PM9311/2/3/5

Enhanced TT1 Chip Set

Enhanced TT1 Switch Fabric

Datasheet

Issue 3: August 2001

© 2000
PMC-Sierra, Inc.
8555 Baxter Place
Burnaby BC Canada V5A 4V7
Phone: (604) 415-6000 FAX: (604) 415-6200

Device Part Numbers

This document contains information on the ETT1™ Chip Set, available from PMC-Sierra, Inc. The following devices comprise the ETT1 Chip Set:

Device Name	PMC Part Number
Scheduler	PM9311-UC
Crossbar	PM9312-UC
Dataslice	PM9313-HC
Enhanced Port Processor	PM9315-HC

Revision History

Issue Number	Issue Date	Details Of Change
1	March 2000	Creation of document
2	July 2000	Added HSTL Power Dissipation values, corrected programming constraints for EPP, added OOB, JTAG and PLL blocks to device dataflow diagrams, added heat sink information to Characteristics section, corrected EPP register section, corrected Frame Format tables, added correct bit definition for EPP Interrupt register, corrected diagram in Appendix C, added send and receive procedure for control packets, added ESOBLCP register explanation, corrected conditions in Table 57, corrected Scheduler Refresh Procedure, added new drawing for Crossbar data flow, corrected token explanation, corrected Figures 36 and 37, corrected spelling and formatting errors throughout document.
3	August 2001	Removed bit 17 from Scheduler Status Register, updated BP_FIFO information in Scheduler Control and Reset Register, corrected bottom view drawings for Scheduler and Crossbar, corrected signal description section in Scheduler and Crossbar for power pins, corrected tper3, and tpl3 in AC Electrical, added memory address information in EPP registers, corrected mechanical drawings, updated state diagram in Scheduler, added information about Scheduler PLL timing, updated initialization sequence for all chips, corrected Dataslice Signal Description for ibpen0, added bit 14(13th and 14th Dataslice enable) to EPP Control register, updated TDM constraints, added Output TDM Queue Overflow to the EPP's Interrupt register, updated EPP Output Backpressure / Unbackpressure Threshold register, updated LCS2 link synchronization in appendix, modified EPP Egress Control Packet Data Format table, Modified ETT1 usage of LCS2 protocol section

1 Functional Description

1.1 OVERVIEW

The ETT1™ Chip Set provides a crossbar-based switch core which is capable of switching cells between 32 ports with each port operating at data rates up to 10 Gbit/s. This section describes the main features of the switch core and how cells flow through a complete system that is based on the ETT1 Chip Set.

This document often refers to port rates of OC-192c or OC-48c. The ETT1 Chip Set itself operates at a fixed cell rate of 25M cells per second per port and thus is unaware of the actual data rate of the attached link. So a switch might be 32 ports of OC-192c, or it could be 32 ports of 10 Gbit/s Ethernet; it is the internal cell rate that is determined by the ETT1 Chip Set, not the link technology.

1.1.1 ETT1 Switch Core Features

The ETT1 switch core provides the following features:

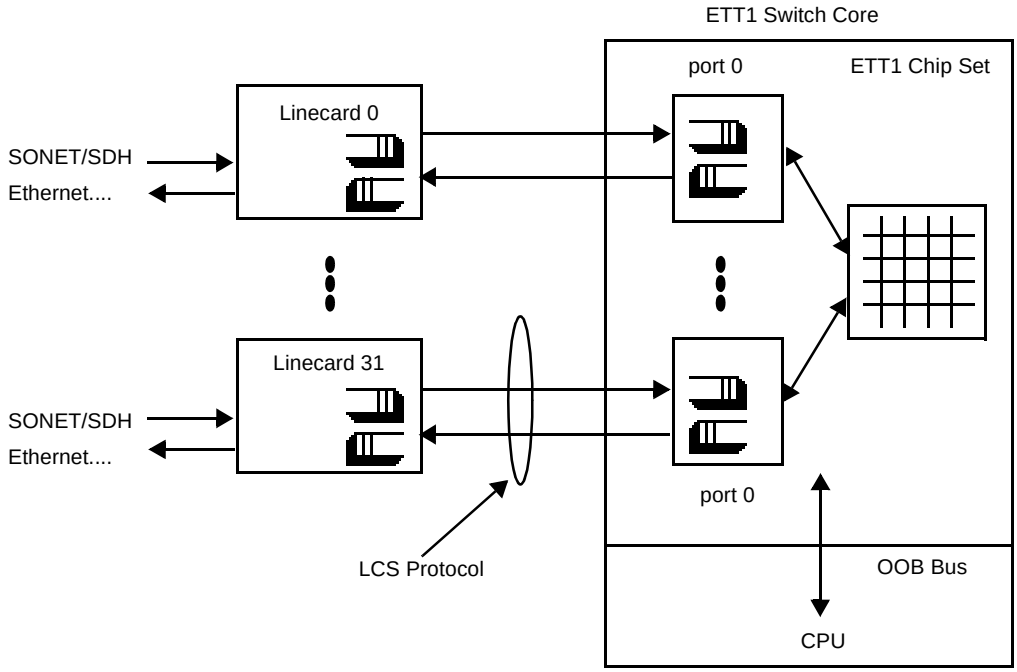
- 320 Gbit/s aggregate bandwidth - up to 32 ports of 10 Gbit/s bandwidth each
- Each port can be configured as 4 x OC-48c or 1 x OC-192c
- Both port configurations support four priorities of best-effort traffic for unicast and multicast data traffic
- TDM support for guaranteed bandwidth and zero delay variation with 10 Mbit/s channel resolution
- LCS™ protocol supports a physical separation of switch core and linecards up to 200 feet (70 m)
- Virtual output queues to eliminate head-of-line blocking on unicast cells
- Internal speedup to provide near-output-queued performance
- Cells are transferred using a credit mechanism to avoid cell losses due to buffer overrun
- In-band management and control via Control Packets
- Out-of-band management and control via a dedicated CPU interface
- Optional redundancy of all shared components for fault tolerance
- Efficient support for multicast with cell replication performed within the switch core

1.1.2 The Switch Core Model

The ETT1 Chip Set is designed to provide a switch core, not a complete packet switch system. A complete switch consists of one or more switch cores, together with a number of linecards. Each linecard connects to one port in the core. The linecard includes a physical interface (fiber, co-axial cable) to a transmission system such as SONET/SDH or Ethernet. The linecard analyzes incoming cells or packets and determines the appropriate egress port and priority. The linecard contains any cell/packet queues that are needed to allow for transient congestion through the switch.

The ETT1 switch core operates on fixed size cells. If the linecard transmission system uses variable length packets, or cells of a size different from those used in the core, then the linecard is responsible for performing any segmentation and reassembly that is needed. Figure 1 illustrates this generic configuration.

Figure 1. The Basic Components of a Switch Built Around the ETT1 Chip Set.



The ETT1 Chip Set has been designed to allow for up to 200 feet (70 meters) of physical separation between the ETT1 core and the linecard. The LCS™(Linecard to Switch) protocol is used between the ETT1 core and the linecard to ensure lossless transfer of cells between the two entities. However, while the LCS protocol must be implemented, the physical separation is not mandatory; the linecard could reside on the same physical board as the ETT1 port devices.

The switch core has two main interfaces. One is the interface between the linecard and the core port, described in the LCS Protocol section.

The second interface is between the ETT1 devices and the local CPU. The ETT1 Chip Set requires a local CPU for configuration, diagnostics and maintenance purposes. A single CPU can control a complete ETT1 core via a common Out-Of-Band (OOB) bus. All of the ETT1 devices have an interface to the OOB bus. The OOB bus is described in Section 1.1.4 “The OOB (Out-Of-Band) Bus” on page 18.

1.1.3 The LCS Protocol

The Linecard-to-Switch (LCS™) protocol provides a simple, clearly defined interface between the linecard and the core. In this section we introduce LCS. There are two aspects to LCS:

- a per-queue, credit-based flow control protocol
- a physical interface

The LCS protocol provides per-queue, credit-based flow control from the ETT1 core to the linecard, which ensures that queues are not overrun. The ETT1 core has shallow (64 cells) queues in both the ingress and egress directions. These queues compensate for the latency between the linecard and the core. One way to think of these queues is simply as extensions of the queues within the linecards. The queues themselves are described further in Section 1.3 “Prioritized Best-Effort Queue Model” on page 30.

The LCS protocol is asymmetrical; it uses different flow control mechanisms for the ingress and egress flows. For the ingress flow LCS uses credits to manage the flow of cells between the linecards and the ETT1 core. The core provides the linecard with a certain number of credits for each ingress queue in the core. These credits correspond to the number of cell requests that the linecard can send to the core. For each cell request that is forwarded to a given queue in the core the linecard must decrement the number of credits for that queue. The core sends a grant (which is also a new credit) to the linecard whenever the core is ready to accept a cell in response to the cell request. At some later time, which is dependent on the complete traffic load, the cell will be forwarded through the ETT1 core to the egress port. In the egress direction a linecard can send hole requests, requesting that the ETT1 core does *not* forward a cell for one celltime. The linecard can issue a hole request for each of the four best effort unicast or multicast priorities. If the linecard continually issued hole requests at all four priorities then the ETT1 core would not forward any best effort traffic to the linecard.

The LCS protocol information is contained within an eight byte header that is added to every cell.

The physical interface that has been implemented in the ETT1 Chip Set is based on a faster version of the Gigabit Ethernet Serdes interface, enabling the use of off-the-shelf parts for the physical link. This interface provides a
1.5 Gbit/s serial link that uses 8b/10b encoded data. Twelve of these links are combined to provide a single LCS link operating at 18 Gbaud, providing an effective data bandwidth that is in excess of an OC-192c link.

NOTE: The LCS protocol is defined in the “LCS Protocol Specification -- Protocol Version 2”, available from PMC-Sierra, Inc. This version of LCS supersedes LCS Version 1. Version 2 is first supported in the TT1 Chip Set with the Enhanced Port Processor device (also referred to as the ETT1 Chip Set) and will be supported in future PMC-Sierra products.

The ETT1 implementation of the LCS protocol is described further in Section 1.6 “ETT1 Usage of the LCS Protocol” on page 64.

1.1.4 The OOB (Out-Of-Band) Bus

The ETT1 Chip Set requires a local CPU to perform initialization and configuration after the Chip Set is reset or if the core configuration is changed - perhaps new ports are added or removed, for example. The OOB bus provides a simple mechanism whereby a local CPU can configure each device.

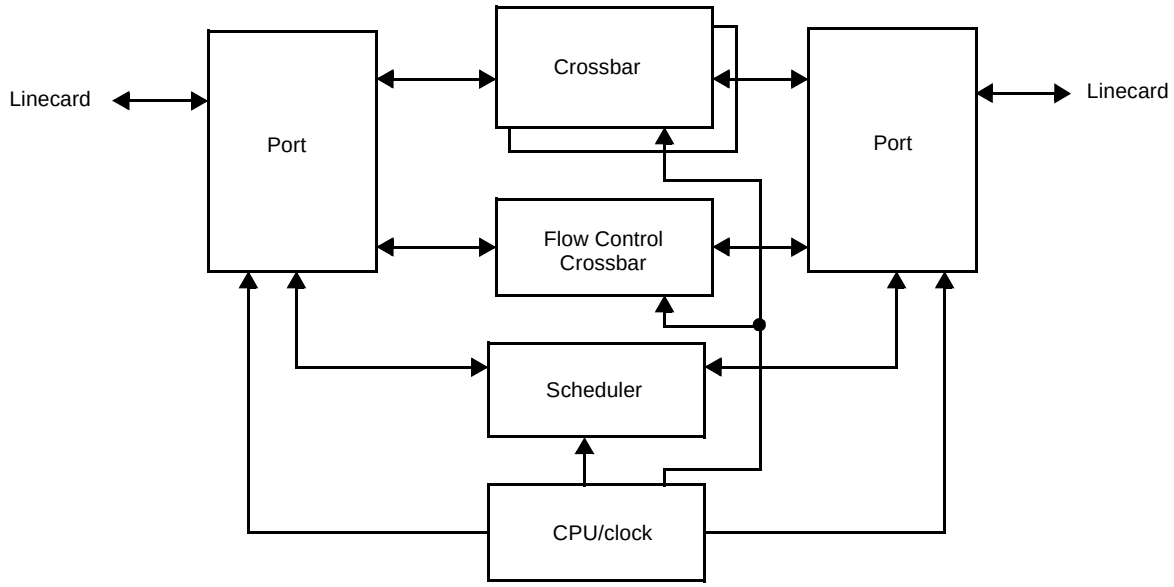
Logically, the OOB bus provides a 32 bit address/data bus with read/write, valid and ready signals. The purpose of the OOB bus is for maintenance and diagnostics; the CPU is not involved in any per-cell operations.

1.2 ARCHITECTURE AND FEATURES

1.2.1 ETT1 Switch Core

An ETT1 switch core consists of four types of entities: Port, Crossbar, Scheduler, and CPU/Clock. There may be one or more instances of each entity within a switch. For example, each entity might be implemented as a separate PCB, with each PCB interconnected via a single midplane PCB. Figure 2 illustrates the logical relationship between these entities.

Figure 2. ETT1 Switch Core Logical Interconnects



Each ETT1 port is attached to one or more linecards. The port contains the shallow cell queues and implements the LCS protocol. The port and Scheduler exchange information about cells that are waiting to be forwarded through the Crossbar core.

The Scheduler maintains local information on the number of cells that are waiting in all of the ingress and egress queues. It arbitrates amongst all cells in the ingress queues, and instructs all of the ports as to which cell they can forward through the Crossbar at each cell time.

Two Crossbars are used. The first, referred to as simply 'the Crossbar', interconnects all of the ports with all of the other ports, enabling cells to be forwarded from the ingress port queues to the egress port queues (in a different port). The Crossbar is reconfigured at every cell time to provide any non-blocking one-to-one or one-to-many mapping from input ports to output ports. Each Crossbar port receives its configuration information from its attached port; the Crossbars do not communicate directly with the Scheduler. The second Crossbar is the flow-control Crossbar. It passes output queue occupancy information from every egress port to every ingress port. The ingress ports use this information to determine when requests should and should not be made to the Scheduler.

The CPU/clock provides clocks and cell boundary information to every ETT1 device. It also has a local CPU which can read and write state information in every ETT1 device via the OOB bus. The CPU/clock entity is a necessary element of the ETT1 switch core, but does not contain any of the ETT1 devices.

1.2.2 Basic Cell Flow

The ETT1 Chip Set consists of four devices. Their names (abbreviations) are:

- Dataslice (DS)
- Enhanced Port Processor (EPP)
- Scheduler (Sched)
- Crossbar (Xbar)

This section describes how cells flow through these four devices.

Figure 3. Two Port Configuration of a ETT1 Switch

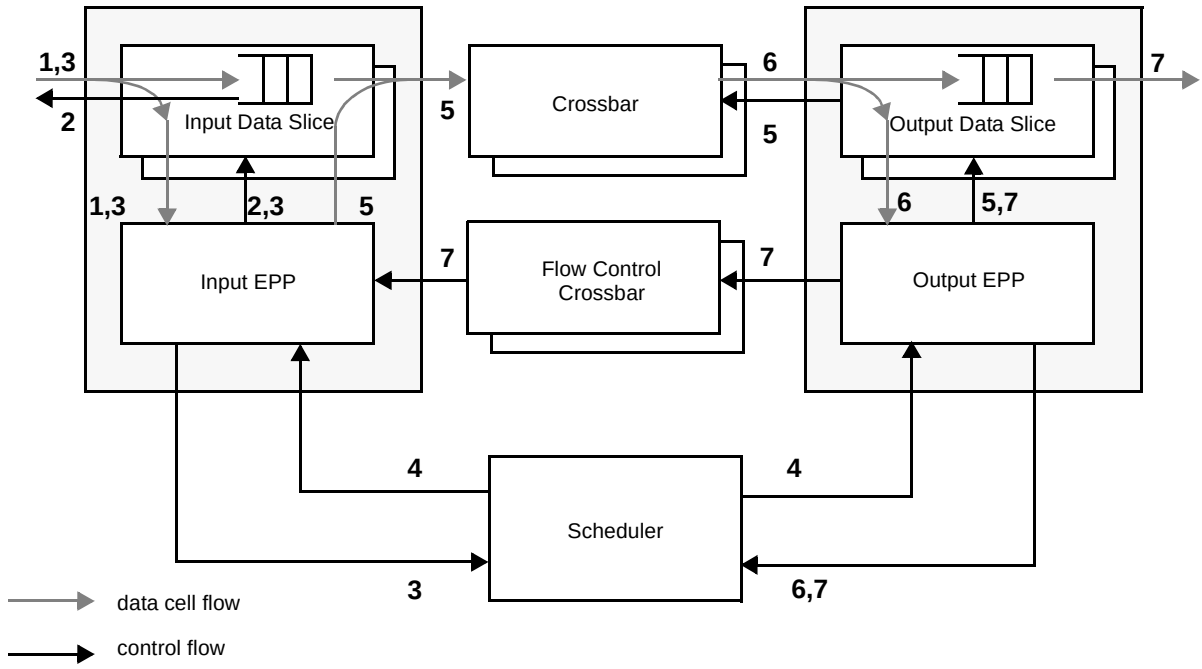


Figure 3 shows a two-port configuration of a ETT1 switch. Only the ingress queues of the left hand port and the egress queues of the right hand port are shown. The port has one EPP and either six or seven DS devices. The DS contains the cell queue memory and also has the Serdes interface to the linecard. A single cell is “sliced” across all of the Dataslice devices, each of which can manage two slices. The EPP is the port controller, and determines where cells should be stored in the DS memories. Multiple Crossbar devices make up a full Crossbar. A single Scheduler device can arbitrate for the entire core.

A cell traverses the switch core in the following sequence of events:

1. A cell request arrives at the ingress port, and is passed to the EPP. The EPP adds the request to any other outstanding cell requests for the same queue.
2. At some later time, the EPP issues a grant/credit to the source linecard, requesting an actual cell for a specific queue. The linecard must respond with the cell within a short period of time.
3. The cell arrives at the ingress port and the LCS header is passed to the EPP. The EPP determines the destination queue from the LCS header, and then tells the Dataslices where to store the cell (each Dataslice stores part of the cell). The EPP also informs the Scheduler that a new cell has arrived and so the Scheduler should add it to the list of cells waiting to be forwarded through the Crossbar. The EPP modifies the LCS label by replacing the destination port with the source port, so the egress port and linecard can see which port sent the cell.
4. The Scheduler arbitrates among all queued cells and sends a grant to those ports that can forward a cell. The Scheduler also sends a *routing tag* to each of the destination (egress) ports; this tag

tells the ports which of the many source ports will be sending it a cell.

5. The source EPP sends a read command to the Dataslices, which then reads the cell from the appropriate queue and sends it to the Crossbar. At the same time, the destination ports send the routing tag information to the Crossbar. This routing tag information is used to configure the internal connections within the Crossbar for the duration of one cell time. The cell then flows through the Crossbar from the source port to the destination port.
6. The cell arrives at the destination port, and the EPP receives the LCS header of the cell. It uses this information to decide in which egress queue the cell should be stored. If this was a multicast cell which caused the egress multicast to reach its occupancy limit, then the EPP would send a congestion notification to the Scheduler.
7. At some later time, the EPP decides to forward the cell to the linecard. The EPP sends a read command to the Dataslices which read the cell from memory and forward the cell out to the linecard. The egress EPP also sends flow control to the ingress EPP, informing it that there now exists free space in one or more of the egress EPP's output queues. Also, if the transmitted cell was a multicast cell then this may cause the egress queue to go from full to not full, in which case the EPP notifies the Scheduler that it (the EPP) can once again accept multicast cells.

The above description does not account for all of the interactions that can take place between ETT1 devices, but it describes the most frequent events. In general, users do not need to be aware of the detailed interactions, however knowledge of the main information flows will assist in gaining an understanding of some of the more complicated sections.

1.2.3 Prioritized Best-effort Service

An ETT1 switch core provides two types of service. The first is a prioritized, best-effort service. The second provides guaranteed bandwidth and is described later.

The best-effort service is very simple. Linecards forward best-effort cells to the ETT1 core where they will be queued. The Scheduler arbitrates among the various cells; the arbitration algorithm has the dual goals of maximizing throughput while providing fair access to all ports. If more than one cell is destined for the same egress port then the Scheduler will grant one of the cells and the others will remain in their ingress queues awaiting another round of arbitration. The service is best-effort in that the Scheduler tries its best to satisfy all queued cells, but in the case of contention then some cells will be delayed.

The Scheduler supports four levels of strict priority for best effort traffic. Level 0 cells have the highest priority, and level 3 cells have the lowest priority. A level 0 cell destined for a given port will always be granted before a cell of a different priority level, in the same ingress port, that is destined for the same egress port.

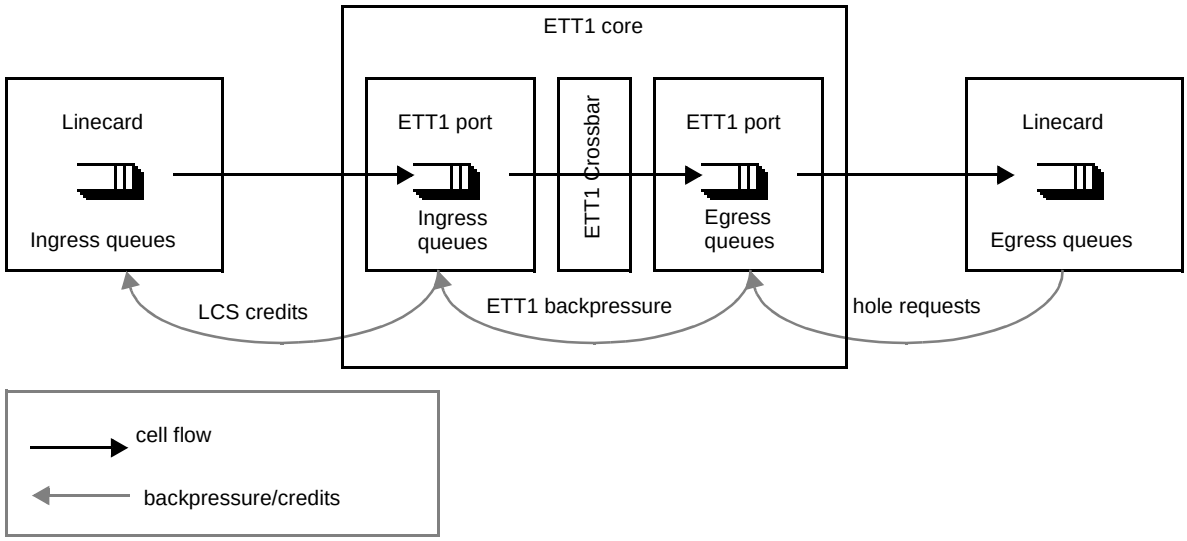
A 'flow' is a sequence of cells from the same ingress port to the same egress port(s) at a given priority. Best-effort flows are either unicast flows (cells in the flow go to only one egress port), or multicast flows (in which case cells can go to many, even all, of the egress ports).

1.2.4 End-to-End Flow Control

The full queueing and flow control model is shown in Figure 4.

NOTE: Credits or backpressure are used at every transfer point to ensure that cells cannot be lost due to lack of buffer space.

Figure 4. Queueing and Flow Control



1.2.5 TDM Service

The ETT1 TDM service provides guaranteed bandwidth and zero cell delay variation. These properties, which are not available from the best-effort service, mean that the TDM service might be used to provide an ATM CBR service, for example. The ETT1 core provides the TDM service at the same time as the best effort service, and TDM cells integrate smoothly with the flow of best-effort traffic. In effect, the TDM cells appear to the Scheduler to be cells of the highest precedence, even greater than level zero best-effort multicast traffic.

The TDM service operates by enabling a ETT1 port (and linecard) to reserve the crossbar fabric at some specified cell time in the future. The Scheduler is notified of this reservation and will not schedule any best-effort cells from the ingress port or to the egress port during that one cell time. Each port can make separate reservations according to whether it will send and/or receive a cell at each cell time.

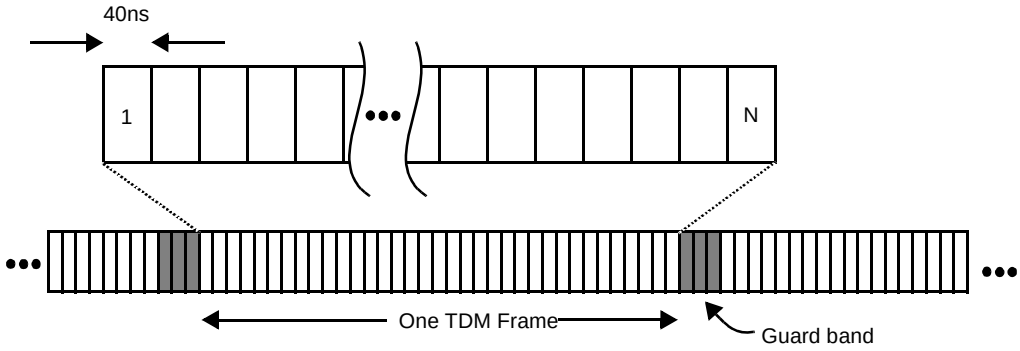
Several egress ports may receive the same cell from a given ingress port; therefore, the TDM service is inherently a multicast service.

In order to provide a guaranteed bandwidth over a long period of time, an ingress port will want to repeat the reservations on a regular basis. To support this the ETT1 core uses an internal construct called a TDM

Frame. A TDM Frame is simply a sequence of ingress and egress reservations. The length (number of cell times) of a TDM Frame is configurable up to a maximum of 1024 cells. The TDM Frame repeats after a certain fixed time. All ports are synchronized to the start of the TDM Frame, and operate cell-synchronously with respect to each other. Thus, at any cell time, every ETT1 port knows whether it has made a reservation to send or receive a TDM cell. See the application note “LCS-2 TDM Service in the ETT1 and TTX Switch Core”, available from PMC-Sierra, Inc.

Figure 5 illustrates the idea of a TDM Frame. The TDM Frame has N slots (N is 1024 or less) where each slot is one 40ns cell time. The TDM Frame is repeated continuously, but adjacent TDM Frames are separated by a guard band of at least 144 cells.

Figure 5. The TDM Frame Concept



All of the ETT1 ports must be synchronized with respect to the start of the TDM Frame. This synchronization information is distributed via the Scheduler. The Scheduler can generate the synchronization pulses itself, or it can receive “Suggested Sync” pulses from a ETT1 port which can in turn receive synchronization signals from a linecard. The linecards do *not* need to be exactly synchronized to either the ETT1 core or the other linecards. The LCS protocol will compensate for the asynchrony between the linecard and the ETT1 core.

1.2.6 Subport Mode (2.5 Gbit/s Linecards)

An ETT1 switch core can have up to 32 ports. Each port supports a linecard bandwidth in excess of 10 Gbit/s. This bandwidth might be used by a single linecard (for example, OC-192c), or the ETT1 port can be configured to share this bandwidth among up to four ports, each of 2.5 Gbit/s. This latter mode, specifically four 2.5 Gbit/s linecards, is referred to as subport mode, or sometimes as quad OC-48c mode.

A single ETT1 switch can have some ports in ‘normal’ mode, and some in subport mode; there is no restriction. The four levels of prioritized best-effort traffic are available in all configurations. However, there are some important differences between the two modes. One difference is that the LCS header must now identify the subport associated with each cell. Two bits in the LCS label fields are used for this purpose, as described below.

A second difference is that the EPP must carefully manage the output rate of each subport, so as not to overflow the buffers at the destination linecard, and this is also described below.

A third difference is that the EPP must maintain separate LCS request counters for each of the subports. It must also maintain separate egress queues. So the number of queues can increase four-fold in order to preserve the independence of each subport. Section 1.3 “Prioritized Best-Effort Queue Model” on page 30 describes the various queuing models in great detail.

1.2.6.1 Identifying the Source Subport

The EPP manages a single physical stream of cells at 25M cells/second. In subport mode the EPP must look at each incoming cell and determine which subport has sent the cell. The LCS label field is used to achieve this. Two bits within the label (referred to in the LCS Specification as MUX bits) are used to denote the source subport, numbered 0 through 3. The MUX bits must be inserted in the LCS label before the cell arrives at the EPP. The MUX bits might be inserted at the source linecards themselves. Alternatively, they might be inserted by a four-to-one multiplexer device placed between the subport linecards and the EPP. In this latter case, the multiplexer device might not be able to re-calculate the LCS CRC. For maximum flexibility, the EPP can be configured to calculate the LCS CRC either with or without the MUX bits.

1.2.6.2 Egress Cell Rate

Within the EPP is an Output Scheduler process which is different from, and should not be confused with, the ETT1 Scheduler device. At every OC-192 cell time, the Output Scheduler looks at the egress queues and decides which cell should be forwarded to the attached egress linecard(s). In subport mode, the Output Scheduler will constrain the egress cell rate so as not to overflow any of the 2.5 Gbit/s links. It does this by operating in a strict round-robin mode, so that at any OC-192 cell time it will only try to send a cell for one of the subports. The four 2.5 Gbit/s subports are labeled as 0 through 3 ; at some cell time the Output Scheduler will only try to send a cell from the egress queues associated with subport 0. In the next time, it will only consider cells destined for subport 1, etc. If, at any cell time, there are no cells to be sent to the selected subport, then an Idle (empty) cell is sent. For example, if subports 0 and 3 are connected to linecards but subports 2 and 4 are disconnected, then the Output Scheduler will send a cell to 0, then an empty cell, then a cell to 3, then an empty cell, and then repeat the sequence. The effective cell rate transmitted to each subport will not exceed 6.25 M cells per second.

1.2.7 LCS Control Packets

The LCS protocol provides in-band control packets. These packets (cells) are distinct from normal cell traffic in that they do not pass through the fabric to an egress linecard, but are intended to cause some effect within the switch.

There are two classes of Control Packets. The first class, referred to as CPU Control Packets, are exchanged between the linecard and the ETT1 CPU (via the EPP and Dataslices). The intention is that CPU Control Packets form the basic mechanism through which the linecard CPU and ETT1 CPU can exchange information. This simple mechanism is subject to cell loss, and so should be supplemented by some form of reliable transport protocol that would operate within the ETT1 CPU and the linecards.

The second class, referred to as LCS Control Packets, are used to manage the link between the linecard and the ETT1 port. These LCS Control Packets can be used to start and stop the flow of cells on the link,

to provide TDM synchronization event information, and can recover from any grant/credit information that is lost if cells are corrupted in transmission.

1.2.7.1 Sending a Control Packet from the OOB to the Linecard

Before sending a CPU (OOB to linecard) control packet, the OOB must first write the control packet header and payload data into the appropriate locations in the Dataslice. (See Section 3.3 “Output Dataslice Queue Memory Allocation with EPP”, Table 23, on page 164.) The header for customer-specific CPU control packets should be written into the Dataslices as shown, but the payload data is completely up to the customer.

To send the CPU control packet which has been written into Dataslice queue memory, the OOB writes the ESOBLCP register with the control packet type select in bits [5:4] (see the bit-breakout), and a linecard fanout in bits [3:0]. If the port is connected to 4 subport linecards, then bits [3:0] are a subport-bitmap. If the port is connected to one OC-192c linecard, then bit 0 must be set when the OOB wishes to send a CP.

When the CP has been sent to the linecard(s) indicated in bits [3:0], bits [3:0] will read back as 0. Since control packets have higher priority than any other traffic type, they will be sent immediately, unless the EPP is programmed to send only idle cells.

1.2.7.2 Sending a Control Packet From the Linecard to the OOB

The linecard sends control packets to OOB using the regular LCS request/grant/cell mechanism. A CPU (linecard to OOB) control packet must have the CPU bit set in its request label (See Section 1.1.3 “The LCS Protocol” on page 17.) When the EPP receives the cell payload for a CPU control packet, it stores the cell in the Dataslices’ Input Queue memories and raises the “Received LC2OOB/CPU Control Packet from Linecard...” interrupt. (In OC-192c mode, it will always be Linecard 0.)

The input queue for CPU control packets from each linecard is only 8 cells deep, so as soon as the OOB sees a “Received LC2OOB...” interrupt, it should read the appropriate “Subport* to OOB FIFO Status” register (0x80..0x8c). Bit [3:0] of that register will tell how many control packet cells are currently in the input queue; bits 6:4 will tell the offset of the head the 8-cell CPU CP input queue. That queue offset should be used to form addresses for the Dataslices’ Input Queue memories. See Section 3.2 “Input Dataslice Queue Memory Allocation with EPP” on page 162, for Dataslice Input Queue memory addressing. Then the OOB should read those addresses to obtain the CPU CP payload data. When the OOB has read the CPU CP payload data, it should write the appropriate “Linecard * to OOB FIFO Status” register (any value). A write to that register, regardless of the write data, will cause the head of the queue to be dequeued, freeing up that space in the CPU CP input queue.

See Section 1.6.3 “Control Packets” on page 69 for more details.

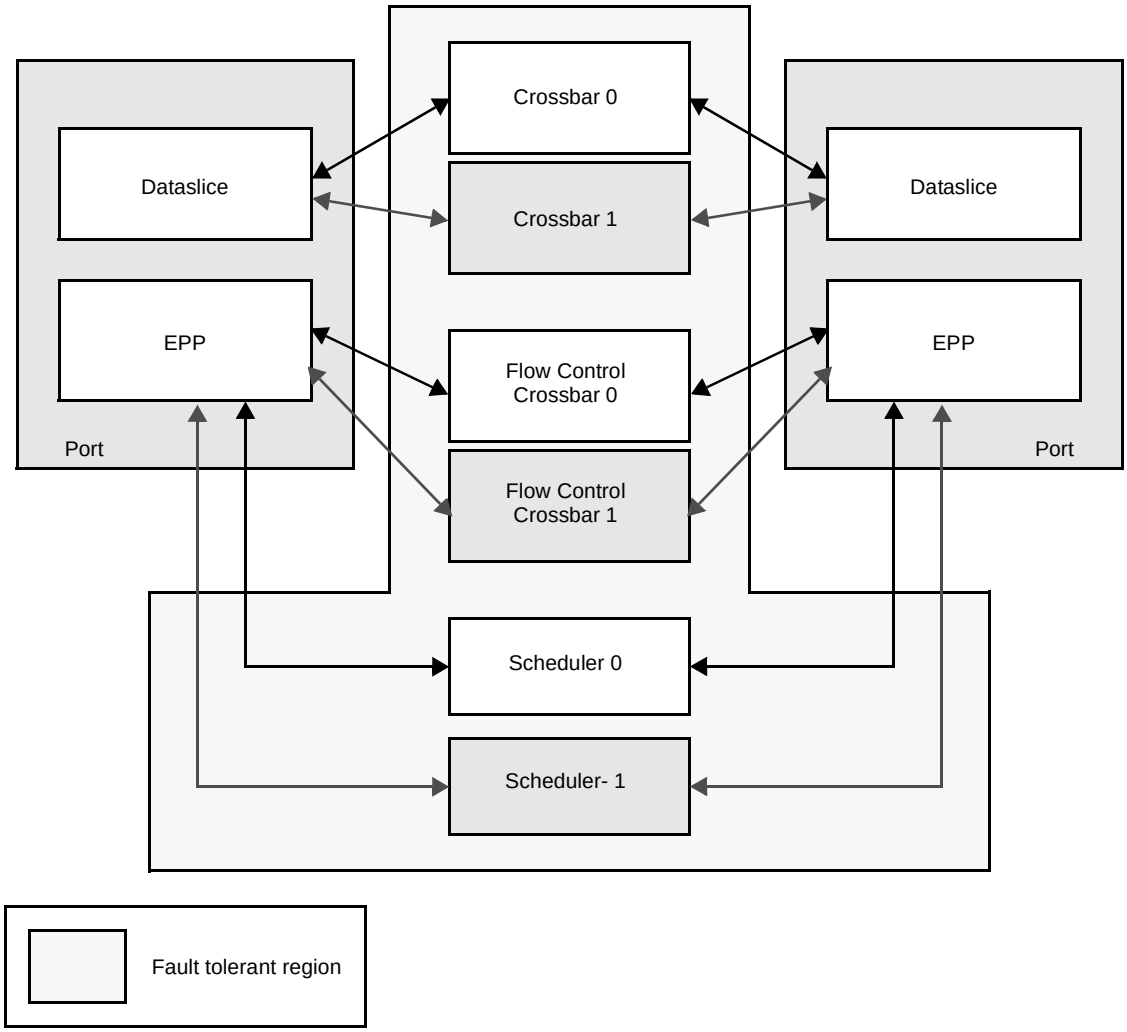
1.2.8 Redundancy

An ETT1 core can be configured with certain redundant (duplicated) elements. A fully redundant core is capable of sustaining single errors within any *shared* device without losing or re-ordering any cells. This section describes the main aspects of a redundant core. A complete switch may have two ETT1 cores,

with linecards dual-homed to both cores, and while this is a valid configuration the ETT1 core does not provide any specific support for such a configuration.

Figure 6 shows a fully redundant core. The Crossbars and Scheduler are fully replicated, as are the links connecting those devices. The port devices (EPP and Dataslices) are *not* replicated. It is important to understand that if an EPP or Dataslice fails (or incurs a transient internal error), then data may be lost, but only within that port.

Figure 6. Fully Redundant Core



A fully redundant core operates in exactly the same way as a non-redundant core. Consider the EPP: it sends new request information to both Schedulers. The two Schedulers operate synchronously, producing *identical* outputs, and the information (grants) received by the EPP will be the same. The Dataslices operate in exactly the same way with their two Crossbar devices.

The information carried on the links is protected by checksums. For example, if the EPP receives information with a checksum error on one link, then it simply ignores that link and uses the information from the other link. All errors are reported to the local CPU via interrupts.

The redundant core is tolerant of such single errors, provided that they are detectable. In general, all errors that occur on the links or within the Scheduler or Crossbar will be detectable.

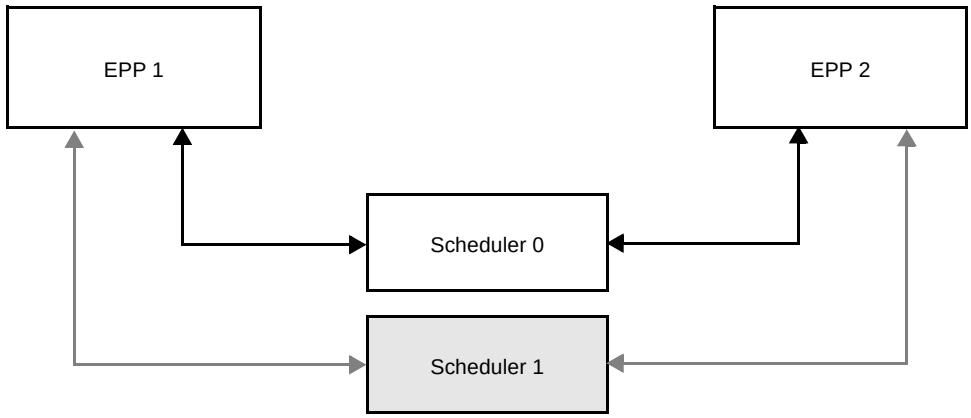
It should be clear from Figure 6 that only the Dataslice and EPP have the redundant connections. An individual Crossbar or Scheduler cannot compensate for information that is received with a checksum error. The following discussion reviews how these two types of devices are affected by receiving corrupted information.

If a Crossbar detects a checksum error then it simply marks the cell as being corrupted. This information is passed to the egress Dataslice which will ignore that link and use the information from the other Crossbar.

The Scheduler is more complicated because it must maintain accurate state information on the occupancy of all queues as well as the backpressure status of some of the egress queues. If a Scheduler receives a checksum error then it must effectively remove itself from the system.

Consider the simple configuration illustrated in Figure 7. Scheduler 0 sees a checksum error on information it receives from EPP 1. Scheduler 0 now has incorrect state information. It immediately disables all of its outbound links so that EPP 1 and EPP 2 know that Scheduler 0 should be ignored. The EPPs now only accept information from Scheduler 1.

Figure 7. Simple Redundant Scheduler Configuration



If a new Scheduler-0 board is inserted in the system, then it must have its internal state updated to match that of Scheduler-1. This is done using a sequence of actions that are controlled by the ETT1 CPU. In essence, the linecards are temporarily stopped from sending new cells, and the state within each EPP is downloaded into the Schedulers, and then the linecards are restarted. The entire process is very rapid (much less than 1ms), but does cause best-effort traffic to be suspended for a brief time.

1.2.9 System Configuration Options

Most of the previous sections have assumed a particular switch configuration consisting of 32 ports of OC-192c, with 64 byte payload cells and full redundancy. The ETT1 Chip Set has four aspects that can be configured according to the user's requirements. Two of these aspects have been described: quad OC48c versus single OC-192c port, and a redundant system versus a non-redundant system. The other two aspects are described in this section.

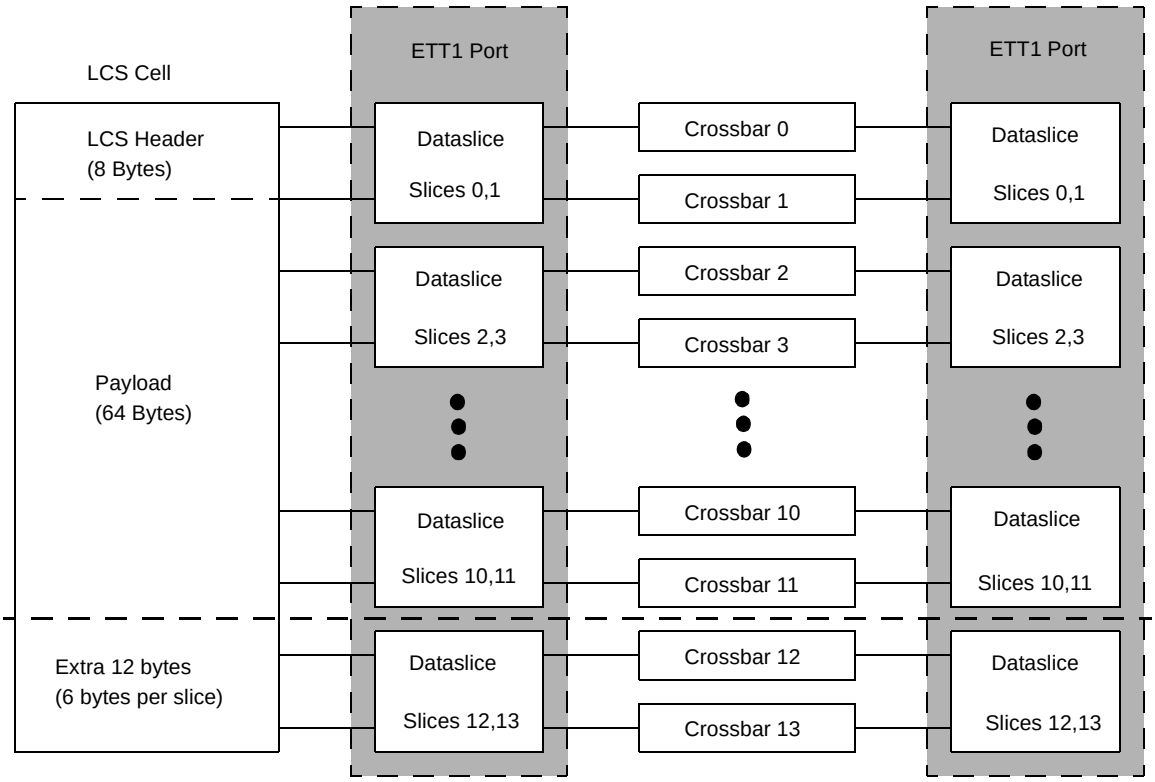
NOTE: All four aspects are orthogonal and so the choice of any one particular aspect does not limit the choices for the other three aspects.

1.2.9.1 Payload: 64 bytes or 76 bytes

The ETT1 core and LCS protocol are designed to forward fixed length cells. Each LCS cell consists of an LCS header (8 bytes) and a payload. The size of this payload can be either 64 bytes or 76 bytes. The choice of payload size is a function of the linecard traffic: 53-byte ATM cells can probably use a 64 byte payload; fragmented IP packets might obtain greater efficiency with a 76 byte payload.

This flexibility in cell size is due to the way the ETT1 Chip Set slices the cells. Figure 8 shows how an LCS cell is sliced across the Dataslices and Crossbars.

Figure 8. LCS Cell Sliced Across Dataslices and Crossbars



The 64 byte payload cell requires six Dataslices per port and 12 Crossbars (32 port, non-redundant). The 76 byte payload cell requires seven Dataslices per port and 14 Crossbars. The EPP is capable of supporting the seventh Dataslice.

Because the extra data is obtained by slicing the cell then this does not affect the clock frequency of the Chip Set. The only effect is that the link between the ETT1 port and the linecard must go up from 18 Gbaud to 21 Gbaud. This is done by making the link wider, not faster.

1.2.9.2 8, 16, or 32 ports

The ETT1 core can support a maximum of 32 ports. Smaller configurations can be supported by simply not using some of the ports. However, if the maximum system is an eight or 16 port configuration, then some

cost savings can be achieved by using fewer Crossbar devices. A Crossbar can be configured to be in an 8- 16- or 32-port core, as shown in Table 1.

Table 1. Crossbar Configurations

Number of OC-192c ports	Number of Crossbars required (non redundant)	Number of Crossbars required (redundant)
1 to 8	3 + 1 Flow Control	6 + 2 Flow Control
9 to 16	6 + 1 Flow Control	12 + 2 Flow Control
17 to 32	12 + 2 Flow Control	24 + 4 Flow Control

The reduction in the number of crossbars is achieved by having a single port use more than one connection on each Crossbar. In a 32 port system, each port will use one connection to each Crossbar device; in an 8 port system, each port will use four connections on each Crossbar. (This DOES NOT apply to Flow Control Crossbars; each port will always use only one connection.)

1.3 PRIORITIZED BEST-EFFORT QUEUE MODEL

The ETT1 switch core has been designed to maximize cell throughput while keeping latency small and providing “fair” throughput among all ports. The ETT1 core supports four levels of strict priority. Level 0 is the highest priority and level 3 the lowest. All conditions being equal, the core will always forward a higher priority cell before a lower priority cell. Furthermore, the core will be fair to all inputs. This fairness is very difficult to quantify and is only meaningful over time periods of many cells. Over short time periods (tens or perhaps hundreds of cells) there may be temporary unfairness between ports.

A cell of a lower priority may pass a cell of a higher priority under certain conditions:

1. Since the scheduling pipeline is shorter for lower priorities, a cell of a lower priority may emerge from the switch sooner than a cell of a higher priority if the lower-priority cell is sent immediately after the higher-priority cell.
2. Hole requests for a higher priority can allow cells of a lower priority to pass cells of the higher priority.
3. If the linecard responds to LCS grants faster than required to meet the system round trip time constraint, then multicast cells of a lower priority may pass unicast cells of a higher priority, due to a (programmable) delay in the EPP’s “Internal Delay Matching Adjustments” register.
4. In subport mode only: multicast output queues (VIQs) are shared by subports, so when a subport is oversubscribed for an extended period of time, blocking may result on all subports. This blocking of multicast can allow unicast cells of the same priority to pass.

In this section we describe the queueing model provided by the ETT1 switch core for this prioritized best-effort traffic. We start by considering only OC-192c ports. The following sections describe how the model differs for OC-48c ports.

1.3.1 Unicast Traffic (OC-192)

Every ETT1 port has a separate request counter and ingress queue for each (output port, priority) combination to which it can send unicast cells. In a full configuration, an ETT1 port can send to 32 ETT1 ports (including itself), at four priorities, and so will have 128 unicast request counters and ingress queues. All of these counters and queues are independent so that the switch does not suffer from head-of-line blocking[1].

Figure 9 illustrates the unicast ingress queueing model for one port.

Figure 9. The Unicast Ingress Queueing Model for One Port

The ingress queues are referred to as Virtual Output Queues (VOQs). This name indicates that each queue holds cells going to just one output port.

Every ETT1 port also has the same number of unicast egress queues; one queue for every (input port, priority) combination that can send cells to the port. Every port will have 128 unicast egress queues. It is not acceptable to have a single egress queue per priority, as this can lead to gross unfairness between ports. These egress unicast ports are called Virtual Input Queues (VIQs) as each queue only holds cells that have come from a single input port. A useful model is to think of a VOQ as being directly connected through the Crossbar to a single VIQ. The cell at the head of the VOQ is waiting to be scheduled in such a way that it can move to the tail of the VIQ in the egress port.

The associated ingress EPP is informed whenever an egress queue is full. In this case, the ingress port will not issue any new requests to the Scheduler until sufficient cell space becomes available at the egress queue. The transfer of cells from ingress to egress queues is lossless from a queueing perspective.

The best effort unicast ingress and egress queues can store up to 64 cells.

1.3.2 Multicast Traffic (OC-192)

The ETT1 core also supports multicast cells. Multicast cells are cells that must be forwarded to one or more egress ports. The LCS header will indicate that a cell is multicast, and it will also contain a multicast group identifier. The ETT1 port uses this identifier to determine the list of ports to which the cell should go. This list of ports is called the multicast fanout. Different multicast groups will probably have different fanouts.

The Scheduler arbitrates among unicast and multicast cells. A multicast cell of a given priority will always have priority over a unicast cell of the same priority, all else being equal. Physical cell replication takes place within the Crossbar.

1. Head -of-line blocking is a phenomenon encountered by input queued switches in the case where a cell destined for one output port is delayed because the cell at the head of the same FIFO queue is destined for some other output which is currently congested.

The Scheduler will not necessarily schedule all ports in a fanout at the same cell time. In other words, the same multicast cell might pass through the Crossbar on several occasions before all egress ports have received their copy of the cell.

Every ETT1 port has a single ingress queue and a single egress queue for multicast traffic at each priority, as shown in Figure 10. (There are also request counters, as for unicast cells, but these are not shown.)

Figure 10. Ingress and Egress Queue for Multicast Traffic

The Scheduler maintains information on the occupancy of the multicast ingress queues, and receives backpressure/unbackpressure signals for the multicast egress queues. If an egress multicast queue is backpressured then there is a question as to what happens to the cell at the head of an ingress queue that is trying to send to that backpressured egress queue (as well as other ports). If the Scheduler blocks the ingress queue, then the system will experience head-of-line blocking for multicast cells at that priority.

Every ETT1 port can select how it wants to handle multicast cells that are *sent* to it. For each of the four multicast egress queues, the local ETT1 CPU can either enable or disable a backpressure signal from being sent to the Scheduler.

NOTE: This selection is at the egress queues, *not* the ingress queues.

If a port is set so that multicast backpressure is disabled, then that port will never cause head-of-line blocking to occur in any ingress multicast queue that will send to it.

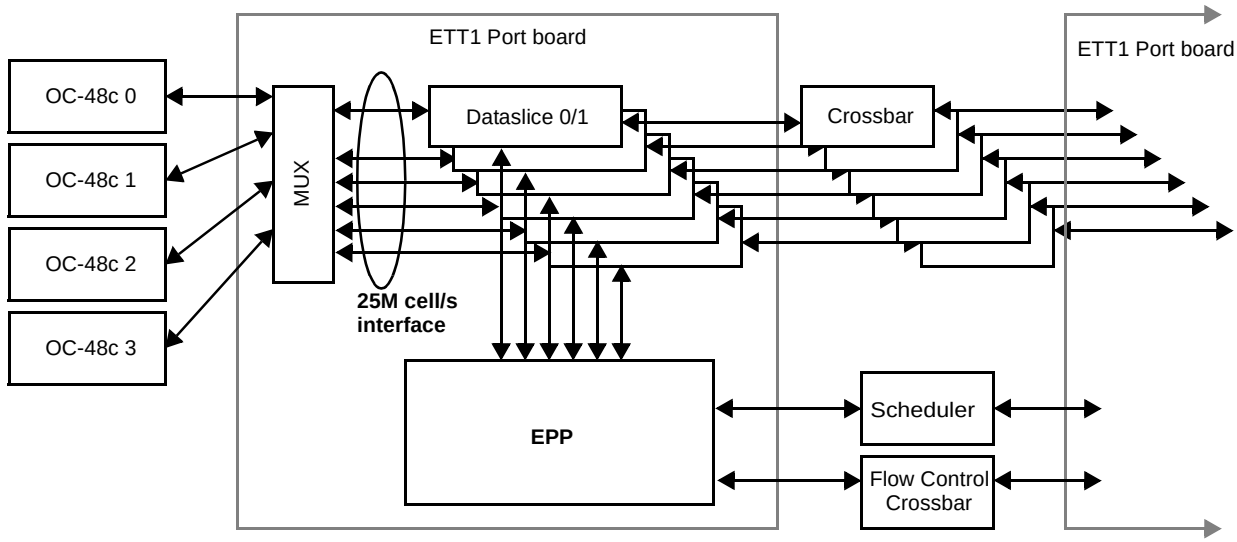
If a port is set so that multicast backpressure is enabled, then that port can assert a backpressure signal if the egress queue gets too full. Consequently, this *might* cause head-of-line blocking at multicast ingress queues that send cells to this port.

