tick signals (eight in total) are created by dividing the previous tick by a factor set up in the `stepDivider` field of the **Core Tick Configuration** register, so `tick1` is `clkDivider` slower than `tick0`, `tick2` is `clkDivider` slower than `tick1`, and so on.

If the **Core Tick Configuration** is updated during runtime, all features relying on the core tick need to be updated accordingly. Meanwhile, inaccurate time measurement will be performed until the first tick after the reconfiguration is generated.

By default the input to the Core Tick divider is the core clock, but using the **Core Tick Select** register the input to the tick divider can be disabled, or chosen to be driven from `debug_write_data` pin 0.



Chapter 25

Multicast Broadcast Storm Control

The multicast/broadcast storm control (MBSC) unit is used to make sure that a switch does not flood the network with too much multicast/broadcast traffic. The MBSC unit prevents several traffic types from transmitting to an egress port if the corresponding traffic rate on that egress port has exceeded a certain limit.

The basic component of the MBSC unit is a token bucket (illustrated in Figure 20.1). For each egress port there is one token bucket per inspected traffic type. In principle a token bucket controls the traffic rate (packet rate or byte rate) on an egress port. A token bucket operates as follows:

1. A configurable number of tokens are periodically added to the token bucket. The bucket level will saturate at the configured capacity.
2. When a packet of the traffic type is received a configurable number of tokens are consumed, i.e. the bucket level is decreased. The number of tokens consumed per packet is either packet length plus IFG adjustment or one per packet.
3. As long as the bucket level is at or above the threshold the bucket will accept all given traffic.
4. When the bucket level drops below the threshold all packets of the inspected traffic type, destined for the corresponding egress port, are dropped. Note that instances of the same packet destined for other egress ports are not affected and have their own token buckets to check the traffic rate.
5. The **MBSC Drop** counter will be incremented once for each egress port where the packet is dropped.

In this core three kinds of traffic are checked by the MBSC unit:

- L2 Broadcast
- L2 Flooding
- L2 Multicast

For each type of traffic there is an individual control unit, consisting of one token bucket per egress port. Every token bucket can be turned on or off separately through a control register (listed in the next section).

25.1 Inspected Traffic

- L2 Broadcast: A Packet with DA = ff:ff:ff:ff:ff:ff.
 - Token bucket configurations:
 - * **L2 Broadcast Storm Control Enable**
 - * **L2 Broadcast Storm Control Bucket Capacity Configuration**
 - * **L2 Broadcast Storm Control Bucket Threshold Configuration**
 - * **L2 Broadcast Storm Control Rate Configuration**
- L2 Flooding: A packet that will be L2 switched but the DA is unknown. In this case the packet is flooded to all VLAN member ports.
 - Token bucket configurations:
 - * **L2 Flooding Storm Control Enable**
 - * **L2 Flooding Storm Control Bucket Capacity Configuration**

- * [L2 Flooding Storm Control Bucket Threshold Configuration](#)
- * [L2 Flooding Storm Control Rate Configuration](#)
- L2 Multicast: A packet that will be L2 switched and has a known multicast DA MAC in the L2 tables. (The DA MAC has Ethernet multicast bit set to 1). The core can optionally include or exclude certain packets as L2 multicast traffic. The configuration is through the [L2 Multicast Handling](#) register.
 - Token bucket configurations:
 - * [L2 Multicast Storm Control Enable](#)
 - * [L2 Multicast Storm Control Bucket Capacity Configuration](#)
 - * [L2 Multicast Storm Control Bucket Threshold Configuration](#)
 - * [L2 Multicast Storm Control Rate Configuration](#)

25.2 Rate Configuration

From the configuration registers a token bucket can be shaped with its capacity, threshold and token settings. The L2 broadcast storm control is here used as an example to demonstrate the operations.

From the [L2 Broadcast Storm Control Rate Configuration](#) register a user can configure how tokens are consumed by a packet, and how new tokens are supplemented to the bucket.

- Token consumption
 1. The token bucket can be set to count either packets or bytes by the [packetsNotBytes](#) field. This setting puts a token bucket in either packet or byte mode to control the maximum packet rate or byte rate on an egress port respectively.
 2.
 - In packet mode, every L2 broadcast packet instance to an egress port will consume one token and the bucket value will be decreased by one.
 - In byte mode, every L2 broadcast packet instance to an egress port will consume as many tokens as there are bytes in the packet plus the specified IFG correction in the [ifgCorrection](#) field.
- Token Injection
 1. The token injection frequency is tick¹ based. The tick timer determines the time period between token injections. The [tick](#) field from the [L2 Broadcast Storm Control Rate Configuration](#) register selects which tick timer to use.
 2. When it is time to inject new tokens, the number of tokens that will be added is configured in the [tokens](#) field.
- Token bucket capacity and threshold. The two configuration registers [L2 Broadcast Storm Control Bucket Capacity Configuration](#) and [L2 Broadcast Storm Control Bucket Threshold Configuration](#) are used to setup how the token bucket handles traffic bursts.

By default the MBSC unit is operating in packet mode, and all token buckets are set to allow the inspected traffic to have at most 5% of the full packet rate for 64-byte packets. Python example code to configure the maximum packet rate to 5% follows:

```
#!/usr/bin/python

rate      = 0.05

minLen    = 64 # bytes
slice     = 4 # switch slices
ifg       = 20 # bytes
pnb       = 1 # = packet mode
portBW    = 100000 # Mbits/s
tickFreqList = [10.0,
                 1.0,
                 0.1,
                 0.01,
                 0.001,
                 0.0001,
                 1e-05,
                 1e-06] # Mhz
```

¹The system ticks are described in Chapter 24.



```
fullByteRate      = portBW/8.0
fullPktRate       = fullByteRate/(minLen+ifg)

pktRate = fullPktRate*rate
pktTokenIn      = 10*slice

tick = len(tickFreqList)-1
for i in range(len(tickFreqList)):
    if tickFreqList[i] * pktTokenIn <= pktRate:
        tick = i
        break

pktTokenIn = int(1.0*pktRate / tickFreqList[tick])

pktCap = pktTokenIn * 20
pktThr = pktTokenIn * 10

# Field settings for the rate configuration register
settings = {
    'packetsNotBytes' : pnb,
    'tokens'          : pktTokenIn,
    'tick'            : tick,
    'ifgCorrection'   : ifg,
    'capacity'        : pktCap,
    'threshold'       : pktThr}
```



Chapter 26

Egress Resource Manager

The core includes an Egress Resource Manager (ERM) unit for controlling the shared buffer memory occupancy of egress ports and queues. The primary objective of the egress resource manager is to avoid persistent buildup of queue length in the buffer memory and prevent the blockage of enqueueing at other ports and queues. Additionally, during buffer memory congestion, ERM facilitates prioritized enqueueing of egress queues with higher priorities.

The resource management granularity is cells and there are 4100 cells, each 256 byte wide, available in the buffer memory. A packet is written to the buffer memory with the original packet data plus a 29 byte ingress to egress header, thus a 1600 byte packet will have 1629 bytes and occupy six cells. A packet plus the ingress to egress header longer than n cells but shorter than $(n+1)$ cells will require $(n+1)$ cells for storage. For example, a 228 byte packet will use two cells. ERM traces the buffer memory occupancy and decides if a cell is allowed to be written to the buffer memory.

The ERM determines the congestion of the buffer memory based on the amount of free space (number of free cells) available. The ERM classifies the congestion levels into Green (no congestion), Yellow (slightly congested) or Red (heavily congested). When the buffer memory is in the yellow or red zone, **Resource Limiter Set** gives four sets of limits to check the queue length for different egress ports and queues. An egress port chooses limit sets for each of its queues from the **Egress Resource Manager Pointer** lookup.

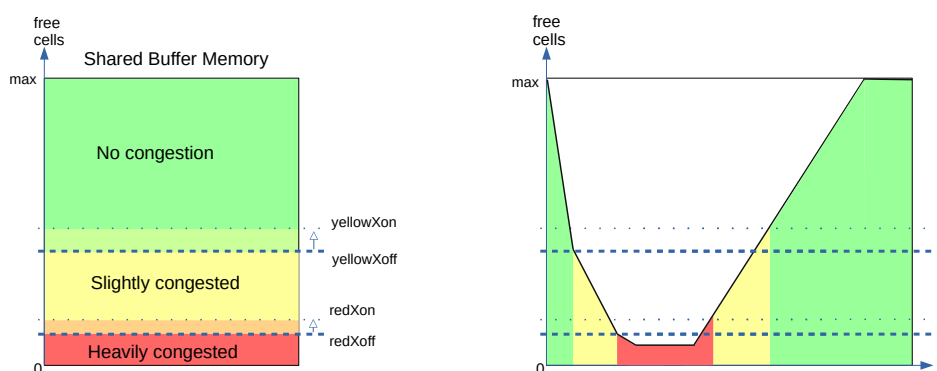


Figure 26.1: Buffer memory congestion zones

26.1 Yellow Zone

ERM Yellow Configuration defines how to enter and exit the yellow zone. The yellow zone is entered when the number of free cells goes below **yellowXoff**. To leave the yellow zone, the number of free cells need to go above **yellowXon**.

ERM checks

The buffer memory is considered partially congested when it is in the yellow zone. The ERM allows moderate buildups in all queues to a certain limit. An incoming cell of a packet is not allowed to be enqueued under two conditions:

1. The number of enqueued cells in the assigned egress queue is more than **yellowLimit**, while the total number of enqueued cells in the same queue and higher priority queues is more than **yellowAccumulated**.
2. **ERM Yellow Configuration** offers an optional check on a per egress port basis. A port can be considered as a red port in the yellow zone if the enqueued cells on that port are above **redPortXoff**. An incoming cell to a red port is not allowed if the length of the assigned queue is larger than **redLimit**.

26.2 Red Zone

ERM Red Configuration defines how to enter and exit the red zone. The red zone is entered when the number of free cells goes below **redXoff**. To leave the red zone, the number of free cells need to go above **redXon**.

ERM checks

The buffer memory is considered severely congested when it is in the red zone and the ERM shall only accept enqueueing to nearly empty queues. An incoming cell of a packet is not allowed to be enqueued in two cases:

1. The number of enqueued cells in the assigned egress queue is more than **redLimit**.
2. The ongoing packet length in cells has exceeded **redMaxCells**.

26.3 Green Zone

When the buffer memory is neither in the yellow zone nor in the red zone, the ERM considers the buffer memory to be uncongested and all incoming cells are accepted and stored in their assigned queues.

26.4 Configuration Example

A commonly used non-default ERM configuration involves allowing a queue to grow up to length **G** without packet drops (guarantees), and preventing new packets from being enqueued when the queue length is beyond **L** (limits). Between queue length **G** and **L** the enqueueing decision is made based on the overall free space in the buffer memory. This configuration imposes the following requirements:

1. $\text{redXon} \geq \text{redXoff} \geq \text{sum}(\text{redLimit})$
The red zone is used as guarantees, its configuration needs to ensure that **redXon** is large enough so that the buffer memory does not get full before all queues reach their **redLimit**. Set **redLimit** a few cells more than the desired guarantee size to have a margin for the latency.
2. Set **yellowAccumulated** to 0, ensuring that **yellowLimit** is always checked in the yellow zone.
3. $\text{yellowXon} \geq \text{yellowXoff} \geq \text{maxBufferFree}$
Put the ERM in the yellow zone even when the buffer memory is empty hence keep **yellowLimit** check under an always on state.

26.5 Restrictions

Be aware that the **Map Queue to Priority** settings need to be done when there is no traffic on any port. Update with ongoing traffic may provide a wrong enqueueing snapshot to the ERM and cause inconsistencies that can not be recovered without a reset.



Chapter 27

Flow Control

The purpose of flow control is to give access to storage in the packet buffer in an fair manner between the ports sending packets to this switch. No single source port or, if configured for it, traffic class, shall be able to behave in a way that punishes other source ports (or traffic classes). For this purpose flow control has two tools at its disposition: Pausing and tail-drop.

27.1 Pausing

Pausing, or Ethernet flow control, is a method of remote controlling the far-end interface's transmissions to this switch using dedicated pause frames. Hence, for successful pause operation the far-end interface also needs to be set up properly. The remote control is done by regularly sending pause frames (by this switch's MACs) to the far-end interfaces.

The switch core will only provide the MACs with a vector of the current pause state. It is up to the MAC to detect state changes and send the appropriate pause frames. The interface for the pause state vector is described in Section 31.5.

The pause frames are entirely handled by the MAC. It both creates frames and consumes incoming frames. The switch does not expect any pause frames on the packet interface from MAC, and the switch will not create any pause frames.

The beauty of pausing is that it can be used to set up flow control without packet drops. If the size of the packet buffer is large enough to cope with the data in flight from all the far end interfaces, and they all support pausing, it is possible to configure a completely drop-less system.

If, however, some far end interfaces do not support pausing, or the amount of data in flight is too large, it is necessary to make use of tail dropping.

27.2 Tail-Drop

Tail-drop is an implicit flow-control scheme. By deliberately dropping incoming packets (tail refers to the tail of the queue) there is an induced limitation of flows by Layer 3 transport protocols with flow control (e.g. TCP). So in contrast to Pausing, Tail-drop is not reliant on features of neighboring interfaces, but on features of higher level protocols. Transport protocols without flow control (e.g. UDP) will not limit their flows due to drops, but tail-drop will still prevent those flows, when misbehaving, from interfering with traffic from other source ports (or traffic classes).

Note that for flow control to function correctly all source ports have to be set up for either pausing or tail-drop (or both). If a single source port is not configured properly, it can starve all the others of buffering resources.

27.2.1 Tail-drop as police for Pausing

Even on Pause-enabled ports it may be useful to set up tail dropping as back-up for Pausing. By setting the tail-drop threshold at a level where we would have stopped receiving data from a Pausing-enabled source port, had it observed our pause frame, we can protect our packet buffering resources even in the case that a remote interface fails to act on the pause frame.

27.3 Buffer partitioning

The packet buffer space is partitioned into reserved and free-for-all (FFA) areas. Properly configured tail-drop will never drop a packet so long as only the reserved areas are used. Below I will use “resource” to mean “source port” on a non-PFC port and “source port/traffic class” on a PFC-enabled port.

The number of FFA cells that are allowed to be consumed by each resource before it will be hit by flow control is configured individually per resource. When the number of used free-for-all cells reaches the configured Xoff threshold, the pause state will be set to Xoff. And when the tail-drop threshold is exceeded a packet may be dropped (depending on whether there are reserves left).

The flow control decision will only be made once the last cell of a packet is about to be written to the packet buffer. Thus the thresholds need to be set so that there is space for one maximum packet per source port set aside.

27.3.1 Reserves

The tail-drop and the pausing share the reserved settings and the counters but the meaning of reserve is different between them. For tail-drop a reserve is really a reserve. Meaning that if, for instance, a source port still has reserves left it will not drop even if the global threshold is exceeded. For pausing, when an Xoff threshold is reached it will cause pausing whether or not there are reserves left. So when the global Xoff threshold is reached all ports with pausing enabled will be paused. Even those that have reserves left.

The reason that tail drop and pausing work differently is that pausing needs hysteresis between Xoff and Xon, and tail drop does not. It would be difficult to maintain the hysteresis if the reserves were observed for pausing.

Each port can be set up to work in either PFC-mode, and non-PFC-mode. In PFC-mode the accounting is done per port and traffic class, while in non-PFC-mode the accounting is only per port.

27.4 Non-PFC mode

In non-PFC mode the traffic class is disregarded, and accounting is only done per source port. The mode is controlled individually per source port by the **Port Pause Settings:mode** fields for pausing and by the **Port Tail-Drop Settings:mode** fields for tail-drop. The **Port Reserved** registers define the number of cells reserved per source port.

These counters are used in non-PFC mode:

- **FFA Used PFC**: The total number of free-for-all cells occupied by ports in PFC-mode
- **FFA Used non-PFC**: Total number of free-for-all cells occupied by ports in non-PFC-mode
- **Port Used**: Number of cells occupied by each source port

Note that the global threshold is for the sum of FFA cells, that is the sum of **FFA Used PFC** and **FFA Used non-PFC**

27.5 PFC-mode

In PFC mode accounting is additionally done per traffic class. The **Port/TC Reserved** registers define the number of cells reserved for each specific source port and traffic class combination.

Figure 27.1 illustrates the partitioning of reserved and FFA areas.

These counters are used in PFC mode:

- **FFA Used PFC**: The total number of free-for-all cells occupied by ports in PFC-mode
- **FFA Used non-PFC**: Total number of free-for-all cells occupied by ports in non-PFC-mode
- **Port FFA Used**: The number of free-for-all cells occupied for each source port
- **TC FFA Used**: The number of free-for-all cells occupied for each traffic class
- **PFC Inc/Dec Counters**: The cell counters per Port/TC are comprised of separate increment and decrement counters per Port/TC. The current counter value is calculated by taking the increment minus the decrement modulo the counter size.



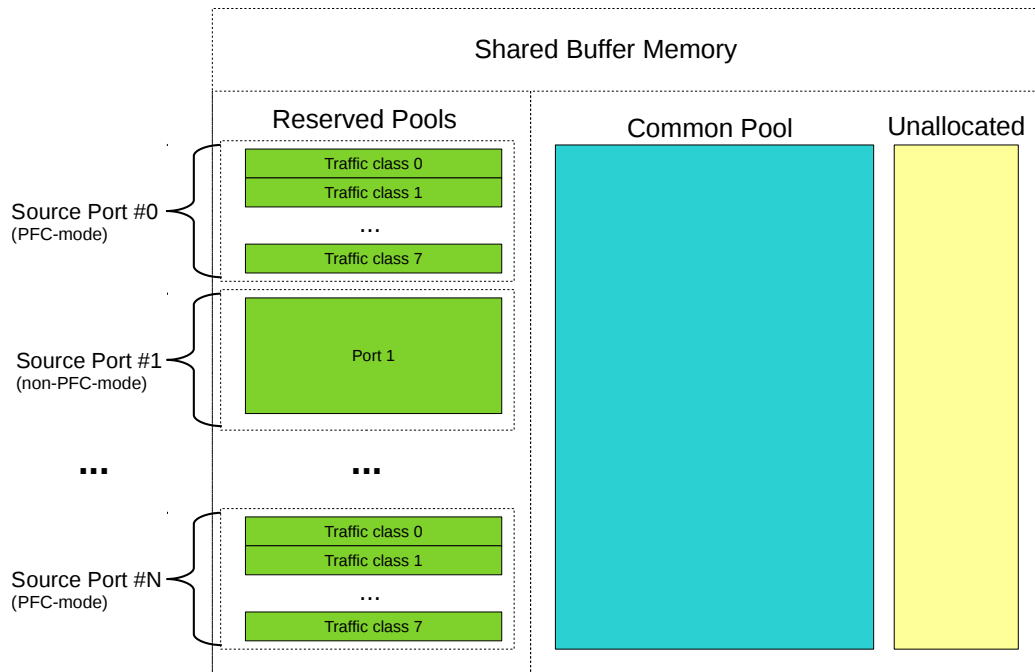


Figure 27.1: The buffer memory is partitioned into Reserved and FFA areas. The unallocated area is the space set aside for the currently incoming packets.

27.5.1 Pausing Thresholds

For tail-drop there is a single set of thresholds above which packets are dropped. For pausing there are two sets of thresholds, Xon thresholds and Xoff thresholds, thus forming a hysteresis area to avoid bursts of pause frames at the threshold. Going above the Xoff threshold will produce a pause frame turning off the packet flow at the remote interface, but to produce a pause frame turning it back on requires going all the way down below the Xon threshold.

These are the pausing thresholds:

- **Xoff FFA Threshold:** When the total number of used FFA cells is at or above this threshold the global pause state is set to paused.
- **Xon FFA Threshold:** When the total number of used FFA cells goes below this threshold the global pause state is set to un-paused.
- **TC Xoff FFA Threshold:** When the total number of used FFA cells for a traffic class is at or above this threshold the traffic class state is set to paused
- **TC Xon FFA Threshold:** When the total number of used FFA cells for a traffic class goes below below this threshold the traffic class state is set to un-paused.
- **Port Xoff FFA Threshold:** When the total number of used FFA cells for a source port is at or above this threshold the source port state will be set to paused.
- **Port Xon FFA Threshold:** When the total number of used FFA cells for a source port goes below this threshold the source port state is set to un-paused.
- **Port/TC Xoff Total Threshold:** When the sum of the FFA and Reserved cells used for a specific source port and traffic class combination is at or above this threshold, the state of this specific source port and traffic class combination will be set to paused.
- **Port/TC Xon Total Threshold:** When the sum of the FFA and Reserved cells used for a specific source port and traffic class combination goes below this threshold the state for this specific source port and traffic class combination is set to un-paused

Note that all thresholds are for the number of FFA cells used, except for the Port/TC threshold which is for the total number of cells used.



In non-PFC-mode each resource is affected by two thresholds: The source port threshold and the global threshold. Both need to be in the un-paused state for the source port to be set to un-paused.

In PFC-mode each resource (source port and traffic class) is affected by four thresholds:

- Source Port/Traffic Class
- Source Port
- Traffic Class
- Global

All four need to be in the un-paused state for the source port and traffic class combination to be set to un-paused.

27.5.2 Tail-drop Thresholds

For tail-drop there is no hysteresis so there is only a single set of thresholds:

- **Tail-Drop FFA Threshold:** When the total number of used FFA cells is above this threshold all packets will be dropped from the tail-drop-enabled ports that have no reserved cells left to spend
- **Port Tail-Drop FFA Threshold:** When the total number of used FFA cells for a source port is above this threshold incoming packets from this source port will be dropped unless the port is in PFC-mode and there are reserved cells left to spend
- **TC Tail-Drop FFA Threshold:** When the total number of used FFA cells for a traffic class is above this threshold any incoming packet belonging to the traffic class will be dropped unless the port/TC has reserved cells left to spend. Only valid in PFC-mode
- **Port/TC Tail-Drop Total Threshold:** When the sum of the FFA and Reserved cells used for a specific source port and traffic class combination is above this threshold any incoming packet from this source port assigned to this traffic class will be dropped. Only valid in PFC-mode

The **Tail-Drop FFA Threshold**, **TC Tail-Drop FFA Threshold** and **Port Tail-Drop FFA Threshold** are not obeyed strictly. The first packet exceeding the threshold may be accepted, causing a one-packet over-shoot.

27.6 Enabling Tail-Drop

Tail-drop is enabled per source port using the **Port Tail-Drop Settings:enable** fields. The individual thresholds are enabled using the enable fields in each threshold register. See Section 27.5.1 above.

27.7 Enabling Pausing

Pausing is enabled per source port using **Port Pause Settings:enable** fields. The individual thresholds are enabled using the enable fields in each threshold register. See Section 27.5.1 above.

27.8 Multiple Switch Slices

Flow control operates independently for the packet buffers in each switch slice. Thus there are individual counters, thresholds and enables per switch slice for each source port. For tail-drop the function is entirely local to a switch slice.

For pausing, however, the status to a MAC is only Xon if none of the slices request Xoff for that port (and priority).

27.9 Dropped packets

Packets that are dropped will still consume resources while they are waiting for deallocation. This applies even to broken packets, for instance packets with CRC errors.

The packets dropped due to exceeding the Tail-Drop thresholds are counted in the **Ingress Resource Manager Drop** register.



27.10 Reconfiguration

The Xon, Xoff and tail-drop thresholds can be reconfigured at any time. The reserved settings, however, cannot be changed on any source port on which there is traffic. The reserved settings also cannot be changed for any source port that has packets queued. If the reserved settings are changed in these cases the flow control counters will be irrevocably corrupted, necessitating a reset for the core to continue normal operation.

27.11 Debug Features

Each threshold can be forced to trigger using the trip fields of the threshold registers. For tail-drop only drop can be forced this way, but accept can of course be assured by disabling the threshold using the enable field.

For pausing a specific pause state can be forced using the force and pattern fields of the [Port Pause Settings](#) register.



Chapter 28

Egress Port Shaper

The egress port rates are shaped by token buckets configured in the [Port Shaper Rate Configuration](#) registers. While the token bucket level is below the threshold configured in the [Port Shaper Bucket Threshold Configuration](#) register, no new packets are scheduled for the corresponding egress port. Ongoing packets are not affected by the shaping bucket status.

The port shapers are enabled using the [Port Shaper Enable](#) register, and the saturation level of the port shaper buckets is controlled by the [Port Shaper Bucket Capacity Configuration](#) register.

An illustration of a token bucket can be seen in [Figure 20.1](#) (despite what the illustration says the shaper will of course never drop any packets).



Chapter 29

Statistics

Short Name	Register Name
1. rxIf	MAC Interface Counters For RX
3. macBrokenPkt	MAC RX Broken Packets
4. macRxMin	MAC RX Short Packet Drop
4. macRxMax	MAC RX Long Packet Drop
5. spOverflow	SP Overflow Drop
11. ippDrop	Unknown Ingress Drop Empty Mask Drop Ingress Spanning Tree Drop: Listen Ingress Spanning Tree Drop: Learning Ingress Spanning Tree Drop: Blocking L2 Lookup Drop Ingress Packet Filtering Drop Reserved MAC DA Drop Reserved MAC SA Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop Invalid Routing Protocol Drop Expired TTL Drop L3 Lookup Drop IP Checksum Drop Second Tunnel Exit Drop Tunnel Exit Miss Action Drop Tunnel Exit Too Small Packet Modification Drop Learning Packet Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Egress Configurable ACL Drop ARP Decoder Drop RARP Decoder Drop L2 IEEE 1588 Decoder Drop L4 IEEE 1588 Decoder Drop IEEE 802.1X and EAPOL Decoder Drop SCTP Decoder Drop LACP Decoder Drop AH Decoder Drop ESP Decoder Drop DNS Decoder Drop BOOTP and DHCP Decoder Drop CAPWAP Decoder Drop IKE Decoder Drop GRE Decoder Drop NAT Action Table Drop

Short Name	Register Name
	L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop Source Port Default ACL Action Drop
11. smon	SMON Set 0 Packet Counter SMON Set 1 Packet Counter SMON Set 2 Packet Counter SMON Set 3 Packet Counter SMON Set 0 Byte Counter SMON Set 1 Byte Counter SMON Set 2 Byte Counter SMON Set 3 Byte Counter
11. ippAcl	Ingress Configurable ACL Match Counter
11. vrfln	Received Packets on Ingress VRF
11. nextHop	Next Hop Hit Status
11. eppAcl	Egress Configurable ACL Match Counter
11. preEppDrop	Queue Off Drop Egress Spanning Tree Drop MBSC Drop Ingress-Egress Packet Filtering Drop L2 Action Table Per Port Drop
11. ip	IP Unicast Received Counter IP Multicast Received Counter IP Unicast Routed Counter IP Multicast Routed Counter IP Multicast ACL Drop Counter
11. ippDebug	Debug IPP Counter Debug EPP Counter
12. ipmOverflow	IPP PM Drop
13. ippTxPkt	IPP Packet Head Counter IPP Packet Tail Counter
14. eopDrop	IPP Empty Destination Drop
14. mmp	Flow Classification And Metering Drop
15. erm	Egress Resource Manager Drop
16. bmOverflow	Buffer Overflow Drop
16. irm	Ingress Resource Manager Drop
18. pbTxPkt	PB Packet Head Counter PB Packet Tail Counter
19. epppDrop	Unknown Egress Drop Egress Port Disabled Drop Egress Port Filtering Drop Tunnel Exit Too Small Packet Modification To Small Drop
19. vrfOut	Transmitted Packets on Egress VRF
19. nat	Ingress NAT Hit Status Egress NAT Hit Status
21. drain	Drain Port Drop
22. epmOverflow	EPP PM Drop
24. rqOverflow	Re-queue Overflow Drop
24. eppTxPkt	EPP Packet Head Counter EPP Packet Tail Counter
25. psError	PS Error Counter
25. psTxPkt	PS Packet Head Counter PS Packet Tail Counter
28. txlf	MAC Interface Counters For TX

Table 29.1: Sequence of Statistics Counters



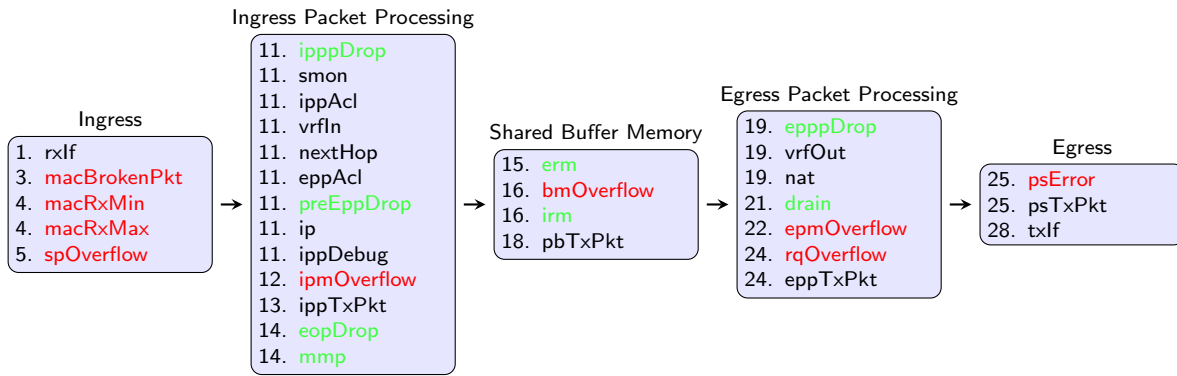


Figure 29.1: Location of Statistics Counters

This core supports full statistics with 64-bit wrap around counters. The statistics is divided into groups depending on the type of statistics and location in the switch. Figure 29.1 gives the location of the counters from ingress to egress, with a sequence number to show their process orders. The counters which are green are for packet drops based on forwarding decisions while the red counters are related to system errors. The details of the counters in Figure 29.1 can be found through Table 29.1.

29.1 Packet Processing Pipeline Drops

During the ingress/egress packet processing, the forwarding algorithm can drop a packet for various reasons. For each type of drop reason at least one drop counter is attached. The counter update is either based on received packets or to-be-transmitted packets.

- **Statistics: IPP Ingress Port Drop.**

Each drop reason has a unique drop identifier (drop ID). The IPP ingress port drop statistics has a counter for each drop ID. In two cases a corresponding drop ID counter can be updated:

1. When a received packet is dropped before any destination port is assigned.
2. When all targeting destination ports are filtered out the **Empty Mask Drop** counter is updated.

- **Statistics: IPP Egress Port Drop.**

This is a per drop ID and per egress port counter located in the ingress processing pipeline. When a packet has obtained one or more destination ports but the following ingress packet process filters out one of the obtained destination ports, a counter is updated for the corresponding egress port with the related drop ID. The **Empty Mask Drop** counter might be updated at the same time if no more destination port is set after the filtering.

- **Statistics: EPP Egress Port Drop.**

This is similar to IPP egress port drop statistics but located in the egress packet processing pipeline. Drops that occur in EPP will cause bubbles on the transmit interface.

29.2 ACL Statistics

When a packet matches an ACL rule as described in Chapter [Classification](#), the result operation can be configured to update a counter. In this case the result operation has a pointer to which counter to update. All the related counters are in Section [Statistics: ACL](#).

29.3 SMON Statistics

There are 4 sets of SMON counters located in the ingress packet processing pipeline, each equipped with one counter per PCP value. The combination of the ingress port number and packet VLAN ID will provide the target SMON set to update through the **SMON Set Search** register. Each SMON set counts both the number of packets and number of bytes as shown in Section [Statistics: SMON](#).



29.4 Routing Statistics

Section [Statistics: Routing](#) has three routing related statistics:

- [Received Packets on Ingress VRF](#). Update when a packet enters a VRF in the ingress processing pipeline.
- [Transmitted Packets on Egress VRF](#). Update when a packet leaves a VRF in the egress processing pipeline.
- [Next Hop Hit Status](#). Update when IPv4/IPv6/MPLS packets hit a next hop entry.

29.5 Ingress Port Receive Statistics

Section [Statistics: IPP Ingress Port Receive](#) lists available statistics for good received packets on a per ingress port basis.

- Good received IP packets
 - [IP Unicast Received Counter](#)
 - [IP Unicast Routed Counter](#)
 - [IP Multicast Received Counter](#)
 - [IP Multicast Routed Counter](#)
 - [IP Multicast ACL Drop Counter](#)

29.6 Packet Datapath Statistics

Section [Statistics: Packet Datapath](#) gives a list of start of packet and end of packet counters in the main blocks of the core. They act as datapath checkpoints and can be helpful in tracing unexpected packet drops or corruptions.

A packet will cross three clock domains on its way through the core:

- RX MAC clock domain.

Packet datapath statistics in the RX MAC clock domain are on the receive edge of the switch, counting received packets as well as illegal packet patterns. Clock crossing synchronizations are applied to these counters in order to share the same configuration bus in the core clock domain. The included counters are:

 1. [MAC Interface Counters For RX](#).
- TX MAC clock domain.

Packet datapath statistics in the TX MAC clock domain are on the transmit edge of the switch, counting transmitted packets as well as protocol errors on the TX interface of the switch. Clock crossing synchronizations are applied to these counters in order to share the same configuration bus in the core clock domain.

 1. [MAC Interface Counters For TX](#).
- Core clock domain.

Packet datapath statistics in the core clock domain are counting in different internal blocks. Each block has a pair of counters for packet heads and tails to identify the pass through of a complete packet. The datapath counting follows the order in [Figure 1.1](#):

 1. [IPP Packet Head Counter](#) and [IPP Packet Tail Counter](#).
 2. [PB Packet Head Counter](#) and [PB Packet Tail Counter](#).
 3. [EPP Packet Head Counter](#) and [EPP Packet Tail Counter](#).
 4. [PS Packet Head Counter](#) and [PS Packet Tail Counter](#).

If a stage has unequal packet head and tail counters while the counters in the previous stages are identical, packets are corrupted in this stage.

29.7 Miscellaneous Statistics

The core is designed to have no silent packet drops and all missing packets on the transmit interface can be found in a dedicated drop counter. Besides the drop counters mentioned above, there are more counters located in all other places where a packet drop might occur. Detailed drop counter list is in [Section Statistics: Misc](#).



29.8 Debug Statistics

Section [Statistics: Debug](#) lists a group of statistics prepared for debug purposes. These counters indicate possible locations when fatal errors occurred inside the core. Typical error events include inaccurate clock frequencies, unacceptable configurations, etc. The switch will try to remain functional after an error state, but a correct behaviour cannot be guaranteed.

29.8.1 Debug Statistics Accuracy

Some of the statistics counters are located in a different clock domain than the configuration bus. The values are therefore transferred through synchronization registers. In order to reduce the hardware cost of these debug counters the synchronization can result in reading incorrect values if readout is done while the counters are incrementing. The counter itself will always have the correct value. It's only the readout that, with a very low probability, can have incorrect value on bits that are toggling.



Chapter 30

Packets To And From The CPU

The CPU port (number 31) has support for two special CPU tags in the packet header. In packets received by the switch on the CPU port, the tag can determine which port the packet shall be sent to. A tag can also be added to packets transmitted by the switch on the CPU port. This allows the software stack to determine where the packet came from and the reason why it was sent to the CPU port.

30.1 Packets From the CPU

Packets sent from the CPU are normally processed as any other packet that enters the switch, so the destination port is determined by the L2 lookup. When the CPU needs to direct a packet to a specific port, bypassing the normal L2 lookup, it is accomplished by adding a protocol header.

Byte Number	Contents of Byte
0-3	[31:0] port bit mask. Bit 0 is port number 0, bit 1 is port number 1 etc. Port 0 is located in bit 0 of byte number 3. The port numbers are physical ports, not link aggregation port numbers. The link aggregation will always be bypassed when sending packets with a From CPU Tag.
4	Bits [2:0] specifies which egress queue the packet shall use. Bit [3] Specifies if the packet shall go out un-modified or modified on the egress ports. If this bit is set to one all ACL actions are bypassed. 0 = Modified. 1 = Unmodified.
5	Bit [0] will set the <i>upd_ts</i> signal on the transmit MAC interface when the packet is transmitted. Bit [1] will set the <i>upd_cf</i> signal on the transmit MAC interface when the packet is transmitted. Bit [2] will set the <i>ts_to_sw</i> signal on the transmit MAC interface when the packet is transmitted.
6-13	PTP Timestamp that will be set on the transmit MAC interface when the packet is transmitted. The lowest numbered byte contains the msb of the timestamp value.
16-18	Reserved. Not used.

Table 30.1: From CPU tag format

The header consists of a specific Ethernet Type (39065) followed by a CPU Tag. The CPU tag has a 4 byte(s)

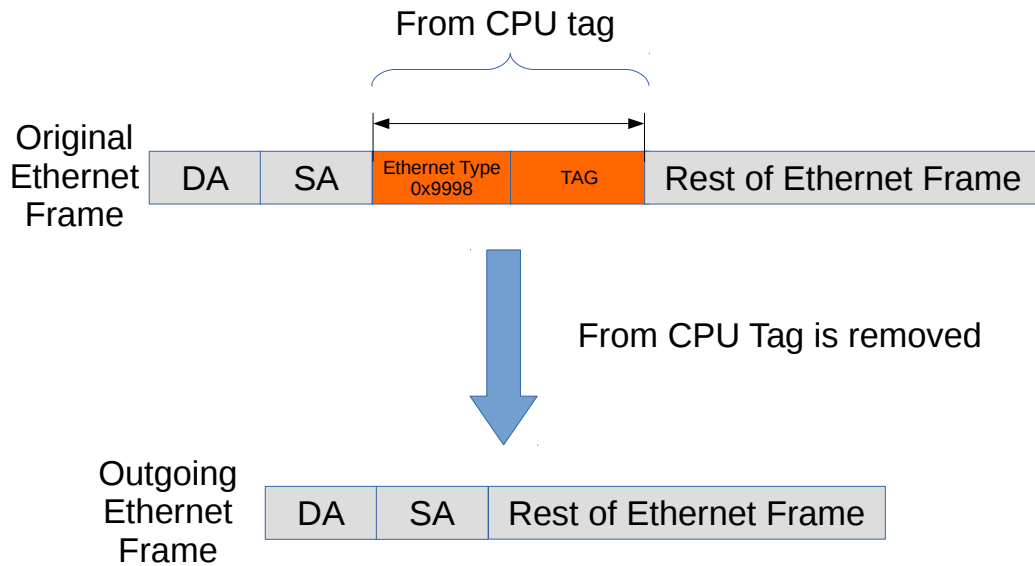


Figure 30.1: Packet from CPU with CPU tag

destination port mask field¹ and 1 byte egress queue field (encoded as specified in table 30.1). The switch core will remove the extra protocol header and send out the packet on the ports requested by the destination port mask in the protocol header. This is shown in the figure 30.1.

The port mask in the CPU Tag field determines which ports the packet shall be sent to. If multiple bits are set in the port mask, the packet is treated as a multicast packet in the resource limiters. The packet will be sent out on all ports with the corresponding bit set.

30.1.1 Identify the From CPU Tag

By default, only packets that are received on the CPU port will be able to support identifying the specific Ethernet type for the from CPU tag. This means that packets with this Ethernet type that are received on other ports of the switch will be treated as unknown and will not enter the packet processing based on the from CPU tag.

If non-CPU ports need to identify the from CPU tag, it can be achieved by the `enableFromCpuTag` from the [Source Port Table](#). Notice the CPU port is not affected by this setting and always decode the from CPU tag.

30.1.2 From CPU Header and Packet Modification and Operations

There are a number of operations which are not carried out when a packet is sent in with the From CPU header. The following lists details this in greater detail what is done and what is not done.

- Link Aggregation is done.
- None of the VLAN operations are carried out.
- Mirroring is done. However with regards to ACL mirroring see below.
- Drops are ignored, example VLAN table , spanning tree / multiple spanning tree drops.
- L2 Lookup result is ignored.
- If the packet hits decoding rules for BPDU, Rapid Stanning Tree, Multiple Spanning tree, or other protocols such as 802.1X-EAPOL AH ARP AVTP DHCP CAPWAP DNS ESP GRE IKE L2 1588 L4 1588 LACP RARP SCTP then the packet will still send a extra copy to the CPU port. This can be disabled by setting the cpu port to zero in the send-to-cpu bitmask in each function.
- Routing is not carried out.
- SMON statistics is performed.
- Basic assignment of MMP is done.

¹The ordering described in 30.1 is the receive/transmit order.



- Meter-Marker-Policer check is done.
- MBSC is bypassed.
- All spanning tree and multiple spanning tree operations are bypassed.
- No learning operation.
- If the From CPU tag has the Modified bit set to one (1) then the following happens:
 - Check Reserved DMAC is bypassed.
 - Check Reserved SMAC is bypassed.
 - ACL operations are not done.
 - ACL statistics are not done.
 - Tunneling are not done (tunnel entry or tunnel exit).
 - SMON statistics is not done.
 - NAT operations are not done.
- If the From CPU tag has the Modified bit set to zero (0) then the following happens:
 - Check Reserved DMAC is done.
 - Check Reserved SMAC is done.
 - ACL operations are done.
 - ACL statistics are done.
 - Tunneling is done (tunnel entry or tunnel exit).
 - SMON statistics is done.
 - NAT operations are done.

30.2 Packets To the CPU

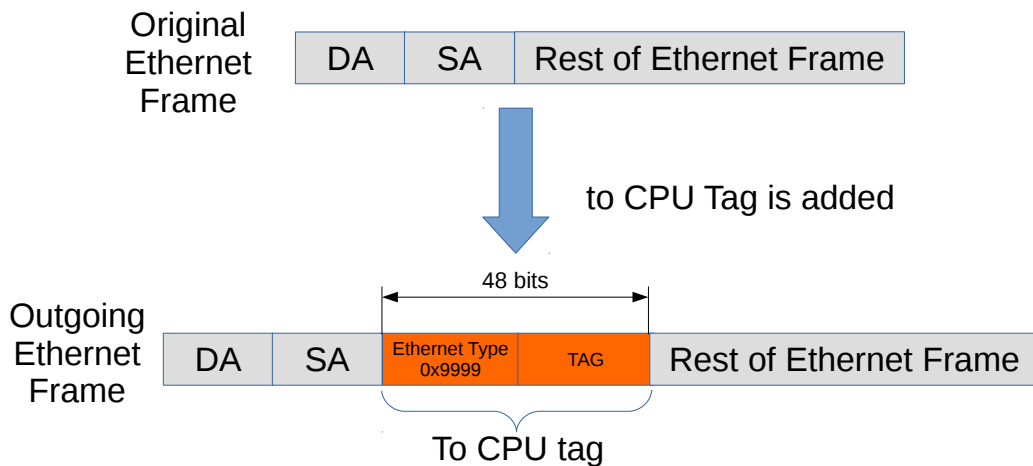


Figure 30.2: Packet to CPU with CPU tag

Packets can also be sent to the CPU port bypassing the normal L2 lookup. By default all packets to the CPU port have an extra protocol header (as shown in Figure 30.2). The header indicates the reason that the packet was sent to the CPU, and the port on which it was received. If the packets shall be the original copy as it came in on a source port or if they shall be the processed version depends first on a register called **Default Packet To CPU Modification**, at some places there also exists extra bits to change this setting.

When packets are sent to the CPU port (number 31 in this core), the packets are tagged with a specific Ethernet Type (type 39321). Figure 30.2 shows the Ethernet type field followed by a tag, and together these constitute the extra protocol header mentioned above. The unmodified incoming packet follows just after this header.

The insertion of the extra protocol header can be disabled by setting the register **Disable CPU tag on CPU Port** to 1.



30.3 To CPU Header format

The following table describes the fields which will be in the toCPU tag. The original bit is set when packets are modified by the egress packet processing, if the modification is the same as the original packet this modification bit will still be set.

Name	Short Name	Field Size	IETF bit index	Description
Ethernet Type	ethType	16	[15:0]	Ethernet Type, 0x9999
Length	length	16	[31:16]	Length of Packet
Packet Type	pktType	6	[37:32]	Packet Type, see table 30.3
IPv4	i	1	[38]	This is a IPv4 Packet. 0 = No 1 = Yes
IPv6	s	1	[39]	This is a IPv6 Packet. 0 = No 1 = Yes
IP Offset	ipo	8	[47:40]	IP Header Offset.
IPv4 length	4l	4	[51:48]	IPv4 Header Length in 4 Octets.
TCP length	tl	4	[55:52]	TCP Header Length in 4 Octets. NOTE: If the packet is a IPv6 and has a segment routing header then this value will be set to zero.
Fragment	f	1	[56]	Fragment Indicator from IPv4 header.
Transmit Type	tt	2	[58:57]	The transmitt type. 0 = Unicast 1 = Multicast 2 = BroadCast 3 = Flooding
Nr Of Vlans	nv	2	[60:59]	The nr of VLANs. 0 = Zero 1 = One 2 = Two 3 = More than two
is_PPpOE	p	1	[61]	PPPoE Header exists in packet.
original	o	1	[62]	Original or modified packet. 0 = Original 1 = Modified
Reserved	c	1	[63]	Reserved.
Outermost VID	outerVid	12	[75:64]	The outermost VLAN ID on the packet
L4 Type	l4t	4	[79:76]	The L4 Type of the packet. 0 = Not known. 1 = Is IPv4 or IPv6 but type is not any L4 type in this list. 2 = UDP 3 = TCP 4 = IGMP 5 = ICMP 6 = ICMPv6 7 = MLD 8..15 - Reserved.
Source Port	srcPort	8	[87:80]	Source Port, bits 4:0 Contains the source port number
Reason	reason	16	[103:88]	Reason Code, Byte 1 is the msb of the reason code. see table 30.4
Meta Data	meta	16	[119:104]	The meta data comes from the forwarding tables. It is setup by software to enable software to determine the reason why a entry was sent to the CPU port.
Reserved	resvd	6	[125:120]	Reserved.
Valid Timestamp	v	1	[126]	If set to one then the Timestamp field is valid.
PTP	p	1	[127]	If set to one then the packet is a PTP packet.

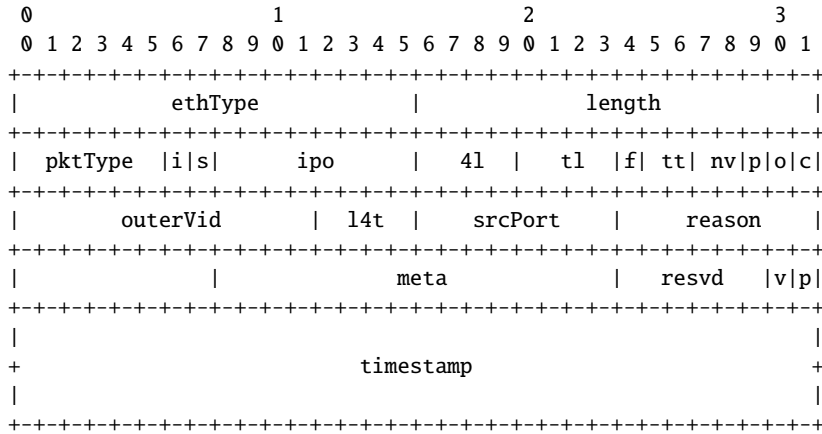


Name	Short Name	Field Size	IETF bit index	Description
Timestamp	timestamp	64	[191:128]	64-bit Timestamp of packet

Table 30.2: To CPU Header

30.3.1 To CPU Header in IETF format

The packets-to-CPU header expressed in a IETF header format below, using the short names from the 30.2 as field names. Bit 0, Byte0 is the first bit and byte which is being transferred out on the CPU port (see appendix B in RFC791):



30.3.2 Packet Type Table

As seen above there is a packet type field and this is determined by the packet decoder information to tell the receiving CPU what type of packet it is.

Packet Type Id	Name	Description
0	ARP	The packet is a ARP packet. Decoding setup and options are available in register ARP Packet Decoder Options
1	RARP	The packet is a RARP packet. Decoding setup and options are available in register RARP Packet Decoder Options
2	LLDP	The packet is a LLDP packet. Decoding setup and options are available in register LLDP Configuration
3	L2_1588	The packet is a IEEE 1588 L2 packet. Decoding setup and options are available in register IEEE 1588 L2 Packet Decoder Options
4	8021X.EAPOL	The packet is a 802.1X or EAPOL packet. Decoding setup and options are available in register IEEE 802.1X and EAPOL Packet Decoder Options
5	MPLS	The packet is a MPLS packet.
6	GRE	The packet is a GRE packet. Decoding setup and options are available in register GRE Packet Decoder Options
7	SCTP	The packet is a SCTP packet. Decoding setup and options are available in register SCTP Packet Decoder Options
8	IGMP	This is a IGMP packet.
9	MLD	This is a MLD packet.
10	ICMP	This is a ICMP packet.
11	LACP	The packet is a LACP packet. Decoding setup and options are available in register LACP Packet Decoder Options
12	AH	The packet is a IPsec AH packet. Decoding setup and options are available in register AH Header Packet Decoder Options
13	ESP	The packet is a IPsec ESP packet. Decoding setup and options are available in register ESP Header Packet Decoder Options
14	DNS	The packet is a DNS packet. Decoding setup and options are available in register DNS Packet Decoder Options
15	BOOTP_DHCP	The packet is a BOOTP or DHCP packet. Decoding setup and options are available in register BOOTP and DHCP Packet Decoder Options
16	L4_1588	The packet is a IEEE 1588 L4 packet. Decoding setup and options are available in register IEEE 1588 L4 Packet Decoder Options
17	CAPWAP	The packet is a CAPWAP packet. Decoding setup and options are available in register CAPWAP Packet Decoder Options
18	IKE	The packet is a IPsec IKE packet. Decoding setup and options are available in register IKE Packet Decoder Options
19	BPDU	The packet is a BPDU packet.
20	UDP_LARGER_THAN_1024	The packet is a UDP packet where destination port \geq 1024.
21	TCP_LARGER_THAN_1024	The packet is a TCP packet where destination port \geq 1024.
22	CANCEL_TE	A tunnel exit was performed but then the original packet shall be sent to the CPU. Hence the inner packet type information is lost. CPU needs to determine packet type by itselfes.
63	default	When all above identifications fails.

Table 30.3: Packet Type Table



30.3.3 Reason Table

The reason codes why a packet was sent to the CPU. Reason code 0 means that the packet was switches or routed and the CPU port was part of the normal forwardings destination ports. If a packet can be directed to the CPU port with multiple reasons, the first hit in the check list below will give the reason code to the egress packet header.

Reason	Description
0	The MAC table, L2 MC table, ACL send to port action, MPLS table, the from-CPU-TAG contained the CPU port or routing tables sent the packet to the CPU port.
1	The packet decoder requires more than one cell.
2	This is a BPDU / RSTP frame.
3	The Unique MAC address to the CPU was hit.
4 + HitIndex	The Source MAC range sent the packet to the CPU..Index to rule.
8 + HitIndex	The Destination MAC range sent the packet to the CPU..Index to rule.
12 + HitIndex	The source port default ACL action sent the packet to the CPU..Index to source port which sent the packet in.
44 + HitIndex	The TCAM in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
60 + HitIndex	The small table in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
316 + HitIndex	The large table in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
2364 + HitIndex	The TCAM in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
2372 + HitIndex	The small table in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
2380 + HitIndex	The large table in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
2508 + HitIndex	The TCAM in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
2532 + HitIndex	The small table in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
2532 + HitIndex	The large table in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
2532 + HitIndex	The TCAM in the configurable ingress ACL engine 3 sent the packet to the CPU..Index to rule.
2548 + HitIndex	The small table in the configurable ingress ACL engine 3 sent the packet to the CPU..Index to rule.
2548 + HitIndex	The large table in the configurable ingress ACL engine 3 sent the packet to the CPU..Index to rule.
2548 + HitIndex	The small table in the configurable egress ACL engine 0 sent the packet to the CPU..Index to rule.
2804 + HitIndex	The large table in the configurable egress ACL engine 0 sent the packet to the CPU..Index to rule.
3828 + HitIndex	The TCAM in the configurable egress ACL engine 0 sent the packet to the CPU..Index to rule.
3844 + HitIndex	The small table in the configurable egress ACL engine 1 sent the packet to the CPU..Index to rule.
3844 + HitIndex	The large table in the configurable egress ACL engine 1 sent the packet to the CPU..Index to rule.
3844 + HitIndex	The TCAM in the configurable egress ACL engine 1 sent the packet to the CPU..Index to rule.
3860	This is an L2 1588 frame.
3861	This is an L4 1588 frame.
3862	This is an ARP frame.
3863	This is an RARP frame.
3864	This is an LLDP frame.
3865	This is an 802.1X EAPOL frame.



Reason	Description
3866	This is an GRE frame.
3867	This is an SCTP frame.
3868	This is an LCAP frame.
3869	This is an AH frame.
3870	This is an ESP frame.
3871	This is an DNS frame.
3872	This is a BOOTP or DHCP frame.
3873	This is an CAPWAP frame.
3874	This is an IKE frame.
3875	The IP TTL field was expired in the packet.
3876	The router ports check about which IPv4/IPv6/MPLS packets was allowed in the router failed.
3877	The default routes send2cpu bit was set.
3878	The IP length exceeded the MTU setup.
3879	The entry in the Next Hop Table is invalid.
3880	The entry in Next Hop Packet Modifications pointed to from the Next Hop Table is invalid.
3881	The next hop entry had a send2cpu bit set.
3882	The IPv4 header size field was not equal to five.
3883	IPv4/IPv6 multicast was detected and redirected to CPU.
3884	The IPv6 routing header contained an unrecognized routing type
3885	The IPv6 segment routing header contained an unexpected routing header length
3886	The IPv6 segment routing header contained TLV field
3887	The maxium number of MPLS tags was detected in a packet.
3888	Packet matched an L2 Multicast Reserved Address
3889	Packet was suppose to do a two tunnel exits.
3890	The first tunnel exit lookup was a hit but the second tunnel exit lookup was a miss and the source port table said this packet shall then be sent to the CPU.
3891	Tunnel Exit result said send to CPU.
3892	After Tunnel entry the MTU was too small for this packet.
3893	The NAT Action Table has sent the packet to the CPU with this code.
3894	The NAT Action Table has sent the packet to the CPU wit this code.
3895	The L2 Action Table has determined that this packet shall be sent to the CPU.
3896	The SNAP LLC Decoding Options has determined that this packet shall be sent to the CPU.

Table 30.4: Reason for packet sent to CPU

The possible reasons are listed in Table 30.4.

1. Hit in the [Reserved Source MAC Address Range](#) with a [sendToCpu](#) action.
2. Hit in the [Reserved Destination MAC Address Range](#) with a [sendToCpu](#) action.
3. Hit in the [L2 Reserved Multicast Address Base](#) with [sendToCpuMask](#) enabled for the corresponding source port.
4. Hit in the [LLDP Configuration](#).
5. Hit in the [Send to CPU](#) register.
 - Notice that when [uniqueCpuMac](#) is enabled then unicast packet will not be switched to the CPU port. Instead packets from any source port with MAC DA equal to [cpuMacAddr](#) will be sent to the CPU. Other mechanism for sending to the CPU port are not affected (e.g. ACL's).
6. Hit in the [Configurable ACL Engine](#) with a [sendToCpu](#) action.

30.3.4 Reason Code Operations

If the packet is sent to the CPU port with a non-zero reason code, the [CPU Reason Code Operation](#) register allows extra actions based on the corresponding reason code. The reason code number is checked in 16 given



ranges from the first entry to the last entry. If the reason code has multiple hits, different operations can be done in parallel and the same operation in the latter one will override the previous hit.

- **mutableCpu** allows the packets that are sent to the CPU port use another port number for the CPU port. In this case the to CPU tag is always inserted to the packet and will not be controlled by **Disable CPU tag on CPU Port**.
- **forceQueue** alters the egress queue of the packets that are sent to the CPU port.



Chapter 31

Core Interface Description

This chapter describes the interfaces to the core. An *input* is an input to the core, and an *output* is a signal driven by the core. In analogy *reception* refers to packets to the core and *transmission* means packets from the core.

31.1 Clock, Reset and Initialization interface

There is a core clock, mac clock signals for the packet interfaces, a global reset signal, mac reset signals for the packet interfaces, and a *doing-init* output (indicating when the core is in initialization and thus not ready to receive packets).

When the global reset, *rstn*, is asserted all packets buffered in the switch will be dropped, the learning and aging engines and all statistics counters will be reset to the initial status. Reset can be pulled at any time, but any ongoing transmit packets will be immediately interrupted and no end of packet signal will be given.

The packet interface resets cannot be used independently. If one reset is asserted, all resets (including the core reset) have to be asserted before any reset can be released.¹

¹Thus the packet interface resets cannot be used to empty a specific packet interface. To do that, follow the procedure in Section 21.8, while making sure that the packet interface halt is kept low.

Signal Name	Size	In Out	Description
clk	1	In	Core clock. For 3200 Gbit/s wire-speed throughput use a core clock frequency of 1200 MHz
rstn	1	In	Global asynchronous reset (active low)
clk_mac_rx_N	1	In	Clock for the RX packet interface for port N .
rstn_mac_rx_N	1	In	Asynchronous reset (active low) for the RX packet interface for port N
clk_mac_tx_N	1	In	Clock for the TX packet interface for port N .
rstn_mac_tx_N	1	In	Asynchronous reset (active low) for the TX packet interface for port N
assert_reset	1	Out	Signal indicating that the core has experienced an unrecoverable error, and should be reset.
consistency_check	1	In	When pulled high internal checks will be made. This is a simulation-only port, it shall be tied low in hardware.
idle	1	Out	Indicates when the packet processing pipelines are empty.
doing_init	1	Out	Indicates that the core is in initialization. The operation of the core is undefined if packets are injected on the rx-interfaces when the core is in initialization

Table 31.1: Clock and Reset interfaces

Core Initialization

Before packets are sent to the core it needs to be initialized. The initialization is initiated when reset is released. Reset activation is asynchronous to any clock. The reset should be kept low at least one cycle of the slowest clock. Releasing reset must be done synchronously with respect to all clocks. During initialization *doing_init* is kept high. See Figure 31.1. The length of the initialization is dependent on the depth of the deepest initialized memory.

During initialization no activity is expected on the configuration interface or on the packet RX interfaces, and the operation of the core is undefined if any such activity occurs.

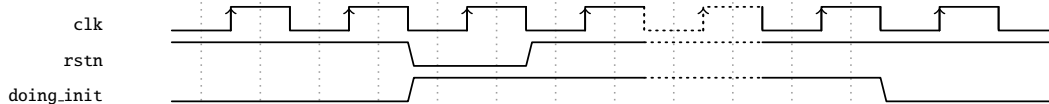


Figure 31.1: Core Initialization

31.1.1 Assert Reset

The *assert_reset* signal will go high, and stay high, if the core experiences an unrecoverable error. The behaviour of the core when *assert_reset* is high is undefined, and the only way to get back to normal operation is to reset the core.

The configuration bus will most likely still work when *assert_reset* is high, but to figure out what went wrong you will probably need to use the debug interface.

31.2 Packet Interface

There are 32 packet interfaces, or ports for short, each divided into a reception part and a transmission part. The ports are numbered from 0 to 31.



Pin	Size	Direction	Description
<code>idata_sp_N</code>	128	In	Packet data.
<code>invalid_bytes_sp_N</code>	5	In	Indicates the number of valid data bytes. For all transactions where <i>last</i> is not high, this shall be equal to the data width in bytes.
<code>ifirst_sp_N</code>	1	In	Start-of-packet flag.
<code>ilast_sp_N</code>	1	In	End-of-packet flag. The <i>last</i> field is also used to signal broken packets. For a correctly transmitted packet <i>last</i> is asserted for the last data transaction of the packet. If <i>last</i> is set high when <i>valid_bytes</i> is zero, the packet is marked as broken, and will be dropped by the core.
<code>invalid_ts_sp_N</code>	1	In	Validates the presence of the timestamp value. Valid when <i>last</i> is set.
<code>its_sp_N</code>	64	In	PTP Timestamp value. Only available when <i>last</i> is set.

Table 31.2: Packet RX interface. **N** is the ingress interface number.

Each direction of a packet interface consists of *first*, *last*, *valid_bytes*, and *data* fields. The transmit direction has an additional *halt* signal to allow the receiving end to moderate the data rate transmitted from the core. There is also a *ptp* flag on the transmit side that is only valid for the first cell. It is copied from the PTP bit in the CPU header, and is thus always zero for all ports except the CPU port.

Packet data is presented in order, i.e. the most recent byte is the, so far, highest numbered byte in the packet. The first valid byte on the bus is byte 0, and all bytes are valid up to the number indicated in *valid_bytes*. Unless the *last* flag is set all bytes or no bytes must be valid.

Sending and Receiving packets

Data transmission, either to or from the core, begins with a transaction where the *first* field is high and the *valid_bytes* field is non-zero, and ends with a data transmission where the *last* field is high. Idle transactions—where *valid_bytes*, *first* and *last* are all zero—are allowed at any time, but unless halted there will be no idle transactions on the transmission interfaces other than between packets.

By default, the core has a short packet size limit of 60 bytes. All shorter packets will be dropped. This assumes that the receiving MAC removes the FCS before sending the packet to the core.

Jumbo packets

The maximum packet length that this core can cope with is 32738 bytes. If this length was allowed to be exceeded either on the ingress or the egress it would corrupt the internal counters.

It should be noted that it is not guaranteed that a packet of that length will always be able to pass through the switch, even if the destination queue is not congested. Depending on the Egress Resource Management settings, and/or the congestion status of other ports, there may not be enough free cells in the packet buffer to store such a large packet. But the switch core will, when properly configured and reasonably uncongested, be able to switch 32738-byte packets.

Longest Packet for No-Overlap Mesh

The longest packet that can pass a no-overlap mesh test is highly dependent on the ERM settings. But with the default settings you can expect to pass a no-overlap mesh test with 7462-byte packets.

Inter-frame gap

For small packets it is possible to feed the switch with more packets than it can handle. This will cause the SP to overflow, and packets to be dropped. To avoid packet drops an inter-frame gap (IFG) of at least 192 bits is



Pin	Size	Direction	Description
odata_ps_ N	128	Out	Packet data.
ovaild.bytes_ps_ N	5	Out	Indicates the number of valid data bytes. For all transactions where <i>last</i> is not high, this is equal to the data width in bytes.
ofirst_ps_ N	1	Out	Start-of-packet flag.
olast_ps_ N	1	Out	End-of-packet flag. For a correctly transmitted packet <i>last</i> is asserted for the last data transaction of the packet. If <i>last</i> is set high when <i>valid.bytes</i> is zero, the packet shall be dropped or terminated with an error by the MAC.
oupd.ts_ps_ N	1	Out	The TX MAC should update the PTP Timestamp field in the current packet. Only valid when <i>first</i> is set.
oupd.cf_ps_ N	1	Out	The TX MAC should update the PTP correction field in the current packet. Only valid when <i>first</i> is set.
ots_to_sw_ps_ N	1	Out	The TX MAC should take a timestamp of the current packet and send to software.
ots_ps_ N	64	Out	PTP Timestamp value. Only valid when <i>first</i> is set.
oudp4_ps_ N	1	Out	The packet is an IPv4/UDP packet. Only valid when <i>first</i> is set.
oudp6_ps_ N	1	Out	The packet is an IPv6/UDP packet.
oudp.csum_ps_ N	9	Out	Byte position of the start of the UDP checksum field. Only valid when <i>first</i> is set.
ots_pos_ps_ N	9	Out	Byte position of the start of the Timestamp field in a PTP packet. Only valid when <i>first</i> is set.
oudp.corr_ps_ N	15	Out	Byte position of the start of the UDP checksum correction position in a PTP packet. Only valid when <i>first</i> is set.
tx_halt_ps_ N	1	In	Interrupt the data transmission from egress port N .

Table 31.3: Packet TX interface. **N** is the egress interface number.

needed between each packet. There is a small fifo in the SP, so a single smaller IFG is fine, but it needs to average at or above the minimum IFG over a window of a few packets.

On the output from the switch packets will be sent back to back, without IFG, and it is up to the receiver to halt the transmission using the *halt* interface to prevent overflows.

Broken packets

A packet ending with *last* set high and *valid.bytes* set to zero is considered a broken packet. Broken packets received by the core will never be output on the egress ports, but will be dropped at the earliest convenience. So any broken packets output from the switch are packet that were somehow corrupted in the core. There are no benign cases where this happens. Depending on the packet length a broken packet input to the core will be dropped either before or after ingress packet processing. Broken packets larger than a cell will pass through the packet processing pipeline and then been dropped, while packets shorter than a cell will be filtered out before the packet processing pipeline.

All broken packets are counted in the [MAC RX Broken Packets](#).



Halts

Data transmission from the transmit interface of the core can be interrupted individually per egress port using the *halt* signals. A high halt signal on the positive edge of mac clock, will cause the transmission to be idle for the corresponding egress port on the same positive edge. Data transmission will resume on the next positive edge of mac clock when halt is again low.

Byte Order

We define the packet byte order by the first transmitted/received byte on the wire labeled byte 0, as in IEEE 802.3. On a packet interface wider than 8 bits the packets byte 0 is placed on the bits data[7:0] followed by byte 1 on bits data[15:8] and so on.

The *valid_bytes* indicates how many of the bytes of the data field that holds valid packet data. From the start of a packet this must always be all bytes on the bus up till the last transfer. At the end of the packet on the last bus transfer the *valid_bytes* can indicate less than the full bus width. In this case the byte order is still the same as previous transfers. For example when *valid_bytes* is 1 the last byte of the packet is placed on bits [7:0] and with *valid_bytes* of 2 the last byte of the packet is placed on bits [15:8] and the second to last is on bits [7:0].

31.3 Configuration Interface

The CPU-accessible registers and tables in the core are accessed using the configuration interface.

Each transaction on the configuration interface consists of a request to the core and a resulting reply from the core.

The pins for the configuration interface are listed in Table 31.4 below.

Pin	Size	Direction	Description
apb_paddr	34	In	Address. This is the APB address bus. The highest address bit (33) on the APB bus is not a normal address bit and is referred to as the Accumulator Bit. This is described further in section 32.
apb_psel	1	In	Select.
apb_penable	1	In	Enable.
apb_pwrite	1	In	Direction. This signal indicates an APB write access when HIGH and an APB read access when LOW.
apb_pwdata	64	In	Write data.
apb_pready	1	Out	Ready. The slave uses this signal to extend an APB transfer.
apb_prdata	64	Out	Read Data.
apb_pslverr	1	Out	Error. This signal indicates a transfer failure.

Table 31.4: The APB interface signals

The *paddr* is a byte address, however the core only supports accessing complete 64-bit words. The lowest address bits, which addresses the byte within a bus word, will always be discarded. The register addresses described in this document always refer to word addresses, not byte addresses.

The core has a varying access latency and therefore an APB master should use *pready*.

The *pslverr* signal is set when a transaction is aborted due to an internal timeout. This can occur if the core clock is lower than required and there is a high traffic rate. It will also occur if the address is outside of any defined register.

For a detailed description of the APB interface see the AMBA APB Protocol Specification Version 2.0, available at developer.arm.com



31.4 Interrupt Interface

The interrupt interface is a vector of interrupt flags. When an interrupt occurs it will become a one cycle long pulse on an interrupt flag. I.e. an interrupt has occurred whenever an interrupt flag is high on the positive edge of the clock.

There is no interrupt mask nor any interrupt status register to be cleared.

Pin	Function	Size	Direction	Description
interrupts[0]	ldf_level	1	Out	Raised when the level of Learning Data FIFO is not below Learning Data FIFO High Watermark Level and receives a push request.
interrupts[1]	ldf_full	1	Out	Raised when Learning Data FIFO is full but still receives a push request.
interrupts[2]	adf_level	1	Out	Raised when the level of Aging Data FIFO is not below Aging Data FIFO High Watermark Level and receives a push request.
interrupts[3]	adf_full	1	Out	Raised when Aging Data FIFO is full but still receives a push request.
interrupts[4]	hdf_level	1	Out	Raised when the level of Hit Update Data FIFO is not below Hit Update Data FIFO High Watermark Level and receives a push request.
interrupts[5]	hdf_full	1	Out	Raised when Hit Update Data FIFO is full but still receives a push request.
interrupts[6]	hash_aging	1	Out	Indicating an aging process on the L2 hash tables is done by either the hardware aging or the software aging. When Aging Engine is operating with Software Aging Enable turned on, it will be silent till Software Aging Start Latch is pulled to one and trigger an aging process immediately. L2 Aging Table entries are evenly divided to 4 buckets while the aging process loops through them in parallel. Each bucket is checked from the first entry to the last entry and in the end raise a corresponding interrupt.
interrupts[7]	cam_aging	1	Out	Indicating an aging process on the L2 collision table is done by either the hardware aging or the software aging. When Software Aging Enable is turned on and Software Aging Start Latch is pulled to one, an aging process will loop through all L2 Aging Collision Table entries immediately from first to last. After it is done this interrupt will be raised.
interrupts[8:31]	reserved	24	Out	Reserved.

Table 31.5: Interrupt interface



31.5 Pause Interfaces

There are separate pause interfaces for sending status information from the switch to the MAC, *opfc_status*, and from the MAC to the switch, *iext_pause*. Note that these interfaces are in the core clock domain, so they have to be synchronized to the MAC clock if connected to the MAC. However the interfaces can be thought of as quasi static. With properly configured pausing thresholds there will never be a short high pulse (due to hysteresis), and losing a short low pulse due to synchronization will create no problems.

31.5.1 PFC Status

The *ipfc_status* interface is used to transfer pause status from the switch resource manager to the MAC, so the MAC can generate pause frames.

The switch will merely indicate its current pause status, it is up to the MAC to generate the necessary pause frames to keep the far end switch in the desired pausing state.

In port mode the status interface will send 0 in unpaused state, and 0xff in paused state.

31.5.2 External Pause

The *iext_pause* interface is used to transfer PFC pause status received by the MAC to the switch egress scheduler. When the status is XOFF the switch egress scheduler will not send any new packets. Ongoing packets are not affected. There is one *iext_pause* interface for each packet interface. Even when priority pause is not enabled the external pause interface is still operating per priority.

Pin	Direction	Size	Description
<i>iext_pause_N</i>	In	8	Xoff=1, Xon=0 status for each PFC channel (0..7)
<i>opfc_status_N</i> [7:0]	Out	8	Xoff=1, Xon=0 status for each PFC channel (0..7)

Table 31.6: The PFC status and External Pause interfaces, where **N** is the packet interface number

31.6 Debug Read Interface

The debug read interface outputs internal debug signals on the *debug_read_data* port. Which signals to observe is selected with the *debug_read_select* port. The mapping between select value and debug signal is described in Table 31.8. Both these signals are pipelined.

Pin	Direction	Size	Description
<i>debug_read_select</i>	In	9	Selects the signal to monitor. See Table 31.8.
<i>debug_read_data</i>	In	32	The debug output data.

