Packet Architects AB

Ethernet Switch
VLAN 5x100M
User Guide

Core Revision unknown

Datasheet Revision unknown

Contents

List of Figures

List of Tables

Chapter 1

Overview

This L2 Ethernet Switching Core offers full wire-speed on all 5 ports, with up to 0.1 Gbit of bandwidth per port. Each port has 4 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 80 bytes. In total the buffer memory has a capacity of 2048 cells.

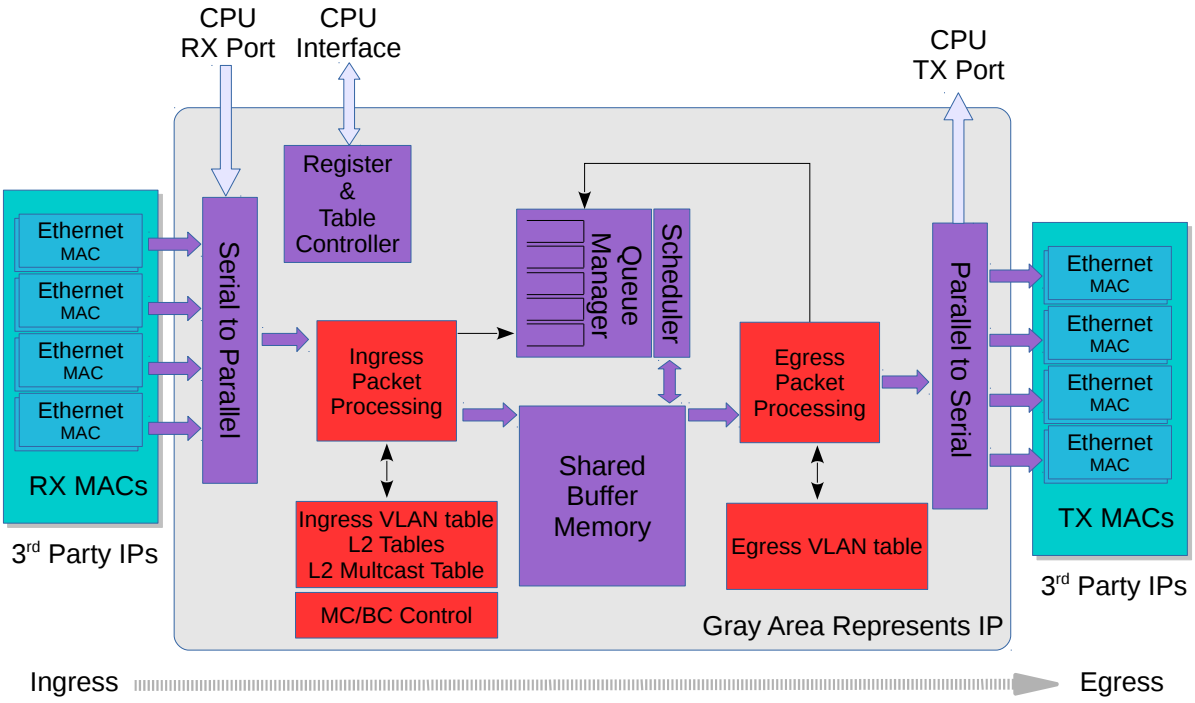


Figure 1.1: Switch Core Overview

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

- 5 x 0.1 Gigabit Ethernet ports.
- Full wire-speed on all ports and all Ethernet frame sizes.
- Store and forward shared memory architecture.
- Support for jumbo packets up to 4087 bytes.
- Passes maximum overlap mesh test (RFC2899) excluding the CPU port, for all packet sizes up to 1518 bytes.
- Queue management operations:
  - Disable scheduling of packets on a port.
  - Disable queuing new packets to a port.
  - Allow a port to be drained without sending out packets.
  - Allow checking if a port is empty or not.
- Input and output mirroring.
- 4,096 entry L2 MAC table, hash based 4-way.
- 16 entry VLAN table.
- 16 entry synthesized CAM to solve hash collisions.
- 4 entries of the synthesized CAM are fully maskable.
- 128 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.
- L2 classification rules. Consists of Source Port, DA MAC, SA MAC, the packets VLAN VID field, the packets VLAN PCP field, the packets VLAN CFI field, Ethernet type.
- 1310720 bits shared packet buffer memory for all ports divided into 2048 cells each of 80 bytes size
- 4 priority queues per egress port.
- Configurable mapping of egress queue from IP TOS, MPLS exp/tc or VLAN PCP bits.
- Deficit Weighted Round Robin Scheduler.
- Egress queue resource limiter with four sets of configurations.
- Configuration interface for accessing configuration and status registers/tables.
- Multicast/Broadcast storm control with separate token buckets for flooding, broadcast and multicast packets.
- Multicast/Broadcast storm control is either packet or byte-based, configurable per egress port.
- LLDP frames can optionally be sent to the CPU.

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 serial to parallel converter accumulates 80 bytes to build a cell, and the cell is sent to ingress processing, if a packet consists of more than 80 bytes then a new cell is built. This is repeated until the *end of packet* signal is asserted.
4. 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.
5. 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.
6. Unless the packet is dropped, the packet is written cell-by-cell into the buffer memory with the I2E header appended.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. The parallel to serial converter divides the cell into MAC-bus sized chunks.
15. Data is transmitted on the output port.
16. Egress port counters are updated.

1.2 Port Numbering Table

Table 1.1 shows the port numbering. Port 4 can serve as a CPU port.

Interface Number	BW	Clock	Clock Frequency	Sync With Core Clock	Port Number & Multicast Table Bit	CPU Port
0	0.1Gbit/s	clk_mac_rx/tx_0	12.50MHz	Yes	0	No
1	0.1Gbit/s	clk_mac_rx/tx_1	12.50MHz	Yes	1	No
2	0.1Gbit/s	clk_mac_rx/tx_2	12.50MHz	Yes	2	No
3	0.1Gbit/s	clk_mac_rx/tx_3	12.50MHz	Yes	3	No
4	0.1Gbit/s	clk_mac_rx/tx_4	12.50MHz	Yes	4	Yes

Table 1.1: Port Numbering Table

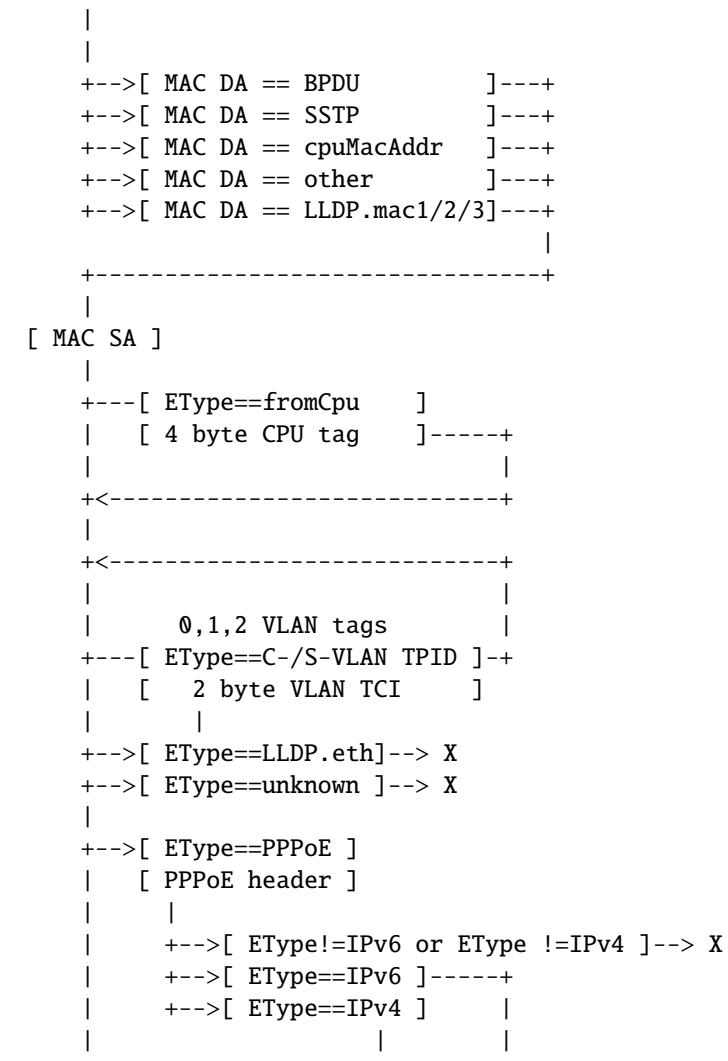
Chapter 2

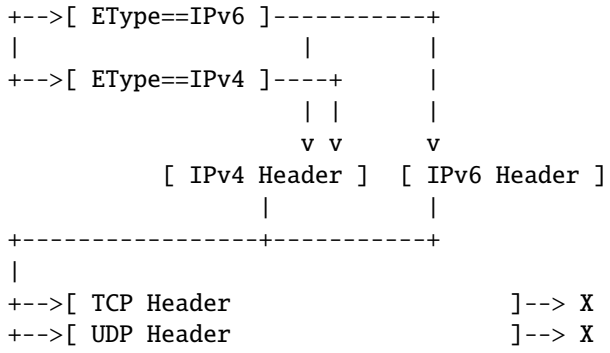
Packet Decoder

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

2.1 Decoding Sequence

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





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. 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.
3. The source MAC address is extracted from the packet.
4. 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) 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**.

(c) From CPU Tags
Packets from CPU will use a Ethernet type value of 0x9988.The From CPU Tag is further described in Chapter 20.

(d) 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. Packet Decoding
   The packet headers are decoded and data extracted. For details see the **Packet Decoding** chapter.
2. 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).
3. 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.
4. 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.
5. 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 and Global ID used in L2 lookups.
6. 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.
7. 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.
8. 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.

9. 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**.
10. 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.
11. Determine Egress Queue Priority
    Egress queues are assigned to packets based on their L2 protocols or classification results. See the **Determine Egress Queue Priority** section.
12. 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.
13. 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.

    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. 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 2 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 0 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 two places (t1-t2) in the core where latency jitter can be introduced. See Figure 4.1 on page 20.

t1 In the SP the ports are visited in a fixed order, thus introducing a jitter the size of the port visitation period.

t2 The egress scheduler visits the ports in a fixed order, introducing a jitter the size of the port visitation period.

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

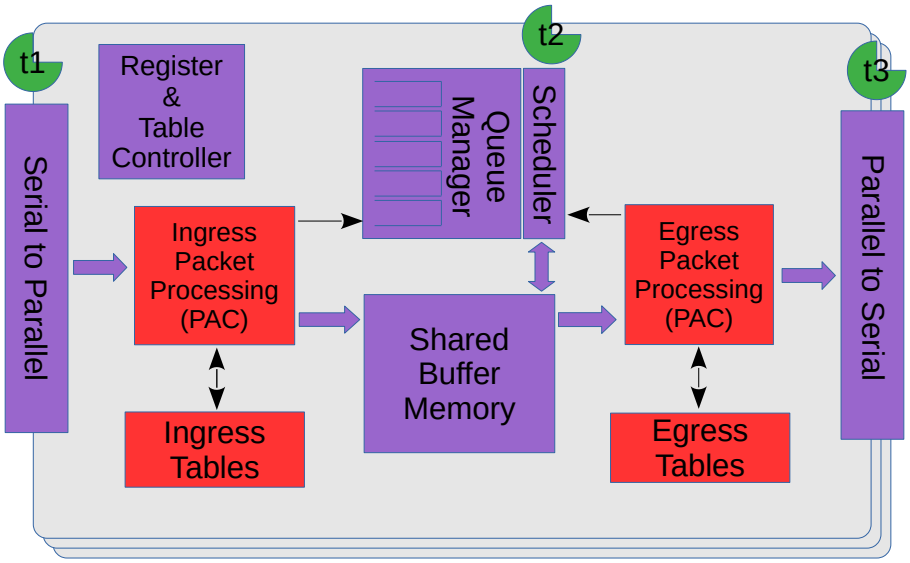


Figure 4.1: Jitter Overview

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 controlled per source port by the **vlanAssignment** in the **Source Port Table**.

5.1.1 Force Ingress VID from L2 ACL

The L2 ACL engine has an option to override the ingress VID assigned above. If the **forceVidValid** field in the **Ingress L2 ACL Result Operation Entries** is set to 1, the corresponding **forceVid** field will be used as the new ingress VID value. The detailed L2 ACL match and action are described in the **L2 Classification** 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.

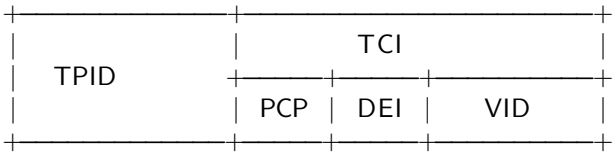
The VLAN membership also determines which multiple spanning tree (MSTP) a packet is part.

The egress queue priority can also be assigned from the VLAN membership (see chapter 14.1).