Therefore, head-of-line blocking is only guaranteed *not* to occur if *all* of the ports that can receive multicast cells have their multicast backpressure signals disabled. If at least one port has backpressure enabled, then head-of-line blocking *might* occur.

1.3.3 Unicast Traffic (Subport Mode)

The EPP can operate in one of two modes. The first mode uses the queueing model described previously, and corresponds to a single channel of approximately 10 Gbit/s. The second mode is called subport mode. In subport mode an EPP can manage four channels of 2.5 Gbit/s each. The physical interface at the Dataslice is always a single channel; in subport mode the four 2.5 Gbit/s channels are time multiplexed onto the single 10 Gbit/s channel. More precisely, given that the ETT1 Chip Set always operates at 25M cells per second, the subport mode correspond to four channels each of up to 6.25M cells per second. Figure 11. shows the block diagram of an ETT1 port card. The six (or seven) Dataslices provide a single interface. A separate MUX device is required to manage the four OC-48c channels and merge them so they appear as a single channel to the EPP/DS.

Figure 11. An ETT1 Port Operating in Subport Mode with Four OC-48c Linecards



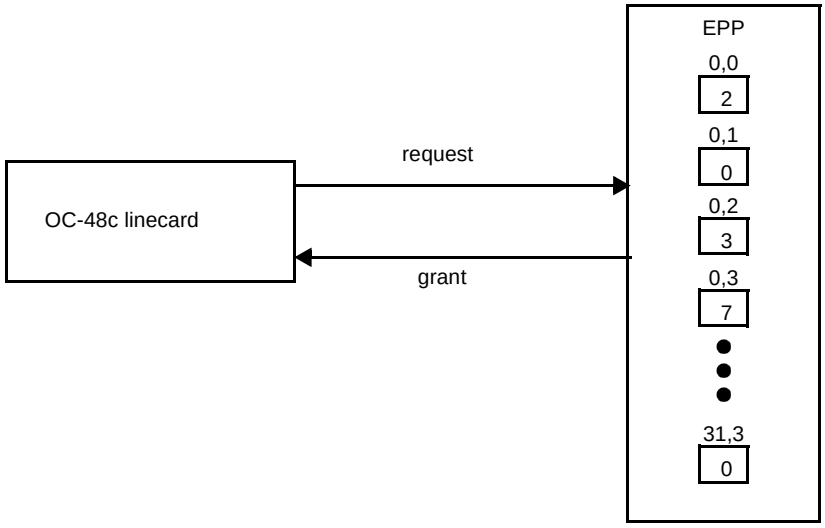
The EPP can support all four best-effort priorities when operating in subport mode. This is the primary feature that differentiates the EPP from the PP. Furthermore, the EPP can operate in a mixed mode switch, so an ETT1 switch can have some ports operating in subport mode (quad OC-48c), while other ports operate in normal mode (OC-192c).

NOTE: The EPP has been designed to operate with the same Dataslice device as the PP, and this has certain limitations that the designer should be aware of. The following sections on subporting explain these limitations.

Ports are numbered 0 through 31and each port has one EPP. Subports are denoted as 0 through 3. Ports with subports are shown as a (port, subport) pair; for example, port 4, subport 3 would be shown as (4,3) Ports without subports (OC-192c) are simply numbered. The following queueing description assumes an ETT1 core configured with all 32 ports in subport mode, supporting 128 OC-48c linecards.

First, consider a single priority. Figure 12 shows the 128 counters needed by one OC-48c linecard. There are currently two requests outstanding for output (0,0). The ETT1 Chip Set uses virtual output queues to avoid head-of-line blocking issues, therefore, the EPP needs to keep track of unicast requests going to each one of the 128 possible output channels (0,0) through (31,3). Consequently, 128 counters are needed. These counters reflect the outstanding *requests* that this one OC-48c ingress linecard has made to the switch.

Figure 12. The Input Request Counters

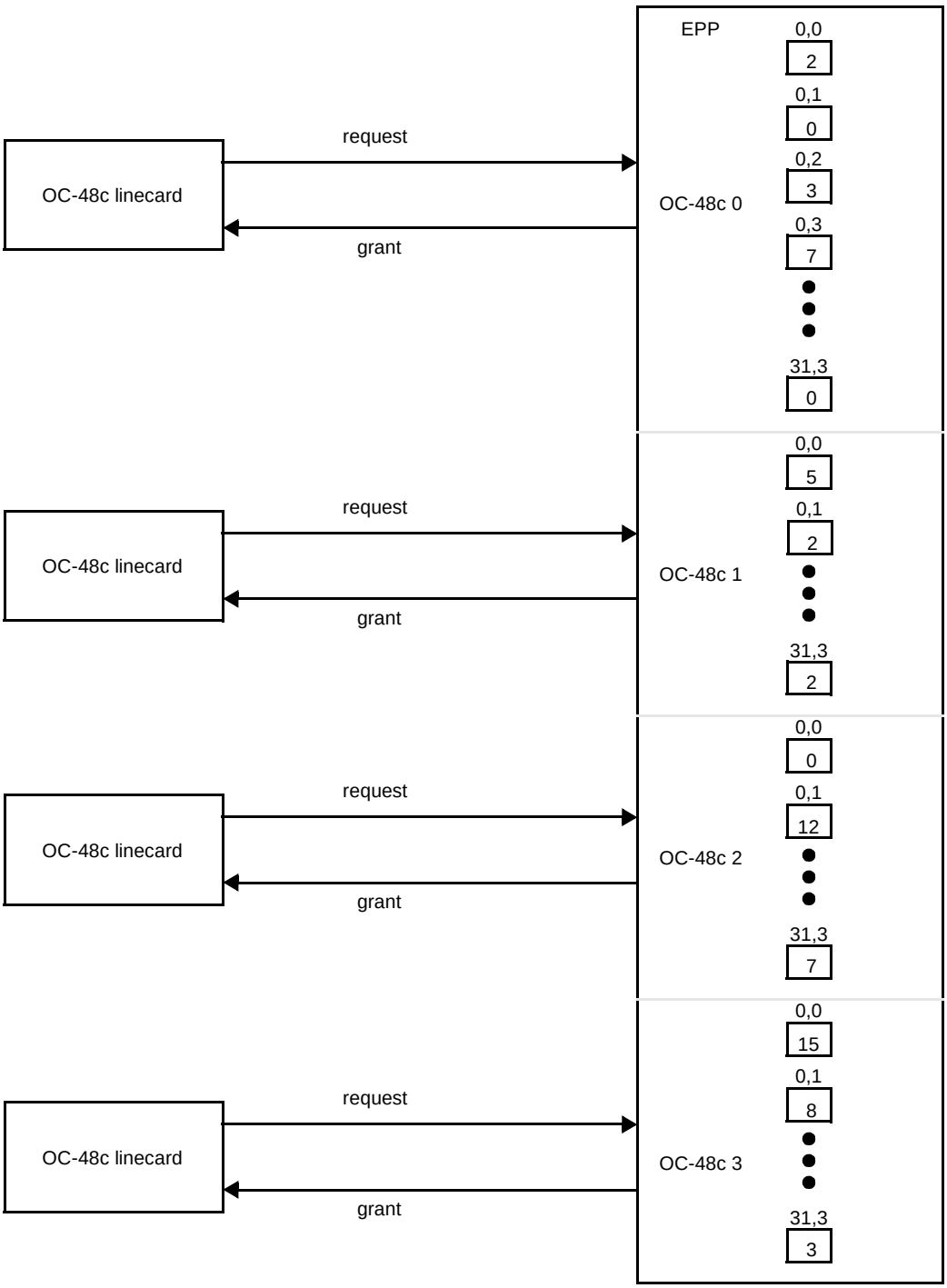


The EPP can support up to four OC-48c channels, and each channel must have its own set of request counters for each of the 128 OC-48c output channels. Figure 13 shows the full set of counters for a single priority.

Within the EPP an LCS Grant Manager process issues grants to the linecards in response to the received requests. Once per cell time the LCS Grant Manager selects one request counter to grant, given that many may qualify. A request counter qualifies to receive a grant if the counter is non-zero and if there is at least one empty cell buffer in the VOQ that is used for cells going to that particular output. Once the LCS Grant Manager has selected a particular VOQ, it decrements the corresponding counter and sends a grant to the appropriate linecard.

The grant indicates a specific VOQ within the linecard. When the linecard receives the grant it does two things. First, it sends to the EPP the cell body that is currently at the head of the granted queue. Second, it increments a local credit counter that is used to determine whether a new request for that queue can be sent.

Figure 13. Full Set of Counters for a Single Priority



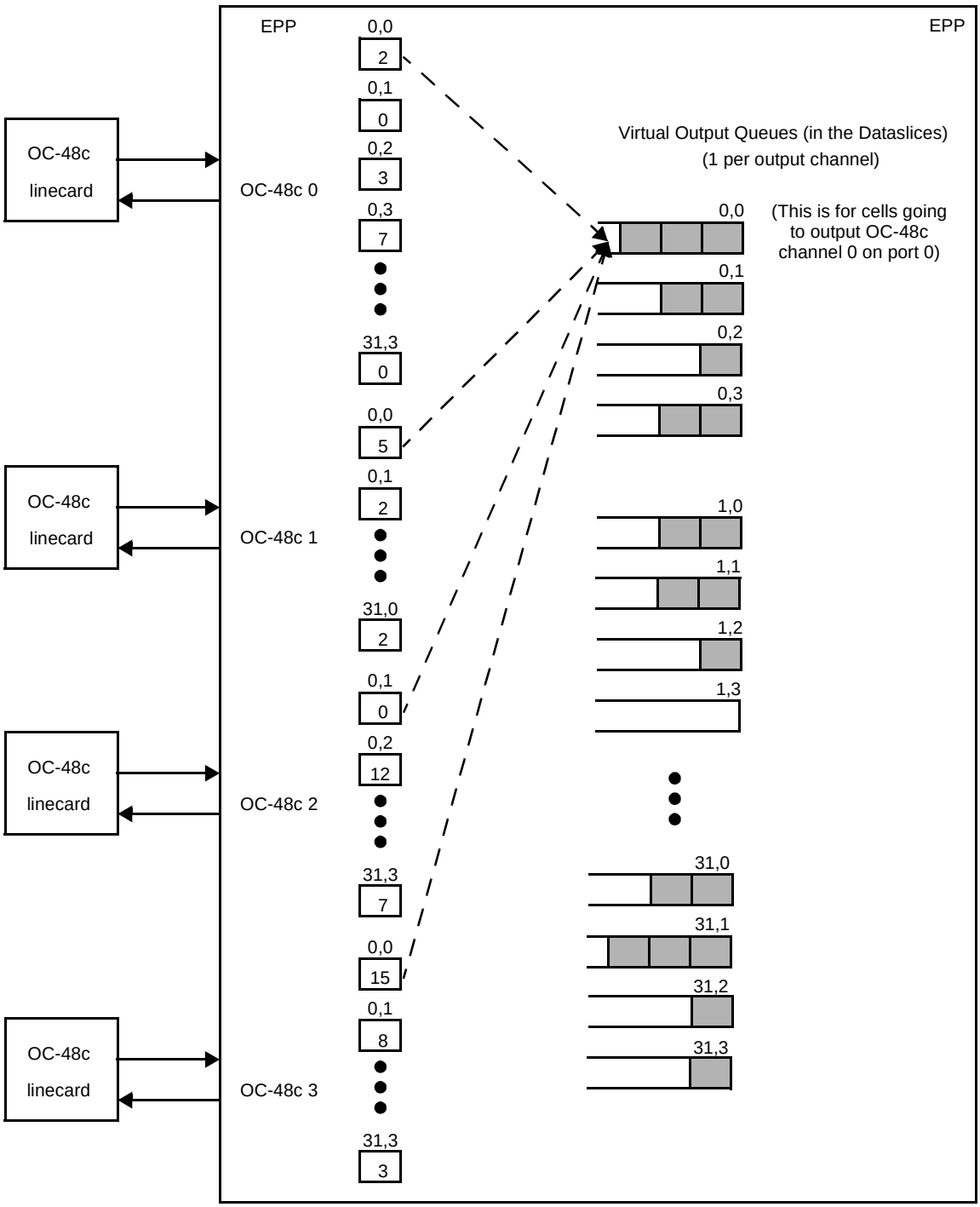
1.3.3.1 Virtual Output Queues

Ideally, the EPP/DS would have a separate VOQ which would correspond to each of the request counters. This would mean that congestion on one VOQ would not impact any other VOQ on the same OC-48c or any VOQs on different OC-48c channels.

In practice, the EPP is constrained by the amount of cell buffering available in the ETT1 Dataslice device (which contains the actual cell buffers). Consequently, all four request counters that correspond to a single output channel share a single queue, as shown in Figure 14.

The EPP only manages a single queue for each of the output channels (whether the output channel is OC-48c or OC-192c). If the EPP is connected to four OC-48c inputs, then those four OC-48c linecards must share the single queue used for each output. The four dotted arrows in Figure 14 show that the four request counters for output (0,0) all map to the same VOQ for output (0,0).

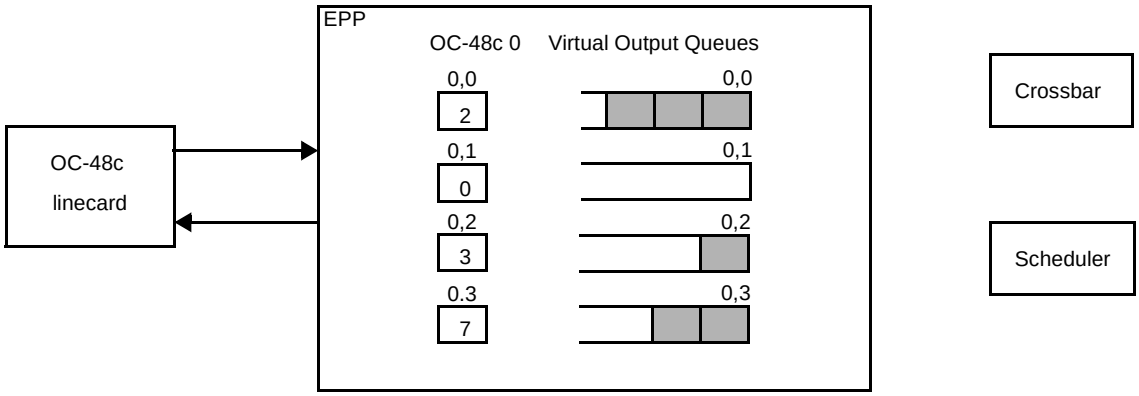
Figure 14. All Request Counters for a Single Output Share a Single Queue



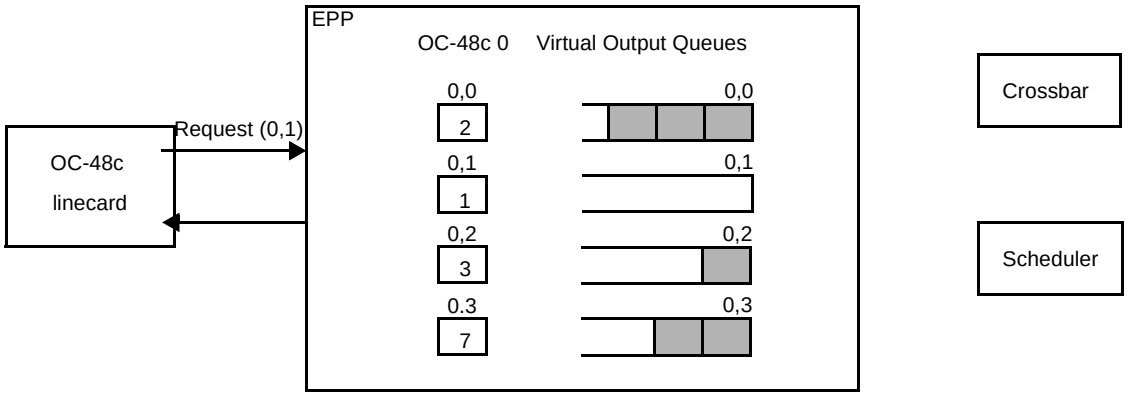
The complete sequence of events at the input side is shown in Figure 15.

1. Linecard a issues a request to the EPP to send a cell to output channel (0,1). The initial request count is zero, so the arrival of the request then increments the request count to one.
2. The LCS Grant Manager sees that there is at least one empty cell buffer in the virtual output queue for output (0,1), and that the request counter for (0,1) is non-zero and so it decrements the request counter (back to zero) and issues a grant to the linecard.
3. The linecard then forwards the actual cell body to the EPP which stores the cell in the virtual output queue. The EPP also issues a request to the Scheduler, indicating that there is a cell waiting to be transferred to output (0).
4. Later the Scheduler issues a grant to the EPP which can forward the cell through the Crossbar to the output channel (the EPP at port 0).

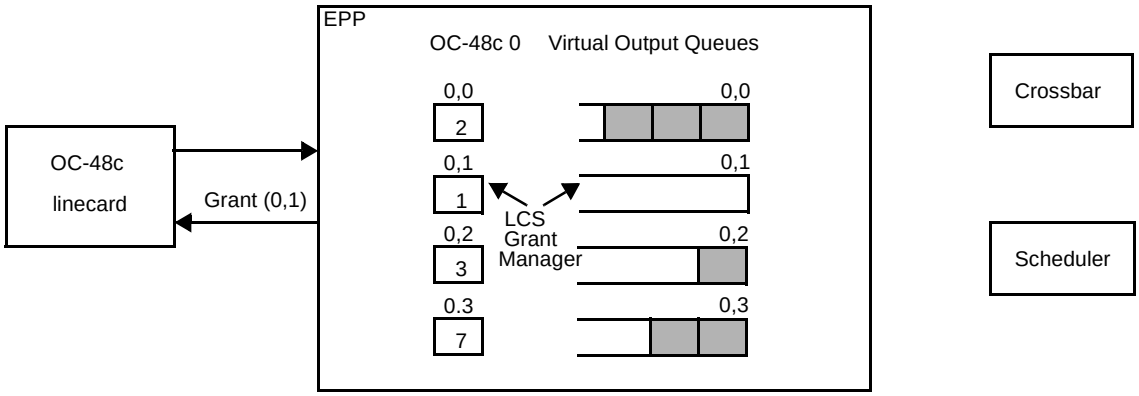
Figure 15. Sequence of Events at the Input Side



Initial state. Subport 0 has no outstanding requests for output (0,1)

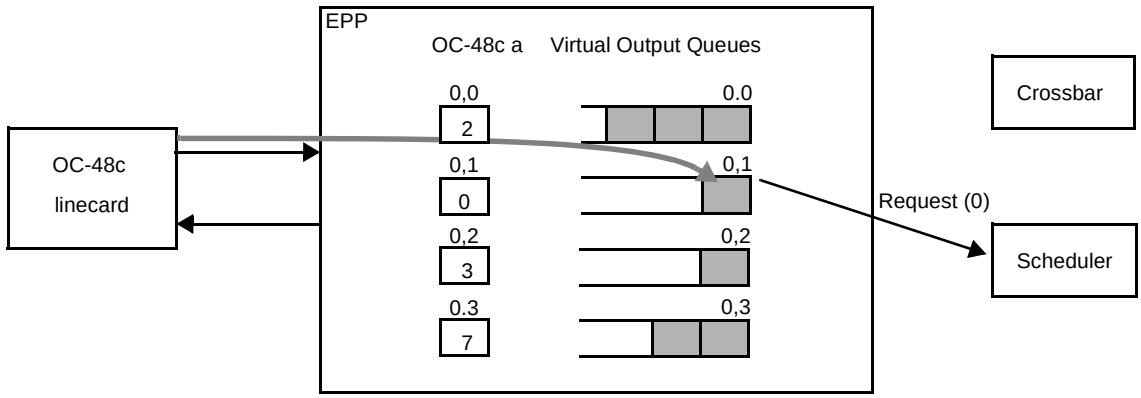


Subport 0 issues a request to send a cell to output (0,1).
The request count for output (0,1) is incremented.

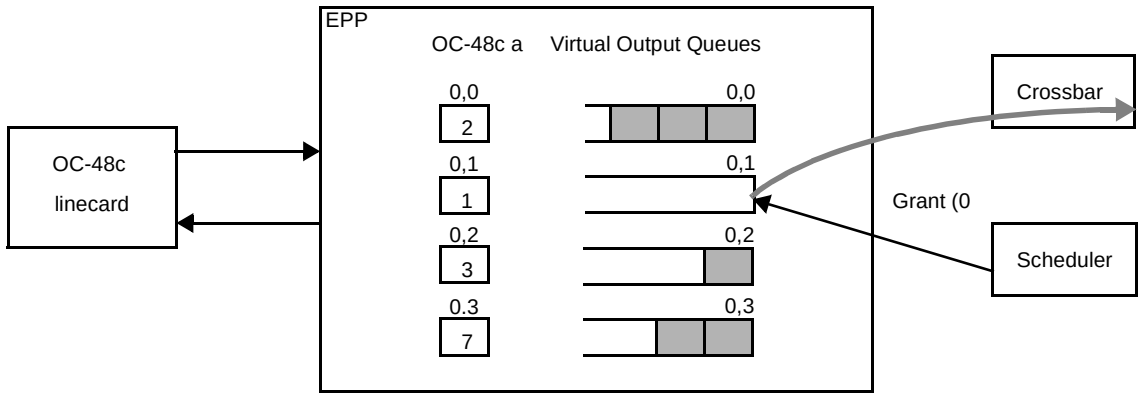


The LCS Grant Manager issues a grant for queue (0,1) to subport 0.

Figure 15. (Continued)



Linecard a receives the grant and forwards cell body to go to output (0,1).
A request is made to the Scheduler.



The Scheduler issues a grant to queue (0,1).
The EPP then sends the cell through the crossbar to the output queue of port 0

The departure of a cell from the VOQ will mean that there is now space for at least one more cell in that VOQ, and that could trigger the LCS Grant Manager to issue a new grant, assuming that a new request had come in for that queue.

1.3.3.2 The Output Process

Each EPP consists of an input (i)EPP and an output (o)EPP. The request counters and virtual output queues described above all reside in the iEPP. The oEPP has virtual input queues that receive the cells forwarded via the Crossbar.

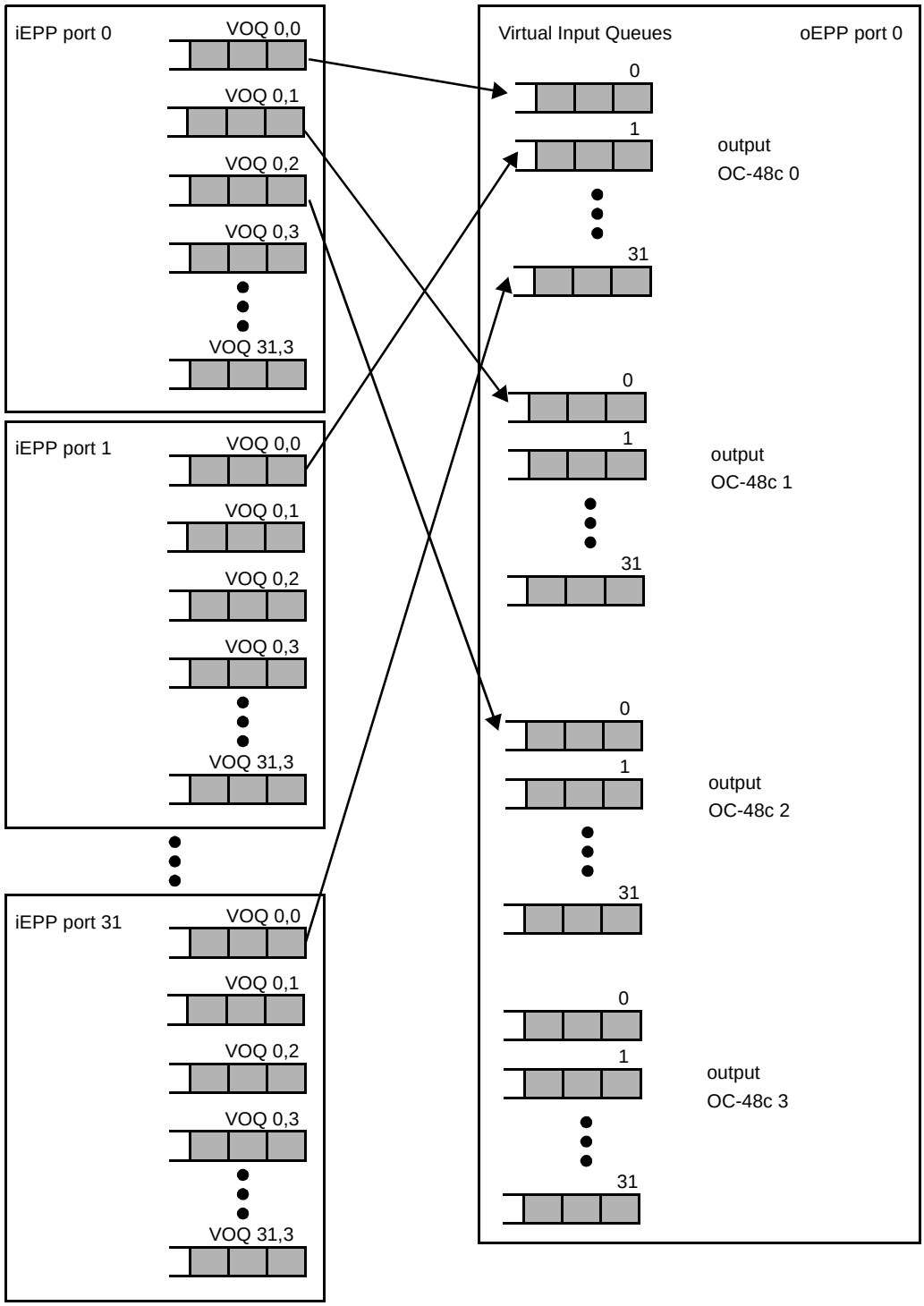
In the ideal scenario, each oEPP would have a separate virtual input queue for every (possibly 128) input channel. However, the same restriction applies, limiting the oEPP to just 32 queues per output OC-48c (and per priority). Figure 16 shows the virtual input queues in each oEPP when attached to four OC-48c ports.

Each virtual input queue maps back to a single virtual output queue in one of the iEPPs. In abstract terms, the virtual input queue is simply an extension of the appropriate virtual output queue.

Each output OC-48c channel within the oEPP has its own set of unicast virtual input queues. If an output queue (virtual input queue) is backpressured, then this will push back to the appropriate virtual output queue. However, each virtual output queue is shared by the four input OC-48c channels within that iEPP, so backpressuring a virtual output queue has the effect of asserting backpressure to all four OC-48c's on the same iEPP.

The oEPP has an Output Scheduler process which determines which cell should be forwarded to the egress linecards.

Figure 16. Virtual Input Queues

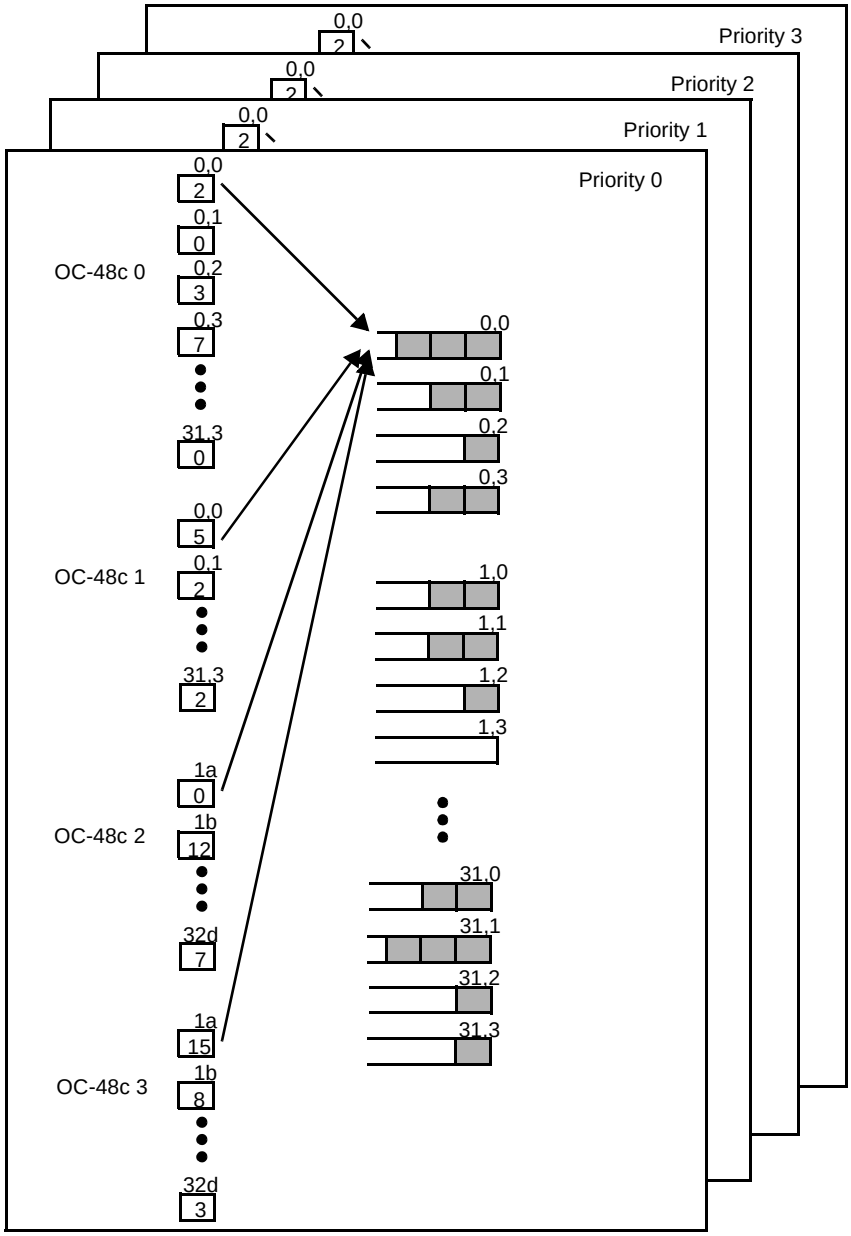


1.3.3.3 Four Priorities

The description in the preceding sections only considered a single priority. The EPP supports four priorities. Therefore, there are four times as many counters and queues as have been described so far , and this is only for unicast traffic. The four priorities can be modeled as separate planes; the input and output queue figures shown earlier can be considered as 2-D planes. The four priorities are then four copies of these planes, stacked next to each other in a third dimension. Figure 17 illustrates this model for the reference counters and virtual output queues. The virtual input queues at the oEPP are not shown.

Consequently, each iEPP has 2048 request counters (4 OC-48c input channels * 128 output channels * 4 priorities) and 512 virtual output queues (128 outputs * 4 priorities). Each oEPP has 512 virtual input queues (4 output channels * 32 ports * 4 priorities). Remember, this calculation is for the configuration of 128 ports of OC-48c. Later on we describe how the number of queues and request counters differs for some other possible configurations.

Figure 17. The Four Priorities



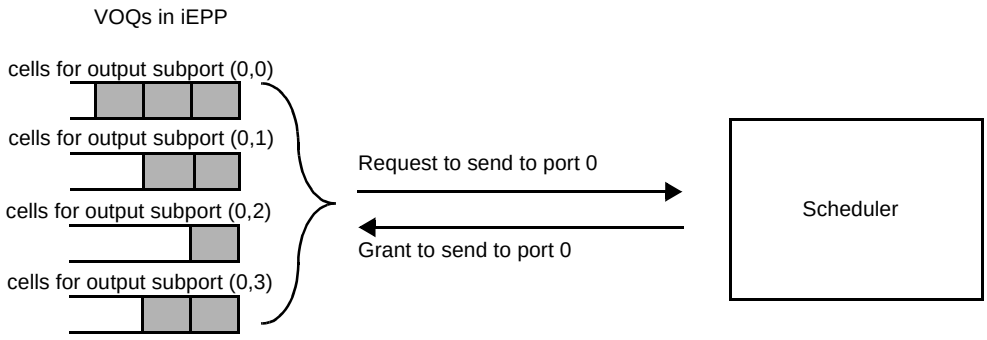
1.3.3.4 Multiplexing Scheduler Requests and Grants.

Figure 16 and Figure 17 imply that the ETT1 Scheduler can accept requests from each of the 32 ports where a request specifies one of 128 output channels. This is not the case; the Scheduler only maintains information on a port basis, not a subport basis.

Consequently, the requests associated with cells going to subports (0,0), (0,1), (0,2) and (0,3) are amalgamated to become requests for port 0.

NOTE: The channel information is not lost; a cell going to channel (0,1) will still be delivered to channel 1, port 0. However, the four queues have their requests multiplexed over a single request/grant connection. Figure 18 illustrates this concept.

Figure 18. Scheduler Requests/Grants are Multiplexed at a Port Level



The EPP has two decisions to make: which request to issue, and which of the four queues to grant.

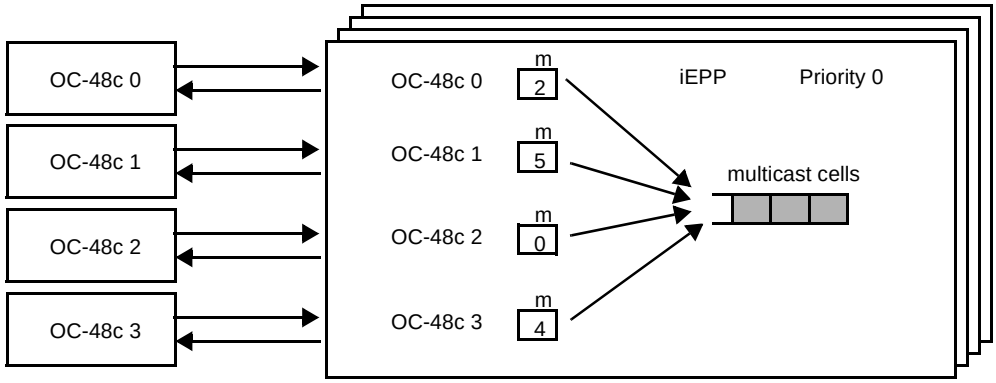
At any cell time, the iEPP may have several possible new requests which it must communicate to the Scheduler. The iEPP has a Scheduler Request Modulator which makes this decision. See Section 3.1.3 “Scheduler Request Modulator” on page 158 for a discussion of this algorithm.

When the iEPP receives a grant from the Scheduler it must decide which of the four VOQs should be serviced. A simple round-robin algorithm is used (across non-empty queues).

1.3.4 Multicast Traffic (subport mode)

Multicast cells are handled at a port level, not a subport level. There is a single cell queue within the iEPP, for each priority of multicast cells. Each OC-48c channel has its own request counter, but, like the unicasts, these four request counters map to a single virtual output queue (per priority), as shown in Figure 19.

Figure 19. Each iEPP has a Single Virtual Output Queue for Multicast Cells

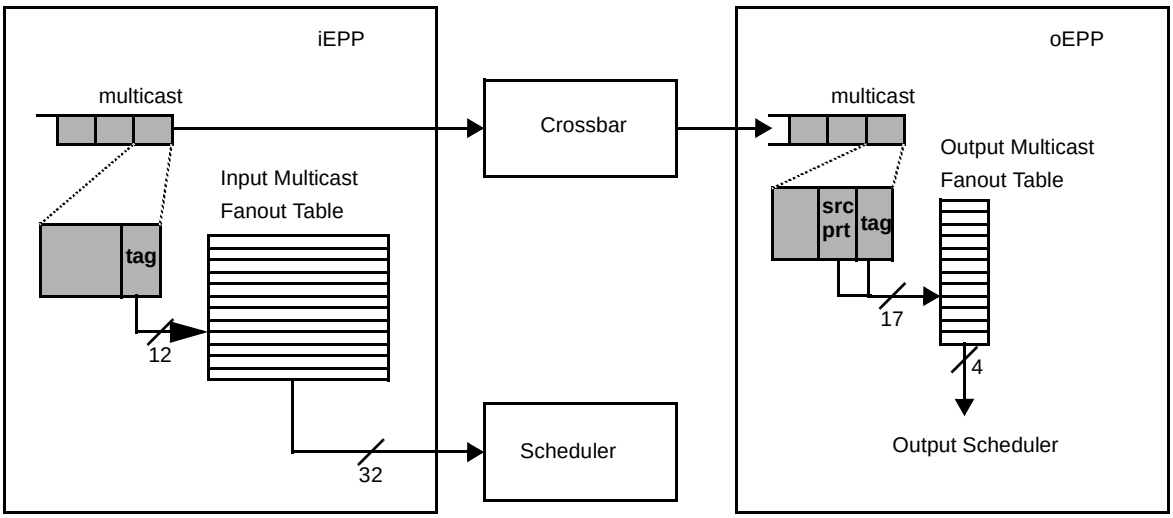


As before, this single queue is not a limitation unless backpressure is asserted to the queue, in which case all four OC-48c subports are backpressured for all multicast cells from this port at this priority.

ETT1 uses a multicast tag in the LCS header to identify the multicast group that this cell should go to. The iEPP then maps the tag into a port vector with one bit per port. This port vector is then passed to the Scheduler to indicate the list of ports that should receive this multicast cell. Again, the Scheduler only recognizes 32 ports and not 128 OC-48c subports; at the input side, a multicast cell destined for any one of output ports (0,0), (0,1), (0,2) or (0,3) would simply be forwarded to the oEPP at port 0.

The oEPP receives the multicast cell, together with its tag. It then maps the (tag, source_port) into a new 4-bit vector that identifies what combination of the four subports should receive this multicast cell. Figure 20 shows the entire process.

Figure 20. Multicast Cell Processing



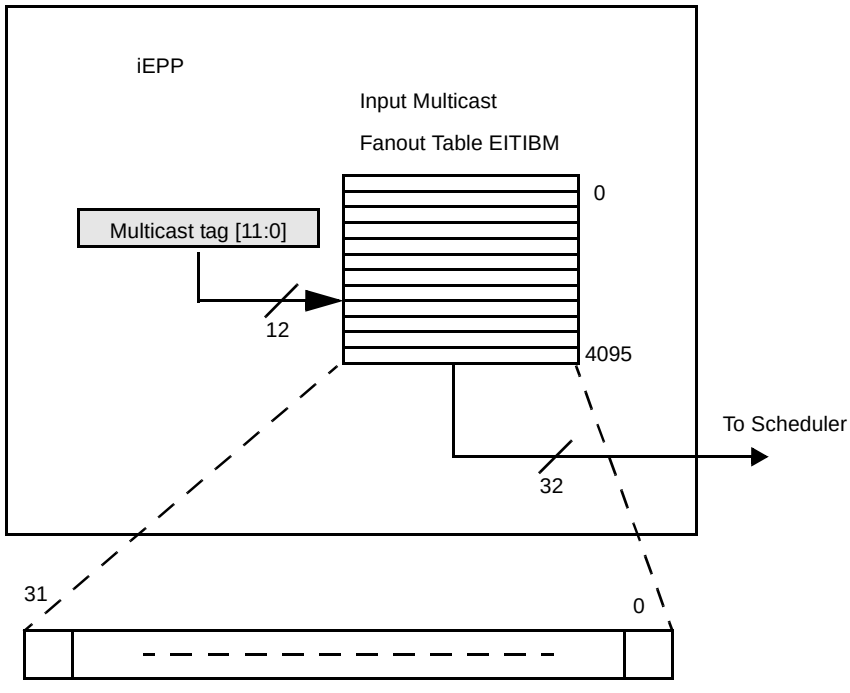
In order to reduce the blocking effects of one OC-48c channel over another channel on the same port, there is a special multicast output queue for each priority. The multicast cells forwarded through the Crossbar are still queued in the order they arrive. This special queue enables cells to be sent out-of-order, but will always be sent in-order for a given egress subport. Therefore, a multicast cell waiting in the output queue memory can only be blocked by cells destined to the same egress subport. Additional output queue memory becomes available whenever a cell is dequeued.

If backpressure is asserted by one of the OC-48c channels to the output multicast queue, then subsequent cells going to other channels on the same port may be blocked if the *entire queue* is occupied by cells destined to the blocked OC-48c channel. This is a consequence of sharing the output queue memory allocation for multicast cells at a given priority. This can cause multicast backpressure to the central Scheduler which can block multicast cells destined to other ports at the same priority if multicast backpressure is enabled.

1.3.4.1 Multicast Tables (OC-192c and Quad OC-48c Modes)

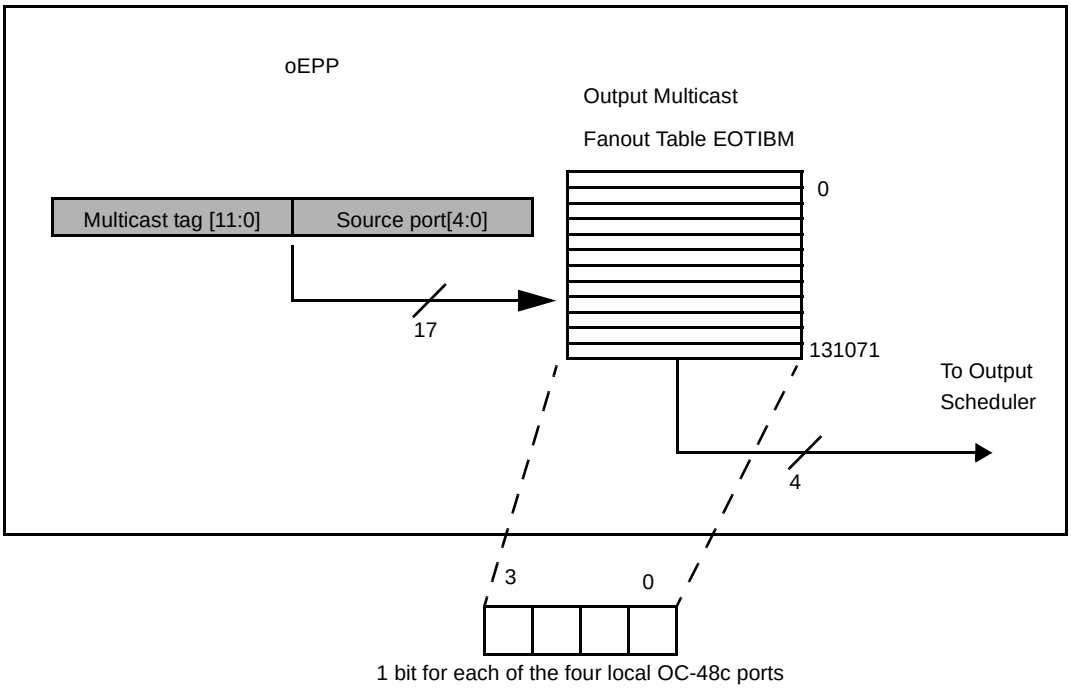
The input multicast fanout table (EITIBM in the EPP) is always used for multicast cells. Every EPP has its own EITIBM table with 4096 entries. The EPP uses the 12-bit multicast tag field from the LCS label field to determine which entry in the table should be used for each multicast cell, as shown in Figure 21. Each entry is simply a 32 bit vector where a 1 means that the cell should go to that port. So if an entry has a 1 at bit 12, for example, then a multicast cell that uses that entry will be sent to port 12 (as well as any other ports that have their bit set to 1).

Figure 21. EITIBM structure



The output multicast fanout table (EOTIBM in the EPP) is only used if the port is in quad OC-48c mode. In quad OC-48c mode a multicast cell arrives at the oEPP via the crossbar. The oEPP must determine what combination of the four OC-48c ports that this multicast cell should be forwarded to (15 possible combinations assuming it goes to at least one of the ports). The oEPP has the multicast tag (12 bits) from the label, but that is not sufficient. It is not sufficient because the multicast tags are managed independently across all ports: each iEPP has its own EITIBM and so a given 12 bit multicast tag might be used to specify different multicast fanouts for two different iEPPs. To resolve the ambiguity, the oEPP must also include the source port number with the 12 bit multicast tag, and combines the 5 bit source port with the 12 bit multicast tag to produce a 17 bit tag that is unique across the entire fabric. This can then be used to index into a table of 4-bit entries as shown in Figure 22.

Figure 22. OITIBM structure



The index is the combination of {Multicast_tag, source_port}, so that the source port forms the least significant bits of the address (index). Each entry has four bits such that bit 0 corresponds to subport 0 and bit 3 corresponds to subport 3.

1.3.5 Combinations of OC-192c and Quad OC-48c Linecards

A switch might contain both OC-192c ports and quad OC-48c ports. Each port card must be configured with information about all of the cards. The number of ingress queues is a function of this mix of ports: given n quad OC-48c ports and m OC-192c ports, then each port will have U unicast queues, where:

U = 4 priorities * n * 4 egress subports + 4 priorities * m

So if n = 32 and m = 0 then U = 512.

If a port is operating in OC-192c mode then it will also have U unicast request counters. If a port is operating in quad OC-48c mode then it will have 4 * U unicast request counters - a separate set of request counters for each of the OC-48c linecards.

Each port always has four multicast ingress queues.

The number of egress queues is a function of the mode of the port itself, not the other ports: a port in OC-192c mode will have one queue for each input port and priority, or 4 * 32 = 128 egress queues. A port in quad OC-48c mode will have four times as many queues, or 512 queues.

Each port always has four multicast egress queues.

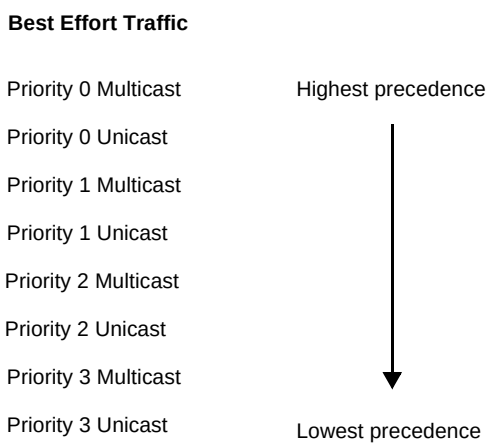
The size and depth of all request counters and queues are defined in Section 3.2 “Input Dataslice Queue Memory Allocation with EPP” on page 162 and Section 3.3 “Output Dataslice Queue Memory Allocation with EPP” on page 162.

1.3.6 Summary

Every ETT1 port has a number of ingress request counters and queues as well as egress queues. Given support for 32 ports, four subports per port, and four priorities, then there are up to 512 unicast ingress queues and four multicast ingress queues, and the same number of egress queues.

At every cell time, the Scheduler arbitrates among all of the ingress queues that are eligible to forward a cell through the Crossbar. An ingress queue is eligible if it is non-empty and its destination egress queue is not backpressured. The Scheduler makes its decision using absolute priorities, with multicast requests having precedence over unicast requests at the same level, as shown in Figure 23.

Figure 23. Scheduler Priorities



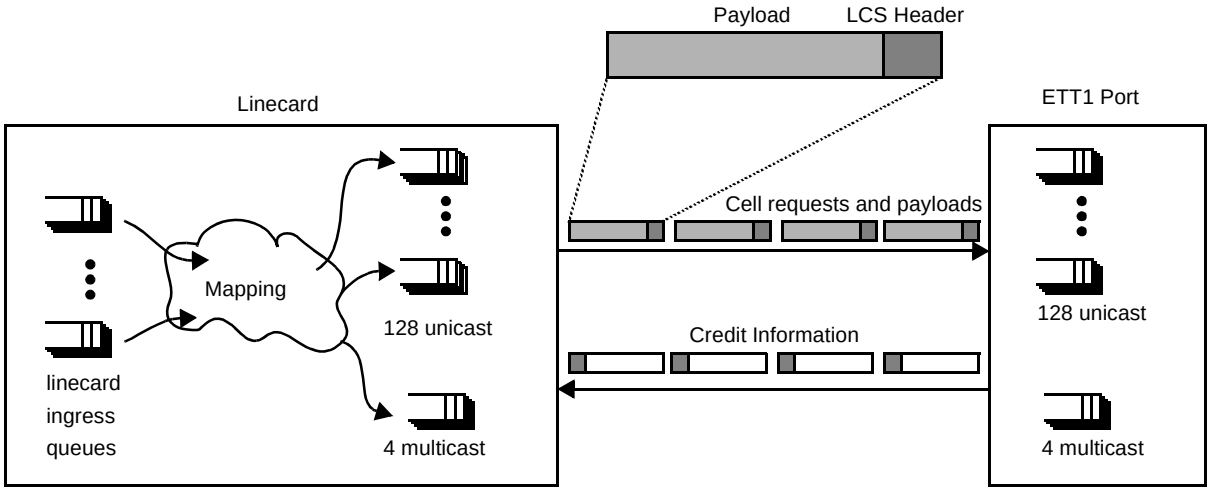
1.3.7 The LCS Protocol

In this section, we describe the LCS protocol from a functional perspective and explain how it works with the queueing model that was described in the previous section.

NOTE: The LCS protocol is defined in the "LCS Protocol Specification -- Protocol Version 2", available from PMC-Sierra, Inc. This version of LCS supersedes LCS Version 1. Version 2 is first supported in the TT1 Chip Set with the Enhanced Port Processor device (also referred to as the ETT1 Chip Set) and will be supported in future PMC-Sierra products. A comparison of the two versions of the LCS protocol is in the same document.

The primary role of the LCS protocol is as a credit-based flow control mechanism that ensures that cells are transferred between the linecard and the switch core only when sufficient buffer space is available. In order to do this, the linecard must have knowledge of the queues that exist in the ETT1 core. The LCS protocol is an asymmetrical protocol; we will first consider the ingress direction. Figure 24 shows the 132 best effort ingress queues in an ETT1 port (this assumes an all-OC-192c switch). The linecard may have a very different number of queues (typically many more), and thus must map its own internal queues to the physical queues present in the ETT1 core.

Figure 24. ETT1 Port Ingress Queues



For example, the linecard might provide an ATM interface which can manage many thousands of virtual circuit flows, and might have a separate internal queue for each flow. The linecard must map its own ingress queues to the 132 ingress queues present in the ETT1 port. The 132 queues shown in the linecard do not have to be physically implemented provided that the linecard can perform the requisite mapping from linecard queues to ETT1 queues on-the-fly. The ETT1 ingress queues are Virtual Output Queues; the mapping function simply has to map each linecard queue to the desired egress port within the ETT1 core. The mapping function must also select the appropriate priority if more than a single priority is required.

Figure 24 shows ingress cell requests and data cells being forwarded to the ETT1 port. These cells are of a fixed size and format: an LCS cell consists of an eight byte LCS header followed by a fixed length payload. The payload can be 64 or 76 bytes (depending on the number of Dataslices and Crossbars used), but all ports in the switch must use the same size payload. The TT1 Chip Set *never* examines or modifies the cell payload unless the cell is used for internal control purposes.

The linecard can send a new cell request whenever a new cell arrives at the linecard and the linecard has at least one request credit remaining for the appropriate queue. So the linecard must keep track of the number of outstanding requests that it can send, and must not issue a request unless it has a corresponding credit.

The ETT1 port returns grant/credit information to the linecard. This grant/credit information reflects the occupancy of the *ingress* queues in the ETT1 port. The linecard can only send a cell to a given ingress queue within the ETT1 port when it receives a grant from the ETT1 port. The linecard increments its corresponding credit counter when it receives a grant/credit.

In order to sustain the maximum possible throughput, the linecard must be able to send a new cell within a short time of receiving a grant/credit from the ETT1 port. If the linecard cannot do this then the VOQ at the ingress ETT1 port may become empty which might, in turn, affect the throughput of the switch. The system designer must be aware of the time budget that is really available to the linecard devices.

The grant/credit mechanism is not used in the egress direction. Rather, a simpler Xon/Xoff-like mechanism is used. The oEPP will forward cells to the egress linecard whenever it has cells waiting in its output queues. The linecard can request the EPP to *not* send a cell of a given priority, or any combination of priorities, by asserting the *hole request* bits in the LCS header. If a hole request bit is asserted at a given priority, then it is a request from the linecard to the EPP that the EPP should *not* forward a cell of that priority at one cell time in the future. The time between the EPP receiving the hole request and observing it is guaranteed to be no more than 64 cell times. Future LCS compliant products will have different response times. Linecard designs requiring backpressure features should accommodate all LCS products to which they will attach.