Table 31.7: The Debug Read interface

id	instance	signal
0	pa_top.switch.mactop	constant-0
1	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck31 {5'valid_bytes, 1'halt, 1'last, 1'first}
2	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck30 {5'valid_bytes, 1'halt, 1'last, 1'first}
3	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck29 {5'valid_bytes, 1'halt, 1'last, 1'first}
4	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck28 {5'valid_bytes, 1'halt, 1'last, 1'first}
5	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck27 {5'valid_bytes, 1'halt, 1'last, 1'first}
6	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck26 {5'valid_bytes, 1'halt, 1'last, 1'first}
7	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck25 {5'valid_bytes, 1'halt, 1'last, 1'first}
8	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck24 {5'valid_bytes, 1'halt, 1'last, 1'first}
9	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck23 {5'valid_bytes, 1'halt, 1'last, 1'first}
10	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck22 {5'valid_bytes, 1'halt, 1'last, 1'first}
11	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck21 {5'valid_bytes, 1'halt, 1'last, 1'first}
12	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck20 {5'valid_bytes, 1'halt, 1'last, 1'first}
13	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck19 {5'valid_bytes, 1'halt, 1'last, 1'first}
14	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck18 {5'valid_bytes, 1'halt, 1'last, 1'first}
15	—"	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck17 {5'valid_bytes, 1'halt, 1'last, 1'first}



id	instance	signal
96	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
97	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
98	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
99	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
100	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
101	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
102	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
103	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
104	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
105	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
106	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
107	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
108	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
109	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
110	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
111	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
112	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
113	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
114	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
115	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
116	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
117	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
118	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
119	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
120	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
121	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
122	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
123	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
124	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
125	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
126	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
127	—"	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
128	—"	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
129	—"	constant-129
130	pa_top.switch.ipp0	constant-130
131	—"	ipp_ipkt_bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
132	—"	ipp_opkt_bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
133	—"	pass_da_0
134	—"	pass_da_1
135	—"	dut_ilpp_iDropper_dbg_drop
136	—"	dut_ilpp_iDropper_dbg_ifirst
137	—"	dut_ilpp_iDropper_dbg_ilast
138	—"	pass_sa_0
139	—"	pass_sa_1
140	—"	constant-140
141	pa_top.switch.ipp0.pm	constant-141
142	—"	pm_fifo_overflow
143	—"	dut_dbg_fifo_full
144	—"	halt_from_pm
145	—"	dut_iFifo_a_debug_in
146	—"	dut_iFifo_a_debug_out
147	—"	constant-147
148	pa_top.switch.ipp1	constant-148
149	—"	ipp_ipkt_bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
150	—"	ipp_opkt_bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
151	—"	pass_da_0
152	—"	pass_da_1
153	—"	dut_ilpp_iDropper_dbg_drop
154	—"	dut_ilpp_iDropper_dbg_ifirst
155	—"	dut_ilpp_iDropper_dbg_ilast
156	—"	pass_sa_0
157	—"	pass_sa_1
158	—"	constant-158
159	pa_top.switch.ipp1.pm	constant-159
160	—"	pm_fifo_overflow
161	—"	dut_dbg_fifo_full
162	—"	halt_from_pm
163	—"	dut_iFifo_a_debug_in
164	—"	dut_iFifo_a_debug_out
165	—"	constant-165
166	pa_top.switch.ipp2	constant-166
167	—"	ipp_ipkt_bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
168	—"	ipp_opkt_bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
169	—"	pass_da_0
170	—"	pass_da_1
171	—"	dut_ilpp_iDropper_dbg_drop
172	—"	dut_ilpp_iDropper_dbg_ifirst
173	—"	dut_ilpp_iDropper_dbg_ilast
174	—"	pass_sa_0
175	—"	pass_sa_1

id	instance	signal
176	—"	constant-176
177	pa_top.switch.ipp2.pm	constant-177
178	—"	pm.fifo.overflow
179	—"	dut_dbg_fifo_full
180	—"	halt_from_pm
181	—"	dut.iFifo.debug_in
182	—"	dut.iFifo.debug_out
183	—"	constant-183
184	pa_top.switch.ipp3	constant-184
185	—"	ipp_ipkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
186	—"	ipp_opkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
187	—"	pass_da_0
188	—"	pass_da_1
189	—"	dut.ilpp.iDropper_dbg_drop
190	—"	dut.ilpp.iDropper_dbg_ifirst
191	—"	dut.ilpp.iDropper_dbg_ilast
192	—"	pass_sa_0
193	—"	pass_sa_1
194	—"	constant-194
195	pa_top.switch.ipp3.pm	constant-195
196	—"	pm.fifo.overflow
197	—"	dut_dbg_fifo_full
198	—"	halt_from_pm
199	—"	dut.iFifo.debug_in
200	—"	dut.iFifo.debug_out
201	—"	constant-201
202	pa_top.switch.sp0	constant-202
203	—"	dut.iSpbridge.assert_reset_sp_bridge
204	—"	dut.iSpbridge.assert_reset_sp_bridge
205	—"	dut.iSpbridge.assert_reset_sp_bridge
206	—"	dut.iSpbridge.assert_reset_sp_bridge
207	—"	dut.iSpbridge.assert_reset_sp_bridge
208	—"	dut.iSpbridge.assert_reset_sp_bridge
209	—"	dut.iSpbridge.assert_reset_sp_bridge
210	—"	dut.iSpbridge.assert_reset_sp_bridge
211	—"	constant-211
212	pa_top.switch.sp1	constant-212
213	—"	dut.iSpbridge.assert_reset_sp_bridge
214	—"	dut.iSpbridge.assert_reset_sp_bridge
215	—"	dut.iSpbridge.assert_reset_sp_bridge
216	—"	dut.iSpbridge.assert_reset_sp_bridge
217	—"	dut.iSpbridge.assert_reset_sp_bridge
218	—"	dut.iSpbridge.assert_reset_sp_bridge
219	—"	dut.iSpbridge.assert_reset_sp_bridge
220	—"	dut.iSpbridge.assert_reset_sp_bridge
221	—"	constant-221
222	pa_top.switch.sp2	constant-222
223	—"	dut.iSpbridge.assert_reset_sp_bridge
224	—"	dut.iSpbridge.assert_reset_sp_bridge
225	—"	dut.iSpbridge.assert_reset_sp_bridge
226	—"	dut.iSpbridge.assert_reset_sp_bridge
227	—"	dut.iSpbridge.assert_reset_sp_bridge
228	—"	dut.iSpbridge.assert_reset_sp_bridge
229	—"	dut.iSpbridge.assert_reset_sp_bridge
230	—"	dut.iSpbridge.assert_reset_sp_bridge
231	—"	constant-231
232	pa_top.switch.sp3	constant-232
233	—"	dut.iSpbridge.assert_reset_sp_bridge
234	—"	dut.iSpbridge.assert_reset_sp_bridge
235	—"	dut.iSpbridge.assert_reset_sp_bridge
236	—"	dut.iSpbridge.assert_reset_sp_bridge
237	—"	dut.iSpbridge.assert_reset_sp_bridge
238	—"	dut.iSpbridge.assert_reset_sp_bridge
239	—"	dut.iSpbridge.assert_reset_sp_bridge
240	—"	dut.iSpbridge.assert_reset_sp_bridge
241	—"	constant-241
242	pa_top.switch.pb0	constant-242
243	—"	dut.iPbu.debug_refc_inc
244	—"	dut.iPbu.debug_port_sch
245	—"	dut.iPbu.dmux_wrr
246	—"	dut.iPbu.debug_qenext
247	—"	dut.iPbu.assert_qediff
248	—"	dut.iPbu.assert_reque_sp
249	—"	Mask of currently receiving packets that have been broken due to BM full
250	—"	dut.iPbu.follow_pfc_accept
251	—"	dut.iPbu.iAssertpacket_3.assert_out
252	—"	pa_top.switch.pb0.iAssertpacket3 {9'valid_bytes, 3'port, 1'last, 1'first}
253	—"	dut.iPbu.iAssertpacket_2.assert_out
254	—"	pa_top.switch.pb0.iAssertpacket2 {9'valid_bytes, 3'port, 1'last, 1'first}
255	—"	dut.iPbu.iAssertpacket_1.assert_out

id	instance	signal
256	—"	pa.top.switch.pb0.iAssertpacket1 {9'valid_bytes, 3'port, 1'last, 1'first}
257	—"	dut.iPbu.iAssertpacket_0.assert_out
258	—"	pa.top.switch.pb0.iAssertpacket0 {9'valid_bytes, 3'port, 1'last, 1'first}
259	—"	dut.iPbu.iPortshaper.iBuckets.reg_stat
260	—"	dut.iPbu.zPassdbgqeread_3.o
261	—"	dut.iPbu.zPassdbgqeread_2.o
262	—"	dut.iPbu.zPassdbgqeread_1.o
263	—"	dut.iPbu.zPassdbgqeread_0.o
264	—"	dut.iPbu.iRequeue.iReFifo_7.iF_iFifos.zFcnt_pop_empty
265	—"	dut.iPbu.iRequeue.iReFifo_7.iF_iFifos.zFcnt_push_full
266	—"	dut.iPbu.iRequeue.iReFifo_6.iF_iFifos.zFcnt_pop_empty
267	—"	dut.iPbu.iRequeue.iReFifo_6.iF_iFifos.zFcnt_push_full
268	—"	dut.iPbu.iRequeue.iReFifo_5.iF_iFifos.zFcnt_pop_empty
269	—"	dut.iPbu.iRequeue.iReFifo_5.iF_iFifos.zFcnt_push_full
270	—"	dut.iPbu.iRequeue.iReFifo_4.iF_iFifos.zFcnt_pop_empty
271	—"	dut.iPbu.iRequeue.iReFifo_4.iF_iFifos.zFcnt_push_full
272	—"	dut.iPbu.iRequeue.iReFifo_3.iF_iFifos.zFcnt_pop_empty
273	—"	dut.iPbu.iRequeue.iReFifo_3.iF_iFifos.zFcnt_push_full
274	—"	dut.iPbu.iRequeue.iReFifo_2.iF_iFifos.zFcnt_pop_empty
275	—"	dut.iPbu.iRequeue.iReFifo_2.iF_iFifos.zFcnt_push_full
276	—"	dut.iPbu.iRequeue.iReFifo_1.iF_iFifos.zFcnt_pop_empty
277	—"	dut.iPbu.iRequeue.iReFifo_1.iF_iFifos.zFcnt_push_full
278	—"	dut.iPbu.iRequeue.iReFifo_0.iF_iFifos.zFcnt_pop_empty
279	—"	dut.iPbu.iRequeue.iReFifo_0.iF_iFifos.zFcnt_push_full
280	—"	dut.iPbu.iRefc.refc_mem_debug
281	—"	dut.iPbu.zPassqesp.zPasslist_3.o
282	—"	dut.iPbu.zPassqesp.zPasslist_2.o
283	—"	dut.iPbu.zPassqesp.zPasslist_1.o
284	—"	dut.iPbu.zPassqesp.zPasslist_0.o
285	—"	Filter mask for packets dropped by ERM
286	—"	Filter mask for packets dropped by ERM
287	—"	Filter mask for packets dropped by ERM
288	—"	Filter mask for packets dropped by ERM
289	—"	dut.iPbu.debug.pb.drop
290	—"	dut.iPbu.debug.pb.drop
291	—"	dut.iPbu.debug.pb.drop
292	—"	dut.iPbu.debug.pb.drop
293	—"	constant-293
294	pa_top.switch.pb0.erm.dut.iEql	constant-294
295	—"	red_zone
296	—"	constant-296
297	pa_top.switch.pb0.pfc	constant-297
298	—"	dut.debug.sp.above_rsv
299	—"	dut.debug.sp.above_rsv
300	—"	dut.debug.sp.above_rsv
301	—"	dut.debug.sp.above_rsv
302	—"	constant-302
303	pa_top.switch.pb0.qe0	constant-303
304	—"	dut.assert.dfifo
305	—"	dut.assert.firstflag
306	—"	dut.assert.reset_next
307	—"	dut.drop.cnt
308	—"	dut.send.cnt
309	—"	dut.iDfifo.iF_iFifos.zFcnt_pop_empty
310	—"	dut.iDfifo.iF_iFifos.zFcnt_push_full
311	—"	dut.ipkt.fifo_7.debug_in
312	—"	dut.ipkt.fifo_7.debug_out
313	—"	dut.ipkt.fifo_6.debug_in
314	—"	dut.ipkt.fifo_6.debug_out
315	—"	dut.ipkt.fifo_5.debug_in
316	—"	dut.ipkt.fifo_5.debug_out
317	—"	dut.ipkt.fifo_4.debug_in
318	—"	dut.ipkt.fifo_4.debug_out
319	—"	dut.ipkt.fifo_3.debug_in
320	—"	dut.ipkt.fifo_3.debug_out
321	—"	dut.ipkt.fifo_2.debug_in
322	—"	dut.ipkt.fifo_2.debug_out
323	—"	dut.ipkt.fifo_1.debug_in
324	—"	dut.ipkt.fifo_1.debug_out
325	—"	dut.ipkt.fifo_0.debug_in
326	—"	dut.ipkt.fifo_0.debug_out
327	—"	dut.pfifo.level
328	—"	dut.pfifo.level
329	—"	dut.pfifo.level
330	—"	dut.pfifo.level
331	—"	dut.pfifo.level
332	—"	dut.pfifo.level
333	—"	dut.pfifo.level
334	—"	dut.pfifo.level
335	—"	constant-335



id	instance	signal
336	pa_top.switch.pb0.qe1	constant-336
337	—"	dut_assert_dfifo
338	—"	dut_assert_firstflag
339	—"	dut_assert_reset_next
340	—"	dut_drop_cnt
341	—"	dut_send_cnt
342	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
343	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
344	—"	dut_ipkt_fifo_7_debug_in
345	—"	dut_ipkt_fifo_7_debug_out
346	—"	dut_ipkt_fifo_6_debug_in
347	—"	dut_ipkt_fifo_6_debug_out
348	—"	dut_ipkt_fifo_5_debug_in
349	—"	dut_ipkt_fifo_5_debug_out
350	—"	dut_ipkt_fifo_4_debug_in
351	—"	dut_ipkt_fifo_4_debug_out
352	—"	dut_ipkt_fifo_3_debug_in
353	—"	dut_ipkt_fifo_3_debug_out
354	—"	dut_ipkt_fifo_2_debug_in
355	—"	dut_ipkt_fifo_2_debug_out
356	—"	dut_ipkt_fifo_1_debug_in
357	—"	dut_ipkt_fifo_1_debug_out
358	—"	dut_ipkt_fifo_0_debug_in
359	—"	dut_ipkt_fifo_0_debug_out
360	—"	dut_pfifo_level
361	—"	dut_pfifo_level
362	—"	dut_pfifo_level
363	—"	dut_pfifo_level
364	—"	dut_pfifo_level
365	—"	dut_pfifo_level
366	—"	dut_pfifo_level
367	—"	dut_pfifo_level
368	—"	constant-368
369	pa_top.switch.pb0.qe2	constant-369
370	—"	dut_assert_dfifo
371	—"	dut_assert_firstflag
372	—"	dut_assert_reset_next
373	—"	dut_drop_cnt
374	—"	dut_send_cnt
375	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
376	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
377	—"	dut_ipkt_fifo_7_debug_in
378	—"	dut_ipkt_fifo_7_debug_out
379	—"	dut_ipkt_fifo_6_debug_in
380	—"	dut_ipkt_fifo_6_debug_out
381	—"	dut_ipkt_fifo_5_debug_in
382	—"	dut_ipkt_fifo_5_debug_out
383	—"	dut_ipkt_fifo_4_debug_in
384	—"	dut_ipkt_fifo_4_debug_out
385	—"	dut_ipkt_fifo_3_debug_in
386	—"	dut_ipkt_fifo_3_debug_out
387	—"	dut_ipkt_fifo_2_debug_in
388	—"	dut_ipkt_fifo_2_debug_out
389	—"	dut_ipkt_fifo_1_debug_in
390	—"	dut_ipkt_fifo_1_debug_out
391	—"	dut_ipkt_fifo_0_debug_in
392	—"	dut_ipkt_fifo_0_debug_out
393	—"	dut_pfifo_level
394	—"	dut_pfifo_level
395	—"	dut_pfifo_level
396	—"	dut_pfifo_level
397	—"	dut_pfifo_level
398	—"	dut_pfifo_level
399	—"	dut_pfifo_level
400	—"	dut_pfifo_level
401	—"	constant-401
402	pa_top.switch.pb0.qe3	constant-402
403	—"	dut_assert_dfifo
404	—"	dut_assert_firstflag
405	—"	dut_assert_reset_next
406	—"	dut_drop_cnt
407	—"	dut_send_cnt
408	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
409	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
410	—"	dut_ipkt_fifo_7_debug_in
411	—"	dut_ipkt_fifo_7_debug_out
412	—"	dut_ipkt_fifo_6_debug_in
413	—"	dut_ipkt_fifo_6_debug_out
414	—"	dut_ipkt_fifo_5_debug_in
415	—"	dut_ipkt_fifo_5_debug_out



id	instance	signal
416	—"	dut_ipkt_fifo_4_debug_in
417	—"	dut_ipkt_fifo_4_debug_out
418	—"	dut_ipkt_fifo_3_debug_in
419	—"	dut_ipkt_fifo_3_debug_out
420	—"	dut_ipkt_fifo_2_debug_in
421	—"	dut_ipkt_fifo_2_debug_out
422	—"	dut_ipkt_fifo_1_debug_in
423	—"	dut_ipkt_fifo_1_debug_out
424	—"	dut_ipkt_fifo_0_debug_in
425	—"	dut_ipkt_fifo_0_debug_out
426	—"	dut_pfifo_level
427	—"	dut_pfifo_level
428	—"	dut_pfifo_level
429	—"	dut_pfifo_level
430	—"	dut_pfifo_level
431	—"	dut_pfifo_level
432	—"	dut_pfifo_level
433	—"	dut_pfifo_level
434	—"	constant-434
435	pa_top.switch.pb0.wrr	constant-435
436	—"	dut_debug_below
437	—"	dut_zPassdebugbvalpipe_zPasslist_7_o
438	—"	dut_zPassdebugbvalpipe_zPasslist_6_o
439	—"	dut_zPassdebugbvalpipe_zPasslist_5_o
440	—"	dut_zPassdebugbvalpipe_zPasslist_4_o
441	—"	dut_zPassdebugbvalpipe_zPasslist_3_o
442	—"	dut_zPassdebugbvalpipe_zPasslist_2_o
443	—"	dut_zPassdebugbvalpipe_zPasslist_1_o
444	—"	dut_zPassdebugbvalpipe_zPasslist_0_o
445	—"	dut_reg_bval
446	—"	dut_reg_bval
447	—"	dut_reg_bval
448	—"	dut_reg_bval
449	—"	dut_reg_bval
450	—"	dut_reg_bval
451	—"	dut_reg_bval
452	—"	dut_reg_bval
453	—"	dut_reg_bval
454	—"	dut_reg_bval
455	—"	dut_reg_bval
456	—"	dut_reg_bval
457	—"	dut_reg_bval
458	—"	dut_reg_bval
459	—"	dut_reg_bval
460	—"	dut_reg_bval
461	—"	dut_reg_bval
462	—"	dut_reg_bval
463	—"	dut_reg_bval
464	—"	dut_reg_bval
465	—"	dut_reg_bval
466	—"	dut_reg_bval
467	—"	dut_reg_bval
468	—"	dut_reg_bval
469	—"	dut_reg_bval
470	—"	dut_reg_bval
471	—"	dut_reg_bval
472	—"	dut_reg_bval
473	—"	dut_reg_bval
474	—"	dut_reg_bval
475	—"	dut_reg_bval
476	—"	dut_reg_bval
477	—"	dut_reg_bval
478	—"	dut_reg_bval
479	—"	dut_reg_bval
480	—"	dut_reg_bval
481	—"	dut_reg_bval
482	—"	dut_reg_bval
483	—"	dut_reg_bval
484	—"	dut_reg_bval
485	—"	dut_reg_bval
486	—"	dut_reg_bval
487	—"	dut_reg_bval
488	—"	dut_reg_bval
489	—"	dut_reg_bval
490	—"	dut_reg_bval
491	—"	dut_reg_bval
492	—"	dut_reg_bval
493	—"	dut_reg_bval
494	—"	dut_reg_bval
495	—"	dut_reg_bval



id	instance	signal
496	—"	dut.reg_bval
497	—"	dut.reg_bval
498	—"	dut.reg_bval
499	—"	dut.reg_bval
500	—"	dut.reg_bval
501	—"	dut.reg_bval
502	—"	dut.reg_bval
503	—"	dut.reg_bval
504	—"	dut.reg_bval
505	—"	dut.reg_bval
506	—"	dut.reg_bval
507	—"	dut.reg_bval
508	—"	dut.reg_bval
509	—"	dut.reg_rank
510	—"	dut.reg_rank
511	—"	dut.reg_rank
512	—"	dut.reg_rank
513	—"	dut.reg_rank
514	—"	dut.reg_rank
515	—"	dut.reg_rank
516	—"	dut.reg_rank
517	—"	constant-517
518	pa_top.switch.pb0.qshp	constant-518
519	—"	dut.iPrioshaper.reg_stat
520	—"	dut.iQueueshaper.reg_stat
521	—"	constant-521
522	pa_top.switch.pb1	constant-522
523	—"	dut.iPbu.debug_refc_inc
524	—"	dut.iPbu.debug_port_sch
525	—"	dut.iPbu.dmux_wrr
526	—"	dut.iPbu.debug_qenext
527	—"	dut.iPbu.assert_qediff
528	—"	dut.iPbu.assert_reque_sp
529	—"	Mask of currently receiving packets that have been broken due to BM full
530	—"	dut.iPbu.follow_pfc_accept
531	—"	dut.iPbu.iAssertpacket_3.assert_out
532	—"	pa.top.switch.pb1.iAssertpacket3 {9'valid_bytes, 3'port, 1'last, 1'first}
533	—"	dut.iPbu.iAssertpacket_2.assert_out
534	—"	pa.top.switch.pb1.iAssertpacket2 {9'valid_bytes, 3'port, 1'last, 1'first}
535	—"	dut.iPbu.iAssertpacket_1.assert_out
536	—"	pa.top.switch.pb1.iAssertpacket1 {9'valid_bytes, 3'port, 1'last, 1'first}
537	—"	dut.iPbu.iAssertpacket_0.assert_out
538	—"	pa.top.switch.pb1.iAssertpacket0 {9'valid_bytes, 3'port, 1'last, 1'first}
539	—"	dut.iPbu.iPortshaper.iBuckets.reg_stat
540	—"	dut.iPbu.zPassdbgqeread_3.o
541	—"	dut.iPbu.zPassdbgqeread_2.o
542	—"	dut.iPbu.zPassdbgqeread_1.o
543	—"	dut.iPbu.zPassdbgqeread_0.o
544	—"	dut.iPbu.iRequeue.iReFifo_7.iF_iFifos.zFcnt_pop_empty
545	—"	dut.iPbu.iRequeue.iReFifo_7.iF_iFifos.zFcnt_push_full
546	—"	dut.iPbu.iRequeue.iReFifo_6.iF_iFifos.zFcnt_pop_empty
547	—"	dut.iPbu.iRequeue.iReFifo_6.iF_iFifos.zFcnt_push_full
548	—"	dut.iPbu.iRequeue.iReFifo_5.iF_iFifos.zFcnt_pop_empty
549	—"	dut.iPbu.iRequeue.iReFifo_5.iF_iFifos.zFcnt_push_full
550	—"	dut.iPbu.iRequeue.iReFifo_4.iF_iFifos.zFcnt_pop_empty
551	—"	dut.iPbu.iRequeue.iReFifo_4.iF_iFifos.zFcnt_push_full
552	—"	dut.iPbu.iRequeue.iReFifo_3.iF_iFifos.zFcnt_pop_empty
553	—"	dut.iPbu.iRequeue.iReFifo_3.iF_iFifos.zFcnt_push_full
554	—"	dut.iPbu.iRequeue.iReFifo_2.iF_iFifos.zFcnt_pop_empty
555	—"	dut.iPbu.iRequeue.iReFifo_2.iF_iFifos.zFcnt_push_full
556	—"	dut.iPbu.iRequeue.iReFifo_1.iF_iFifos.zFcnt_pop_empty
557	—"	dut.iPbu.iRequeue.iReFifo_1.iF_iFifos.zFcnt_push_full
558	—"	dut.iPbu.iRequeue.iReFifo_0.iF_iFifos.zFcnt_pop_empty
559	—"	dut.iPbu.iRequeue.iReFifo_0.iF_iFifos.zFcnt_push_full
560	—"	dut.iPbu.iRefc.refc_mem_debug
561	—"	dut.iPbu.zPassqesp.zPasslist_3.o
562	—"	dut.iPbu.zPassqesp.zPasslist_2.o
563	—"	dut.iPbu.zPassqesp.zPasslist_1.o
564	—"	dut.iPbu.zPassqesp.zPasslist_0.o
565	—"	Filter mask for packets dropped by ERM
566	—"	Filter mask for packets dropped by ERM
567	—"	Filter mask for packets dropped by ERM
568	—"	Filter mask for packets dropped by ERM
569	—"	dut.iPbu.debug_pb_drop
570	—"	dut.iPbu.debug_pb_drop
571	—"	dut.iPbu.debug_pb_drop
572	—"	dut.iPbu.debug_pb_drop
573	—"	constant-573
574	pa_top.switch.pb1.erm.dut.iEqI	constant-574
575	—"	red_zone



id	instance	signal
576	—"	constant-576
577	pa_top.switch.pb1.pfc	constant-577
578	—"	dut_debug_sp_above_rsv
579	—"	dut_debug_sp_above_rsv
580	—"	dut_debug_sp_above_rsv
581	—"	dut_debug_sp_above_rsv
582	—"	constant-582
583	pa_top.switch.pb1.qe0	constant-583
584	—"	dut_assert_dfifo
585	—"	dut_assert_firstflag
586	—"	dut_assert_reset_next
587	—"	dut_drop_cnt
588	—"	dut_send_cnt
589	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
590	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
591	—"	dut_ipkt_fifo_7_debug_in
592	—"	dut_ipkt_fifo_7_debug_out
593	—"	dut_ipkt_fifo_6_debug_in
594	—"	dut_ipkt_fifo_6_debug_out
595	—"	dut_ipkt_fifo_5_debug_in
596	—"	dut_ipkt_fifo_5_debug_out
597	—"	dut_ipkt_fifo_4_debug_in
598	—"	dut_ipkt_fifo_4_debug_out
599	—"	dut_ipkt_fifo_3_debug_in
600	—"	dut_ipkt_fifo_3_debug_out
601	—"	dut_ipkt_fifo_2_debug_in
602	—"	dut_ipkt_fifo_2_debug_out
603	—"	dut_ipkt_fifo_1_debug_in
604	—"	dut_ipkt_fifo_1_debug_out
605	—"	dut_ipkt_fifo_0_debug_in
606	—"	dut_ipkt_fifo_0_debug_out
607	—"	dut_pfifo_level
608	—"	dut_pfifo_level
609	—"	dut_pfifo_level
610	—"	dut_pfifo_level
611	—"	dut_pfifo_level
612	—"	dut_pfifo_level
613	—"	dut_pfifo_level
614	—"	dut_pfifo_level
615	—"	constant-615
616	pa_top.switch.pb1.qe1	constant-616
617	—"	dut_assert_dfifo
618	—"	dut_assert_firstflag
619	—"	dut_assert_reset_next
620	—"	dut_drop_cnt
621	—"	dut_send_cnt
622	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
623	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
624	—"	dut_ipkt_fifo_7_debug_in
625	—"	dut_ipkt_fifo_7_debug_out
626	—"	dut_ipkt_fifo_6_debug_in
627	—"	dut_ipkt_fifo_6_debug_out
628	—"	dut_ipkt_fifo_5_debug_in
629	—"	dut_ipkt_fifo_5_debug_out
630	—"	dut_ipkt_fifo_4_debug_in
631	—"	dut_ipkt_fifo_4_debug_out
632	—"	dut_ipkt_fifo_3_debug_in
633	—"	dut_ipkt_fifo_3_debug_out
634	—"	dut_ipkt_fifo_2_debug_in
635	—"	dut_ipkt_fifo_2_debug_out
636	—"	dut_ipkt_fifo_1_debug_in
637	—"	dut_ipkt_fifo_1_debug_out
638	—"	dut_ipkt_fifo_0_debug_in
639	—"	dut_ipkt_fifo_0_debug_out
640	—"	dut_pfifo_level
641	—"	dut_pfifo_level
642	—"	dut_pfifo_level
643	—"	dut_pfifo_level
644	—"	dut_pfifo_level
645	—"	dut_pfifo_level
646	—"	dut_pfifo_level
647	—"	dut_pfifo_level
648	—"	constant-648
649	pa_top.switch.pb1.qe2	constant-649
650	—"	dut_assert_dfifo
651	—"	dut_assert_firstflag
652	—"	dut_assert_reset_next
653	—"	dut_drop_cnt
654	—"	dut_send_cnt
655	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty

id	instance	signal
656	—"	dut.iDfifo.iF.iFifos.zFcnt.push_full
657	—"	dut.ipkt_fifo_7.debug_in
658	—"	dut.ipkt_fifo_7.debug_out
659	—"	dut.ipkt_fifo_6.debug_in
660	—"	dut.ipkt_fifo_6.debug_out
661	—"	dut.ipkt_fifo_5.debug_in
662	—"	dut.ipkt_fifo_5.debug_out
663	—"	dut.ipkt_fifo_4.debug_in
664	—"	dut.ipkt_fifo_4.debug_out
665	—"	dut.ipkt_fifo_3.debug_in
666	—"	dut.ipkt_fifo_3.debug_out
667	—"	dut.ipkt_fifo_2.debug_in
668	—"	dut.ipkt_fifo_2.debug_out
669	—"	dut.ipkt_fifo_1.debug_in
670	—"	dut.ipkt_fifo_1.debug_out
671	—"	dut.ipkt_fifo_0.debug_in
672	—"	dut.ipkt_fifo_0.debug_out
673	—"	dut.pfifo_level
674	—"	dut.pfifo_level
675	—"	dut.pfifo_level
676	—"	dut.pfifo_level
677	—"	dut.pfifo_level
678	—"	dut.pfifo_level
679	—"	dut.pfifo_level
680	—"	dut.pfifo_level
681	—"	constant-681
682	pa_top.switch.pb1.qe3	constant-682
683	—"	dut.assert_dfifo
684	—"	dut.assert_firstflag
685	—"	dut.assert_reset_next
686	—"	dut.drop_cnt
687	—"	dut.send_cnt
688	—"	dut.iDfifo.iF.iFifos.zFcnt.pop_empty
689	—"	dut.iDfifo.iF.iFifos.zFcnt.push_full
690	—"	dut.ipkt_fifo_7.debug_in
691	—"	dut.ipkt_fifo_7.debug_out
692	—"	dut.ipkt_fifo_6.debug_in
693	—"	dut.ipkt_fifo_6.debug_out
694	—"	dut.ipkt_fifo_5.debug_in
695	—"	dut.ipkt_fifo_5.debug_out
696	—"	dut.ipkt_fifo_4.debug_in
697	—"	dut.ipkt_fifo_4.debug_out
698	—"	dut.ipkt_fifo_3.debug_in
699	—"	dut.ipkt_fifo_3.debug_out
700	—"	dut.ipkt_fifo_2.debug_in
701	—"	dut.ipkt_fifo_2.debug_out
702	—"	dut.ipkt_fifo_1.debug_in
703	—"	dut.ipkt_fifo_1.debug_out
704	—"	dut.ipkt_fifo_0.debug_in
705	—"	dut.ipkt_fifo_0.debug_out
706	—"	dut.pfifo_level
707	—"	dut.pfifo_level
708	—"	dut.pfifo_level
709	—"	dut.pfifo_level
710	—"	dut.pfifo_level
711	—"	dut.pfifo_level
712	—"	dut.pfifo_level
713	—"	dut.pfifo_level
714	—"	constant-714
715	pa_top.switch.pb1.wrr	constant-715
716	—"	dut.debug_below
717	—"	dut.zPassdebugbvalpipe.zPasslist_7_o
718	—"	dut.zPassdebugbvalpipe.zPasslist_6_o
719	—"	dut.zPassdebugbvalpipe.zPasslist_5_o
720	—"	dut.zPassdebugbvalpipe.zPasslist_4_o
721	—"	dut.zPassdebugbvalpipe.zPasslist_3_o
722	—"	dut.zPassdebugbvalpipe.zPasslist_2_o
723	—"	dut.zPassdebugbvalpipe.zPasslist_1_o
724	—"	dut.zPassdebugbvalpipe.zPasslist_0_o
725	—"	dut.reg_bval
726	—"	dut.reg_bval
727	—"	dut.reg_bval
728	—"	dut.reg_bval
729	—"	dut.reg_bval
730	—"	dut.reg_bval
731	—"	dut.reg_bval
732	—"	dut.reg_bval
733	—"	dut.reg_bval
734	—"	dut.reg_bval
735	—"	dut.reg_bval

id	instance	signal
736	—"	dut_reg_bval
737	—"	dut_reg_bval
738	—"	dut_reg_bval
739	—"	dut_reg_bval
740	—"	dut_reg_bval
741	—"	dut_reg_bval
742	—"	dut_reg_bval
743	—"	dut_reg_bval
744	—"	dut_reg_bval
745	—"	dut_reg_bval
746	—"	dut_reg_bval
747	—"	dut_reg_bval
748	—"	dut_reg_bval
749	—"	dut_reg_bval
750	—"	dut_reg_bval
751	—"	dut_reg_bval
752	—"	dut_reg_bval
753	—"	dut_reg_bval
754	—"	dut_reg_bval
755	—"	dut_reg_bval
756	—"	dut_reg_bval
757	—"	dut_reg_bval
758	—"	dut_reg_bval
759	—"	dut_reg_bval
760	—"	dut_reg_bval
761	—"	dut_reg_bval
762	—"	dut_reg_bval
763	—"	dut_reg_bval
764	—"	dut_reg_bval
765	—"	dut_reg_bval
766	—"	dut_reg_bval
767	—"	dut_reg_bval
768	—"	dut_reg_bval
769	—"	dut_reg_bval
770	—"	dut_reg_bval
771	—"	dut_reg_bval
772	—"	dut_reg_bval
773	—"	dut_reg_bval
774	—"	dut_reg_bval
775	—"	dut_reg_bval
776	—"	dut_reg_bval
777	—"	dut_reg_bval
778	—"	dut_reg_bval
779	—"	dut_reg_bval
780	—"	dut_reg_bval
781	—"	dut_reg_bval
782	—"	dut_reg_bval
783	—"	dut_reg_bval
784	—"	dut_reg_bval
785	—"	dut_reg_bval
786	—"	dut_reg_bval
787	—"	dut_reg_bval
788	—"	dut_reg_bval
789	—"	dut_reg_rank
790	—"	dut_reg_rank
791	—"	dut_reg_rank
792	—"	dut_reg_rank
793	—"	dut_reg_rank
794	—"	dut_reg_rank
795	—"	dut_reg_rank
796	—"	dut_reg_rank
797	—"	constant-797
798	pa_top.switch.pb1.qshp	constant-798
799	—"	dut.iPrioShaper_reg_stat
800	—"	dut.iQueueShaper_reg_stat
801	—"	constant-801
802	pa_top.switch.pb2	constant-802
803	—"	dut.iPbu_debug_refc_inc
804	—"	dut.iPbu_debug_port_sch
805	—"	dut.iPbu_dmux_wrr
806	—"	dut.iPbu_debug_qenext
807	—"	dut.iPbu_assert_qediff
808	—"	dut.iPbu_assert_reque_sp
809	—"	Mask of currently receiving packets that have been broken due to BM full
810	—"	dut.iPbu_follow_pfc_accept
811	—"	dut.iPbu.iAssertpacket_3_assert_out
812	—"	pa.top.switch.pb2.iAssertpacket3 {9'valid.bytes, 3'port, 1'last, 1'first}
813	—"	dut.iPbu.iAssertpacket_2_assert_out
814	—"	pa.top.switch.pb2.iAssertpacket2 {9'valid.bytes, 3'port, 1'last, 1'first}
815	—"	dut.iPbu.iAssertpacket_1_assert_out



id	instance	signal
816	—"	pa.top.switch.pb2.iAssertpacket1 {9'valid_bytes, 3'port, 1'last, 1'first}
817	—"	dut.iPbu.iAssertpacket_0.assert_out
818	—"	pa.top.switch.pb2.iAssertpacket0 {9'valid_bytes, 3'port, 1'last, 1'first}
819	—"	dut.iPbu.iPortshaper.iBuckets.reg_stat
820	—"	dut.iPbu.zPassdbgqeread_3.o
821	—"	dut.iPbu.zPassdbgqeread_2.o
822	—"	dut.iPbu.zPassdbgqeread_1.o
823	—"	dut.iPbu.zPassdbgqeread_0.o
824	—"	dut.iPbu.iRequeue.iReFifo.7.iF.iFifos.zFcnt_pop_empty
825	—"	dut.iPbu.iRequeue.iReFifo.7.iF.iFifos.zFcnt_push_full
826	—"	dut.iPbu.iRequeue.iReFifo.6.iF.iFifos.zFcnt_pop_empty
827	—"	dut.iPbu.iRequeue.iReFifo.6.iF.iFifos.zFcnt_push_full
828	—"	dut.iPbu.iRequeue.iReFifo.5.iF.iFifos.zFcnt_pop_empty
829	—"	dut.iPbu.iRequeue.iReFifo.5.iF.iFifos.zFcnt_push_full
830	—"	dut.iPbu.iRequeue.iReFifo.4.iF.iFifos.zFcnt_pop_empty
831	—"	dut.iPbu.iRequeue.iReFifo.4.iF.iFifos.zFcnt_push_full
832	—"	dut.iPbu.iRequeue.iReFifo.3.iF.iFifos.zFcnt_pop_empty
833	—"	dut.iPbu.iRequeue.iReFifo.3.iF.iFifos.zFcnt_push_full
834	—"	dut.iPbu.iRequeue.iReFifo.2.iF.iFifos.zFcnt_pop_empty
835	—"	dut.iPbu.iRequeue.iReFifo.2.iF.iFifos.zFcnt_push_full
836	—"	dut.iPbu.iRequeue.iReFifo.1.iF.iFifos.zFcnt_pop_empty
837	—"	dut.iPbu.iRequeue.iReFifo.1.iF.iFifos.zFcnt_push_full
838	—"	dut.iPbu.iRequeue.iReFifo.0.iF.iFifos.zFcnt_pop_empty
839	—"	dut.iPbu.iRequeue.iReFifo.0.iF.iFifos.zFcnt_push_full
840	—"	dut.iPbu.iRefc.refc_mem_debug
841	—"	dut.iPbu.zPassqesp.zPasslist_3.o
842	—"	dut.iPbu.zPassqesp.zPasslist_2.o
843	—"	dut.iPbu.zPassqesp.zPasslist_1.o
844	—"	dut.iPbu.zPassqesp.zPasslist_0.o
845	—"	Filter mask for packets dropped by ERM
846	—"	Filter mask for packets dropped by ERM
847	—"	Filter mask for packets dropped by ERM
848	—"	Filter mask for packets dropped by ERM
849	—"	dut.iPbu.debug.pb.drop
850	—"	dut.iPbu.debug.pb.drop
851	—"	dut.iPbu.debug.pb.drop
852	—"	dut.iPbu.debug.pb.drop
853	—"	constant-853
854	pa_top.switch.pb2.erm.dut.iEqI	constant-854
855	—"	red_zone
856	—"	constant-856
857	pa_top.switch.pb2.pfc	constant-857
858	—"	dut.debug.sp.above_rsv
859	—"	dut.debug.sp.above_rsv
860	—"	dut.debug.sp.above_rsv
861	—"	dut.debug.sp.above_rsv
862	—"	constant-862
863	pa_top.switch.pb2.qe0	constant-863
864	—"	dut.assert.dfifo
865	—"	dut.assert.firstflag
866	—"	dut.assert.reset_next
867	—"	dut.drop.cnt
868	—"	dut.send.cnt
869	—"	dut.iDfifo.iF.iFifos.zFcnt_pop_empty
870	—"	dut.iDfifo.iF.iFifos.zFcnt_push_full
871	—"	dut.ipkt.fifo.7.debug_in
872	—"	dut.ipkt.fifo.7.debug_out
873	—"	dut.ipkt.fifo.6.debug_in
874	—"	dut.ipkt.fifo.6.debug_out
875	—"	dut.ipkt.fifo.5.debug_in
876	—"	dut.ipkt.fifo.5.debug_out
877	—"	dut.ipkt.fifo.4.debug_in
878	—"	dut.ipkt.fifo.4.debug_out
879	—"	dut.ipkt.fifo.3.debug_in
880	—"	dut.ipkt.fifo.3.debug_out
881	—"	dut.ipkt.fifo.2.debug_in
882	—"	dut.ipkt.fifo.2.debug_out
883	—"	dut.ipkt.fifo.1.debug_in
884	—"	dut.ipkt.fifo.1.debug_out
885	—"	dut.ipkt.fifo.0.debug_in
886	—"	dut.ipkt.fifo.0.debug_out
887	—"	dut.pfifo.level
888	—"	dut.pfifo.level
889	—"	dut.pfifo.level
890	—"	dut.pfifo.level
891	—"	dut.pfifo.level
892	—"	dut.pfifo.level
893	—"	dut.pfifo.level
894	—"	dut.pfifo.level
895	—"	constant-895



id	instance	signal
896	pa_top.switch.pb2.qe1	constant-896
897	—"	dut_assert_dfifo
898	—"	dut_assert_firstflag
899	—"	dut_assert_reset_next
900	—"	dut_drop_cnt
901	—"	dut_send_cnt
902	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
903	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
904	—"	dut_ipkt_fifo_7_debug_in
905	—"	dut_ipkt_fifo_7_debug_out
906	—"	dut_ipkt_fifo_6_debug_in
907	—"	dut_ipkt_fifo_6_debug_out
908	—"	dut_ipkt_fifo_5_debug_in
909	—"	dut_ipkt_fifo_5_debug_out
910	—"	dut_ipkt_fifo_4_debug_in
911	—"	dut_ipkt_fifo_4_debug_out
912	—"	dut_ipkt_fifo_3_debug_in
913	—"	dut_ipkt_fifo_3_debug_out
914	—"	dut_ipkt_fifo_2_debug_in
915	—"	dut_ipkt_fifo_2_debug_out
916	—"	dut_ipkt_fifo_1_debug_in
917	—"	dut_ipkt_fifo_1_debug_out
918	—"	dut_ipkt_fifo_0_debug_in
919	—"	dut_ipkt_fifo_0_debug_out
920	—"	dut_pfifo_level
921	—"	dut_pfifo_level
922	—"	dut_pfifo_level
923	—"	dut_pfifo_level
924	—"	dut_pfifo_level
925	—"	dut_pfifo_level
926	—"	dut_pfifo_level
927	—"	dut_pfifo_level
928	—"	constant-928
929	pa_top.switch.pb2.qe2	constant-929
930	—"	dut_assert_dfifo
931	—"	dut_assert_firstflag
932	—"	dut_assert_reset_next
933	—"	dut_drop_cnt
934	—"	dut_send_cnt
935	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
936	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
937	—"	dut_ipkt_fifo_7_debug_in
938	—"	dut_ipkt_fifo_7_debug_out
939	—"	dut_ipkt_fifo_6_debug_in
940	—"	dut_ipkt_fifo_6_debug_out
941	—"	dut_ipkt_fifo_5_debug_in
942	—"	dut_ipkt_fifo_5_debug_out
943	—"	dut_ipkt_fifo_4_debug_in
944	—"	dut_ipkt_fifo_4_debug_out
945	—"	dut_ipkt_fifo_3_debug_in
946	—"	dut_ipkt_fifo_3_debug_out
947	—"	dut_ipkt_fifo_2_debug_in
948	—"	dut_ipkt_fifo_2_debug_out
949	—"	dut_ipkt_fifo_1_debug_in
950	—"	dut_ipkt_fifo_1_debug_out
951	—"	dut_ipkt_fifo_0_debug_in
952	—"	dut_ipkt_fifo_0_debug_out
953	—"	dut_pfifo_level
954	—"	dut_pfifo_level
955	—"	dut_pfifo_level
956	—"	dut_pfifo_level
957	—"	dut_pfifo_level
958	—"	dut_pfifo_level
959	—"	dut_pfifo_level
960	—"	dut_pfifo_level
961	—"	constant-961
962	pa_top.switch.pb2.qe3	constant-962
963	—"	dut_assert_dfifo
964	—"	dut_assert_firstflag
965	—"	dut_assert_reset_next
966	—"	dut_drop_cnt
967	—"	dut_send_cnt
968	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
969	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
970	—"	dut_ipkt_fifo_7_debug_in
971	—"	dut_ipkt_fifo_7_debug_out
972	—"	dut_ipkt_fifo_6_debug_in
973	—"	dut_ipkt_fifo_6_debug_out
974	—"	dut_ipkt_fifo_5_debug_in
975	—"	dut_ipkt_fifo_5_debug_out



id	instance	signal
976	—"	dut_ipkt_fifo_4_debug_in
977	—"	dut_ipkt_fifo_4_debug_out
978	—"	dut_ipkt_fifo_3_debug_in
979	—"	dut_ipkt_fifo_3_debug_out
980	—"	dut_ipkt_fifo_2_debug_in
981	—"	dut_ipkt_fifo_2_debug_out
982	—"	dut_ipkt_fifo_1_debug_in
983	—"	dut_ipkt_fifo_1_debug_out
984	—"	dut_ipkt_fifo_0_debug_in
985	—"	dut_ipkt_fifo_0_debug_out
986	—"	dut_pfifo_level
987	—"	dut_pfifo_level
988	—"	dut_pfifo_level
989	—"	dut_pfifo_level
990	—"	dut_pfifo_level
991	—"	dut_pfifo_level
992	—"	dut_pfifo_level
993	—"	dut_pfifo_level
994	—"	constant-994
995	pa_top.switch.pb2.wrr	constant-995
996	—"	dut_debug_below
997	—"	dut_zPassdebugbvalpipe_zPasslist_7_o
998	—"	dut_zPassdebugbvalpipe_zPasslist_6_o
999	—"	dut_zPassdebugbvalpipe_zPasslist_5_o
1000	—"	dut_zPassdebugbvalpipe_zPasslist_4_o
1001	—"	dut_zPassdebugbvalpipe_zPasslist_3_o
1002	—"	dut_zPassdebugbvalpipe_zPasslist_2_o
1003	—"	dut_zPassdebugbvalpipe_zPasslist_1_o
1004	—"	dut_zPassdebugbvalpipe_zPasslist_0_o
1005	—"	dut_reg_bval
1006	—"	dut_reg_bval
1007	—"	dut_reg_bval
1008	—"	dut_reg_bval
1009	—"	dut_reg_bval
1010	—"	dut_reg_bval
1011	—"	dut_reg_bval
1012	—"	dut_reg_bval
1013	—"	dut_reg_bval
1014	—"	dut_reg_bval
1015	—"	dut_reg_bval
1016	—"	dut_reg_bval
1017	—"	dut_reg_bval
1018	—"	dut_reg_bval
1019	—"	dut_reg_bval
1020	—"	dut_reg_bval
1021	—"	dut_reg_bval
1022	—"	dut_reg_bval
1023	—"	dut_reg_bval
1024	—"	dut_reg_bval
1025	—"	dut_reg_bval
1026	—"	dut_reg_bval
1027	—"	dut_reg_bval
1028	—"	dut_reg_bval
1029	—"	dut_reg_bval
1030	—"	dut_reg_bval
1031	—"	dut_reg_bval
1032	—"	dut_reg_bval
1033	—"	dut_reg_bval
1034	—"	dut_reg_bval
1035	—"	dut_reg_bval
1036	—"	dut_reg_bval
1037	—"	dut_reg_bval
1038	—"	dut_reg_bval
1039	—"	dut_reg_bval
1040	—"	dut_reg_bval
1041	—"	dut_reg_bval
1042	—"	dut_reg_bval
1043	—"	dut_reg_bval
1044	—"	dut_reg_bval
1045	—"	dut_reg_bval
1046	—"	dut_reg_bval
1047	—"	dut_reg_bval
1048	—"	dut_reg_bval
1049	—"	dut_reg_bval
1050	—"	dut_reg_bval
1051	—"	dut_reg_bval
1052	—"	dut_reg_bval
1053	—"	dut_reg_bval
1054	—"	dut_reg_bval
1055	—"	dut_reg_bval



id	instance	signal
1056	—"	dut_reg_bval
1057	—"	dut_reg_bval
1058	—"	dut_reg_bval
1059	—"	dut_reg_bval
1060	—"	dut_reg_bval
1061	—"	dut_reg_bval
1062	—"	dut_reg_bval
1063	—"	dut_reg_bval
1064	—"	dut_reg_bval
1065	—"	dut_reg_bval
1066	—"	dut_reg_bval
1067	—"	dut_reg_bval
1068	—"	dut_reg_bval
1069	—"	dut_reg_rank
1070	—"	dut_reg_rank
1071	—"	dut_reg_rank
1072	—"	dut_reg_rank
1073	—"	dut_reg_rank
1074	—"	dut_reg_rank
1075	—"	dut_reg_rank
1076	—"	dut_reg_rank
1077	—"	constant-1077
1078	pa_top.switch.pb2.qshp	constant-1078
1079	—"	dut.iPrioshaper.reg_stat
1080	—"	dut.iQueueshaper.reg_stat
1081	—"	constant-1081
1082	pa_top.switch.pb3	constant-1082
1083	—"	dut.iPbu.debug_refc.inc
1084	—"	dut.iPbu.debug_port.sch
1085	—"	dut.iPbu.dmux.wrr
1086	—"	dut.iPbu.debug.qenext
1087	—"	dut.iPbu.assert.qediff
1088	—"	dut.iPbu.assert.reqeue.sp
1089	—"	Mask of currently receiving packets that have been broken due to BM full
1090	—"	dut.iPbu.follow.pfc.accept
1091	—"	dut.iPbu.iAssertpacket_3.assert.out
1092	—"	pa.top.switch.pb3.iAssertpacket3 {9'valid_bytes, 3'port, 1'last, 1'first}
1093	—"	dut.iPbu.iAssertpacket_2.assert.out
1094	—"	pa.top.switch.pb3.iAssertpacket2 {9'valid_bytes, 3'port, 1'last, 1'first}
1095	—"	dut.iPbu.iAssertpacket_1.assert.out
1096	—"	pa.top.switch.pb3.iAssertpacket1 {9'valid_bytes, 3'port, 1'last, 1'first}
1097	—"	dut.iPbu.iAssertpacket_0.assert.out
1098	—"	pa.top.switch.pb3.iAssertpacket0 {9'valid_bytes, 3'port, 1'last, 1'first}
1099	—"	dut.iPbu.iPortshaper.iBuckets.reg_stat
1100	—"	dut.iPbu.zPassdbgqeread_3.o
1101	—"	dut.iPbu.zPassdbgqeread_2.o
1102	—"	dut.iPbu.zPassdbgqeread_1.o
1103	—"	dut.iPbu.zPassdbgqeread_0.o
1104	—"	dut.iPbu.iRequeue.iReFifo_7.iF_iFifos.zFcnt.pop.empty
1105	—"	dut.iPbu.iRequeue.iReFifo_7.iF_iFifos.zFcnt.push.full
1106	—"	dut.iPbu.iRequeue.iReFifo_6.iF_iFifos.zFcnt.pop.empty
1107	—"	dut.iPbu.iRequeue.iReFifo_6.iF_iFifos.zFcnt.push.full
1108	—"	dut.iPbu.iRequeue.iReFifo_5.iF_iFifos.zFcnt.pop.empty
1109	—"	dut.iPbu.iRequeue.iReFifo_5.iF_iFifos.zFcnt.push.full
1110	—"	dut.iPbu.iRequeue.iReFifo_4.iF_iFifos.zFcnt.pop.empty
1111	—"	dut.iPbu.iRequeue.iReFifo_4.iF_iFifos.zFcnt.push.full
1112	—"	dut.iPbu.iRequeue.iReFifo_3.iF_iFifos.zFcnt.pop.empty
1113	—"	dut.iPbu.iRequeue.iReFifo_3.iF_iFifos.zFcnt.push.full
1114	—"	dut.iPbu.iRequeue.iReFifo_2.iF_iFifos.zFcnt.pop.empty
1115	—"	dut.iPbu.iRequeue.iReFifo_2.iF_iFifos.zFcnt.push.full
1116	—"	dut.iPbu.iRequeue.iReFifo_1.iF_iFifos.zFcnt.pop.empty
1117	—"	dut.iPbu.iRequeue.iReFifo_1.iF_iFifos.zFcnt.push.full
1118	—"	dut.iPbu.iRequeue.iReFifo_0.iF_iFifos.zFcnt.pop.empty
1119	—"	dut.iPbu.iRequeue.iReFifo_0.iF_iFifos.zFcnt.push.full
1120	—"	dut.iPbu.iRefc.refc_mem.debug
1121	—"	dut.iPbu.zPassqesp.zPasslist_3.o
1122	—"	dut.iPbu.zPassqesp.zPasslist_2.o
1123	—"	dut.iPbu.zPassqesp.zPasslist_1.o
1124	—"	dut.iPbu.zPassqesp.zPasslist_0.o
1125	—"	Filter mask for packets dropped by ERM
1126	—"	Filter mask for packets dropped by ERM
1127	—"	Filter mask for packets dropped by ERM
1128	—"	Filter mask for packets dropped by ERM
1129	—"	dut.iPbu.debug.pb.drop
1130	—"	dut.iPbu.debug.pb.drop
1131	—"	dut.iPbu.debug.pb.drop
1132	—"	dut.iPbu.debug.pb.drop
1133	—"	constant-1133
1134	pa_top.switch.pb3.erm.dut.iEql	constant-1134
1135	—"	red_zone



id	instance	signal
1136	—"	constant-1136
1137	pa_top.switch.pb3.pfc	constant-1137
1138	—"	dut_debug_sp_above_rsv
1139	—"	dut_debug_sp_above_rsv
1140	—"	dut_debug_sp_above_rsv
1141	—"	dut_debug_sp_above_rsv
1142	—"	constant-1142
1143	pa_top.switch.pb3.qe0	constant-1143
1144	—"	dut_assert_dfifo
1145	—"	dut_assert_firstflag
1146	—"	dut_assert_reset_next
1147	—"	dut_drop_cnt
1148	—"	dut_send_cnt
1149	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
1150	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
1151	—"	dut_ipkt_fifo_7_debug_in
1152	—"	dut_ipkt_fifo_7_debug_out
1153	—"	dut_ipkt_fifo_6_debug_in
1154	—"	dut_ipkt_fifo_6_debug_out
1155	—"	dut_ipkt_fifo_5_debug_in
1156	—"	dut_ipkt_fifo_5_debug_out
1157	—"	dut_ipkt_fifo_4_debug_in
1158	—"	dut_ipkt_fifo_4_debug_out
1159	—"	dut_ipkt_fifo_3_debug_in
1160	—"	dut_ipkt_fifo_3_debug_out
1161	—"	dut_ipkt_fifo_2_debug_in
1162	—"	dut_ipkt_fifo_2_debug_out
1163	—"	dut_ipkt_fifo_1_debug_in
1164	—"	dut_ipkt_fifo_1_debug_out
1165	—"	dut_ipkt_fifo_0_debug_in
1166	—"	dut_ipkt_fifo_0_debug_out
1167	—"	dut_pfifo_level
1168	—"	dut_pfifo_level
1169	—"	dut_pfifo_level
1170	—"	dut_pfifo_level
1171	—"	dut_pfifo_level
1172	—"	dut_pfifo_level
1173	—"	dut_pfifo_level
1174	—"	dut_pfifo_level
1175	—"	constant-1175
1176	pa_top.switch.pb3.qe1	constant-1176
1177	—"	dut_assert_dfifo
1178	—"	dut_assert_firstflag
1179	—"	dut_assert_reset_next
1180	—"	dut_drop_cnt
1181	—"	dut_send_cnt
1182	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
1183	—"	dut_iDfifo_iF_iFifos_zFcnt_push_full
1184	—"	dut_ipkt_fifo_7_debug_in
1185	—"	dut_ipkt_fifo_7_debug_out
1186	—"	dut_ipkt_fifo_6_debug_in
1187	—"	dut_ipkt_fifo_6_debug_out
1188	—"	dut_ipkt_fifo_5_debug_in
1189	—"	dut_ipkt_fifo_5_debug_out
1190	—"	dut_ipkt_fifo_4_debug_in
1191	—"	dut_ipkt_fifo_4_debug_out
1192	—"	dut_ipkt_fifo_3_debug_in
1193	—"	dut_ipkt_fifo_3_debug_out
1194	—"	dut_ipkt_fifo_2_debug_in
1195	—"	dut_ipkt_fifo_2_debug_out
1196	—"	dut_ipkt_fifo_1_debug_in
1197	—"	dut_ipkt_fifo_1_debug_out
1198	—"	dut_ipkt_fifo_0_debug_in
1199	—"	dut_ipkt_fifo_0_debug_out
1200	—"	dut_pfifo_level
1201	—"	dut_pfifo_level
1202	—"	dut_pfifo_level
1203	—"	dut_pfifo_level
1204	—"	dut_pfifo_level
1205	—"	dut_pfifo_level
1206	—"	dut_pfifo_level
1207	—"	dut_pfifo_level
1208	—"	constant-1208
1209	pa_top.switch.pb3.qe2	constant-1209
1210	—"	dut_assert_dfifo
1211	—"	dut_assert_firstflag
1212	—"	dut_assert_reset_next
1213	—"	dut_drop_cnt
1214	—"	dut_send_cnt
1215	—"	dut_iDfifo_iF_iFifos_zFcnt_pop_empty

id	instance	signal
1216	—"	dut.iDfifo.iF.iFifos.zFcnt.push_full
1217	—"	dut.ipkt_fifo_7.debug_in
1218	—"	dut.ipkt_fifo_7.debug_out
1219	—"	dut.ipkt_fifo_6.debug_in
1220	—"	dut.ipkt_fifo_6.debug_out
1221	—"	dut.ipkt_fifo_5.debug_in
1222	—"	dut.ipkt_fifo_5.debug_out
1223	—"	dut.ipkt_fifo_4.debug_in
1224	—"	dut.ipkt_fifo_4.debug_out
1225	—"	dut.ipkt_fifo_3.debug_in
1226	—"	dut.ipkt_fifo_3.debug_out
1227	—"	dut.ipkt_fifo_2.debug_in
1228	—"	dut.ipkt_fifo_2.debug_out
1229	—"	dut.ipkt_fifo_1.debug_in
1230	—"	dut.ipkt_fifo_1.debug_out
1231	—"	dut.ipkt_fifo_0.debug_in
1232	—"	dut.ipkt_fifo_0.debug_out
1233	—"	dut.pfifo_level
1234	—"	dut.pfifo_level
1235	—"	dut.pfifo_level
1236	—"	dut.pfifo_level
1237	—"	dut.pfifo_level
1238	—"	dut.pfifo_level
1239	—"	dut.pfifo_level
1240	—"	dut.pfifo_level
1241	—"	constant-1241
1242	pa_top.switch.pb3.qe3	constant-1242
1243	—"	dut.assert_dfifo
1244	—"	dut.assert_firstflag
1245	—"	dut.assert_reset_next
1246	—"	dut.drop_cnt
1247	—"	dut.send_cnt
1248	—"	dut.iDfifo.iF.iFifos.zFcnt.pop_empty
1249	—"	dut.iDfifo.iF.iFifos.zFcnt.push_full
1250	—"	dut.ipkt_fifo_7.debug_in
1251	—"	dut.ipkt_fifo_7.debug_out
1252	—"	dut.ipkt_fifo_6.debug_in
1253	—"	dut.ipkt_fifo_6.debug_out
1254	—"	dut.ipkt_fifo_5.debug_in
1255	—"	dut.ipkt_fifo_5.debug_out
1256	—"	dut.ipkt_fifo_4.debug_in
1257	—"	dut.ipkt_fifo_4.debug_out
1258	—"	dut.ipkt_fifo_3.debug_in
1259	—"	dut.ipkt_fifo_3.debug_out
1260	—"	dut.ipkt_fifo_2.debug_in
1261	—"	dut.ipkt_fifo_2.debug_out
1262	—"	dut.ipkt_fifo_1.debug_in
1263	—"	dut.ipkt_fifo_1.debug_out
1264	—"	dut.ipkt_fifo_0.debug_in
1265	—"	dut.ipkt_fifo_0.debug_out
1266	—"	dut.pfifo_level
1267	—"	dut.pfifo_level
1268	—"	dut.pfifo_level
1269	—"	dut.pfifo_level
1270	—"	dut.pfifo_level
1271	—"	dut.pfifo_level
1272	—"	dut.pfifo_level
1273	—"	dut.pfifo_level
1274	—"	constant-1274
1275	pa_top.switch.pb3.wrr	constant-1275
1276	—"	dut.debug_below
1277	—"	dut.zPassdebugbvalpipe.zPasslist_7_o
1278	—"	dut.zPassdebugbvalpipe.zPasslist_6_o
1279	—"	dut.zPassdebugbvalpipe.zPasslist_5_o
1280	—"	dut.zPassdebugbvalpipe.zPasslist_4_o
1281	—"	dut.zPassdebugbvalpipe.zPasslist_3_o
1282	—"	dut.zPassdebugbvalpipe.zPasslist_2_o
1283	—"	dut.zPassdebugbvalpipe.zPasslist_1_o
1284	—"	dut.zPassdebugbvalpipe.zPasslist_0_o
1285	—"	dut.reg_bval
1286	—"	dut.reg_bval
1287	—"	dut.reg_bval
1288	—"	dut.reg_bval
1289	—"	dut.reg_bval
1290	—"	dut.reg_bval
1291	—"	dut.reg_bval
1292	—"	dut.reg_bval
1293	—"	dut.reg_bval
1294	—"	dut.reg_bval
1295	—"	dut.reg_bval

id	instance	signal
1296	—"	dut_reg_bval
1297	—"	dut_reg_bval
1298	—"	dut_reg_bval
1299	—"	dut_reg_bval
1300	—"	dut_reg_bval
1301	—"	dut_reg_bval
1302	—"	dut_reg_bval
1303	—"	dut_reg_bval
1304	—"	dut_reg_bval
1305	—"	dut_reg_bval
1306	—"	dut_reg_bval
1307	—"	dut_reg_bval
1308	—"	dut_reg_bval
1309	—"	dut_reg_bval
1310	—"	dut_reg_bval
1311	—"	dut_reg_bval
1312	—"	dut_reg_bval
1313	—"	dut_reg_bval
1314	—"	dut_reg_bval
1315	—"	dut_reg_bval
1316	—"	dut_reg_bval
1317	—"	dut_reg_bval
1318	—"	dut_reg_bval
1319	—"	dut_reg_bval
1320	—"	dut_reg_bval
1321	—"	dut_reg_bval
1322	—"	dut_reg_bval
1323	—"	dut_reg_bval
1324	—"	dut_reg_bval
1325	—"	dut_reg_bval
1326	—"	dut_reg_bval
1327	—"	dut_reg_bval
1328	—"	dut_reg_bval
1329	—"	dut_reg_bval
1330	—"	dut_reg_bval
1331	—"	dut_reg_bval
1332	—"	dut_reg_bval
1333	—"	dut_reg_bval
1334	—"	dut_reg_bval
1335	—"	dut_reg_bval
1336	—"	dut_reg_bval
1337	—"	dut_reg_bval
1338	—"	dut_reg_bval
1339	—"	dut_reg_bval
1340	—"	dut_reg_bval
1341	—"	dut_reg_bval
1342	—"	dut_reg_bval
1343	—"	dut_reg_bval
1344	—"	dut_reg_bval
1345	—"	dut_reg_bval
1346	—"	dut_reg_bval
1347	—"	dut_reg_bval
1348	—"	dut_reg_bval
1349	—"	dut_reg_rank
1350	—"	dut_reg_rank
1351	—"	dut_reg_rank
1352	—"	dut_reg_rank
1353	—"	dut_reg_rank
1354	—"	dut_reg_rank
1355	—"	dut_reg_rank
1356	—"	dut_reg_rank
1357	—"	constant-1357
1358	pa_top.switch.pb3.qshp	constant-1358
1359	—"	dut_iPrioshaper_reg_stat
1360	—"	dut_iQueueshaper_reg_stat
1361	—"	constant-1361
1362	pa_top.switch.bm0	constant-1362
1363	—"	dut_bm_ifree_4.debug_free
1364	—"	dut_bm_ifree_3.debug_free
1365	—"	dut_bm_ifree_2.debug_free
1366	—"	dut_bm_ifree_1.debug_free
1367	—"	dut_bm_ifree_0.debug_free
1368	—"	constant-1368
1369	pa_top.switch.bm1	constant-1369
1370	—"	dut_bm_ifree_4.debug_free
1371	—"	dut_bm_ifree_3.debug_free
1372	—"	dut_bm_ifree_2.debug_free
1373	—"	dut_bm_ifree_1.debug_free
1374	—"	dut_bm_ifree_0.debug_free
1375	—"	constant-1375



id	instance	signal
1376	pa_top.switch.bm2	constant-1376
1377	—"	dut_bm_ifree_4.debug_free
1378	—"	dut_bm_ifree_3.debug_free
1379	—"	dut_bm_ifree_2.debug_free
1380	—"	dut_bm_ifree_1.debug_free
1381	—"	dut_bm_ifree_0.debug_free
1382	—"	constant-1382
1383	pa_top.switch.bm3	constant-1383
1384	—"	dut_bm_ifree_4.debug_free
1385	—"	dut_bm_ifree_3.debug_free
1386	—"	dut_bm_ifree_2.debug_free
1387	—"	dut_bm_ifree_1.debug_free
1388	—"	dut_bm_ifree_0.debug_free
1389	—"	constant-1389
1390	pa_top.switch.ps0	constant-1390
1391	—"	halt_from_ps
1392	—"	dut_iPs2_zPsAssert_item
1393	—"	dut_iPs2_iBridge_6.assert_reset
1394	—"	dut_iPs2_iBridge_5.assert_reset
1395	—"	dut_iPs2_iBridge_4.assert_reset
1396	—"	dut_iPs2_iBridge_3.assert_reset
1397	—"	dut_iPs2_iBridge_2.assert_reset
1398	—"	dut_iPs2_iBridge_1.assert_reset
1399	—"	dut_iPs2_iBridge_0.assert_reset
1400	—"	dut_iPs2_iSplitter_0.assert_noend
1401	—"	dut_iPs2_iSplitter_0.assert_ptr
1402	—"	dut_iPs2_iSplitter_0.used_mem
1403	—"	dut_iPs2_iSplitter_0.used_mem
1404	—"	dut_iPs2_iSplitter_0.used_mem
1405	—"	dut_iPs2_iSplitter_0.used_mem
1406	—"	dut_iPs2_iSplitter_0.used_mem
1407	—"	dut_iPs2_iSplitter_0.used_mem
1408	—"	dut_iPs2_iSplitter_0.used_mem
1409	—"	dut_iPs2_iSplitter_0.used_mem
1410	—"	constant-1410
1411	pa_top.switch.ps1	constant-1411
1412	—"	halt_from_ps
1413	—"	dut_iPs2_zPsAssert_item
1414	—"	dut_iPs2_iBridge_6.assert_reset
1415	—"	dut_iPs2_iBridge_5.assert_reset
1416	—"	dut_iPs2_iBridge_4.assert_reset
1417	—"	dut_iPs2_iBridge_3.assert_reset
1418	—"	dut_iPs2_iBridge_2.assert_reset
1419	—"	dut_iPs2_iBridge_1.assert_reset
1420	—"	dut_iPs2_iBridge_0.assert_reset
1421	—"	dut_iPs2_iSplitter_0.assert_noend
1422	—"	dut_iPs2_iSplitter_0.assert_ptr
1423	—"	dut_iPs2_iSplitter_0.used_mem
1424	—"	dut_iPs2_iSplitter_0.used_mem
1425	—"	dut_iPs2_iSplitter_0.used_mem
1426	—"	dut_iPs2_iSplitter_0.used_mem
1427	—"	dut_iPs2_iSplitter_0.used_mem
1428	—"	dut_iPs2_iSplitter_0.used_mem
1429	—"	dut_iPs2_iSplitter_0.used_mem
1430	—"	dut_iPs2_iSplitter_0.used_mem
1431	—"	constant-1431
1432	pa_top.switch.ps2	constant-1432
1433	—"	halt_from_ps
1434	—"	dut_iPs2_zPsAssert_item
1435	—"	dut_iPs2_iBridge_6.assert_reset
1436	—"	dut_iPs2_iBridge_5.assert_reset
1437	—"	dut_iPs2_iBridge_4.assert_reset
1438	—"	dut_iPs2_iBridge_3.assert_reset
1439	—"	dut_iPs2_iBridge_2.assert_reset
1440	—"	dut_iPs2_iBridge_1.assert_reset
1441	—"	dut_iPs2_iBridge_0.assert_reset
1442	—"	dut_iPs2_iSplitter_0.assert_noend
1443	—"	dut_iPs2_iSplitter_0.assert_ptr
1444	—"	dut_iPs2_iSplitter_0.used_mem
1445	—"	dut_iPs2_iSplitter_0.used_mem
1446	—"	dut_iPs2_iSplitter_0.used_mem
1447	—"	dut_iPs2_iSplitter_0.used_mem
1448	—"	dut_iPs2_iSplitter_0.used_mem
1449	—"	dut_iPs2_iSplitter_0.used_mem
1450	—"	dut_iPs2_iSplitter_0.used_mem
1451	—"	dut_iPs2_iSplitter_0.used_mem
1452	—"	constant-1452
1453	pa_top.switch.ps3	constant-1453
1454	—"	halt_from_ps
1455	—"	dut_iPs2_zPsAssert_item

id	instance	signal
1456	—"	dut.iPs2.iBridge_6.assert.reset
1457	—"	dut.iPs2.iBridge_5.assert.reset
1458	—"	dut.iPs2.iBridge_4.assert.reset
1459	—"	dut.iPs2.iBridge_3.assert.reset
1460	—"	dut.iPs2.iBridge_2.assert.reset
1461	—"	dut.iPs2.iBridge_1.assert.reset
1462	—"	dut.iPs2.iBridge_0.assert.reset
1463	—"	dut.iPs2.iSplitter_0.assert.noend
1464	—"	dut.iPs2.iSplitter_0.assert.ptr
1465	—"	dut.iPs2.iSplitter_0.used.mem
1466	—"	dut.iPs2.iSplitter_0.used.mem
1467	—"	dut.iPs2.iSplitter_0.used.mem
1468	—"	dut.iPs2.iSplitter_0.used.mem
1469	—"	dut.iPs2.iSplitter_0.used.mem
1470	—"	dut.iPs2.iSplitter_0.used.mem
1471	—"	dut.iPs2.iSplitter_0.used.mem
1472	—"	dut.iPs2.iSplitter_0.used.mem
1473	—"	constant-1473
1474	pa_top.switch.epp0	constant-1474
1475	—"	dut.iEpp.assert.ipkt
1476	—"	dut.iEpp.assert.opkt
1477	—"	epp.ipkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1478	—"	epp.opkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1479	—"	dut.iEpp.iDropper.da_0
1480	—"	dut.iEpp.iDropper.da_1
1481	—"	dut.iEpp.iDropper.dbg_drop
1482	—"	dut.iEpp.iDropper.dbg_ifirst
1483	—"	dut.iEpp.iDropper.dbg_ilast
1484	—"	dut.iEpp.iDropper.sa_0
1485	—"	dut.iEpp.iDropper.sa_1
1486	—"	pa.top.switch.epp0.iPacketassertpm {9'valid_bytes, 3'port, 1'last, 1'first}
1487	—"	pa.top.switch.epp0.iPacketassertin {9'valid_bytes, 3'port, 1'last, 1'first}
1488	—"	constant-1488
1489	pa_top.switch.epp0.pm	constant-1489
1490	—"	pm.fifo.overflow
1491	—"	dut.dbg.fifo.full
1492	—"	halt.from.pm
1493	—"	dut.iFifoa.debug.in
1494	—"	dut.iFifoa.debug.out
1495	—"	constant-1495
1496	pa_top.switch.epp1	constant-1496
1497	—"	dut.iEpp.assert.ipkt
1498	—"	dut.iEpp.assert.opkt
1499	—"	epp.ipkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1500	—"	epp.opkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1501	—"	dut.iEpp.iDropper.da_0
1502	—"	dut.iEpp.iDropper.da_1
1503	—"	dut.iEpp.iDropper.dbg_drop
1504	—"	dut.iEpp.iDropper.dbg_ifirst
1505	—"	dut.iEpp.iDropper.dbg_ilast
1506	—"	dut.iEpp.iDropper.sa_0
1507	—"	dut.iEpp.iDropper.sa_1
1508	—"	pa.top.switch.epp1.iPacketassertpm {9'valid_bytes, 3'port, 1'last, 1'first}
1509	—"	pa.top.switch.epp1.iPacketassertin {9'valid_bytes, 3'port, 1'last, 1'first}
1510	—"	constant-1510
1511	pa_top.switch.epp1.pm	constant-1511
1512	—"	pm.fifo.overflow
1513	—"	dut.dbg.fifo.full
1514	—"	halt.from.pm
1515	—"	dut.iFifoa.debug.in
1516	—"	dut.iFifoa.debug.out
1517	—"	constant-1517
1518	pa_top.switch.epp2	constant-1518
1519	—"	dut.iEpp.assert.ipkt
1520	—"	dut.iEpp.assert.opkt
1521	—"	epp.ipkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1522	—"	epp.opkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1523	—"	dut.iEpp.iDropper.da_0
1524	—"	dut.iEpp.iDropper.da_1
1525	—"	dut.iEpp.iDropper.dbg_drop
1526	—"	dut.iEpp.iDropper.dbg_ifirst
1527	—"	dut.iEpp.iDropper.dbg_ilast
1528	—"	dut.iEpp.iDropper.sa_0
1529	—"	dut.iEpp.iDropper.sa_1
1530	—"	pa.top.switch.epp2.iPacketassertpm {9'valid_bytes, 3'port, 1'last, 1'first}
1531	—"	pa.top.switch.epp2.iPacketassertin {9'valid_bytes, 3'port, 1'last, 1'first}
1532	—"	constant-1532
1533	pa_top.switch.epp2.pm	constant-1533
1534	—"	pm.fifo.overflow
1535	—"	dut.dbg.fifo.full

id	instance	signal
1536	—"	halt_from_pm
1537	—"	dut.iFifo.debug_in
1538	—"	dut.iFifo.debug_out
1539	—"	constant-1539
1540	pa_top.switch.epp3	constant-1540
1541	—"	dut.iEpp.assert_ipkt
1542	—"	dut.iEpp.assert_opkt
1543	—"	epp.ipkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1544	—"	epp.opkt.bus {18'data, 9'valid_bytes, 3'id, 1'last, 1'first}
1545	—"	dut.iEpp.iDropper_da_0
1546	—"	dut.iEpp.iDropper_da_1
1547	—"	dut.iEpp.iDropper_dbg_drop
1548	—"	dut.iEpp.iDropper_dbg_ifirst
1549	—"	dut.iEpp.iDropper_dbg_ilast
1550	—"	dut.iEpp.iDropper_sa_0
1551	—"	dut.iEpp.iDropper_sa_1
1552	—"	pa.top.switch.epp3.iPacketassertpm {9'valid_bytes, 3'port, 1'last, 1'first}
1553	—"	pa.top.switch.epp3.iPacketassertin {9'valid_bytes, 3'port, 1'last, 1'first}
1554	—"	constant-1554
1555	pa_top.switch.epp3.pm	constant-1555
1556	—"	pm.fifo.overflow
1557	—"	dut.dbg_fifo_full
1558	—"	halt_from_pm
1559	—"	dut.iFifo.debug_in
1560	—"	dut.iFifo.debug_out
1561	—"	constant-1561
1562	pa_top.switch.ingress_common	constant-1562
1563	—"	dut.iLearnage.iHitUpdate.iFifo_3.iF.iFifos.zFcnt_pop_empty
1564	—"	dut.iLearnage.iHitUpdate.iFifo_3.iF.iFifos.zFcnt_push_full
1565	—"	dut.iLearnage.iHitUpdate.iFifo_2.iF.iFifos.zFcnt_pop_empty
1566	—"	dut.iLearnage.iHitUpdate.iFifo_2.iF.iFifos.zFcnt_push_full
1567	—"	dut.iLearnage.iHitUpdate.iFifo_1.iF.iFifos.zFcnt_pop_empty
1568	—"	dut.iLearnage.iHitUpdate.iFifo_1.iF.iFifos.zFcnt_push_full
1569	—"	dut.iLearnage.iHitUpdate.iFifo_0.iF.iFifos.zFcnt_pop_empty
1570	—"	dut.iLearnage.iHitUpdate.iFifo_0.iF.iFifos.zFcnt_push_full
1571	—"	dut.iLearnage.iLearn_hw.learning_aborted
1572	—"	dut.iLearnage.iLearn.iFifo_3.iF.iFifos.zFcnt_pop_empty
1573	—"	dut.iLearnage.iLearn.iFifo_3.iF.iFifos.zFcnt_push_full
1574	—"	dut.iLearnage.iLearn.iFifo_2.iF.iFifos.zFcnt_pop_empty
1575	—"	dut.iLearnage.iLearn.iFifo_2.iF.iFifos.zFcnt_push_full
1576	—"	dut.iLearnage.iLearn.iFifo_1.iF.iFifos.zFcnt_pop_empty
1577	—"	dut.iLearnage.iLearn.iFifo_1.iF.iFifos.zFcnt_push_full
1578	—"	dut.iLearnage.iLearn.iFifo_0.iF.iFifos.zFcnt_pop_empty
1579	—"	dut.iLearnage.iLearn.iFifo_0.iF.iFifos.zFcnt_push_full
1580	—"	dut.iLearnage.iConf.iFifo_2.iFifo.iF.iFifos.zFcnt_pop_empty
1581	—"	dut.iLearnage.iConf.iFifo_2.iFifo.iF.iFifos.zFcnt_push_full
1582	—"	dut.iLearnage.iConf.iFifo_1.iFifo.iF.iFifos.zFcnt_pop_empty
1583	—"	dut.iLearnage.iConf.iFifo_1.iFifo.iF.iFifos.zFcnt_push_full
1584	—"	dut.iLearnage.iConf.iFifo_0.iFifo.iF.iFifos.zFcnt_pop_empty
1585	—"	dut.iLearnage.iConf.iFifo_0.iFifo.iF.iFifos.zFcnt_push_full
1586	—"	dut.iMbsc.iFlood_reg_stat
1587	—"	dut.iMbsc.iMc_reg_stat
1588	—"	dut.iMbsc.iBc_reg_stat
1589	—"	constant-1589
1590	pa_top.switch.interface_common	constant-1590
1591	—"	dut.zFaii.iMf.zMf_1.item
1592	—"	dut.zFaii.iMf.zMf_1.zSs2l_2.o
1593	—"	dut.zFaii.iMf.zMf_1.zSs2l_1.o
1594	—"	dut.zFaii.iMf.zMf_1.zSs2l_0.o
1595	—"	dut.zFaip.iMf.zMf_1.item
1596	—"	dut.zFaip.iMf.zMf_1.zSs2l_2.o
1597	—"	dut.zFaip.iMf.zMf_1.zSs2l_1.o
1598	—"	dut.zFaip.iMf.zMf_1.zSs2l_0.o
1599	—"	dut.zFaie.iMf.zMf_1.item
1600	—"	dut.zFaie.iMf.zMf_1.zSs2l_2.o
1601	—"	dut.zFaie.iMf.zMf_1.zSs2l_1.o
1602	—"	dut.zFaie.iMf.zMf_1.zSs2l_0.o
1603	—"	dut.zFaiq.iMf.zMf_1.item
1604	—"	dut.zFaiq.iMf.zMf_1.zSs2l_2.o
1605	—"	dut.zFaiq.iMf.zMf_1.zSs2l_1.o
1606	—"	dut.zFaiq.iMf.zMf_1.zSs2l_0.o
1607	—"	dut.zFais.iMf.zMf_1.item
1608	—"	dut.zFais.iMf.zMf_1.zSs2l_2.o
1609	—"	dut.zFais.iMf.zMf_1.zSs2l_1.o
1610	—"	dut.zFais.iMf.zMf_1.zSs2l_0.o
1611	—"	constant-1611

Table 31.8: Debug Selection Map



31.7 Debug Write Interface

The debug write interface is an input port to the Switch Core that can be used for debugging purposes. In normal operation the *debug.write.data* pins must be tied low. The function of the debug write interface is controlled by registers in the individual blocks. In this core only the tick dividers use the debug write interface. See [Core Tick Select](#).

Pin	Direction	Size	Description
debug.write.data	In	1	The debug write input data. Must be tied low for normal switch operation.

Table 31.9: The Debug Write interface

Chapter 32

Configuration Interface

The configuration interface is an AMBA APB interface used for monitoring the core and for configuration of internal registers and tables. The pins are described in Table 31.4 on page 187, but for a detailed description of the APB interface see the AMBA APB Protocol Specification Version 2.0, available at developer.arm.com

32.1 Response time

The response time may vary between registers, and even vary for the same register depending on how busy the core is switching packets. The response time is in the order of tens of core clock cycles.

32.2 Out of range accesses

There is no range check on the configuration interface, so an access to an address that is not mapped to any register will result in a internal timeout and raise the *pslverr* on the bus.

32.3 Atomic Wide Access

There are a few recommendations how to access wide registers (registers that are wider than the APB data bus). The interface does allow a more flexible access pattern than what is described here. If that is needed then see the next section.

The highest address bit (33) on the APB bus is not a normal address bit. It is used to control wide register access. It will be referred to as the Accumulator Bit in the following description.

- Wide Reads
 - always read wide register starting with the lowest address and ending with the highest address.
 - when reading the lowest address of the register the Accumulator Bit should be 0.
 - when reading the other addresses of the register the Accumulator Bit should be 1.
- Wide Writes
 - always write wide register starting with the lowest address and ending with the highest address.
 - when writing the highest address of the register the Accumulator Bit should be 0.
 - when writing the other addresses of the register the Accumulator Bit should be 1.
- Narrow reads and writes
 - If the register fits within the APB data bus width then the Accumulator Bit should be 0.

Note that if there are bridges between the CPU and the APB bus then they need to be set up to guarantee the order of accesses.

The software API implementation provided with the switch handles the Accumulator Bit thereby hiding it completely for the software that use the API.

32.4 Accumulator Accesses

Each table or register bank where the data is wider than the configuration data bus, will be equipped with a shadow-register called an accumulator. The accumulator allows the full data width to be updated atomically even though the bus width is narrower than the data. The accumulator is accessed by setting bit 33 of the address high during a normal register access. An access with bit 33 of the address low we call a **DEFAULT** access, while an access with bit 33 of the address high is called an **ACCUMULATOR** access. The register section of the datasheet will only list the addresses for **DEFAULT** access to the registers. Address bit 33 is considered an accumulator flag, and not a part of the address.