5.3 VLAN operations

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

For reference here is the 802.1Q VLAN header:



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

The VLAN operations that can be performed are:

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

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

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 VLAN Table Operation

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

5.3.3 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:
```

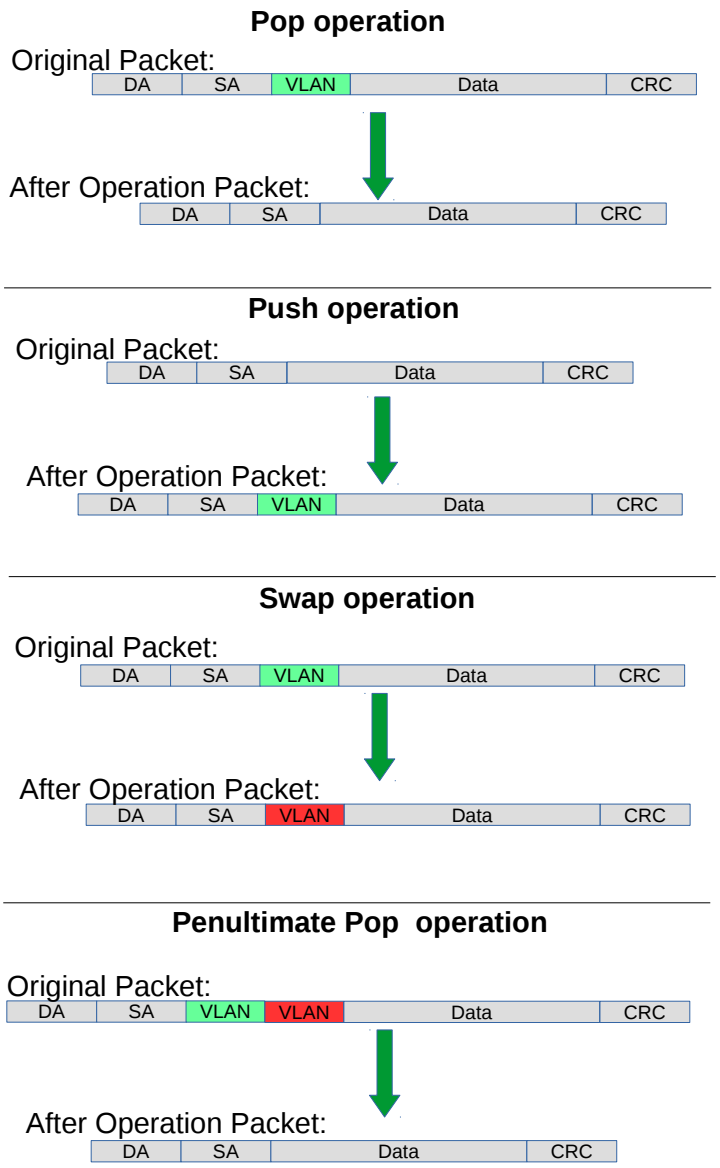


Figure 5.1: VLAN Packet Operations

```
[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]
```

5.3.4 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 hash is calculated with {GID,DA MAC} as key (see MAC Table Hashing).
- The hash is used as index into the **L2 DA Hash Lookup Table**. 4 entries are read out in parallel, each corresponding to a hash bucket.
- The bucket entries are all compared with the {GID,DA MAC} key and if one entry is equal to the key that entry is considered a match.
- The {GID, DA MAC} key is also compared with all the entries in the **L2 Lookup Collision Table** CAM. The CAM is searched starting from entry 0 and the first matching entry is treated as a match. Any following matching entries are ignored.
- Some entries in **L2 Lookup Collision Table** has per-bit masks. These are set up in the **L2 Lookup Collision Table Masks** registers. Using the mask an entry can define with single-bit granularity what shall be included in the comparison. A zero in the mask means that the corresponding bit shall be ignored, while a one means that the bit shall be compared.
- An entry in the **L2 DA Hash Lookup Table** is only compared if the corresponding valid bits are set. The valid bits are located in the **L2 Aging Table** 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.

  A packet can be sent to its source port only when it hits the corresponding unicast entry in the **L2 Destination Table**. Broadcast, flooded, **L2 Multicast Table** hit packet will have its source port excluded from the destination portmask.
- 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 there is a hit then the hit bit in the **L2 Aging Table** is set to one.
- 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.

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.

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.

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 ten packets per egress port.

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

Chapter 8

Classification

8.1 L2 Classification

- L2 ACL engine search data fields are setup in table **Ingress L2 ACL Match Data Entries** and result actions are setup in register **Ingress L2 ACL Result Operation Entries**.
  - The entries in the table are searched starting with entry 0.
  - The statistics counter which can be updated are located in the **Ingress L2 ACL Match Counter**
  - When multiple entries match (are hit) the associated actions from all matching entries will be executed.
  - If two or more entries which match contain the same action then data from the highest (last) entry will be choosen. For example if two entries has the action *force to queue priority* and the lowest hit has a destination queue of 2 while the highest hit has a destination queue of 4 then the packet will have a destination queue of 4.

Chapter 9

VLAN and Packet Type Filtering

This chapter gives an overview of the filtering options available on ingress. 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.

The settings are unique for each port.

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

Chapter 10

Hashing

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

10.1 Hashing Functions

This section describes the hash functions used in this core.

10.1.1 MAC Table Hashing

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

Hash Function for MAC Table

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

When learning random MAC addresses the hash function results in an average utilization of the L2 table of 34% (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: 52 bits hash key
          key[51:48] = GID
          key[47:0] = MAC
      fold count = 6
      returns: 10 bits hash value
  """
  hashval = key & 0b1111111111
  hashval = hashval ^ (key>>10)
  hashval = hashval & 0b1111111111
  hashval = hashval ^ (key>>20)
  hashval = hashval & 0b1111111111
  hashval = hashval ^ (key>>30)
  hashval = hashval & 0b1111111111
  hashval = hashval ^ (key>>40)
  hashval = hashval & 0b1111111111
  hashval = hashval ^ (key>>50)
  hashval = hashval & 0b1111111111
  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 & 0xf) << 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:4
    key = (4)<< 48 | 0x466162bc84dd
    hashval = calc_l2_hash(key) # the hash value is used as index into the L2 DA Hash T
    assert hashval == 21
```

Chapter 11

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.

11.1 L2 Forwarding Information Base (FIB)

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

11.1.1 Tables for MAC DA lookup

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

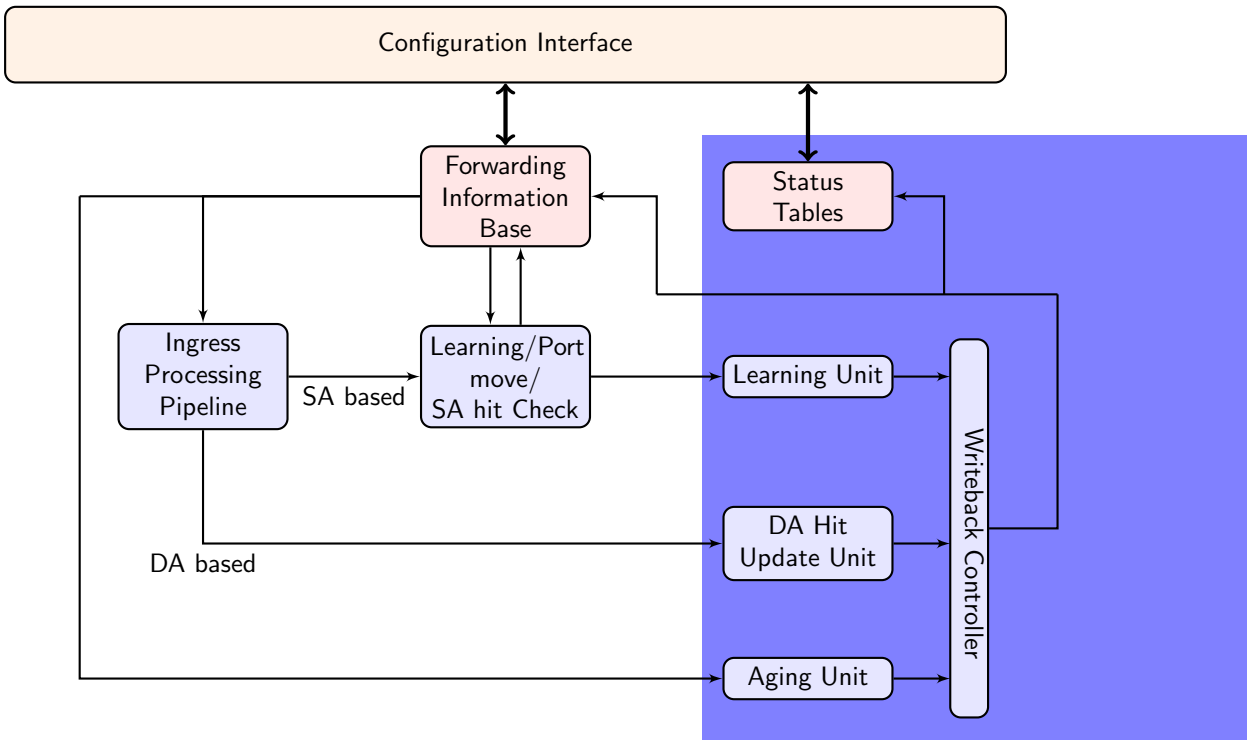


Figure 11.1: Learning and Aging Engine

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 52-bit hash key and the hash function returns a 10-bit hash value.

11.1.2 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 11.1 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.

11.1.3 Hash Collision Accommodation

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

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

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

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

11.2 Hardware Learning and Aging

11.2.1 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.

11.2.2 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.
- 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.
- 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**.

11.2.3 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.

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

[1] Check all reason codes in Table 20.2
[2] See more in Chapter Spanning Tree.
[3] The system ticks are described in Chapter Tick.

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 11.1. 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 11.1: Hardware Aging Operations

11.2.4 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 11.1, 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.

11.3 Software Learning and Aging

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

11.3.1 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 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.

11.3.2 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 12

Spanning Tree

Spanning-Tree Protocol (STP) 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 protocol 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 20

12.1 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**.

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.

12.2 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.

Chapter 13

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

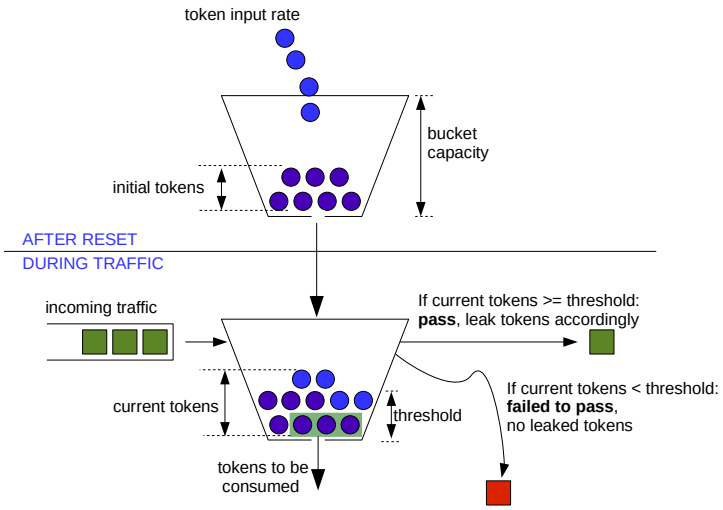


Figure 13.1: General Token Bucket Illustration

Figure 13.1 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

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 five core ticks.

Chapter 14

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 four 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 three. 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.

14.1 Determine Egress Queue

Figure 14.1 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. **forceQueue** in **Ingress L2 ACL Result Operation Entries**
2. **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
- Classification results

The available classification engines are described in the Classification chapter.

Egress queue from packet headers can be mapped from:

[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

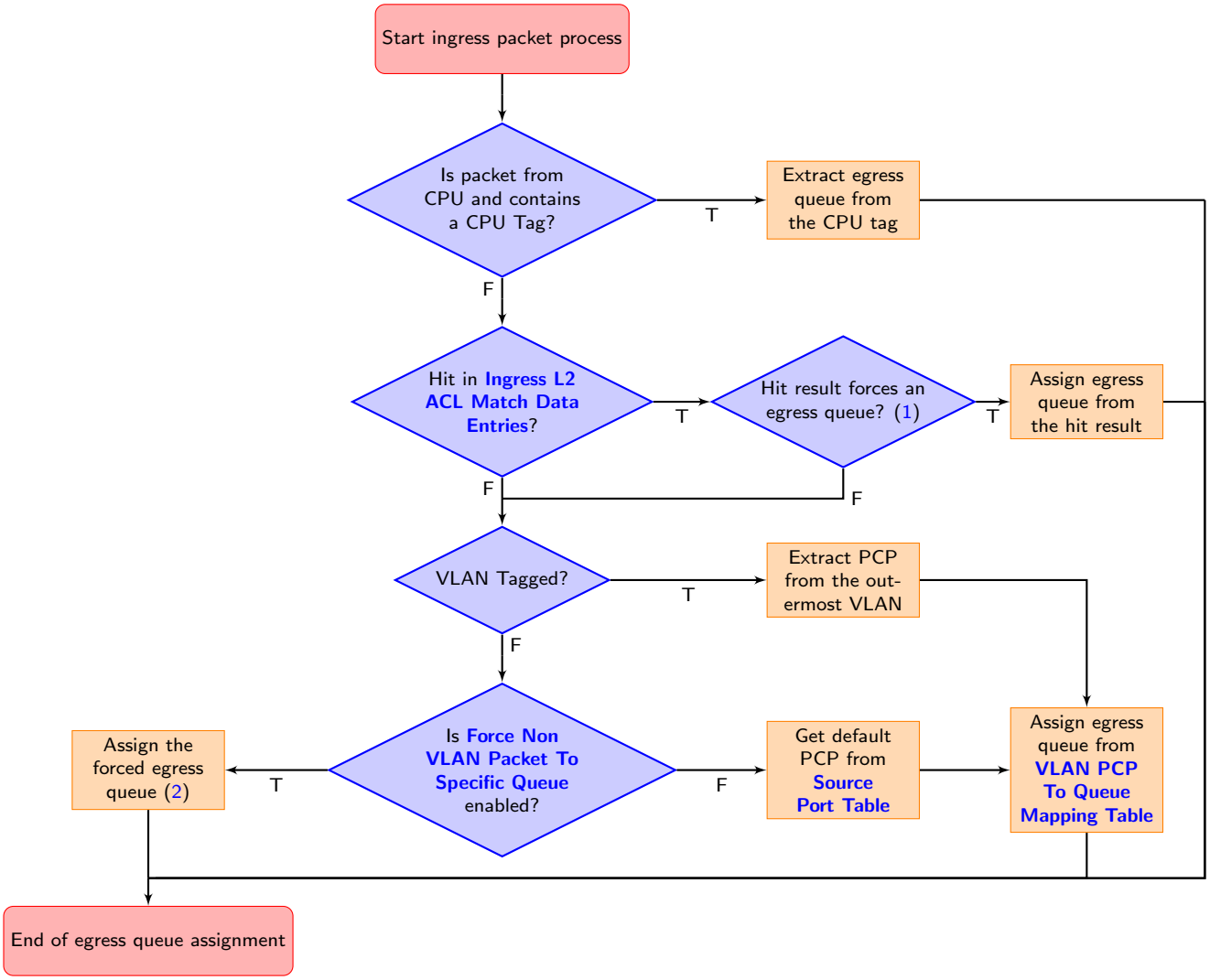


Figure 14.1: Egress Queue Selection Diagram

- 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.

14.2 Priority Mapping

Each queue is mapped to one of four 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 14.4, below for the details.

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

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.

