Packet Architects AB

Ethernet Switch
Advance L2/VLAN 48x1G + 5x10G
User Guide

Core Revision unknown

Datasheet Revision unknown

March 29, 2024

Contents

List of Figures

List of Tables

Chapter 1

Overview

This L2 Ethernet Switching Core offers full wire-speed on all 53 ports. 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 160 bytes. In total the buffer memory has a capacity of 13466 cells.

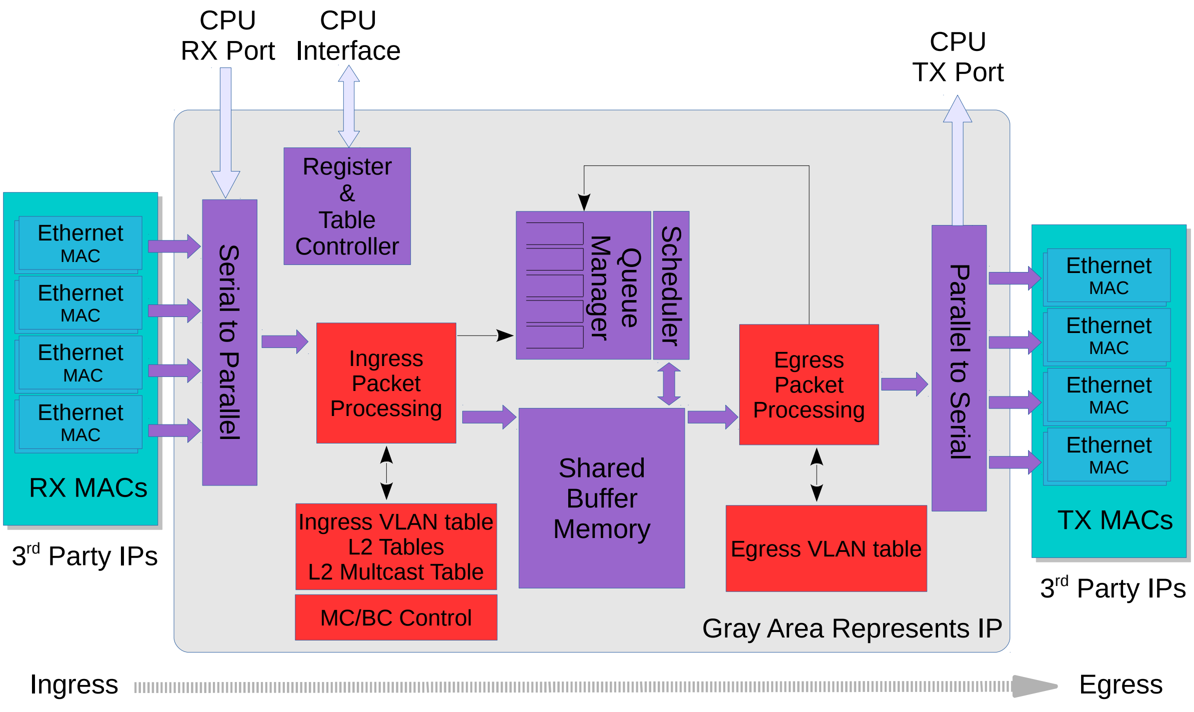

Figure 1.1: Switch Core Overview

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

- 48 ports of 1 Gigabit Ethernet.
- 5 ports of 10 Gigabit Ethernet.
- Full wire-speed on all ports and all Ethernet frame sizes.
- Store and forward shared memory architecture.
- Support for jumbo packets up to 16359 bytes.
- Passes maximum overlap mesh test (RFC2899) excluding the CPU port, 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.
- RSPAN - Remote Switch Port Analyzer
- 8 source MAC address ranges with a number of different actions.
- 8 destination MAC address ranges with a number of different actions.
- 32,768 entry L2 MAC table, hash based 8-way.
- 4,096 entry VLAN table.
- 64 entry synthesized CAM to solve hash collisions.
- 8 entries of the synthesized CAM are fully maskable.
- 1,024 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.
- 64 multiple spanning trees, ingress and egress checks.
- 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.
- Support for masking all look-up keys for L2 MAC table.
- 4432 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 11.2 The classificaiton / ACL key can be up to 322 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 256 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.
- 17236480 bits shared packet buffer memory for all ports divided into 13466 cells each of 160 bytes size
- 8 priority queues per egress port.
- Configurable mapping of egress queue from IP TOS, MPLS exp/tc or VLAN PCP bits.
- 128 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 with 27 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.
- Attack prevention by TCP flag rules combined with TCP-port and IP address checks, this also includes IMCP length attack checks.
- 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 form software to transmit MAC.

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 160 bytes to build a cell, and the cell is sent to ingress processing, if a packet consists of more than 160 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 128 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. The buffer memory has enough read and write performance for any traffic scenario and will never cause head of line blocking due to read / write conflicts.
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.
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. Register **CPU Port** determines the port that can serve as a CPU port, the default CPU port number is 52.

Interface Number	BW	Clock	Clock Frequency	Sync With Core Clock	Port Number & Multicast Table Bit	CPU Port
0	1.0Gbit/s	clk_mac_rx/tx_0	125.00MHz	No	0	Optional
1	1.0Gbit/s	clk_mac_rx/tx_1	125.00MHz	No	1	Optional
2	1.0Gbit/s	clk_mac_rx/tx_2	125.00MHz	No	2	Optional
3	1.0Gbit/s	clk_mac_rx/tx_3	125.00MHz	No	3	Optional
4	1.0Gbit/s	clk_mac_rx/tx_4	125.00MHz	No	4	Optional
5	1.0Gbit/s	clk_mac_rx/tx_5	125.00MHz	No	5	Optional
6	1.0Gbit/s	clk_mac_rx/tx_6	125.00MHz	No	6	Optional
7	1.0Gbit/s	clk_mac_rx/tx_7	125.00MHz	No	7	Optional
8	1.0Gbit/s	clk_mac_rx/tx_8	125.00MHz	No	8	Optional
9	1.0Gbit/s	clk_mac_rx/tx_9	125.00MHz	No	9	Optional
10	1.0Gbit/s	clk_mac_rx/tx_10	125.00MHz	No	10	Optional
11	1.0Gbit/s	clk_mac_rx/tx_11	125.00MHz	No	11	Optional
12	1.0Gbit/s	clk_mac_rx/tx_12	125.00MHz	No	12	Optional
13	1.0Gbit/s	clk_mac_rx/tx_13	125.00MHz	No	13	Optional
14	1.0Gbit/s	clk_mac_rx/tx_14	125.00MHz	No	14	Optional
15	1.0Gbit/s	clk_mac_rx/tx_15	125.00MHz	No	15	Optional
16	1.0Gbit/s	clk_mac_rx/tx_16	125.00MHz	No	16	Optional
17	1.0Gbit/s	clk_mac_rx/tx_17	125.00MHz	No	17	Optional
18	1.0Gbit/s	clk_mac_rx/tx_18	125.00MHz	No	18	Optional
19	1.0Gbit/s	clk_mac_rx/tx_19	125.00MHz	No	19	Optional
20	1.0Gbit/s	clk_mac_rx/tx_20	125.00MHz	No	20	Optional
21	1.0Gbit/s	clk_mac_rx/tx_21	125.00MHz	No	21	Optional
22	1.0Gbit/s	clk_mac_rx/tx_22	125.00MHz	No	22	Optional
23	1.0Gbit/s	clk_mac_rx/tx_23	125.00MHz	No	23	Optional
24	1.0Gbit/s	clk_mac_rx/tx_24	125.00MHz	No	24	Optional
25	1.0Gbit/s	clk_mac_rx/tx_25	125.00MHz	No	25	Optional
26	1.0Gbit/s	clk_mac_rx/tx_26	125.00MHz	No	26	Optional
27	1.0Gbit/s	clk_mac_rx/tx_27	125.00MHz	No	27	Optional
28	1.0Gbit/s	clk_mac_rx/tx_28	125.00MHz	No	28	Optional
29	1.0Gbit/s	clk_mac_rx/tx_29	125.00MHz	No	29	Optional
30	1.0Gbit/s	clk_mac_rx/tx_30	125.00MHz	No	30	Optional
31	1.0Gbit/s	clk_mac_rx/tx_31	125.00MHz	No	31	Optional
32	1.0Gbit/s	clk_mac_rx/tx_32	125.00MHz	No	32	Optional
33	1.0Gbit/s	clk_mac_rx/tx_33	125.00MHz	No	33	Optional
34	1.0Gbit/s	clk_mac_rx/tx_34	125.00MHz	No	34	Optional
35	1.0Gbit/s	clk_mac_rx/tx_35	125.00MHz	No	35	Optional
36	1.0Gbit/s	clk_mac_rx/tx_36	125.00MHz	No	36	Optional
37	1.0Gbit/s	clk_mac_rx/tx_37	125.00MHz	No	37	Optional
38	1.0Gbit/s	clk_mac_rx/tx_38	125.00MHz	No	38	Optional
39	1.0Gbit/s	clk_mac_rx/tx_39	125.00MHz	No	39	Optional
40	1.0Gbit/s	clk_mac_rx/tx_40	125.00MHz	No	40	Optional
41	1.0Gbit/s	clk_mac_rx/tx_41	125.00MHz	No	41	Optional
42	1.0Gbit/s	clk_mac_rx/tx_42	125.00MHz	No	42	Optional
43	1.0Gbit/s	clk_mac_rx/tx_43	125.00MHz	No	43	Optional

Interface Number	**BW**	**Clock**	**Clock Frequency**	**Sync With Core Clock**	**Port Number & Multicast Table Bit**	**CPU Port**
44	1.0Gbit/s	clk_mac_rx/tx_44	125.00MHz	No	44	Optional
45	1.0Gbit/s	clk_mac_rx/tx_45	125.00MHz	No	45	Optional
46	1.0Gbit/s	clk_mac_rx/tx_46	125.00MHz	No	46	Optional
47	1.0Gbit/s	clk_mac_rx/tx_47	125.00MHz	No	47	Optional
48	10.0Gbit/s	clk_mac_rx/tx_48	312.50MHz	No	48	Optional
49	10.0Gbit/s	clk_mac_rx/tx_49	312.50MHz	No	49	Optional
50	10.0Gbit/s	clk_mac_rx/tx_50	312.50MHz	No	50	Optional
51	10.0Gbit/s	clk_mac_rx/tx_51	312.50MHz	No	51	Optional
52	10.0Gbit/s	clk_mac_rx/tx_52	312.50MHz	No	52	Default

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*.

```
      |
      +----------+
 [ Timestamp ]   |
      +----------+
      |
      +-->[ MAC DA == BPDU         ]---+
      +-->[ MAC DA == SSTP         ]---+
      +-->[ MAC DA == cpuMacAddr   ]---+
      +-->[ MAC DA == other        ]---+
      +-->[ MAC DA == LLDP.mac1/2/3]---+
      +-->[ MAC DA == LACP.mac     ]---+
                                       |
      +--------------------------------+
      |
  [ MAC SA ]
      |
      +---[ EType==fromCpu    ]
      |   [ 19 byte CPU tag    ]-----+
      |                              |
      +<-----------------------------+
      |
      +<-----------------------------+
      |                              |
      |      0,1,2 VLAN tags         |
      +---[ EType==C-/S-VLAN TPID ]-+
      |   [   2 byte VLAN TCI     ]
      |       |
      +-->[ EType==LLDP.eth]--> X
      +-->[ EType==IEEE_1722_AVTP.eth]--> X
      +-->[ EType==ARP.eth]--> X
      +-->[ EType==RARP.eth]--> X
      +-->[ EType==ieee1588EthType.eth]--> X
      +-->[ EType==ieee8021xEthType.eth]--> X
```

```
+-->[ EType==PTP]--> X
+---[ EType==MPLS ]
|   [ MPLS tag 1  ]--> X
|
+-->[ EType==unknown ]--> X
|
+-->[ EType==PPPoE ]
|   [ PPPoE header ]
|     |
|     +-->[ EType!=IPv6 or EType !=IPv4 ]--> X
|     +-->[ EType==IPv6 ]-----+
|     +-->[ EType==IPv4 ]     |
|                  |          |
+-->[ EType==IPv6 ]-----------+
|                  |          |
+-->[ EType==IPv4 ]----+      |
                   | |        |
                   v v        v
           [ IPv4 Header ]  [ IPv6 Header ]
                 |            |
+----------------+------------+
|
+-->[ TCP Header                    ]--> X
+-->[ L4Proto == ahHeader.l4Proto   ]--> X
+-->[ L4Proto == espHeader.l4Proto  ]--> X
+-->[ L4Proto == gre.l4Proto        ]--> X
+-->[ L4Proto == sctp.l4Proto       ]--> X
+-->[ IGMP Header                   ]--> X
+-->[ ICMP Header                   ]--> X
+-->[ UDP Header                    ]----+
                                         |
+----------------------------------------+
|
+-->[ UDP Dest Port == bootp.udp1/udp2  ] --> X
+-->[ UDP Dest Port == capwap.udp1/udp2 ] --> X
+-->[ UDP Dest Port == gre.udp1/udp2    ] --> X
+-->[ UDP Dest Port == Unknown          ] --> X
```

The packet decoding is done according to the figure above. The packet decoding steps are described below.

1. A packet arrives at the ingress packet processing pipeline.
2. A packet can optionally have a timestamp prepended to the Ethernet frame by the MAC. This is configured per source port in **Ingress Ports With Timestamp**.
3. The destination MAC address is extracted and compared.
   (a) If the address matches the BPDU multicast address (01:80:C2:00:00:00) the packet can be sent to the CPU if enabled in **Send to CPU**. There is no decoding done apart from the MAC address comparison. BPDU frames are usually 802.3 encapsulated with a 802.2 LLC header. This decoding is not done by the switch. Note that packets that match the LLDP criteria described below will not be considered BPDU packets.
   (b) If the address matches the SSTP (Shared Spanning Tree Protocol) multicast address (01:00:0C:CC:CC:CD) the packet can be sent to the CPU if enabled in **Send to CPU**. There is no decoding done apart from the MAC address comparison.

(c) If the address matches the configurable **cpuMacAddr** and this feature is enabled then the packet will be sent to the CPU port.

(d) If the address matches one of the mac1/mac2/mac3 addresses in the **LLDP Configuration** the packet will subject to further LLDP decoding.

(e) If the DA MAC is equal to the register **LACP Packet Decoder Options** field **mac** 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.

4. The source MAC address is extracted from the packet.

5. The Ethernet type is extracted from the packet and is then compared to known types.

(a) 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 critera will not be considered a BPDU packet even if it matches the BPDU multicast address.

(b) 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.

(c) 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.

(d) 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**.

(e) 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.

(f) 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.

(g) 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**.

(h) MPLS.
One MPLS tag is decoded. No other L3 decoding will be done after this.

(i) From CPU Tags
Packets from CPU will use a Ethernet type value of 0x9988.The From CPU Tag is further described in Chapter 28.

(j) 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.

(k) 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 **l4Proto** 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 **l4Proto** 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 **l4Proto** 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 **l4Proto** 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.

(l) 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 **l4Port** 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. 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.

(m) 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
   The packet headers are decoded and data extracted. For details see the **Packet Decoding** chapter.

3. 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.

4. 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.

5. 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).

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

11. TTL routing check and drop
TTL check is enabled when the packet has an ACL action to decrease the TTL. **Expired TTL to CPU** determines if the packet with expired TTL shall be dropped or sent to the CPU port.

12. 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.

13. Attack prevention drop
TCP/UDP packets are checked by **TCP/UDP Flag Rules** to prevent security or DOS attacks.

14. L2 Switching
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.

15. 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 (**l2ActionTablePortState**) 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).

16. 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.

17. 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.

18. Learning Lookup
The source MAC address is searched in the **L2 DA 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.

19. 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**.

20. 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.

21. 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**.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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 groupping process is through sequence of ingress packet processing steps, the metering process is after all other ingress packet processing are done and before the enqueuing 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. Egress Port VLAN
A VLAN header operation can be performed based on the physical output port. See the **VLAN Processing** chapter.

3. 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.

4. Egress VLAN Translation
Potentially replace the outgoing VID and Ethernet Type on a specific port with a specific VID. Uses a Dleft lookup in **Egress VLAN Translation Small Table**, **Egress VLAN Translation Large Table** and **Egress VLAN Translation TCAM**.

5. RSPAN
Perform a push or pop of an RSPAN tag if enabled in **Egress RSPAN Configuration**.

6. Reassemble Packet Headers
The final step in the egress processing is to reassembly the outgoing packet 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 15 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 2 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 tree places (t1-t3) in the core where latency jitter can be introduced. See Figure 4.1 on page 32.

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 asyncronous 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 asyncronous 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.

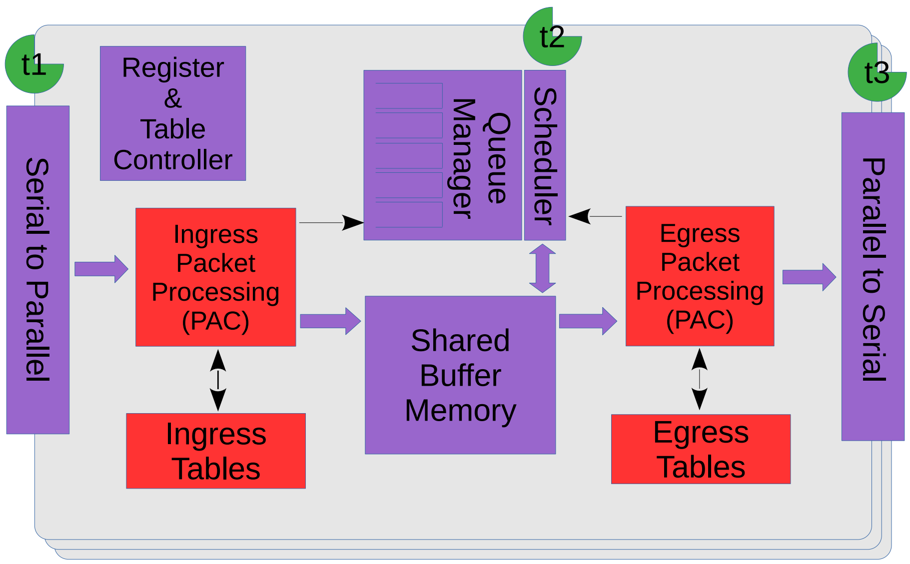

Figure 4.1: Jitter Overview

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.

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 19.1).

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:

```
+-------------+-------------------------+
|             |           TCI           |
|    TPID     +------+------+-----------+
|             | PCP  | DEI  |    VID    |
+-------------+------+------+-----------+
```

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.

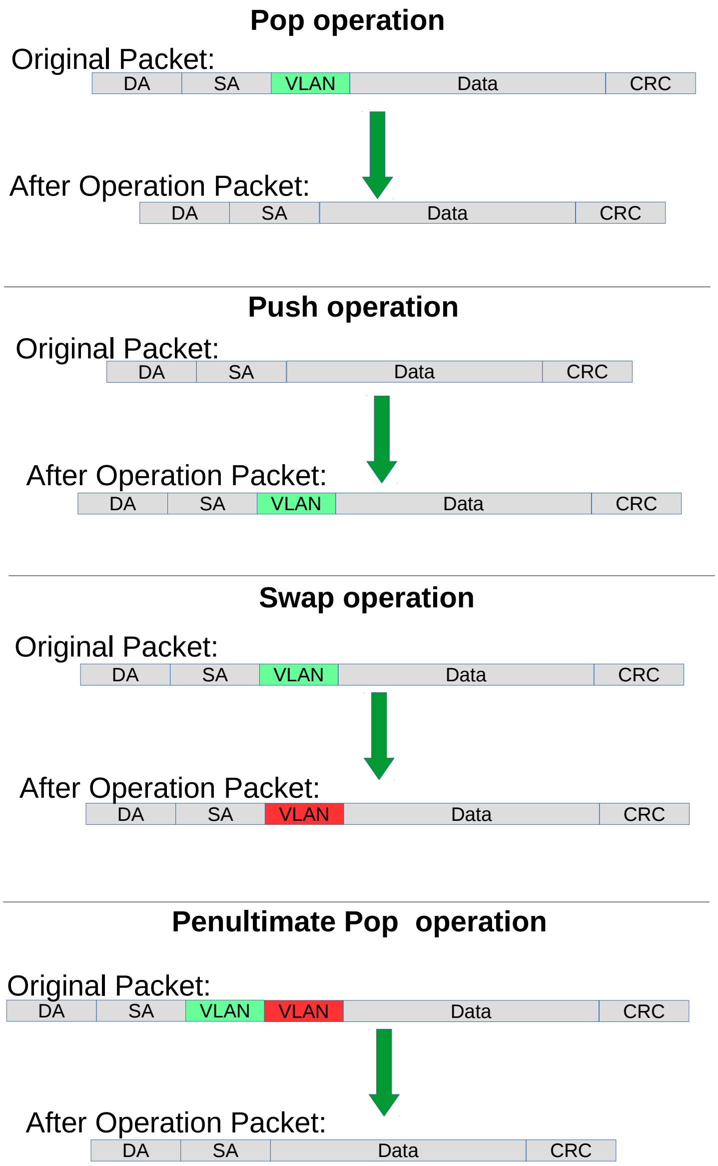

Figure 5.1: VLAN Packet Operations

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**.

5.3.3 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.

5.3.4 VLAN Table Operation

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

5.3.5 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**.

5.3.6 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 Dleft lookup using the **Egress VLAN Translation Small Table**, **Egress VLAN Translation Large Table** and **Egress VLAN Translation TCAM** Tables. If multiple hits the **Egress VLAN Translation Selection** can be used to determine which result to select. It is possible to mask the search data using **Egress VLAN Translation Search Mask**..

5.3.7 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 19.1 **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.

5.3.8 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.
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.
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.

5.3.9 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]
```

5.3.10 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 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 concatenation {GID,DA MAC} is AND:ed with the global masks. The global mask for the DA MAC lookup is set up in the **Mask MAC Table Lookup** register.
- 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**. 8 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** and the **L2 Aging Status Shadow Table**. 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**.

- 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 a 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 is two tables which allows 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 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 used a number of bits from the source port table, egress port state, SA and DA MAC lookups to form a 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**.

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 carrie 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 a 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 be 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 **l2ActionTablePortState** 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 **l2ActionTableSaStatus** 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 **l2ActionTableDaStatus** 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

Mirroring

This core supports both input and output mirroring.

7.1 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.

7.2 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

7.2.1 Requeueing FIFO

Output mirroring (and input mirroring to oneself) is accomplished by requeuing 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 requeuing 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 fourteen packets per egress port.

Drops due to the requeueing FIFOs overflowing are counted in the **Re-queue Overflow Drop** register.

Chapter 8

RSPAN - Remote Switch Port Analyzer

RSPAN is a function that allows mirroring traffic to other switches by encapsulating the packets in a VLAN tag.

An RSPAN network consists of switches with three roles.

1. *Source Device*
   The source device is where the mirrored traffic originates. It uses the normal mirroring functions to send the mirror copies. The mirrored packets are encapsulated in a RSPAN tag and output on a port.

2. *Intermediate Device*
   An intermediate device just forwards the RSPAN tagged packets.

3. *Destination Device*
   The destination device removes the RSPAN tag and output the packet on a port.

8.1 Source Device

Input and output mirroring can be used to create the mirror copies. A dedicated RSPAN port, reflector port, is used. On this port only mirror traffic should be sent. No other traffic should be switched to this port, i.e. normal switching functions should not use this port as a destination.

The reflector port must be configured to push a RSPAN tag by setting **pushRspanTag** in **Egress RSPAN Configuration**.

The RSPAN tag is a normal VLAN tag and the content of the tag is configured in **Egress RSPAN Configuration**.

A switch can have multiple reflector ports.

8.2 Intermediate Device

An intermediate device must be configured to allow receiving RSPAN tagged packets and to forward them to a dedicated port. This can be accomplished by setting up a source port VLAN with a GID only used for this purpose. The VLAN will have two member ports, the RSPAN ingress port and the RSPAN egress port. Learning should be disabled for the ingress port. The ingress packets will then be flooded to the egress port.

8.3 Destination Device

The destination device receives the RSPAN packet on a dedicated ingress port and forwards them to the dedicated monitor port. This forwarding can be done in the same way as an intermediate device.

On the egress port the RSPAN tag is popped by setting **popRspanTag** in **Egress RSPAN Configuration**.

Chapter 9

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.

9.0.1 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.

9.1 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 addres, 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
```

9.2 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, 6 bits

If a field is disabled in the **Link Aggregation Ctrl** register then the field in the hash key 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,
              saIp,useIpInHash,
              daIp,
              srcPort):
    # This function builds the key to be
    #   used for calculating the hash.
    final_data = 0
    if useDaMacInHash==0:
        daMac = 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 | saIp
    final_data = final_data <<128
    if useIpInHash==1:
        final_data = final_data | daIp
    final_data = final_data <<6
    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 10

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.

10.1 Timestamp from RX MAC

Each ingress port can be configured in **Ingress Ports With Timestamp** to use a prepended timestamp before the normal L2 header on all packets. The timestamp should be created by the MAC and added before the MAC sends the packet to the switch. The transfer of the timestamp must be done during the inter frame gap period in order to not affect performance.

The timestamp must be added on all packets on a port also on non-PTP packets.

The timestamp size is 8 bytes.

10.1.1 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.

10.2 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.

10.2.1 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)

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 10.1: PTP Header Format

MAC DA	MAC SA	EtherType=0x88F7	PTP

Table 10.2: PTP over 802.3 Ethernet

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.

10.2.2 PTP over UDP

MAC DA	MAC SA	EtherType	IPv4	UDP	PTP

Table 10.3: PTP over UDP/IPv4

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.

10.3 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.

MAC DA	MAC SA	EtherType	IPv6	UDP	PTP	Checksum Correction

Table 10.4: PTP over UDP/IPv6

10.3.1 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.

10.4 Support for Ordinary Clock

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

10.4.1 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.

10.4.2 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**.

10.4.3 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.

10.4.4 Master receiving DelayReq

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

10.4.5 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.

10.4.6 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.

10.5 Support for 1-step Peer to Peer

10.5.1 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.

10.5.2 Peer receiving PDelayReq

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

10.5.3 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.

10.5.4 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 11

Classification

11.1 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.
- L2 Source MAC Drop is setup in table **L2 Destination Table** using field **pktDropSa**. This will drop all packets which matches this SA MAC address.
- 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.

11.2 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 parallell 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.

11.2.1 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	222	19	6	4	256	2048	32
1	322	31	6	4	128	1024	16
2	222	31	6	4	64	512	16
3	222	31	6	4	64	256	16

Table 11.1: 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 poitner 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	3 bits	The source port bits from source port table **preLookupAclBits**.
Number of VLANS	2 bits	The packets number of incoming VLANs.
L2 Protocol	1 bits	The packets L2 Type 0 = Other than this list. 1 = IEEE 1722 AVTP .
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 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	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
3	Has VLANs	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANs in packet .
4	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 .
5	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 .
6	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
7	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.

Bit in Select Bit-mask	Field Name	Size in Bits	When is field valid?	Description
8	Inner VID	12	When packet has a two VLANs.	The packes innermost VLAN Identifier (VID).
9	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.
10	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
11	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.
12	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.
13	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
14	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
15	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
16	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
17	Source Port	6	Always valid	The source port of the packet.
18	Rule Pointer	4	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

11.2.2 Example Of Selecting Fields For Configurable Ingress ACL Table 0

Since this ACL engine can select up to 6 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 222 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 19 bits , will be set as follows 100000000000000 in binary format (Hex value of 0x4000) and the lookup data will be located as follows:

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

Table 11.4: Hash Key Example for Ethernet Type

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 19 bits , will be set as follows 101 in binary format (Hex value of 0x5) and the lookup data will be located as follows:

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

Table 11.5: Hash Key Example for Destiantion MAC Address and Outer LAN VID

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the field-SelectBitmask, which is 19 bits , will be set as follows 100000000000011 in binary format (Hex value of 0x4003) and the lookup data will be located as follows:

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

Table 11.6: Hash Key Example for Simple L2 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 19 bits , will be set as follows 110011100000000000 in binary format (Hex value of 0x33800) and the lookup data will be located as follows:

0 - 53	Source Port Width : 6 52 47	L3 Type Width : 2 46 45	L4 Protocol Width : 8 44 37	L4 Destination Port Width : 16 36 21	L4 Source Port Width : 16 20 5	Valid 5 4 0

Table 11.7: Hash Key Example for L4 ACL

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 poitner 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	3 bits	The source port bits from source port table **preLookupAclBits**.
Number of VLANS	2 bits	The packets number of incoming VLANs.

Packet Field	Size in Bits	Description
L2 Protocol	1 bits	The packets L2 Type 0 = Other than this list. 1 = IEEE 1722 AVTP
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	MAC DA	48	Always valid	The packets destination MAC address.
1	MAC SA	48	Always valid	The packets source MAC address
2	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
3	Has VLANs	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANs in packet
4	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
5	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
6	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
7	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.
8	Inner VID	12	When packet has a two VLANs.	The packes innermost VLAN Identifier (VID).
9	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.
10	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
11	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.

Bit in Select Bit-mask	Field Name	Size in Bits	When is field valid?	Description
12	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
13	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
14	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
15	Outer MPLS	20	When L2 frame holds a MPLS packet.	Outermost MPLS label.
16	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
17	TTL	8	When packet is a IPv4,IPv6 or MPLS	IPv4, IPv6 or MPLS Time-To-Live (TTL) byte.
18	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.
19	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.
20	MLD Address	128	When packet is a IPv6 and the ICMPv6 type is equal to 130,131,132	The MLD headers Multicast Address field.
21	ICMP Type	8	When L4 packet is a ICMP packet	ICMP Type.
22	ICMP Code	8	When L4 packet is a ICMP packet	ICMP Code.
23	IGMP Type	8	When L4 packet is a IGMP	IGMP Type.
24	IGMP Group Address	32	When L4 packet is a IGMP	IGMP Group Address.
25	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
26	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.

Bit in Select Bit-mask	Field Name	Size in Bits	When is field valid?	Description
27	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
28	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
29	Source Port	6	Always valid	The source port of the packet.
30	Rule Pointer	4	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

11.2.3 Example Of Selecting Fields For Configurable Ingress ACL Table 1

Since this ACL engine can select up to 6 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 322 bits in search key looks like when different fields are selected.

Example ACL with Outer VLAN ID

In this example we only want to create a rule with one field which is the Outer VLAN ID. This means that the fieldSelectBitmask, which is 31 bits , will be set as follows 100 in binary format (Hex value of 0x4) and the lookup data will be located as follows:

0 - 13	Outer VID Width : 12 12 1	Valid 1 0 0

Table 11.10: Hash Key Example for Outer VLAN ID

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 31 bits , will be set as follows 101 in binary format (Hex value of 0x5) and the lookup data will be located as follows:

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

Table 11.11: Hash Key Example for Destiantion MAC Address and Outer LAN VID

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the field-

SelectBitmask, which is 31 bits , will be set as follows 1000000000000000000000000011 in binary format (Hex value of 0x4000003) 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 11.12: 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 Destiantion 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 31 bits , will be set as follows 10010000000000001100000000000 in binary format (Hex value of 0x12001800) 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 11.13: 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 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 31 bits , will be set as follows 110010000011000000000000000000 in binary format (Hex value of 0x320c0000) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 6	Width : 2	Width : 8	Width : 16	Width : 16	5
53	52 47	46 45	44 37	36 21	20 5	4 0

Table 11.14: 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 selecing source port, destiantion MAC, source MAC, Ethernet Type, inner VLAN, outer VLAN, L3 Type, IPv4 SA, IPv4 DA, L4 protocol, L4 Source port and L4 Destiantion port and finally the rule pointer. All in all 13 fields are selected. This means that the fieldSelectBitmask, which is 31 bits , will be set as follows 1110110000011000001100100000111 in binary format (Hex value of 0x760c1907) 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 : 6	Width : 48	Width : 48	Width : 12	Width : 12	Width : 16	Width : 2
265	264 259	258 211	210 163	162 151	150 139	138 123	122 121

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 : 4	13
120 89	88 57	56 49	48 33	32 17	16 13	12 0

Table 11.15: Hash Key Example for Openflow 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 poitner 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	3 bits	The source port bits from source port table **preLookupAclBits**.
Number of VLANS	2 bits	The packets number of incoming VLANs.
L2 Protocol	1 bits	The packets L2 Type 0 = Other than this list. 1 = IEEE 1722 AVTP .
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 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	MAC DA	48	Always valid	The packets destination MAC address.
1	MAC SA	48	Always valid	The packets source MAC address
2	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
3	Has VLANs	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANs in packet .
4	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 .
5	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 .
6	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
7	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.
8	Inner VID	12	When packet has a two VLANs.	The packes innermost VLAN Identifier (VID).
9	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.

Bit in Select Bit-mask	Field Name	Size in Bits	When is field valid?	Description
10	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
11	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
12	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
13	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
14	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
15	Outer MPLS	20	When L2 frame holds a MPLS packet.	Outermost MPLS label.
16	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
17	TTL	8	When packet is a IPv4,IPv6 or MPLS	IPv4, IPv6 or MPLS Time-To-Live (TTL) byte.
18	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.
19	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.
20	MLD Address	128	When packet is a IPv6 and the ICMPv6 type is equal to 130,131,132	The MLD headers Multicast Address field.
21	ICMP Type	8	When L4 packet is a ICMP packet	ICMP Type.
22	ICMP Code	8	When L4 packet is a ICMP packet	ICMP Code.
23	IGMP Type	8	When L4 packet is a IGMP	IGMP Type.
24	IGMP Group Address	32	When L4 packet is a IGMP	IGMP Group Address.
25	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
26	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.