A **DEFAULT** read will return the requested data in the reply, and at the same time load the full data width into the accumulator. Thus following up the **DEFAULT** read with **ACCUMULATOR** reads will allow reading the state of the register at the time of the original **DEFAULT** read. If data consistency is not important, all the reads can be of the **DEFAULT** type, but there is no point because the read performance is the same. In fact reading a table will potentially be faster using the accumulator, because only the first access will have to wait for access to the physical memory.

Writes work similarly, but the other way around. The accumulator will first be loaded using **ACCUMULATOR** writes and then the contents of the accumulator is written to the register. The final **DEFAULT** write will use the data given as *wdata*, and fill it out with the data in the accumulator. Writing data wider than the bus cannot be done without taking the accumulator into account.

If only a part of a very wide register is to be written, the most efficient approach may be to do a **DEFAULT** read (loading the accumulator) followed by a **DEFAULT** write. But note that there is no way to do a truly atomic read-modify-write. Any write that the core slips in while the accumulator is loaded will be over-written by the following **DEFAULT** write.

When the data is wider than the bus the address is stepped by 2^n between table indexes or registers. For instance a 32-bit bus and a 65 bit wide table will result in index 1 starting at address 4, with address 3 unused and address 2 only containing a single valid bit.

Chapter 33

Debugging the Design

The design contains debug points. They are available as registers in the design. For each debug point there is a counter. The fields which are more than a single bit also have a comparison register. This register is used for updating the counter only for specific matching values.

33.1 Debug Counters in Ingress Packet Processing

The Cnt Id field in the table below points to the counter to be updated in the counter bank of the **Debug IPP Counter** register.

Register	Cnt Id	Bits	Description
IPP Debug finalVid	0	13	The VID used to lookup in the VLAN table The setup of mask and compare is located in register Debug Counter finalVid Setup . This register enables the user to see if a specific value has been seen.
IPP Debug vlanVidOp	1	3	The VLAN Table VID Operation The setup of mask and compare is located in register Debug Counter vlanVidOp Setup . This register enables the user to see if a specific value has been seen.
IPP Debug I2DaTcamHitsAndCast	2	15	If the L2 TCAM was hit and which type was returned. The setup of mask and compare is located in register Debug Counter I2DaTcamHitsAndCast Setup . This register enables the user to see if a specific value has been seen.
IPP Debug I2DaHashKey	3	60	The hash value for the packet. The setup of mask and compare is located in register Debug Counter I2DaHashKey Setup . This register enables the user to see if a specific value has been seen.
IPP Debug I2DaHash	4	10	The hash value for the packet. The setup of mask and compare is located in register Debug Counter I2DaHash Setup . This register enables the user to see if a specific value has been seen.
IPP Debug I2DaHashHitAndBucket	5	3	The L2 bucket used and hit bit. The setup of mask and compare is located in register Debug Counter I2DaHashHitAndBucket Setup . This register enables the user to see if a specific value has been seen.
IPP Debug I2DaHashHitAndBucket	5	3	The L2 bucket used and hit bit. The setup of mask and compare is located in register Debug Counter I2DaHashHitAndBucket Setup . This register enables the user to see if a specific value has been seen.
IPP Debug routerHit	6	1	The router was hit
IPP Debug nextHopPtrLpm	7	10	The LPM functions next hop pointer The setup of mask and compare is located in register Debug Counter nextHopPtrLpm Setup . This register enables the user to see if a specific value has been seen.
IPP Debug nextHopPtrHash	8	10	The L3 hash functions next hop pointer The setup of mask and compare is located in register Debug Counter nextHopPtrHash Setup . This register enables the user to see if a specific value has been seen.
IPP Debug nextHopPtrLpmHit	9	1	The LPM functions had a hit in the LPM table.
IPP Debug nextHopPtrHashHit	10	1	The L3 hash functions had a hit in the L3 hash table.



Register	Cnt Id	Bits	Description
IPP Debug nextHopPtrFinal	11	10	The final next hop pointer after ECMP and default route. The setup of mask and compare is located in register Debug Counter nextHopPtrFinal Setup . This register enables the user to see if a specific value has been seen.
IPP Debug srcPort	12	5	The source port which the packet came in on. The setup of mask and compare is located in register Debug Counter srcPort Setup . This register enables the user to see if a specific value has been seen.
IPP Debug dropPktAfterL2Decode	13	1	Packet was dropped after L2 packet decoder
IPP Debug nrVlans	14	2	The number of VLANs the incoming packet has The setup of mask and compare is located in register Debug Counter nrVlans Setup . This register enables the user to see if a specific value has been seen.
IPP Debug dropPktAfterL3Decode	15	1	Packet was dropped after L3 packet decoder
IPP Debug spVidOp	16	3	The Source port VID Operation The setup of mask and compare is located in register Debug Counter spVidOp Setup . This register enables the user to see if a specific value has been seen.
IPP Debug routed	17	1	The packet was routed
IPP Debug isFlooding	18	1	Was the packet flooded
IPP Debug isBroadcast	19	1	Was the packet broadcasted
IPP Debug doL2Lookup	20	1	This packet shall do lookup in L2 tables.
IPP Debug dstPortmask	21	32	The packets final portmask The setup of mask and compare is located in register Debug Counter dstPortmask Setup . This register enables the user to see if a specific value has been seen.
IPP Debug debugMatchIPP0	22	22	This allows a user to match all the above debug registers to make a counter update. This allows a user to update a counter based on multiple events happening for the same packet. The Cnt bit indicates which bit is in the bit-field. The setup of mask and compare is located in register Debug Counter debugMatchIPP0 Setup . This register enables the user to see if a specific value has been seen.

Table 33.1: IPP Debug List

33.2 Debug Counters in Egress Packet Processing

The Cnt Id field in the table below points to the counter to be updated in the counter bank of the **Debug EPP Counter** register.



Register	Cnt Id	Bits	Description
EPP Debug delSpecificVlan	0	1	This packet has a vid which shall be viewed as a priority VID and it will be deleted from the outgoing packet.
EPP Debug updateTosExp	1	1	This packet shall have a updated TOS/EXP field.
EPP Debug isIPv4	2	1	Packet is a IPv4 packet.
EPP Debug isIPv6	3	1	Packet is a IPv6 packet.
EPP Debug addNewMpls	4	1	Packet shall add a new MPLS header.
EPP Debug isPPPoE	5	1	Packet has a PPPoE header.
EPP Debug imActive	6	1	This packet shall be input mirrored.
EPP Debug imActive	6	1	This packet shall be input mirrored by sending out a second copy to the same destination port.
EPP Debug imExtra	7	1	This packet will send a extra input mirrored packet copy since the packet is already going out on this port.
EPP Debug omEnabled	8	1	This packet shall be output mirrored.
EPP Debug omImActive	9	1	This packet shall be both input mirrored and output mirrored.
EPP Debug reQueue	10	1	This packet shall be requeued.
EPP Debug reQueuePortId	11	3	This packet shall be requeued to this port. The setup of mask and compare is located in Debug Counter reQueuePortId Setup . This register enables the user to see if a specific value has been seen.
EPP Debug reQueuePkt	12	1	This packet will be requeued one more time since on the same port there shall be multiple copies.
EPP Debug fromPort	13	32	The port which the packet is going to be sent out on. The setup of mask and compare is located in Debug Counter fromPort Setup . This register enables the user to see if a specific value has been seen.
EPP Debug debugMatchEPP0	14	14	This allows a user to match all or part of them. This allows a user to update a counter based on multiple events happening for the same packet. The Cnt bit indicates which bit is in the bit-field. The setup of mask and compare is located in Debug Counter debugMatchEPP0 Setup . This register enables the user to see if a specific value has been seen.

Table 33.2: EPP Debug List





Chapter 34

Implementation

34.1 Floorplanning

The top of the core is the *pa_top* level, it wraps the switch core, *pa_top_switch*, and may also contain interface bridges.

TBD: Floorplanning info for multi-slice

34.1.1 Pipelining

The number of pipeline stages in the data paths between the floorplan blocks can be set freely when the RTL is generated. The same goes for the number of input flops and output flops on each floorplan block. If you need to change the number of pipeline stages it is a trivial task, but the RTL has to be re-generated. It cannot be adjusted in the existing verilog files.

Connection	Pipeline stages
SP ↔ IPP	1
IPP ↔ PB/BM	1
$IPP_x \leftrightarrow PB_y/BM_y$	0
PB ↔ BM	1
BM ↔ EPP	1
EPP ↔ PS	1

Table 34.1: The settings for pipeline flops between floorplan blocks

Floorplan block	Input flops	Output flops
SP	0	0
IPP	0	0
PB	0	1
BM	0	0
EPP	0	0
PS	1	1

Table 34.2: The settings for input and output flops for the floorplan blocks

The pipeline settings used when generating this core are shown in Table 34.1, and the input/output flops are listed in Table 34.2¹.

The $IPP_x \leftrightarrow PB_y/BM_y$ number is for busses going from an IPP in one slice to PB/BM in a different slice. To the number of pipeline stages between specific IPP and PB/BMs will be added $\text{abs}(x-y)$ times the number in the table above, where x and y are the slice numbers.

¹It should be noted that the input/output flops for the PS is not as clear cut as for the other blocks, due to the slightly more complex interface to the MAC.

34.1.2 Configuration and debug

The configuration and debug busses are in principle extremely flexible in how they can be pipelined. Flops can be added and removed anywhere so long as each bus is still in sync. This, as the other changes in pipelining, can only be done by generating new RTL.

34.2 Clock crossings

The bulk of the core is in a single clock domain, the core domain, driven by the *clk* clock. Each packet interface has separate clock domains for TX and RX. All paths between these domains are synchronized by either two synchronization flops, or by an asynchronous memory. The synchronization flops are always instantiations of the *verilog_sync_flops* verilog module, and the asynchronous memories are always instantiations of *verilog_memory_2c*.

34.2.1 IPP and EPP Structure

The IPP and EPP modules are both pipelines with a main dataflow from input to output. The floorplan is recommended to follow the pipeline dataflow. The logic input to a memory comes from the preceding pipeline stage and the output goes to the following pipeline stage. Which pipeline stage a specific memory belongs to is documented in the delivered files *eppp0_raw_opt.ramstat* and *ipp0_raw_opt.ramstat*.

In addition to the memory instances, the pipeline flipflops belonging to each pipeline stage is documented in *ipp0_raw_opt.fflist* and *eppp0_raw_opt.fflist*.

The exact Verilog instance names are not listed in these files but the names in the lists are part of the instance names and uniquely identify them.

In addition to the main dataflow there is also a configuration bus that has access to all memory instances and to the configuration registers. These paths are normally not in the critical path.

The configuration registers as opposed to the configuration memories can be accessed in multiple pipeline stages and therefore does not have a simple placement strategy.

34.3 Memory wrappers

The memories in the core are instantiated using the *verilog_memory.v* wrapper. It is expected that this wrapper is replaced, or modified, by the customer to instantiate appropriate memory macros. The macros needed are listed in Table 34.3. For memories with the *write_through* attribute set, simultaneous reading and writing the of same address is expected to yield the write data as read result. For memories with *write_through* set to 0 simultaneous reading and writing to the same address shall not occur.

type	width	depth	write through	write mask	input flops	output flops
dp	3	1024	1	None	0	0
dp	107	32	1	None	0	0
dp	118	32	1	None	0	0
dp	321	32	1	None	0	0
dp	206	16	1	None	0	0
dp	8	256	1	None	0	0
dp	471	512	1	None	0	0
dp	471	64	1	None	0	0
dp	140	16	1	None	0	0
dp	325	64	1	None	0	0
dp	189	24	1	None	0	0
dp	178	32	1	None	0	0
dp	176	4096	1	None	0	0
dp	152	512	1	None	0	0



dp	47	1024	1	None	0	0
dp	51	1024	1	None	0	0
dp	60	1024	1	None	0	0
dp	26	4128	1	None	0	0
dp	19	128	1	None	0	0
dp	189	256	1	None	0	0
dp	189	64	1	None	0	0
dp	92	32	1	None	0	0
dp	124	32	1	None	0	0
dp	2048	11	0	None	0	0
dp	122	26	0	None	0	0
dp	2048	14	0	None	0	0
dp	263	14	0	None	0	0
dp	180	26	0	None	0	0
dp	2048	8	0	None	0	0
dp	258	8	0	None	0	0
dp	258	39	0	None	0	0
dp	288	20	0	None	0	0
dp	128	17	0	None	0	0
dp	248	12	0	None	0	0
dp	246	64	0	None	0	0
dp	32	64	0	None	0	0
dp	249	20	0	None	0	0
dp	248	52	0	None	0	0
dp	128	37	0	None	0	0
dp	130	37	0	None	0	0
dp	96	51	0	None	0	0
dp	72	51	0	None	0	0
dp	48	72	0	None	0	0
dp	263	12	0	None	0	0
dp	257	50	0	None	0	0
dp	256	9	0	None	0	0
dp	127	21	0	None	0	0
dp	67	99	0	None	0	0
dp	34	99	0	None	0	0
dp	38	103	0	None	0	0
dp	96	103	0	None	0	0
dp	48	64	0	None	0	0
dp	48	65	0	None	0	0
dp	263	13	0	None	0	0
dp	134	52	0	None	0	0
dp	110	52	0	None	0	0
dp	263	11	0	None	0	0
dp	32	120	0	None	0	0
dp	102	32	0	None	0	0
dp	109	19	0	None	0	0
dp	48	62	0	None	0	0
dp	64	56	0	None	0	0
dp	48	46	0	None	0	0
dp	32	147	0	None	0	0
dp	49	108	0	None	0	0
dp	70	60	0	None	0	0
dp	232	153	0	None	0	0
dp	1816	153	0	None	0	0
dp	2417	34	0	None	0	0



dp	2048	8	1	None	0	0
dp	4	4100	1	None	0	0
dp	9	820	0	None	0	0
dp	23	4100	0	None	0	0
dp	4	4100	0	None	0	0
dp	82	4100	0	None	0	0
dp	14	4100	0	None	0	0
dp	33	4100	1	None	0	0
dp	2048	820	0	None	0	0
dp	10	820	1	None	0	0
dp	640	16	1	None	0	0
dp	48	1024	1	None	0	0
dp	28	1024	1	None	0	0
dp	136	16	1	None	0	0
dp	28	128	1	None	0	0
dp	51	2048	1	None	0	0
dp	10	512	1	None	0	0
dp	18	256	1	None	0	0
dp	9	256	1	None	0	0
dp	1	2048	1	None	0	0
dp	710	46	0	None	0	0
dp	197	12	0	None	0	0
dp	880	13	0	None	0	0
dp	2977	13	0	None	0	0
dp	160	18	0	None	0	0
dp	512	45	0	None	0	0
dp	64	42	0	None	0	0
dp	272	42	0	None	0	0
dp	416	45	0	None	0	0
dp	480	45	0	None	0	0
dp	256	42	0	None	0	0
dp	320	42	0	None	0	0
dp	53	46	0	None	0	0
dp	2904	28	0	None	0	0
dp	576	30	0	None	0	0
dp	320	28	0	None	0	0
dp	70	49	0	None	0	0
dp	240	28	0	None	0	0
dp	432	29	0	None	0	0
dp	3169	29	0	None	0	0
dp	192	29	0	None	0	0
dp	640	9	0	None	0	0
dp	69	30	0	None	0	0
dp	131	28	0	None	0	0
dp	11	200	0	None	0	0
dp	3007	202	0	None	0	0
dp	358	56	0	None	0	0
dc	13	16	0	None	0	0
dc	193	16	0	None	0	0
dc	230	16	0	None	0	0

Table 34.3: The memory macros needed for this core. Types: dp=two ports, one read and one write, running on the same clock. dc=two ports, one read and one write, with separate clocks for read and write.



For this design all dual-clock memories are generated as memory instances, but for synchronous memories only those with 2048 bits or more have been generated as a memory instance. Smaller synchronous memories are created as arrays of flops in the verilog source code. To change the criterium for making a memory as an instance or as an array of flops, new RTL has to be generated².

34.4 Dual ported memories

All memories are dual ported. Some dual-port memories have different clocks for the two ports, these are all instantiated using *verilog_memory_2c* wrapper. For these a real dual-port memory macro is the preferred choice. Most dual-port memories, however, are running on a single clock, and for these a better approach is to use a single-port memory macro clocked at twice the frequency. Unless, of course, the frequency would be prohibitively high. Note in the example timing diagram that the write is done in the first clock cycle to satisfy the *write_through* criterium. For memories that are not *write_through* it may be desirable for timing reasons to have the read in the first clock cycle.

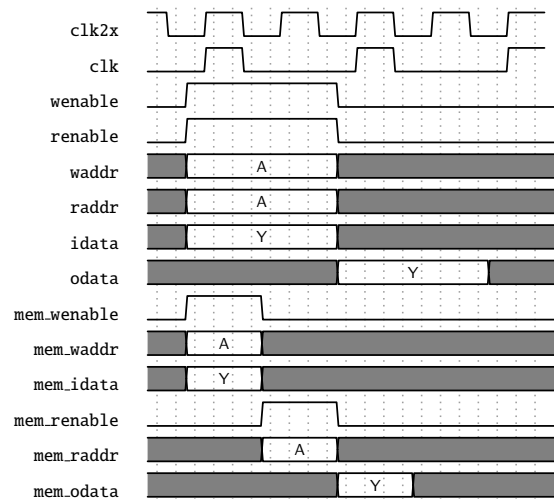


Figure 34.1: Timing diagram for a single ported memory used in the dual ported memory wrapper. In this case a concurrent read and write to the same address of a memory wrapper set for one cycle latency and with the write through attribute set.

There is no dedicated double frequency clock connected to the memories, it has to be provided using the **meminst_in* busses to the memory wrappers.

34.5 Memory timing

All memories in the design can be selected to have either:

- One cycle latency
- Two cycles latency, with the flop added on the input to the memory
- Two cycles latency, with the flop added on the output from the memory
- Three cycles latency, with flops added on both the input and the output

Which setting is used for each memory instance can be seen in the *input flops* and *output flops* columns of Table 34.3.

²Although, any instantiated memory wrapper can of course be left as is, and thus be implemented as an array of flops in synthesis.

34.6 Lint set up

For spyglass linting the following settings are assumed:

- `set_parameter ignore_local_variables yes`
- `set_parameter handle_zero_padding "W362"`

34.6.1 Waivers

Besides the inline waivers in the code these blanket waivers shall be applied:

- `waive -rule STARC05-2.11.3.1 -comment "Case statements are used in the sequential blocks of state-machines. This is not an issue"`
- `waive -rule STARC05-2.2.3.3 -comment "Flip-flops may be written several times in the same sequential block. This is not an issue"`
- `waive -regex -du "consistency_check.*" -rule "W240" -comment "consistency_check is guarded by SYNTHESIS, and is not used in hardware."`
- `waive -rule W415a -comment "Assigning multiple times in the same always block is a code style we use. This is not an issue"`
- `waive -rule W528 -comment "The way we pipeline will leave a lot of unread signals. This is not an issue"`

Chapter 35

Registers and Tables

Contents

35.1	Address Space For Tables and Registers	233
35.2	Byte Order	233
35.3	Register Banks	234
35.4	Registers and Tables in Alphabetical Order	268
35.5	Active Queue Manager	298
35.5.1	ERM Red Configuration	298
35.5.2	ERM Yellow Configuration	299
35.5.3	Egress Resource Manager Pointer	300
35.5.4	Resource Limiter Set	300
35.6	Core Information	301
35.6.1	Core Version	301
35.7	Egress Packet Processing	301
35.7.1	Beginning of Packet Tunnel Entry Instruction Table	301
35.7.2	Color Remap From Egress Port	302
35.7.3	Color Remap From Ingress Admission Control	303
35.7.4	Debug Counter debugMatchEPP0 Setup	303
35.7.5	Debug Counter fromPort Setup	304
35.7.6	Debug Counter reQueuePortId Setup	304
35.7.7	Disable CPU tag on CPU Port	305
35.7.8	Drain Port	305
35.7.9	EPP Debug addNewMpls	306
35.7.10	EPP Debug debugMatchEPP0	306
35.7.11	EPP Debug delSpecificVlan	306
35.7.12	EPP Debug fromPort	307
35.7.13	EPP Debug imActive	307
35.7.14	EPP Debug imExtra	307
35.7.15	EPP Debug isIPv4	308
35.7.16	EPP Debug isIPv6	308
35.7.17	EPP Debug isPPPoE	308
35.7.18	EPP Debug omEnabled	309
35.7.19	EPP Debug omImActive	309
35.7.20	EPP Debug reQueue	309
35.7.21	EPP Debug reQueuePkt	310
35.7.22	EPP Debug reQueuePortId	310
35.7.23	EPP Debug updateTosExp	310
35.7.24	Egress Ethernet Type for VLAN tag	311
35.7.25	Egress MPLS Decoding Options	311

35.7.26	Egress MPLS TTL Table	312
35.7.27	Egress Multiple Spanning Tree State	312
35.7.28	Egress NAT Operation	312
35.7.29	Egress Port Configuration	313
35.7.30	Egress Port VID Operation	316
35.7.31	Egress Queue To MPLS EXP Mapping Table	317
35.7.32	Egress Queue To PCP And CFI/DEI Mapping Table	318
35.7.33	Egress Router Table	318
35.7.34	Egress Tunnel Exit Table	318
35.7.35	Egress VLAN Translation TCAM	319
35.7.36	Egress VLAN Translation TCAM Answer	320
35.7.37	IP QoS Mapping Table	320
35.7.38	Ingress NAT Operation	321
35.7.39	L2 QoS Mapping Table	321
35.7.40	L2 Tunnel Entry Instruction Table	322
35.7.41	L3 Tunnel Entry Instruction Table	323
35.7.42	MPLS QoS Mapping Table	323
35.7.43	NAT Add Egress Port for NAT Calculation	324
35.7.44	Next Hop DA MAC	324
35.7.45	Next Hop MPLS Table	324
35.7.46	Next Hop Packet Insert MPLS Header	325
35.7.47	Output Mirroring Table	327
35.7.48	Router Port Egress SA MAC Address	327
35.7.49	Select Which Egress QoS Mapping Table To Use	328
35.7.50	TOS QoS Mapping Table	328
35.7.51	Tunnel Entry Header Data	329
35.7.52	Tunnel Entry Instruction Table	330
35.8	Flow Control	330
35.8.1	FFA Used PFC	330
35.8.2	FFA Used non-PFC	331
35.8.3	PFC Dec Counters for ingress ports 0 to 7	331
35.8.4	PFC Dec Counters for ingress ports 16 to 23	332
35.8.5	PFC Dec Counters for ingress ports 24 to 31	332
35.8.6	PFC Dec Counters for ingress ports 8 to 15	332
35.8.7	PFC Inc Counters for ingress ports 0 to 7	333
35.8.8	PFC Inc Counters for ingress ports 16 to 23	333
35.8.9	PFC Inc Counters for ingress ports 24 to 31	334
35.8.10	PFC Inc Counters for ingress ports 8 to 15	334
35.8.11	Port FFA Used	334
35.8.12	Port Pause Settings	335
35.8.13	Port Reserved	335
35.8.14	Port Tail-Drop FFA Threshold	336
35.8.15	Port Tail-Drop Settings	336
35.8.16	Port Used	337
35.8.17	Port Xoff FFA Threshold	337
35.8.18	Port Xon FFA Threshold	338
35.8.19	Port/TC Reserved	338
35.8.20	Port/TC Tail-Drop Total Threshold	338
35.8.21	Port/TC Xoff Total Threshold	339
35.8.22	Port/TC Xon Total Threshold	339
35.8.23	TC FFA Used	340
35.8.24	TC Tail-Drop FFA Threshold	340
35.8.25	TC Xoff FFA Threshold	341



35.8.26	TC Xon FFA Threshold	341
35.8.27	Tail-Drop FFA Threshold	342
35.8.28	Xoff FFA Threshold	342
35.8.29	Xon FFA Threshold	342
35.9	Global Configuration	343
35.9.1	Core Tick Configuration	343
35.9.2	Core Tick Select	343
35.9.3	MAC RX Maximum Packet Length	344
35.9.4	Scratch	344
35.10	Ingress Packet Processing	344
35.10.1	AH Header Packet Decoder Options	344
35.10.2	ARP Packet Decoder Options	345
35.10.3	Aging Data FIFO	345
35.10.4	Aging Data FIFO High Watermark Level	346
35.10.5	Allow Special Frame Check For L2 Action Table	346
35.10.6	BOOTP and DHCP Packet Decoder Options	348
35.10.7	CAPWAP Packet Decoder Options	349
35.10.8	CPU Reason Code Operation	349
35.10.9	Check IPv4 Header Checksum	350
35.10.10	DNS Packet Decoder Options	351
35.10.11	Debug Counter debugMatchIPPO Setup	351
35.10.12	Debug Counter dstPortmask Setup	352
35.10.13	Debug Counter finalVid Setup	352
35.10.14	Debug Counter l2DaHash Setup	352
35.10.15	Debug Counter l2DaHashHitAndBucket Setup	353
35.10.16	Debug Counter l2DaHashKey Setup	353
35.10.17	Debug Counter l2DaTcamHitsAndCast Setup	354
35.10.18	Debug Counter nextHopPtrFinal Setup	354
35.10.19	Debug Counter nextHopPtrHash Setup	355
35.10.20	Debug Counter nextHopPtrLpm Setup	355
35.10.21	Debug Counter nrVlans Setup	356
35.10.22	Debug Counter spVidOp Setup	356
35.10.23	Debug Counter srcPort Setup	357
35.10.24	Debug Counter vlanVidOp Setup	357
35.10.25	Default Packet To CPU Modification	357
35.10.26	ESP Header Packet Decoder Options	358
35.10.27	Egress ACL Rule Pointer TCAM	358
35.10.28	Egress ACL Rule Pointer TCAM Answer	359
35.10.29	Egress Configurable ACL 0 Large Table	360
35.10.30	Egress Configurable ACL 0 Rules Setup	361
35.10.31	Egress Configurable ACL 0 Search Mask	362
35.10.32	Egress Configurable ACL 0 Selection	362
35.10.33	Egress Configurable ACL 0 Small Table	363
35.10.34	Egress Configurable ACL 0 TCAM	364
35.10.35	Egress Configurable ACL 0 TCAM Answer	365
35.10.36	Egress Configurable ACL 1 Rules Setup	366
35.10.37	Egress Configurable ACL 1 TCAM	366
35.10.38	Egress Configurable ACL 1 TCAM Answer	367
35.10.39	Egress Port NAT State	368
35.10.40	Egress Spanning Tree State	368
35.10.41	Enable Enqueue To Ports And Queues	368
35.10.42	Flooding Action Send to Port	369
35.10.43	Force Non VLAN Packet To Specific Color	369



35.10.44	Force Non VLAN Packet To Specific Queue	370
35.10.45	Force Unknown L3 Packet To Specific Color	370
35.10.46	Force Unknown L3 Packet To Specific Egress Queue	371
35.10.47	Forward From CPU	371
35.10.48	GRE Packet Decoder Options	371
35.10.49	Hairpin Enable	372
35.10.50	Hardware Learning Configuration	372
35.10.51	Hardware Learning Counter	373
35.10.52	Hash Based L3 Routing Table	373
35.10.53	Hit Update Data FIFO	374
35.10.54	Hit Update Data FIFO High Watermark Level	375
35.10.55	IEEE 1588 L2 Packet Decoder Options	375
35.10.56	IEEE 1588 L4 Packet Decoder Options	376
35.10.57	IEEE 802.1X and EAPOL Packet Decoder Options	377
35.10.58	IKE Packet Decoder Options	377
35.10.59	IPP Debug debugMatchIPP0	378
35.10.60	IPP Debug doL2Lookup	378
35.10.61	IPP Debug dropPktAfterL2Decode	379
35.10.62	IPP Debug dropPktAfterL3Decode	379
35.10.63	IPP Debug dstPortmask	379
35.10.64	IPP Debug finalVid	380
35.10.65	IPP Debug isBroadcast	380
35.10.66	IPP Debug isFlooding	380
35.10.67	IPP Debug l2DaHash	381
35.10.68	IPP Debug l2DaHashHitAndBucket	381
35.10.69	IPP Debug l2DaHashKey	381
35.10.70	IPP Debug l2DaTcamHitsAndCast	382
35.10.71	IPP Debug nextHopPtrFinal	382
35.10.72	IPP Debug nextHopPtrHash	382
35.10.73	IPP Debug nextHopPtrHashHit	383
35.10.74	IPP Debug nextHopPtrLpm	383
35.10.75	IPP Debug nextHopPtrLpmHit	383
35.10.76	IPP Debug nrVlans	384
35.10.77	IPP Debug routed	384
35.10.78	IPP Debug routerHit	384
35.10.79	IPP Debug spVidOp	385
35.10.80	IPP Debug srcPort	385
35.10.81	IPP Debug vlanVidOp	385
35.10.82	IPv4 TOS Field To Egress Queue Mapping Table	386
35.10.83	IPv4 TOS Field To Packet Color Mapping Table	386
35.10.84	IPv6 Class of Service Field To Egress Queue Mapping Table	386
35.10.85	IPv6 Class of Service Field To Packet Color Mapping Table	387
35.10.86	Ingress Admission Control Current Status	387
35.10.87	Ingress Admission Control Initial Pointer	387
35.10.88	Ingress Admission Control Mark All Red	388
35.10.89	Ingress Admission Control Mark All Red Enable	388
35.10.90	Ingress Admission Control Reset	389
35.10.91	Ingress Admission Control Token Bucket Configuration	389
35.10.92	Ingress Configurable ACL 0 Large Table	390
35.10.93	Ingress Configurable ACL 0 Pre Lookup	393
35.10.94	Ingress Configurable ACL 0 Rules Setup	393
35.10.95	Ingress Configurable ACL 0 Search Mask	394
35.10.96	Ingress Configurable ACL 0 Selection	394



35.10.97	Ingress Configurable ACL 0 Small Table	395
35.10.98	Ingress Configurable ACL 0 TCAM	397
35.10.99	Ingress Configurable ACL 0 TCAM Answer	398
35.10.100	Ingress Configurable ACL 1 Large Table	400
35.10.101	Ingress Configurable ACL 1 Pre Lookup	404
35.10.102	Ingress Configurable ACL 1 Rules Setup	404
35.10.103	Ingress Configurable ACL 1 Search Mask	405
35.10.104	Ingress Configurable ACL 1 Selection	405
35.10.105	Ingress Configurable ACL 1 Small Table	406
35.10.106	Ingress Configurable ACL 1 TCAM	410
35.10.107	Ingress Configurable ACL 1 TCAM Answer	410
35.10.108	Ingress Configurable ACL 2 Pre Lookup	413
35.10.109	Ingress Configurable ACL 2 Rules Setup	414
35.10.110	Ingress Configurable ACL 2 TCAM	414
35.10.111	Ingress Configurable ACL 2 TCAM Answer	415
35.10.112	Ingress Configurable ACL 3 Rules Setup	418
35.10.113	Ingress Configurable ACL 3 TCAM	418
35.10.114	Ingress Configurable ACL 3 TCAM Answer	418
35.10.115	Ingress Drop Options	419
35.10.116	Ingress Egress Port Packet Type Filter	420
35.10.117	Ingress Ethernet Type for VLAN tag	422
35.10.118	Ingress MMP Drop Mask	423
35.10.119	Ingress Multiple Spanning Tree State	423
35.10.120	Ingress Port Packet Type Filter	424
35.10.121	Ingress Router Table	425
35.10.122	Ingress VID Ethernet Type Range Assignment Answer	427
35.10.123	Ingress VID Ethernet Type Range Search Data	427
35.10.124	Ingress VID Inner VID Range Assignment Answer	428
35.10.125	Ingress VID Inner VID Range Search Data	428
35.10.126	Ingress VID MAC Range Assignment Answer	429
35.10.127	Ingress VID MAC Range Search Data	429
35.10.128	Ingress VID Outer VID Range Assignment Answer	430
35.10.129	Ingress VID Outer VID Range Search Data	430
35.10.130	L2 Action Table	430
35.10.131	L2 Action Table Egress Port State	432
35.10.132	L2 Action Table Source Port	432
35.10.133	L2 Aging Collision Shadow Table	434
35.10.134	L2 Aging Collision Table	434
35.10.135	L2 Aging Status Shadow Table	435
35.10.136	L2 Aging Status Shadow Table - Replica	435
35.10.137	L2 Aging Table	435
35.10.138	L2 DA Hash Lookup Table	436
35.10.139	L2 Destination Table	436
35.10.140	L2 Destination Table - Replica	437
35.10.141	L2 Lookup Collision Table	438
35.10.142	L2 Lookup Collision Table Masks	438
35.10.143	L2 Multicast Handling	439
35.10.144	L2 Multicast Table	439
35.10.145	L2 Reserved Multicast Address Action	440
35.10.146	L2 Reserved Multicast Address Base	440
35.10.147	L2 SA Hash Lookup Table	441
35.10.148	L2 Tunnel Decoder Setup	441
35.10.149	L3 LPM Result	442



35.10.150	L3 Routing Default	442
35.10.151	L3 Routing TCAM	443
35.10.152	LACP Packet Decoder Options	444
35.10.153	LLDP Configuration	445
35.10.154	Learning And Aging Enable	445
35.10.155	Learning And Aging Writeback Control	446
35.10.156	Learning Conflict	447
35.10.157	Learning DA MAC	447
35.10.158	Learning Data FIFO	447
35.10.159	Learning Data FIFO High Watermark Level	448
35.10.160	Learning Overflow	448
35.10.161	Link Aggregate Weight	449
35.10.162	Link Aggregation Ctrl	449
35.10.163	Link Aggregation Membership	450
35.10.164	Link Aggregation To Physical Ports Members	450
35.10.165	MPLS EXP Field To Egress Queue Mapping Table	451
35.10.166	MPLS EXP Field To Packet Color Mapping Table	451
35.10.167	NAT Action Table	452
35.10.168	NAT Action Table Force Original Packet	452
35.10.169	Next Hop Packet Modifications	453
35.10.170	Next Hop Table	454
35.10.171	Port Move Options	455
35.10.172	RARP Packet Decoder Options	456
35.10.173	Reserved Destination MAC Address Range	456
35.10.174	Reserved Source MAC Address Range	457
35.10.175	Router Egress Queue To VLAN Data	458
35.10.176	Router MTU Table	459
35.10.177	Router Port MAC Address	459
35.10.178	SCTP Packet Decoder Options	460
35.10.179	SMON Set Search	460
35.10.180	SNAP LLC Decoding Options	461
35.10.181	Second Tunnel Exit Lookup TCAM	461
35.10.182	Second Tunnel Exit Lookup TCAM Answer	462
35.10.183	Second Tunnel Exit Miss Action	463
35.10.184	Send to CPU	463
35.10.185	Software Aging Enable	464
35.10.186	Software Aging Start Latch	464
35.10.187	Source Port Default ACL Action	465
35.10.188	Source Port Table	467
35.10.189	Time to Age	474
35.10.190	Tunnel Entry MTU Length Check	474
35.10.191	Tunnel Exit Lookup TCAM	475
35.10.192	Tunnel Exit Lookup TCAM Answer	476
35.10.193	VLAN PCP And DEI To Color Mapping Table	477
35.10.194	VLAN PCP To Queue Mapping Table	477
35.10.195	VLAN Table	478
35.11	MBSC	482
35.11.1	L2 Broadcast Storm Control Bucket Capacity Configuration	482
35.11.2	L2 Broadcast Storm Control Bucket Threshold Configuration	482
35.11.3	L2 Broadcast Storm Control Enable	482
35.11.4	L2 Broadcast Storm Control Rate Configuration	483
35.11.5	L2 Flooding Storm Control Bucket Capacity Configuration	483
35.11.6	L2 Flooding Storm Control Bucket Threshold Configuration	483



35.11.7	L2 Flooding Storm Control Enable	484
35.11.8	L2 Flooding Storm Control Rate Configuration	484
35.11.9	L2 Multicast Storm Control Bucket Capacity Configuration	484
35.11.10	L2 Multicast Storm Control Bucket Threshold Configuration	485
35.11.11	L2 Multicast Storm Control Enable	485
35.11.12	L2 Multicast Storm Control Rate Configuration	485
35.12	Scheduling	486
35.12.1	DWRR Bucket Capacity Configuration	486
35.12.2	DWRR Bucket Misc Configuration	486
35.12.3	DWRR Weight Configuration	487
35.12.4	Map Queue to Priority	487
35.12.5	Output Disable	488
35.13	Shapers	488
35.13.1	Port Shaper Bucket Capacity Configuration	488
35.13.2	Port Shaper Bucket Threshold Configuration	489
35.13.3	Port Shaper Enable	489
35.13.4	Port Shaper Rate Configuration	489
35.13.5	Prio Shaper Bucket Capacity Configuration	490
35.13.6	Prio Shaper Bucket Threshold Configuration	490
35.13.7	Prio Shaper Enable	491
35.13.8	Prio Shaper Rate Configuration	491
35.13.9	Queue Shaper Bucket Capacity Configuration	492
35.13.10	Queue Shaper Bucket Threshold Configuration	492
35.13.11	Queue Shaper Enable	492
35.13.12	Queue Shaper Rate Configuration	493
35.14	Shared Buffer Memory	493
35.14.1	Buffer Free	493
35.14.2	Egress Port Depth	494
35.14.3	Egress Queue Depth	494
35.14.4	Minimum Buffer Free	495
35.14.5	Packet Buffer Status	495
35.15	Statistics: ACL	495
35.15.1	Egress Configurable ACL Match Counter	495
35.15.2	Ingress Configurable ACL Match Counter	496
35.16	Statistics: Debug	496
35.16.1	Debug EPP Counter	496
35.16.2	Debug IPP Counter	497
35.16.3	EPP PM Drop	497
35.16.4	IPP PM Drop	497
35.16.5	PS Error Counter	498
35.16.6	SP Overflow Drop	498
35.17	Statistics: EPP Egress Port Drop	499
35.17.1	Egress Port Disabled Drop	499
35.17.2	Egress Port Filtering Drop	499
35.17.3	Tunnel Exit Too Small Packet Modification To Small Drop	500
35.17.4	Unknown Egress Drop	500
35.18	Statistics: IPP Egress Port Drop	500
35.18.1	Egress Spanning Tree Drop	500
35.18.2	Ingress-Egress Packet Filtering Drop	501
35.18.3	L2 Action Table Per Port Drop	501
35.18.4	MBSC Drop	502
35.18.5	Queue Off Drop	502
35.19	Statistics: IPP Ingress Port Drop	502



35.19.1	AH Decoder Drop	502
35.19.2	ARP Decoder Drop	503
35.19.3	BOOTP and DHCP Decoder Drop	503
35.19.4	CAPWAP Decoder Drop	504
35.19.5	DNS Decoder Drop	504
35.19.6	ESP Decoder Drop	504
35.19.7	Egress Configurable ACL Drop	505
35.19.8	Empty Mask Drop	505
35.19.9	Expired TTL Drop	505
35.19.10	GRE Decoder Drop	506
35.19.11	IEEE 802.1X and EAPOL Decoder Drop	506
35.19.12	IKE Decoder Drop	507
35.19.13	IP Checksum Drop	507
35.19.14	Ingress Configurable ACL Drop	507
35.19.15	Ingress Packet Filtering Drop	508
35.19.16	Ingress Spanning Tree Drop: Blocking	508
35.19.17	Ingress Spanning Tree Drop: Learning	508
35.19.18	Ingress Spanning Tree Drop: Listen	509
35.19.19	Invalid Routing Protocol Drop	509
35.19.20	L2 Action Table Drop	510
35.19.21	L2 Action Table Port Move Drop	510
35.19.22	L2 Action Table Special Packet Type Drop	510
35.19.23	L2 IEEE 1588 Decoder Drop	511
35.19.24	L2 Lookup Drop	511
35.19.25	L2 Reserved Multicast Address Drop	512
35.19.26	L3 Lookup Drop	512
35.19.27	L4 IEEE 1588 Decoder Drop	512
35.19.28	LACP Decoder Drop	513
35.19.29	Learning Packet Drop	513
35.19.30	Maximum Allowed VLAN Drop	513
35.19.31	Minimum Allowed VLAN Drop	514
35.19.32	NAT Action Table Drop	514
35.19.33	RARP Decoder Drop	515
35.19.34	Reserved MAC DA Drop	515
35.19.35	Reserved MAC SA Drop	515
35.19.36	SCTP Decoder Drop	516
35.19.37	Second Tunnel Exit Drop	516
35.19.38	Source Port Default ACL Action Drop	516
35.19.39	Tunnel Exit Miss Action Drop	517
35.19.40	Tunnel Exit Too Small Packet Modification Drop	517
35.19.41	Unknown Ingress Drop	518
35.19.42	VLAN Member Drop	518
35.20	Statistics: IPP Ingress Port Receive	518
35.20.1	IP Multicast ACL Drop Counter	518
35.20.2	IP Multicast Received Counter	519
35.20.3	IP Multicast Routed Counter	519
35.20.4	IP Unicast Received Counter	520
35.20.5	IP Unicast Routed Counter	520
35.21	Statistics: Misc	521
35.21.1	Buffer Overflow Drop	521
35.21.2	Drain Port Drop	521
35.21.3	Egress Resource Manager Drop	521
35.21.4	Flow Classification And Metering Drop	522



35.21.5	IPP Empty Destination Drop	522
35.21.6	Ingress Resource Manager Drop	523
35.21.7	MAC RX Broken Packets	523
35.21.8	MAC RX Long Packet Drop	523
35.21.9	MAC RX Short Packet Drop	524
35.21.10	Re-queue Overflow Drop	524
35.22	Statistics: NAT	524
35.22.1	Egress NAT Hit Status	524
35.22.2	Ingress NAT Hit Status	525
35.23	Statistics: Packet Datapath	525
35.23.1	EPP Packet Head Counter	525
35.23.2	EPP Packet Tail Counter	526
35.23.3	IPP Packet Head Counter	526
35.23.4	IPP Packet Tail Counter	526
35.23.5	MAC Interface Counters For RX	527
35.23.6	MAC Interface Counters For TX	527
35.23.7	PB Packet Head Counter	527
35.23.8	PB Packet Tail Counter	528
35.23.9	PS Packet Head Counter	528
35.23.10	PS Packet Tail Counter	529
35.24	Statistics: Routing	529
35.24.1	Next Hop Hit Status	529
35.24.2	Received Packets on Ingress VRF	529
35.24.3	Transmitted Packets on Egress VRF	530
35.25	Statistics: SMON	530
35.25.1	SMON Set 0 Byte Counter	530
35.25.2	SMON Set 0 Packet Counter	531
35.25.3	SMON Set 1 Byte Counter	531
35.25.4	SMON Set 1 Packet Counter	532
35.25.5	SMON Set 2 Byte Counter	532
35.25.6	SMON Set 2 Packet Counter	532
35.25.7	SMON Set 3 Byte Counter	533
35.25.8	SMON Set 3 Packet Counter	533

All registers and tables that are accessible from a configuration interface are listed in this chapter. A user guide for the configuration interface is found in Chapter 32, and the pins for the configuration interfaces are described in Section 31.3.

35.1 Address Space For Tables and Registers

All tables in the address space are linear. The size of a table entry is always rounded up to nearest power of two of the bus width. For example if the bus is 64 bits and a entry in a table is 65 bits wide, it will then use two addresses per entry. Second example, the bus is still 64 bits, but the entry is 341 bits wide, the entry will then use a address space of 8 addresses per table entry (341 bits fits within 6 bus words but is rounded up to nearest power of two). This is shown in figure 35.1. The total address space used by this core is 395531 addresses.

35.2 Byte Order

When a register field is wider than a byte and the field represents an integer value or the field is related to a packet header field, the order of the bytes needs to be defined.

Integer fields in the registers have a little endian byte order so that the lowest bits in a field will be at lowest bits on the configuration bus. When a field spans multiple configuration bus addresses the lowest



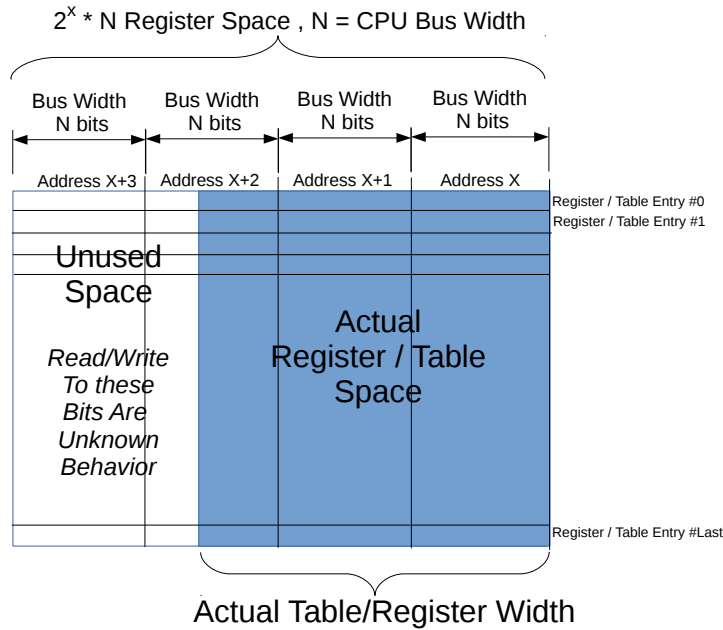


Figure 35.1: Address space usage by tables

address will hold the lowest bits of the field. If this is memory mapped and accessed by a host CPU it will be in the correct byte order for a little endian CPU.

In network byte order the first transmitted or received byte has byte number 0. One example is the Ethernet MAC address with the printed representation *a1-b2-c3-d4-e5-f6* where *a1* would be sent first and would be byte 0). When used in a register field the highest bits in the register field corresponds to the lowest network byte. Therefore the MAC address above would be the value *0xa1b2c3d4e5f6* and as seen by a little endian host CPU the byte *0xf6* would be at the lowest address.

A special case are IPv6 addresses. In the standard printed representation *0102:0304:0506:...* the leftmost byte *01* is byte 0 in the network order followed by byte *02* as network byte 1. When configuring this in a register field the lowest bytes are from the lowest network byte numbers. However each pair of bytes are also swapped. The address above would therefore be the value *0x...050603040102*.

35.3 Register Banks

A bank is a hardware unit which holds a number of registers or a single table. In a bank containing data wider than 64 bits, registers (or table entries) must be accessed one at a time, or the accesses will interfere with each other.

Bank Name	Connected Registers or Tables
switch_info_regbank	Core Version
top_regs	Buffer Free Scratch Core Tick Configuration Core Tick Select
pa top switch mactop iRxedgcheck iProtocolcheck0 table	MAC Interface Counters For RX
pa top switch mactop iRxedgcheck iProtocolcheck1 table	MAC Interface Counters For RX
pa top switch mactop iRxedgcheck iProtocolcheck2 table	MAC Interface Counters For RX



Bank Name	Connected Registers or Tables
pa top switch mactop iRxedgecheck iProtocolcheck3 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck4 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck5 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck6 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck7 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck8 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck9 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck10 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck11 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck12 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck13 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck14 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck15 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck16 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck17 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck18 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck19 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck20 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck21 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck22 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck23 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck24 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck25 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck26 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck27 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck28 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck29 table	MAC Interface Counters For RX



Bank Name	Connected Registers or Tables
pa top switch mactop iRxedgecheck iProtocolcheck30 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck31 table	MAC Interface Counters For RX
rx_length_ref	MAC RX Maximum Packet Length[0..31]
rx_length_drop	MAC RX Broken Packets[0..31] MAC RX Short Packet Drop[0..31] MAC RX Long Packet Drop[0..31]
pa top switch mactop iTxedgecheck iProtocolcheck0 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck1 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck2 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck3 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck4 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck5 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck6 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck7 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck8 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck9 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck10 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck11 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck12 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck13 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck14 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck15 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck16 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck17 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck18 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck19 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck20 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck21 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck22 table	MAC Interface Counters For TX



Bank Name	Connected Registers or Tables
pa top switch mactop iTxedgecheck iProtocolcheck23 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck24 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck25 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck26 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck27 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck28 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck29 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck30 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck31 table	MAC Interface Counters For TX
l2_broadcast_storm_control_rate_settings	L2 Broadcast Storm Control Rate Configuration
l2_broadcast_storm_control_bucket_settings	L2 Broadcast Storm Control Bucket Capacity Configuration L2 Broadcast Storm Control Bucket Threshold Configuration
l2_broadcast_storm_control_misc	L2 Broadcast Storm Control Enable
l2_multicast_storm_control_rate_settings	L2 Multicast Storm Control Rate Configuration
l2_multicast_storm_control_bucket_settings	L2 Multicast Storm Control Bucket Capacity Configuration L2 Multicast Storm Control Bucket Threshold Configuration
l2_multicast_storm_control_misc	L2 Multicast Storm Control Enable
l2_flooding_storm_control_rate_settings	L2 Flooding Storm Control Rate Configuration
l2_flooding_storm_control_bucket_settings	L2 Flooding Storm Control Bucket Capacity Configuration L2 Flooding Storm Control Bucket Threshold Configuration
l2_flooding_storm_control_misc	L2 Flooding Storm Control Enable
le_ae_status	Learning Conflict Learning Overflow
le_ae_control	Learning And Aging Enable Software Aging Enable Learning And Aging Writeback Control Learning Data FIFO High Watermark Level Aging Data FIFO High Watermark Level Hit Update Data FIFO High Watermark Level Hardware Learning Configuration[0..31] Time to Age
age_cam_register_bank	L2 Aging Collision Table[0..31]
mac_cnt_register_bank	Hardware Learning Counter[0..31]
L2 Aging Table	L2 Aging Table
le_ae_control_latch	Software Aging Start Latch
ldf	Learning Data FIFO
adf	Aging Data FIFO
hdf	Hit Update Data FIFO
count_sp_ss0	SP Overflow Drop
count_broken_pkt_ss0	IPP PM Drop IPP Empty Destination Drop
count_pa top switch ipp0 conf	Unknown Ingress Drop Empty Mask Drop Ingress Spanning Tree Drop: Listen Ingress Spanning Tree Drop: Learning Ingress Spanning Tree Drop: Blocking



Bank Name	Connected Registers or Tables
	L2 Lookup Drop Ingress Packet Filtering Drop Reserved MAC DA Drop Reserved MAC SA Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop Invalid Routing Protocol Drop Expired TTL Drop L3 Lookup Drop IP Checksum Drop Second Tunnel Exit Drop Tunnel Exit Miss Action Drop Tunnel Exit Too Small Packet Modification Drop Learning Packet Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Egress Configurable ACL Drop ARP Decoder Drop RARP Decoder Drop L2 IEEE 1588 Decoder Drop L4 IEEE 1588 Decoder Drop IEEE 802.1X and EAPOL Decoder Drop SCTP Decoder Drop LACP Decoder Drop AH Decoder Drop ESP Decoder Drop DNS Decoder Drop BOOTP and DHCP Decoder Drop CAPWAP Decoder Drop IKE Decoder Drop GRE Decoder Drop NAT Action Table Drop L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop Source Port Default ACL Action Drop
count_opkt_pa top switch ipp0 conf	IPP Packet Head Counter IPP Packet Tail Counter
Tunnel Exit Lookup TCAM Answer Slice 0	Tunnel Exit Lookup TCAM Answer Slice 0
Ingress Admission Control Initial Pointer Slice 0	Ingress Admission Control Initial Pointer Slice 0
Ingress Configurable ACL 0 Large Table Slice 0	Ingress Configurable ACL 0 Large Table Slice 0
Ingress Configurable ACL 0 Small Table Slice 0	Ingress Configurable ACL 0 Small Table Slice 0
Ingress Configurable ACL 0 TCAM Answer Slice 0	Ingress Configurable ACL 0 TCAM Answer Slice 0
Ingress Configurable ACL 1 Large Table Slice 0	Ingress Configurable ACL 1 Large Table Slice 0
Ingress Configurable ACL 1 Small Table Slice 0	Ingress Configurable ACL 1 Small Table Slice 0
Ingress Configurable ACL 1 TCAM Answer Slice 0	Ingress Configurable ACL 1 TCAM Answer Slice 0



Bank Name	Connected Registers or Tables
Ingress Configurable ACL 2 TCAM Answer Slice 0	Ingress Configurable ACL 2 TCAM Answer Slice 0
Ingress Configurable ACL 3 TCAM Answer Slice 0	Ingress Configurable ACL 3 TCAM Answer Slice 0
Source Port Default ACL Action Slice 0	Source Port Default ACL Action Slice 0
VLAN Table Slice 0	VLAN Table Slice 0
Ingress Multiple Spanning Tree State Slice 0	Ingress Multiple Spanning Tree State Slice 0
Ingress Router Table Slice 0	Ingress Router Table Slice 0
L3 LPM Result Slice 0	L3 LPM Result Slice 0
Hash Based L3 Routing Table Slice 0	Hash Based L3 Routing Table Slice 0
Next Hop Table Slice 0	Next Hop Table Slice 0
Next Hop Packet Modifications Slice 0	Next Hop Packet Modifications Slice 0
L2 Aging Status Shadow Table Slice 0	L2 Aging Status Shadow Table Slice 0
L2 DA Hash Lookup Table Slice 0	L2 DA Hash Lookup Table Slice 0
L2 Destination Table Slice 0	L2 Destination Table Slice 0
L2 SA Hash Lookup Table Slice 0	L2 SA Hash Lookup Table Slice 0
L2 Aging Status Shadow Table - Replica Slice 0	L2 Aging Status Shadow Table - Replica Slice 0
L2 Destination Table - Replica Slice 0	L2 Destination Table - Replica Slice 0
L2 Action Table Slice 0	L2 Action Table Slice 0
L2 Action Table Source Port Slice 0	L2 Action Table Source Port Slice 0
Egress ACL Rule Pointer TCAM Answer Slice 0	Egress ACL Rule Pointer TCAM Answer Slice 0
Egress Configurable ACL 0 Large Table Slice 0	Egress Configurable ACL 0 Large Table Slice 0
Egress Configurable ACL 0 Small Table Slice 0	Egress Configurable ACL 0 Small Table Slice 0
Egress Configurable ACL 0 TCAM Answer Slice 0	Egress Configurable ACL 0 TCAM Answer Slice 0
Egress Configurable ACL 1 TCAM Answer Slice 0	Egress Configurable ACL 1 TCAM Answer Slice 0
Tunnel Entry MTU Length Check Slice 0	Tunnel Entry MTU Length Check Slice 0
ipp_register_bank_ss0	Link Aggregation Ctrl Debug Counter srcPort Setup L2 Tunnel Decoder Setup Learning DA MAC SNAP LLC Decoding Options Ingress Ethernet Type for VLAN tag L2 Reserved Multicast Address Base Debug Counter nrVlans Setup Debug Counter spVidOp Setup Ingress Configurable ACL 0 Selection Ingress Configurable ACL 1 Selection Debug Counter finalVid Setup Debug Counter vlanVidOp Setup Debug Counter nextHopPtrLpm Setup Debug Counter nextHopPtrHash Setup Debug Counter nextHopPtrFinal Setup Check IPv4 Header Checksum Force Non VLAN Packet To Specific Queue Force Unknown L3 Packet To Specific Egress Queue Force Non VLAN Packet To Specific Color Force Unknown L3 Packet To Specific Color Debug Counter l2DaHash Setup



Bank Name	Connected Registers or Tables
	Debug Counter l2DaHashHitAndBucket Setup Debug Counter l2DaTcamHitsAndCast Setup Forward From CPU Port Move Options L2 Action Table Egress Port State L2 Multicast Handling Egress Configurable ACL 0 Selection Egress Port NAT State NAT Action Table Force Original Packet Ingress MMP Drop Mask Debug Counter dstPortmask Setup Debug Counter debugMatchIPP0 Setup IPP Debug srcPort IPP Debug dropPktAfterL2Decode IPP Debug nrVlans IPP Debug dropPktAfterL3Decode IPP Debug spVidOp IPP Debug finalVid IPP Debug vlanVidOp IPP Debug routerHit IPP Debug nextHopPtrLpm IPP Debug nextHopPtrHash IPP Debug nextHopPtrLpmHit IPP Debug nextHopPtrHashHit IPP Debug nextHopPtrFinal IPP Debug l2DaHashKey IPP Debug l2DaHash IPP Debug l2DaHashHitAndBucket IPP Debug l2DaTcamHitsAndCast IPP Debug routed IPP Debug isFlooding IPP Debug isBroadcast IPP Debug doL2Lookup IPP Debug dstPortmask IPP Debug debugMatchIPP0 Enable Enqueue To Ports And Queues Flooding Action Send to Port Link Aggregation To Physical Ports Members Link Aggregate Weight CPU Reason Code Operation Ingress Egress Port Packet Type Filter NAT Action Table Egress Configurable ACL 1 Rules Setup Egress Configurable ACL 0 Rules Setup Allow Special Frame Check For L2 Action Table Egress Multiple Spanning Tree State Router MTU Table Hairpin Enable L2 Multicast Table L2 Lookup Collision Table Masks L2 Lookup Collision Table L2 Aging Collision Shadow Table Router Egress Queue To VLAN Data MPLS EXP Field To Packet Color Mapping Table IPv6 Class of Service Field To Packet Color Mapping Table IPv4 TOS Field To Packet Color Mapping Table



Bank Name	Connected Registers or Tables
	VLAN PCP And DEI To Color Mapping Table MPLS EXP Field To Egress Queue Mapping Table IPv6 Class of Service Field To Egress Queue Mapping Table IPv4 TOS Field To Egress Queue Mapping Table VLAN PCP To Queue Mapping Table L3 Routing Default Ingress VID Ethernet Type Range Assignment Answer Ingress VID Ethernet Type Range Search Data Ingress VID Inner VID Range Assignment Answer Ingress VID Inner VID Range Search Data Ingress VID Outer VID Range Assignment Answer Ingress VID Outer VID Range Search Data Ingress VID MAC Range Assignment Answer Ingress Configurable ACL 3 Rules Setup Ingress Configurable ACL 2 Rules Setup Ingress Configurable ACL 2 Pre Lookup Ingress Configurable ACL 1 Rules Setup Ingress Configurable ACL 1 Pre Lookup Ingress Configurable ACL 0 Rules Setup Ingress Configurable ACL 0 Pre Lookup Ingress Port Packet Type Filter SMON Set Search Default Packet To CPU Modification L2 Reserved Multicast Address Action Second Tunnel Exit Miss Action Second Tunnel Exit Lookup TCAM Answer Link Aggregation Membership Source Port Table Egress ACL Rule Pointer TCAM ARP Packet Decoder Options RARP Packet Decoder Options IEEE 1588 L2 Packet Decoder Options IEEE 802.1X and EAPOL Packet Decoder Options GRE Packet Decoder Options SCTP Packet Decoder Options LACP Packet Decoder Options AH Header Packet Decoder Options ESP Header Packet Decoder Options DNS Packet Decoder Options BOOTP and DHCP Packet Decoder Options CAPWAP Packet Decoder Options IKE Packet Decoder Options Debug Counter I2DaHashKey Setup Egress Spanning Tree State Tunnel Exit Lookup TCAM Ingress Configurable ACL 0 TCAM IEEE 1588 L4 Packet Decoder Options Ingress Configurable ACL 0 Search Mask Second Tunnel Exit Lookup TCAM Router Port MAC Address Ingress VID MAC Range Search Data Ingress Configurable ACL 3 TCAM Reserved Source MAC Address Range Reserved Destination MAC Address Range Send to CPU LLDP Configuration



Bank Name	Connected Registers or Tables
	Ingress Configurable ACL 1 Search Mask Egress Configurable ACL 0 Search Mask Egress Configurable ACL 0 TCAM Ingress Configurable ACL 1 TCAM Ingress Configurable ACL 2 TCAM Egress Configurable ACL 1 TCAM
ipp_register_bank_misc_ss0	Ingress Drop Options
L3 Routing TCAM Slice 0	L3 Routing TCAM Slice 0
count_packets_ipp0_smonStatisticsBlock	SMON Set 0 Packet Counter[0..7] SMON Set 1 Packet Counter[0..7] SMON Set 2 Packet Counter[0..7] SMON Set 3 Packet Counter[0..7]
count_bytes_ipp0_smonStatisticsBlock	SMON Set 0 Byte Counter[0..7] SMON Set 1 Byte Counter[0..7] SMON Set 2 Byte Counter[0..7] SMON Set 3 Byte Counter[0..7]
count_ipp0_aclConfStatisticsBlock	Ingress Configurable ACL Match Counter[0..63]
count_ipp0_vrflnStatisticsBlock	Received Packets on Ingress VRF[0..3]
Next Hop Hit Status Slice 0	Next Hop Hit Status Slice 0
count_ipp0_egressAclStatisticsBlock	Egress Configurable ACL Match Counter[0..63]
count_ipp0_egressDropStatisticsBlock	Queue Off Drop[0..31] Egress Spanning Tree Drop[0..31] MBSC Drop[0..31] Ingress-Egress Packet Filtering Drop[0..31] L2 Action Table Per Port Drop[0..31]
count_ucipp0_igrPortMibBlock	IP Unicast Received Counter[0..31]
count_mcipp0_igrPortMibBlock	IP Multicast Received Counter[0..31]
count_uc_routedipp0_igrPortMibBlock	IP Unicast Routed Counter[0..31]
count_mc_routedipp0_igrPortMibBlock	IP Multicast Routed Counter[0..31]
count_mc_acl_dropipp0_igrPortMibBlock	IP Multicast ACL Drop Counter[0..31]
count_ipp0_debugIppStatisticsBlock	Debug IPP Counter[0..22]
bk_mmp_stat_0	Flow Classification And Metering Drop
bk_ingress_admission_control_all_red_en_0	Ingress Admission Control Mark All Red Enable
bk_ingress_admission_control_all_red_0	Ingress Admission Control Mark All Red
Ingress Admission Control Token Bucket Configuration Slice 0	Ingress Admission Control Token Bucket Configuration Slice 0
Ingress Admission Control Reset Slice 0	Ingress Admission Control Reset Slice 0
Ingress Admission Control Current Status Slice 0	Ingress Admission Control Current Status Slice 0
bk_erm_ss0	ERM Yellow Configuration ERM Red Configuration Egress Resource Manager Pointer[0..7] Resource Limiter Set[0..3]
count_erm_ss0	Egress Resource Manager Drop[0..7]
pb_info_regbank_ss0	Packet Buffer Status
count_drop_pa_top_switch_pb0	Buffer Overflow Drop Ingress Resource Manager Drop
pb_queue_manage_register_bank_ss0	Map Queue to Priority[0..7]
count_drop_pa_top_switch_pb0_iRequeue	Re-queue Overflow Drop
pfc_regbank_rsv_size_ss0	Port/TC Reserved[0..63]
pfc_regbank_port_rsv_size_ss0	Port Reserved[0..7]
PFC Inc Counters for ingress ports 0 to 7 Slice 0	PFC Inc Counters for ingress ports 0 to 7 Slice 0



Bank Name	Connected Registers or Tables
PFC Inc Counters for ingress ports 8 to 15 Slice 0	PFC Inc Counters for ingress ports 8 to 15 Slice 0
PFC Inc Counters for ingress ports 16 to 23 Slice 0	PFC Inc Counters for ingress ports 16 to 23 Slice 0
PFC Inc Counters for ingress ports 24 to 31 Slice 0	PFC Inc Counters for ingress ports 24 to 31 Slice 0
PFC Dec Counters for ingress ports 0 to 7 Slice 0	PFC Dec Counters for ingress ports 0 to 7 Slice 0
PFC Dec Counters for ingress ports 8 to 15 Slice 0	PFC Dec Counters for ingress ports 8 to 15 Slice 0
PFC Dec Counters for ingress ports 16 to 23 Slice 0	PFC Dec Counters for ingress ports 16 to 23 Slice 0
PFC Dec Counters for ingress ports 24 to 31 Slice 0	PFC Dec Counters for ingress ports 24 to 31 Slice 0
pfc_regbank_cmn_misc_ss0	Port FFA Used[0..7] Port Used[0..7] TC FFA Used[0..7] FFA Used PFC FFA Used non-PFC
pfc_regbank_pause_settings1_ss0	Port Pause Settings[0..7]
pfc_regbank_taildrop_settings0_ss0	Port Tail-Drop Settings[0..7]
pfc_regbank_misc_ss0	Xon FFA Threshold Xoff FFA Threshold Tail-Drop FFA Threshold TC Xon FFA Threshold[0..7] TC Xoff FFA Threshold[0..7] TC Tail-Drop FFA Threshold[0..7] Port Xon FFA Threshold[0..7] Port Xoff FFA Threshold[0..7] Port Tail-Drop FFA Threshold[0..7] Port/TC Xon Total Threshold[0..63] Port/TC Xoff Total Threshold[0..63] Port/TC Tail-Drop Total Threshold[0..63]
qe_register_bank_ss0_sp0	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss0_sp1	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss0_sp2	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss0_sp3	Egress Port Depth[0..7] Egress Queue Depth[0..63]
pb_r_register_bank_ss0	Minimum Buffer Free
disable_queue_output_register_bank_ss0	Output Disable[0..7]
dwrr_bucket_capacity_settings_ss0	DWRR Bucket Capacity Configuration[0..7]
dwrr_bucket_misc_settings_ss0	DWRR Bucket Misc Configuration[0..7]
dwrr_weight_settings_ss0	DWRR Weight Configuration[0..63]
queue_shaper_rate_settings_ss0	Queue Shaper Rate Configuration
queue_shaper_bucket_settings_ss0	Queue Shaper Bucket Capacity Configuration Queue Shaper Bucket Threshold Configuration
queue_shaper_misc_ss0	Queue Shaper Enable
prio_shaper_rate_settings_ss0	Prio Shaper Rate Configuration
prio_shaper_bucket_settings_ss0	Prio Shaper Bucket Capacity Configuration Prio Shaper Bucket Threshold Configuration
prio_shaper_misc_ss0	Prio Shaper Enable



Bank Name	Connected Registers or Tables
port_shaper_rate_settings_ss0	Port Shaper Rate Configuration
port_shaper_bucket_settings_ss0	Port Shaper Bucket Capacity Configuration Port Shaper Bucket Threshold Configuration
port_shaper_misc_ss0	Port Shaper Enable
count_opkt_pa top switch pb0	PB Packet Head Counter PB Packet Tail Counter
drain_port_ss0	Drain Port
drain_drop_ss0	Drain Port Drop[0..7]
count_pa top switch epp0 conf	Unknown Egress Drop[0..7] Egress Port Disabled Drop[0..7] Egress Port Filtering Drop[0..7] Tunnel Exit Too Small Packet Modification To Small Drop[0..7] EPP PM Drop
count_opkt_pa top switch epp0 conf	EPP Packet Head Counter EPP Packet Tail Counter
Egress Port Configuration Slice 0	Egress Port Configuration Slice 0
Egress Tunnel Exit Table Slice 0	Egress Tunnel Exit Table Slice 0
Tunnel Entry Instruction Table Slice 0	Tunnel Entry Instruction Table Slice 0
Tunnel Entry Header Data Slice 0	Tunnel Entry Header Data Slice 0
Beginning of Packet Tunnel Entry Instruction Table Slice 0	Beginning of Packet Tunnel Entry Instruction Table Slice 0
L2 Tunnel Entry Instruction Table Slice 0	L2 Tunnel Entry Instruction Table Slice 0
L3 Tunnel Entry Instruction Table Slice 0	L3 Tunnel Entry Instruction Table Slice 0
Color Remap From Egress Port Slice 0	Color Remap From Egress Port Slice 0
Color Remap From Ingress Admission Control Slice 0	Color Remap From Ingress Admission Control Slice 0
Egress Router Table Slice 0	Egress Router Table Slice 0
Next Hop DA MAC Slice 0	Next Hop DA MAC Slice 0
Router Port Egress SA MAC Address Slice 0	Router Port Egress SA MAC Address Slice 0
Next Hop MPLS Table Slice 0	Next Hop MPLS Table Slice 0
Egress MPLS TTL Table Slice 0	Egress MPLS TTL Table Slice 0
Next Hop Packet Insert MPLS Header Slice 0	Next Hop Packet Insert MPLS Header Slice 0
Egress Queue To PCP And CFI/DEI Mapping Table Slice 0	Egress Queue To PCP And CFI/DEI Mapping Table Slice 0
Egress VLAN Translation TCAM Answer Slice 0	Egress VLAN Translation TCAM Answer Slice 0
Ingress NAT Operation Slice 0	Ingress NAT Operation Slice 0
Egress NAT Operation Slice 0	Egress NAT Operation Slice 0
Select Which Egress QoS Mapping Table To Use Slice 0	Select Which Egress QoS Mapping Table To Use Slice 0
L2 QoS Mapping Table Slice 0	L2 QoS Mapping Table Slice 0
IP QoS Mapping Table Slice 0	IP QoS Mapping Table Slice 0
TOS QoS Mapping Table Slice 0	TOS QoS Mapping Table Slice 0
MPLS QoS Mapping Table Slice 0	MPLS QoS Mapping Table Slice 0
epp_register_bank_ss0	Output Mirroring Table Egress VLAN Translation TCAM Egress Queue To MPLS EXP Mapping Table Debug Counter reQueuePortId Setup Debug Counter fromPort Setup Egress MPLS Decoding Options Egress Ethernet Type for VLAN tag NAT Add Egress Port for NAT Calculation



Bank Name	Connected Registers or Tables
	Disable CPU tag on CPU Port Debug Counter debugMatchEPP0 Setup EPP Debug imActive EPP Debug imExtra EPP Debug omEnabled EPP Debug omImActive EPP Debug reQueue EPP Debug reQueuePortId EPP Debug reQueuePkt EPP Debug fromPort EPP Debug delSpecificVlan EPP Debug updateTosExp EPP Debug isIPv4 EPP Debug isIPv6 EPP Debug addNewMpls EPP Debug isPPPoE EPP Debug debugMatchEPP0 Egress Port VID Operation
count_esp0_vrfOutStatisticsBlock	Transmitted Packets on Egress VRF[0..3]
Ingress NAT Hit Status Slice 0	Ingress NAT Hit Status Slice 0
Egress NAT Hit Status Slice 0	Egress NAT Hit Status Slice 0
count_esp0_debugEspStatisticsBlock	Debug ESP Counter[0..14]
count_ps0_ss0	PS Error Counter
count_ps1_ss0	PS Error Counter
count_ps2_ss0	PS Error Counter
count_ps3_ss0	PS Error Counter
count_ps4_ss0	PS Error Counter
count_ps5_ss0	PS Error Counter
count_ps6_ss0	PS Error Counter
count_ps7_ss0	PS Error Counter
count_opkt_pa top switch ps0 ps_wrap.bridge	PS Packet Head Counter PS Packet Tail Counter
count_sp_ss1	SP Overflow Drop
count_broken_pkt_ss1	IPP PM Drop IPP Empty Destination Drop
count_pa top switch ipp1 conf	Unknown Ingress Drop Empty Mask Drop Ingress Spanning Tree Drop: Listen Ingress Spanning Tree Drop: Learning Ingress Spanning Tree Drop: Blocking L2 Lookup Drop Ingress Packet Filtering Drop Reserved MAC DA Drop Reserved MAC SA Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop Invalid Routing Protocol Drop Expired TTL Drop L3 Lookup Drop IP Checksum Drop Second Tunnel Exit Drop Tunnel Exit Miss Action Drop Tunnel Exit Too Small Packet Modification Drop



Bank Name	Connected Registers or Tables
	Learning Packet Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Egress Configurable ACL Drop ARP Decoder Drop RARP Decoder Drop L2 IEEE 1588 Decoder Drop L4 IEEE 1588 Decoder Drop IEEE 802.1X and EAPOL Decoder Drop SCTP Decoder Drop LACP Decoder Drop AH Decoder Drop ESP Decoder Drop DNS Decoder Drop BOOTP and DHCP Decoder Drop CAPWAP Decoder Drop IKE Decoder Drop GRE Decoder Drop NAT Action Table Drop L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop Source Port Default ACL Action Drop
count_opkt.pa top switch ipp1 conf	IPP Packet Head Counter IPP Packet Tail Counter
Tunnel Exit Lookup TCAM Answer Slice 1	Tunnel Exit Lookup TCAM Answer Slice 1
Ingress Admission Control Initial Pointer Slice 1	Ingress Admission Control Initial Pointer Slice 1
Ingress Configurable ACL 0 Large Table Slice 1	Ingress Configurable ACL 0 Large Table Slice 1
Ingress Configurable ACL 0 Small Table Slice 1	Ingress Configurable ACL 0 Small Table Slice 1
Ingress Configurable ACL 0 TCAM Answer Slice 1	Ingress Configurable ACL 0 TCAM Answer Slice 1
Ingress Configurable ACL 1 Large Table Slice 1	Ingress Configurable ACL 1 Large Table Slice 1
Ingress Configurable ACL 1 Small Table Slice 1	Ingress Configurable ACL 1 Small Table Slice 1
Ingress Configurable ACL 1 TCAM Answer Slice 1	Ingress Configurable ACL 1 TCAM Answer Slice 1
Ingress Configurable ACL 2 TCAM Answer Slice 1	Ingress Configurable ACL 2 TCAM Answer Slice 1
Ingress Configurable ACL 3 TCAM Answer Slice 1	Ingress Configurable ACL 3 TCAM Answer Slice 1
Source Port Default ACL Action Slice 1	Source Port Default ACL Action Slice 1
VLAN Table Slice 1	VLAN Table Slice 1
Ingress Multiple Spanning Tree State Slice 1	Ingress Multiple Spanning Tree State Slice 1
Ingress Router Table Slice 1	Ingress Router Table Slice 1
L3 LPM Result Slice 1	L3 LPM Result Slice 1
Hash Based L3 Routing Table Slice 1	Hash Based L3 Routing Table Slice 1
Next Hop Table Slice 1	Next Hop Table Slice 1
Next Hop Packet Modifications Slice 1	Next Hop Packet Modifications Slice 1
L2 Aging Status Shadow Table Slice 1	L2 Aging Status Shadow Table Slice 1



Bank Name	Connected Registers or Tables
L2 DA Hash Lookup Table Slice 1	L2 DA Hash Lookup Table Slice 1
L2 Destination Table Slice 1	L2 Destination Table Slice 1
L2 SA Hash Lookup Table Slice 1	L2 SA Hash Lookup Table Slice 1
L2 Aging Status Shadow Table - Replica Slice 1	L2 Aging Status Shadow Table - Replica Slice 1
L2 Destination Table - Replica Slice 1	L2 Destination Table - Replica Slice 1
L2 Action Table Slice 1	L2 Action Table Slice 1
L2 Action Table Source Port Slice 1	L2 Action Table Source Port Slice 1
Egress ACL Rule Pointer TCAM Answer Slice 1	Egress ACL Rule Pointer TCAM Answer Slice 1
Egress Configurable ACL 0 Large Table Slice 1	Egress Configurable ACL 0 Large Table Slice 1
Egress Configurable ACL 0 Small Table Slice 1	Egress Configurable ACL 0 Small Table Slice 1
Egress Configurable ACL 0 TCAM Answer Slice 1	Egress Configurable ACL 0 TCAM Answer Slice 1
Egress Configurable ACL 1 TCAM Answer Slice 1	Egress Configurable ACL 1 TCAM Answer Slice 1
Tunnel Entry MTU Length Check Slice 1	Tunnel Entry MTU Length Check Slice 1
ipp_register.bank.ss1	Link Aggregation Ctrl Debug Counter srcPort Setup L2 Tunnel Decoder Setup Learning DA MAC SNAP LLC Decoding Options Ingress Ethernet Type for VLAN tag L2 Reserved Multicast Address Base Debug Counter nrVlans Setup Debug Counter spVidOp Setup Ingress Configurable ACL 0 Selection Ingress Configurable ACL 1 Selection Debug Counter finalVid Setup Debug Counter vlanVidOp Setup Debug Counter nextHopPtrLpm Setup Debug Counter nextHopPtrHash Setup Debug Counter nextHopPtrFinal Setup Check IPv4 Header Checksum Force Non VLAN Packet To Specific Queue Force Unknown L3 Packet To Specific Egress Queue Force Non VLAN Packet To Specific Color Force Unknown L3 Packet To Specific Color Debug Counter I2DaHash Setup Debug Counter I2DaHashHitAndBucket Setup Debug Counter I2DaTcamHitsAndCast Setup Forward From CPU Port Move Options L2 Action Table Egress Port State L2 Multicast Handling Egress Configurable ACL 0 Selection Egress Port NAT State NAT Action Table Force Original Packet Ingress MMP Drop Mask Debug Counter dstPortmask Setup Debug Counter debugMatchIPP0 Setup IPP Debug srcPort IPP Debug dropPktAfterL2Decode