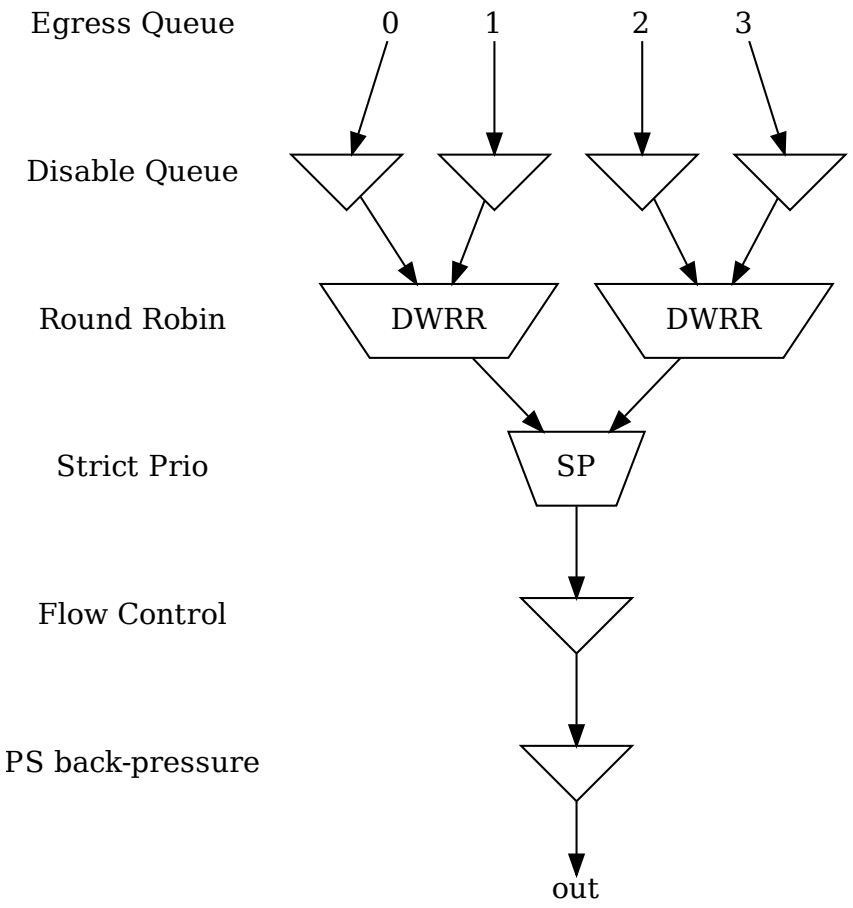


Figure 14.2: Egress Queue Scheduling example. Here using half the priorities, with two queues mapped to each.

14.3 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 14.2. 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:

- 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).

14.4 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.

14.5 Queue Management

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

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

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

- 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.

14.6 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 15

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 (five 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 16

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 13.1). For each egress port there is one token bucket per inspected traffic type. In principle a token bucket controls the traffic rate (packet rate or byte rate) on an egress port. A token bucket operates as follows:

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

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

- L2 Broadcast
- L2 Flooding
- L2 Multicast

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

16.1 Inspected Traffic

- L2 Broadcast: A Packet with DA = ff:ff:ff:ff:ff:ff.
  - Token bucket configurations:
    - **L2 Broadcast Storm Control Enable**
    - **L2 Broadcast Storm Control Bucket Capacity Configuration**
    - **L2 Broadcast Storm Control Bucket Threshold Configuration**

        * **L2 Broadcast Storm Control Rate Configuration**

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

16.2 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:

[1] The system ticks are described in Chapter 15.

```
#!/usr/bin/python

rate    = 0.05

minLen  = 64 # bytes
slice   = 1 # switch slices
ifg     = 20 # bytes
pnb     = 1 # = packet mode
portBW  = 100 # Mbits/s
tickFreqList = [1.25,
                0.125,
                0.0125,
                0.00125,
                0.000125] # 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}
```

Chapter 17

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 2048 cells, each 80 byte wide, available in the buffer memory. A packet is written to the buffer memory with the original packet data plus a 8 byte ingress to egress header, thus a 1600 byte packet will have 1608 bytes and occupy 20 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 73 byte packet will use two cells. ERM traces the buffer memory occupancy and decides if a cell is allowed to be written to the buffer memory.

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

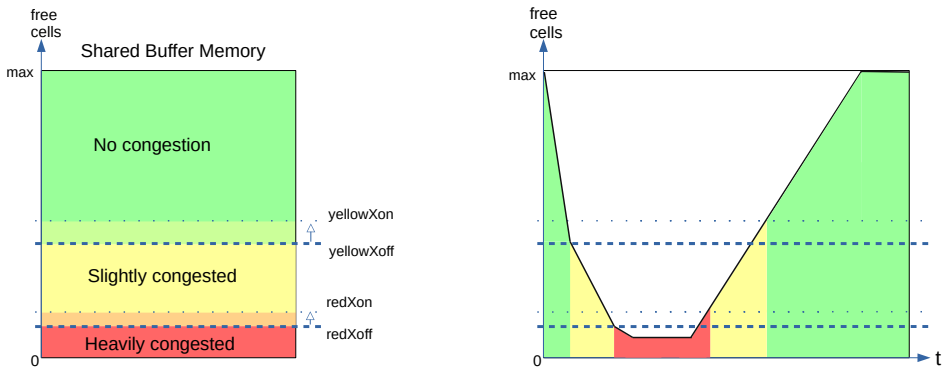


Figure 17.1: Buffer memory congestion zones

17.1 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**.

17.2 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**.

17.3 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.

17.4 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. **yellowXon** $\geq$ **yellowXoff** $\geq$ **maxBufferFree**
   Put the ERM in the yellow zone even when the buffer memory is empty hence keep **yellowLimit** check under an always on state.

17.5 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 18

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.

18.1 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 21.4.

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.

18.2 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.

18.2.1 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.

18.3 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.

18.3.1 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.

18.3.2 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.

18.3.3 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.

18.3.4 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).

18.4 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 18.3.2 above.

18.5 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 18.3.2 above.

18.6 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.

18.7 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.

18.8 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 19

Statistics

Short Name	Register Name
3. macBrokenPkt	MAC RX Broken Packets
4. macRxMin	MAC RX Short 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 Ingress L2 ACL Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop
11. smon	SMON Set 0 Packet Counter SMON Set 1 Packet Counter SMON Set 2 Packet Counter SMON Set 3 Packet Counter SMON Set 0 Byte Counter SMON Set 1 Byte Counter SMON Set 2 Byte Counter SMON Set 3 Byte Counter
11. ippAcl	Ingress L2 ACL Match Counter
11. preEppDrop	Queue Off Drop Egress Spanning Tree Drop MBSC Drop
12. ipmOverflow	IPP PM Drop
13. ippTxPkt	IPP Packet Head Counter IPP Packet Tail Counter
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
21. drain	Drain Port Drop
22. epmOverflow	EPP PM Drop
24. rqOverflow	Re-queue Overflow Drop

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

Table 19.1: 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 19.1 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 19.1 can be found through Table 19.1.

19.1 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.

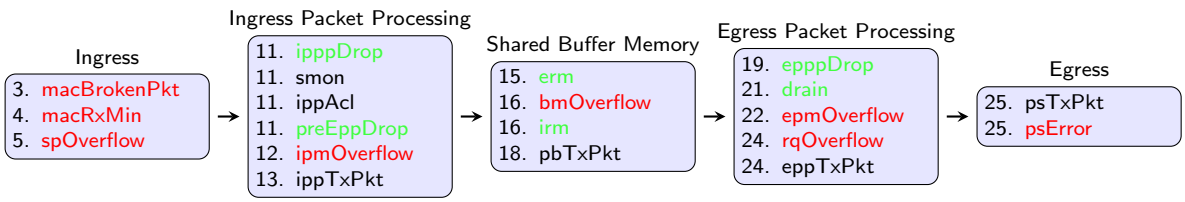


Figure 19.1: Location of Statistics Counters

19.2 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.

19.3 SMON Statistics

There are 4 sets of SMON counters located in the ingress packet processing pipeline, each equipped with one counter per PCP value. The combination of the ingress port number and packet VLAN ID will provide the target SMON set to update through the **SMON Set Search** register. Each SMON set counts both the number of packets and number of bytes as shown in Section Statistics: SMON.

19.4 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.

19.5 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.

19.6 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.

Chapter 20

Packets To And From The CPU

The CPU port (number 4) 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.

20.1 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	[4:0] 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 0.
1	Bits [1:0] specifies which egress queue the packet shall use.

Table 20.1: From CPU tag format

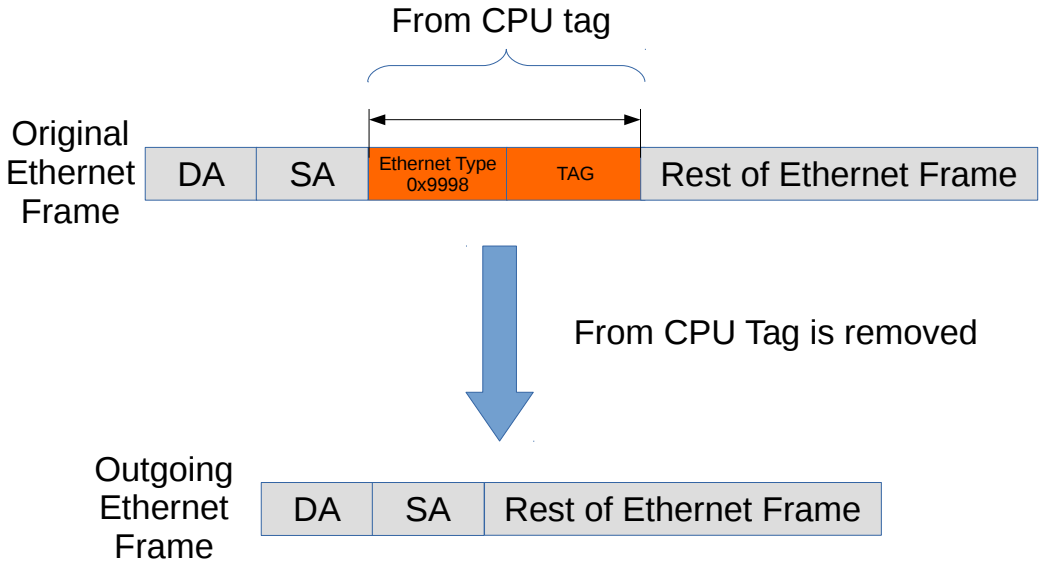


Figure 20.1: Packet from CPU with CPU tag

The header consists of a specific Ethernet Type (39065) followed by a CPU Tag. The CPU tag has a 1 byte(s) destination port mask field[1] and 1 byte egress queue field (encoded as specified in table 20.1). 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 20.1.

The port mask in the CPU Tag field determines which ports the packet shall be sent to. If multiple bits are set in the port mask, the packet is treated as a multicast packet in the resource limiters. The packet will be sent out on all ports with the corresponding bit set.

20.1.1 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.

- 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 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 operations are bypassed.
- No learning operation.
- ACL operations are done.
- ACL statistics are done.
- SMON statistics is done.

20.2 Packets To the CPU

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 20.2). 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 4 in this core), the packets are tagged with a specific Ethernet Type (type 39321). Figure 20.2 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.

[1] The ordering described in 20.1 is the receive/transmit order.

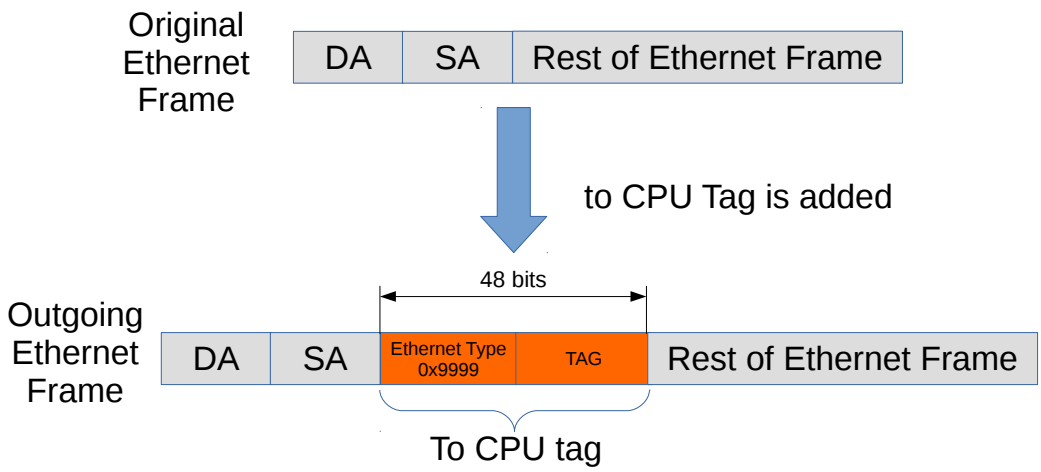


Figure 20.2: Packet to CPU with CPU tag

Byte Number	Contents of Byte
0	Bits [2:0] contains the source port where the packet entered the switch.
1	Reason for packet sent to CPU. See table 20.3.
2	Reserved
3	Reserved

Table 20.2: To CPU tag format

20.2.1 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 first L2 classification sent the packet to the CPU..Index to rule.
20	This is an LLDP frame.

Table 20.3: Reason for packet sent to CPU

The possible reasons are listed in Table 20.3.

1. Hit in the **LLDP Configuration**.
2. 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).

3. Hit in **Ingress L2 ACL Match Data Entries** with a **sendToCpu** action.

Chapter 21

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.

21.1 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 two *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 14.6, while making sure that the packet interface halt is kept low.

Signal Name	Size	In Out	Description
clk	1	In	Core clock. For 0.5 Gbit/s wire-speed throughput use a core clock frequency of 6.25 MHz
rstn	1	In	Global asynchronous reset (active low)
clk_mac_rx_N	1	In	Clock for the RX packet interface for port **N**. Synchronous with the core clock.
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**. Synchronous with the core clock.
rstn_mac_tx_N	1	In	Asynchronous reset (active low) for the TX packet interface for port **N**
clk_mult_0	1	In	A 62.5MHz clock, synchronous with the core clock.
clk_mult_1	1	In	A 12.5MHz 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 21.1: 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 21.1. 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.