Bit in Select Bit-mask	Field Name	Size in Bits	When is field valid?	Description
27	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
28	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
29	Source Port	6	Always valid	The source port of the packet.
30	Rule Pointer	4	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

11.2.4 Example Of Selecting Fields For Configurable Ingress ACL Table 2

Since this ACL engine can select up to 6 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 222 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 31 bits , will be set as follows 1000000000000 in binary format (Hex value of 0x1000) and the lookup data will be located as follows:

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

Table 11.18: Hash Key Example for IPv4 DA

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 31 bits , will be set as follows 101 in binary format (Hex value of 0x5) and the lookup data will be located as follows:

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

Table 11.19: 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 31 bits , will be set as follows 10001000000000000000000100000111 in binary format (Hex value of 0x44000107) 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 : 4	6
146	145 98	97 50	49 34	33 22	21 10	9 6	5 0

Table 11.20: 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 Destiantion 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 31 bits , will be set as follows 100100000000000110000000000000 in binary format (Hex value of 0x12006000) 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 11.21: 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 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 31 bits , will be set as follows 1100100000110000000000000000000 in binary format (Hex value of 0x320c0000) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 6	Width : 2	Width : 8	Width : 16	Width : 16	5
53	52 47	46 45	44 37	36 21	20 5	4 0

Table 11.22: 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 selecing source port, destiantion MAC, source MAC, Ethernet Type, inner VLAN, outer VLAN, L3 Type, IPv4 SA, IPv4 DA, L4 protocol, L4 Source port and L4 Destiantion port and finally the rule pointer. All in all 13 fields are selected. This means that the fieldSelectBitmask, which is 31 bits , will be set as follows 1110110000011000001100100000111 in binary format (Hex value of 0x760c1907) 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 : 6	Width : 48	Width : 48	Width : 12	Width : 12	Width : 16	Width : 2
265	264 259	258 211	210 163	162 151	150 139	138 123	122 121

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 : 4	13
120 89	88 57	56 49	48 33	32 17	16 13	12 0

Table 11.23: Hash Key Example for Openflow Entry

Pre Lookup for Configurable Ingress ACL Table 3

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 poitner then the rule pointer from the source port table will be selected. The prelookup is setup in **Ingress Configurable ACL 3 Pre Lookup**

Packet Field	Size in Bits	Description
Source Port Bits	3 bits	The source port bits from source port table **preLookupAclBits**.
Number of VLANS	2 bits	The packets number of incoming VLANs.
L2 Protocol	1 bits	The packets L2 Type 0 = Other than this list. 1 = IEEE 1722 AVTP .
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 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	MAC DA	48	Always valid	The packets destination MAC address.
1	MAC SA	48	Always valid	The packets source MAC address
2	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
3	Has VLANs	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANs in packet .
4	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 .
5	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 .
6	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
7	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.

Bit in Select Bit-mask	Field Name	Size in Bits	When is field valid?	Description
8	Inner VID	12	When packet has a two VLANs.	The packes innermost VLAN Identifier (VID).
9	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.
10	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
11	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
12	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
13	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
14	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
15	Outer MPLS	20	When L2 frame holds a MPLS packet.	Outermost MPLS label.
16	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
17	TTL	8	When packet is a IPv4,IPv6 or MPLS	IPv4, IPv6 or MPLS Time-To-Live (TTL) byte.
18	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.
19	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.
20	MLD Address	128	When packet is a IPv6 and the ICMPv6 type is equal to 130,131,132	The MLD headers Multicast Address field.
21	ICMP Type	8	When L4 packet is a ICMP packet	ICMP Type.
22	ICMP Code	8	When L4 packet is a ICMP packet	ICMP Code.
23	IGMP Type	8	When L4 packet is a IGMP	IGMP Type.
24	IGMP Group Address	32	When L4 packet is a IGMP	IGMP Group Address.
25	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
26	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.

Bit in Select Bit-mask	Field Name	Size in Bits	When is field valid?	Description
27	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
28	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
29	Source Port	6	Always valid	The source port of the packet.
30	Rule Pointer	4	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

11.2.5 Example Of Selecting Fields For Configurable Ingress ACL Table 3

Since this ACL engine can select up to 6 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 222 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 31 bits , will be set as follows 10000000000000000 in binary format (Hex value of 0x10000) and the lookup data will be located as follows:

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

Table 11.26: 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 31 bits , will be set as follows 101 in binary format (Hex value of 0x5) 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 11.27: Hash Key Example for Destiantion MAC Address and Outer LAN VID

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the field-

SelectBitmask, which is 31 bits , will be set as follows 100000000000000000000000011 in binary format (Hex value of 0x4000003) 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 11.28: 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 Destiantion 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 31 bits , will be set as follows 10010000000000001100000000000 in binary format (Hex value of 0x12001800) 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 11.29: 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 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 31 bits , will be set as follows 110010000011000000000000000000 in binary format (Hex value of 0x320c0000) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 6	Width : 2	Width : 8	Width : 16	Width : 16	5
53	52 47	46 45	44 37	36 21	20 5	4 0

Table 11.30: 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 selecing source port, destiantion MAC, source MAC, Ethernet Type, inner VLAN, outer VLAN, L3 Type, IPv4 SA, IPv4 DA, L4 protocol, L4 Source port and L4 Destiantion port and finally the rule pointer. All in all 13 fields are selected. This means that the fieldSelectBitmask, which is 31 bits , will be set as follows 1110110000011000001100100000111 in binary format (Hex value of 0x760c1907) 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 : 6	Width : 48	Width : 48	Width : 12	Width : 12	Width : 16	Width : 2
265	264 259	258 211	210 163	162 151	150 139	138 123	122 121

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 : 4	13
120 89	88 57	56 49	48 33	32 17	16 13	12 0

Table 11.31: Hash Key Example for Openflow Entry

11.2.6 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** .

11.2.7 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**

The MAC operation to be done on the packet is located in the table **Egress MAC Operations** which enables changing both Source and Destination MAC address. A example use case for this operation is to do routing from the ACL table.

11.3 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.

11.3.1 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 Action	Ingress Acl 1 Has Action	Ingress Acl 2 Has Action	Ingress Acl 3 Has Action
noLearning	Yes	Yes	Yes	Yes
decTtl	Yes	Yes	Yes	Yes
dropEnable	Yes	Yes	Yes	Yes
sendToCpu	Yes	Yes	Yes	Yes

Table 11.32: 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.

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**	Yes	Yes	Yes	Yes
forceQueue	**forceQueuePrio**	**eQueue**	Yes	Yes	Yes	Yes
forceColor	**forceColorPrio**	**color**	Yes	Yes	Yes	Yes
mmpValid	**mmpOrder**	**mmpPtr**	Yes	Yes	Yes	Yes
macOp	**macPrio**	**macOpPtr**	Yes	Yes	Yes	Yes
updateCfiDei	**cfiDeiPrio**	**newCfiDeiValue**	Yes	Yes	Yes	Yes
updatePcp	**pcpPrio**	**newPcpValue**	Yes	Yes	Yes	Yes
updateVid	**vidPrio**	**newVidValue**	Yes	Yes	Yes	Yes
updateEType	**ethPrio**	**newEthType**	Yes	Yes	Yes	Yes
imPrio	**inputMirror**	**destInputMirror**	Yes	Yes	Yes	Yes
sendToPort	N/A	**destPort**	Yes	Yes	Yes	Yes
updateCounter	N/A	**counter**	Yes	Yes	Yes	Yes

Table 11.33: The lowest numbered takes effect if no priority else the highest numbered with priority set.

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.

11.3.2 ACL Routing

This ACL engine can be used to achieve routing functionality. This is done by:

- Setup the ACLs lookup table search data to use the routers MAC DA and destination IP address in a single entry. If needed the TCAM entries of the ACLs can be used to achieve Longest Prefix Match (LPM) lookups.
- If LPM lookups are needed then place the entries with the longest prefix at the lowest entries (starting at entry 0) in the TCAM. The TCAM tables are searched from the lowest entry (0) to the highest entry and hence will hit the entry with the longest prefix first.

- If both IPv4 and IPv6 are needed then two seperate ACL tables need to be setup. Include the L3 Type in the searches to avoid false positives.
- Optionally the VID from the incoming packet can be added to the match if the router is only available on a certain VID.
- If a default route shall be used then setup a TCAM entry as the last in TCAM table where none of the IP bits are compared (But the MAC DA bits shall be compared).
- Turn on the IP checksum checker in register **Check IPv4 Header Checksum**.
- Use four actions listed below to achieve routing, both IPv6 and IPv4 routing are supported.
  - Use the action **sendToPort** to send the packet to the correct destination port.
  - Use the action **macOp** and point into the **Egress MAC Operations** where the egress packet modification operation should be set as follows:
    * The new MAC SA shall be copied from the original MAC DA.
    * The new MAC DA shall be taken from the table. This is the Next Hop MAC DA address.
  - Use the action **decTtl** to decrement the incoming packets TTL. If the new TTL ends up being zero then the packet can either be sent to the CPU (using register **Expired TTL to CPU**) or dropped. The drop counter is recorded in **Expired TTL Drop**.
  - For statistics use the action **updateCounter**

11.3.3 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.

Chapter 12

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 13

Attack Prevention

The switch has the possibility to decode TCP/UDP packets and detect and drop packets that matches patterns in order to prevent security or DOS attacks.

If a packet is a TCP/UDP packet (IPv4 or IPv6) the TCP/UDP flags will be compared to all the **TCP/UDP Flag Rules**. The flag comparison can also be combined with a check if the IP Source address equals the IP Destination address. There is also a check if the TCP/UDP source port number matches the TCP/UDP destination port number.

The switch also provides a length check for ICMP packets. **ICMP Length Check** allows the packet to be dropped if the ICMP protocol data size is more than a certain bytes.

If a packet matches any of these rules the packet will be dropped and the **Attack Prevention Drop** will be incremented. When a packet fails either the ICMP length check or the TCP/UDP flag check, the ACL rules can still be hit. However, the ACL action to send the packet to the CPU or any egress port will only override drop decisions based on TCP/UDP rules. In other words, if a packet fails the ICMP length check, it cannot be redirected to an egress port using ACL actions.

Chapter 14

Hashing

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

14.1 Hashing Functions

This section describes the hash functions used in this core.

Before each hash is calculated the values are masked by doing a AND function with the masks setup in **Mask MAC Table Lookup**.

14.1.1 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 (8). 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 5 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 44% (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 = 5
        returns: 12 bits hash value
    """
    hashval = key & 0b111111111111
    hashval = hashval ^ (key>>12)
    hashval = hashval & 0b111111111111
    hashval = hashval ^ (key>>24)
    hashval = hashval & 0b111111111111
    hashval = hashval ^ (key>>36)
    hashval = hashval & 0b111111111111
    hashval = hashval ^ (key>>48)
    hashval = hashval & 0b111111111111
```

```
  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:1125
    key = (1125)<< 48 | 0x466162bc84dd
    hashval = calc_l2_hash(key) # the hash value is used as index into the L2 DA Hash T
    assert hashval == 3700
```

14.1.2 Hash function for Ingress Configurable ACL 0

The hash function recevies 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 222 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: 222 bits hash key
      fold count = 37
      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
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 & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffff
  return calc_confAcl_small0_hash( key )


def calc_confAcl_large0_hash( key ):
  """ key: 222 bits hash key
      fold count = 25
      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 & 0b111111111
  hashval = hashval ^ (key>>153)
  hashval = hashval & 0b111111111
  hashval = hashval ^ (key>>162)
  hashval = hashval & 0b111111111
  hashval = hashval ^ (key>>171)
  hashval = hashval & 0b111111111
  hashval = hashval ^ (key>>180)
  hashval = hashval & 0b111111111
  hashval = hashval ^ (key>>189)
  hashval = hashval & 0b111111111
  hashval = hashval ^ (key>>198)
  hashval = hashval & 0b111111111
```

```
  hashval = hashval ^ (key>>207)
  hashval = hashval & 0b111111111
  hashval = hashval ^ (key>>216)
  hashval = hashval & 0b111111111
  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 & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffff
  return calc_confAcl_large0_hash( key )



def confAcl0_hash_test():
    key = 1077741769851287329901319011142359533502979116624325718010924180272
    hashval = confAcl_small0_hash(key)
    assert hashval == 58

    hashval = confAcl_large0_hash(key)
    assert hashval == 58
```

14.1.3 Hash function for Ingress Configurable ACL 1

The hash function recevies 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 322 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: 322 bits hash key
      fold count = 65
      returns: 5 bits hash value
  """
  hashval = key & 0b11111
  hashval = hashval ^ (key>>5)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>10)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>15)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>20)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>25)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>30)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>35)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>40)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>45)
  hashval = hashval & 0b11111
  hashval = hashval ^ (key>>50)
  hashval = hashval & 0b11111
```

```
hashval = hashval ^ (key>>55)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>60)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>65)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>70)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>75)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>80)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>85)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>90)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>95)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>100)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>105)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>110)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>115)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>125)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>130)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>135)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>140)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>145)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>150)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>155)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>160)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>165)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>170)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>175)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>180)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>185)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>190)
hashval = hashval & 0b11111
hashval = hashval ^ (key>>195)
```

```
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>200)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>205)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>210)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>215)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>220)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>225)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>230)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>235)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>240)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>245)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>250)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>255)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>260)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>265)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>270)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>275)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>280)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>285)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>290)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>295)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>300)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>305)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>310)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>315)
    hashval = hashval & 0b11111
    hashval = hashval ^ (key>>320)
    hashval = hashval & 0b11111
    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 & 0x3ffffffffffffffffffffffffffffffffffffffffffffffffffffffff
```

```
  return calc_confAcl_small1_hash( key )


def calc_confAcl_large1_hash( key ):
  """ key: 322 bits hash key
      fold count = 41
      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
  hashval = hashval ^ (key>>136)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>144)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>152)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>160)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>168)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>176)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>184)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>192)
  hashval = hashval & 0b11111111
```

```
  hashval = hashval ^ (key>>200)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>208)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>216)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>224)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>232)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>240)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>248)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>256)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>264)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>272)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>280)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>288)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>296)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>304)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>312)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>320)
  hashval = hashval & 0b11111111
  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 & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffff
  return calc_confAcl_large1_hash( key )



def confAcl1_hash_test():
    key = 794863248110500551464916705323710578824365242952345287517876619958948392583886
    hashval = confAcl_small1_hash(key)
    assert hashval == 31

    hashval = confAcl_large1_hash(key)
    assert hashval == 62
```

14.1.4 Hash function for Ingress Configurable ACL 2

The hash function recevies the lookup key created by selecting the fields from the packet determined by the **Ingress Configurable ACL 2 Rules Setup** The lookup key is up to 222 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_small2_hash( key ):
  """ key: 222 bits hash key
      fold count = 56
      returns: 4 bits hash value
  """
  hashval = key & 0b1111
  hashval = hashval ^ (key>>4)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>8)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>12)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>16)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>20)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>24)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>28)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>32)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>36)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>40)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>44)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>48)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>52)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>56)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>60)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>64)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>68)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>72)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>76)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>80)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>84)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>88)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>92)
  hashval = hashval & 0b1111
  hashval = hashval ^ (key>>96)
  hashval = hashval & 0b1111
```

```
hashval = hashval ^ (key>>100)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>104)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>108)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>112)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>116)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>124)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>128)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>132)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>136)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>140)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>144)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>148)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>152)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>156)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>160)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>164)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>168)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>172)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>176)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>180)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>184)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>188)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>192)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>196)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>200)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>204)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>208)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>212)
```

```
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>216)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>220)
    hashval = hashval & 0b1111
    return hashval

def confAcl_small2_hash( destination_address ):
    """ Calculate index into confAcl_small2 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_small2_hash( key )


def calc_confAcl_large2_hash( key ):
    """ key: 222 bits hash key
        fold count = 32
        returns: 7 bits hash value
    """
    hashval = key & 0b1111111
    hashval = hashval ^ (key>>7)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>14)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>21)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>28)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>35)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>42)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>49)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>56)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>63)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>70)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>77)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>84)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>91)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>98)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>105)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>112)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>119)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>126)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>133)
```

```
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>140)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>147)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>154)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>161)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>168)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>175)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>182)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>189)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>196)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>203)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>210)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>217)
    hashval = hashval & 0b1111111
    return hashval

def confAcl_large2_hash( destination_address ):
    """ Calculate index into confAcl_large2 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_large2_hash( key )



def confAcl2_hash_test():
      key = 12967117406310246089580885594751098639837666856524645656208956O4170
      hashval = confAcl_small2_hash(key)
      assert hashval == 10

      hashval = confAcl_large2_hash(key)
      assert hashval == 119
```

14.1.5 Hash function for Ingress Configurable ACL 3

The hash function recevies the lookup key created by selecting the fields from the packet determined by the **Ingress Configurable ACL 3 Rules Setup** The lookup key is up to 222 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_small3_hash( key ):
    """ key: 222 bits hash key
        fold count = 56
        returns: 4 bits hash value
    """
```

```
hashval = key & 0b1111
hashval = hashval ^ (key>>4)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>8)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>12)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>16)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>20)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>24)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>28)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>32)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>36)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>40)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>44)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>48)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>52)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>56)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>60)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>64)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>68)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>72)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>76)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>80)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>84)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>88)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>92)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>100)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>104)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>108)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>112)
hashval = hashval & 0b1111
```

```
    hashval = hashval ^ (key>>116)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>120)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>124)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>128)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>132)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>136)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>140)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>144)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>148)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>152)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>156)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>160)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>164)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>168)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>172)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>176)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>180)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>184)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>188)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>192)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>196)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>200)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>204)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>208)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>212)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>216)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>220)
    hashval = hashval & 0b1111
    return hashval

def confAcl_small3_hash( destination_address ):
```

```
    """ Calculate index into confAcl_small3 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_small3_hash( key )


def calc_confAcl_large3_hash( key ):
    """ key: 222 bits hash key
        fold count = 37
        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
    return hashval

def confAcl_large3_hash( destination_address ):
   """ Calculate index into confAcl_large3 hash table from
       the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x3ffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_large3_hash( key )



def confAcl3_hash_test():
      key = 670621934316019683801298571508479579371087208237775909055983367 6
      hashval = confAcl_small3_hash(key)
      assert hashval == 11

      hashval = confAcl_large3_hash(key)
      assert hashval == 21
```

14.1.6 Hash function for Egress Vlan Translation

The hash function receives the outermost VID of the modified packet at egress, the egress port number, along with the VLAN Ethernet type (C or S tag). 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_egressVlanTranslation_small_hash( outermostVidType,
                                            outermostVid,
                                            dstPort ):
```

```
    """ key: 19 bits hash key
        fold count = 4
        returns: 6 bits hash value
    """
    key = 0
    key = key << 1 | (outermostVidType & 0x1)
    key = key << 12 | (outermostVid & 0xfff)
    key = key << 6 | (dstPort & 0x3f)
    hashval = key & 0b111111
    hashval = hashval ^ (key>>6)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>12)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b111111
    return hashval
def egressVlanTranslation_small_hash( outermostVidType,
                                      outermostVid,
                                      dstPort ):
    """ Calculate index into egressVlanTranslation_small hash table from
        the different fields. The parameter must be an integer. """

    return calc_egressVlanTranslation_small_hash( outermostVidType=outermostVidType,
                                                  outermostVid=outermostVid,
                                                  dstPort=dstPort )



def calc_egressVlanTranslation_large_hash( outermostVidType,
                                           outermostVid,
                                           dstPort ):
    """ key: 19 bits hash key
        fold count = 3
        returns: 9 bits hash value
    """
    key = 0
    key = key << 1 | (outermostVidType & 0x1)
    key = key << 12 | (outermostVid & 0xfff)
    key = key << 6 | (dstPort & 0x3f)
    hashval = key & 0b111111111
    hashval = hashval ^ (key>>9)
    hashval = hashval & 0b111111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b111111111
    return hashval
def egressVlanTranslation_large_hash( outermostVidType,
                                      outermostVid,
                                      dstPort ):
    """ Calculate index into egressVlanTranslation_large hash table from
        the different fields. The parameter must be an integer. """

    return calc_egressVlanTranslation_large_hash( outermostVidType=outermostVidType,
                                                  outermostVid=outermostVid,
                                                  dstPort=dstPort )
```

```
def egressVlanTranslation_hash_test():
    dstPort = 19
    outermostVid = 639
    outermostVidType = 1

    hashval = egressVlanTranslation_small_hash( outermostVidType=outermostVidType,
                                                outermostVid=outermostVid,
                                                dstPort=dstPort)
    assert hashval == 36

    hashval = egressVlanTranslation_large_hash( outermostVidType=outermostVidType,
                                                outermostVid=outermostVid,
                                                dstPort=dstPort)
    assert hashval == 413
```

Chapter 15

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 15.1.

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.

15.1 Functions using D-left

The following functions use D-left Lookup.

15.1.1 Egress VLAN Translation

The Egress VLAN Translation table:

- The hash tables are **Egress VLAN Translation Small Table** and **Egress VLAN Translation Large Table**. Each of the the hash tables has 2 buckets for each hash index.
- The search data/hash key is the egress port, the outermost VID and the outermost VID Type, a C-tag (0) or S-tag (1).
- The TCAM is **Egress VLAN Translation TCAM**.
- The hash functions used to index the **Egress VLAN Translation Small Table** and **Egress VLAN Translation Large Table** are described in section Hash function for Egress VLAN Translation.
- The masks for the hash tables are **Egress VLAN Translation Search Mask**.
- The configuration for resolving multiple hits is in **Egress VLAN Translation Selection**.

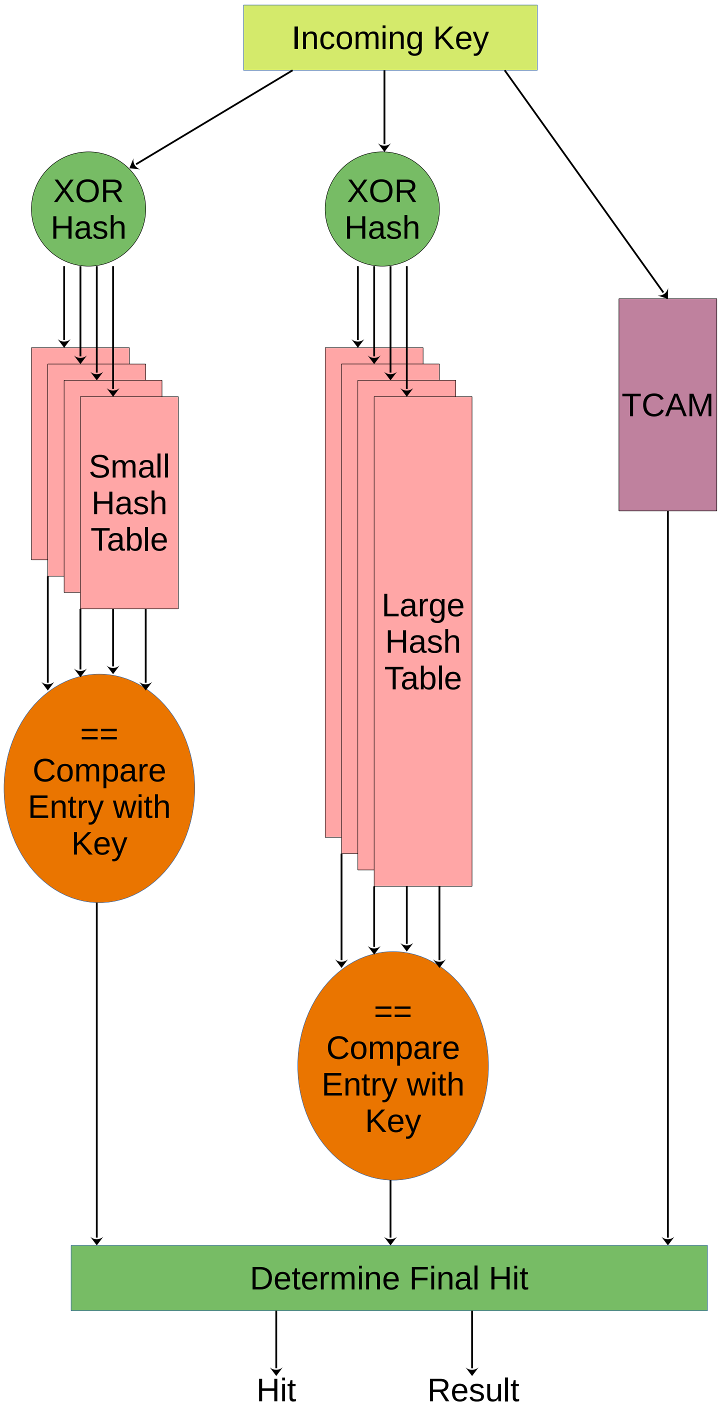

Figure 15.1: D-left Function

- While the hash tables stores the answer in the same memories as the lookup key, the TCAM has a seperate table holding the answer: **Egress VLAN Translation TCAM Answer**.

15.1.2 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**
  - **Ingress Configurable ACL 2 Small Table**
  - **Ingress Configurable ACL 2 Large Table**
  - **Ingress Configurable ACL 3 Small Table**
  - **Ingress Configurable ACL 3 Large Table**
- 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**
  - **Ingress Configurable ACL 2 Search Mask**
  - **Ingress Configurable ACL 3 Search Mask**
- Configuration for resolving multiple hits
  - **Ingress Configurable ACL 0 Selection**
  - **Ingress Configurable ACL 1 Selection**
  - **Ingress Configurable ACL 2 Selection**
  - **Ingress Configurable ACL 3 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**

Chapter 16

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.

16.1 L2 Forwarding Information Base (FIB)

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

16.1.1 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.

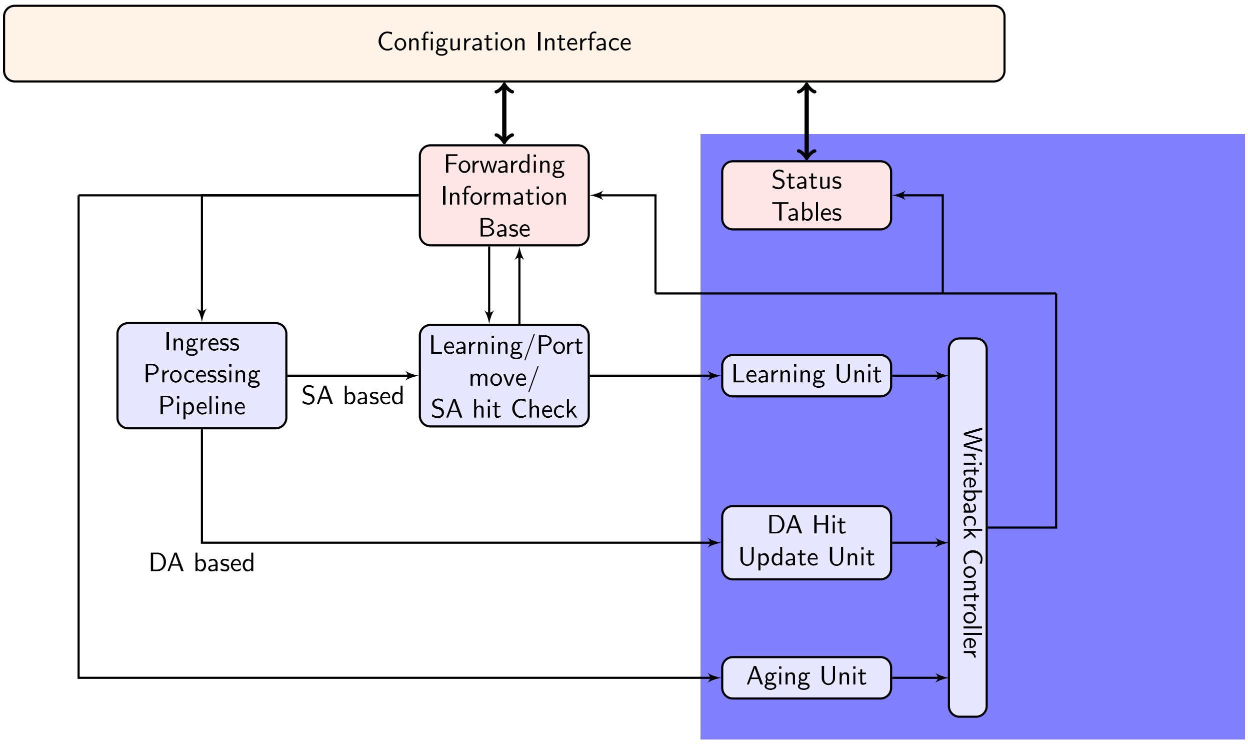

Figure 16.1: Learning and Aging Engine

(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 12-bit hash value.

16.1.2 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 16.1 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. Unexpected behavior will occur if the tables do not have the same content.

16.1.3 Hash Collision Accommodation

In order to solve hash collisions, the **L2 DA Hash Lookup Table** has 8 buckets each with 4,096 entries. A given key-hash pair can search in the 8 buckets in parallel by reading from the address that equals the hash value. The 8 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 8 keys, the first 8 keys can be accommondated in the 8 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 15-bit hit index with the hash value in the lower 12 bits and the bucket number in the higher 3 bits. The corresponding **L2 Destination Table** address equals the hit index.
- Hit in the **L2 Lookup Collision Table**: get a 6-bit hit index from the hit entry address. The corresponding **L2 Destination Table** address is (hit index + 32,768).

16.2 Hardware Learning and Aging

16.2.1 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 accomondate 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 sends decisions to the FIB.

16.2.2 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 varity 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.
  - If the packet hit in the **Configurable ACL Engine** with **noLearning** enabled.
- 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**.

[1] Check all reason codes in Table 28.2

[2] See more in Chapter Spanning Tree.

16.2.3 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 based [3] 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 16.1. 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 16.1: Hardware Aging Operations

16.2.4 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 the FIB.

According to Table 16.1, the automatic **hit** bit update for an non-static L2 entry will keep the hardware aging unit away from setting the **valid** bit to 0, hence avoid aging out the entry.

16.3 Software Learning and Aging

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

16.3.1 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

[3] The system ticks are described in Chapter Tick.

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.

16.3.2 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.

Chapter 17

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 28

17.1 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.

17.2 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.

17.3 Spanning Tree Drop Counters

When a port's ingress or egress spanning tree states causes a frame to be dropped, the frames direction and spanning-tree state are used to select which drop counter 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. There is one egress register per port.

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 18

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

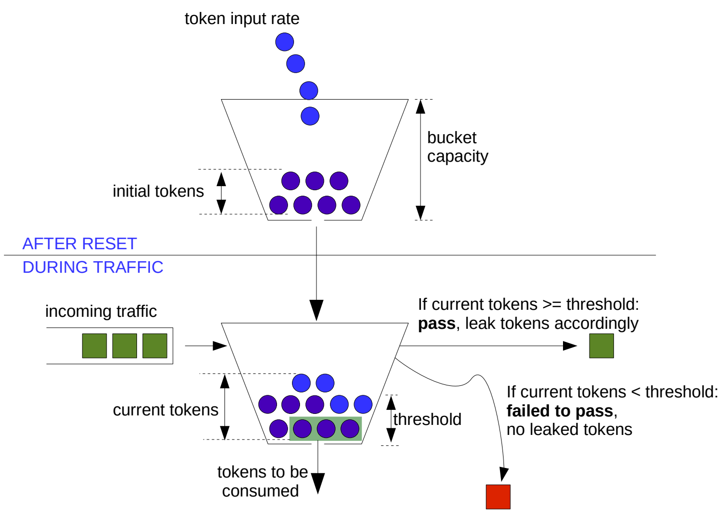

Figure 18.1: General Token Bucket Illustration

Figure 18.1 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 six core ticks.

Chapter 19

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 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 queue[1].

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 assiged. 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 higher[2] 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.

19.1 Determine Egress Queue

Figure 19.1 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**

[1] That is unless the packet is to be dropped, because then the pointer is instead added to the end of the throw queue.
[2] Priorities are numbered backward, so zero is the highest priority

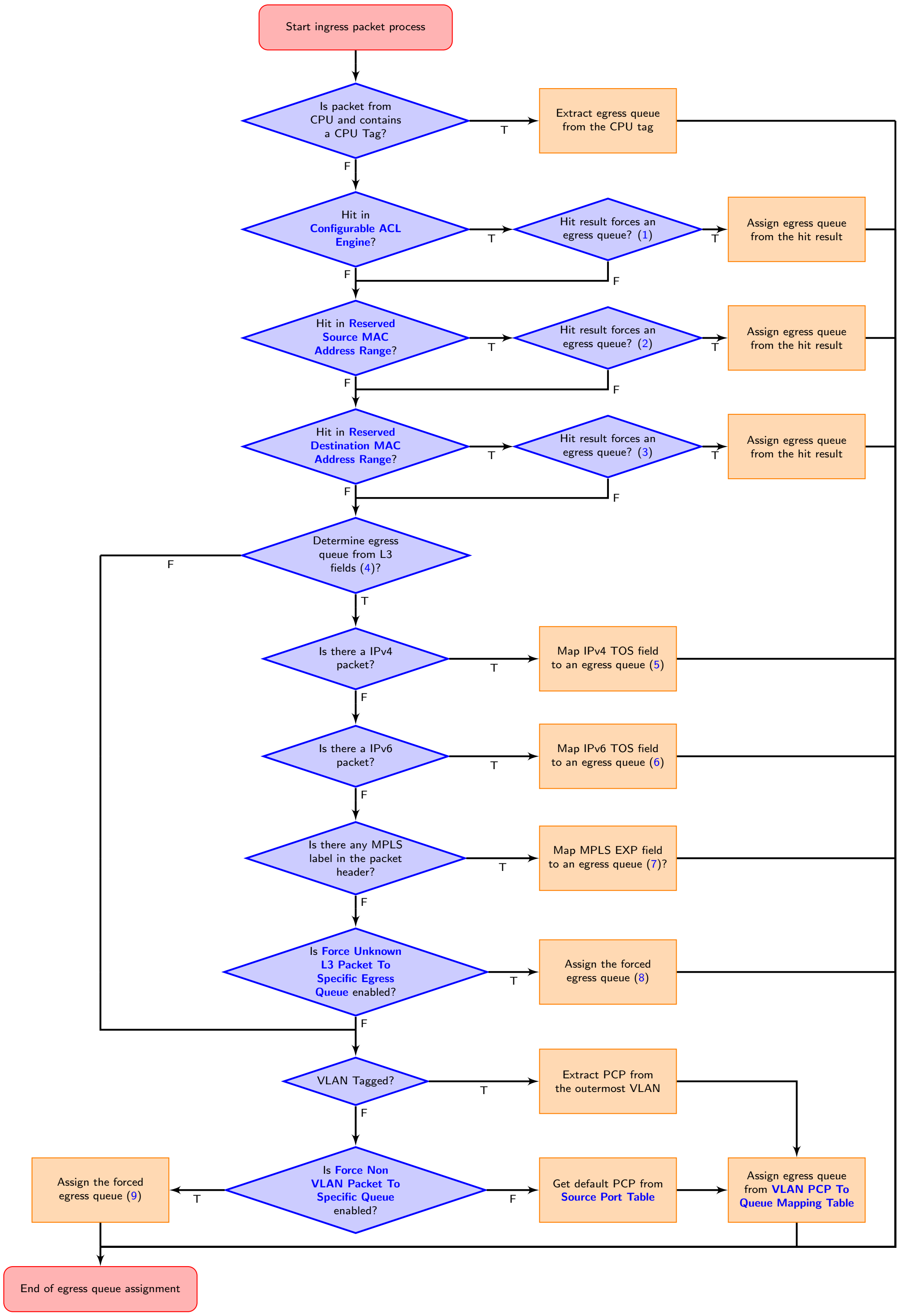

Figure 19.1: Egress Queue Selection Diagram

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:

- Packet L2 headers
- Packet L3 headers
- 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.

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

19.2.1 Remap Egress Queue to Packet Headers

This core supports remapping determined egress queues to outgoing packets' headers.

- 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.

19.3 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 19.6, 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.

19.4 Shapers

For a queue to be eligable 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.

Additionaly 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.

19.4.1 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.

19.4.2 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.