If a linecard has nearly exhausted its egress buffers for priority 2 cells, then it might *continuously* assert hole request for priority 2 until it can recover some additional buffers.

NOTE: The hole request mechanism operates per-priority and not per-source port.

Refer to the “LCS Protocol Specification - Protocol Version 2”, available from PMC-Sierra, Inc., for further details on the LCS protocol and the definition of the various fields used within the LCS header in each direction.

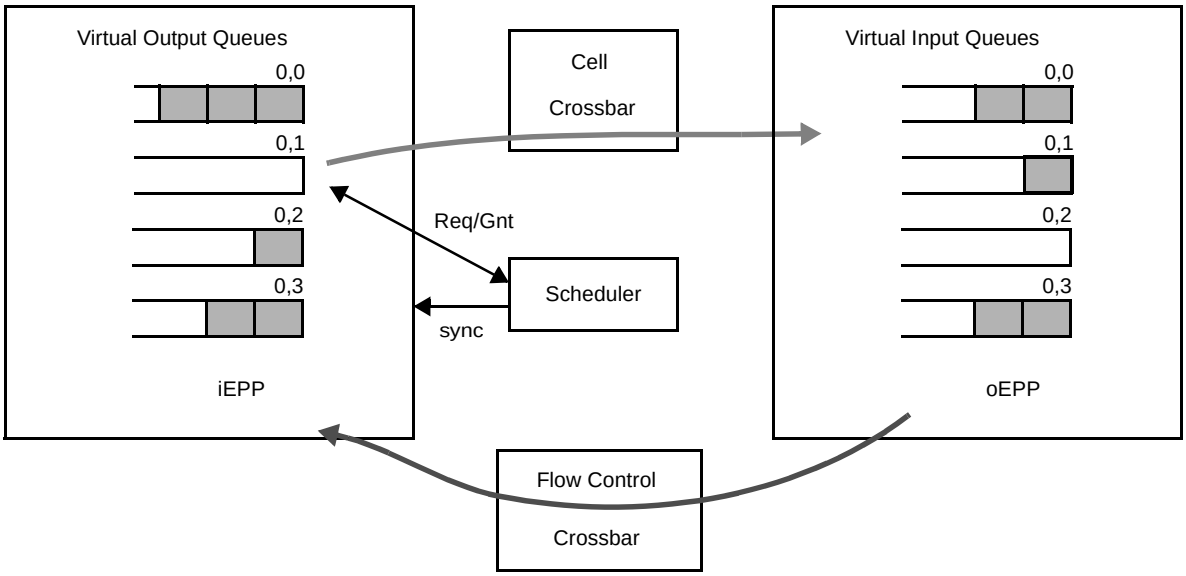
1.4 EXPLICIT BACKPRESSURE FROM OEPP TO IEPP

The iEPP does not make a unicast request to the Scheduler unless the destination VIQ has a sufficient number of empty buffers. For each unicast egress VIQ there is one corresponding ingress VOQ. Every iEPP maintains state information on the occupancy of the relevant egress VIQs in all of the EPPs (including itself) in order to ensure that it does not forward a cell to a full VIQ.

The actual occupancy of a VIQ is dynamic: the number of free cell locations will increment whenever a cell is forwarded to the attached linecard, and the number of free cell locations will decrement whenever a new cell enters the VIQ from the crossbar. The iEPP knows the latter information - it is the iEPP that forwards the cell to the VIQ. But the iEPP does not know when a cell gets sent out to the linecard. So the oEPP must tell each iEPP how many cells have been sent from each relevant VIQ, and this information must be sent with sufficient frequency that the throughput of a single flow is not adversely affected.

In the ETT1 fabric, this update information is transmitted from the oEPPs to the iEPPs via a second Crossbar, referred to as the Flow Control Crossbar. Figure 25 shows the cell flow from left to right and the reverse flow of the update information.

Figure 25. A Separate Crossbar is Used to Convey Backpressure Information to the iEPPs



Every iEPP maintains a set of counters (Output Queue Debit Counters) that track the number of cells in each of the relevant VIQs of every oEPP. At reset, these counters are all set to zero. A counter is incremented whenever the iEPP makes a request to forward a cell to the appropriate VIQ. If the counter reaches a pre-determined maximum, then the iEPP will not issue further requests. Once some cells are forwarded from the VIQ, the iEPP will reduce the counter and resume issuing requests for that flow.

In general, the user does not need to be aware of the details of this mechanism except for the case when update information is lost, perhaps due to a noise-induced error on one or more of the links attached to the Flow Control Crossbar. The details of this error recovery mechanism are described in Section “Enhanced Port Processor - Flow Control Crossbar” on page 94.

1.4.1 Flow Control Crossbar Synchronization

The Flow Control Crossbars interconnect every EPP with every other EPP. During each cell time, the Flow Control Crossbar sets the crosspoints such that every iEPP receives update information from one other oEPP. Further, no two iEPPs receive the same update information from the same oEPP. In the next cell time the crosspoints must be re-organized so that every iEPP receives update information from a different oEPP.

The system must ensure that the Flow Control Crossbars and the EPPs are all synchronized, so that every iEPP knows from which oEPP it is receiving information at any given time. This synchronization information is generated by the Scheduler using the FCCSYN register to send a periodic pulse out to all EPPs. Further, if a redundant Scheduler is used then the two Schedulers must be synchronized. This Scheduler-to-Scheduler synchronization occurs after a Scheduler Refresh operation. Refer to Section 1.9.4 “Scheduler Refresh Procedure” on page 98.

1.5 TDM SERVICE

The TDM service enables linecards to periodically reserve bandwidth within the ETT1 core. Cells that are switched via this service will have a fixed latency through the switch and will not be delayed due to best effort contention for an output port. 1-in-N idle cells and egress control packets may cause temporary delays, but these should be absorbed by the guardband and speedup.

The TDM service is implemented within the ETT1 core using:

- dedicated queues in the ETT1 ports
- a table-based reservation mechanism in the ETT1 ports
- a flexible synchronization mechanism

1.5.1 TDM Queues

The linecard uses the normal LCS request-grant mechanism for TDM cells. When TDM cells arrive at an ETT1 ingress port, they are queued in a single TDM queue which can buffer up to 96 cells. The cells are stored in the ingress queue until their reservation time occurs, at which time they are forwarded through the crossbar to the appropriate egress TDM queue. Given that there can be no contention for TDM cells then a TDM queue might seem unnecessary. The queue is present so that the linecard does not need to be closely synchronized with the ETT1 fabric.

If the port is operating in subport mode then four ingress queues are used, one for each subport. Each ETT1 port has a single egress TDM queue, even when the port is in subport mode. The ETT1 can transmit cells out-of-order between the subports, and so a single shared TDM queue does not cause a problem. Figure 26 illustrates the queueing structure.

Figure 26. The Queueing Structure

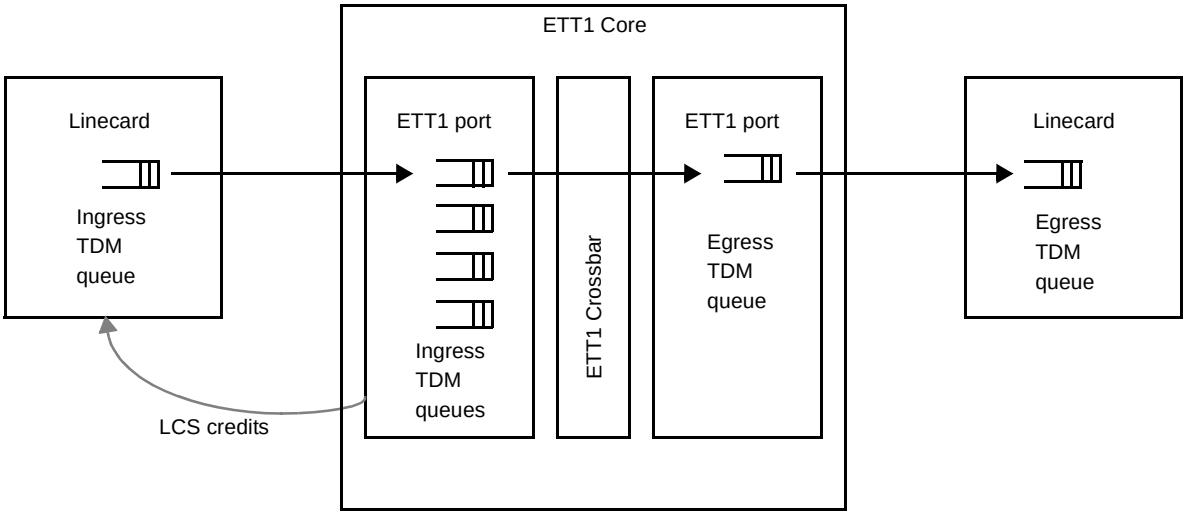


Figure 26 also highlights some important points with respect to the TDM service:

- The linecard may also require a TDM ingress and egress queue. Unless the arrival of TDM cells at the linecard is synchronized with the start of the TDM Frame, the linecard will need to buffer those cells until the appropriate time.
- There is no backpressure from the ETT1 egress TDM queue. Backpressure is not meaningful for a TDM service within the ETT1 core. The egress linecard must accept TDM cells that are sent to it (or else discard them).
- The normal LCS credit mechanism is used for ingress TDM cells.

1.5.2 TDM Reservation Tables

The TDM service operates by enabling ports to reserve bandwidth through the ETT1 crossbar at specific times. Each port has two logically separate tables; one for sending and one for receiving TDM cells. The Send and Receive tables are physically implemented in a single table called the TDM table. This TDM table has 1,024 entries, although fewer entries can be used if the Frame length is less than 1024. Each entry is of the form shown in Table 2.

Table 2. TDM Reservation Table

Field	Size (bits)	Meaning
Input VLD	1	1 if the input side TDM entry is valid
Input Tag	10	Tag for incoming cells (TDM flow ID)

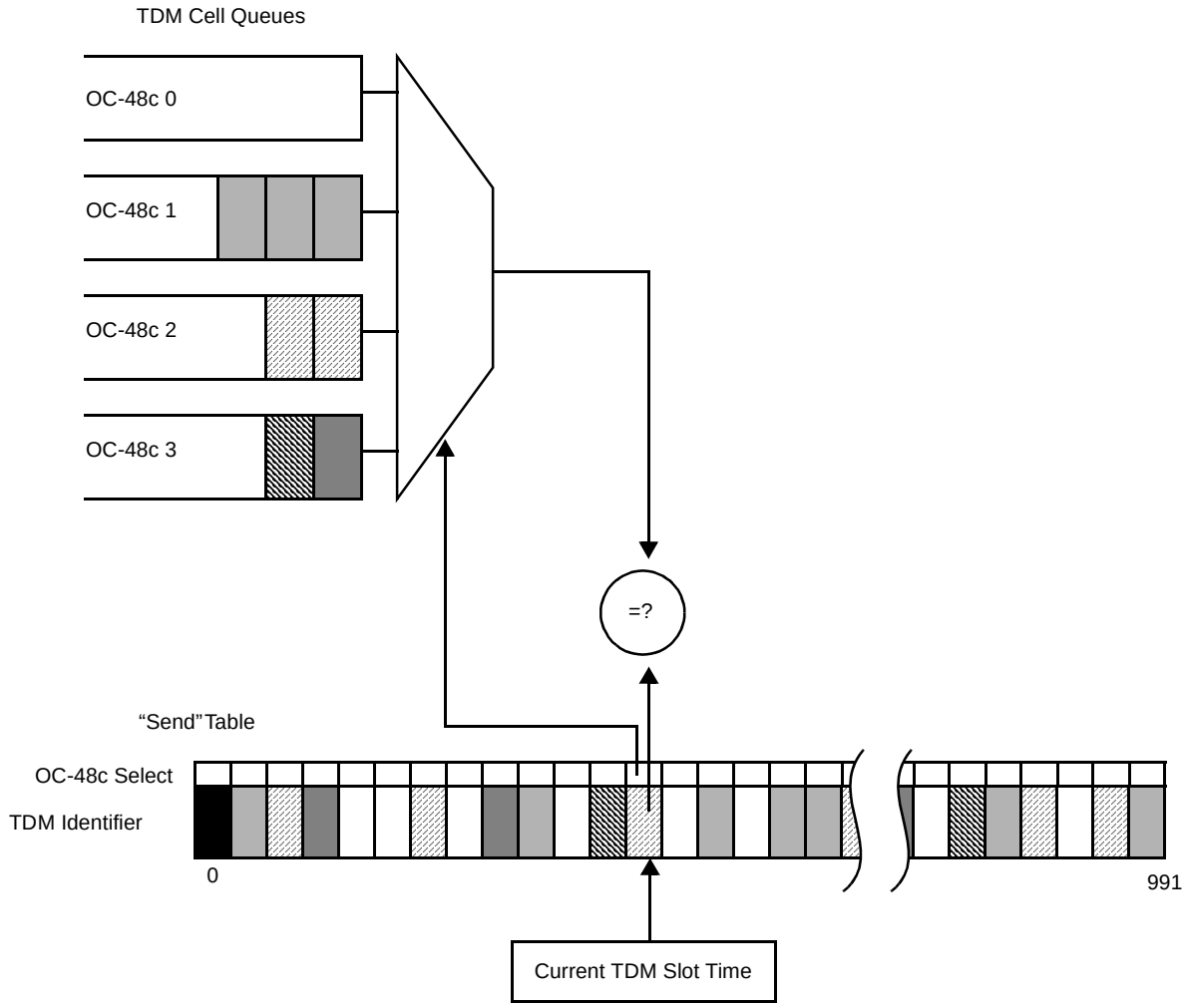
Table 2. TDM Reservation Table (Continued)

Field	Size (bits)	Meaning
Subport	2	Identifies the source subport (if in subport mode, else 0)
Output VLD	1	1 if the output side TDM entry is valid
Input Port	5	Input port that the output expects to receive from
Output vector	4	A four-bit vector which defines the fanout of the subports that should receive a TDM cell.

The EPP contains two TDM tables so that one table can be used by the fabric while the system software is configuring the other table. All EPPs must use the same table at the same time. A TDM LCS Control Packet is used to specify which of the two tables is being used at any given time.

At the start of the Frame, a TDM pointer in each port begins at entry 0 in the table and advances one entry every cell time. During every cell time, if the “current” TDM table Send entry is valid, the EPP checks to see if the cell at the head of the appropriate TDM queue has a TDM flow ID (tag) that matches the entry in the table. If not, then nothing special is done. If they do match then the EPP informs the Scheduler that it will be sending a TDM cell at some fixed time in the future. This prevents the Scheduler from trying to schedule a best-effort cell from that port at the same time. Figure 27 shows the logic at the ingress port.

Figure 27. Ingress Port Logic



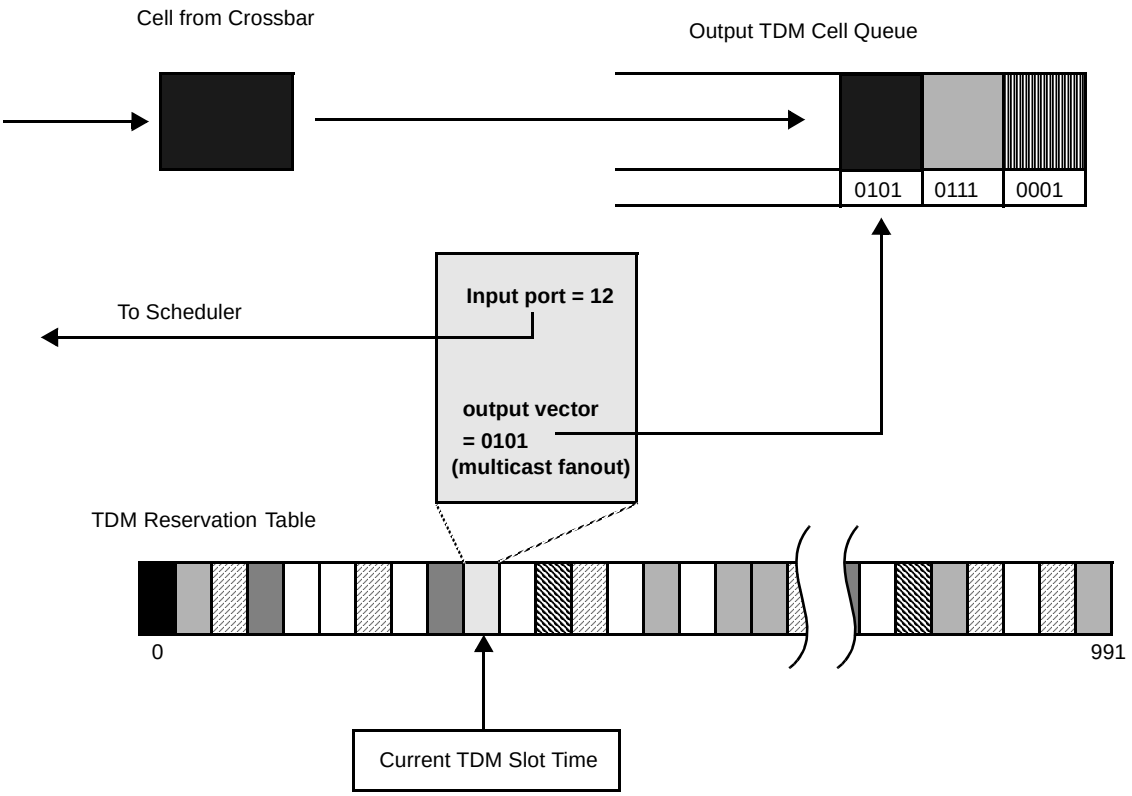
The egress side of the port also looks at the TDM table. If the Output Valid bit in the current entry is set, then the port tells the Scheduler that it expects to receive a TDM cell at some fixed time in the future. This prevents the Scheduler from trying to schedule a best-effort cell to that port at the same time.

If the Scheduler doesn't receive the input TDM request bit from the specified input port, then it assumes that there is no TDM cell ready to be transmitted, and thus any egress port that expected to receive that cell will now become available for best-effort cell scheduling.

When the TDM cell enters the egress port, it is placed in a separate, 96-entry, egress queue dedicated for TDM traffic. Whenever this queue is non-empty, the cell at the head of the queue is sent immediately to the linecard (provided that the Output Scheduler is not frozen).

The TDM service is inherently a multicast service. The egress ETT1 must therefore support the ability for a TDM cell to go to any combination of the four subports within a port. A four-bit field in every TDM table entry provides the multicast fanout for each TDM cell that is received at the egress port. The EPP keeps track of this fanout information and sends the TDM cell to each subport that is a member of the fanout. Figure 28 illustrates how the EPP uses the TDM table to determine the source port of a TDM cell, as well as the local subport multicast fanout.

Figure 28. The oEPP Must Treat the Transferred TDM Cell as a Multicast Cell



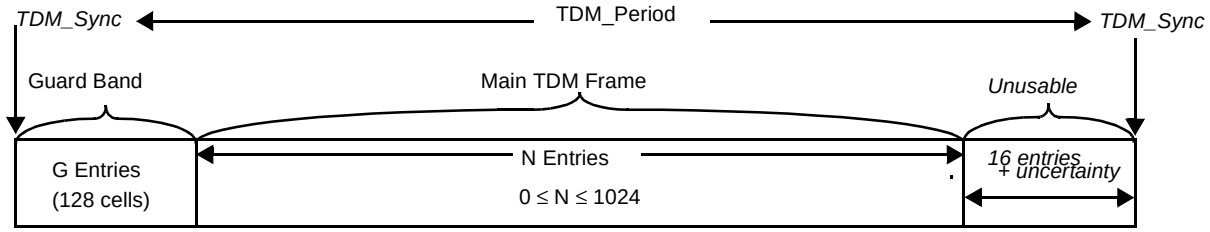
The output TDM cell queue does not need to be large as TDM cells are always forwarded to the output linecards at the highest priority, and the TDM queue is never backpressured. The size of this queue does, however, restrict the possible TDM table configuration. The following constraints are required to prevent dropped TDM cells:

- In a TDM table time slot, an input port can not send more than one cell and an output port can not receive more than one cell.
- Subport mode: The TDM Reservation Table can specify the same ingress subport only once every four time slots.
- Subport mode: The TDM Reservation Table can not specify the same egress subport more than 48 time slots within a moving window of 192 time slots. This constraint is caused by the TDM output queue.

1.5.3 TDM Frame Timing

A single TDM table is shown in Figure 29. In addition to the 1,024 entry portion, there is an additional "guard band" of G entries preceding the table. These G entries are unusable and are explained below. An additional 16 time slots (plus uncertainty/jitter in TDM sync timing) are unusable following the last valid entry in any of the TDM tables due to pipeline latency through the Scheduler and EPP. These 16 entries cannot be used for TDM traffic.

Figure 29. Single TDM Table



The time between TDM Sync signals is referred to as the TDM period. This period is set to the sum of G plus the length of table N plus the final latency allowance of 16 + uncertainty. In the EPP chapter, G is referred to as the "TDM offset value".

1.5.4 TDM Synchronization

There are several different levels of synchronization required for the TDM service to work properly. First, all EPPs must point to the same entry in their TDM tables in order for the Send and Receive reservations to match up in the Scheduler. Second, each linecard must be synchronized (approximately) with its ETT1 port so that it can send TDM cells at the right time and third, so that the linecard will switch TDM tables synchronously with the ETT1 port.

The first level of synchronization, between ETT1 ports, is handled automatically by the ETT1 Chip Set. All of the ETT1 devices operate cell synchronously.

The second and third levels of synchronization, between the linecard and ETT1, are handled by the TDM Sync mechanism. The Scheduler is configured in one of three ways, described below, to periodically send out a TDM_Sync to all ports. All of the ETT1 ports receive the TDM_Sync signal from the Scheduler at the same cell time. Upon receiving the TDM_Sync from the Scheduler, every port then (1) flushes any cells currently in the TDM input queue, (2) transmits a TDM Control Packet to the linecard and (3) starts a counter which increments every cell time. When this counter reaches the value of the TDM offset (guardband), the EPP begins comparing the tag of the cell at the head of the TDM FIFO with the entry in TDM table location 0, and normal TDM processing commences.

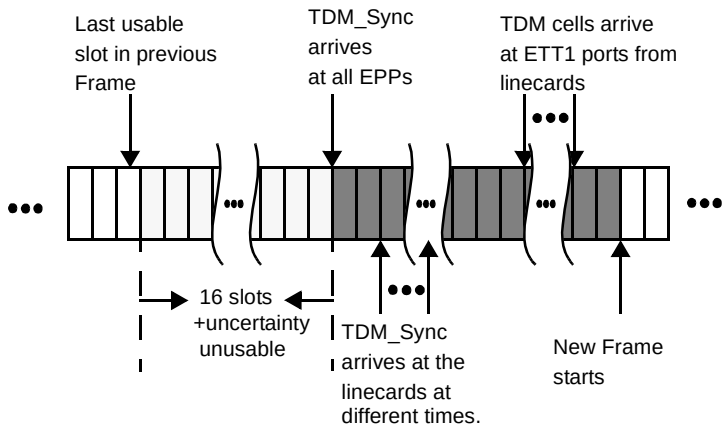
The TDM_Sync and TDM Control Packet both include a Frame Select bit to tell the ports and linecards which TDM table to use. This allows all ports and linecards to switch TDM tables synchronously.

TDM_Sync signals do not arrive at the linecards at exactly the same cell time. There may be several cell times between the first linecard receiving TDM_Sync and the last. The variation is due to variations in link lengths and also due to timing uncertainties when signals cross asynchronous clock domains. The guard band absorbs these variations, so the linecard does not need to be aware of the link delays or clock domain issues.

As soon as the linecard receives the TDM_Sync signal it starts to forward TDM requests to the ETT1 port. The port has an ingress queue for these cells and so can buffer them until they are required. The ingress queue is 96 cells deep, and is controlled by the normal LCS request-grant flow-control mechanism. The linecard *must* forward TDM cells in the correct order (corresponding to its entries in its TDM Send table), but it does not need to be closely synchronized to the ETT1 core. The linecard can skip TDM cells (i.e. not send them), but must not re-order cells relative to the TDM tables.

The ETT1 core waits for 128 cell times (the TDM offset) after it has sent TDM_Sync before the new TDM Frame is started. This provides all linecards with sufficient time to receive the TDM_Sync signal and to request and send in the first TDM cells. Figure 30 shows the sequence of events in more detail.

Figure 30. TDM Synchronization



The Scheduler can be configured to send the TDM_Sync in three ways. Each way is driven by one of these three sources:

1. A Suggested_TDM_Sync from a counter in the EPP.
2. A counter in the Scheduler.
3. A Suggested_TDM_Sync from a designated linecard (through the EPP).

There are pros and cons to each of these methods. Each method is described in more detail below.

1.5.4.1 TDM Sync Driven by the EPP

The EPP can be programmed to send a Suggested_TDM_Sync to the Scheduler once every N+1 celltimes by writing N to the EPP's TDM Sync Generation Control register and writing 1 to the Enable TDM Sync Generation register. The TDM table select is also a field of the TDM Sync Generation Control register. The Scheduler is programmed to listen to the Suggested_TDM_Sync from a designated port.

The advantage of this method over the third method is that the only exposure to jitter (uncertainties in delay) on the TDM_Sync is after the EPP sends the TDM Control Packet to the linecard, due to crossing asynchronous clock domains (and subport-ordering delay in subport mode). The advantage of this method over the second method is that when using two Schedulers for fault tolerance, the second method requires synchronous OOB writes to program the Schedulers at exactly the same time; this method provides a synchronous Suggested_TDM_Sync to the two Schedulers instead.

A possible disadvantage of this method is that TDM Sync is synchronous to the ETT1's core clock, not to the linecard's clock or any other external synchronization signal.

1.5.4.2 TDM Sync Driven by the Scheduler

The Scheduler can be programmed to generate its own TDM_Sync signals once every N+1 celltimes using the Scheduler's TDM Control register. The TDM table select is also controlled via the TDM Control register. The Scheduler ignores Suggested_TDM_Sync from all ports.

The advantages of this method are simplicity and low jitter, similar to the first method.

The disadvantage of this method is that when using two Schedulers for fault tolerance, this method requires synchronous OOB writes to program the Schedulers at exactly the same time; if the Schedulers' TDM Control registers are enabled to send TDM Sync at different times, then all EPPs will report mismatching frames from the two Schedulers. Scheduler refresh will not synchronize Scheduler-generated TDM Syncs.

1.5.4.3 TDM Sync Driven by a Designated Linecard

Any linecard can send a Suggested_TDM_Sync to its ETT1 port, using a LCS TDM Control Packet, every TDM_Period cells. The TDM Control Packet also specifies which of the two TDM tables should be used. When the EPP receives a TDM Control Packet it sends a Suggested_TDM_Sync signal to the Scheduler. The Scheduler is programmed to listen to the Suggested_TDM_Sync from a designated port.

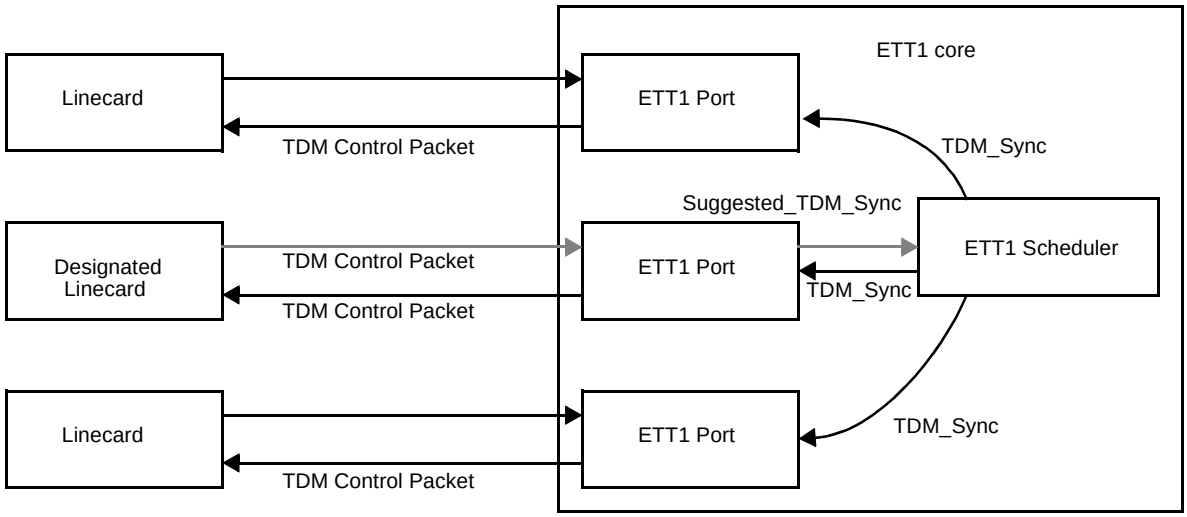
The advantage of this method is that an external synchronization source can be used.

The disadvantage of this method is that there is much more exposure to jitter in TDM_Sync. Asynchronous clock domain crossings (and subport-ordering delays in subport mode) will cause delay variations at multiple stages: the LCS request for the Suggested_TDM_Sync Control Packet, the LCS grant, the delivery of the Suggested_TDM_Sync Control Packet payload, and finally the delivery of the TDM_Sync Control Packet from the ETT1 to the linecards. Additionally, the Scheduler's internal digital TDM_Sync PLL

(described below) adds jitter. See the "TDM Service in the ETT1 and TTX Switch Cores" App Note for a more thorough discussion of TDM_Sync jitter and its consequences.

On receiving a Suggested_TDM_Sync signal, the Scheduler computes the period at which the Suggested_TDM_Sync signals are arriving. The Scheduler then uses this period to determine the next period at which it sends a TDM_Sync to all ETT1 ports. The period of the TDM_Sync signals will track the period of the incoming Suggested_TDM_Sync signals. However the phase of the TDM_Sync signals may be shifted relative to the Suggested_TDM_Sync signals. Internally the Scheduler has a digital PLL which will gradually cause the transmitted TDM_Sync signal to have a fixed phase offset relative to the incoming Suggested_TDM_Sync signal. Figure 31 shows the passage of signals.

Figure 31. Signal Flow



After a count of TDM_Period cell times, the designated line card will send another Suggested_TDM_Sync, the line card TDM period counter will re-initialize and the process will repeat. If the Scheduler sees three TDM periods go by without receiving another Suggested TDM sync, it will trigger an interrupt and continue to send TDM_Syncs to the ports at the current period.

1.5.5 Configuring the TDM Service

Before TDM cells can flow the system must be configured. This involves the following:

1. Configure the TDM tables in all ports that will send or receive TDM cells.

2. Configure the Scheduler in the appropriate mode for TDM synchronization.

3. Enable TDM cells.

Configuring the tables is straightforward, as described in Section 1.5.2 “TDM Reservation Tables”. The rules for the tables are:

(a) No port can source more than one TDM cell at any cell time.

(b) No port can receive more than one cell at any cell time.

(c) The linecard must send the cells to the EPP:
- in order, and
- before the current TDM slot reaches the slot reserved for that cell.

(d) Subport mode: The TDM Reservation Table can specify the same ingress subport only once every four time slots.

(e) Subport mode: The TDM Reservation Table can not specify the same egress subport more than 48 time slots within a moving window of 192 time slots. This constraint is caused by the TDM output queue.

Item (c) above is important to understand: the EPP will buffer all ingress TDM cells before they are sent through the ETT1 fabric. The EPP can only buffer 96 TDM cells at the ingress. The linecard must be able to send TDM cells to meet the average and the burst rate that are implied by the TDM table configuration. So, if the linecard is operating at 20M cells/sec, for example, while the ETT1 fabric operates at 25M cells/sec then the linecard must not reserve a contiguous burst of 128 TDM slots because it would be unable to supply the last few cells in time for their reservations.

The next step is to configure the Scheduler. The Scheduler TDM service is controlled via its TDM Control register. The first aspect to configure is the source of the Suggested_TDM_sync signal. If a linecard or EPP will supply the Suggested_TDM_sync signal then configure the TDM Control register with the selected port and set the Current Port valid bit. If a linecard or EPP does not supply the Suggested_TDM_sync signal then the Scheduler must be configured with the desired TDM_Period: write the desired period into the Initial Period field and toggle (0 to 1 to 0) the Load Initial Period control bit. In all cases the Enable bit (bit 31) of the Scheduler’s TDM Control register should be set to 1.

The final stage of configuring the Scheduler is to write the desired value to the Enable TDM traffic register. Each of the 32 bits should be set to 1 for those ports that will send or receive TDM traffic.

1.5.6 Changing the Source of the Suggested Sync

If the Scheduler is configured to use an external (port or linecard) source for the Suggested_TDM_sync, then you can change the source port by writing a new value into the port field of the TDM Control register. Once the new port starts to supply the Suggested_TDM_sync signal then the Scheduler will adjust the frequency and phase of the transmitted TDM_Sync signal to match that of the new incoming signal. The Scheduler will change the frequency of the outgoing sync signal once it has seen three Suggested_TDM_sync signals, but it will only gradually adjust the phase. The phase of the outgoing TDM_Sync will be adjusted by one cell time in every TDM_period.

NOTE: The phase reference for the Scheduler is the received Suggested_TDM_sync signal,

which will be several cell times after the linecard has sent the Suggested_TDM_sync to the ETT1 port.

If the Scheduler does not see an incoming Suggested_TDM_sync signal for three TDM_periods then it will issue an interrupt indicating that the sync source has failed. However it will continue to operate with the current values of frequency and phase.

1.6 ETT1 USAGE OF THE LCS PROTOCOL

This section provides details of how the LCS header is used in an ETT1 system. Further information on the operation of the LCS protocol is in the “LCS Protocol Specification -- Protocol Version 2”, available from PMC-Sierra, Inc.

1.6.1 LCS Protocol Prepend

LCS segments are 72 or 84 bytes in length. The line to switch format is shown in Table 3, while Table 5 shows the field definitions of this format. Likewise, Tables 3 and 4 show the switch to line format and field definitions, respectively.

Table 3. LCS Format from Linecard to Switch

		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	**Request Field**	Req Valid	Label_1 [13:7]						
Byte 1		Label_1 [6:0]							Type_1 [3]
Byte 2		Type_1 [2:0]			Rsvd				
Byte 3	**Payload Field**	Payload Valid	Seq_Number[9:3]						
Byte 4		Seq_Number[2:0]			Rsvd				
Byte 5	**Hole Field**	Payload_Hole_Request[3:0]				Rsvd			
Byte 6	**CRC Field**	CRC-16[15:8]							
Byte 7		CRC-16 [7:0]							
Bytes 8-71 or 8-83	**Payload**	Payload Data							

Table 4. Definitions of LCS Fields from Linecard to Switch

Field	Size (bits)	Definition
Req Valid	1	Indicates that request field is valid (Request Label_1).
Label_1	14	Label that is being requested (e.g. unicast-QID/multicast-label/TDM-label).
Type_1	4	Segment type that is being requested (e.g. Control Packet).
Payload Valid	1	Indicates that the Payload Data field contains a valid payload.
Seq_Number	10	A sequence number for synchronizing grants with payloads.
Payload_Hole_ Request	4	Request for a “hole” in the payload stream from switch to linecard. (Priority 3,2,1,0 for bits [3:0], respectively.) (See Section 1.2.4 “End-to-End Flow Control” on page 22 for explanation).
CRC-16	16	16-bit CRC calculated over complete 64-bit prepend.[a]
Payload Data	64 or 76 bytes	Payload Data

a. See Section 1.6.4.1 “Use of CRC-16 Field” on page 73 for details of CRC calculation and Section 3.4.2.2 “Reset and Control” on page 169, bit 4.

Table 5. LCS Format from Switch to Linecard

		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	**Grant Field**	Grnt Valid	Label_1 [13:7]						
Byte 1		Label_1 [6:0]							Type_1 [3]
Byte 2		Type_1 [2:0]			Seq_Num [9:5]				
Byte 3		Seq_Num [4:0]					Payload Valid	Label_2 [13:12]	
Byte 4	**Payload Field**	Label_2 [11:4]							
Byte 5		Label_2 [3:0]				Type_2 [3:0]			
Byte 6	**CRC Field**	CRC-16 [15:8]							
Byte 7		CRC-16 [7:0]							
Bytes 8-71 or 8-83	**Payload**	Payload Data							

Table 6. Definitions of LCS Fields from Switch to Linecard

Field	Size (bits)	Definition
Grnt Valid	1	Indicates that grant field is valid (Grant Label_1 and Sequence Number).
Label_1	14	Label for credit (e.g. QID/mcast-label/TDM-label).
Type_1	4	Segment type for credit (e.g. TDM).
Seq_Num	10	Sequence Number associated with grant.
Payload Valid	1	Indicates whether the egress payload is valid (Payload Label_2 and Payload Data).
Label_2	14	Label for egress payload (e.g. ingress port identifier).
Type_2	4	Segment type for egress payload (e.g. Control Packet).
CRC-16	16	16-bit CRC calculated over complete 64-bit prepend.[a]
Payload Data	64 or 76 bytes	Payload Data

a. See Section 1.6.4.1 “Use of CRC-16 Field” on page 73 for details of CRC calculation and Section 3.4.2.2 “Reset and Control” on page 169, bit 4.

1.6.2 Use of Label and Type Fields

This section provides detail about the generic 14-bit labels and 4-bit types defined in Table 7, Table 8, Table 9, and Table 10. In all cases, in multi-bit fields, the MSB is transmitted first. For each segment, the "Type Field" and "Label Field" are encoded as follows:

Table 7. Encoding of 4-bit Type Fields

Type[3:0]	Segment Type	Use
0000	Control Packet	Provides for inband control information between the linecard and the switch core.
0001,...,0110	Reserved for future use	Currently not defined for use.
0111	TDM	TDM service provides preallocated bandwidth at a regular time interval.
1000,...,1011	Unicast Best-Effort (Priority 0,1,2,3)	Prioritized unicast service where priority 0 is the highest priority.
1100,...,1111	Multicast Best-Effort (Priority 0,1,2,3)	Prioritized multicast service where priority 0 is the highest priority.

Table 8. Usage of the Ingress Request Label_1

Segment Type	Type_1	Encoding of Request Label_1[13:0]
Control Packet	0000	Rsvd[a] (13), CP-category (1). CP-category = 0 for LCS Control Packets; CP-category = 1 for CPU Control Packets.
TDM	0111	MUX (2)[b], Expansion[c] (2), TDM Tag (10)
Unicast Best-Effort (Priority 0,1,2,3)	10xx	MUX (2)[b], Expansion[d] (5), Destination Port (5), Destination subport[e] (2)
Multicast Best-Effort (Priority 0,1,2,3)	11xx	MUX (2)[b], Multicast Tag (12)

a. Reserved. All bits labeled Rsvd must be set to zero.
b. The MUX[1:0] field is placed at the beginning of the Request Label_1 field in all segments (i.e. both data traffic and control packets). If this field is not used for a MUX function, it can be used as additional bits for expansion. If not used, it must be set to zero.
c. TDM Expansion field used in products if the number of TDM slots is increased. If this field is used, the "TDM Tag" field will increase contiguously. If not used, must be set to zero.
d. Unicast Expansion field used in products if the number of ports is increased. If this field is used, the "Destination Port" field will increase contiguously. If not used, must be set to zero.
e. If the port has no subports (i.e. if it is connected to a single linecard without multiplexing), this field must be set to zero.

Table 9. Usage of the Egress Grant Label_1

Segment Type	Type_1	Encoding of Grant Label_1[13:0]
Control Packet	0000	Rsvd[a] (13), CP-category (1). CP-category = 0 for LCS Control Packets; CP-category = 1 for CPU Control Packets.
TDM	0111	MUX (2)[b], Expansion[c] (2), TDM Tag (10)
Unicast Best-Effort (Priority 0,1,2,3)	10xx	MUX (2)[b], Expansion[d] (5), Destination Port (5), Destination subport[e] (2)
Multicast Best-Effort (Priority 0,1,2,3)	11xx	MUX (2)[b], Multicast Tag (12)

a. Reserved. All bits labeled Rsvd must be set to zero.
b. The MUX[1:0] field is placed at the beginning of the Grant Label_1 field in all segments (i.e. both data traffic and control packets). If this field is not used for a MUX function, it can be used as additional bits for expansion. If not used, it must be set to zero.
c. TDM Expansion field used in products if the number of TDM slots is increased. If this field is used, the “TDM Tag” field will increase contiguously. If not used, must be set to zero.
d. Unicast Expansion field used in products if the number of ports is increased. If this field is used, the “Destination Port” field will increase contiguously. If not used, must be set to zero.
e. If the port has no subports (i.e. if it is connected to a single linecard without multiplexing), this field must be set to zero.

Table 10. Usage of the Egress Payload Label_2

Segment Type	Type_2	Encoding of Payload Label_2[13:0]
Control Packet	0000	Rsvd[a] (13), CP-category (1). CP-category = 0 for LCS Control Packets; CP-category = 1 for CPU Control Packets.
TDM	0111	Expansion[b] (7), Source Port (5), Source subport[c] (2)
Unicast Best-Effort (Priority 0,1,2,3)	10xx	Expansion[d] (7), Source Port (5), Source subport[c] (2)
Multicast Best-Effort (Priority 0,1,2,3)	11xx	Expansion[e] (7), Source Port (5), Source subport[c] (2)

a. Reserved. All bits labeled Rsvd must be set to zero.
b. TDM Expansion field used in products if the number of ports is increased. If this field is used, the “Source Port” field will increase contiguously. If not used, must be set to zero.
c. If the source port has no subports (i.e. if it is connected to a single linecard without multiplexing), this field will be zero.
d. Unicast Expansion field used in products if the number of ports is increased. If this field is used, the “Source Port” field will increase contiguously. If not used, must be set to zero.
e. Multicast Expansion field used in products if the number of ports is increased. If this field is used, the “Source Port” field will increase contiguously. If not used, must be set to zero.

1.6.2.1 Subport Mode Label and MUX Fields

Label fields are arranged such that port addresses can be subdivided into four multiplexed, subport addresses. Each segment type with a subport field utilizes bits [1:0] for the subport addresses. When subports are not utilized, these bits will be set to zero.

Subport multiplexing shares a common physical connection to carry segments to/from each subported linecard to a switch port. The time multiplexing between subports may be done on a quad linecard, or may be done at an intermediate stage. When time-multiplexed, segment transmissions are sequenced between each of the four subports through the use of a MUX field.

When a port of an LCS device uses the time-multiplexed subport mode, two bits in the Request Label_1 field of ingress and Grant Label_1 field of egress segments are used to multiplex/demultiplex each of four subport channels. The 2-bit field is called the MUX[1:0] field. For ingress segments, the MUX[1:0] bits are carried in the Request Label_1 field; for egress segments, the MUX[1:0] bits are carried in the Grant Label_1 (but not the Payload Label_2). Note that for ingress traffic, this field is used to designate the source subport of the ingress segment and is independent of the Request Label_1 subport field. Likewise for egress traffic, the MUX field is used to designate the destination subport of the egress segment and is independent of the Grant Label_1 and Payload Label_2 subport fields.

Segments entering the switch core must have the MUX[1:0] field correctly set. The MUX[1:0] field in arriving segments must follow the sequence 00,01,10,11,00,... in consecutive segments. The MUX[1:0] field in departing segments must also follow the sequence 00,01,10,11,00,... in consecutive segments. When the MUX subport sequence must be interrupted by the need to send an Idle frame, the sequence should continue with the next subport in sequence following the previously sent subport. The MUX[1:0] field is placed at the beginning of the Request Label_1 or Grant Label_1 field in all segments (i.e. both data traffic and control packets).

Segments departing or arriving at the subported line card may have the MUX[1:0] field set to correspond to the linecard's designated subport. Optionally, the linecard may set the MUX[1:0] field to zero, relying on an intermediary multiplexer device to correctly set the MUX[1:0] field. In such a configuration, the LCS prepend CRC is generated and checked masking the MUX[1:0] field to zero, enabling the intermediary multiplexer device to independently set the MUX[1:0] field without termination and regeneration of the CRC field. Note that without the requirement of setting the MUX[1:0] field, the linecard's framing function does not have to be subport aware in a system with an intermediary, time-multiplexed data stream.

1.6.3 Control Packets

The LCS link is controlled by Control Packets (CPs) that are generated by the linecard or the switch core. When the switch core sends a CP to the linecard, it marks the correct Payload Label_2 and Type_2 fields as a Control Packet and sends the control information in the payload data. When the linecard wishes to send a CP to the switch core, it goes through the normal three phase request/grant/transmit process. Control Packets have a higher priority over other traffic; therefore, grants will be returned in response to CP request prior to other outstanding requests.

There are three categories of CPs: (1) LCS Control Packets that are used to perform low-level functions such as link-synchronization, (2) CPU Control Packets which are used for communication between the

linecard and the switch core CPU, and (3) Single Phase Control Packets which are used as a special form of LCS Control packets for timing sensitive information.

1.6.3.1 LCS Control Packets

LCS CPs are specified by setting the Label_1/Label_2 Field to 14'h0000 (CP-category for LCS Control Packets) and associated Type Field to 4'h0 (Type for Control Packets). Table 11 describes the contents of the first 10-bytes of the LCS segment payload for an LCS CP format.

Table 11. LCS Control Packet Format

Field	Size (bits)	Definition
CP_Type	8	Indicates the type of LCS Control Packet (e.g. TDM)
CP_Type Extension	56	Indicates the information carried with this type of control packet. Unused bits must be set to zero.
CRC-16	16	16-bit CRC calculated over 10-byte control packet payload.[a]

a. See Section 1.6.4.1 "Use of CRC-16 Field" on page 73 for details of CRC calculation.

Because there is no flow control mechanism for LCS CPs sent to the switch fabric, and a CP request/grant does not distinguish the CP_Type, care must be taken so as not to overflow the receiving CP queues. At most, only one request for each CP_Type should be outstanding.

In what follows, the three CP_Types of currently defined LCS CPs are described.

NOTE: The actual LCS headers for Switch-to-Linecard Control Packets are not built in to the EPP. Instead, the EPP expects to find the headers, preconfigured at specific queue locations within the Dataslices. Thus, part of the initialization sequence of a new ETT1 port card is to write the values of the Control Packets into the Dataslice queues. Section 3.3 "Output Dataslice Queue Memory Allocation with EPP" on page 162, provides details of these locations.

TDM Control Packets

The format of TDM CPs is described in Table 12.

Table 12. TDM Control Packet Format

Field	Size (bits)	Definition
CP_Type	8	00000000
TDM Frame Sel	1	Indicates which TDM frame to use.
Unused	55	Set to zero.
CRC-16	16	16-bit CRC calculated over 10-byte control packet payload.

Note that when a TDM CP is sent from the linecard to the switch core, no reply is generated by the switch core.

Request Count Control Packets

Request Count CPs are only sent from the linecard to the switch, and are used to keep request counter values consistent between the linecard and the switch core. When the switch core receives this CP, it updates the corresponding request counter (specified by Label_1 and Type_1) with the count field. This control packet can be sent periodically or when the linecard suspects that the linecard and switch core have different values (e.g. following a CRC error).

Request Count CPs are required for unicast traffic and also required for multicast and TDM requested traffic when supported. For unicast traffic, the Label_1 field is set as described in Table 8. For multicast traffic, the Label_1 field is undefined and should be set to zero. The format of the Request Count CP is described in Table 13

Table 13. Request Count Control Packet Format

Field	Size (bits)	Definition
CP_Type	8	00000001
Label_1	14	Label for request counter.
Type_1	4	10xx unicast 11xx multicast
Count	14	Update counter value, always set to zero for multicast.
Unused	24	Set to zero.
CRC-16	16	16-bit CRC calculated over 10-byte control packet payload.

Start/Stop Control Packets

Start/Stop Control Packets are used to toggle the flow of data traffic over the link (here, data traffic is defined to be TDM, best-effort unicast and multicast segment payloads and associated requests and grants. i.e. all non-Control Packet segments). Start/Stop Control Packets can be sent over an LCS link in either direction.

If a linecard sends a Stop Control Packet to the switch core, the switch core will stop sending grants to the linecard for ingress data traffic, and will stop sending egress data traffic to the linecard. When a Start Control Packet is sent, grants and egress data traffic resume.

If the switch core sends a Stop Control Packet to a linecard, the linecard must stop sending new data traffic requests into the switch core until a Start Control Packet is sent.

Note that when either end has received a Stop Control Packet, it can continue to request, grant, send and receive all categories of Control Packets.

When an LCS link powers-up, it will be in Stop mode: i.e. it will act as if a Stop Control Packet has been sent in both directions.

Table 14. Start/Stop Control Packet Format

Field	Size (bits)	Definition
CP_Type	8	00000010
Start/Stop	1	1=Start, 0=Stop.
Unused	55	Set to zero.
CRC-16	16	16-bit CRC calculated over 10-byte control packet payload.

1.6.3.2 CPU Control Packets

CPU Control Packets are specified by setting the Label_1/Label_2 Field to 14'h0001 (CP-category for CPU Control Packets) and associated Type Field to 4'h0 (Type for Control Packets).

When the linecard sends a CPU CP to the switch core CPU, the complete payload data is passed to the switch core CPU. Likewise, when the switch core sends a CPU CP to the linecard, the complete payload data is passed to the linecard.

No flow control mechanism is explicitly implemented for CPU CPs, and will be implementation dependent.

1.6.4 Use of CRC Fields

1.6.4.1 Use of CRC-16 Field

Control Packets and LCS protocol segment prepends use the CRC-16 polynomial: $x^{16} + x^{12} + x^5 + 1$. This is the same polynomial used in X.25.

1. Prior to starting the calculation, the remainder is set to 0xffff.
2. In all cases, the CRC-16 is calculated over all bits (80-bits of the CP or 64-bits of the prepend) with the CRC field set to all zeroes.
3. Transmission of the CRC-16 is done with the most significant bit sent first.

Note that the use of the CRC-16 option in the ETT1 Switch to Linecard LCS protocol segment prepend requires that the first byte of the CRC-16 be masked when the CRC-16 is checked (see example).

The following are some sample CPs and prepends, showing the CRC-16 generated for each. The CRC-16 field is always the last 16 bits.

LCS Start CP:

0x0280000000000000C566

Request Count CP:

0x01037280240000006FDB

LCS Prepend from Switch to Linecard:

0x8719362C09DCxxC7 [1] (subport = 0)

0xE719362C09DCxxC7 [1] (subport = 3, MUX masked to 0 for CRC)

0xE719362C09DCxxDF (subport = 3, MUX included in CRC)

The use of the CRC-16 polynomial in an ETT1 system is the same as above, however only the last eight bits of the 16-bit CRC are valid from the Switch to the Linecard. Therefore, the receiving linecard should calculate the CRC for the entire prepend with the CRC field set to all 0's and compare the last eight bits to those of the received CRC as shown here. "xx" means these eight bits should be masked from the compare.

LCS Prepend from Linecard to Switch:

0x81B9409D00106039 [1] (subport = 0)

1. Note the MUX[1:0] field is assumed equal to "00" in these CRC-8 calculations.

0xE1B9409D00106039 [1] (subport = 3, MUX masked to 0 for CRC)

0xE1B9409D00103F21 (subport = 3, MUX included in CRC)

The CRC-16 from the Linecard to the Switch is the same as the standard CRC-16.

1.6.4.2 Use of CRC-8 Option

Optionally, an ETT1 based system can use LCS segment prepends with the CRC-8 polynomial: $x^8 + x^2 + x + 1$.

1. Prior to starting the calculation, the remainder is set to 0xff.
2. In all cases, the CRC-8 is calculated over all bits (64-bits of the prepend) with the CRC field set to all zeroes.
3. Transmission of the CRC-8 is done with the most significant bit sent first.

The following are some sample prepends, showing the CRC-8 generated for each. The CRC-8 field is always the last 8 bits.

LCS Prepend from Switch to Linecard:

0x8719362C09DCxxFD [1] (subport = 0)

0xE719362C09DCxxFD [1] (subport = 3, MUX masked to 0 for CRC)

0xE719362C09DCxx19 (subport = 3, MUX included in CRC)

The CRC-8 polynomial in an ETT1 system uses only the last eight bits of the 16-bit CRC field. Therefore, the receiving linecard should calculate the CRC for the entire prepend with the entire CRC field set to all 0's and compare the calculated CRC-8 to those of the received CRC-8 as shown here. "xx" means these eight bits should be masked from the compare.

LCS Prepend from Linecard to Switch:

0x81B9409D00100096 [1] (subport = 0)

0xE1B9409D00100096 [1] (subport = 3, MUX masked to 0 for CRC)

0xE1B9409D00100072 (subport = 3, MUX included in CRC)

The CRC-8 from the Linecard to the Switch is sent with the leading eight bits of the CRC-16 field set to zero.