Bank Name	Connected Registers or Tables
	IPP Debug nrVlans IPP Debug dropPktAfterL3Decode IPP Debug spVidOp IPP Debug finalVid IPP Debug vlanVidOp IPP Debug routerHit IPP Debug nextHopPtrLpm IPP Debug nextHopPtrHash IPP Debug nextHopPtrLpmHit IPP Debug nextHopPtrHashHit IPP Debug nextHopPtrFinal IPP Debug l2DaHashKey IPP Debug l2DaHash IPP Debug l2DaHashHitAndBucket IPP Debug l2DaTcamHitsAndCast IPP Debug routed IPP Debug isFlooding IPP Debug isBroadcast IPP Debug doL2Lookup IPP Debug dstPortmask IPP Debug debugMatchIPP0 Enable Enqueue To Ports And Queues Flooding Action Send to Port Link Aggregation To Physical Ports Members Link Aggregate Weight CPU Reason Code Operation Ingress Egress Port Packet Type Filter NAT Action Table Egress Configurable ACL 1 Rules Setup Egress Configurable ACL 0 Rules Setup Allow Special Frame Check For L2 Action Table Egress Multiple Spanning Tree State Router MTU Table Hairpin Enable L2 Multicast Table L2 Lookup Collision Table Masks L2 Lookup Collision Table L2 Aging Collision Shadow Table Router Egress Queue To VLAN Data MPLS EXP Field To Packet Color Mapping Table IPv6 Class of Service Field To Packet Color Mapping Table IPv4 TOS Field To Packet Color Mapping Table VLAN PCP And DEI To Color Mapping Table MPLS EXP Field To Egress Queue Mapping Table IPv6 Class of Service Field To Egress Queue Mapping Table IPv4 TOS Field To Egress Queue Mapping Table VLAN PCP To Queue Mapping Table L3 Routing Default Ingress VID Ethernet Type Range Assignment Answer Ingress VID Ethernet Type Range Search Data Ingress VID Inner VID Range Assignment Answer Ingress VID Inner VID Range Search Data Ingress VID Outer VID Range Assignment Answer Ingress VID Outer VID Range Search Data Ingress VID MAC Range Assignment Answer Ingress Configurable ACL 3 Rules Setup



Bank Name	Connected Registers or Tables
	Ingress Configurable ACL 2 Rules Setup Ingress Configurable ACL 2 Pre Lookup Ingress Configurable ACL 1 Rules Setup Ingress Configurable ACL 1 Pre Lookup Ingress Configurable ACL 0 Rules Setup Ingress Configurable ACL 0 Pre Lookup Ingress Port Packet Type Filter SMON Set Search Default Packet To CPU Modification L2 Reserved Multicast Address Action Second Tunnel Exit Miss Action Second Tunnel Exit Lookup TCAM Answer Link Aggregation Membership Source Port Table Egress ACL Rule Pointer TCAM ARP Packet Decoder Options RARP Packet Decoder Options IEEE 1588 L2 Packet Decoder Options IEEE 802.1X and EAPOL Packet Decoder Options GRE Packet Decoder Options SCTP Packet Decoder Options LACP Packet Decoder Options AH Header Packet Decoder Options ESP Header Packet Decoder Options DNS Packet Decoder Options BOOTP and DHCP Packet Decoder Options CAPWAP Packet Decoder Options IKE Packet Decoder Options Debug Counter I2DaHashKey Setup Egress Spanning Tree State Tunnel Exit Lookup TCAM Ingress Configurable ACL 0 TCAM IEEE 1588 L4 Packet Decoder Options Ingress Configurable ACL 0 Search Mask Second Tunnel Exit Lookup TCAM Router Port MAC Address Ingress VID MAC Range Search Data Ingress Configurable ACL 3 TCAM Reserved Source MAC Address Range Reserved Destination MAC Address Range Send to CPU LLDP Configuration Ingress Configurable ACL 1 Search Mask Egress Configurable ACL 0 Search Mask Egress Configurable ACL 0 TCAM Ingress Configurable ACL 1 TCAM Ingress Configurable ACL 2 TCAM Egress Configurable ACL 1 TCAM
ipp_register_bank_misc_ss1	Ingress Drop Options
L3 Routing TCAM Slice 1	L3 Routing TCAM Slice 1
count_packets ipp1_smonStatisticsBlock	SMON Set 0 Packet Counter[0..7] SMON Set 1 Packet Counter[0..7] SMON Set 2 Packet Counter[0..7] SMON Set 3 Packet Counter[0..7]
count_bytes ipp1_smonStatisticsBlock	SMON Set 0 Byte Counter[0..7]



Bank Name	Connected Registers or Tables
	SMON Set 1 Byte Counter[0..7] SMON Set 2 Byte Counter[0..7] SMON Set 3 Byte Counter[0..7]
count_ipp1_aclConfStatisticsBlock	Ingress Configurable ACL Match Counter[0..63]
count_ipp1_vrflnStatisticsBlock	Received Packets on Ingress VRF[0..3]
Next Hop Hit Status Slice 1	Next Hop Hit Status Slice 1
count_ipp1_egressAclStatisticsBlock	Egress Configurable ACL Match Counter[0..63]
count_ipp1_egressDropStatisticsBlock	Queue Off Drop[0..31] Egress Spanning Tree Drop[0..31] MBSC Drop[0..31] Ingress-Egress Packet Filtering Drop[0..31] L2 Action Table Per Port Drop[0..31]
count_ucipp1_igrPortMibBlock	IP Unicast Received Counter[0..31]
count_mcipp1_igrPortMibBlock	IP Multicast Received Counter[0..31]
count_uc_routedipp1_igrPortMibBlock	IP Unicast Routed Counter[0..31]
count_mc_routedipp1_igrPortMibBlock	IP Multicast Routed Counter[0..31]
count_mc_acl_dropipp1_igrPortMibBlock	IP Multicast ACL Drop Counter[0..31]
count_ipp1_debugIppStatisticsBlock	Debug IPP Counter[0..22]
bk_mmp_stat_1	Flow Classification And Metering Drop
bk_ingress_admission_control_all_red_en_1	Ingress Admission Control Mark All Red Enable
bk_ingress_admission_control_all_red_1	Ingress Admission Control Mark All Red
Ingress Admission Control Token Bucket Configuration Slice 1	Ingress Admission Control Token Bucket Configuration Slice 1
Ingress Admission Control Reset Slice 1	Ingress Admission Control Reset Slice 1
Ingress Admission Control Current Status Slice 1	Ingress Admission Control Current Status Slice 1
bk_erm_ss1	ERM Yellow Configuration ERM Red Configuration Egress Resource Manager Pointer[0..7] Resource Limiter Set[0..3]
count_erm_ss1	Egress Resource Manager Drop[0..7]
pb_info_regbank_ss1	Packet Buffer Status
count_drop_pa_top_switch_pb1	Buffer Overflow Drop Ingress Resource Manager Drop
pb_queue_manage_register_bank_ss1	Map Queue to Priority[0..7]
count_drop_pa_top_switch_pb1_iRequeue	Re-queue Overflow Drop
pfc_regbank_rsv_size_ss1	Port/TC Reserved[0..63]
pfc_regbank_port_rsv_size_ss1	Port Reserved[0..7]
PFC Inc Counters for ingress ports 0 to 7 Slice 1	PFC Inc Counters for ingress ports 0 to 7 Slice 1
PFC Inc Counters for ingress ports 8 to 15 Slice 1	PFC Inc Counters for ingress ports 8 to 15 Slice 1
PFC Inc Counters for ingress ports 16 to 23 Slice 1	PFC Inc Counters for ingress ports 16 to 23 Slice 1
PFC Inc Counters for ingress ports 24 to 31 Slice 1	PFC Inc Counters for ingress ports 24 to 31 Slice 1
PFC Dec Counters for ingress ports 0 to 7 Slice 1	PFC Dec Counters for ingress ports 0 to 7 Slice 1
PFC Dec Counters for ingress ports 8 to 15 Slice 1	PFC Dec Counters for ingress ports 8 to 15 Slice 1
PFC Dec Counters for ingress ports 16 to 23 Slice 1	PFC Dec Counters for ingress ports 16 to 23 Slice 1
PFC Dec Counters for ingress ports 24 to 31 Slice 1	PFC Dec Counters for ingress ports 24 to 31 Slice 1



Bank Name	Connected Registers or Tables
pfc_regbank_cmn_misc_ss1	Port FFA Used[0..7] Port Used[0..7] TC FFA Used[0..7] FFA Used PFC FFA Used non-PFC
pfc_regbank_pause_settings1_ss1	Port Pause Settings[0..7]
pfc_regbank_taildrop_settings0_ss1	Port Tail-Drop Settings[0..7]
pfc_regbank_misc_ss1	Xon FFA Threshold Xoff FFA Threshold Tail-Drop FFA Threshold TC Xon FFA Threshold[0..7] TC Xoff FFA Threshold[0..7] TC Tail-Drop FFA Threshold[0..7] Port Xon FFA Threshold[0..7] Port Xoff FFA Threshold[0..7] Port Tail-Drop FFA Threshold[0..7] Port/TC Xon Total Threshold[0..63] Port/TC Xoff Total Threshold[0..63] Port/TC Tail-Drop Total Threshold[0..63]
qe_register_bank_ss1_sp0	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss1_sp1	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss1_sp2	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss1_sp3	Egress Port Depth[0..7] Egress Queue Depth[0..63]
pb_r_register_bank_ss1	Minimum Buffer Free
disable_queue_output_register_bank_ss1	Output Disable[0..7]
dwrr_bucket_capacity_settings_ss1	DWRR Bucket Capacity Configuration[0..7]
dwrr_bucket_misc_settings_ss1	DWRR Bucket Misc Configuration[0..7]
dwrr_weight_settings_ss1	DWRR Weight Configuration[0..63]
queue_shaper_rate_settings_ss1	Queue Shaper Rate Configuration
queue_shaper_bucket_settings_ss1	Queue Shaper Bucket Capacity Configuration Queue Shaper Bucket Threshold Configuration
queue_shaper_misc_ss1	Queue Shaper Enable
prio_shaper_rate_settings_ss1	Prio Shaper Rate Configuration
prio_shaper_bucket_settings_ss1	Prio Shaper Bucket Capacity Configuration Prio Shaper Bucket Threshold Configuration
prio_shaper_misc_ss1	Prio Shaper Enable
port_shaper_rate_settings_ss1	Port Shaper Rate Configuration
port_shaper_bucket_settings_ss1	Port Shaper Bucket Capacity Configuration Port Shaper Bucket Threshold Configuration
port_shaper_misc_ss1	Port Shaper Enable
count_opkt_pa_top_switch_pb1	PB Packet Head Counter PB Packet Tail Counter
drain_port_ss1	Drain Port
drain_drop_ss1	Drain Port Drop[0..7]
count_pa_top_switch_epp1_conf	Unknown Egress Drop[0..7] Egress Port Disabled Drop[0..7] Egress Port Filtering Drop[0..7] Tunnel Exit Too Small Packet Modification To Small Drop[0..7] EPP PM Drop
count_opkt_pa_top_switch_epp1_conf	EPP Packet Head Counter



Bank Name	Connected Registers or Tables
	EPP Packet Tail Counter
Egress Port Configuration Slice 1	Egress Port Configuration Slice 1
Egress Tunnel Exit Table Slice 1	Egress Tunnel Exit Table Slice 1
Tunnel Entry Instruction Table Slice 1	Tunnel Entry Instruction Table Slice 1
Tunnel Entry Header Data Slice 1	Tunnel Entry Header Data Slice 1
Beginning of Packet Tunnel Entry Instruction Table Slice 1	Beginning of Packet Tunnel Entry Instruction Table Slice 1
L2 Tunnel Entry Instruction Table Slice 1	L2 Tunnel Entry Instruction Table Slice 1
L3 Tunnel Entry Instruction Table Slice 1	L3 Tunnel Entry Instruction Table Slice 1
Color Remap From Egress Port Slice 1	Color Remap From Egress Port Slice 1
Color Remap From Ingress Admission Control Slice 1	Color Remap From Ingress Admission Control Slice 1
Egress Router Table Slice 1	Egress Router Table Slice 1
Next Hop DA MAC Slice 1	Next Hop DA MAC Slice 1
Router Port Egress SA MAC Address Slice 1	Router Port Egress SA MAC Address Slice 1
Next Hop MPLS Table Slice 1	Next Hop MPLS Table Slice 1
Egress MPLS TTL Table Slice 1	Egress MPLS TTL Table Slice 1
Next Hop Packet Insert MPLS Header Slice 1	Next Hop Packet Insert MPLS Header Slice 1
Egress Queue To PCP And CFI/DEI Mapping Table Slice 1	Egress Queue To PCP And CFI/DEI Mapping Table Slice 1
Egress VLAN Translation TCAM Answer Slice 1	Egress VLAN Translation TCAM Answer Slice 1
Ingress NAT Operation Slice 1	Ingress NAT Operation Slice 1
Egress NAT Operation Slice 1	Egress NAT Operation Slice 1
Select Which Egress QoS Mapping Table To Use Slice 1	Select Which Egress QoS Mapping Table To Use Slice 1
L2 QoS Mapping Table Slice 1	L2 QoS Mapping Table Slice 1
IP QoS Mapping Table Slice 1	IP QoS Mapping Table Slice 1
TOS QoS Mapping Table Slice 1	TOS QoS Mapping Table Slice 1
MPLS QoS Mapping Table Slice 1	MPLS QoS Mapping Table Slice 1
epp_register_bank_ss1	Output Mirroring Table Egress VLAN Translation TCAM Egress Queue To MPLS EXP Mapping Table Debug Counter reQueuePortId Setup Debug Counter fromPort Setup Egress MPLS Decoding Options Egress Ethernet Type for VLAN tag NAT Add Egress Port for NAT Calculation Disable CPU tag on CPU Port Debug Counter debugMatchEPP0 Setup EPP Debug imActive EPP Debug imExtra EPP Debug omEnabled EPP Debug omImActive EPP Debug reQueue EPP Debug reQueuePortId EPP Debug reQueuePkt EPP Debug fromPort EPP Debug delSpecificVlan EPP Debug updateTosExp EPP Debug isIPv4 EPP Debug isIPv6 EPP Debug addNewMpls



Bank Name	Connected Registers or Tables
	EPP Debug isPPPoE EPP Debug debugMatchEPP0 Egress Port VID Operation
count_epp1_vrfOutStatisticsBlock	Transmitted Packets on Egress VRF[0..3]
Ingress NAT Hit Status Slice 1	Ingress NAT Hit Status Slice 1
Egress NAT Hit Status Slice 1	Egress NAT Hit Status Slice 1
count_epp1_debugEppStatisticsBlock	Debug EPP Counter[0..14]
count_ps0_ss1	PS Error Counter
count_ps1_ss1	PS Error Counter
count_ps2_ss1	PS Error Counter
count_ps3_ss1	PS Error Counter
count_ps4_ss1	PS Error Counter
count_ps5_ss1	PS Error Counter
count_ps6_ss1	PS Error Counter
count_ps7_ss1	PS Error Counter
count_opkt.pa top switch ps1 ps_wrap.bridge	PS Packet Head Counter PS Packet Tail Counter
count_sp_ss2	SP Overflow Drop
count_broken_pkt_ss2	IPP PM Drop IPP Empty Destination Drop
count_pa top switch ipp2 conf	Unknown Ingress Drop Empty Mask Drop Ingress Spanning Tree Drop: Listen Ingress Spanning Tree Drop: Learning Ingress Spanning Tree Drop: Blocking L2 Lookup Drop Ingress Packet Filtering Drop Reserved MAC DA Drop Reserved MAC SA Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop Invalid Routing Protocol Drop Expired TTL Drop L3 Lookup Drop IP Checksum Drop Second Tunnel Exit Drop Tunnel Exit Miss Action Drop Tunnel Exit Too Small Packet Modification Drop Learning Packet Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Egress Configurable ACL Drop ARP Decoder Drop RARP Decoder Drop L2 IEEE 1588 Decoder Drop L4 IEEE 1588 Decoder Drop IEEE 802.1X and EAPOL Decoder Drop SCTP Decoder Drop LACP Decoder Drop AH Decoder Drop ESP Decoder Drop DNS Decoder Drop BOOTP and DHCP Decoder Drop



Bank Name	Connected Registers or Tables
	CAPWAP Decoder Drop IKE Decoder Drop GRE Decoder Drop NAT Action Table Drop L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop Source Port Default ACL Action Drop
count_opkt_pa top switch ipp2 conf	IPP Packet Head Counter IPP Packet Tail Counter
Tunnel Exit Lookup TCAM Answer Slice 2	Tunnel Exit Lookup TCAM Answer Slice 2
Ingress Admission Control Initial Pointer Slice 2	Ingress Admission Control Initial Pointer Slice 2
Ingress Configurable ACL 0 Large Table Slice 2	Ingress Configurable ACL 0 Large Table Slice 2
Ingress Configurable ACL 0 Small Table Slice 2	Ingress Configurable ACL 0 Small Table Slice 2
Ingress Configurable ACL 0 TCAM Answer Slice 2	Ingress Configurable ACL 0 TCAM Answer Slice 2
Ingress Configurable ACL 1 Large Table Slice 2	Ingress Configurable ACL 1 Large Table Slice 2
Ingress Configurable ACL 1 Small Table Slice 2	Ingress Configurable ACL 1 Small Table Slice 2
Ingress Configurable ACL 1 TCAM Answer Slice 2	Ingress Configurable ACL 1 TCAM Answer Slice 2
Ingress Configurable ACL 2 TCAM Answer Slice 2	Ingress Configurable ACL 2 TCAM Answer Slice 2
Ingress Configurable ACL 3 TCAM Answer Slice 2	Ingress Configurable ACL 3 TCAM Answer Slice 2
Source Port Default ACL Action Slice 2	Source Port Default ACL Action Slice 2
VLAN Table Slice 2	VLAN Table Slice 2
Ingress Multiple Spanning Tree State Slice 2	Ingress Multiple Spanning Tree State Slice 2
Ingress Router Table Slice 2	Ingress Router Table Slice 2
L3 LPM Result Slice 2	L3 LPM Result Slice 2
Hash Based L3 Routing Table Slice 2	Hash Based L3 Routing Table Slice 2
Next Hop Table Slice 2	Next Hop Table Slice 2
Next Hop Packet Modifications Slice 2	Next Hop Packet Modifications Slice 2
L2 Aging Status Shadow Table Slice 2	L2 Aging Status Shadow Table Slice 2
L2 DA Hash Lookup Table Slice 2	L2 DA Hash Lookup Table Slice 2
L2 Destination Table Slice 2	L2 Destination Table Slice 2
L2 SA Hash Lookup Table Slice 2	L2 SA Hash Lookup Table Slice 2
L2 Aging Status Shadow Table - Replica Slice 2	L2 Aging Status Shadow Table - Replica Slice 2
L2 Destination Table - Replica Slice 2	L2 Destination Table - Replica Slice 2
L2 Action Table Slice 2	L2 Action Table Slice 2
L2 Action Table Source Port Slice 2	L2 Action Table Source Port Slice 2
Egress ACL Rule Pointer TCAM Answer Slice 2	Egress ACL Rule Pointer TCAM Answer Slice 2
Egress Configurable ACL 0 Large Table Slice 2	Egress Configurable ACL 0 Large Table Slice 2
Egress Configurable ACL 0 Small Table Slice 2	Egress Configurable ACL 0 Small Table Slice 2



Bank Name	Connected Registers or Tables
Egress Configurable ACL 0 TCAM Answer Slice 2	Egress Configurable ACL 0 TCAM Answer Slice 2
Egress Configurable ACL 1 TCAM Answer Slice 2	Egress Configurable ACL 1 TCAM Answer Slice 2
Tunnel Entry MTU Length Check Slice 2	Tunnel Entry MTU Length Check Slice 2
ipp_register_bank_ss2	Link Aggregation Ctrl Debug Counter srcPort Setup L2 Tunnel Decoder Setup Learning DA MAC SNAP LLC Decoding Options Ingress Ethernet Type for VLAN tag L2 Reserved Multicast Address Base Debug Counter nrVlans Setup Debug Counter spVidOp Setup Ingress Configurable ACL 0 Selection Ingress Configurable ACL 1 Selection Debug Counter finalVid Setup Debug Counter vlanVidOp Setup Debug Counter nextHopPtrLpm Setup Debug Counter nextHopPtrHash Setup Debug Counter nextHopPtrFinal Setup Check IPv4 Header Checksum Force Non VLAN Packet To Specific Queue Force Unknown L3 Packet To Specific Egress Queue Force Non VLAN Packet To Specific Color Force Unknown L3 Packet To Specific Color Debug Counter l2DaHash Setup Debug Counter l2DaHashHitAndBucket Setup Debug Counter l2DaTcamHitsAndCast Setup Forward From CPU Port Move Options L2 Action Table Egress Port State L2 Multicast Handling Egress Configurable ACL 0 Selection Egress Port NAT State NAT Action Table Force Original Packet Ingress MMP Drop Mask Debug Counter dstPortmask Setup Debug Counter debugMatchIPP0 Setup IPP Debug srcPort IPP Debug dropPktAfterL2Decode IPP Debug nrVlans IPP Debug dropPktAfterL3Decode IPP Debug spVidOp IPP Debug finalVid IPP Debug vlanVidOp IPP Debug routerHit IPP Debug nextHopPtrLpm IPP Debug nextHopPtrHash IPP Debug nextHopPtrLpmHit IPP Debug nextHopPtrHashHit IPP Debug nextHopPtrFinal IPP Debug l2DaHashKey IPP Debug l2DaHash IPP Debug l2DaHashHitAndBucket IPP Debug l2DaTcamHitsAndCast



Bank Name	Connected Registers or Tables
	IPP Debug routed IPP Debug isFlooding IPP Debug isBroadcast IPP Debug doL2Lookup IPP Debug dstPortmask IPP Debug debugMatchIPP0 Enable Enqueue To Ports And Queues Flooding Action Send to Port Link Aggregation To Physical Ports Members Link Aggregate Weight CPU Reason Code Operation Ingress Egress Port Packet Type Filter NAT Action Table Egress Configurable ACL 1 Rules Setup Egress Configurable ACL 0 Rules Setup Allow Special Frame Check For L2 Action Table Egress Multiple Spanning Tree State Router MTU Table Hairpin Enable L2 Multicast Table L2 Lookup Collision Table Masks L2 Lookup Collision Table L2 Aging Collision Shadow Table Router Egress Queue To VLAN Data MPLS EXP Field To Packet Color Mapping Table IPv6 Class of Service Field To Packet Color Mapping Table IPv4 TOS Field To Packet Color Mapping Table VLAN PCP And DEI To Color Mapping Table MPLS EXP Field To Egress Queue Mapping Table IPv6 Class of Service Field To Egress Queue Mapping Table IPv4 TOS Field To Egress Queue Mapping Table VLAN PCP To Queue Mapping Table L3 Routing Default Ingress VID Ethernet Type Range Assignment Answer Ingress VID Ethernet Type Range Search Data Ingress VID Inner VID Range Assignment Answer Ingress VID Inner VID Range Search Data Ingress VID Outer VID Range Assignment Answer Ingress VID Outer VID Range Search Data Ingress VID MAC Range Assignment Answer Ingress Configurable ACL 3 Rules Setup Ingress Configurable ACL 2 Rules Setup Ingress Configurable ACL 2 Pre Lookup Ingress Configurable ACL 1 Rules Setup Ingress Configurable ACL 1 Pre Lookup Ingress Configurable ACL 0 Rules Setup Ingress Configurable ACL 0 Pre Lookup Ingress Port Packet Type Filter SMON Set Search Default Packet To CPU Modification L2 Reserved Multicast Address Action Second Tunnel Exit Miss Action Second Tunnel Exit Lookup TCAM Answer Link Aggregation Membership Source Port Table Egress ACL Rule Pointer TCAM



Bank Name	Connected Registers or Tables
	ARP Packet Decoder Options RARP Packet Decoder Options IEEE 1588 L2 Packet Decoder Options IEEE 802.1X and EAPOL Packet Decoder Options GRE Packet Decoder Options SCTP Packet Decoder Options LACP Packet Decoder Options AH Header Packet Decoder Options ESP Header Packet Decoder Options DNS Packet Decoder Options BOOTP and DHCP Packet Decoder Options CAPWAP Packet Decoder Options IKE Packet Decoder Options Debug Counter l2DaHashKey Setup Egress Spanning Tree State Tunnel Exit Lookup TCAM Ingress Configurable ACL 0 TCAM IEEE 1588 L4 Packet Decoder Options Ingress Configurable ACL 0 Search Mask Second Tunnel Exit Lookup TCAM Router Port MAC Address Ingress VID MAC Range Search Data Ingress Configurable ACL 3 TCAM Reserved Source MAC Address Range Reserved Destination MAC Address Range Send to CPU LLDP Configuration Ingress Configurable ACL 1 Search Mask Egress Configurable ACL 0 Search Mask Egress Configurable ACL 0 TCAM Ingress Configurable ACL 1 TCAM Ingress Configurable ACL 2 TCAM Egress Configurable ACL 1 TCAM
ipp_register.bank.misc.ss2	Ingress Drop Options
L3 Routing TCAM Slice 2	L3 Routing TCAM Slice 2
count_packets_ipp2_smonStatisticsBlock	SMON Set 0 Packet Counter[0..7] SMON Set 1 Packet Counter[0..7] SMON Set 2 Packet Counter[0..7] SMON Set 3 Packet Counter[0..7]
count_bytes_ipp2_smonStatisticsBlock	SMON Set 0 Byte Counter[0..7] SMON Set 1 Byte Counter[0..7] SMON Set 2 Byte Counter[0..7] SMON Set 3 Byte Counter[0..7]
count_ipp2_aclConfStatisticsBlock	Ingress Configurable ACL Match Counter[0..63]
count_ipp2_vrflnStatisticsBlock	Received Packets on Ingress VRF[0..3]
Next Hop Hit Status Slice 2	Next Hop Hit Status Slice 2
count_ipp2_egressAclStatisticsBlock	Egress Configurable ACL Match Counter[0..63]
count_ipp2_egressDropStatisticsBlock	Queue Off Drop[0..31] Egress Spanning Tree Drop[0..31] MBSC Drop[0..31] Ingress-Egress Packet Filtering Drop[0..31] L2 Action Table Per Port Drop[0..31]
count_ucipp2_igrPortMibBlock	IP Unicast Received Counter[0..31]
count_mcipp2_igrPortMibBlock	IP Multicast Received Counter[0..31]
count_uc_routedipp2_igrPortMibBlock	IP Unicast Routed Counter[0..31]



Bank Name	Connected Registers or Tables
count_mc_routedipp2_igrPortMibBlock	IP Multicast Routed Counter[0..31]
count_mc_acl_dropipp2_igrPortMibBlock	IP Multicast ACL Drop Counter[0..31]
count_ipp2_debuglppStatisticsBlock	Debug IPP Counter[0..22]
bk_mmp_stat_2	Flow Classification And Metering Drop
bk_ingress_admission_control_all_red_en_2	Ingress Admission Control Mark All Red Enable
bk_ingress_admission_control_all_red_2	Ingress Admission Control Mark All Red
Ingress Admission Control Token Bucket Configuration Slice 2	Ingress Admission Control Token Bucket Configuration Slice 2
Ingress Admission Control Reset Slice 2	Ingress Admission Control Reset Slice 2
Ingress Admission Control Current Status Slice 2	Ingress Admission Control Current Status Slice 2
bk_erm_ss2	ERM Yellow Configuration ERM Red Configuration Egress Resource Manager Pointer[0..7] Resource Limiter Set[0..3]
count_erm_ss2	Egress Resource Manager Drop[0..7]
pb_info_regbank_ss2	Packet Buffer Status
count_drop_pa_top_switch_pb2	Buffer Overflow Drop Ingress Resource Manager Drop
pb_queue_manage_register_bank_ss2	Map Queue to Priority[0..7]
count_drop_pa_top_switch_pb2_iRequeue	Re-queue Overflow Drop
pfc_regbank_rsv_size_ss2	Port/TC Reserved[0..63]
pfc_regbank_port_rsv_size_ss2	Port Reserved[0..7]
PFC Inc Counters for ingress ports 0 to 7 Slice 2	PFC Inc Counters for ingress ports 0 to 7 Slice 2
PFC Inc Counters for ingress ports 8 to 15 Slice 2	PFC Inc Counters for ingress ports 8 to 15 Slice 2
PFC Inc Counters for ingress ports 16 to 23 Slice 2	PFC Inc Counters for ingress ports 16 to 23 Slice 2
PFC Inc Counters for ingress ports 24 to 31 Slice 2	PFC Inc Counters for ingress ports 24 to 31 Slice 2
PFC Dec Counters for ingress ports 0 to 7 Slice 2	PFC Dec Counters for ingress ports 0 to 7 Slice 2
PFC Dec Counters for ingress ports 8 to 15 Slice 2	PFC Dec Counters for ingress ports 8 to 15 Slice 2
PFC Dec Counters for ingress ports 16 to 23 Slice 2	PFC Dec Counters for ingress ports 16 to 23 Slice 2
PFC Dec Counters for ingress ports 24 to 31 Slice 2	PFC Dec Counters for ingress ports 24 to 31 Slice 2
pfc_regbank_cm_n_misc_ss2	Port FFA Used[0..7] Port Used[0..7] TC FFA Used[0..7] FFA Used PFC FFA Used non-PFC
pfc_regbank_pause_settings1_ss2	Port Pause Settings[0..7]
pfc_regbank_taildrop_settings0_ss2	Port Tail-Drop Settings[0..7]
pfc_regbank_misc_ss2	Xon FFA Threshold Xoff FFA Threshold Tail-Drop FFA Threshold TC Xon FFA Threshold[0..7] TC Xoff FFA Threshold[0..7] TC Tail-Drop FFA Threshold[0..7] Port Xon FFA Threshold[0..7] Port Xoff FFA Threshold[0..7] Port Tail-Drop FFA Threshold[0..7]



Bank Name	Connected Registers or Tables
	Port/TC Xon Total Threshold[0..63] Port/TC Xoff Total Threshold[0..63] Port/TC Tail-Drop Total Threshold[0..63]
qe_register_bank_ss2_sp0	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss2_sp1	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss2_sp2	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss2_sp3	Egress Port Depth[0..7] Egress Queue Depth[0..63]
pb_r_register_bank_ss2	Minimum Buffer Free
disable_queue_output_register_bank_ss2	Output Disable[0..7]
dwrr_bucket_capacity_settings_ss2	DWRR Bucket Capacity Configuration[0..7]
dwrr_bucket_misc_settings_ss2	DWRR Bucket Misc Configuration[0..7]
dwrr_weight_settings_ss2	DWRR Weight Configuration[0..63]
queue_shaper_rate_settings_ss2	Queue Shaper Rate Configuration
queue_shaper_bucket_settings_ss2	Queue Shaper Bucket Capacity Configuration Queue Shaper Bucket Threshold Configuration
queue_shaper_misc_ss2	Queue Shaper Enable
prio_shaper_rate_settings_ss2	Prio Shaper Rate Configuration
prio_shaper_bucket_settings_ss2	Prio Shaper Bucket Capacity Configuration Prio Shaper Bucket Threshold Configuration
prio_shaper_misc_ss2	Prio Shaper Enable
port_shaper_rate_settings_ss2	Port Shaper Rate Configuration
port_shaper_bucket_settings_ss2	Port Shaper Bucket Capacity Configuration Port Shaper Bucket Threshold Configuration
port_shaper_misc_ss2	Port Shaper Enable
count_opkt_pa_top_switch_pb2	PB Packet Head Counter PB Packet Tail Counter
drain_port_ss2	Drain Port
drain_drop_ss2	Drain Port Drop[0..7]
count_pa_top_switch_epp2_conf	Unknown Egress Drop[0..7] Egress Port Disabled Drop[0..7] Egress Port Filtering Drop[0..7] Tunnel Exit Too Small Packet Modification To Small Drop[0..7] EPP PM Drop
count_opkt_pa_top_switch_epp2_conf	EPP Packet Head Counter EPP Packet Tail Counter
Egress Port Configuration Slice 2	Egress Port Configuration Slice 2
Egress Tunnel Exit Table Slice 2	Egress Tunnel Exit Table Slice 2
Tunnel Entry Instruction Table Slice 2	Tunnel Entry Instruction Table Slice 2
Tunnel Entry Header Data Slice 2	Tunnel Entry Header Data Slice 2
Beginning of Packet Tunnel Entry Instruction Table Slice 2	Beginning of Packet Tunnel Entry Instruction Table Slice 2
L2 Tunnel Entry Instruction Table Slice 2	L2 Tunnel Entry Instruction Table Slice 2
L3 Tunnel Entry Instruction Table Slice 2	L3 Tunnel Entry Instruction Table Slice 2
Color Remap From Egress Port Slice 2	Color Remap From Egress Port Slice 2
Color Remap From Ingress Admission Control Slice 2	Color Remap From Ingress Admission Control Slice 2
Egress Router Table Slice 2	Egress Router Table Slice 2
Next Hop DA MAC Slice 2	Next Hop DA MAC Slice 2



Bank Name	Connected Registers or Tables
Router Port Egress SA MAC Address Slice 2	Router Port Egress SA MAC Address Slice 2
Next Hop MPLS Table Slice 2	Next Hop MPLS Table Slice 2
Egress MPLS TTL Table Slice 2	Egress MPLS TTL Table Slice 2
Next Hop Packet Insert MPLS Header Slice 2	Next Hop Packet Insert MPLS Header Slice 2
Egress Queue To PCP And CFI/DEI Mapping Table Slice 2	Egress Queue To PCP And CFI/DEI Mapping Table Slice 2
Egress VLAN Translation TCAM Answer Slice 2	Egress VLAN Translation TCAM Answer Slice 2
Ingress NAT Operation Slice 2	Ingress NAT Operation Slice 2
Egress NAT Operation Slice 2	Egress NAT Operation Slice 2
Select Which Egress QoS Mapping Table To Use Slice 2	Select Which Egress QoS Mapping Table To Use Slice 2
L2 QoS Mapping Table Slice 2	L2 QoS Mapping Table Slice 2
IP QoS Mapping Table Slice 2	IP QoS Mapping Table Slice 2
TOS QoS Mapping Table Slice 2	TOS QoS Mapping Table Slice 2
MPLS QoS Mapping Table Slice 2	MPLS QoS Mapping Table Slice 2
epp_register_bank_ss2	Output Mirroring Table Egress VLAN Translation TCAM Egress Queue To MPLS EXP Mapping Table Debug Counter reQueuePortId Setup Debug Counter fromPort Setup Egress MPLS Decoding Options Egress Ethernet Type for VLAN tag NAT Add Egress Port for NAT Calculation Disable CPU tag on CPU Port Debug Counter debugMatchEPP0 Setup EPP Debug imActive EPP Debug imExtra EPP Debug omEnabled EPP Debug omImActive EPP Debug reQueue EPP Debug reQueuePortId EPP Debug reQueuePkt EPP Debug fromPort EPP Debug delSpecificVlan EPP Debug updateTosExp EPP Debug isIPv4 EPP Debug isIPv6 EPP Debug addNewMpls EPP Debug isPPPoE EPP Debug debugMatchEPP0 Egress Port VID Operation
count_epp2_vrfOutStatisticsBlock	Transmitted Packets on Egress VRF[0..3]
Ingress NAT Hit Status Slice 2	Ingress NAT Hit Status Slice 2
Egress NAT Hit Status Slice 2	Egress NAT Hit Status Slice 2
count_epp2_debugEppStatisticsBlock	Debug EPP Counter[0..14]
count_ps0_ss2	PS Error Counter
count_ps1_ss2	PS Error Counter
count_ps2_ss2	PS Error Counter
count_ps3_ss2	PS Error Counter
count_ps4_ss2	PS Error Counter
count_ps5_ss2	PS Error Counter



Bank Name	Connected Registers or Tables
count_ps6_ss2	PS Error Counter
count_ps7_ss2	PS Error Counter
count_opkt_pa top switch ps2 ps_wrap.bridge	PS Packet Head Counter PS Packet Tail Counter
count_sp_ss3	SP Overflow Drop
count_broken_pkt_ss3	IPP PM Drop IPP Empty Destination Drop
count_pa top switch ipp3 conf	Unknown Ingress Drop Empty Mask Drop Ingress Spanning Tree Drop: Listen Ingress Spanning Tree Drop: Learning Ingress Spanning Tree Drop: Blocking L2 Lookup Drop Ingress Packet Filtering Drop Reserved MAC DA Drop Reserved MAC SA Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop Invalid Routing Protocol Drop Expired TTL Drop L3 Lookup Drop IP Checksum Drop Second Tunnel Exit Drop Tunnel Exit Miss Action Drop Tunnel Exit Too Small Packet Modification Drop Learning Packet Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Egress Configurable ACL Drop ARP Decoder Drop RARP Decoder Drop L2 IEEE 1588 Decoder Drop L4 IEEE 1588 Decoder Drop IEEE 802.1X and EAPOL Decoder Drop SCTP Decoder Drop LACP Decoder Drop AH Decoder Drop ESP Decoder Drop DNS Decoder Drop BOOTP and DHCP Decoder Drop CAPWAP Decoder Drop IKE Decoder Drop GRE Decoder Drop NAT Action Table Drop L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop Source Port Default ACL Action Drop
count_opkt_pa top switch ipp3 conf	IPP Packet Head Counter IPP Packet Tail Counter
Tunnel Exit Lookup TCAM Answer Slice 3	Tunnel Exit Lookup TCAM Answer Slice 3



Bank Name	Connected Registers or Tables
Ingress Admission Control Initial Pointer Slice 3	Ingress Admission Control Initial Pointer Slice 3
Ingress Configurable ACL 0 Large Table Slice 3	Ingress Configurable ACL 0 Large Table Slice 3
Ingress Configurable ACL 0 Small Table Slice 3	Ingress Configurable ACL 0 Small Table Slice 3
Ingress Configurable ACL 0 TCAM Answer Slice 3	Ingress Configurable ACL 0 TCAM Answer Slice 3
Ingress Configurable ACL 1 Large Table Slice 3	Ingress Configurable ACL 1 Large Table Slice 3
Ingress Configurable ACL 1 Small Table Slice 3	Ingress Configurable ACL 1 Small Table Slice 3
Ingress Configurable ACL 1 TCAM Answer Slice 3	Ingress Configurable ACL 1 TCAM Answer Slice 3
Ingress Configurable ACL 2 TCAM Answer Slice 3	Ingress Configurable ACL 2 TCAM Answer Slice 3
Ingress Configurable ACL 3 TCAM Answer Slice 3	Ingress Configurable ACL 3 TCAM Answer Slice 3
Source Port Default ACL Action Slice 3	Source Port Default ACL Action Slice 3
VLAN Table Slice 3	VLAN Table Slice 3
Ingress Multiple Spanning Tree State Slice 3	Ingress Multiple Spanning Tree State Slice 3
Ingress Router Table Slice 3	Ingress Router Table Slice 3
L3 LPM Result Slice 3	L3 LPM Result Slice 3
Hash Based L3 Routing Table Slice 3	Hash Based L3 Routing Table Slice 3
Next Hop Table Slice 3	Next Hop Table Slice 3
Next Hop Packet Modifications Slice 3	Next Hop Packet Modifications Slice 3
L2 Aging Status Shadow Table Slice 3	L2 Aging Status Shadow Table Slice 3
L2 DA Hash Lookup Table Slice 3	L2 DA Hash Lookup Table Slice 3
L2 Destination Table Slice 3	L2 Destination Table Slice 3
L2 SA Hash Lookup Table Slice 3	L2 SA Hash Lookup Table Slice 3
L2 Aging Status Shadow Table - Replica Slice 3	L2 Aging Status Shadow Table - Replica Slice 3
L2 Destination Table - Replica Slice 3	L2 Destination Table - Replica Slice 3
L2 Action Table Slice 3	L2 Action Table Slice 3
L2 Action Table Source Port Slice 3	L2 Action Table Source Port Slice 3
Egress ACL Rule Pointer TCAM Answer Slice 3	Egress ACL Rule Pointer TCAM Answer Slice 3
Egress Configurable ACL 0 Large Table Slice 3	Egress Configurable ACL 0 Large Table Slice 3
Egress Configurable ACL 0 Small Table Slice 3	Egress Configurable ACL 0 Small Table Slice 3
Egress Configurable ACL 0 TCAM Answer Slice 3	Egress Configurable ACL 0 TCAM Answer Slice 3
Egress Configurable ACL 1 TCAM Answer Slice 3	Egress Configurable ACL 1 TCAM Answer Slice 3
Tunnel Entry MTU Length Check Slice 3	Tunnel Entry MTU Length Check Slice 3
ipp_register.bank.ss3	Link Aggregation Ctrl Debug Counter srcPort Setup L2 Tunnel Decoder Setup Learning DA MAC SNAP LLC Decoding Options Ingress Ethernet Type for VLAN tag L2 Reserved Multicast Address Base Debug Counter nrVlans Setup



Bank Name	Connected Registers or Tables
	Debug Counter spVidOp Setup Ingress Configurable ACL 0 Selection Ingress Configurable ACL 1 Selection Debug Counter finalVid Setup Debug Counter vlanVidOp Setup Debug Counter nextHopPtrLpm Setup Debug Counter nextHopPtrHash Setup Debug Counter nextHopPtrFinal Setup Check IPv4 Header Checksum Force Non VLAN Packet To Specific Queue Force Unknown L3 Packet To Specific Egress Queue Force Non VLAN Packet To Specific Color Force Unknown L3 Packet To Specific Color Debug Counter l2DaHash Setup Debug Counter l2DaHashHitAndBucket Setup Debug Counter l2DaTcamHitsAndCast Setup Forward From CPU Port Move Options L2 Action Table Egress Port State L2 Multicast Handling Egress Configurable ACL 0 Selection Egress Port NAT State NAT Action Table Force Original Packet Ingress MMP Drop Mask Debug Counter dstPortmask Setup Debug Counter debugMatchIPP0 Setup IPP Debug srcPort IPP Debug dropPktAfterL2Decode IPP Debug nrVlans IPP Debug dropPktAfterL3Decode IPP Debug spVidOp IPP Debug finalVid IPP Debug vlanVidOp IPP Debug routerHit IPP Debug nextHopPtrLpm IPP Debug nextHopPtrHash IPP Debug nextHopPtrLpmHit IPP Debug nextHopPtrHashHit IPP Debug nextHopPtrFinal IPP Debug l2DaHashKey IPP Debug l2DaHash IPP Debug l2DaHashHitAndBucket IPP Debug l2DaTcamHitsAndCast IPP Debug routed IPP Debug isFlooding IPP Debug isBroadcast IPP Debug doL2Lookup IPP Debug dstPortmask IPP Debug debugMatchIPP0 Enable Enqueue To Ports And Queues Flooding Action Send to Port Link Aggregation To Physical Ports Members Link Aggregate Weight CPU Reason Code Operation Ingress Egress Port Packet Type Filter NAT Action Table



Bank Name	Connected Registers or Tables
	Egress Configurable ACL 1 Rules Setup Egress Configurable ACL 0 Rules Setup Allow Special Frame Check For L2 Action Table Egress Multiple Spanning Tree State Router MTU Table Hairpin Enable L2 Multicast Table L2 Lookup Collision Table Masks L2 Lookup Collision Table L2 Aging Collision Shadow Table Router Egress Queue To VLAN Data MPLS EXP Field To Packet Color Mapping Table IPv6 Class of Service Field To Packet Color Mapping Table IPv4 TOS Field To Packet Color Mapping Table VLAN PCP And DEI To Color Mapping Table MPLS EXP Field To Egress Queue Mapping Table IPv6 Class of Service Field To Egress Queue Mapping Table IPv4 TOS Field To Egress Queue Mapping Table VLAN PCP To Queue Mapping Table L3 Routing Default Ingress VID Ethernet Type Range Assignment Answer Ingress VID Ethernet Type Range Search Data Ingress VID Inner VID Range Assignment Answer Ingress VID Inner VID Range Search Data Ingress VID Outer VID Range Assignment Answer Ingress VID Outer VID Range Search Data Ingress VID MAC Range Assignment Answer Ingress Configurable ACL 3 Rules Setup Ingress Configurable ACL 2 Rules Setup Ingress Configurable ACL 2 Pre Lookup Ingress Configurable ACL 1 Rules Setup Ingress Configurable ACL 1 Pre Lookup Ingress Configurable ACL 0 Rules Setup Ingress Configurable ACL 0 Pre Lookup Ingress Port Packet Type Filter SMON Set Search Default Packet To CPU Modification L2 Reserved Multicast Address Action Second Tunnel Exit Miss Action Second Tunnel Exit Lookup TCAM Answer Link Aggregation Membership Source Port Table Egress ACL Rule Pointer TCAM ARP Packet Decoder Options RARP Packet Decoder Options IEEE 1588 L2 Packet Decoder Options IEEE 802.1X and EAPOL Packet Decoder Options GRE Packet Decoder Options SCTP Packet Decoder Options LACP Packet Decoder Options AH Header Packet Decoder Options ESP Header Packet Decoder Options DNS Packet Decoder Options BOOTP and DHCP Packet Decoder Options CAPWAP Packet Decoder Options IKE Packet Decoder Options



Bank Name	Connected Registers or Tables
	Debug Counter I2DaHashKey Setup Egress Spanning Tree State Tunnel Exit Lookup TCAM Ingress Configurable ACL 0 TCAM IEEE 1588 L4 Packet Decoder Options Ingress Configurable ACL 0 Search Mask Second Tunnel Exit Lookup TCAM Router Port MAC Address Ingress VID MAC Range Search Data Ingress Configurable ACL 3 TCAM Reserved Source MAC Address Range Reserved Destination MAC Address Range Send to CPU LLDP Configuration Ingress Configurable ACL 1 Search Mask Egress Configurable ACL 0 Search Mask Egress Configurable ACL 0 TCAM Ingress Configurable ACL 1 TCAM Ingress Configurable ACL 2 TCAM Egress Configurable ACL 1 TCAM
ipp_register_bank_misc_ss3	Ingress Drop Options
L3 Routing TCAM Slice 3	L3 Routing TCAM Slice 3
count_packets_ipp3_smonStatisticsBlock	SMON Set 0 Packet Counter[0..7] SMON Set 1 Packet Counter[0..7] SMON Set 2 Packet Counter[0..7] SMON Set 3 Packet Counter[0..7]
count_bytes_ipp3_smonStatisticsBlock	SMON Set 0 Byte Counter[0..7] SMON Set 1 Byte Counter[0..7] SMON Set 2 Byte Counter[0..7] SMON Set 3 Byte Counter[0..7]
count_ipp3_aclConfStatisticsBlock	Ingress Configurable ACL Match Counter[0..63]
count_ipp3_vrflnStatisticsBlock	Received Packets on Ingress VRF[0..3]
Next Hop Hit Status Slice 3	Next Hop Hit Status Slice 3
count_ipp3_egressAclStatisticsBlock	Egress Configurable ACL Match Counter[0..63]
count_ipp3_egressDropStatisticsBlock	Queue Off Drop[0..31] Egress Spanning Tree Drop[0..31] MBSC Drop[0..31] Ingress-Egress Packet Filtering Drop[0..31] L2 Action Table Per Port Drop[0..31]
count_ucipp3_igrPortMibBlock	IP Unicast Received Counter[0..31]
count_mcipp3_igrPortMibBlock	IP Multicast Received Counter[0..31]
count_uc_routedipp3_igrPortMibBlock	IP Unicast Routed Counter[0..31]
count_mc_routedipp3_igrPortMibBlock	IP Multicast Routed Counter[0..31]
count_mc_acl_dropipp3_igrPortMibBlock	IP Multicast ACL Drop Counter[0..31]
count_ipp3_debugIppStatisticsBlock	Debug IPP Counter[0..22]
bk_mmp_stat_3	Flow Classification And Metering Drop
bk_ingress_admission_control_all_red_en_3	Ingress Admission Control Mark All Red Enable
bk_ingress_admission_control_all_red_3	Ingress Admission Control Mark All Red
Ingress Admission Control Token Bucket Configuration Slice 3	Ingress Admission Control Token Bucket Configuration Slice 3
Ingress Admission Control Reset Slice 3	Ingress Admission Control Reset Slice 3
Ingress Admission Control Current Status Slice 3	Ingress Admission Control Current Status Slice 3
bk_erm_ss3	ERM Yellow Configuration ERM Red Configuration



Bank Name	Connected Registers or Tables
	Egress Resource Manager Pointer[0..7] Resource Limiter Set[0..3]
count_erm_ss3	Egress Resource Manager Drop[0..7]
pb_info_regbank_ss3	Packet Buffer Status
count_drop_pa top switch pb3	Buffer Overflow Drop Ingress Resource Manager Drop
pb_queue_manage_register_bank_ss3	Map Queue to Priority[0..7]
count_drop_pa top switch pb3 iRequeue	Re-queue Overflow Drop
pfc_regbank_rsv_size_ss3	Port/TC Reserved[0..63]
pfc_regbank_port_rsv_size_ss3	Port Reserved[0..7]
PFC Inc Counters for ingress ports 0 to 7 Slice 3	PFC Inc Counters for ingress ports 0 to 7 Slice 3
PFC Inc Counters for ingress ports 8 to 15 Slice 3	PFC Inc Counters for ingress ports 8 to 15 Slice 3
PFC Inc Counters for ingress ports 16 to 23 Slice 3	PFC Inc Counters for ingress ports 16 to 23 Slice 3
PFC Inc Counters for ingress ports 24 to 31 Slice 3	PFC Inc Counters for ingress ports 24 to 31 Slice 3
PFC Dec Counters for ingress ports 0 to 7 Slice 3	PFC Dec Counters for ingress ports 0 to 7 Slice 3
PFC Dec Counters for ingress ports 8 to 15 Slice 3	PFC Dec Counters for ingress ports 8 to 15 Slice 3
PFC Dec Counters for ingress ports 16 to 23 Slice 3	PFC Dec Counters for ingress ports 16 to 23 Slice 3
PFC Dec Counters for ingress ports 24 to 31 Slice 3	PFC Dec Counters for ingress ports 24 to 31 Slice 3
pfc_regbank_cm_n_misc_ss3	Port FFA Used[0..7] Port Used[0..7] TC FFA Used[0..7] FFA Used PFC FFA Used non-PFC
pfc_regbank_pause_settings1_ss3	Port Pause Settings[0..7]
pfc_regbank_tai_l_drop_settings0_ss3	Port Tail-Drop Settings[0..7]
pfc_regbank_misc_ss3	Xon FFA Threshold Xoff FFA Threshold Tail-Drop FFA Threshold TC Xon FFA Threshold[0..7] TC Xoff FFA Threshold[0..7] TC Tail-Drop FFA Threshold[0..7] Port Xon FFA Threshold[0..7] Port Xoff FFA Threshold[0..7] Port Tail-Drop FFA Threshold[0..7] Port/TC Xon Total Threshold[0..63] Port/TC Xoff Total Threshold[0..63] Port/TC Tail-Drop Total Threshold[0..63]
qe_register_bank_ss3_sp0	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss3_sp1	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss3_sp2	Egress Port Depth[0..7] Egress Queue Depth[0..63]
qe_register_bank_ss3_sp3	Egress Port Depth[0..7] Egress Queue Depth[0..63]
pb_r_register_bank_ss3	Minimum Buffer Free



Bank Name	Connected Registers or Tables
disable_queue_output_register_bank_ss3	Output Disable[0..7]
dwrr_bucket_capacity_settings_ss3	DWRR Bucket Capacity Configuration[0..7]
dwrr_bucket_misc_settings_ss3	DWRR Bucket Misc Configuration[0..7]
dwrr_weight_settings_ss3	DWRR Weight Configuration[0..63]
queue_shaper_rate_settings_ss3	Queue Shaper Rate Configuration
queue_shaper_bucket_settings_ss3	Queue Shaper Bucket Capacity Configuration Queue Shaper Bucket Threshold Configuration
queue_shaper_misc_ss3	Queue Shaper Enable
prio_shaper_rate_settings_ss3	Prio Shaper Rate Configuration
prio_shaper_bucket_settings_ss3	Prio Shaper Bucket Capacity Configuration Prio Shaper Bucket Threshold Configuration
prio_shaper_misc_ss3	Prio Shaper Enable
port_shaper_rate_settings_ss3	Port Shaper Rate Configuration
port_shaper_bucket_settings_ss3	Port Shaper Bucket Capacity Configuration Port Shaper Bucket Threshold Configuration
port_shaper_misc_ss3	Port Shaper Enable
count_opkt.pa top switch pb3	PB Packet Head Counter PB Packet Tail Counter
drain_port_ss3	Drain Port
drain_drop_ss3	Drain Port Drop[0..7]
count.pa top switch epp3 conf	Unknown Egress Drop[0..7] Egress Port Disabled Drop[0..7] Egress Port Filtering Drop[0..7] Tunnel Exit Too Small Packet Modification To Small Drop[0..7] EPP PM Drop
count_opkt.pa top switch epp3 conf	EPP Packet Head Counter EPP Packet Tail Counter
Egress Port Configuration Slice 3	Egress Port Configuration Slice 3
Egress Tunnel Exit Table Slice 3	Egress Tunnel Exit Table Slice 3
Tunnel Entry Instruction Table Slice 3	Tunnel Entry Instruction Table Slice 3
Tunnel Entry Header Data Slice 3	Tunnel Entry Header Data Slice 3
Beginning of Packet Tunnel Entry Instruction Table Slice 3	Beginning of Packet Tunnel Entry Instruction Table Slice 3
L2 Tunnel Entry Instruction Table Slice 3	L2 Tunnel Entry Instruction Table Slice 3
L3 Tunnel Entry Instruction Table Slice 3	L3 Tunnel Entry Instruction Table Slice 3
Color Remap From Egress Port Slice 3	Color Remap From Egress Port Slice 3
Color Remap From Ingress Admission Control Slice 3	Color Remap From Ingress Admission Control Slice 3
Egress Router Table Slice 3	Egress Router Table Slice 3
Next Hop DA MAC Slice 3	Next Hop DA MAC Slice 3
Router Port Egress SA MAC Address Slice 3	Router Port Egress SA MAC Address Slice 3
Next Hop MPLS Table Slice 3	Next Hop MPLS Table Slice 3
Egress MPLS TTL Table Slice 3	Egress MPLS TTL Table Slice 3
Next Hop Packet Insert MPLS Header Slice 3	Next Hop Packet Insert MPLS Header Slice 3
Egress Queue To PCP And CFI/DEI Mapping Table Slice 3	Egress Queue To PCP And CFI/DEI Mapping Table Slice 3
Egress VLAN Translation TCAM Answer Slice 3	Egress VLAN Translation TCAM Answer Slice 3
Ingress NAT Operation Slice 3	Ingress NAT Operation Slice 3
Egress NAT Operation Slice 3	Egress NAT Operation Slice 3



Bank Name	Connected Registers or Tables
Select Which Egress QoS Mapping Table To Use Slice 3	Select Which Egress QoS Mapping Table To Use Slice 3
L2 QoS Mapping Table Slice 3	L2 QoS Mapping Table Slice 3
IP QoS Mapping Table Slice 3	IP QoS Mapping Table Slice 3
TOS QoS Mapping Table Slice 3	TOS QoS Mapping Table Slice 3
MPLS QoS Mapping Table Slice 3	MPLS QoS Mapping Table Slice 3
epp_register_bank_ss3	Output Mirroring Table Egress VLAN Translation TCAM Egress Queue To MPLS EXP Mapping Table Debug Counter reQueuePortId Setup Debug Counter fromPort Setup Egress MPLS Decoding Options Egress Ethernet Type for VLAN tag NAT Add Egress Port for NAT Calculation Disable CPU tag on CPU Port Debug Counter debugMatchEPP0 Setup EPP Debug imActive EPP Debug imExtra EPP Debug omEnabled EPP Debug omImActive EPP Debug reQueue EPP Debug reQueuePortId EPP Debug reQueuePkt EPP Debug fromPort EPP Debug delSpecificVlan EPP Debug updateTosExp EPP Debug isIPv4 EPP Debug isIPv6 EPP Debug addNewMpls EPP Debug isPPPoE EPP Debug debugMatchEPP0 Egress Port VID Operation
count_epp3_vrfOutStatisticsBlock	Transmitted Packets on Egress VRF[0..3]
Ingress NAT Hit Status Slice 3	Ingress NAT Hit Status Slice 3
Egress NAT Hit Status Slice 3	Egress NAT Hit Status Slice 3
count_epp3_debugEppStatisticsBlock	Debug EPP Counter[0..14]
count_ps0_ss3	PS Error Counter
count_ps1_ss3	PS Error Counter
count_ps2_ss3	PS Error Counter
count_ps3_ss3	PS Error Counter
count_ps4_ss3	PS Error Counter
count_ps5_ss3	PS Error Counter
count_ps6_ss3	PS Error Counter
count_ps7_ss3	PS Error Counter
count_opkt_pa top switch ps3 ps_wrap_bridge	PS Packet Head Counter PS Packet Tail Counter

35.4 Registers and Tables in Alphabetical Order



Name	Slice	Address Range
AH Decoder Drop	Slice 0	4928
	Slice 1	102608
	Slice 2	200288
	Slice 3	297968
AH Header Packet Decoder Options	Slice 0	85501
	Slice 1	183181
	Slice 2	280861
	Slice 3	378541
ARP Decoder Drop	Slice 0	4921
	Slice 1	102601
	Slice 2	200281
	Slice 3	297961
ARP Packet Decoder Options	Slice 0	85487
	Slice 1	183167
	Slice 2	280847
	Slice 3	378527
Aging Data FIFO		4801
Aging Data FIFO High Watermark Level		603
Allow Special Frame Check For L2 Action Table	Slice 0	82887 - 82890
	Slice 1	180567 - 180570
	Slice 2	278247 - 278250
	Slice 3	375927 - 375930
BOOTP and DHCP Decoder Drop	Slice 0	4931
	Slice 1	102611
	Slice 2	200291
	Slice 3	297971
BOOTP and DHCP Packet Decoder Options	Slice 0	85507
	Slice 1	183187
	Slice 2	280867
	Slice 3	378547
Beginning of Packet Tunnel Entry Instruction Table	Slice 0	91822 - 91837
	Slice 1	189502 - 189517
	Slice 2	287182 - 287197
	Slice 3	384862 - 384877
Buffer Free		1
Buffer Overflow Drop	Slice 0	89751
	Slice 1	187431
	Slice 2	285111
	Slice 3	382791
CAPWAP Decoder Drop	Slice 0	4932
	Slice 1	102612
	Slice 2	200292
	Slice 3	297972
CAPWAP Packet Decoder Options	Slice 0	85509
	Slice 1	183189
	Slice 2	280869
	Slice 3	378549
CPU Reason Code Operation	Slice 0	82315 - 82330
	Slice 1	179995 - 180010
	Slice 2	277675 - 277690
	Slice 3	375355 - 375370
Check IPv4 Header Checksum	Slice 0	81922
	Slice 1	179602
	Slice 2	277282
	Slice 3	374962

Name	Slice	Address Range
Color Remap From Egress Port	Slice 0	91870 - 91901
	Slice 1	189550 - 189581
	Slice 2	287230 - 287261
	Slice 3	384910 - 384941
Color Remap From Ingress Admission Control	Slice 0	91902 - 91933
	Slice 1	189582 - 189613
	Slice 2	287262 - 287293
	Slice 3	384942 - 384973
Core Tick Configuration		3
Core Tick Select		4
Core Version		0
DNS Decoder Drop	Slice 0	4930
	Slice 1	102610
	Slice 2	200290
	Slice 3	297970
DNS Packet Decoder Options	Slice 0	85505
	Slice 1	183185
	Slice 2	280865
	Slice 3	378545
DWRR Bucket Capacity Configuration	Slice 0	90928 - 90935
	Slice 1	188608 - 188615
	Slice 2	286288 - 286295
	Slice 3	383968 - 383975
DWRR Bucket Misc Configuration	Slice 0	90936 - 90943
	Slice 1	188616 - 188623
	Slice 2	286296 - 286303
	Slice 3	383976 - 383983
DWRR Weight Configuration	Slice 0	90944 - 91007
	Slice 1	188624 - 188687
	Slice 2	286304 - 286367
	Slice 3	383984 - 384047
Debug Counter debugMatchEPP0 Setup	Slice 0	99300
	Slice 1	196980
	Slice 2	294660
	Slice 3	392340
Debug Counter debugMatchIPP0 Setup	Slice 0	81939
	Slice 1	179619
	Slice 2	277299
	Slice 3	374979
Debug Counter dstPortmask Setup	Slice 0	81938
	Slice 1	179618
	Slice 2	277298
	Slice 3	374978
Debug Counter finalVid Setup	Slice 0	81917
	Slice 1	179597
	Slice 2	277277
	Slice 3	374957
Debug Counter fromPort Setup	Slice 0	99295
	Slice 1	196975
	Slice 2	294655
	Slice 3	392335
Debug Counter I2DaHash Setup	Slice 0	81927
	Slice 1	179607
	Slice 2	277287
	Slice 3	374967



Name	Slice	Address Range
Debug Counter I2DaHashHitAndBucket Setup	Slice 0	81928
	Slice 1	179608
	Slice 2	277288
	Slice 3	374968
Debug Counter I2DaHashKey Setup	Slice 0	85513
	Slice 1	183193
	Slice 2	280873
	Slice 3	378553
Debug Counter I2DaTcamHitsAndCast Setup	Slice 0	81929
	Slice 1	179609
	Slice 2	277289
	Slice 3	374969
Debug Counter nextHopPtrFinal Setup	Slice 0	81921
	Slice 1	179601
	Slice 2	277281
	Slice 3	374961
Debug Counter nextHopPtrHash Setup	Slice 0	81920
	Slice 1	179600
	Slice 2	277280
	Slice 3	374960
Debug Counter nextHopPtrLpm Setup	Slice 0	81919
	Slice 1	179599
	Slice 2	277279
	Slice 3	374959
Debug Counter nrVlans Setup	Slice 0	81913
	Slice 1	179593
	Slice 2	277273
	Slice 3	374953
Debug Counter reQueuePortId Setup	Slice 0	99294
	Slice 1	196974
	Slice 2	294654
	Slice 3	392334
Debug Counter spVidOp Setup	Slice 0	81914
	Slice 1	179594
	Slice 2	277274
	Slice 3	374954
Debug Counter srcPort Setup	Slice 0	81907
	Slice 1	179587
	Slice 2	277267
	Slice 3	374947
Debug Counter vlanVidOp Setup	Slice 0	81918
	Slice 1	179598
	Slice 2	277278
	Slice 3	374958
Debug EPP Counter	Slice 0	102424 - 102438
	Slice 1	200104 - 200118
	Slice 2	297784 - 297798
	Slice 3	395464 - 395478
Debug IPP Counter	Slice 0	89466 - 89488
	Slice 1	187146 - 187168
	Slice 2	284826 - 284848
	Slice 3	382506 - 382528

Name	Slice	Address Range
Default Packet To CPU Modification	Slice 0	84955 - 84986
	Slice 1	182635 - 182666
	Slice 2	280315 - 280346
	Slice 3	377995 - 378026
Disable CPU tag on CPU Port	Slice 0	99299
	Slice 1	196979
	Slice 2	294659
	Slice 3	392339
Drain Port	Slice 0	91458
	Slice 1	189138
	Slice 2	286818
	Slice 3	384498
Drain Port Drop	Slice 0	91459 - 91466
	Slice 1	189139 - 189146
	Slice 2	286819 - 286826
	Slice 3	384499 - 384506
EPP Debug addNewMpls	Slice 0	99313
	Slice 1	196993
	Slice 2	294673
	Slice 3	392353
EPP Debug debugMatchEPP0	Slice 0	99315
	Slice 1	196995
	Slice 2	294675
	Slice 3	392355
EPP Debug delSpecificVlan	Slice 0	99309
	Slice 1	196989
	Slice 2	294669
	Slice 3	392349
EPP Debug fromPort	Slice 0	99308
	Slice 1	196988
	Slice 2	294668
	Slice 3	392348
EPP Debug imActive	Slice 0	99301
	Slice 1	196981
	Slice 2	294661
	Slice 3	392341
EPP Debug imExtra	Slice 0	99302
	Slice 1	196982
	Slice 2	294662
	Slice 3	392342
EPP Debug isIPv4	Slice 0	99311
	Slice 1	196991
	Slice 2	294671
	Slice 3	392351
EPP Debug isIPv6	Slice 0	99312
	Slice 1	196992
	Slice 2	294672
	Slice 3	392352
EPP Debug isPPPoE	Slice 0	99314
	Slice 1	196994
	Slice 2	294674
	Slice 3	392354

Name	Slice	Address Range
EPP Debug omEnabled	Slice 0	99303
	Slice 1	196983
	Slice 2	294663
	Slice 3	392343
EPP Debug omImActive	Slice 0	99304
	Slice 1	196984
	Slice 2	294664
	Slice 3	392344
EPP Debug reQueue	Slice 0	99305
	Slice 1	196985
	Slice 2	294665
	Slice 3	392345
EPP Debug reQueuePkt	Slice 0	99307
	Slice 1	196987
	Slice 2	294667
	Slice 3	392347
EPP Debug reQueuePortId	Slice 0	99306
	Slice 1	196986
	Slice 2	294666
	Slice 3	392346
EPP Debug updateTosExp	Slice 0	99310
	Slice 1	196990
	Slice 2	294670
	Slice 3	392350
EPP PM Drop	Slice 0	91499
	Slice 1	189179
	Slice 2	286859
	Slice 3	384539
EPP Packet Head Counter	Slice 0	91500
	Slice 1	189180
	Slice 2	286860
	Slice 3	384540
EPP Packet Tail Counter	Slice 0	91501
	Slice 1	189181
	Slice 2	286861
	Slice 3	384541
ERM Red Configuration	Slice 0	89729
	Slice 1	187409
	Slice 2	285089
	Slice 3	382769
ERM Yellow Configuration	Slice 0	89728
	Slice 1	187408
	Slice 2	285088
	Slice 3	382768
ESP Decoder Drop	Slice 0	4929
	Slice 1	102609
	Slice 2	200289
	Slice 3	297969
ESP Header Packet Decoder Options	Slice 0	85503
	Slice 1	183183
	Slice 2	280863
	Slice 3	378543

Name	Slice	Address Range
Egress ACL Rule Pointer TCAM	Slice 0	85359 - 85486
	Slice 1	183039 - 183166
	Slice 2	280719 - 280846
	Slice 3	378399 - 378526
Egress ACL Rule Pointer TCAM Answer	Slice 0	76674 - 76737
	Slice 1	174354 - 174417
	Slice 2	272034 - 272097
	Slice 3	369714 - 369777
Egress Configurable ACL 0 Large Table	Slice 0	76738 - 80833
	Slice 1	174418 - 178513
	Slice 2	272098 - 276193
	Slice 3	369778 - 373873
Egress Configurable ACL 0 Rules Setup	Slice 0	82879 - 82886
	Slice 1	180559 - 180566
	Slice 2	278239 - 278246
	Slice 3	375919 - 375926
Egress Configurable ACL 0 Search Mask	Slice 0	86317
	Slice 1	183997
	Slice 2	281677
	Slice 3	379357
Egress Configurable ACL 0 Selection	Slice 0	81934
	Slice 1	179614
	Slice 2	277294
	Slice 3	374974
Egress Configurable ACL 0 Small Table	Slice 0	80834 - 81857
	Slice 1	178514 - 179537
	Slice 2	276194 - 277217
	Slice 3	373874 - 374897
Egress Configurable ACL 0 TCAM	Slice 0	86325 - 86452
	Slice 1	184005 - 184132
	Slice 2	281685 - 281812
	Slice 3	379365 - 379492
Egress Configurable ACL 0 TCAM Answer	Slice 0	81858 - 81873
	Slice 1	179538 - 179553
	Slice 2	277218 - 277233
	Slice 3	374898 - 374913
Egress Configurable ACL 1 Rules Setup	Slice 0	82875 - 82878
	Slice 1	180555 - 180558
	Slice 2	278235 - 278238
	Slice 3	375915 - 375918
Egress Configurable ACL 1 TCAM	Slice 0	87285 - 87796
	Slice 1	184965 - 185476
	Slice 2	282645 - 283156
	Slice 3	380325 - 380836
Egress Configurable ACL 1 TCAM Answer	Slice 0	81874 - 81889
	Slice 1	179554 - 179569
	Slice 2	277234 - 277249
	Slice 3	374914 - 374929
Egress Configurable ACL Drop	Slice 0	4920
	Slice 1	102600
	Slice 2	200280
	Slice 3	297960



Name	Slice	Address Range
Egress Configurable ACL Match Counter	Slice 0	89082 - 89145
	Slice 1	186762 - 186825
	Slice 2	284442 - 284505
	Slice 3	382122 - 382185
Egress Ethernet Type for VLAN tag	Slice 0	99297
	Slice 1	196977
	Slice 2	294657
	Slice 3	392337
Egress MPLS Decoding Options	Slice 0	99296
	Slice 1	196976
	Slice 2	294656
	Slice 3	392336
Egress MPLS TTL Table	Slice 0	93994 - 93997
	Slice 1	191674 - 191677
	Slice 2	289354 - 289357
	Slice 3	387034 - 387037
Egress Multiple Spanning Tree State	Slice 0	82891 - 82906
	Slice 1	180571 - 180586
	Slice 2	278251 - 278266
	Slice 3	375931 - 375946
Egress NAT Hit Status	Slice 0	101400 - 102423
	Slice 1	199080 - 200103
	Slice 2	296760 - 297783
	Slice 3	394440 - 395463
Egress NAT Operation	Slice 0	96246 - 97269
	Slice 1	193926 - 194949
	Slice 2	291606 - 292629
	Slice 3	389286 - 390309
Egress Port Configuration	Slice 0	91502 - 91533
	Slice 1	189182 - 189213
	Slice 2	286862 - 286893
	Slice 3	384542 - 384573
Egress Port Depth	Slice 0 to 0	90631 - 90638
	Slice 1 to 0	90703 - 90710
	Slice 2 to 0	90775 - 90782
	Slice 3 to 0	90847 - 90854
	Slice 0 to 1	188311 - 188318
	Slice 1 to 1	188383 - 188390
	Slice 2 to 1	188455 - 188462
	Slice 3 to 1	188527 - 188534
	Slice 0 to 2	285991 - 285998
	Slice 1 to 2	286063 - 286070
	Slice 2 to 2	286135 - 286142
	Slice 3 to 2	286207 - 286214
	Slice 0 to 3	383671 - 383678
	Slice 1 to 3	383743 - 383750
	Slice 2 to 3	383815 - 383822
	Slice 3 to 3	383887 - 383894
Egress Port Disabled Drop	Slice 0	91475 - 91482
	Slice 1	189155 - 189162
	Slice 2	286835 - 286842
	Slice 3	384515 - 384522

Name	Slice	Address Range
Egress Port Filtering Drop	Slice 0	91483 - 91490
	Slice 1	189163 - 189170
	Slice 2	286843 - 286850
	Slice 3	384523 - 384530
Egress Port NAT State	Slice 0	81935
	Slice 1	179615
	Slice 2	277295
	Slice 3	374975
Egress Port VID Operation	Slice 0	99316 - 99347
	Slice 1	196996 - 197027
	Slice 2	294676 - 294707
	Slice 3	392356 - 392387
Egress Queue Depth	Slice 0 to 0	90639 - 90702
	Slice 1 to 0	90711 - 90774
	Slice 2 to 0	90783 - 90846
	Slice 3 to 0	90855 - 90918
	Slice 0 to 1	188319 - 188382
	Slice 1 to 1	188391 - 188454
	Slice 2 to 1	188463 - 188526
	Slice 3 to 1	188535 - 188598
	Slice 0 to 2	285999 - 286062
	Slice 1 to 2	286071 - 286134
	Slice 2 to 2	286143 - 286206
	Slice 3 to 2	286215 - 286278
	Slice 0 to 3	383679 - 383742
	Slice 1 to 3	383751 - 383814
	Slice 2 to 3	383823 - 383886
	Slice 3 to 3	383895 - 383958
Egress Queue To MPLS EXP Mapping Table	Slice 0	99286 - 99293
	Slice 1	196966 - 196973
	Slice 2	294646 - 294653
	Slice 3	392326 - 392333
Egress Queue To PCP And CFI/DEI Mapping Table	Slice 0	94062 - 94069
	Slice 1	191742 - 191749
	Slice 2	289422 - 289429
	Slice 3	387102 - 387109
Egress Resource Manager Drop	Slice 0	89742 - 89749
	Slice 1	187422 - 187429
	Slice 2	285102 - 285109
	Slice 3	382782 - 382789
Egress Resource Manager Pointer	Slice 0	89730 - 89737
	Slice 1	187410 - 187417
	Slice 2	285090 - 285097
	Slice 3	382770 - 382777
Egress Router Table	Slice 0	91934 - 91937
	Slice 1	189614 - 189617
	Slice 2	287294 - 287297
	Slice 3	384974 - 384977
Egress Spanning Tree Drop	Slice 0	89178 - 89209
	Slice 1	186858 - 186889
	Slice 2	284538 - 284569
	Slice 3	382218 - 382249



Name	Slice	Address Range
Egress Spanning Tree State	Slice 0	85515
	Slice 1	183195
	Slice 2	280875
	Slice 3	378555
Egress Tunnel Exit Table	Slice 0	91534 - 91549
	Slice 1	189214 - 189229
	Slice 2	286894 - 286909
	Slice 3	384574 - 384589
Egress VLAN Translation TCAM	Slice 0	99158 - 99285
	Slice 1	196838 - 196965
	Slice 2	294518 - 294645
	Slice 3	392198 - 392325
Egress VLAN Translation TCAM Answer	Slice 0	94070 - 94197
	Slice 1	191750 - 191877
	Slice 2	289430 - 289557
	Slice 3	387110 - 387237
Empty Mask Drop	Slice 0	4899
	Slice 1	102579
	Slice 2	200259
	Slice 3	297939
Enable Enqueue To Ports And Queues	Slice 0	81963 - 81994
	Slice 1	179643 - 179674
	Slice 2	277323 - 277354
	Slice 3	375003 - 375034
Expired TTL Drop	Slice 0	4911
	Slice 1	102591
	Slice 2	200271
	Slice 3	297951
FFA Used PFC	Slice 0	90370
	Slice 1	188050
	Slice 2	285730
	Slice 3	383410
FFA Used non-PFC	Slice 0	90371
	Slice 1	188051
	Slice 2	285731
	Slice 3	383411
Flooding Action Send to Port	Slice 0	81995 - 82026
	Slice 1	179675 - 179706
	Slice 2	277355 - 277386
	Slice 3	375035 - 375066
Flow Classification And Metering Drop	Slice 0	89489
	Slice 1	187169
	Slice 2	284849
	Slice 3	382529
Force Non VLAN Packet To Specific Color	Slice 0	81925
	Slice 1	179605
	Slice 2	277285
	Slice 3	374965
Force Non VLAN Packet To Specific Queue	Slice 0	81923
	Slice 1	179603
	Slice 2	277283
	Slice 3	374963

Name	Slice	Address Range
Force Unknown L3 Packet To Specific Color	Slice 0	81926
	Slice 1	179606
	Slice 2	277286
	Slice 3	374966
Force Unknown L3 Packet To Specific Egress Queue	Slice 0	81924
	Slice 1	179604
	Slice 2	277284
	Slice 3	374964
Forward From CPU	Slice 0	81930
	Slice 1	179610
	Slice 2	277290
	Slice 3	374970
GRE Decoder Drop	Slice 0	4934
	Slice 1	102614
	Slice 2	200294
	Slice 3	297974
GRE Packet Decoder Options	Slice 0	85495
	Slice 1	183175
	Slice 2	280855
	Slice 3	378535
Hairpin Enable	Slice 0	83035 - 83066
	Slice 1	180715 - 180746
	Slice 2	278395 - 278426
	Slice 3	376075 - 376106
Hardware Learning Configuration		605 - 636
Hardware Learning Counter		670 - 701
Hash Based L3 Routing Table	Slice 0	41538 - 49729
	Slice 1	139218 - 147409
	Slice 2	236898 - 245089
	Slice 3	334578 - 342769
Hit Update Data FIFO		4802
Hit Update Data FIFO High Watermark Level		604
IEEE 1588 L2 Packet Decoder Options	Slice 0	85491
	Slice 1	183171
	Slice 2	280851
	Slice 3	378531
IEEE 1588 L4 Packet Decoder Options	Slice 0	86029
	Slice 1	183709
	Slice 2	281389
	Slice 3	379069
IEEE 802.1X and EAPOL Decoder Drop	Slice 0	4925
	Slice 1	102605
	Slice 2	200285
	Slice 3	297965
IEEE 802.1X and EAPOL Packet Decoder Options	Slice 0	85493
	Slice 1	183173
	Slice 2	280853
	Slice 3	378533
IKE Decoder Drop	Slice 0	4933
	Slice 1	102613
	Slice 2	200293
	Slice 3	297973