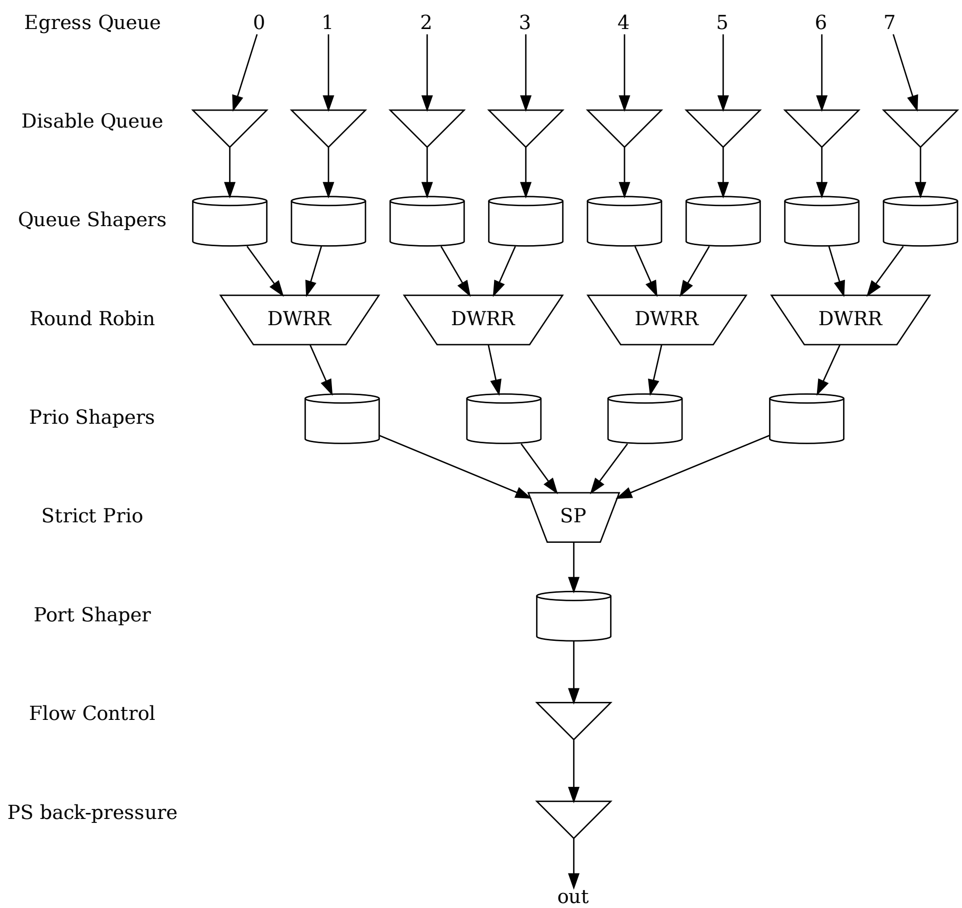

Figure 19.2: Egress Queue Scheduling example. Here using half the priorities, with two queues mapped to each.

19.5 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 19.2. This figure is an example where half the priorities are used and two queues map to each priority[3].

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).

19.6 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** $<< X$, 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.

$X = max(0, highestSetBit(cellBytes) - highestSetBit(weigth))$

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.

19.7 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

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

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.

19.8 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 20

Packet Coloring

20.1 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 20.1.

Color	Code
Green	0
Yellow	1
Red	2

Table 20.1: 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 19.1.

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 20.1 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.

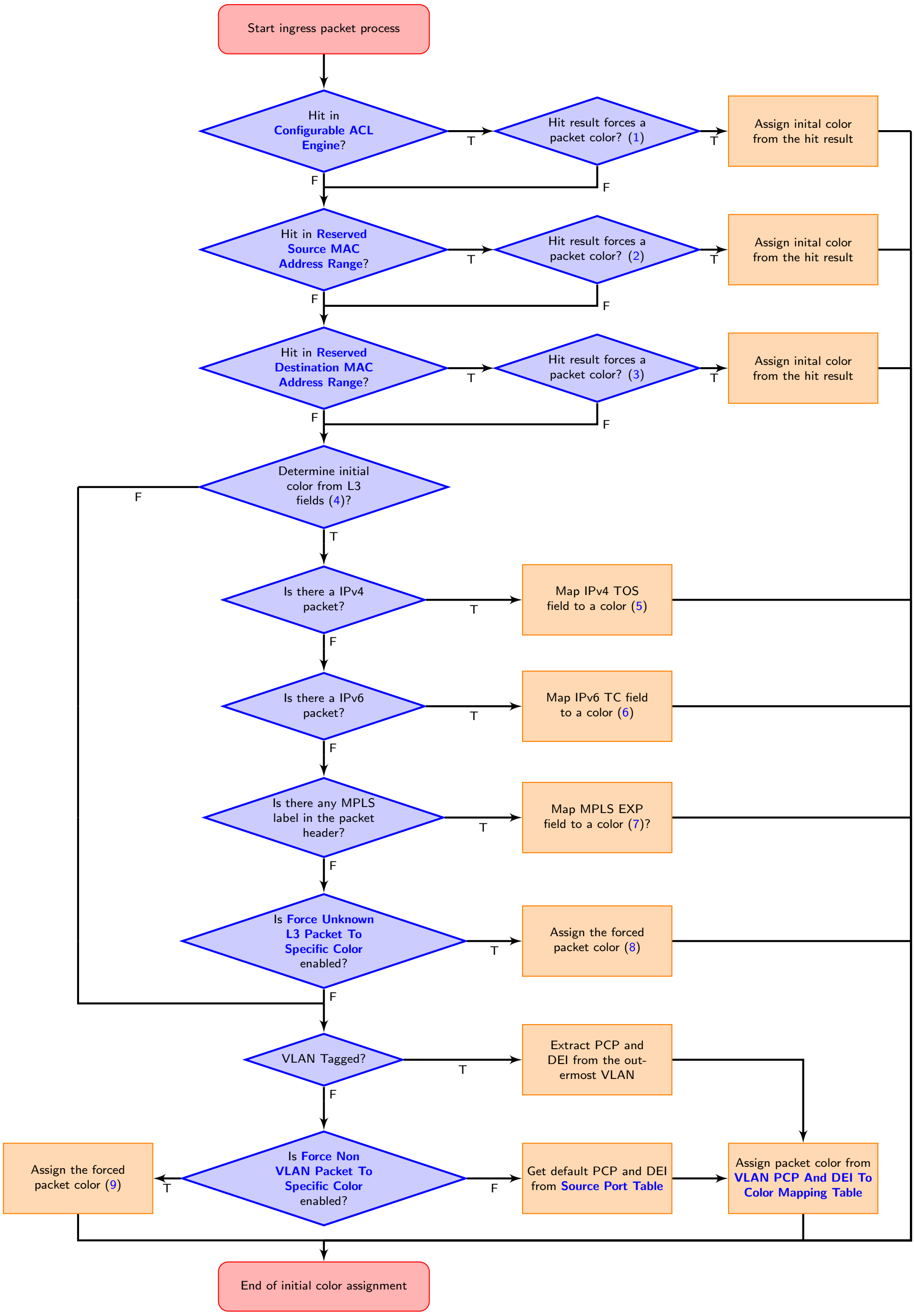

Figure 20.1: Packet Initial Color Selection Diagram

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.

20.2 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.

  ```
  tos[i] = ( color2Tos[i] & tosMask[i] ) | ( tos[i] & (~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 21

Admission Control

21.1 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.

21.1.1 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 21.1. 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**.

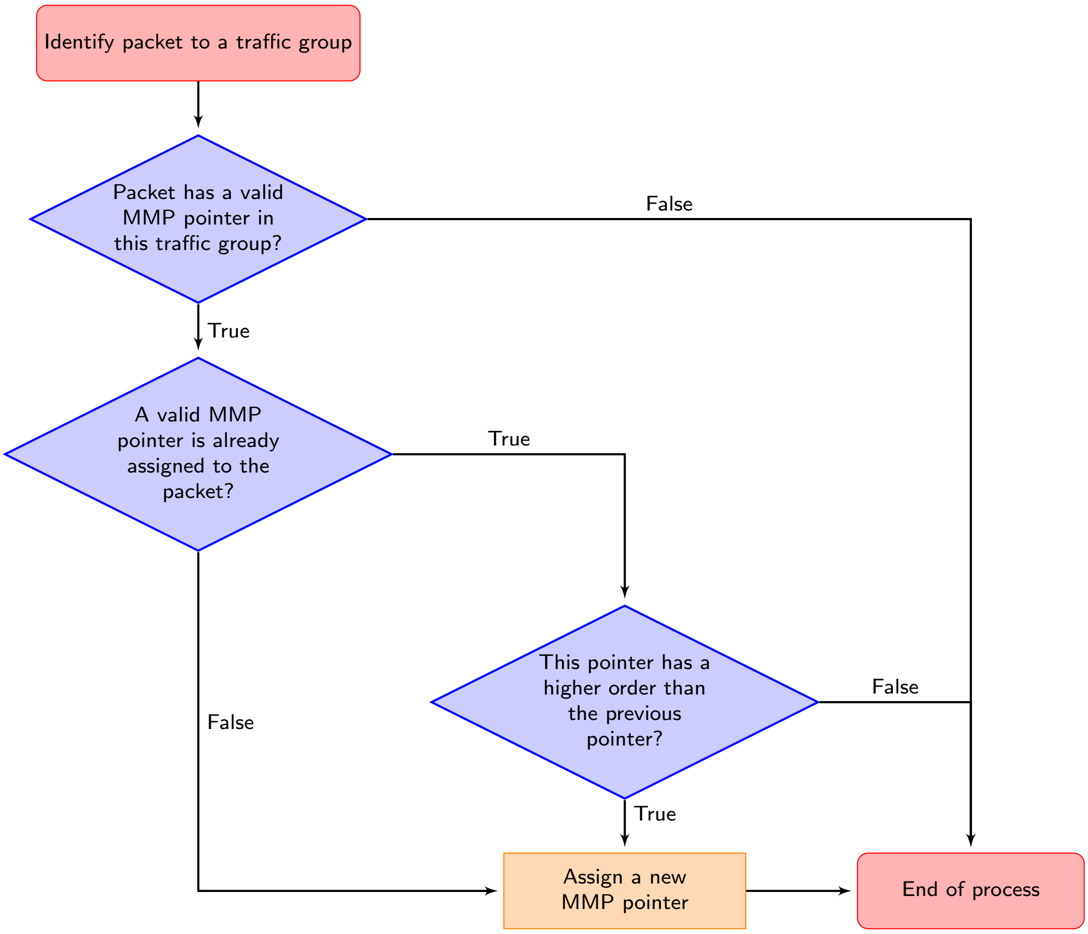

Figure 21.1: MMP pointer Selection Diagram

5. Ingress VID:
   Lookup in **VLAN Table** based on the **ingress VID**.

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.

21.2 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 128 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 21.1. 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 21.1: Rate Configuration Example (Assume tickFreqList = [1MHz, 100KHz, 10KHz, 1KHz, 100Hz])

Chapter 22

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 (six in total) are created by dividing the previuous 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 23

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 18.1). 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 four kinds of traffic are checked by the MBSC unit:

- L2 Broadcast
- L2 Unknown Multicast Flooding
- L2 Unknown Unicast 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).

23.1 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 Unknown Multicast: A Packet that will be L2 switchecd but the DA is unknown. The unknown DA MAC has Ethernet multicast bit set to 1. In this case the packet is flooded to all VLAN member ports.
    - Token bucket configurations:
        * **L2 Unknown Multicast Storm Control Enable**
        * **L2 Unknown Multicast Storm Control Bucket Capacity Configuration**
        * **L2 Unknown Multicast Storm Control Bucket Threshold Configuration**
        * **L2 Unknown Multicast Storm Control Rate Configuration**
- L2 Unknown Unicast: A Packet that will be L2 switchecd but the DA is unknown. The unknown DA MAC has Ethernet multicast bit set to 0. In this case the packet is flooded to all VLAN member ports.
    - Token bucket configurations:
        * **L2 Unknown Unicast Storm Control Enable**
        * **L2 Unknown Unicast Storm Control Bucket Capacity Configuration**
        * **L2 Unknown Unicast Storm Control Bucket Threshold Configuration**
        * **L2 Unknown Unicast 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**

23.2 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 tick [1] 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    = 1 # switch slices
ifg      = 20 # bytes
pnb      = 1 # = packet mode
portBW   = 1000 # Mbits/s
tickFreqList = [1.0,
                0.1,
                0.01,
                0.001,
                0.0001,
                1e-05] # Mhz

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}
```

[1] The system ticks are described in Chapter 22.

Chapter 24

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 enqueuing at other ports and queues. Additionally, during buffer memory congestion, ERM facilitates prioritized enqueuing of egress queues with higher priorities.

The resource management granularity is cells and there are 13466 cells, each 160 byte wide, available in the buffer memory. A packet is written to the buffer memory with the original packet data plus a 24 byte ingress to egress header, thus a 1600 byte packet will have 1624 bytes and occupy ten 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 137 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 27 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.

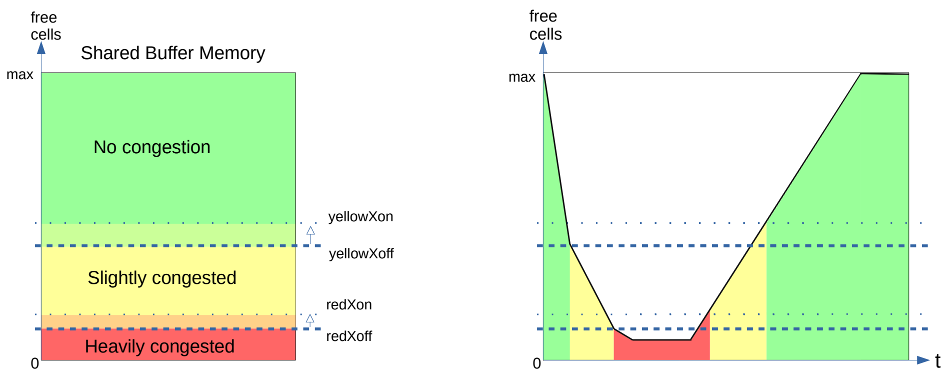

Figure 24.1: Buffer memory congestion zones

24.1 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**.

24.2 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 enqueuing 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**.

24.3 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.

24.4 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 enqueuing decision is made based on the overall free space in the buffer memory. This configuration imposes the following requirements:

1. $\texttt{redXon} \geq \texttt{redXoff} \geq sum(\texttt{redLimit})$
   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. $\texttt{yellowXon} \geq \texttt{yellowXoff} \geq \texttt{maxBufferFree}$
   Put the ERM in the yellow zone even when the buffer memory is empty hence keep **`yellowLimit`** check under an always on state.

24.5 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 enqueuing snapshot to the ERM and cause inconsistencies that can not be recovered without a reset.

Chapter 25

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 shall be able to behave in a way that punishes other source ports. For this purpose flow control has two tools at its disposition: Pausing and tail-drop.

25.1 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 29.4.

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.

25.2 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.

25.2.1 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.

25.3 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.

The number of FFA cells that are are allowed to be consumed by each source port before it will be hit by flow control is configured individually per source port. 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.

25.3.1 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 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.

The **Port Reserved** registers define the number of cells reserved per source port.

25.3.2 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.
- **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.

Each source port 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 the set to un-paused.

25.3.3 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

The **Tail-Drop FFA Threshold** is not obeyed strictly. The first packet exceeding the threshold may be accepted, causing a one-packet over-shoot.

25.3.4 Counters

These are the counters that the thresholds are compared to:

- **FFA Used**: The total number of cells used from the FFA area.
- **Port Used**: The total number of cells used for each port (FFA+reserved).

25.4 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 25.3.2 above.

25.5 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 25.3.2 above.

25.6 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.

25.7 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.

25.8 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 26

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 18.1 (despite what the illustration says the shaper will of course never drop any packets).

Chapter 27

Statistics

Short Name	Register Name
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. ipppDrop	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 Expired TTL Drop IP Checksum Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Attack Prevention 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 GRE Decoder Drop L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop L2 Destination Table SA Lookup Drop Source Port Default ACL Action Drop
11. smon	SMON Set 0 Packet Counter

Short Name	Register Name
	SMON Set 1 Packet Counter SMON Set 2 Packet Counter SMON Set 3 Packet Counter SMON Set 4 Packet Counter SMON Set 5 Packet Counter SMON Set 6 Packet Counter SMON Set 7 Packet Counter SMON Set 8 Packet Counter SMON Set 9 Packet Counter SMON Set 10 Packet Counter SMON Set 11 Packet Counter SMON Set 12 Packet Counter SMON Set 13 Packet Counter SMON Set 14 Packet Counter SMON Set 15 Packet Counter SMON Set 0 Byte Counter SMON Set 1 Byte Counter SMON Set 2 Byte Counter SMON Set 3 Byte Counter SMON Set 4 Byte Counter SMON Set 5 Byte Counter SMON Set 6 Byte Counter SMON Set 7 Byte Counter SMON Set 8 Byte Counter SMON Set 9 Byte Counter SMON Set 10 Byte Counter SMON Set 11 Byte Counter SMON Set 12 Byte Counter SMON Set 13 Byte Counter SMON Set 14 Byte Counter SMON Set 15 Byte Counter
11. ippAcl	Ingress 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
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
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

Short Name	Register Name
25. psTxPkt	**PS Packet Head Counter** **PS Packet Tail Counter**
25. psError	**PS Error Counter**

Table 27.1: Sequence of Statistics Counters

This core supports full statistics with 32-bit wrap around counters. The statistics is divided into groups depending on the type of statistics and location in the switch. Figure 27.1 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 27.1 can be found through Table 27.1.

27.1 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.

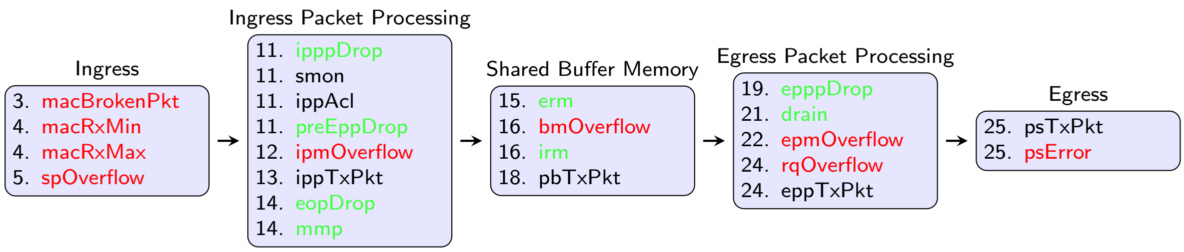

Figure 27.1: Location of Statistics Counters

27.2 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.

27.3 SMON Statistics

There are 16 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.

27.4 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.

  There are no packet statistics in the RX MAC clock domain.

- 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.

- 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.

27.5 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.

27.6 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.

27.6.1 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 28

Packets To And From The CPU

The CPU port (number 52 by default) 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.

28.1 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
0-6	[52:0] 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 6. 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.
7	Bits [2:0] specifies which egress queue the packet shall use.
8	Bit [0] will set the *upd_ts* signal on the transmit MAC interface when the packet is transmitted. Bit [1] will set the *upd_cf* signal on the transmit MAC interface when the packet is transmitted. Bit [2] will set the *ts_to_sw* signal on the transmit MAC interface when the packet is transmitted.
9-16	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.

Table 28.1: From CPU tag format

The header consists of a specific Ethernet Type (39065) followed by a CPU Tag. The CPU tag has a 7 byte(s) destination port mask field[1] and 1 byte egress queue field (encoded as specified in table 28.1). 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 28.1.

[1]The ordering described in 28.1 is the receive/transmit order.

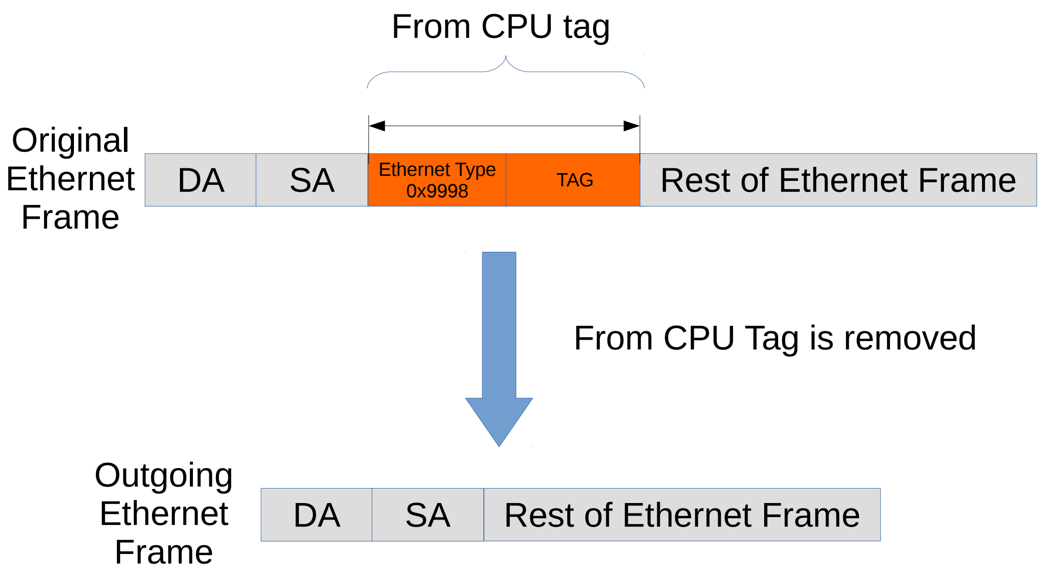

Figure 28.1: Packet from CPU with CPU tag

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.

28.1.1 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 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.
- Meter-Marker-Policer check is done.
- MBSC is bypassed.
- All spanning tree and multiple spanning treeperations are bypassed.
- No learning operation.
- Check Reserved DMAC is done.
- Check Reserved SMAC is done.
- ACL operations are done.

- ACL statistics are done.
- SMON statistics is done.

28.2 Packets To the CPU

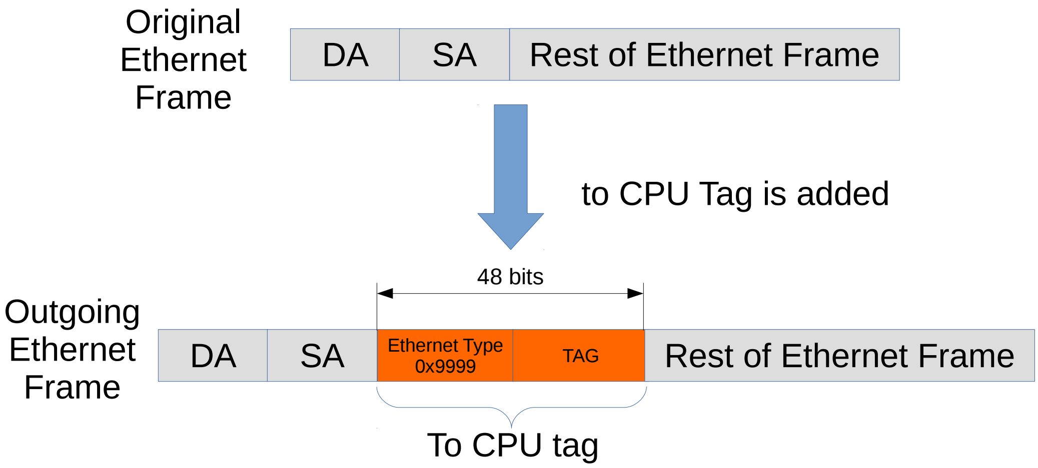

Figure 28.2: Packet to CPU with CPU tag

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 28.2). The header indicates the reason that the packet was sent to the CPU, and the port on which it was received. Packets which arrives on the CPU Port are modified according to what actions the packet was subjected to one example is VLAN header modifications.

When packets are sent to the CPU port (number 52 in this core), the packets are tagged with a specific Ethernet Type (type 39321). Figure 28.2 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.

Byte Number	Contents of Byte
0	Bits [5:0] contains the source port where the packet entered the switch.
1 to 2	Reason for packet sent to CPU. See table 28.3. Byte 1 is the msb of the reason code.
3	PTP bit, if bit 0 is set to one then the packet is a PTP packet and the Timestamp field is valid.
4 to 11	Timestamp (64 bits). The lowest numbered byte contains the msb of the timestamp value.

Table 28.2: To CPU tag format

28.2.1 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 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.
12 + HitIndex	The Destination MAC range sent the packet to the CPU..Index to rule.
20 + HitIndex	The source port default ACL action sent the packet to the CPU..Index to source port which sent the packet in.
73 + HitIndex	The TCAM in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
105 + HitIndex	The small table in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
361 + HitIndex	The large table in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
2409 + HitIndex	The TCAM in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
2425 + HitIndex	The small table in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
2553 + HitIndex	The large table in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
3577 + HitIndex	The TCAM in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
3593 + HitIndex	The small table in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
3657 + HitIndex	The large table in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
4169 + HitIndex	The TCAM in the configurable ingress ACL engine 3 sent the packet to the CPU..Index to rule.
4185 + HitIndex	The small table in the configurable ingress ACL engine 3 sent the packet to the CPU..Index to rule.
4249 + HitIndex	The large table in the configurable ingress ACL engine 3 sent the packet to the CPU..Index to rule.
4505	This is an L2 1588 frame.
4506	This is an L4 1588 frame.
4507	This is an ARP frame.
4508	This is an RARP frame.
4509	This is an LLDP frame.
4510	This is an 802.1X EAPOL frame.
4511	This is an GRE frame.
4512	This is an SCTP frame.
4513	This is an LCAP frame.
4514	This is an AH frame.
4515	This is an ESP frame.
4516	This is an DNS frame.
4517	This is a BOOTP or DHCP frame.
4518	This is an CAPWAP frame.
4519	The IP TTL field was expired in the packet.
4520	Packet matched an L2 Multicast Reserved Address
4521	The **L2 Action Table** has determined that this packet shall be sent to the CPU.

Table 28.3: Reason for packet sent to CPU

The possible reasons are listed in Table 28.3.

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.

Chapter 29

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.

29.1 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). The one *clk_mult* are higher frequency clocks, syncronous with the core clock, that are used in a few places in the core where a higher clock gives a substantial area savings.

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 immidiately 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]

[1] Thus the packet interface resets cannot be used to empty a specific packet interface. To do that, follow the procedure in Section 19.8, while making sure that the packet interface halt is kept low.

Signal Name	Size	In Out	Description
clk	1	In	Core clock. For 98 Gbit/s wire-speed throughput use a core clock frequency of 156.25 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**
clk_mult_0	1	In	A 312.5MHz clock, synchronous with the core clock.
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 29.1: 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 29.1. 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.

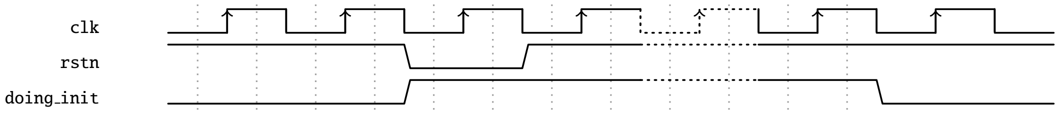

Figure 29.1: Core Initialization

29.1.1 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.

29.2 Packet Interface

There are 53 packet interfaces, or ports for short, each divided into a reception part and a transmission part. The ports are numbered from 0 to 52.

Pin	Size	Direction	Description
rx_axis_tvalid_N	1	In	Set high to indicate that the current bus cycle is valid. The core must accept the data, there is no backpressure mechanism.
rx_axis_tlast_N	1	In	End-of-packet flag. Indicates that the current bus cycle contains the last data transfer for the packet. This is the only time a partially-filled data word is permitted.
rx_axis_tdata_N	8	In	Packet data.
rx_axis_tkeep_N	1	In	A per-byte data valid indication for the last word. Only valid when tlast is high. If tkeep[0] is high, tdata[7:0] is valid; if tkeep[1] is high, tdata[15:8] is also valid; and so on and so forth. The axis_tkeep port shall be connected to the LSBś of axis_tkeep_user.
rx_axis_tuser_N	1	In	Error indication for the packet. Valid only when tlast is high. The axis_tuser port shall be connected to the MSB of axis_tkeep_user.

Table 29.2: Packet RX interface for ports 0-47. **N** is the ingress interface number.

The port interfaces are not all the same. There are two different port interface variants in this core, each with an RX and a TX direction:

1. Ports 0-47: RX-interface see Table 29.2 on page 165, TX-interface see Table 29.3 on page 166
2. Ports 48-52: RX-interface see Table 29.4 on page 167, TX-interface see Table 29.5 on page 168

Each direction of a packet interface consists of *tvalid*, *tlast*, *tkeep*, *tdata* and *tuser* fields. The transmit direction has an additional *tready* signal to allow the receiving end to moderate the data rate transmitted from the core.

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 last byte indicated by *tkeep*. Unless the *tlast* 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 *tvalid* field is high and the *valid_bytes* field is non-zero, and ends with a data transmission where the *tlast* field is high. Idle transactions—where tkeep, *tvalid* and *tlast* 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 16359 bytes. If this length was allowed to be exceeded either on the ingress or the egress it would corrupt the internal counters.

Pin	Size	Direction	Description
tx_axis_tvalid_N	1	Out	Set high to indicate that the current bus cycle is valid.
tx_axis_tlast_N	1	Out	End-of-packet flag. Indicates that the current bus cycle contains the last data transfer for the packet. This is the only time a partially-filled data word is permitted.
tx_axis_tdata_N	8	Out	Packet data.
tx_axis_tkeep_N	1	Out	A per-byte data valid indication for the last word. Only valid when tlast is high. If tkeep[0] is high, tdata[7:0] is valid; if tkeep[1] is high, tdata[15:8] is also valid; and so on and so forth. The axis_tkeep_user signal is created by concatenating {axis_tuser,axis_tkeep}.
tx_axis_tuser_N	1	Out	Error indication for the packet. Valid only when tlast is high.
tx_axis_tready_N	1	In	Driven by the MAC to indicate that the interface is able to accept the data currently present on the bus. If the tready signal deasserts during a transfer, the current data on the bus must be held until tready is asserted again.

Table 29.3: Packet TX interface for ports 0-47. **N** is the egress interface number.

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 16359-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 9600-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 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 *tready* interface to prevent overflows.

Broken packets

A packet ending with *tuser* set high 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.

Pin	Size	Direction	Description
rx_axis_tvalid_**N**	1	In	Set high to indicate that the current bus cycle is valid. The core must accept the data, there is no backpressure mechanism.
rx_axis_tlast_**N**	1	In	End-of-packet flag. Indicates that the current bus cycle contains the last data transfer for the packet. This is the only time a partially-filled data word is permitted.
rx_axis_tdata_**N**	32	In	Packet data.
rx_axis_tkeep_**N**	4	In	A per-byte data valid indication for the last word. Only valid when tlast is high. If tkeep[0] is high, tdata[7:0] is valid; if tkeep[1] is high, tdata[15:8] is also valid; and so on and so forth. The axis_tkeep port shall be connected to the LSBś of axis_tkeep_user.
rx_axis_tuser_**N**	1	In	Error indication for the packet. Valid only when tlast is high. The axis_tuser port shall be connected to the MSB of axis_tkeep_user.

Table 29.4: Packet RX interface for ports 48-52. **N** is the ingress interface number.

All broken packets are counted in the **MAC RX Broken Packets**.

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 *tkeep* 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 *tkeep* can indicate less than the full bus width. In this case the byte order is still the same as previous transfers. For example when *tkeep* is 1 the last byte of the packet is placed on bits [7:0] and with *tkeep* of 3 the last byte of the packet is placed on bits [15:8] and the second to last is on bits [7:0].

29.3 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 29.6 below.

Pin	Size	Direction	Description
tx_axis_tvalid_N	1	Out	Set high to indicate that the current bus cycle is valid.
tx_axis_tlast_N	1	Out	End-of-packet flag. Indicates that the current bus cycle contains the last data transfer for the packet. This is the only time a partially-filled data word is permitted.
tx_axis_tdata_N	32	Out	Packet data.
tx_axis_tkeep_N	4	Out	A per-byte data valid indication for the last word. Only valid when tlast is high. If tkeep[0] is high, tdata[7:0] is valid; if tkeep[1] is high, tdata[15:8] is also valid; and so on and so forth. The axis_tkeep_user signal is created by concatenating {axis_tuser,axis_tkeep}.
tx_axis_tuser_N	1	Out	Error indication for the packet. Valid only when tlast is high.
tx_axis_tready_N	1	In	Driven by the MAC to indicate that the interface is able to accept the data currently present on the bus. If the tready signal deasserts during a transfer, the current data on the bus must be held until tready is asserted again.

Table 29.5: Packet TX interface for ports 48-52. **N** is the egress interface number.

Pin	Size	Direction	Description
apb_paddr	24	In	Address. This is the APB address bus. The highest address bit (23) on the APB bus is not a normal address bit and is referred to as the Accumulator Bit. This is described further in section 30.
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	32	In	Write data.
apb_pready	1	Out	Ready. The slave uses this signal to extend an APB transfer.
apb_prdata	32	Out	Read Data.
apb_pslverr	1	Out	Error. This signal indicates a transfer failure.

Table 29.6: The APB interface signals

The *paddr* is a byte address, however the core only supports accessing complete 32-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

29.4 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 syncronized to the MAC clock if connected to the MAC. However the interfaces can be though 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.

29.4.1 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.

29.4.2 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.

Pin	Direction	Size	Description
iext_pause_**N**	In	1	Xoff=1, Xon=0.
opfc_status_**N**	Out	1	Xoff=1, Xon=0.

Table 29.7: ThePFC status and External Pause interfaces, where **N** is the packet interface number

29.5 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 29.9. Both these signals are pipelined.

Pin	Direction	Size	Description
debug_read_select	In	11	Selects the signal to monitor. See Table 29.9.
debug_read_data	In	32	The debug output data.