1. Note the MUX[1:0] field is assumed equal to "00" in these CRC-8 calculations.

1.6.5 LCS PHY Layer

The LCS PHY provides a high-speed parallel transfer of data in the form of framed, fixed-sized segments that have been channelized across multiple media connections. Higher segment rates are also supported through trunking options, while lower segment rates are supported through multiplexing options.

The channelization, framing, and transmission functions of the LCS PHY have been specified to allow use of a variety of available Serdes (serialization/deserialization) and parallel fiber optics components. Portions of this specification will refer to examples of media types. The LCS PHY specification does not preclude other compatible options.

Channelization requirements include the mapping of segments to multiple channelized frames, channel skew specifications, and interchannel (segment) alignment operation.

Frame alignment, encapsulation, retiming and the required code-points for a parallel, full-duplex interface are included in the framing function.

Transmission requirements include the clocking requirements and electrical characteristics of the parallel interface.

The parallel interface used by the LCS PHY is based on widely available Serdes implementations that include their own endec (encoder/decoder) functionality. These components also fit well with the available and emerging parallel fiber optics components. Optionally, an LCS device can include the Serdes and 8B/10B Endec functions, but these are not strictly required by the LCS specification. In fact, LCS implementations that forego the use of optical transceivers or Serdes and endecs altogether for single board design are also compliant with the LCS specification.

1.6.5.1 Link Speed

The LCS PHY for ETT1 supports one in link speed option, 1.5 Gbaud. This link speed has various requirements that affect channelization and framing functions, discussed here and in the LCS specification. Serdes and optical transceiver components are available to support the 1.5 Gbaud links.

1.6.5.2 Segment Payload Options

The LCS Protocol for ETT1 uses an eight byte prepend and supports two segment payload sizes, a 64-byte payload with an eight byte LCS prepend (8+64 byte segment) and a 76-byte payload with an eight byte LCS prepend (8+76 byte segment). Discussion of these options is also covered in the channelization and framing sections here and in the LCS specification.

1.6.5.3 Segment Channelization

The LCS Physical Layer includes the external interface for receiving and transmitting LCS segments from and to the linecard. The interface consists of parallel “ganged” interfaces that each carry a serialized

portion of a segment called a frame. An LCS linecard or switch should therefore state the number of channels in a gang and the bytes per channel utilized for its particular implementation.

In the receive direction each channel receives a serial stream of 8B/10B coded data which can be decoded into a parallel byte stream plus two additional bits - one to indicate a control code-point and one for an errored code-point. In the transmit direction, an eight bit byte of parallel data plus one bit to indicate control word is sent through each channel as serialized 8B/10B encoded data.

The optional parallel interface has separate ingress and egress 10-bit parallel buses. The busses operate in source synchronous mode and so carry a clock with them to latch the data. A source synchronous transmit clock is driven from the LCS device interface to the Serdes devices. For data arriving from the remote LCS device, each Serdes device recovers the receive clock from the bit stream and provides it to the local LCS device's parallel bus receiver interface.

Table 15 shows the different external interface configurations supported by the ETT1. When utilizing the parallel interface, a Serdes device is used to convert between the byte-stream and the 8B/10B encoded serial stream. This Serdes will be a Single Data Rate (SDR) interface for the 1.5 Gbit/s interface.

Table 15. Fiber 1.5 Gbit/s per Channel Interface Configurations

Gang of Channels	Data Frame Size	Segment Size	Segment Time	Segments per second	Channel Data Rate	Channel Line Rate
12	6	72	40 ns	25 M	1.50 Gbits/s	1.5 SDR
14	6	84	40 ns	25 M	1.50 Gbits/s	1.5 SDR

Table 16. Mapping Segments to 1.5 Gbit/s Channels

Gang of Chnls	Data Frame Size	Channel 1	Channel 2	Channel 3	Channel 4	Channel 8	Channel 9	Channel 12	Channel 14	Channel 15
14	6	Prepend 0-5	Prepend 6-7, Pyld 0-3	Payload 4-9	Payload 10-15	Payload 34-39	Payload 40-45	Payload 58-63	Payload 70-75	
12	6	Prepend 0-5	Prepend 6-7, Pyld 0-3	Payload 4-9	Payload 10-15	Payload 34-39	Payload 40-45	Payload 58-63		

An LCS segment consists of an eight byte prepend and either a 64- or 76-byte payload. The start of the prepend is sent on the first channel, any remainder of the prepend is sent on the next channel, followed by the start of the payload. The remaining channels carry the subsequent bytes of the payload. The exact mapping of bytes to channels is shown Table 16.

An 8+64 byte segment is not explicitly supported when using the 2.5 Gbaud links. However, any smaller segment payload than 76 bytes can be supported by padding the fixed length segments to the full 76-byte payload.

1.6.5.4 Alignment, Framing, and Transmission Functions

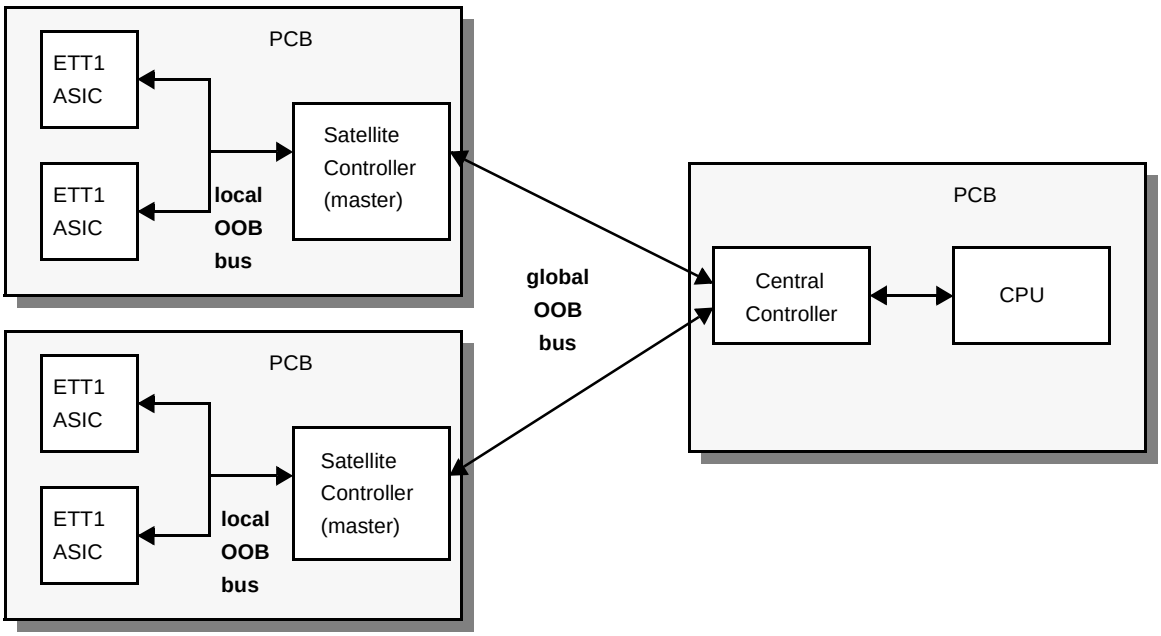
Details of the alignment, framing, and transmission function of the LCS Protocol are covered in the “LCS Protocol Specification -- Protocol Version 2”, available from PMC-Sierra, Inc.

1.7 THE OUT-OF-BAND (OOB) BUS INTERFACE

The Out-of-Band (OOB) bus provides a mechanism for a local CPU to control some or all ETT1 devices within a switch fabric. The OOB bus provides about 1Mbyte/sec of bandwidth which should be sufficient for the intended task. The OOB is a “local” bus in the sense that it only interconnects devices that are co-located on the same printed circuit board. A complete fabric will consist of many boards and therefore many separate OOB buses. The multiple OOB buses could be extended into a single logical bus via the use of a “satellite” bus controller on each board as shown in Figure 32. Alternatively, each OOB bus could be controlled by a local CPU and the CPUs could then communicate with the other CPUs via some other mechanism such as an Ethernet network. This document describes only the “local” OOB bus.

All of the ETT1 devices have an OOB interface. The ETT1 devices are slave devices and so there *must* be a local master device, such as a satellite controller, that initiates bus transactions.

Figure 32. One Implementation: The OOB Bus Consists of Both Local and Global Buses



1.7.1 The OOB Bus

The OOB bus is an eight-bit, tri-state, multiplexed address/data bus with separate control and interrupt signals.

1.7.1.1 OOB Bus Signals

Thirteen signals are used:

Table 17. OOB Control Bus Signals

Signal Name	Description	Driven by M(aster) or S(lave)
AD[7:0]	8 bit address/data bus. AD[7] indicates the type of cycle (1 = Write, 0 = Read) during the first address cycle, and is used for for address/data during other cycles. A[6:0] must provide the board select address when VALID_L is high.	Both. Address driven by master. Data supplied by master for a write cycle. Data supplied by Slave for a read cycle.
VALID_L	Bus cycle is valid. Active low.	Master.
WAIT_L	Wait. Active low. Initially asserted to indicate slave will respond. Then asserted, if necessary, to allow slave extra cycles to respond to current transaction.	Slave
CLK	Clock.	Clock source.
INT_HI	High priority interrupt	Slave
INT_LO	Low priority interrupt	Slave

NOTE: Active low signals are indicated by "_L".

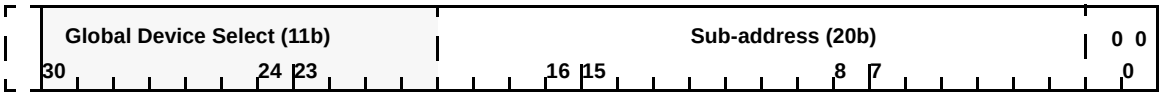
All signals are point to point except for the AD bus which is a multipoint bus. In particular, WAIT_L and the interrupt signals are not open collector/gate drivers.

1.7.1.2 Address Space

The OOB bus provides 31 bits of address - A[30:0]. Of these, bits A[1] and A[0] are always 0 as the bus does not support 8- or 16-bit accesses: all accesses are 32-bits wide, and must be aligned on quad byte boundaries.

Bit A[31] is used to indicate the cycle type: Write or $\overline{\text{Read}}$.

The remaining 31 bits are divided into a device select block and a sub-address block.



The global device select block is 11 bits and is used to identify a target device. The sub-address is used by the target device to identify a register or memory location within the device.

The device select block is divided into three categories as follows:

0x7FF	**Broadcast Write**. All devices should respond to this device select value but will choose to ignore the write if they do not support a broadcast write to the specified sub-address. All devices assert WAIT_L until their local write operation completes.
0x7F0 to **0x7FE**	**Multicast** device selects. (15 multicast groups). See Table 18 below for groups.
0x0 to **0x7EF**	**Individual** device selects. See Table 19 for details of one possible address scheme.

The current multicast groups are shown in Table 18.

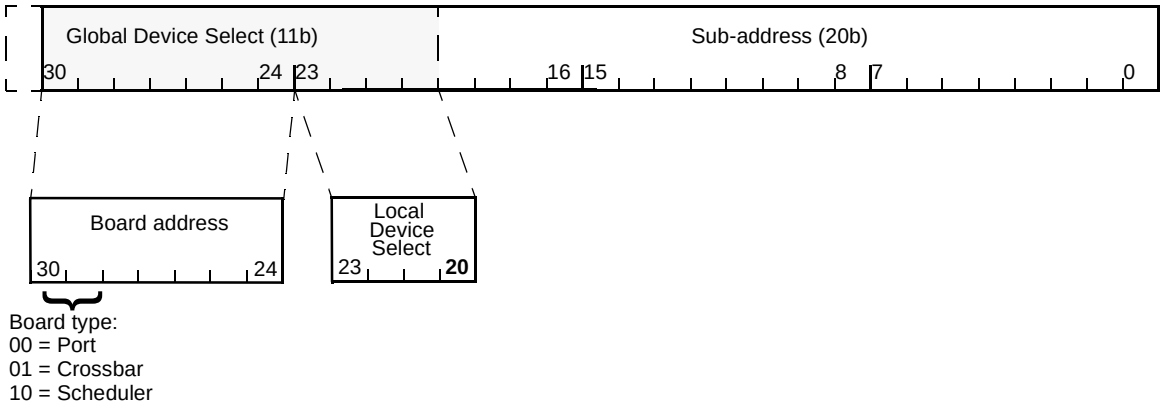
Table 18. Multicast Group Selects

Group	Device select value (11bits)
All EPPs	0x7FE
All Dataslices	0x7FD
All Schedulers	0x7FC
All Crossbars	0x7FB
All Satellites (Not part of the ETT1 Chip Set)	0x7F0

1.7.1.3 Individual Device Selects

The individual device selects (30:20) are hierarchical: the most significant seven bits (30:24) are the board address and identify a single board (PCB) in the system; the four least significant bits (23:20) determine which particular device on a board is being selected. In addition, the two most significant bits (30:29) indicate a board type: Port board, Crossbar board or Scheduler board. Figure 33 shows this organization.

Figure 33. The Global Device Select Defines the Board Number and the Device on Each Board



At power-on the ETT1 devices do not know which board they are on. They learn the board address (bits 30:24) by continuously observing the OOB bus itself: every board must have an OOB controller which must drive the unique board address on AD[6:0] whenever VALID_L is not active. The OOB controller must continue to do this while the system is active. Each port board (and EPP) obtains its port number by looking at AD[4:0] when VALID_L is not active, which is why AD[6:0] must be valid whenever VALID_L is inactive.

Each device obtains its own local device select (bits 23:20) either from static pins on the device, or through a preset address. For the Enhanced Port Processor, Crossbar, and Scheduler devices, four dev_sel pins (oobdev_sel[3:0]), are tied to logical 1 or 0 to provide the four least significant bits of the unique device address. For the Dataslice device, three dev_sel pins (oobdev_sel[2:0]) are tied to logical 1 or 0 to provide three of the four least significant bits of the unique device address. Since there are two logical Dataslices within each package, the fourth implied bit is used to select which of the two logical Dataslices is addressed.

The combinations of permissible board addresses and device addresses are shown in Table 19.

NOTE: If the satellite OOB controller needs its own address within the OOB space then it can use 0xPPe (i.e. local device select = 0x00e).

Table 19. Unicast Group Selects

Devices	Address Format (bin)	Address range allocated
EPP on port P (P=0..31)	{00P_PPPP_1111}	0x00f, 0x01f, ... 0x1ff
Dataslice D (D=0..13) on port P (P=0..31)	{00P_PPPP_DDDD}[a]	0x000-0x00d,..0x1f0-0x1fd
Scheduler S (S=0,1)	{100_000S_0000}	0x400 or 0x410
Crossbar C (C=0..31)	{010_CCCC_000C}[b]	0x200, 0x201, 0x210...0x2f1

a. In the ETT1 reference system design, P_PPPP corresponds to the port number (0-1f hex) and DDDD corresponds to the Dataslice number (0-d hex)
b. In the ETT1 reference system design, the first four bits of the CCCC_000C designation correspond to the crossbar board number, and the last bit of the CCCC_000C designation specifies which of the two crossbar chips on that board is being addressed. Note that in the reference design, logical crossbars 0 and 6 are on the board with CCCC=0000, logical crossbars 1 and 7 are on the board with CCCC=0001, etc.

1.7.1.4 Port Board Assignment

The port boards must have unique board select addresses in the range 0 to 31. These addresses identify the logical port number of a port for LCS purposes. The assignment of port addresses is not random and must reflect the physical connections made between the port boards and the crossbar boards. For example, one port board will have connections to port 6 on every Crossbar device and port 6 on every Scheduler. That port board must then have a board address of 6.

1.7.1.5 Bus Cycles

Two types of bus cycles are supported: write and read. Each type may have any number of wait states before the cycle completes.

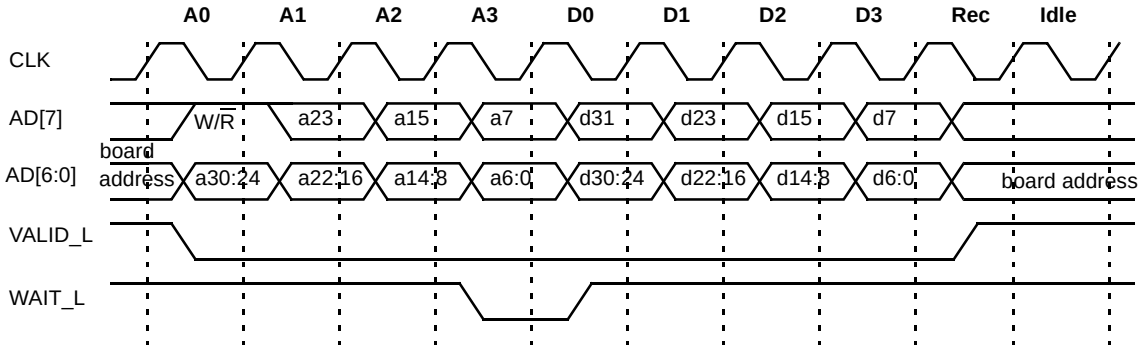
1.7.1.6 A Write Cycle - No Wait

A simple 32-bit write. The master drives out a 31-bit address, 8 bits at a time, most significant bits first; AD[7] is high to indicate a write operation. VALID_L is asserted by the master to indicate the new cycle.

NOTE: The target slave asserts WAIT_L during the final address cycle to indicate that the target is active. The target then de-asserts WAIT_L with the first data cycle. The master must de-assert VALID_L for at least one cycle before re-asserting it for the next cycle.

Whenever VALID_L is de-asserted the master *must* drive the local board address onto AD[6:0].

Figure 34. Write Cycle - No Wait



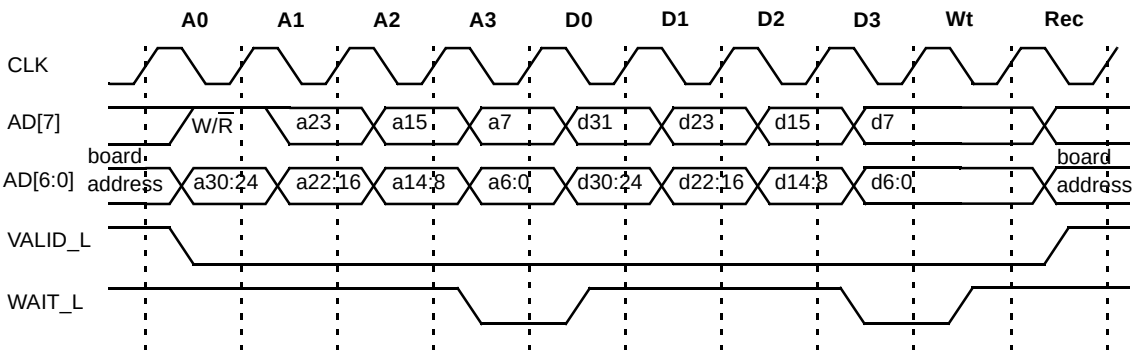
Addresses are presented during cycles A0-A3. The write data is presented during D0-D3. There is then a recover cycle during which the target performs the write.

1.7.1.7 A Write Cycle - One Wait

If, during D3, the target decides that it may not be able to complete the write immediately, then it must assert WAIT_L. It continues to assert WAIT_L until the write is completed or WAIT_L has been asserted for 256 cycles, at which time WAIT_L is de-asserted. (If WAIT_L is asserted for 256 cycles then the master will consider this to be a bus error).

Figure 35. shows a write cycle with the target forcing a one cycle wait. The target slave asserts WAIT_L during A3 to indicate that the target is active. The target then de-asserts WAIT_L and re-asserts it during D3. The master must maintain d7:0 until WAIT_L is de-asserted.

Figure 35. Write Cycle - One Wait

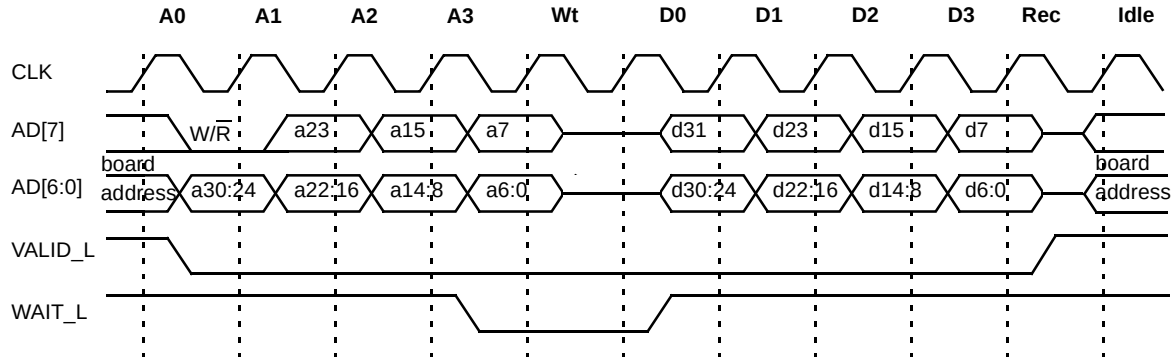


1.7.1.8 A Read Cycle - One Wait

A simple 32-bit read. The master drives out a 31-bit address, 8 bits at a time, most significant bits first; AD[7] is driven low in the first cycle to indicate a read cycle. VALID_L is asserted by the master to indicate the new cycle. The target slave asserts WAIT_L during A3 to indicate that the target is active. The target

keeps WAIT_L asserted through bus turn-around cycle (effectively a wait cycle). The target de-asserts WAIT_L once the first valid data byte is on the bus. VALID_L must be high for at least two rising clocks before the next cycle. There is an additional bus turn-around cycle during the Recover cycle. This is the fastest read cycle: there is *always* at least one bus turn-around cycle whenever AD is tri-stated. The master must drive the AD signals before the next rising edge of the clock after the VALID_L is deasserted.

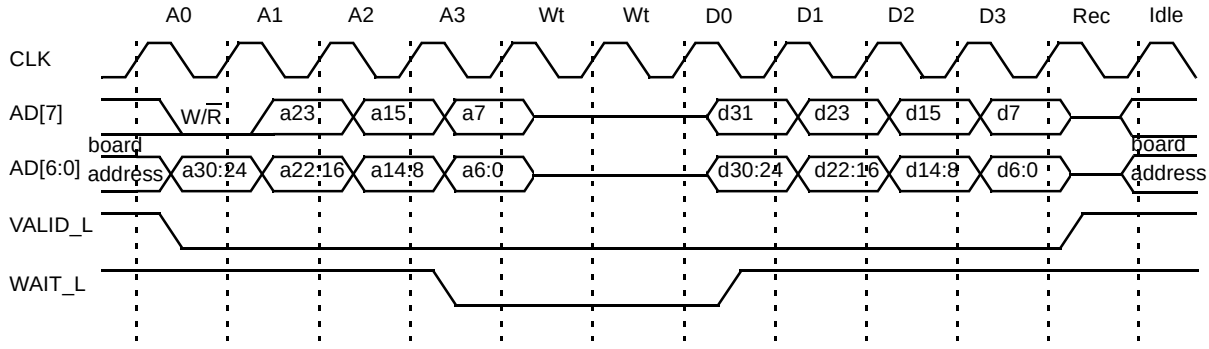
Figure 36. Read Cycle - One Wait



1.7.1.9 A Read Cycle - Two Waits

This is the same as a single wait read cycle except that there is an extra wait cycle before D31:24 are valid and WAIT_L is de-asserted.

Figure 37. Read Cycle - Two Waits

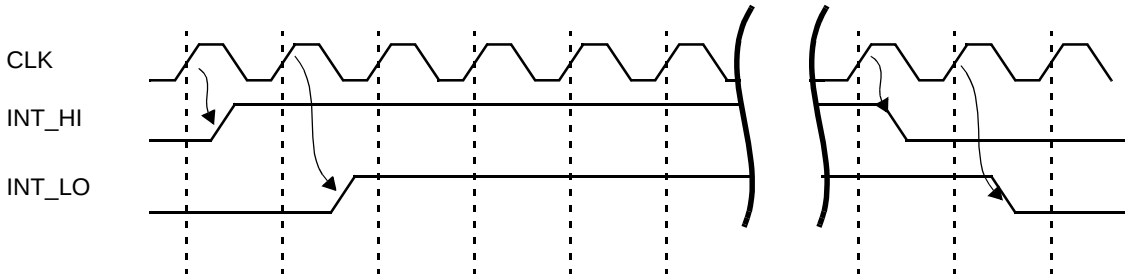


1.7.1.10 Interrupts

Two active high interrupt lines are provided. INT_HI is for “emergency” interrupts such as a card failing. INT_LO is for non-emergency interrupts such as a device requesting to be polled for some reason. Interrupts are asserted and de-asserted synchronously with respect to the clock. Interrupts can be asserted at any time, even during an access.

The bus master *must* clear an interrupt by reading an appropriate register within the device that has asserted the interrupt.

Figure 38. Interrupts are Active High and Synchronous



Bus Rules

1. The target slave *must* assert WAIT_L during the final address cycle. If the master does not see WAIT_L asserted during the final address cycle then it concludes that there is no device at that address and terminates the cycle (takes VALID_L high).
2. VALID_L *must* be de-asserted (high) for at least one clock cycle between accesses. The master *must* drive the board address on AD[6:0] during a positive clock edge while VALID_L is de-asserted (high).
3. A slave *may* insert up to 255 wait states before it *must* de-assert WAIT_L. Failure to de-assert WAIT_L within that time will result in a Bus Error. (The master will terminate the cycle by de-asserting VALID_L)

1.8 INITIALIZATION PROCEDURE

Every ETT1 component must be correctly initialized before being used within a switch system. In general, the process of initialization depends on whether the whole switch is being initialized (initial power-on sequence) or whether a component is being added to a switch system that is already operating. However, to simplify the process, this document describes an initialization sequence that can be used in either case, but does incur some redundant operations when used at system initialization.

The initialization sequence is also a function of the physical organization of the ETT1 devices. The sequence described here assumes the physical configuration that has been used in the ETT1 Reference System: each port board has one Enhanced Port Processor and six or seven Dataslice devices; each Crossbar board has two Crossbar devices; each Scheduler board has one Scheduler device. The system has redundant Crossbars and Schedulers. The link between the linecards and the ETT1 ports uses industry standard Serdes devices.

NOTE: For more information on the link synchronization involving the Serdes and the fiber optics, see Appendix C, Section 2 “The 8b/10b Interface to the Dataslice” on page 328.

1.8.1 Initial Power-on:

- Wait for voltage ramp-up
- Suggested:
  - Initialize the OOB satellite FPGA
- Reset Enhanced Port Processor
- Reset Dataslices
- Reset Scheduler
- Reset Crossbar
- Enable PLLs on all devices (turn off PLL reset)
- Program Dataslices' DINBY and DOUTBY to 3
- Set interrupt masks (disable all except AIB ready up and external link interrupts) on EPP & DSs
- Set primary Crossbar in DSs to one that exists
- Program 8b10b tables of DSs
- Program pre-defined switch to Linecard Control Packets location in all DSs. (See section 3.3, table 28)
- EPP Waiting Scheduler Request Count Memory EWSRCT address offset 1C000-1DFFCh: After resetting an EPP, write 00000000h to address offsets 1D000h through 1D7FCh in that EPP (a total of 512 OOB writes). Whan all EPPs are resetsimultaneously (using a broadcast OOB write to the EPP Reset and Control register), 512 broadcast OOB writes can be used to reset those locations in all EPPs.
- Release VCSEL and PIN diode resets on DSs
- Suggested:
  - enable clock phase shifting on Serdes,
  - configure Serdes loopback controls for normal operation,
  - enable Serdes synchronization,
  - send idle-not-rdy from linecard
- Enable DS 8b10b encoder

- Enable DS 8b10b decoder
- Suggested:
    - disable Serdes clock phase shifting when sync is established
- Set Scheduler BP depth field of Control register to 0x14.
- Set Scheduler action bits (Configuration dependent. See "Enhanced Port Processor - Scheduler" on page 83)
- Enable the Flow Control Crossbar sync from the Scheduler
- Reset DS AIB (for all ports present)
- Reset Crossbar AIB (for all ports present)(includes Flow Control Crossbars)
- Reset EPP AIB (for all ports present)
- Reset Scheduler AIB (for all ports present)
- Enable DS AIB (for all ports present)
- Enable Crossbar AIB (for all ports present)(includes Flow Control Crossbars)
- Release reset on DS AIB
- Release reset on Crossbar AIB
- Release reset on EPP AIB
- Release reset on Scheduler AIB

These sequences should complete with only the "ready up" interrupts enabled. When these interrupts occur (which may be during later parts of the above sequences) then perform the following:

When DS AIB ready up:

- Inform local processes that the DS is operational
- Enable all interrupts on DS, except for Transceiver Comma Detect and 8b10b Decoder Comma Detect which are set every time an idle is received.

When EPP AIB ready up for at least one Scheduler and at least one set of Flow Control Crossbars:

- Inform local processes that the EPP is operational
- Enable all interrupts on EPP

- Set Fault Tolerance Control
- Enable Port Processor
- Define 1-in-N Idle Count value
- Take EPP out of LCS Stop mode

When Crossbar AIB ready up:

- Inform local processes that the Crossbar is operational
- Enable all interrupts on Crossbar

When Scheduler AIB ready up:

- Inform local processes that the Scheduler is operational
- Enable all interrupts on Scheduler
- Enable ports that are present and up
- Enable non-TDM traffic
- If TDM traffic, define TDM control and Enable TDM traffic

In a system with redundant Schedulers, it is essential to perform a refresh operation as the final step, once ports are configured and operational. This synchronizes the two Schedulers. Refresh schedulers if necessary (do not refresh if only one in system)

1.8.2 Port Board being added:

- Wait for voltage ramp-up
- Suggested:
  - Initialize the OOB satellite FPGA
- If a Scheduler is present and has its Control register's CRC Action bit set, save and clear the CRC Action bit.
- Reset Enhanced Port Processor
- Reset Dataslices
- Enable PLLs on EPP and DSs (turn off PLL reset)

- Program Dataslices' DINBY and DOUTBY to 3
- Set interrupt masks (disable all except AIB ready up and external link interrupts) on EPP & DSs
- Set primary Crossbar in DSs to one that exists
- Program 8b10b tables of DSs
- Program pre-defined switch to Linecard Control Packets location in all DSs. (See section 3.3, table 28)
- EPP Waiting Scheduler Request Count Memory EWSRCT address offset 1C000-1DFFCh: After resetting an EPP, write 00000000h to address offsets 1D000h through 1D7FCh in that EPP (a total of 512 OOB writes). Whan all EPPs are resetsimultaneously (using a broadcast OOB write to the EPP Reset and Control register), 512 broadcast OOB writes can be used to reset those locations in all EPPs.
- Release VCSEL and PIN diode resets on DSs
- Suggested:
    - enable clock phase shifting on Serdes,
    - configure Serdes loopback controls for normal operation,
    - enable Serdes synchronization,
    - send idle-not-rdy from linecard
- Enable DS 8b10b encoder
- Enable DS 8b10b decoder
- Suggested:
    - disable Serdes clock phase shifting when sync is established

If neighboring Crossbar cards are present:

- Reset DS AIB and Crossbar AIB
- Enable DS AIB and Crossbar AIB
- Release reset on DS AIB and Crossbar AIB

If neighboring Flow Control Crossbar cards are present:

- Reset EPP AIB and Flow Control Crossbar AIB

- Enable EPP AIB and Flow Control Crossbar AIB
- Release reset on EPP AIB and Flow Control Crossbar AIB

If neighboring Scheduler cards are present:

- Reset EPP AIB and Scheduler AIB
- Enable EPP AIB and Scheduler AIB
- Release reset on EPP AIB and Scheduler AIB
- Restore the Scheduler Control register CRC Action bit if it was set before initializing this Port Board.

These sequences should complete with only the "ready up" interrupts enabled. When these interrupts occur (which may be during later parts of the above sequences) then perform the following:

When DS AIB ready up:

- Inform local processes that the DS is operational
- Enable all interrupts on DS, except for Transceiver Comma Detect and 8b10b Decoder Comma Detect which are set every time an idle is received.

When EPP AIB ready up for at least one Scheduler and at least one set of Flow Control Crossbars:

- Inform local processes that the EPP is operational
- Enable all interrupts on EPP
- Set Fault Tolerance Control
- Enable Port Processor
- Define 1-in-N Idle Count value
- Take EPP out of LCS Stop mode

1.8.3 Scheduler Board being added:

- Wait for voltage ramp-up
- Suggested:
    - Initialize the OOB satellite FPGA

- Reset Scheduler
- Enable PLLs on SCH (turn off PLL reset)
- Set BP depth field of Control register to 0x14.
- Set Scheduler action bits (Configuration dependent. See "Enhanced Port Processor - Scheduler" on page 83)
- Enable the Flow Control Crossbar sync
- Set interrupt masks (disable all expect ready up)

If neighboring Port cards are present:

- Reset EPP AIB and Scheduler AIB (for all ports present)
- Enable EPP AIB and Scheduler AIB (for all ports present)
- Release reset on EPP AIB and Scheduler AIB (for all ports present)
- Reset ports on Scheduler

These sequences should complete with only the "ready up" interrupts enabled. When these interrupts occur (which may be during later parts of the above sequences) then perform the following:

When Scheduler AIB ready up:

- Inform local processes that the Scheduler is operational
- Enable all interrupts on Scheduler
- Enable ports that are present and up
- Enable non-TDM traffic
- If TDM traffic, define TDM control and Enable TDM traffic

In a system with redundant Schedulers, it is essential to perform a refresh operation as the final step, once ports are configured and operational. This synchronizes the two Schedulers. Refresh schedulers if necessary (do not refresh if only one in system)

1.8.4 Crossbar Board being added:

- Wait for voltage ramp-up
- Suggested:

- Initialize the OOB satellite FPGA

- Reset Crossbar
- Enable PLLs on Crossbar (turn off PLL reset)
- Set interrupt masks (disable all except ready up)

If neighboring Port cards are present and this is a Flow Control Crossbar:

- Reset EPP AIB and Flow Control Crossbar AIB (for all ports present)
- Enable EPP AIB and Flow Control Crossbar AIB (for all ports present)
- Release reset on EPP AIB and Flow Control Crossbar AIB (for all ports present)

If neighboring Port cards are present and this is a data Crossbar:

- Reset DS AIB and Crossbar AIB (for all ports present)
- Enable DS AIB and Crossbar AIB (for all ports present)
- Release reset on DS AIB and Crossbar AIB (for all ports present)

These sequences should complete with only the "ready up" interrupts enabled. When these interrupts occur (which may be during later parts of the above sequences) then perform the following:

When Crossbar AIB ready up:

- Inform local processes that the Crossbar is operational
- Enable all interrupts on Crossbar

1.9 FAULT TOLERANCE

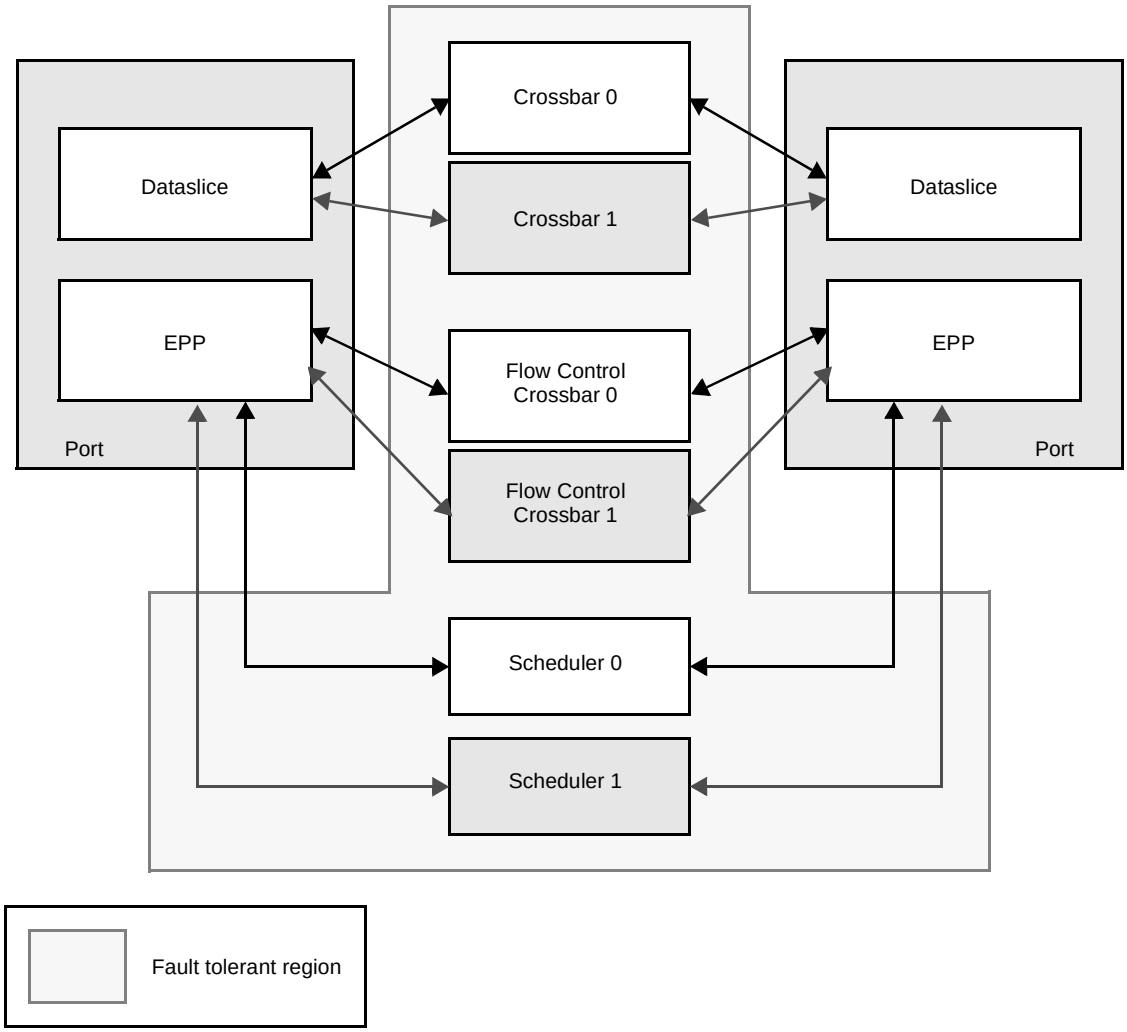
A ETT1 fabric can optionally be configured with redundant elements. When the ETT1 system contains redundant Schedulers and Crossbars, it is capable of sustaining certain faults and yet not lose or corrupt any of the cells within the fabric.

1.9.1 Fault Tolerance Model

The fault tolerance model assumes that any failure within a redundant element will manifest itself as a link error, and thus will be observable by the local CPU. The redundant elements are the Scheduler and Crossbar which are connected via AIB links to the Enhanced Port Processor and Dataslice, as shown in Figure 39. These AIB links are protected by a CRC such that a failure in a device will likely result in a CRC error, which can be detected by the local CPU through interrupts. Furthermore, even if a CRC error is not

generated, the Enhanced Port Processor and Dataslice will detect any difference in the information received from the two, supposedly identical elements and will act as though an error had occurred.

Figure 39. Redundant Connections



In this section, we consider two types of errors that may occur: soft errors and hard errors. Soft errors are transient errors that do not require replacement of any of the boards. Hard errors are caused by a device failure resulting in a defective board which should be replaced. On the AIB links a soft error would appear as a transient CRC error; a hard error would be indicated by the link ready signal going inactive (see Section 1.10.3 “AIB Links (A)” on page 103).

The significance of this fault model is that all faults associated with redundant elements will appear as a link error of some form. So the fault detection and recovery procedures for any ETT1 fabric can be

specified by considering the various AIB links. The following sections explain these procedures for redundant and non-redundant configurations.

1.9.2 Soft/Transient Errors

Intermittent CRC errors between the Dataslice and Crossbar and the Enhanced Port Processor and Scheduler are considered in this section. The corrective action that needs to take place due to these errors are different for non-redundant and redundant configurations.

1.9.2.1 Non-Redundant Configurations

Dataslice - Crossbar

In a non-redundant Crossbar configuration, CRC errors that occur between the Dataslice and Crossbar imply that part of a cell has been corrupted. In Figure 40, Dataslice 1 and Dataslice 2 are connected to Crossbar 0.

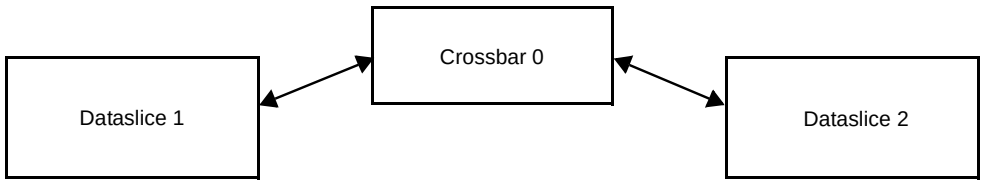
If a CRC error occurs from Dataslice 1 to Crossbar- 0 when valid data is to be transferred, then that slice of information for that cell has been corrupted. As a result, Dataslice 2 will receive an invalid cell. Crossbar 0 will indicate to the CPU that a CRC error has occurred.

If Dataslice 2 receives a CRC error from Crossbar-0 when valid data is to be transferred, then the cell has been corrupted. Dataslice 2 will indicate to the CPU that a CRC error has occurred.

In both of the above, if the corrupted cell did contain cell data (as opposed to being an empty cell) Dataslice 2 will store the corrupted information in its local memory and forward it to the egress linecard. The ETT1 fabric will not make the decision to discard the cell; the linecard must detect the error and discard the cell. The linecard would detect such an error with it's own payload error detection scheme.

If the CRC error occurred in an empty cell then no information is lost. However the occurrence of a CRC error is always indicated to the local CPU via a maskable interrupt.

Figure 40. Simple Non-Redundant Crossbar Configuration



Enhanced Port Processor - Flow Control Crossbar

If CRC errors occur between the Flow Control Crossbar device and the EPP, the input EPP will not have the correct view of the available space in the egress queues of the output EPP. Thus, a refresh of the output queue credits must be performed. The invalid incremental credit interrupt register is used to indicate which input port to output port flow has lost output queue credits. For example, if port 5's invalid incremental credit interrupt register has a value of 0x400, then there are lost credits for all Unicast flows from input port 5 to output port 10. See Section 1.9.3 "Flow Control Refresh Procedure" on page 97.

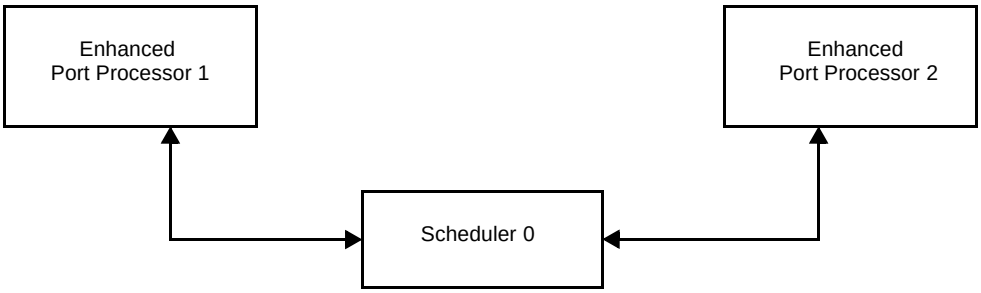
Enhanced Port Processor - Scheduler

In a non-redundant Scheduler configuration, CRC errors that occur between the Enhanced Port Processor and Scheduler might result in inconsistent state information. The Scheduler must have accurate information on the number of outstanding requests as well as the backpressure state of all VIQs. A lost request or grant can cause a discrepancy between the Enhanced Port Processor and Scheduler. Figure 41 describes a simple non-redundant Scheduler configuration where Enhanced Port Processor 1 and Enhanced Port Processor 2 are connected to Scheduler 0.

If Enhanced Port Processor 1 receives a CRC error on information from Scheduler 0, then Enhanced Port Processor 1 does not know if that information contained a valid grant. If the corrupted grant was for unicast traffic, then the Enhanced Port Processor would have one more request than the Scheduler; this might delay the forwarding of one cell. However, if the corrupted grant was for multicast traffic, then the next multicast grant might cause a cell to be sent to the wrong destination port. This is considered unacceptable and so a CRC error from Scheduler-0 makes Enhanced Port Processor 1 ignore all subsequent Scheduler grants until the state information can be restored. As always, the CRC error is indicated to the local CPU which must refresh Scheduler 0 before traffic can continue to flow to/from this port. The Enhanced Port Processor continues to receive routing tags from the Scheduler, and will forward cells received from other ports.

If Scheduler 0 receives a CRC error on information from Enhanced Port Processor- , then Scheduler 0 may have lost a new request or backpressure information. If the lost information was a unicast request, then the Scheduler would have one less request than the Enhanced Port Processor. This might delay the forwarding of one cell. If the lost information was a multicast request, the Scheduler might send a cell to the wrong destination port. If back pressure assertion was lost, the Scheduler could possibly overflow the output queues. These behaviors are considered unacceptable. Thus, when Scheduler 0 detects a CRC error it disables traffic to/from Enhanced Port Processor 1 until the CPU refreshes Scheduler 0's state information. Traffic to/from other ports continues to flow. The CRC error is indicated to the CPU which must then refresh the state information in Scheduler 0. See Section 1.9.4 "Scheduler Refresh Procedure" on page 98.

Figure 41. Simple Non-Redundant Scheduler Configuration



1.9.2.2 Redundant Configurations

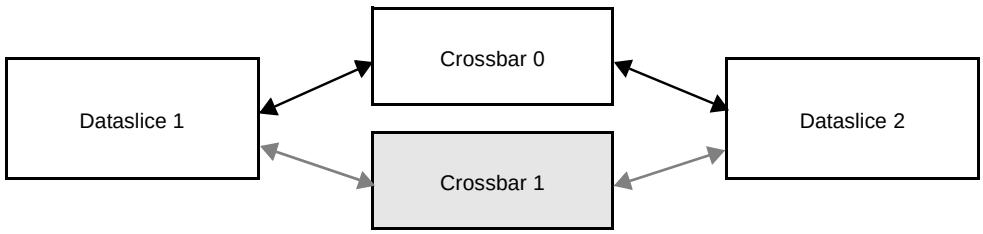
For a redundant Crossbar configuration, the redundant connection provides the uncorrupted information between the Dataslice and Crossbar. In Figure 42, Dataslice 1 and Dataslice 2 are connected to both Crossbar 0 and Crossbar 1. The same information is sent from the Dataslices to each of the Crossbars, and therefore, the same information is received by the Dataslices from each of the Crossbars.

Consider when a CRC error occurs from Dataslice 1 to Crossbar 0, and no CRC error occurs from Dataslice 1 to Crossbar 1. In this situation, Dataslice 2 will receive invalid information from Crossbar 0, but will receive valid information from Crossbar 1. Dataslice 2 uses the valid information and no cells are corrupted. Crossbar 0 will indicate to the CPU that a CRC error has occurred.

If a CRC error occurs from Crossbar 0 to Dataslice 2, Dataslice 2 would not receive valid information from Crossbar-0. Since Crossbar 1 receives the same information as Crossbar 0, Dataslice 2 will get the valid information from Crossbar 1. Dataslice-2 uses the valid information and no cells are corrupted. Dataslice 2 will indicate to the CPU that a CRC error has occurred.

In both situations, the simple redundant Crossbar configuration has prevented cell loss in the event of a single CRC error. The occurrence of a CRC error between the Dataslice and Crossbar is always noted, but no corrective action is required.

Figure 42. Simple Redundant Crossbar Configuration



Enhanced Port Processor - Flow Control Crossbar

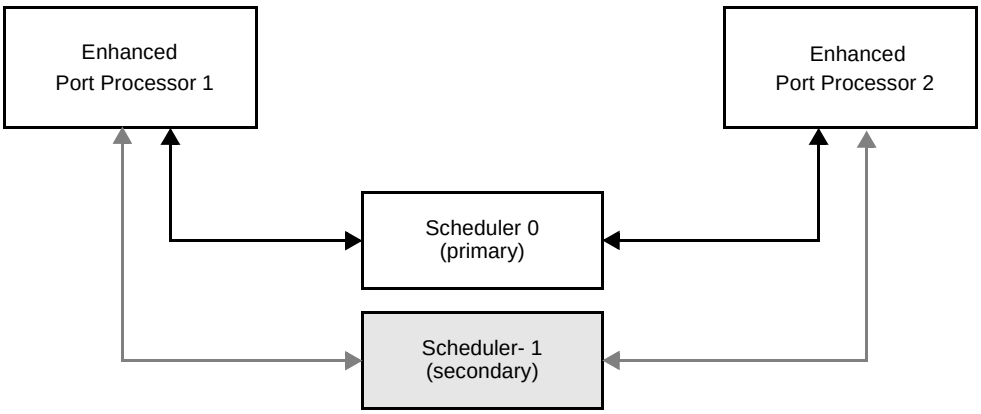
The redundant configuration is equivalent to that of the Dataslice to Crossbar configuration: A single CRC error on one of the links can be ignored because the EPP will automatically use the valid information from the other link. However, if the EPP receives four consecutive CRC errors on its primary FC crossbar link then it will consider the link to have failed, and will make the other link the new primary link. See the EFTCTRL register in the EPP for more details.

If CRC errors occur on both links during the same cell time then information will be lost, and the recovery process described in the non-redundant configuration must be used to restore the correct state in the input EPP.

Enhanced Port Processor - Scheduler

In a redundant Scheduler configuration, the redundant connection provides the uncorrupted information between the Enhanced Port Processor and Scheduler. Figure 43 describes a simple redundant Scheduler configuration where Enhanced Port Processor 1 and Enhanced Port Processor 2 are connected to both Scheduler 0 and Scheduler 1. In this example, Scheduler 0 is the primary Scheduler and Scheduler 1 is the secondary Scheduler.

Figure 43. Simple Redundant Scheduler Configuration



If, due to an error, the two Schedulers send different information to the EPPs then it is clearly essential that all of the EPPs must use the information from the same Scheduler. Therefore, all of the Enhanced Port Processors must use a consistent setting for their primary/secondary Scheduler, which is defined in the EFTCTRL register in the EPP. The corrective action taken is different for CRC errors to/from the primary Scheduler and CRC errors to/from the secondary Scheduler. These are described below.

If Enhanced Port Processor 1 receives a CRC error on information from Scheduler 0 (primary), then it uses the uncorrupted grant information from Scheduler 1. Similarly, if Enhanced Port Processor 1 receives a CRC error on information from Scheduler 1, then it uses the information from Scheduler 0. In both these case, the CPU is notified of the error, but no corrective action is required by the CPU since the state information remains consistent. However if the EPP sees four consecutive CRC errors on the link from its

current primary Scheduler then it will automatically consider the other Scheduler as the new primary Scheduler.

If Scheduler 0 (primary Scheduler) receives a CRC error on information from Enhanced Port Processor 1, then Scheduler 0 and Scheduler 1 might differ in their state information. In this situation Scheduler 0 immediately disables its AIB links to Enhanced Port Processor 1 and Enhanced Port Processor 2 (and all Enhanced Port Processors). The Enhanced Port Processors immediately detect the loss of connection to Scheduler 0 and automatically select Scheduler 1 (secondary Scheduler) to be their new primary Scheduler. Without CPU intervention, the system has disabled the primary Scheduler, and switched the Enhanced Port Processors to use the secondary Scheduler. The CPU is informed of this error, and must undertake the process to refresh the Schedulers.The EPP's may register Scheduler mismatch interrupts until the Schedulers are refreshed. The AIB links to other EPPs are not affected.

If Scheduler 1 (secondary Scheduler) receives a CRC error on information from Enhanced Port Processor 1, then Scheduler 1 immediately disables its AIB links to Enhanced Port Processor 1. The Enhanced Port Processors do not alter their primary/secondary setting, since Scheduler 0 is already selected as the primary Scheduler. The CPU is notified of this error, and must go through the steps to refresh the Schedulers.

The actions the Scheduler performs upon detecting a CRC error are different in the redundant and non-redundant configurations. Consequently the Scheduler must be configured to be in redundant or non-redundant mode. This configuration is accomplished via the CRC Action bit in the Scheduler's control register (SCTRLRS). The CRC Action bit should *only* be set to 1 for the primary Scheduler in a redundant configuration.