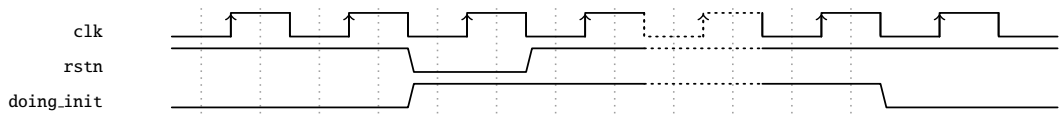


Figure 21.1: Core Initialization

21.1.1 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.

21.2 Packet Interface

There are 5 packet interfaces, or ports for short, each divided into a reception part and a transmission part. The ports are numbered from 0 to 4.

Pin	Size	Direction	Description
idata_sp_**N**	8	In	Packet data.
ivalid_bytes_sp_**N**	1	In	Indicates the number of valid data bytes. For all transactions where *last* is not high, this shall be equal to the data width in bytes.
ifirst_sp_**N**	1	In	Start-of-packet flag.
ilast_sp_**N**	1	In	End-of-packet flag. The *last* field is also used to signal broken packets. For a correctly transmitted packet *last* is asserted for the last data transaction of the packet. If *last* is set high when *valid_bytes* is zero, the packet is marked as broken, and will be dropped by the core.

Table 21.2: Packet RX interface. **N** is the ingress interface number.

Pin	Size	Direction	Description
odata_ps_**N**	8	Out	Packet data.
ovalid_bytes_ps_**N**	1	Out	Indicates the number of valid data bytes. For all transactions where *last* is not high, this is equal to the data width in bytes.
ofirst_ps_**N**	1	Out	Start-of-packet flag.
olast_ps_**N**	1	Out	End-of-packet flag. For a correctly transmitted packet *last* is asserted for the last data transaction of the packet. If *last* is set high when *valid_bytes* is zero, the packet shall be dropped or terminated with an error by the MAC.
tx_halt_ps_**N**	1	In	Interrupt the data transmission from egress port **N**.

Table 21.3: Packet TX interface. **N** is the egress interface number.

Each direction of a packet interface consists of *first*, *last*, *valid_bytes*, and *data* fields. The transmit direction has an additional *halt* signal to allow the receiving end to moderate the data rate transmitted from the core.

Packet data is presented in order, i.e. the most recent byte is the, so far, highest numbered byte in the packet. The first valid byte on the bus is byte 0, and all bytes are valid up to the number indicated in *valid_bytes*. Unless the *last* flag is set all bytes or no bytes must be valid.

Sending and Receiving packets

Data transmission, either to or from the core, begins with a transaction where the *first* field is high and the *valid_bytes* field is non-zero, and ends with a data transmission where the *last* field is high. Idle transactions—where valid_bytes, *first* and *last* are all zero—are allowed at any time, but unless halted there will be no idle transactions on the transmission interfaces other than between packets.

By default, the core has a short packet size limit of 60 bytes. All shorter packets will be dropped. This assumes that the receiving MAC removes the FCS before sending the packet to the core.

Jumbo packets

The maximum packet length that this core can cope with is 4087 bytes. If this length is exceeded either on the ingress or the egress it may corrupt the internal counters.

It should be noted that it is not guaranteed that a packet of that length will always be able to pass through the switch, even if the destination queue is not congested. Depending on the Egress Resource Management settings, and/or the congestion status of other ports, there may not be enough free cells in the packet buffer to store such a large packet. But the switch core will, when properly configured and reasonably uncongested, be able to switch 4087-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 2048-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 *halt* interface to prevent overflows.

Broken packets

A packet ending with *last* set high and *valid_bytes* set to zero is considered a broken packet. Broken packets received by the core will never be output on the egress ports, but will be dropped at the earliest convenience. So any broken packets output from the switch are packet that were somehow corrupted in the core. There are no benign cases where this happens. Depending on the packet length a broken packet input to the core will be dropped either before or after ingress packet processing. Broken packets larger than a cell will pass through the packet processing pipeline and then been dropped, while packets shorter than a cell will be filtered out before the packet processing pipeline.

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

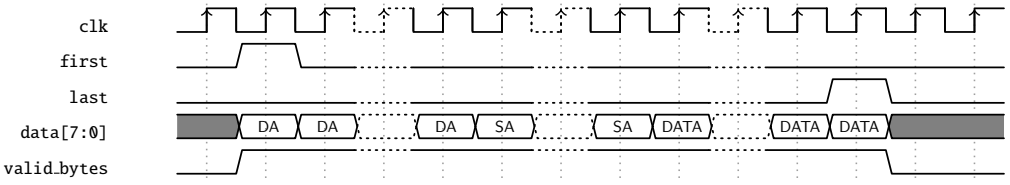


Figure 21.2: Sending and Receiving packets without error (8-bit)

Halts

Data transmission from the transmit interface of the core can be interrupted individually per egress port using the *halt* signals. A high halt signal on the positive edge of mac clock, will cause the transmission to be idle for the corresponding egress port on the same positive edge. Data transmission will resume on the next positive edge of mac clock when halt is again low.

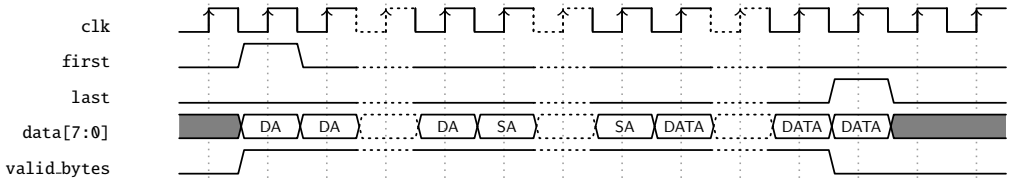


Figure 21.3: Sending and Receiving packets with error (8-bit)

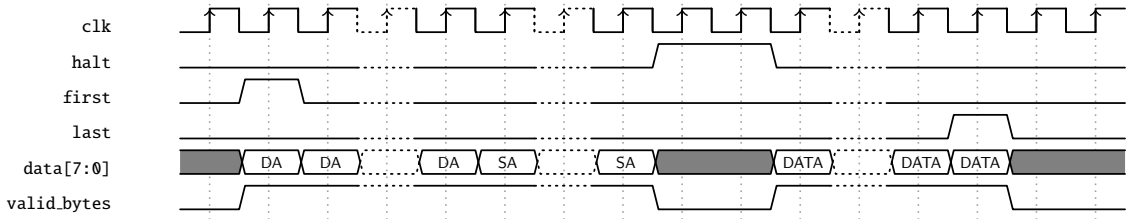


Figure 21.4: Halted transmit packet (8-bit)

Byte Order

We define the packet byte order by the first transmitted/received byte on the wire labeled byte 0, as in IEEE 802.3. On a packet interface wider than 8 bits the packets byte 0 is placed on the bits data[7:0] followed by byte 1 on bits data[15:8] and so on.

The *valid_bytes* indicates how many of the bytes of the data field that holds valid packet data. From the start of a packet this must always be all bytes on the bus up till the last transfer. At the end of the packet on the last bus transfer the *valid_bytes* can indicate less than the full bus width. In this case the byte order is still the same as previous transfers. For example when *valid_bytes* is 1 the last byte of the packet is placed on bits [7:0] and with *valid_bytes* of 2 the last byte of the packet is placed on bits [15:8] and the second to last is on bits [7:0].

21.3 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 21.4 below.

A user guide for the configuration interface follows in Chapter 22.

Pin	Size	Direction	Description
request_data_N	32	In	The request data
request_address_N	16	In	The request address
request_re_N	1	In	Read enable for the transaction. Active high
request_we_N	1	In	Write enable for the transaction. Active high
request_type_N	2	In	The request type 0 Default 1 Accumulator 2 *Reserved* 3 *Reserved*
request_id_N	1	In	The request identifier.
reply_status_N	2	Out	The reply status 0 Idle (NONE) 1 Read OK (ROK) 2 Write OK (WOK) 3 Fail (FAIL)
reply_id_N	1	Out	The reply identifier
reply_data_N	32	Out	The reply data.

Table 21.4: The signals for an instance of the configuration interface

21.4 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.

21.4.1 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.

21.4.2 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 21.5: ThePFC status and External Pause interfaces, where **N** is the packet interface number

21.5 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 21.7. Both these signals are pipelined.

Pin	Direction	Size	Description
debug_read_select	In	9	Selects the signal to monitor. See Table 21.7.
debug_read_data	In	32	The debug output data.

Table 21.6: The Debug Read interface

id	instance	signal
0	pa_top.switch.mactop	constant-0
1	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
2	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
3	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
4	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
5	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
6	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
7	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
8	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
9	—"—	rx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
10	—"—	tx_pkt_bus {8'data, 1'valid_bytes, 1'last, 1'first}
11	—"—	constant-11
12	pa_top.switch.ipp0	constant-12
13	—"—	ipp_ipkt_bus {20'data, 7'valid_bytes, 3'id, 1'last, 1'first}
14	—"—	ipp_opkt_bus {20'data, 7'valid_bytes, 3'id, 1'last, 1'first}
15	—"—	pass_da_0
16	—"—	pass_da_1
17	—"—	dut_iIpp_iDropper_dbg_drop
18	—"—	dut_iIpp_iDropper_dbg_ifirst
19	—"—	dut_iIpp_iDropper_dbg_ilast
20	—"—	pass_sa_0
21	—"—	pass_sa_1
22	—"—	constant-22
23	pa_top.switch.ipp0.pm	constant-23
24	—"—	pm_fifo_overflow
25	—"—	dut_dbg_fifo_full
26	—"—	halt_from_pm
27	—"—	dut_iFifoa_debug_in
28	—"—	dut_iFifoa_debug_out
29	—"—	constant-29
30	pa_top.switch.sp0	constant-30
31	—"—	dut_iSpbridge_iBridge_4_iSyncFifo_iF_iFifos_zFcnt_pop_empty
32	—"—	dut_iSpbridge_iBridge_4_iSyncFifo_iF_iFifos_zFcnt_push_full
33	—"—	dut_iSpbridge_iBridge_3_iSyncFifo_iF_iFifos_zFcnt_pop_empty
34	—"—	dut_iSpbridge_iBridge_3_iSyncFifo_iF_iFifos_zFcnt_push_full
35	—"—	dut_iSpbridge_iBridge_2_iSyncFifo_iF_iFifos_zFcnt_pop_empty
36	—"—	dut_iSpbridge_iBridge_2_iSyncFifo_iF_iFifos_zFcnt_push_full
37	—"—	dut_iSpbridge_iBridge_1_iSyncFifo_iF_iFifos_zFcnt_pop_empty
38	—"—	dut_iSpbridge_iBridge_1_iSyncFifo_iF_iFifos_zFcnt_push_full
39	—"—	dut_iSpbridge_iBridge_0_iSyncFifo_iF_iFifos_zFcnt_pop_empty
40	—"—	dut_iSpbridge_iBridge_0_iSyncFifo_iF_iFifos_zFcnt_push_full
41	—"—	dut_iSpbridge_assert_reset_sp_bridge
42	—"—	dut_iSpbridge_assert_reset_sp_bridge
43	—"—	dut_iSpbridge_assert_reset_sp_bridge
44	—"—	dut_iSpbridge_assert_reset_sp_bridge
45	—"—	dut_iSpbridge_assert_reset_sp_bridge
46	—"—	constant-46
47	pa_top.switch.pb0	constant-47
48	—"—	dut_iPbu_debug_refc_inc
49	—"—	dut_iPbu_debug_port_sch
50	—"—	dut_iPbu_dmux_wrr
51	—"—	dut_iPbu_debug_qenext
52	—"—	dut_iPbu_assert_qediff
53	—"—	dut_iPbu_assert_reque_sp
54	—"—	Mask of currently receiving packets that have been broken due to BM full
55	—"—	dut_iPbu_follow_pfc_accept
56	—"—	dut_iPbu_iAssertpacket_0_assert_out
57	—"—	pa.top.switch.pb0.iAssertpacket0 {7'valid_bytes, 3'port, 1'last, 1'first}
58	—"—	dut_iPbu_zPassdbgqeread_0_o
59	—"—	dut_iPbu_iRequeue_iReFifo_4_iF_iFifos_zFcnt_pop_empty
60	—"—	dut_iPbu_iRequeue_iReFifo_4_iF_iFifos_zFcnt_push_full
61	—"—	dut_iPbu_iRequeue_iReFifo_3_iF_iFifos_zFcnt_pop_empty
62	—"—	dut_iPbu_iRequeue_iReFifo_3_iF_iFifos_zFcnt_push_full
63	—"—	dut_iPbu_iRequeue_iReFifo_2_iF_iFifos_zFcnt_pop_empty
64	—"—	dut_iPbu_iRequeue_iReFifo_2_iF_iFifos_zFcnt_push_full
65	—"—	dut_iPbu_iRequeue_iReFifo_1_iF_iFifos_zFcnt_pop_empty
66	—"—	dut_iPbu_iRequeue_iReFifo_1_iF_iFifos_zFcnt_push_full
67	—"—	dut_iPbu_iRequeue_iReFifo_0_iF_iFifos_zFcnt_pop_empty
68	—"—	dut_iPbu_iRequeue_iReFifo_0_iF_iFifos_zFcnt_push_full
69	—"—	dut_iPbu_iRefc_refc_mem_debug