Name	Slice	Address Range
IKE Packet Decoder Options	Slice 0	85511
	Slice 1	183191
	Slice 2	280871
	Slice 3	378551
IP Checksum Drop	Slice 0	4913
	Slice 1	102593
	Slice 2	200273
	Slice 3	297953
IP Multicast ACL Drop Counter	Slice 0	89434 - 89465
	Slice 1	187114 - 187145
	Slice 2	284794 - 284825
	Slice 3	382474 - 382505
IP Multicast Received Counter	Slice 0	89338 - 89369
	Slice 1	187018 - 187049
	Slice 2	284698 - 284729
	Slice 3	382378 - 382409
IP Multicast Routed Counter	Slice 0	89402 - 89433
	Slice 1	187082 - 187113
	Slice 2	284762 - 284793
	Slice 3	382442 - 382473
IP QoS Mapping Table	Slice 0	97846 - 98101
	Slice 1	195526 - 195781
	Slice 2	293206 - 293461
	Slice 3	390886 - 391141
IP Unicast Received Counter	Slice 0	89306 - 89337
	Slice 1	186986 - 187017
	Slice 2	284666 - 284697
	Slice 3	382346 - 382377
IP Unicast Routed Counter	Slice 0	89370 - 89401
	Slice 1	187050 - 187081
	Slice 2	284730 - 284761
	Slice 3	382410 - 382441
IPP Debug debugMatchIPP0	Slice 0	81962
	Slice 1	179642
	Slice 2	277322
	Slice 3	375002
IPP Debug doL2Lookup	Slice 0	81960
	Slice 1	179640
	Slice 2	277320
	Slice 3	375000
IPP Debug dropPktAfterL2Decode	Slice 0	81941
	Slice 1	179621
	Slice 2	277301
	Slice 3	374981
IPP Debug dropPktAfterL3Decode	Slice 0	81943
	Slice 1	179623
	Slice 2	277303
	Slice 3	374983
IPP Debug dstPortmask	Slice 0	81961
	Slice 1	179641
	Slice 2	277321
	Slice 3	375001

Name	Slice	Address Range
IPP Debug finalVid	Slice 0	81945
	Slice 1	179625
	Slice 2	277305
	Slice 3	374985
IPP Debug isBroadcast	Slice 0	81959
	Slice 1	179639
	Slice 2	277319
	Slice 3	374999
IPP Debug isFlooding	Slice 0	81958
	Slice 1	179638
	Slice 2	277318
	Slice 3	374998
IPP Debug I2DaHash	Slice 0	81954
	Slice 1	179634
	Slice 2	277314
	Slice 3	374994
IPP Debug I2DaHashHitAndBucket	Slice 0	81955
	Slice 1	179635
	Slice 2	277315
	Slice 3	374995
IPP Debug I2DaHashKey	Slice 0	81953
	Slice 1	179633
	Slice 2	277313
	Slice 3	374993
IPP Debug I2DaTcamHitsAndCast	Slice 0	81956
	Slice 1	179636
	Slice 2	277316
	Slice 3	374996
IPP Debug nextHopPtrFinal	Slice 0	81952
	Slice 1	179632
	Slice 2	277312
	Slice 3	374992
IPP Debug nextHopPtrHash	Slice 0	81949
	Slice 1	179629
	Slice 2	277309
	Slice 3	374989
IPP Debug nextHopPtrHashHit	Slice 0	81951
	Slice 1	179631
	Slice 2	277311
	Slice 3	374991
IPP Debug nextHopPtrLpm	Slice 0	81948
	Slice 1	179628
	Slice 2	277308
	Slice 3	374988
IPP Debug nextHopPtrLpmHit	Slice 0	81950
	Slice 1	179630
	Slice 2	277310
	Slice 3	374990
IPP Debug nrVlans	Slice 0	81942
	Slice 1	179622
	Slice 2	277302
	Slice 3	374982

Name	Slice	Address Range
IPP Debug routed	Slice 0	81957
	Slice 1	179637
	Slice 2	277317
	Slice 3	374997
IPP Debug routerHit	Slice 0	81947
	Slice 1	179627
	Slice 2	277307
	Slice 3	374987
IPP Debug spVidOp	Slice 0	81944
	Slice 1	179624
	Slice 2	277304
	Slice 3	374984
IPP Debug srcPort	Slice 0	81940
	Slice 1	179620
	Slice 2	277300
	Slice 3	374980
IPP Debug vlanVidOp	Slice 0	81946
	Slice 1	179626
	Slice 2	277306
	Slice 3	374986
IPP Empty Destination Drop	Slice 0	4897
	Slice 1	102577
	Slice 2	200257
	Slice 3	297937
IPP PM Drop	Slice 0	4896
	Slice 1	102576
	Slice 2	200256
	Slice 3	297936
IPP Packet Head Counter	Slice 0	4940
	Slice 1	102620
	Slice 2	200300
	Slice 3	297980
IPP Packet Tail Counter	Slice 0	4941
	Slice 1	102621
	Slice 2	200301
	Slice 3	297981
IPv4 TOS Field To Egress Queue Mapping Table	Slice 0	84007 - 84262
	Slice 1	181687 - 181942
	Slice 2	279367 - 279622
	Slice 3	377047 - 377302
IPv4 TOS Field To Packet Color Mapping Table	Slice 0	83471 - 83726
	Slice 1	181151 - 181406
	Slice 2	278831 - 279086
	Slice 3	376511 - 376766
IPv6 Class of Service Field To Egress Queue Mapping Table	Slice 0	83751 - 84006
	Slice 1	181431 - 181686
	Slice 2	279111 - 279366
	Slice 3	376791 - 377046
IPv6 Class of Service Field To Packet Color Mapping Table	Slice 0	83215 - 83470
	Slice 1	180895 - 181150
	Slice 2	278575 - 278830
	Slice 3	376255 - 376510

Name	Slice	Address Range
Ingress Admission Control Current Status	Slice 0	89650 - 89681
	Slice 1	187330 - 187361
	Slice 2	285010 - 285041
	Slice 3	382690 - 382721
Ingress Admission Control Initial Pointer	Slice 0	5006 - 5261
	Slice 1	102686 - 102941
	Slice 2	200366 - 200621
	Slice 3	298046 - 298301
Ingress Admission Control Mark All Red	Slice 0	89522 - 89553
	Slice 1	187202 - 187233
	Slice 2	284882 - 284913
	Slice 3	382562 - 382593
Ingress Admission Control Mark All Red Enable	Slice 0	89490 - 89521
	Slice 1	187170 - 187201
	Slice 2	284850 - 284881
	Slice 3	382530 - 382561
Ingress Admission Control Reset	Slice 0	89618 - 89649
	Slice 1	187298 - 187329
	Slice 2	284978 - 285009
	Slice 3	382658 - 382689
Ingress Admission Control Token Bucket Configuration	Slice 0	89554 - 89617
	Slice 1	187234 - 187297
	Slice 2	284914 - 284977
	Slice 3	382594 - 382657
Ingress Configurable ACL 0 Large Table	Slice 0	5262 - 21645
	Slice 1	102942 - 119325
	Slice 2	200622 - 217005
	Slice 3	298302 - 314685
Ingress Configurable ACL 0 Pre Lookup	Slice 0	84903 - 84918
	Slice 1	182583 - 182598
	Slice 2	280263 - 280278
	Slice 3	377943 - 377958
Ingress Configurable ACL 0 Rules Setup	Slice 0	84895 - 84902
	Slice 1	182575 - 182582
	Slice 2	280255 - 280262
	Slice 3	377935 - 377942
Ingress Configurable ACL 0 Search Mask	Slice 0	86045
	Slice 1	183725
	Slice 2	281405
	Slice 3	379085
Ingress Configurable ACL 0 Selection	Slice 0	81915
	Slice 1	179595
	Slice 2	277275
	Slice 3	374955
Ingress Configurable ACL 0 Small Table	Slice 0	21646 - 23693
	Slice 1	119326 - 121373
	Slice 2	217006 - 219053
	Slice 3	314686 - 316733
Ingress Configurable ACL 0 TCAM	Slice 0	85773 - 86028
	Slice 1	183453 - 183708
	Slice 2	281133 - 281388
	Slice 3	378813 - 379068

Name	Slice	Address Range
Ingress Configurable ACL 0 TCAM Answer	Slice 0	23694 - 23757
	Slice 1	121374 - 121437
	Slice 2	219054 - 219117
	Slice 3	316734 - 316797
Ingress Configurable ACL 1 Large Table	Slice 0	23758 - 24781
	Slice 1	121438 - 122461
	Slice 2	219118 - 220141
	Slice 3	316798 - 317821
Ingress Configurable ACL 1 Pre Lookup	Slice 0	84383 - 84894
	Slice 1	182063 - 182574
	Slice 2	279743 - 280254
	Slice 3	377423 - 377934
Ingress Configurable ACL 1 Rules Setup	Slice 0	84375 - 84382
	Slice 1	182055 - 182062
	Slice 2	279735 - 279742
	Slice 3	377415 - 377422
Ingress Configurable ACL 1 Search Mask	Slice 0	86309
	Slice 1	183989
	Slice 2	281669
	Slice 3	379349
Ingress Configurable ACL 1 Selection	Slice 0	81916
	Slice 1	179596
	Slice 2	277276
	Slice 3	374956
Ingress Configurable ACL 1 Small Table	Slice 0	24782 - 24845
	Slice 1	122462 - 122525
	Slice 2	220142 - 220205
	Slice 3	317822 - 317885
Ingress Configurable ACL 1 TCAM	Slice 0	86453 - 86516
	Slice 1	184133 - 184196
	Slice 2	281813 - 281876
	Slice 3	379493 - 379556
Ingress Configurable ACL 1 TCAM Answer	Slice 0	24846 - 24877
	Slice 1	122526 - 122557
	Slice 2	220206 - 220237
	Slice 3	317886 - 317917
Ingress Configurable ACL 2 Pre Lookup	Slice 0	84311 - 84374
	Slice 1	181991 - 182054
	Slice 2	279671 - 279734
	Slice 3	377351 - 377414
Ingress Configurable ACL 2 Rules Setup	Slice 0	84307 - 84310
	Slice 1	181987 - 181990
	Slice 2	279667 - 279670
	Slice 3	377347 - 377350
Ingress Configurable ACL 2 TCAM	Slice 0	86517 - 87284
	Slice 1	184197 - 184964
	Slice 2	281877 - 282644
	Slice 3	379557 - 380324
Ingress Configurable ACL 2 TCAM Answer	Slice 0	24878 - 24973
	Slice 1	122558 - 122653
	Slice 2	220238 - 220333
	Slice 3	317918 - 318013

Name	Slice	Address Range
Ingress Configurable ACL 3 Rules Setup	Slice 0	84303 - 84306
	Slice 1	181983 - 181986
	Slice 2	279663 - 279666
	Slice 3	377343 - 377346
Ingress Configurable ACL 3 TCAM	Slice 0	86205 - 86268
	Slice 1	183885 - 183948
	Slice 2	281565 - 281628
	Slice 3	379245 - 379308
Ingress Configurable ACL 3 TCAM Answer	Slice 0	24974 - 24989
	Slice 1	122654 - 122669
	Slice 2	220334 - 220349
	Slice 3	318014 - 318029
Ingress Configurable ACL Drop	Slice 0	4919
	Slice 1	102599
	Slice 2	200279
	Slice 3	297959
Ingress Configurable ACL Match Counter	Slice 0	87990 - 88053
	Slice 1	185670 - 185733
	Slice 2	283350 - 283413
	Slice 3	381030 - 381093
Ingress Drop Options	Slice 0	87797
	Slice 1	185477
	Slice 2	283157
	Slice 3	380837
Ingress Egress Port Packet Type Filter	Slice 0	82331 - 82362
	Slice 1	180011 - 180042
	Slice 2	277691 - 277722
	Slice 3	375371 - 375402
Ingress Ethernet Type for VLAN tag	Slice 0	81911
	Slice 1	179591
	Slice 2	277271
	Slice 3	374951
Ingress MMP Drop Mask	Slice 0	81937
	Slice 1	179617
	Slice 2	277297
	Slice 3	374977
Ingress Multiple Spanning Tree State	Slice 0	41502 - 41517
	Slice 1	139182 - 139197
	Slice 2	236862 - 236877
	Slice 3	334542 - 334557
Ingress NAT Hit Status	Slice 0	99352 - 101399
	Slice 1	197032 - 199079
	Slice 2	294712 - 296759
	Slice 3	392392 - 394439
Ingress NAT Operation	Slice 0	94198 - 96245
	Slice 1	191878 - 193925
	Slice 2	289558 - 291605
	Slice 3	387238 - 389285
Ingress Packet Filtering Drop	Slice 0	4904
	Slice 1	102584
	Slice 2	200264
	Slice 3	297944

Name	Slice	Address Range
Ingress Port Packet Type Filter	Slice 0	84919 - 84950
	Slice 1	182599 - 182630
	Slice 2	280279 - 280310
	Slice 3	377959 - 377990
Ingress Resource Manager Drop	Slice 0	89752
	Slice 1	187432
	Slice 2	285112
	Slice 3	382792
Ingress Router Table	Slice 0	41518 - 41521
	Slice 1	139198 - 139201
	Slice 2	236878 - 236881
	Slice 3	334558 - 334561
Ingress Spanning Tree Drop: Blocking	Slice 0	4902
	Slice 1	102582
	Slice 2	200262
	Slice 3	297942
Ingress Spanning Tree Drop: Learning	Slice 0	4901
	Slice 1	102581
	Slice 2	200261
	Slice 3	297941
Ingress Spanning Tree Drop: Listen	Slice 0	4900
	Slice 1	102580
	Slice 2	200260
	Slice 3	297940
Ingress VID Ethernet Type Range Assignment Answer	Slice 0	84275 - 84278
	Slice 1	181955 - 181958
	Slice 2	279635 - 279638
	Slice 3	377315 - 377318
Ingress VID Ethernet Type Range Search Data	Slice 0	84279 - 84282
	Slice 1	181959 - 181962
	Slice 2	279639 - 279642
	Slice 3	377319 - 377322
Ingress VID Inner VID Range Assignment Answer	Slice 0	84283 - 84286
	Slice 1	181963 - 181966
	Slice 2	279643 - 279646
	Slice 3	377323 - 377326
Ingress VID Inner VID Range Search Data	Slice 0	84287 - 84290
	Slice 1	181967 - 181970
	Slice 2	279647 - 279650
	Slice 3	377327 - 377330
Ingress VID MAC Range Assignment Answer	Slice 0	84299 - 84302
	Slice 1	181979 - 181982
	Slice 2	279659 - 279662
	Slice 3	377339 - 377342
Ingress VID MAC Range Search Data	Slice 0	86189 - 86204
	Slice 1	183869 - 183884
	Slice 2	281549 - 281564
	Slice 3	379229 - 379244
Ingress VID Outer VID Range Assignment Answer	Slice 0	84291 - 84294
	Slice 1	181971 - 181974
	Slice 2	279651 - 279654
	Slice 3	377331 - 377334

Name	Slice	Address Range
Ingress VID Outer VID Range Search Data	Slice 0	84295 - 84298
	Slice 1	181975 - 181978
	Slice 2	279655 - 279658
	Slice 3	377335 - 377338
Ingress-Egress Packet Filtering Drop	Slice 0	89242 - 89273
	Slice 1	186922 - 186953
	Slice 2	284602 - 284633
	Slice 3	382282 - 382313
Invalid Routing Protocol Drop	Slice 0	4910
	Slice 1	102590
	Slice 2	200270
	Slice 3	297950
L2 Action Table	Slice 0	76418 - 76545
	Slice 1	174098 - 174225
	Slice 2	271778 - 271905
	Slice 3	369458 - 369585
L2 Action Table Drop	Slice 0	4937
	Slice 1	102617
	Slice 2	200297
	Slice 3	297977
L2 Action Table Egress Port State	Slice 0	81932
	Slice 1	179612
	Slice 2	277292
	Slice 3	374972
L2 Action Table Per Port Drop	Slice 0	89274 - 89305
	Slice 1	186954 - 186985
	Slice 2	284634 - 284665
	Slice 3	382314 - 382345
L2 Action Table Port Move Drop	Slice 0	4938
	Slice 1	102618
	Slice 2	200298
	Slice 3	297978
L2 Action Table Source Port	Slice 0	76546 - 76673
	Slice 1	174226 - 174353
	Slice 2	271906 - 272033
	Slice 3	369586 - 369713
L2 Action Table Special Packet Type Drop	Slice 0	4936
	Slice 1	102616
	Slice 2	200296
	Slice 3	297976
L2 Aging Collision Shadow Table	Slice 0	83167 - 83198
	Slice 1	180847 - 180878
	Slice 2	278527 - 278558
	Slice 3	376207 - 376238
L2 Aging Collision Table		638 - 669
L2 Aging Status Shadow Table	Slice 0	51778 - 55873
	Slice 1	149458 - 153553
	Slice 2	247138 - 251233
	Slice 3	344818 - 348913
L2 Aging Status Shadow Table - Replica	Slice 0	68194 - 72289
	Slice 1	165874 - 169969
	Slice 2	263554 - 267649
	Slice 3	361234 - 365329
L2 Aging Table		702 - 4797
L2 Broadcast Storm Control Bucket Capacity Configuration		336 - 367



Name	Slice	Address Range
L2 Broadcast Storm Control Bucket Threshold Configuration		368 - 399
L2 Broadcast Storm Control Enable		400
L2 Broadcast Storm Control Rate Configuration		304 - 335
L2 DA Hash Lookup Table	Slice 0	55874 - 59969
	Slice 1	153554 - 157649
	Slice 2	251234 - 255329
	Slice 3	348914 - 353009
L2 Destination Table	Slice 0	59970 - 64097
	Slice 1	157650 - 161777
	Slice 2	255330 - 259457
	Slice 3	353010 - 357137
L2 Destination Table - Replica	Slice 0	72290 - 76417
	Slice 1	169970 - 174097
	Slice 2	267650 - 271777
	Slice 3	365330 - 369457
L2 Flooding Storm Control Bucket Capacity Configuration		530 - 561
L2 Flooding Storm Control Bucket Threshold Configuration		562 - 593
L2 Flooding Storm Control Enable		594
L2 Flooding Storm Control Rate Configuration		498 - 529
L2 IEEE 1588 Decoder Drop	Slice 0	4923
	Slice 1	102603
	Slice 2	200283
	Slice 3	297963
L2 Lookup Collision Table	Slice 0	83135 - 83166
	Slice 1	180815 - 180846
	Slice 2	278495 - 278526
	Slice 3	376175 - 376206
L2 Lookup Collision Table Masks	Slice 0	83131 - 83134
	Slice 1	180811 - 180814
	Slice 2	278491 - 278494
	Slice 3	376171 - 376174
L2 Lookup Drop	Slice 0	4903
	Slice 1	102583
	Slice 2	200263
	Slice 3	297943
L2 Multicast Handling	Slice 0	81933
	Slice 1	179613
	Slice 2	277293
	Slice 3	374973
L2 Multicast Storm Control Bucket Capacity Configuration		433 - 464
L2 Multicast Storm Control Bucket Threshold Configuration		465 - 496
L2 Multicast Storm Control Enable		497
L2 Multicast Storm Control Rate Configuration		401 - 432
L2 Multicast Table	Slice 0	83067 - 83130
	Slice 1	180747 - 180810
	Slice 2	278427 - 278490
	Slice 3	376107 - 376170
L2 QoS Mapping Table	Slice 0	97782 - 97845
	Slice 1	195462 - 195525
	Slice 2	293142 - 293205
	Slice 3	390822 - 390885
L2 Reserved Multicast Address Action	Slice 0	84987 - 85242
	Slice 1	182667 - 182922
	Slice 2	280347 - 280602
	Slice 3	378027 - 378282



Name	Slice	Address Range
L2 Reserved Multicast Address Base	Slice 0	81912
	Slice 1	179592
	Slice 2	277272
	Slice 3	374952
L2 Reserved Multicast Address Drop	Slice 0	4918
	Slice 1	102598
	Slice 2	200278
	Slice 3	297958
L2 SA Hash Lookup Table	Slice 0	64098 - 68193
	Slice 1	161778 - 165873
	Slice 2	259458 - 263553
	Slice 3	357138 - 361233
L2 Tunnel Decoder Setup	Slice 0	81908
	Slice 1	179588
	Slice 2	277268
	Slice 3	374948
L2 Tunnel Entry Instruction Table	Slice 0	91838 - 91853
	Slice 1	189518 - 189533
	Slice 2	287198 - 287213
	Slice 3	384878 - 384893
L3 LPM Result	Slice 0	41522 - 41537
	Slice 1	139202 - 139217
	Slice 2	236882 - 236897
	Slice 3	334562 - 334577
L3 Lookup Drop	Slice 0	4912
	Slice 1	102592
	Slice 2	200272
	Slice 3	297952
L3 Routing Default	Slice 0	84271 - 84274
	Slice 1	181951 - 181954
	Slice 2	279631 - 279634
	Slice 3	377311 - 377314
L3 Routing TCAM	Slice 0	87798 - 87925
	Slice 1	185478 - 185605
	Slice 2	283158 - 283285
	Slice 3	380838 - 380965
L3 Tunnel Entry Instruction Table	Slice 0	91854 - 91869
	Slice 1	189534 - 189549
	Slice 2	287214 - 287229
	Slice 3	384894 - 384909
L4 IEEE 1588 Decoder Drop	Slice 0	4924
	Slice 1	102604
	Slice 2	200284
	Slice 3	297964
LACP Decoder Drop	Slice 0	4927
	Slice 1	102607
	Slice 2	200287
	Slice 3	297967
LACP Packet Decoder Options	Slice 0	85499
	Slice 1	183179
	Slice 2	280859
	Slice 3	378539

Name	Slice	Address Range
LLDP Configuration	Slice 0	86305
	Slice 1	183985
	Slice 2	281665
	Slice 3	379345
Learning And Aging Enable		599
Learning And Aging Writeback Control		601
Learning Conflict		595
Learning DA MAC	Slice 0	81909
	Slice 1	179589
	Slice 2	277269
	Slice 3	374949
Learning Data FIFO		4799
Learning Data FIFO High Watermark Level		602
Learning Overflow		597
Learning Packet Drop	Slice 0	4917
	Slice 1	102597
	Slice 2	200277
	Slice 3	297957
Link Aggregate Weight	Slice 0	82059 - 82314
	Slice 1	179739 - 179994
	Slice 2	277419 - 277674
	Slice 3	375099 - 375354
Link Aggregation Ctrl	Slice 0	81906
	Slice 1	179586
	Slice 2	277266
	Slice 3	374946
Link Aggregation Membership	Slice 0	85263 - 85294
	Slice 1	182943 - 182974
	Slice 2	280623 - 280654
	Slice 3	378303 - 378334
Link Aggregation To Physical Ports Members	Slice 0	82027 - 82058
	Slice 1	179707 - 179738
	Slice 2	277387 - 277418
	Slice 3	375067 - 375098
MAC Interface Counters For RX		48 - 79
MAC Interface Counters For TX		208 - 271
MAC RX Broken Packets		112 - 143
MAC RX Long Packet Drop		176 - 207
MAC RX Maximum Packet Length		80 - 111
MAC RX Short Packet Drop		144 - 175
MBSC Drop	Slice 0	89210 - 89241
	Slice 1	186890 - 186921
	Slice 2	284570 - 284601
	Slice 3	382250 - 382281
MPLS EXP Field To Egress Queue Mapping Table	Slice 0	83743 - 83750
	Slice 1	181423 - 181430
	Slice 2	279103 - 279110
	Slice 3	376783 - 376790
MPLS EXP Field To Packet Color Mapping Table	Slice 0	83207 - 83214
	Slice 1	180887 - 180894
	Slice 2	278567 - 278574
	Slice 3	376247 - 376254



Name	Slice	Address Range
MPLS QoS Mapping Table	Slice 0	98614 - 99125
	Slice 1	196294 - 196805
	Slice 2	293974 - 294485
	Slice 3	391654 - 392165
Map Queue to Priority	Slice 0	89753 - 89760
	Slice 1	187433 - 187440
	Slice 2	285113 - 285120
	Slice 3	382793 - 382800
Maximum Allowed VLAN Drop	Slice 0	4909
	Slice 1	102589
	Slice 2	200269
	Slice 3	297949
Minimum Allowed VLAN Drop	Slice 0	4908
	Slice 1	102588
	Slice 2	200268
	Slice 3	297948
Minimum Buffer Free	Slice 0	90919
	Slice 1	188599
	Slice 2	286279
	Slice 3	383959
NAT Action Table	Slice 0	82363 - 82874
	Slice 1	180043 - 180554
	Slice 2	277723 - 278234
	Slice 3	375403 - 375914
NAT Action Table Drop	Slice 0	4935
	Slice 1	102615
	Slice 2	200295
	Slice 3	297975
NAT Action Table Force Original Packet	Slice 0	81936
	Slice 1	179616
	Slice 2	277296
	Slice 3	374976
NAT Add Egress Port for NAT Calculation	Slice 0	99298
	Slice 1	196978
	Slice 2	294658
	Slice 3	392338
Next Hop DA MAC	Slice 0	91938 - 92961
	Slice 1	189618 - 190641
	Slice 2	287298 - 288321
	Slice 3	384978 - 386001
Next Hop Hit Status	Slice 0	88058 - 89081
	Slice 1	185738 - 186761
	Slice 2	283418 - 284441
	Slice 3	381098 - 382121
Next Hop MPLS Table	Slice 0	92970 - 93993
	Slice 1	190650 - 191673
	Slice 2	288330 - 289353
	Slice 3	386010 - 387033
Next Hop Packet Insert MPLS Header	Slice 0	93998 - 94061
	Slice 1	191678 - 191741
	Slice 2	289358 - 289421
	Slice 3	387038 - 387101

Name	Slice	Address Range
Next Hop Packet Modifications	Slice 0	50754 - 51777
	Slice 1	148434 - 149457
	Slice 2	246114 - 247137
	Slice 3	343794 - 344817
Next Hop Table	Slice 0	49730 - 50753
	Slice 1	147410 - 148433
	Slice 2	245090 - 246113
	Slice 3	342770 - 343793
Output Disable	Slice 0	90920 - 90927
	Slice 1	188600 - 188607
	Slice 2	286280 - 286287
	Slice 3	383960 - 383967
Output Mirroring Table	Slice 0	99126 - 99157
	Slice 1	196806 - 196837
	Slice 2	294486 - 294517
	Slice 3	392166 - 392197
PB Packet Head Counter	Slice 0	91456
	Slice 1	189136
	Slice 2	286816
	Slice 3	384496
PB Packet Tail Counter	Slice 0	91457
	Slice 1	189137
	Slice 2	286817
	Slice 3	384497
PFC Dec Counters for ingress ports 0 to 7	Slice 0	90090 - 90153
	Slice 1	187770 - 187833
	Slice 2	285450 - 285513
	Slice 3	383130 - 383193
PFC Dec Counters for ingress ports 16 to 23	Slice 0	90218 - 90281
	Slice 1	187898 - 187961
	Slice 2	285578 - 285641
	Slice 3	383258 - 383321
PFC Dec Counters for ingress ports 24 to 31	Slice 0	90282 - 90345
	Slice 1	187962 - 188025
	Slice 2	285642 - 285705
	Slice 3	383322 - 383385
PFC Dec Counters for ingress ports 8 to 15	Slice 0	90154 - 90217
	Slice 1	187834 - 187897
	Slice 2	285514 - 285577
	Slice 3	383194 - 383257
PFC Inc Counters for ingress ports 0 to 7	Slice 0	89834 - 89897
	Slice 1	187514 - 187577
	Slice 2	285194 - 285257
	Slice 3	382874 - 382937
PFC Inc Counters for ingress ports 16 to 23	Slice 0	89962 - 90025
	Slice 1	187642 - 187705
	Slice 2	285322 - 285385
	Slice 3	383002 - 383065
PFC Inc Counters for ingress ports 24 to 31	Slice 0	90026 - 90089
	Slice 1	187706 - 187769
	Slice 2	285386 - 285449
	Slice 3	383066 - 383129



Name	Slice	Address Range
PFC Inc Counters for ingress ports 8 to 15	Slice 0	89898 - 89961
	Slice 1	187578 - 187641
	Slice 2	285258 - 285321
	Slice 3	382938 - 383001
PS Error Counter	Slice 0	102480 - 102487
	Slice 1	200160 - 200167
	Slice 2	297840 - 297847
	Slice 3	395520 - 395527
PS Packet Head Counter	Slice 0	102488
	Slice 1	200168
	Slice 2	297848
	Slice 3	395528
PS Packet Tail Counter	Slice 0	102489
	Slice 1	200169
	Slice 2	297849
	Slice 3	395529
Packet Buffer Status	Slice 0	89750
	Slice 1	187430
	Slice 2	285110
	Slice 3	382790
Port FFA Used	Slice 0	90346 - 90353
	Slice 1	188026 - 188033
	Slice 2	285706 - 285713
	Slice 3	383386 - 383393
Port Move Options	Slice 0	81931
	Slice 1	179611
	Slice 2	277291
	Slice 3	374971
Port Pause Settings	Slice 0	90372 - 90379
	Slice 1	188052 - 188059
	Slice 2	285732 - 285739
	Slice 3	383412 - 383419
Port Reserved	Slice 0	89826 - 89833
	Slice 1	187506 - 187513
	Slice 2	285186 - 285193
	Slice 3	382866 - 382873
Port Shaper Bucket Capacity Configuration	Slice 0	91402 - 91409
	Slice 1	189082 - 189089
	Slice 2	286762 - 286769
	Slice 3	384442 - 384449
Port Shaper Bucket Threshold Configuration	Slice 0	91410 - 91417
	Slice 1	189090 - 189097
	Slice 2	286770 - 286777
	Slice 3	384450 - 384457
Port Shaper Enable	Slice 0	91418
	Slice 1	189098
	Slice 2	286778
	Slice 3	384458
Port Shaper Rate Configuration	Slice 0	91394 - 91401
	Slice 1	189074 - 189081
	Slice 2	286754 - 286761
	Slice 3	384434 - 384441

Name	Slice	Address Range
Port Tail-Drop FFA Threshold	Slice 0	90431 - 90438
	Slice 1	188111 - 188118
	Slice 2	285791 - 285798
	Slice 3	383471 - 383478
Port Tail-Drop Settings	Slice 0	90380 - 90387
	Slice 1	188060 - 188067
	Slice 2	285740 - 285747
	Slice 3	383420 - 383427
Port Used	Slice 0	90354 - 90361
	Slice 1	188034 - 188041
	Slice 2	285714 - 285721
	Slice 3	383394 - 383401
Port Xoff FFA Threshold	Slice 0	90423 - 90430
	Slice 1	188103 - 188110
	Slice 2	285783 - 285790
	Slice 3	383463 - 383470
Port Xon FFA Threshold	Slice 0	90415 - 90422
	Slice 1	188095 - 188102
	Slice 2	285775 - 285782
	Slice 3	383455 - 383462
Port/TC Reserved	Slice 0	89762 - 89825
	Slice 1	187442 - 187505
	Slice 2	285122 - 285185
	Slice 3	382802 - 382865
Port/TC Tail-Drop Total Threshold	Slice 0	90567 - 90630
	Slice 1	188247 - 188310
	Slice 2	285927 - 285990
	Slice 3	383607 - 383670
Port/TC Xoff Total Threshold	Slice 0	90503 - 90566
	Slice 1	188183 - 188246
	Slice 2	285863 - 285926
	Slice 3	383543 - 383606
Port/TC Xon Total Threshold	Slice 0	90439 - 90502
	Slice 1	188119 - 188182
	Slice 2	285799 - 285862
	Slice 3	383479 - 383542
Prio Shaper Bucket Capacity Configuration	Slice 0	91265 - 91328
	Slice 1	188945 - 189008
	Slice 2	286625 - 286688
	Slice 3	384305 - 384368
Prio Shaper Bucket Threshold Configuration	Slice 0	91329 - 91392
	Slice 1	189009 - 189072
	Slice 2	286689 - 286752
	Slice 3	384369 - 384432
Prio Shaper Enable	Slice 0	91393
	Slice 1	189073
	Slice 2	286753
	Slice 3	384433
Prio Shaper Rate Configuration	Slice 0	91201 - 91264
	Slice 1	188881 - 188944
	Slice 2	286561 - 286624
	Slice 3	384241 - 384304

Name	Slice	Address Range
Queue Off Drop	Slice 0	89146 - 89177
	Slice 1	186826 - 186857
	Slice 2	284506 - 284537
	Slice 3	382186 - 382217
Queue Shaper Bucket Capacity Configuration	Slice 0	91072 - 91135
	Slice 1	188752 - 188815
	Slice 2	286432 - 286495
	Slice 3	384112 - 384175
Queue Shaper Bucket Threshold Configuration	Slice 0	91136 - 91199
	Slice 1	188816 - 188879
	Slice 2	286496 - 286559
	Slice 3	384176 - 384239
Queue Shaper Enable	Slice 0	91200
	Slice 1	188880
	Slice 2	286560
	Slice 3	384240
Queue Shaper Rate Configuration	Slice 0	91008 - 91071
	Slice 1	188688 - 188751
	Slice 2	286368 - 286431
	Slice 3	384048 - 384111
RARP Decoder Drop	Slice 0	4922
	Slice 1	102602
	Slice 2	200282
	Slice 3	297962
RARP Packet Decoder Options	Slice 0	85489
	Slice 1	183169
	Slice 2	280849
	Slice 3	378529
Re-queue Overflow Drop	Slice 0	89761
	Slice 1	187441
	Slice 2	285121
	Slice 3	382801
Received Packets on Ingress VRF	Slice 0	88054 - 88057
	Slice 1	185734 - 185737
	Slice 2	283414 - 283417
	Slice 3	381094 - 381097
Reserved Destination MAC Address Range	Slice 0	86285 - 86300
	Slice 1	183965 - 183980
	Slice 2	281645 - 281660
	Slice 3	379325 - 379340
Reserved MAC DA Drop	Slice 0	4905
	Slice 1	102585
	Slice 2	200265
	Slice 3	297945
Reserved MAC SA Drop	Slice 0	4906
	Slice 1	102586
	Slice 2	200266
	Slice 3	297946
Reserved Source MAC Address Range	Slice 0	86269 - 86284
	Slice 1	183949 - 183964
	Slice 2	281629 - 281644
	Slice 3	379309 - 379324

Name	Slice	Address Range
Resource Limiter Set	Slice 0	89738 - 89741
	Slice 1	187418 - 187421
	Slice 2	285098 - 285101
	Slice 3	382778 - 382781
Router Egress Queue To VLAN Data	Slice 0	83199 - 83206
	Slice 1	180879 - 180886
	Slice 2	278559 - 278566
	Slice 3	376239 - 376246
Router MTU Table	Slice 0	82907 - 83034
	Slice 1	180587 - 180714
	Slice 2	278267 - 278394
	Slice 3	375947 - 376074
Router Port Egress SA MAC Address	Slice 0	92962 - 92969
	Slice 1	190642 - 190649
	Slice 2	288322 - 288329
	Slice 3	386002 - 386009
Router Port MAC Address	Slice 0	86125 - 86188
	Slice 1	183805 - 183868
	Slice 2	281485 - 281548
	Slice 3	379165 - 379228
SCTP Decoder Drop	Slice 0	4926
	Slice 1	102606
	Slice 2	200286
	Slice 3	297966
SCTP Packet Decoder Options	Slice 0	85497
	Slice 1	183177
	Slice 2	280857
	Slice 3	378537
SMON Set 0 Byte Counter	Slice 0	87958 - 87965
	Slice 1	185638 - 185645
	Slice 2	283318 - 283325
	Slice 3	380998 - 381005
SMON Set 0 Packet Counter	Slice 0	87926 - 87933
	Slice 1	185606 - 185613
	Slice 2	283286 - 283293
	Slice 3	380966 - 380973
SMON Set 1 Byte Counter	Slice 0	87966 - 87973
	Slice 1	185646 - 185653
	Slice 2	283326 - 283333
	Slice 3	381006 - 381013
SMON Set 1 Packet Counter	Slice 0	87934 - 87941
	Slice 1	185614 - 185621
	Slice 2	283294 - 283301
	Slice 3	380974 - 380981
SMON Set 2 Byte Counter	Slice 0	87974 - 87981
	Slice 1	185654 - 185661
	Slice 2	283334 - 283341
	Slice 3	381014 - 381021
SMON Set 2 Packet Counter	Slice 0	87942 - 87949
	Slice 1	185622 - 185629
	Slice 2	283302 - 283309
	Slice 3	380982 - 380989

Name	Slice	Address Range
SMON Set 3 Byte Counter	Slice 0	87982 - 87989
	Slice 1	185662 - 185669
	Slice 2	283342 - 283349
	Slice 3	381022 - 381029
SMON Set 3 Packet Counter	Slice 0	87950 - 87957
	Slice 1	185630 - 185637
	Slice 2	283310 - 283317
	Slice 3	380990 - 380997
SMON Set Search	Slice 0	84951 - 84954
	Slice 1	182631 - 182634
	Slice 2	280311 - 280314
	Slice 3	377991 - 377994
SNAP LLC Decoding Options	Slice 0	81910
	Slice 1	179590
	Slice 2	277270
	Slice 3	374950
SP Overflow Drop	Slice 0	4848 - 4855
	Slice 1	102528 - 102535
	Slice 2	200208 - 200215
	Slice 3	297888 - 297895
Scratch		2
Second Tunnel Exit Drop	Slice 0	4914
	Slice 1	102594
	Slice 2	200274
	Slice 3	297954
Second Tunnel Exit Lookup TCAM	Slice 0	86061 - 86124
	Slice 1	183741 - 183804
	Slice 2	281421 - 281484
	Slice 3	379101 - 379164
Second Tunnel Exit Lookup TCAM Answer	Slice 0	85247 - 85262
	Slice 1	182927 - 182942
	Slice 2	280607 - 280622
	Slice 3	378287 - 378302
Second Tunnel Exit Miss Action	Slice 0	85243 - 85246
	Slice 1	182923 - 182926
	Slice 2	280603 - 280606
	Slice 3	378283 - 378286
Select Which Egress QoS Mapping Table To Use	Slice 0	97270 - 97781
	Slice 1	194950 - 195461
	Slice 2	292630 - 293141
	Slice 3	390310 - 390821
Send to CPU	Slice 0	86301
	Slice 1	183981
	Slice 2	281661
	Slice 3	379341
Software Aging Enable		600
Software Aging Start Latch		4798
Source Port Default ACL Action	Slice 0	24990 - 25117
	Slice 1	122670 - 122797
	Slice 2	220350 - 220477
	Slice 3	318030 - 318157
Source Port Default ACL Action Drop	Slice 0	4939
	Slice 1	102619
	Slice 2	200299
	Slice 3	297979



Name	Slice	Address Range
Source Port Table	Slice 0	85295 - 85358
	Slice 1	182975 - 183038
	Slice 2	280655 - 280718
	Slice 3	378335 - 378398
TC FFA Used	Slice 0	90362 - 90369
	Slice 1	188042 - 188049
	Slice 2	285722 - 285729
	Slice 3	383402 - 383409
TC Tail-Drop FFA Threshold	Slice 0	90407 - 90414
	Slice 1	188087 - 188094
	Slice 2	285767 - 285774
	Slice 3	383447 - 383454
TC Xoff FFA Threshold	Slice 0	90399 - 90406
	Slice 1	188079 - 188086
	Slice 2	285759 - 285766
	Slice 3	383439 - 383446
TC Xon FFA Threshold	Slice 0	90391 - 90398
	Slice 1	188071 - 188078
	Slice 2	285751 - 285758
	Slice 3	383431 - 383438
TOS QoS Mapping Table	Slice 0	98102 - 98613
	Slice 1	195782 - 196293
	Slice 2	293462 - 293973
	Slice 3	391142 - 391653
Tail-Drop FFA Threshold	Slice 0	90390
	Slice 1	188070
	Slice 2	285750
	Slice 3	383430
Time to Age		637
Transmitted Packets on Egress VRF	Slice 0	99348 - 99351
	Slice 1	197028 - 197031
	Slice 2	294708 - 294711
	Slice 3	392388 - 392391
Tunnel Entry Header Data	Slice 0	91566 - 91821
	Slice 1	189246 - 189501
	Slice 2	286926 - 287181
	Slice 3	384606 - 384861
Tunnel Entry Instruction Table	Slice 0	91550 - 91565
	Slice 1	189230 - 189245
	Slice 2	286910 - 286925
	Slice 3	384590 - 384605
Tunnel Entry MTU Length Check	Slice 0	81890 - 81905
	Slice 1	179570 - 179585
	Slice 2	277250 - 277265
	Slice 3	374930 - 374945
Tunnel Exit Lookup TCAM	Slice 0	85517 - 85772
	Slice 1	183197 - 183452
	Slice 2	280877 - 281132
	Slice 3	378557 - 378812
Tunnel Exit Lookup TCAM Answer	Slice 0	4942 - 5005
	Slice 1	102622 - 102685
	Slice 2	200302 - 200365
	Slice 3	297982 - 298045

Name	Slice	Address Range
Tunnel Exit Miss Action Drop	Slice 0	4915
	Slice 1	102595
	Slice 2	200275
	Slice 3	297955
Tunnel Exit Too Small Packet Modification Drop	Slice 0	4916
	Slice 1	102596
	Slice 2	200276
	Slice 3	297956
Tunnel Exit Too Small Packet Modification To Small Drop	Slice 0	91491 - 91498
	Slice 1	189171 - 189178
	Slice 2	286851 - 286858
	Slice 3	384531 - 384538
Unknown Egress Drop	Slice 0	91467 - 91474
	Slice 1	189147 - 189154
	Slice 2	286827 - 286834
	Slice 3	384507 - 384514
Unknown Ingress Drop	Slice 0	4898
	Slice 1	102578
	Slice 2	200258
	Slice 3	297938
VLAN Member Drop	Slice 0	4907
	Slice 1	102587
	Slice 2	200267
	Slice 3	297947
VLAN PCP And DEI To Color Mapping Table	Slice 0	83727 - 83742
	Slice 1	181407 - 181422
	Slice 2	279087 - 279102
	Slice 3	376767 - 376782
VLAN PCP To Queue Mapping Table	Slice 0	84263 - 84270
	Slice 1	181943 - 181950
	Slice 2	279623 - 279630
	Slice 3	377303 - 377310
VLAN Table	Slice 0	25118 - 41501
	Slice 1	122798 - 139181
	Slice 2	220478 - 236861
	Slice 3	318158 - 334541
Xoff FFA Threshold	Slice 0	90389
	Slice 1	188069
	Slice 2	285749
	Slice 3	383429
Xon FFA Threshold	Slice 0	90388
	Slice 1	188068
	Slice 2	285748
	Slice 3	383428

35.5 Active Queue Manager

35.5.1 ERM Red Configuration

Configurations to mark the buffer memory congestion status as Red (heavily congested).



Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	89729
Slice 1 :	187409
Slice 2 :	285089
Slice 3 :	382769

Field Description

Bits	Field Name	Description	Default Value
12:0	redXoff	Number of free cells below this value will invoke the red congestion check for the incoming cells. The checks include the number of enqueued cells in the current queue and the packet length. The incoming packet might be terminated and dropped based on the check result.	0x19a
25:13	redXon	Once the red congestion check is applied, number of free cells need to go above this value to disable the check again. The value needs to be larger than redXoff to provide an effective hysteresis.	0x401
33:26	redMaxCells	Maximum allowed packet length in cells when the buffer memory congestion status is red.	0x7

35.5.2 ERM Yellow Configuration

Configurations to mark the buffer memory congestion status as Yellow (slightly congested).

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	89728
Slice 1 :	187408
Slice 2 :	285088
Slice 3 :	382768

Field Description

Bits	Field Name	Description	Default Value
12:0	yellowXoff	Number of free cells below this value will invoke yellow congestion checks for the incoming cells. The checks include the number of enqueued cells in the current queue, higher priority queues and optionally the total number of enqueued cells for the current egress port. Incoming packets might be terminated and dropped based on the check result.	0x559
25:13	yellowXon	Once the yellow congestion check is applied, number of free cells need to go above this value to disable the check again. The value needs to be larger than yellowXoff to provide an effective hysteresis.	0x790



Bits	Field Name	Description	Default Value
33:26	redPortEn	When the buffer memory congestion status is yellow and a single port consumes more than redPortXoff cells, this field can apply the redLimit check on a per port basis.	0xff
46:34	redPortXoff	When the buffer memory congestion status is yellow and the total number of cells enqueued on an egress port is larger than this value, redLimit check for that port will be invoked. Only valid when redPortEn is turned on.	0x401

35.5.3 Egress Resource Manager Pointer

This table provides each egress port a set of limiters. Different egress queues can have different pointers to the **Resource Limiter Set**.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space

Slice 0 :	89730 to 89737
Slice 1 :	187410 to 187417
Slice 2 :	285090 to 285097
Slice 3 :	382770 to 382777

Field Description

Bits	Field Name	Description	Default Value
1:0	q0	Pointer to the Resource Limiter Set for egress queue 0.	0x0
3:2	q1	Pointer to the Resource Limiter Set for egress queue 1.	0x0
5:4	q2	Pointer to the Resource Limiter Set for egress queue 2.	0x0
7:6	q3	Pointer to the Resource Limiter Set for egress queue 3.	0x0
9:8	q4	Pointer to the Resource Limiter Set for egress queue 4.	0x0
11:10	q5	Pointer to the Resource Limiter Set for egress queue 5.	0x0
13:12	q6	Pointer to the Resource Limiter Set for egress queue 6.	0x0
15:14	q7	Pointer to the Resource Limiter Set for egress queue 7.	0x0

35.5.4 Resource Limiter Set

This resource limiter is for comparing how many cells are ahead of the incoming cell for scheduling, that includes cells are enqueued in the same egress queue and all cells with a higher scheduling priority.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : Pointer from the **Egress Resource Manager Pointer**

Address Space

Slice 0 :	89738 to 89741
Slice 1 :	187418 to 187421
Slice 2 :	285098 to 285101
Slice 3 :	382778 to 382781



Field Description

Bits	Field Name	Description	Default Value
12:0	yellowAccumulated	When the buffer memory is slightly congested (yellow), the ERM allows accumulation of cells with the same queue or higher scheduling priorities to the limit in this field before applying the yellowLimit .	0xab
25:13	yellowLimit	When the buffer memory is slightly congested (yellow) and yellowAccumulated is reached, the packet will be terminated and dropped if the enqueued cells in the corresponding queue is more than this value.	0x1f
38:26	redLimit	When the buffer memory is heavily congested (red), the incoming packet will be terminated and dropped if the enqueued cells in the corresponding egress queue is more than this value.	0x10
46:39	maxCells	Maximum allowed packet length in cells for this limiter. Packet with cells more than this value will be dropped.	0xff

35.6 Core Information**35.6.1 Core Version**

Address 0 is reserved for the core version. Make sure the register value is the same as the revision number in the front page of the datasheet.

Number of Entries : 1
 Type of Operation : Read Only
 Address Space : 0

Field Description

Bits	Field Name	Description	Default Value
31:0	version	Version of the core.	0xcda53817

35.7 Egress Packet Processing**35.7.1 Beginning of Packet Tunnel Entry Instruction Table**

This is the L2 tunnel entry instruction which describes how a tunnel entry should be done after the L3 header. If the L3Type is either IPv4, IPv6 then the length fields are updated in the IP headers, for IPv4 the checksum is re-calculated. If the hasUDP is turned on then the UDP length-field is updated.



Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : Tunnel entry pointer

Address Space

Slice 0 :	91822 to 91837
Slice 1 :	189502 to 189517
Slice 2 :	287182 to 287197
Slice 3 :	384862 to 384877

Field Description

Bits	Field Name	Description	Default Value
1:0	l3Type	Inserted header type, when selecting MPLS/Other no updates will be done to the data. 0 = IPv4 1 = IPv6 2 = MPLS/Other. 3 = Reserved.	0x0
7:2	ipHeaderOffset	Where does the IPv4/IPv6 header start in this header. Only valid if the L3-Header type is IPv4 or IPv6.	0x0
8	hasUdp	If the header is a IPv4 or IPv6 then a an UDP header is after the IP header. 0 = No. 1 = Yes.	0x0

35.7.2 Color Remap From Egress Port

Options for remapping internal packet color to outgoing packet headers. Each egress port has a separate color to field mapping.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Port

Address Space

Slice 0 :	91870 to 91901
Slice 1 :	189550 to 189581
Slice 2 :	287230 to 287261
Slice 3 :	384910 to 384941

Field Description

Bits	Field Name	Description	Default Value						
1:0	colorMode	0 = Skip remap 1 = Remap to L3 only 2 = Remap to L2 only 3 = Remap to L2 and L3	0x1						
25:2	color2Tos	New TOS/TC value based on packet color. <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">bits [0:7] :</td> <td>TOS/TC value for green</td> </tr> <tr> <td>bits [8:15] :</td> <td>TOS/TC value for yellow</td> </tr> <tr> <td>bits [16:23] :</td> <td>TOS/TC value for red</td> </tr> </table>	bits [0:7] :	TOS/TC value for green	bits [8:15] :	TOS/TC value for yellow	bits [16:23] :	TOS/TC value for red	0x0
bits [0:7] :	TOS/TC value for green								
bits [8:15] :	TOS/TC value for yellow								
bits [16:23] :	TOS/TC value for red								
33:26	tosMask	Mask for updating the TOS/TC field. For each bit in the mask, 0 means keep original value, 1 means update new value to that bit.	0x0						



Bits	Field Name	Description	Default Value						
36:34	color2Dei	New DEI value based on packet color. This is located in the outermost VLAN of the transmitted packet. <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">bit 0 :</td> <td>DEI value for green</td> </tr> <tr> <td>bit 1 :</td> <td>DEI value for yellow</td> </tr> <tr> <td>bit 2 :</td> <td>DEI value for red</td> </tr> </table>	bit 0 :	DEI value for green	bit 1 :	DEI value for yellow	bit 2 :	DEI value for red	0x0
bit 0 :	DEI value for green								
bit 1 :	DEI value for yellow								
bit 2 :	DEI value for red								

35.7.3 Color Remap From Ingress Admission Control

Options from ingress admission control to remap internal packet color to outgoing packet headers.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Meter Pointer

Address Space

Slice 0 :	91902 to 91933
Slice 1 :	189582 to 189613
Slice 2 :	287262 to 287293
Slice 3 :	384942 to 384973

Field Description

Bits	Field Name	Description	Default Value						
0	enable	If set, the colorMode field determines the remap process. Otherwise color remapping based on the ingress admission control is skipped.	0x0						
2:1	colorMode	0 = Remap disabled 1 = Remap to L3 only 2 = Remap to L2 only 3 = Remap to L2 and L3	0x0						
26:3	color2Tos	New TOS/TC value based on packet color. <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">bits [0:7] :</td> <td>TOS/TC value for green</td> </tr> <tr> <td>bits [8:15] :</td> <td>TOS/TC value for yellow</td> </tr> <tr> <td>bits [16:23] :</td> <td>TOS/TC value for red</td> </tr> </table>	bits [0:7] :	TOS/TC value for green	bits [8:15] :	TOS/TC value for yellow	bits [16:23] :	TOS/TC value for red	0x0
bits [0:7] :	TOS/TC value for green								
bits [8:15] :	TOS/TC value for yellow								
bits [16:23] :	TOS/TC value for red								
34:27	tosMask	Mask for updating the TOS/TC field. For each bit in the mask, 0 means keep original value, 1 means update new value to that bit.	0x0						
37:35	color2Dei	New DEI value based on packet color. This is located in the outermost VLAN of the transmitted packet. <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 20%;">bit 0 :</td> <td>DEI value for green</td> </tr> <tr> <td>bit 1 :</td> <td>DEI value for yellow</td> </tr> <tr> <td>bit 2 :</td> <td>DEI value for red</td> </tr> </table>	bit 0 :	DEI value for green	bit 1 :	DEI value for yellow	bit 2 :	DEI value for red	0x0
bit 0 :	DEI value for green								
bit 1 :	DEI value for yellow								
bit 2 :	DEI value for red								

35.7.4 Debug Counter debugMatchEPP0 Setup

Packet processing debug setup for registerDebug debugMatchEPP0.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space

Slice 0 :	99300
Slice 1 :	196980
Slice 2 :	294660
Slice 3 :	392340



Field Description

Bits	Field Name	Description	Default Value
13:0	mask	Mask for comparison to update debug counter.	0x0
27:14	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.7.5 Debug Counter fromPort Setup

Packet processing debug setup for registerDebug fromPort.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99295
Slice 1 :	196975
Slice 2 :	294655
Slice 3 :	392335

Field Description

Bits	Field Name	Description	Default Value
31:0	mask	Mask for comparison to update debug counter.	0x0
63:32	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.7.6 Debug Counter reQueuePortId Setup

Packet processing debug setup for registerDebug reQueuePortId.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99294
Slice 1 :	196974
Slice 2 :	294654
Slice 3 :	392334

Field Description

Bits	Field Name	Description	Default Value
2:0	mask	Mask for comparison to update debug counter.	0x0



Bits	Field Name	Description	Default Value
5:3	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.7.7 Disable CPU tag on CPU Port

When a packet is sent to the CPU port normally a To CPU Tag will be added to the packet. This register provides a option to disable the CPU tag

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99299
Slice 1 :	196979
Slice 2 :	294659
Slice 3 :	392339

Field Description

Bits	Field Name	Description	Default Value
0	disable	When set, the CPU port will no longer add a CPU Tag to packets going to the CPU port. 0 = To CPU Tag enabled 1 = To CPU Tag disabled	0x0
1	disableReason0	When set, the CPU port will no longer add a CPU Tag to packets going to the CPU port with reason code 0(default reason). 0 = To CPU Tag enabled 1 = To CPU Tag disabled	0x0

35.7.8 Drain Port

Drop all packets on all queues to egress ports. The dropped packets are counted in the [Drain Port Drop](#) counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	91458
Slice 1 :	189138
Slice 2 :	286818
Slice 3 :	384498

Field Description

Bits	Field Name	Description	Default Value
7:0	drainMask	Egress ports to be drained. One bit for each port in the current switch slice where bit 0 corresponds to local port 0.	0x0



35.7.9 EPP Debug addNewMpls

Packet processing pipeline status for addNewMpls.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99313
Slice 1 :	196993
Slice 2 :	294673
Slice 3 :	392353

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.10 EPP Debug debugMatchEPP0

Packet processing pipeline status for debugMatchEPP0.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99315
Slice 1 :	196995
Slice 2 :	294675
Slice 3 :	392355

Field Description

Bits	Field Name	Description	Default Value
13:0	value	Status from last processed packet.	0x0

35.7.11 EPP Debug delSpecificVlan

Packet processing pipeline status for delSpecificVlan.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99309
Slice 1 :	196989
Slice 2 :	294669
Slice 3 :	392349



Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.12 EPP Debug fromPort

Packet processing pipeline status for fromPort.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99308
Slice 1 :	196988
Slice 2 :	294668
Slice 3 :	392348

Field Description

Bits	Field Name	Description	Default Value
31:0	value	Status from last processed packet.	0x0

35.7.13 EPP Debug imActive

Packet processing pipeline status for imActive.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99301
Slice 1 :	196981
Slice 2 :	294661
Slice 3 :	392341

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.14 EPP Debug imExtra

Packet processing pipeline status for imExtra.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99302
Slice 1 :	196982
Slice 2 :	294662
Slice 3 :	392342



Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.15 EPP Debug isIPv4

Packet processing pipeline status for isIPv4.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99311
Slice 1 :	196991
Slice 2 :	294671
Slice 3 :	392351

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.16 EPP Debug isIPv6

Packet processing pipeline status for isIPv6.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99312
Slice 1 :	196992
Slice 2 :	294672
Slice 3 :	392352

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.17 EPP Debug isPPPoE

Packet processing pipeline status for isPPPoE.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99314
Slice 1 :	196994
Slice 2 :	294674
Slice 3 :	392354



Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.18 EPP Debug omEnabled

Packet processing pipeline status for omEnabled.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99303
Slice 1 :	196983
Slice 2 :	294663
Slice 3 :	392343

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.19 EPP Debug omlmActive

Packet processing pipeline status for omlmActive.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99304
Slice 1 :	196984
Slice 2 :	294664
Slice 3 :	392344

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.20 EPP Debug reQueue

Packet processing pipeline status for reQueue.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99305
Slice 1 :	196985
Slice 2 :	294665
Slice 3 :	392345



Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.21 EPP Debug reQueuePkt

Packet processing pipeline status for reQueuePkt.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99307
Slice 1 :	196987
Slice 2 :	294667
Slice 3 :	392347

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.22 EPP Debug reQueuePortId

Packet processing pipeline status for reQueuePortId.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99306
Slice 1 :	196986
Slice 2 :	294666
Slice 3 :	392346

Field Description

Bits	Field Name	Description	Default Value
2:0	value	Status from last processed packet.	0x0

35.7.23 EPP Debug updateTosExp

Packet processing pipeline status for updateTosExp.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99310
Slice 1 :	196990
Slice 2 :	294670
Slice 3 :	392350



Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.7.24 Egress Ethernet Type for VLAN tag

Ethernet type used in VLAN operations when typeSel selects User Defined VLAN type. This Ethernet type is only used in VLAN push operations. In VLAN filtering a pushed user defined VLAN will be considered to be a C-VLAN.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99297
Slice 1 :	196977
Slice 2 :	294657
Slice 3 :	392337

Field Description

Bits	Field Name	Description	Default Value
15:0	typeValue	Ethernet Type value.	0xffff

35.7.25 Egress MPLS Decoding Options

When doing a Penultimate Pop then compare the first nibble after the innermost MPLS tag with this registers field nibbleForIpv4 to determine if the outgoing packet should have an IPv4 or IPv6 Ethernet Type.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99296
Slice 1 :	196976
Slice 2 :	294656
Slice 3 :	392336

Field Description

Bits	Field Name	Description	Default Value
3:0	nibbleForIpv4	The nibble value which is used to identify a IPv4 packet after a MPLS header. If the nibble does not match this value it is assumed to be an IPv6 packet.	0x4



35.7.26 Egress MPLS TTL Table

Configuration of what modification shall be done on the TTL field in MPLS routed packets.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : Packets VRF

Address Space

Slice 0 :	93994 to 93997
Slice 1 :	191674 to 191677
Slice 2 :	289354 to 289357
Slice 3 :	387034 to 387037

Field Description

Bits	Field Name	Description	Default Value
0	addNewTTL	Select if the router should decremented TTL in the outgoing packet or if it should be set to a fixed value. 0 = Decrement TTL 1 = Set the TTL to newTTL	0x0
8:1	newTTL	New TTL for the packet. Only used when addNewTTL is set to 1	0x0

35.7.27 Egress Multiple Spanning Tree State

Table of egress Multiple Spanning Tree Protocol Instances. Depends on routed or not, the pointer used to address the instance/entry in this table can from **msptPtr** in the **Next Hop Packet Modifications** table or **msptPtr** in the **VLAN Table**. Each entry contains the ingress spanning tree states for all ports in this MSTI.

Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : msptPtr from VLAN Table or Next Hop Packet Modifications Table

Address Space

Slice 0 :	82891 to 82906
Slice 1 :	180571 to 180586
Slice 2 :	278251 to 278266
Slice 3 :	375931 to 375946

Field Description

Bits	Field Name	Description	Default Value
63:0	portSptState	The egress spanning tree state for this MSTI. Bit[1:0] is the state for port #0, bit[3:2] is the state for port #1, etc. 0 = Forwarding 1 = Discarding 2 = Learning	0x0

35.7.28 Egress NAT Operation

Egress NAT Operation Table.



Number of Entries : 1024
 Type of Operation : Read/Write
 Addressing : Egress ACL NAT Pointer plus egress port number.

Address Space	Slice 0 : 96246 to 97269
	Slice 1 : 193926 to 194949
	Slice 2 : 291606 to 292629
	Slice 3 : 389286 to 390309

Field Description

Bits	Field Name	Description	Default Value
0	replaceSrc	Replace Source or Destination. 0 = Destination 1 = Source	0x0
1	replaceIP	Replace IP address. 0 = No. 1 = Yes.	0x0
2	replaceL4Port	Replace TCP/UDP port. 0 = No. 1 = Yes.	0x0
34:3	ipAddress	The new IP Address.	0x0
50:35	port	The new L4 Port.	0x0

35.7.29 Egress Port Configuration

This table configures various functions that are dependent on which port the packet leaves the switch. A VLAN operation (e.g. push, pop, swap) to be performed can be selected by the [vlanSingleOp](#) field. For the push and swap operations the information used to create the new VLAN header is controlled by the fields [vidSel](#), [cfiDeiSel](#), [pcpSel](#) and [typeSel](#). Other configurations are VLAN LUT index, port disable and different filtering rules based on packet VLAN fields when the egress processing is done.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space	Slice 0 : 91502 to 91533
	Slice 1 : 189182 to 189213
	Slice 2 : 286862 to 286893
	Slice 3 : 384542 to 384573

Field Description

Bits	Field Name	Description	Default Value
0	colorRemap	If set, color remapping to outgoing packet headers is allowed. The default color remapping options are based on the egress port number from the Color Remap From Egress Port table. If a packet is subjected to ingress admission control, its ingress admission control pointer can provide remap options from the Color Remap From Ingress Admission Control table to override default options.	0x0



Bits	Field Name	Description	Default Value
3:1	vlanSingleOp	The egress port VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0
4	removeSNAP	If a packet which has SNAP/LLC encoding then remove it before sending out the packet on this egress port. 0 = No. Keep it. 1 = Yes. Remove it.	0x0
6:5	typeSel	Selects which TPID to use when building a new VLAN header in a push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag field typeValue .	0x0
8:7	vidSel	Selects which VID to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's vid will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's vid . 2 = From the Ingress VID as selected in the Source Port Table .	0x0
10:9	cfiDeiSel	Selects which CFI/DEI to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's cfiDei will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's cfiDei . 2 = From Egress Queue To PCP And CFI/DEI Mapping Table .	0x0
12:11	pcpSel	Selects which PCP to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's cfiDei will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's pcp . 2 = From Egress Queue To PCP And CFI/DEI Mapping Table .	0x0
24:13	vid	The VID used in egress port VLAN push or swap operation if selected by vidSel .	0x0
25	cfiDei	The CFI/DEI used in egress port VLAN push or swap operation if selected by cfiDeiSel .	0x0



Bits	Field Name	Description	Default Value
28:26	pcp	The PCP used in egress port VLAN push or swap operation if selected by pcpSel .	0x0
29	disabled	Disabling this port. All packets to this port is dropped and Egress Port Disabled Drop is incremented. 0 = All packets will be sent out. 1 = All packets will be dropped.	0x0
30	dropCtaggedVlans	Drop or allow customer VLANs tagged packets on this egress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow C-VLANs. 1 = Drop C-VLANs.	0x0
31	dropStaggedVlans	Drop or allow service VLANs tagged packets on this egress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow S-VLANs. 1 = Drop S-VLANs.	0x0
32	moreThanOneVlans	When filtering with dropCtaggedVlans or dropStaggedVlans then this field must be set to 1.	0x0
33	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
34	dropSingleTaggedVlans	Drop or Allow packets that has one VLAN tag on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
35	dropDualTaggedVlans	Drop or allow packets which has more than one VLAN tag on this egress port. 0 = Allow packets which has more than one VLAN tag. 1 = Drop packets which has more than one VLAN tag.	0x0
36	dropCStaggedVlans	Drop or allow packets which has a C-VLAN followed by a S-VLAN tagged on this egress port. 0 = Allow packets which has a C-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a S-VLAN tag.	0x0
37	dropSCTaggedVlans	Drop or allow packets which has a S-VLAN followed by a C-VLAN tagged on this egress port. 0 = Allow packets which has a S-VLAN followed by a C-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a C-VLAN tag.	0x0
38	dropCCtaggedVlans	Drop or allow packets which has a C-VLAN followed by a C-VLAN tagged on this egress port. 0 = Allow packets which has a C-VLAN tag followed by a C-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a C-VLAN tag.	0x0



Bits	Field Name	Description	Default Value
39	dropSStaggedVlans	Drop or allow packets which has a S-VLAN followed by a S-VLAN tagged on this egress port. 0 = Allow packets which has a S-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a S-VLAN tag.	0x0
40	useEgressQueueRemapping	Which remapping to final PCP, DEI, EXP and TOS fields shall be used for this port. 0 = Only use Egress Queue Remapping Tables 1 = First use the Egress Queue Remapping Tables then use the Select Which Egress QoS Mapping Table To Use to determine the final DEI,CFI,TOS and EXP fields.	0x0

35.7.30 Egress Port VID Operation

This search table checks the ingress VID and the number of VLANs before the egress port VLAN operation. If both ingress VID and number of VLANs are in the defined range then the VLAN operation in this table will override egress port VLAN operations. In case of multiple hit, VLAN operation from the first hit takes effect.

Number of Entries : 16
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space

Slice 0 :	99316 to 99347
Slice 1 :	196996 to 197027
Slice 2 :	294676 to 294707
Slice 3 :	392356 to 392387

Field Description

Bits	Field Name	Description	Default Value
2:0	vlanSingleOpIf	If this entry is hit, then this VLAN operation will override egress port VLAN operation. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0
4:3	typeSelf	If this entry is hit, selects which TPID to use when building a new VLAN header in a push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag field typeValue .	0x0



Bits	Field Name	Description	Default Value
6:5	vidSelf	Selects which VID to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's vidlf will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's vidlf . 2 = From the Ingress VID as selected in the Source Port Table .	0x0
8:7	cfiDeiSelf	Selects which CFI/DEI to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's cfiDei will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's cfiDei . 2 = From Egress Queue To PCP And CFI/DEI Mapping Table .	0x0
10:9	pcpSelf	Selects which PCP to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's cfiDei will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's pcp . 2 = From Egress Queue To PCP And CFI/DEI Mapping Table .	0x0
22:11	vidlf	VID used in VLAN push or swap operation if vidSelf chooses VID from this table.	0x0
23	cfiDei	CFI/DEI used in VLAN push or swap operation if cfiDeiSelf chooses CFI/DEI from this table.	0x0
26:24	pcplf	PCP used in VLAN push or swap operation if pcpSelf chooses PCP from this table.	0x0
38:27	startVid	Start of ingress VID to hit.	0x0
50:39	endVid	End of ingress VID to hit.	0x0
53:51	minNrVlans	Minimum number of VLANs to hit	0x0
56:54	maxNrVlans	Maximum number of VLANs to hit	0x0
88:57	validPorts	Determine the valid egress port list.	0x0

35.7.31 Egress Queue To MPLS EXP Mapping Table

Map from egress queue number to MPLS EXP value to be used in MPLS operations selected by **Next Hop MPLS Table** and by **Next Hop Packet Insert MPLS Header** .

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress Queue

Address Space

Slice 0 :	99286 to 99293
Slice 1 :	196966 to 196973
Slice 2 :	294646 to 294653
Slice 3 :	392326 to 392333

Field Description



Bits	Field Name	Description	Default Value
2:0	exp	The outgoing Exp value for this queue.	0x0

35.7.32 Egress Queue To PCP And CFI/DEI Mapping Table

Get PCP and CFI/DEI from egress queues if selected by egress port VLAN operations push or swap.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress Queue

Address Space	Slice 0 : 94062 to 94069
	Slice 1 : 191742 to 191749
	Slice 2 : 289422 to 289429
	Slice 3 : 387102 to 387109

Field Description

Bits	Field Name	Description	Default Value
0	cfiDei	Map from egress queue to CFI/DEI.	0x0
3:1	pcp	Map from egress queue to PCP.	0x0

35.7.33 Egress Router Table

Configuration of what modification shall be done on the TTL field in routed packets.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : Packets VRF

Address Space	Slice 0 : 91934 to 91937
	Slice 1 : 189614 to 189617
	Slice 2 : 287294 to 287297
	Slice 3 : 384974 to 384977

Field Description

Bits	Field Name	Description	Default Value
0	addNewTTL	Select if the router should decremented TTL in the outgoing packet or if it should be set to a fixed value. 0 = Decrement TTL 1 = Set the TTL to newTTL	0x0
8:1	newTTL	New TTL for the packet. Only used when addNewTTL is set to 1	0x0

35.7.34 Egress Tunnel Exit Table

The same packet exit which is done at ingress described in the second tunnel exit lookup. Setting must be the same. This tunnel exit can also be used by the L2, L3 and ACL actions.



Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : From Various tables during ingress packet processing

Address Space	Slice 0 : 91534 to 91549
	Slice 1 : 189214 to 189229
	Slice 2 : 286894 to 286909
	Slice 3 : 384574 to 384589

Field Description

Bits	Field Name	Description	Default Value
8:0	howManyBytesToRemove	How many bytes to remove.	0x0
9	updateEthType	If packet is removed after L2+VLAN headers then update the Ethernet Header Type Field	0x0
25:10	ethType	If packet is removed after L2+VLAN headers then the New Ethernet Type which will overwrite the existing lowest 16 bits after the removal operation.	0x0
26	removeVlan	If packet is removed after L2+VLAN headers then remove the VLAN headers on the incoming packet.	0x0
27	updateL4Protocol	If packet is removed after L3 headers then update the L4 Protocol in IP header.	0x0
35:28	l4Protocol	If packet is removed after L3 headers then this new L4 Protocol will be written.	0x0
37:36	whereToRemove	Where to do the tunnel exit from 0 = At Byte Zero 1 = After L2 and up to two VLAN headers. 2 = After L3 IPv4/IPv6 headers. 3 = Reserved.	0x0

35.7.35 Egress VLAN Translation TCAM

The outermost VID and VID Ethernet Type (Service tag or Customer tag types) of the outgoing packet is compared.

Number of Entries : 128
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space	Slice 0 : 99158 to 99285
	Slice 1 : 196838 to 196965
	Slice 2 : 294518 to 294645
	Slice 3 : 392198 to 392325

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
5:1	dstPort_mask	Mask for dstPort.	0x1f
10:6	dstPort	The destination port which the packet is going out on	0x0
22:11	outermostVid_mask	Mask for outermostVid.	0xffff
34:23	outermostVid	The outermost VID of the modified packet.	0x0



Bits	Field Name	Description	Default Value
35	outermostVidType_mask	Mask for outermostVidType.	0x1
36	outermostVidType	The outermost VID is a S-tag or C-Tag. 0 = Customer tag 1 = Service tag	0x0

35.7.36 Egress VLAN Translation TCAM Answer

This is the table holding the answer for the [Egress VLAN Translation TCAM](#).

Number of Entries : 128
 Type of Operation : Read/Write
 Addressing : [Egress VLAN Translation TCAM](#) hit index

Address Space	Slice 0 :	94070 to 94197
	Slice 1 :	191750 to 191877
	Slice 2 :	289430 to 289557
	Slice 3 :	387110 to 387237

Field Description

Bits	Field Name	Description	Default Value
11:0	newVid	The new VID for the outgoing packet.	0x0
27:12	ethType	The new Ethernet Type for the outgoing packet	0x0

35.7.37 IP QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID and ECN bits in the TOS Byte if selected from [Select Which Egress QoS Mapping Table To Use](#). The rest of the TOS bits comes from the coloring mapping or MMP mapping tables.

Number of Entries : 256
 Type of Operation : Read/Write

Addressing :	Address [2:0] :	The egress queue which the packet was queued on.
	Address [4:3]:	The color of the packet.
	Address [6:5] :	The ECN ToS bits TOS[1:0] after coloring operation.
	Address [7] :	The Pointer from the Select Which Egress QoS Mapping Table To Use whichTablePtr .
Address Space	Slice 0 :	97846 to 98101
	Slice 1 :	195526 to 195781
	Slice 2 :	293206 to 293461
	Slice 3 :	390886 to 391141

Field Description



Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP	0x0
7:6	ecnTos	The outgoing TOS [1:0] ECN bits	0x0
8	updateExp	If the packet enters a new MPLS tunnel using the Next Hop Packet Insert MPLS Header then use this Exp for the outermost MPLS label. 0 = No. Dont Remap. 1 = Yes. Remap to this new value	0x0
11:9	newExp	New Exp value to be used.	0x0

35.7.38 Ingress NAT Operation

Ingress NAT Operation Table.

Number of Entries : 2048
 Type of Operation : Read/Write
 Addressing : Ingress ACL NAT Pointer plus egress port number.

Address Space

Slice 0 :	94198 to 96245
Slice 1 :	191878 to 193925
Slice 2 :	289558 to 291605
Slice 3 :	387238 to 389285

Field Description

Bits	Field Name	Description	Default Value
0	replaceSrc	Replace Source or Destination. 0 = Destiantion 1 = Source	0x0
1	replaceIP	Replace IP address. 0 = No. 1 = Yes.	0x0
2	replaceL4Port	Replace TCP/UDP port. 0 = No. 1 = Yes.	0x0
34:3	ipAddress	The new IP Address.	0x0
50:35	port	The new L4 Port.	0x0

35.7.39 L2 QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID if selected from [Select Which Egress QoS Mapping Table To Use](#).



Number of Entries : 64

Type of Operation : Read/Write

Addressing :	Address [2:0] :	The egress queue which the packet was queued on.
	Address [4:3]:	The color of the packet.
	Address [5] :	The Pointer from the Select Which Egress QoS Mapping Table To Use whichTablePtr.
Address Space	Slice 0 :	97782 to 97845
	Slice 1 :	195462 to 195525
	Slice 2 :	293142 to 293205
	Slice 3 :	390822 to 390885

Field Description

Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI.	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP.	0x0

35.7.40 L2 Tunnel Entry Instruction Table

The is the L2 tunnel entry instruction which described how a tunnel entry should be done after the L2 MAC and VLAN headers in the packet. If the L3Type is either IPv4, IPv6 then the length fields are updated in the IP headers, for IPv4 the checksum is re-calculated. If the hasUDP is turned on then the UDP length-field is updated.

Number of Entries : 16

Type of Operation : Read/Write

Addressing : Tunnel entry pointer

Address Space	Slice 0 :	91838 to 91853
	Slice 1 :	189518 to 189533
	Slice 2 :	287198 to 287213
	Slice 3 :	384878 to 384893

Field Description

Bits	Field Name	Description	Default Value
1:0	l3Type	Insert header type. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Other.	0x0
2	hasUdp	If the header is a IPv4 or IPv6 then a insert an UDP header after IP header.	0x0
3	updateEtherType	Shall the Ethernet Type be updated. 0 = No 1 = Yes	0x0
19:4	outerEtherType	EtherType preceding the tunnel entry point.	0x0



35.7.41 L3 Tunnel Entry Instruction Table

The is the L3 tunnel entry instruction which described how a tunnel entry should be done after the L3 IPv4/IPv6/MPLS headers in the packet.

Number of Entries :	16
Type of Operation :	Read/Write
Addressing :	Tunnel entry pointer
Address Space	Slice 0 : 91854 to 91869
	Slice 1 : 189534 to 189549
	Slice 2 : 287214 to 287229
	Slice 3 : 384894 to 384909

Field Description

Bits	Field Name	Description	Default Value
1:0	updateL4Type	If the packet is a IPv4 or IPv6 then the Next Header/Protocol field shall be updated. IPv4 Packet will see a updated header checksum.	0x0
9:2	I4Protocol	The new Next Header/Protocol byte	0x0

35.7.42 MPLS QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID and outermost EXP MPLS label if selected from [Select Which Egress QoS Mapping Table To Use](#).

Number of Entries :	512	
Type of Operation :	Read/Write	
Addressing :	Address [2:0] :	The egress queue which the packet was queued on.
	Address [4:3]:	The color of the packet.
	Address [7:5] :	The outermost label EXP bits.
	Address [8] :	The Pointer from the Select Which Egress QoS Mapping Table To Use whichTablePtr .
Address Space	Slice 0 : 98614 to 99125	
	Slice 1 : 196294 to 196805	
	Slice 2 : 293974 to 294485	
	Slice 3 : 391654 to 392165	

Field Description

Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI.	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP.	0x0



Bits	Field Name	Description	Default Value
8:6	exp	The outgoing Exp value for this queue in the outer-most MPLS label.	0x0

35.7.43 NAT Add Egress Port for NAT Calculation

Should the ingress and egress NAT pointers from the ingress and egress ACL be added with the egress port number.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	99298
Slice 1 :	196978
Slice 2 :	294658
Slice 3 :	392338

Field Description

Bits	Field Name	Description	Default Value
0	dontAddIngress	Do not add egress port when calculating the ingress NAT offset pointer. 0 = Add Egress Port. 1 = Do not add Egress Port.	0x0
1	dontAddEgress	Do not add egress port when calculating the egress NAT offset pointer. 0 = Add Egress Port. 1 = Do not add Egress Port.	0x0

35.7.44 Next Hop DA MAC

Determines the destination MAC address to use in the packet exiting the router.

Number of Entries : 1024

Type of Operation : Read/Write

Addressing : [nextHopPacketMod](#)

Address Space

Slice 0 :	91938 to 92961
Slice 1 :	189618 to 190641
Slice 2 :	287298 to 288321
Slice 3 :	384978 to 386001

Field Description

Bits	Field Name	Description	Default Value
47:0	daMac	The destination MAC address for the next hop.	0x0

35.7.45 Next Hop MPLS Table

Determines the MPLS tag operation to perform.