1.9.3 Flow Control Refresh Procedure

The previous section refers to the process of refreshing flow control state. The purpose of this procedure is to insure consistency of the state information between the iEPPs and the oEPPs. In the non-redundant Flow Control Crossbar configuration, the refresh procedure ensures that iEPPs will have correct queue depth information about oEPPs

The refresh procedure is as follows:

1. Detect invalid incremental credit interrupt on port *i.* Read invalid incremental credit interrupt register on port *i*. The value specifies to which output port *j* flow could have missing output queue credits.
2. Write freeze unicast Output Scheduler flows register on port *j* with a value of $1 \ll i$. This will cause the Output Scheduler on port *j* to stop sending cells from port *i*.
3. Disable the non TDM enable for port *i* in the Scheduler.
4. Freeze the Scheduler Request Modulator on port *i*.
5. Read the output unicast queue information memory (queue length) on port *j* for queues corresponding to input port *i*. If port *j* is in OC-192c mode, then 4 queue lengths are stored. If port *j* is in OC-48c mode, then 16 queue lengths are stored.
6. Read the waiting scheduler request count memory on port *i* for queues corresponding to output

port *j*. The number of request counts stored should be the same as above.

7. Add the stored queue length with the waiting scheduler request count and write that value into the output unicast queue debit count memory for each corresponding *i,j* output queue. The number of write accesses should be 4 in OC-192c mode and 16 in OC-48c mode.
8. At this point in time, the output unicast queue debit count memory is up to date, and the unicast Output Scheduler can be unfrozen, the non TDM traffic enabled, and the scheduler request modulator unfrozen.

1.9.4 Scheduler Refresh Procedure

A previous section refers to the process of refreshing a Scheduler. The purpose of this procedure is to ensure consistency of the state information between the Enhanced Port Processors and the Scheduler(s). In the redundant scheduler configuration, the refresh procedure ensures that the two Schedulers will contain identical state information and make the same scheduling decisions.

The refresh procedure is as follows:

1. Disable non-TDM traffic in the Scheduler(s): set the enable non-TDM traffic register (SNTDMEN, 0x80) to 0x0. Do not modify the Enable Port register (SENBPRT,0x38). Do not modify the TDM traffic register.
2. Issue the “Go Refresh” command to the Enhanced Port Processors. The EPPs then send their state information to the Scheduler(s).
3. Wait until the refresh process has completed: for each port that issued a “Go Refresh” command, wait until the appropriate bit in the Scheduler’s ‘Port is Refreshed’ register (SRFRS, 0x94) is set to 1.
4. Enable non-TDM traffic in the Scheduler(s): set the enable non-TDM traffic register (SNTDMEN, 0x80) to 1 for each port that is enabled.

   NOTE: An ETT1 Scheduler device will not schedule best-effort traffic until a “Go Scheduler” command is sent from all of the Enhanced Port Processors.

   NOTE: Extra writes are required at this point. Refer to the “Scheduler Device Errata, issue 3” document, available from PMC-Sierra, Inc.

5. Issue the “Go Scheduler” command to all of the Enhanced Port Processors that issued “Go Refresh”. This command synchronizes the two Schedulers to start scheduling best-effort traffic at the same time.
6. If the EPP’s EFTCTRL register’s “Ignore Central Scheduler Grants” or “Freeze SRM” bits are set, clear these bits.
7. The Scheduler(s) now contain the same state information as the EPPs and will start scheduling cells. In the redundant Scheduler configuration, the Schedulers will also make identical scheduling decisions.

1.9.5 Refresh Schedulers After Modifying Registers

The CPU cannot guarantee to modify the same register in both Schedulers at exactly the same instant, even using an OOB multicast write to all Schedulers. Consequently there are certain Scheduler registers which, when modified, require a refresh process to occur (assuming a system with two Schedulers). Table 20 list these registers.

Table 20. Refresh-sensitive Registers in the Scheduler

Address	Access	Register	Symbol
00004h	Read/Write	Control and Reset	SCTRLRS
00038H	Read/Write	Enable Port	SENBPRT
00080h	Read/Write	Enable non-TDM Traffic	SNTDMEN
00084h	Read/Write	Enable TDM Traffic	STDMEN
00088h	Read/Write	Reset Port	SPORTRS
0008Ch	Read/Write	Freeze Port	SPORTFRZ
0009Ch	Read/Write	TDM Control	STDMCTRL
00100h	Read/Write	PLL control/status	SPLL

1.9.6 Hard Errors

This section describes the sequence of events that need to take place to correctly replace a failed board. When a board is replaced, it must be resynchronized with an already running system. In this section, we address each type of board replacement and describe how to resynchronize the board.

1.9.6.1 Port Board

When a port board fails, the Schedulers must disable traffic flowing to/from this port. All the cells currently stored in the input and output queues of the failed port board will be lost. These are the following steps required to replace the failed port board.

NOTE: The failure of the port board may also cause the primary Scheduler to shut down.

1. Detect failed port board.
2. Disable TDM and best-effort traffic in the Scheduler(s) to/from this port.
3. Disable this port in the Scheduler's SENBPRT register.
4. Disable the AIB links to/from the Crossbars and Schedulers.
5. Replace failed port board.

6. Bring up new board. See “Port Board being added:” on page 87.
7. Enable TDM traffic in Scheduler for this port. TDM Traffic can now flow to/from this new port board.
8. Refresh state information from this new port to Schedulers. (See Section 1.9.4 “Scheduler Refresh Procedure”) This is only necessary if the linecard sends traffic before the link to the Scheduler board comes up. Re-establish primary and secondary Schedulers in every port.
9. Best-effort traffic can now flow to/from this new port board.

1.9.6.2 Crossbar Board

The steps needed to replace a failed Crossbar board on a non-redundant Crossbar configuration and a redundant Crossbar configuration are different.

For a non-redundant Crossbar configuration, a failed Crossbar board causes all traffic in the entire system to be disabled. These are the following steps required after detection of a failed Crossbar board.

1. Detect failed Crossbar board. For example, CPU is indicated of loss of connectivity from this board to any one of the existing port boards.
2. Immediately disable TDM and non-TDM traffic for all ports in the Schedulers.
3. Disable the AIB links to/from the Port board.
4. Replace failed Crossbar board.
5. Bring up new board (see “Crossbar Board being added:” on page 90)
6. Enable TDM and best-effort traffic for all ports in the Scheduler.
7. Traffic can now flow.

The steps needed to cope with a failed Crossbar board on a redundant Crossbar configuration are described here. A failed Crossbar board does not require traffic to be disabled. The redundant Crossbar board carries traffic while the failed Crossbar board is replaced. These are the necessary steps to follow.

1. Detect failed Crossbar board. For example, CPU is indicated of loss of connectivity from this board to any one of the existing port boards.
2. Disable the AIB links to/from the Port board.
3. Replace failed Crossbar board.
4. Bring up new board (see “Crossbar Board being added:” on page 90)
5. Crossbars are now in fault tolerant mode.

1.9.6.3 Scheduler Board

In the non-redundant Scheduler configuration, a failed Scheduler board will prevent all traffic from flowing in the entire system. The following steps are required to cope with this failure.

1. Detect failed Scheduler board. For example, CPU is indicated of loss of connectivity from this board to any one of the existing port boards.
2. Disable TDM and non-TDM traffic to/from ports that still have connectivity.
3. Disable the AIB links to/from the Port board.
4. Replace failed Scheduler board.
5. Bring up new board (see “Scheduler Board being added:” on page 89)
6. Configure this Scheduler to operate in non-redundant Scheduler mode.
7. Enable TDM traffic in Scheduler for connected ports. TDM traffic can now flow.
8. Refresh state information from all ports to this Scheduler. (See Section 1.9.4 “Scheduler Refresh Procedure”)
9. Best-effort traffic can now flow.

The steps required for a redundant Scheduler configuration are more complicated. The redundant Scheduler can still schedule traffic while the failed Scheduler board is replaced. When the new Scheduler is inserted into the system, it must be resynchronized with the other Scheduler and the Enhanced Port Processors. This will be accomplished by the following steps.

1. Detect failed Scheduler board. For example, CPU is indicated of loss of connectivity from this board to any one of the existing port boards. This failure will cause this Scheduler to be disabled.
2. Disable the AIB links to/from the Port board.
3. Replace failed Scheduler board.
4. Bring up new board (see “Scheduler Board being added:” on page 89)
5. Configure this Scheduler to operate in redundant Scheduler mode.
6. If the Scheduler is generating the TDM sync, then TDM traffic must be stopped by writing a value of 0 to the TDM Sync Period and then writing the actual value to both Schedulers.
7. Enable TDM traffic in new Scheduler for appropriate ports.

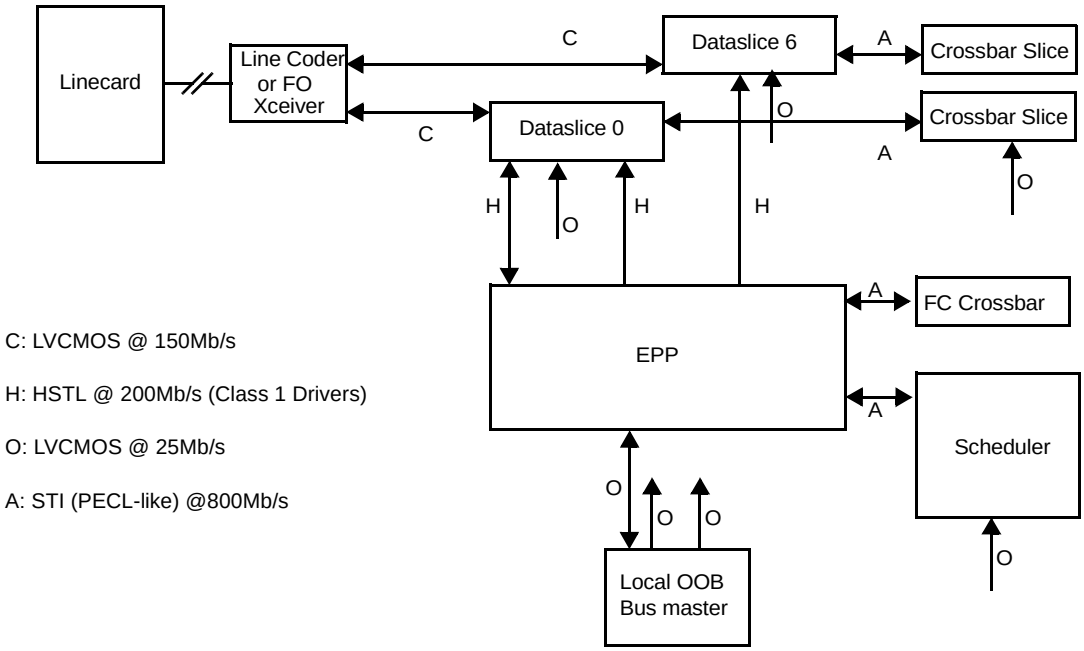
   NOTE: The redundant Scheduler is still scheduling TDM and non-TDM traffic.

8. Refresh state information from all ports to this Scheduler. (See Section 1.9.4 “Scheduler Refresh Procedure”) TDM traffic still flows.
9. Best-effort traffic can now flow. Schedulers are now in fault tolerant mode and synchronized.

1.10 ETT1 SIGNALS AND INTERCONNECTIONS

This section describes the different types of signals/interconnects used in a ETT1 switch core. Figure 44 shows the four types of devices required and lists the types of signals present.

Figure 44. ETT1 Signals and Interconnects



1.10.1 LVCMOS (C and O)

These are standard 2.5V LVCMOS I/Os (3.3V tolerant inputs). The ‘C’ signals provide the interface to the linecoder (if present) such as a Serdes device. These are intra-board signals. The ‘O’ signals are the local OOB bus, and operate using a local 25MHz clock. See Section 1.7 “The Out-of-Band (OOB) Bus Interface” on page 77 for more information.

1.10.2 HSTL (H)

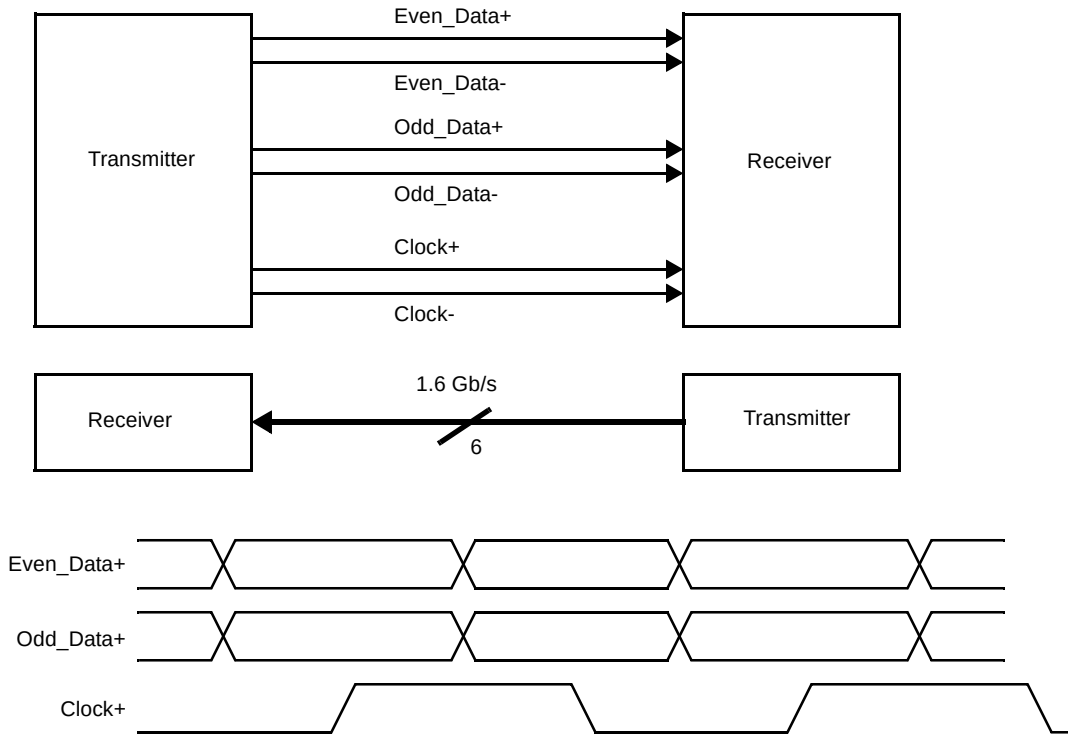
The HSTL signals provide the high-speed intra-board links between the EPP and the Dataslices. These are point-to-point signals. All signals have only one driver.

These point-to-point signals carry the data between the first two Dataslice devices and the EPP. These signals are HSTL Class 1 (50 Ohm load).

1.10.3 AIB Links (A)

The inter-board device-to-device links are all AIB serial links. These serial links are full duplex with 1.6Gbits/s data in both directions. Each unidirectional link comprises three pairs of differential signals: Even_Data, Odd_Data and Clock. The two data pairs carry link data while the clock provides source-synchronous clock information. The clock is a 400MHz signal. Data is transmitted using both edges of the clock to provide 800Mb/s on each of the two data pairs as shown in Figure 45.

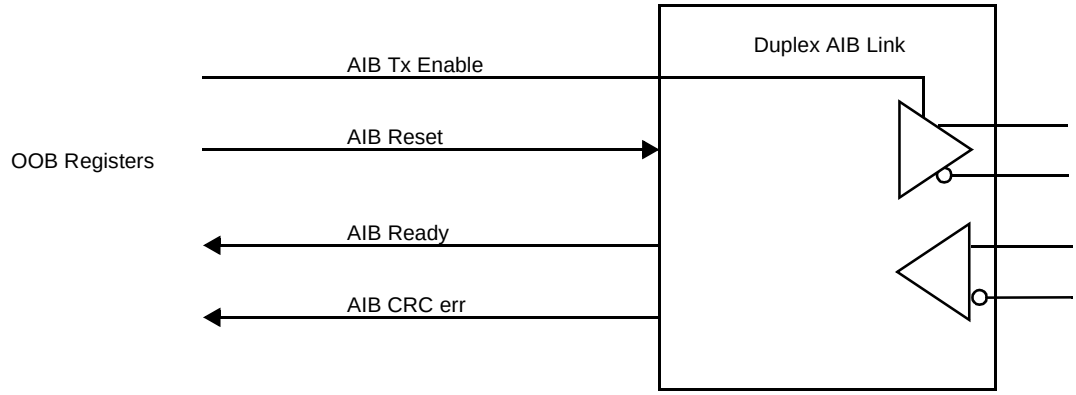
Figure 45. AIB Links



The AIB links are very straightforward to use, and are the same in all four devices. After a link is first enabled it will go through a training sequence with its peer. This is to ensure that the link is operating correctly before data is transmitted. All data carried over AIB links are protected by a CRC field; the status of the link is always visible to the local CPU via registers that can be accessed by the OOB bus.

The local CPU must configure all appropriate links before the links can operate. Figure 46 shows the AIB control/status signals that are visible to the CPU.

Figure 46. AIB Control/Status Signals.



Assuming that the link is connected to an AIB peer, then the first step is to enable the transmitter (AIB_Tx_Enable) and reset the link (AIB_Reset). The driver is inactive until AIB_Tx_Enable is set to 1, as indicated in the Figure 46. The link will start to train as soon as reset is deasserted. Assuming that the link trains correctly then the AIB_Ready signal for this link will be asserted. If the AIB_Ready signal is asserted then the bidirectional link is operational, so both receivers have trained correctly.

The link will now be operating in normal mode, transmitting data frames. If a transient error corrupts a frame, causing a CRC mismatch, then the AIB_CRC_err signal will be asserted. If four consecutive frames encounter CRC errors then the receiver will consider the link to have failed. In this case the training sequence will be restarted and AIB_Ready signal will be deasserted.

Of course a link might fail, restart and retrain faster than software can respond to an interrupt. In that case the ETT1 devices have separate OOB registers that indicate whether the AIB_Ready signal has transitioned. Thus, software can determine which link is having a problem. A failing link will also cause the AIB_CRC_err bit to be set.

1.10.4 Live Insertion

The AIB drivers should remain disabled until the attached receiver has reached its intended operating voltage. So do not enable an AIB driver unless the peer AIB receiver is inserted in the system and operating (i.e. voltage levels are steady). At reset all AIB drivers will be disabled (tri-stated). If the CPU detects that a board is being withdrawn from the system then it should immediately assert AIB reset for the link that is not being removed and deassert AIB_Tx_Enable for the AIB drivers at both ends of the link.

1.11 SYSTEM LATENCIES

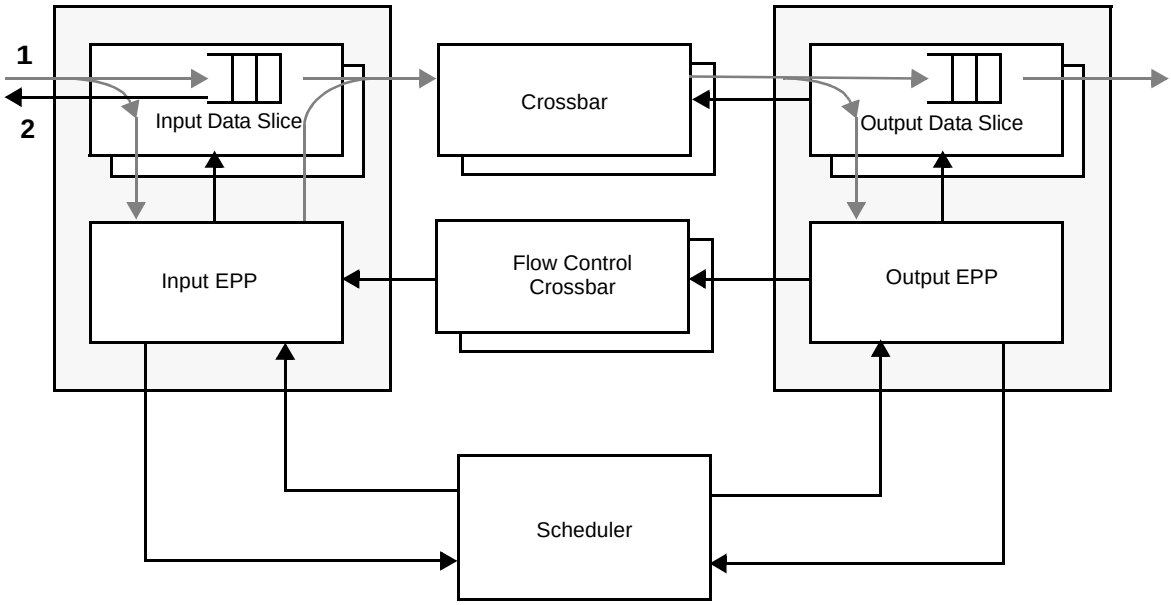
The ETT1 Chip Set operates in a cell-synchronous mode. Therefore all event latencies described here are defined in terms of ETT1 cell times (40ns). Further, all events are defined relative to the Dataslice to linecard interface (Serdes interface).

1.11.1 LCS Request to Grant Minimum Latency

This is the minimum latency from when the EPP receives a request from the linecard (**1** in Figure 47.) to when the EPP sends an LCS grant (**2**). These are relative to the Dataslice interface as shown, not the EPP.

This minimum latency is 16 cell times.

Figure 47. Request to Grant latency

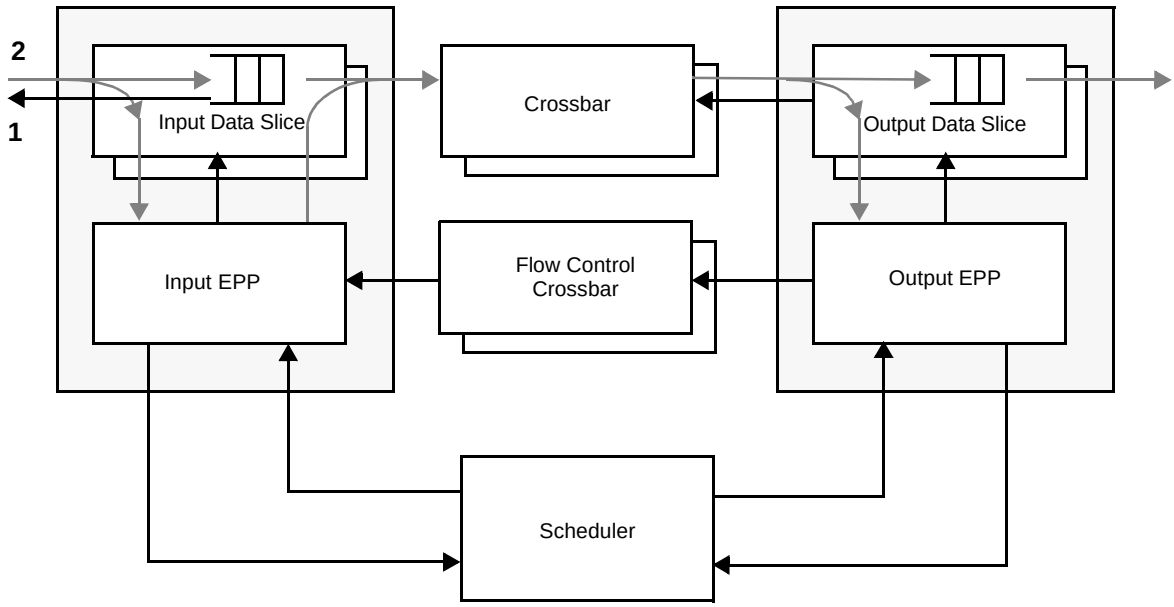


1.11.2 LCS Grant to Linecard Cell Latency

This is the maximum latency from when the iEPP issues an LCS grant (**1** in Figure 48.) to when the iEPP expects to receive the cell body associated with the grant (**2**). These are relative to the Dataslice interface as shown, not the EPP.

The maximum latency is 42 cell times minus the programmed value of the “DSiFIFO Token Mechanism Delay” field of the EPP EIDMA register (0009Ch). This is explained in more detail in C.2.2.1 , “Synchronization,” on page 333.

Figure 48. ETT1 Event Latencies

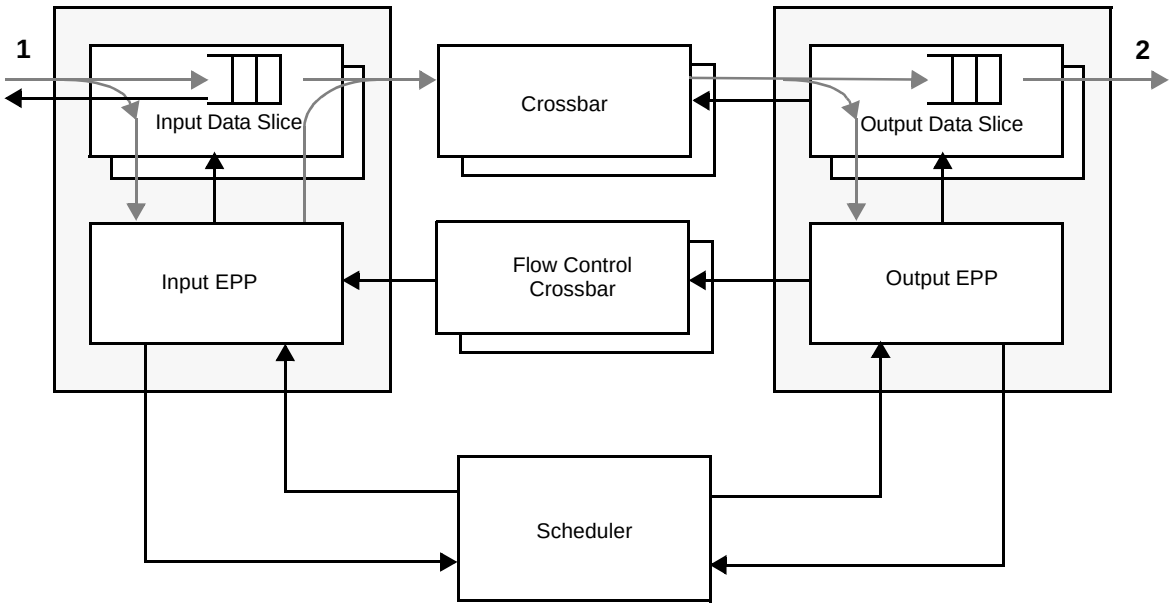


1.11.3 Cell latency Through an Empty System

This is the minimum cell latency that a cell body can experience as it passes from the input Dataslice (**1** in Figure 49.) to the output Dataslice (**2**).

The minimum cell latency is 29 cell times.

Figure 49. Cell Latency



1.11.4 LCS Hole Request to Hole Grant Latency

This is the maximum latency from when a hole request arrives at the Dataslice (**1** in Figure 47.) to when the hole request is granted (**2**). The egress cell does *not* contain a valid cell at the requested priority.

This maximum latency for ETT1 is 64 cell times.

1.12 ETT1 DIAGNOSTICS

An ETT1-based switch is a highly sophisticated system which is typically required to demonstrate very high levels of uptime. In order to achieve this uptime it is important to be able to:

- Ensure that a subsystem is operating correctly before it is brought online.
- Identify subsystems that have failed as quickly as possible.
- Diagnose failures to determine the specific entity that should be replaced.

This document describes a number of diagnostics that customers might find useful in addressing these needs.

The following terminology is used to describe a subsystem within a switch:

offline: the subsystem is not interacting with the rest of the system except via the OOB bus. AIB links may be taken up and down but will not cause changes to other subsystems.

online, inactive:the subsystem may be receiving signals from the system, and may be sending cells to itself, thus contending for fabric bandwidth. It only sends/receives diagnostic cells, not customer traffic.

online, active:the subsystem is participating in the normal operation of the switch, sending and receiving customer traffic and maintenance/diagnostic cells.

1.12.1 Device Tests

All of the ETT1 devices have a number of registers. The local ETT1 CPU can read from and write to many of these registers. Some care must be taken when modifying the contents of registers as they may have side effects that will affect the operation or cell flow of system components that are already online and active. In general, configuration registers should not be modified once a subsystem is online and active. The CPU can read most registers at any time without effect, however some registers, particularly interrupt registers and statistics counters, are cleared when they are read.

Some of the ETT1 devices contain memory structures (RAMs). Many of these RAMs can be written to and read from by the local CPU. Since the internal BIST mechanism is not available to the CPU, it is recommended that a customer verify the correct operation of such memory structures by a soak test in which the CPU writes and reads to all locations. Such a test should perform the widely known test patterns such as walking 1's, 0's, checkerboard, etc. Of course these tests are destructive to the current contents and can only be performed on RAMs that are not being used. In practice very few of the RAMs can be modified once the devices are active. The RAMs can be read non-destructively, and a great deal of internal system information is available this way; however unless the exact state of the system is known at the instance the RAM is read, then it is difficult to know if the returned value is correct. Obviously, RAMs that implement TDM and multicast tables are designed to be modified with due care during normal operation.

1.12.2 AIB Link Tests

All of the ETT1 devices use the high-speed AIB links for inter-device communication. The AIBs are used exclusively for inter-device links between PCBs. The AIB links associated with a subsystem can be tested while the subsystem is offline.

The following actions describe the procedure that should be used to test an AIB link attached to an offline subsystem (one end of the link may be attached to an online subsystem).

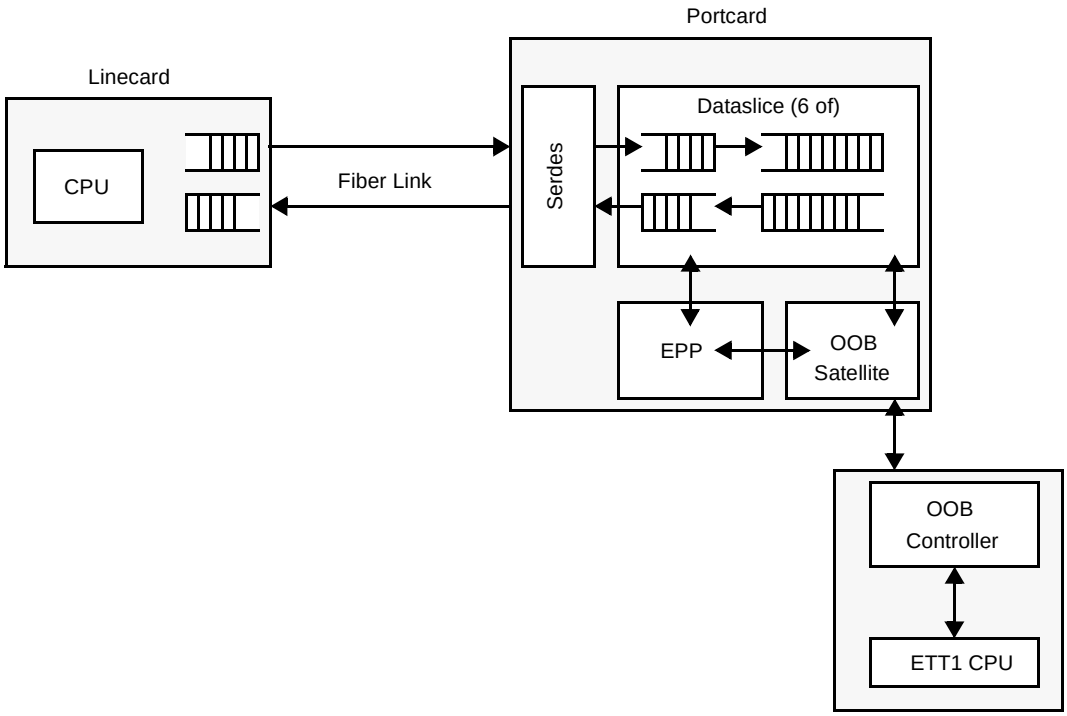
1. If the other end of the AIB is attached to an online (active) subsystem then ensure that the AIB is ignored by that subsystem, as follows:
   - Dataslice: this (test) AIB should be the secondary link.
   - EPP: this (test) AIB should be the secondary link.
   - Scheduler: disable the port associated with this (test) AIB. (Use the Enable Port register).
   - Xbar: no action needed.
2. Assert the reset control to both ends of the test AIB.
3. Enable the transmitters at both ends of the AIB.
4. De-assert the reset control to both ends of the test AIB.
5. The link should now come up -- the Ready bits associated with the link should be ‘1’. Interrupts will also be generated if the mask bits are set appropriately.
6. Disable the transmitter at one end of the AIB.
7. The far end of the AIB should indicate CRC errors and that the link is not ready. (A few CRC errors are generated before the link is considered down.)
8. Repeat steps 2 - 6 as needed to ensure that the link is OK. Also, leave the link up for a soak test and ensure that no CRC errors are received.

1.12.3 Linecard to ETT1 Link Diagnostics

When a new portcard in installed in the ETT1 Fabric it will be necessary to ensure that the portcard is operating correctly before bring it online. This section describes a number of tests that can be performed to verify the linecard-to-ETT1 side of the card.

Figure 50 shows an example architecture of the portcard and its connection to the linecard on the left and the ETT1 CPU on the right. The Dataslice is shown with two sets of queues.

Figure 50. Illustrating the Flow of Cells from the Linecard CPU to the ETT1 CPU

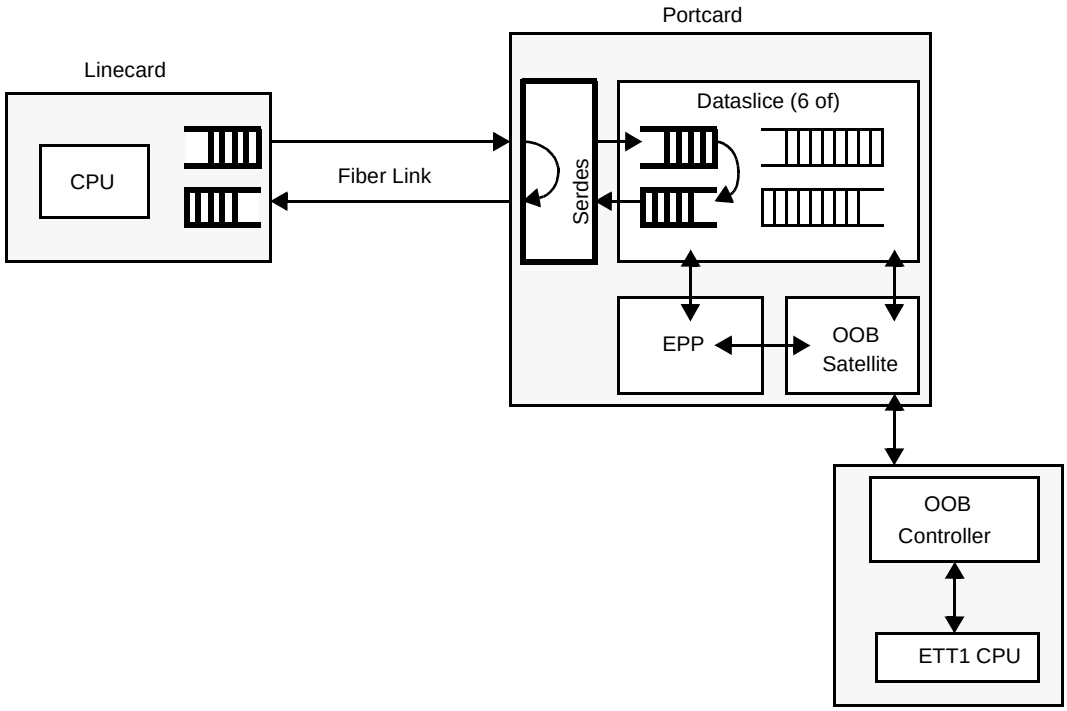


The shallow queues on the left are simply rate-matching FIFOs of 8 cells depth, used to compensate for asynchrony between the ETT1 core 200MHz clock and the 150MHz Serdes clock. The deeper queues on the right represent the virtual output queues and virtual input queues, where cells are stored pending being forwarded to their egress port or on to the linecard. We assume the presence of a linecard CPU which can generate and receive Control cells to/from the ETT1 portcard

1.12.3.1 Verifying the Fiber Link

In order to verify that the optical fiber link between linecard and portcard is operational, the portcard can set its local Serdes devices to operate in remote loopback mode. In addition the Dataslice devices can operate in a remote loopback mode in which cells received at the ingress are looped back directly to the egress rate matching FIFOs. Using either point of loopback the linecard can send cells to the portcard and have them looped back directly so that the linecard can check the cells itself. This can provide a useful post-failure diagnostic to determine if a fault is in the link or within the portcard or linecard. Figure 51 shows the loopback path.

Figure 51. Loopback Path



The loopback function is controlled from the portcard, and so the linecard must have some form of out-of-band communication in order to set either the Serdes or Dataslice into loopback mode.

NOTE: While the Dataslice is in loopback, the ingress cells are still received and passed to the EPP. The EPP should be disabled to prevent it from attempting to forward the incoming cells while the portcard is in loopback mode. Also, the EPP must be programmed to send only idle cells. (Set the EIDLCT register to 0x0.)

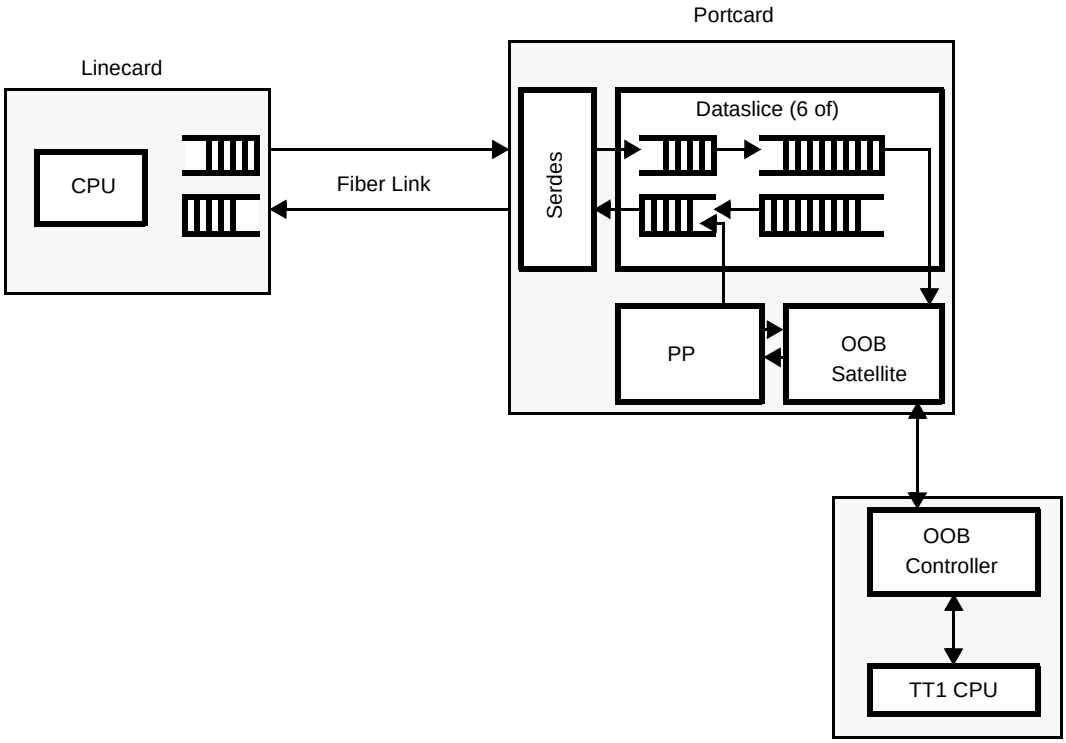
NOTE: If a Mux device is used then the mux bits in the LCS header should be set correctly so as to be returned to the correct OC-48c port. These mux bits will not be modified when the portcard is in loopback mode, and thus must be set correctly for both ingress and egress directions.

Some Serdes parts can be set to loopback at their serial outputs - so that the egress cells can be looped back to the ingress direction of the portcard. A portcard could use this feature to test the Serdes logic without needing a linecard to be attached.

1.12.3.2 Establishing Linecard to ETT1 Communication.

The LCS protocol provides support for Control packets which can be exchanged between the linecard CPU and the ETT1 CPU. Figure 52 shows the flow of cells between CPUs.

Figure 52. Control Packet Exchange Between Linecard CPU and ETT1 CPU



The linecard can send a Control packet to the portcard (once the fiber link is operational). This Control packet is stored in a special input queue that can store up to eight of these packets. Once the packet is stored the EPP can issue an interrupt to the ETT1 CPU indicating the arrival of a Control packet. The ETT1 CPU can then read the Control packet directly from the Dataslice memory.

NOTE: The ingress Control packet queue is eight cells deep and there is no flow control imposed on the Control packets sent from the linecard - if the linecard sends control packets faster than the ETT1 CPU can read them then Control packets will be lost. Some form of reliable flow control should be handled between the two CPUs.

In the egress direction, the ETT1 CPU can create a Control packet within the Dataslice memory. Once created the CPU issues a write command to the EPP and the cell will be transmitted to the linecard.

This mechanism allows the ETT1 CPU to communicate directly with the linecard. This also tests the fiber link, although at a limited cell rate.

1.12.3.3 Testing Error Detection Logic

The ETT1 CPU can send any arbitrary cell to the linecard CPU. The ETT1 CPU simply writes the entire cell into the Dataslice memory, then instructs the EPP to send it by writing to the EPP's "Output UC Queue

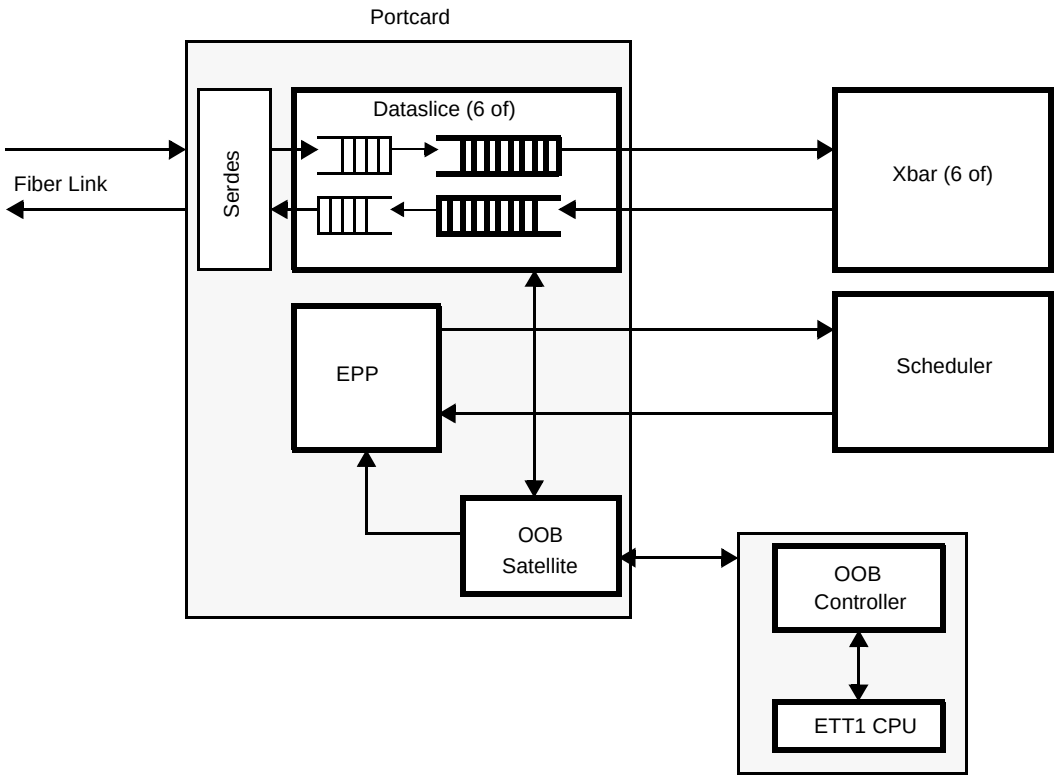
Information Memory". Therefore, the ETT1 CPU can create a cell that has an incorrect CRC value. This can be used to test the CRC detection logic at the linecard.

Also, the Dataslice uses internal tables to encode and decode the symbols sent to/received from the Serdes. Thus it is possible to configure these tables so that they will produce erroneous codes. This can be used to check that the 8b/10b decode is operating correctly.

1.12.4 ETT1 Internal Datapath Tests

The previous section describes tests that can verify the integrity of the link between the linecard and the ETT1 fabric. This section describes a test that can verify the internal datapaths. Figure 53 shows the datapaths that will be tested.

Figure 53. Testing the ETT1 Internal Datapaths



The objective is to create a packet in the Dataslice and have it traverse the usual internal ETT1 sequence of Scheduler-request, Scheduler-grant, forward-cell.

The steps are as follows:

1. The CPU writes to the Dataslices, creating a cell in one of the virtual output queues corresponding to itself -- i.e. it is going to send a cell to itself.

2. Turn off the EPP Output Scheduler for cells coming *from* itself. This will ensure that the cell will stay in the virtual input queue in the egress buffers of the Dataslice.
3. Bring up all AIB links and enable non-tdm traffic in the Scheduler for this port. Enable the EPP and take it out of LCS-2 Stop mode.
4. Write to the Outstanding Scheduler Request Counter in the EPP for this VOQ.

   This will cause a request to be made to the Scheduler which will immediately issue a grant to the EPP. The cell will be forwarded through the crossbar and back to the test port. The cell will then be stored in the egress queue but will not be forwarded to the linecard because the Output Scheduler was disabled in step 2.
5. The CPU can read the cell that is waiting in the egress queue and check that it passed through OK.
6. Repeat as needed.

A number of variations can be done on this basic theme:

- Because the Output Scheduler in each EPP can be stopped on a per *source* port basis, then in step 1 the CPU can create a cell going to *any* egress port, not just to itself. This is possible even if the egress port is currently online with real traffic. Thus it is possible to verify the datapath from the new port to all other ports without interfering with existing flows.
- Do not turn off the Output Scheduler in step 2 (or unfreeze it after Step 6). The linecard can then verify that the cell is OK, thus testing more of the system.
- Send enough cells so that the egress queue fills up and backpressures the input. So after sending 64 cells (OC-192 configuration) the egress queue should be full and thus prevent any more cells being sent to this queue.

2 Dataslice

This chapter contains information on the Dataslice device, part number PM9313-HC, available from PMC-Sierra, Inc.

The Dataslice contains queue storage for all cells that have arrived at the input port from the linecard and all cells that have passed through the Crossbar to the output port before departing to the linecard. The Dataslice interfaces to the linecards with 8B/10B encoded physical links, operating at 1.5 Gbit/s. Each Dataslice device packages two independent logical Dataslice units. The ingress and egress datapath is striped between Dataslices such that the first six bytes of the LCS cell are sent through logical Dataslice 0; the next six bytes through logical Dataslice 1, etc. Six Dataslice devices are required on each port card in order to process 72-byte LCS cells. An alternative configuration uses seven Dataslice devices to process 84-byte LCS cells, carrying a 76-byte payload.

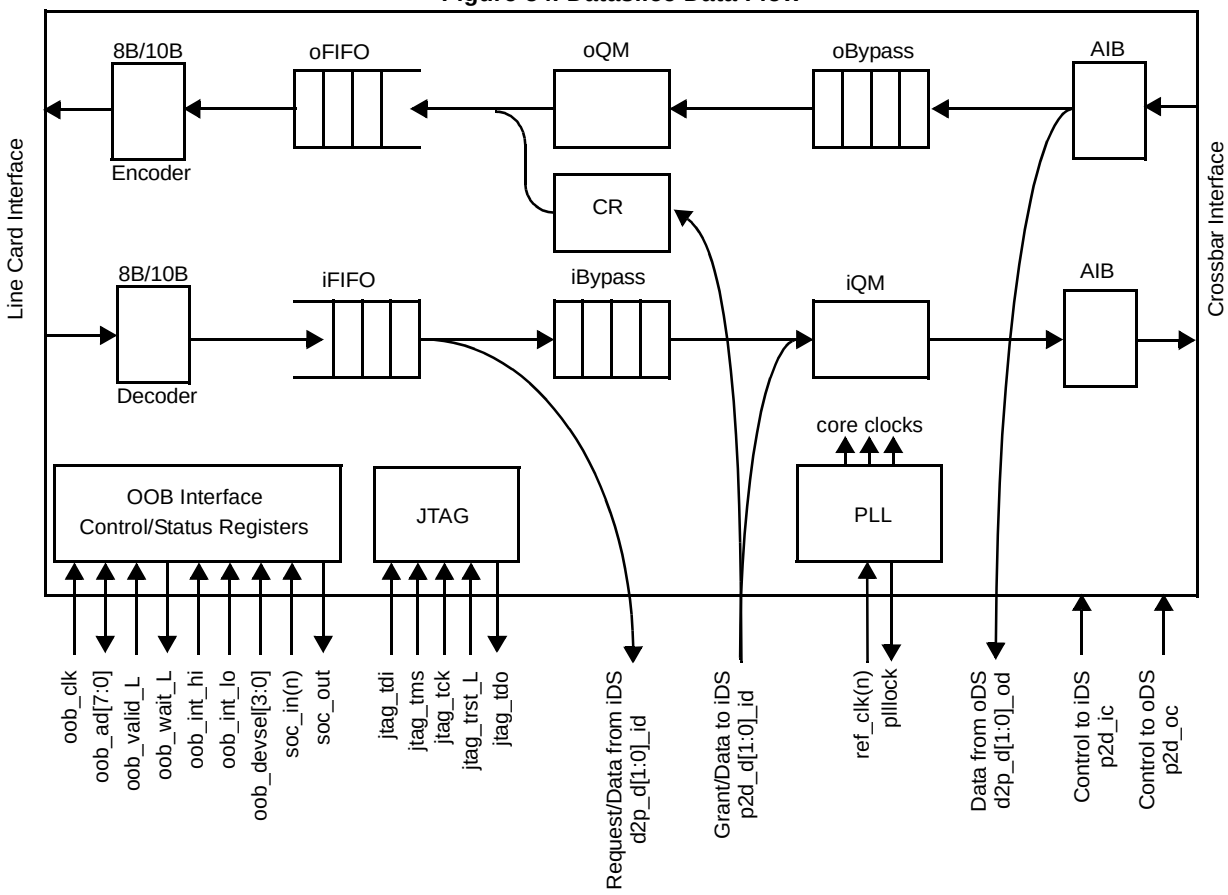
2.1 DATASLICE BLOCKS

The following discussion assumes 12 logical Dataslices per port. (6 devices).

There are 12 logical Dataslice units, packaged in six ICs, on a 72-byte LCS cell port card. Each logical Dataslice unit has a full-duplex, 10-bit wide, parallel bus connection to a Serdes device on the linecard interface, two bidirectional AIB connections to two Crossbars, two 8-bit busses for data from the ingress and egress paths to the Enhanced Port Processor, one bus for processed ingress data from the EPP, and two unidirectional 8-bit busses for ingress and egress control information from the EPP.

Figure 54 shows how cells flow through the Dataslice.

Figure 54. Dataslice Data Flow



The iFIFO is an eight-cell-deep and the oFIFO is a four-cell-deep FIFO that perform clock decoupling functions between the port card and the linecard and provide buffering between the queue memory and the outside world.

The iBypass and oBypass registers are temporary storage areas; eight-cell-deep FIFOs that provide the cell bypass function of the Dataslice. When a cell arrives from the Linecard, nine of the 12 Dataslices store the cell temporarily in their iBypass register. The three remaining Dataslices send their portion of the cell to the EPP. The EPP instructs the Dataslices to transfer the cell from the iBypass register to a specific input queue. The same general flow occurs for the oBypass, except that all 12 Dataslices use the oBypass. There is a constant, programmable depth for each of the bypass registers. This depth is set during initialization through the Out-Of-Band (OOB) bus by the CPU.

The iQM and oQM are the on-device queue memories used for storing cells in the input queues and output queues, respectively. The iQM and oQM queue memories are each 8,704 cells deep and 48 bits wide.

2.1.1 OOB Interface and Control/Status Registers

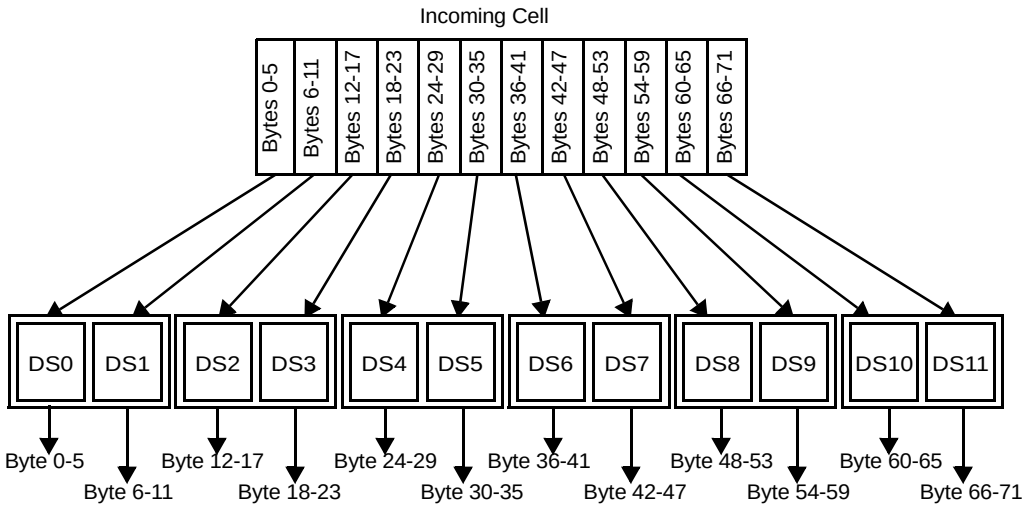
All of the devices have an OOB interface. This interface allows a single local CPU to control and monitor all of the devices within a core fabric. Internally, each device provides registers that can be mapped into the CPU's address space. These registers are described in more detail in Section 2.3 "Dataslice Registers".