id	instance	signal
70	—"—	dut_iPbu_zPassqesp_zPasslist_0_o
71	—"—	Filter mask for packets dropped by ERM
72	—"—	dut_iPbu_debug_pb_drop
73	—"—	constant-73
74	pa_top.switch.pb0.erm.dut_iEql	constant-74
75	—"—	red_zone
76	—"—	constant-76
77	pa_top.switch.pb0.pfc	constant-77
78	—"—	dut_debug_pause
79	—"—	constant-79
80	pa_top.switch.pb0.qe0	constant-80
81	—"—	dut_assert_dfifo
82	—"—	dut_assert_firstflag
83	—"—	dut_assert_reset_next
84	—"—	dut_drop_cnt
85	—"—	dut_send_cnt
86	—"—	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
87	—"—	dut_iDfifo_iF_iFifos_zFcnt_push_full
88	—"—	dut_ipkt_fifo_4_debug_in
89	—"—	dut_ipkt_fifo_4_debug_out
90	—"—	dut_ipkt_fifo_3_debug_in
91	—"—	dut_ipkt_fifo_3_debug_out
92	—"—	dut_ipkt_fifo_2_debug_in
93	—"—	dut_ipkt_fifo_2_debug_out
94	—"—	dut_ipkt_fifo_1_debug_in
95	—"—	dut_ipkt_fifo_1_debug_out
96	—"—	dut_ipkt_fifo_0_debug_in
97	—"—	dut_ipkt_fifo_0_debug_out
98	—"—	dut_pfifo_level
99	—"—	dut_pfifo_level
100	—"—	dut_pfifo_level
101	—"—	dut_pfifo_level
102	—"—	dut_pfifo_level
103	—"—	constant-103
104	pa_top.switch.pb0.wrr	constant-104
105	—"—	dut_debug_below
106	—"—	dut_zPassdebugbvalpipe_zPasslist_3_o
107	—"—	dut_zPassdebugbvalpipe_zPasslist_2_o
108	—"—	dut_zPassdebugbvalpipe_zPasslist_1_o
109	—"—	dut_zPassdebugbvalpipe_zPasslist_0_o
110	—"—	dut_reg_bval
111	—"—	dut_reg_bval
112	—"—	dut_reg_bval
113	—"—	dut_reg_bval
114	—"—	dut_reg_bval
115	—"—	dut_reg_bval
116	—"—	dut_reg_bval
117	—"—	dut_reg_bval
118	—"—	dut_reg_bval
119	—"—	dut_reg_bval
120	—"—	dut_reg_bval
121	—"—	dut_reg_bval
122	—"—	dut_reg_bval
123	—"—	dut_reg_bval
124	—"—	dut_reg_bval
125	—"—	dut_reg_bval
126	—"—	dut_reg_bval
127	—"—	dut_reg_bval
128	—"—	dut_reg_bval
129	—"—	dut_reg_bval
130	—"—	dut_reg_rank
131	—"—	dut_reg_rank
132	—"—	dut_reg_rank
133	—"—	dut_reg_rank
134	—"—	dut_reg_rank
135	—"—	constant-135
136	pa_top.switch.bm0	constant-136
137	—"—	dut_bm_ifree_debug_free
138	—"—	constant-138
139	pa_top.switch.ps0	constant-139
140	—"—	halt_from_ps
141	—"—	dut_iPs2_zPsAssert_item
142	—"—	dut_iPs2_iBridge_4_iSyncFifo_iF_iFifos_zFcnt_pop_empty
143	—"—	dut_iPs2_iBridge_4_iSyncFifo_iF_iFifos_zFcnt_push_full
144	—"—	dut_iPs2_iBridge_3_assert_reset
145	—"—	dut_iPs2_iBridge_3_iSyncFifo_iF_iFifos_zFcnt_pop_empty
146	—"—	dut_iPs2_iBridge_3_iSyncFifo_iF_iFifos_zFcnt_push_full
147	—"—	dut_iPs2_iBridge_2_assert_reset
148	—"—	dut_iPs2_iBridge_2_iSyncFifo_iF_iFifos_zFcnt_pop_empty
149	—"—	dut_iPs2_iBridge_2_iSyncFifo_iF_iFifos_zFcnt_push_full

id	instance	signal
150	—"—	dut_iPs2_iBridge_1_assert_reset
151	—"—	dut_iPs2_iBridge_1_iSyncFifo_iF_iFifos_zFcnt_pop_empty
152	—"—	dut_iPs2_iBridge_1_iSyncFifo_iF_iFifos_zFcnt_push_full
153	—"—	dut_iPs2_iBridge_0_assert_reset
154	—"—	dut_iPs2_iBridge_0_iSyncFifo_iF_iFifos_zFcnt_pop_empty
155	—"—	dut_iPs2_iBridge_0_iSyncFifo_iF_iFifos_zFcnt_push_full
156	—"—	dut_iPs2_iSplitter_0_assert_noend
157	—"—	dut_iPs2_iSplitter_0_assert_ptr
158	—"—	dut_iPs2_iSplitter_0_used_mem
159	—"—	dut_iPs2_iSplitter_0_used_mem
160	—"—	dut_iPs2_iSplitter_0_used_mem
161	—"—	dut_iPs2_iSplitter_0_used_mem
162	—"—	dut_iPs2_iSplitter_0_used_mem
163	—"—	constant-163
164	pa_top.switch.epp0	constant-164
165	—"—	dut_iEpp_assert_ipkt
166	—"—	dut_iEpp_assert_opkt
167	—"—	epp_ipkt_bus {20'data, 7'valid_bytes, 3'id, 1'last, 1'first}
168	—"—	epp_opkt_bus {20'data, 7'valid_bytes, 3'id, 1'last, 1'first}
169	—"—	dut_iEpp_iDropper_da_0
170	—"—	dut_iEpp_iDropper_da_1
171	—"—	dut_iEpp_iDropper_dbg_drop
172	—"—	dut_iEpp_iDropper_dbg_ifirst
173	—"—	dut_iEpp_iDropper_dbg_ilast
174	—"—	dut_iEpp_iDropper_sa_0
175	—"—	dut_iEpp_iDropper_sa_1
176	—"—	pa.top.switch.epp0.iPacketassertpm {7'valid_bytes, 3'port, 1'last, 1'first}
177	—"—	pa.top.switch.epp0.iPacketassertin {7'valid_bytes, 3'port, 1'last, 1'first}
178	—"—	constant-178
179	pa_top.switch.epp0.pm	constant-179
180	—"—	pm_fifo_overflow
181	—"—	dut_dbg_fifo_full
182	—"—	halt_from_pm
183	—"—	dut_iFifoa_debug_in
184	—"—	dut_iFifoa_debug_out
185	—"—	constant-185
186	pa_top.switch.ingress_common	constant-186
187	—"—	dut_iLearnage_iHitUpdate_iFifo_0_iF_iFifos_zFcnt_pop_empty
188	—"—	dut_iLearnage_iHitUpdate_iFifo_0_iF_iFifos_zFcnt_push_full
189	—"—	dut_iMbsc_iFlood_reg_stat
190	—"—	dut_iMbsc_iMc_reg_stat
191	—"—	dut_iMbsc_iBc_reg_stat
192	—"—	constant-192
193	pa_top.switch.interface_common	constant-193
194	—"—	dut_zFaii_iMf_zMf_1_item
195	—"—	dut_zFaip_iMf_zMf_1_item
196	—"—	dut_zFaie_iMf_zMf_1_item
197	—"—	dut_zFaiq_iMf_zMf_1_item
198	—"—	dut_zFais_iMf_zMf_1_item
199	—"—	constant-199

Table 21.7: Debug Selection Map

21.6 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 21.8: The Debug Write interface

Chapter 22

Configuration Interface

The configuration interface is used for monitoring the core and for configuration of internal registers and tables. The pins are described in Table 21.4 on page 80.

Even if you are just doing a quick and dirty bus implementation, please read the short implementation note, Section 22.5, at the end of this chapter.

22.1 Request Types

Requests can be of either read or write type. Asserting the read- and write-enables concurrently is not supported. Reads and writes can be of `DEFAULT` or `ACCUMULATOR` type. Although registers and tables where the data width is less than or equal to the configuration interface data width support only the `DEFAULT` type. The purpose of the `ACCUMULATOR` request type is to access data that is wider than the bus without the risk of data inconsistency.

Requests for registers which exceed the bus width are discussed in more detail in Section 22.4 below.

22.2 Reply Types

A write access will produce either a `WOK`, reply indicating that the write was successful, or a `FAIL` reply, indicating that the write failed. A read access will similarly produce either a `ROK` or a `FAIL` response. When the response is `ROK`, the read data is available on the data pins. All valid requests will result in a reply, but no reply, not even a fail, will be produced for an access to an unmapped address.

If the core clock frequency is set below the recommended frequency and the core is running at full capacity then a request to a memory may take an infinite time to complete. In practice the recommended frequency is set so that there are sufficient cycles for firmware accesses even under full load.

Figure 22.1 shows two write accesses to the same register taking different time to complete.

22.3 Transaction Identifier

This core has only a single transaction ID, so the request_id shall be tied low. Normally you should always wait for a transaction to return a reply before issuing a new transaction, because issuing concurrent transactions can cause the loss of replies. But for writes to registers it is relatively safe to re-use the same ID for back-to-back accesses. The replies may be inconsistent, but since registers (unlike tables) will never block an access, the writes will succeed.

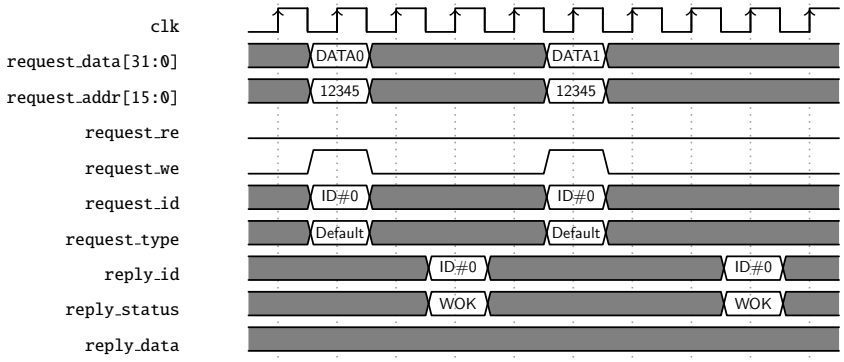


Figure 22.1: Completion time, even to the same register, may vary

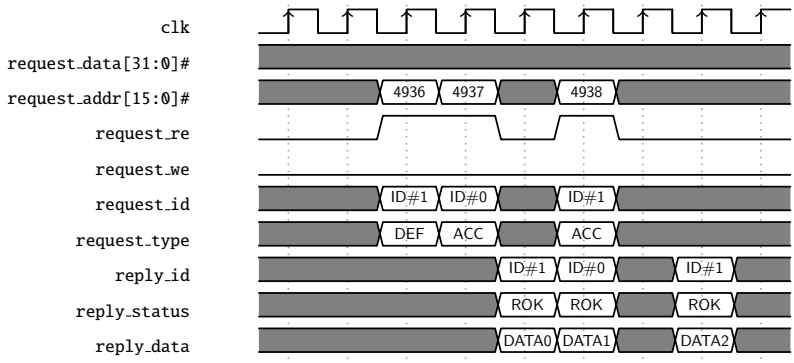


Figure 22.2: Read from a wide register

22.4 Atomic Wide Access - Accumulator Access

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 tough the bus width is narrower than the data. Accesses to the accumulator are done using the same address that would be used to directly access the data it shadows, the only difference being that the request type is set to `ACCUMULATOR`.

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 request_data, and fill it out with the data in the accumulator. Thus writing data wider than the bus cannot be done without taking the accumulator into account.

If only a part of the data is to be written, the most efficient approach is to do a default read (loading the accumulator) followed by a default write. The accesses should be issued as close as possible, to minimize the risk of the core updating the memory data while the accumulator is loaded. 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.

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 table will result in index 1 starting at address 4, with address 3 unused and address 2 only containing a single valid bit.

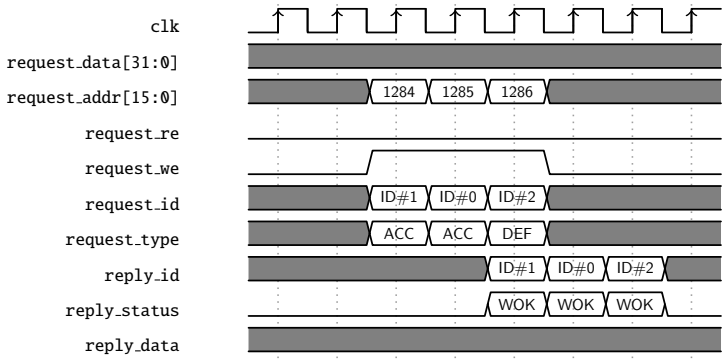


Figure 22.3: Write to a wide register

22.5 Implementation note

Mapping the pins of the configuration bus to your bus of choice is usually a non-trivial task. We recommend two things:

- Map the request_type pin to an unused address bit
- Do all requests within wide registers or tables in consecutive order.

The first makes it easy to do default and accumulator accesses using any bus protocol. The second makes it easier to later optimize your access performance should the need arise.

So, a read from a wide register would start with a default read on the lowest address of the register, and then continue upward using accumulator accesses. A write would start with an accumulator write to the lowest address, continue upward with accumulator writes until the last address where a default write finishes the transaction. The software API implementation provided with the switch supports both of these thereby hiding it completely for the software that use the API.

Note that for this to work your bus needs to be set up to guarantee the order of accesses in the bridge to the configuration bus.

Chapter 23

Implementation

23.1 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.

23.1.1 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 23.1: 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 23.2: The settings for input and output flops for the floorplan blocks

The pipeline settings used when generating this core are shown in Table 23.1, and the input/output flops are listed in Table 23.2[1].

23.1.2 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.

23.1.3 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.

23.2 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 23.2. For memories with the *write_through* attribute set, simultaneous reading and writing the of same address is expected to yield the write data as read result. For memories with *write_through* set to 0 simultaneous reading and writing to the same address shall not occur.

type	width	depth	write through	write mask	input flops	output flops
dp	3	1024	1	None	0	0
dp	3	1024	0	None	0	0
dp	64	200	1	None	0	0
dp	1	1024	1	None	0	0
dp	1	1024	0	None	0	0
dp	52	1024	1	None	0	0
dp	52	1024	0	None	0	0
dp	9	4112	1	None	0	0
dp	9	4112	0	None	0	0
dp	72	220	0	None	0	0
dp	64	50	0	None	0	0
dp	3	2048	1	None	0	0
dp	7	2048	0	None	0	0
dp	16	2048	0	None	0	0
dp	3	2048	0	None	0	0
dp	12	2048	0	None	0	0
dp	22	2048	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	640	2048	0	None	0	0
dp	11	2048	1	None	0	0
dp	9	30	0	None	0	0
dp	66	320	0	None	0	0
dp	64	350	0	None	0	0

Table 23.3: The memory macros needed for this core. dp=two ports, one read and one write, running on the same clock.

Only memories with 256 bits or more have been generated as a memory instance. Smaller 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].

23.3 Dual ported memories

All memories are dual ported. Unless the frequency would be prohibitively high, the best approach is to implement the memories using single ported memory macros clocked at twice the speed. 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.