Table 29.8: The Debug Read interface

id	instance	signal
0	pa_top.switch.mactop	constant-0
1	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
2	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
3	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
4	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
5	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
6	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
7	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
8	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
9	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
10	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
11	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
12	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
13	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
14	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
15	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
16	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
17	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}

id	instance	signal
18	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
19	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
20	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
21	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
22	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
23	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
24	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
25	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
26	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
27	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
28	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
29	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
30	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
31	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
32	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
33	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
34	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
35	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
36	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
37	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
38	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
39	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
40	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
41	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
42	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
43	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
44	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
45	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
46	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
47	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
48	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
49	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
50	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
51	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
52	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
53	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
54	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
55	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
56	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
57	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
58	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
59	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
60	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
61	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
62	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
63	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
64	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
65	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
66	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
67	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
68	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
69	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
70	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
71	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
72	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
73	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
74	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
75	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
76	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
77	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
78	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
79	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
80	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
81	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
82	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
83	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
84	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
85	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
86	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
87	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
88	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
89	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
90	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
91	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
92	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
93	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
94	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
95	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
96	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
97	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}

id	instance	signal
98	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
99	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
100	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
101	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
102	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
103	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
104	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
105	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
106	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
107	—"—	constant-107
108	pa_top.switch.ipp0	constant-108
109	—"—	ipp_ipkt_bus {16'data, 8'valid_bytes, 6'id, 1'last, 1'first}
110	—"—	ipp_opkt_bus {16'data, 8'valid_bytes, 6'id, 1'last, 1'first}
111	—"—	pass_da_0
112	—"—	pass_da_1
113	—"—	dut_iIpp_iDropper_dbg_drop
114	—"—	dut_iIpp_iDropper_dbg_ifirst
115	—"—	dut_iIpp_iDropper_dbg_ilast
116	—"—	pass_sa_0
117	—"—	pass_sa_1
118	—"—	constant-118
119	pa_top.switch.ipp0.pm	constant-119
120	—"—	pm_fifo_overflow
121	—"—	dut_dbg_fifo_full
122	—"—	halt_from_pm
123	—"—	dut_iFifoa_debug_in
124	—"—	dut_iFifoa_debug_out
125	—"—	constant-125
126	pa_top.switch.sp0	constant-126
127	—"—	dut_iSpbridge_assert_reset_sp_bridge
128	—"—	dut_iSpbridge_assert_reset_sp_bridge
129	—"—	dut_iSpbridge_assert_reset_sp_bridge
130	—"—	dut_iSpbridge_assert_reset_sp_bridge
131	—"—	dut_iSpbridge_assert_reset_sp_bridge
132	—"—	dut_iSpbridge_assert_reset_sp_bridge
133	—"—	dut_iSpbridge_assert_reset_sp_bridge
134	—"—	dut_iSpbridge_assert_reset_sp_bridge
135	—"—	dut_iSpbridge_assert_reset_sp_bridge
136	—"—	dut_iSpbridge_assert_reset_sp_bridge
137	—"—	dut_iSpbridge_assert_reset_sp_bridge
138	—"—	dut_iSpbridge_assert_reset_sp_bridge
139	—"—	dut_iSpbridge_assert_reset_sp_bridge
140	—"—	dut_iSpbridge_assert_reset_sp_bridge
141	—"—	dut_iSpbridge_assert_reset_sp_bridge
142	—"—	dut_iSpbridge_assert_reset_sp_bridge
143	—"—	dut_iSpbridge_assert_reset_sp_bridge
144	—"—	dut_iSpbridge_assert_reset_sp_bridge
145	—"—	dut_iSpbridge_assert_reset_sp_bridge
146	—"—	dut_iSpbridge_assert_reset_sp_bridge
147	—"—	dut_iSpbridge_assert_reset_sp_bridge
148	—"—	dut_iSpbridge_assert_reset_sp_bridge
149	—"—	dut_iSpbridge_assert_reset_sp_bridge
150	—"—	dut_iSpbridge_assert_reset_sp_bridge
151	—"—	dut_iSpbridge_assert_reset_sp_bridge
152	—"—	dut_iSpbridge_assert_reset_sp_bridge
153	—"—	dut_iSpbridge_assert_reset_sp_bridge
154	—"—	dut_iSpbridge_assert_reset_sp_bridge
155	—"—	dut_iSpbridge_assert_reset_sp_bridge
156	—"—	dut_iSpbridge_assert_reset_sp_bridge
157	—"—	dut_iSpbridge_assert_reset_sp_bridge
158	—"—	dut_iSpbridge_assert_reset_sp_bridge
159	—"—	dut_iSpbridge_assert_reset_sp_bridge
160	—"—	dut_iSpbridge_assert_reset_sp_bridge
161	—"—	dut_iSpbridge_assert_reset_sp_bridge
162	—"—	dut_iSpbridge_assert_reset_sp_bridge
163	—"—	dut_iSpbridge_assert_reset_sp_bridge
164	—"—	dut_iSpbridge_assert_reset_sp_bridge
165	—"—	dut_iSpbridge_assert_reset_sp_bridge
166	—"—	dut_iSpbridge_assert_reset_sp_bridge
167	—"—	dut_iSpbridge_assert_reset_sp_bridge
168	—"—	dut_iSpbridge_assert_reset_sp_bridge
169	—"—	dut_iSpbridge_assert_reset_sp_bridge
170	—"—	dut_iSpbridge_assert_reset_sp_bridge
171	—"—	dut_iSpbridge_assert_reset_sp_bridge
172	—"—	dut_iSpbridge_assert_reset_sp_bridge
173	—"—	dut_iSpbridge_assert_reset_sp_bridge
174	—"—	dut_iSpbridge_assert_reset_sp_bridge
175	—"—	dut_iSpbridge_assert_reset_sp_bridge
176	—"—	dut_iSpbridge_assert_reset_sp_bridge
177	—"—	dut_iSpbridge_assert_reset_sp_bridge

id	instance	signal
178	—"—	dut_iSpbridge_assert_reset_sp_bridge
179	—"—	dut_iSpbridge_assert_reset_sp_bridge
180	—"—	constant-180
181	pa_top.switch.pb0	constant-181
182	—"—	dut_iPbu_debug_refc_inc
183	—"—	dut_iPbu_debug_port_sch
184	—"—	dut_iPbu_dmux_wrr
185	—"—	dut_iPbu_debug_qenext
186	—"—	dut_iPbu_assert_qediff
187	—"—	dut_iPbu_assert_reque_sp
188	—"—	Mask of currently receiving packets that have been broken due to BM full
189	—"—	dut_iPbu_follow_pfc_accept
190	—"—	dut_iPbu_iAssertpacket_0_assert_out
191	—"—	pa.top.switch.pb0.iAssertpacket0 {8'valid_bytes, 6'port, 1'last, 1'first}
192	—"—	dut_iPbu_iPortshaper_iBuckets_reg_stat
193	—"—	dut_iPbu_zPassdbgqeread_0_o
194	—"—	dut_iPbu_iRequeue_iReFifo_52_iF_iFifos_zFcnt_pop_empty
195	—"—	dut_iPbu_iRequeue_iReFifo_52_iF_iFifos_zFcnt_push_full
196	—"—	dut_iPbu_iRequeue_iReFifo_51_iF_iFifos_zFcnt_pop_empty
197	—"—	dut_iPbu_iRequeue_iReFifo_51_iF_iFifos_zFcnt_push_full
198	—"—	dut_iPbu_iRequeue_iReFifo_50_iF_iFifos_zFcnt_pop_empty
199	—"—	dut_iPbu_iRequeue_iReFifo_50_iF_iFifos_zFcnt_push_full
200	—"—	dut_iPbu_iRequeue_iReFifo_49_iF_iFifos_zFcnt_pop_empty
201	—"—	dut_iPbu_iRequeue_iReFifo_49_iF_iFifos_zFcnt_push_full
202	—"—	dut_iPbu_iRequeue_iReFifo_48_iF_iFifos_zFcnt_pop_empty
203	—"—	dut_iPbu_iRequeue_iReFifo_48_iF_iFifos_zFcnt_push_full
204	—"—	dut_iPbu_iRequeue_iReFifo_47_iF_iFifos_zFcnt_pop_empty
205	—"—	dut_iPbu_iRequeue_iReFifo_47_iF_iFifos_zFcnt_push_full
206	—"—	dut_iPbu_iRequeue_iReFifo_46_iF_iFifos_zFcnt_pop_empty
207	—"—	dut_iPbu_iRequeue_iReFifo_46_iF_iFifos_zFcnt_push_full
208	—"—	dut_iPbu_iRequeue_iReFifo_45_iF_iFifos_zFcnt_pop_empty
209	—"—	dut_iPbu_iRequeue_iReFifo_45_iF_iFifos_zFcnt_push_full
210	—"—	dut_iPbu_iRequeue_iReFifo_44_iF_iFifos_zFcnt_pop_empty
211	—"—	dut_iPbu_iRequeue_iReFifo_44_iF_iFifos_zFcnt_push_full
212	—"—	dut_iPbu_iRequeue_iReFifo_43_iF_iFifos_zFcnt_pop_empty
213	—"—	dut_iPbu_iRequeue_iReFifo_43_iF_iFifos_zFcnt_push_full
214	—"—	dut_iPbu_iRequeue_iReFifo_42_iF_iFifos_zFcnt_pop_empty
215	—"—	dut_iPbu_iRequeue_iReFifo_42_iF_iFifos_zFcnt_push_full
216	—"—	dut_iPbu_iRequeue_iReFifo_41_iF_iFifos_zFcnt_pop_empty
217	—"—	dut_iPbu_iRequeue_iReFifo_41_iF_iFifos_zFcnt_push_full
218	—"—	dut_iPbu_iRequeue_iReFifo_40_iF_iFifos_zFcnt_pop_empty
219	—"—	dut_iPbu_iRequeue_iReFifo_40_iF_iFifos_zFcnt_push_full
220	—"—	dut_iPbu_iRequeue_iReFifo_39_iF_iFifos_zFcnt_pop_empty
221	—"—	dut_iPbu_iRequeue_iReFifo_39_iF_iFifos_zFcnt_push_full
222	—"—	dut_iPbu_iRequeue_iReFifo_38_iF_iFifos_zFcnt_pop_empty
223	—"—	dut_iPbu_iRequeue_iReFifo_38_iF_iFifos_zFcnt_push_full
224	—"—	dut_iPbu_iRequeue_iReFifo_37_iF_iFifos_zFcnt_pop_empty
225	—"—	dut_iPbu_iRequeue_iReFifo_37_iF_iFifos_zFcnt_push_full
226	—"—	dut_iPbu_iRequeue_iReFifo_36_iF_iFifos_zFcnt_pop_empty
227	—"—	dut_iPbu_iRequeue_iReFifo_36_iF_iFifos_zFcnt_push_full
228	—"—	dut_iPbu_iRequeue_iReFifo_35_iF_iFifos_zFcnt_pop_empty
229	—"—	dut_iPbu_iRequeue_iReFifo_35_iF_iFifos_zFcnt_push_full
230	—"—	dut_iPbu_iRequeue_iReFifo_34_iF_iFifos_zFcnt_pop_empty
231	—"—	dut_iPbu_iRequeue_iReFifo_34_iF_iFifos_zFcnt_push_full
232	—"—	dut_iPbu_iRequeue_iReFifo_33_iF_iFifos_zFcnt_pop_empty
233	—"—	dut_iPbu_iRequeue_iReFifo_33_iF_iFifos_zFcnt_push_full
234	—"—	dut_iPbu_iRequeue_iReFifo_32_iF_iFifos_zFcnt_pop_empty
235	—"—	dut_iPbu_iRequeue_iReFifo_32_iF_iFifos_zFcnt_push_full
236	—"—	dut_iPbu_iRequeue_iReFifo_31_iF_iFifos_zFcnt_pop_empty
237	—"—	dut_iPbu_iRequeue_iReFifo_31_iF_iFifos_zFcnt_push_full
238	—"—	dut_iPbu_iRequeue_iReFifo_30_iF_iFifos_zFcnt_pop_empty
239	—"—	dut_iPbu_iRequeue_iReFifo_30_iF_iFifos_zFcnt_push_full
240	—"—	dut_iPbu_iRequeue_iReFifo_29_iF_iFifos_zFcnt_pop_empty
241	—"—	dut_iPbu_iRequeue_iReFifo_29_iF_iFifos_zFcnt_push_full
242	—"—	dut_iPbu_iRequeue_iReFifo_28_iF_iFifos_zFcnt_pop_empty
243	—"—	dut_iPbu_iRequeue_iReFifo_28_iF_iFifos_zFcnt_push_full
244	—"—	dut_iPbu_iRequeue_iReFifo_27_iF_iFifos_zFcnt_pop_empty
245	—"—	dut_iPbu_iRequeue_iReFifo_27_iF_iFifos_zFcnt_push_full
246	—"—	dut_iPbu_iRequeue_iReFifo_26_iF_iFifos_zFcnt_pop_empty
247	—"—	dut_iPbu_iRequeue_iReFifo_26_iF_iFifos_zFcnt_push_full
248	—"—	dut_iPbu_iRequeue_iReFifo_25_iF_iFifos_zFcnt_pop_empty
249	—"—	dut_iPbu_iRequeue_iReFifo_25_iF_iFifos_zFcnt_push_full
250	—"—	dut_iPbu_iRequeue_iReFifo_24_iF_iFifos_zFcnt_pop_empty
251	—"—	dut_iPbu_iRequeue_iReFifo_24_iF_iFifos_zFcnt_push_full
252	—"—	dut_iPbu_iRequeue_iReFifo_23_iF_iFifos_zFcnt_pop_empty
253	—"—	dut_iPbu_iRequeue_iReFifo_23_iF_iFifos_zFcnt_push_full
254	—"—	dut_iPbu_iRequeue_iReFifo_22_iF_iFifos_zFcnt_pop_empty
255	—"—	dut_iPbu_iRequeue_iReFifo_22_iF_iFifos_zFcnt_push_full
256	—"—	dut_iPbu_iRequeue_iReFifo_21_iF_iFifos_zFcnt_pop_empty
257	—"—	dut_iPbu_iRequeue_iReFifo_21_iF_iFifos_zFcnt_push_full

id	instance	signal
258	—"—	dut_iPbu_iRequeue_iReFifo_20_iF_iFifos_zFcnt_pop_empty
259	—"—	dut_iPbu_iRequeue_iReFifo_20_iF_iFifos_zFcnt_push_full
260	—"—	dut_iPbu_iRequeue_iReFifo_19_iF_iFifos_zFcnt_pop_empty
261	—"—	dut_iPbu_iRequeue_iReFifo_19_iF_iFifos_zFcnt_push_full
262	—"—	dut_iPbu_iRequeue_iReFifo_18_iF_iFifos_zFcnt_pop_empty
263	—"—	dut_iPbu_iRequeue_iReFifo_18_iF_iFifos_zFcnt_push_full
264	—"—	dut_iPbu_iRequeue_iReFifo_17_iF_iFifos_zFcnt_pop_empty
265	—"—	dut_iPbu_iRequeue_iReFifo_17_iF_iFifos_zFcnt_push_full
266	—"—	dut_iPbu_iRequeue_iReFifo_16_iF_iFifos_zFcnt_pop_empty
267	—"—	dut_iPbu_iRequeue_iReFifo_16_iF_iFifos_zFcnt_push_full
268	—"—	dut_iPbu_iRequeue_iReFifo_15_iF_iFifos_zFcnt_pop_empty
269	—"—	dut_iPbu_iRequeue_iReFifo_15_iF_iFifos_zFcnt_push_full
270	—"—	dut_iPbu_iRequeue_iReFifo_14_iF_iFifos_zFcnt_pop_empty
271	—"—	dut_iPbu_iRequeue_iReFifo_14_iF_iFifos_zFcnt_push_full
272	—"—	dut_iPbu_iRequeue_iReFifo_13_iF_iFifos_zFcnt_pop_empty
273	—"—	dut_iPbu_iRequeue_iReFifo_13_iF_iFifos_zFcnt_push_full
274	—"—	dut_iPbu_iRequeue_iReFifo_12_iF_iFifos_zFcnt_pop_empty
275	—"—	dut_iPbu_iRequeue_iReFifo_12_iF_iFifos_zFcnt_push_full
276	—"—	dut_iPbu_iRequeue_iReFifo_11_iF_iFifos_zFcnt_pop_empty
277	—"—	dut_iPbu_iRequeue_iReFifo_11_iF_iFifos_zFcnt_push_full
278	—"—	dut_iPbu_iRequeue_iReFifo_10_iF_iFifos_zFcnt_pop_empty
279	—"—	dut_iPbu_iRequeue_iReFifo_10_iF_iFifos_zFcnt_push_full
280	—"—	dut_iPbu_iRequeue_iReFifo_9_iF_iFifos_zFcnt_pop_empty
281	—"—	dut_iPbu_iRequeue_iReFifo_9_iF_iFifos_zFcnt_push_full
282	—"—	dut_iPbu_iRequeue_iReFifo_8_iF_iFifos_zFcnt_pop_empty
283	—"—	dut_iPbu_iRequeue_iReFifo_8_iF_iFifos_zFcnt_push_full
284	—"—	dut_iPbu_iRequeue_iReFifo_7_iF_iFifos_zFcnt_pop_empty
285	—"—	dut_iPbu_iRequeue_iReFifo_7_iF_iFifos_zFcnt_push_full
286	—"—	dut_iPbu_iRequeue_iReFifo_6_iF_iFifos_zFcnt_pop_empty
287	—"—	dut_iPbu_iRequeue_iReFifo_6_iF_iFifos_zFcnt_push_full
288	—"—	dut_iPbu_iRequeue_iReFifo_5_iF_iFifos_zFcnt_pop_empty
289	—"—	dut_iPbu_iRequeue_iReFifo_5_iF_iFifos_zFcnt_push_full
290	—"—	dut_iPbu_iRequeue_iReFifo_4_iF_iFifos_zFcnt_pop_empty
291	—"—	dut_iPbu_iRequeue_iReFifo_4_iF_iFifos_zFcnt_push_full
292	—"—	dut_iPbu_iRequeue_iReFifo_3_iF_iFifos_zFcnt_pop_empty
293	—"—	dut_iPbu_iRequeue_iReFifo_3_iF_iFifos_zFcnt_push_full
294	—"—	dut_iPbu_iRequeue_iReFifo_2_iF_iFifos_zFcnt_pop_empty
295	—"—	dut_iPbu_iRequeue_iReFifo_2_iF_iFifos_zFcnt_push_full
296	—"—	dut_iPbu_iRequeue_iReFifo_1_iF_iFifos_zFcnt_pop_empty
297	—"—	dut_iPbu_iRequeue_iReFifo_1_iF_iFifos_zFcnt_push_full
298	—"—	dut_iPbu_iRequeue_iReFifo_0_iF_iFifos_zFcnt_pop_empty
299	—"—	dut_iPbu_iRequeue_iReFifo_0_iF_iFifos_zFcnt_push_full
300	—"—	dut_iPbu_iRefc_refc_mem_debug
301	—"—	dut_iPbu_zPassqesp_zPasslist_0_o
302	—"—	Filter mask for packets dropped by ERM
303	—"—	dut_iPbu_debug_pb_drop
304	—"—	constant-304
305	pa_top.switch.pb0.erm.dut_iEql	constant-305
306	—"—	red_zone
307	—"—	constant-307
308	pa_top.switch.pb0.pfc	constant-308
309	—"—	dut_debug_pause
310	—"—	constant-310
311	pa_top.switch.pb0.qe0	constant-311
312	—"—	dut_assert_dfifo
313	—"—	dut_assert_firstflag
314	—"—	dut_assert_reset_next
315	—"—	dut_drop_cnt
316	—"—	dut_send_cnt
317	—"—	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
318	—"—	dut_iDfifo_iF_iFifos_zFcnt_push_full
319	—"—	dut_ipkt_fifo_52_debug_in
320	—"—	dut_ipkt_fifo_52_debug_out
321	—"—	dut_ipkt_fifo_51_debug_in
322	—"—	dut_ipkt_fifo_51_debug_out
323	—"—	dut_ipkt_fifo_50_debug_in
324	—"—	dut_ipkt_fifo_50_debug_out
325	—"—	dut_ipkt_fifo_49_debug_in
326	—"—	dut_ipkt_fifo_49_debug_out
327	—"—	dut_ipkt_fifo_48_debug_in
328	—"—	dut_ipkt_fifo_48_debug_out
329	—"—	dut_ipkt_fifo_47_debug_in
330	—"—	dut_ipkt_fifo_47_debug_out
331	—"—	dut_ipkt_fifo_46_debug_in
332	—"—	dut_ipkt_fifo_46_debug_out
333	—"—	dut_ipkt_fifo_45_debug_in
334	—"—	dut_ipkt_fifo_45_debug_out
335	—"—	dut_ipkt_fifo_44_debug_in
336	—"—	dut_ipkt_fifo_44_debug_out
337	—"—	dut_ipkt_fifo_43_debug_in

id	instance	signal
338	—"—	dut_ipkt_fifo_43_debug_out
339	—"—	dut_ipkt_fifo_42_debug_in
340	—"—	dut_ipkt_fifo_42_debug_out
341	—"—	dut_ipkt_fifo_41_debug_in
342	—"—	dut_ipkt_fifo_41_debug_out
343	—"—	dut_ipkt_fifo_40_debug_in
344	—"—	dut_ipkt_fifo_40_debug_out
345	—"—	dut_ipkt_fifo_39_debug_in
346	—"—	dut_ipkt_fifo_39_debug_out
347	—"—	dut_ipkt_fifo_38_debug_in
348	—"—	dut_ipkt_fifo_38_debug_out
349	—"—	dut_ipkt_fifo_37_debug_in
350	—"—	dut_ipkt_fifo_37_debug_out
351	—"—	dut_ipkt_fifo_36_debug_in
352	—"—	dut_ipkt_fifo_36_debug_out
353	—"—	dut_ipkt_fifo_35_debug_in
354	—"—	dut_ipkt_fifo_35_debug_out
355	—"—	dut_ipkt_fifo_34_debug_in
356	—"—	dut_ipkt_fifo_34_debug_out
357	—"—	dut_ipkt_fifo_33_debug_in
358	—"—	dut_ipkt_fifo_33_debug_out
359	—"—	dut_ipkt_fifo_32_debug_in
360	—"—	dut_ipkt_fifo_32_debug_out
361	—"—	dut_ipkt_fifo_31_debug_in
362	—"—	dut_ipkt_fifo_31_debug_out
363	—"—	dut_ipkt_fifo_30_debug_in
364	—"—	dut_ipkt_fifo_30_debug_out
365	—"—	dut_ipkt_fifo_29_debug_in
366	—"—	dut_ipkt_fifo_29_debug_out
367	—"—	dut_ipkt_fifo_28_debug_in
368	—"—	dut_ipkt_fifo_28_debug_out
369	—"—	dut_ipkt_fifo_27_debug_in
370	—"—	dut_ipkt_fifo_27_debug_out
371	—"—	dut_ipkt_fifo_26_debug_in
372	—"—	dut_ipkt_fifo_26_debug_out
373	—"—	dut_ipkt_fifo_25_debug_in
374	—"—	dut_ipkt_fifo_25_debug_out
375	—"—	dut_ipkt_fifo_24_debug_in
376	—"—	dut_ipkt_fifo_24_debug_out
377	—"—	dut_ipkt_fifo_23_debug_in
378	—"—	dut_ipkt_fifo_23_debug_out
379	—"—	dut_ipkt_fifo_22_debug_in
380	—"—	dut_ipkt_fifo_22_debug_out
381	—"—	dut_ipkt_fifo_21_debug_in
382	—"—	dut_ipkt_fifo_21_debug_out
383	—"—	dut_ipkt_fifo_20_debug_in
384	—"—	dut_ipkt_fifo_20_debug_out
385	—"—	dut_ipkt_fifo_19_debug_in
386	—"—	dut_ipkt_fifo_19_debug_out
387	—"—	dut_ipkt_fifo_18_debug_in
388	—"—	dut_ipkt_fifo_18_debug_out
389	—"—	dut_ipkt_fifo_17_debug_in
390	—"—	dut_ipkt_fifo_17_debug_out
391	—"—	dut_ipkt_fifo_16_debug_in
392	—"—	dut_ipkt_fifo_16_debug_out
393	—"—	dut_ipkt_fifo_15_debug_in
394	—"—	dut_ipkt_fifo_15_debug_out
395	—"—	dut_ipkt_fifo_14_debug_in
396	—"—	dut_ipkt_fifo_14_debug_out
397	—"—	dut_ipkt_fifo_13_debug_in
398	—"—	dut_ipkt_fifo_13_debug_out
399	—"—	dut_ipkt_fifo_12_debug_in
400	—"—	dut_ipkt_fifo_12_debug_out
401	—"—	dut_ipkt_fifo_11_debug_in
402	—"—	dut_ipkt_fifo_11_debug_out
403	—"—	dut_ipkt_fifo_10_debug_in
404	—"—	dut_ipkt_fifo_10_debug_out
405	—"—	dut_ipkt_fifo_9_debug_in
406	—"—	dut_ipkt_fifo_9_debug_out
407	—"—	dut_ipkt_fifo_8_debug_in
408	—"—	dut_ipkt_fifo_8_debug_out
409	—"—	dut_ipkt_fifo_7_debug_in
410	—"—	dut_ipkt_fifo_7_debug_out
411	—"—	dut_ipkt_fifo_6_debug_in
412	—"—	dut_ipkt_fifo_6_debug_out
413	—"—	dut_ipkt_fifo_5_debug_in
414	—"—	dut_ipkt_fifo_5_debug_out
415	—"—	dut_ipkt_fifo_4_debug_in
416	—"—	dut_ipkt_fifo_4_debug_out
417	—"—	dut_ipkt_fifo_3_debug_in

id	instance	signal
418	—"—	dut_ipkt_fifo_3_debug_out
419	—"—	dut_ipkt_fifo_2_debug_in
420	—"—	dut_ipkt_fifo_2_debug_out
421	—"—	dut_ipkt_fifo_1_debug_in
422	—"—	dut_ipkt_fifo_1_debug_out
423	—"—	dut_ipkt_fifo_0_debug_in
424	—"—	dut_ipkt_fifo_0_debug_out
425	—"—	dut_pfifo_level
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	—"—	dut_pfifo_level
435	—"—	dut_pfifo_level
436	—"—	dut_pfifo_level
437	—"—	dut_pfifo_level
438	—"—	dut_pfifo_level
439	—"—	dut_pfifo_level
440	—"—	dut_pfifo_level
441	—"—	dut_pfifo_level
442	—"—	dut_pfifo_level
443	—"—	dut_pfifo_level
444	—"—	dut_pfifo_level
445	—"—	dut_pfifo_level
446	—"—	dut_pfifo_level
447	—"—	dut_pfifo_level
448	—"—	dut_pfifo_level
449	—"—	dut_pfifo_level
450	—"—	dut_pfifo_level
451	—"—	dut_pfifo_level
452	—"—	dut_pfifo_level
453	—"—	dut_pfifo_level
454	—"—	dut_pfifo_level
455	—"—	dut_pfifo_level
456	—"—	dut_pfifo_level
457	—"—	dut_pfifo_level
458	—"—	dut_pfifo_level
459	—"—	dut_pfifo_level
460	—"—	dut_pfifo_level
461	—"—	dut_pfifo_level
462	—"—	dut_pfifo_level
463	—"—	dut_pfifo_level
464	—"—	dut_pfifo_level
465	—"—	dut_pfifo_level
466	—"—	dut_pfifo_level
467	—"—	dut_pfifo_level
468	—"—	dut_pfifo_level
469	—"—	dut_pfifo_level
470	—"—	dut_pfifo_level
471	—"—	dut_pfifo_level
472	—"—	dut_pfifo_level
473	—"—	dut_pfifo_level
474	—"—	dut_pfifo_level
475	—"—	dut_pfifo_level
476	—"—	dut_pfifo_level
477	—"—	dut_pfifo_level
478	—"—	constant-478
479	pa_top.switch.pb0.wrr	constant-479
480	—"—	dut_debug_below
481	—"—	dut_zPassdebugbvalpipe_zPasslist_7_o
482	—"—	dut_zPassdebugbvalpipe_zPasslist_6_o
483	—"—	dut_zPassdebugbvalpipe_zPasslist_5_o
484	—"—	dut_zPassdebugbvalpipe_zPasslist_4_o
485	—"—	dut_zPassdebugbvalpipe_zPasslist_3_o
486	—"—	dut_zPassdebugbvalpipe_zPasslist_2_o
487	—"—	dut_zPassdebugbvalpipe_zPasslist_1_o
488	—"—	dut_zPassdebugbvalpipe_zPasslist_0_o
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
496	—"—	dut_reg_bval
497	—"—	dut_reg_bval

id	instance	signal
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_bval
510	—"—	dut_reg_bval
511	—"—	dut_reg_bval
512	—"—	dut_reg_bval
513	—"—	dut_reg_bval
514	—"—	dut_reg_bval
515	—"—	dut_reg_bval
516	—"—	dut_reg_bval
517	—"—	dut_reg_bval
518	—"—	dut_reg_bval
519	—"—	dut_reg_bval
520	—"—	dut_reg_bval
521	—"—	dut_reg_bval
522	—"—	dut_reg_bval
523	—"—	dut_reg_bval
524	—"—	dut_reg_bval
525	—"—	dut_reg_bval
526	—"—	dut_reg_bval
527	—"—	dut_reg_bval
528	—"—	dut_reg_bval
529	—"—	dut_reg_bval
530	—"—	dut_reg_bval
531	—"—	dut_reg_bval
532	—"—	dut_reg_bval
533	—"—	dut_reg_bval
534	—"—	dut_reg_bval
535	—"—	dut_reg_bval
536	—"—	dut_reg_bval
537	—"—	dut_reg_bval
538	—"—	dut_reg_bval
539	—"—	dut_reg_bval
540	—"—	dut_reg_bval
541	—"—	dut_reg_bval
542	—"—	dut_reg_bval
543	—"—	dut_reg_bval
544	—"—	dut_reg_bval
545	—"—	dut_reg_bval
546	—"—	dut_reg_bval
547	—"—	dut_reg_bval
548	—"—	dut_reg_bval
549	—"—	dut_reg_bval
550	—"—	dut_reg_bval
551	—"—	dut_reg_bval
552	—"—	dut_reg_bval
553	—"—	dut_reg_bval
554	—"—	dut_reg_bval
555	—"—	dut_reg_bval
556	—"—	dut_reg_bval
557	—"—	dut_reg_bval
558	—"—	dut_reg_bval
559	—"—	dut_reg_bval
560	—"—	dut_reg_bval
561	—"—	dut_reg_bval
562	—"—	dut_reg_bval
563	—"—	dut_reg_bval
564	—"—	dut_reg_bval
565	—"—	dut_reg_bval
566	—"—	dut_reg_bval
567	—"—	dut_reg_bval
568	—"—	dut_reg_bval
569	—"—	dut_reg_bval
570	—"—	dut_reg_bval
571	—"—	dut_reg_bval
572	—"—	dut_reg_bval
573	—"—	dut_reg_bval
574	—"—	dut_reg_bval
575	—"—	dut_reg_bval
576	—"—	dut_reg_bval
577	—"—	dut_reg_bval

id	instance	signal
578	—"—	dut_reg_bval
579	—"—	dut_reg_bval
580	—"—	dut_reg_bval
581	—"—	dut_reg_bval
582	—"—	dut_reg_bval
583	—"—	dut_reg_bval
584	—"—	dut_reg_bval
585	—"—	dut_reg_bval
586	—"—	dut_reg_bval
587	—"—	dut_reg_bval
588	—"—	dut_reg_bval
589	—"—	dut_reg_bval
590	—"—	dut_reg_bval
591	—"—	dut_reg_bval
592	—"—	dut_reg_bval
593	—"—	dut_reg_bval
594	—"—	dut_reg_bval
595	—"—	dut_reg_bval
596	—"—	dut_reg_bval
597	—"—	dut_reg_bval
598	—"—	dut_reg_bval
599	—"—	dut_reg_bval
600	—"—	dut_reg_bval
601	—"—	dut_reg_bval
602	—"—	dut_reg_bval
603	—"—	dut_reg_bval
604	—"—	dut_reg_bval
605	—"—	dut_reg_bval
606	—"—	dut_reg_bval
607	—"—	dut_reg_bval
608	—"—	dut_reg_bval
609	—"—	dut_reg_bval
610	—"—	dut_reg_bval
611	—"—	dut_reg_bval
612	—"—	dut_reg_bval
613	—"—	dut_reg_bval
614	—"—	dut_reg_bval
615	—"—	dut_reg_bval
616	—"—	dut_reg_bval
617	—"—	dut_reg_bval
618	—"—	dut_reg_bval
619	—"—	dut_reg_bval
620	—"—	dut_reg_bval
621	—"—	dut_reg_bval
622	—"—	dut_reg_bval
623	—"—	dut_reg_bval
624	—"—	dut_reg_bval
625	—"—	dut_reg_bval
626	—"—	dut_reg_bval
627	—"—	dut_reg_bval
628	—"—	dut_reg_bval
629	—"—	dut_reg_bval
630	—"—	dut_reg_bval
631	—"—	dut_reg_bval
632	—"—	dut_reg_bval
633	—"—	dut_reg_bval
634	—"—	dut_reg_bval
635	—"—	dut_reg_bval
636	—"—	dut_reg_bval
637	—"—	dut_reg_bval
638	—"—	dut_reg_bval
639	—"—	dut_reg_bval
640	—"—	dut_reg_bval
641	—"—	dut_reg_bval
642	—"—	dut_reg_bval
643	—"—	dut_reg_bval
644	—"—	dut_reg_bval
645	—"—	dut_reg_bval
646	—"—	dut_reg_bval
647	—"—	dut_reg_bval
648	—"—	dut_reg_bval
649	—"—	dut_reg_bval
650	—"—	dut_reg_bval
651	—"—	dut_reg_bval
652	—"—	dut_reg_bval
653	—"—	dut_reg_bval
654	—"—	dut_reg_bval
655	—"—	dut_reg_bval
656	—"—	dut_reg_bval
657	—"—	dut_reg_bval

id	instance	signal
658	—"—	dut_reg_bval
659	—"—	dut_reg_bval
660	—"—	dut_reg_bval
661	—"—	dut_reg_bval
662	—"—	dut_reg_bval
663	—"—	dut_reg_bval
664	—"—	dut_reg_bval
665	—"—	dut_reg_bval
666	—"—	dut_reg_bval
667	—"—	dut_reg_bval
668	—"—	dut_reg_bval
669	—"—	dut_reg_bval
670	—"—	dut_reg_bval
671	—"—	dut_reg_bval
672	—"—	dut_reg_bval
673	—"—	dut_reg_bval
674	—"—	dut_reg_bval
675	—"—	dut_reg_bval
676	—"—	dut_reg_bval
677	—"—	dut_reg_bval
678	—"—	dut_reg_bval
679	—"—	dut_reg_bval
680	—"—	dut_reg_bval
681	—"—	dut_reg_bval
682	—"—	dut_reg_bval
683	—"—	dut_reg_bval
684	—"—	dut_reg_bval
685	—"—	dut_reg_bval
686	—"—	dut_reg_bval
687	—"—	dut_reg_bval
688	—"—	dut_reg_bval
689	—"—	dut_reg_bval
690	—"—	dut_reg_bval
691	—"—	dut_reg_bval
692	—"—	dut_reg_bval
693	—"—	dut_reg_bval
694	—"—	dut_reg_bval
695	—"—	dut_reg_bval
696	—"—	dut_reg_bval
697	—"—	dut_reg_bval
698	—"—	dut_reg_bval
699	—"—	dut_reg_bval
700	—"—	dut_reg_bval
701	—"—	dut_reg_bval
702	—"—	dut_reg_bval
703	—"—	dut_reg_bval
704	—"—	dut_reg_bval
705	—"—	dut_reg_bval
706	—"—	dut_reg_bval
707	—"—	dut_reg_bval
708	—"—	dut_reg_bval
709	—"—	dut_reg_bval
710	—"—	dut_reg_bval
711	—"—	dut_reg_bval
712	—"—	dut_reg_bval
713	—"—	dut_reg_bval
714	—"—	dut_reg_bval
715	—"—	dut_reg_bval
716	—"—	dut_reg_bval
717	—"—	dut_reg_bval
718	—"—	dut_reg_bval
719	—"—	dut_reg_bval
720	—"—	dut_reg_bval
721	—"—	dut_reg_bval
722	—"—	dut_reg_bval
723	—"—	dut_reg_bval
724	—"—	dut_reg_bval
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
736	—"—	dut_reg_bval
737	—"—	dut_reg_bval