2.1.2 Dataslice Cell Flow

2.1.2.1 Input Side Data Flow

On the input side, LCS cells are sent to the Dataslice from the Serdes interface and enter the iFIFO. The header and first four bytes of data enter DSs 0 and 1, the next six bytes of data enter DS2, and the last six bytes of the data enter DS11, as illustrated in Figure 55. Every cell time, one cell is removed from the iFIFO and sent to the iBypass. If the Dataslice is DS0, DS1, or DS2, the cell is copied and sent through the Request/Data from iDS, on d2p_d[1:0]_id, to the EPP. If the iFIFO is empty, the DSs write idle (and non valid) frames to the EPP and iBypass.

Figure 55. Slicing of a Data Cell into Logical Dataslices



NOTE: For more information on Dataslice synchronization, see Section 2.2.1 "Synchronization" on page 333.

The Control to iDS, p2d_ic, provides the write address and write request. If the write request is valid, the cell is written to the iQM location specified by the write address. For DS0, DS1, and DS2, the data from Grant/data to iDS, p2d_d[1:0]_id, is written to memory. For DS3 through DS11, the head cell from the iBypass is written to memory.

If the write request is not valid, nothing is written to memory. The cell being transferred out of the head of the iBypass registers is dropped.

Regardless of the Control to iDS information (on p2d_ic), the Credit Information (byte 0 through bit 5 of byte 4) of the cell header being written on Grant/data to iDS, is saved in a CR (credit register); it is used to indicate backpressure information back to the linecard. The CRC contained in the LCS header of the frame coming from the Grant/Data to iDS is computed only over the Tag portion of the LCS header. Another CRC in the frame that is computed only over the Credit portion of the LCS header is sent to DS0. This credit CRC is stored in the CR with the credits.

The Control to iDS frame contains a read request and read address. If the read request is asserted, the iQM location specified by the read address is read and the cell is sent through the Crossbar AIB Interface. The routing tag carried in the Control to iDS link is placed at the head of the cell as it is sent to the Crossbar.

If the “SendZero” bit is set, the DS sends an all-zero frame into the Crossbar instead of reading the cell from the queue memory.

2.1.2.2 Output Side Data Flow

Data cells entering the oDS devices from the Crossbar AIB Interface are placed into the oBypass. For DS0, DS1, and DS2, the cells are sent to the EPPs via the Data from oDS busses, d2p_d[1:0]_id. The VLD bit sent to the EPPs is the same as the VLD bit received from the Crossbars.

The EPP, having decided to store this new cell, sends a control frame over the Control to oDS bus, p2d_ic. This frame contains a write address and write request. If the write request is asserted, the cell at the head of the oBypass is placed into the oQM location designated by the write address. If the write request is not asserted, the cell at the head of the oBypass is removed from the FIFO and discarded.

The Control to oDS frame also contains a read request and read address. If the read request is asserted, then the cell in the oQM location specified by the read address is removed from the memory and placed in the oFIFO.

If the “SendZero” bit is set, the DS inserts an all-zero frame into the oFIFO instead of reading a cell from queue memory.

As the cell is inserted into the oFIFO, bytes 0 through the first part of byte 4 (the Credit Information fields of the LCS header) are overwritten by the contents of the CR (Credit Register) by DS0, and the LCS CRC is computed as the XOR of the tag CRC contained in the cell and the Credit CRC contained in the CR. If the Serdes clock rate is slightly less than the core clock rate, this oFIFO occasionally fills up. The EPP can be programmed to insert one “idle” cell periodically into the stream entering the oFIFO to avoid filling the oFIFO. The EPP does not assert either a read request or a “send zero” command to the oDS for these “idle” cells, nor does the oDS insert any cell into the oFIFO. The frequency of these idle cells is programmable and determined by the relative clock frequencies of the switch core and the linecard.

2.2 8B/10B INTERFACE

The Dataslice connects to the outside world via a 10-bit wide transmit and receive interface. In the ingress direction, an 8B/10B decoder lookup table maps received 10-bit characters into LCS data bytes. In the egress direction, an 8B/10B encoder lookup table maps LCS data bytes into 10-bit characters to be transmitted. The 8B/10B encoder and decoder lookup tables can be programmed to use any arbitrary 8B/10B code.

2.3 DATASLICE REGISTERS

The Dataslice device select low-order bits are hard wired on the board by three pins (oobdev_sel[2:0]) on the Dataslice package. Since there are two logical Dataslices within each package, the fourth implied bit is used to select which of the two logical Dataslices is addressed.

2.3.1 Dataslice Summary

The following table is a summary of information for all registers in the Dataslice. See the following Descriptions section for more information on individual registers.

Read and Clear means that reading the register causes it to be cleared (reset to zero)

Table 21. Dataslice Register Summary

Address	Symbol	Register	Access	Default Value
00000h	DSTS	Status	Read Only	10000000h
00004h	DRS	Reset	Read/Write	00000004h
00008h	DIRLMSK	Low Priority Mask	Read/Write	00000000h
0000Ch	DIRHMSK	High Priority Mask	Read/Write	00000000h
00010h	DIR	Interrupt Register	Read and Clear	00000000h
00030h	DAIBRS	AIB Reset	Read/Write	00000003h
00034h	DAIBRDY	AIB Ready	Read Only	00000003h
0003Ch	DAIBEN	AIB Tx Enable	Read/Write	00000000h
00080h	DPRIXB	Primary Crossbar Select	Read/Write	00000000h
00084h	DINBY	Input Bypass Shift Register Depth	Read/Write	00000002h
00088h	DOUTBY	Output Bypass Shift Register Depth	Read/Write	00000002h
0008Ch	DCODEN	8B/10B Encoder Enable	Read/Write	00000000h
00090h	DDECEN	8B/10B Decoder Enable	Read/Write	00000000h
00094h	DLSRRS	VCSEL Laser Reset (Active-Low)	Read/Write	00000000h
00098h	DLSRACT	VCSEL Laser Active Status	Read Only	00000000h
0009Ch	DPINRS	PIN Diode Reset (Active-Low)	Read/Write	00000000h
000A0h	DPINDET	PIN Diode Signal Detect	Read Only	00000000h

Table 21. Dataslice Register Summary (Continued)

Address	Symbol	Register	Access	Default Value
000A4h	DXCVDET	Transceiver Signal Detect	Read Only	00000000h
000A8h	DXCVLB	Transceiver Loopback Mode	Read/Write	00000000h
000ACh	DFORDY	Fiber Optics Link Ready	Read Only	00000000h
000B0h	DINTLB	Dataslice Loopback Mode	Read/Write	00000000h
00100h	DPLL	PLL Control/Status	Read/Write	0001447Ch
40000-5FFFCh	DIQM	Input Queue Memory (IQM)	Read/Write	Unknown
60000- 7FFFCh	DOQM	Output Queue Memory (OQM)	Read/Write	Unknown
80000-807FCh	D8BCOD	8B/10B Encoder Lookup Table	Read/Write	Unknown
81000-81FFCh	D8BDEC	8B/10B Decoder Lookup Table	Read/Write	Unknown

2.3.2 Dataslice Register Descriptions

2.3.2.1 Status

Symbol: DSTS

Address Offset: 00000h
Default Value: 10000000h
Access: Read Only

Status register.

Bits	Description
31:28	**Device ID Number.** Identifies the specific device.
27:24	**Device Revision Number.**
23:2	Reserved.
1	**High Priority Interrupt.** 1 = an outstanding high priority interrupt. One of the bits in the Interrupt Register is set, and is enabled via its corresponding high priority mask.
0	**Low Priority Interrupt.** 1 = an outstanding low priority interrupt. One of the bits in the Interrupt Register is set, and is enabled via its corresponding low priority mask.

2.3.2.2 Reset

Symbol: DRS

Address Offset: 00004h
Default Value: 00000000h
Access: Read/Write

Reset register.

Bits	Description
31-1	Reserved.
0	**Reset**. Writing a 1 to this location will reset the entire device. It is equivalent to a hardware reset. This register is cleared automatically when the device is reset. Writing a 0 is not necessary. Soft reset takes 1mS to complete.

2.3.2.3 Low Priority Mask

Symbol: DIRLMSK

Address Offset: 00008h
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:24	Reserved.
23:0	**Low Priority Mask**. Mask bits for low priority interrupts. Each mask bit is set to 1 to enable a low priority interrupt when the corresponding bit in the Interrupt Register is 1.

2.3.2.4 High Priority Mask

Symbol: DIRHMSK

Address Offset: 0000Ch
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:24	Reserved.
23:0	**High Priority Mask**. Mask bits for high priority interrupts. Each mask bit is set to 1 to enable a high priority interrupt when the corresponding bit in the General Interrupt Register is 1.

2.3.2.5 Interrupt Register

Symbol: DIR

Address Offset: 00010h
Default Value: 00000000h
Access: Read and Clear

Interrupt Register.

Bits	Description
31:24	Reserved.
23	**8B/10B Decoder Error.** This is set if a 10B character received from the fiber optics interface is not a valid 8B/10B code as defined in the 8B/10B decoder lookup table.
22	**Output Fifo Overflow.** This is set if the EPP attempts to push a cell into the output fifo when it is full. This output fifo can hold up to 4 cells. This is likely to be caused by incorrect programming of the “1-in-N” counter on the EPP.
21	**Input Fifo Overflow.** This is set if the fiber optics receiver attempts to push a cell into the input fifo when it is full. The input fifo can hold up to 8 cells. This is likely to be caused by a malfunction in the “1-in-N” counter on the ingress linecard.
20	**VCSEL Laser Active Up**. This is set if the VCSEL laser array raises its normal operation status.
19	**VCSEL Laser Active Down.** This is set if the VCSEL laser array drops its normal operation status.
18	**PIN Diode Signal Detect Up.** This is set if the PIN diode receiver raises the detection of a light stream from the fiber optical link.
17	**PIN Diode Signal Detect Down.** This is set if the PIN diode receiver drops the detection of a light stream from the fiber optical link.
16	**Transceiver Signal Detect Up.** This is set if the transceiver raises the detection of a bit stream from the fiber optics PIN diode receiver.
15	**Transceiver Signal Detect Down.** This is set if the transceiver drops the detection of a bit stream from the fiber optics PIN diode receiver.
14	**Transceiver Comma Detect**. This is set if the transceiver indicates the detection of a comma character “0011111xxx”.
13	**8B/10B Decoder Comma Detect**. This is set if the fiber optics receiver looks up a 10B control character that maps to “0100000000” in the 8B/10B decoder lookup table.
12	**Fiber Optics Egress Ready Up.** This is set if the fiber optics receiver on the egress linecard raises its ready status as a consequence of synchronization being locked on the 8B/10B control sequences.
11	**Fiber Optics Egress Ready Down.** This is set if the fiber optics receiver on the egress linecard drops its ready status as a consequence of synchronization being lost on the 8B/10B control sequences.

Bits	Description (Continued)
10	**Fiber Optics Ingress Ready Up.** This is set if the fiber optics receiver raises its ready status as a consequence of synchronization being locked on the 8B/10B control sequences.
9	**Fiber Optics Ingress Ready Down.** This is set if the fiber optics receiver drops its ready status as a consequence of synchronization being lost on the 8B/10B control sequences.
8	**Crossbar 1 Ready Up.** This is set if the Crossbar 1 AIB link raises its ready status.
7	**Crossbar 0 Ready Up.** This is set if the Crossbar 0 AIB link raises its ready status.
6	**Crossbar 1 Ready Down.** This is set if the Crossbar 1 AIB link drops its ready status.
5	**Crossbar 0 Ready Down.** This is set if the Crossbar 0 AIB link drops its ready status.
4	**Crossbar 0 and Crossbar 1 Ready Down Together.** This is set if both Crossbar 0 and Crossbar 1 AIB links drop their ready status simultaneously.
3	**Crossbar 1 CRC Error.** This is set if a CRC error is flagged by the Crossbar 1 link receiver.
2	**Crossbar 0 CRC Error.** This is set if a CRC error is flagged by the Crossbar 0 link receiver.
1	**Crossbar 0 and Crossbar 1 CRC Errors Together.** This is set if CRC errors are simultaneously flagged by both Crossbar 0 and Crossbar 1 AIB link receivers.
0	**Crossbar 0 and Crossbar 1 Mismatch.** This is set if both Crossbar 0 and Crossbar 1 AIB links are ready, there is no CRC error, and the data received from Crossbar 0 does not match that received from Crossbar 1.

2.3.2.6 AIB Reset

Symbol: DAIBRS

Address Offset: 00030h
Default Value: 00000003h
Access: Read/Write

Used to assert reset to the corresponding Crossbar AIB link. Bit 0 is for the Crossbar 0 AIB link, and bit 1 is for the Crossbar 1 AIB link. If reset is asserted, the link will not operate or transition to ready. Reset is asserted to 1 on power-up reset or if the reset bit in the control register is asserted.

Bits	Description
31:2	Reserved.
1	**Crossbar 1 AIB Link Reset**
0	**Crossbar 0 AIB Link Reset**

2.3.2.7 AIB Ready

Symbol: DAIBRDY

Address Offset: 00034h
Default Value: 00000003h
Access: Read Only

Indicates if the corresponding Crossbar AIB link is ready.

Bits	Description
31:2	Reserved.
1	**Crossbar 1 AIB link**
0	**Crossbar 0 AIB link**

NOTE: Default value assumes that Primary and Secondary Crossbars are installed.

2.3.2.8 AIB Tx Enable

Symbol: DAIBEN

Address Offset: 0003Ch
Default Value: 00000000h
Access: Read/Write

Used to enable the transmitter of the corresponding Crossbar AIB link. If the corresponding Crossbar is not physically present in the system, the bit for the Crossbar should be set to zero to reduce electrical noise and to prevent electrical glitches when the Crossbar board is inserted or removed.

Bits	Description
31:2	
1	**Crossbar 1 AIB Link Enable Transmitter**
0	**Crossbar 0 AIB Link Enable Transmitter**

2.3.2.9 Primary Crossbar Select

Symbol: DPRIXB

Address Offset: 00080h
Default Value: 00000000h
Access: Read/Write

Selects Crossbar 0 or Crossbar 1 as the primary Crossbar for fault tolerance.

Bits	Description
31:1	Reserved.
0	**Primary Crossbar Select.** A value of 0 selects Crossbar 0. A value of 1 selects Crossbar 1.

2.3.2.10 Input Bypass Shift Register Depth

Symbol: DINBY

Address Offset: 00084h
Default Value: 00000002h
Access: Read/Write

Programs the depth of the Input Bypass Shift Register between 0 and 7 cells. This register defaults to a depth of 2 at power-up and reset time and should be changed to a value of 3 for use in a ETT1 system..

Bits	Description
31:3	Reserved.
2:0	**Depth Range.** Values from 0 to 7.

2.3.2.11 Output Bypass Shift Register Depth

Symbol: DOUTBY

Address Offset: 00088h
Default Value: 00000002h
Access: Read/Write

Programs the depth of the Output Bypass Shift Register between 0 and 7 cells. This register defaults to a depth of 2 at power-up and reset time and should be changed to a value of 3 for use in a ETT1 system.

Bits	Description
31:3	Reserved.
2:0	**Depth Range.** Values from 0 to 7.

2.3.2.12 8B/10B Encoder Enable

Symbol: DCODEN

Address Offset: 0008Ch
Default Value: 00000000h
Access: Read/Write

Enables the 8B/10B Encoder

Bits	Description
31:1	Reserved.
0	**8B/10B Encoder Enable.** This bit enables the 8B/10B encoder logic for the fiber optics transmitter. If the 8B/10B encoder is disabled, no cells are popped from the output fifo. It is generally a good idea to program the 8B/10B encoder lookup table before the 8B/10B encoder is enabled.

2.3.2.13 8B/10B Decoder Enable

Symbol: DDECEN

Address Offset: 00090h
Default Value: 00000000h
Access: Read/Write

Enables the 8B/10B Decoder.

Bits	Description
31:1	Reserved.
0	**8B/10B Decoder Enable.** This bit enables the 8B/10B decoder logic for the fiber optics receiver. If the 8B/10B decoder is disabled, no synchronization is locked onto and therefore the fiber optics ready status is not asserted, and no tokens/cells are pushed into the token/input fifos. It is generally a good idea to program the 8B/10B decoder lookup table before the 8B/10B decoder is enabled.

2.3.2.14 VCSEL Laser Reset (Active-Low)

Symbol: DLSRRS

Address Offset: 00094h
Default Value: 00000000h
Access: Read/Write

Drive the active-low reset line on the Siemens PAROLI VCSEL laser array.

NOTE: This is a general output pin and may be used for other purposes.

Bits	Description
31-1	Reserved.
0	**VCSEL Laser Reset (Active-Low).** This bit can be used to drive the active-low reset line on the VCSEL laser array. It defaults to a low level at power-up and reset time which causes the laser array to be switched off.

2.3.2.15 VCSEL Laser Active Status

Symbol: DLSRACT

Address Offset: 00098h
Default Value: 00000000h
Access: Read Only

Indicates the status of the Siemens PAROLI laser array.

NOTE: This is a general input pin and may be used for other purposes.

Bits	Description
31:1	Reserved.
0	**VCSEL Laser Active Status.** This bit indicates whether the VCSEL laser array is in normal operation (1) or there is a laser fault (0) or the reset line is asserted active-low (0).

2.3.2.16 PIN Diode Reset (Active-Low)

Symbol: DPINRS

Address Offset: 0009Ch
Default Value: 00000000h
Access: Read/Write

Drives the active-low reset line on the Siemens PAROLI PIN diode array.

NOTE: This is a general output pin and may be used for other purposes.

Bits	Description
31:1	Reserved.
0	**PIN Diode Reset (Active-Low).** This bit can be used to drive the active-low reset line on the PIN diode array. It defaults to a low level at power-up and reset time which causes the PIN diode array outputs to be driven low.

2.3.2.17 PIN Diode Signal Detect

Symbol: DPINDET

Address Offset: 000A0h
Default Value: 00000000h
Access: Read Only

Indicates whether the Siemens PAROLI PIN diode array detects a signal of sufficient AC power on the fiber optical link.

NOTE: This is a general input pin and may be used for other purposes.

Bits	Description
31:1	Reserved.
0	**PIN Diode Signal Detect.** This bit indicates whether the PIN diode array detects a signal of sufficient AC power on the fiber optical link.

2.3.2.18 Transceiver Signal Detect

Symbol: DXCVDET

Address Offset: 000A4h
Default Value: 00000000h
Access: Read Only

Indicates whether the transceiver detects a bit stream of sufficient AC power from the PIN diode receiver.

NOTE: This is a general input pin and may be used for other purposes.

Bits	Description
31:1	Reserved.
0	**Transceiver Signal Detect.**

2.3.2.19 Transceiver Loopback Mode

Symbol: DXCVLB

Address Offset: 000A8h
Default Value: 00000000h
Access: Read/Write

Enables a local loopback mode as provided by the transceiver.

NOTE: This is a general output pin and may be used for other purposes.

Bits	Description
31:1	Reserved.
0	**Transceiver Loopback Mode.** This bit can be used to enable a local loopback mode as provided by the transceiver.

2.3.2.20 Fiber Optics Link Ready

Symbol: DFORDY

Address Offset: 000ACh
Default Value: 00000000h
Access: Read Only

Indicates synchronization and idle-ready state of the fiber optics transmitter.

NOTE: This is a general input pin and may be used for other purposes.

Bits	Description
31:2	Reserved.
1	**Egress Link Ready.** This bit indicates that the linecard fiber optics transmitter transitioned to the idle-ready 8B/10B control sequence. The linecard should transition from the idle 8B/10B control sequence to the idle-ready 8B/10B control sequence when its fiber optics receiver is synchronized.
0	**Ingress Link Ready.** This bit indicates that the fiber optics receiver is locked to the synchronization transmitted by the linecard. It causes the fiber optics transmitter to transition from the idle 8B/10B control sequence to the idle-ready 8B/10B control sequence.

2.3.2.21 Dataslice Loopback Mode

Symbol: DINTLB

Address Offset: 000B0h
Default Value: 00000000h
Access: Read/Write

Enables a local loopback mode on the Dataslice.

Bits	Description
31:1	Reserved.
0	**Dataslice Loopback Mode.** This bit enables a local loopback mode on the Dataslice that causes data to be looped internally from the input fifo to the output fifo.

2.3.2.22 PLL Control/Status

Symbol: DPLL

Address Offset: 00100h
Default Value: 0001447Ch
Access: Read/Write

Controls operation of the internal PLL (Phase Locked Loop). After a power on reset, the PLL itself is held in reset. This is reflected in bit 16 of this register. The local CPU must reset this bit to 0 to enable operation of the device, and thus should write 0000447Ch to this register.

Bit	Description
31:17	**PLL status.** These bits reflect internal PLL operation status and should be ignored.
16	**Reset PLL.** When set to 1 the PLL is held reset. The supplied reference clock will be used as the internal clock. The serial links will not be operational. This bit will be 1 after power-up reset and should be deasserted for normal operation. PLL reset takes 10mS to complete.
15:0	**PLL control.**

2.3.2.23 Input Queue Memory (IQM)

Symbol: DIQM

Address Offset: 40000-5FFFCh
Default Value: Unknown
Access: Read/Write

The actual IQM address range extends from 40000h to 50FFCh. Beyond IQM address 50FFCh, aliasing occurs within the IQM 512x48 RAM bank.

If the IQM is accessed at an even 32-bit aligned word address, the data bitmap consists of the following:

Bits	Description
31:16	Reserved.
15:0	This field contains the 16 MSBs of the addressed IQM 48-bit word.

If the IQM is accessed at an odd 32-bit aligned word address, the data bitmap consists of the following:

Bits	Description
31:0	This field contains the 32 LSBs of the addressed IQM 48-bit word

2.3.2.24 Output Queue Memory (OQM)

Symbol: DOQM

Address Offset: 60000- 7FFFCh
Default Value: Unknown
Access: Read/Write

The actual OQM address range extends from 60000h to 70FFCh. Beyond IQM address 70FFCh, aliasing occurs within the OQM 512x48 RAM bank.

If the OQM is accessed at an even 32-bit aligned word address, the data bitmap consists of the following:

Bits	Description
31:16	Reserved.
15:0	This field contains the 16 MSBs of the addressed OQM 48-bit word.

If the IQM is accessed at an odd 32-bit aligned word address, the data bitmap consists of the following:

Bits	Description
31:0	This field contains the 32 LSBs of the addressed OQM 48-bit word

2.3.2.25 8B/10B Encoder Lookup Table

Symbol: D8BCOD

Address Offset: 80000-807FCh
Default Value: Unknown
Access: Read/Write

The actual 8B/10B encoder lookup table address range extends from 0x80000 to 0x804FC. Beyond 8B/10B encoder lookup table address 0x804FC aliasing will occur within the 8B/10B encoder lookup table 320x26 RAM.

The 8B/10B encoder lookup table address space is defined as:

- 0x80000-0x803FC: Lookup of 8B data bytes into 10B data characters.
- 0x80400-0x80414: Lookup of idle 8B/10B control sequence (suggested 10B control sequence: 0x80400 = K28.5 and 0x80404-0x80414 = K27.7).
- 0x80420-0x80434: Lookup of idle-ready 8B/10B control sequence (suggested 10B control sequence: 0x80420 = K28.5 and 0x80424-0x80434 = K29.7).

The 8B/10B encoder lookup table data bitmap consists of the following:

Bits	Description
31:26	Reserved.
25:23	Weight of the positive RDS 10B character.
22:13	Positive RDS 10B character.
12:10	Absolute weight of the negative RDS 10B character.
9:0	Negative RDS 10B character.

NOTE: Bit 0 and bit 13 represent the bit that would be transmitted first for each respective 10B character.

2.3.2.26 8B/10B Decoder Lookup Table

Symbol: D8BDEC

Address Offset: 81000-81FFCh
Default Value: Unknown
Access: Read/Write

The 8B/10B decoder lookup table data bitmap consists of the following:

Bits	Description
31:10	Reserved.
9	This bit flags an 8B/10B decode error. The entry that the 10B character indexes does not correspond to a valid 8B control or data byte.
8	This bit identifies an 8B/10B control character. If the 10B character indexes an entry that has this control bit set and the data field [7:0] is 0x0, the 10B character is recognized as the comma character for synchronization. If the 10B character indexes an entry that has this control bit set and the data field [7:0] is 0xFF, the 10B character is recognized as an indication that the linecard fiber optics is ready.
7:0	This data field contains the corresponding 8B value.

2.4 DATASLICE SIGNAL DESCRIPTIONS

This section describes the Dataslice signals.

The following notation is used to describe the signal type:

I Input pin

O Output pin

B Bidirectional Input/Output pin

The signal description also includes the type of buffer used for the particular signal:

PECL PECL are Pseudo-ECL (positive voltage ECL) compatible signals.

STI STI is a very high-speed asynchronous communications protocol used to connect devices that may be 1 to 15 meters apart.

HSTL All HSTL specifications are consistent with EIA/JEDEC Standard, EIA/JESD8-6 “High Speed Transceivers Logic (HSTL): A 1.5V Output Buffer Supply Voltage based Interface Standard for Digital Integrated Circuits”, dated 8/95. Refer to these specifications for more information**.**

CMOS The CMOS Buffers are either 3.3 V compatible or 2.5 V Low Voltage TTL compatible signals, as noted.

Table 22. Dataslice Signal Descriptions

Name	I/O	Type	Description
OOB Interface			
oob_ad[7:0]	B	CMOS	OOB Address bus
oob_clk	I	CMOS	OOB CLock
oob_devsel0	I	CMOS	OOB device select
oob_devsel1	I	CMOS	OOB device select
oob_devsel2	I	CMOS	OOB device select
oob_valid_L	I	CMOS	OOB Control
oob_wait_L	O	CMOS	OOB Control
DS0 OOB Interface			
oob0_int_hi	O	CMOS	OOB Interrupt

Table 22. Dataslice Signal Descriptions (Continued)

Name	I/O	Type	Description
oob0_int_lo	O	CMOS	OOB Interrupt
DS1 OOB Interface			
oob1_int_hi	O	CMOS	OOB Interrupt
oob1_int_lo	O	CMOS	OOB Interrupt
DS/EPP Interface			
p2d_ic[7:0]	I	HSTL (Class 2)	EPP to DS ingress control
p2d_oc[7:0]	I	HSTL (Class 2)	EPP to DS egress control
Vref	I	HSTL	Input Reference Voltage
DS0 DS/EPP Interface			
d2p_d0_id[7:0]	O	HSTL (Class 1)	DS to EPP ingress data unmodified
d2p_d0_od[7:0]	O	HSTL (Class 1)	DS to EPP egress data
p2d_d0_id[7:0]	I	HSTL (Class 1)	EPP to DS ingress data modified
DS1 DS/EPP Interface			
d2p_d1_id[7:0]	O	HSTL (Class 1)	DS to EPP ingress data unmodified
d2p_d1_od[7:0]	O	HSTL (Class 1)	DS to EPP egress data
p2d_d1_id[7:0]	I	HSTL (Class 1)	EPP to DS ingress data modified
DS0 AIB Interface			
d2x_d0a_[c, cn, e, en, o, on]	O	STI	DS to Xbar cell data
d2x_d0b_[c, cn, e, en, o, on]	O	STI	DS to Xbar cell data
x2d_d0a_[c, cn, e, en, o, on]	I	STI	Xbar to DS cell data
x2d_d0b_[c, cn, e, en, o, on]	I	STI	Xbar to DS cell data

Table 22. Dataslice Signal Descriptions (Continued)

Name	I/O	Type	Description
DS1 AIB Interface			
d2x_d1a_[c, cn, e, en, o, on]	O	STI	DS to Xbar cell data
d2x_d1b_[c, cn, e, en, o, on]	O	STI	DS to Xbar cell data
x2d_d1a_[c, cn, e, en, o, on]	I	STI	Xbar to DS cell data
x2d_d1b_[c, cn, e, en, o, on]	I	STI	Xbar to DS cell data
Serdes Interface			
fotx_clk_out	O	CMOS	Clock output sync'd to fotx_txd
DS0 Serdes Interface			
forx0_clk	I	CMOS	Serdes to DS 8b/10b receive clock
forx0_rxd[9:0]	I	CMOS	Serdes to DS 8b/10b encoded data
fotx0_txd[9:0]	O	CMOS	DS to Serdes 8b/10b encoded data
pin_diode0_reset_L	O	CMOS	VCSEL and Serdes control signals
pin_diode0_sig_det	I	CMOS	VCSEL and Serdes status signals
vcsel0_laser_act	I	CMOS	VCSEL and Serdes status signals
vcsel0_reset_L	O	CMOS	VCSEL and Serdes control signals
xcvr0_com_det	I	CMOS	VCSEL and Serdes status signals
xcvr0_loopback	O	CMOS	VCSEL and Serdes control signals
xcvr0_sig_det	I	CMOS	VCSEL and Serdes status signals
DS1 Serdes Interface			
forx1_clk	I	CMOS	Serdes to DS 8b/10b receive clock
forx1_rxd[9:0]	I	CMOS	Serdes to DS 8b/10b encoded data
fotx1_txd[9:0]	O	CMOS	DS to Serdes 8b/10b encoded data
pin_diode1_reset_L	O	CMOS	VCSEL and Serdes control signals
pin_diode1_sig_det	I	CMOS	VCSEL and Serdes status signals
vcsel1_laser_act	I	CMOS	VCSEL and Serdes status signals
vcsel1_reset_L	O	CMOS	VCSEL and Serdes control signals

Table 22. Dataslice Signal Descriptions (Continued)

Name	I/O	Type	Description
xcvr1_com_det	I	CMOS	VCSEL and Serdes status signals
xcvr1_loopback	O	CMOS	VCSEL and Serdes control signals
xcvr1_sig_det	I	CMOS	VCSEL and Serdes status signals
Power Supply, Clock Source, Reset, and Diagnostics			
fotx_clk_in	I	CMOS	Clock input used for fotx_clk_out
plllock	O	CMOS	Test output
pwrup_reset_L	I	CMOS	Synchronous, Active Low Reset
VDDA		Isolated Supply	VDD (2.5 V) for PLL
VDD		Supply	VDD (2.5 V)
VDDQ		Supply	VDDQ (1.6 V)
GND		Supply	GND (0 V)
ref_clk and ref_clkn	I	PECL	System 200MHz clock (differential)
soc_in and soc_inn	I	PECL	System Start-of-cell (differential)
JTAG Interface			
jtag_tck	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tdi	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tdo	O	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tms	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_trst_L	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
Dataslice Configuration			
crdcrcen0	I	CMOS	DS0 credit CRC enable tied low on all DS devices
crdcrcen1	I	CMOS	DS1 credit CRC enable tied high on physical DS device 0 only

Table 22. Dataslice Signal Descriptions (Continued)

Name	I/O	Type	Description
crden0	I	CMOS	DS0 credit enable tied high on physical DS device 0 only
crden1	I	CMOS	DS1 credit enable tied low on all DS devices
ibpen0	I	CMOS	DS0 iBypass enable tied low on physical DS device 0 only
ibpen1	I	CMOS	DS1 iBypass enable tied low on physical DS device 0 only
ASIC Manufacturing Test Interface (see Appendix B.2)			
ce0_io	I	CMOS	Test (GND)
ce0_scan	I	CMOS	Test (GND)
ce0_tstm3	I	CMOS	Test (GND)
lssd_ce1_a	I	CMOS	Test (VDD)
lssd_ce1_b	I	CMOS	Test (VDD)
lssd_ce1_c1	I	CMOS	Test (VDD)
lssd_ce1_c2	I	CMOS	Test (VDD)
lssd_ce1_c3	I	CMOS	Test (VDD)
lssd_scan_in[15:0]	I	CMOS	Test: Scan input (GND)
lssd_scan_out[15:0]	O	CMOS	Test: Scan output
mon[15:0]	O	CMOS	Test (NC)
plltest_in	I	CMOS	Test (GND)
plltest_out	O	CMOS	Test output (NC)
test_di1	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_di2	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_lt	I	CMOS	Test (VDD)
test_re	I	CMOS	Test (GND)
test_ri	I	CMOS	Test (VDD)

2.5 PINOUT AND PACKAGE INFORMATION

2.5.1 Pinout Tables

Table 23. Dataslice Pinout (left side)

	01	02	03	04	05	06	07	08	09	10
A	No Pin	GND	test_di2	VDDQ	test_di1	GND	test_ri	VDD	UNUSED	GND
B	d2p_d1_id0	crdcrcen1	ibpen1	oob_devsel2	oob_wait_L	pwrup_reset_L	UNUSED	oob_valid_L	UNUSED	UNUSED
C	ref_clkn	VDDQ	ref_clk	GND	test_re	VDD	test_lt	GND	UNUSED	VDDQ
D	d2p_d1_id4	crden1	oob1_int_lo	oob1_int_hi	UNUSED	oob_clk	p2d_ic7	p2d_ic2	UNUSED	UNUSED
E	VDDA	GND	plltest_in	VDDQ	plllock	GND	p2d_ic0	VDD	lssd_scan_out9	GND
F	d2p_d1_id5	d2p_d1_id2	d2p_d1_id1	p2d_d1_id2	UNUSED	soc_inn	p2d_ic5	p2d_ic4	UNUSED	UNUSED
G	UNUSED	VDD	plltest_out	GND	p2d_d1_id3	VDDQ	soc_in	GND	lssd_scan_out8	VDD
H	p2d_d1_id6	p2d_d1_id1	d2p_d1_id7	d2p_d1_id3	d2p_d1_od0	d2p_d1_id6	oob_devsel0	p2d_ic6	p2d_oc6	UNUSED
J	d2p_d1_od1	GND	p2d_d1_id4	VDD	d2p_d1_od3	GND	UNUSED	VDDQ	p2d_ic1	GND
K	p2d_d1_id0	d2p_d1_od5	d2p_d1_od2	fotx1_txd1	d2p_d1_od6	d2p_d1_od4	Vref	oob_devsel1	Vref	p2d_oc7
L	p2d_d1_id5	VDDQ	d2p_d1_od7	GND	fotx1_txd0	VDDQ	p2d_d1_id7	GND	p2d_ic3	VDD
M	UNUSED	UNUSED	UNUSED	UNUSED	lssd_scan_out6	UNUSED	lssd_scan_out7	Vref	GND	fotx0_txd7
N	UNUSED	GND	UNUSED	VDDQ	UNUSED	GND	UNUSED	VDD	fotx1_txd3	GND
P	UNUSED	UNUSED	UNUSED	UNUSED	lssd_scan_out5	UNUSED	lssd_scan_out4	fotx1_txd9	VDD	mon6
R	d2x_d1a_c	VDD	fotx1_txd8	GND	fotx1_txd2	VDD	d2x_d1a_o	GND	mon3	VDD
T	x2d_d1a_o	forx1_rxd1	forx1_rxd2	fotx1_txd5	fotx1_txd4	forx1_rxd0	d2x_d1a_en	d2x_d1b_c	mon2	mon5
U	forx1_rxd3	GND	forx1_rxd4	VDD	fotx1_txd6	GND	x2d_d1a_en	VDDQ	fotx_clk_in	GND
V	d2x_d1a_cn	x2d_d1a_on	forx1_rxd5	fotx1_txd7	d2x_d1a_on	x2d_d1a_e	d2x_d1b_on	x2d_d1b_e	fotx_clk_out	UNUSED
W	UNUSED	VDDQ	lssd_scan_out3	GND	x2d_d1a_c	VDDQ	d2x_d1b_cn	GND	lssd_scan_in9	VDD
Y	forx1_rxd6	forx1_rxd8	forx1_rxd9	d2x_d1a_e	x2d_d1a_cn	d2x_d1b_o	x2d_d1b_o	x2d_d1b_en	UNUSED	UNUSED
AA	UNUSED	GND	lssd_scan_out2	VDDQ	lssd_scan_in15	GND	x2d_d1b_on	VDD	lssd_scan_in8	GND
AB	forx1_rxd7	vcsel1_reset_L	pin_diode1_reset_L	xcvr1_sig_det	d2x_d1b_e	xcvr1_loopback	mon1	mon4	UNUSED	UNUSED
AC	lssd_scan_out0	VDD	lssd_scan_out1	GND	lssd_scan_in12	VDD	lssd_scan_in10	GND	UNUSED	VDDQ
AD	UNUSED	pin_diode1_sig_det	forx1_clk	xcvr1_com_det	x2d_d1b_c	mon0	d2x_d1b_en	x2d_d1b_cn	UNUSED	UNUSED
AE	vcsel1_laser_act	GND	lssd_scan_in14	VDDQ	lssd_scan_in13	GND	lssd_scan_in11	VDD	UNUSED	GND

Table 24. Dataslice Pinout (right side)

11	12	13	14	15	16	17	18	19	
UNUSED	VDD	lssd_scan_ out13	GND	lssd_scan_ out15	VDDQ	UNUSED	GND	oob0_int_hi	**A**
UNUSED	UNUSED	UNUSED	oob_ad7	oob_ad3	oob_ad0	ibpen0	UNUSED	d2p_d0_id7	**B**
UNUSED	GND	lssd_scan_ out12	VDD	lssd_scan_ out14	GND	jtag_tms	VDDQ	jtag_tdi	**C**
UNUSED	p2d_oc4	p2d_oc1	oob_ad4	UNUSED	UNUSED	crdcrcen0	oob0_int_lo	crden0	**D**
lssd_scan_ out10	VDDQ	oob_ad6	GND	jtag_tdo	VDDQ	jtag_tck	GND	UNUSED	**E**
UNUSED	p2d_oc3	oob_ad1	UNUSED	d2p_d0_id2	p2d_d0_id0	d2p_d0_id6	d2p_d0_id4	d2p_d0_id3	**F**
lssd_scan_ out11	GND	oob_ad2	VDDQ	p2d_d0_id6	GND	jtag_trst_L	VDD	UNUSED	**G**
p2d_oc5	oob_ad5	UNUSED	Vref	p2d_d0_id1	d2p_d0_id5	d2p_d0_id1	p2d_d0_id7	p2d_d0_id2	**H**
p2d_oc2	VDD	UNUSED	GND	d2p_d0_od2	VDD	p2d_d0_id4	GND	d2p_d0_od7	**J**
Vref	UNUSED	p2d_d0_id5	d2p_d0_od5	Vref	fotx0_txd9	d2p_d0_od6	d2p_d0_od3	d2p_d0_id0	**K**
p2d_oc0	GND	d2p_d0_od4	VDDQ	fotx0_txd8	GND	d2p_d0_od0	VDDQ	p2d_d0_id3	**L**
VDD	d2p_d0_od1	ce0_scan	UNUSED	ce0_io	UNUSED	UNUSED	UNUSED	UNUSED	**M**
fotx0_txd6	VDD	UNUSED	GND	UNUSED	VDDQ	UNUSED	GND	UNUSED	**N**
GND	fotx0_txd1	lssd_ce1_a	UNUSED	ce0_tstm3	UNUSED	UNUSED	UNUSED	UNUSED	**P**
mon12	GND	d2x_d0a_en	VDD	fotx0_txd5	GND	fotx0_txd2	VDD	d2x_d0a_cn	**R**
mon13	x2d_d0b_c	x2d_d0a_en	d2x_d0a_o	fotx0_txd4	fotx0_txd3	forx0_rxd8	forx0_rxd9	x2d_d0a_on	**T**
mon10	VDD	x2d_d0b_e	GND	d2x_d0a_on	VDD	forx0_rxd6	GND	forx0_rxd7	**U**
mon8	d2x_d0b_o	x2d_d0b_cn	x2d_d0a_e	d2x_d0a_e	fotx0_txd0	forx0_rxd5	x2d_d0a_o	d2x_d0a_c	**V**
lssd_scan_ in6	GND	d2x_d0b_e	VDDQ	x2d_d0a_c	GND	lssd_ce1_b	VDDQ	UNUSED	**W**
UNUSED	mon7	d2x_d0b_on	d2x_d0b_en	x2d_d0b_en	x2d_d0a_cn	forx0_rxd1	forx0_rxd2	forx0_rxd4	**Y**
lssd_scan_ in7	VDDQ	mon9	GND	lssd_scan_ in0	VDDQ	lssd_ce1_c1	GND	UNUSED	**AA**
UNUSED	mon11	mon14	xcvr0_ loopback	x2d_d0b_on	xcvr0_sig_ det	pin_diode0_ reset_L	vcsel0_ reset_L	forx0_rxd3	**AB**
UNUSED	GND	lssd_scan_ in5	VDD	lssd_scan_ in3	GND	lssd_ce1_c2	VDD	lssd_ce1_c3	**AC**
UNUSED	d2x_d0b_c	x2d_d0b_o	mon15	d2x_d0b_cn	xcvr0_com_ det	forx0_clk	pin_diode0_ sig_det	forx0_rxd0	**AD**
UNUSED	VDD	lssd_scan_ in4	GND	lssd_scan_ in2	VDDQ	lssd_scan_ in1	GND	vcsel0_ laser_act	**AE**

Table 18. Dataslice Alpha Pin List

Signal Name	Pin
ce0_io	M15
ce0_scan	M13
ce0_tstm3	P15
crdcrcen0	D17
crdcrcen1	B02
crden0	D19
crden1	D02
d2p_d0_id0	K19
d2p_d0_id1	H17
d2p_d0_id2	F15
d2p_d0_id3	F19
d2p_d0_id4	F18
d2p_d0_id5	H16
d2p_d0_id6	F17
d2p_d0_id7	B19
d2p_d0_od0	L17
d2p_d0_od1	M12
d2p_d0_od2	J15
d2p_d0_od3	K18
d2p_d0_od4	L13
d2p_d0_od5	K14
d2p_d0_od6	K17
d2p_d0_od7	J19
d2p_d1_id0	B01
d2p_d1_id1	F03
d2p_d1_id2	F02
d2p_d1_id3	H04
d2p_d1_id4	D01
d2p_d1_id5	F01
d2p_d1_id6	H06
d2p_d1_id7	H03
d2p_d1_od0	H05
d2p_d1_od1	J01
d2p_d1_od2	K03
d2p_d1_od3	J05
d2p_d1_od4	K06
d2p_d1_od5	K02
d2p_d1_od6	K05
d2p_d1_od7	L03
d2x_d0a_c	V19
d2x_d0a_cn	R19
d2x_d0a_e	V15
d2x_d0a_en	R13
d2x_d0a_o	T14
d2x_d0a_on	U15
d2x_d0b_c	AD12
d2x_d0b_cn	AD15
d2x_d0b_e	W13
d2x_d0b_en	Y14
d2x_d0b_o	V12
d2x_d0b_on	Y13
d2x_d1a_c	R01
d2x_d1a_cn	V01
d2x_d1a_e	Y04
d2x_d1a_en	T07
d2x_d1a_o	R07
d2x_d1a_on	V05
d2x_d1b_c	T08
d2x_d1b_cn	W07
d2x_d1b_e	AB05
d2x_d1b_en	AD07

Signal Name	Pin
d2x_d1b_o	Y06
d2x_d1b_on	V07
forx0_clk	AD17
forx0_rxd0	AD19
forx0_rxd1	Y17
forx0_rxd2	Y18
forx0_rxd3	AB19
forx0_rxd4	Y19
forx0_rxd5	V17
forx0_rxd6	U17
forx0_rxd7	U19
forx0_rxd8	T17
forx0_rxd9	T18
forx1_clk	AD03
forx1_rxd0	T06
forx1_rxd1	T02
forx1_rxd2	T03
forx1_rxd3	U01
forx1_rxd4	U03
forx1_rxd5	V03
forx1_rxd6	Y01
forx1_rxd7	AB01
forx1_rxd8	Y02
forx1_rxd9	Y03
fotx0_txd0	V16
fotx0_txd1	P12
fotx0_txd2	R17
fotx0_txd3	T16
fotx0_txd4	T15
fotx0_txd5	R15
fotx0_txd6	N11
fotx0_txd7	M10
fotx0_txd8	L15
fotx0_txd9	K16
fotx1_txd0	L05
fotx1_txd1	K04
fotx1_txd2	R05
fotx1_txd3	N09
fotx1_txd4	T05
fotx1_txd5	T04
fotx1_txd6	U05
fotx1_txd7	V04
fotx1_txd8	R03
fotx1_txd9	P08
fotx_clk_in	U09
fotx_clk_out	V09
GND	A02
GND	A06
GND	A10
GND	A14
GND	A18
GND	C04
GND	C08
GND	C12
GND	C16
GND	E02
GND	E06
GND	E10
GND	E14
GND	E18
GND	G04

Signal Name	Pin
GND	G08
GND	G12
GND	G16
GND	J02
GND	J06
GND	J10
GND	J14
GND	J18
GND	L04
GND	L08
GND	L12
GND	L16
GND	M09
GND	N02
GND	N06
GND	N10
GND	N14
GND	N18
GND	P11
GND	R04
GND	R08
GND	R12
GND	R16
GND	U02
GND	U06
GND	U10
GND	U14
GND	U18
GND	W04
GND	W08
GND	W12
GND	W16
GND	AA02
GND	AA06
GND	AA10
GND	AA14
GND	AA18
GND	AC04
GND	AC08
GND	AC12
GND	AC16
GND	AE02
GND	AE06
GND	AE10
GND	AE14
GND	AE18
ibpen0	B17
ibpen1	B03
jtag_tck	E17
jtag_tdi	C19
jtag_tdo	E15
jtag_tms	C17
jtag_trst_L	G17
lssd_ce1_a	P13
lssd_ce1_b	W17
lssd_ce1_c1	AA17
lssd_ce1_c2	AC17
lssd_ce1_c3	AC19
lssd_scan_in0	AA15
lssd_scan_in1	AE17
lssd_scan_in10	AC07

Signal Name	Pin
lssd_scan_in11	AE07
lssd_scan_in12	AC05
lssd_scan_in13	AE05
lssd_scan_in14	AE03
lssd_scan_in15	AA05
lssd_scan_in2	AE15
lssd_scan_in3	AC15
lssd_scan_in4	AE13
lssd_scan_in5	AC13
lssd_scan_in6	W11
lssd_scan_in7	AA11
lssd_scan_in8	AA09
lssd_scan_in9	W09
lssd_scan_out0	AC01
lssd_scan_out1	AC03
lssd_scan_out10	E11
lssd_scan_out11	G11
lssd_scan_out12	C13
lssd_scan_out13	A13
lssd_scan_out14	C15
lssd_scan_out15	A15
lssd_scan_out2	AA03
lssd_scan_out3	W03
lssd_scan_out4	P07
lssd_scan_out5	P05
lssd_scan_out6	M05
lssd_scan_out7	M07
lssd_scan_out8	G09
lssd_scan_out9	E09
mon0	AD06
mon1	AB07
mon10	U11
mon11	AB12
mon12	R11
mon13	T11
mon14	AB13
mon15	AD14
mon2	T09
mon3	R09
mon4	AB08
mon5	T10
mon6	P10
mon7	Y12
mon8	V11
mon9	AA13
No Pin	A01
oob0_int_hi	A19
oob0_int_lo	D18
oob1_int_hi	D04
oob1_int_lo	D03
oob_ad0	B16
oob_ad1	F13
oob_ad2	G13
oob_ad3	B15
oob_ad4	D14
oob_ad5	H12
oob_ad6	E13
oob_ad7	B14
oob_clk	D06
oob_devsel0	H07

Signal Name	Pin
oob_devsel1	K08
oob_devsel2	B04
oob_valid_L	B08
oob_wait_L	B05
p2d_d0_id0	F16
p2d_d0_id1	H15
p2d_d0_id2	H19
p2d_d0_id3	L19
p2d_d0_id4	J17
p2d_d0_id5	K13
p2d_d0_id6	G15
p2d_d0_id7	H18
p2d_d1_id0	K01
p2d_d1_id1	H02
p2d_d1_id2	F04
p2d_d1_id3	G05
p2d_d1_id4	J03
p2d_d1_id5	L01
p2d_d1_id6	H01
p2d_d1_id7	L07
p2d_ic0	E07
p2d_ic1	J09
p2d_ic2	D08
p2d_ic3	L09
p2d_ic4	F08
p2d_ic5	F07
p2d_ic6	H08
p2d_ic7	D07
p2d_oc0	L11
p2d_oc1	D13
p2d_oc2	J11
p2d_oc3	F12
p2d_oc4	D12
p2d_oc5	H11
p2d_oc6	H09
p2d_oc7	K10
pin_diode0_reset_L	AB17
pin_diode0_sig_det	AD18
pin_diode1_reset_L	AB03
pin_diode1_sig_det	AD02
plllock	E05
plltest_in	E03
plltest_out	G03
pwrup_reset_L	B06
ref_clk	C03
ref_clkn	C01
soc_in	G07
soc_inn	F06
test_di1	A05
test_di2	A03
test_lt	C07
test_re	C05
test_ri	A07
UNUSED	A09
UNUSED	A11
UNUSED	A17
UNUSED	B07
UNUSED	B09
UNUSED	B10
UNUSED	B11

Signal Name	Pin
UNUSED	B12
UNUSED	B13
UNUSED	B18
UNUSED	C09
UNUSED	C11
UNUSED	D05
UNUSED	D09
UNUSED	D10
UNUSED	D11
UNUSED	D15
UNUSED	D16
UNUSED	E19
UNUSED	F05
UNUSED	F09
UNUSED	F10
UNUSED	F11
UNUSED	F14
UNUSED	G01
UNUSED	G19
UNUSED	H10
UNUSED	H13
UNUSED	J07
UNUSED	J13
UNUSED	K12
UNUSED	M01
UNUSED	M02
UNUSED	M03
UNUSED	M04
UNUSED	M06
UNUSED	M14
UNUSED	M16
UNUSED	M17
UNUSED	M18
UNUSED	M19
UNUSED	N01
UNUSED	N03
UNUSED	N05
UNUSED	N07
UNUSED	N13
UNUSED	N15
UNUSED	N17
UNUSED	N19
UNUSED	P01
UNUSED	P02
UNUSED	P03
UNUSED	P04
UNUSED	P06
UNUSED	P14
UNUSED	P16
UNUSED	P17
UNUSED	P18
UNUSED	P19
UNUSED	V10
UNUSED	W01
UNUSED	W19
UNUSED	Y09
UNUSED	Y10
UNUSED	Y11
UNUSED	AA01
UNUSED	AA19

Signal Name	Pin
UNUSED	AB09
UNUSED	AB10
UNUSED	AB11
UNUSED	AC09
UNUSED	AC11
UNUSED	AD01
UNUSED	AD09
UNUSED	AD10
UNUSED	AD11
UNUSED	AE09
UNUSED	AE11
vcsel0_laser_act	AE19
vcsel0_reset_L	AB18
vcsel1_laser_act	AE01
vcsel1_reset_L	AB02
VDD	A08
VDD	A12
VDD	C06
VDD	C14
VDD	E08
VDD	G02
VDD	G10
VDD	G18
VDD	J04
VDD	J12
VDD	J16
VDD	L10
VDD	M11
VDD	N08
VDD	N12
VDD	P09
VDD	R02
VDD	R06
VDD	R10
VDD	R14
VDD	R18
VDD	U04
VDD	U12
VDD	U16
VDD	W10
VDD	AA08
VDD	AC02
VDD	AC06
VDD	AC14
VDD	AC18
VDD	AE08
VDD	AE12
VDDQ	C18
VDDQ	E16
VDDQ	L14
VDDQ	L18
VDDQ	N16
VDDQ	W18
VDDQ	AA16
VDDQ	U08
VDDQ	W06
VDDQ	W14
VDDQ	AA12
VDDQ	AC10
VDDQ	AE04

Signal Name	Pin
VDDQ	AE16
VDDQ	C02
VDDQ	E04
VDDQ	L02
VDDQ	L06
VDDQ	N04
VDDQ	W02
VDDQ	AA04
VDDQ	A04
VDDQ	A16
VDDQ	C10
VDDQ	E12
VDDQ	G06
VDDQ	G14
VDDQ	J08
VDDA	E01
Vref	M08
Vref	K07
Vref	H14
Vref	K15
Vref	K09
Vref	K11
x2d_d0a_c	W15
x2d_d0a_cn	Y16
x2d_d0a_e	V14
x2d_d0a_en	T13
x2d_d0a_o	V18
x2d_d0a_on	T19
x2d_d0b_c	T12
x2d_d0b_cn	V13
x2d_d0b_e	U13
x2d_d0b_en	Y15
x2d_d0b_o	AD13
x2d_d0b_on	AB15
x2d_d1a_c	W05
x2d_d1a_cn	Y05
x2d_d1a_e	V06
x2d_d1a_en	U07
x2d_d1a_o	T01
x2d_d1a_on	V02
x2d_d1b_c	AD05
x2d_d1b_cn	AD08
x2d_d1b_e	V08
x2d_d1b_en	Y08
x2d_d1b_o	Y07
x2d_d1b_on	AA07
xcvr0_com_det	AD16
xcvr0_loopback	AB14
xcvr0_sig_det	AB16
xcvr1_com_det	AD04
xcvr1_loopback	AB06
xcvr1_sig_det	AB04

2.5.2 Package Dimensions

This section outlines the mechanical dimensions for the Dataslice device. The package is a 474 ceramic ball grid array (CBGA).