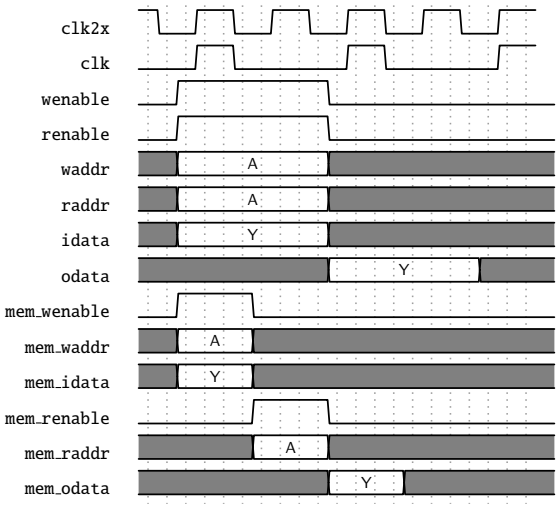


Figure 23.1: Timing diagram for a single ported memory used in the dual ported memory wrapper. In this case a concurrent read and write to the same address of a memory wrapper set for one cycle latency and with the write through attribute set.

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.

23.4 Memory timing

All memories in the design can be selected to have either:

- One cycle latency

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

- 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 23.2.

23.5 Lint set up

For spyglass linting the following settings are assumed:

- set_parameter ignore_local_variables yes
- set_parameter handle_zero_padding "W362"

23.5.1 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 24

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 22, and the pins for the configuration interfaces are described in Section 21.3.

24.1 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 24.1. The total address space used by this core is 21560 addresses.

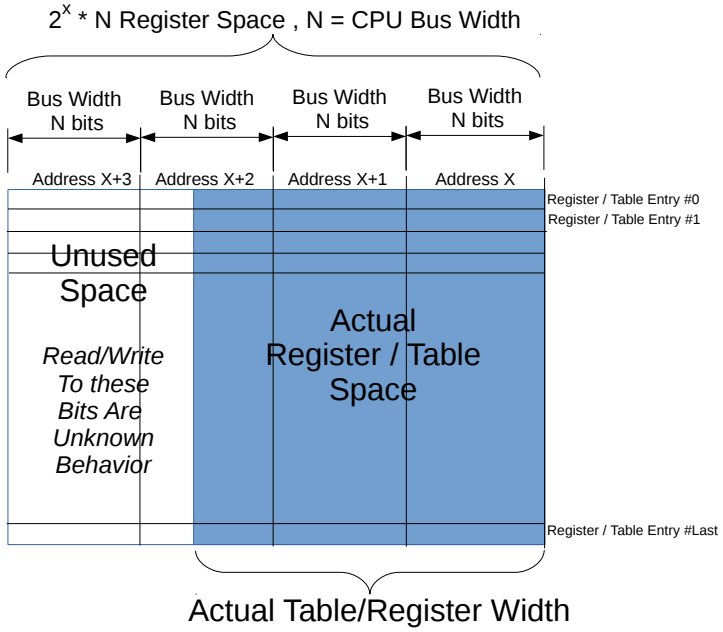


Figure 24.1: Address space usage by tables

24.2 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*.

24.3 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 Scratch
rx_length_drop	MAC RX Broken Packets[0..4] MAC RX Short Packet Drop[0..4]
l2_broadcast_storm_control_rate_settings	L2 Broadcast Storm Control Rate Configuration
l2_broadcast_storm_control_bucket_settings	L2 Broadcast Storm Control Bucket Capacity Configuration L2 Broadcast Storm Control Bucket Threshold Configuration
l2_broadcast_storm_control_misc	L2 Broadcast Storm Control Enable
l2_multicast_storm_control_rate_settings	L2 Multicast Storm Control Rate Configuration
l2_multicast_storm_control_bucket_settings	L2 Multicast Storm Control Bucket Capacity Configuration L2 Multicast Storm Control Bucket Threshold Configuration
l2_multicast_storm_control_misc	L2 Multicast Storm Control Enable
l2_flooding_storm_control_rate_settings	L2 Flooding Storm Control Rate Configuration
l2_flooding_storm_control_bucket_settings	L2 Flooding Storm Control Bucket Capacity Configuration L2 Flooding Storm Control Bucket Threshold Configuration
l2_flooding_storm_control_misc	L2 Flooding Storm Control Enable
le_ae_status	Learning Conflict Learning Overflow
le_ae_control	Learning And Aging Enable Hardware Learning Configuration Time to Age
age_cam_register_bank	L2 Aging Collision Table[0..15]
mac_cnt_register_bank	Hardware Learning Counter[0..4]
L2 Aging Table	L2 Aging Table
count_sp_ss0	SP Overflow Drop
count_broken_pkt_ss0	IPP PM 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 Ingress L2 ACL Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop
count_opkt_pa top switch ipp0 conf	IPP Packet Head Counter IPP Packet Tail Counter
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
ipp_register_bank_ss0	Source Port Table L2 Lookup Collision Table Masks L2 Lookup Collision Table VLAN Table

Bank Name	Connected Registers or Tables
	Send to CPU Ingress Ethernet Type for VLAN tag Force Non VLAN Packet To Specific Queue Egress Spanning Tree State Forward From CPU Port Move Options L2 Multicast Handling Debug srcPort Debug dstPortmask Enable Enqueue To Ports And Queues L2 Multicast Table L2 Aging Collision Shadow Table VLAN PCP To Queue Mapping Table Ingress L2 ACL Result Operation Entries Ingress Port Packet Type Filter SMON Set Search LLDP Configuration Ingress L2 ACL Match Data Entries
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]
count_bytes ipp0_smonStatisticsBlock	SMON Set 0 Byte Counter[0..7] SMON Set 1 Byte Counter[0..7] SMON Set 2 Byte Counter[0..7] SMON Set 3 Byte Counter[0..7]
count_ipp0_aclL2StatisticsBlock	Ingress L2 ACL Match Counter[0..15]
count_ipp0_egressDropStatisticsBlock	Queue Off Drop[0..4] Egress Spanning Tree Drop[0..4] MBSC Drop[0..4]
bk_erm_ss0	ERM Yellow Configuration Resource Limiter Set[0..3] ERM Red Configuration Egress Resource Manager Pointer[0..4]
count_erm_ss0	Egress Resource Manager Drop[0..4]
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..4]
count_drop_pa top switch pb0 iRequeue	Re-queue Overflow Drop
pfc_regbank_port_rsv_size_ss0	Port Reserved[0..4]
pfc_regbank_cmn_misc_ss0	Port Used[0..4] FFA Used
pfc_regbank_pause_settings1_ss0	Port Pause Settings[0..4]
pfc_regbank_taildrop_settings0_ss0	Port Tail-Drop Settings[0..4]
pfc_regbank_misc_ss0	Xon FFA Threshold Xoff FFA Threshold Tail-Drop FFA Threshold Port Xon FFA Threshold[0..4] Port Xoff FFA Threshold[0..4] Port Tail-Drop FFA Threshold[0..4]
qe_register_bank_ss0_sp0	Egress Port Depth[0..4] Egress Queue Depth[0..19]
pb_r_register_bank_ss0	Minimum Buffer Free

Bank Name	Connected Registers or Tables
disable_queue_output_register_bank_ss0	Output Disable[0..4]
dwrr_bucket_capacity_settings_ss0	DWRR Bucket Capacity Configuration[0..4]
dwrr_bucket_misc_settings_ss0	DWRR Bucket Misc Configuration[0..4]
dwrr_weight_settings_ss0	DWRR Weight Configuration[0..19]
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..4]
count_pa top switch epp0 conf	Unknown Egress Drop[0..4] Egress Port Disabled Drop[0..4] EPP PM Drop
count_opkt_pa top switch epp0 conf	EPP Packet Head Counter EPP Packet Tail Counter
epp_register_bank_ss0	Output Mirroring Table Egress Ethernet Type for VLAN tag Disable CPU tag on CPU Port
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

24.4 Registers and Tables in Alphabetical Order

Name	Address Range
Buffer Free	1
Buffer Overflow Drop	21334
Core Tick Configuration	2
Core Tick Select	3
Core Version	0
DWRR Bucket Capacity Configuration	21412 - 21416
DWRR Bucket Misc Configuration	21417 - 21421
DWRR Weight Configuration	21422 - 21441
Debug dstPortmask	20857
Debug srcPort	20856
Disable CPU tag on CPU Port	21515
Drain Port	21490
Drain Port Drop	21491 - 21495
EPP PM Drop	21506
EPP Packet Head Counter	21507
EPP Packet Tail Counter	21508
ERM Red Configuration	21322
ERM Yellow Configuration	21312
Egress Ethernet Type for VLAN tag	21514
Egress Port Depth	21381 - 21385
Egress Port Disabled Drop	21501 - 21505
Egress Queue Depth	21386 - 21405
Egress Resource Manager Drop	21328 - 21332
Egress Resource Manager Pointer	21323 - 21327
Egress Spanning Tree Drop	21262 - 21266

Name	Address Range
Egress Spanning Tree State	20852
Empty Mask Drop	4354
Enable Enqueue To Ports And Queues	20858 - 20862
FFA Used	21352
Force Non VLAN Packet To Specific Queue	20851
Forward From CPU	20853
Hardware Learning Configuration	149
Hardware Learning Counter	168 - 172
IPP PM Drop	4352
IPP Packet Head Counter	4364
IPP Packet Tail Counter	4365
Ingress Drop Options	21176
Ingress Ethernet Type for VLAN tag	20850
Ingress L2 ACL Drop	4360
Ingress L2 ACL Match Counter	21241 - 21256
Ingress L2 ACL Match Data Entries	21048 - 21175
Ingress L2 ACL Result Operation Entries	21015 - 21030
Ingress Packet Filtering Drop	4359
Ingress Port Packet Type Filter	21031 - 21035
Ingress Resource Manager Drop	21335
Ingress Spanning Tree Drop: Blocking	4357
Ingress Spanning Tree Drop: Learning	4356
Ingress Spanning Tree Drop: Listen	4355
L2 Aging Collision Shadow Table	20991 - 21006
L2 Aging Collision Table	152 - 167
L2 Aging Status Shadow Table	4366 - 8461
L2 Aging Table	173 - 4268
L2 Broadcast Storm Control Bucket Capacity Configuration	101 - 105
L2 Broadcast Storm Control Bucket Threshold Configuration	106 - 110
L2 Broadcast Storm Control Enable	111
L2 Broadcast Storm Control Rate Configuration	96 - 100
L2 DA Hash Lookup Table	8462 - 16653
L2 Destination Table	16654 - 20765
L2 Flooding Storm Control Bucket Capacity Configuration	133 - 137
L2 Flooding Storm Control Bucket Threshold Configuration	138 - 142
L2 Flooding Storm Control Enable	143
L2 Flooding Storm Control Rate Configuration	128 - 132
L2 Lookup Collision Table	20784 - 20815
L2 Lookup Collision Table Masks	20776 - 20783
L2 Lookup Drop	4358
L2 Multicast Handling	20855
L2 Multicast Storm Control Bucket Capacity Configuration	117 - 121
L2 Multicast Storm Control Bucket Threshold Configuration	122 - 126
L2 Multicast Storm Control Enable	127
L2 Multicast Storm Control Rate Configuration	112 - 116
L2 Multicast Table	20863 - 20990
LLDP Configuration	21040
Learning And Aging Enable	148
Learning Conflict	144
Learning Overflow	146
MAC RX Broken Packets	48 - 52
MAC RX Short Packet Drop	53 - 57
MBSC Drop	21267 - 21271

Name	Address Range
Map Queue to Priority	21336 - 21340
Maximum Allowed VLAN Drop	4363
Minimum Allowed VLAN Drop	4362
Minimum Buffer Free	21406
Output Disable	21407 - 21411
Output Mirroring Table	21509 - 21513
PB Packet Head Counter	21488
PB Packet Tail Counter	21489
PS Error Counter	21554 - 21558
PS Packet Head Counter	21552
PS Packet Tail Counter	21553
Packet Buffer Status	21333
Port Move Options	20854
Port Pause Settings	21353 - 21357
Port Reserved	21342 - 21346
Port Tail-Drop FFA Threshold	21376 - 21380
Port Tail-Drop Settings	21358 - 21362
Port Used	21347 - 21351
Port Xoff FFA Threshold	21371 - 21375
Port Xon FFA Threshold	21366 - 21370
Queue Off Drop	21257 - 21261
Re-queue Overflow Drop	21341
Resource Limiter Set	21314 - 21321
SMON Set 0 Byte Counter	21209 - 21216
SMON Set 0 Packet Counter	21177 - 21184
SMON Set 1 Byte Counter	21217 - 21224
SMON Set 1 Packet Counter	21185 - 21192
SMON Set 2 Byte Counter	21225 - 21232
SMON Set 2 Packet Counter	21193 - 21200
SMON Set 3 Byte Counter	21233 - 21240
SMON Set 3 Packet Counter	21201 - 21208
SMON Set Search	21036 - 21039
SP Overflow Drop	4304 - 4308
Scratch	4
Send to CPU	20848
Source Port Table	20766 - 20775
Tail-Drop FFA Threshold	21365
Time to Age	150
Unknown Egress Drop	21496 - 21500
Unknown Ingress Drop	4353
VLAN Member Drop	4361
VLAN PCP To Queue Mapping Table	21007 - 21014
VLAN Table	20816 - 20847
Xoff FFA Threshold	21364
Xon FFA Threshold	21363

24.5 Active Queue Manager

24.5.1 ERM Red Configuration

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

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21322

Field Description

Bits	Field Name	Description	Default Value
11:0	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.	0xcc
23:12	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.	0x200
29:24	redMaxCells	Maximum allowed packet length in cells when the buffer memory congestion status is red.	0x13

24.5.2 ERM Yellow Configuration

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

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 21312

Field Description

Bits	Field Name	Description	Default Value
11:0	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.	0x3b0
23:12	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.	0x574
28:24	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.	0x1f

Bits	Field Name	Description	Default Value
40:29	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.	0x334

24.5.3 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 : 5
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 21323 to 21327

Field Description

Bits	Field Name	Description	Default Value
1:0	q0	Pointer to the **Resource Limiter Set** for egress queue 0.	0x0
3:2	q1	Pointer to the **Resource Limiter Set** for egress queue 1.	0x0
5:4	q2	Pointer to the **Resource Limiter Set** for egress queue 2.	0x0
7:6	q3	Pointer to the **Resource Limiter Set** for egress queue 3.	0x0

24.5.4 Resource Limiter Set

This resource limiter is for comparing how many cells are ahead of the incoming cell for scheduling, that includes cells are enqueued in the same egress queue and all cells with a higher scheduling priority.