id	instance	signal
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_bval
790	—"—	dut_reg_bval
791	—"—	dut_reg_bval
792	—"—	dut_reg_bval
793	—"—	dut_reg_bval
794	—"—	dut_reg_bval
795	—"—	dut_reg_bval
796	—"—	dut_reg_bval
797	—"—	dut_reg_bval
798	—"—	dut_reg_bval
799	—"—	dut_reg_bval
800	—"—	dut_reg_bval
801	—"—	dut_reg_bval
802	—"—	dut_reg_bval
803	—"—	dut_reg_bval
804	—"—	dut_reg_bval
805	—"—	dut_reg_bval
806	—"—	dut_reg_bval
807	—"—	dut_reg_bval
808	—"—	dut_reg_bval
809	—"—	dut_reg_bval
810	—"—	dut_reg_bval
811	—"—	dut_reg_bval
812	—"—	dut_reg_bval
813	—"—	dut_reg_bval
814	—"—	dut_reg_bval
815	—"—	dut_reg_bval
816	—"—	dut_reg_bval
817	—"—	dut_reg_bval

id	instance	signal
818	—"—	dut_reg_bval
819	—"—	dut_reg_bval
820	—"—	dut_reg_bval
821	—"—	dut_reg_bval
822	—"—	dut_reg_bval
823	—"—	dut_reg_bval
824	—"—	dut_reg_bval
825	—"—	dut_reg_bval
826	—"—	dut_reg_bval
827	—"—	dut_reg_bval
828	—"—	dut_reg_bval
829	—"—	dut_reg_bval
830	—"—	dut_reg_bval
831	—"—	dut_reg_bval
832	—"—	dut_reg_bval
833	—"—	dut_reg_bval
834	—"—	dut_reg_bval
835	—"—	dut_reg_bval
836	—"—	dut_reg_bval
837	—"—	dut_reg_bval
838	—"—	dut_reg_bval
839	—"—	dut_reg_bval
840	—"—	dut_reg_bval
841	—"—	dut_reg_bval
842	—"—	dut_reg_bval
843	—"—	dut_reg_bval
844	—"—	dut_reg_bval
845	—"—	dut_reg_bval
846	—"—	dut_reg_bval
847	—"—	dut_reg_bval
848	—"—	dut_reg_bval
849	—"—	dut_reg_bval
850	—"—	dut_reg_bval
851	—"—	dut_reg_bval
852	—"—	dut_reg_bval
853	—"—	dut_reg_bval
854	—"—	dut_reg_bval
855	—"—	dut_reg_bval
856	—"—	dut_reg_bval
857	—"—	dut_reg_bval
858	—"—	dut_reg_bval
859	—"—	dut_reg_bval
860	—"—	dut_reg_bval
861	—"—	dut_reg_bval
862	—"—	dut_reg_bval
863	—"—	dut_reg_bval
864	—"—	dut_reg_bval
865	—"—	dut_reg_bval
866	—"—	dut_reg_bval
867	—"—	dut_reg_bval
868	—"—	dut_reg_bval
869	—"—	dut_reg_bval
870	—"—	dut_reg_bval
871	—"—	dut_reg_bval
872	—"—	dut_reg_bval
873	—"—	dut_reg_bval
874	—"—	dut_reg_bval
875	—"—	dut_reg_bval
876	—"—	dut_reg_bval
877	—"—	dut_reg_bval
878	—"—	dut_reg_bval
879	—"—	dut_reg_bval
880	—"—	dut_reg_bval
881	—"—	dut_reg_bval
882	—"—	dut_reg_bval
883	—"—	dut_reg_bval
884	—"—	dut_reg_bval
885	—"—	dut_reg_bval
886	—"—	dut_reg_bval
887	—"—	dut_reg_bval
888	—"—	dut_reg_bval
889	—"—	dut_reg_bval
890	—"—	dut_reg_bval
891	—"—	dut_reg_bval
892	—"—	dut_reg_bval
893	—"—	dut_reg_bval
894	—"—	dut_reg_bval
895	—"—	dut_reg_bval
896	—"—	dut_reg_bval
897	—"—	dut_reg_bval

id	instance	signal
898	—"—	dut_reg_bval
899	—"—	dut_reg_bval
900	—"—	dut_reg_bval
901	—"—	dut_reg_bval
902	—"—	dut_reg_bval
903	—"—	dut_reg_bval
904	—"—	dut_reg_bval
905	—"—	dut_reg_bval
906	—"—	dut_reg_bval
907	—"—	dut_reg_bval
908	—"—	dut_reg_bval
909	—"—	dut_reg_bval
910	—"—	dut_reg_bval
911	—"—	dut_reg_bval
912	—"—	dut_reg_bval
913	—"—	dut_reg_rank
914	—"—	dut_reg_rank
915	—"—	dut_reg_rank
916	—"—	dut_reg_rank
917	—"—	dut_reg_rank
918	—"—	dut_reg_rank
919	—"—	dut_reg_rank
920	—"—	dut_reg_rank
921	—"—	dut_reg_rank
922	—"—	dut_reg_rank
923	—"—	dut_reg_rank
924	—"—	dut_reg_rank
925	—"—	dut_reg_rank
926	—"—	dut_reg_rank
927	—"—	dut_reg_rank
928	—"—	dut_reg_rank
929	—"—	dut_reg_rank
930	—"—	dut_reg_rank
931	—"—	dut_reg_rank
932	—"—	dut_reg_rank
933	—"—	dut_reg_rank
934	—"—	dut_reg_rank
935	—"—	dut_reg_rank
936	—"—	dut_reg_rank
937	—"—	dut_reg_rank
938	—"—	dut_reg_rank
939	—"—	dut_reg_rank
940	—"—	dut_reg_rank
941	—"—	dut_reg_rank
942	—"—	dut_reg_rank
943	—"—	dut_reg_rank
944	—"—	dut_reg_rank
945	—"—	dut_reg_rank
946	—"—	dut_reg_rank
947	—"—	dut_reg_rank
948	—"—	dut_reg_rank
949	—"—	dut_reg_rank
950	—"—	dut_reg_rank
951	—"—	dut_reg_rank
952	—"—	dut_reg_rank
953	—"—	dut_reg_rank
954	—"—	dut_reg_rank
955	—"—	dut_reg_rank
956	—"—	dut_reg_rank
957	—"—	dut_reg_rank
958	—"—	dut_reg_rank
959	—"—	dut_reg_rank
960	—"—	dut_reg_rank
961	—"—	dut_reg_rank
962	—"—	dut_reg_rank
963	—"—	dut_reg_rank
964	—"—	dut_reg_rank
965	—"—	dut_reg_rank
966	—"—	constant-966
967	pa_top.switch.pb0.qshp	constant-967
968	—"—	dut_iPrioshaper_reg_stat
969	—"—	dut_iQueueshaper_reg_stat
970	—"—	constant-970
971	pa_top.switch.bm0	constant-971
972	—"—	dut_bm_ifree_debug_free
973	—"—	constant-973
974	pa_top.switch.ps0	constant-974
975	—"—	halt_from_ps
976	—"—	dut_iPs2_zPsAssert_item
977	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_52.iPsassertout {3'valid_bytes, 1'last, 1'first}

id	instance	signal
978	—"—	dut_iPs2_iBridge_51_assert_reset
979	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_51.iPsassertout {3'valid_bytes, 1'last, 1'first}
980	—"—	dut_iPs2_iBridge_50_assert_reset
981	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_50.iPsassertout {3'valid_bytes, 1'last, 1'first}
982	—"—	dut_iPs2_iBridge_49_assert_reset
983	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_49.iPsassertout {3'valid_bytes, 1'last, 1'first}
984	—"—	dut_iPs2_iBridge_48_assert_reset
985	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_48.iPsassertout {3'valid_bytes, 1'last, 1'first}
986	—"—	dut_iPs2_iBridge_47_assert_reset
987	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_47.iPsassertout {1'valid_bytes, 1'last, 1'first}
988	—"—	dut_iPs2_iBridge_46_assert_reset
989	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_46.iPsassertout {1'valid_bytes, 1'last, 1'first}
990	—"—	dut_iPs2_iBridge_45_assert_reset
991	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_45.iPsassertout {1'valid_bytes, 1'last, 1'first}
992	—"—	dut_iPs2_iBridge_44_assert_reset
993	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_44.iPsassertout {1'valid_bytes, 1'last, 1'first}
994	—"—	dut_iPs2_iBridge_43_assert_reset
995	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_43.iPsassertout {1'valid_bytes, 1'last, 1'first}
996	—"—	dut_iPs2_iBridge_42_assert_reset
997	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_42.iPsassertout {1'valid_bytes, 1'last, 1'first}
998	—"—	dut_iPs2_iBridge_41_assert_reset
999	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_41.iPsassertout {1'valid_bytes, 1'last, 1'first}
1000	—"—	dut_iPs2_iBridge_40_assert_reset
1001	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_40.iPsassertout {1'valid_bytes, 1'last, 1'first}
1002	—"—	dut_iPs2_iBridge_39_assert_reset
1003	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_39.iPsassertout {1'valid_bytes, 1'last, 1'first}
1004	—"—	dut_iPs2_iBridge_38_assert_reset
1005	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_38.iPsassertout {1'valid_bytes, 1'last, 1'first}
1006	—"—	dut_iPs2_iBridge_37_assert_reset
1007	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_37.iPsassertout {1'valid_bytes, 1'last, 1'first}
1008	—"—	dut_iPs2_iBridge_36_assert_reset
1009	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_36.iPsassertout {1'valid_bytes, 1'last, 1'first}
1010	—"—	dut_iPs2_iBridge_35_assert_reset
1011	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_35.iPsassertout {1'valid_bytes, 1'last, 1'first}
1012	—"—	dut_iPs2_iBridge_34_assert_reset
1013	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_34.iPsassertout {1'valid_bytes, 1'last, 1'first}
1014	—"—	dut_iPs2_iBridge_33_assert_reset
1015	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_33.iPsassertout {1'valid_bytes, 1'last, 1'first}
1016	—"—	dut_iPs2_iBridge_32_assert_reset
1017	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_32.iPsassertout {1'valid_bytes, 1'last, 1'first}
1018	—"—	dut_iPs2_iBridge_31_assert_reset
1019	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_31.iPsassertout {1'valid_bytes, 1'last, 1'first}
1020	—"—	dut_iPs2_iBridge_30_assert_reset
1021	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_30.iPsassertout {1'valid_bytes, 1'last, 1'first}
1022	—"—	dut_iPs2_iBridge_29_assert_reset
1023	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_29.iPsassertout {1'valid_bytes, 1'last, 1'first}
1024	—"—	dut_iPs2_iBridge_28_assert_reset
1025	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_28.iPsassertout {1'valid_bytes, 1'last, 1'first}
1026	—"—	dut_iPs2_iBridge_27_assert_reset
1027	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_27.iPsassertout {1'valid_bytes, 1'last, 1'first}
1028	—"—	dut_iPs2_iBridge_26_assert_reset
1029	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_26.iPsassertout {1'valid_bytes, 1'last, 1'first}
1030	—"—	dut_iPs2_iBridge_25_assert_reset
1031	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_25.iPsassertout {1'valid_bytes, 1'last, 1'first}
1032	—"—	dut_iPs2_iBridge_24_assert_reset
1033	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_24.iPsassertout {1'valid_bytes, 1'last, 1'first}
1034	—"—	dut_iPs2_iBridge_23_assert_reset
1035	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_23.iPsassertout {1'valid_bytes, 1'last, 1'first}
1036	—"—	dut_iPs2_iBridge_22_assert_reset
1037	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_22.iPsassertout {1'valid_bytes, 1'last, 1'first}
1038	—"—	dut_iPs2_iBridge_21_assert_reset
1039	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_21.iPsassertout {1'valid_bytes, 1'last, 1'first}
1040	—"—	dut_iPs2_iBridge_20_assert_reset
1041	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_20.iPsassertout {1'valid_bytes, 1'last, 1'first}
1042	—"—	dut_iPs2_iBridge_19_assert_reset
1043	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_19.iPsassertout {1'valid_bytes, 1'last, 1'first}
1044	—"—	dut_iPs2_iBridge_18_assert_reset
1045	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_18.iPsassertout {1'valid_bytes, 1'last, 1'first}
1046	—"—	dut_iPs2_iBridge_17_assert_reset
1047	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_17.iPsassertout {1'valid_bytes, 1'last, 1'first}
1048	—"—	dut_iPs2_iBridge_16_assert_reset
1049	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_16.iPsassertout {1'valid_bytes, 1'last, 1'first}
1050	—"—	dut_iPs2_iBridge_15_assert_reset
1051	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_15.iPsassertout {1'valid_bytes, 1'last, 1'first}
1052	—"—	dut_iPs2_iBridge_14_assert_reset
1053	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_14.iPsassertout {1'valid_bytes, 1'last, 1'first}
1054	—"—	dut_iPs2_iBridge_13_assert_reset
1055	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_13.iPsassertout {1'valid_bytes, 1'last, 1'first}
1056	—"—	dut_iPs2_iBridge_12_assert_reset
1057	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_12.iPsassertout {1'valid_bytes, 1'last, 1'first}

id	instance	signal
1058	—"—	dut_iPs2_iBridge_11_assert_reset
1059	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_11.iPsassertout {1'valid_bytes, 1'last, 1'first}
1060	—"—	dut_iPs2_iBridge_10_assert_reset
1061	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_10.iPsassertout {1'valid_bytes, 1'last, 1'first}
1062	—"—	dut_iPs2_iBridge_9_assert_reset
1063	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_9.iPsassertout {1'valid_bytes, 1'last, 1'first}
1064	—"—	dut_iPs2_iBridge_8_assert_reset
1065	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_8.iPsassertout {1'valid_bytes, 1'last, 1'first}
1066	—"—	dut_iPs2_iBridge_7_assert_reset
1067	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_7.iPsassertout {1'valid_bytes, 1'last, 1'first}
1068	—"—	dut_iPs2_iBridge_6_assert_reset
1069	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_6.iPsassertout {1'valid_bytes, 1'last, 1'first}
1070	—"—	dut_iPs2_iBridge_5_assert_reset
1071	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_5.iPsassertout {1'valid_bytes, 1'last, 1'first}
1072	—"—	dut_iPs2_iBridge_4_assert_reset
1073	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_4.iPsassertout {1'valid_bytes, 1'last, 1'first}
1074	—"—	dut_iPs2_iBridge_3_assert_reset
1075	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_3.iPsassertout {1'valid_bytes, 1'last, 1'first}
1076	—"—	dut_iPs2_iBridge_2_assert_reset
1077	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_2.iPsassertout {1'valid_bytes, 1'last, 1'first}
1078	—"—	dut_iPs2_iBridge_1_assert_reset
1079	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_1.iPsassertout {1'valid_bytes, 1'last, 1'first}
1080	—"—	dut_iPs2_iBridge_0_assert_reset
1081	—"—	pa.top.switch.ps0.ps_wrap_bridge.mem_bridge_0.iPsassertout {1'valid_bytes, 1'last, 1'first}
1082	—"—	dut_iPs2_iSplitter_0_assert_noend
1083	—"—	dut_iPs2_iSplitter_0_assert_ptr
1084	—"—	dut_iPs2_iSplitter_0_used_mem
1085	—"—	dut_iPs2_iSplitter_0_used_mem
1086	—"—	dut_iPs2_iSplitter_0_used_mem
1087	—"—	dut_iPs2_iSplitter_0_used_mem
1088	—"—	dut_iPs2_iSplitter_0_used_mem
1089	—"—	dut_iPs2_iSplitter_0_used_mem
1090	—"—	dut_iPs2_iSplitter_0_used_mem
1091	—"—	dut_iPs2_iSplitter_0_used_mem
1092	—"—	dut_iPs2_iSplitter_0_used_mem
1093	—"—	dut_iPs2_iSplitter_0_used_mem
1094	—"—	dut_iPs2_iSplitter_0_used_mem
1095	—"—	dut_iPs2_iSplitter_0_used_mem
1096	—"—	dut_iPs2_iSplitter_0_used_mem
1097	—"—	dut_iPs2_iSplitter_0_used_mem
1098	—"—	dut_iPs2_iSplitter_0_used_mem
1099	—"—	dut_iPs2_iSplitter_0_used_mem
1100	—"—	dut_iPs2_iSplitter_0_used_mem
1101	—"—	dut_iPs2_iSplitter_0_used_mem
1102	—"—	dut_iPs2_iSplitter_0_used_mem
1103	—"—	dut_iPs2_iSplitter_0_used_mem
1104	—"—	dut_iPs2_iSplitter_0_used_mem
1105	—"—	dut_iPs2_iSplitter_0_used_mem
1106	—"—	dut_iPs2_iSplitter_0_used_mem
1107	—"—	dut_iPs2_iSplitter_0_used_mem
1108	—"—	dut_iPs2_iSplitter_0_used_mem
1109	—"—	dut_iPs2_iSplitter_0_used_mem
1110	—"—	dut_iPs2_iSplitter_0_used_mem
1111	—"—	dut_iPs2_iSplitter_0_used_mem
1112	—"—	dut_iPs2_iSplitter_0_used_mem
1113	—"—	dut_iPs2_iSplitter_0_used_mem
1114	—"—	dut_iPs2_iSplitter_0_used_mem
1115	—"—	dut_iPs2_iSplitter_0_used_mem
1116	—"—	dut_iPs2_iSplitter_0_used_mem
1117	—"—	dut_iPs2_iSplitter_0_used_mem
1118	—"—	dut_iPs2_iSplitter_0_used_mem
1119	—"—	dut_iPs2_iSplitter_0_used_mem
1120	—"—	dut_iPs2_iSplitter_0_used_mem
1121	—"—	dut_iPs2_iSplitter_0_used_mem
1122	—"—	dut_iPs2_iSplitter_0_used_mem
1123	—"—	dut_iPs2_iSplitter_0_used_mem
1124	—"—	dut_iPs2_iSplitter_0_used_mem
1125	—"—	dut_iPs2_iSplitter_0_used_mem
1126	—"—	dut_iPs2_iSplitter_0_used_mem
1127	—"—	dut_iPs2_iSplitter_0_used_mem
1128	—"—	dut_iPs2_iSplitter_0_used_mem
1129	—"—	dut_iPs2_iSplitter_0_used_mem
1130	—"—	dut_iPs2_iSplitter_0_used_mem
1131	—"—	dut_iPs2_iSplitter_0_used_mem
1132	—"—	dut_iPs2_iSplitter_0_used_mem
1133	—"—	dut_iPs2_iSplitter_0_used_mem
1134	—"—	dut_iPs2_iSplitter_0_used_mem
1135	—"—	dut_iPs2_iSplitter_0_used_mem
1136	—"—	dut_iPs2_iSplitter_0_used_mem
1137	—"—	constant-1137

id	instance	signal
1138	pa_top.switch.epp0	constant-1138
1139	—"—	dut_iEpp_assert_ipkt
1140	—"—	dut_iEpp_assert_opkt
1141	—"—	epp_ipkt_bus {16'data, 8'valid_bytes, 6'id, 1'last, 1'first}
1142	—"—	epp_opkt_bus {16'data, 8'valid_bytes, 6'id, 1'last, 1'first}
1143	—"—	dut_iEpp_iDropper_da_0
1144	—"—	dut_iEpp_iDropper_da_1
1145	—"—	dut_iEpp_iDropper_dbg_drop
1146	—"—	dut_iEpp_iDropper_dbg_ifirst
1147	—"—	dut_iEpp_iDropper_dbg_ilast
1148	—"—	dut_iEpp_iDropper_sa_0
1149	—"—	dut_iEpp_iDropper_sa_1
1150	—"—	pa.top.switch.epp0.iPacketassertpm {8'valid_bytes, 6'port, 1'last, 1'first}
1151	—"—	pa.top.switch.epp0.iPacketassertin {8'valid_bytes, 6'port, 1'last, 1'first}
1152	—"—	constant-1152
1153	pa_top.switch.epp0.pm	constant-1153
1154	—"—	pm_fifo_overflow
1155	—"—	dut_dbg_fifo_full
1156	—"—	halt_from_pm
1157	—"—	dut_iFifoa_debug_in
1158	—"—	dut_iFifoa_debug_out
1159	—"—	constant-1159
1160	pa_top.switch.ingress_common	constant-1160
1161	—"—	dut_iLearnage_iHitUpdate_iFifo_0_iF_iFifos_zFcnt_pop_empty
1162	—"—	dut_iLearnage_iHitUpdate_iFifo_0_iF_iFifos_zFcnt_push_full
1163	—"—	dut_iMbsc_iFloodMc_reg_stat
1164	—"—	dut_iMbsc_iFloodUc_reg_stat
1165	—"—	dut_iMbsc_iMc_reg_stat
1166	—"—	dut_iMbsc_iBc_reg_stat
1167	—"—	constant-1167
1168	pa_top.switch.interface_common	constant-1168
1169	—"—	dut_zFaii_iMf_zMf_1_item
1170	—"—	dut_zFaip_iMf_zMf_1_item
1171	—"—	dut_zFaie_iMf_zMf_1_item
1172	—"—	dut_zFaiq_iMf_zMf_1_item
1173	—"—	dut_zFais_iMf_zMf_1_item
1174	—"—	constant-1174

Table 29.9: Debug Selection Map

29.6 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 29.10: The Debug Write interface

Chapter 30

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 29.6 on page 168, but for a detailed description of the APB interface see the AMBA APB Protocol Specification Version 2.0, available at developer.arm.com

30.1 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.

30.2 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.

30.3 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 (23) 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.

30.4 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 23 of the address high during a normal register access. An access with bit 23 of the address low we call a `DEFAULT` access, while an access with bit 23 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 23 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 31

Implementation

31.1 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.

The switch hierarchy is divided into six major blocks that we call floorplan blocks. These are: SP, IPP, BM, PB, EPP, and PS. There is also two smaller blocks: ingress_common, interface_common. In some configurations these are very small, but in some the ingress_common can be quite substantial.

Besides the configuration bus, which spreads it's tentacles to every corner of the core, the dataflow through the floorplan blocks is basically that of the path of a packet. The flow from ingress to egress is SP, IPP, BM/PB, EPP, and PS. The PB/BM are lumped together in the list because the packet data goes through the BM, and the control data through the PB. The ingress_common contains auxillary functions for the ingress packet processing and thus mainly talks to the IPP. The other small block, interface_common, is mostly comprised of shim logic for the external interfaces.

31.1.1 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 $\leftrightarrow$ IPP	0
IPP $\leftrightarrow$ PB/BM	0
PB $\leftrightarrow$ BM	0
BM $\leftrightarrow$ EPP	0
EPP $\leftrightarrow$ PS	0

Table 31.1: 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 31.2: The settings for input and output flops for the floorplan blocks

The pipeline settings used when generating this core are shown in Table 31.1, and the input/output flops are listed in Table 31.2[1].

31.1.2 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.

31.2 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*.

31.2.1 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 ippp0_raw_opt.ramstat.

In addition to the memory instances, the pipeline flipflops belonging to each pipeline stage is documented in ippp0_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.

31.3 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 instanciate appropriate memory macros. The macros needed are listed in Table 31.3. 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	4096	1	None	0	0
dp	3	4096	0	None	0	0
dp	256	212	1	None	0	0
dp	165	256	1	None	0	0
dp	10	512	1	None	0	0
dp	5	2048	1	None	0	0
dp	321	512	1	None	0	0
dp	321	64	1	None	0	0
dp	98	32	1	None	0	0

[1] It 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.

dp	421	256	1	None	0	0
dp	421	32	1	None	0	0
dp	321	128	1	None	0	0
dp	321	16	1	None	0	0
dp	89	53	1	None	0	0
dp	108	4096	1	None	0	0
dp	106	64	1	None	0	0
dp	1	4096	1	None	0	0
dp	1	4096	0	None	0	0
dp	60	4096	1	None	0	0
dp	60	4096	0	None	0	0
dp	15	32832	1	None	0	0
dp	15	32832	0	None	0	0
dp	53	1024	1	None	0	0
dp	20	128	1	None	0	0
dp	91	128	1	None	0	0
dp	32	128	1	None	0	0
dp	118	128	1	None	0	0
dp	10	212	0	None	0	0
dp	1583	214	0	None	0	0
dp	1280	53	1	None	0	0
dp	6	13466	1	None	0	0
dp	41	53	0	None	0	0
dp	8	13466	0	None	0	0
dp	22	13466	0	None	0	0
dp	6	13466	0	None	0	0
dp	16	13466	0	None	0	0
dp	15	13466	0	None	0	0
dp	77	13466	1	None	0	0
dp	1280	13466	0	None	0	0
dp	14	13466	1	None	0	0
dp	39	53	1	None	0	0
dp	52	512	1	None	0	0
dp	38	128	1	None	0	0
dp	48	512	1	None	0	0
dp	48	64	1	None	0	0
dp	10	318	0	None	0	0
dp	9	318	1	None	0	0
dp	1361	320	0	None	0	0
dp	9	320	1	None	0	0
dp	256	318	0	None	0	0
dc	13	16	0	None	0	0
dc	8	16	0	None	0	0
dc	32	16	0	None	0	0

Table 31.3: 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 generated[2].

31.4 Dual ported memories

All memories are dual ported. Some dual-ported memories have different clocks for the two ports, these are all instanciated 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 approch 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.

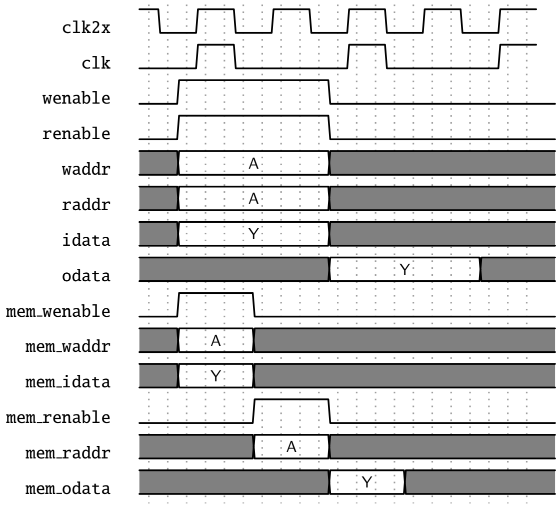

Figure 31.1: Timing diagram for a single ported memory used in the dual ported memory wrapper. In this case a concurrent read and write to the same address of a memory wrapper set for one cycle latency and with the write through attribute set.

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.

31.5 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 31.3.

31.6 Lint set up

For spyglass linting the following settings are assumed:

[2]Although, any instantiated memory wrapper can of course be left as is, and thus be implemented as an array of flops in synthesis.

- set_parameter ignore_local_variables yes
- set_parameter handle_zero_padding “W362”

31.6.1 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 -regexp -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 32

Registers and Tables

Contents

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 30, and the pins for the configuration interfaces are described in Section 29.3.

32.1 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 32 bits and a entry in a table is 33 bits wide, it will then use two addresses per entry. Second example, the bus is still 32 bits, but the entry is 181 bits wide, the entry will then use a address space of 8 addresses per table entry (181 bits fits within 6 bus words but is rounded up to nearest power of two). This is shown in figure 32.1. The total address space used by this core is 280648 addresses.

32.2 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 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*.

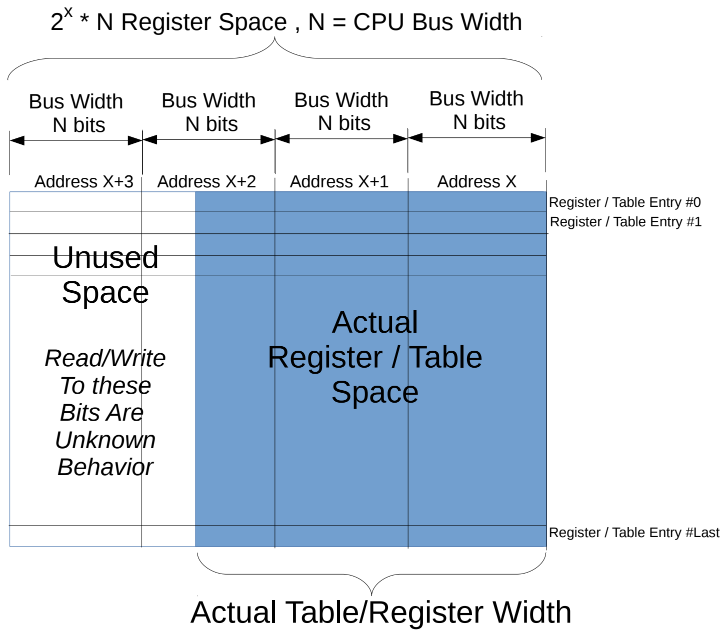

Figure 32.1: Address space usage by tables

32.3 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 32 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 Core Tick Configuration Core Tick Select CPU Port Scratch
rx_length_ref	MAC RX Maximum Packet Length[0..52]
rx_length_drop	MAC RX Broken Packets[0..52] MAC RX Short Packet Drop[0..52] MAC RX Long Packet Drop[0..52]
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_unknown_unicast_storm_control_rate_settings	L2 Unknown Unicast Storm Control Rate Configuration
l2_unknown_unicast_storm_control_bucket_settings	L2 Unknown Unicast Storm Control Bucket Capacity Configuration L2 Unknown Unicast Storm Control Bucket Threshold Configuration
l2_unknown_unicast_storm_control_misc	L2 Unknown Unicast Storm Control Enable
l2_unknown_multicast_storm_control_rate_settings	L2 Unknown Multicast Storm Control Rate Configuration

Bank Name	Connected Registers or Tables
l2_unknown_multicast_storm_control_bucket_settings	L2 Unknown Multicast Storm Control Bucket Capacity Configuration L2 Unknown Multicast Storm Control Bucket Threshold Configuration
l2_unknown_multicast_storm_control_misc	L2 Unknown Multicast Storm Control Enable
le_ae_status	Learning Conflict Learning Overflow
le_ae_control	Learning And Aging Enable Hardware Learning Configuration[0..52] Time to Age
age_cam_register_bank	L2 Aging Collision Table[0..63]
mac_cnt_register_bank	Hardware Learning Counter[0..52]
L2 Aging Table	L2 Aging Table
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 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 Expired TTL Drop IP Checksum Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Attack Prevention 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 GRE Decoder Drop L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop L2 Destination Table SA Lookup Drop Source Port Default ACL Action Drop
count_opkt_pa top switch ipp0 conf	IPP Packet Head Counter IPP Packet Tail Counter
L2 Reserved Multicast Address Action	L2 Reserved Multicast Address Action
Ingress Admission Control Initial Pointer	Ingress Admission Control Initial Pointer

Bank Name	Connected Registers or Tables
Ingress Configurable ACL 0 Pre Lookup	Ingress Configurable ACL 0 Pre Lookup
Ingress Configurable ACL 0 Large Table	Ingress Configurable ACL 0 Large Table
Ingress Configurable ACL 0 Small Table	Ingress Configurable ACL 0 Small Table
Ingress Configurable ACL 0 TCAM Answer	Ingress Configurable ACL 0 TCAM Answer
Ingress Configurable ACL 1 Pre Lookup	Ingress Configurable ACL 1 Pre Lookup
Ingress Configurable ACL 1 Large Table	Ingress Configurable ACL 1 Large Table
Ingress Configurable ACL 1 Small Table	Ingress Configurable ACL 1 Small Table
Ingress Configurable ACL 1 TCAM Answer	Ingress Configurable ACL 1 TCAM Answer
Ingress Configurable ACL 2 Pre Lookup	Ingress Configurable ACL 2 Pre Lookup
Ingress Configurable ACL 2 Large Table	Ingress Configurable ACL 2 Large Table
Ingress Configurable ACL 2 Small Table	Ingress Configurable ACL 2 Small Table
Ingress Configurable ACL 2 TCAM Answer	Ingress Configurable ACL 2 TCAM Answer
Ingress Configurable ACL 3 Pre Lookup	Ingress Configurable ACL 3 Pre Lookup
Ingress Configurable ACL 3 Large Table	Ingress Configurable ACL 3 Large Table
Ingress Configurable ACL 3 Small Table	Ingress Configurable ACL 3 Small Table
Ingress Configurable ACL 3 TCAM Answer	Ingress Configurable ACL 3 TCAM Answer
Source Port Default ACL Action	Source Port Default ACL Action
Ingress VID MAC Range Assignment Answer	Ingress VID MAC Range Assignment Answer
Ingress VID Outer VID Range Assignment Answer	Ingress VID Outer VID Range Assignment Answer
Ingress VID Inner VID Range Assignment Answer	Ingress VID Inner VID Range Assignment Answer
VLAN Table	VLAN Table
Ingress Multiple Spanning Tree State	Ingress Multiple Spanning Tree State
IPv4 TOS Field To Egress Queue Mapping Table	IPv4 TOS Field To Egress Queue Mapping Table
IPv6 Class of Service Field To Egress Queue Mapping Table	IPv6 Class of Service Field To Egress Queue Mapping Table
VLAN PCP And DEI To Color Mapping Table	VLAN PCP And DEI To Color Mapping Table
IPv4 TOS Field To Packet Color Mapping Table	IPv4 TOS Field To Packet Color Mapping Table
IPv6 Class of Service Field To Packet Color Mapping Table	IPv6 Class of Service Field To Packet Color Mapping Table
MPLS EXP Field To Packet Color Mapping Table	MPLS EXP Field To Packet Color Mapping Table
L2 Aging Status Shadow Table	L2 Aging Status Shadow Table
L2 DA Hash Lookup Table	L2 DA Hash Lookup Table
L2 Destination Table	L2 Destination Table
L2 Multicast Table	L2 Multicast Table
Egress Multiple Spanning Tree State	Egress Multiple Spanning Tree State
L2 Action Table	L2 Action Table
L2 Action Table Source Port	L2 Action Table Source Port
ipp_register_bank_ss0	Link Aggregation Ctrl ICMP Length Check Ingress Configurable ACL 0 Selection Ingress Configurable ACL 1 Selection Ingress Configurable ACL 2 Selection Ingress Configurable ACL 3 Selection