Number of Entries : 1024
 Type of Operation : Read/Write
 Addressing : [nextHopPacketMod](#)

Address Space

Slice 0 :	92970 to 93993
Slice 1 :	190650 to 191673
Slice 2 :	288330 to 289353
Slice 3 :	386010 to 387033

Field Description

Bits	Field Name	Description	Default Value
2:0	mplsOperation	The egress MPLS tag operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate Pop(remove all MPLS tags).	0x0
4:3	expSel	Select which EXP bits to use when building a new MPLS tag in Push or Swap operation. 0 = From this entries EXP field. 1 = From egress queue remapping in Egress Queue To MPLS EXP Mapping Table 2 = From the MPLS label (outermost MPLS tag if a swap and innermost if a push).	0x0
7:5	exp	Value to use for the EXP field when building a new MPLS tag in a swap or push operation.	0x0
27:8	label	MPLS label to use when building a new MPLS tag in a swap or push operation.	0x0

35.7.46 Next Hop Packet Insert MPLS Header

Shall MPLS labels (up to 4) be inserted on the packet before it is sent out. This enables a IP packet to go into a MPLS tunnel. Header is placed after L2 and VLANs before the IP packet header. MPLS EXP field comes from destination queue to EXP mapping table defined in [Egress Queue To MPLS EXP Mapping Table](#). Only the lowest entries from 0 to 16-1 in the

fieldNext Hop Table[nextHopPacketMod](#) can be used to insert a MPLS header.

Number of Entries : 16
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : [nextHopPacketMod](#) bits [3 : 0]

Address Space

Slice 0 :	93998 to 94061
Slice 1 :	191678 to 191741
Slice 2 :	289358 to 289421
Slice 3 :	387038 to 387101

Field Description

Bits	Field Name	Description	Default Value
2:0	howManyLabelsToInsert	How many labels shall be inserted. Setting a zero here means no labels will be added.	0x0



Bits	Field Name	Description	Default Value
3	whichEthernetType	Which Ethernet Type shall be used for these MPLS labels. 0 = 0x8847 1 = 0x8848	0x0
23:4	mplsLabel0	First/Outermost MPLS label to be inserter.	0x0
24	copyTtl0	Where shall the TTL come from in the MPLS label 0. 0 = From this table, field ttl0. 1 = From the inner packet.	0x0
32:25	ttl0	TTL table value for MPLS label 0.	0x0
33	expFromQueue0	Where shall the EXP come from in the MPLS label 0. 0 = From this table, field exp0. 1 = From the Egress Queue To MPLS EXP Mapping Table .	0x0
36:34	exp0	EXP table value for MPLS label 0.	0x0
56:37	mplsLabel1	MPLS label 1 to be inserter.	0x0
57	copyTtl1	Where shall the TTL come from in the MPLS label 1. 0 = From this table, field ttl1. 1 = From the inner packet.	0x0
65:58	ttl1	TTL table value for MPLS label 1.	0x0
66	expFromQueue1	Where shall the EXP come from in the MPLS label 1. 0 = From this table, field exp1. 1 = From the Egress Queue To MPLS EXP Mapping Table .	0x0
69:67	exp1	EXP table value for MPLS label 1.	0x0
89:70	mplsLabel2	MPLS label 2 to be inserter.	0x0
90	copyTtl2	Where shall the TTL come from in the MPLS label 2. 0 = From this table, field ttl2. 1 = From the inner packet.	0x0
98:91	ttl2	TTL table value for MPLS label 2.	0x0
99	expFromQueue2	Where shall the EXP come from in the MPLS label 2. 0 = From this table, field exp2. 1 = From the Egress Queue To MPLS EXP Mapping Table .	0x0
102:100	exp2	EXP table value for MPLS label 2.	0x0
122:103	mplsLabel3	MPLS label 3 to be inserter.	0x0
123	copyTtl3	Where shall the TTL come from in the MPLS label 3. 0 = From this table, field ttl3. 1 = From the inner packet.	0x0
131:124	ttl3	TTL table value for MPLS label 3.	0x0
132	expFromQueue3	Where shall the EXP come from in the MPLS label 3. 0 = From this table, field exp3. 1 = From the Egress Queue To MPLS EXP Mapping Table .	0x0
135:133	exp3	EXP table value for MPLS label 3.	0x0



35.7.47 Output Mirroring Table

Output mirroring configuration. An egress port can be set to have a mirrored port, but output mirroring cannot link more than one port. i.e. If Port A has an output mirroring Port B, Port B has an output mirroring Port C, packets sent to port A will not be mirrored to Port C.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space	Slice 0 : 99126 to 99157
	Slice 1 : 196806 to 196837
	Slice 2 : 294486 to 294517
	Slice 3 : 392166 to 392197

Field Description

Bits	Field Name	Description	Default Value
0	outputMirrorEnabled	If set to one, output mirroring is enabled for this port.	0x0
5:1	outputMirrorPort	Destination of output mirroring. Only valid if outputMirrorEnabled is set. Notice if the design contains more than one switch slice, packets egressed on one slice cannot be mirrored to another slice.	0x0

35.7.48 Router Port Egress SA MAC Address

The routers SA MAC address to use when a packet exits the router. In normal cases this would be the incoming Destination MAC address. However when using NAT there are cases which this does not work and hence this table allows the usage of a alternative MAC address.

Number of Entries : 4
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write
 Addressing : VRF

Address Space	Slice 0 : 92962 to 92969
	Slice 1 : 190642 to 190649
	Slice 2 : 288322 to 288329
	Slice 3 : 386002 to 386009

Field Description

Bits	Field Name	Description	Default Value
31:0	selectMacEntryPortMask	Portmask to select which SA MAC address to use as router MAC address. One bit per destination port. 0 = use incoming packets DA MAC address. 1 = use altMacAddress.	0x0
79:32	altMacAddress	The alternative base destination MAC address that is used to identify packets to the router.	0x0



35.7.49 Select Which Egress QoS Mapping Table To Use

This is the initial table which is looked up by all packets in order to determine how the mapping from internal QoS to packets final PCP, DEI, TOS/EXP field shall look like. In order for this table to be executed the field [useEgressQueueRemapping](#) must be set to one.

Number of Entries : 512

Type of Operation : Read/Write

Addressing :	Address Bit [1:0]:	Forwarding type to this port. 0 = Switched Packet 1 = Routed Packet 2 = Classification Rule Forwarded Packet 3 = Others - Send-to-CPU and packet from CPU
	Address Bit [3:2]:	Packet type 0 = L2 - Not IPv4/IPv6/MPLS 1 = IPv4 2 = IPv6 3 = MPLS
	Address Bit [9:4]:	Egress Port
Address Space	Slice 0 :	97270 to 97781
	Slice 1 :	194950 to 195461
	Slice 2 :	292630 to 293141
	Slice 3 :	390310 to 390821

Field Description

Bits	Field Name	Description	Default Value
2:0	whichTableToUse	Select which table type to use. 0 = None. No remapping 1 = L2 QoS Mapping Table 2 = IP QoS Mapping Table 3 = TOS QoS Mapping Table 4 = MPLS QoS Mapping Table 5 = Use this tables remapping of DEI and PCP bits.	0x0
3	whichTablePtr	Which index of the tables to use. For most QoS tables there exists multiple tables to choose from.	0x0
4	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5	cfiDei	Packets new CFI/DEI.	0x0
6	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
9:7	pcp	Packets new PCP.	0x0

35.7.50 TOS QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID and TOS Byte if selected from [Select Which Egress QoS Mapping Table To Use](#). The input TOS byte to this mapping table comes from the coloring or MMP mapping tables.



Number of Entries : 512

Type of Operation : Read/Write

Addressing :	Address [7:0] :	The TOS byte.
	Address [8] :	The Pointer from the Select Which Egress QoS Mapping Table To Use whichTablePtr .
Address Space	Slice 0 :	98102 to 98613
	Slice 1 :	195782 to 196293
	Slice 2 :	293462 to 293973
	Slice 3 :	391142 to 391653

Field Description

Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP	0x0
13:6	newTos	The outgoing new TOS bits	0x0
14	updateExp	If the packet enters a new MPLS tunnel using the Next Hop Packet Insert MPLS Header then use this Exp for the outermost MPLS label. 0 = No. Dont Remap. 1 = Yes. Remap to this new value	0x0
17:15	newExp	New Exp value to be used.	0x0

35.7.51 Tunnel Entry Header Data

The this is the byte data which is used to do tunnel insertions. The data to be used is pointed to from the [Tunnel Entry Instruction Table](#)

Number of Entries : 16

Number of Addresses per Entry : 16

Type of Operation : Read/Write

Addressing : [tunnelHeaderPtr](#)

Address Space	Slice 0 :	91566 to 91821
	Slice 1 :	189246 to 189501
	Slice 2 :	286926 to 287181
	Slice 3 :	384606 to 384861

Field Description

Bits	Field Name	Description	Default Value
639:0	data	Tunnel header data (bytes) to be inserted at tunnel entry point in packet. Byte 0 is the start of the tunnel header.	0x0



35.7.52 Tunnel Entry Instruction Table

The tunnel entry instruction describes how a tunnel shall be entered. The same pointer address is used to read out the [Beginning of Packet Tunnel Entry Instruction Table](#) , [L2 Tunnel Entry Instruction Table](#) and [L3 Tunnel Entry Instruction Table](#). The field tunnelEntryType determine which tunnel entry table to use. The insertion of the length field is independent from the other tunnel header length updates which is done.

Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : Tunnel Entry Pointer from various tables

Address Space

Slice 0 :	91550 to 91565
Slice 1 :	189230 to 189245
Slice 2 :	286910 to 286925
Slice 3 :	384590 to 384605

Field Description

Bits	Field Name	Description	Default Value
1:0	tunnelEntryType	A tunnel entry shall be done. Where shall the tunnel entry be done 0 = At Byte Zero described in Beginning of Packet Tunnel Entry Instruction Table 1 = After L2 and up to two VLAN headers. described in L2 Tunnel Entry Instruction Table 2 = After L3 IPv4/IPv6/MPLS headers. 3 = Reserved.	0x0
2	insertLength	Insert the a packet length fields. The 2 byte length of the frame will overwrite current 2 bytes in the header data to be inserted at lengthPos . 0 = Yes. Insert a length field. 1 = No. Don't insert a length field.	0x0
9:3	lengthPos	If length shall be inserted , where shall it be inserted. A value of 0 means beginning of tunnel entry data.	0x0
23:10	lengthNegOffset	How much shall be decremented from the total packet (frame) length.	0x0
37:24	lengthPosOffset	How much shall be incremented from the total packet (frame) length.	0x0
38	incVlansInLength	Should the outgoing packets number of VLANs be included in the length calculation? 0 = No. 1 = Yes.	0x0
42:39	tunnelHeaderPtr	Points to which header to insert from register Tunnel Entry Header Data .	0x0
49:43	tunnelHeaderLen	The length of the tunnel header, in bytes, to insert from register Tunnel Entry Header Data .	0x0

35.8 Flow Control

35.8.1 FFA Used PFC

Total number of cells from the common pool used by ports in PFC-mode.



Number of Entries :	1
Type of Operation :	Read Only
Address Space	Slice 0 : 90370
	Slice 1 : 188050
	Slice 2 : 285730
	Slice 3 : 383410

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.2 FFA Used non-PFC

Total number of cells used from the common pool by ports in non-PFC mode.

Number of Entries :	1
Type of Operation :	Read Only
Address Space	Slice 0 : 90371
	Slice 1 : 188051
	Slice 2 : 285731
	Slice 3 : 383411

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.3 PFC Dec Counters for ingress ports 0 to 7

Wrapping counters of deallocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port}) + \text{Traffic class}$
Address Space	Slice 0 : 90090 to 90153
	Slice 1 : 187770 to 187833
	Slice 2 : 285450 to 285513
	Slice 3 : 383130 to 383193

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0



35.8.4 PFC Dec Counters for ingress ports 16 to 23

Wrapping counters of deallocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port} - 16) + \text{Traffic class}$
Address Space	Slice 0 : 90218 to 90281
	Slice 1 : 187898 to 187961
	Slice 2 : 285578 to 285641
	Slice 3 : 383258 to 383321

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.5 PFC Dec Counters for ingress ports 24 to 31

Wrapping counters of deallocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port} - 24) + \text{Traffic class}$
Address Space	Slice 0 : 90282 to 90345
	Slice 1 : 187962 to 188025
	Slice 2 : 285642 to 285705
	Slice 3 : 383322 to 383385

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.6 PFC Dec Counters for ingress ports 8 to 15

Wrapping counters of deallocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port}) + \text{Traffic class}$
Address Space	Slice 0 : 90154 to 90217
	Slice 1 : 187834 to 187897
	Slice 2 : 285514 to 285577
	Slice 3 : 383194 to 383257



Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.7 PFC Inc Counters for ingress ports 0 to 7

Wrapping counters of allocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port-0}) + \text{Traffic class}$
Address Space	Slice 0 : 89834 to 89897
	Slice 1 : 187514 to 187577
	Slice 2 : 285194 to 285257
	Slice 3 : 382874 to 382937

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.8 PFC Inc Counters for ingress ports 16 to 23

Wrapping counters of allocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port-2}) + \text{Traffic class}$
Address Space	Slice 0 : 89962 to 90025
	Slice 1 : 187642 to 187705
	Slice 2 : 285322 to 285385
	Slice 3 : 383002 to 383065

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0



35.8.9 PFC Inc Counters for ingress ports 24 to 31

Wrapping counters of allocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port}-3) + \text{Traffic class}$
Address Space	Slice 0 : 90026 to 90089
	Slice 1 : 187706 to 187769
	Slice 2 : 285386 to 285449
	Slice 3 : 383066 to 383129

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.10 PFC Inc Counters for ingress ports 8 to 15

Wrapping counters of allocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries :	64
Type of Operation :	Read Only
Addressing :	$8 * (\text{Source port}-1) + \text{Traffic class}$
Address Space	Slice 0 : 89898 to 89961
	Slice 1 : 187578 to 187641
	Slice 2 : 285258 to 285321
	Slice 3 : 382938 to 383001

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.11 Port FFA Used

Number of cells used from the common pool for this source port

Number of Entries :	8
Type of Operation :	Read Only
Addressing :	Source port
Address Space	Slice 0 : 90346 to 90353
	Slice 1 : 188026 to 188033
	Slice 2 : 285706 to 285713
	Slice 3 : 383386 to 383393



Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.12 Port Pause Settings

Pause settings per source port.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Source port

Address Space

Slice 0 :	90372 to 90379
Slice 1 :	188052 to 188059
Slice 2 :	285732 to 285739
Slice 3 :	383412 to 383419

Field Description

Bits	Field Name	Description	Default Value
0	enable	0 = Pausing disabled 1 = Pausing enabled	0x0
1	mode	On a port where both pausing and tail-drop is enabled the modes must match for the calculation of used FFA to be correct. 0 = Priority mode 1 = Port mode	0x0
3:2	reserved	Reserved.	0x0
11:4	force	Each bit refers to one traffic class (bit 0 = TC 0) 0 = No force 1 = Force the pause state to that set in the pattern field Only valid if pausing is enabled.	0x0
19:12	pattern	Each bit refers to one traffic class (bit 0 = TC 0) 0 = Not paused 1 = Paused	0x0

35.8.13 Port Reserved

Number of cells reserved in the buffer memory for this source port. Shall be set to zero for prio-mode ports
 Note that this setting can only be changed for an empty port.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Source port

Address Space

Slice 0 :	89826 to 89833
Slice 1 :	187506 to 187513
Slice 2 :	285186 to 285193
Slice 3 :	382866 to 382873



Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x7

35.8.14 Port Tail-Drop FFA Threshold

Settings for the Port Tail-Drop FFA Threshold

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Source port

Address Space

Slice 0 :	90431 to 90438
Slice 1 :	188111 to 188118
Slice 2 :	285791 to 285798
Slice 3 :	383471 to 383478

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Tail-drop threshold in number of cells. When the FFA cells used by the source port reaches this threshold no further packets will be accepted for this source port	0x1004
13	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

35.8.15 Port Tail-Drop Settings

Tail-drop settings per source port.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Source port

Address Space

Slice 0 :	90380 to 90387
Slice 1 :	188060 to 188067
Slice 2 :	285740 to 285747
Slice 3 :	383420 to 383427

Field Description

Bits	Field Name	Description	Default Value
0	enable	0 = Tail-drop is disabled for this source port 1 = Tail-drop is enabled for this source port	0x0



Bits	Field Name	Description	Default Value
1	mode	On a port where both pausing and tail-drop is enabled the modes must match for the calculation of used FFA to be correct. 0 = Priority mode 1 = Port mode	0x0

35.8.16 Port Used

Total number of cells used for this source port

Number of Entries : 8
Type of Operation : Read Only
Addressing : Source port

Address Space

Slice 0 :	90354 to 90361
Slice 1 :	188034 to 188041
Slice 2 :	285714 to 285721
Slice 3 :	383394 to 383401

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.17 Port Xoff FFA Threshold

Settings for Port Xoff FFA Threshold

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Source port

Address Space

Slice 0 :	90423 to 90430
Slice 1 :	188103 to 188110
Slice 2 :	285783 to 285790
Slice 3 :	383463 to 383470

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xoff threshold for the number of used FFA cells for this source port	0x0
13	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0



35.8.18 Port Xon FFA Threshold

Settings for Port Xon FFA Threshold

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	Source port
Address Space	Slice 0 : 90415 to 90422
	Slice 1 : 188095 to 188102
	Slice 2 : 285775 to 285782
	Slice 3 : 383455 to 383462

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xon threshold for the number of used FFA cells for this source port	0x0

35.8.19 Port/TC Reserved

Number of cells reserved in the buffer memory for this source port and traffic class. For ports set to port-mode this should be 0 for all queues. Note that this setting can only be changed for an empty port.

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	8 * Source port + Traffic class
Address Space	Slice 0 : 89762 to 89825
	Slice 1 : 187442 to 187505
	Slice 2 : 285122 to 285185
	Slice 3 : 382802 to 382865

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.20 Port/TC Tail-Drop Total Threshold

Settings for Port/TC Tail-Drop Total Threshold

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	8 * Source port + Traffic class
Address Space	Slice 0 : 90567 to 90630
	Slice 1 : 188247 to 188310
	Slice 2 : 285927 to 285990
	Slice 3 : 383607 to 383670



Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Tail-drop threshold in number of cells. When the sum of reserved and FFA cells used by this specific source port and traffic class combination reaches this threshold no further packets will be accepted for this source port and traffic class	0x1004
13	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

35.8.21 Port/TC Xoff Total Threshold

Settings for Port/TC Xoff Total Threshold

Number of Entries : 64
 Type of Operation : Read/Write
 Addressing : 8 * Source port + Traffic class

Address Space

Slice 0 :	90503 to 90566
Slice 1 :	188183 to 188246
Slice 2 :	285863 to 285926
Slice 3 :	383543 to 383606

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xoff threshold for the sum of reserved and FFA cells used for this source port and traffic class combination	0x0
13	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

35.8.22 Port/TC Xon Total Threshold

Settings for Port/TC Xon Total Threshold

Number of Entries : 64
 Type of Operation : Read/Write
 Addressing : 8 * Source port + Traffic class

Address Space

Slice 0 :	90439 to 90502
Slice 1 :	188119 to 188182
Slice 2 :	285799 to 285862
Slice 3 :	383479 to 383542



Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xon threshold for the sum of reserved and FFA cells used for this source port and traffic class combination	0x0

35.8.23 TC FFA Used

Number of cells used from the common pool for this traffic class.

Number of Entries : 8
 Type of Operation : Read Only
 Addressing : Traffic class

Address Space	Slice 0 :	90362 to 90369
	Slice 1 :	188042 to 188049
	Slice 2 :	285722 to 285729
	Slice 3 :	383402 to 383409

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells	0x0

35.8.24 TC Tail-Drop FFA Threshold

Settings for TC Tail-Drop FFA Threshold

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Traffic class

Address Space	Slice 0 :	90407 to 90414
	Slice 1 :	188087 to 188094
	Slice 2 :	285767 to 285774
	Slice 3 :	383447 to 383454

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Tail-drop threshold in number of cells. When the FFA cells used by the traffic class reaches this threshold no further packets will be accepted for this traffic class	0x1004
13	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0



Bits	Field Name	Description	Default Value
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

35.8.25 TC Xoff FFA Threshold

Settings for TC Xoff FFA Threshold

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Traffic class

Address Space	Slice 0 : 90399 to 90406
	Slice 1 : 188079 to 188086
	Slice 2 : 285759 to 285766
	Slice 3 : 383439 to 383446

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xoff threshold for the number of used FFA cells for this traffic class	0x0
13	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

35.8.26 TC Xon FFA Threshold

Settings for TC Xon FFA Threshold

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Traffic class

Address Space	Slice 0 : 90391 to 90398
	Slice 1 : 188071 to 188078
	Slice 2 : 285751 to 285758
	Slice 3 : 383431 to 383438

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xon threshold for the number of used FFA cells for this traffic class	0x0



35.8.27 Tail-Drop FFA Threshold

Settings for Tail-Drop FFA Threshold

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 90390
	Slice 1 : 188070
	Slice 2 : 285750
	Slice 3 : 383430

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Tail-drop threshold in number of cells. When the total number of FFA cells used reaches this threshold no further packets will be accepted.	0xf08
13	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

35.8.28 Xoff FFA Threshold

Settings for Xoff FFA Threshold

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 90389
	Slice 1 : 188069
	Slice 2 : 285749
	Slice 3 : 383429

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xoff threshold for the total number of used FFA cells	0x0
13	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
14	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

35.8.29 Xon FFA Threshold

Settings for Xon FFA Threshold



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 90388
	Slice 1 : 188068
	Slice 2 : 285748
	Slice 3 : 383428

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Xon threshold for the total number of used FFA cells	0x0

35.9 Global Configuration**35.9.1 Core Tick Configuration**

Global register for setting the frequency of the core tick

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	3

Field Description

Bits	Field Name	Description	Default Value
20:0	clkDivider	The master Core Tick will be issued once every $rg_tick_div.clkDivider$ core clock cycles. If set to zero, there will be no tick.	0x78
24:21	stepDivider	The seven ticks derived from the master core tick are issued once every $rg_tick_div.stepDivider^{tick_number+1}$ master ticks. The master tick is tick number 0. If stepDivider is set to zero, there will be no ticks except possibly the master tick.	0xa

35.9.2 Core Tick Select

Global register for setting clock input to the core tick divider

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	4

Field Description

Bits	Field Name	Description	Default Value
1:0	clkSelect	Select the source clock for the Core Tick divider. 0: disabled, 1: core clock, 2: debug_write_data[0], 3: reserved	0x1

35.9.3 MAC RX Maximum Packet Length

Packets with length above this value will be dropped.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Ingress Port
 Address Space : 80 to 111

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x4003

35.9.4 Scratch

Scratch Register

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 2

Field Description

Bits	Field Name	Description	Default Value
63:0	scratch	scratch field.	0x0

35.10 Ingress Packet Processing

35.10.1 AH Header Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a Authentical Header, the underlying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write

Address Space

Slice 0 :	85501
Slice 1 :	183181
Slice 2 :	280861
Slice 3 :	378541



Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	I4Proto	The value to be used to find this packet type.	0x33
40:9	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
72:41	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.2 ARP Packet Decoder Options

The Ethernet type used to determine if a packet is a ARP packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write

Address Space

Slice 0 :	85487
Slice 1 :	183167
Slice 2 :	280847
Slice 3 :	378527

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x806
48:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
80:49	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.3 Aging Data FIFO

This register exposes the output of a FIFO which is holding all aging requests from the aging unit. Under hardware aging writeback mode, the entry pushed to this FIFO is in sync with the [FIB](#). If hardware aging writeback is turned off, the final aging decision should be issued from software injected learning packet and what is pushed to this FIFO is not updated to L2 tables.



Number of Entries : 1
 Type of Operation : Read Only
 Address Space : 4801

Field Description

Bits	Field Name	Description	Default Value
3:0	hashClearValid	One bit per bucket, each bit set to 1 means the aging unit has requested to change the corresponding hash bucket valid bit from 1 to 0 hence clear out this entry.	0x0
7:4	hashClearHit	One bit per bucket, each bit set to 1 means the aging unit has requested to change corresponding hash bucket hit bit from 1 to 0.	0x0
17:8	hashValue	Hash of GID, MAC.	0x0
18	camClearValid	When this field is 1, the aging unit has requested to change the corresponding cam entry valid bit from 1 to 0 hence clear out this entry.	0x0
19	camClearHit	When this field is 1, the aging unit has requested to change the corresponding cam entry hit bit from 1 to 0.	0x0
24:20	camIndex	Index to the entry in L2 Aging Collision Table .	0x0
25	valid	0 = Empty FIFO, entry is not valid 1 = Valid entry	0x0

35.10.4 Aging Data FIFO High Watermark Level

The High Watermark Interrupt will occur when a push to [Aging Data FIFO](#) is done and the number of existing entries after the push is larger than this setting.

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 603

Field Description

Bits	Field Name	Description	Default Value
5:0	level	Number of used entries.	0x0

35.10.5 Allow Special Frame Check For L2 Action Table

The result in [L2 Action Table](#) is a pointer field [allowPtr](#) which allows result from the L2 SA Action Table to setup rules of which types of packets/frames are allowed to be sent in on a port. If any of there is a match and packet is not allowed then all instances are dropped of this packet. The drop counter [L2 Action Table Special Packet Type Drop](#) is updated.



Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : Result from **L2 Action Table**

Address Space

Slice 0 :	82887 to 82890
Slice 1 :	180567 to 180570
Slice 2 :	278247 to 278250
Slice 3 :	375927 to 375930

Field Description

Bits	Field Name	Description	Default Value
0	dontAllowBPDU	Allow BPDU frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
1	dontAllow8021X_EAPOL	Allow 802.1X EAPOL frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
2	dontAllowCAPWAP	Allow CAPWAP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
3	dontAllowARP	Allow ARP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
4	dontAllowRARP	Allow RARP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
5	dontAllowDNS	Allow DNS frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
6	dontAllowBOOTP_DHCP	Allow BOOTP_DHCP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
7	dontAllowSCTP	Allow STCP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
8	dontAllowLLDP	Allow LLDP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
9	dontAllowGRE	Allow GRE frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
10	dontAllowESP	Allow ESP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
11	dontAllowAH	Allow AH frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
12	dontAllowL2_1588	Allow L2 1588 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
13	dontAllowL4_1588	Allow L4 1588 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0



Bits	Field Name	Description	Default Value
14	dontAllowICMP	Allow ICMP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
15	dontAllowIGMP	Allow IGMP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
16	dontAllowL2McReserved	Allow L2 Reserved Da frames, see register L2 Reserved Multicast Address Base . 0 = Allow frame. 1 = Do not allow frame.	0x0
17	dontAllowIPV4	Allow IPV4 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
18	dontAllowIPV6	Allow IPV6 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
19	dontAllowUDP	Allow UDP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
20	dontAllowTCP	Allow TCP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
21	dontAllowMPLS	Allow MPLS frames. 0 = Allow frame. 1 = Do not allow frame.	0x0

35.10.6 BOOTP and DHCP Packet Decoder Options

The UDP port 1 number used by the BOOTP protocol, the underlying packet must be a IPv4 packet. If L4 Source Port is this value then L4 Destination Port must be registerbootpUdpPort2 value and vice versa. . If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space	Slice 0 : 85507
	Slice 1 : 183187
	Slice 2 : 280867
	Slice 3 : 378547

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	udp1	The value to be used to find this packet type.	0x43
32:17	udp2	The value to be used to find this packet type.	0x44



Bits	Field Name	Description	Default Value
64:33	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
96:65	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.7 CAPWAP Packet Decoder Options

The fields needs to determine if a packet is a CAPWAP packet the underlying packet must be a IPv4 or IPv6 packet. . If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write

Address Space

Slice 0 :	85509
Slice 1 :	183189
Slice 2 :	280869
Slice 3 :	378549

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	udp1	The value to be used to find this packet type.	0x147e
32:17	udp2	The value to be used to find this packet type.	0x147f
64:33	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
96:65	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.8 CPU Reason Code Operation

When a packet raises a send to CPU action during the ingress packet process, follow-up operations can be performed based on the reason code. In this table 16 ranges are searched in order and the same action hit in the latter range overrides the previous hit.



Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space	Slice 0 : 82315 to 82330
	Slice 1 : 179995 to 180010
	Slice 2 : 277675 to 277690
	Slice 3 : 375355 to 375370

Field Description

Bits	Field Name	Description	Default Value
0	mutableCpu	Force the packet to another port instead of the CPU port when the CPU reason code hit in the range.	0x0
5:1	port	The new destination to replace the CPU port.	0x0
6	forceQueue	Force the packet to the CPU port with a new egress queue when the CPU reason code hit in the range.	0x0
9:7	eQueue	Egress queue	0x0
10	forceUpdateOrigCpuPkt	If this reason code is hit shall the origCpuPkt field be updated? 0 = No, no update. 1 = Yes, update.	0x0
11	origCpuPkt	Force the packet to the CPU to be the original, unmodified, packet. 0 = No, modification will happen to packet. 1 = Yes, force the packet to be unmodified.	0x0
27:12	start	Start of CPU reason code.	0x0
43:28	end	End of CPU reason code.	0x0

35.10.9 Check IPv4 Header Checksum

This register provides an option to drop the IPv4 packet if its header checksum field has an incorrect value. The option is only for not routed IPv4 packet. For a routed IPv4 packet, the checksum check is always performed.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 81922
	Slice 1 : 179602
	Slice 2 : 277282
	Slice 3 : 374962

Field Description

Bits	Field Name	Description	Default Value
0	dropErrorChkSum	If set, always calculate the checksum of the received IPv4 packet. If the calculated value does not match the IPv4 checksum field, the packet is dropped.	0x0



35.10.10 DNS Packet Decoder Options

The TCP/UDP destination port number used to determine if a packet is a DNS packet, the underlying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space	Slice 0 : 85505
	Slice 1 : 183185
	Slice 2 : 280865
	Slice 3 : 378545

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	I4Port	The value to be used to find this packet type.	0x35
48:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
80:49	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.11 Debug Counter debugMatchIPP0 Setup

Packet processing debug setup for registerDebug debugMatchIPP0.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81939
	Slice 1 : 179619
	Slice 2 : 277299
	Slice 3 : 374979

Field Description

Bits	Field Name	Description	Default Value
21:0	mask	Mask for comparison to update debug counter.	0x0
43:22	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0



35.10.12 Debug Counter dstPortmask Setup

Packet processing debug setup for registerDebug dstPortmask.

Number of Entries : 1
Type of Operation : Read/Write

Address Space

Slice 0 :	81938
Slice 1 :	179618
Slice 2 :	277298
Slice 3 :	374978

Field Description

Bits	Field Name	Description	Default Value
31:0	mask	Mask for comparison to update debug counter.	0x0
63:32	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.13 Debug Counter finalVid Setup

Packet processing debug setup for registerDebug finalVid.

Number of Entries : 1
Type of Operation : Read/Write

Address Space

Slice 0 :	81917
Slice 1 :	179597
Slice 2 :	277277
Slice 3 :	374957

Field Description

Bits	Field Name	Description	Default Value
12:0	mask	Mask for comparison to update debug counter.	0x0
25:13	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.14 Debug Counter l2DaHash Setup

Packet processing debug setup for registerDebug l2DaHash.



Number of Entries : 1

Type of Operation : Read/Write

Address Space	Slice 0 : 81927
	Slice 1 : 179607
	Slice 2 : 277287
	Slice 3 : 374967

Field Description

Bits	Field Name	Description	Default Value
9:0	mask	Mask for comparison to update debug counter.	0x0
19:10	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.15 Debug Counter I2DaHashHitAndBucket Setup

Packet processing debug setup for
registerDebug I2DaHashHitAndBucket.

Number of Entries : 1

Type of Operation : Read/Write

Address Space	Slice 0 : 81928
	Slice 1 : 179608
	Slice 2 : 277288
	Slice 3 : 374968

Field Description

Bits	Field Name	Description	Default Value
2:0	mask	Mask for comparison to update debug counter.	0x0
5:3	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.16 Debug Counter I2DaHashKey Setup

Packet processing debug setup for
registerDebug I2DaHashKey.

Number of Entries : 1

Number of Addresses per Entry : 2

Type of Operation : Read/Write

Address Space	Slice 0 : 85513
	Slice 1 : 183193
	Slice 2 : 280873
	Slice 3 : 378553



Field Description

Bits	Field Name	Description	Default Value
59:0	mask	Mask for comparison to update debug counter.	0x0
119:60	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.17 Debug Counter I2DaTcamHitsAndCast Setup

Packet processing debug setup for registerDebug I2DaTcamHitsAndCast.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81929
Slice 1 :	179609
Slice 2 :	277289
Slice 3 :	374969

Field Description

Bits	Field Name	Description	Default Value
14:0	mask	Mask for comparison to update debug counter.	0x0
29:15	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.18 Debug Counter nextHopPtrFinal Setup

Packet processing debug setup for registerDebug nextHopPtrFinal.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81921
Slice 1 :	179601
Slice 2 :	277281
Slice 3 :	374961

Field Description

Bits	Field Name	Description	Default Value
9:0	mask	Mask for comparison to update debug counter.	0x0



Bits	Field Name	Description	Default Value
19:10	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.19 Debug Counter nextHopPtrHash Setup

Packet processing debug setup for registerDebug nextHopPtrHash.

Number of Entries : 1
Type of Operation : Read/Write

Address Space	Slice 0 : 81920
	Slice 1 : 179600
	Slice 2 : 277280
	Slice 3 : 374960

Field Description

Bits	Field Name	Description	Default Value
9:0	mask	Mask for comparison to update debug counter.	0x0
19:10	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.20 Debug Counter nextHopPtrLpm Setup

Packet processing debug setup for registerDebug nextHopPtrLpm.

Number of Entries : 1
Type of Operation : Read/Write

Address Space	Slice 0 : 81919
	Slice 1 : 179599
	Slice 2 : 277279
	Slice 3 : 374959

Field Description

Bits	Field Name	Description	Default Value
9:0	mask	Mask for comparison to update debug counter.	0x0
19:10	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0



35.10.21 Debug Counter nrVlans Setup

Packet processing debug setup for registerDebug nrVlans.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81913
	Slice 1 : 179593
	Slice 2 : 277273
	Slice 3 : 374953

Field Description

Bits	Field Name	Description	Default Value
1:0	mask	Mask for comparison to update debug counter.	0x0
3:2	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.22 Debug Counter spVidOp Setup

Packet processing debug setup for registerDebug spVidOp.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81914
	Slice 1 : 179594
	Slice 2 : 277274
	Slice 3 : 374954

Field Description

Bits	Field Name	Description	Default Value
2:0	mask	Mask for comparison to update debug counter.	0x0
5:3	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0



35.10.23 Debug Counter srcPort Setup

Packet processing debug setup for registerDebug srcPort.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81907
	Slice 1 : 179587
	Slice 2 : 277267
	Slice 3 : 374947

Field Description

Bits	Field Name	Description	Default Value
4:0	mask	Mask for comparison to update debug counter.	0x0
9:5	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.24 Debug Counter vlanVidOp Setup

Packet processing debug setup for registerDebug vlanVidOp.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81918
	Slice 1 : 179598
	Slice 2 : 277278
	Slice 3 : 374958

Field Description

Bits	Field Name	Description	Default Value
2:0	mask	Mask for comparison to update debug counter.	0x0
5:3	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparison is carried out. If comparison results in true the counter is updated	0x0

35.10.25 Default Packet To CPU Modification

Shall packets which are sent to the CPU be modified or original incoming packets. If a packet is switch / routed the to the CPU port then it will come out as the modified packet. This register only is relevant when a packet is sent to the cpu using Send-to-CPU flag (ie. when reason code != 0).



Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Source Port

Address Space

Slice 0 :	84955 to 84986
Slice 1 :	182635 to 182666
Slice 2 :	280315 to 280346
Slice 3 :	377995 to 378026

Field Description

Bits	Field Name	Description	Default Value
0	origCpuPkt	Force the packet to the CPU to be the original, unmodified, packet. The exception to this is rule is the tunnel exit which will still be carried out. 0 = No, modification will happen to packet. 1 = Yes, force the packet to be unmodified.	0x0

35.10.26 ESP Header Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a Authentic Header, the underlying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write

Address Space

Slice 0 :	85503
Slice 1 :	183183
Slice 2 :	280863
Slice 3 :	378543

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	I4Proto	The value to be used to find this packet type.	0x32
40:9	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
72:41	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.27 Egress ACL Rule Pointer TCAM

D-left search that determines which ACL rule pointers to use when building the search key for the egress ACL lookups.



Number of Entries : 64
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space	Slice 0 : 85359 to 85486
	Slice 1 : 183039 to 183166
	Slice 2 : 280719 to 280846
	Slice 3 : 378399 to 378526

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
32:1	destPortMask_mask	Mask for destPortMask.	2 ³² - 1
64:33	destPortMask	The packets egress ports, one bit per port.	0x0
65	routed_mask	Mask for routed.	0x1
66	routed	The packet was routed.	0x0
68:67	vrf_mask	Mask for vrf.	0x3
70:69	vrf	The VRF used when routed.	0x0
71	flooded_mask	Mask for flooded.	0x1
72	flooded	The packet was flooded due to L2 table miss.	0x0
73	ucSwitched_mask	Mask for ucSwitched.	0x1
74	ucSwitched	The packet was L2 switched to a unicast destination port.	0x0
75	mcSwitched_mask	Mask for mcSwitched.	0x1
76	mcSwitched	The packet was L2 switched to a multicast group.	0x0
88:77	vid_mask	Mask for vid.	0xffff
100:89	vid	The index used in the VLAN table lookup.	0x0
102:101	l3Type_mask	Mask for l3Type.	0x3
104:103	l3Type	The packets L3 Type. ab- FourIPv4IPv6MPLSOther	0x0
107:105	l4Type_mask	Mask for l4Type.	0x7
110:108	l4Type	The packets L4 Type. abEightNot known.Is IPv4 or IPv6 but type is not any L4 type in this list.UDPTCPiGMPiCmPv6MLD	0x0
115:111	srcPort_mask	Mask for srcPort.	0x1f
120:116	srcPort	The packets source port.	0x0

35.10.28 Egress ACL Rule Pointer TCAM Answer

This is the table holding the answer for the **Egress ACL Rule Pointer TCAM**.

Number of Entries : 64
 Type of Operation : Read/Write
 Addressing : **Egress ACL Rule Pointer TCAM** hit index

Address Space	Slice 0 : 76674 to 76737
	Slice 1 : 174354 to 174417
	Slice 2 : 272034 to 272097
	Slice 3 : 369714 to 369777



Field Description

Bits	Field Name	Description	Default Value
2:0	rulePtr0	Rule Pointer for egress ACL engine 0.	0x0
4:3	rulePtr1	Rule Pointer for egress ACL engine 1.	0x0

35.10.29 Egress Configurable ACL 0 Large Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 1024

Number of Addresses per Entry : 4

Type of Operation : Read/Write

Addressing :	address[7:0] : hash of {compareData }
	address[9:8] : bucket number
	Slice 0 : 76738 to 80833
	Slice 1 : 174418 to 178513
Address Space	Slice 2 : 272098 to 276193
	Slice 3 : 369778 to 373873

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	compareData	The data which shall be compared in this entry.	0x0
136	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
137	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
138	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
154:139	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
155	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
156	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the Egress Configurable ACL Drop counter is incremented.	0x0
157	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0



Bits	Field Name	Description	Default Value
162:158	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
163	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
169:164	counter	This is a result field used when this entry is hit. Which counter in Egress Configurable ACL Match Counter to update.	0x0
170	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
180:171	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
181	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
182	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
183	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
187:184	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
188	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0

35.10.30 Egress Configurable ACL 0 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 7 fields (bits) which are set to one are selected to build the lookup key for this ACL. It is not allowed to set more than 7 bit in the bitmask. The fields are described in [ACL Fields](#)

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : ACL rule pointer

Address Space	Slice 0 : 82879 to 82886
	Slice 1 : 180559 to 180566
	Slice 2 : 278239 to 278246
	Slice 3 : 375919 to 375926

Field Description



Bits	Field Name	Description	Default Value
17:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 7 bits should be set.	0x0

35.10.31 Egress Configurable ACL 0 Search Mask

Before the hashing and searching is done in the [Egress Configurable ACL 0 Large Table](#) and [Egress Configurable ACL 0 Small Table](#). The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Separate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Address Space	Slice 0 : 86317
	Slice 1 : 183997
	Slice 2 : 281677
	Slice 3 : 379357

Field Description

Bits	Field Name	Description	Default Value
134:0	mask.small	Which bits to compare in the Egress Configurable ACL 0 Small Table lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{135} - 1$
269:135	mask.large	Which bits to compare in the Egress Configurable ACL 0 Large Table lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{135} - 1$

35.10.32 Egress Configurable ACL 0 Selection

This register selects which result to use when there are multiple hits.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81934
	Slice 1 : 179614
	Slice 2 : 277294
	Slice 3 : 374974

Field Description



Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

35.10.33 Egress Configurable ACL 0 Small Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 256

Number of Addresses per Entry : 4

Type of Operation : Read/Write

Addressing :	address[5:0] :	hash of {compareData }
	address[7:6] :	bucket number
	Slice 0 :	80834 to 81857
	Slice 1 :	178514 to 179537
Address Space	Slice 2 :	276194 to 277217
	Slice 3 :	373874 to 374897

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	compareData	The data which shall be compared in this entry.	0x0
136	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
137	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
138	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
154:139	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
155	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
156	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the Egress Configurable ACL Drop counter is incremented.	0x0
157	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0



Bits	Field Name	Description	Default Value
162:158	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
163	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
169:164	counter	This is a result field used when this entry is hit. Which counter in Egress Configurable ACL Match Counter to update.	0x0
170	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
180:171	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
181	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
182	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
183	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
187:184	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
188	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0

35.10.34 Egress Configurable ACL 0 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries : 16
 Number of Addresses per Entry : 8
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space

Slice 0 :	86325 to 86452
Slice 1 :	184005 to 184132
Slice 2 :	281685 to 281812
Slice 3 :	379365 to 379492

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0



Bits	Field Name	Description	Default Value
135:1	mask	Which bits to compare in this entry.	$2^{135} - 1$
270:136	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

35.10.35 Egress Configurable ACL 0 TCAM Answer

This is the table holding the answer for the [Egress Configurable ACL 0 TCAM](#).

Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : [Egress Configurable ACL 0 TCAM](#) hit index

Address Space

Slice 0 :	81858 to 81873
Slice 1 :	179538 to 179553
Slice 2 :	277218 to 277233
Slice 3 :	374898 to 374913

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the Egress Configurable ACL Drop counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
26:22	destPort	The port which the packet shall be sent to.	0x0
27	updateCounter	When set the selected statistics counter will be updated.	0x0
33:28	counter	Which counter in Egress Configurable ACL Match Counter to update.	0x0
34	natOpValid	NAT operation pointer is valid.	0x0
44:35	natOpPtr	NAT operation pointer.	0x0
45	natOpPrio	If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
46	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0
47	tunnelEntryUcMc	Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0



Bits	Field Name	Description	Default Value
51:48	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
52	tunnelEntryPrio	If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0

35.10.36 Egress Configurable ACL 1 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 20 fields (bits) which are set to one are selected to build the lookup key for this ACL. It is not allowed to set more than 20 bit in the bitmask. The fields are described in [ACL Fields](#)

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : ACL rule pointer

Address Space

Slice 0 :	82875 to 82878
Slice 1 :	180555 to 180558
Slice 2 :	278235 to 278238
Slice 3 :	375915 to 375918

Field Description

Bits	Field Name	Description	Default Value
19:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 20 bits should be set.	0x0

35.10.37 Egress Configurable ACL 1 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries : 16
 Number of Addresses per Entry : 32
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space

Slice 0 :	87285 to 87796
Slice 1 :	184965 to 185476
Slice 2 :	282645 to 283156
Slice 3 :	380325 to 380836

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
540:1	mask	Which bits to compare in this entry.	$2^{540} - 1$



Bits	Field Name	Description	Default Value
1080:541	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

35.10.38 Egress Configurable ACL 1 TCAM Answer

This is the table holding the answer for the [Egress Configurable ACL 1 TCAM](#).

Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : [Egress Configurable ACL 1 TCAM](#) hit index

Address Space	Slice 0 : 81874 to 81889
	Slice 1 : 179554 to 179569
	Slice 2 : 277234 to 277249
	Slice 3 : 374914 to 374929

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the Egress Configurable ACL Drop counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
26:22	destPort	The port which the packet shall be sent to.	0x0
27	updateCounter	When set the selected statistics counter will be updated.	0x0
33:28	counter	Which counter in Egress Configurable ACL Match Counter to update.	0x0
34	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0
35	tunnelEntryUcMc	Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
39:36	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
40	tunnelEntryPrio	If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0



35.10.39 Egress Port NAT State

At end of ingress processing a check is done to determine what to do with packets which has different port states and what the ingress and egress ACLs says what shall be done with the packets. The table needs to be enabled in the [natActionTableEnable](#).

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81935
	Slice 1 : 179615
	Slice 2 : 277295
	Slice 3 : 374975

Field Description

Bits	Field Name	Description	Default Value
31:0	portState	Egress Port NAT state (Bit 0 is port 0, bit 1 is port 1 etc.). 0 = Private 1 = Public	0x0

35.10.40 Egress Spanning Tree State

Spanning tree state for each egress port. The state Disabled implies that spanning tree protocol is not enabled and hence frames will be forwarded on this egress port.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space	Slice 0 : 85515
	Slice 1 : 183195
	Slice 2 : 280875
	Slice 3 : 378555

Field Description

Bits	Field Name	Description	Default Value
95:0	sptState	State of the spanning tree protocol. Bit[2:0] is port #0, bit[5:3] is port #1 etc. 0 = Disabled 1 = Blocking 2 = Listening 3 = Learning 4 = Forwarding	0x0

35.10.41 Enable Enqueue To Ports And Queues

This register is used to control if a particular port and queue shall be able to enqueue new packets. One queue mask exists for each port, setting a bit in the queue mask means packet is allowed to be queued on the respective queue. Packets that are directed to a queue that is turned off will be dropped and counted in [Queue Off Drop](#).



Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space	Slice 0 : 81963 to 81994
	Slice 1 : 179643 to 179674
	Slice 2 : 277323 to 277354
	Slice 3 : 375003 to 375034

Field Description

Bits	Field Name	Description	Default Value
7:0	q_on	If a bit is set, the corresponding queue is on.	0xff

35.10.42 Flooding Action Send to Port

If a packet is flooded and this function is enabled on the source port then the packet is sent to a single egress port instead of being flooded to all ports part of the packets VLAN membership.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Source Port
Address Space	Slice 0 : 81995 to 82026
	Slice 1 : 179675 to 179706
	Slice 2 : 277355 to 277386
	Slice 3 : 375035 to 375066

Field Description

Bits	Field Name	Description	Default Value
0	enable	Enable sent to port instead of flooding. 0 = Disable 1 = Enable	0x0
5:1	destPort	Once enabled this is the destination port to sent the packet to in case of flooding.	0x0

35.10.43 Force Non VLAN Packet To Specific Color

If a packet is non-VLAN tagged, there is an option to force these packets to a certain initial color.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81925
	Slice 1 : 179605
	Slice 2 : 277285
	Slice 3 : 374965

Field Description

Bits	Field Name	Description	Default Value
0	forceColor	When set, packets which are non-VLAN tagged are forced to a color.	0x0
2:1	color	Initial color of the packet	0x0

35.10.44 Force Non VLAN Packet To Specific Queue

If a packet is non-VLAN tagged, there is an option to force these packets to a certain ingress/egress queue.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81923
Slice 1 :	179603
Slice 2 :	277283
Slice 3 :	374963

Field Description

Bits	Field Name	Description	Default Value
0	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
3:1	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

35.10.45 Force Unknown L3 Packet To Specific Color

If a packet does not contain IPv4, IPv6, MPLS or PPPoE carrying IPv4/IPv6 field there is an option to force the packet to a certain initial color.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81926
Slice 1 :	179606
Slice 2 :	277286
Slice 3 :	374966

Field Description

Bits	Field Name	Description	Default Value
0	forceColor	When set, unknown L3 packet types are forced to a color.	0x0
2:1	color	Initial color of the packet	0x0



35.10.46 Force Unknown L3 Packet To Specific Egress Queue

If a packet does not contain IPv4, IPv6, MPLS or PPPoE carrying IPv4/IPv6 field there is an option to force the packet to a certain egress queue.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81924
	Slice 1 : 179604
	Slice 2 : 277284
	Slice 3 : 374964

Field Description

Bits	Field Name	Description	Default Value
0	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
3:1	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

35.10.47 Forward From CPU

Indicates if all frames received on the CPU port shall be forwarded while ignoring the egress port's spanning tree status.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81930
	Slice 1 : 179610
	Slice 2 : 277290
	Slice 3 : 374970

Field Description

Bits	Field Name	Description	Default Value
0	enable	If set, any frame received on the CPU port is forwarded without consideration of the egress port's spanning tree state.	0x0

35.10.48 GRE Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a GRE header. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space	Slice 0 : 85495
	Slice 1 : 183175
	Slice 2 : 280855
	Slice 3 : 378535



Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	I4Proto	The value to be used to find this packet type.	0x2f
24:9	udp1	The value to be used to find this packet type.	0x1292
40:25	udp2	The value to be used to find this packet type.	0x1293
72:41	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
104:73	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.49 Hairpin Enable

Decide if the L2 switching allows a packet to be switched back on the same port it entered the switch. There are separate controls for flooding due to unknown MAC DA, multicast and unicast.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : Ingress port

Address Space

Slice 0 :	83035 to 83066
Slice 1 :	180715 to 180746
Slice 2 :	278395 to 278426
Slice 3 :	376075 to 376106

Field Description

Bits	Field Name	Description	Default Value
0	allowFlood	Allow flooding to source port.	0x0
1	allowMc	Allow multicast to source port.	0x0
2	allowUc	Allow unicast to source port.	0x1

35.10.50 Hardware Learning Configuration

Configure default status for a newly learned entry, learning limits and learning exceptions.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : Ingress Port
Address Space : 605 to 636

Field Description

Bits	Field Name	Description	Default Value
0	valid	For a new packet which is to be learned what value shall the valid bit have?	0x1
1	stat	For a new packet which is to be learned what value shall the static bit have?	0x0
2	hit	For a new packet which is to be learned what value shall the hit bit have?	0x1
15:3	learnLimit	Maximum number of entries can be learned on this port. 0 means no limit.	0x0
16	portMoveException	When the hardware learning unit is turned on and the ingress packet processing determines to bypass the hardware learning check, set this field to one to still perform the port move action.	0x0
17	saHitException	When the hardware learning unit is turned on and the ingress packet processing determines to bypass the hardware learning check, set this field to one to still perform the SA hit update action.	0x0

35.10.51 Hardware Learning Counter

Number of MAC addresses learned by the hardware learning unit. Write 0 to clear.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Ingress Port
 Address Space : 670 to 701

Field Description

Bits	Field Name	Description	Default Value
12:0	cnt	Number of learned L2 entries.	0x0

35.10.52 Hash Based L3 Routing Table

This is the routing table used to determine the next hop. The IP lookup is done by hashing the VRF and the destination address extracted from the incoming packet. The hash is used to index this table. For each hash value the table has 4 buckets. The incoming IP address is compared with the destIPAddr field in all the buckets for the selected hash value. The packets assigned VRF is compared with the vrf fields and the protocol type is compared against the entries protocol. If there is a match in any bucket then the other fields in the matched bucket will be used for next hop processing. If ECMP is enabled for this entry an offset is added to the [nextHopPointer](#) and used when indexing the [Next Hop Table](#).



Number of Entries :	2048
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	address[0:8] : hash of {VRF, IP destination address} or {Source port and outermost MPLS label}
	address[9:10] : bucket number
Address Space	Slice 0 : 41538 to 49729
	Slice 1 : 139218 to 147409
	Slice 2 : 236898 to 245089
	Slice 3 : 334578 to 342769

Field Description

Bits	Field Name	Description	Default Value
0	ipVersion	Select if this is an IPv4 or IPv6 entry. 0 = IPv4 entry. 1 = IPv6 entry.	0x0
1	mpls	This is an MPLS entry, 0 = IP entry. 1 = MPLS entry.	0x0
3:2	vrf	This entries VRF. The packets assigned VRF will be compared with this field.	0x0
131:4	destIPAddr	The IP or MPLS address to be matched. If the entry is an IPv4 entry then only bits [31:0] is used. If the entry is a MPLS entry then bits [5-1:0] contains the source port while bits [5+19:5] contains the MPLS label to match.	0x0
141:132	nextHopPointer	Index into the Next Hop Table for this destination.	0x0
142	useECMP	Enables the use of ECMP hash to calculate the next hop pointer. 0 = Use ECMP hash. 1 = Do not use ECMP hash.	0x0
148:143	ecmpMask	How many bits of the ECMP hash will be used when calculating the ECMP offset. This byte is AND:ed with the ECMP hash to determine which bits shall be used as offset.	0x0
151:149	ecmpShift	How many bits the masked ECMP hash will be right shifted.	0x0

35.10.53 Hit Update Data FIFO

This register exposes the output of a FIFO which is holding all hit update requests to refresh the hit state. Under hardware hit writeback mode, the entry pushed to this FIFO is in sync with the [FIB](#). If hardware hit writeback is turned off, the final hit update decision should be issued from software injected learning packet and what is pushed to this FIFO is not updated to L2 tables.

Number of Entries :	1
Type of Operation :	Read Only
Address Space :	4802

Field Description



Bits	Field Name	Description	Default Value
0	hashRefreshHit	When this field is 1, the learning and aging engine has requested to refresh the hit state from 0 to 1 for the hash table in FIB.	0x0
10:1	hashValue	Hash of GID, MAC.	0x0
12:11	hashBucket	Bucket number of the hash lookup table.	0x0
13	camRefreshHit	When this field is 1, the learning and aging engine has requested to refresh the hit state from 0 to 1 for the cam entry.	0x0
18:14	camIndex	Index to the entry in L2 Aging Collision Table .	0x0
19	valid	0 = Empty FIFO, entry is not valid 1 = Valid entry	0x0

35.10.54 Hit Update Data FIFO High Watermark Level

The High Watermark Interrupt will occur when a push to [Hit Update Data FIFO](#) is done and the number of existing entries after the push is larger than this setting.

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 604

Field Description

Bits	Field Name	Description	Default Value
5:0	level	Number of used entries	0x0

35.10.55 IEEE 1588 L2 Packet Decoder Options

The Ethernet type used to determine if a packet is a IEEE 1588 L2 Packet. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write

Address Space

Slice 0 :	85491
Slice 1 :	183171
Slice 2 :	280851
Slice 3 :	378531

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x88f7



Bits	Field Name	Description	Default Value
48:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
80:49	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0
81	ptp	If a packet is sent to the CPU and this bit is set and the packet has a timestamp then it will show having a valid timestamp in the CPU-header.	0x0

35.10.56 IEEE 1588 L4 Packet Decoder Options

IEEE 1588 L4 packet is determined by this register. Fields from L2/L3/L4 are required for the comparison, including two optional DA MAC, five optional IPv4 DA, two optional IPv6 DA with the first one maskable, and two optional UDP destination ports. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 16
 Type of Operation : Read/Write

Address Space

Slice 0 :	86029
Slice 1 :	183709
Slice 2 :	281389
Slice 3 :	379069

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
48:1	da_mac1	DA MAC to match.	0x11b19000000
96:49	da_mac2	DA MAC to match.	0x180c200000e
128:97	da_ipv4_addr1	IPv4 DA to match.	0xe0000181
160:129	da_ipv4_addr2	IPv4 DA to match.	0xe0000182
192:161	da_ipv4_addr3	IPv4 DA to match.	0xe0000183
224:193	da_ipv4_addr4	IPv4 DA to match.	0xe0000184
256:225	da_ipv4_addr5	IPv4 DA to match.	0xe000016b
384:257	da_ipv6_addr1	IPv6 DA to match. This address is maskable.	0x1810000000000000000000000000ff0
512:385	da_ipv6_mask1	Bit mask for da_ipv6_addr1. For each bit of the mask, 1 means valid for comparison.	0xffff0ffffffffffffffffffffffffffff
640:513	da_ipv6_addr2	IPv6 DA to match.	0x6b000000000000000000000000ff02
656:641	udp1	UDP destination to match.	0x13f
672:657	udp2	UDP destination to match.	0x140
704:673	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0



Bits	Field Name	Description	Default Value
736:705	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0
737	ptp	If a packet is sent to the CPU and this bit is set and the packet has a timestamp then it will show having a valid timestamp in the CPU-header.	0x0

35.10.57 IEEE 802.1X and EAPOL Packet Decoder Options

The Ethernet type used to determine if a packet is a 802.1X or EAPOL packet. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
	Slice 0 : 85493
	Slice 1 : 183173
	Slice 2 : 280853
	Slice 3 : 378533

Address Space

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x888e
48:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
80:49	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.58 IKE Packet Decoder Options

The UDP ports used to detect a IKE packet the underlying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
	Slice 0 : 85511
	Slice 1 : 183191
	Slice 2 : 280871
	Slice 3 : 378551

Address Space



Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	udp1	The value to be used to find this packet type.	0x1f4
32:17	udp2	The value to be used to find this packet type.	0x1194
64:33	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
96:65	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.59 IPP Debug debugMatchIPP0

Packet processing pipeline status for debugMatchIPP0.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81962
Slice 1 :	179642
Slice 2 :	277322
Slice 3 :	375002

Field Description

Bits	Field Name	Description	Default Value
21:0	value	Status from last processed packet.	0x0

35.10.60 IPP Debug doL2Lookup

Packet processing pipeline status for doL2Lookup.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81960
Slice 1 :	179640
Slice 2 :	277320
Slice 3 :	375000

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0



35.10.61 IPP Debug dropPktAfterL2Decode

Packet processing pipeline status for dropPktAfterL2Decode.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81941
	Slice 1 : 179621
	Slice 2 : 277301
	Slice 3 : 374981

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.10.62 IPP Debug dropPktAfterL3Decode

Packet processing pipeline status for dropPktAfterL3Decode.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81943
	Slice 1 : 179623
	Slice 2 : 277303
	Slice 3 : 374983

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.10.63 IPP Debug dstPortmask

Packet processing pipeline status for dstPortmask.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81961
	Slice 1 : 179641
	Slice 2 : 277321
	Slice 3 : 375001

Field Description

Bits	Field Name	Description	Default Value
31:0	value	Status from last processed packet.	0x0



35.10.64 IPP Debug finalVid

Packet processing pipeline status for finalVid.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81945
	Slice 1 : 179625
	Slice 2 : 277305
	Slice 3 : 374985

Field Description

Bits	Field Name	Description	Default Value
12:0	value	Status from last processed packet.	0x0

35.10.65 IPP Debug isBroadcast

Packet processing pipeline status for isBroadcast.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81959
	Slice 1 : 179639
	Slice 2 : 277319
	Slice 3 : 374999

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.10.66 IPP Debug isFlooding

Packet processing pipeline status for isFlooding.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81958
	Slice 1 : 179638
	Slice 2 : 277318
	Slice 3 : 374998

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0



35.10.67 IPP Debug I2DaHash

Packet processing pipeline status for I2DaHash.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81954
	Slice 1 : 179634
	Slice 2 : 277314
	Slice 3 : 374994

Field Description

Bits	Field Name	Description	Default Value
9:0	value	Status from last processed packet.	0x0

35.10.68 IPP Debug I2DaHashHitAndBucket

Packet processing pipeline status for I2DaHashHitAndBucket.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81955
	Slice 1 : 179635
	Slice 2 : 277315
	Slice 3 : 374995

Field Description

Bits	Field Name	Description	Default Value
2:0	value	Status from last processed packet.	0x0

35.10.69 IPP Debug I2DaHashKey

Packet processing pipeline status for I2DaHashKey.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81953
	Slice 1 : 179633
	Slice 2 : 277313
	Slice 3 : 374993

Field Description

Bits	Field Name	Description	Default Value
59:0	value	Status from last processed packet.	0x0



35.10.70 IPP Debug I2DaTcamHitsAndCast

Packet processing pipeline status for I2DaTcamHitsAndCast.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81956
	Slice 1 : 179636
	Slice 2 : 277316
	Slice 3 : 374996

Field Description

Bits	Field Name	Description	Default Value
14:0	value	Status from last processed packet.	0x0

35.10.71 IPP Debug nextHopPtrFinal

Packet processing pipeline status for nextHopPtrFinal.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81952
	Slice 1 : 179632
	Slice 2 : 277312
	Slice 3 : 374992

Field Description

Bits	Field Name	Description	Default Value
9:0	value	Status from last processed packet.	0x0

35.10.72 IPP Debug nextHopPtrHash

Packet processing pipeline status for nextHopPtrHash.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81949
	Slice 1 : 179629
	Slice 2 : 277309
	Slice 3 : 374989

Field Description

Bits	Field Name	Description	Default Value
9:0	value	Status from last processed packet.	0x0



35.10.73 IPP Debug nextHopPtrHashHit

Packet processing pipeline status for nextHopPtrHashHit.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81951
	Slice 1 : 179631
	Slice 2 : 277311
	Slice 3 : 374991

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.10.74 IPP Debug nextHopPtrLpm

Packet processing pipeline status for nextHopPtrLpm.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81948
	Slice 1 : 179628
	Slice 2 : 277308
	Slice 3 : 374988

Field Description

Bits	Field Name	Description	Default Value
9:0	value	Status from last processed packet.	0x0

35.10.75 IPP Debug nextHopPtrLpmHit

Packet processing pipeline status for nextHopPtrLpmHit.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81950
	Slice 1 : 179630
	Slice 2 : 277310
	Slice 3 : 374990

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0



35.10.76 IPP Debug nrVlans

Packet processing pipeline status for nrVlans.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81942
	Slice 1 : 179622
	Slice 2 : 277302
	Slice 3 : 374982

Field Description

Bits	Field Name	Description	Default Value
1:0	value	Status from last processed packet.	0x0

35.10.77 IPP Debug routed

Packet processing pipeline status for routed.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81957
	Slice 1 : 179637
	Slice 2 : 277317
	Slice 3 : 374997

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

35.10.78 IPP Debug routerHit

Packet processing pipeline status for routerHit.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81947
	Slice 1 : 179627
	Slice 2 : 277307
	Slice 3 : 374987

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0



35.10.79 IPP Debug spVidOp

Packet processing pipeline status for spVidOp.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81944
	Slice 1 : 179624
	Slice 2 : 277304
	Slice 3 : 374984

Field Description

Bits	Field Name	Description	Default Value
2:0	value	Status from last processed packet.	0x0

35.10.80 IPP Debug srcPort

Packet processing pipeline status for srcPort.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81940
	Slice 1 : 179620
	Slice 2 : 277300
	Slice 3 : 374980

Field Description

Bits	Field Name	Description	Default Value
4:0	value	Status from last processed packet.	0x0

35.10.81 IPP Debug vlanVidOp

Packet processing pipeline status for vlanVidOp.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 81946
	Slice 1 : 179626
	Slice 2 : 277306
	Slice 3 : 374986

Field Description

Bits	Field Name	Description	Default Value
2:0	value	Status from last processed packet.	0x0



35.10.82 IPv4 TOS Field To Egress Queue Mapping Table

Mapping table from TOS in the IPv4 header to an egress queue.

Number of Entries :	256
Type of Operation :	Read/Write
Addressing :	Incoming IPv4 packets TOS
Address Space	Slice 0 : 84007 to 84262
	Slice 1 : 181687 to 181942
	Slice 2 : 279367 to 279622
	Slice 3 : 377047 to 377302

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

35.10.83 IPv4 TOS Field To Packet Color Mapping Table

Mapping table from TOS in the IPv4 header to a packet initial color.

Number of Entries :	256
Type of Operation :	Read/Write
Addressing :	Incoming IPv4 packets TOS pointer
Address Space	Slice 0 : 83471 to 83726
	Slice 1 : 181151 to 181406
	Slice 2 : 278831 to 279086
	Slice 3 : 376511 to 376766

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

35.10.84 IPv6 Class of Service Field To Egress Queue Mapping Table

Mapping table from Class of Service in the IPv6 header to an egress queue.

Number of Entries :	256
Type of Operation :	Read/Write
Addressing :	Incoming IPv6 packets Class of Service
Address Space	Slice 0 : 83751 to 84006
	Slice 1 : 181431 to 181686
	Slice 2 : 279111 to 279366
	Slice 3 : 376791 to 377046

Field Description



Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

35.10.85 IPv6 Class of Service Field To Packet Color Mapping Table

Mapping table from Class of service in the IPv6 header to a packet initial color.

Number of Entries : 256
 Type of Operation : Read/Write
 Addressing : Incoming IPv6 packets Class of Service pointer

Address Space	Slice 0 : 83215 to 83470
	Slice 1 : 180895 to 181150
	Slice 2 : 278575 to 278830
	Slice 3 : 376255 to 376510

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

35.10.86 Ingress Admission Control Current Status

Number of tokens currently in the token bucket.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Meter Pointer

Address Space	Slice 0 : 89650 to 89681
	Slice 1 : 187330 to 187361
	Slice 2 : 285010 to 285041
	Slice 3 : 382690 to 382721

Field Description

Bits	Field Name	Description	Default Value
15:0	tokens_0	Number of tokens after the last visit for token bucket 0.	0x0
31:16	tokens_1	Number of tokens after the last visit for token bucket 1.	0x0

35.10.87 Ingress Admission Control Initial Pointer

Initial ingress admission control pointer based on source port number and L2 priority. L2 priority is from either the outermost VLAN PCP field or **defaultPcp**. Further processes may overwrite the initial pointer by comparing the order of the pointer.



Number of Entries :	256
Type of Operation :	Read/Write
Addressing :	address[4:0] : Ingress Port
	address[7:5] : L2 Priority
Address Space	Slice 0 : 5006 to 5261
	Slice 1 : 102686 to 102941
	Slice 2 : 200366 to 200621
	Slice 3 : 298046 to 298301

Field Description

Bits	Field Name	Description	Default Value
0	mmpValid	If set, this entry contains a valid MMP pointer	0x0
5:1	mmpPtr	Initial pointer to the ingress MMP.	0x0
7:6	mmpOrder	Order of the initial ingress MMP pointer.	0x0

35.10.88 Ingress Admission Control Mark All Red

Blocking status of the MMP entry due to packet drops in the MMP.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space	Slice 0 : 89522 to 89553
	Slice 1 : 187202 to 187233
	Slice 2 : 284882 to 284913
	Slice 3 : 382562 to 382593

Field Description

Bits	Field Name	Description	Default Value
0	markAllRed	When this field is set to 1 by the core, the corresponding MMP entry is under the blocking status. As a consequence, all packets with this MMP pointer will be dropped. Clear this field to allow packets enter the MMP entry again.	0x0

35.10.89 Ingress Admission Control Mark All Red Enable

Option to block metering after MMP packet drops.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space	Slice 0 : 89490 to 89521
	Slice 1 : 187170 to 187201
	Slice 2 : 284850 to 284881
	Slice 3 : 382530 to 382561



Field Description

Bits	Field Name	Description	Default Value
0	markAllRedEn	After setting this field to 1, if a packet is dropped by a MMP entry, this MMP entry will stop metering and drop all packets with the corresponding MMP pointer.	0x0

35.10.90 Ingress Admission Control Reset

Reset token buckets so that it is back to the initial status. The reset will be kept high till new traffic arrives, then the traffic is metered with a bucket full of tokens and the reset is deactivated. It is helpful when the token bucket configuration is changed during runtime.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Meter Pointer

Address Space	Slice 0 :	89618 to 89649
	Slice 1 :	187298 to 187329
	Slice 2 :	284978 to 285009
	Slice 3 :	382658 to 382689

Field Description

Bits	Field Name	Description	Default Value
0	bucketReset	if set, reload with full tokens for token buckets in this entry.	0x1

35.10.91 Ingress Admission Control Token Bucket Configuration

Configuration options for token buckets used by Ingress Admission Control. Each entry refers to either a single rate three color marker (srTCM) or a two rate three color marker (trTCM) with two token buckets. For each token bucket the rate is configured by filling in a certain number of tokens at one of the available frequencies. Token bucket 0 shall always use the committed information rate (CIR). Runtime configuration update requires writing 1 to the [Ingress Admission Control Reset](#) first.

Number of Entries : 32
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write
 Addressing : Meter Pointer

Address Space	Slice 0 :	89554 to 89617
	Slice 1 :	187234 to 187297
	Slice 2 :	284914 to 284977
	Slice 3 :	382594 to 382657

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity_0	Capacity for token bucket 0.	0x0
27:16	tokens_0	Number of tokens added each tick for token bucket 0.	0x0
30:28	tick_0	Select one of the 8 available ticks for token bucket 0. The tick frequencies are configured globally in the Core Tick Configuration register.	0x0
46:31	bucketCapacity_1	Capacity for token bucket 1.	0x0
58:47	tokens_1	Number of tokens added each tick for token bucket 1.	0x0
61:59	tick_1	Select one of the 8 available ticks for token bucket 1. The tick frequencies are configured globally in the Core Tick Configuration register.	0x0
62	bucketMode	0 = srTCM 1 = trTCM	0x0
63	colorBlind	0 = color-aware: The metering result is based on the initial coloring from the ingress process pipeline. 1 = color-blind: The metering ignores any pre-coloring.	0x0
66:64	dropMask	Drop mask for the three colors obtained from the metering result. For each bit set to 1 the corresponding color shall drop the packet. Bit 0, 1, 2 represents drop or not for green, yellow and red respectively	0x4
81:67	maxLength	Maximum allowed packet length in bytes. Packets with bytes larger than this value will be dropped before metering.	0x7fff
83:82	tokenMode	0 = Count in bytes and add extra bytes for metering. 1 = Count in bytes and subtract extra bytes for metering. 2 = Count in packets. 3 = No tokens are counted.	0x0
91:84	byteCorrection	Extra bytes per packet for IFG correction, only valid under byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

35.10.92 Ingress Configurable ACL 0 Large Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.



Number of Entries : 2048
 Number of Addresses per Entry : 8
 Type of Operation : Read/Write

Addressing :	address[8:0] : hash of {compareData }
	address[10:9] : bucket number
	Slice 0 : 5262 to 21645
	Slice 1 : 102942 to 119325
	Slice 2 : 200622 to 217005

Address Space	Slice 3 : 298302 to 314685
---------------	----------------------------

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
330:1	compareData	The data which shall be compared in this entry.	0x0
331	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
332	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
333	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
349:334	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
350	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
351	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
352	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
357:353	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
358	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
363:359	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
364	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
365	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0



Bits	Field Name	Description	Default Value
371:366	counter	This is a result field used when this entry is hit. Which counter in Ingress Configurable ACL Match Counter to update.	0x0
372	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
380:373	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0
388:381	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
389	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
390	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
422:391	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
423	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
424	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
440:425	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
441	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
452:442	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
453	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
454	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
456:455	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0

Bits	Field Name	Description	Default Value
457	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
458	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
463:459	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
465:464	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
466	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
469:467	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
470	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0

35.10.93 Ingress Configurable ACL 0 Pre Lookup

The pre ACL lookup allows the user to defined a specific rules for certain packet types in the ACL engine 0. Setting the valid bit and a new rule will override the default rule pointer from the source port table.

Number of Entries : 16

Type of Operation : Read/Write

Addressing :

Address bits [1:0]	Value from preLookupAc1Bits .
Address bits [3:2]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS

Address Space

Slice 0 :	84903 to 84918
Slice 1 :	182583 to 182598
Slice 2 :	280263 to 280278
Slice 3 :	377943 to 377958

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
3:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

35.10.94 Ingress Configurable ACL 0 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 7 fields (bits) which are set to one are selected. It is not allowed to set more than 7 bit in the bitmask. The fields are described in [ACL Fields](#)



Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	ACL rule pointer
Address Space	Slice 0 : 84895 to 84902
	Slice 1 : 182575 to 182582
	Slice 2 : 280255 to 280262
	Slice 3 : 377935 to 377942

Field Description

Bits	Field Name	Description	Default Value
13:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 7 bits should be set.	0x0

35.10.95 Ingress Configurable ACL 0 Search Mask

Before the hashing and searching is done in the [Ingress Configurable ACL 0 Large Table](#) and [Ingress Configurable ACL 0 Small Table](#). The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Separate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Address Space	Slice 0 : 86045
	Slice 1 : 183725
	Slice 2 : 281405
	Slice 3 : 379085

Field Description

Bits	Field Name	Description	Default Value
329:0	mask_small	Which bits to compare in the Ingress Configurable ACL 0 Small Table lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{330} - 1$
659:330	mask_large	Which bits to compare in the Ingress Configurable ACL 0 Large Table lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{330} - 1$

35.10.96 Ingress Configurable ACL 0 Selection

This register selects which result to use when there are multiple hits.



Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 81915
	Slice 1 : 179595
	Slice 2 : 277275
	Slice 3 : 374955

Field Description

Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

35.10.97 Ingress Configurable ACL 0 Small Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 256
 Number of Addresses per Entry : 8
 Type of Operation : Read/Write

Addressing :	address[5:0] : hash of {compareData }
	address[7:6] : bucket number
	Slice 0 : 21646 to 23693
	Slice 1 : 119326 to 121373
	Slice 2 : 217006 to 219053

Address Space

Slice 3 : 314686 to 316733

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
330:1	compareData	The data which shall be compared in this entry.	0x0
331	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
332	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
333	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
349:334	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0



Bits	Field Name	Description	Default Value
350	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
351	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
352	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
357:353	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
358	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
363:359	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
364	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
365	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
371:366	counter	This is a result field used when this entry is hit. Which counter in Ingress Configurable ACL Match Counter to update.	0x0
372	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
380:373	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0
388:381	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
389	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
390	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0



Bits	Field Name	Description	Default Value
422:391	newIpv4Value	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
423	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
424	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
440:425	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
441	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
452:442	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
453	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
454	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
456:455	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
457	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
458	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
463:459	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
465:464	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
466	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
469:467	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
470	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0

35.10.98 Ingress Configurable ACL 0 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.



Number of Entries : 16
 Number of Addresses per Entry : 16
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space

Slice 0 :	85773 to 86028
Slice 1 :	183453 to 183708
Slice 2 :	281133 to 281388
Slice 3 :	378813 to 379068

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
330:1	mask	Which bits to compare in this entry.	$2^{330} - 1$
660:331	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

35.10.99 Ingress Configurable ACL 0 TCAM Answer

This is the table holding the answer for the [Ingress Configurable ACL 0 TCAM](#).

Number of Entries : 16
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : [Ingress Configurable ACL 0 TCAM](#) hit index

Address Space

Slice 0 :	23694 to 23757
Slice 1 :	121374 to 121437
Slice 2 :	219054 to 219117
Slice 3 :	316734 to 316797

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
26:22	destPort	The port which the packet shall be sent to.	0x0



Bits	Field Name	Description	Default Value
27	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
32:28	destInputMirror	Destination physical port for input mirroring.	0x0
33	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
34	updateCounter	When set the selected statistics counter will be updated.	0x0
40:35	counter	Which counter in Ingress Configurable ACL Match Counter to update.	0x0
41	updateTosExp	Force TOS/EXP update.	0x0
49:42	newTosExp	New TOS/EXP value.	0x0
57:50	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
58	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
59	updateSaOrDa	Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
91:60	newIpValue	Update the SA or DA IPv4 address value.	0x0
92	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
93	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
109:94	newL4Value	Update the L4 SP or DP with this value	0x0
110	natOpValid	NAT operation pointer is valid.	0x0
121:111	natOpPtr	NAT operation pointer.	0x0
122	natOpPrio	If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
123	forceColor	If set, the packet shall have a forced color.	0x0
125:124	color	Initial color of the packet if the forceColor field is set.	0x0
126	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
127	mmpValid	If set, this entry contains a valid MMP pointer	0x0
132:128	mmpPtr	Ingress MMP pointer.	0x0



Bits	Field Name	Description	Default Value
134:133	mmpOrder	Ingress MMP pointer order.	0x0
135	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
138:136	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
139	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0

35.10.100 Ingress Configurable ACL 1 Large Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 128

Number of Addresses per Entry : 8

Type of Operation : Read/Write

Addressing :

address[5:0]	: hash of {compareData }
address[6:6]	: bucket number

Address Space

Slice 0 :	23758 to 24781
Slice 1 :	121438 to 122461
Slice 2 :	219118 to 220141
Slice 3 :	316798 to 317821

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	compareData	The data which shall be compared in this entry.	0x0
136	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
137	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
138	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
154:139	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
155	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
156	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
157	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0



Bits	Field Name	Description	Default Value
162:158	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
163	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
168:164	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
169	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
170	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
171	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
177:172	counter	This is a result field used when this entry is hit. Which counter in Ingress Configurable ACL Match Counter to update.	0x0
178	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
186:179	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0
194:187	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
195	updateCfiDei	This is a result field used when this entry is hit. The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
196	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
197	updatePcp	This is a result field used when this entry is hit. The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
200:198	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
201	updateVid	This is a result field used when this entry is hit. The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
213:202	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0



Bits	Field Name	Description	Default Value
214	updateEType	This is a result field used when this entry is hit. The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
216:215	newEthType	This is a result field used when this entry is hit. Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag .	0x0
217	cfiDeiPrio	This is a result field used when this entry is hit. If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
218	pcpPrio	This is a result field used when this entry is hit. If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
219	vidPrio	This is a result field used when this entry is hit. If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
220	ethPrio	This is a result field used when this entry is hit. If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
221	enableUpdateIpf	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
222	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
254:223	newIpfValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
255	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
256	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
272:257	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
273	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0

Bits	Field Name	Description	Default Value
284:274	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
285	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
286	ptp	This is a result field used when this entry is hit. When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
287	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
288	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
292:289	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
293	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
294	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
296:295	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
297	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
298	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
303:299	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
305:304	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
306	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
309:307	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
310	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
311	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing .	0x0

Bits	Field Name	Description	Default Value
323:312	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
324	forceVidPrio	This is a result field used when this entry is hit. If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

35.10.101 Ingress Configurable ACL 1 Pre Lookup

The pre ACL lookup allows the user to defined a specific rules for certain packet types in the ACL engine 1. Setting the valid bit and a new rule will override the default rule pointer from the source port table.