Number of Entries : 4
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : Pointer from the **Egress Resource Manager Pointer**
Address Space : 21314 to 21321

Field Description

Bits	Field Name	Description	Default Value
11:0	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**.	0x89

Bits	Field Name	Description	Default Value
23:12	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.	0x3a
35:24	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.	0x2f
41:36	maxCells	Maximum allowed packet length in cells for this limiter. Packet with cells more than this value will be dropped.	0x3f

24.6 Core Information

24.6.1 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

24.7 Egress Packet Processing

24.7.1 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 : 21515

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

24.7.2 Drain Port

Drop all packets on all queues to egress ports. The dropped packets are counted in the **Drain Port Drop** counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21490

Field Description

Bits	Field Name	Description	Default Value
4:0	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

24.7.3 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.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21514

Field Description

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

24.7.4 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 : 5
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 21509 to 21513

Field Description

Bits	Field Name	Description	Default Value
0	outputMirrorEnabled	If set to one, output mirroring is enabled for this port.	0x0
3:1	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
4	omUnderVlanMembership	If set, output mirroring to a destination that not a member of the VLAN will be ignored.	0x0

24.8 Flow Control

24.8.1 FFA Used

Total number of cells used from the common pool.

Number of Entries : 1
Type of Operation : Read Only
Address Space : 21352

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Number of cells	0x0

24.8.2 Port Pause Settings

Pause settings per source port.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Source port
Address Space : 21353 to 21357

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

24.8.3 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 : 5
Type of Operation : Read/Write
Addressing : Source port
Address Space : 21342 to 21346

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Number of cells	0x13

24.8.4 Port Tail-Drop FFA Threshold

Settings for the Port Tail-Drop FFA Threshold

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Source port
Address Space : 21376 to 21380

Field Description

Bits	Field Name	Description	Default Value
11:0	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	0x800
12	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
13	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

24.8.5 Port Tail-Drop Settings

Tail-drop settings per source port.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Source port
Address Space : 21358 to 21362

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

24.8.6 Port Used

Total number of cells used for this source port

Number of Entries : 5
Type of Operation : Read Only
Addressing : Source port
Address Space : 21347 to 21351

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Number of cells	0x0

24.8.7 Port Xoff FFA Threshold

Settings for Port Xoff FFA Threshold

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Source port
Address Space : 21371 to 21375

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Xoff threshold for the number of used FFA cells for this source port	0x0
12	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0

Bits	Field Name	Description	Default Value
13	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

24.8.8 Port Xon FFA Threshold

Settings for Port Xon FFA Threshold

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Source port
Address Space : 21366 to 21370

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Xon threshold for the number of used FFA cells for this source port	0x0

24.8.9 Tail-Drop FFA Threshold

Settings for Tail-Drop FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21365

Field Description

Bits	Field Name	Description	Default Value
11:0	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.	0x78e
12	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
13	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

24.8.10 Xoff FFA Threshold

Settings for Xoff FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21364

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Xoff threshold for the total number of used FFA cells	0x0
12	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
13	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

24.8.11 Xon FFA Threshold

Settings for Xon FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21363

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Xon threshold for the total number of used FFA cells	0x0

24.9 Global Configuration

24.9.1 Core Tick Configuration

Global register for setting the frequency of the core tick

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

Field Description

Bits	Field Name	Description	Default Value
12:0	clkDivider	The master Core Tick will be issued once every $rg\_tick\_div.clkDivider/8$ core clock cycles. If set to zero, there will be no tick.	0x28
16:13	stepDivider	The four 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

24.9.2 Core Tick Select

Global register for setting clock input to the core tick divider

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

24.9.3 Scratch

Scratch Register

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 4

Field Description

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

24.10 Ingress Packet Processing

24.10.1 Debug dstPortmask

Packet processing pipeline status for dstPortmask.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 20857

Field Description

Bits	Field Name	Description	Default Value
4:0	value	Status from last processed packet.	0x0

24.10.2 Debug srcPort

Packet processing pipeline status for srcPort.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 20856

Field Description

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

24.10.3 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
Type of Operation : Read/Write
Address Space : 20852

Field Description

Bits	Field Name	Description	Default Value
14:0	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

24.10.4 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 : 5
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 20858 to 20862

Field Description

Bits	Field Name	Description	Default Value
3:0	q_on	If a bit is set, the corresponding queue is on.	0xf

24.10.5 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 : 20851

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 14.1	0x0
2:1	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

24.10.6 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 : 20853

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

24.10.7 Hardware Learning Configuration

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

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 149

Field Description

Bits	Field Name	Description	Default Value
0	valid	For a new packet which is to be learned what value shall the valid bit have?	0x1
1	stat	For a new packet which is to be learned what value shall the static bit have?	0x0

Bits	Field Name	Description	Default Value
2	hit	For a new packet which is to be learned what value shall the hit bit have?	0x1
15:3	learnLimit	Maximum number of entries can be learned on a port. 0 means no limit.	0x0
16	portMoveException	When the hardware learning unit is turned on and the ingress packet processing determines to bypass the hardware learning check, set this field to one to still perform the port move action.	0x0
17	saHitException	When the hardware learning unit is turned on and the ingress packet processing determines to bypass the hardware learning check, set this field to one to still perform the SA hit update action.	0x0

24.10.8 Hardware Learning Counter

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

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Ingress Port
Address Space : 168 to 172

Field Description

Bits	Field Name	Description	Default Value
12:0	cnt	Number of learned L2 entries.	0x0

24.10.9 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 : 21176

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

24.10.10 Ingress Ethernet Type for VLAN tag

When decoding VLAN tags, if the Ethernet Type matches the **typeValue** it will be considered to be a VLAN tag in addition to the standard values of 0x8100 and 0x88A8. The **type** field determines if the VLAN should be regarded as a Service VLAN or Customer VLAN.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 20850

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

24.10.11 Ingress L2 ACL Match Data Entries

All packets entering the switch will be subjected to ACL filtering. This allows custom packet processing to be done for certain selected packets. For each entry it can be selected which fields that shall be part of the comparison. The entries in the table are searched starting with entry 0.

Number of Entries : 16
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 21048 to 21175

Field Description

Bits	Field Name	Description	Default Value
0	compareEthType	Determines if the EthType field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0
1	typeOfComparisonEthType	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
17:2	ethType	Ethernet Type after VLAN.	0x0
18	compareDaMac	Determines if the DaMac field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0

Bits	Field Name	Description	Default Value
19	typeOfComparisonDaMac	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
67:20	daMac	Destination MAC address.	0x0
68	compareSaMac	Determines if the SaMac field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0
69	typeOfComparisonSaMac	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
117:70	saMac	Source MAC address.	0x0
118	compareVid	Determines if the Vid field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0
119	typeOfComparisonVid	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
131:120	vid	Compared with the packets VLAN VID after Ingress VID assignment and Source Port Table VLAN operation.	0x0
132	comparePcp	Determines if the Pcp field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0
133	typeOfComparisonPcp	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
136:134	pcp	Compared with the packets VLAN PCP after Source Port Table VLAN operation.	0x0
137	compareDei	Determines if the Dei field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0
138	typeOfComparisonDei	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
139	dei	Compared with the packets VLAN CFI/DEI after Source Port Table VLAN operation.	0x0

Bits	Field Name	Description	Default Value
140	compareHasVlans	Determines if the HasVlans field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0
141	typeOfComparisonHasVlans	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
142	hasVlans	Is there at least one VLAN in the packet. 0 = No VLAN 1 = One or More VLAN	0x0
143	compareCstag	Determines if the Cstag field in this entry shall be compared. 0 = Do not compare. 1 = Include the comparison in the entry comparison.	0x0
144	typeOfComparisonCstag	What type of comparison shall be considered as hit. 0 = Not equal shall be considered a hit. 1 = Equal as hit.	0x1
145	cstag	Is the outermost VLAN tag a C-tag or S-Tag. If a packet does not have a VLAN or the VLAN was removed due to a pop operation in the source port vlan operation then the value will be set to zero(0). 0 = C-tag 1 = S-tag	0x0
150:146	ports	Ports that this filter rule applies to.	0x0

24.10.12 Ingress L2 ACL Result Operation Entries

The highest entry rule that is matched by the ACL comparison will determine what operations to perform from the corresponding entry number in this table.

Number of Entries : 16
Type of Operation : Read/Write
Addressing : **Ingress L2 ACL Match Data Entries** hit index
Address Space : 21015 to 21030

Field Description

Bits	Field Name	Description	Default Value
0	dropEnable	If set, the packet shall be dropped and the **Ingress L2 ACL Drop** counter is incremented.	0x0
1	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
2	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 14.1	0x0
4:3	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

Bits	Field Name	Description	Default Value
5	sendToPort	Send the packet to a specific port. 0 = Do not sent to a port. 1 = Send to port.	0x0
8:6	destPort	The port which the packet shall be sent to.	0x0
9	forceVidValid	A new ingress VID shall be used when doing the VLAN table lookup. This is the VID which is used for the VLAN lookup overriding the source port tables VID assignment.	0x0
14:10	forceVid	The new ingress VID which shall be used in the VLAN lookup.	0x0
15	updateCounter	When set the selected statistics counter will be updated.	0x0
19:16	counter	Which counter in **Ingress L2 ACL Match Counter** to update.	0x0

24.10.13 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 : 5
Type of Operation : Read/Write
Addressing : Ingress port
Address Space : 21031 to 21035

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

Bits	Field Name	Description	Default Value
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
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	dropIPv4Packets	Drop or allow IPv4 packets on this ingress port. 0 = Allow IPv4 packets. 1 = Drop IPv4 packets.	0x0
15	dropIPv6Packets	Drop or allow IPv6 packets on this ingress port. 0 = Allow IPv6 packets. 1 = Drop IPv6 packets.	0x0
16	dropMPLSPackets	Drop or allow MPLS packets on this ingress port. 0 = Allow MPLS packets. 1 = Drop MPLS packets.	0x0
17	dropIPv4MulticastPackets	Drop or allow IPv4 multicast packets on this ingress port. 0 = Allow IPv4 MC packets. 1 = Drop IPv4 MC packets.	0x0
18	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
19	dropL2BroadcastFrames	Drop or allow L2 broadcast packets on this ingress port. 0 = Drop L2 broadcast packets. 1 = Allow L2 broadcast packets.	0x0
20	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
21	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
22	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
23	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
24	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
25	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

24.10.14 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 :	16
Type of Operation :	Read/Write
Addressing :	**L2 Lookup Collision Table** hit index
Address Space :	20991 to 21006

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

24.10.15 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 : 16
Type of Operation : Read/Write
Addressing : **L2 Lookup Collision Table** hit index
Address Space : 152 to 167

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

24.10.16 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 : 4096
Type of Operation : Read/Write
Addressing :

address[0:9] :	hash of {GID, destination MAC}
address[10:11] :	bucket number

Address Space : 4366 to 8461

Field Description

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

24.10.17 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 :	4096
Type of Operation :	Read/Write
Addressing :	address[0:9] : hash of {GID, destination MAC} address[10:11] : bucket number
Address Space :	173 to 4268

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

24.10.18 L2 DA Hash Lookup Table

The L2 table is used for hash search based on the destination MAC address and a GID from the **VLAN Table**. When performing a L2 destination port lookup, {GID, destination MAC} is used as key for a hash calculation (see Section MAC Table Hashing). The hash is then used as index into this table to read out the 4 buckets. The incoming {GID, destination MAC} are compared to all the buckets. If any of the buckets match then address was known. The result of the lookup will be read from the **L2 Destination Table** at the same address as the matching hash index and bucket. .

Number of Entries :	4096
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address[0:9] : hash of {GID, destination MAC} address[10:11] : bucket number
Address Space :	8462 to 16653

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address.	0x0
51:48	gid	Global identifier from the VLAN Table.	0x0

24.10.19 L2 Destination Table

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

Number of Entries : 4112
Type of Operation : Read/Write
Addressing :

address 0 to 4095**L2 DA Hash Lookup Table** address :
address 4096 to4111 :**L2 Lookup Collision Table** address

Address Space : 16654 to 20765

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
7:1	destPort_or_mcAddr	Destination port number or pointer into the **L2 Multicast Table**.	0x0
8	pktDrop	If set, the packet will be dropped and the **L2 Lookup Drop** incremented.	0x0

24.10.20 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 : 16
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 20784 to 20815

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address	0x0
51:48	gid	Global identifier for learning	0x0

24.10.21 L2 Lookup Collision Table Masks

Masks for collision memory for the MAC address and the global identifier. Only the first 4entries has masks on them.

Number of Entries : 4
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 20776 to 20783

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address mask	$2^{48}-1$
51:48	gid	Global identifier for learning mask	0xf

24.10.22 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 : 20855

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

24.10.23 L2 Multicast Table

L2 multicast table.

Number of Entries : 128
Type of Operation : Read/Write
Addressing : mcAddr field from **L2 Destination Table**
Address Space : 20863 to 20990

Field Description

Bits	Field Name	Description	Default Value
4:0	mcPortMask	L2 portmask entry members. If set, the port is part of multicast group and shall be transmitted to.	0x1f

24.10.24 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 :	21040

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
165:161	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.	0xf

24.10.25 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 :	148

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

Bits	Field Name	Description	Default Value
2	daHitEnable	If set, MAC DA hit in the forwarding information base will update the hit bit for non-static entries.	0x1
3	lru	If set, the learning unit will try to overwrite a least recently used non-static entry in either the hash table or the collision table when there is no free entry to use. Otherwise the learning unit will try to overwrite a non-static entry in the collision table.	0x0

24.10.26 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 : 2
Type of Operation : Read/Write
Address Space : 144

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
52:49	gid	Global identifier from the VLAN Table.	0x0
55:53	port	Port number.	0x0

24.10.27 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 : 2
Type of Operation : Read/Write
Address Space : 146

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
52:49	gid	Global identifier from the VLAN Table.	0x0

Bits	Field Name	Description	Default Value
55:53	port	Port number.	0x0

24.10.28 Port Move Options

Determine if port move is allowed on static entries.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 20854

Field Description

Bits	Field Name	Description	Default Value
4:0	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.	0x1f

24.10.29 SMON Set Search

If both source port and VLAN ID match one of the entries, the corresponding SMON counter will be updated.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : SMON set number
Address Space : 21036 to 21039

Field Description

Bits	Field Name	Description	Default Value
2:0	srcPort	Source port	0x0
14:3	vid	VLAN ID	0x0

24.10.30 Send to CPU

Configuration of MAC addresses used to redirect packets to CPU.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	20848

Field Description

Bits	Field Name	Description	Default Value
4:0	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.	0x1f
9:5	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.	0x1f
14:10	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
62:15	cpuMacAddr	Packets with this destination MAC address will be sent to the CPU. Only valid if **uniqueCpuMac** on the source port is set.	0x0

24.10.31 Source Port Table

This table configures various functions that are dependent on which port the packet enters the switch. Configurations including input mirroring, spanning tree state, Ingress VID offset, special VID treatment, multicast learning, and min/max number of VLANs.