Bank Name	Connected Registers or Tables
	Expired TTL to CPU 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 Forward From CPU L2 Multicast Handling Debug srcPort Enable Enqueue To Ports And Queues Flooding Action Send to Port Allow Special Frame Check For L2 Action Table Hairpin Enable L2 Aging Collision Shadow Table MPLS EXP Field To Egress Queue Mapping Table VLAN PCP To Queue Mapping Table TCP/UDP Flag Rules Ingress VID Ethernet Type Range Assignment Answer Ingress Configurable ACL 3 Rules Setup Ingress Configurable ACL 2 Rules Setup Ingress Configurable ACL 1 Rules Setup Ingress Configurable ACL 0 Rules Setup Ingress Port Packet Type Filter SMON Set Search Link Aggregation Membership Source Port Table Ingress Egress Port Packet Type Filter Ingress VID Ethernet Type Range Search Data Ingress VID Inner VID Range Search Data Ingress VID Outer VID Range Search Data Ingress Ethernet Type for VLAN tag ARP Packet Decoder Options RARP Packet Decoder Options IEEE 1588 L2 Packet Decoder Options IEEE 802.1X and EAPOL Packet Decoder Options SCTP Packet Decoder Options AH Header Packet Decoder Options ESP Header Packet Decoder Options DNS Packet Decoder Options Ingress Ports With Timestamp L2 Reserved Multicast Address Base Mask MAC Table Lookup Port Move Options L2 Action Table Egress Port State Ingress MMP Drop Mask Debug dstPortmask Link Aggregation To Physical Ports Members Link Aggregate Weight L2 Lookup Collision Table Masks L2 Lookup Collision Table Send to CPU LLDP Configuration GRE Packet Decoder Options LACP Packet Decoder Options BOOTP and DHCP Packet Decoder Options CAPWAP Packet Decoder Options

Bank Name	Connected Registers or Tables
	Egress Spanning Tree State Ingress VID MAC Range Search Data Reserved Source MAC Address Range Reserved Destination MAC Address Range IEEE 1588 L4 Packet Decoder Options Ingress Configurable ACL 1 Search Mask Ingress Configurable ACL 1 TCAM Ingress Configurable ACL 0 Search Mask Ingress Configurable ACL 2 Search Mask Ingress Configurable ACL 3 Search Mask Ingress Configurable ACL 3 TCAM Ingress Configurable ACL 2 TCAM Ingress Configurable ACL 0 TCAM
ipp_register_bank_misc_ss0	Ingress Drop Options
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] SMON Set 4 Packet Counter[0..7] SMON Set 5 Packet Counter[0..7] SMON Set 6 Packet Counter[0..7] SMON Set 7 Packet Counter[0..7] SMON Set 8 Packet Counter[0..7] SMON Set 9 Packet Counter[0..7] SMON Set 10 Packet Counter[0..7] SMON Set 11 Packet Counter[0..7] SMON Set 12 Packet Counter[0..7] SMON Set 13 Packet Counter[0..7] SMON Set 14 Packet Counter[0..7] SMON Set 15 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] SMON Set 4 Byte Counter[0..7] SMON Set 5 Byte Counter[0..7] SMON Set 6 Byte Counter[0..7] SMON Set 7 Byte Counter[0..7] SMON Set 8 Byte Counter[0..7] SMON Set 9 Byte Counter[0..7] SMON Set 10 Byte Counter[0..7] SMON Set 11 Byte Counter[0..7] SMON Set 12 Byte Counter[0..7] SMON Set 13 Byte Counter[0..7] SMON Set 14 Byte Counter[0..7] SMON Set 15 Byte Counter[0..7]
count_ipp0_aclConfStatisticsBlock	Ingress Configurable ACL Match Counter[0..255]
count_ipp0_egressDropStatisticsBlock	Queue Off Drop[0..52] Egress Spanning Tree Drop[0..52] MBSC Drop[0..52] Ingress-Egress Packet Filtering Drop[0..52] L2 Action Table Per Port Drop[0..52]
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

Bank Name	Connected Registers or Tables
Ingress Admission Control Token Bucket Configuration	Ingress Admission Control Token Bucket Configuration
Ingress Admission Control Reset	Ingress Admission Control Reset
Ingress Admission Control Current Status	Ingress Admission Control Current Status
bk_erm_ss0	ERM Yellow Configuration ERM Red Configuration Egress Resource Manager Pointer[0..52] Resource Limiter Set[0..26]
count_erm_ss0	Egress Resource Manager Drop[0..52]
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..52]
count_drop_pa top switch pb0 iRequeue	Re-queue Overflow Drop
pfc_regbank_port_rsv_size_ss0	Port Reserved[0..52]
pfc_regbank_cmn_misc_ss0	Port Used[0..52] FFA Used
pfc_regbank_pause_settings1_ss0	Port Pause Settings[0..52]
pfc_regbank_taildrop_settings0_ss0	Port Tail-Drop Settings[0..52]
pfc_regbank_misc_ss0	Xon FFA Threshold Xoff FFA Threshold Tail-Drop FFA Threshold Port Xon FFA Threshold[0..52] Port Xoff FFA Threshold[0..52] Port Tail-Drop FFA Threshold[0..52]
qe_register_bank_ss0_sp0	Egress Port Depth[0..52] Egress Queue Depth[0..423]
pb_r_register_bank_ss0	Minimum Buffer Free
disable_queue_output_register_bank_ss0	Output Disable[0..52]
dwrr_bucket_capacity_settings_ss0	DWRR Bucket Capacity Configuration[0..52]
dwrr_bucket_misc_settings_ss0	DWRR Bucket Misc Configuration[0..52]
dwrr_weight_settings_ss0	DWRR Weight Configuration[0..423]
queue_shaper_rate_settings	Queue Shaper Rate Configuration
queue_shaper_bucket_settings	Queue Shaper Bucket Capacity Configuration Queue Shaper Bucket Threshold Configuration
queue_shaper_misc	Queue Shaper Enable
prio_shaper_rate_settings	Prio Shaper Rate Configuration
prio_shaper_bucket_settings	Prio Shaper Bucket Capacity Configuration Prio Shaper Bucket Threshold Configuration
prio_shaper_misc	Prio Shaper Enable
port_shaper_rate_settings	Port Shaper Rate Configuration
port_shaper_bucket_settings	Port Shaper Bucket Capacity Configuration Port Shaper Bucket Threshold Configuration
port_shaper_misc	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..52]
count_pa top switch epp0 conf	Unknown Egress Drop[0..52] Egress Port Disabled Drop[0..52] Egress Port Filtering Drop[0..52] EPP PM Drop
count_opkt_pa top switch epp0 conf	EPP Packet Head Counter EPP Packet Tail Counter

Bank Name	Connected Registers or Tables
Egress Port Configuration	Egress Port Configuration
Egress MAC Operations	Egress MAC Operations
Color Remap From Egress Port	Color Remap From Egress Port
Color Remap From Ingress Admission Control	Color Remap From Ingress Admission Control
Egress Queue To PCP And CFI/DEI Mapping Table	Egress Queue To PCP And CFI/DEI Mapping Table
Egress VLAN Translation Large Table	Egress VLAN Translation Large Table
Egress VLAN Translation Small Table	Egress VLAN Translation Small Table
Egress VLAN Translation TCAM Answer	Egress VLAN Translation TCAM Answer
epp_register_bank_ss0	Output Mirroring Table Egress Ethernet Type for VLAN tag Egress VLAN Translation Selection Disable CPU tag on CPU Port Egress VLAN Translation Search Mask Egress RSPAN Configuration Egress VLAN Translation TCAM
count_opkt_pa top switch ps0 ps_wrap_bridge	PS Packet Head Counter PS Packet Tail Counter
count_error_pa top switch ps0 ps_wrap_bridge	PS Error Counter

32.4 Registers and Tables in Alphabetical Order

Name	Address Range
AH Decoder Drop	34058
AH Header Packet Decoder Options	267343
ARP Decoder Drop	34051
ARP Packet Decoder Options	267323
Allow Special Frame Check For L2 Action Table	266504 - 266507
Attack Prevention Drop	34050
BOOTP and DHCP Decoder Drop	34061
BOOTP and DHCP Packet Decoder Options	268163
Buffer Free	1
Buffer Overflow Drop	272061
CAPWAP Decoder Drop	34062
CAPWAP Packet Decoder Options	268171
CPU Port	4
Check IPv4 Header Checksum	266390
Color Remap From Egress Port	277685 - 277790
Color Remap From Ingress Admission Control	277791 - 278046
Core Tick Configuration	2
Core Tick Select	3
Core Version	0
DNS Decoder Drop	34060
DNS Packet Decoder Options	267351
DWRR Bucket Capacity Configuration	273023 - 273075
DWRR Bucket Misc Configuration	273076 - 273128
DWRR Weight Configuration	273129 - 273552

Name	Address Range
Debug dstPortmask	267367
Debug srcPort	266397
Disable CPU tag on CPU Port	280422
Drain Port	276338
Drain Port Drop	276340 - 276392
EPP PM Drop	276552
EPP Packet Head Counter	276553
EPP Packet Tail Counter	276554
ERM Red Configuration	271844
ERM Yellow Configuration	271840
ESP Decoder Drop	34059
ESP Header Packet Decoder Options	267347
Egress Ethernet Type for VLAN tag	280420
Egress MAC Operations	276661 - 277684
Egress Multiple Spanning Tree State	265871 - 266126
Egress Port Configuration	276555 - 276660
Egress Port Depth	272492 - 272544
Egress Port Disabled Drop	276446 - 276498
Egress Port Filtering Drop	276499 - 276551
Egress Queue Depth	272545 - 272968
Egress Queue To PCP And CFI/DEI Mapping Table	278047 - 278054
Egress RSPAN Configuration	280425 - 280530
Egress Resource Manager Drop	272006 - 272058
Egress Resource Manager Pointer	271846 - 271951
Egress Spanning Tree Drop	270561 - 270613
Egress Spanning Tree State	268179
Egress VLAN Translation Large Table	278055 - 280102
Egress VLAN Translation Search Mask	280423
Egress VLAN Translation Selection	280421
Egress VLAN Translation Small Table	280103 - 280358
Egress VLAN Translation TCAM	280531 - 280546
Egress VLAN Translation TCAM Answer	280359 - 280366
Empty Mask Drop	34035
Enable Enqueue To Ports And Queues	266398 - 266450
Expired TTL Drop	34046
Expired TTL to CPU	266389
FFA Used	272223
Flooding Action Send to Port	266451 - 266503
Flow Classification And Metering Drop	270773
Force Non VLAN Packet To Specific Color	266393
Force Non VLAN Packet To Specific Queue	266391
Force Unknown L3 Packet To Specific Color	266394
Force Unknown L3 Packet To Specific Egress Queue	266392
Forward From CPU	266395
GRE Decoder Drop	34063
GRE Packet Decoder Options	268147
Hairpin Enable	266508 - 266560
Hardware Learning Configuration	957 - 1009
Hardware Learning Counter	1076 - 1128
ICMP Length Check	266384
IEEE 1588 L2 Packet Decoder Options	267331
IEEE 1588 L4 Packet Decoder Options	268347
IEEE 802.1X and EAPOL Decoder Drop	34055

Name	Address Range
IEEE 802.1X and EAPOL Packet Decoder Options	267335
IP Checksum Drop	34047
IPP Empty Destination Drop	34033
IPP PM Drop	34032
IPP Packet Head Counter	34069
IPP Packet Tail Counter	34070
IPv4 TOS Field To Egress Queue Mapping Table	131639 - 131894
IPv4 TOS Field To Packet Color Mapping Table	132167 - 132422
IPv6 Class of Service Field To Egress Queue Mapping Table	131895 - 132150
IPv6 Class of Service Field To Packet Color Mapping Table	132423 - 132678
Ingress Admission Control Current Status	271670 - 271797
Ingress Admission Control Initial Pointer	36119 - 36630
Ingress Admission Control Mark All Red	270902 - 271029
Ingress Admission Control Mark All Red Enable	270774 - 270901
Ingress Admission Control Reset	271542 - 271669
Ingress Admission Control Token Bucket Configuration	271030 - 271541
Ingress Configurable ACL 0 Large Table	38679 - 71446
Ingress Configurable ACL 0 Pre Lookup	36631 - 38678
Ingress Configurable ACL 0 Rules Setup	266709 - 266724
Ingress Configurable ACL 0 Search Mask	268923
Ingress Configurable ACL 0 Selection	266385
Ingress Configurable ACL 0 Small Table	71447 - 75542
Ingress Configurable ACL 0 TCAM	269483 - 269994
Ingress Configurable ACL 0 TCAM Answer	75543 - 75670
Ingress Configurable ACL 1 Large Table	77719 - 94102
Ingress Configurable ACL 1 Pre Lookup	75671 - 77718
Ingress Configurable ACL 1 Rules Setup	266693 - 266708
Ingress Configurable ACL 1 Search Mask	268379
Ingress Configurable ACL 1 Selection	266386
Ingress Configurable ACL 1 Small Table	94103 - 96150
Ingress Configurable ACL 1 TCAM	268411 - 268922
Ingress Configurable ACL 1 TCAM Answer	96151 - 96214
Ingress Configurable ACL 2 Large Table	98263 - 106454
Ingress Configurable ACL 2 Pre Lookup	96215 - 98262
Ingress Configurable ACL 2 Rules Setup	266677 - 266692
Ingress Configurable ACL 2 Search Mask	268939
Ingress Configurable ACL 2 Selection	266387
Ingress Configurable ACL 2 Small Table	106455 - 107478
Ingress Configurable ACL 2 TCAM	269227 - 269482
Ingress Configurable ACL 2 TCAM Answer	107479 - 107542
Ingress Configurable ACL 3 Large Table	109591 - 113686
Ingress Configurable ACL 3 Pre Lookup	107543 - 109590
Ingress Configurable ACL 3 Rules Setup	266661 - 266676
Ingress Configurable ACL 3 Search Mask	268955
Ingress Configurable ACL 3 Selection	266388
Ingress Configurable ACL 3 Small Table	113687 - 114710
Ingress Configurable ACL 3 TCAM	268971 - 269226
Ingress Configurable ACL 3 TCAM Answer	114711 - 114774
Ingress Configurable ACL Drop	34049
Ingress Configurable ACL Match Counter	270252 - 270507
Ingress Drop Options	269995
Ingress Egress Port Packet Type Filter	267059 - 267270
Ingress Ethernet Type for VLAN tag	267319

Name	Address Range
Ingress MMP Drop Mask	267365
Ingress Multiple Spanning Tree State	131383 - 131638
Ingress Packet Filtering Drop	34040
Ingress Port Packet Type Filter	266725 - 266777
Ingress Ports With Timestamp	267355
Ingress Resource Manager Drop	272062
Ingress Spanning Tree Drop: Blocking	34038
Ingress Spanning Tree Drop: Learning	34037
Ingress Spanning Tree Drop: Listen	34036
Ingress VID Ethernet Type Range Assignment Answer	266657 - 266660
Ingress VID Ethernet Type Range Search Data	267271 - 267286
Ingress VID Inner VID Range Assignment Answer	114995 - 114998
Ingress VID Inner VID Range Search Data	267287 - 267302
Ingress VID MAC Range Assignment Answer	114987 - 114990
Ingress VID MAC Range Search Data	268187 - 268218
Ingress VID Outer VID Range Assignment Answer	114991 - 114994
Ingress VID Outer VID Range Search Data	267303 - 267318
Ingress-Egress Packet Filtering Drop	270667 - 270719
L2 Action Table	266127 - 266254
L2 Action Table Drop	34065
L2 Action Table Egress Port State	267363
L2 Action Table Per Port Drop	270720 - 270772
L2 Action Table Port Move Drop	34066
L2 Action Table Source Port	266255 - 266382
L2 Action Table Special Packet Type Drop	34064
L2 Aging Collision Shadow Table	266561 - 266624
L2 Aging Collision Table	1012 - 1075
L2 Aging Status Shadow Table	132687 - 165454
L2 Aging Table	1129 - 33896
L2 Broadcast Storm Control Bucket Capacity Configuration	357 - 409
L2 Broadcast Storm Control Bucket Threshold Configuration	410 - 462
L2 Broadcast Storm Control Enable	463
L2 Broadcast Storm Control Rate Configuration	304 - 356
L2 DA Hash Lookup Table	165455 - 230990
L2 Destination Table	230991 - 263822
L2 Destination Table SA Lookup Drop	34067
L2 IEEE 1588 Decoder Drop	34053
L2 Lookup Collision Table	268003 - 268130
L2 Lookup Collision Table Masks	267987 - 268002
L2 Lookup Drop	34039
L2 Multicast Handling	266396
L2 Multicast Storm Control Bucket Capacity Configuration	518 - 570
L2 Multicast Storm Control Bucket Threshold Configuration	571 - 623
L2 Multicast Storm Control Enable	624
L2 Multicast Storm Control Rate Configuration	465 - 517
L2 Multicast Table	263823 - 265870
L2 Reserved Multicast Address Action	34071 - 36118
L2 Reserved Multicast Address Base	267357
L2 Reserved Multicast Address Drop	34048
L2 Unknown Multicast Storm Control Bucket Capacity Configuration	840 - 892
L2 Unknown Multicast Storm Control Bucket Threshold Configuration	893 - 945

Name	Address Range
L2 Unknown Multicast Storm Control Enable	946
L2 Unknown Multicast Storm Control Rate Configuration	787 - 839
L2 Unknown Unicast Storm Control Bucket Capacity Configuration	679 - 731
L2 Unknown Unicast Storm Control Bucket Threshold Configuration	732 - 784
L2 Unknown Unicast Storm Control Enable	785
L2 Unknown Unicast Storm Control Rate Configuration	626 - 678
L4 IEEE 1588 Decoder Drop	34054
LACP Decoder Drop	34057
LACP Packet Decoder Options	268155
LLDP Configuration	268139
Learning And Aging Enable	956
Learning Conflict	948
Learning Overflow	952
Link Aggregate Weight	267475 - 267986
Link Aggregation Ctrl	266383
Link Aggregation Membership	266794 - 266846
Link Aggregation To Physical Ports Members	267369 - 267474
MAC RX Broken Packets	101 - 153
MAC RX Long Packet Drop	207 - 259
MAC RX Maximum Packet Length	48 - 100
MAC RX Short Packet Drop	154 - 206
MBSC Drop	270614 - 270666
MPLS EXP Field To Egress Queue Mapping Table	266625 - 266632
MPLS EXP Field To Packet Color Mapping Table	132679 - 132686
Map Queue to Priority	272063 - 272115
Mask MAC Table Lookup	267359
Maximum Allowed VLAN Drop	34045
Minimum Allowed VLAN Drop	34044
Minimum Buffer Free	272969
Output Disable	272970 - 273022
Output Mirroring Table	280367 - 280419
PB Packet Head Counter	276336
PB Packet Tail Counter	276337
PS Error Counter	280594 - 280646
PS Packet Head Counter	280592
PS Packet Tail Counter	280593
Packet Buffer Status	272059
Port Move Options	267361
Port Pause Settings	272224 - 272276
Port Reserved	272117 - 272169
Port Shaper Bucket Capacity Configuration	276182 - 276234
Port Shaper Bucket Threshold Configuration	276235 - 276287
Port Shaper Enable	276288
Port Shaper Rate Configuration	276129 - 276181
Port Tail-Drop FFA Threshold	272439 - 272491
Port Tail-Drop Settings	272277 - 272329
Port Used	272170 - 272222
Port Xoff FFA Threshold	272386 - 272438
Port Xon FFA Threshold	272333 - 272385
Prio Shaper Bucket Capacity Configuration	275265 - 275688
Prio Shaper Bucket Threshold Configuration	275689 - 276112

Name	Address Range
Prio Shaper Enable	276113
Prio Shaper Rate Configuration	274841 - 275264
Queue Off Drop	270508 - 270560
Queue Shaper Bucket Capacity Configuration	273977 - 274400
Queue Shaper Bucket Threshold Configuration	274401 - 274824
Queue Shaper Enable	274825
Queue Shaper Rate Configuration	273553 - 273976
RARP Decoder Drop	34052
RARP Packet Decoder Options	267327
Re-queue Overflow Drop	272116
Reserved Destination MAC Address Range	268283 - 268346
Reserved MAC DA Drop	34041
Reserved MAC SA Drop	34042
Reserved Source MAC Address Range	268219 - 268282
Resource Limiter Set	271952 - 272005
SCTP Decoder Drop	34056
SCTP Packet Decoder Options	267339
SMON Set 0 Byte Counter	270124 - 270131
SMON Set 0 Packet Counter	269996 - 270003
SMON Set 1 Byte Counter	270132 - 270139
SMON Set 1 Packet Counter	270004 - 270011
SMON Set 10 Byte Counter	270204 - 270211
SMON Set 10 Packet Counter	270076 - 270083
SMON Set 11 Byte Counter	270212 - 270219
SMON Set 11 Packet Counter	270084 - 270091
SMON Set 12 Byte Counter	270220 - 270227
SMON Set 12 Packet Counter	270092 - 270099
SMON Set 13 Byte Counter	270228 - 270235
SMON Set 13 Packet Counter	270100 - 270107
SMON Set 14 Byte Counter	270236 - 270243
SMON Set 14 Packet Counter	270108 - 270115
SMON Set 15 Byte Counter	270244 - 270251
SMON Set 15 Packet Counter	270116 - 270123
SMON Set 2 Byte Counter	270140 - 270147
SMON Set 2 Packet Counter	270012 - 270019
SMON Set 3 Byte Counter	270148 - 270155
SMON Set 3 Packet Counter	270020 - 270027
SMON Set 4 Byte Counter	270156 - 270163
SMON Set 4 Packet Counter	270028 - 270035
SMON Set 5 Byte Counter	270164 - 270171
SMON Set 5 Packet Counter	270036 - 270043
SMON Set 6 Byte Counter	270172 - 270179
SMON Set 6 Packet Counter	270044 - 270051
SMON Set 7 Byte Counter	270180 - 270187
SMON Set 7 Packet Counter	270052 - 270059
SMON Set 8 Byte Counter	270188 - 270195
SMON Set 8 Packet Counter	270060 - 270067
SMON Set 9 Byte Counter	270196 - 270203
SMON Set 9 Packet Counter	270068 - 270075
SMON Set Search	266778 - 266793
SP Overflow Drop	33936 - 33988
Scratch	5
Send to CPU	268131

Name	Address Range
Source Port Default ACL Action	114775 - 114986
Source Port Default ACL Action Drop	34068
Source Port Table	266847 - 267058
TCP/UDP Flag Rules	266641 - 266656
Tail-Drop FFA Threshold	272332
Time to Age	1010
Unknown Egress Drop	276393 - 276445
Unknown Ingress Drop	34034
VLAN Member Drop	34043
VLAN PCP And DEI To Color Mapping Table	132151 - 132166
VLAN PCP To Queue Mapping Table	266633 - 266640
VLAN Table	114999 - 131382
Xoff FFA Threshold	272331
Xon FFA Threshold	272330

32.5 Active Queue Manager

32.5.1 ERM Red Configuration

Configurations to mark the buffer memory congestion status as Red (heavily congested).

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	271844

Field Description

Bits	Field Name	Description	Default Value
13:0	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.	0x212
27:14	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.	0xd26
34:28	redMaxCells	Maximum allowed packet length in cells when the buffer memory congestion status is red.	0xb

32.5.2 ERM Yellow Configuration

Configurations to mark the buffer memory congestion status as Yellow (slightly congested).

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 271840

Field Description

Bits	Field Name	Description	Default Value
13:0	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.	0x212
27:14	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.	0xf07
80:28	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.	$2^{53} - 1$
94:81	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.	0x247

32.5.3 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 : 53
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 271846 to 271951

Field Description

Bits	Field Name	Description	Default Value
4:0	q0	Pointer to the **Resource Limiter Set** for egress queue 0.	0x0
9:5	q1	Pointer to the **Resource Limiter Set** for egress queue 1.	0x0
14:10	q2	Pointer to the **Resource Limiter Set** for egress queue 2.	0x0
19:15	q3	Pointer to the **Resource Limiter Set** for egress queue 3.	0x0
24:20	q4	Pointer to the **Resource Limiter Set** for egress queue 4.	0x0

Bits	Field Name	Description	Default Value
29:25	q5	Pointer to the **Resource Limiter Set** for egress queue 5.	0x0
34:30	q6	Pointer to the **Resource Limiter Set** for egress queue 6.	0x0
39:35	q7	Pointer to the **Resource Limiter Set** for egress queue 7.	0x0

32.5.4 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 :	27
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Pointer from the **Egress Resource Manager Pointer**
Address Space :	271952 to 272005

Field Description

Bits	Field Name	Description	Default Value
13:0	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 appling the **yellowLimit**.	0x55
27:14	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.	0x3d
41:28	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.	0x1a
48:42	maxCells	Maximum allowed packet length in cells for this limiter. Packet with cells more than this value will be dropped.	0x7f

32.6 Core Information

32.6.1 Core Version

Adress 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

32.7 Egress Packet Processing

32.7.1 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 :	53
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space :	277685 to 277790

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. 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
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
36:34	color2Dei	New DEI value based on packet color. This is located in the outermost VLAN of the transmitted packet. bit 0 : DEI value for green bit 1 : DEI value for yellow bit 2 : DEI value for red	0x0

32.7.2 Color Remap From Ingress Admission Control

Options from ingress admission control to remap internal packet color to outgoing packet headers.

Number of Entries :	128
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space :	277791 to 278046

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. 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
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. bit 0 : DEI value for green bit 1 : DEI value for yellow bit 2 : DEI value for red	0x0

32.7.3 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 : 280422

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

32.7.4 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
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 276338

Field Description

Bits	Field Name	Description	Default Value
52:0	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

32.7.5 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 : 280420

Field Description

Bits	Field Name	Description	Default Value
15:0	typeValue	Ethernet Type value.	0xffff

32.7.6 Egress MAC Operations

The operation to do on the packets MAC fields.

Number of Entries : 512
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : From ingress ACL lookup
Address Space : 276661 to 277684

Field Description

Bits	Field Name	Description	Default Value
1:0	saOp	Where shall the MAC SA come from. 0 = No Change 1 = Use DA MAC 2 = Use data from this table 3 = Reserved	0x0

Bits	Field Name	Description	Default Value
3:2	daOp	Where shall the MAC DA come from. 0 = No Change 1 = Use SA MAC 2 = Use data from this table 3 = Reserved	0x0
51:4	macData	The data which can be used to update SA or DA MAC.	0x0

32.7.7 Egress Multiple Spanning Tree State

Table of egress Multiple Spanning Tree Protocol Instances. The field **msptPtr** in the **VLAN Table** is used to address the instance/entry in this table. Each entry contains the egress spanning tree states for all ports in this MSTI.

Number of Entries :	64
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	**mstpPtr** from **VLAN Table**
Address Space :	265871 to 266126

Field Description

Bits	Field Name	Description	Default Value
105:0	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

32.7.8 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 :	53
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	276555 to 276660

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
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
5:4	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
7:6	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
9:8	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
11:10	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
23:12	vid	The VID used in egress port VLAN push or swap operation if selected by **vidSel**.	0x0
24	cfiDei	The CFI/DEI used in egress port VLAN push or swap operation if selected by **cfiDeiSel**.	0x0

Bits	Field Name	Description	Default Value
27:25	pcp	The PCP used in egress port VLAN push or swap operation if selected by **pcpSel**.	0x0
28	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
29	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
30	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
31	moreThanOneVlans	When filtering with dropCtaggedVlans or dropStaggedVlans then this field must be set to 1.	0x0
32	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
33	dropSingleTaggedVlans	Drop or Allow packets that has one VLAN tag on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
34	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
35	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
36	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
37	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
38	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

32.7.9 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 : 278047 to 278054

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

32.7.10 Egress RSPAN Configuration

Configuration for RSPAN tags on each egress port. When configured to push or pop a RSPAN tag then all packets will unconditionally be subject to this operation. When pushing an RSPAN tag the content of the tag is specified in this register.

Number of Entries : 53
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 280425 to 280530

Field Description

Bits	Field Name	Description	Default Value
0	pushRspanTag	Push an RSPAN tag to all packets on this port.	0x0
1	popRspanTag	Pop an RSPAN tag from all packets on this port.	0x0
17:2	rspanTagEthType	The EtherType used when pushing an RSPAN tag.	0x0
29:18	rspanTagVid	The VID used when pushing an RSPAN tag.	0x0
30	rspanTagCfiDei	The DEI used when pushing an RSPAN tag.	0x0
33:31	rspanTagPcp	The PCP used when pushing an RSPAN tag.	0x0

32.7.11 Egress VLAN Translation Large Table

The outermost VID and VID Ethernet Type (Service tag or Customer tag types) of the outgoing packet is compared.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries :	1024
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address[8:0] : hash of { dstPort outermostVid outermostVid-Type } address[9:9] : bucket number
Address Space :	278055 to 280102

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
6:1	dstPort	This is a field which is used as search data. The destination port which the packet is going out on	0x0
18:7	outermostVid	This is a field which is used as search data. The outermost VID of the modified packet.	0x0
19	outermostVidType	This is a field which is used as search data. The outermost VID is a S-tag or C-Tag. 0 = Customer tag 1 = Service tag	0x0
31:20	newVid	This is a result field used when this entry is hit. The new VID for the outgoing packet.	0x0
47:32	ethType	This is a result field used when this entry is hit. The new Ethernet Type for the outgoing packet	0x0

32.7.12 Egress VLAN Translation Search Mask

Before the hashing and searching is done in the **Egress VLAN Translation Large Table** and **Egress VLAN Translation 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. Seperate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	280423

Field Description

Bits	Field Name	Description	Default Value
5:0	dstPort_mask_small	Which bits to compare in the field dstPort in **Egress VLAN Translation 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.	0x3f

Bits	Field Name	Description	Default Value
11:6	dstPort_mask_large	Which bits to compare in the field dstPort **Egress VLAN Translation 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.	0x3f
23:12	outermostVid_mask_small	Which bits to compare in the field outermostVid in **Egress VLAN Translation 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.	0xfff
35:24	outermostVid_mask_large	Which bits to compare in the field outermostVid **Egress VLAN Translation 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.	0xfff
36	outermostVidType_mask_small	Which bits to compare in the field outermostVidType in **Egress VLAN Translation 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.	0x1
37	outermostVidType_mask_large	Which bits to compare in the field outermostVidType **Egress VLAN Translation 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.	0x1

32.7.13 Egress VLAN Translation Selection

This register selects which result to use when there are multiple hits.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	280421

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

32.7.14 Egress VLAN Translation Small Table

The outermost VID and VID Ethernet Type (Service tag or Customer tag types) of the outgoing packet is compared.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries :	128
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address[5:0] : hash of { dstPort outermostVid outermostVidType } address[6:6] : bucket number
Address Space :	280103 to 280358

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
6:1	dstPort	This is a field which is used as search data. The destination port which the packet is going out on	0x0
18:7	outermostVid	This is a field which is used as search data. The outermost VID of the modified packet.	0x0
19	outermostVidType	This is a field which is used as search data. The outermost VID is a S-tag or C-Tag. 0 = Customer tag 1 = Service tag	0x0
31:20	newVid	This is a result field used when this entry is hit. The new VID for the outgoing packet.	0x0
47:32	ethType	This is a result field used when this entry is hit. The new Ethernet Type for the outgoing packet	0x0

32.7.15 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 :	8
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	280531 to 280546

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
6:1	dstPort_mask	Mask for dstPort.	0x3f
12:7	dstPort	The destination port which the packet is going out on	0x0
24:13	outermostVid_mask	Mask for outermostVid.	0xfff
36:25	outermostVid	The outermost VID of the modified packet.	0x0
37	outermostVidType_mask	Mask for outermostVidType.	0x1

Bits	Field Name	Description	Default Value
38	outermostVidType	The outermost VID is a S-tag or C-Tag. 0 = Customer tag 1 = Service tag	0x0

32.7.16 Egress VLAN Translation TCAM Answer

This is the table holding the answer for the **Egress VLAN Translation TCAM**.

Number of Entries : 8
Type of Operation : Read/Write
Addressing : **Egress VLAN Translation TCAM** hit index
Address Space : 280359 to 280366

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

32.7.17 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 : 53
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 280367 to 280419

Field Description

Bits	Field Name	Description	Default Value
0	outputMirrorEnabled	If set to one, output mirroring is enabled for this port.	0x0
6:1	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
7	omUnderVlanMembership	If set, output mirroring to a destination that not a member of the VLAN will be ignored.	0x0

32.8 Flow Control

32.8.1 FFA Used

Total number of cells used from the common pool.