Number of Entries : 512

Type of Operation : Read/Write

Addressing :

Address bits [1:0]	Value from preLookupAcIBits .
Address bits [3:2]	Number of VLANs in incoming Packet.
Address bits [5:4]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS
Address bits [8:6]	L4 Type Of Packet. 0 = Not known. 1 = Is IPv4 or IPv6 but type is not any L4 type in this list. 2 = UDP 3 = TCP 4 = IGMP 5 = ICMP 6 = ICMPv6 7 = MLD

Address Space

Slice 0 :	84383 to 84894
Slice 1 :	182063 to 182574
Slice 2 :	279743 to 280254
Slice 3 :	377423 to 377934

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
3:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

35.10.102 Ingress Configurable ACL 1 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 7 fields (bits) which are set to one are selected. It is not allowed to set more than 7 bit in the bitmask. The fields are described in [ACL Fields](#)



Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	ACL rule pointer
Address Space	Slice 0 : 84375 to 84382
	Slice 1 : 182055 to 182062
	Slice 2 : 279735 to 279742
	Slice 3 : 377415 to 377422

Field Description

Bits	Field Name	Description	Default Value
32:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 7 bits should be set.	0x0

35.10.103 Ingress Configurable ACL 1 Search Mask

Before the hashing and searching is done in the [Ingress Configurable ACL 1 Large Table](#) and [Ingress Configurable ACL 1 Small Table](#). The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Separate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Address Space	Slice 0 : 86309
	Slice 1 : 183989
	Slice 2 : 281669
	Slice 3 : 379349

Field Description

Bits	Field Name	Description	Default Value
134:0	mask_small	Which bits to compare in the Ingress Configurable ACL 1 Small Table lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{135} - 1$
269:135	mask_large	Which bits to compare in the Ingress Configurable ACL 1 Large Table lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{135} - 1$

35.10.104 Ingress Configurable ACL 1 Selection

This register selects which result to use when there are multiple hits.



Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 81916
	Slice 1 : 179596
	Slice 2 : 277276
	Slice 3 : 374956

Field Description

Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

35.10.105 Ingress Configurable ACL 1 Small Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 8
 Number of Addresses per Entry : 8
 Type of Operation : Read/Write

Addressing :

address[1:0] :	hash of {compareData }
address[2:2] :	bucket number

Address Space

Slice 0 :	24782 to 24845
Slice 1 :	122462 to 122525
Slice 2 :	220142 to 220205
Slice 3 :	317822 to 317885

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	compareData	The data which shall be compared in this entry.	0x0
136	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
137	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
138	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
154:139	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0



Bits	Field Name	Description	Default Value
155	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
156	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
157	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
162:158	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
163	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
168:164	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
169	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
170	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
171	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
177:172	counter	This is a result field used when this entry is hit. Which counter in Ingress Configurable ACL Match Counter to update.	0x0
178	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
186:179	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0
194:187	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
195	updateCfiDei	This is a result field used when this entry is hit. The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
196	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
197	updatePcp	This is a result field used when this entry is hit. The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0



Bits	Field Name	Description	Default Value
200:198	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
201	updateVid	This is a result field used when this entry is hit. The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
213:202	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
214	updateEType	This is a result field used when this entry is hit. The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
216:215	newEthType	This is a result field used when this entry is hit. Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag .	0x0
217	cfiDeiPrio	This is a result field used when this entry is hit. If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
218	pcpPrio	This is a result field used when this entry is hit. If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
219	vidPrio	This is a result field used when this entry is hit. If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
220	ethPrio	This is a result field used when this entry is hit. If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
221	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
222	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
254:223	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
255	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0



Bits	Field Name	Description	Default Value
256	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
272:257	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
273	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
284:274	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
285	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
286	ptp	This is a result field used when this entry is hit. When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
287	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
288	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
292:289	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
293	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
294	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
296:295	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
297	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
298	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
303:299	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
305:304	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
306	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0



Bits	Field Name	Description	Default Value
309:307	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
310	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
311	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing .	0x0
323:312	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
324	forceVidPrio	This is a result field used when this entry is hit. If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

35.10.106 Ingress Configurable ACL 1 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries :	8
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
	Slice 0 : 86453 to 86516
	Slice 1 : 184133 to 184196
	Slice 2 : 281813 to 281876
	Slice 3 : 379493 to 379556
Address Space	

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	mask	Which bits to compare in this entry.	$2^{135} - 1$
270:136	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

35.10.107 Ingress Configurable ACL 1 TCAM Answer

This is the table holding the answer for the [Ingress Configurable ACL 1 TCAM](#).



Number of Entries : 8
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : **Ingress Configurable ACL 1 TCAM** hit index

Address Space

Slice 0 :	24846 to 24877
Slice 1 :	122526 to 122557
Slice 2 :	220206 to 220237
Slice 3 :	317886 to 317917

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
26:22	destPort	The port which the packet shall be sent to.	0x0
27	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
32:28	destInputMirror	Destination physical port for input mirroring.	0x0
33	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
34	noLearning	If set this packets MAC SA will not be learned.	0x0
35	updateCounter	When set the selected statistics counter will be updated.	0x0
41:36	counter	Which counter in Ingress Configurable ACL Match Counter to update.	0x0
42	updateTosExp	Force TOS/EXP update.	0x0
50:43	newTosExp	New TOS/EXP value.	0x0
58:51	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
59	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
60	newCfiDeiValue	The value to update to.	0x0



Bits	Field Name	Description	Default Value
61	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
64:62	newPcpValue	The PCP value to update to.	0x0
65	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
77:66	newVidValue	The VID value to update to.	0x0
78	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
80:79	newEthType	Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag .	0x0
81	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
82	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
83	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
84	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
85	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
86	updateSaOrDa	Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
118:87	newIpValue	Update the SA or DA IPv4 address value.	0x0
119	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
120	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
136:121	newL4Value	Update the L4 SP or DP with this value	0x0
137	natOpValid	NAT operation pointer is valid.	0x0
148:138	natOpPtr	NAT operation pointer.	0x0
149	natOpPrio	If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0



Bits	Field Name	Description	Default Value
150	ptp	When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
151	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0
152	tunnelEntryUcMc	Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
156:153	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
157	tunnelEntryPrio	If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
158	forceColor	If set, the packet shall have a forced color.	0x0
160:159	color	Initial color of the packet if the forceColor field is set.	0x0
161	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
162	mmpValid	If set, this entry contains a valid MMP pointer	0x0
167:163	mmpPtr	Ingress MMP pointer.	0x0
169:168	mmpOrder	Ingress MMP pointer order.	0x0
170	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
173:171	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
174	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
175	forceVidValid	Override the Ingress VID, see chapter VLAN Processing .	0x0
187:176	forceVid	The new Ingress VID.	0x0
188	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

35.10.108 Ingress Configurable ACL 2 Pre Lookup

The pre ACL lookup allows the user to defined a specific rules for certain packet types in the ACL engine 2. Setting the valid bit and a new rule will override the default rule pointer from the source port table.

Number of Entries : 64

Type of Operation : Read/Write

Addressing :

Address bits [1:0]	Value from preLookupAcIbBits .
Address bits [3:2]	Number of VLANs in incoming Packet.
Address bits [5:4]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS

Address Space

Slice 0 :	84311 to 84374
Slice 1 :	181991 to 182054
Slice 2 :	279671 to 279734
Slice 3 :	377351 to 377414



Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
2:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

35.10.109 Ingress Configurable ACL 2 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 20 fields (bits) which are set to one are selected. It is not allowed to set more than 20 bit in the bitmask. The fields are described in [ACL Fields](#)

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : ACL rule pointer

Address Space

Slice 0 :	84307 to 84310
Slice 1 :	181987 to 181990
Slice 2 :	279667 to 279670
Slice 3 :	377347 to 377350

Field Description

Bits	Field Name	Description	Default Value
27:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 20 bits should be set.	0x0

35.10.110 Ingress Configurable ACL 2 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries : 24
 Number of Addresses per Entry : 32
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space

Slice 0 :	86517 to 87284
Slice 1 :	184197 to 184964
Slice 2 :	281877 to 282644
Slice 3 :	379557 to 380324

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
540:1	mask	Which bits to compare in this entry.	$2^{540} - 1$
1080:541	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

35.10.111 Ingress Configurable ACL 2 TCAM Answer

This is the table holding the answer for the [Ingress Configurable ACL 2 TCAM](#).

Number of Entries : 24
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : [Ingress Configurable ACL 2 TCAM](#) hit index

Address Space

Slice 0 :	24878 to 24973
Slice 1 :	122558 to 122653
Slice 2 :	220238 to 220333
Slice 3 :	317918 to 318013

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
26:22	destPort	The port which the packet shall be sent to.	0x0
27	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
32:28	destInputMirror	Destination physical port for input mirroring.	0x0
33	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
34	noLearning	If set this packets MAC SA will not be learned.	0x0
35	updateCounter	When set the selected statistics counter will be updated.	0x0



Bits	Field Name	Description	Default Value
41:36	counter	Which counter in Ingress Configurable ACL Match Counter to update.	0x0
42	updateTosExp	Force TOS/EXP update.	0x0
50:43	newTosExp	New TOS/EXP value.	0x0
58:51	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
59	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
60	newCfiDeiValue	The value to update to.	0x0
61	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
64:62	newPcpValue	The PCP value to update to.	0x0
65	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
77:66	newVidValue	The VID value to update to.	0x0
78	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
80:79	newEthType	Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag .	0x0
81	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
82	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
83	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
84	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
85	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
86	updateSaOrDa	Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0



Bits	Field Name	Description	Default Value
118:87	newIpv4Value	Update the SA or DA IPv4 address value.	0x0
119	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
120	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
136:121	newL4Value	Update the L4 SP or DP with this value	0x0
137	natOpValid	NAT operation pointer is valid.	0x0
148:138	natOpPtr	NAT operation pointer.	0x0
149	natOpPrio	If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
150	ptp	When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
151	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0
152	tunnelEntryUcMc	Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
156:153	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
157	tunnelEntryPrio	If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
158	forceColor	If set, the packet shall have a forced color.	0x0
160:159	color	Initial color of the packet if the forceColor field is set.	0x0
161	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
162	mmpValid	If set, this entry contains a valid MMP pointer	0x0
167:163	mmpPtr	Ingress MMP pointer.	0x0
169:168	mmpOrder	Ingress MMP pointer order.	0x0
170	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
173:171	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
174	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
175	forceVidValid	Override the Ingress VID, see chapter VLAN Processing .	0x0
187:176	forceVid	The new Ingress VID.	0x0
188	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

35.10.112 Ingress Configurable ACL 3 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 10 fields (bits) which are set to one are selected. It is not allowed to set more than 10 bit in the bitmask. The fields are described in [ACL Fields](#)

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	ACL rule pointer
Address Space	Slice 0 : 84303 to 84306
	Slice 1 : 181983 to 181986
	Slice 2 : 279663 to 279666
	Slice 3 : 377343 to 377346

Field Description

Bits	Field Name	Description	Default Value
9:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 10 bits should be set.	0x0

35.10.113 Ingress Configurable ACL 3 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries :	16
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space	Slice 0 : 86205 to 86268
	Slice 1 : 183885 to 183948
	Slice 2 : 281565 to 281628
	Slice 3 : 379245 to 379308

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
80:1	mask	Which bits to compare in this entry.	$2^{80} - 1$
160:81	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

35.10.114 Ingress Configurable ACL 3 TCAM Answer

This is the table holding the answer for the [Ingress Configurable ACL 3 TCAM](#).



Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : **Ingress Configurable ACL 3 TCAM** hit index

Address Space	Slice 0 : 24974 to 24989
	Slice 1 : 122654 to 122669
	Slice 2 : 220334 to 220349
	Slice 3 : 318014 to 318029

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
26:22	destPort	The port which the packet shall be sent to.	0x0
27	forceColor	If set, the packet shall have a forced color.	0x0
29:28	color	Initial color of the packet if the forceColor field is set.	0x0
30	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
31	mmpValid	If set, this entry contains a valid MMP pointer	0x0
36:32	mmpPtr	Ingress MMP pointer.	0x0
38:37	mmpOrder	Ingress MMP pointer order.	0x0
39	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
42:40	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
43	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0

35.10.115 Ingress Drop Options

Options to enable or disable learning when the the L2 forwarding process drops the packet.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 87797
	Slice 1 : 185477
	Slice 2 : 283157
	Slice 3 : 380837

Field Description



Bits	Field Name	Description	Default Value
0	learnL2DestDrop	Allow learning when L2 Destination Table drops the packet.	0x0
1	learnL2FloodDrop	Allow learning when the packet is dropped due to unknown DA.	0x0
2	learnL2DestVlanMemberDrop	Allow learning when the packet is dropped due to destination VLAN membership check.	0x1
3	learnL2HairpinDrop	Allow learning when the packet is dropped due to hairpin configurations.	0x0

35.10.116 Ingress Egress Port Packet Type Filter

This sets up which packets are to be dropped or allowed to be transmitted on each of the egress ports. This filtering is done after the source port tables VLAN operation and the VLAN tables VLAN operation. Notice this filter applies to L2 L3 forwarding result only, any other special rules could bypass it (traffic to/from CPU port, classifications, etc). Packets dropped due to this filter will be counted in [Ingress-Egress Packet Filtering Drop](#).

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space	Slice 0 : 82331 to 82362
	Slice 1 : 180011 to 180042
	Slice 2 : 277691 to 277722
	Slice 3 : 375371 to 375402

Field Description

Bits	Field Name	Description	Default Value
0	dropCtaggedVlans	Drop or allow customer VLAN tagged packets on this egress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow C-VLANs. 1 = Drop C-VLANs.	0x0
1	dropSstagedVlans	Drop or allow service VLAN tagged packets on this egress port. Must set moreThanOneVlans when this is used. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow S-VLANs. 1 = Drop S-VLANs.	0x0
2	moreThanOneVlans	When filtering with dropCtaggedVlans or dropSstagedVlans then this field must be set to 1.	0x0
3	dropSingleTaggedVlans	Drop or Allow packets that are VLAN untagged on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0



Bits	Field Name	Description	Default Value
4	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
5	dropIPv4Packets	Drop or allow IPv4 packets on this egress port. 0 = Allow IPv4 packets. 1 = Drop IPv4 packets.	0x0
6	dropIPv6Packets	Drop or allow IPv6 packets on this egress port. 0 = Allow IPv6 packets. 1 = Drop IPv6 packets.	0x0
7	dropMPLSPackets	Drop or allow MPLS packets on this source port. 0 = Allow MPLS packets. 1 = Drop MPLS packets.	0x0
8	dropIPv4MulticastPackets	Drop or allow IPv4 Multicast packets on this egress port. 0 = Allow IPv4 MC packets. 1 = Drop IPv4 MC packets.	0x0
9	dropIPv6MulticastPackets	Drop or allow IPv6 Multicast packets on this egress port. 0 = Allow IPv6 MC packets. 1 = Drop IPv6 MC packets.	0x0
10	dropL2BroadcastFrames	Drop or allow L2 broadcast packets on this egress port. 0 = Allow L2 broadcast packets. 1 = Drop L2 broadcast packets.	0x0
11	dropL2FloodingFrames	Drop or allow L2 flooding packets on this egress port. Observe that this rule takes the unknownL2McFilterRule into account. 0 = Allow L2 flooding packets. 1 = Drop L2 flooding packets.	0x0
12	dropL2MulticastFrames	Drop or allow L2 multicast packets on this egress port. Observe that this L2 multicast bit takes the register L2 Multicast Handling into account to determine if this packet is a L2 multicast packet or not. 0 = Allow L2 multicast packets 1 = Drop L2 multicast packets.	0x0
13	dropDualTaggedVlans	Drop or allow packets with has more than one VLAN tag on this egress port. 0 = Allow packets which has more than one VLAN tag. 1 = Drop packets which has more than one VLAN tag.	0x0
14	dropCStaggedVlans	Drop or allow packets with has a C-VLAN followed by a S-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a C-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a S-VLAN tag.	0x0



Bits	Field Name	Description	Default Value
15	dropSCTaggedVlans	Drop or allow packets with has a S-VLAN followed by a C-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a S-VLAN followed by a C-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a C-VLAN tag.	0x0
16	dropCCTaggedVlans	Drop or allow packets with has a C-VLAN followed by a C-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a C-VLAN tag followed by a C-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a C-VLAN tag.	0x0
17	dropSSTaggedVlans	Drop or allow packets with has a S-VLAN followed by a S-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a S-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a S-VLAN tag.	0x0
18	dropRouted	Drop or allow packets which has been routed on this egress port. 0 = Allow packets which has been routed. 1 = Drop packets which has been routed.	0x0
50:19	srcPortFilter	Each egress port has an optional way of ensuring that a specific source port does not send out a packet on a specific egress port. By setting a bit in this port mask, the packets originating from that source port will be dropped and not be allowed to reach this egress port.	0x0

35.10.117 Ingress Ethernet Type for VLAN tag

When decoding VLAN tags, if the Ethernet Type matches the **typeValue** it will be considered to be a VLAN tag in addition to the standard values of 0x8100 and 0x88A8. The **type** field determines if the VLAN should be regarded as a Service VLAN or Customer VLAN.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81911
Slice 1 :	179591
Slice 2 :	277271
Slice 3 :	374951

Field Description

Bits	Field Name	Description	Default Value
15:0	typeValue	Ethernet Type value.	0xffff



Bits	Field Name	Description	Default Value
16	type	User defined VLAN type. 0 = Customer VLAN. 1 = Service VLAN.	0x0
17	valid	User defined VLAN is valid. 0 = Not Valid. 1 = Valid.	0x0

35.10.118 Ingress MMP Drop Mask

This register provides an option to let ingress MMP not drop packets on certain ports after metering.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81937
Slice 1 :	179617
Slice 2 :	277297
Slice 3 :	374977

Field Description

Bits	Field Name	Description	Default Value
31:0	dropMask	Each bit in this mask refers to if ingress MMP drop is allowed on the corresponding egress port.	$2^{32} - 1$

35.10.119 Ingress Multiple Spanning Tree State

Table of ingress Multiple Spanning Tree Protocol Instances. For routed packets the pointer used to address this table is from the **msptPtr** field in the **Next Hop Packet Modifications** table. For switched packets is from the **msptPtr** field in the **VLAN Table**. Each entry contains the ingress spanning tree states for all ports in this MSTI.

Number of Entries : 16

Type of Operation : Read/Write

Addressing : msptPtr from VLAN Table or Next Hop Packet Modifications Table

Address Space

Slice 0 :	41502 to 41517
Slice 1 :	139182 to 139197
Slice 2 :	236862 to 236877
Slice 3 :	334542 to 334557

Field Description

Bits	Field Name	Description	Default Value
63:0	portSptState	The ingress spanning tree state for this MSTI. Bit[1:0] is the state for port #0, bit[3:2] is the state for port #1, etc. 0 = Forwarding 1 = Discarding 2 = Learning	0x0



35.10.120 Ingress Port Packet Type Filter

This configures which packet types that are to be dropped or allowed on each source port. Each entry corresponds to one ingress port. Packets dropped due to the filter are counted in [Ingress Packet Filtering Drop](#).

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space	Slice 0 : 84919 to 84950
	Slice 1 : 182599 to 182630
	Slice 2 : 280279 to 280310
	Slice 3 : 377959 to 377990

Field Description

Bits	Field Name	Description	Default Value
0	dropCtaggedVlans	Drop or allow customer VLAN tagged packet on this ingress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow C-VLANs. 1 = Drop C-VLANs.	0x0
1	dropStaggedVlans	Drop or allow service VLANs tagged packets on this ingress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow S-VLANs. 1 = Drop S-VLANs.	0x0
2	moreThanOneVlans	When filtering with dropCtaggedVlans or dropStaggedVlans then this field must be set to 1.	0x0
3	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this ingress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
4	dropSingleTaggedVlans	Drop or Allow packets that are VLAN untagged on this ingress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
5	dropIPv4Packets	Drop or allow IPv4 packets on this ingress port. 0 = Allow IPv4 packets. 1 = Drop IPv4 packets.	0x0
6	dropIPv6Packets	Drop or allow IPv6 packets on this ingress port. 0 = Allow IPv6 packets. 1 = Drop IPv6 packets.	0x0
7	dropMPLSPackets	Drop or allow MPLS packets on this ingress port. 0 = Allow MPLS packets. 1 = Drop MPLS packets.	0x0
8	dropIPv4MulticastPackets	Drop or allow IPv4 multicast packets on this ingress port. 0 = Allow IPv4 MC packets. 1 = Drop IPv4 MC packets.	0x0



Bits	Field Name	Description	Default Value
9	dropIPv6MulticastPackets	Drop or allow IPv6 multicast packets on this ingress port. 0 = Allow IPv6 MC packets. 1 = Drop IPv6 MC packets.	0x0
10	dropL2BroadcastFrames	Drop or allow L2 broadcast packets on this ingress port. 0 = Drop L2 broadcast packets. 1 = Allow L2 broadcast packets.	0x0
11	dropL2MulticastFrames	Drop or allow L2 multicast packets on this ingress port. Observe that this L2 multicast bit takes the register L2 Multicast Handling into account to determine if this packet is a L2 multicast packet or not. 0 = Allow L2 multicast packets 1 = Drop L2 multicast packets.	0x0
12	dropDualTaggedVlans	Drop or allow packets which has more than one VLAN tag on this ingress port. 0 = Allow packets which has dual tags. 1 = Drop packets which has dual tags.	0x0
13	dropCStaggedVlans	Drop or allow packets which has a C-VLAN followed by a S-VLAN tagged on this ingress port. 0 = Allow packets which has a C-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a S-VLAN tag.	0x0
14	dropSCTaggedVlans	Drop or allow packets which has a S-VLAN followed by a C-VLAN tagged on this ingress port. 0 = Allow packets which has a S-VLAN followed by a C-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a C-VLAN tag.	0x0
15	dropCCtaggedVlans	Drop or allow packets which has a C-VLAN followed by a C-VLAN tagged on this ingress port. 0 = Allow packets which has a C-VLANs tag followed by a C-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a C-VLAN tag.	0x0
16	dropSStaggedVlans	Drop or allow packets which has a S-VLAN followed by a S-VLAN tagged on this source port. 0 = Allow packets which has a S-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a S-VLAN tag.	0x0

35.10.121 Ingress Router Table

The ingress router table or the Virtual Router Function (VRF), controls which packets are allowed to get access to this router. If a packet is dropped due to the settings of **Ingress Router Table** accept fields then the **Invalid Routing Protocol Drop** will be incremented. Updates for the **Next Hop Hit Status** is also controlled in this table.



Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : **vrf**

Address Space

Slice 0 :	41518 to 41521
Slice 1 :	139198 to 139201
Slice 2 :	236878 to 236881
Slice 3 :	334558 to 334561

Field Description

Bits	Field Name	Description	Default Value
0	acceptIPv4	Accept IPv4 packets. If disabled and an IPv4 packet reaches the router the packet will be dropped and the Invalid Routing Protocol Drop incremented. 0 = Deny 1 = Accept	0x0
1	acceptIPv6	Accept IPv6 packets. If disabled and an IPv6 packet reaches the router the packet will be dropped and the Invalid Routing Protocol Drop incremented. 0 = Deny 1 = Accept	0x0
2	acceptMPLS	Accept MPLS packets. If disabled and an MPLS packet reaches the router the packet will be dropped and the Invalid Routing Protocol Drop incremented. 0 = Deny 1 = Accept	0x0
10:3	minTTL	Minimum TTL. Packets with a TTL below this value will not be accepted. The packet will be dropped and the Expired TTL Drop counter incremented. If the minTtlToCpu is set the packet will be sent to CPU instead of being dropped. The TTL check is done for IPv4, IPv6 and MPLS routed packets.	0x0
11	minTtlToCpu	If this is set then packets below minimum TTL will be send to CPU instead of dropped.	0x0
12	ipv4HitUpdates	Enable updates of the Next Hop Hit Status for routed IPv4 packets. 0 = Disable 1 = Enable	0x0
13	ipv6HitUpdates	Enable updates of the Next Hop Hit Status for routed IPv6 packets. 0 = Disable 1 = Enable	0x0
14	mplsHitUpdates	Enable updates of the Next Hop Hit Status for routed MPLS packets. 0 = Disable 1 = Enable	0x0
15	ecmpUseIpDa	Use IP destination address as part of ECMP hash key.	0x1
16	ecmpUseIpSa	Use IP source address as part of ECMP hash key.	0x1
17	ecmpUseIpTos	Use IP TOS/Traffic Class as part of ECMP hash key.	0x0
18	ecmpUseIpProto	Use IP Protocol/Next Header as part of ECMP hash key.	0x1
19	ecmpUseIpL4Sp	Use TCP/UDP source port as part of ECMP hash key.	0x1



Bits	Field Name	Description	Default Value
20	ecmpUselpL4Dp	Use TCP/UDP destination port as part of ECMP hash key.	0x1
21	mmpValid	If set, this entry contains a valid MMP pointer. Only valid when packets get routed	0x0
26:22	mmpPtr	Ingress MMP pointer.	0x0
28:27	mmpOrder	Ingress MMP pointer order.	0x0
29	sendToCpuOrDrop	When a check if the packet protocols are allowed on this Ingress Router Table shall the packets be dropped or sent-to-CPU? 0 = Dropped. 1 = Sent-To-CPU	0x0

35.10.122 Ingress VID Ethernet Type Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : [Ingress VID Ethernet Type Range Search Data](#) hit index

Address Space	Slice 0 : 84275 to 84278
	Slice 1 : 181955 to 181958
	Slice 2 : 279635 to 279638
	Slice 3 : 377315 to 377318

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

35.10.123 Ingress VID Ethernet Type Range Search Data

This Ethernet type range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the [Ingress VID Ethernet Type Range Assignment Answer](#) table.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space	Slice 0 : 84279 to 84282
	Slice 1 : 181959 to 181962
	Slice 2 : 279639 to 279642
	Slice 3 : 377319 to 377322

Field Description



Bits	Field Name	Description	Default Value
31:0	ports	Ports that this range search is activated on.	0x0
47:32	start	Start of Ethernet type range.	0x0
63:48	end	End of Ethernet type range.	0x0

35.10.124 Ingress VID Inner VID Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : [Ingress VID Inner VID Range Search Data](#) hit index

Address Space	Slice 0 : 84283 to 84286
	Slice 1 : 181963 to 181966
	Slice 2 : 279643 to 279646
	Slice 3 : 377323 to 377326

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

35.10.125 Ingress VID Inner VID Range Search Data

If a packet has an inner VLAN tag, this inner VID range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the [Ingress VID Inner VID Range Assignment Answer](#) table.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space	Slice 0 : 84287 to 84290
	Slice 1 : 181967 to 181970
	Slice 2 : 279647 to 279650
	Slice 3 : 377327 to 377330

Field Description

Bits	Field Name	Description	Default Value
31:0	ports	Ports that this range search is activated on.	0x0
32	vtype	Shall this entry match S-Type or C-Type VLAN. 0 = C-Type 1 = S-Type	0x0
44:33	start	Start of VID range.	0x0



Bits	Field Name	Description	Default Value
56:45	end	End of VID range.	0x0

35.10.126 Ingress VID MAC Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : [Ingress VID MAC Range Search Data](#) hit index

Address Space

Slice 0 :	84299 to 84302
Slice 1 :	181979 to 181982
Slice 2 :	279659 to 279662
Slice 3 :	377339 to 377342

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

35.10.127 Ingress VID MAC Range Search Data

This MAC address range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the [Ingress VID MAC Range Assignment Answer](#) table.

Number of Entries : 4
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space

Slice 0 :	86189 to 86204
Slice 1 :	183869 to 183884
Slice 2 :	281549 to 281564
Slice 3 :	379229 to 379244

Field Description

Bits	Field Name	Description	Default Value
31:0	ports	Ports that this range search is activated on.	0x0
32	saOrDa	Is this rule for source or destination MAC address. 0 = Source MAC 1 = Destination MAC	0x0
80:33	start	Start of MAC address range.	0x0
128:81	end	End of MAC address range.	0x0



35.10.128 Ingress VID Outer VID Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	Ingress VID Outer VID Range Search Data hit index
Address Space	Slice 0 : 84291 to 84294
	Slice 1 : 181971 to 181974
	Slice 2 : 279651 to 279654
	Slice 3 : 377331 to 377334

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

35.10.129 Ingress VID Outer VID Range Search Data

If a packet has an outer VLAN tag, this outer VID range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the [Ingress VID Outer VID Range Assignment Answer](#) table.

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space	Slice 0 : 84295 to 84298
	Slice 1 : 181975 to 181978
	Slice 2 : 279655 to 279658
	Slice 3 : 377335 to 377338

Field Description

Bits	Field Name	Description	Default Value
31:0	ports	Ports that this range search is activated on.	0x0
32	vtype	Shall this entry match S-Type or C-Type VLAN. 0 = C-Type 1 = S-Type	0x0
44:33	start	Start of VID range.	0x0
56:45	end	End of VID range.	0x0

35.10.130 L2 Action Table

The L2 action table can be used to limit what type of traffic shall be able to enter a port depending on which port its coming from and going to. There are three table results which can be taken into consideration, the I2 destination MAC lookup, the I2 source MAC lookup and finally the ingress ACL lookup. The [L2 Action Table Egress Port State](#) defines the highest bit in the address. This table is looked up for each



of the destination ports which the packet is going to. If a packet is dropped then it is recorded in the drop counter **L2 Action Table Drop**.

Number of Entries : 128
 Type of Operation : Read/Write

Addressing :

Address Bit 0:	Source Port State Bit from Source Port Table field I2ActionTablePortState .
Address Bit 1:	L2 SA Table was a hit. 0 = Miss. 1 = Hit.
Address Bit 2:	L2 SA Table - L2 Action Table Status bit. If this table was a miss then this bit will be zero.
Address Bit 3:	L2 DA Table - L2 Action Table Status bit. If this table was a miss then this bit will be zero.
Address Bit [5:4]:	L2 Packet Type. 0 = L2 Dest Table was a Unicast. 1 = L2 Dest Table was Multicast. 2 = L2 DA table was a miss and packet is being flooded. 3 = Packet was a Broadcast packet and L2 Dest Table did not hit. If both flooded and L2 Broadcast packet then this option will be selected.
Address Bit 6:	Destination Port State Bit comes from the L2 Action Table Egress Port State .

Address Space

Slice 0 :	76418 to 76545
Slice 1 :	174098 to 174225
Slice 2 :	271778 to 271905
Slice 3 :	369458 to 369585

Field Description

Bits	Field Name	Description	Default Value
0	noLearningUc	The packet shall not be learned. This is applied to L2 DA MAC unicast packets.	0x0
1	noLearningMc	If the packet is a L2 Multicast then the packet shall not be learned. If a packet is a L2 Multicast depends on if the SA MAC MC bit is set.	0x0
2	dropAll	The packet shall drop all instances and update counter L2 Action Table Drop . However special packets which are allowed will still be allowed into the switch (using the field useSpecialAllow set to one and register Allow Special Frame Check For L2 Action Table)	0x0
3	drop	The packet shall only drop on the ports which hits this action.	0x0
4	dropPortMove	The packet shall be dropped if the result from the learning lookup is port-move.	0x0
5	sendToCpu	The packet shall be send to the CPU.	0x0
6	forceSendToCpuOrigPkt	Force the packet to the CPU to be the original,unmodified, packet. The exception to this is rule is the tunnel exit which will still be carried out.	0x0
7	noPortMove	No port move is allowed for this packet.	0x0



Bits	Field Name	Description	Default Value
8	useSpecialAllow	Use the special frame checks on this port. 0 = No. 1 = Yes.	0x0
10:9	allowPtr	Pointer to allow special packets defined in Allow Special Frame Check For L2 Action Table .	0x0
11	mmpValid	If set, this entry contains a valid MMP pointer	0x0
16:12	mmpPtr	Ingress MMP pointer.	0x0
18:17	mmpOrder	Ingress MMP pointer order.	0x0

35.10.131 L2 Action Table Egress Port State

The egress port state for the L2 Action Table Lookup.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81932
Slice 1 :	179612
Slice 2 :	277292
Slice 3 :	374972

Field Description

Bits	Field Name	Description	Default Value
31:0	state	What is the egress port status bits in the L2 Action Table for the egress port. Bit [0] are used for port 0, Bits [1] are used for port 1 and so on.	0x0

35.10.132 L2 Action Table Source Port

The L2 action table for source port is looked up at the same time as the [L2 Action Table](#) and its result is merged with the lookup from the [L2 Action Table](#) table, this lookup is active when enabled in the [Source Port Table](#) field [enableL2ActionTable](#) is set to one. The [L2 Action Table](#) is enabled for each of the destination ports the packet is going to, this table is looked up based on the source port and even if the packet is going to no destination ports this lookup is still carried out. Another difference between [L2 Action Table](#) and this table is that the highest address bit (bit 6) which uses the status from the L2 SA Lookup and if the packet is going to do a port move then this address bit is high.



Number of Entries : 128
 Type of Operation : Read/Write

Addressing :	Address Bit 0:	Source Port State Bit from Source Port Table field I2ActionTablePortState .
	Address Bit 1:	L2 SA Table was a hit. 0 = Miss. 1 = Hit.
	Address Bit 2:	L2 SA Table - L2 Action Table Status bit.
	Address Bit 3:	L2 DA Table - L2 Action Table Status bit. If this table was a miss then this bit will be zero.
	Address Bit [5:4]:	L2 Packet Type. 0 = L2 Dest Table was a Unicast. 1 = L2 Dest Table was Multicast. 2 = L2 DA table was a miss and packet is being flooded. 3 = Packet was a Broadcast packet and L2 Dest Table did not hit. If both flooded and L2 Broadcast packet then this option will be selected.
	Address Bit [6]:	Port Move. Result bit from L2 SA lookup if the packet shall do a port move or not.
Address Space	Slice 0 :	76546 to 76673
	Slice 1 :	174226 to 174353
	Slice 2 :	271906 to 272033
	Slice 3 :	369586 to 369713

Field Description

Bits	Field Name	Description	Default Value
0	noLearningUc	The packet shall not be learned. This is applied to L2 DA MAC unicast packets.	0x0
1	noLearningMc	If the packet is a L2 Multicast then the packet shall not be learned. If a packet is a L2 Multicast depends on if the SA MAC MC bit is set.	0x0
2	dropAll	The packet shall drop all instances and update counter L2 Action Table Drop . However special packets which are allowed will still be allowed into the switch (using the field useSpecialAllow set to one and register Allow Special Frame Check For L2 Action Table)	0x0
3	drop	The packet shall only drop on the ports which hits this action.	0x0
4	dropPortMove	The packet shall be dropped if the result from the learning lookup is port-move.	0x0
5	sendToCpu	The packet shall be send to the CPU.	0x0
6	forceSendToCpuOrigPkt	Force the packet to the CPU to be the original,unmodified, packet. The exception to this is rule is the tunnel exit which will still be carried out.	0x0
7	noPortMove	No port move is allowed for this packet.	0x0
8	useSpecialAllow	Use the special frame checks on this port. 0 = No. 1 = Yes.	0x0
10:9	allowPtr	Pointer to allow special packets defined in Allow Special Frame Check For L2 Action Table .	0x0
11	mmpValid	If set, this entry contains a valid MMP pointer	0x0



Bits	Field Name	Description	Default Value
16:12	mmpPtr	Ingress MMP pointer.	0x0
18:17	mmpOrder	Ingress MMP pointer order.	0x0

35.10.133 L2 Aging Collision Shadow Table

This table traces the **valid** field of the **L2 Aging Collision Table** and is used by L2 forwarding to check if a hit in the **L2 Lookup Collision Table** is valid. Any software write to this table shall be updated to the **valid** field of the **L2 Aging Collision Table**.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : **L2 Lookup Collision Table** hit index

Address Space	Slice 0 : 83167 to 83198
	Slice 1 : 180847 to 180878
	Slice 2 : 278527 to 278558
	Slice 3 : 376207 to 376238

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding L2 Lookup Collision Table entry is valid.	0x0

35.10.134 L2 Aging Collision Table

This table holds the status of the entries in the **L2 Lookup Collision Table**. Any software write to the **valid** field in this table shall be done in the **L2 Aging Collision Shadow Table**.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : **L2 Lookup Collision Table** hit index
 Address Space : 638 to 669

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding L2 Lookup Collision Table entry is valid.	0x0
1	stat	If this is set, then the corresponding L2 Lookup Collision Table entry will not be aged out.	0x0
2	hit	If this is set, then the corresponding L2 Lookup Collision Table entry has a L2 SA/DA search hit since the last aging scan.	0x0



35.10.135 L2 Aging Status Shadow Table

This table traces the **valid** field of the **L2 Aging Table** and is used by L2 forwarding to check if a hit in the **L2 DA Hash Lookup Table** is valid. Any software write to this table shall be updated to the **valid** field of the **L2 Aging Table**. Any software write to this table shall be copied to the **L2 Aging Status Shadow Table - Replica**

Number of Entries :	4096
Type of Operation :	Read/Write
Addressing :	address[0:9] : hash of {GID, destination MAC}
	address[10:11] : bucket number
Address Space	Slice 0 : 51778 to 55873
	Slice 1 : 149458 to 153553
	Slice 2 : 247138 to 251233
	Slice 3 : 344818 to 348913

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding hash table entry is valid.	0x0

35.10.136 L2 Aging Status Shadow Table - Replica

This table traces the **valid** field of the **L2 Aging Table** and is used by L2 forwarding to check if a hit in the **L2 SA Hash Lookup Table** is valid. Content of this table shall be identical as the **L2 Aging Status Shadow Table**.

Number of Entries :	4096
Type of Operation :	Read/Write
Addressing :	address[0:9] : hash of {GID, source MAC}
	address[10:11] : bucket number
Address Space	Slice 0 : 68194 to 72289
	Slice 1 : 165874 to 169969
	Slice 2 : 263554 to 267649
	Slice 3 : 361234 to 365329

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding hash table entry is valid.	0x0

35.10.137 L2 Aging Table

This table uses the same addressing as the **L2 DA Hash Lookup Table** to show the status of each entries in that table. Any software write to any valid field in this table shall be done in the **L2 Aging Status Shadow Table**. Any software write to this table shall be copied to the **L2 Aging Status Shadow Table - Replica**



Number of Entries :	4096
Type of Operation :	Read/Write
Addressing :	address[0:9] : hash of {GID, destination MAC}
	address[10:11] : bucket number
Address Space :	702 to 4797

Field Description

Bits	Field Name	Description	Default Value
0	valid	If set, then the corresponding hash table entry is valid.	0x0
1	stat	If set, then the corresponding hash table entry will not be aged out.	0x0
2	hit	If set, then the corresponding hash table entry has a L2 DA search hit since the last aging scan.	0x0

35.10.138 L2 DA Hash Lookup Table

The L2 table is used for hash search based on the destination MAC address and a GID from the [VLAN Table](#). When performing a L2 destination port lookup, {GID, destination MAC} is used as key for a hash calculation (see Section [MAC Table Hashing](#)). The hash is then used as index into this table to read out the 4 buckets. The incoming {GID, destination MAC} are compared to all the buckets. If any of the buckets match then address was known. The result of the lookup will be read from the [L2 Destination Table](#) at the same address as the matching hash index and bucket. Any software write to this table shall be copied to the [L2 SA Hash Lookup Table](#).

Number of Entries :	4096
Type of Operation :	Read/Write
Addressing :	address[0:9] : hash of {GID, destination MAC}
	address[10:11] : bucket number
Address Space	Slice 0 : 55874 to 59969
	Slice 1 : 153554 to 157649
	Slice 2 : 251234 to 255329
	Slice 3 : 348914 to 353009

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address.	0x0
59:48	gid	Global identifier from the VLAN Table.	0x0

35.10.139 L2 Destination Table

This table contains either a destination port or a pointer to the L2 multicast table. Any software write to this table shall be copied to the [L2 Destination Table - Replica](#).



Number of Entries :	4128
Type of Operation :	Read/Write
Addressing :	address 0 to 4095 L2 DA Hash Lookup Table address
	address 4096 to L2 Lookup Collision Table address 4127 :
Address Space	Slice 0 : 59970 to 64097
	Slice 1 : 157650 to 161777
	Slice 2 : 255330 to 259457
	Slice 3 : 353010 to 357137

Field Description

Bits	Field Name	Description	Default Value
0	uc	Unicast if set; multicast if cleared. Multicast means that a lookup to the L2 Multicast Table will occur and determine a list of destination ports.	0x0
6:1	destPort_or_mcAddr	Destination port number or pointer into the L2 Multicast Table .	0x0
7	pktDrop	If set, the packet will be dropped and the L2 Lookup Drop incremented.	0x0
8	l2ActionTableDaStatus	The status DA bit to be used in the addressing for the table L2 Action Table Lookup.	0x0
9	l2ActionTableSaStatus	The status SA bit to be used in the addressing for the table L2 Action Table Lookup.	0x0
25:10	metaData	Meta data for to CPU tag.	0x0

35.10.140 L2 Destination Table - Replica

This table is replicated from the **L2 Destination Table** and used by the learning engine allowing the learning engine and packet forwarding to process in parallel. Content of this table shall be identical as the **L2 Destination Table**.

Number of Entries :	4128
Type of Operation :	Read/Write
Addressing :	address 0 to 4095 L2 SA Hash Lookup Table address
	address 4096 to L2 Lookup Collision Table address 4127 :
Address Space	Slice 0 : 72290 to 76417
	Slice 1 : 169970 to 174097
	Slice 2 : 267650 to 271777
	Slice 3 : 365330 to 369457

Field Description



Bits	Field Name	Description	Default Value
0	uc	Unicast if set; multicast if cleared. Multicast means that a lookup to the L2 Multicast Table will occur and determine a list of destination ports.	0x0
6:1	destPort_or_mcAddr	Destination port number or pointer into the L2 Multicast Table .	0x0
7	pktDrop	If set, the packet will be dropped and the L2 Lookup Drop incremented.	0x0
8	l2ActionTableDaStatus	The status DA bit to be used in the addressing for the table L2 Action Table Lookup.	0x0
9	l2ActionTableSaStatus	The status SA bit to be used in the addressing for the table L2 Action Table Lookup.	0x0
25:10	metaData	Meta data for to CPU tag.	0x0

35.10.141 L2 Lookup Collision Table

Collision table for the **L2 DA Hash Lookup Table**. If there is a hash collision and all the buckets for that hash index are occupied then additional entries can be stored in the collision table. When searching this table, all entries are compared in parallel and the matching entry with the lowest address will be used as a match result. Chapter [Learning and Aging](#) describes how to search and write to this table.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space	Slice 0 : 83135 to 83166
	Slice 1 : 180815 to 180846
	Slice 2 : 278495 to 278526
	Slice 3 : 376175 to 376206

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address	0x0
59:48	gid	Global identifier for learning	0x0

35.10.142 L2 Lookup Collision Table Masks

Masks for collision memory for the MAC address and the global identifier. Only the first 4 entries has masks on them.

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space	Slice 0 : 83131 to 83134
	Slice 1 : 180811 to 180814
	Slice 2 : 278491 to 278494
	Slice 3 : 376171 to 376174



Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address mask	$2^{48} - 1$
59:48	gid	Global identifier for learning mask	0xfff

35.10.143 L2 Multicast Handling

Exceptions for L2 multicast flag handling, only valid for the Multicast Broadcast Storm Control and the Ingress Egress Port Packet Type Filter. The switch sets by default a L2 multicast flag when DA is an Ethernet multicast address (i.e. DA with the least-significant bit of the first octet equals 1 (e.g. 01:80:c2:00:00:00) but not equal to ff:ff:ff:ff:ff:ff).

Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 81933
	Slice 1 : 179613
	Slice 2 : 277293
	Slice 3 : 374973

Field Description

Bits	Field Name	Description	Default Value
0	exclIPv4Mc	If set, IPv4 packets with IPv4 multicast MAC address will NOT have a L2 multicast flag.	0x0
1	exclIPv6Mc	If set, IPv6 packets with IPv6 multicast MAC address will NOT have a L2 multicast flag.	0x0
2	inclL2McLut	If set, packets that are forwarded by L2 Multicast Table will internally be treated as the L2 multicast bit in the L2 DA address would have been set to one.	0x1
3	inclMultiPorts	If set, packets that end up in more than one destination port but not due to broadcast or flooding will have a L2 multicast flag. Observe that mirroring is not a valid multiport destination.	0x0
4	unknownL2McFilterRule	Select the filtering rules for unknown L2 multicast MAC DA in the Ingress Egress Port Packet Type Filter . 0 = dropL2FloodingFrames 1 = dropL2MulticastFrames	0x0

35.10.144 L2 Multicast Table

L2 multicast table.

Number of Entries : 64
 Type of Operation : Read/Write
 Addressing : mcAddr field from **L2 Destination Table** or from **Next Hop Table**

Address Space	Slice 0 : 83067 to 83130
	Slice 1 : 180747 to 180810
	Slice 2 : 278427 to 278490
	Slice 3 : 376107 to 376170



Field Description

Bits	Field Name	Description	Default Value
31:0	mcPortMask	L2 portmask entry members. If set, the port is part of multicast group and shall be transmitted to.	$2^{32} - 1$
47:32	metaData	Meta data for to CPU tag.	0x0

35.10.145 L2 Reserved Multicast Address Action

If the higher bits of the incoming packets MAC DA address matches the [L2 Reserved Multicast Address Base](#) then the lower bits are used as index into this table. The action can be to drop the packet, send the packet to the CPU or just process the packet in the normal L2 pipeline.

Number of Entries : 256
 Type of Operation : Read/Write
 Addressing : MAC DA[7:0]

Address Space

Slice 0 :	84987 to 85242
Slice 1 :	182667 to 182922
Slice 2 :	280347 to 280602
Slice 3 :	378027 to 378282

Field Description

Bits	Field Name	Description	Default Value
31:0	dropMask	Determines which source ports that are not allowed to receive this multicast address. Each bit set to 1 will result in dropping this multicast address on that source port. Bit 0 is port 0, bit 1 is port 1 etc. Each drop will be counted in L2 Reserved Multicast Address Drop .	0x0
63:32	sendToCpuMask	Received packets on these source ports will be sent to the CPU. Bit 0 represents port 0, bit 1 represents port 1 etc. LLDP frames sent to the CPU takes priority over this.	0x0

35.10.146 L2 Reserved Multicast Address Base

Certain L2 Destination MAC addresses shall be treated special when entering the switch. If the first 40 bits of the Destination MAC address matches the macBase field then the lowest 8 bits are used as index into the [L2 Reserved Multicast Address Action](#) table.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space

Slice 0 :	81912
Slice 1 :	179592
Slice 2 :	277272
Slice 3 :	374952



Field Description

Bits	Field Name	Description	Default Value
39:0	macBase	The first 40 bits of the reserved MAC address, and the lower 16 bits of it can be masked. The default is 01:80:c2:00:00	0x180c20000
55:40	mask	Bit comparison mask for the lower 2 bytes in macBase (marked with XX as in 01:80:c2:XX:XX). If a bit is set in the mask then the corresponding bit will be compared. Otherwise the bits are dont care.	0xffff

35.10.147 L2 SA Hash Lookup Table

L2 table used for hash search based on the source MAC and a GID from the [VLAN Table](#). When performing a SA MAC learning check {GID, Source MAC} is used as key for a hash function (see Section [MAC Table Hashing](#)) which calculates a hash index. The hash index points to this table that has 4 buckets. The incoming {GID, source MAC} are compared to all the 4 buckets. If any of the buckets match then address was known. The result of the lookup will be read from the [L2 Destination Table - Replica](#) at the same address as the matching hash index and bucket. Content of this table shall be identical as the [L2 DA Hash Lookup Table](#).

Number of Entries : 4096

Type of Operation : Read/Write

Addressing :	address[0:9] :	hash of {GID, source MAC}
	address[10:11] :	bucket number
Address Space	Slice 0 :	64098 to 68193
	Slice 1 :	161778 to 165873
	Slice 2 :	259458 to 263553
	Slice 3 :	357138 to 361233

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address.	0x0
59:48	gid	Global identifier from the VLAN Table.	0x0

35.10.148 L2 Tunnel Decoder Setup

The tunnel TPID setup is setup in this register. This is used by the tunnel packet decoder. Besides the configurable values the default Ethernet Type values of 0x8100 is detected as a C-type VLAN ID while 0x9100, 0x9200 and 0x88A8 is discoverable as S-type VLAN IDs.

Number of Entries : 1

Type of Operation : Read/Write

Address Space	Slice 0 :	81908
	Slice 1 :	179588
	Slice 2 :	277268
	Slice 3 :	374948



Field Description

Bits	Field Name	Description	Default Value
0	defaultEthCTypeValid	The configurable Ethernet Type C-type is valid.	0x0
16:1	defaultEthCType	A configurable Ethernet Type which shall be used to determine a C-Type VLAN.	0x0
17	defaultEthSTypeValid	The configurable Ethernet Type S-type is valid.	0x0
33:18	defaultEthSType	A configurable Ethernet Type which shall be used to determine a S-Type VLAN.	0x0

35.10.149 L3 LPM Result

This is the longest prefix routing table result. The index into the table is the hit index from the [L3 Routing TCAM](#).

Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : [L3 Routing TCAM](#) hit index

Address Space

Slice 0 :	41522 to 41537
Slice 1 :	139202 to 139217
Slice 2 :	236882 to 236897
Slice 3 :	334562 to 334577

Field Description

Bits	Field Name	Description	Default Value
0	useECMP	Enables the use of ECMP hash to calculate the next hop pointer. 0 = Use ECMP hash. 1 = Do not use ECMP hash.	0x0
6:1	ecmpMask	How many bits of the ECMP hash will be used when calculating the ECMP offset. This byte is AND:ed with the ECMP hash to determine which bits shall be used as offset.	0x0
9:7	ecmpShift	How many bits the masked ECMP hash will be right shifted.	0x0
19:10	nextHopPointer	Index into the Next Hop Table for this destination.	0x0

35.10.150 L3 Routing Default

The default router to be used if the destination lookup in L3 tables fails, i.e does not match either the LPM or the hash tables.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : [vrf](#)

Address Space

Slice 0 :	84271 to 84274
Slice 1 :	181951 to 181954
Slice 2 :	279631 to 279634
Slice 3 :	377311 to 377314



Field Description

Bits	Field Name	Description	Default Value
9:0	nextHop	The default next hop to be used. Index into the Next Hop Table .	0x0
10	pktDrop	If set the packet will be drop and the L3 Lookup Drop counter incremented.	0x0
11	sendToCpu	If set then the packet will be sent to the CPU.	0x0
12	mmpValid	If set, this entry contains a valid MMP pointer	0x0
17:13	mmpPtr	Ingress MMP pointer.	0x0
19:18	mmpOrder	Ingress MMP pointer order.	0x0

35.10.151 L3 Routing TCAM

This is the longest prefix match routing table used to determine the next hop. This table is compared from the highest address and downwards. The match which has the highest entry number is selected. The selected entry number is used to index the [L3 LPM Result](#) table to provide the next hop result. For each entry the mask determines which bits that shall be compared. An entry contains three parts: valid flag, comparison fields and field masks. Each comparison field is associated with a mask to optionally ignore several bits or even the entire field during comparison. To allow any value on a certain bit, the corresponding bit in the mask shall be set to 1. As a consequence, the field will have that bit nailed to 0 if read and ignored during lookup. Hit in multiple entries will return the first hit index (lowest address/index) to lookup in the result table.

Number of Entries : 16
 Number of Addresses per Entry : 8
 Type of Operation : Read/Write
 Addressing : Entry number

Address Space

Slice 0 :	87798 to 87925
Slice 1 :	185478 to 185605
Slice 2 :	283158 to 283285
Slice 3 :	380838 to 380965

Field Description

Bits	Field Name	Description	Default Value
1:0	proto	Select if this is an IPv4, IPv6 or MPLS entry. 0 = Reserved 1 = MPLS Entry. 2 = IPv4 entry. 3 = IPv6 entry. protoMaskN determines the bits in the field that can be ignored for comparison.	0x0
3:2	vrf	This entries VRF. The packets assigned VRF will be compared with this field. vrfMaskN determines the bits in the field that can be ignored for comparison.	0x0



Bits	Field Name	Description	Default Value
131:4	destIPAddr	The IP or MPLS address to be matched. If the entry is an IPv4 entry then bits [31:0] should be set to the IPv4 address. If the entry is an MPLS entry then bits [5-1:0] should contain the source port while bits [5+19:5] should contain the MPLS label. destIPAddrMaskN determines the bits in the field that can be ignored for comparison.	0x0
133:132	protoMaskN	Mask for the proto field. For each bit in the mask, 0 means the bit is valid for comparison, 1 means the comparison ignores this bit.	0x0
135:134	vrfMaskN	Mask for the vrf field. For each bit in the mask, 0 means the bit is valid for comparison, 1 means the comparison ignores this bit.	0x0
263:136	destIPAddrMaskN	Mask for the destIPAddr field. For each bit in the mask, 0 means the bit is valid for comparison, 1 means the comparison ignores this bit.	0x0
264	valid	If set, this entry is valid	0x0

35.10.152 LACP Packet Decoder Options

This is the MAC address used to determine that a packet is a LACP packet. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
	Slice 0 : 85499
	Slice 1 : 183179
	Slice 2 : 280859
	Slice 3 : 378539

Address Space

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
48:1	mac	The value to be used to find this packet type.	0x180c2000002
80:49	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
112:81	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0



35.10.153 LLDP Configuration

A LLDP packet is identified as a LLDP frame if the packets MAC DA matches one of the mac1-mac3 fields and the packets EtherType matches eth. The portmask field determines if an identified LLDP packet will bypass the normal packet processing and instead be sent to the CPU or if the packet should pass through normal packet processing.

Number of Entries :	1
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Address Space	Slice 0 : 86305
	Slice 1 : 183985
	Slice 2 : 281665
	Slice 3 : 379345

Field Description

Bits	Field Name	Description	Default Value
47:0	mac1	DA MAC address to match for LLDP packet.	0x180c20000e
95:48	mac2	DA MAC address to match for LLDP packet.	0x180c200003
143:96	mac3	DA MAC address to match for LLDP packet.	0x180c200000
159:144	eth	The Ethernet Type for a LLDP	0x88cc
160	bpduOption	If both LLDP and BPDU are valid, because the BPDU has same MAC address as LLDP, then this option allows the BPDU identification to be turned off 0 = Don't do anything. Both LLDP and BPDU can be valid at same time. 1 = Remove BPDU valid causing that the packet will only be seen as a LLDP packet and not a BPDU frame and the new frame will not be sent to the CPU because the switch will no longer consider it a BPDU frame, this includes Rapid Spanning Tree BPDUs also.	0x0
192:161	portmask	One bit per source port, bit 0 for port 0, bit 1 for port 1 etc. 0 = Do not sent a matched LLDP packet to the CPU from this port. Packet will pass through normal packet processing. 1 = Send a matched LLDP packet to CPU from this source port and hence bypassing normal processing.	$2^{31} - 1$

35.10.154 Learning And Aging Enable

Enable/Disable the learning and aging function. If software needs to take fully control over learning and aging tables by writing to the [FIB](#) directly, the learning and aging units should be completely turned off, which means all fields in this register have to be cleared to 0, partly reset is not allowed. When the learning and aging units are turned on, software still has controllablity over learning and aging by injecting user defined learning packets.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	599

Field Description



Bits	Field Name	Description	Default Value
0	learningEnable	If set, the learning unit will be activated.	0x1
1	agingEnable	If set, the aging unit will be activated.	0x1
2	daHitEnable	If set, MAC DA hit in the forwarding information base will update the hit bit for non-static entries.	0x1
3	lru	If set, the learning unit will try to overwrite a least recently used non-static entry in either the hash table or the collision table when there is no free entry to use. Otherwise the learning unit will try to overwrite a non-static entry in the collision table.	0x0

35.10.155 Learning And Aging Writeback Control

Determine how the hardware learning and aging engine act on injected learning packets. By default all the hardware and software learning/aging/hit results can be updated to the [FIB](#). If software needs more controllability, the learning/aging/hit decisions from hardware can be configured to only send to corresponding writeback FIFOs but not write to the [FIB](#).

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 601

Field Description

Bits	Field Name	Description	Default Value
0	hwLearningWriteBack	If set, the hardware learning result from unknown or port moved source MAC will be pushed to the Learning Data FIFO and written to the FIB simultaneously. Otherwise the result is only pushed to the FIFO.	0x1
1	hwAgingWriteBack	If set, the aging result will be pushed to the Aging Data FIFO and written to the FIB simultaneously. Otherwise the result is only pushed to the FIFO and software has to read it out then send in a corresponding learning packet if this aging result should be written to the tables.	0x1
2	hwHitWriteBack	If set, the hit update of a learned destination MAC will be pushed to Hit Update Data FIFO and written to the FIB simultaneously. Otherwise the result is only pushed to the FIFO and then software decides the FIB writes.	0x1
3	adfPushOption	By default the Aging Data FIFO contains all hardware aging requests, including modifying the hit state and clearing the entry. Set this field to 1 to only push when an entry needs to be cleared/aged out.	0x0



35.10.156 Learning Conflict

Status register for the failed port move operation. A valid status means the L2 Forwarding Information Base cannot bind the existing GID, MAC to a new port. Once the status register is updated from the hardware, no more fails can be updated until the software clears the valid field.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write
 Address Space : 595

Field Description

Bits	Field Name	Description	Default Value
0	valid	Indicates hardware has written a learning conflict to this status register. Write 0 to clear.	0x0
48:1	macAddr	MAC address.	0x0
60:49	gid	Global identifier from the VLAN Table.	0x0
65:61	port	Port number.	0x0

35.10.157 Learning DA MAC

The MAC address to be used by packets which are injected by software to be learned.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 81909
	Slice 1 : 179589
	Slice 2 : 277269
	Slice 3 : 374949

Field Description

Bits	Field Name	Description	Default Value
47:0	mac	The destination MAC address to be used by software when injecting new addresses to be learned	0x0
48	enable	Shall the switch accept learning packets? 0 = No 1 = Yes	0x0

35.10.158 Learning Data FIFO

This register exposes the output of a FIFO which is holding all learning requests that have been processed by the learning unit. A read from this register will pop one entry from the fifo. Under hardware learning writeback mode, all valid entries have been updated to the [FIB](#) regardless of hardware or software learning. When hardware learning writeback is turned off, software takes full control of the learning unit, hardware learning result will only be pushed to this FIFO but not update the related L2 tables.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read Only
 Address Space : 4799



Field Description

Bits	Field Name	Description	Default Value
47:0	mac	MAC address for a learning request.	0x0
63:48	gid	Global IDentifier from the gid field in the VLAN Table .	0x0
76:64	destAddress	The L2 Destination Table address decided by the learning unit.	0x0
77	uc	The uc field in the L2 Destination Table decided by the learning unit.	0x0
83:78	port_or_ptr	The destPort or mcAddr field in the L2 Destination Table decided by the learning unit.	0x0
84	drop	The pktDrop field in the L2 Destination Table decided by the learning unit.	0x0
85	l2ActionTableDaStatus	l2ActionTableDaStatus field in the L2 Destination Table	0x0
86	l2ActionTableSaStatus	l2ActionTableSaStatus field in the L2 Destination Table	0x0
102:87	metaData	metaData field in the L2 Destination Table	0x0
105:103	status	Entry status either refers to the L2 Aging Table or the L2 Aging Collision Table based on the destAddress field.	0x0
106	type	Type of the learning request. 0 = Hardware learning result 1 = Software learning result from a injected learning packet	0x0
107	valid	0 = FIFO is empty 1 = FIFO is not empty and the data is valid	0x0

35.10.159 Learning Data FIFO High Watermark Level

The High Watermark Interrupt will occur when a push to **Learning Data FIFO** is done and the number of existing entries after the push is larger than this setting.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 602

Field Description

Bits	Field Name	Description	Default Value
5:0	level	Number of used entries.	0x0

35.10.160 Learning Overflow

Status register for the failed hardware learning operation. A valid status means the L2 Forwarding Information Base cannot find an available slot for the unknown GID, MAC. Once the status register is updated from the hardware, no more fails can be updated until the software clears the valid field.



Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write
 Address Space : 597

Field Description

Bits	Field Name	Description	Default Value
0	valid	Indicates hardware has written a learning overflow to this status register, Write 0 to clear.	0x0
48:1	macAddr	MAC address.	0x0
60:49	gid	Global identifier from the VLAN Table.	0x0
65:61	port	Port number.	0x0

35.10.161 Link Aggregate Weight

The link aggregate hash will index into this table to determine which physical port within the aggregate that a packet should be output to. The number of bits set for a port will determine the ratio of packets that will go out on that port. For each hash index only one of the ports that belong to the same link aggregate must be set. The number of bits set divided by number of hash values determines the ratio of traffic going to that port. All link aggregates share this table since each physical port can only belong to one link aggregate. When a link aggregate only has one port then all bits for that port must be set.

Number of Entries : 256
 Type of Operation : Read/Write
 Addressing : The link aggregate hash.

Address Space

Slice 0 :	82059 to 82314
Slice 1 :	179739 to 179994
Slice 2 :	277419 to 277674
Slice 3 :	375099 to 375354

Field Description

Bits	Field Name	Description	Default Value
31:0	ports	One bit per physical port.	0x0

35.10.162 Link Aggregation Ctrl

This register controls whether link aggregation is enabled and which packet header fields that will be used to calculate the link aggregate hash value.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space

Slice 0 :	81906
Slice 1 :	179586
Slice 2 :	277266
Slice 3 :	374946



Field Description

Bits	Field Name	Description	Default Value
0	enable	Is Link aggregation enabled or not. 0 = Link Aggregation is disabled 1 = Link Aggregation is enabled	0x0
1	useSaMacInHash	The packets source MAC address shall be part of the hash key when calculating the link aggregate hash value	0x0
2	useDaMacInHash	The packets destination MAC addresses shall be part of the hash key when calculating the link aggregate hash value	0x0
3	useIpInHash	The packets IP source and destination addresses shall be part of the hash key when calculating the link aggregate hash value	0x0
4	useL4InHash	The packets L4 SP / DP and L4 protocol byte shall be part of the hash key when calculating the link aggregate hash value	0x0
5	useTosInHash	The incoming packets TOS byte shall be part of the hash key when calculating the link aggregate hash value	0x0
6	useNextHopInHash	For routed packets the next hop entry shall be part of the hash key when calculating the link aggregate hash value.	0x0
7	useVlanIdInHash	The packets VLAN Identifier tag shall be part of the hash key when calculating the link aggregate hash value.	0x0

35.10.163 Link Aggregation Membership

This register is used to determine which link aggregation a specific source port is membership of. If link aggregation is enabled then this port number is used for all source lookups instead of the port where the packet entered the switch.

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Ingress port

Address Space

Slice 0 :	85263 to 85294
Slice 1 :	182943 to 182974
Slice 2 :	280623 to 280654
Slice 3 :	378303 to 378334

Field Description

Bits	Field Name	Description	Default Value
4:0	la	The Link aggregation which this port is a member of	0x0

35.10.164 Link Aggregation To Physical Ports Members

This link aggregate portmasks are setup to determine which physical ports are members of each link aggregate.



Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : The link aggregate number.

Address Space	Slice 0 : 82027 to 82058
	Slice 1 : 179707 to 179738
	Slice 2 : 277387 to 277418
	Slice 3 : 375067 to 375098

Field Description

Bits	Field Name	Description	Default Value
31:0	members	Physical ports that are members of this link aggregate. One bit per port.	0x0

35.10.165 MPLS EXP Field To Egress Queue Mapping Table

Mapping table from MPLS EXP priority fields to egress queues.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Incoming packets MPLS EXP priority bits

Address Space	Slice 0 : 83743 to 83750
	Slice 1 : 181423 to 181430
	Slice 2 : 279103 to 279110
	Slice 3 : 376783 to 376790

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue	0x1

35.10.166 MPLS EXP Field To Packet Color Mapping Table

Mapping table from MPLS EXP priority fields to packet initial color.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Incoming packets MPLS EXP priority bits

Address Space	Slice 0 : 83207 to 83214
	Slice 1 : 180887 to 180894
	Slice 2 : 278567 to 278574
	Slice 3 : 376247 to 376254

Field Description



Bits	Field Name	Description	Default Value
1:0	color	Packet initial color	0x0

35.10.167 NAT Action Table

At end of ingress processing a check is done to determine what to do with the packets. This table is used to setup operations based on the port states [Egress Port NAT State](#) and [natPortState](#)

Number of Entries : 512

Type of Operation : Read/Write

Addressing :

Address Bit 0 :	Source Port NAT State natPortState
Address Bit 1 :	Egress Port NAT State Egress Port NAT State
Address Bit [3:2] :	Was packet Switched or Routed 0 = Other - sendToCpu,sendToPort from classification. 1 = Routed. 2 = Flooded. 3 = Switched - The packet hit a DA table entry and was not routed.
Address Bit 4 :	Was ingress ACL NAT operation enabled. 0 = No 1 = Yes
Address Bit 5 :	Was egress ACL NAT operation enabled. 0 = No 1 = Yes
Address Bit 6 :	Was the packet switched/routed unicast/multicast. Only valid for switching and routing. Otherwise set to zero. 0 = Multicast. 1 = Unicast.
Address Bit [8:7] :	L3 Packet Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Other

Address Space

Slice 0 :	82363 to 82874
Slice 1 :	180043 to 180554
Slice 2 :	277723 to 278234
Slice 3 :	375403 to 375914

Field Description

Bits	Field Name	Description	Default Value
1:0	action	What to do with the packet depending on what port states are. 0 = No Operation 1 = Send to CPU Reason NAT Action Table Code 1 2 = Send to CPU Reason NAT Action Table Code 2 3 = Drop the packet. Update counter NAT Action Table Drop .	0x0

35.10.168 NAT Action Table Force Original Packet

If the NAT Action Table forces packets to be send to the CPU then they can either be the processed packet or the original packet. This register sets up for each reason what the packet to the CPU shall be.



Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	81936
Slice 1 :	179616
Slice 2 :	277296
Slice 3 :	374976

Field Description

Bits	Field Name	Description	Default Value
0	reasonOne	Force the packet to the CPU from the NAT action table Reason type 1 to be the original packet.	0x0
1	reasonTwo	Force the packet to the CPU from the NAT action table Reason type 2 to be the original packet.	0x0

35.10.169 Next Hop Packet Modifications

Determines the VLAN operations to perform on the packet exiting the router. One or two VLAN headers can be added to the outgoing packet.

Number of Entries : 1024

Type of Operation : Read/Write

Addressing : [nextHopPacketMod](#)

Address Space

Slice 0 :	50754 to 51777
Slice 1 :	148434 to 149457
Slice 2 :	246114 to 247137
Slice 3 :	343794 to 344817

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this a valid entry. If the router points to an entry with this field cleared the packet will be sent to CPU. 0 = Invalid 1 = Valid	0x0
1	outerVlanAppend	Insert/push an outer VLAN header in the packet. The information used to create the new VLAN header is controlled by the fields outerVid , outerPcpSel , outerCfiDeiSel and outerEthType . If the selected outermost VLAN header field doesn't exist in the packet then the new VLAN header field will be taken from Router Egress Queue To VLAN Data . 0 = No operation. 1 = Insert/push an outer VLAN tag.	0x0
3:2	outerPcpSel	Selects which PCP bits to use when building an outer VLAN header. 0 = From outermost VLAN header in the original packet (if any). 1 = From this entrie's outerPcp field. 2 = From Router Egress Queue To VLAN Data .	0x0



Bits	Field Name	Description	Default Value
5:4	outerCfiDeiSel	Selects which CFI/DEI bit to use when building an outer VLAN header. 0 = From outermost VLAN header in the original packet (if any). 1 = From this entrie's outerCfiDei field. 2 = From Router Egress Queue To VLAN Data .	0x0
7:6	outerEthType	Pointer to the VLAN type. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN.	0x0
19:8	outerVid	The VID used when building an outer VLAN header.	0x0
22:20	outerPcp	The PCP bits to use when building an outer VLAN header. If selected by outerPcpSel .	0x0
23	outerCfiDei	The CFI/DEI bit to use when building an outer VLAN header. If selected by outerCfiDeiSel .	0x0
24	innerVlanAppend	Insert/push an inner VLAN header in the packet. The information used to create the new VLAN header is controlled by the fields innerVid , innerPcpSel , innerCfiDeiSel and innerEthType . If the selected innermost VLAN header field doesn't exist in the packet then the new VLAN header field will be taken from Router Egress Queue To VLAN Data . 0 = No operation 1 = Insert/push an inner VLAN tag.	0x0
26:25	innerPcpSel	Selects which PCP bits to use when building an inner VLAN header. 0 = From innermost VLAN header in the original packet (if any). 1 = From this entrie's innerPcp field. 2 = From Router Egress Queue To VLAN Data .	0x0
28:27	innerCfiDeiSel	Selects which CFI/DEI bit to use when building an inner VLAN header. 0 = From innermost VLAN header in the original packet (if any). 1 = From this entrie's innerCfiDei field. 2 = From Router Egress Queue To VLAN Data .	0x0
30:29	innerEthType	Pointer to the VLAN type. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN.	0x0
42:31	innerVid	The VID used when building an inner VLAN header.	0x0
45:43	innerPcp	The PCP bits to use when building an inner VLAN header. If selected by innerPcpSel .	0x0
46	innerCfiDei	The CFI/DEI bit to use when building an inner VLAN header. If selected by innerCfiDeiSel .	0x0
50:47	msptPtr	The multiple spanning tree to be used by packets for egress spanning tree check for this next hop. Points to an entry in Egress Multiple Spanning Tree State	0x0

35.10.170 Next Hop Table

Forwarding decision for a routed packet including destination port(s), or if packet shall be dropped or sent to the CPU port.



Number of Entries : 1024
 Type of Operation : Read/Write
 Addressing : Next Hop Pointer

Address Space	Slice 0 : 49730 to 50753
	Slice 1 : 147410 to 148433
	Slice 2 : 245090 to 246113
	Slice 3 : 342770 to 343793

Field Description

Bits	Field Name	Description	Default Value
0	validEntry	Is this a valid entry or not. If the entry is not valid then the packet shall be sent to the CPU for further processing	0x0
1	srv6Sid	If set, this entry is a locally instantiated SRv6 segment identifier	0x0
11:2	nextHopPacketMod	Pointer into the Next Hop Packet Modifications table and the Next Hop DA MAC table.	0x0
12	l2Uc	L2 unicast or multicast. A multicast means that a lookup in the L2 Multicast Table will take place to determine the destination portmask. 0 = L2 multicast. 1 = L2 unicast.	0x0
18:13	destPort_or_mcAddr	Destination port number or a pointer into the L2 Multicast Table	0x0
19	pktDrop	If set then the packet will be dropped and the L3 Lookup Drop incremented.	0x0
20	sendToCpu	If set then the packet will be sent to the CPU.	0x0
21	tunnelEntry	Shall this packet enter into a tunnel.	0x0
25:22	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the router. If field l2Uc is set to L2 multicast then Tunnel Entry Instruction Table uses the egress port as a offset from this base pointer.	0x0
26	tunnelExit	Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
30:27	tunnelExitPtr	Pointer to tunnel exit described in Egress Tunnel Exit Table .	0x0
46:31	metaData	Meta data for to CPU tag.	0x0

35.10.171 Port Move Options

Determine if port move is allowed on static entries.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 81931
	Slice 1 : 179611
	Slice 2 : 277291
	Slice 3 : 374971

Field Description



Bits	Field Name	Description	Default Value
31:0	allowPortMoveOnStatic	This field configures which source ports that are allowed to change their static GID and MAC to other ports. One bit for each port where bit 0 corresponds to port 0. When the L2 forwarding information base identifies a GID, MAC SA and source port combination that conflicts with a existing static entry, if the previous binded port has a corresponding bit set to 1 in this field, it allows the learning engine to update the GID and MAC to the current source port.	$2^{32} - 1$

35.10.172 RARP Packet Decoder Options

The Ethernet type used to determine if a packet is a RARP packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write

Address Space

Slice 0 :	85489
Slice 1 :	183169
Slice 2 :	280849
Slice 3 :	378529

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x8035
48:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
80:49	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.173 Reserved Destination MAC Address Range

The mac addresses ranges that the packets destination MAC address are compared with and the corresponding actions. A range is matched if the packets MAC address is $\geq startAddr$ and the address is $\leq stopAddr$. The table is searched starting from entry 0. When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated. If multiple ranges are matched, any matching range that sets drop will cause a drop. Any match that sets sendToCpu will cause send to CPU (this



has priority over drop). When multiple ranges that match has set the forceQueue field then the highest numbered entry will determine the value.

Number of Entries :	4
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space	Slice 0 : 86285 to 86300
	Slice 1 : 183965 to 183980
	Slice 2 : 281645 to 281660
	Slice 3 : 379325 to 379340

Field Description

Bits	Field Name	Description	Default Value
47:0	startAddr	The start MAC address of the range. A packets destination MAC address must be equal or greater than this value to match the range.	0x0
95:48	stopAddr	The end MAC address of the range. A packets destination MAC address must be equal or less than this value to match the range.	0x0
96	dropEnable	If the MAC address was within the range the packet shall be dropped and the Reserved MAC DA Drop counter incremented.	0x0
97	sendToCpu	If the MAC address was within the range the packet shall be sent to the CPU.	0x0
98	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
101:99	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
103:102	color	Initial color of the packet.	0x0
104	forceColor	If set, the packet shall have a forced color.	0x0
105	mmpValid	If set, this entry contains a valid MMP pointer	0x0
110:106	mmpPtr	Ingress MMP pointer.	0x0
112:111	mmpOrder	Ingress MMP pointer order.	0x0
144:113	enable	Enable the reserved MAC DA check per source port. One bit for each port where bit 0 corresponds to port 0. If a bit is set to one, the reserved MAC DA range is activated for that source port.	0x0

35.10.174 Reserved Source MAC Address Range

The mac addresses ranges that the packets source MAC address are compared with and the corresponding actions. A range is matched if the packets MAC address is $\geq startAddr$ and the address is $\leq stopAddr$. The table is searched starting from entry 0. When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated. If multiple ranges are matched, any matching range that sets drop will cause a drop. Any match that sets sendToCpu will cause send to CPU (this has priority over drop). When multiple ranges that match has set the forceQueue then the highest numbered entry will determine the value.



Number of Entries : 4
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space	Slice 0 : 86269 to 86284
	Slice 1 : 183949 to 183964
	Slice 2 : 281629 to 281644
	Slice 3 : 379309 to 379324

Field Description

Bits	Field Name	Description	Default Value
47:0	startAddr	The start MAC address of the range. A packets source MAC address must be equal or greater than this value to match the range.	0x0
95:48	stopAddr	The end MAC address of the range. A packets source MAC address must be equal or less than this value to match the range.	0x0
96	dropEnable	If the MAC address was within the range the packet shall be dropped and the Reserved MAC SA Drop counter incremented.	0x0
97	sendToCpu	If the MAC address was within the range the packet shall be sent to the CPU.	0x0
98	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
101:99	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
103:102	color	Initial color of the packet.	0x0
104	forceColor	If set, the packet shall have a forced color.	0x0
105	mmpValid	If set, this entry contains a valid MMP pointer	0x0
110:106	mmpPtr	Ingress MMP pointer.	0x0
112:111	mmpOrder	Ingress MMP pointer order.	0x0
144:113	enable	Enable the reserved source MAC check per source port. One bit for each port where bit 0 corresponds to port 0. If a bit is set to one, the reserved source MAC range is activated for that source port.	0x0

35.10.175 Router Egress Queue To VLAN Data

Map from egress queue number to VLAN PCP and CFI/DEI values to be used in router VLAN operations selected by **Next Hop Packet Modifications**.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress Queue

Address Space	Slice 0 : 83199 to 83206
	Slice 1 : 180879 to 180886
	Slice 2 : 278559 to 278566
	Slice 3 : 376239 to 376246

Field Description



Bits	Field Name	Description	Default Value
0	cfiDei	Map from egress queue to CFI/DEI	0x0
3:1	pcp	Map from egress queue to PCP	0x0

35.10.176 Router MTU Table

An MTU check is done on each routed packet by comparing the IPv4 Total Length field with the **maxIPv4MTU** limit. Correspondingly IPv6 Payload Length field is compared with **maxIPv6MTU**. If the length field exceeds the limit the packet will be sent to the CPU. Each router VRF has a MTU limit for each port.