NOTE: Drawings are not to scale.

Figure 56. Dataslice CBGA Package Dimensions - Top and Side Views

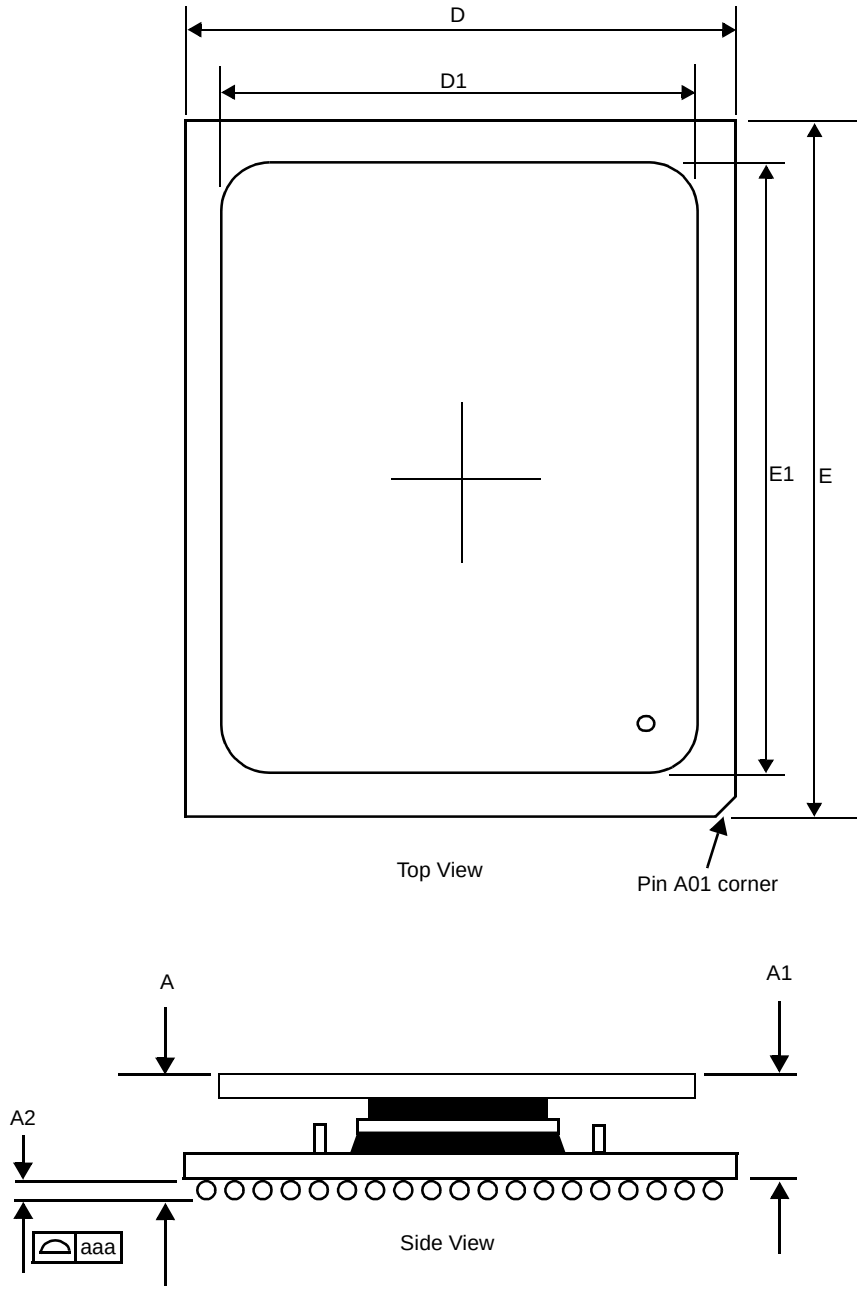


Figure 57. Dataslice CBGA Package Dimensions - Bottom View

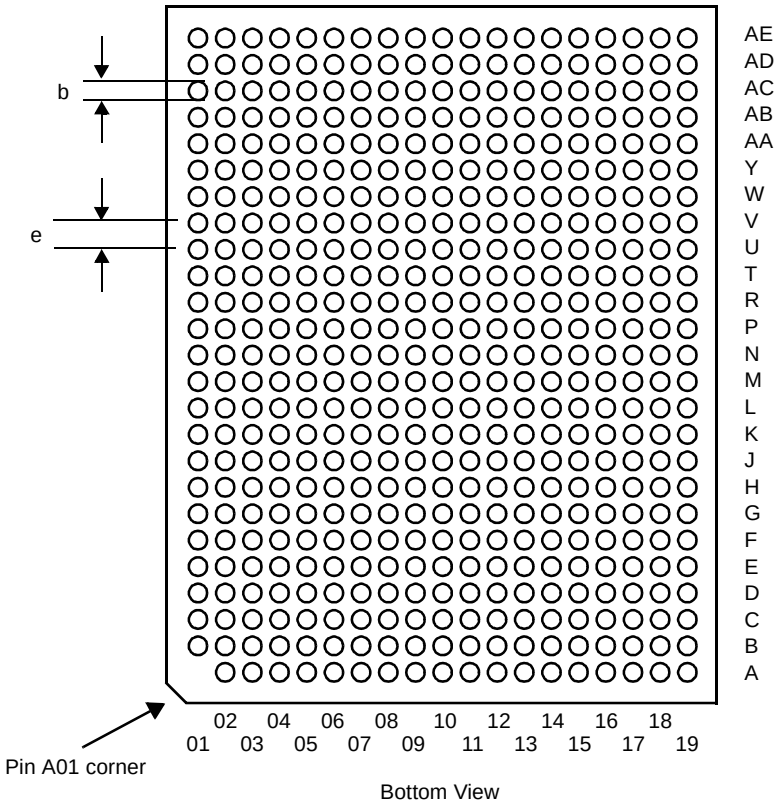


Table 19. Dataslice CBGA Mechanical Specifications

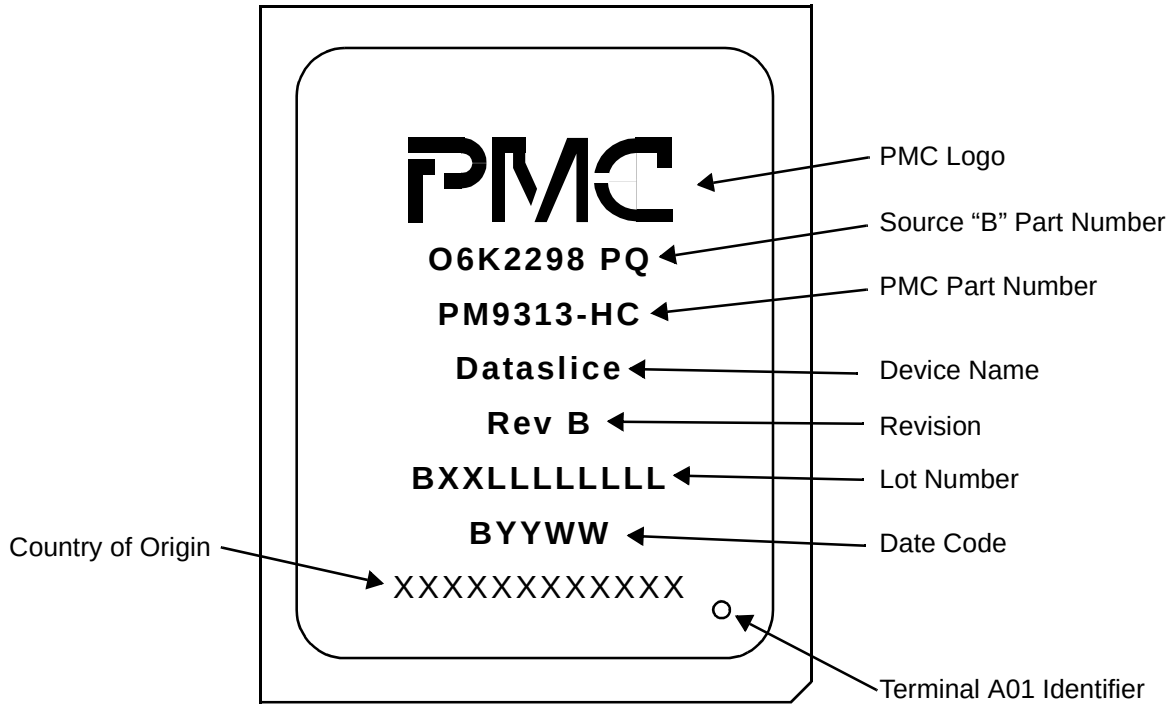
Symbol	e =1.27 mm			Units
	Minimum	Nominal	Maximum	
A		5.22		mm
A1		4.32		mm
A2		0.90		mm
aaa		0.15		mm
D		25.00		mm
D1		22.86		mm
E		32.50		mm
E1		30.48		mm
M		19 x 25		
N		474		
b	0.82		0.93	mm

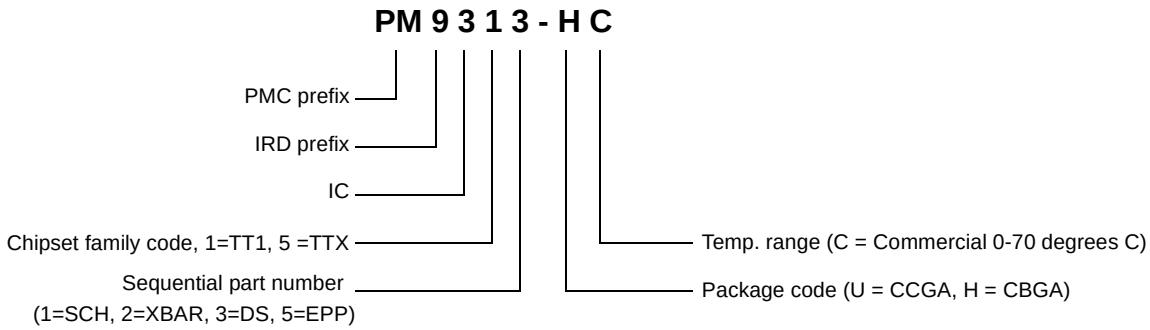
NOTE: Ball alloy is 90/10 PbSn

2.5.3 Part Number

The PMC-Sierra Part Number for the Dataslice is:

Dataslice	PM9313-HC





3 Enhanced Port Processor

This chapter contains information on the Enhanced Port Processor device, part number PM9315-HC, available from PMC-Sierra, Inc.

The Enhanced Port Processor (EPP) is based on the original TT1™ Port Processor, but has been greatly improved in terms of its ability to handle OC-48c channels. The original Port Processor (PP) could only support OC-48c at a single priority and required all linecards to be OC-48c. The EPP can support OC-48c at the four priority levels and can be used in OC-48c or OC-192c mode on a port-by-port basis. This enables the EPP to be used with the rest of the ETT1 Chip Set and to provide up to 128 ports of OC-48c. Also, multicast handling has been substantially improved over the PP; now the EPP can handle multicast at the OC-48c port level. However, the most important feature of the EPP is that it operates with the original TT1 Dataslice, Crossbar and Scheduler devices.

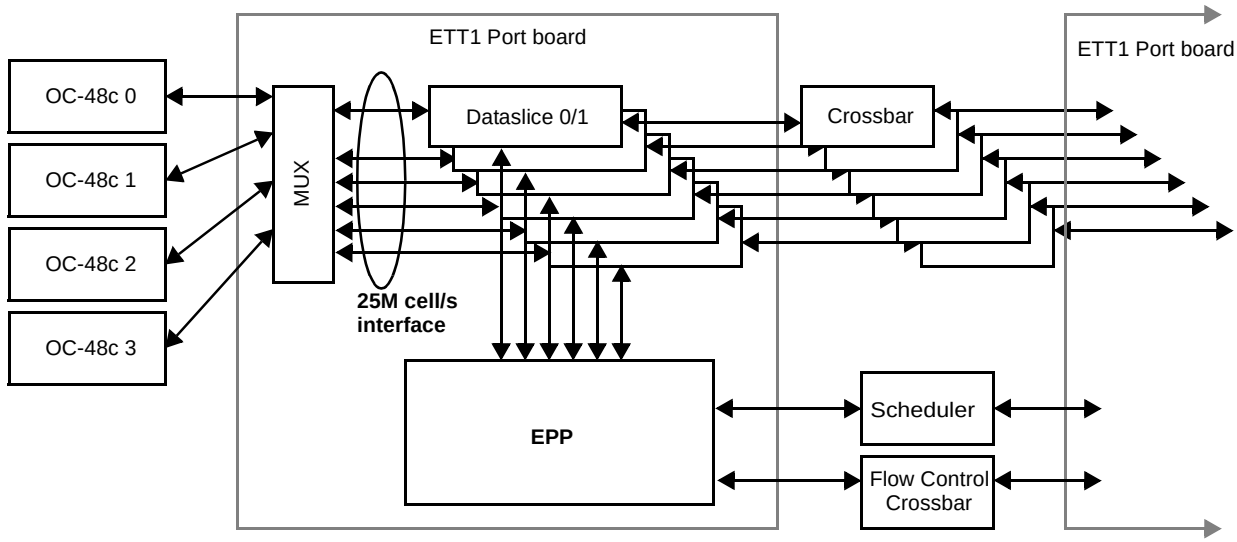
The main features of the EPP are:

- Four ports of OC-48c or one port of OC-192c.
- Communicates with other EPPs (*not* PPs) which can be connected either to OC-48c linecards or OC-192c linecards.
- Supports four priority levels for unicast and multicast for OC-48c or OC-192c.
- Supports TDM channels for OC-48c and OC-192c.
- Uses the LCS (Linecard to Switch) protocol.

NOTE: The EPP itself operates on an OC-192c rate stream; therefore, a multiplexer function is needed for OC-48c mode. This multiplexer sits between the four OC-48c rate linecards and the ETT1 port card, and maps the four incoming OC-48c rate streams onto a single stream to the EPP. Each request label contains the information needed to identify its source OC-48c linecard.

Figure 58 shows the interconnection of the EPP to other ETT1 devices.

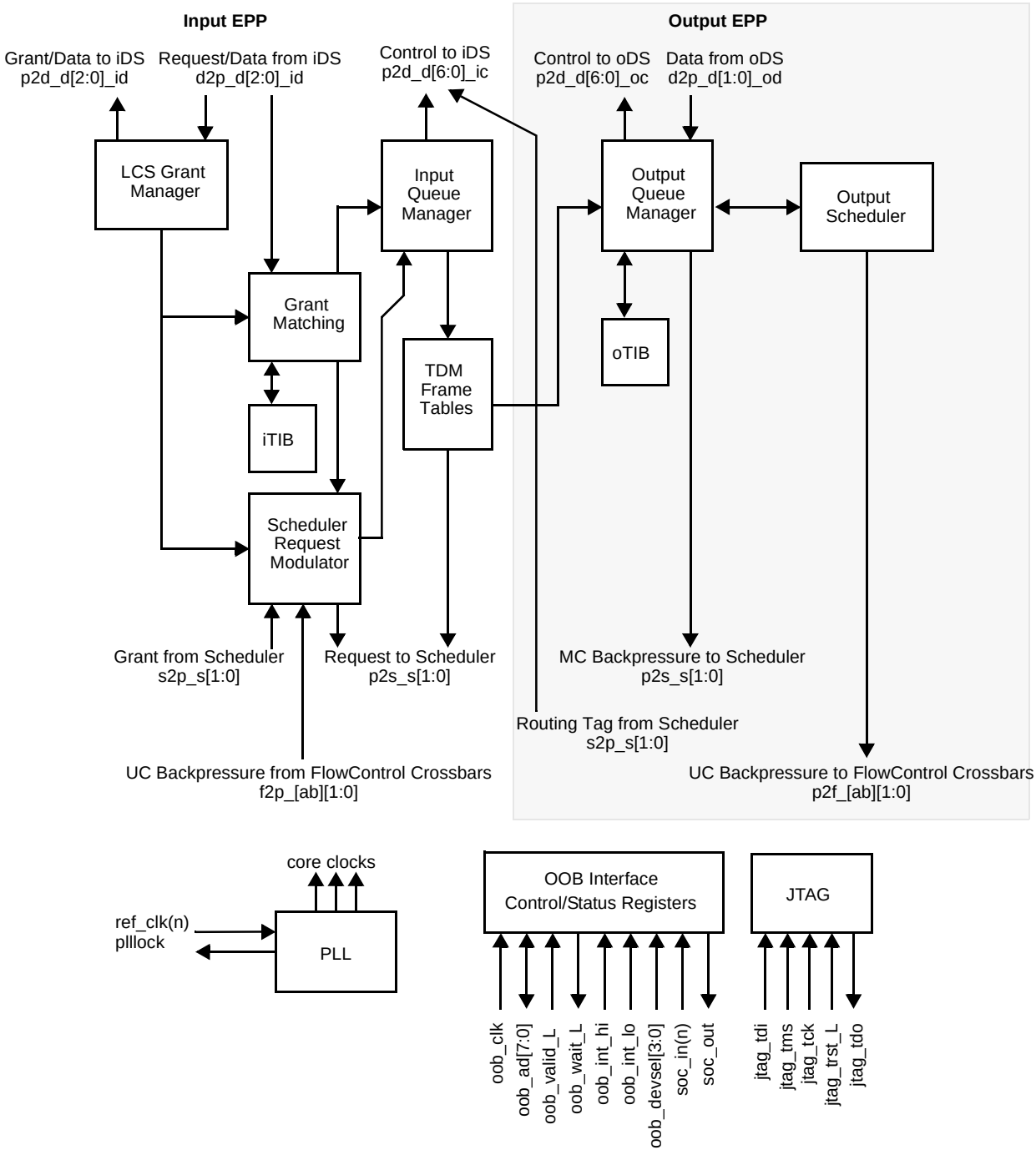
Figure 58. An ETT1 Port Operating in Sub-port Mode with Four OC-48c Linecards



3.1 EPP DATA FLOWS AND BLOCKS

This section describes the functionality of the EPP and its basic structure.

Figure 59. Functional Diagram of LCS Enhanced Port Processor



The EPP is logically split up into an input/ingress EPP (iEPP) and an output/egress EPP (oEPP). The LCS Grant Manager sends grants for input queues which have some space left, and arbitrates among competing flows. The Grant Matching logic matches incoming cell bodies with their LCS grant labels. The iEPP Tag Information Base (iTIB) is a table of multicast port fanouts indexed by LCS multicast tags. The TDM Frame Tables are indexed by TDM slot and contain scheduling information. The Scheduler Request Modulator provides OC-48c granularity backpressure for unicast flows, arbitrates among competing best-effort flows, and restores OC-48c granularity to port-granularity grants from the Scheduler chip. The Output Scheduler block arbitrates among all egress flows to determine which flow may send a cell to the linecard each cell time.

The following section describes cell flows within the EPP. The remaining sections describe the blocks of the EPP in more detail.

3.1.1 EPP Data Flows

3.1.1.1 iEPP Cell Arrival: LCS Request-Grant-Cell Flow

The EPP uses the LCS, version 2 (Linecard-to-Switch) protocol. This is a major enhancement to the original LCS protocol in that requests from the linecard are separated from their cell body, reducing the storage requirements of ETT1. The details of LCS are described elsewhere, but the basic operation is as follows:

1. A cell arrives at the ingress linecard and is destined for some output channel, O.
2. The linecard issues a request to the EPP, indicating the output queue (O). The cell body is *not* sent to the EPP at this time.
3. The EPP returns a grant to the linecard, indicating that the cell body can now be forwarded to the switch. This grant is also an implicit credit, so that the linecard can issue a further request for this output (O).

The input Dataslices, iDS0, iDS1 and iDS2, send their input request and cell data to the input EPP on d2p_d0_id[7:0], d2p_d1_id[7:0], d2p_d2_id[7:0], respectively. The LCS ingress header and first 10 bytes of cell payload are sent. The request portion of the LCS ingress header goes to the LCS Grant Manager, which arbitrates among all flows with requests (and space in the input queues) to choose the flow which will receive the next grant. The LCS Grant Manager sends the grant label and sequence number back to input Dataslice 0 via p2d_d0_id[7:0].

If the grant is unicast, its label information is put into a programmable delay line to the Scheduler Request Modulator. This allows requests to be sent to the Scheduler before the arrival of unicast cell bodies. The delay is programmed so that Scheduler grants will not come back before the arrival of the cell bodies, providing that the system round trip time constraint (64 celltimes) is met.

The LCS Grant Manager also sends the grant label and sequence number to the Grant Matching block, which matches incoming cell sequence numbers with previously stored grant sequence numbers, and restores the associated label information to the cell whose payload has just arrived. When a match is made, several things happen:

- The cell's label information is sent to the input Queue Manager, which determines where the cell will be stored in input Dataslice queue memory. If the flow's queue is already full, the cell is dropped and an interrupt is asserted. Otherwise, the EPP sends a write command & write address to all input Dataslices via p2d_d[6:0]_ic[7:0]. At the same time, the EPP sends egress cell label information to the input Dataslices, via p2d_d0_id[7:0] and p2d_d1_id[7:0]. This information is written into the egress cell header location of the cell.

- If the cell is multicast, its tag (from the label) is used as the index into the iTIB, to look up the port fanout. If the fanout is all-0's, the cell is dropped. Otherwise the fanout is sent to the Scheduler Request Modulator for arbitration with other unicast and multicast cells waiting to be requested.

3.1.1.2 iEPP Cell Departure: Scheduler Request-Grant Flow

Unicast and multicast requests to the Scheduler come from the Scheduler Request Modulator, while TDM requests to the Scheduler come from the TDM Frames block. Requests are sent to both (redundant) Schedulers, p2s_s[1:0].

The Scheduler Request Modulator receives unicast egress queue occupancy information from all EPPs' Output Schedulers via the Flow Control Crossbars, f2p_[a:b][1:0]. It uses this information to decide whether it can pass on requests to the Scheduler for cells destined for those queues. The Scheduler Request Modulator also keeps track of subport information for each request sent to the Scheduler, and restores the subport information to grants coming back from the Scheduler via s2p_s[1:0]. This is because the Scheduler only supports port-granularity in its arbitration.

When subport information has been restored to a valid Scheduler grant, the grant info is sent to the Input Queue Manager. If the specified input queue is empty, then an interrupt is asserted and no cell is sent. Otherwise, the Input Queue Manager generates a read command and read address, sent to the input Dataslices via p2d_d[6:0]_ic[7:0]. The input Dataslices then send the cell across the Data Crossbars.

The Scheduler also sends a reverse routing tag to the EPP, to indicate from which port this port will be receiving the next cell. This is passed to the input Dataslices via p2d_d[6:0]_ic[7:0]. The input Dataslices pass the tag on to the data Crossbars, to configure the Crossbar fabric.

3.1.1.3 oEPP Cell Arrival: Data Crossbar to oEPP Flow

After a cell traverses the data Crossbars, it is sent out to the output Dataslices. oDS0 and oDS1 send their cell data to the oEPP via d2p_d0_od[7:0] and d2p_d1_od[7:0]. The Output Queue Manager determines which output queue the cell is destined for, and generates a write command and write address for the output Dataslices. This is sent via p2d_d[6:0]_oc[7:0]. The Output Scheduler is notified that a cell has been added to that output queue.

If the incoming cell causes a multicast output queue to grow larger than the programmable backpressure threshold, then backpressure is asserted to the Scheduler via p2s_s[1:0]. Optionally, queue size can be ignored and arriving multicast cells will simply be dropped if the queue fills up.

The Scheduler's backpressure mechanism is not used for unicast because the Output Scheduler on each port provides the Scheduler Request Modulator on each port with subport-granularity queue information via the Flow Control Crossbars. The Scheduler's backpressure mechanism is not used for TDM because TDM programming should ensure that the output queues cannot overflow.

When multicast cells arrive at an oEPP in OC-48c mode, the source port and multicast tag are used to index the oTIB, to look up the egress subport fanout.

The egress subport fanout for TDM cells is taken from the TDM Frame table entry for the slot whose cells are currently arriving at oEPPs.

3.1.1.4 oEPP Cell Departure: oEPP to Linecard Flow

The Output Scheduler arbitrates among all flows with cells waiting to be sent to the linecards. When the oEPP is in OC-48 mode, it arbitrates for each output linecard in turn (subport 0, subport 1, subport 2, subport 3, subport 0,...).

When the Output Scheduler decides to send a cell, it passes label information to the Output Queue Manager which sends a read command and read address to the Output Dataslices via p2d_d[6;0]_oc[7:0].

When a multicast queue size falls below the programmable unbackpressure threshold, unbackpressure is sent to the Scheduler.

The Output Scheduler sends incremental credits for unicast egress queues to all EPPs' Scheduler Request Modulators via p2f_[a.b][1:0].

3.1.2 LCS Grant Manager

The purpose of the LCS Grant Manager is to prevent any input queue from overflowing, and to arbitrate among competing flows. It consists of linecard request counters, input queue debit counters, and an arbiter.

3.1.2.1 Linecard Request Counters

Each linecard request counter is initialized to 0 at reset, incremented when a new LCS Request arrives for the corresponding flow, and decremented when the LCS Grant Manager sends an LCS Grant to the requesting linecard. Request counters can also be updated by LCS Control Packet or OOB access. They are stored in the Linecard Request Counters memory. See Section 3.4 "Enhanced Port Processor Registers" for detailed address and data formats of this memory. These counters are maintained within the EPP and no customer interaction is required. However, customers may choose to monitor request counts on various flows, or to implement request count updates using this memory instead of LCS Control Packets.

For Control Packet and TDM queues, there is always one request counter per input queue. For multicast and unicast, if the EPP is in OC-48c mode, there are four request counters per input queue, one for each

subport. The LCS Grant Manager arbiter ensures that no request counter can be starved by the others; this is described in Section 3.1.2.3 “LCS Grant Manager Arbitration”.

The size of the request counters for TDM, multicast and unicast flows allows for request counts much larger than the input queue sizes, in order to avoid blocking requests from the input linecards. Maximum request counter values depend on whether the EPP is in OC-48c mode, and are shown Table 20.

Table 20. Linecard Request Count Maximum Values

Traffic Type	OC-48 port Max Request Count	OC-192c port Max Request Count
TDM	255	1023
Multicast	255	1023
Unicast	255	1023
Control Packets	7	7

Control packet requests have the highest priority of all traffic types, are always considered to have sufficient queue space, and are usually granted immediately. However, when the 1-in-N Idle Counter or 1-in-N NoGrant Counter expires, the LCS Grant Manager is prohibited from sending a grant during that celltime. To allow some margin for these cases, the control packet request counters are three (3) bits wide instead of one (1)bit.

When a TDM Sync command arrives from the Scheduler, all TDM request counters are immediately reset to 0. If the request counters weren’t already 0, the “Linecard Requests Flushed by TDM Sync” interrupt is raised. If a linecard sends too many requests for some flow and overflows that request counter, then the request counter will stick at the maximum value and the “Linecard Request Count Overflow” interrupt will be raised.

Multicast, TDM and CP tags from LCS Request labels must be stored in FIFOs corresponding to the request counters, since these tags must be returned to the linecards as part of LCS Grant labels. These FIFOs are contained in the Linecard Multicast Tags memory, the Linecard TDM Tags memory, and some auxiliary registers accessed along with CP request counters in the Linecard Request Count memory. These FIFOs are maintained within the EPP and no customer interaction is required.

3.1.2.2 Input Queue Debit Counters

Each input queue has one debit counter which is initialized to 0 at reset, incremented when the EPP sends an LCS Grant for that queue to a linecard, and decremented when the Scheduler sends a grant for that queue.

Control Packet queues are always considered to have 0 debits; Control Packet requests will always be immediately granted. So there are no debit counters for Control Packet input queues. Linecards must make sure to not overflow the Control Packet input queues (only CPU Control Packets go to the input queues; LCS Processed Control Packets are processed and dropped immediately).

3.1.2.3 LCS Grant Manager Arbitration

The LCS Grant Manager arbiter makes a linecard grant decision every celltime, based on information from the Linecard Request Counters and Input Queue Debit Counters. When the EPP is in OC-48c mode, each queue is shared by 4 request counters, one per subport. Therefore, in OC-48c mode, additional state information about multicast and unicast flows is used in the decision, in order to prevent any request counter from being starved by the other request counters contending for the same queue.

When the EPP is in OC-48c mode, the LCS Grant Manager arbitrates between the eligible flows requested by each subport in turn (subport 0, subport 1, subport 2, subport 3, subport 0,...). Thus each OC-48c linecard will receive at most one grant every 4 celltimes.

A flow is eligible for arbitration if its request counter is nonzero and its debit counter is less than the queue size. The arbiter enforces the ETT1 service class priority scheme (Control Packets > TDM > MC0 > UC0 > MC1 > ... > UC3). Within each unicast priority, the arbiter performs a round-robin on eligible flows. The order of flows considered in the round-robin is increasing {egress_subport, egress_port}. So for an egress port that is in OC-48c mode, its subports will appear once each 32 steps in the ordering. For an egress port in OC-192c mode, there is only one valid subport (0), but each egress_subport step for that port will map to subport 0.

Because of the strict order in which grants are given to each subport in OC-48c mode, and the unpredictable timing of cells leaving a congested queue, it would be possible for some request counters to never receive grants if the arbitration were left as a free-for-all. In order to avoid this, the LCS Grant Manager “reserves” the last space in each unicast or multicast queue for a rotating favorite subport. When a debit counter indicates that there is only one space left in a queue, only the request counter of the favorite subport for that queue is eligible. Whenever the favorite subport gets a grant, regardless of whether the queue is down to its last space, the favorite subport changes to the next subport with a non-zero request count. This algorithm ensures that no request counter can be starved.

3.1.3 Scheduler Request Modulator

The purpose of the Scheduler Request Modulator is to prevent unicast output queues from overflowing. The Scheduler’s backpressure mechanism only supports port-granularity backpressuring, which is not acceptable for unicast (subport-granularity) queues. So instead of sending backpressure commands from the oEPP to the Scheduler, all oEPPs send queue occupancy information to all iEPPs via the Flow Control Crossbars. The Scheduler Request Modulator uses the output queue occupancy information to prevent overflowing any unicast output queue; if there is not enough room left in an output queue, no more requests for that output queue will be sent to the Scheduler until more room becomes available.

Since unicast and multicast are interleaved per-priority in the overall ETT1 priority scheme, multicast requests are also arbitrated by the SRM.

The SRM consists of input queue request counters, output queue debit counters, an arbiter for deciding which flow to request to the Scheduler, and state for restoring subport information to Scheduler grants.

3.1.3.1 Input Queue Request Counters

Input Queue request counters are initialized to 0 at reset. Unicast and multicast request counters behave slightly differently.

Unicast request counters are incremented when unicast request/grant labels arrive via a programmable delay line from the LCS Grant Manager. Using the programmable delay line instead of waiting for unicast cells to arrive allows unicast requests to be sent to the Scheduler ahead of arrival so that Scheduler Grants can arrive at the same time as the cells, effectively masking Scheduler latency from the round trip time. The delay line is programmed so that Scheduler grants cannot arrive before the cells arrive, provided that the system round trip time constraint is met (64 celltimes from the grant to the cell arrival).

Multicast request counters are incremented when the Grant Matching block matches arriving multicast cells with their request/grant labels.

Unicast and multicast request counters are decremented as requests are sent to the Scheduler.

3.1.3.2 Output Queue Debit Counters

Output queue debit counters are initialized to 0 at reset. They are incremented when unicast requests are sent to the Scheduler. When incremental output queue credits arrive via the Flow Control Crossbars, they are subtracted from the appropriate debit counters. The counters are contained in the Output Queue Debit Counters Memory. See Section 3.4 “Enhanced Port Processor Registers” for detailed address and data formats. These counters are maintained within the EPP and no customer interaction is required, unless the customer chooses to implement the Flow Control recovery procedure outlined in “Enhanced Port Processor - Flow Control Crossbar” on page 94.

Output queue debit counts are maintained for unicast only, not multicast, since the Scheduler’s backpressure mechanism can be used to prevent multicast output queues from overflowing.

3.1.3.3 Scheduler Request Count Modulator Arbiter

The arbiter decides every celltime which best-effort flow, if any, to request to the Scheduler. A unicast flow is eligible for arbitration if its input queue request counter is nonzero and its output queue debit counter is less than the queue size. A multicast flow is eligible for arbitration if its request count is nonzero and if fewer than 64 requests for that multicast flow are currently waiting for grants from the Scheduler.. Multicast flows are included in the arbitration in order to enforce the ETT1 best-effort service class priority scheme (MC0 > UC0 > MC1 > ... > UC3), and to prevent 96-cell multicast input queues from exceeding the maximum Scheduler request count of 64.

Within each unicast priority, a round-robin is used to arbitrate among eligible flows. The order of the round-robin is increasing (egress_subport, egress_port), like the LCS Grant Manager arbiter.

3.1.4 Input Queue Manager

The Input Queue Manager stores arriving cells when the Grant Matching block has restored their grant label information, and sends departing cells when the Scheduler Request Modulator restores subport granularity to Scheduler grants. Cells are dequeued only when the Scheduler sends a dequeue command along with a grant.

Unicast input queues are per-priority, per-destination (port, subport). If the destination port is OC-192c, the only valid subport is 0 and the queue length is 64 cells. If the destination port is OC-48c, there is a separate queue per subport, 16 cells each.

Multicast input queues are per-priority and are 96 cells in length.

TDM input queues are per-subport. If this port is OC-192c, the only valid subport is 0 and the queue length is 96. If this port is OC-48c, there is a separate queue per subport, 24 cells each. When a TDM Sync command arrives from the Scheduler, TDM input queues are instantly emptied. If cells were present in any TDM input queue, the “Cells Flushed by TDM Sync” interrupt is raised.

Control Packet input queues are per-subport. The length is 8 regardless of OC-48c mode. Control Packet queues never receive grants from the Scheduler. Instead, OOB must write (any value) to the appropriate “Linecard to OOB FIFO Subport [3:0] Status” register in order to dequeue a Control Packet cell.

If a Control Packet arrives for a full queue, an “LC2OOB/CPU CP FIFO Overflow, LC[3:0]” interrupt is raised. If the Scheduler grants to an empty (unicast/multicast/tdm) queue, a “Scheduler Grant to Empty * Queue” is raised. If a cell of any traffic type arrives for a full queue, the cell will be dropped (no write command to the Dataslices). If there is a Scheduler grant to any empty queue, no cell will be transmitted (no read command to the Dataslices).

All input queues are simple circular buffers.

3.1.5 Output Queue Manager

The Output Queue Manager stores arriving cells from the Data Crossbars, and sends/dequeues departing cells under control of the Output Scheduler.

Unicast output queues are per-priority, per-source port, per-destination subport. If this EPP is OC-192c, the only valid subport is 0, and the length is 64. If this EPP is OC-48c, there is a separate output queue for each subport, and the length is 16. All unicast output queues are simple circular buffers.

Multicast output queues are per-priority, shared among output subports. When a multicast cell arrives, it is stored at one location regardless of how many destination subports it is to be sent to. In OC-48c mode, a separate list of queue locations is maintained for each subport, so that each subport appears to have a different queue with up to 96 locations. Since the 96 locations are actually shared by all subports, it is possible for congestion on one subport to back up the entire queue. If multicast backpressure is enabled, then multicast traffic destined for other ports at the same priority may be blocked. The multicast output queues are not simple circular buffers, instead they use a linked-list representation.

There is only one TDM output queue, length 96, shared among subports the using the same mechanism as the multicast output queues.

There is no queue for egress Control Packets; instead, there is one allocated cell location in Dataslice Output Queue Memory for each type of egress Control Packet. Several of these locations must be initialized by the OOB before the switch begins to operate; see Section 3.3 “Output Dataslice Queue Memory Allocation with EPP” for details.

The Output Queue Manager maintains head and length information for unicast queues in the Output Unicast Queue Information Memory, which is documented in Section 3.4 “Enhanced Port Processor Registers”. This information is maintained within the EPP and no customer interaction is required. However, in order to implement the Flow Control recovery procedure outlined in “Enhanced Port Processor - Flow Control Crossbar” on page 94, the customer will need to read the length of each affected unicast queue.

The Output Queue Manager is the source of the interrupts “Output UC or MC Queue Overflow” and “output TDM Queue Overflow”.

3.1.6 Output Scheduler

The purpose of the Output Scheduler is to arbitrate among cells in the output queues waiting to be sent to the output linecard(s). When the EPP is in OC-48c mode, the Output Scheduler arbitrates among cells destined for each subport in turn (subport0, subport1, subport2, subport3, subport0,...).

Unicast traffic can be shut off on a per-source-port basis in the Output Scheduler, by the “Freeze Unicast Output Scheduler Flows” register (see Section 3.4 “Enhanced Port Processor Registers”).

Hole requests from the linecard are sent to the Output Scheduler. When a hole request for a given priority is valid, the Output Scheduler will not send any unicast or multicast cell of that priority during that celltime. In OC-48c mode, separate hole requests come from each subport, and are delayed until that subport is due to receive a cell.

When in LCS Stop mode, the Output Scheduler does not send cells to the linecard. Since the LCS Start/Stop status is per linecard, subports on the same OC-48c mode port could have different Start/Stop status. Even though cells are not sent to the linecard, TDM cells continue to be dequeued from the TDM output queues. This is to prevent one stopped subport from blocking traffic to other subports, since the output TDM/MC queues are shared among subports. If LCS Lossy mode is in effect, then multicast cells will also continue to be dequeued.

The Output Scheduler also sends Control Packets to the linecard(s) under control of the “Send OOB2LC Control Packet” register. When the Scheduler sends a TDM Sync command, the Output Scheduler sends a TDM Sync Control Packet to the linecard(s). Several types of egress Control Packet must be written into the correct locations in the Dataslice Output Queue Memory by software before the switch begins to operate; see Section 3.3 “Output Dataslice Queue Memory Allocation with EPP” for details.

3.1.7 OOB Interface and Control/Status Registers

All of the devices have an OOB interface. This interface allows a single local CPU to control and monitor all of the devices within a core fabric. Internally, each device provides registers that can be mapped into the CPU’s address space. These registers are described in more detail in Section 3.4 “Enhanced Port Processor Registers”.

3.2 INPUT DATASLICE QUEUE MEMORY ALLOCATION WITH EPP

The following table lists the raw addresses within input Dataslice queue memory for all input queues (VOQs) managed by the iEPP. These addresses apply to each input Dataslice in the port; to read or write a complete cell in input Dataslice queue memory, the raw address must be translated into OOB address offsets, and then those offsets should be read/written for each Dataslice.

Priority is the lower 2 bits of the traffic type for Unicast or Multicast queues; Port is 5 bits, and Subport is 2 bits. “x” is a single bit that can be 0 or 1, used to denote offsets within queues. “yy” can only be 00, 01, or 10, and is used to denote offsets within queues that are 24 or 96 cells in length.

Table 21. Input Dataslice Queue Memory Allocation

Traffic Type	Type of Ingress Linecard	Queue size	Total # of Queues	Dataslice Queue Mem Address Mapping	Raw Address Range
Unicast	either (OC-48c egress port)	16	512	0,Priority,Egress Port, Egress Subport,xxxx	0x0000-0x1FFF
Unicast	either (OC-192c egress port)	64	128	0,Priority,Egress Port,xxxxxx	0x0000-0x1FFF
Multicast	either	96	4	10000,yy,xxxxx,Priority	0x2000-0x217F
TDM	OC-48c	24	4	1000011,yy,xxx,Ingress Subport	0x2180-0x21DF
TDM	OC-192c	96	1	1000011,yy,xxxxx	0x2180-0x21DF
CPU CP	OC-48c	8	4	100001111,Ingress Subport,xxx	0x21E0-0x21FF
CPU CP	OC-192c	8	1	100001111,00,xxx	0x21E0-0x21E7

Each raw address given above is for a 48-bit word in Dataslice queue memory. In order to access the memory with OOB, a raw address must be converted into two OOB address offsets because OOB accesses are 32-bit. The following formula is used:

```
oob_offset_16_MSBs = 0x40000 | (raw_address << 3)
oob_offset_32_LSBs = 0x40000 | (raw_address << 3) | 0x4
```

3.3 OUTPUT DATASLICE QUEUE MEMORY ALLOCATION WITH EPP

The following table lists the raw addresses within output Dataslice queue memory for all output queues (VIQs) managed by the oEPP. These addresses apply to each output Dataslice in the port; to read or write a complete cell in output Dataslice queue memory, the raw address must be translated into OOB address offsets, and then those offsets should be read/written for each Dataslice.

NOTE: There is no output queue for Control Packets. Instead, one cell location is statically allocated for each type of outgoing Control Packet. OOB must initialize the processed CP locations in output Dataslice queue memory before starting EPP operation.

Priority is the lower 2 bits of the traffic type for Unicast or Multicast queues; Port is 5 bits, and Subport is 2 bits. “x” is a single bit that can be 0 or 1, used to denote offsets within queues. “yy” can only be 00, 01, or 10, and is used to denote offsets within queues that are 96 cells in length.

Table 22. Output Dataslice Queue Memory Allocation

Traffic Type	Type of Egress Linecard	Queue Size	Total # of Queues	Dataslice Queue Mem Address Mapping	Raw Address Range
Unicast	OC-48c	16	512	0,Priority,Ingress Port,Egress Subport,xxxx	0x0000-0x1FFF
Unicast	OC-192c	64	128	0,Priority,Ingress Port,xxxxxx	0x0000-0x1FFF
Multicast	OC-48c OC-192c	96	4	10000,yy,xxxxx,Priority	0x2000-0x217F
TDM	OC-48c OC-192c	96	1	1000011,yy,xxxxx	0x2180-0x21DF
CPU (OOB to LC) CP	OC-48c OC-192c	1	1	10000111100000	0x21E0
LCS Stop CP	OC-48c OC-192c	1	1	10000111100010	0x21E2
LCS Start CP	OC-48c OC-192c	1	1	10000111100011	0x21E3
TDM Sync Sel 0 CP	OC-48c OC-192c	1	1	10000111100100	0x21E4
TDM Sync Sel 1 CP	OC-48c OC-192c	1	1	10000111100101	0x21E5

Each raw address given above is for a 48-bit word in Dataslice queue memory. In order to access the memory with OOB, a raw address must be converted into two OOB address offsets because OOB accesses are 32-bit. The following formula is used:

```
oob_offset_16_MSBs = 0x60000 | (raw_address << 3)
oob_offset_32_LSBs = 0x60000 | (raw_address << 3) | 0x4
```

The table below shows the data format of egress Control Packets, and the OOB address offsets within each logical Dataslice where data must be written. The values written to shaded locations are left up to the

customer's implementation. The CPU Control Packet locations should be written whenever the customer's software is about to send a CPU control packet to the linecard. The given locations for the other types of Control Packets **must** be initialized after the Dataslices are reset and before traffic is sent through the switch.

Table 23. EPP Egress Control Packet Data Format and Dataslice OOB Addressing

Control Packet Type	OOB Address Offset	DS0	DS1	DS2	DS3...DS11
CPU (CRC-16 header mode)	(msb) 0x70F00	0x0000	0x0031		
	(lsb) 0x70F04	0x00000010			
CPU (CRC-8 header mode)	(msb) 0x70F00	0x0000	0x0057		
	(lsb) 0x70F04	0x00000010			
LCS Stop	(msb) 0x70F10	0x0000	0x0000	0x0000	
	(lsb) 0x70F14	0x00000000	0x02000000	0x00006E9F	
LCS Start	(msb) 0x70F18	0x0000	0x0000	0x0000	
	(lsb) 0x70F1C	0x00000000	0x02800000	0x0000C566	
TDM Sync Sel 0	(msb) 0x70F20	0x0000	0x0000	0x0000	
	(lsb) 0x70F24	0x00000000	0x00000000	0x0000E139	
TDM Sync Sel 1	(msb) 0x70F28	0x0000	0x0000	0x0000	
	(lsb) 0x70F2C	0x00000000	0x00800000	0x00004AC0	

3.4 ENHANCED PORT PROCESSOR REGISTERS

The Enhanced Port Processor device select low-order bits are hard wired on the board by four pins (oobdev_sel[3:0]) on the Enhanced Port Processor package. See section 1.7.1.3 “Individual Device Selects” on page 79 for more information.

3.4.1 Enhanced Port Processor Summary

The following table is a summary of information for all registers in the Enhanced Port Processor. See the following Descriptions section for more information on individual registers.

Read and Clear means that reading the register causes it to be cleared (reset to zero).

Table 24. Enhanced Port Processor Register Summary

Address	Symbol	Register	Access	Default Value
00000h	ESTS	Status	Read Only	51000000h
00004h	ETKNRS	Reset and Control	Read/Write	00000031h
00008h	EIRLMSK	Low Priority Mask	Read/Write	00000000h
0000Ch	EIRHMSK	High Priority Mask	Read/Write	00000000h
00010h	EIR	Interrupt Register	Read and Clear	00000000h
00014h	ELPAIBIM	Low Priority AIB Interrupt Mask	Read/Write	00000000h
00018h	EHPAIBM	High Priority AIB Interrupt Mask	Read/Write	00000000h
0001Ch	EAIBIR	AIB Interrupt Register	Read and Clear	00000000h
00020h	ELPICIR	Low Priority Incremental Credit Interrupt Mask	Read/Write	00000000h
00024h	EHPICIR	High Priority Incremental Credit Interrupt Mask	Read/Write	00000000h
00028h	EIICIR	Invalid Incremental Credit Interrupt Register	Read and Clear	00000000h
00030h	EAIBRS	AIB Reset	Read/Write	0000003Fh
00034h	EAIBRDY	AIB Ready	Read Only	00000000h
00038h	EEN	Enable Port Processor	Read/Write	00000000h

Table 24. Enhanced Port Processor Register Summary (Continued)

Address	Symbol	Register	Access	Default Value
0003Ch	EAIBEN	AIB Enable Transmitter	Read/Write	00000000h
00040h	ELCSSSS	LCS Started/Stopped Status/Control	Read/Write	00000000h
00044h	ELGCBD	Lost Grant or Cell Body Debug Info	Read and Clear	00000000h
00048h	ESGEQD	Scheduler Grant to Empty Queue Debug Info	Read and Clear	00000000h
0004Ch	ELCSECT	LCS Header CRC Error Count	Read and Clear	00000000h
00050h	ELCSP0CT	LCS Processor CP Subport 0 CRC Error Count	Read and Clear	00000000h
00054h	ELCSP1EC	LCS Processor CP Subport 1 CRC Error Count	Read and Clear	00000000h
00058h	ELCSP2EC	LCS Processor CP Subport 2 CRC Error Count	Read and Clear	00000000h
0005Ch	ELCSP3EC	LCS Processor CP Subport 3 CRC Error Count	Read and Clear	00000000h
00060h	EIS0VCT	Input Subport 0 Valid Count	Read and Clear	00000000h
00064h	EIS1VCT	Input Subport 1 Valid Count	Read and Clear	00000000h
00068h	EIS2VCT	Input Subport 2 Valid Count	Read and Clear	00000000h
0006Ch	EIS3VCT	Input Subport 3 Valid Count	Read and Clear	00000000h
00070h	EOS0VCT	Output Subport 0 Valid Count	Read and Clear	00000000h
00074h	EOS1VCT	Output Subport 1 Valid Count	Read and Clear	00000000h
00078h	EOS2VCT	Output Subport 2 Valid Count	Read and Clear	00000000h
0007Ch	EOS3VCT	Output Subport 3 Valid Count	Read and Clear	00000000h

Table 24. Enhanced Port Processor Register Summary (Continued)

Address	Symbol	Register	Access	Default Value
00080h	ES0OOBS	Subport 0 to OOB FIFO Status	Read/Write	00000000h
00084h	ES1OOBS	Subport 1 to OOB FIFO Status	Read/Write	00000000h
00088h	ES2OOBS	Subport 2 to OOB FIFO Status	Read/Write	00000000h
0008Ch	ES3OOBS	Subport 3 to OOB FIFO Status	Read/Write	00000000h
00090h	ESOBLCP	Send OOB to Linecard Control Packet	Read/Write	00000000h
00094h	ESTDMSSS	Suggested TDM Sync Subport Select	Read/Write	00000000h
00098h	EMCIFCT	Multicast Invalid Fanout Count	Read and Clear	00000000h
0009Ch	EIDMA	Internal Delay Matching Adjustments	Read/Write	03A8D66Bh
000A0h	EOPBPTHR	Output Backpressure/Unbackpressure Threshold	Read/Write	00001C46h
000A4h	EEMM	Egress OC-48/OC-192c Mode Map	Read/Write	00000000h
000A8h	ETDMO	TDM Offset Value	Read/Write	00000080h
000ACh	EFCTL	Freeze Control	Read/Write	00000000h
000B0h	EFUOS	Freeze Unicast Output Scheduler Flows	Read/Write	00000000h
000B4h	EMPD	Multicast Priority Drop	Read/Write	00000000h
000B8h	EFTCTRL	Fault Tolerance Status/Control	Read/Write	00000000h
000BCh	ERFRSSC	Refresh Schedulers Command	Write Only	00000000h
000C0h	ESTRTSC	Go Schedulers Command	Write Only	00000000h
000C4h	EIDLCT	Idle Count Value	Read/Write	00000000h
000C8h	ETDMEN	Enable TDM Sync Generation	Read/Write	00000000h
000CCh	ETDMCTRL	TDM Sync Generation Control	Read/Write	00000000h
000DOh	ENOGCT	No Grant Count	Read/Write	FFFFFFFFh
000D4h	EESR48SD	End of Sch Refresh/Sch OC-48 Sync ModCount	Write Only	00000000h
00100h	EPLLREG	PLL Status/Control	Read/Write	0001447Ch
02000-02FFCh	ETDMFT0M	TDM Frame Table 0 Memory	Read/Write	000000000h

Table 24. Enhanced Port Processor Register Summary (Continued)

Address	Symbol	Register	Access	Default Value
03000-03FFCh	ETDMFT1M	TDM Frame Table 1 Memory	Read/Write	000000000h
04000-07FFCh	EITIBM	ITIB Memory	Read/Write	000000000h
08000-0FFFCh	ESRCTM	Subport Request Count Memory	Read/Write	000000000h
16000-17FFCh	EIQIM	Input Queue Information Memory	Read/Write	000000000h
1C000-1DFFCh	EWSRCT	Waiting Scheduler Request Count Memory	Read/Write	Unknown
1E000-1E7FCh	EOUCQDCT	Output UC Queue Debit Count Memory	Read/Write	000000000h
20000-21FFCh	EOUCQI	Output UC Queue Information Memory	Read/Write	000000000h
3E000-3FFFCh	EOMCQDCT	Output MC Queue Drop Counters	Read/Write	000000000h
08000-FFFFCh	EOTIBM	OTIB Memory	Read/Write	000000000h

3.4.2 Enhanced Port Processor Register Descriptions

Read and Clear means that reading the register causes it to be cleared (reset to zero).

All bits labeled as Reserved should be set to 0.