Number of Entries : 1
Type of Operation : Read Only
Address Space : 272223

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Number of cells	0x0

32.8.2 Port Pause Settings

Pause settings per source port.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Source port
Address Space : 272224 to 272276

Field Description

Bits	Field Name	Description	Default Value
0	enable	0 = Pausing disabled 1 = Pausing enabled	0x0
1	force	Force pause to the value in pause_pattern 0 = No force 1 = Force Only valid if pausing is enabled.	0x0
2	pattern	The value forced when pause_force is set 0 = Not paused 1 = Paused	0x0

32.8.3 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 : 53
Type of Operation : Read/Write
Addressing : Source port
Address Space : 272117 to 272169

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Number of cells	0xb

32.8.4 Port Tail-Drop FFA Threshold

Settings for the Port Tail-Drop FFA Threshold

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Source port
Address Space :	272439 to 272491

Field Description

Bits	Field Name	Description	Default Value
13:0	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	0x349a
14	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
15	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

32.8.5 Port Tail-Drop Settings

Tail-drop settings per source port.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Source port
Address Space :	272277 to 272329

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

32.8.6 Port Used

Total number of cells used for this source port

Number of Entries :	53
Type of Operation :	Read Only
Addressing :	Source port
Address Space :	272170 to 272222

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Number of cells	0x0

32.8.7 Port Xoff FFA Threshold

Settings for Port Xoff FFA Threshold

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Source port
Address Space :	272386 to 272438

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Xoff threshold for the number of used FFA cells for this source port	0x0
14	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
15	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

32.8.8 Port Xon FFA Threshold

Settings for Port Xon FFA Threshold

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Source port
Address Space :	272333 to 272385

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Xon threshold for the number of used FFA cells for this source port	0x0

32.8.9 Tail-Drop FFA Threshold

Settings for Tail-Drop FFA Threshold

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	272332

Field Description

Bits	Field Name	Description	Default Value
13:0	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.	0x3248
14	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
15	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

32.8.10 Xoff FFA Threshold

Settings for Xoff FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 272331

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Xoff threshold for the total number of used FFA cells	0x0
14	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
15	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

32.8.11 Xon FFA Threshold

Settings for Xon FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 272330

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Xon threshold for the total number of used FFA cells	0x0

32.9 Global Configuration

32.9.1 CPU Port

Select which port is the CPU port.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4

Field Description

Bits	Field Name	Description	Default Value
5:0	port	Port number	0x34

32.9.2 Core Tick Configuration

Global register for setting the frequency of the core tick

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 2

Field Description

Bits	Field Name	Description	Default Value
19:0	clkDivider	The master Core Tick will be issued once every $rg\_tick\_div.clkDivider/4$ core clock cycles. If set to zero, there will be no tick.	0x271
23:20	stepDivider	The five 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

32.9.3 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 : 3

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

32.9.4 MAC RX Maximum Packet Length

Packets with length above this value will be dropped.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Ingress Port
Address Space :	48 to 100

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x2580

32.9.5 Scratch

Scratch Register

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	5

Field Description

Bits	Field Name	Description	Default Value
63:0	scratch	scratch field.	0x0

32.10 Ingress Packet Processing

32.10.1 AH Header Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a Authentical Header, the underlaying 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 :	4
Type of Operation :	Read/Write
Address Space :	267343

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	The value to be used to find this packet type.	0x33
61:9	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
114:62	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

32.10.2 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 :	4
Type of Operation :	Read/Write
Address Space :	267323

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
69:17	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
122:70	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

32.10.3 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 :	266504 to 266507

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
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

Bits	Field Name	Description	Default Value
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

32.10.4 BOOTP and DHCP Packet Decoder Options

The UDP port 1 number used by the BOOTP protocol, the underlaying packet must be a IPv4 packet. If L4 Source Port is this value then L4 Destination Port must be egisterbootpUdpPort2 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 :	8
Type of Operation :	Read/Write
Address Space :	268163

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
85:33	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
138:86	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

32.10.5 CAPWAP Packet Decoder Options

The fields needs to determine if a packet is a CAPWAP packet the underlaying 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 : 8
Type of Operation : Read/Write
Address Space : 268171

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
85:33	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
138:86	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

32.10.6 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 : 266390

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

32.10.7 DNS Packet Decoder Options

The TCP/UDP destination port number used to determine if a packet is a DNS packet, the underlaying 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 : 4
Type of Operation : Read/Write
Address Space : 267351

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	l4Port	The value to be used to find this packet type.	0x35
69:17	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
122:70	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

32.10.8 Debug dstPortmask

Packet processing pipeline status for dstPortmask.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	267367

Field Description

Bits	Field Name	Description	Default Value
52:0	value	Status from last processed packet.	0x0

32.10.9 Debug srcPort

Packet processing pipeline status for srcPort.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	266397

Field Description

Bits	Field Name	Description	Default Value
31:0	value	Status from last processed packet.	0x0

32.10.10 ESP Header Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a Authentical Header, the underlaying 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 :	4
Type of Operation :	Read/Write
Address Space :	267347

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	The value to be used to find this packet type.	0x32
61:9	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
114:62	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

32.10.11 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 :	8
Type of Operation :	Read/Write
Address Space :	268179

Field Description

Bits	Field Name	Description	Default Value
158:0	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

32.10.12 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 : 53
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 266398 to 266450

Field Description

Bits	Field Name	Description	Default Value
7:0	q_on	If a bit is set, the corresponding queue is on.	0xff

32.10.13 Expired TTL to CPU

This register provides an option to forward IPv4/IPv6 packets to the CPU port when they hit the ACL action to decrease the value of the TTL field and cause the decreased TTL equals 0. Without enabling this register, the corresponding packets will be dropped.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 266389

Field Description

Bits	Field Name	Description	Default Value
0	enable	If set, IP Packet with TTL less than 2 and hit the ACL action to decrease its TTL will be sent to the CPU port instead of dropped.	0x0

32.10.14 Flooding Action Send to Port

If a packet is flooded and this function is enabled on the source port then the packet is send to a single egress port instead of being flooded to all ports part of the packets VLAN membership.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Source Port
Address Space : 266451 to 266503

Field Description

Bits	Field Name	Description	Default Value
0	enable	Enable sent to port instead of flooding. 0 = Disable 1 = Enable	0x0

Bits	Field Name	Description	Default Value
6:1	destPort	Once enabled this is the destination port to sent the packet to in case of flooding.	0x0

32.10.15 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 : 266393

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

32.10.16 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 : 266391

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 19.1	0x0
3:1	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

32.10.17 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 : 266394

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

32.10.18 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 :	266392

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 19.1	0x0
3:1	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

32.10.19 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 :	266395

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

32.10.20 GRE Packet Decoder Options

The L4 protocol number which is used to detemine 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 :	8
Type of Operation :	Read/Write
Address Space :	268147

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	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
93:41	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
146:94	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

32.10.21 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 :	53
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	266508 to 266560

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

32.10.22 Hardware Learning Configuration

Configure default status for a newly learned entry, learning limits and learning exceptions.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Ingress Port
Address Space :	957 to 1009

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

Bits	Field Name	Description	Default Value
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
18:3	learnLimit	Maximum number of entries can be learned on this port. 0 means no limit.	0x0
19	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
20	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

32.10.23 Hardware Learning Counter

Number of MAC addresses learned by the hardware learning unit. Write 0 to clear.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Ingress Port
Address Space :	1076 to 1128

Field Description

Bits	Field Name	Description	Default Value
15:0	cnt	Number of learned L2 entries.	0x0

32.10.24 ICMP Length Check

Length check for IP packets carrying ICMP protocol data. IP payload length larger than the maximum size defined in this register can cause the packet get dropped.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	266384

Field Description

Bits	Field Name	Description	Default Value
0	dropMaxICMPv4	If set, the IPv4 packet carrying ICMPv4 data size larger than the defined maximum length will be dropped	0x0
14:1	maxICMPv4Bytes	Maximum size of ICMPv4	0x200

Bits	Field Name	Description	Default Value
15	dropMaxICMPv6	If set, the IPv6 packet carrying ICMPv6 data size larager than the defined maximum length will be dropped	0x0
29:16	maxICMPv6Bytes	Maximum size of ICMPv6	0x200

32.10.25 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 :	4
Type of Operation :	Read/Write
Address Space :	267331

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
69:17	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
122:70	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
123	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

32.10.26 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 :	32
Type of Operation :	Read/Write
Address Space :	268347

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.	0x18100000000000000000000000000ff0
512:385	da_ipv6_mask1	Bit mask for da_ipv6_addr1. For each bit of the mask, 1 means valid for comparison.	0xfff0ffffffffffffffffffffffffffff
640:513	da_ipv6_addr2	IPv6 DA to match.	0x6b00000000000000000000000000ff02
656:641	udp1	UDP destination to match.	0x13f
672:657	udp2	UDP destination to match.	0x140
725:673	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
778:726	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
779	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

32.10.27 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 :	4
Type of Operation :	Read/Write
Address Space :	267335

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
69:17	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

Bits	Field Name	Description	Default Value
122:70	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

32.10.28 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 :	131639 to 131894

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

32.10.29 IPv4 TOS Field To Packet Color Mapping Table

Mapping table from TOS in the IPv4 header to a packet inital color.

Number of Entries :	256
Type of Operation :	Read/Write
Addressing :	Incoming IPv4 packets TOS pointer
Address Space :	132167 to 132422

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

32.10.30 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 :	131895 to 132150

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

32.10.31 IPv6 Class of Service Field To Packet Color Mapping Table

Mapping table from Class of service in the IPv6 header to a packet inital color.

Number of Entries :	256
Type of Operation :	Read/Write
Addressing :	Incoming IPv6 packets Class os Service pointer
Address Space :	132423 to 132678

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

32.10.32 Ingress Admission Control Current Status

Number of tokens currently in the token bucket.

Number of Entries :	128
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space :	271670 to 271797

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

32.10.33 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 :	512
Type of Operation :	Read/Write
Addressing :	address[5:0] : Ingress Port address[8:6] : L2 Priority
Address Space :	36119 to 36630

Field Description

Bits	Field Name	Description	Default Value
0	mmpValid	If set, this entry contains a valid MMP pointer	0x0
7:1	mmpPtr	Initial pointer to the ingress MMP.	0x0
9:8	mmpOrder	Order of the initial ingress MMP pointer.	0x0

32.10.34 Ingress Admission Control Mark All Red

Blocking status of the MMP entry due to packet drops in the MMP.

Number of Entries :	128
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space :	270902 to 271029

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

32.10.35 Ingress Admission Control Mark All Red Enable

Option to block metering after MMP packet drops.

Number of Entries :	128
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space :	270774 to 270901

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

32.10.36 Ingress Admission Control Reset

Reset token buckets so that it is back to the inital 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 : 128
Type of Operation : Read/Write
Addressing : Meter Pointer
Address Space : 271542 to 271669

Field Description

Bits	Field Name	Description	Default Value
0	bucketReset	if set, reload with full tokens for token buckets in this entry.	0x1

32.10.37 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 writting 1 to the **Ingress Admission Control Reset** first.

Number of Entries : 128
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Addressing : Meter Pointer
Address Space : 271030 to 271541

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 6 available ticks for token bucket 0. The tick frequencies are configured globaly 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 6 available ticks for token bucket 1. The tick frequencies are configured globaly 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

Bits	Field Name	Description	Default Value
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
80:67	maxLength	Maximum allowed packet length in bytes. Packets with bytes larger than this value will be dropped before metering.	0x3fff
82:81	tokenMode	0 = Count in bytes and add extra bytes for metering. 1 = Count in bytes and substract extra bytes for metering. 2 = Count in packets. 3 = No tokens are counted.	0x0
90:83	byteCorrection	Extra bytes per packet for IFG correction, only valid under byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

32.10.38 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 :	16
Type of Operation :	Read/Write
Addressing :	address[8:0] : hash of {compareData } address[10:9] : bucket number
Address Space :	38679 to 71446

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	compareData	The data which shall be compared in this entry.	0x0
223	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
232:224	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
233	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
234	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0

Bits	Field Name	Description	Default Value
235	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
236	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
237	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
243:238	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
244	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
250:245	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
251	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
252	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
253	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
261:254	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
262	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
263	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
264	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
267:265	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
268	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

Bits	Field Name	Description	Default Value
280:269	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
281	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
283:282	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
284	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
285	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
286	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
287	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
288	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
290:289	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
291	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
292	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
299:293	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
301:300	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
302	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 19.1	0x0
305:303	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
306	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
307	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
319:308	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
320	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

32.10.39 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 : 2048
Type of Operation : Read/Write
Addressing :

Address bits [2:0]	Value from **preLookupAclBits**.
Address bits [4:3]	Number of VLANs in incoming Packet.
Address bits [5:5]	L2 Type Of Packet. 0 = Others - Not listed in this list. 1 = IEEE 1722/AVTP
Address bits [7:6]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS
Address bits [10:8]	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 : 36631 to 38678

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
4:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

32.10.40 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 6 fields (bits) which are set to one are selected. It is not allowed to set more than 6 bit in the bitmask. The fields are described in ACL Fields

Number of Entries : 16
Type of Operation : Read/Write
Addressing : ACL rule pointer
Address Space : 266709 to 266724

Field Description

Bits	Field Name	Description	Default Value
18:0	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 6 bits should be set.	0x0

32.10.41 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. Seperate 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 :	268923

Field Description

Bits	Field Name	Description	Default Value
221:0	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.	$2^{222} - 1$
443:222	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.	$2^{222} - 1$

32.10.42 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 :	266385

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

32.10.43 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 :	16
Type of Operation :	Read/Write
Addressing :	address[5:0] : hash of {compareData } address[7:6] : bucket number
Address Space :	71447 to 75542

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	compareData	The data which shall be compared in this entry.	0x0
223	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
232:224	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
233	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
234	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
235	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
236	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
237	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
243:238	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
244	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
250:245	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
251	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
252	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0

Bits	Field Name	Description	Default Value
253	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
261:254	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
262	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
263	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
264	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
267:265	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
268	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
280:269	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
281	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
283:282	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
284	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
285	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
286	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
287	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
288	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
290:289	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
291	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
292	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
299:293	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
301:300	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
302	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 19.1	0x0
305:303	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
306	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
307	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
319:308	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
320	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

32.10.44 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 :	32
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	269483 to 269994

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	mask	Which bits to compare in this entry.	$2^{222}-1$
444:223	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

32.10.45 Ingress Configurable ACL 0 TCAM Answer

This is the table holding the answer for the **Ingress Configurable ACL 0 TCAM**.

Number of Entries :	32
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	**Ingress Configurable ACL 0 TCAM** hit index
Address Space :	75543 to 75670

Field Description

Bits	Field Name	Description	Default Value
0	macOp	If set this packets MAC SA and DA can be changed.	0x0
9:1	macOpPtr	Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
10	macPrio	If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
11	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
12	decTtl	If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
13	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
14	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
20:15	destPort	The port which the packet shall be sent to.	0x0
21	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
27:22	destInputMirror	Destination physical port for input mirroring.	0x0
28	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
29	noLearning	If set this packets MAC SA will not be learned.	0x0
30	updateCounter	When set the selected statistics counter will be updated.	0x0
38:31	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
39	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
40	newCfiDeiValue	The value to update to.	0x0
41	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
44:42	newPcpValue	The PCP value to update to.	0x0

Bits	Field Name	Description	Default Value
45	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
57:46	newVidValue	The VID value to update to.	0x0
58	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
60:59	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
61	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
62	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
63	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
64	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
65	forceColor	If set, the packet shall have a forced color.	0x0
67:66	color	Initial color of the packet if the forceColor field is set.	0x0
68	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
69	mmpValid	If set, this entry contains a valid MMP pointer	0x0
76:70	mmpPtr	Ingress MMP pointer.	0x0
78:77	mmpOrder	Ingress MMP pointer order.	0x0
79	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 19.1	0x0
82:80	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
83	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
84	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
96:85	forceVid	The new Ingress VID.	0x0
97	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

32.10.46 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 :	1024
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	address[7:0] : hash of {compareData } address[9:8] : bucket number
Address Space :	77719 to 94102

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
322:1	compareData	The data which shall be compared in this entry.	0x0
323	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
332:324	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
333	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
334	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
335	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
336	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
337	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
343:338	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
344	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
350:345	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
351	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
352	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
353	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
361:354	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0

Bits	Field Name	Description	Default Value
362	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
363	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
364	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
367:365	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
368	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
380:369	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
381	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
383:382	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
384	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
385	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
386	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
387	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
388	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
390:389	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
391	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
392	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
399:393	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0

Bits	Field Name	Description	Default Value
401:400	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
402	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 19.1	0x0
405:403	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
406	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
407	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
419:408	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
420	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

32.10.47 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 : 2048

Type of Operation : Read/Write

Addressing :

Address bits [2:0]	Value from **preLookupAclBits**.
Address bits [4:3]	Number of VLANs in incoming Packet.
Address bits [5:5]	L2 Type Of Packet. 0 = Others - Not listed in this list. 1 = IEEE 1722/AVTP
Address bits [7:6]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS
Address bits [10:8]	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 : 75671 to 77718

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
4:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

32.10.48 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 6 fields (bits) which are set to one are selected. It is not allowed to set more than 6 bit in the bitmask. The fields are described in ACL Fields

Number of Entries :	16
Type of Operation :	Read/Write
Addressing :	ACL rule pointer
Address Space :	266693 to 266708

Field Description

Bits	Field Name	Description	Default Value
30:0	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 6 bits should be set.	0x0

32.10.49 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. Seperate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	32
Type of Operation :	Read/Write
Address Space :	268379

Field Description

Bits	Field Name	Description	Default Value
321:0	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.	$2^{322} - 1$
643:322	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.	$2^{322} - 1$

32.10.50 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 :	266386

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

32.10.51 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 :	128
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	address[4:0] : hash of {compareData } address[6:5] : bucket number
Address Space :	94103 to 96150

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
322:1	compareData	The data which shall be compared in this entry.	0x0
323	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
332:324	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
333	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
334	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
335	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
336	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

Bits	Field Name	Description	Default Value
337	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
343:338	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
344	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
350:345	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
351	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
352	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
353	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
361:354	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
362	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
363	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
364	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
367:365	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
368	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
380:369	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
381	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

Bits	Field Name	Description	Default Value
383:382	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
384	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
385	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
386	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
387	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
388	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
390:389	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
391	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
392	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
399:393	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
401:400	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
402	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 19.1	0x0
405:403	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
406	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
407	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
419:408	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
420	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

32.10.52 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 : 16
Number of Addresses per Entry : 32
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 268411 to 268922

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
322:1	mask	Which bits to compare in this entry.	$2^{322}-1$
644:323	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

32.10.53 Ingress Configurable ACL 1 TCAM Answer

This is the table holding the answer for the **Ingress Configurable ACL 1 TCAM**.

Number of Entries : 16
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Addressing : **Ingress Configurable ACL 1 TCAM** hit index
Address Space : 96151 to 96214

Field Description

Bits	Field Name	Description	Default Value
0	macOp	If set this packets MAC SA and DA can be changed.	0x0
9:1	macOpPtr	Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
10	macPrio	If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
11	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
12	decTtl	If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
13	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
14	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
20:15	destPort	The port which the packet shall be sent to.	0x0

Bits	Field Name	Description	Default Value
21	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
27:22	destInputMirror	Destination physical port for input mirroring.	0x0
28	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
29	noLearning	If set this packets MAC SA will not be learned.	0x0
30	updateCounter	When set the selected statistics counter will be updated.	0x0
38:31	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
39	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
40	newCfiDeiValue	The value to update to.	0x0
41	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
44:42	newPcpValue	The PCP value to update to.	0x0
45	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
57:46	newVidValue	The VID value to update to.	0x0
58	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
60:59	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
61	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
62	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
63	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
64	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
65	forceColor	If set, the packet shall have a forced color.	0x0
67:66	color	Initial color of the packet if the forceColor field is set.	0x0
68	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
69	mmpValid	If set, this entry contains a valid MMP pointer	0x0
76:70	mmpPtr	Ingress MMP pointer.	0x0
78:77	mmpOrder	Ingress MMP pointer order.	0x0

Bits	Field Name	Description	Default Value
79	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 19.1	0x0
82:80	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
83	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
84	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
96:85	forceVid	The new Ingress VID.	0x0
97	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

32.10.54 Ingress Configurable ACL 2 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 :	512		
Number of Addresses per Entry :	16		
Type of Operation :	Read/Write		
Addressing :	address[6:0]	:	hash of {compareData }
	address[8:7]	:	bucket number
Address Space :	98263 to 106454		

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	compareData	The data which shall be compared in this entry.	0x0
223	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
232:224	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
233	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
234	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
235	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0

Bits	Field Name	Description	Default Value
236	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
237	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
243:238	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
244	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
250:245	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
251	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
252	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
253	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
261:254	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
262	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
263	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
264	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
267:265	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
268	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
280:269	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
281	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

Bits	Field Name	Description	Default Value
283:282	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
284	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
285	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
286	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
287	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
288	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
290:289	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
291	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
292	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
299:293	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
301:300	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
302	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 19.1	0x0
305:303	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
306	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
307	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
319:308	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
320	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

32.10.55 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 : 2048
Type of Operation : Read/Write
Addressing :

Address bits [2:0]	Value from **preLookupAclBits**.
Address bits [4:3]	Number of VLANs in incoming Packet.
Address bits [5:5]	L2 Type Of Packet. 0 = Others - Not listed in this list. 1 = IEEE 1722/AVTP
Address bits [7:6]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS
Address bits [10:8]	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 : 96215 to 98262

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
4:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

32.10.56 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 6 fields (bits) which are set to one are selected. It is not allowed to set more than 6 bit in the bitmask. The fields are described in ACL Fields

Number of Entries : 16
Type of Operation : Read/Write
Addressing : ACL rule pointer
Address Space : 266677 to 266692

Field Description

Bits	Field Name	Description	Default Value
30:0	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 6 bits should be set.	0x0

32.10.57 Ingress Configurable ACL 2 Search Mask

Before the hashing and searching is done in the **Ingress Configurable ACL 2 Large Table** and **Ingress Configurable ACL 2 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. Seperate 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 :	268939

Field Description

Bits	Field Name	Description	Default Value
221:0	mask_small	Which bits to compare in the **Ingress Configurable ACL 2 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.	$2^{222} - 1$
443:222	mask_large	Which bits to compare in the **Ingress Configurable ACL 2 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.	$2^{222} - 1$

32.10.58 Ingress Configurable ACL 2 Selection

This register selects which result to use when there are multiple hits.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	266387

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

32.10.59 Ingress Configurable ACL 2 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 :	64
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	address[3:0] : hash of {compareData } address[5:4] : bucket number
Address Space :	106455 to 107478

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	compareData	The data which shall be compared in this entry.	0x0
223	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
232:224	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
233	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
234	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
235	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
236	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
237	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
243:238	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
244	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
250:245	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
251	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
252	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0

Bits	Field Name	Description	Default Value
253	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
261:254	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
262	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
263	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
264	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
267:265	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
268	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
280:269	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
281	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
283:282	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
284	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
285	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
286	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
287	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
288	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
290:289	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
291	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
292	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
299:293	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
301:300	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
302	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 19.1	0x0
305:303	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
306	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
307	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
319:308	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
320	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

32.10.60 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 :	16
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	269227 to 269482

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	mask	Which bits to compare in this entry.	$2^{222}-1$
444:223	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

32.10.61 Ingress Configurable ACL 2 TCAM Answer

This is the table holding the answer for the **Ingress Configurable ACL 2 TCAM**.

Number of Entries :	16
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	**Ingress Configurable ACL 2 TCAM** hit index
Address Space :	107479 to 107542

Field Description

Bits	Field Name	Description	Default Value
0	macOp	If set this packets MAC SA and DA can be changed.	0x0
9:1	macOpPtr	Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
10	macPrio	If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
11	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
12	decTtl	If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
13	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
14	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
20:15	destPort	The port which the packet shall be sent to.	0x0
21	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
27:22	destInputMirror	Destination physical port for input mirroring.	0x0
28	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
29	noLearning	If set this packets MAC SA will not be learned.	0x0
30	updateCounter	When set the selected statistics counter will be updated.	0x0
38:31	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
39	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
40	newCfiDeiValue	The value to update to.	0x0
41	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
44:42	newPcpValue	The PCP value to update to.	0x0

Bits	Field Name	Description	Default Value
45	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
57:46	newVidValue	The VID value to update to.	0x0
58	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
60:59	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
61	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
62	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
63	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
64	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
65	forceColor	If set, the packet shall have a forced color.	0x0
67:66	color	Initial color of the packet if the forceColor field is set.	0x0
68	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
69	mmpValid	If set, this entry contains a valid MMP pointer	0x0
76:70	mmpPtr	Ingress MMP pointer.	0x0
78:77	mmpOrder	Ingress MMP pointer order.	0x0
79	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 19.1	0x0
82:80	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
83	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
84	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
96:85	forceVid	The new Ingress VID.	0x0
97	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

32.10.62 Ingress Configurable ACL 3 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 :	256
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	address[5:0] : hash of {compareData } address[7:6] : bucket number
Address Space :	109591 to 113686

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	compareData	The data which shall be compared in this entry.	0x0
223	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
232:224	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
233	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
234	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
235	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
236	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
237	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
243:238	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
244	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
250:245	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
251	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
252	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
253	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
261:254	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0

Bits	Field Name	Description	Default Value
262	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
263	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
264	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
267:265	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
268	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
280:269	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
281	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
283:282	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
284	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
285	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
286	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
287	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
288	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
290:289	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
291	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
292	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
299:293	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0

Bits	Field Name	Description	Default Value
301:300	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
302	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 19.1	0x0
305:303	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
306	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
307	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
319:308	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
320	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

32.10.63 Ingress Configurable ACL 3 Pre Lookup

The pre ACL lookup allows the user to defined a specific rules for certain packet types in the ACL engine 3. Setting the valid bit and a new rule will override the default rule pointer from the source port table.

Number of Entries : 2048

Type of Operation : Read/Write

Addressing :

Address bits [2:0]	Value from **preLookupAclBits**.
Address bits [4:3]	Number of VLANs in incoming Packet.
Address bits [5:5]	L2 Type Of Packet. 0 = Others - Not listed in this list. 1 = IEEE 1722/AVTP
Address bits [7:6]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS
Address bits [10:8]	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 : 107543 to 109590

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
4:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

32.10.64 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 6 fields (bits) which are set to one are selected. It is not allowed to set more than 6 bit in the bitmask. The fields are described in ACL Fields

Number of Entries :	16
Type of Operation :	Read/Write
Addressing :	ACL rule pointer
Address Space :	266661 to 266676

Field Description

Bits	Field Name	Description	Default Value
30:0	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 6 bits should be set.	0x0

32.10.65 Ingress Configurable ACL 3 Search Mask

Before the hashing and searching is done in the **Ingress Configurable ACL 3 Large Table** and **Ingress Configurable ACL 3 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. Seperate 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 :	268955

Field Description

Bits	Field Name	Description	Default Value
221:0	mask_small	Which bits to compare in the **Ingress Configurable ACL 3 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.	$2^{222}-1$
443:222	mask_large	Which bits to compare in the **Ingress Configurable ACL 3 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.	$2^{222}-1$

32.10.66 Ingress Configurable ACL 3 Selection

This register selects which result to use when there are multiple hits.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	266388

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

32.10.67 Ingress Configurable ACL 3 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 :	64
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	address[3:0] : hash of {compareData } address[5:4] : bucket number
Address Space :	113687 to 114710

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	compareData	The data which shall be compared in this entry.	0x0
223	macOp	This is a result field used when this entry is hit. If set this packets MAC SA and DA can be changed.	0x0
232:224	macOpPtr	This is a result field used when this entry is hit. Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
233	macPrio	This is a result field used when this entry is hit. If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
234	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
235	decTtl	This is a result field used when this entry is hit. If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
236	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

Bits	Field Name	Description	Default Value
237	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
243:238	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
244	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
250:245	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
251	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
252	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
253	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
261:254	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
262	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
263	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
264	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
267:265	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
268	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
280:269	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
281	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

Bits	Field Name	Description	Default Value
283:282	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
284	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
285	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
286	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
287	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
288	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
290:289	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
291	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
292	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
299:293	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
301:300	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
302	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 19.1	0x0
305:303	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
306	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
307	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
319:308	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
320	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

32.10.68 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 :	16
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	268971 to 269226

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
222:1	mask	Which bits to compare in this entry.	$2^{222} - 1$
444:223	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

32.10.69 Ingress Configurable ACL 3 TCAM Answer

This is the table holding the answer for the **Ingress Configurable ACL 3 TCAM**.

Number of Entries :	16
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	**Ingress Configurable ACL 3 TCAM** hit index
Address Space :	114711 to 114774

Field Description

Bits	Field Name	Description	Default Value
0	macOp	If set this packets MAC SA and DA can be changed.	0x0
9:1	macOpPtr	Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0
10	macPrio	If multiple mac operations are set and this prio bit is set then this mac operation pointer will be selected.	0x0
11	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
12	decTtl	If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
13	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
14	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
20:15	destPort	The port which the packet shall be sent to.	0x0

Bits	Field Name	Description	Default Value
21	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
27:22	destInputMirror	Destination physical port for input mirroring.	0x0
28	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
29	noLearning	If set this packets MAC SA will not be learned.	0x0
30	updateCounter	When set the selected statistics counter will be updated.	0x0
38:31	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
39	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
40	newCfiDeiValue	The value to update to.	0x0
41	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
44:42	newPcpValue	The PCP value to update to.	0x0
45	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
57:46	newVidValue	The VID value to update to.	0x0
58	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
60:59	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
61	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
62	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
63	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
64	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
65	forceColor	If set, the packet shall have a forced color.	0x0
67:66	color	Initial color of the packet if the forceColor field is set.	0x0
68	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
69	mmpValid	If set, this entry contains a valid MMP pointer	0x0
76:70	mmpPtr	Ingress MMP pointer.	0x0
78:77	mmpOrder	Ingress MMP pointer order.	0x0

Bits	Field Name	Description	Default Value
79	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 19.1	0x0
82:80	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
83	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
84	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
96:85	forceVid	The new Ingress VID.	0x0
97	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

32.10.70 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 : 269995

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 packt is dropped due to destination VLAN membership check.	0x1
3	learnL2HairpinDrop	Allow learning when the packet is dropped due to hairpin configurations.	0x0

32.10.71 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 : 53
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 267059 to 267270

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	dropStaggedVlans	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 dropStaggedVlans 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
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 = 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

Bits	Field Name	Description	Default Value
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
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
70:18	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

32.10.72 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
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Address Space :	267319

Field Description

Bits	Field Name	Description	Default Value
15:0	typeValue	Ethernet Type value.	0xffff
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
70:18	ignoreStag	If set, type value 0x88A8 is not parsed as Service VLAN type.	0x0

32.10.73 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
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	267365

Field Description

Bits	Field Name	Description	Default Value
52:0	dropMask	Each bit in this mask refers to if ingress MMP drop is allowed on the corresponding egress port.	$2^{53} - 1$

32.10.74 Ingress Multiple Spanning Tree State

Table of ingress Multiple Spanning Tree Protocol Instances. The field **msptPtr** in the **VLAN Table** is used to address this table. Each entry contains the ingress spanning tree states for all ports in this MSTI.

Number of Entries :	64
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	**msptPtr** from **VLAN Table**
Address Space :	131383 to 131638

Field Description

Bits	Field Name	Description	Default Value
105:0	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

32.10.75 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 :	53
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	266725 to 266777

Field Description

Bits	Field Name	Description	Default Value
0	dropMacDaLocal	If bit 47 in the DA MAC address is set to zero then packet will be dropped. This is sometimes referred to as the Local / Globally Administered bit.	0x0
1	dropMacDaGlobal	If bit 47 in the DA MAC is set to one then packet will be dropped. This is sometimes referred to as the Local / Globally Administered bit.	0x0
2	dropMacDaUnicast	If bit 48 in the DA MAC is set to zero then packet will be dropped. This is sometimes referred to as the Multicast/Unicast bit, 0 being a unicast DA Address.	0x0
3	dropMacSaLocal	If bit 47 in the SA MAC address is set to zero then packet will be dropped. This is sometimes referred to as the Local / Globally Administered bit.	0x0
4	dropMacSaGlobal	If bit 47 in the SA MAC is set to one then packet will be dropped. This is sometimes referred to as the Local / Globally Administered bit.	0x0
5	dropMacSaNotSourceRouted	If bit 48 in the SA MAC address is set to zero then packet will be dropped. This is sometimes referred to as the Routing Information Indicator bit.	0x0
6	dropMacSaSourceRouted	If bit 48 in the SA MAC is set to one then packet will be dropped. This is sometimes referred to as the Routing Information Indicator bit.	0x0
7	dropDaMac0	Drop or allow DA MAC 00:00:00:00:00:00. 0 = Allow 1 = Drop	0x0

Bits	Field Name	Description	Default Value
8	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
9	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
10	moreThanOneVlans	When filtering with dropCtaggedVlans or dropStaggedVlans then this field must be set to 1.	0x0
11	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this ingress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
12	dropSingleTaggedVlans	Drop or Allow packets that are VLAN untagged on this ingress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
13	dropMacDaEqSa	Drop or allow MAC packets which has a DA==SA on this ingress port. 0 = Allow MAC DA == MAC SA packets. 1 = Drop MAC DA == MAC SA packets.	0x0
14	dropIPv4DaEqSa	Drop or allow IPv4 packets which has a DA IP==SA IP on this ingress port. 0 = Allow IPv4 DA == IPv4 SA packets. 1 = Drop IPv4 DA == IPv4 SA packets.	0x0
15	dropIPv6DaEqSa	Drop or allow IPv6 packets which has a DA IP==SA IP on this ingress port. 0 = Allow IPv6 DA == IPv6 SA packets. 1 = Drop IPv6 DA == IPv6 SA packets.	0x0
16	dropIPv4Packets	Drop or allow IPv4 packets on this ingress port. 0 = Allow IPv4 packets. 1 = Drop IPv4 packets.	0x0
17	dropIPv6Packets	Drop or allow IPv6 packets on this ingress port. 0 = Allow IPv6 packets. 1 = Drop IPv6 packets.	0x0
18	dropMPLSPackets	Drop or allow MPLS packets on this ingress port. 0 = Allow MPLS packets. 1 = Drop MPLS packets.	0x0
19	dropIPv4MulticastPackets	Drop or allow IPv4 multicast packets on this ingress port. 0 = Allow IPv4 MC packets. 1 = Drop IPv4 MC packets.	0x0
20	dropIPv6MulticastPackets	Drop or allow IPv6 multicast packets on this ingress port. 0 = Allow IPv6 MC packets. 1 = Drop IPv6 MC packets.	0x0

Bits	Field Name	Description	Default Value
21	dropL2BroadcastFrames	Drop or allow L2 broadcast packets on this ingress port. 0 = Drop L2 broadcast packets. 1 = Allow L2 broadcast packets.	0x0
22	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
23	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
24	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
25	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
26	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
27	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

32.10.76 Ingress Ports With Timestamp

Determines which ports that have a timestamp of 8-bytes first in the incoming packet. The timestamp bytes are removed in the normal L2/L3 decoding but are inserted in the To CPU Tag.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	267355

Field Description

Bits	Field Name	Description	Default Value
52:0	hasTimestamp	Each bit set corresponds to an ingress port that have a timestamp prepended to all packets. Bit 0 corresponds to port 0.	0x0

32.10.77 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 : 266657 to 266660

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

32.10.78 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
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 267271 to 267286

Field Description

Bits	Field Name	Description	Default Value
52:0	ports	Ports that this range search is activated on.	0x0
68:53	start	Start of Ethernet type range.	0x0
84:69	end	End of Ethernet type range.	0x0

32.10.79 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 : 114995 to 114998

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

32.10.80 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
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 267287 to 267302

Field Description

Bits	Field Name	Description	Default Value
52:0	ports	Ports that this range search is activated on.	0x0
53	vtype	Shall this entry match S-Type or C-Type VLAN. 0 = C-Type 1 = S-Type	0x0
65:54	start	Start of VID range.	0x0
77:66	end	End of VID range.	0x0

32.10.81 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 : 114987 to 114990

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0

Bits	Field Name	Description	Default Value
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

32.10.82 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 :	8
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	268187 to 268218

Field Description

Bits	Field Name	Description	Default Value
52:0	ports	Ports that this range search is activated on.	0x0
53	saOrDa	Is this rule for source or destination MAC address. 0 = Source MAC 1 = Destination MAC	0x0
101:54	start	Start of MAC address range.	0x0
149:102	end	End of MAC address range.	0x0

32.10.83 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 :	114991 to 114994

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

32.10.84 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
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	267303 to 267318

Field Description

Bits	Field Name	Description	Default Value
52:0	ports	Ports that this range search is activated on.	0x0
53	vtype	Shall this entry match S-Type or C-Type VLAN. 0 = C-Type 1 = S-Type	0x0
65:54	start	Start of VID range.	0x0
77:66	end	End of VID range.	0x0

32.10.85 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 l2 destination MAC lookup, the l2 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 destiantion 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 **l2ActionTablePortState**.
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:	Destiantion Port State Bit comes from the **L2 Action Table Egress Port State**.

Address Space :	266127 to 266254

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	noPortMove	No port move is allowed for this packet.	0x0
7	useSpecialAllow	Use the special frame checks on this port. 0 = No. 1 = Yes.	0x0
9:8	allowPtr	Pointer to allow special packets defined in **Allow Special Frame Check For L2 Action Table**.	0x0
10	mmpValid	If set, this entry contains a valid MMP pointer	0x0
17:11	mmpPtr	Ingress MMP pointer.	0x0
19:18	mmpOrder	Ingress MMP pointer order.	0x0

32.10.86 L2 Action Table Egress Port State

The egress port state for the L2 Action Table Lookup.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	267363

Field Description

Bits	Field Name	Description	Default Value
52:0	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

32.10.87 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 **l2ActionTablePortState**.
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 : 266255 to 266382

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	noPortMove	No port move is allowed for this packet.	0x0
7	useSpecialAllow	Use the special frame checks on this port. 0 = No. 1 = Yes.	0x0
9:8	allowPtr	Pointer to allow special packets defined in **Allow Special Frame Check For L2 Action Table**.	0x0
10	mmpValid	If set, this entry contains a valid MMP pointer	0x0
17:11	mmpPtr	Ingress MMP pointer.	0x0
19:18	mmpOrder	Ingress MMP pointer order.	0x0

32.10.88 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 :	64
Type of Operation :	Read/Write
Addressing :	**L2 Lookup Collision Table** hit index
Address Space :	266561 to 266624

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

32.10.89 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 :	64
Type of Operation :	Read/Write
Addressing :	**L2 Lookup Collision Table** hit index
Address Space :	1012 to 1075

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

32.10.90 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**.