Number of Entries :	5
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	20766 to 20775

Field Description

Bits	Field Name	Description	Default Value
0	learningEn	If hardware learning is turned on and this is set to one, the unknown source MAC address from this port will be learned.	0x1
1	dropUnknownDa	If set to one packets with unknown destination MAC address from this port will be dropped.	0x0

Bits	Field Name	Description	Default Value
3:2	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
5:4	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
17:6	defaultVid	The default VID. 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
18	defaultCfiDei	The default CFI / DEI bit. It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
21:19	defaultPcp	The default PCP bits. It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
23:22	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
25:24	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
26	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
27	learnMulticastSaMac	If set, the learning engine allows Ethernet multicast source MAC addresses to be learned.	0x0

Bits	Field Name	Description	Default Value
28	learnMacDaEqSa	Set to zero to ignore the hardware learning request when MAC DA equals SA.	0x1
29	learnSaMac0	Set to zero to ignore the hardware learning request on MAC SA 00:00:00:00:00:00.	0x0
30	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
31	imUnderVlanMembership	If set, input mirroring to a destination that not a member of the VLAN will be ignored.	0x0
34:32	destInputMirror	Destination port for input mirroring. Only valid if **inputMirrorEnabled** is set.	0x0
37:35	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

24.10.32 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 :	150

Field Description

Bits	Field Name	Description	Default Value
31:0	tickCnt	Number of ticks (see Chapter Tick) between starts of the aging process.	$2^{32} - 1$
34:32	tick	Select one of the 5 available ticks. The tick frequencies are configured globaly in the **Core Tick Configuration** register.	0x0

24.10.33 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 :	21007 to 21014

Field Description

Bits	Field Name	Description	Default Value
1:0	pQueue	Egress queue.	0x1

24.10.34 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 :	16
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	The packet's Ingress VID plus offset as defined in **Source Port Table**.
Address Space :	20816 to 20847

Field Description

Bits	Field Name	Description	Default Value
4:0	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.	0x1f
8:5	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
11:9	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
13:12	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

Bits	Field Name	Description	Default Value
15:14	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
17:16	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
29:18	vid	The VID used in VLAN push or swap operation if selected by **vidSel**.	0x0
32:30	pcp	The PCP used in VLAN push or swap operation if selected by **pcpSel**.	0x0
33	cfiDei	The CFI/DEI used in VLAN push or swap operation if selected by **cfiDeiSel**	0x0
35:34	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

24.11 MBSC

24.11.1 L2 Broadcast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Broadcast Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 101 to 105

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x49c

24.11.2 L2 Broadcast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Broadcast Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 106 to 110

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.	0x24e

24.11.3 L2 Broadcast Storm Control Enable

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

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 111

Field Description

Bits	Field Name	Description	Default Value
4:0	enable	Bitmask where the index is the Egress Ports	0x0

24.11.4 L2 Broadcast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Broadcast Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 96 to 100

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	0x3b
15:13	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x4
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

24.11.5 L2 Flooding Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Flooding Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 133 to 137

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x49c

24.11.6 L2 Flooding Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Flooding Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 138 to 142

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.	0x24e

24.11.7 L2 Flooding Storm Control Enable

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

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 143

Field Description

Bits	Field Name	Description	Default Value
4:0	enable	Bitmask where the index is the Egress Ports	0x0

24.11.8 L2 Flooding Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Flooding Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 128 to 132

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	0x3b
15:13	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x4
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

24.11.9 L2 Multicast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Multicast Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 117 to 121

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	0x49c

24.11.10 L2 Multicast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Multicast Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 122 to 126

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.	0x24e

24.11.11 L2 Multicast Storm Control Enable

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

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 127

Field Description

Bits	Field Name	Description	Default Value
4:0	enable	Bitmask where the index is the Egress Ports	0x0

24.11.12 L2 Multicast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Multicast Storm Control

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 112 to 116

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	0x3b
15:13	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x4
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

24.12 Scheduling

24.12.1 DWRR Bucket Capacity Configuration

Token Bucket Capacity Configuration for DWRR

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 21412 to 21416

Field Description

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

24.12.2 DWRR Bucket Misc Configuration

Bucket Configurations for DWRR

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 21417 to 21421

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.	0xb
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

24.12.3 DWRR Weight Configuration

Weight Configuration for DWRR

Number of Entries : 20
Type of Operation : Read/Write
Addressing : Egress port * 4 + queue
Address Space : 21422 to 21441

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

24.12.4 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 : 5
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 21336 to 21340

Field Description

Bits	Field Name	Description	Default Value
1:0	prio0	The priority for queue 0	0x0
3:2	prio1	The priority for queue 1	0x1
5:4	prio2	The priority for queue 2	0x2
7:6	prio3	The priority for queue 3	0x3

24.12.5 Output Disable

Bitmask for disabling the egress queues on egress ports.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 21407 to 21411

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

24.13 Shared Buffer Memory

24.13.1 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
11:0	cells	Number of free cells.	0x800

24.13.2 Egress Port Depth

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

Number of Entries : 5
Type of Operation : Read Only
Addressing : Egress Port
Address Space : 21381 to 21385

Field Description

Bits	Field Name	Description	Default Value
11:0	packets	Number of packet currently queued.	0x0

24.13.3 Egress Queue Depth

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

Number of Entries : 20
Type of Operation : Read Only
Addressing : Global queue number
Address Space : 21386 to 21405

Field Description

Bits	Field Name	Description	Default Value
11:0	packets	Number of packets currently queued.	0x0

24.13.4 Minimum Buffer Free

Minimum number of cells available in the buffer memory

Number of Entries : 1
Type of Operation : Read Only
Address Space : 21406

Field Description

Bits	Field Name	Description	Default Value
11:0	cells	Number of cells.	0x800

24.13.5 Packet Buffer Status

Queue status of the packet buffer

Number of Entries : 1
Type of Operation : Read Only
Address Space : 21333

Field Description

Bits	Field Name	Description	Default Value
4:0	empty	Empty flags for the egress ports	0x1f

24.14 Statistics: ACL

24.14.1 Ingress L2 ACL Match Counter

Number of packets hit in entries of **Ingress L2 ACL Match Data Entries**.
In Figure 19.1, **ippAcl** with process sequence **11** represents the internal location of this counter.

Number of Entries : 16
Type of Operation : Read/Write
Addressing : See **Ingress L2 ACL Match Data Entries** for how ACL rules are mapped to counters.
Address Space : 21241 to 21256

Field Description

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

24.15 Statistics: Debug

24.15.1 EPP PM Drop

Number of drops due to FIFO overflows in EPP PM.
In Figure 19.1, **epmOverflow** with process sequence **22** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21506

Field Description

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

24.15.2 IPP PM Drop

Number of drops due to FIFO overflows in IPP PM.
In Figure 19.1, **ipmOverflow** with process sequence **12** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4352

Field Description

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

24.15.3 PS Error Counter

Number of errors occured in the PS-converter.
In Figure 19.1, **psError** with process sequence **25** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 21554 to 21558

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
15:8	overflow	Number of FIFO overflows in the PS-converter. This error will cause packet corruptions.	0x0

24.15.4 SP Overflow Drop

Number of packets dropped due to: FIFO overflow in the SP-converter.
In Figure 19.1, **spOverflow** with process sequence **5** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read Only
Addressing : Ingress port
Address Space : 4304 to 4308

Field Description

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

24.16 Statistics: EPP Egress Port Drop

24.16.1 Egress Port Disabled Drop

Number of packets dropped due to egress port disabled.
In Figure 19.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 21501 to 21505

Field Description

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

24.16.2 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 19.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 21496 to 21500

Field Description

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

24.17 Statistics: IPP Egress Port Drop

24.17.1 Egress Spanning Tree Drop

Number of packets dropped due to egress spanning tree check configured in **Egress Spanning Tree State**
In Figure 19.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 21262 to 21266

Field Description

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

24.17.2 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 19.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 21267 to 21271

Field Description

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

24.17.3 Queue Off Drop

Number of packets dropped due to the queue being turned off.
In Figure 19.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 21257 to 21261

Field Description

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

24.18 Statistics: IPP Ingress Port Drop

24.18.1 Empty Mask Drop

Number of packets dropped due to an empty destination port mask.
In Figure 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4354

Field Description

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

24.18.2 Ingress L2 ACL Drop

Number of packets dropped due to the L2 ingress ACL.
In Figure 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4360

Field Description

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

24.18.3 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 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4359

Field Description

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

24.18.4 Ingress Spanning Tree Drop: Blocking

Number of packets dropped due to that a ports's ingress spanning tree protocol state was **Blocking**. In Figure 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4357

Field Description

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

24.18.5 Ingress Spanning Tree Drop: Learning

Number of packets dropped due to that a port's ingress spanning tree protocol state was **Learning**. In Figure 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4356

Field Description

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

24.18.6 Ingress Spanning Tree Drop: Listen

Number of packets dropped due to that a port's ingress spanning tree protocol state was **Listening**. In Figure 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4355

Field Description

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

24.18.7 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 .
In Figure 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4358

Field Description

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

24.18.8 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 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4363

Field Description

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

24.18.9 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 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4362

Field Description

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

24.18.10 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 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4353

Field Description

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

24.18.11 VLAN Member Drop

Number of packets dropped due to the packets source port notbeing part of the packets VLAN membership.
In Figure 19.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4361

Field Description

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

24.19 Statistics: Misc

24.19.1 Buffer Overflow Drop

Counter for the number of packets dropped due to the shared buffer memory being full.
In Figure 19.1, **bmOverflow** with process sequence **16** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21334

Field Description

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

24.19.2 Drain Port Drop

Number of packets dropped due to the port is drained.
In Figure 19.1, **drain** with process sequence **21** represents the internal location of this counter.

Number of Entries :	5
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	21491 to 21495

Field Description

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

24.19.3 Egress Resource Manager Drop

Number of packets dropped by the egress resource manager.
In Figure 19.1, **erm** with process sequence **15** represents the internal location of this counter.

Number of Entries :	5
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space :	21328 to 21332

Field Description

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

24.19.4 Ingress Resource Manager Drop

Counter for the number of packets dropped due to exeeding thresholds set up in the ingress resource manager.
In Figure 19.1, **irm** with process sequence **16** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	21335

Field Description

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

24.19.5 MAC RX Broken Packets

Number of broken packets dropped (packets with last=1 and valid_bytes=0).
In Figure 19.1, **macBrokenPkt** with process sequence **3** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read Only
Addressing : Ingress Port
Address Space : 48 to 52

Field Description

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

24.19.6 MAC RX Short Packet Drop

Number of packets dropped due to length below 60 bytes.
In Figure 19.1, **macRxMin** with process sequence **4** represents the internal location of this counter.

Number of Entries : 5
Type of Operation : Read Only
Addressing : Ingress Port
Address Space : 53 to 57

Field Description

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

24.19.7 Re-queue Overflow Drop

Counter for the number of packets dropped due to a FIFO overflow in re-queue.
In Figure 19.1, **rqOverflow** with process sequence **24** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21341

Field Description

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

24.20 Statistics: Packet Datapath

24.20.1 EPP Packet Head Counter

Number of packet first cells through the Egress Packet Process module.
In Figure 19.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21507

Field Description

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

24.20.2 EPP Packet Tail Counter

Number of packet last cells through the Egress Packet Process module.
In Figure 19.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21508

Field Description

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

24.20.3 IPP Packet Head Counter

Number of packet first cells through the Ingress Packet Process module.
In Figure 19.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4364

Field Description

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

24.20.4 IPP Packet Tail Counter

Number of packet last cells through the Ingress Packet Process module.
In Figure 19.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 4365

Field Description

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

24.20.5 PB Packet Head Counter

Number of packet first cells through the Shared Buffer Memory module.
In Figure 19.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21488

Field Description

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

24.20.6 PB Packet Tail Counter

Number of packet last cells through the Shared Buffer Memory module.
In Figure 19.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21489

Field Description

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

24.20.7 PS Packet Head Counter

Number of packet first cells through the Parallel to Serial module.
In Figure 19.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21552

Field Description

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

24.20.8 PS Packet Tail Counter

Number of packet last cells through the Parallel to Serial module.
In Figure 19.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 21553

Field Description

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

24.21 Statistics: SMON

24.21.1 SMON Set 0 Byte Counter

Number of bytes counted in SMON Set 0.
In Figure 19.1, **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 : 21209 to 21216

Field Description

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

24.21.2 SMON Set 0 Packet Counter

Number of packets counted in SMON Set 0.
In Figure 19.1, **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 :	21177 to 21184

Field Description

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

24.21.3 SMON Set 1 Byte Counter

Number of bytes counted in SMON Set 1.
In Figure 19.1, **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 :	21217 to 21224

Field Description

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

24.21.4 SMON Set 1 Packet Counter

Number of packets counted in SMON Set 1.
In Figure 19.1, **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 :	21185 to 21192

Field Description

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

24.21.5 SMON Set 2 Byte Counter

Number of bytes counted in SMON Set 2.
In Figure 19.1, **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 :	21225 to 21232

Field Description

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

24.21.6 SMON Set 2 Packet Counter

Number of packets counted in SMON Set 2.
In Figure 19.1, **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 :	21193 to 21200

Field Description

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

24.21.7 SMON Set 3 Byte Counter

Number of bytes counted in SMON Set 3.
In Figure 19.1, **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 :	21233 to 21240

Field Description

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

24.21.8 SMON Set 3 Packet Counter

Number of packets counted in SMON Set 3.
In Figure 19.1, **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 :	21201 to 21208

Field Description

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