Number of Entries :	128
Type of Operation :	Read/Write
Addressing :	destination-port * 4 + VRF
Address Space	Slice 0 : 82907 to 83034
	Slice 1 : 180587 to 180714
	Slice 2 : 278267 to 278394
	Slice 3 : 375947 to 376074

Field Description

Bits	Field Name	Description	Default Value
15:0	maxIPv4MTU	The maximum MTU allowed for IPv4 packets	0xffff
31:16	maxIPv6MTU	The maximum MTU allowed for IPv6 packets	0xffff

35.10.177 Router Port MAC Address

The incoming packets destination MAC address is compared against all the entries in the table. If there is a match after the macMask has been applied the packet will enter the routing function with the VRF identifier assigned from the matching entry. The table must be configured so that only one match is possible.

Number of Entries :	16
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space	Slice 0 : 86125 to 86188
	Slice 1 : 183805 to 183868
	Slice 2 : 281485 to 281548
	Slice 3 : 379165 to 379228

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddress	The base destination MAC address that is used to identify packets to the router.	0x0



Bits	Field Name	Description	Default Value
95:48	macMask	Each bit says if the bit in the DA MAC shall be compared. 0 = Dont compare bit. 1 = Compare bit.	0x0
127:96	selectMacEntryPortMask	Portmask to select which MAC address to use. One bit per source port. 0 = use macAddress. 1 = use altMacAddress.	0x0
175:128	altMacAddress	The alternative base destination MAC address that is used to identify packets to the router.	0x0
176	valid	If set, this entry is valid for comparison.	0x0
178:177	vrf	The VRF to use for this router	0x0

35.10.178 SCTP Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a SCTP header. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write

Address Space

Slice 0 :	85497
Slice 1 :	183177
Slice 2 :	280857
Slice 3 :	378537

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	I4Proto	The value to be used to find this packet type.	0x84
40:9	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
72:41	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

35.10.179 SMON Set Search

If both source port and VLAN ID match one of the entries, the corresponding SMON counter will be updated.



Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : SMON set number

Address Space	Slice 0 : 84951 to 84954
	Slice 1 : 182631 to 182634
	Slice 2 : 280311 to 280314
	Slice 3 : 377991 to 377994

Field Description

Bits	Field Name	Description	Default Value
4:0	srcPort	Source port	0x0
16:5	vid	VLAN ID	0x0

35.10.180 SNAP LLC Decoding Options

When a SNAP/LLC packet is received there are some options which allows the packet if not recognized to be sent to the CPU. The packet will have a special reason code

Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 : 81910
	Slice 1 : 179590
	Slice 2 : 277270
	Slice 3 : 374950

Field Description

Bits	Field Name	Description	Default Value
15:0	ethSize	What maximum size of packet shall be interpreted as SNAP packet.	0x5dc
47:16	sendToCpu	When a LLC is not equal to (dsap==0xAA and ssap==0xAA and ctrl==0x03) then packet will be sent to cpu. Bit 0 is from port 0, bit 1 is for port 1, etc.	0x0

35.10.181 Second Tunnel Exit Lookup TCAM

The extracted key from packet which is described in the tunnel exit lookup.

Number of Entries : 16
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : All entries are read out in parallel

Address Space	Slice 0 : 86061 to 86124
	Slice 1 : 183741 to 183804
	Slice 2 : 281421 to 281484
	Slice 3 : 379101 to 379164



Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
80:1	pktKey_mask	Mask for pktKey.	$2^{80} - 1$
160:81	pktKey	The extracted key from the packet according to the first lookup.	0x0
162:161	tabKey_mask	Mask for tabKey.	0x3
164:163	tabKey	The key from the first tunnel exit lookup result table.	0x0

35.10.182 Second Tunnel Exit Lookup TCAM Answer

This is the table holding the answer for the [Second Tunnel Exit Lookup TCAM](#).

Number of Entries : 16
 Type of Operation : Read/Write
 Addressing : [Second Tunnel Exit Lookup TCAM](#) hit index

Address Space

Slice 0 :	85247 to 85262
Slice 1 :	182927 to 182942
Slice 2 :	280607 to 280622
Slice 3 :	378287 to 378302

Field Description

Bits	Field Name	Description	Default Value
8:0	howManyBytesToRemove	How many bytes to remove.	0x0
9	updateEthType	If packet is removed after L2+VLAN headers then update the Ethernet Header Type Field	0x0
25:10	ethType	If packet is removed after L2+VLAN headers then the New Ethernet Type which will overwrite the existing lowest 16 bits after the removal operation.	0x0
26	removeVlan	If packet is removed after L2+VLAN headers then remove the VLAN headers on the incoming packet.	0x0
27	updateL4Protocol	If packet is removed after L3 headers then update the L4 Protocol in IP header.	0x0
35:28	l4Protocol	If packet is removed after L3 headers then this new L4 Protocol will be written.	0x0
37:36	whereToRemove	Where to do the tunnel exit from 0 = At Byte Zero 1 = After L2 and up to two VLAN headers. 2 = After L3 IPv4/IPv6 headers. 3 = Reserved.	0x0
38	dropPkt	Drop the packet.	0x0
39	dontExit	Do not do a tunnel exit on this packet.	0x0
40	replaceVid	Replace the assigned VID. This is the VID which shall be used in the VLAN table lookup. This forces a new VID into this packet and bypassing all but the ACL force VID operation.	0x0
52:41	newVid	The new to be used VID.	0x0



Bits	Field Name	Description	Default Value
57:53	tunnelExitEgressPtr	Tunnel Exit Egress Pointer. Shall point to same tunnel / packet decapsulation operation but setup in egress pipeline in Egress Tunnel Exit Table	0x0
58	removeFromCpuTag	If packet came in and had a From CPU Tag does the tunnel exit lookup remove the From CPU Tag or should the design remove this TAG? 0 = Ignore the FROM CPU Tag, the tunnel exit will remove this. 1 = The hardware should remove the From CPU Tag.	0x0

35.10.183 Second Tunnel Exit Miss Action

When a packet misses in the tunnel second lookup table shall this packet be dropped or not?

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : The tblIndex result field from the first tunnel exit lookup

Address Space

Slice 0 :	85243 to 85246
Slice 1 :	182923 to 182926
Slice 2 :	280603 to 280606
Slice 3 :	378283 to 378286

Field Description

Bits	Field Name	Description	Default Value
0	droplfMiss	If miss in this table then drop packet 0 = No 1 = Yes	0x0

35.10.184 Send to CPU

Configuration of MAC addresses used to redirect packets to CPU.

Number of Entries : 1
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write

Address Space

Slice 0 :	86301
Slice 1 :	183981
Slice 2 :	281661
Slice 3 :	379341

Field Description

Bits	Field Name	Description	Default Value
31:0	allowBpdu	Send to CPU portmask, bit 0 port 0, bit 1 port 1 etc. If source port bit is set then packets that have the destination MAC address equal to 01:80:C2:00:00:00 are sent to the CPU port.	$2^{32} - 1$



Bits	Field Name	Description	Default Value
63:32	allowRstBpdu	Send to CPU portmask, bit 0 port 0, bit 1 port 1 etc. If the source port bit is set then packets that have the destination MAC address equal to 01:00:0C:CC:CC:CD are sent to the CPU port.	$2^{32} - 1$
95:64	uniqueCpuMac	If set then unicast packets can not be switched or routed to the CPU port. Other mechanism for sending to the CPU port are not affected (e.g. ACL's). This also enables detection of a specific MAC address, cpuMacAddr , that will be sent to the CPU.	0x0
143:96	cpuMacAddr	Packets with this destination MAC address will be sent to the CPU. Only valid if uniqueCpuMac on the source port is set.	0x0

35.10.185 Software Aging Enable

If set, the hardware aging unit will stop the countdown - age out loop, instead software is responsible for counting down and triggering an age out round by [Software Aging Start Latch](#).

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 600

Field Description

Bits	Field Name	Description	Default Value
0	enable	Enable software aging.	0x0

35.10.186 Software Aging Start Latch

This is used under software aging mode when [Software Aging Enable](#) is set.

Number of Entries : 1
 Type of Operation : Write Only
 Address Space : 4798

Field Description

Bits	Field Name	Description	Default Value
0	start	When register is written with start bit set an age out process is started.	0x0



35.10.187 Source Port Default ACL Action

The default ACL action which will be taken on a source port if the [enableDefaultPortAcl](#) is set and the ACL lookup misses. The action will also be taken if the [forcePortAclAction](#) is set and then it will override the result from the ACL even if the ACL was hit or not.

Number of Entries : 32
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : Source Port

Address Space

Slice 0 :	24990 to 25117
Slice 1 :	122670 to 122797
Slice 2 :	220350 to 220477
Slice 3 :	318030 to 318157

Field Description

Bits	Field Name	Description	Default Value
0	metaDataValid	Is the meta_data field valid.	0x0
16:1	metaData	Meta data for packets going to the CPU.	0x0
17	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
22:18	destInputMirror	Destination physical port for input mirroring.	0x0
23	noLearning	If set this packets MAC SA will not be learned.	0x0
24	updateCounter	When set the selected statistics counter will be updated.	0x0
30:25	counter	Which counter in Ingress Configurable ACL Match Counter to update.	0x0
31	updateTosExp	Force TOS/EXP update.	0x0
39:32	newTosExp	New TOS/EXP value.	0x0
47:40	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
48	forceVidValid	Override the Ingress VID, see chapter VLAN Processing .	0x0
60:49	forceVid	The new Ingress VID.	0x0
61	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
62	newCfiDeiValue	The value to update to.	0x0
63	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
66:64	newPcpValue	The PCP value to update to.	0x0



Bits	Field Name	Description	Default Value
67	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
79:68	newVidValue	The VID value to update to.	0x0
80	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
82:81	newEthType	Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag .	0x0
83	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
84	updateSaOrDa	Update the SA or DA IPv4 address. The Destination IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
116:85	newIpValue	Update the SA or DA IPv4 address value.	0x0
117	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
118	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
134:119	newL4Value	Update the L4 SP or DP with this value	0x0
135	dropEnable	If set, the packet shall be dropped and the Ingress Configurable ACL Drop counter is incremented.	0x0
136	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
137	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming original packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
138	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
143:139	destPort	The port which the packet shall be sent to.	0x0
144	ptp	When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
145	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0



Bits	Field Name	Description	Default Value
146	tunnelEntryUcMc	Shall this entry point to the Tunnel Entry Instruction Table with or without a egress port offset. 0 = Unicast Tunnel Entry Instruction Table without offset for each port 1 = Multicast Tunnel Entry Instruction Table with offset for each port.	0x0
150:147	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
151	forceColor	If set, the packet shall have a forced color.	0x0
153:152	color	Initial color of the packet if the forceColor field is set.	0x0
154	mmpValid	If set, this entry contains a valid MMP pointer	0x0
159:155	mmpPtr	Ingress MMP pointer.	0x0
161:160	mmpOrder	Ingress MMP pointer order.	0x0
162	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 21.1	0x0
165:163	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
166	natOpValid	NAT operation pointer is valid.	0x0
177:167	natOpPtr	NAT operation pointer.	0x0

35.10.188 Source Port Table

This table configures various functions that are dependent on which port the packet enters the switch. A VLAN operation (e.g. push, pop, swap) to be performed can be selected by the **vlanSingleOp** field in **Source Port Table**. For the push and swap operations the information used to create the new VLAN header is controlled by the fields **vidSel**, **cfiDeiSel**, **pcpSel** and **typeSel**. Other configurations are VLAN LUT index, input mirroring, spanning tree state, Ingress VID offset, special VID treatment, multicast learning, min/max number of VLANs and L3 priority selection.

Number of Entries : 32
 Number of Addresses per Entry : 2
 Type of Operation : Read/Write
 Addressing : Ingress port

Address Space

Slice 0 :	85295 to 85358
Slice 1 :	182975 to 183038
Slice 2 :	280655 to 280718
Slice 3 :	378335 to 378398

Field Description

Bits	Field Name	Description	Default Value
0	learningEn	If hardware learning is turned on and this is set to one, the unknown source MAC address from this port will be learned.	0x1
1	dropUnknownDa	If set to one packets with unknown destination MAC address from this port will be dropped.	0x0



Bits	Field Name	Description	Default Value
2	prioFromL3	If the packet is IP/MPLS and this is set the egress queue will be selected from Layer 3 decoding described in Determine Egress Queue .	0x0
3	colorFromL3	If the packet is IP/MPLS and this bit is set the packet initial color will be selected from Layer 3 decoding.	0x0
4	useAcl0	Use ACL on this source port. 0 = No. No ACL lookup is done 1 = Yes. The aclRule0 pointer selects which fields that are part of the lookup	0x0
7:5	aclRule0	Pointer into the Ingress Configurable ACL 0 Rules Setup table selecting which ACL fields to select to do the ACL lookup with.	0x0
8	useAcl1	Use ACL on this source port. 0 = No. No ACL lookup is done 1 = Yes. The aclRule1 pointer selects which fields that are part of the lookup	0x0
11:9	aclRule1	Pointer into the Ingress Configurable ACL 1 Rules Setup table selecting which ACL fields to select to do the ACL lookup with.	0x0
12	useAcl2	Use ACL on this source port. 0 = No. No ACL lookup is done 1 = Yes. The aclRule2 pointer selects which fields that are part of the lookup	0x0
15:13	aclRule2	Pointer into the Ingress Configurable ACL 2 Rules Setup table selecting which ACL fields to select to do the ACL lookup with.	0x0
16	useAcl3	Use ACL on this source port. 0 = No. No ACL lookup is done 1 = Yes. The aclRule3 pointer selects which fields that are part of the lookup	0x0
19:17	aclRule3	Pointer into the Ingress Configurable ACL 3 Rules Setup table selecting which ACL fields to select to do the ACL lookup with.	0x0
22:20	vlanSingleOp	The source port VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0



Bits	Field Name	Description	Default Value
24:23	vidSel	Selects which VID to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's defaultVid will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's defaultVid . 2 = From the second VLAN in the original packet (if any).	0x0
26:25	cfiDeiSel	Selects which CFI/DEI to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's defaultCfiDei will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's defaultCfiDei . 2 = From the second VLAN in the original packet (if any).	0x0
28:27	pcpSel	Selects which PCP to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's defaultPcp will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's defaultPcp . 2 = From the second VLAN in the original packet (if any).	0x0
30:29	nrVlansVidOperationIf	This alternative VID operation for port VLAN operation is selected if the following operation is true. 0 = Nr of VLANS in incoming packet is zero. 1 = Nr of VLANS in incoming packet is one. 2 = Nr of VLANS in incoming packet is two. 3 = Reserved and Disabled	0x3
33:31	vlanSingleOpIf	If the field nrVlansVidOperationIf is true then this operation will override the default port vid operation vlanSingleOp . The source port VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0



Bits	Field Name	Description	Default Value
35:34	vidSelf	<p>If the field nrVlansVidOperationIf is true then this operation will override the default port vid operation vidSel. Selects which VID to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's defaultVidIf will be used.</p> <p>0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's defaultVid. 2 = From the second VLAN in the original packet (if any).</p>	0x0
37:36	cfiDeiSelf	<p>If the field nrVlansVidOperationIf is true then this operation will override the default port vid operation cfiDeiSel. Selects which CFI/DEI to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's defaultCfiDeiIf will be used.</p> <p>0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's defaultCfiDei. 2 = From the second VLAN in the original packet (if any).</p>	0x0
39:38	pcpSelf	<p>If the field nrVlansVidOperationIf is true then this operation will override the default port vid operation pcpSel. Selects which PCP to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's defaultPcpIf will be used.</p> <p>0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's defaultPcp. 2 = From the second VLAN in the original packet (if any).</p>	0x0
41:40	typeSelf	<p>If the field nrVlansVidOperationIf is true then this operation will override the default port vid operation typeSel. Selects which TPID to use when building a new VLAN header in a source port push or swap operation.</p> <p>0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag.</p>	0x0



Bits	Field Name	Description	Default Value
53:42	defaultVidIf	The default VID if nrVlansVidOperationIf is true. This is used in source port VLAN operations (see vidSel). It is used to assign Ingress VID (see vlanAssignment). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
54	defaultCfiDeiIf	The default CFI / DEI bit if nrVlansVidOperationIf is true. This is used in source port VLAN operations (see cfiDeiSel). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
57:55	defaultPcpIf	The default PCP bits if nrVlansVidOperationIf is true. This is used in source port VLAN operations (see pcpSel). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
59:58	typeSel	Selects which TPID to use when building a new VLAN header in a source port push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag .	0x0
61:60	vlanAssignment	Controls how a packets Ingress VID is assigned. If the selected source is from a VLAN header in the incoming packet and the packet doesn't have that header, then this table entry's defaultVid will be used. 0 = packet based - the Ingress VID is assigned from the incoming packets outermost VLAN header. 1 = port-based - the packets Ingress VID is assigned from this table entry's defaultVid 2 = mixed - if there are two VLANs in the incoming packet, the inner VLAN is chosen. If the incoming packet has only 0 or 1 VLAN, then it will select this table entry's defaultVid	0x0
73:62	defaultVid	The default VID. This is used in source port VLAN operations (see vidSel). It is used to assign Ingress VID (see vlanAssignment). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
74	defaultCfiDei	The default CFI / DEI bit. This is used in source port VLAN operations (see cfiDeiSel). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0



Bits	Field Name	Description	Default Value
77:75	defaultPcp	The default PCP bits. This is used in source port VLAN operations (see .pcpSel). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
79:78	defaultVidOrder	When a new hit is done in the result in the L2,L3,L4 VID range checks the ingress VID will only be changed if the result has a higher order value.	0x0
81:80	minAllowedVlans	The minimum number of VLAN headers a packet must have to be allowed on this port. Otherwise the packet will be dropped and the Minimum Allowed VLAN Drop will be incremented. 0 = All packets are accepted. 1 = 1 or more tags are accepted. 2 = 2 or more tags are accepted. 3 = No packets are accepted.	0x0
83:82	maxAllowedVlans	The maximum number of VLAN headers a packet is allowed to have to enter on this port. Otherwise the packet will be dropped and the Maximum Allowed VLAN Drop will be incremented. 0 = Only untagged packets are accepted. 1 = 0 to 1 tags are accepted. 2 = Any number of VLANs are accepted. 3 = Any number of VLANs are accepted.	0x2
84	ignoreVlanMembership	By default packets on non-VLAN member source port are dropped before entering the L2 lookup process. Set this field to one to ignore the VLAN membership check on the source port. However L2 lookup can never forward packets to non-VLAN member destinations.	0x0
85	learnMulticastSaMac	If set, the learning engine allows Ethernet multicast source MAC addresses to be learned.	0x0
86	inputMirrorEnabled	If set, input mirroring is enabled on this port. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destInputMirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
87	imUnderVlanMembership	If set, input mirroring to a destination that not a member of the VLAN will be ignored.	0x0
88	imUnderPortIsolation	If set, input mirroring to a destination that isolated the source port in the srcPortFilter will be ignored.	0x0
93:89	destInputMirror	Destination physical port for input mirroring. Only valid if inputMirrorEnabled is set.	0x0



Bits	Field Name	Description	Default Value
96:94	spt	The spanning tree state for this ingress port. The state Disabled implies that spanning tree protocol is not enabled and hence frames will be forwarded on this egress port. 0 = Disabled. 1 = Blocking. 2 = Listening. 3 = Learning. 4 = Forwarding.	0x0
97	enablePriorityTag	An outer VLAN tag with VID matching priorityVid will have PCP bits extracted and used to determine output queue but in remaining VLAN processing this tag will not be treated as a VLAN tag. If the packet has an inner VLAN tag this will be treated as an outer VLAN tag in the following VLAN processing. The VID will only be matched in a VLAN header located immediately after DA and SA MAC, i.e. no custom tags allowed. In egress processing the outer VLAN tag will be removed. 0 = Disable comparison. 1 = Enable comparison.	0x0
109:98	priorityVid	The VID used in the outer VLAN tag comparison, see enablePriorityTag .	0x0
110	enableFromCpuTag	This option can validate the from CPU tag decoding on packets from non-CPU ports. The CPU port is not affected by this field and always decode the from CPU tag.	0x0
111	disableTunnelExit	On this source port are the packets allowed to do a tunnel exit. 0 = Yes 1 = No	0x0
112	firstHitSecondMissSendToCpu	If first tunnel lookup exit hit but second tunnel exit lookup fails then send the packet to the CPU. 0 = Do nothing. 1 = Send the packet to the CPU.	0x0
113	disableRouting	On this source port are the packets allowed to do L3 routing. 0 = No 1 = Yes	0x0
114	natActionTableEnable	Packets coming in on this source port should be checked in the NAT Action Table . 0 = No. 1 = Yes.	0x0
115	natPortState	What is this ports NAT status. 0 = Private 1 = Public	0x0



Bits	Field Name	Description	Default Value
116	enableL2ActionTable	On packets coming in on this port should be checked with the L2 Action Table and L2 Action Table Source Port . 0 = No, Do not lookup on the L2 Action Table and L2 Action Table Source Port . 1 = Yes. Do Lookup in the L2 Action Table and L2 Action Table Source Port	0x0
117	l2ActionTablePortState	What is the source port status bit. Used in table L2 Action Table and L2 Action Table Source Port .	0x0
118	enableDefaultPortAcl	If enabled then the default acl for this port will be done if the ACL misses in its lookup. 0 = Disabled. No default action taken. 1 = Enabled. If ACL lookup misses then this ACL actil will be carried out instead.	0x0
119	forcePortAclAction	If enabled then the default acl for this port will always be done, if the ACL is hit then the port ACL will overwrite the ACL result. 0 = Disabled. Not action forced. 1 = Enabled. The port ACL overwrites and result from the ingress ACL.	0x0
121:120	preLookupAclBits	Pre lookup bits which is used by this port in the pre-lookup tables in the ingress ACLS. Same value is used for all pre ACL lookups which has the source port bits in it.	0x0

35.10.189 Time to Age

Interval period after which [FIB](#) entries are aged out.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 637

Field Description

Bits	Field Name	Description	Default Value
31:0	tickCnt	Number of ticks (see Chapter Tick) between starts of the aging process.	$2^{32} - 1$
34:32	tick	Select one of the 8 available ticks. The tick frequencies are configured globally in the Core Tick Configuration register.	0x0

35.10.190 Tunnel Entry MTU Length Check

If a packet is routed and if the tunnel entry updates the IPv4 or IPv6 packet length then this table shall be setup to enable the too long packets to be sent to the CPU for fragmentation.



Number of Entries :	16
Type of Operation :	Read/Write
Addressing :	Tunnel entry pointer
Address Space	Slice 0 : 81890 to 81905
	Slice 1 : 179570 to 179585
	Slice 2 : 277250 to 277265
	Slice 3 : 374930 to 374945

Field Description

Bits	Field Name	Description	Default Value
6:0	length	The added length of a IPv4 or IPv6 packet.	0x0

35.10.191 Tunnel Exit Lookup TCAM

The tunnel exit lookup which is performed on the incoming original packet

Number of Entries :	16
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space	Slice 0 : 85517 to 85772
	Slice 1 : 183197 to 183452
	Slice 2 : 280877 to 281132
	Slice 3 : 378557 to 378812

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
1	snapLlc_mask	Mask for snapLlc.	0x1
2	snapLlc	This is a SNAP and LLC packet.	0x0
18:3	ethType_mask	Mask for ethType.	0xffff
34:19	ethType	Ethernet Type for the incoming packet.	0x0
37:35	l3Type_mask	Mask for l3Type.	0x7
40:38	l3Type	The L3 type which shall be matched. If unknown L3 type then this will set to 7. 0 = IPv4 1 = IPv6 2 = One MPLS Label 3 = Two MPLS Labels 4 = Three MPLS labels 5 = Four MPLS labels	0x0
43:41	frag_mask	Mask for frag.	0x7
46:44	frag	IPv4 header fragments bits, if IPv6/MPLS then these bits are set to zero. The bit 0 is the dont-fragment flag (DF bit), bit 1 is the multi-fragment bit (MF bit), bit 2 is if fragment offset is non-zero.	0x0
174:47	ipSa_mask	Mask for ipSa.	$2^{128} - 1$



Bits	Field Name	Description	Default Value
302:175	ipSa	The IP Source Address. IPv4 is located in bits [31:0].	0x0
430:303	ipDa_mask	Mask for ipDa.	$2^{128} - 1$
558:431	ipDa	The IP Destination or MPLS Address. IPv4 is located in bits [31:0]. First MPLS bits are located at [19:0], second MPLS label [39:20], third MPLS label is [59:40] and forth label is at [79:60].	0x0
560:559	l4Type_mask	Mask for l4Type.	0x3
562:561	l4Type	The L4 type which shall be matched. If not UDP or TCP value 2 will be set in this register. 0 = TCP 1 = UDP 2 = Others 3 = Reserved.	0x0
570:563	l4Protocol_mask	Mask for l4Protocol.	0xff
578:571	l4Protocol	The L4 protocol from the IPv4 or IPv6 headers which shall be matched.	0x0
594:579	l4Sp_mask	Mask for l4Sp.	0xffff
610:595	l4Sp	L4 Source port, if packet is a TCP or UDP, otherwise set to zero.	0x0
626:611	l4Dp_mask	Mask for l4Dp.	0xffff
642:627	l4Dp	L4 destination port, if packet is a TCP or UDP, otherwise set to zero.	0x0
643	fromCpuTag_mask	Mask for fromCpuTag.	0x1
644	fromCpuTag	This packet contains a From CPU Tag. 0 = No. 1 = Yes.	0x0

35.10.192 Tunnel Exit Lookup TCAM Answer

This is the table holding the answer for the [Tunnel Exit Lookup TCAM](#).

Number of Entries : 16
 Number of Addresses per Entry : 4
 Type of Operation : Read/Write
 Addressing : [Tunnel Exit Lookup TCAM](#) hit index

Address Space

Slice 0 :	4942 to 5005
Slice 1 :	102622 to 102685
Slice 2 :	200302 to 200365
Slice 3 :	297982 to 298045

Field Description

Bits	Field Name	Description	Default Value
31:0	srcPortMask	Which source ports shall this tunnel exit be done on? The portmask which has one bit per source port. 0 = No, do not do tunnel exit 1 = Yes, if second tunnel lookup is a hit then do tunnel exit.	0x0



Bits	Field Name	Description	Default Value
32	sendToCpu	This packet shall be sent to the CPU. 0 = No. 1 = Yes.	0x0
41:33	secondShift	Second tunnel exit lookup shift to get the data for the second lookup, this value is in number of bytes, this value can at maximum be 217.	0x0
42	secondIncludeVlan	Shall second tunnel exit lookup shift be updated according to how many VLANs the packet has? 0 = No 1 = Yes	0x0
43	direct	Use direct value in this table in the Second Tunnel Exit Lookup Table Lookup. 0 = False 1 = True	0x0
123:44	key	Direct Value to use in instead of value from packet.	0x0
203:124	lookupMask	Mask for second tunnel exit lookup data. Before the lookup in the second lookup takes place this value from first lookup/packet data is AND:ed with this value.	0x0
205:204	tabIndex	Index to be used in second tunnel exit dleft lookup. This is used in conjunciton with the key extracted from this table or from packet data.	0x0

35.10.193 VLAN PCP And DEI To Color Mapping Table

Mapping table from VLAN PCP and DEI field to packet initial color.

Number of Entries : 16

Type of Operation : Read/Write

Addressing :

address[0:2] :	PCP
address[3] :	DEI
Slice 0 :	83727 to 83742
Slice 1 :	181407 to 181422
Slice 2 :	279087 to 279102
Slice 3 :	376767 to 376782

Address Space

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

35.10.194 VLAN PCP To Queue Mapping Table

Mapping table from VLAN PCP priority bits to ingress/egress queues.



Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	Incoming packets VLAN priority bits
Address Space	Slice 0 : 84263 to 84270
	Slice 1 : 181943 to 181950
	Slice 2 : 279623 to 279630
	Slice 3 : 377303 to 377310

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

35.10.195 VLAN Table

Defines the VLAN port membership, which VID to use in L2 lookups, the MSPT to use, if routing is allowed and a VLAN operation (e.g. push, pop, swap) to be performed.

The VLAN operation is selected by the [vlanSingleOp](#) field. For the push and swap operations the information used to create the new VLAN header is controlled by the fields [vidSel](#), [cfiDeiSel](#), [pcpSel](#) and [typeSel](#).

Number of Entries :	4096
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	The packet's Ingress VID plus offset as defined in Source Port Table .
Address Space	Slice 0 : 25118 to 41501
	Slice 1 : 122798 to 139181
	Slice 2 : 220478 to 236861
	Slice 3 : 318158 to 334541

Field Description

Bits	Field Name	Description	Default Value
31:0	vlanPortMask	VLAN membership portmask. The packets source port must be a member of the VLAN, otherwise the packet will be dropped and the VLAN Member Drop will be incremented. The membership mask will also limit the destination ports for L2 unicast, multicast, broadcast and flooding. If this results in an empty destination port mask then the packet is dropped and the Empty Mask Drop will be incremented.	$2^{32} - 1$
43:32	gid	The packet will be assigned a global identifier that is used during L2 lookup to allow multiple VLANs to share the same L2 tables.	0x0
44	mmpValid	If set, this entry contains a valid MMP pointer	0x0



Bits	Field Name	Description	Default Value
49:45	mmpPtr	Ingress MMP pointer.	0x0
51:50	mmpOrder	Ingress MMP pointer order.	0x0
55:52	msptPtr	The multiple spanning tree to be used by packets on this VLAN. Points to entries in the Ingress Multiple Spanning Tree State and Egress Multiple Spanning Tree State tables	0x0
58:56	vlanSingleOp	The ingress VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate Pop(remove all VLANs).	0x0
60:59	vidSel	Selects which VID to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's vid will be used. 0 = From the outermost VLAN in the original packet (if any). 1 = From this table entry's vid . 2 = From the second VLAN in the original packet (if any).	0x0
62:61	cfiDeiSel	Selects which CFI/DEI to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's cfiDei will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's cfiDei . 2 = From the second VLAN in the original packet (if any).	0x0
64:63	pcpSel	Selects which PCP to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's pcp will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's pcp . 2 = From the second VLAN in the original packet (if any).	0x0
76:65	vid	The VID used in VLAN push or swap operation if selected by vidSel .	0x0
79:77	pcp	The PCP used in VLAN push or swap operation if selected by pcpSel .	0x0
80	cfiDei	The CFI/DEI used in VLAN push or swap operation if selected by cfiDeiSel	0x0



Bits	Field Name	Description	Default Value
82:81	typeSel	<p>Selects which TPID to use when building a new VLAN header in a push or swap operation.</p> <p>0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag field typeValue.</p>	0x0
146:83	nrVlansVidOperationIf	<p>A per source port setting. Port 0 uses bits [1:0], port 2 uses bits [3:2] and so on. If the packet coming in on the source port has this amount of VLANs then this operation will override the VLAN Tables VID operation and all associated data. This operation does take into account what operation the source port VID operation performed on the packet. If a already has 2 VLANs and a push operation is done it will still be counted as a packet with two vlans. If a packet has zero vlans and a pop operation is carried out it will still have zero VLANs. Swap operations does not change the number of VLANs on the packet.</p> <p>0 = Incoming packet after source port VID op has zero VLANs 1 = Incoming packet after source port VID op has one VLAN 2 = Incoming packet after source port VID op has Two VLANs 3 = Reserved and Disabled</p>	$2^{64} - 1$
149:147	vlanSingleOpIf	<p>This operation depends on if the nrVlansVidOperationIf is done on this port. Then the default operation is overridden with this value. The ingress VLAN operation to perform on the packet.</p> <p>0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate Pop(remove all VLANs).</p>	0x0
151:150	vidSelf	<p>This operation depends on if the nrVlansVidOperationIf is done on this port. Then the default operation is overridden with this value. Selects which VID to use when building a new VLAN header in a push or swap operation. this table entry's pcp will be used.</p> <p>0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's pcp. 2 = From the second VLAN in the original packet (if any).</p>	0x0



Bits	Field Name	Description	Default Value
153:152	cfiDeiSelf	This operation depends on if the nrVlansVidOperationIf is done on this port. Selects which CFI/DEI to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's cfiDei will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's cfiDei . 2 = From the second VLAN in the original packet (if any).	0x0
155:154	pcpSelf	This operation depends on if the nrVlansVidOperationIf is done on this port. Selects which PCP to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's pcp will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's pcp . 2 = From the second VLAN in the original packet (if any).	0x0
157:156	typeSelf	This operation depends on if the nrVlansVidOperationIf is done on this port. Then the default operation is overridden with this value. Selects which TPID to use when building a new VLAN header in a push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register Egress Ethernet Type for VLAN tag field typeValue .	0x0
169:158	vidIf	If this data is used depends on if the nrVlansVidOperationIf is done on this port. Then the default operation is overridden with this value. The VID used in VLAN push or swap operation if selected by vidSel .	0x0
172:170	pcpIf	If this data is used depends on if the nrVlansVidOperationIf is done on this port. Then the default operation is overridden with this value. The PCP used in VLAN push or swap operation if selected by pcpSel .	0x0
173	cfiDeiIf	If this data is used depends on if the nrVlansVidOperationIf is done on this port. Then the default operation is overridden with this value. The CFI/DEI used in VLAN push or swap operation if selected by cfiDeiSel	0x0



Bits	Field Name	Description	Default Value
174	allowRouting	Allow routing. 0 = The router will not process the packet but L2 processing will be done normally. 1 = Packet will be processed by the router.	0x1
175	sendIpMcToCpu	Send all IPv4 and IPv6 multicast packets to CPU, bypassing L2 processing and L3 routing.	0x0

35.11 MBSC

35.11.1 L2 Broadcast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Broadcast Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 336 to 367

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x5c8

35.11.2 L2 Broadcast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Broadcast Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 368 to 399

Field Description

Bits	Field Name	Description	Default Value
15:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	0x2e4

35.11.3 L2 Broadcast Storm Control Enable

Bitmask to turn L2 Broadcast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 400



Field Description

Bits	Field Name	Description	Default Value
31:0	enable	Bitmask where the index is the Egress Ports	0x0

35.11.4 L2 Broadcast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Broadcast Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 304 to 335

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x1
12:1	tokens	The number of tokens added each tick	0x4a
15:13	tick	Select one of the eight available core ticks. The tick frequencies are configured globally in the core Tick Configuration register.	0x2
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

35.11.5 L2 Flooding Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Flooding Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 530 to 561

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x5c8

35.11.6 L2 Flooding Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Flooding Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 562 to 593



Field Description

Bits	Field Name	Description	Default Value
15:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	0x2e4

35.11.7 L2 Flooding Storm Control Enable

Bitmask to turn L2 Flooding Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 594

Field Description

Bits	Field Name	Description	Default Value
31:0	enable	Bitmask where the index is the Egress Ports	0x0

35.11.8 L2 Flooding Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Flooding Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 498 to 529

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x1
12:1	tokens	The number of tokens added each tick	0x4a
15:13	tick	Select one of the eight available core ticks. The tick frequencies are configured globally in the core Tick Configuration register.	0x2
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

35.11.9 L2 Multicast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Multicast Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 433 to 464



Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x5c8

35.11.10 L2 Multicast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Multicast Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 465 to 496

Field Description

Bits	Field Name	Description	Default Value
15:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	0x2e4

35.11.11 L2 Multicast Storm Control Enable

Bitmask to turn L2 Multicast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries : 1
 Type of Operation : Read/Write
 Address Space : 497

Field Description

Bits	Field Name	Description	Default Value
31:0	enable	Bitmask where the index is the Egress Ports	0x0

35.11.12 L2 Multicast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Multicast Storm Control

Number of Entries : 32
 Type of Operation : Read/Write
 Addressing : Egress Ports
 Address Space : 401 to 432

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x1
12:1	tokens	The number of tokens added each tick	0x4a
15:13	tick	Select one of the eight available core ticks. The tick frequencies are configured globally in the core Tick Configuration register.	0x2
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

35.12 Scheduling

35.12.1 DWRR Bucket Capacity Configuration

Token Bucket Capacity Configuration for DWRR

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress Ports

Address Space

Slice 0 :	90928 to 90935
Slice 1 :	188608 to 188615
Slice 2 :	286288 to 286295
Slice 3 :	383968 to 383975

Field Description

Bits	Field Name	Description	Default Value
17:0	bucketCapacity	Capacity of the byte bucket	$2^{18} - 1$

35.12.2 DWRR Bucket Misc Configuration

Bucket Configurations for DWRR

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress Ports

Address Space

Slice 0 :	90936 to 90943
Slice 1 :	188616 to 188623
Slice 2 :	286296 to 286303
Slice 3 :	383976 to 383983

Field Description

Bits	Field Name	Description	Default Value
4:0	threshold	When the number of bytes in any bucket goes below $2^{**}thr$, all buckets mapped to the same prio will be replenished.	0xf



Bits	Field Name	Description	Default Value
5	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:6	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode.	0x14

35.12.3 DWRR Weight Configuration

Weight Configuration for DWRR

Number of Entries : 64
 Type of Operation : Read/Write
 Addressing : Egress port * 8 + queue

Address Space	Slice 0 : 90944 to 91007
	Slice 1 : 188624 to 188687
	Slice 2 : 286304 to 286367
	Slice 3 : 383984 to 384047

Field Description

Bits	Field Name	Description	Default Value
7:0	weight	The relative weight of the queue. A queue with weight 0 is not part of the round robin scheduling but will always be selected last.	0x1

35.12.4 Map Queue to Priority

Map from egress queue to egress priority. Note that this setting must not be changed for any queue with packets queued.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space	Slice 0 : 89753 to 89760
	Slice 1 : 187433 to 187440
	Slice 2 : 285113 to 285120
	Slice 3 : 382793 to 382800

Field Description

Bits	Field Name	Description	Default Value
2:0	prio0	The priority for queue 0	0x0
5:3	prio1	The priority for queue 1	0x1
8:6	prio2	The priority for queue 2	0x2
11:9	prio3	The priority for queue 3	0x3
14:12	prio4	The priority for queue 4	0x4
17:15	prio5	The priority for queue 5	0x5
20:18	prio6	The priority for queue 6	0x6
23:21	prio7	The priority for queue 7	0x7



35.12.5 Output Disable

Bitmask for disabling the egress queues on egress ports.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space	Slice 0 :	90920 to 90927
	Slice 1 :	188600 to 188607
	Slice 2 :	286280 to 286287
	Slice 3 :	383960 to 383967

Field Description

Bits	Field Name	Description	Default Value
0	egressQueue0Disabled	If set, stop scheduling new packets for output from queue 0 on this egress port.	0x0
1	egressQueue1Disabled	If set, stop scheduling new packets for output from queue 1 on this egress port.	0x0
2	egressQueue2Disabled	If set, stop scheduling new packets for output from queue 2 on this egress port.	0x0
3	egressQueue3Disabled	If set, stop scheduling new packets for output from queue 3 on this egress port.	0x0
4	egressQueue4Disabled	If set, stop scheduling new packets for output from queue 4 on this egress port.	0x0
5	egressQueue5Disabled	If set, stop scheduling new packets for output from queue 5 on this egress port.	0x0
6	egressQueue6Disabled	If set, stop scheduling new packets for output from queue 6 on this egress port.	0x0
7	egressQueue7Disabled	If set, stop scheduling new packets for output from queue 7 on this egress port.	0x0

35.13 Shapers

35.13.1 Port Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Port Shaper

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress Port

Address Space	Slice 0 :	91402 to 91409
	Slice 1 :	189082 to 189089
	Slice 2 :	286762 to 286769
	Slice 3 :	384442 to 384449



Field Description

Bits	Field Name	Description	Default Value
16:0	bucketCapacity	Capacity of the token bucket	0x927c

35.13.2 Port Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Port Shaper

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress Port

Address Space

Slice 0 :	91410 to 91417
Slice 1 :	189090 to 189097
Slice 2 :	286770 to 286777
Slice 3 :	384450 to 384457

Field Description

Bits	Field Name	Description	Default Value
16:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	0x30d4

35.13.3 Port Shaper Enable

Bitmask to turn Port Shaper ON/OFF (1/0) for Egress Port

Number of Entries : 1
 Type of Operation : Read/Write

Address Space

Slice 0 :	91418
Slice 1 :	189098
Slice 2 :	286778
Slice 3 :	384458

Field Description

Bits	Field Name	Description	Default Value
7:0	enable	Bitmask where the index is the Egress Port	0x0

35.13.4 Port Shaper Rate Configuration

Token Bucket rate Configuration for Port Shaper



Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space	Slice 0 : 91394 to 91401
	Slice 1 : 189074 to 189081
	Slice 2 : 286754 to 286761
	Slice 3 : 384434 to 384441

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:1	tokens	The number of tokens added each tick	0x4e2
16:14	tick	Select one of the eight available core ticks. The tick frequencies are configured globally in the core Tick Configuration register.	0x0
24:17	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

35.13.5 Prio Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Prio Shaper

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Prio
Address Space	Slice 0 : 91265 to 91328
	Slice 1 : 188945 to 189008
	Slice 2 : 286625 to 286688
	Slice 3 : 384305 to 384368

Field Description

Bits	Field Name	Description	Default Value
16:0	bucketCapacity	Capacity of the token bucket	0x927c

35.13.6 Prio Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Prio Shaper

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Prio
Address Space	Slice 0 : 91329 to 91392
	Slice 1 : 189009 to 189072
	Slice 2 : 286689 to 286752
	Slice 3 : 384369 to 384432



Field Description

Bits	Field Name	Description	Default Value
16:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	0x30d4

35.13.7 Prio Shaper Enable

Bitmask to turn Prio Shaper ON/OFF (1/0) for Egress Port * 8 + Egress Prio

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	91393
Slice 1 :	189073
Slice 2 :	286753
Slice 3 :	384433

Field Description

Bits	Field Name	Description	Default Value
63:0	enable	Bitmask where the index is the Egress Port * 8 + Egress Prio	0x0

35.13.8 Prio Shaper Rate Configuration

Token Bucket rate Configuration for Prio Shaper

Number of Entries : 64

Type of Operation : Read/Write

Addressing : Egress Port * 8 + Egress Prio

Address Space

Slice 0 :	91201 to 91264
Slice 1 :	188881 to 188944
Slice 2 :	286561 to 286624
Slice 3 :	384241 to 384304

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:1	tokens	The number of tokens added each tick	0x4e2
16:14	tick	Select one of the eight available core ticks. The tick frequencies are configured globally in the core Tick Configuration register.	0x0
24:17	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18



35.13.9 Queue Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Queue Shaper

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Queue
Address Space	Slice 0 : 91072 to 91135
	Slice 1 : 188752 to 188815
	Slice 2 : 286432 to 286495
	Slice 3 : 384112 to 384175

Field Description

Bits	Field Name	Description	Default Value
16:0	bucketCapacity	Capacity of the token bucket	0x927c

35.13.10 Queue Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Queue Shaper

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Queue
Address Space	Slice 0 : 91136 to 91199
	Slice 1 : 188816 to 188879
	Slice 2 : 286496 to 286559
	Slice 3 : 384176 to 384239

Field Description

Bits	Field Name	Description	Default Value
16:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	0x30d4

35.13.11 Queue Shaper Enable

Bitmask to turn Queue Shaper ON/OFF (1/0) for Egress Port * 8 + Egress Queue

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 91200
	Slice 1 : 188880
	Slice 2 : 286560
	Slice 3 : 384240

Field Description



Bits	Field Name	Description	Default Value
63:0	enable	Bitmask where the index is the Egress Port * 8 + Egress Queue	0x0

35.13.12 Queue Shaper Rate Configuration

Token Bucket rate Configuration for Queue Shaper

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Queue
Address Space	Slice 0 : 91008 to 91071
	Slice 1 : 188688 to 188751
	Slice 2 : 286368 to 286431
	Slice 3 : 384048 to 384111

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:1	tokens	The number of tokens added each tick	0x4e2
16:14	tick	Select one of the eight available core ticks. The tick frequencies are configured globally in the core Tick Configuration register.	0x0
24:17	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

35.14 Shared Buffer Memory

35.14.1 Buffer Free

The number of cells available in the buffer memory for incoming packets.

Number of Entries :	1
Type of Operation :	Read Only
Address Space :	1

Field Description

Bits	Field Name	Description	Default Value
14:0	cells	Number of free cells.	0x4010



35.14.2 Egress Port Depth

Number of packets available in the buffer memory for each egress port.

Number of Entries : 8

Type of Operation : Read Only

Addressing : Egress Port

Address Space

Slice 0 to 0 :	90631 to 90638
Slice 1 to 0 :	90703 to 90710
Slice 2 to 0 :	90775 to 90782
Slice 3 to 0 :	90847 to 90854
Slice 0 to 1 :	188311 to 188318
Slice 1 to 1 :	188383 to 188390
Slice 2 to 1 :	188455 to 188462
Slice 3 to 1 :	188527 to 188534
Slice 0 to 2 :	285991 to 285998
Slice 1 to 2 :	286063 to 286070
Slice 2 to 2 :	286135 to 286142
Slice 3 to 2 :	286207 to 286214
Slice 0 to 3 :	383671 to 383678
Slice 1 to 3 :	383743 to 383750
Slice 2 to 3 :	383815 to 383822
Slice 3 to 3 :	383887 to 383894

Field Description

Bits	Field Name	Description	Default Value
12:0	packets	Number of packet currently queued.	0x0

35.14.3 Egress Queue Depth

Number of packets available in the buffer memory for each egress queue.

Number of Entries : 64

Type of Operation : Read Only

Addressing : Slice-global queue number

Address Space

Slice 0 to 0 :	90639 to 90702
Slice 1 to 0 :	90711 to 90774
Slice 2 to 0 :	90783 to 90846
Slice 3 to 0 :	90855 to 90918
Slice 0 to 1 :	188319 to 188382
Slice 1 to 1 :	188391 to 188454
Slice 2 to 1 :	188463 to 188526
Slice 3 to 1 :	188535 to 188598
Slice 0 to 2 :	285999 to 286062
Slice 1 to 2 :	286071 to 286134
Slice 2 to 2 :	286143 to 286206
Slice 3 to 2 :	286215 to 286278
Slice 0 to 3 :	383679 to 383742
Slice 1 to 3 :	383751 to 383814
Slice 2 to 3 :	383823 to 383886
Slice 3 to 3 :	383895 to 383958



Field Description

Bits	Field Name	Description	Default Value
12:0	packets	Number of packets currently queued.	0x0

35.14.4 Minimum Buffer Free

Minimum number of cells available in the buffer memory

Number of Entries : 1

Type of Operation : Read Only

Address Space

Slice 0 :	90919
Slice 1 :	188599
Slice 2 :	286279
Slice 3 :	383959

Field Description

Bits	Field Name	Description	Default Value
12:0	cells	Number of cells.	0x1004

35.14.5 Packet Buffer Status

Queue status of the packet buffer

Number of Entries : 1

Type of Operation : Read Only

Address Space

Slice 0 :	89750
Slice 1 :	187430
Slice 2 :	285110
Slice 3 :	382790

Field Description

Bits	Field Name	Description	Default Value
7:0	empty	Empty flags for the egress ports	0xff

35.15 Statistics: ACL**35.15.1 Egress Configurable ACL Match Counter**

Number of packets hit in entries from Egress configurable ACL lookup.



Number of Entries : 64
 Type of Operation : Read/Write
 Addressing : Index from result of Egress configurable ACL.

Address Space	Slice 0 : 89082 to 89145
	Slice 1 : 186762 to 186825
	Slice 2 : 284442 to 284505
	Slice 3 : 382122 to 382185

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.15.2 Ingress Configurable ACL Match Counter

Number of packets hit in entries from Ingress configurable ACL lookup.

Number of Entries : 64
 Type of Operation : Read/Write
 Addressing : Index from result of Ingress configurable ACL.

Address Space	Slice 0 : 87990 to 88053
	Slice 1 : 185670 to 185733
	Slice 2 : 283350 to 283413
	Slice 3 : 381030 to 381093

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.16 Statistics: Debug

35.16.1 Debug EPP Counter

Number of packets hit in entries from Debug points in EPP.

Number of Entries : 15
 Type of Operation : Read/Write
 Addressing : Epp Debug Counter table.

Address Space	Slice 0 : 102424 to 102438
	Slice 1 : 200104 to 200118
	Slice 2 : 297784 to 297798
	Slice 3 : 395464 to 395478

Field Description



Bits	Field Name	Description	Default Value
15:0	packets	Number of packets.	0x0

35.16.2 Debug IPP Counter

Number of packets hit in entries from Debug points in IPP.

Number of Entries : 23
 Type of Operation : Read/Write
 Addressing : lpp Debug Counter table.

Address Space	Slice 0 :	89466 to 89488
	Slice 1 :	187146 to 187168
	Slice 2 :	284826 to 284848
	Slice 3 :	382506 to 382528

Field Description

Bits	Field Name	Description	Default Value
15:0	packets	Number of packets.	0x0

35.16.3 EPP PM Drop

Number of drops due to FIFO overflows in EPP PM.

In Figure 29.1, **epmOverflow** with process sequence 22 represents the internal location of this counter.

Number of Entries : 1
 Type of Operation : Read/Write

Address Space	Slice 0 :	91499
	Slice 1 :	189179
	Slice 2 :	286859
	Slice 3 :	384539

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.16.4 IPP PM Drop

Number of drops due to FIFO overflows in IPP PM.

In Figure 29.1, **ipmOverflow** with process sequence 12 represents the internal location of this counter.



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4896
	Slice 1 : 102576
	Slice 2 : 200256
	Slice 3 : 297936

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.16.5 PS Error Counter

Number of errors occurred in the PS-converter.

In Figure 29.1, **psError** with process sequence 25 represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read Only
Addressing :	Egress port
Address Space	Slice 0 : 102480 to 102487
	Slice 1 : 200160 to 200167
	Slice 2 : 297840 to 297847
	Slice 3 : 395520 to 395527

Field Description

Bits	Field Name	Description	Default Value
31:0	underrun	Number of packets which have empty cycles caused by the internal PS-converter but not the external halt during packet transmissions.	0x0
63:32	overflow	Number of FIFO overflows in the PS-converter. This error will cause packet corruptions.	0x0

35.16.6 SP Overflow Drop

Number of packets dropped due to: FIFO overflow in the SP-converter.

In Figure 29.1, **spOverflow** with process sequence 5 represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read Only
Addressing :	Ingress port
Address Space	Slice 0 : 4848 to 4855
	Slice 1 : 102528 to 102535
	Slice 2 : 200208 to 200215
	Slice 3 : 297888 to 297895



Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets on this ingress port.	0x0

35.17 Statistics: EPP Egress Port Drop**35.17.1 Egress Port Disabled Drop**

Number of packets dropped due to egress port disabled. The addressing of this table continues between slices, which means the first entry in a higher slice number should be appended to the last entry of the previous slice to get the correct table index.

In Figure 29.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space

Slice 0 :	91475 to 91482
Slice 1 :	189155 to 189162
Slice 2 :	286835 to 286842
Slice 3 :	384515 to 384522

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.17.2 Egress Port Filtering Drop

Number of packets dropped due to egress port filtering. The addressing of this table continues between slices, which means the first entry in a higher slice number should be appended to the last entry of the previous slice to get the correct table index.

In Figure 29.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : Egress port

Address Space

Slice 0 :	91483 to 91490
Slice 1 :	189163 to 189170
Slice 2 :	286843 to 286850
Slice 3 :	384523 to 384530

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.17.3 Tunnel Exit Too Small Packet Modification To Small Drop

The packet modification after the tunnel exit resulted in a packet size that was less than zero. The addressing of this table continues between slices, which means the first entry in a higher slice number should be appended to the last entry of the previous slice to get the correct table index.

In Figure 29.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space	Slice 0 : 91491 to 91498
	Slice 1 : 189171 to 189178
	Slice 2 : 286851 to 286858
	Slice 3 : 384531 to 384538

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.17.4 Unknown Egress Drop

Number of packets dropped during egress packet processing due to unknown reasons. Internal error caused by packet drop with an invalid Drop ID. The addressing of this table continues between slices, which means the first entry in a higher slice number should be appended to the last entry of the previous slice to get the correct table index.

In Figure 29.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space	Slice 0 : 91467 to 91474
	Slice 1 : 189147 to 189154
	Slice 2 : 286827 to 286834
	Slice 3 : 384507 to 384514

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.18 Statistics: IPP Egress Port Drop

35.18.1 Egress Spanning Tree Drop

Number of packets dropped due to egress spanning tree check configured in **Egress Spanning Tree State** and **Egress Multiple Spanning Tree State**

In Figure 29.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.



Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space	Slice 0 : 89178 to 89209
	Slice 1 : 186858 to 186889
	Slice 2 : 284538 to 284569
	Slice 3 : 382218 to 382249

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.18.2 Ingress-Egress Packet Filtering Drop

Number of packets dropped due to ingress-egress packet filtering configured in [Ingress Egress Port Packet Type Filter](#).

In Figure 29.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space	Slice 0 : 89242 to 89273
	Slice 1 : 186922 to 186953
	Slice 2 : 284602 to 284633
	Slice 3 : 382282 to 382313

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.18.3 L2 Action Table Per Port Drop

Number of packets dropped due to L2 Action Table per egress port drop configured in [L2 Action Table Drop](#).

In Figure 29.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space	Slice 0 : 89274 to 89305
	Slice 1 : 186954 to 186985
	Slice 2 : 284634 to 284665
	Slice 3 : 382314 to 382345

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.18.4 MBSC Drop

Number of packets dropped due to MBSC. When the egress port exceeds the multicast/broadcast traffic limits any multicast/broadcast packets will be dropped.

In Figure 29.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space	Slice 0 : 89210 to 89241
	Slice 1 : 186890 to 186921
	Slice 2 : 284570 to 284601
	Slice 3 : 382250 to 382281

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.18.5 Queue Off Drop

Number of packets dropped due to the queue being turned off.

In Figure 29.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space	Slice 0 : 89146 to 89177
	Slice 1 : 186826 to 186857
	Slice 2 : 284506 to 284537
	Slice 3 : 382186 to 382217

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19 Statistics: IPP Ingress Port Drop

35.19.1 AH Decoder Drop

Number of packets dropped due to setting in register **AH Header Packet Decoder Options**.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4928
	Slice 1 : 102608
	Slice 2 : 200288
	Slice 3 : 297968

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.2 ARP Decoder Drop

Number of packets dropped due to setting in register [ARP Packet Decoder Options](#).
 In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4921
	Slice 1 : 102601
	Slice 2 : 200281
	Slice 3 : 297961

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.3 BOOTP and DHCP Decoder Drop

Number of packets dropped due to setting in register [BOOTP and DHCP Packet Decoder Options](#).
 In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4931
	Slice 1 : 102611
	Slice 2 : 200291
	Slice 3 : 297971

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.19.4 CAPWAP Decoder Drop

Number of packets dropped due to setting in register [CAPWAP Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4932
	Slice 1 : 102612
	Slice 2 : 200292
	Slice 3 : 297972

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.5 DNS Decoder Drop

Number of packets dropped due to setting in register [DNS Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4930
	Slice 1 : 102610
	Slice 2 : 200290
	Slice 3 : 297970

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.6 ESP Decoder Drop

Number of packets dropped due to setting in register [ESP Header Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4929
	Slice 1 : 102609
	Slice 2 : 200289
	Slice 3 : 297969

Field Description



Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.7 Egress Configurable ACL Drop

Number of packets dropped due to matching an Egress Configurable ACL with drop.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4920
Slice 1 :	102600
Slice 2 :	200280
Slice 3 :	297960

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.8 Empty Mask Drop

Number of packets dropped due to an empty destination port mask.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4899
Slice 1 :	102579
Slice 2 :	200259
Slice 3 :	297939

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.9 Expired TTL Drop

Number of packets dropped due to expired TTL.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4911
	Slice 1 : 102591
	Slice 2 : 200271
	Slice 3 : 297951

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.10 GRE Decoder Drop

Number of packets dropped due to setting in register [GRE Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4934
	Slice 1 : 102614
	Slice 2 : 200294
	Slice 3 : 297974

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.11 IEEE 802.1X and EAPOL Decoder Drop

Number of packets dropped due to setting in register [IEEE 802.1X and EAPOL Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4925
	Slice 1 : 102605
	Slice 2 : 200285
	Slice 3 : 297965

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.19.12 IKE Decoder Drop

Number of packets dropped due to setting in register **IKE Packet Decoder Options**.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4933
	Slice 1 : 102613
	Slice 2 : 200293
	Slice 3 : 297973

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.13 IP Checksum Drop

Number of packets dropped due to incorrect IP checksum.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4913
	Slice 1 : 102593
	Slice 2 : 200273
	Slice 3 : 297953

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.14 Ingress Configurable ACL Drop

Number of packets dropped due to matching an Ingress Configurable ACL with drop.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4919
	Slice 1 : 102599
	Slice 2 : 200279
	Slice 3 : 297959

Field Description



Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.15 Ingress Packet Filtering Drop

Number of packets dropped due to ingress port packet type filtering as configured in [Ingress Port Packet Type Filter](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4904
Slice 1 :	102584
Slice 2 :	200264
Slice 3 :	297944

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.16 Ingress Spanning Tree Drop: Blocking

Number of packets dropped due to that a ports's ingress spanning tree protocol state was **Blocking** or that port and packet VLAN's ingress multiple spanning tree instance state was **Discarding**.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4902
Slice 1 :	102582
Slice 2 :	200262
Slice 3 :	297942

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.17 Ingress Spanning Tree Drop: Learning

Number of packets dropped due to that a port's ingress spanning tree protocol state was **Learning** or that port and packet VLAN's ingress multiple spanning tree instance state was **Learning**.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4901
	Slice 1 : 102581
	Slice 2 : 200261
	Slice 3 : 297941

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.18 Ingress Spanning Tree Drop: Listen

Number of packets dropped due to that a port's ingress spanning tree protocol state was **Listening**. In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4900
	Slice 1 : 102580
	Slice 2 : 200260
	Slice 3 : 297940

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.19 Invalid Routing Protocol Drop

Number of packets dropped due to invalid routing protocol. This occurs when a packet enters the router port but the protocol type is not allowed to be routed as configured in **Ingress Router Table**.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4910
	Slice 1 : 102590
	Slice 2 : 200270
	Slice 3 : 297950

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.19.20 L2 Action Table Drop

Number of packets dropped due to the **L2 Action Table** says drop all instances.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4937
	Slice 1 : 102617
	Slice 2 : 200297
	Slice 3 : 297977

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.21 L2 Action Table Port Move Drop

Number of packets dropped due to the **L2 Action Table** says drop due to port move packet.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4938
	Slice 1 : 102618
	Slice 2 : 200298
	Slice 3 : 297978

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.22 L2 Action Table Special Packet Type Drop

Number of packets dropped due to the **Allow Special Frame Check For L2 Action Table** dit not allow a certain packet/frame type.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4936
	Slice 1 : 102616
	Slice 2 : 200296
	Slice 3 : 297976



Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.23 L2 IEEE 1588 Decoder Drop

Number of packets dropped due to setting in register [IEEE 1588 L4 Packet Decoder Options](#). In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4923
Slice 1 :	102603
Slice 2 :	200283
Slice 3 :	297963

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.24 L2 Lookup Drop

Number of packets dropped in the L2 destination port lookup process. Either due to a drop flag in an [L2 Destination Table](#) entry, or due to destination port not being member of the VLAN or due to not allowing destination port being the same as the source port.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4903
Slice 1 :	102583
Slice 2 :	200263
Slice 3 :	297943

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.19.25 L2 Reserved Multicast Address Drop

Number of packets dropped due to the L2 Reserved Multicast Addresses on counter 0

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4918
	Slice 1 : 102598
	Slice 2 : 200278
	Slice 3 : 297958

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.26 L3 Lookup Drop

Number of packets dropped due to a drop flag in **L3 Routing Default** or **Next Hop Table**.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4912
	Slice 1 : 102592
	Slice 2 : 200272
	Slice 3 : 297952

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.27 L4 IEEE 1588 Decoder Drop

Number of packets dropped due to setting in register **IEEE 1588 L4 Packet Decoder Options**.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4924
	Slice 1 : 102604
	Slice 2 : 200284
	Slice 3 : 297964

Field Description



Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.28 LACP Decoder Drop

Number of packets dropped due to setting in register [LACP Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4927
	Slice 1 : 102607
	Slice 2 : 200287
	Slice 3 : 297967

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.29 Learning Packet Drop

Number of learning packets dropped. After learning information is extracted all learning packets are dropped.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4917
	Slice 1 : 102597
	Slice 2 : 200277
	Slice 3 : 297957

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.30 Maximum Allowed VLAN Drop

Number of packets dropped due to too many VLAN tags. Packets are dropped if number of VLANS is above the limit setup in the [Source Port Table](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4909
	Slice 1 : 102589
	Slice 2 : 200269
	Slice 3 : 297949

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.31 Minimum Allowed VLAN Drop

Number of packets dropped due to insufficient VLAN tags. Packets are dropped if number of VLANS is below the limit setup in the [Source Port Table](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4908
	Slice 1 : 102588
	Slice 2 : 200268
	Slice 3 : 297948

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.32 NAT Action Table Drop

Number of packets dropped due to the [NAT Action Table](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4935
	Slice 1 : 102615
	Slice 2 : 200295
	Slice 3 : 297975

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.19.33 RARP Decoder Drop

Number of packets dropped due to setting in register [RARP Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4922
	Slice 1 : 102602
	Slice 2 : 200282
	Slice 3 : 297962

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.34 Reserved MAC DA Drop

Number of packets dropped due to the packets destination MAC address match a [Reserved Destination MAC Address Range](#) that is configured to be dropped.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4905
	Slice 1 : 102585
	Slice 2 : 200265
	Slice 3 : 297945

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.35 Reserved MAC SA Drop

Number of packets dropped due to the packets source MAC address match a [Reserved Source MAC Address Range](#) that is configured to be dropped.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4906
	Slice 1 : 102586
	Slice 2 : 200266
	Slice 3 : 297946



Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.36 SCTP Decoder Drop

Number of packets dropped due to setting in register [SCTP Packet Decoder Options](#).

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4926
Slice 1 :	102606
Slice 2 :	200286
Slice 3 :	297966

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.37 Second Tunnel Exit Drop

Number of packets dropped due to second tunnel exit lookup says drop packet.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4914
Slice 1 :	102594
Slice 2 :	200274
Slice 3 :	297954

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.38 Source Port Default ACL Action Drop

Number of packets dropped due to the table [Source Port Default ACL Action](#) says drop.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4939
	Slice 1 : 102619
	Slice 2 : 200299
	Slice 3 : 297979

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.39 Tunnel Exit Miss Action Drop

Number of packets dropped due to second tunnel exit lookup was a miss while the tunnel exit table [Second Tunnel Exit Miss Action](#) says that the second tunnel table must be a hit.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4915
	Slice 1 : 102595
	Slice 2 : 200275
	Slice 3 : 297955

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.40 Tunnel Exit Too Small Packet Modification Drop

The packet modification after the tunnel exit resulted in a packet size that was less than zero.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4916
	Slice 1 : 102596
	Slice 2 : 200276
	Slice 3 : 297956

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.19.41 Unknown Ingress Drop

Number of packets dropped during ingress packet processing due to unknown reasons. Internal error caused by packet drop with an invalid Drop ID.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4898
	Slice 1 : 102578
	Slice 2 : 200258
	Slice 3 : 297938

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.19.42 VLAN Member Drop

Number of packets dropped due to the packets source port not being part of the packets VLAN membership.

In Figure 29.1, **ippDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4907
	Slice 1 : 102587
	Slice 2 : 200267
	Slice 3 : 297947

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.20 Statistics: IPP Ingress Port Receive

35.20.1 IP Multicast ACL Drop Counter

Number of IP multicast packets received and hit in ACL drop rules. IP multicast packets are counted for IPv4 packets with destination MAC in range 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff or IPv6 packets with destination MAC matches 33:33:xx:xx:xx:xx.

In Figure 29.1, **ip** with process sequence **11** represents the internal location of this counter.



Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space	Slice 0 : 89434 to 89465
	Slice 1 : 187114 to 187145
	Slice 2 : 284794 to 284825
	Slice 3 : 382474 to 382505

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.20.2 IP Multicast Received Counter

Number of IP multicast packets received on ingress. IP multicast packets are counted for IPv4 packets with destination MAC in range 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff or IPv6 packets with destination MAC matches 33:33:xx:xx:xx:xx.

In Figure 29.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space	Slice 0 : 89338 to 89369
	Slice 1 : 187018 to 187049
	Slice 2 : 284698 to 284729
	Slice 3 : 382378 to 382409

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.20.3 IP Multicast Routed Counter

Number of IP multicast packets received and routed on ingress. IP multicast packets are counted for IPv4 packets with destination MAC in range 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff or IPv6 packets with destination MAC matches 33:33:xx:xx:xx:xx.

In Figure 29.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space	Slice 0 : 89402 to 89433
	Slice 1 : 187082 to 187113
	Slice 2 : 284762 to 284793
	Slice 3 : 382442 to 382473



Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.20.4 IP Unicast Received Counter

Number of IP unicast packets received on ingress. Any IP packet with destination MAC not in IP multicast range (01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff for IPv4 and 33:33:xx:xx:xx:xx for IPv6) are counted as IP unicast packets.

In Figure 29.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries : 32

Type of Operation : Read/Write

Addressing : Ingress port

Address Space

Slice 0 :	89306 to 89337
Slice 1 :	186986 to 187017
Slice 2 :	284666 to 284697
Slice 3 :	382346 to 382377

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.20.5 IP Unicast Routed Counter

Number of IP unicast packets received and routed on ingress. Any IP packet with destination MAC not in IP multicast range (01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff for IPv4 and 33:33:xx:xx:xx:xx for IPv6) are counted as IP unicast packets.

In Figure 29.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries : 32

Type of Operation : Read/Write

Addressing : Ingress port

Address Space

Slice 0 :	89370 to 89401
Slice 1 :	187050 to 187081
Slice 2 :	284730 to 284761
Slice 3 :	382410 to 382441

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0



35.21 Statistics: Misc

35.21.1 Buffer Overflow Drop

Counter for the number of packets dropped due to the shared buffer memory being full.

In Figure 29.1, **bmOverflow** with process sequence **16** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 89751
	Slice 1 : 187431
	Slice 2 : 285111
	Slice 3 : 382791

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.21.2 Drain Port Drop

Number of packets dropped due to the port is drained.

In Figure 29.1, **drain** with process sequence **21** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space	Slice 0 : 91459 to 91466
	Slice 1 : 189139 to 189146
	Slice 2 : 286819 to 286826
	Slice 3 : 384499 to 384506

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.21.3 Egress Resource Manager Drop

Number of packets dropped by the egress resource manager.

In Figure 29.1, **erm** with process sequence **15** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space	Slice 0 : 89742 to 89749
	Slice 1 : 187422 to 187429
	Slice 2 : 285102 to 285109
	Slice 3 : 382782 to 382789



Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.21.4 Flow Classification And Metering Drop

Number of packets dropped due to flow classification and metering.

In Figure 29.1, **mmp** with process sequence **14** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	89489
Slice 1 :	187169
Slice 2 :	284849
Slice 3 :	382529

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.21.5 IPP Empty Destination Drop

Number of drops due to the determined destination is cleared during post-ingress packet processing and causing no cell to be enqueued in the buffer memory. This happens on single cell packet with end-of-packet drop actions.

In Figure 29.1, **eopDrop** with process sequence **14** represents the internal location of this counter.

Number of Entries : 1

Type of Operation : Read/Write

Address Space

Slice 0 :	4897
Slice 1 :	102577
Slice 2 :	200257
Slice 3 :	297937

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0



35.21.6 Ingress Resource Manager Drop

Counter for the number of packets dropped due to exceeding thresholds set up in the ingress resource manager.

In Figure 29.1, **irm** with process sequence **16** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 89752
	Slice 1 : 187432
	Slice 2 : 285112
	Slice 3 : 382792

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

35.21.7 MAC RX Broken Packets

Number of broken packets dropped (packets with last=1 and valid_bytes=0).

In Figure 29.1, **macBrokenPkt** with process sequence **3** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read Only (unreliable)
Addressing :	Ingress Port
Address Space :	112 to 143

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.21.8 MAC RX Long Packet Drop

Number of packets dropped due to length above **MAC RX Maximum Packet Length**.

In Figure 29.1, **macRxMax** with process sequence **4** represents the internal location of this counter.

Number of Entries :	32
Type of Operation :	Read Only (unreliable)
Addressing :	Ingress Port
Address Space :	176 to 207

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0



35.21.9 MAC RX Short Packet Drop

Number of packets dropped due to length below 60 bytes.

In Figure 29.1, **macRxMin** with process sequence 4 represents the internal location of this counter.

Number of Entries : 32
 Type of Operation : Read Only (**unreliable**)
 Addressing : Ingress Port
 Address Space : 144 to 175

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.21.10 Re-queue Overflow Drop

Counter for the number of packets dropped due to a FIFO overflow in re-queue.

In Figure 29.1, **rqOverflow** with process sequence 24 represents the internal location of this counter.

Number of Entries : 1
 Type of Operation : Read/Write

Slice 0 :	89761
Slice 1 :	187441
Slice 2 :	285121
Slice 3 :	382801

Address Space

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets	0x0

35.22 Statistics: NAT

35.22.1 Egress NAT Hit Status

Status bit is set if there was a hit in the **Egress NAT Operation**.

In Figure 29.1, **nat** with process sequence 19 represents the internal location of this counter.

Number of Entries : 1024
 Type of Operation : Read/Write
 Addressing : Egress NAT pointer + Egress Port

Slice 0 :	101400 to 102423
Slice 1 :	199080 to 200103
Slice 2 :	296760 to 297783
Slice 3 :	394440 to 395463

Address Space



Field Description

Bits	Field Name	Description	Default Value
0	hit	If set, the corresponding entry in the Egress NAT Operation is hit.	0x0

35.22.2 Ingress NAT Hit Status

Status bit is set if there was a hit in the [Ingress NAT Operation](#).

In Figure 29.1, **nat** with process sequence **19** represents the internal location of this counter.

Number of Entries :	2048
Type of Operation :	Read/Write
Addressing :	Ingress NAT pointer + Egress Port
Address Space	Slice 0 : 99352 to 101399
	Slice 1 : 197032 to 199079
	Slice 2 : 294712 to 296759
	Slice 3 : 392392 to 394439

Field Description

Bits	Field Name	Description	Default Value
0	hit	If set, the corresponding entry in the Ingress NAT Operation is hit.	0x0

35.23 Statistics: Packet Datapath**35.23.1 EPP Packet Head Counter**

Number of packet first cells through the Egress Packet Process module.

In Figure 29.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 91500
	Slice 1 : 189180
	Slice 2 : 286860
	Slice 3 : 384540

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0



35.23.2 EPP Packet Tail Counter

Number of packet last cells through the Egress Packet Process module.

In Figure 29.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 91501
	Slice 1 : 189181
	Slice 2 : 286861
	Slice 3 : 384541

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

35.23.3 IPP Packet Head Counter

Number of packet first cells through the Ingress Packet Process module.

In Figure 29.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4940
	Slice 1 : 102620
	Slice 2 : 200300
	Slice 3 : 297980

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0

35.23.4 IPP Packet Tail Counter

Number of packet last cells through the Ingress Packet Process module.

In Figure 29.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 4941
	Slice 1 : 102621
	Slice 2 : 200301
	Slice 3 : 297981

Field Description



Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

35.23.5 MAC Interface Counters For RX

Counters for the interface protocol checkers. The counters wrap.

In Figure 29.1, **rxIf** with process sequence **1** represents the internal location of this counter.

Number of Entries : 32
 Type of Operation : Read Only (**unreliable**)
 Addressing : Ingress Port
 Address Space : 48 to 79

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Correct packets completed	0x0
63:32	error	Bus protocol errors.	0x0

35.23.6 MAC Interface Counters For TX

Counters for the interface protocol checkers. The counters wrap.

In Figure 29.1, **txIf** with process sequence **28** represents the internal location of this counter.

Number of Entries : 32
 Number of Addresses per Entry : 2
 Type of Operation : Read Only
 Addressing : Egress Port
 Address Space : 208 to 271

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Correct packets completed	0x0
63:32	error	Bus protocol errors.	0x0
95:64	halt	Halt errors. Incremented if first, last or valid_bytes is non-zero when halt is high.	0x0

35.23.7 PB Packet Head Counter

Number of packet first cells through the Shared Buffer Memory module.

In Figure 29.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.



Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 91456
	Slice 1 : 189136
	Slice 2 : 286816
	Slice 3 : 384496

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0

35.23.8 PB Packet Tail Counter

Number of packet last cells through the Shared Buffer Memory module.

In Figure 29.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 91457
	Slice 1 : 189137
	Slice 2 : 286817
	Slice 3 : 384497

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

35.23.9 PS Packet Head Counter

Number of packet first cells through the Parallel to Serial module.

In Figure 29.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 102488
	Slice 1 : 200168
	Slice 2 : 297848
	Slice 3 : 395528

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0



35.23.10 PS Packet Tail Counter

Number of packet last cells through the Parallel to Serial module.

In Figure 29.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space	Slice 0 : 102489
	Slice 1 : 200169
	Slice 2 : 297849
	Slice 3 : 395529

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

35.24 Statistics: Routing

35.24.1 Next Hop Hit Status

Status bit is set if a packet was routed using the corresponding entry in the [Next Hop Table](#).

In Figure 29.1, **nextHop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1024
Type of Operation :	Read/Write
Addressing :	Next Hop
Address Space	Slice 0 : 88058 to 89081
	Slice 1 : 185738 to 186761
	Slice 2 : 283418 to 284441
	Slice 3 : 381098 to 382121

Field Description

Bits	Field Name	Description	Default Value
0	ipv4	The next hop entry was hit with an IPv4 packet.	0x0
1	ipv6	The next hop entry was hit with an IPv6 packet.	0x0
2	mpls	The next hop entry was hit with an MPLS packet.	0x0

35.24.2 Received Packets on Ingress VRF

Number of packets enter a VRF on ingress.

In Figure 29.1, **vrfIn** with process sequence **11** represents the internal location of this counter.



Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : vrf

Address Space

Slice 0 :	88054 to 88057
Slice 1 :	185734 to 185737
Slice 2 :	283414 to 283417
Slice 3 :	381094 to 381097

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.24.3 Transmitted Packets on Egress VRF

Number of packets leave a VRF on egress.

In Figure 29.1, **vrfOut** with process sequence 19 represents the internal location of this counter.

Number of Entries : 4
 Type of Operation : Read/Write
 Addressing : vrf

Address Space

Slice 0 :	99348 to 99351
Slice 1 :	197028 to 197031
Slice 2 :	294708 to 294711
Slice 3 :	392388 to 392391

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.25 Statistics: SMON

35.25.1 SMON Set 0 Byte Counter

Number of bytes counted in SMON Set 0.

In Figure 29.1, **smon** with process sequence 11 represents the internal location of this counter.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : VLAN PCP

Address Space

Slice 0 :	87958 to 87965
Slice 1 :	185638 to 185645
Slice 2 :	283318 to 283325
Slice 3 :	380998 to 381005



Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

35.25.2 SMON Set 0 Packet Counter

Number of packets counted in SMON Set 0.

In Figure 29.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : VLAN PCP

Address Space

Slice 0 :	87926 to 87933
Slice 1 :	185606 to 185613
Slice 2 :	283286 to 283293
Slice 3 :	380966 to 380973

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.25.3 SMON Set 1 Byte Counter

Number of bytes counted in SMON Set 1.

In Figure 29.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : VLAN PCP

Address Space

Slice 0 :	87966 to 87973
Slice 1 :	185646 to 185653
Slice 2 :	283326 to 283333
Slice 3 :	381006 to 381013

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0



35.25.4 SMON Set 1 Packet Counter

Number of packets counted in SMON Set 1.

In Figure 29.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space	Slice 0 : 87934 to 87941
	Slice 1 : 185614 to 185621
	Slice 2 : 283294 to 283301
	Slice 3 : 380974 to 380981

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.25.5 SMON Set 2 Byte Counter

Number of bytes counted in SMON Set 2.

In Figure 29.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space	Slice 0 : 87974 to 87981
	Slice 1 : 185654 to 185661
	Slice 2 : 283334 to 283341
	Slice 3 : 381014 to 381021

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

35.25.6 SMON Set 2 Packet Counter

Number of packets counted in SMON Set 2.

In Figure 29.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space	Slice 0 : 87942 to 87949
	Slice 1 : 185622 to 185629
	Slice 2 : 283302 to 283309
	Slice 3 : 380982 to 380989



Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

35.25.7 SMON Set 3 Byte Counter

Number of bytes counted in SMON Set 3.

In Figure 29.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : VLAN PCP

Address Space

Slice 0 :	87982 to 87989
Slice 1 :	185662 to 185669
Slice 2 :	283342 to 283349
Slice 3 :	381022 to 381029

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

35.25.8 SMON Set 3 Packet Counter

Number of packets counted in SMON Set 3.

In Figure 29.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries : 8
 Type of Operation : Read/Write
 Addressing : VLAN PCP

Address Space

Slice 0 :	87950 to 87957
Slice 1 :	185630 to 185637
Slice 2 :	283310 to 283317
Slice 3 :	380990 to 380997

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0