Number of Entries :	32768
Type of Operation :	Read/Write
Addressing :	address[0:11] : hash of {GID, destination MAC} address[12:14] : bucket number
Address Space :	132687 to 165454

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding hash table entry is valid.	0x0

32.10.91 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**.

Number of Entries :	32768	
Type of Operation :	Read/Write	
Addressing :	address[0:11] :	hash of {GID, destination MAC}
	address[12:14] :	bucket number
Address Space :	1129 to 33896	

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

32.10.92 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 8 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. .

Number of Entries :	32768	
Number of Addresses per Entry :	2	
Type of Operation :	Read/Write	
Addressing :	address[0:11] :	hash of {GID, destination MAC}
	address[12:14] :	bucket number
Address Space :	165455 to 230990	

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address.	0x0
59:48	gid	Global identifier from the VLAN Table.	0x0

32.10.93 L2 Destination Table

This table contains either a destination port or a pointer to the L2 multicast table..

Number of Entries :	32832	
Type of Operation :	Read/Write	
Addressing :	address 0 to 32767 :	**L2 DA Hash Lookup Table** address
	address 32768 to 32831 :	**L2 Lookup Collision Table** address
Address Space :	230991 to 263822	

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
10:1	destPort_or_mcAddr	Destination port number or pointer into the **L2 Multicast Table**.	0x0
11	pktDrop	If set, the packet will be dropped and the **L2 Lookup Drop** incremented.	0x0
12	pktDropSa	If set, the packet will be dropped if this packet was hit with the SA search and the **L2 Destination Table SA Lookup Drop** incremented.	0x0
13	l2ActionTableDaStatus	The status DA bit to be used in the addressing for the table **L2 Action Table** Lookup.	0x0
14	l2ActionTableSaStatus	The status SA bit to be used in the addressing for the table **L2 Action Table** Lookup.	0x0

32.10.94 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 :	64
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	268003 to 268130

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address	0x0
59:48	gid	Global identifier for learning	0x0

32.10.95 L2 Lookup Collision Table Masks

Masks for collision memory for the MAC address and the global identifier. Only the first 8entries has masks on them.

Number of Entries :	8
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	267987 to 268002

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

32.10.96 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 :	266396

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

32.10.97 L2 Multicast Table

L2 multicast table.

Number of Entries :	1024
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	mcAddr field from **L2 Destination Table**
Address Space :	263823 to 265870

Field Description

Bits	Field Name	Description	Default Value
52:0	mcPortMask	L2 portmask entry members. If set, the port is part of multicast group and shall be transmitted to.	$2^{53}-1$

32.10.98 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
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Addressing :	MAC DA[7:0]
Address Space :	34071 to 36118

Field Description

Bits	Field Name	Description	Default Value
52:0	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
105:53	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
158:106	sendToPortMask	Send the packet to a specific port. 0 = Do not sent to a port. 1 = Send to port.	0x0
164:159	destPort	The port which the packet shall be sent to.	0x0

32.10.99 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
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	267357

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

32.10.100 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 :	8
Type of Operation :	Read/Write
Address Space :	268155

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
101:49	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
154:102	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

32.10.101 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 :	8
Type of Operation :	Read/Write
Address Space :	268139

Field Description

Bits	Field Name	Description	Default Value
47:0	mac1	DA MAC address to match for LLDP packet.	0x180c200000e
95:48	mac2	DA MAC address to match for LLDP packet.	0x180c2000003
143:96	mac3	DA MAC address to match for LLDP packet.	0x180c2000000
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
213:161	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.	$2^{52} - 1$

32.10.102 Learning And Aging Enable

Enable/Disable the learning and aging function. If software needs to take fully control over learning and aging tables by writting 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.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	956

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

Bits	Field Name	Description	Default Value
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

32.10.103 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 untill the software clears the valid field.

Number of Entries :	1
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Address Space :	948

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
66:61	port	Port number.	0x0

32.10.104 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 untill the software clears the valid field.

Number of Entries :	1
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Address Space :	952

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
66:61	port	Port number.	0x0

32.10.105 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
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	The link aggregate hash.
Address Space :	267475 to 267986

Field Description

Bits	Field Name	Description	Default Value
52:0	ports	One bit per physical port.	0x0

32.10.106 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 :	266383

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

Bits	Field Name	Description	Default Value
6	useVlanIdInHash	The packets VLAN Identifier tag shall be part of the hash key when calculating the link aggregate hash value.	0x0

32.10.107 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 enterned the switch.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	266794 to 266846

Field Description

Bits	Field Name	Description	Default Value
5:0	la	The Link aggregation which this port is a member of	0x0

32.10.108 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 :	53
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	The link aggregate number.
Address Space :	267369 to 267474

Field Description

Bits	Field Name	Description	Default Value
52:0	members	Physical ports that are members of this link aggregate. One bit per port.	0x0

32.10.109 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 :	266625 to 266632

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue	0x1

32.10.110 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 :	132679 to 132686

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color	0x0

32.10.111 Mask MAC Table Lookup

Which bits shall be used in the hash function and which bits shall be compared in the L2 lookup.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	267359

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddrMask	MAC address mask. 0 = Bit will not be used, 1= bit will be used.	$2^{48} - 1$
59:48	gidMask	Global identifier mask. 0 = Bit will not be used, 1= bit will be used.	0xfff

32.10.112 Port Move Options

Determine if port move is allowed on static entries.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	267361

Field Description

Bits	Field Name	Description	Default Value
52:0	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 coressponding bit set to 1 in this field, it allows the learning engine to update the GID and MAC to the current source port.	$2^{53}-1$

32.10.113 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 :	4
Type of Operation :	Read/Write
Address Space :	267327

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
69:17	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
122:70	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

32.10.114 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 :	8
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	268283 to 268346

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 19.1	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	Inital 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
112:106	mmpPtr	Ingress MMP pointer.	0x0
114:113	mmpOrder	Ingress MMP pointer order.	0x0
167:115	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

32.10.115 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 :	8
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	268219 to 268282

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 19.1	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	Inital 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
112:106	mmpPtr	Ingress MMP pointer.	0x0
114:113	mmpOrder	Ingress MMP pointer order.	0x0
167:115	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

32.10.116 SCTP Packet Decoder Options

The L4 protocol number which is used to detemine 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 :	4
Type of Operation :	Read/Write
Address Space :	267339

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	The value to be used to find this packet type.	0x84
61:9	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

Bits	Field Name	Description	Default Value
114:62	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

32.10.117 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 :	16
Type of Operation :	Read/Write
Addressing :	SMON set number
Address Space :	266778 to 266793

Field Description

Bits	Field Name	Description	Default Value
5:0	srcPort	Source port	0x0
17:6	vid	VLAN ID	0x0

32.10.118 Send to CPU

Configuration of MAC addresses used to redirect packets to CPU.

Number of Entries :	1
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Address Space :	268131

Field Description

Bits	Field Name	Description	Default Value
52:0	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.	$2^{53} - 1$
105:53	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.	$2^{53} - 1$
158:106	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

Bits	Field Name	Description	Default Value
206:159	cpuMacAddr	Packets with this destination MAC address will be sent to the CPU. Only valid if **uniqueCpuMac** on the source port is set.	0x0

32.10.119 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 :	53
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	Source Port
Address Space :	114775 to 114986

Field Description

Bits	Field Name	Description	Default Value
0	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
6:1	destInputMirror	Destination physical port for input mirroring.	0x0
7	noLearning	If set this packets MAC SA will not be learned.	0x0
8	updateCounter	When set the selected statistics counter will be updated.	0x0
16:9	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
17	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
29:18	forceVid	The new Ingress VID.	0x0
30	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
31	newCfiDeiValue	The value to update to.	0x0
32	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
35:33	newPcpValue	The PCP value to update to.	0x0
36	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
48:37	newVidValue	The VID value to update to.	0x0

Bits	Field Name	Description	Default Value
49	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
51:50	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
52	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
53	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
54	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
60:55	destPort	The port which the packet shall be sent to.	0x0
61	forceColor	If set, the packet shall have a forced color.	0x0
63:62	color	Initial color of the packet if the forceColor field is set.	0x0
64	mmpValid	If set, this entry contains a valid MMP pointer	0x0
71:65	mmpPtr	Ingress MMP pointer.	0x0
73:72	mmpOrder	Ingress MMP pointer order.	0x0
74	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 19.1	0x0
77:75	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
78	decTtl	If set this packets L3 (IPv4,IPv6) TTL field will be decremented. If the field is already zero then it will be kept at zero. If this action leads to TTL=0 then the packet is dropped or sent to the CPU port according to **Expired TTL to CPU**	0x0
79	macOp	If set this packets MAC SA and DA can be changed.	0x0
88:80	macOpPtr	Pointer to egress MAC action, defined in **Egress MAC Operations** on what changes shall be done to MAC addresses of the packet.	0x0

32.10.120 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 :	53
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	266847 to 267058

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
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
8:5	aclRule0	Pointer into the **Ingress Configurable ACL 0 Rules Setup** table selecting which ACL fields to select to do the ACL lookup with.	0x0
9	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
13:10	aclRule1	Pointer into the **Ingress Configurable ACL 1 Rules Setup** table selecting which ACL fields to select to do the ACL lookup with.	0x0
14	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
18:15	aclRule2	Pointer into the **Ingress Configurable ACL 2 Rules Setup** table selecting which ACL fields to select to do the ACL lookup with.	0x0
19	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
23:20	aclRule3	Pointer into the **Ingress Configurable ACL 3 Rules Setup** table selecting which ACL fields to select to do the ACL lookup with.	0x0
26:24	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
28:27	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
30:29	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
32:31	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
34:33	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
36:35	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
48:37	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

Bits	Field Name	Description	Default Value
49	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
52:50	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
54:53	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
56:55	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
58:57	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
59	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
60	learnMulticastSaMac	If set, the learning engine allows Ethernet multicast source MAC addresses to be learned.	0x0
61	learnMacDaEqSa	Set to zero to ignore the hardware learning request when MAC DA equals SA.	0x1
62	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
63	imUnderVlanMembership	If set, input mirroring to a destination that not a member of the VLAN will be ignored.	0x0
64	imUnderPortIsolation	If set, input mirroring to a destination that isolated the source port in the **srcPortFilter** will be ignored.	0x0
70:65	destInputMirror	Destination physical port for input mirroring. Only valid if **inputMirrorEnabled** is set.	0x0

Bits	Field Name	Description	Default Value
73:71	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
74	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
86:75	priorityVid	The VID used in the outer VLAN tag comparison, see **enablePriorityTag**.	0x0
87	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
88	l2ActionTablePortState	What is the source port status bit. Used in table **L2 Action Table** and **L2 Action Table Source Port**.	0x0
89	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
90	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
93:91	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

32.10.121 TCP/UDP Flag Rules

IPv4/IPv6 TCP/UDP packets will be compared to all entries in this table. The TCP/UDP flags values can be compared by enabling some of the comparisons. The packets flags will be compared with the values in the entries for all flags that have comparison enabled. If comparison is disabled the flags values will be ignored. In addition the packets IP source and destination addresses are compared and if they are equal this status can also be used in the rules. The TCP source and destination ports are also compared if equal and this status can also be used in the rules. If a packet matches any of these rules the packet will be dropped and the **Attack Prevention Drop** will be incremented.

Number of Entries : 16
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 266641 to 266656

Field Description

Bits	Field Name	Description	Default Value
0	urg	TCP flag URG compare value.	0x0
1	ack	TCP flag ACK compare value.	0x0
2	psh	TCP flag PSH compare value.	0x0
3	rst	TCP flag RST compare value.	0x0
4	syn	TCP flag SYN compare value.	0x0
5	fin	TCP flag FIN compare value.	0x0
6	DaSa	Value of IP address comparison.	0x0
7	SpDpTcp	Value of TCP port comparison.	0x0
8	SpDpUdp	Value of UDP port comparison.	0x0
9	cmpUrg	Enable comparison of URG.	0x0
10	cmpAck	Enable comparison of ACK.	0x0
11	cmpPsh	Enable comparison of PSH.	0x0
12	cmpRst	Enable comparison of RST.	0x0
13	cmpSyn	Enable comparison of SYN.	0x0
14	cmpFin	Enable comparison of FIN.	0x0
15	cmpDaSa	Enable comparison of IP DA equal to SA.	0x0
16	cmpSpDpTcp	Enable comparison of TCP source port equal to destination port.	0x0
17	cmpSpDpUdp	Enable comparison of UDP source port equal to destination port.	0x0
18	enable	Enable this rule.	0x0

32.10.122 Time to Age

Interval period after which FIB entries are aged out.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1010

Field Description

Bits	Field Name	Description	Default Value
31:0	tickCnt	Number of ticks (see Chapter Tick) between starts of the aging process.	$2^{32} - 1$

Bits	Field Name	Description	Default Value
34:32	tick	Select one of the 6 available ticks. The tick frequencies are configured globaly in the **Core Tick Configuration** register.	0x0

32.10.123 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
Address Space :	132151 to 132166	

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

32.10.124 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 :	266633 to 266640

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

32.10.125 VLAN Table

Defines the VLAN port membership, which GID 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 :	114999 to 131382

Field Description

Bits	Field Name	Description	Default Value
52:0	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.	$2^{53}-1$
64:53	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
65	mmpValid	If set, this entry contains a valid MMP pointer	0x0
72:66	mmpPtr	Ingress MMP pointer.	0x0
74:73	mmpOrder	Ingress MMP pointer order.	0x0
80:75	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
83:81	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
85:84	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
87:86	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

Bits	Field Name	Description	Default Value
89:88	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
101:90	vid	The VID used in VLAN push or swap operation if selected by **vidSel**.	0x0
104:102	pcp	The PCP used in VLAN push or swap operation if selected by **pcpSel**.	0x0
105	cfiDei	The CFI/DEI used in VLAN push or swap operation if selected by **cfiDeiSel**	0x0
107:106	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

32.11 MBSC

32.11.1 L2 Broadcast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Broadcast Storm Control

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 357 to 409

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x5c8

32.11.2 L2 Broadcast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Broadcast Storm Control

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 410 to 462

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

32.11.3 L2 Broadcast Storm Control Enable

Bitmask to turn L2 Broadcast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	463

Field Description

Bits	Field Name	Description	Default Value
52:0	enable	Bitmask where the index is the Egress Ports	0x0

32.11.4 L2 Broadcast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Broadcast Storm Control

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Ports
Address Space :	304 to 356

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 six available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	Index / Value 0-47: 0x3 48-52: 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

32.11.5 L2 Multicast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Multicast Storm Control

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 518 to 570

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x5c8

32.11.6 L2 Multicast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Multicast Storm Control

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 571 to 623

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

32.11.7 L2 Multicast Storm Control Enable

Bitmask to turn L2 Multicast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 624

Field Description

Bits	Field Name	Description	Default Value
52:0	enable	Bitmask where the index is the Egress Ports	0x0

32.11.8 L2 Multicast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Multicast Storm Control

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 465 to 517

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 six available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	Index / Value: 0-47 0x3; 48-52 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

32.11.9 L2 Unknown Multicast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Unknown Multicast Storm Control

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 840 to 892

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x5c8

32.11.10 L2 Unknown Multicast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Unknown Multicast Storm Control

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 893 to 945

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

32.11.11 L2 Unknown Multicast Storm Control Enable

Bitmask to turn L2 Unknown Multicast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	946

Field Description

Bits	Field Name	Description	Default Value
52:0	enable	Bitmask where the index is the Egress Ports	0x0

32.11.12 L2 Unknown Multicast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Unknown Multicast Storm Control

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Ports
Address Space :	787 to 839

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 six available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	Index / Value 0-47 / 0x3 48-52 / 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

32.11.13 L2 Unknown Unicast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Unknown Unicast Storm Control

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Ports
Address Space :	679 to 731

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x5c8

32.11.14 L2 Unknown Unicast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Unknown Unicast Storm Control

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Ports
Address Space :	732 to 784

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

32.11.15 L2 Unknown Unicast Storm Control Enable

Bitmask to turn L2 Unknown Unicast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	785

Field Description

Bits	Field Name	Description	Default Value
52:0	enable	Bitmask where the index is the Egress Ports	0x0

32.11.16 L2 Unknown Unicast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Unknown Unicast Storm Control

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Ports
Address Space :	626 to 678

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 six available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	Index / Value: 0-47 → 0x3; 48-52 → 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

32.12 Scheduling

32.12.1 DWRR Bucket Capacity Configuration

Token Bucket Capacity Configuration for DWRR

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 273023 to 273075

Field Description

Bits	Field Name	Description	Default Value
17:0	bucketCapacity	Capacity of the byte bucket	$2^{18} - 1$

32.12.2 DWRR Bucket Misc Configuration

Bucket Configurations for DWRR

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 273076 to 273128

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.	0xe
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

32.12.3 DWRR Weight Configuration

Weight Configuration for DWRR

Number of Entries :	424
Type of Operation :	Read/Write
Addressing :	Egress port * 8 + queue
Address Space :	273129 to 273552

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

32.12.4 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 :	53
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	272063 to 272115

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

32.12.5 Output Disable

Bitmask for disabling the egress queues on egress ports.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	272970 to 273022

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

32.13 Shapers

32.13.1 Port Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Port Shaper

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space :	276182 to 276234

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	Index: 0-47 Value: 0xea6 Index: 48-52 Value: 0x927c

32.13.2 Port Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Port Shaper

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space :	276235 to 276287

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.	Index Value 0-47 0x4e2 48-52 0x30d4

32.13.3 Port Shaper Enable

Bitmask to turn Port Shaper ON/OFF (1/0) for Egress Port

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	276288

Field Description

Bits	Field Name	Description	Default Value
52:0	enable	Bitmask where the index is the Egress Port	0x0

32.13.4 Port Shaper Rate Configuration

Token Bucket rate Configuration for Port Shaper

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space :	276129 to 276181

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
12:1	tokens	The number of tokens added each tick	Index Value 0-47 0x7d 48-52 0x4e2
15:13	tick	Select one of the six available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x0
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

32.13.5 Prio Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Prio Shaper

Number of Entries :	424
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Prio
Address Space :	275265 to 275688

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	Index / Value 0-383: 0xea6 384-423: 0x927c

32.13.6 Prio Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Prio Shaper

Number of Entries :	424
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Prio
Address Space :	275689 to 276112

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.	Index / Value 0-383: 0x4e2 384-423: 0x30d4

32.13.7 Prio Shaper Enable

Bitmask to turn Prio Shaper ON/OFF (1/0) for Egress Port * 8 + Egress Prio

Number of Entries :	1
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Address Space :	276113

Field Description

Bits	Field Name	Description	Default Value
423:0	enable	Bitmask where the index is the Egress Port * 8 + Egress Prio	0x0

32.13.8 Prio Shaper Rate Configuration

Token Bucket rate Configuration for Prio Shaper

Number of Entries :	424
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Prio
Address Space :	274841 to 275264

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
12:1	tokens	The number of tokens added each tick	Index / Value 0-383: 0x7d 384-423: 0x4e2
15:13	tick	Select one of the six available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x0
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

32.13.9 Queue Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Queue Shaper

Number of Entries :	424
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Queue
Address Space :	273977 to 274400

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	Index / Value 0-383: 0xea6 384-423: 0x927c

32.13.10 Queue Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Queue Shaper

Number of Entries :	424
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Queue
Address Space :	274401 to 274824

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.	Index / Value 0-383: 0x4e2 384-423: 0x30d4

32.13.11 Queue Shaper Enable

Bitmask to turn Queue Shaper ON/OFF (1/0) for Egress Port * 8 + Egress Queue

Number of Entries :	1
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Address Space :	274825

Field Description

Bits	Field Name	Description	Default Value
423:0	enable	Bitmask where the index is the Egress Port * 8 + Egress Queue	0x0

32.13.12 Queue Shaper Rate Configuration

Token Bucket rate Configuration for Queue Shaper

Number of Entries :	424
Type of Operation :	Read/Write
Addressing :	Egress Port * 8 + Egress Queue
Address Space :	273553 to 273976

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
12:1	tokens	The number of tokens added each tick	Index / Value 0-383: 0x7d 384-423: 0x4e2
15:13	tick	Select one of the six available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x0
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

32.14 Shared Buffer Memory

32.14.1 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
13:0	cells	Number of free cells.	0x349a

32.14.2 Egress Port Depth

Number of packets available in the buffer memory for each egress port.

Number of Entries : 53
Type of Operation : Read Only
Addressing : Egress Port
Address Space : 272492 to 272544

Field Description

Bits	Field Name	Description	Default Value
13:0	packets	Number of packet currently queued.	0x0

32.14.3 Egress Queue Depth

Number of packets available in the buffer memory for each egress queue.

Number of Entries : 424
Type of Operation : Read Only
Addressing : Global queue number
Address Space : 272545 to 272968

Field Description

Bits	Field Name	Description	Default Value
13:0	packets	Number of packets currently queued.	0x0

32.14.4 Minimum Buffer Free

Minimum number of cells available in the buffer memory

Number of Entries :	1
Type of Operation :	Read Only
Address Space :	272969

Field Description

Bits	Field Name	Description	Default Value
13:0	cells	Number of cells.	0x349a

32.14.5 Packet Buffer Status

Queue status of the packet buffer

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read Only
Address Space :	272059

Field Description

Bits	Field Name	Description	Default Value
52:0	empty	Empty flags for the egress ports	$2^{53} - 1$

32.15 Statistics: ACL

32.15.1 Ingress Configurable ACL Match Counter

Number of packets hit in entries from Ingress configurable ACL lookup.

Number of Entries :	256
Type of Operation :	Read/Write
Addressing :	Index from result of Ingress configurable ACL.
Address Space :	270252 to 270507

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.16 Statistics: Debug

32.16.1 EPP PM Drop

Number of drops due to FIFO overflows in EPP PM.
In Figure 27.1, **epmOverflow** with process sequence **22** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 276552

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.16.2 IPP PM Drop

Number of drops due to FIFO overflows in IPP PM.
In Figure 27.1, **ipmOverflow** with process sequence **12** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34032

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.16.3 PS Error Counter

Number of errors occured in the PS-converter.
In Figure 27.1, **psError** with process sequence **25** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 280594 to 280646

Field Description

Bits	Field Name	Description	Default Value
7:0	underrun	Number of packets which have empty cycles caused by the internal PS-converter but not the external halt during packet transmissions.	0x0

Bits	Field Name	Description	Default Value
15:8	overflow	Number of FIFO overflows in the PS-converter. This error will cause packet corruptions.	0x0

32.16.4 SP Overflow Drop

Number of packets dropped due to: FIFO overflow in the SP-converter.
In Figure 27.1, **spOverflow** with process sequence **5** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read Only
Addressing : Ingress port
Address Space : 33936 to 33988

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets on this ingress port.	0x0

32.17 Statistics: EPP Egress Port Drop

32.17.1 Egress Port Disabled Drop

Number of packets dropped due to egress port disabled.
In Figure 27.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 276446 to 276498

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.17.2 Egress Port Filtering Drop

Number of packets dropped due to egress port filtering.
In Figure 27.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 276499 to 276551

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.17.3 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.
In Figure 27.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	276393 to 276445

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.18 Statistics: IPP Egress Port Drop

32.18.1 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 27.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	53
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space :	270561 to 270613

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.18.2 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 27.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Port (not aggregated)
Address Space : 270667 to 270719

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.18.3 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 27.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Port (not aggregated)
Address Space : 270720 to 270772

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.18.4 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 27.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Port (not aggregated)
Address Space : 270614 to 270666

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.18.5 Queue Off Drop

Number of packets dropped due to the queue being turned off.
In Figure 27.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Port (not aggregated)
Address Space : 270508 to 270560

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19 Statistics: IPP Ingress Port Drop

32.19.1 AH Decoder Drop

Number of packets dropped due to setting in register **AH Header Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34058

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.2 ARP Decoder Drop

Number of packets dropped due to setting in register **ARP Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34051

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.3 Attack Prevention Drop

Number of packets dropped due to matching TCP/UDP flag rule.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34050

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.4 BOOTP and DHCP Decoder Drop

Number of packets dropped due to setting in register **BOOTP and DHCP Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34061

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.5 CAPWAP Decoder Drop

Number of packets dropped due to setting in register **CAPWAP Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34062

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.6 DNS Decoder Drop

Number of packets dropped due to setting in register **DNS Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34060

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.7 ESP Decoder Drop

Number of packets dropped due to setting in register **ESP Header Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34059

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.8 Empty Mask Drop

Number of packets dropped due to an empty destination port mask.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34035

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.9 Expired TTL Drop

Number of packets dropped due to expired TTL.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34046

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.10 GRE Decoder Drop

Number of packets dropped due to setting in register **GRE Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34063

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.11 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34055

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.12 IP Checksum Drop

Number of packets dropped due to incorrect IP checksum.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34047

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.13 Ingress Configurable ACL Drop

Number of packets dropped due to matching an Ingress Configurable ACL with drop.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34049

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.14 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34040

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.15 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34038

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.16 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34037

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.17 Ingress Spanning Tree Drop: Listen

Number of packets dropped due to that a port's ingress spanning tree protocol state was **Listening**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34036

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.18 L2 Action Table Drop

Number of packets dropped due to the **L2 Action Table** says drop all instances.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34065

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.19 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34066

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.20 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34064

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.21 L2 Destination Table SA Lookup Drop

Number of packets dropped due to the table **L2 Destination Table** field

ieldL2 Destination TablepktDropSa says drop.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34067

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.22 L2 IEEE 1588 Decoder Drop

Number of packets dropped due to setting in register **IEEE 1588 L4 Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34053

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.23 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34039

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.24 L2 Reserved Multicast Address Drop

Number of packets dropped due to the L2 Reserved Multicast Addresses on counter 0
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34048

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.25 L4 IEEE 1588 Decoder Drop

Number of packets dropped due to setting in register **IEEE 1588 L4 Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34054

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.26 LACP Decoder Drop

Number of packets dropped due to setting in register **LACP Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34057

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.27 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34045

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.28 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34044

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.29 RARP Decoder Drop

Number of packets dropped due to setting in register **RARP Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34052

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.30 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34041

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.31 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34042

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.32 SCTP Decoder Drop

Number of packets dropped due to setting in register **SCTP Packet Decoder Options**.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34056

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.33 Source Port Default ACL Action Drop

Number of packets dropped due to the table **Source Port Default ACL Action** says drop.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34068

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.34 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 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34034

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.19.35 VLAN Member Drop

Number of packets dropped due to the packets source port notbeing part of the packets VLAN membership.
In Figure 27.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 34043

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.20 Statistics: Misc

32.20.1 Buffer Overflow Drop

Counter for the number of packets dropped due to the shared buffer memory being full.
In Figure 27.1, **bmOverflow** with process sequence **16** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 272061

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.20.2 Drain Port Drop

Number of packets dropped due to the port is drained.
In Figure 27.1, **drain** with process sequence **21** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 276340 to 276392

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.20.3 Egress Resource Manager Drop

Number of packets dropped by the egress resource manager.
In Figure 27.1, **erm** with process sequence **15** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 272006 to 272058

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.20.4 Flow Classification And Metering Drop

Number of packets dropped due to flow classification and metering.
In Figure 27.1, **mmp** with process sequence **14** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	270773

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.20.5 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 27.1, **eopDrop** with process sequence **14** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34033

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.20.6 Ingress Resource Manager Drop

Counter for the number of packets dropped due to exeeding thresholds set up in the ingress resource manager.
In Figure 27.1, **irm** with process sequence **16** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	272062

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets.	0x0

32.20.7 MAC RX Broken Packets

Number of broken packets dropped.
In Figure 27.1, **macBrokenPkt** with process sequence **3** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read Only (unreliable)
Addressing : Ingress Port
Address Space : 101 to 153

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.20.8 MAC RX Long Packet Drop

Number of packets dropped due to length above **MAC RX Maximum Packet Length**.
In Figure 27.1, **macRxMax** with process sequence **4** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read Only (unreliable)
Addressing : Ingress Port
Address Space : 207 to 259

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.20.9 MAC RX Short Packet Drop

Number of packets dropped due to length below 60 bytes.
In Figure 27.1, **macRxMin** with process sequence **4** represents the internal location of this counter.

Number of Entries : 53
Type of Operation : Read Only (unreliable)
Addressing : Ingress Port
Address Space : 154 to 206

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.20.10 Re-queue Overflow Drop

Counter for the number of packets dropped due to a FIFO overflow in re-queue.
In Figure 27.1, **rqOverflow** with process sequence **24** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	272116

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of dropped packets	0x0

32.21 Statistics: Packet Datapath

32.21.1 EPP Packet Head Counter

Number of packet first cells through the Egress Packet Process module.
In Figure 27.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	276553

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet headers.	0x0

32.21.2 EPP Packet Tail Counter

Number of packet last cells through the Egress Packet Process module.
In Figure 27.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	276554

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet tails.	0x0

32.21.3 IPP Packet Head Counter

Number of packet first cells through the Ingress Packet Process module.
In Figure 27.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34069

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet headers.	0x0

32.21.4 IPP Packet Tail Counter

Number of packet last cells through the Ingress Packet Process module.
In Figure 27.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	34070

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet tails.	0x0

32.21.5 PB Packet Head Counter

Number of packet first cells through the Shared Buffer Memory module.
In Figure 27.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	276336

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet headers.	0x0

32.21.6 PB Packet Tail Counter

Number of packet last cells through the Shared Buffer Memory module.
In Figure 27.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 276337

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet tails.	0x0

32.21.7 PS Packet Head Counter

Number of packet first cells through the Parallel to Serial module.
In Figure 27.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 280592

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet headers.	0x0

32.21.8 PS Packet Tail Counter

Number of packet last cells through the Parallel to Serial module.
In Figure 27.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 280593

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packet tails.	0x0

32.22 Statistics: SMON

32.22.1 SMON Set 0 Byte Counter

Number of bytes counted in SMON Set 0.
In Figure 27.1, **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 :	270124 to 270131

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.2 SMON Set 0 Packet Counter

Number of packets counted in SMON Set 0.
In Figure 27.1, **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 :	269996 to 270003

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.3 SMON Set 1 Byte Counter

Number of bytes counted in SMON Set 1.
In Figure 27.1, **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 :	270132 to 270139

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.4 SMON Set 1 Packet Counter

Number of packets counted in SMON Set 1.
In Figure 27.1, **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 :	270004 to 270011

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.5 SMON Set 10 Byte Counter

Number of bytes counted in SMON Set 10.
In Figure 27.1, **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 :	270204 to 270211

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.6 SMON Set 10 Packet Counter

Number of packets counted in SMON Set 10.
In Figure 27.1, **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 :	270076 to 270083

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.7 SMON Set 11 Byte Counter

Number of bytes counted in SMON Set 11.
In Figure 27.1, **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 :	270212 to 270219

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.8 SMON Set 11 Packet Counter

Number of packets counted in SMON Set 11.
In Figure 27.1, **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 :	270084 to 270091

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.9 SMON Set 12 Byte Counter

Number of bytes counted in SMON Set 12.
In Figure 27.1, **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 :	270220 to 270227

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.10 SMON Set 12 Packet Counter

Number of packets counted in SMON Set 12.
In Figure 27.1, **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 :	270092 to 270099

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.11 SMON Set 13 Byte Counter

Number of bytes counted in SMON Set 13.
In Figure 27.1, **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 :	270228 to 270235

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.12 SMON Set 13 Packet Counter

Number of packets counted in SMON Set 13.
In Figure 27.1, **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 :	270100 to 270107

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.13 SMON Set 14 Byte Counter

Number of bytes counted in SMON Set 14.
In Figure 27.1, **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 : 270236 to 270243

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.14 SMON Set 14 Packet Counter

Number of packets counted in SMON Set 14.
In Figure 27.1, **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 : 270108 to 270115

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.15 SMON Set 15 Byte Counter

Number of bytes counted in SMON Set 15.
In Figure 27.1, **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 : 270244 to 270251

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.16 SMON Set 15 Packet Counter

Number of packets counted in SMON Set 15.
In Figure 27.1, **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 :	270116 to 270123

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.17 SMON Set 2 Byte Counter

Number of bytes counted in SMON Set 2.
In Figure 27.1, **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 :	270140 to 270147

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.18 SMON Set 2 Packet Counter

Number of packets counted in SMON Set 2.
In Figure 27.1, **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 :	270012 to 270019

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.19 SMON Set 3 Byte Counter

Number of bytes counted in SMON Set 3.
In Figure 27.1, **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 :	270148 to 270155

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.20 SMON Set 3 Packet Counter

Number of packets counted in SMON Set 3.
In Figure 27.1, **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 :	270020 to 270027

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.21 SMON Set 4 Byte Counter

Number of bytes counted in SMON Set 4.
In Figure 27.1, **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 :	270156 to 270163

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.22 SMON Set 4 Packet Counter

Number of packets counted in SMON Set 4.
In Figure 27.1, **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 :	270028 to 270035

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.23 SMON Set 5 Byte Counter

Number of bytes counted in SMON Set 5.
In Figure 27.1, **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 :	270164 to 270171

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.24 SMON Set 5 Packet Counter

Number of packets counted in SMON Set 5.
In Figure 27.1, **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 :	270036 to 270043

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.25 SMON Set 6 Byte Counter

Number of bytes counted in SMON Set 6.
In Figure 27.1, **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 :	270172 to 270179

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.26 SMON Set 6 Packet Counter

Number of packets counted in SMON Set 6.
In Figure 27.1, **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 :	270044 to 270051

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.27 SMON Set 7 Byte Counter

Number of bytes counted in SMON Set 7.
In Figure 27.1, **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 :	270180 to 270187

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.28 SMON Set 7 Packet Counter

Number of packets counted in SMON Set 7.
In Figure 27.1, **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 :	270052 to 270059

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.29 SMON Set 8 Byte Counter

Number of bytes counted in SMON Set 8.
In Figure 27.1, **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 :	270188 to 270195

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.30 SMON Set 8 Packet Counter

Number of packets counted in SMON Set 8.
In Figure 27.1, **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 :	270060 to 270067

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0

32.22.31 SMON Set 9 Byte Counter

Number of bytes counted in SMON Set 9.
In Figure 27.1, **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 :	270196 to 270203

Field Description

Bits	Field Name	Description	Default Value
23:0	bytes	Number of bytes.	0x0

32.22.32 SMON Set 9 Packet Counter

Number of packets counted in SMON Set 9.
In Figure 27.1, **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 :	270068 to 270075

Field Description

Bits	Field Name	Description	Default Value
23:0	packets	Number of packets.	0x0