3.4.2.1 Status

Symbol: ESTS

Address Offset: 00000h
Default Value: 51000000h
Access: Read Only

Bits	Description
31:28	**Chip ID Number.** Identifies the specific ETT1 device.
27:24	**Chip Revision Number.**
23:2	Reserved.
1	**High Priority Interrupt.** 1 = an outstanding high priority interrupt. One of the bits in the Interrupt Register is set, and is enabled via its corresponding high priority mask.
0	**Low Priority Interrupt.** 1 = an outstanding low priority interrupt. One of the bits in the Interrupt Register is set, and is enabled via its corresponding low priority mask.

3.4.2.2 Reset and Control

Symbol: ETKNRS

Address Offset: 00004h
Default Value: 00000031h
Access: Read/Write

Reset and Control register.

Bits	Description
31-15	Reserved.
14	**13th and 14th Dataslice Enable.** This bit must be set before bringing up the Dataslices' fiber-optics links in systems that use 14 Dataslices. Note: This bit will read out in bit position 6.
13:6	Reserved
5	**Generate Truncated 16-bit CRC.** For Switch to Linecard LCS headers, generate a 16-bit CRC (0x11021) over the header, but only send the lower 8 bits, if this bit is 1. If this bit is 0, generate an 8-bit CRC (0x107) over the header and send that.

Bits	Description (Continued)
4	**Check full 16-bit CRC.** For Linecard to Switch LCS headers, check a 16-bit CRC (0x11021) over the header, if this bit is 1. If this bit is 0, check an 8-bit CRC (0x107). This applies only to the LCS header and not to Processed Control Packet payload data.
3	**Include MuxChip's Subport ID in CRC gen.** If this bit is 1, generate the Switch to Linecard LCS header CRC including the MUX field in the grant label. If 0, generate the CRC with the MUX field masked to 0.
2	**Include MuxChip's Subport ID in CRC check.** If this bit is 1, check the Linecard to Switch LCS header CRC including the MUX field in the request label. If 0, check the CRC with the MUX field masked to 0.
1	**Small System Mode.** If this bit is set, only Flow Control Crossbar A is used; FC Crossbar B is ignored. Port numbers must be even numbers. This corresponds to the Data Crossbars' 16x16 and 8x8 modes.
0	**Reset**. Writing a 1 to this location will reset the entire chip. It is equivalent to a hardware reset. This register is cleared automatically when the chip is reset. Soft reset takes 1mS to complete. **NOTE:** After resetting an EPP, write 0 to offsets 0x1d000 through 0x1d7fc in that EPP (a total of 512 OOB writes). When all EPPs are reset simultaneously (using a broadcast OOB write to the EPP Control register), 512 broadcast OOB writes can be used to reset those locations in all EPPs.

3.4.2.3 Low Priority Mask

Symbol: EIRLMSK

Address Offset: 00008h
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:0	**Low Priority Mask**. Mask bits for low priority interrupts. Each mask bit is set to 1 to enable a low priority interrupt when the corresponding bit in the Interrupt Register is 1.

3.4.2.4 High Priority Mask

Symbol: EIRHMSK

Address Offset: 0000Ch
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:0	**High Priority Mask**. Mask bits for high priority interrupts. Each mask bit is set to 1 to enable a high priority interrupt when the corresponding bit in the Interrupt Register is 1.

3.4.2.5 Interrupt Register

Symbol: EIR

Address Offset: 00010h
Default Value: 00000000h
Access: Read and Clear

Interrupt Register.

Bits	Description
31	**Scheduler Grant to Empty TDM Queue.** This is set if a Scheduler grant arrives for a TDM input queue (VOQ) which has no cells waiting.
30	**Scheduler Grant to Empty MC Queue**. This is set if a Scheduler grant arrives for a MC input queue (VOQ) which has no cells waiting.
29	**Cells Flushed by TDM Sync.** This is set if a TDM Sync clears waiting cells from TDM input queues. If a TDM Sync arrives and the TDM input queues are already empty, this is not set.
28	**Linecard Requests Flushed by TDM Sync.** This is set if a TDM Sync clears nonzero TDM Subport request counters.
27	**Scheduler Grant to Nonwaiting UC Queue.** This is set if a Scheduler Grant arrives for a priority and port for which no subport queues are expecting grants.
26	**oDS to oEPP Valid Mismatch.** This is set if the three oDS to oEPP frame's valid bit does not match.
25	**iDS to iEPP Valid Mismatch.** This is set if the three iDS to iEPP frame's valid bit does not match.
24	**Grant or Cell Body Dropped.** This is set if 128 OC-192c celltimes have passed since a grant, but no cell payload has arrived with the grant sequence number.
23	**Invalid Cell Sequence Number from Subport 3.** This is sent if Subport 3 has sent a cell payload with a sequence number which does not match any outstanding grant.
22	**Invalid Cell Sequence Number from Subport 2.** This is sent if Subport 2 has sent a cell payload with a sequence number which does not match any outstanding grant.
21	**Invalid Cell Sequence Number from Subport 1.** This is sent if Subport 1 has sent a cell payload with a sequence number which does not match any outstanding grant.
20	**Invalid Cell Sequence Number from Subport 0.** This is sent if Subport 0 has sent a cell payload with a sequence number which does not match any outstanding grant.

Bits	Description (Continued)
19	**Received Multicast Cell with Invalid Fanout.** This is set if a multicast cell arrives and its tag's fanout is all-0s.
18	**CRC Error in LCS Header from LC??.** This is set if an incoming cell has an LCS header CRC error or has a code error in input dataslices 0, 1, or 2. Since the MUX field of the request label could be corrupted, the source subport is unknown.
17	**CRC Error in Processed Control Packet from Subport 3.** This is set if there is a CRC Error over the first 10 bytes of payload data for a PCP from Subport 3.
16	**CRC Error in Processed Control Packet from Subport 2.** This is set if there is a CRC Error over the first 10 bytes of payload data for a PCP from Subport 2.
15	**CRC Error in Processed Control Packet from Subport 1.** This is set if there is a CRC Error over the first 10 bytes of payload data for a PCP from Subport 1.
14	**CRC Error in Processed Control Packet from Subport 0.** This is set if there is a CRC Error over the first 10 bytes of payload data for a PCP from Subport 0.
13	**Received LCS Start Control Packet.** This notifies OOB that a Subport has sent a Processed Control Packet of type "LCS Start".
12	**Received LCS Stop Control Packet.** This notifies OOB that a Subport has sent a Processed Control Packet of type "LCS Stop".
11	**Received LC2OOB/CPU Control Packet from Subport 3.** This notifies OOB that Subport 3 has sent a CPU Control Packet.
10	**Received LC2OOB/CPU Control Packet from Subport 2.** This notifies OOB that Subport 2 has sent a CPU Control Packet.
9	**Received LC2OOB/CPU Control Packet from Subport 1.** This notifies OOB that Subport 1 has sent a CPU Control Packet.
8	**Received LC2OOB/CPU Control Packet from Subport 0.** This notifies OOB that Subport 0 has sent a CPU Control Packet.
7	**LC2OOB/CPU Control Packet FIFO Overflow, Subport 3.** This is set if Subport 3 overflows its 8-cell FIFO of CPU Control Packets to OOB.
6	**LC2OOB/CPU Control Packet FIFO Overflow, Subport 2.** This is set if Subport 2 overflows its 8-cell FIFO of CPU Control Packets to OOB.
5	**LC2OOB/CPU Control Packet FIFO Overflow, Subport 1.** This is set if Subport 1 overflows its 8-cell FIFO of CPU Control Packets to OOB.
4	**LC2OOB/CPU Control Packet FIFO Overflow, Subport 0.** This is set if Subport 0 overflows its 8-cell FIFO of CPU Control Packets to OOB.
3	**Line Card Request Count Overflow.** This is set if a Subport sends too many requests (the counter will stick at max value).
2	**Scheduler Grant to Empty UC Queue.** This is set if the scheduler grants to an empty UC input queue (VOQ).
1	**Output MC or UC Queue Overflow.** This is set if a cell from the crossbar overflows a MC or UC output queue.
0	**Output TDM Queue Overflow.** This is set if a cell from the Crossbar overflows a TDM output queue.

3.4.2.6 Low Priority AIB Interrupt Mask

Symbol: ELPAIBIM

Address Offset: 00014h
Default Value: 00000000h
Access: Read/Write

Enables low priority interrupts.

Bits	Description
31:24	Reserved.
23:0	**Mask bits for low priority interrupts.** Each mask bit is set to 1 to enable a low priority interrupt when the corresponding bit in the AIB Interrupt register is 1.

3.4.2.7 High Priority AIB Interrupt Mask

Symbol: EHPAIBM

Address Offset: 00018h
Default Value: 00000000h
Access: Read/Write

Enables low priority interrupts.

Bits	Description
31:24	Reserved.
23:0	**Mask bits for high priority interrupts.** Each mask bit is set to 1 to enable a high priority interrupt when the corresponding bit in the AIB Interrupt register is 1.

3.4.2.8 AIB Interrupt Register

Symbol: EAIBIR

Address Offset: 0001Ch
Default Value: 00000000h
Access: Read and Clear

Interrupt Register

Bits	Description
31:24	Reserved.
23	**Flow Control Crossbar A1 CRC Error**
22	**Flow Control Crossbar A0 CRC Error**
21	**Flow Control Crossbar A1 Ready Down**
20	**Flow Control Crossbar A0 Ready Down**
19	**Flow Control Crossbar A1 Ready Up**
18	**Flow Control Crossbar A0 Ready Up**
17	**Flow Control Crossbar A Both Links CRC Error.** If set, neither A0 nor A1 provided valid incremental credits during at least one celltime, so check the Invalid Incremental Credits interrupt register to see which ports' output queue credits need repair.
16	**Flow Control Crossbar A Frame Mismatch.** Though apparently valid, the frames received from A0 and A1 were not identical. In this case, the Primary Flow Control Crossbar A is selected.
15	**Flow Control Crossbar B1 CRC Error**
14	**Flow Control Crossbar B0 CRC Error**
13	**Flow Control Crossbar B1 Ready Down**
12	**Flow Control Crossbar B0 Ready Down**
11	**Flow Control Crossbar B1 Ready Up**
10	**Flow Control Crossbar B0 Ready Up**
9	**Flow Control Crossbar B Both Links CRC Error**. If set, neither B0 nor B1 provided valid incremental credits during at least one celltime, so check the Invalid Incremental Credits interrupt register to see which ports' output queue credits need repair.
8	**Flow Control Crossbar B Frame Mismatch**. Though apparently valid, the frames received from B0 and B1 were not identical. In this case the Primary Flow Control Crossbar B is selected.
7	**Scheduler 1 CRC Error**
6	**Scheduler 0 CRC Error**
5	**Scheduler 1 Ready Down**
4	**Scheduler 0 Ready Down**
3	**Scheduler 1 Ready Up**
2	**Scheduler 0 Ready Up**

Bits	Description (Continued)
1	**Scheduler Both Links CRC Error.** If set, traffic for this port is frozen and OOB must initiate Scheduler Refresh to repair Scheduler state for this port.
0	**Scheduler Frame Mismatch.** Though apparently valid, the frames received from Sched0 and Sched1 were not identical. In this case the Primary Sched is selected.

3.4.2.9 Low Priority Incremental Credit Interrupt Mask

Symbol: ELPICIR

Address Offset: 00020h
Default Value: 00000000h
Access: Read/Write

Enables a low priority interrup

Bits	Description
31:0	**Mask bits for low priority interrupts.** Each mask bit is set to 1 to enable a low priority interrupt when the corresponding bit in the Invalid Incremental Credit Interrupt register is 1.

3.4.2.10 High Priority Incremental Credit Interrupt Mask

Symbol: EHPICIR

Address Offset: 00024h
Default Value: 00000000h
Access: Read/Write

Enables a high priority interrupt.

Bits	Description
31:0	**Mask bits for High priority interrupts.** Each mask bit is set to 1 to enable a high priority interrupt when the corresponding bit in the Invalid Incremental Credit Interrupt register is 1.

3.4.2.11 Invalid Incremental Credit Interrupt Register

Symbol: EIICIR

Address Offset: 00028h
Default Value: 00000000h
Access: Read and Clear

Interrupt register.

Bits	Description
31:0	**Each bit corresponds to a single port. Invalid** incremental credits were received from either Flow Control Crossbar A or Flow Control Crossbar B.

3.4.2.12 AIB Reset

Symbol: EAIBRS

Address Offset: 00030h
Default Value: 0000003Fh
Access: Read/Write

Resets AIB link for each port.

Bits	Description
31:6	Reserved.
5	**Flow Control Crossbar A1 AIB Reset.**
4	**Flow Control Crossbar A0 AIB Reset.**
3	**Flow Control Crossbar B1 AIB Reset.**
2	**Flow Control Crossbar B0 AIB Reset.**
1	**Scheduler 1 AIB Reset.**
0	**Scheduler 0 AIB Reset.**

3.4.2.13 AIB Ready

Symbol: EAIBRDY

Address Offset: 00034h
Default Value: 00000000h
Access: Read Only

Indicates if the corresponding AIB link is ready.

Bits	Description
31:6	Reserved.
5	**Flow Control Crossbar A1 AIB Ready.**
4	**Flow Control Crossbar A0 AIB Ready.**
3	**Flow Control Crossbar B1 AIB Ready.**
2	**Flow Control Crossbar B0 AIB Ready.**
1	**Scheduler 1 AIB Ready.**
0	**Scheduler 0 AIB Ready.**

3.4.2.14 Enable Port Processor

Symbol: EEN

Address Offset: 00038h
Default Value: 00000000h
Access: Read/Write

Indicates that the Port Processor is enabled (1) or disabled (0).

Bits	Description
31:1	Reserved
0	**This bit indicates that the port processor is enabled (1) or disabled (0)**. The input port processor drops all cells (including Control Packets) from the linecard or crossbar and ignores all scheduler control information if this bit is 0.

3.4.2.15 AIB Enable Transmitter

Symbol: EAIBEN

Address Offset: 0003Ch
Default Value: 00000000h
Access: Read/Write

Used to enable the transmitter of the corresponding AIB link. If the corresponding central Scheduler is not physically present in the system, this bit should be set to zero to reduce unwanted electrical noise and to minimize unwanted effects when either Scheduler board is inserted.

Bits	Description
31:6	Reserved
5	**Flow Control Crossbar A1 AIB Enable Tx.**
4	**Flow Control Crossbar A0 AIB Enable Tx.**
3	**Flow Control Crossbar B1 AIB Enable Tx.**
2	**Flow Control Crossbar B0 AIB Enable Tx.**
1	**Scheduler 1 AIB Enable Tx.**
0	**Scheduler 0 AIB Enable Tx.**

3.4.2.16 LCS Started/Stopped Status/Control

Symbol: ELCSSSS

Address Offset: 00040h
Default Value: 00000000h
Access: Read/Write

This register resets to 0 (all subports stopped). Traffic must be started by Processed Control Packet or OOB after reset.

Bits	Description
31:8	Reserved
7	**Subport 3 egress MC queues are “lossy” if LCS Stopped**
6	**Subport 2 egress MC queues are “lossy” if LCS Stopped**
5	**Subport 1 egress MC queues are “lossy” if LCS Stopped**
4	**Subport 0 egress MC queues are “lossy” if LCS Stopped**
3	**Subport 3 is started if 1, stopped if 0**
2	**Subport 2 is started if 1, stopped if 0**
1	**Subport 1 is started if 1, stopped if 0**

Bits	Description (Continued)
0	**Subport 0 is started if 1, stopped if 0**

NOTE: The “lossy” behavior is defined in the “LCS Protocol Specification”.

3.4.2.17 Lost Grant or Cell Body Debug Info

Symbol: ELGCBD

Address Offset: 00044h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of lost grants or cells.

Bits	Description
31:30	Reserved
29:12	**The first LCS grant label for which a cell was expected but lost.**
11:0	**The total number of cells which have been expected but lost.**

3.4.2.18 Scheduler Grant to Empty Queue Debug Info

Symbol: ESGEQD

Address Offset: 00048h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of grants to an empty queue.

Bits	Description
31:27	Reserved
26:16	**The first QID for which there was a scheduler grant to empty queue**
15:0	**The total number of scheduler grants to empty queues**

3.4.2.19 LCS Header CRC Error Count

Symbol: ELCSECT

Address Offset: 0004Ch
Default Value: 00000000h
Access: Read and Clear

Indicates the number of CRC Errors in Linecard to Switch LCS headers

Bits	Description
31:0	**The total number of detected CRC Errors in Linecard to Switch LCS headers**

3.4.2.20 LCS Processor CP Subport 0 CRC Error Count

Symbol: ELCSP0CT

Address Offset: 00050h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of CRC Errors in processed CP from Subport 0.

Bits	Description
31:0	**The total number of detected CRC Errors in Processed CPs from Subport 0.**

3.4.2.21 LCS Processor CP Subport 1 CRC Error Count

Symbol: ELCSP1EC

Address Offset: 00054h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of CRC Errors in processed CP from Subport1.

Bits	Description
31:0	**The total number of detected CRC Errors in Processed CPs from Subport 1.**

3.4.2.22 LCS Processor CP Subport 2 CRC Error Count

Symbol: ELCSP2EC

Address Offset: 00058h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of CRC Errors in processed CP from Subport 2.

Bits	Description
31:0	**The total number of detected CRC Errors in Processed CPs from Subport 2.**

3.4.2.23 LCS Processor CP Subport 3 CRC Error Count

Symbol: ELCSP3EC

Address Offset: 0005Ch
Default Value: 00000000h
Access: Read and Clear

Indicates the number of CRC Errors in processed CP from Subport 3.

Bits	Description
31:0	**The total number of detected CRC Errors in Processed CPs from Subport 3.**

3.4.2.24 Input Subport 0 Valid Count

Symbol: EIS0VCT

Address Offset: 00060h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells matched with grant sequence numbers for Subport 0.**

3.4.2.25 Input Subport 1 Valid Count

Symbol: EIS1VCT

Address Offset: 00064h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells matched with grant sequence numbers for Subport 1.**

3.4.2.26 Input Subport 2 Valid Count

Symbol: EIS2VCT

Address Offset: 00068h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells matched with grant sequence numbers for Subport 2.**

3.4.2.27 Input Subport 3 Valid Count

Symbol: EIS3VCT

Address Offset: 0006Ch
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells matched with grant sequence numbers for Subport 3.**

3.4.2.28 Output Subport 0 Valid Count

Symbol: EOS0VCT

Address Offset: 00070h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells sent to Subport 0.**

3.4.2.29 Output Subport 1 Valid Count

Symbol: EOS1VCT

Address Offset: 00074h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells sent to Subport 1.**

3.4.2.30 Output Subport 2 Valid Count

Symbol: EOS2VCT

Address Offset: 00078h
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells sent to Subport 2.**

3.4.2.31 Output Subport 3 Valid Count

Symbol: EOS3VCT

Address Offset: 0007Ch
Default Value: 00000000h
Access: Read and Clear

Indicates the number of cells matched with grant sequence numbers

Bits	Description
31:0	**The total number of cells sent to Subport 3.**

3.4.2.32 Subport 0 to OOB FIFO Status

Symbol: ES0OOBS

Address Offset: 00080h
Default Value: 00000000h
Access: Read/Write

Write of any value will dequeue the head of the FIFO.

Bits	Description
31:7	Reserved
6:4	**The offset of the head of the 8-deep FIFO in DS queue memory.**
3:0	**The length (0 to 8) of the FIFO in DS queue memory.**

3.4.2.33 Subport 1 to OOB FIFO Status

Symbol: ES1OOBS

Address Offset: 00084h
Default Value: 00000000h
Access: Read/Write

Write of any value will dequeue the head of the FIFO.

Bits	Description
31:7	Reserved
6:4	**The offset of the head of the 8-deep FIFO in DS queue memory.**
3:0	**The length (0 to 8) of the FIFO in DS queue memory.**

3.4.2.34 Subport 2 to OOB FIFO Status

Symbol: ES2OOBS

Address Offset: 00088h
Default Value: 00000000h
Access: Read/Write

Write of any value will dequeue the head of the FIFO.

Bits	Description
31:7	Reserved
6:4	**The offset of the head of the 8-deep FIFO in DS queue memory.**
3:0	**The length (0 to 8) of the FIFO in DS queue memory.**

3.4.2.35 Subport 3 to OOB FIFO Status

Symbol: ES3OOBS

Address Offset: 0008Ch
Default Value: 00000000h
Access: Read/Write

Write of any value will dequeue the head of the FIFO.

Bits	Description
31:7	Reserved
6:4	**The offset of the head of the 8-deep FIFO in DS queue memory.**
3:0	**The length (0 to 8) of the FIFO in DS queue memory.**

3.4.2.36 Send OOB to Linecard Control Packet

Symbol: ESOBLCP

Address Offset: 00090h
Default Value: 00000000h
Access: Read/Write

Before sending a CPU (OOB to linecard) control packet, the OOB must first write the control packet header and payload data into the appropriate locations in the Dataslice. See section 3.3, Table 28, “EPP Egress Control Packet Data Format and Dataslice OOB Addressing.” The header for customer-specific CPU control packets should be written into the Dataslices as shown, but the payload data is completely up to the customer. To send the CPU control packet which has been written into Dataslice queue memory, the OOB writes the ESOBLCP register with the control packet type select in bits 5:4 (see the bit-breakout), and a linecard fanout in bits 3:0. If the port is connected to 4 subport linecards, then bits 3:0 are a subport-bitmap. If the port is connected to one OC-192-rate linecard, then bit 0 must be set when the OOB wishes to send a CP. When the CP has been sent to the linecard(s) indicated in bits 3:0, bits 3:0 will read back as 0. Since control packets have higher priority than any other traffic type, they will be sent immediately, unless the EPP is programmed to send only idle cells.

Bits	Description
31:6	Reserved
5:4	**Control Packet type select** 0: Generic OOB to Subport Control Packet 2: LCS Stop Control Packet 3: LCS Start Control Packet
3	**Subport 3 fanout for OOB to Subport Control Packet**
2	**Subport 2 fanout for OOB to Subport Control Packet**
1	**Subport 1 fanout for OOB to Subport Control Packet**
0	**Subport 0 fanout for OOB to Subport Control Packet**

3.4.2.37 Suggested TDM Sync Subport Select

Symbol: ESTDMSSS

Address Offset: 00094h
Default Value: 00000000h
Access: Read/Write

Sync Support Select

Bits	Description
31:2	Reserved
1:0	**Subport select.** Subport from which (suggested) TDM Sync Processed Control packets will be accepted and forwarded to the scheduler, if this port is not enabled to internally generate suggested TDM Syncs.

3.4.2.38 Multicast Invalid Fanout Count

Symbol: EMCIFCT

Address Offset: 00098h
Default Value: 00000000h
Access: Read and Clear

Fanout count

Bits	Description
31:30	Reserved
29:28	**First subport from which a multicast cell with all-0 fanout was received.**
27:16	**First tag which yielded the all-0 fanout.**
15:0	**Total number of cells with invalid fanouts that have been received (and dropped).**

3.4.2.39 Internal Delay Matching Adjustments

Symbol: EIDMA

Address Offset: 0009Ch
Default Value: 03A8D66Bh
Access: Read/Write

Delay matching

Bits	Description
31:29	Reserved
28:27	**DS iFIFO Token Mechanism Delay.** Specifies the depth of the token shift register. Default at reset is 0x0. See Appendix C.2.2.1 for details.
26:23	**Flow Control Crossbar oEPP->iEPP Delay.** Specifies the time it takes for a Flow Control frame to reach the iEPP after it sent from any oEPP. The value of this register will not require adjustment unless there is a change in timing in a future revision of the EPP or Crossbar.
22:18	**Unicast Scheduler Request to Waiting Delay.** Specifies the minimum delay for a priority 0 UC scheduler request to be granted. The value of this register will not require adjustment unless there is a change in timing in a future revision of the EPP or Scheduler.
17:12	**TDM Ingress Subport to Scheduler Grant Delay.** Specifies the delay between a valid input TDM slot and the corresponding scheduler TDM grant. The value of this register will not require adjustment unless there is a change in timing in a future revision of the EPP or Scheduler.
11:6	**TDM Egress Fanout Delay.** Specifies the delay between a valid output TDM slot and the arrival of the TDM cell. The value of this register will not require adjustment unless there is a change in timing in a future revision of the EPP, DS, Crossbar or Scheduler.
5:0	**LCS Unicast Grant to Scheduler Request Delay.** The value of this register will not require adjustment, provided that the customer can meet the RTT constraint given in Section 1.1 “Preventing Underflow of the VOQ” on page 327.

3.4.2.40 Output Backpressure/Unbackpressure Threshold

Symbol: EOPBPTHR

Address Offset: 000A0h
Default Value: 00001C46h
Access: Read/Write

These thresholds are for Multicast queues only.

Bits	Description
31:15	Reserved
14:8	**Output Unbackpressure Threshold.** This specifies when the output port processor de-asserts backpressure. When the Output Scheduler grants to an output queue, the output port processor checks if the number of cells in that queue is equal to the threshold value. If the two values are equal, then an unbackpressure signal is sent to the central scheduler. If the EPP is in OC-48c mode, then this value should be changed to 1Fh.
7	Reserved
6:0	**Output Backpressure Threshold.** This specifies when the output port processor asserts backpressure. When a cell from crossbar enters an output queue, the output port processor checks if the number of cells already in that queue is equal to the threshold value. If the two values are equal, then a backpressure signal is sent to the central scheduler.

3.4.2.41 Egress OC-48/OC-192c Mode Map

Symbol: EEMM

Address Offset: 000A4h
Default Value: 00000000h
Access: Read/Write

Chooses OC-48c/OC-192c Mode.

Bits	Description
31:0	**Port Mode select**. Bits 0:31correspond to individual ports. If the bit is set to 1, the port is OC-48; if set to 0, the port is OC-192c.

3.4.2.42 TDM Offset Value

Symbol: ETDMO

Address Offset: 000A8h
Default Value: 00000080h
Access: Read/Write

Determines TDM offset value.

Bits	Description
31:8	Reserved
7:0	**Specifies TDM Offset Value.** The number of cell times from the arrival of TDM sync from the central scheduler to the recognition of the first time slot in the TDM table. Note that this value must be identical in all ports participating in TDM traffic.

3.4.2.43 Freeze Control

Symbol: EFCTL

Address Offset: 000ACh
Default Value: 00000000h
Access: Read/Write

Freezes TDM grants, requests and cells.

Bits	Description
31:6	Reserved
5	**Freeze TDM grants from LCS Grant Manager.** If this bit is 1, the LCS Grant Manager sends no TDM grants to the linecard.
4	**Freeze TDM Scheduler Requests.** If this bit is 1, TDM input requests are not sent to the Scheduler.
3	**Freeze TDM cells from Output Scheduler.** If this bit is 1, the Output Scheduler does not schedule any TDM cells onto the output line.
2	**Freeze MC/UC grants from LCS Grant Manager.** If this bit is 1, the LCS Grant manager sends no MC or UC grants to the linecard.
1	**Freeze Scheduler Request Modulator.** If this bit is 1, the SRM sends no MC or UC requests to the scheduler.
0	**Freeze MC/UC cells from Output Scheduler.** If this bit is 1, the Output Scheduler does not schedule any MC or UC cells onto the output line.

3.4.2.44 Freeze Unicast Output Scheduler Flows

Symbol: EFUOS

Address Offset: 000B0h
Default Value: 00000000h
Access: Read/Write

Determines whether or not the Output Scheduler sends unicast cells from specific ports.

Bits	Description
31:0	**Bits 31:0 correspond to specific ports.** If the bit for the port is set to 1, the Output Scheduler sends no unicast cells from that port.

3.4.2.45 Multicast Priority Drop

Symbol: EMPD

Address Offset: 000B4h
Default Value: 00000000h
Access: Read/Write

This specifies if a given priority should allow multicast cells to be dropped.

Bits	Description
31:4	Reserved
3:0	**This specifies if a given priority should allow multicast cells to be dropped.** If a bit is 1, then the output port processor does not assert backpressure for the corresponding multicast queue. If a bit is 0, then the output port processor prevents dropping of multicast cells by asserting backpressure to the central scheduler. Bit 3 is for priority 3, bit 2 is for priority 2, bit 1 is for priority 1, and bit 0 is for priority 0.

3.4.2.46 Fault Tolerance Status/Control

Symbol: EFTCTRL

Address Offset: 000B8h
Default Value: 00000000h
Access: Read/Write

Fault Tolerence Status/Control Register.

Bits	Description
31:5	Reserved
4	**Primary Flow Control Crossbar B Select.** This bit specifies which Flow Control Crossbar B is primary. If this bit is 0, then Flow Control Crossbar B 0 is primary. If this bit is 1, then Flow Control Crossbar B 1 is primary. This bit is both a status and control bit. When the primary Flow Control Crossbar B goes down, the EPP must switch over to the secondary Flow Control Crossbar B (causes this bit to flip). OOB can also change the configuration.
3	**Primary Flow Control Crossbar A Select.** This bit specifies which Flow Control Crossbar A is primary. If this bit is 0, then Flow Control Crossbar A 0 is primary. If this bit is 1, then Flow Control Crossbar A 1 is primary. This bit is both a status and control bit. When the primary Flow Control Crossbar A goes down, the EPP must switch over to the secondary Flow Control Crossbar A (causes this bit to flip). OOB can also change the configuration.
2	**Ignore Central Scheduler Grants.** This bit indicates that the EPP is ignoring grants from both primary and secondary central schedulers. The EPP asserts this bit if both schedulers have transmitted CRC errors in the same frame, if the only operational scheduler has transmitted a CRC error, or if the only operational scheduler's AIB link dropped ready. Note that the OOB must write a 0 to this bit to clear it.
1	**Freeze Scheduler Request Modulator (Multicast & Unicast only).** This bit indicates that the SRM is frozen (TDM traffic can still flow) due to fault tolerance errors. The EPP asserts this bit if both schedulers have transmitted CRC errors in the same frame, if the only operational scheduler has transmitted a CRC error, or if the only operational scheduler's AIB link dropped ready. Note that the OOB must write a 0 to this bit to clear it.
0	**Primary Scheduler Select.** This bit specifies which scheduler is primary. If this bit is 0, then scheduler 0 is primary. If this bit is 1, then scheduler 1 is primary. This value must be consistent on all the port processors. This bit is both a status and control bit. When the primary scheduler goes down, all EPPs must switch over to the secondary scheduler (causes this bit to flip). The OOB processor can also change the configuration.

3.4.2.47 Refresh Schedulers Command

Symbol: ERFRSSC

Address Offset: 000BCh
Default Value: 00000000h
Access: Write Only

.Causes the EPP to start the refresh operation.

Bits	Description
31:0	**Start Refresh Schedulers Operation.** A write to this address (any data) causes the EPP to start the scheduler refresh operation. The EPP freezes the Scheduler Request Modulator, freezes the Output Scheduler, and waits for in-flight cells to arrive before transferring state information to the scheduler(s). Note that OOB must must first disable the scheduler's non-TDM traffic register for this port before issuing this command.

3.4.2.48 Go Schedulers Command

Symbol: ESTRTSC

Address Offset: 000C0h
Default Value: 00000000h
Access: Write Only

A write to this address causes an EPP to end the refresh operation.

Bits	Description
31:0	**Synchronously Start Schedulers Operation.** A write to this address (any data) causes the EPP to end the refresh operation. Since the primary and secondary scheduler must transmit the same control in lock step, this command is used to synchronize the two schedulers. Note that OOB must first enable the scheduler's non-TDM traffic register for this port before issuing this command.

3.4.2.49 Idle Count Value

Symbol: EIDLCT

Address Offset: 000C4h
Default Value: 00000000h
Access: Read/Write

This register is used to compensate for the clock frequency variation between the ETT1 switch and the linecard.

Bits	Description
31:0	**Idle Count Value.** This represents the frequency of sending idle frames to the line card. This register is used to compensate for the clock frequency variation between the ETT1 switch and the linecard. If this register is set to 0x0, only idle frames are transmitted to the linecard. If this register is set to 0xFFFFFFFF, no idle frames are transmitted. If this register is set to N, an idle frame is sent every N + 1 frame times. Note that this register resets to 0x0.

3.4.2.50 Enable TDM Sync Generation

Symbol: ETDMEN

Address Offset: 000C8h
Default Value: 00000000h
Access: Read/Write

Indicates if the TDM sync generation is enabled (1) or disabled (0).

Bits	Description
31:1	Reserved.
0	**Enable TDM Sync Generation.** This bit indicates if the TDM sync generation is enabled (1) or disabled (0). If the linecard chooses not to provide the TDM sync signal, any one of the EPPs can be programmed to generate this signal.

3.4.2.51 TDM Sync Generation Control

Symbol: ETDMCTRL

Address Offset: 000CCh
Default Value: 00000000h
Access: Read/Write

TDM Sync Generation Control register.

Bits	Description
31:13	Reserved.
12	**TDM Frame Select.** This bit specifies which TDM frame table to select.
11:0	**TDM Sync Frequency.** This represents the frequency of the TDM sync pulse. If this register is set to N, a TDM sync signal is sent to the central Scheduler every N + 1 frame times.

3.4.2.52 No Grant Count

Symbol: ENOGCT

Address Offset: 000DOh
Default Value: FFFFFFFFh
Access: Read/Write

No Grant Count register.

Bits	Description
31:0	**Controls the frequency of sending cells without grants to the line card.** This register is used to guarantee that the linecard gets a gap in the stream of grants often enough so that it can send idle cells without holding up cell payloads. If this register is set to 0x0, no grants are sent to the linecard. If this register is set to 0xFFFFFFFF, no grants are held up. If this register is set to N, a cell without a grant is sent every N+1 *non-idle* frame times. Note that this register resets to 0xFFFFFFFF. In OC-48 mode, (N+1) should be odd so that no-grant celltimes are distributed equally to all four linecards, so the value written to this register (N) should be even.

3.4.2.53 End of Sch Refresh/Sch OC-48 Sync ModCount

Symbol: EESR48SD

Address Offset: 000D4h
Default Value: 00000000h
Access: Write Only

Specifies the value of an internal OC-48 Sync counter at which the OOB Go Schedulers command should trigger the end of Scheduler refresh.

Bits	Description
31:5	Reserved.
4:0	**Prevents OC-48 Sync phase change after Scheduler refresh.** The value of this register will not require adjustment unless there is a change in timing in a future revision of the EPP or Scheduler.

3.4.2.54 PLL Status/Control

Symbol: EPLLREG

Address Offset: 00100h
Default Value: 0001447Ch
Access: Read/Write

For more details, refer to the IBM SA-12E databook.

Bits	Description
31:21	Reserved.
30:20	**PLL_OBSERVE (read only).**
19:17	Reserved
16	**PLL_RESET.** When set, VCO is held at minimum frequency and PLL is bypassed. PLL reset takes 10mS to complete.
15:10	**PLL_TUNE.**
9:6	**PLL_MULT.**
5:3	**PLL_RANGE_B.**
2:0	**PLL_RANGE_A.**

3.4.2.55 TDM Frame Table 0 Memory

Symbol: ETDMFT0M

Address Offset: 02000-02FFCh
Default Value: 000000000h
Access: Read/Write

Each entry represents a time slot in the TDM table.

Bits	Description
31:23	Reserved.
22	**Input Valid.** This bit is 1 if the input side TDM entry is valid.
21:20	**Input Subport.** (OC-48 only) Input linecard which provides the TDM cell to send.
19:10	**Input Tag .** The tag that the head of the input TDM tag FIFO must match.
9	**Output Valid.** This bit is 1 if the output side TDM entry is valid.
8:4	**Input Port.** The input port from which the output expects to receive a TDM cell.
3:0	**Egress Subport Fanout.** Specifies which linecard(s) will receive the TDM cell.

3.4.2.56 TDM Frame Table 1 Memory

Symbol: ETDMFT1M

Address Offset: 03000-03FFCh
Default Value: 000000000h
Access: Read/Write

Each entry represents a time slot in the TDM table.

Bits	Description
31:23	Reserved.
22	**Input Valid.** This bit is 1 if the input side TDM entry is valid.
21:20	**Input Subport.** (OC-48 only) Input linecard which provides the TDM cell to send.
19:10	**Input Tag.** The tag that the head of the input TDM tag FIFO must match.
9	**Output Valid.** This bit is 1 if the output side TDM entry is valid.
8:4	**Input Port.** The input port from which the output expects to receive a TDM cell.
3:0	**Egress Subport Fanout.** Specifies which linecard(s) will receive the TDM cell.

3.4.2.57 ITIB Memory

Symbol: EITIBM

Address Offset: 04000-07FFCh
Default Value: 000000000h
Access: Read/Write

ITIB memory.

Bits	Description
31:0	**ITIB Memory.** Each entry is a multicast port fanout value, addressed by a12-bit multicast tag. Note that an all-zero fanout is considered invalid.

3.4.2.58 Linecard Request Count Memory

Symbol: ESRCTM

Address Offset: 08000-0FFFCh
Default Value: 000000000h
Access: Read/Write

This memory is addressed by {source_subport, QID}. Each traffic type has a different address and data format.

For Unicast Traffic: Each request count is 10 bits wide if OC-192c (9:0), 8 bits wide if OC-48(7:0).

Address

Bits	Description
19:15	**Must be set to 0x1**
14:13	**Source Subport**
12:11	**Must be set to 0x2 for Unicast Traffic type**
10:9	**Priority**
8:4	**Destination Port**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:10	Reserved.
9:0	**Request Count**

For Multicast traffic: Each request count is 10 bits wide if OC-192c, 8 bits wide if OC-48. Each request count corresponds to a FIFO in Linecard Multicast Tag FIFO Memory; if the request count is changed by OOB, the FIFO length also changes. OOB should specify a new head address for the FIFO in the upper data bits. OOB should also write the correct number of tags into the Linecard Multicast Tag FIFO memory so that the state of the request counter and its tag FIFO are consistent**.**

Address

Bits	Description
19:15	**Must be set to 0x1**
14:13	**Source Subport**
12:11	**Must be set to 0x3 for Multicast Traffic type**
10:9	**Priority**
8:0	**Must be set to 0**

Data

Bits	Description
31:20	Reserved.
19:10	**Linecard Multicast Tag Head**
9:0	**Request Count**

For TDM traffic: Each request count is 10 bits wide if OC-192c, 8 bits wide if OC-48. Each request count corresponds to a FIFO in Linecard TDM Tag FIFO Memory; if the request count is changed by OOB, the FIFO length also changes. OOB should specify a new head address for the FIFO in the upper data bits. OOB should also write the correct number of tags into the Linecard TDM Tag FIFO memory so that the state of the request counter and its tag FIFO are consistent.

Address

Bits	Description
19:15	**Must be set to 0x1**
14:13	**Source Subport**
12:9	**Must be set to 0x7 for TDM Traffic type**
8:4	**Must be set to 0**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:20	Reserved.
19:10	**Linecard TDM Tag Head**
9:0	**Request Count**

For Control Packet: Each request count is 3 bits wide. Each request count corresponds to a 7-deep, 1-bit-wide FIFO. Whenever OOB changes a request count value, it should also supply any tags associated with requests in the upper data bits. The LSB of the “Control Packet Tag FIFO” field is the head of the tag FIFO, i.e. the next tag that will be sent with a grant.

Address

Bits	Description
19:15	**Must be set to 0x1**
14:13	**Source Subport**
12:9	**Must be set to 0x0 for Control Packet Traffic type**
8:4	**Must be set to 0**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:10	Reserved.
9:3	**Control Packet Tag Head**
2:0	**Request Count**

3.4.2.59 Input Queue Information Memory

Symbol: EIQIM

Address Offset: 16000-17FFCh
Default Value: 000000000h
Access: Read/Write

This memory is addressed by QID. This memory contains input queue information for 4 TDM, 4 multicast, 128 unicast, and 4 control packet queues.

For Unicast Traffic: 128-512 queues depending on Egress OC48 Mode Map register; each OC-192 queue holds 0-64 cells, each OC-48 queue holds 0-16 cells.

Address

Bits	Description
19:13	**Must be set to 0xB**
12:11	**Must be set to 0x2 for Unicast Traffic type**
10:9	**Priority**
8:4	**Destination Port**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:13	Reserved.
12:7	**Head**
6:0	**Length**

For Multicast traffic: 4 queues regardless of OC-48 modes; each queue holds 0-96 cells.

Address

Bits	Description
19:13	**Must be set to 0xB**
12:11	**Must be set to 0x3 for Multicast Traffic type**
10:9	**Priority**
8:0	**Must be set to 0**

Data

Bits	Description
31:15	Reserved.
14	**Full**
13:7	**Head**
6:0	**Tail**

For TDM traffic: 1 or 4 queues depending on this EPP's OC-48 mode. If OC-192, up to 96 cells; if OC-48, up to 24 cells each queue.

Address

Bits	Description
19:13	**Must be set to 0xB**
12:9	**Must be set to 0x7 for TDM Traffic type**
8:4	**Must be set to 0**
3:2	**Source Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:15	Reserved.
14	**Full**
13:7	**Head**
6:0	**Tail**

For Control Packet: 1 or 4 queues depending on this EPP's OC-48 mode. 8 cells per queue regardless of mode

Address

Bits	Description
19:13	**Must be set to 0xB**
12:9	**Must be set to 0x0 for Control Packet Traffic type**
8:4	**Must be set to 0**
3:2	**Source Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:7	Reserved.
6:4	**Head**
3:0	**Length**

3.4.2.60 Outstanding Scheduler Request Count Memory

Symbol: EOSRM

Address Offset: 1A000-1BFFCh
Default Value: Unknown
Access: Read/Write

The Outstanding Scheduler Request Count Memory contains the number of cells in a VOQ that the EPP must request a grant from the Scheduler. This memory is addressed by QID, but only Unicast and Multicast addresses are valid.

For Unicast traffic: 128-512 queues depending on Egress OC-48c Mode Map register; maximum 64 requests for OC-192c, max 16 requests for OC-48.

Address

Bits	Description
19:13	**Must be set to 0xD**
12:11	**Must be set to 0x2 for Unicast traffic type**
10:9	**Priority**
8:4	**Destination Port**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:7	Reserved.
6:0	**Requests**

For Multicast traffic: 4 queues regardless of OC-48 modes; max 96 requests.

Address

Bits	Description
19:13	**Must be set to 0xD**
12:11	**Must be set to 0x3 for Multicast traffic type**
10:9	**Priority**
8:0	**Must be set to 0**

Data

Bits	Description
31:7	Reserved.
6:0	**Requests**

3.4.2.61 Waiting Scheduler Request Count Memory

Symbol: EWSRCT

Address Offset: 1C000-1DFFCh
Default Value: Unknown
Access: Read/Write

The Waiting Scheduler request Count Memory contains the number of requests that the EPP is waiting to get a grant back from the Scheduler. This memory is addressed by QID, but only Unicast and Multicast addresses are valid. After resetting an EPP, write 00000000h to address offsets 1D000h through 1D7FCh in that EPP (a total of 512 OOB writes). Whan all EPPs are reset simultaneously (using a broadcast OOB write to the EPP Reset and Control register), 512 broadcast OOB writes can be used to reset those locations in all EPPs.

For Unicast traffic: 128-512 queues depending on Egress OC-48c Mode Map register; maximum 64 requests for OC-192c, max 16 requests for OC-48.

Address

Bits	Description
19:13	**Must be set to 0xE**
12:11	**Must be set to 0x2 for Unicast traffic type**
10:9	**Priority**
8:4	**Destination Port**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:7	Reserved.
6:0	**Requests**

For Multicast traffic: 4 queues regardless of OC-48 modes; max 96 requests.

Address

Bits	Description
19:13	**Must be set to 0xE**
12:11	**Must be set to 0x3 for Multicast traffic type**
10:9	**Priority**
8:0	**Must be set to 0**

Data

Bits	Description
31:7	Reserved.
6:0	**Requests**

3.4.2.62 Output UC Queue Debit Count Memory

Symbol: EOUCQDCT

Address Offset: 1E000-1E7FCh
Default Value: 000000000h
Access: Read/Write

This memory contains debits for UC only.

Address

Bits	Description
19:13	**Must be set to 0xF**
12:11	**Must be set to 0x0 for Unicast traffic type**
10:9	**Priority**
8:4	**Destination Port**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:7	Reserved.
6:0	**Debits.** 128-512 queues depending on Egress OC-48c Mode Map register; maximum 64 debits for OC-192c, max 16 debits for OC-48.

3.4.2.63 Output UC Queue Information Memory

Symbol: EOUCQI

Address Offset: 20000-21FFCh
Default Value: 000000000h
Access: Read/Write

This memory is addressed by QID, but only UC qid addresses are valid. 128-512 queues depending on Egress OC-48c Mode Map register; each OC-192c queue holds 0-64 cells, each OC-48 queue holds 0-16 cells.

Address

Bits	Description
19:13	**Must be set to 0x10**
12:11	**Must be set to 0x2 for Unicast traffic type**
10:9	**Priority**
8:4	**Destination Port**
3:2	**Destination Subport**
1:0	**Must be set to 0**

Data

Bits	Description
31:13	Reserved.
12:7	**Head**
6:0	**Length**

3.4.2.64 Output MC Queue Drop Counters

Symbol: EOMCQDCT

Address Offset: 3E000-3FFFCh
Default Value: 000000000h
Access: Read/Write

This set of registers is addressed by QID, but only MC addresses are valid. Each register contains the number of MC cells dropped by the output EPP for this QID. Note that this value will stick at 0xFFFFFFFF. There is only one register for each MC output queue (total of 4 registers, regardless of OC-48 mode).

Address

Bits	Description
19:13	**Must be set to 0x1F**
12:11	**Must be set to 0x3 for Multicast traffic type**
10:9	**Priority**
8:2	**Must be set to 0**
1:0	**Must be set to 0**

Data

Bits	Description
31:0	**Drop Count.**

3.4.2.65 OTIB Memory

Symbol: EOTIBM

Address Offset: 08000-FFFFCh
Default Value: 000000000h
Access: Read/Write

OTIB memory. Each entry is a multicast egress subport fanout value, addressed by {multicast_tag, source_port}. Note that if the egress port is OC-48, an all-0 fanout will cause the cell to be dropped. If the egress port is OC-192c, this memory is not used.

Bits	Description
31:4	Reserved.
3:0	**Fanout**

3.5 ENHANCED PORT PROCESSOR SIGNAL DESCRIPTIONS

This section describes the Enhanced Port Processor signals.

The following notation is used to describe the signal type:

I Input pin

O Output pin

B Bidirectional Input/Output pin

The signal description also includes the type of buffer used for the particular signal:

PECL PECL are Pseudo-ECL (positive voltage ECL) compatible signals.

STI STI is a very high-speed asynchronous communications protocol used to connect devices that may be 1 to 15 meters apart.

HSTL All HSTL specifications are consistent with EIA/JEDEC Standard, EIA/JESD8-6 High Speed Transceivers Logic (HSTL): A 1.5V Output Buffer Supply Voltage based Interface Standard for Digital Integrated Circuits", dated 8/95. Refer to these specifications for more information**.**

CMOS The CMOS Buffers are either 3.3 V compatible or 2.5 V Low Voltage TTL compatible signals, as noted.

Table 25. Enhanced Port Processor Signal Descriptions

Name	I/O	Type	Description
OOB Interface			
oob_ad[7:0]	B	CMOS	OOB Address bus
oob_clk	I	CMOS	OOB Clock
oob_int_hi	O	CMOS	OOB Interrupt
oob_int_lo	O	CMOS	OOB Interrupt
oob_valid_L	I	CMOS	OOB Control
oob_wait_L	O	CMOS	OOB Control
oob_devsel0	I	CMOS	OOB Device Select
oob_devsel1	I	CMOS	OOB Device Select
oob_devsel2	I	CMOS	OOB Device Select

Table 25. Enhanced Port Processor Signal Descriptions (Continued)

Name	I/O	Type	Description
oob_devsel3	I	CMOS	OOB Device Select
EPP/DS Interface			
d2p_d0_id[7:0]	I	HSTL Class 1	DS to EPP ingress data unmodified
d2p_d0_od[7:0]	I	HSTL Class 1	DS to EPP egress data
d2p_d1_id[7:0]	I	HSTL Class 1	DS to EPP ingress data unmodified
d2p_d1_od[7:0]	I	HSTL Class 1	DS to EPP egress data
d2p_d2_id[7:0]	I	HSTL Class 1	DS to EPP ingress data unmodified
p2d_d[6:0]_ic[7:0]	O	HSTL Class 1	EPP to DS ingress control
p2d_d[6:0]_oc[7:0]	O	HSTL Class 1	EPP to DS egress control
p2d_d0_id[7:0]	O	HSTL Class 1	EPP to DS ingress data modified
p2d_d1_id[7:0]	O	HSTL Class 1	EPP to DS ingress data modified
Vref	I	HSTL	Input Reference Voltage
AIB Interface			
p2s_s0_[c cn, e, en, o, on]	O	STI	EPP to Sched AIB
p2s_s1_[c cn, e, en, o, on]	O	STI	EPP to Sched AIB
s2p_s0_[c cn, e, en, o, on]	I	STI	Sched to EPP AIB
s2p_s1_[c cn, e, en, o, on]	I	STI	Sched to EPP AIB
f2p_a0_[c cn, e, en, o, on]	I	STI	Fc Crossbar to EPP AIB

Table 25. Enhanced Port Processor Signal Descriptions (Continued)

Name	I/O	Type	Description
f2p_a1_[c cn, e, en, o, on]	I	STI	Fc Crossbar to EPP AIB
f2p_b0_[c cn, e, en, o, on]	I	STI	Fc Crossbar to EPP AIB
f2p_b1_[c cn, e, en, o, on]	I	STI	Fc Crossbar to EPP AIB
p2f_a0_[c cn, e, en, o, on]	I	STI	EPP to Fc Crossbar AIB
p2f_a1_[c cn, e, en, o, on]	I	STI	EPP to Fc Crossbar AIB
p2f_b0_[c cn, e, en, o, on]	I	STI	EPP to Fc Crossbar AIB
p2f_b1_[c cn, e, en, o, on]	I	STI	EPP to Fc Crossbar AIB
Power Supply, Clock Source, Reset, and Diagnostics			
soc_out	O	CMOS	Buffered soc_in (NC)
VDDA		Isolated Supply	VDD (2.5 V) for PLL
VDD		Supply	VDD (2.5 V)
VDDQ		Supply	VDDQ (1.6 V)
ref_clk	I	PECL	System 200MHz clock
ref_clkn	I	PECL	System 200MHz clock
soc_in	I	PECL	System Start-of-cell
soc_inn	I	PECL	System Start-of-cell
plllock	O	CMOS	Test output
pwrup_reset_in_L	I	CMOS	Synchronous, Active Low Reset
JTAG Interface			
jtag_tck	I	CMOS	1149.1 JTAG 2.5V ONLY!
jtag_tdi	I	CMOS	1149.1 JTAG 2.5V ONLY!

Table 25. Enhanced Port Processor Signal Descriptions (Continued)

Name	I/O	Type	Description
jtag_tdo	O	CMOS	1149.1 JTAG 2.5V ONLY!
jtag_tms	I	CMOS	1149.1 JTAG 2.5V ONLY!
jtag_trst_L	I	CMOS	1149.1 JTAG 2.5V ONLY!
ASIC Manufacturing Test Interface			
ce0_io	I	CMOS	Test (GND)
ce0_scan	I	CMOS	Test (GND)
ce0_test	I	CMOS	Test (GND)
ce0_tstm3	I	CMOS	Test (GND)
lssd_ce1_a	I	CMOS	Test (VDD)
lssd_ce1_b	I	CMOS	Test (VDD)
lssd_ce1_c1	I	CMOS	Test (VDD)
lssd_ce1_c2	I	CMOS	Test (VDD)
lssd_ce1_c3	I	CMOS	Test (VDD)
lssd_scan_in[19:0]	I	CMOS	Scan input (GND)
lssd_scan_out[19:0]	O	CMOS	Scan output
plltest_in	I	CMOS	Test (GND)
plltest_out	O	CMOS	Test output (NC)
test_di1	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_di2	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_lt	I	CMOS	Test (VDD)
test_re	I	CMOS	Test (GND)
test_ri	I	CMOS	Test (VDD)

3.6 PINOUT AND PACKAGE INFORMATION

3.6.1 Pinout Tables

Table 26. Enhanced Port Processor Pinout (left side)

	01	02	03	04	05	06	07	08
A	No Pin	UNUSED	test_di2	UNUSED	ce0_test	UNUSED	ce0_io	UNUSED
B	UNUSED	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDDQ
C	lssd_scan_out18	UNUSED	lssd_scan_in18	UNUSED	ce0_scan	UNUSED	ce0_tstm3	s2p_s0_o
D	UNUSED	VDDQ	UNUSED	GND	UNUSED	VDDQ	UNUSED	GND
E	lssd_scan_out17	UNUSED	lssd_scan_in17	UNUSED	test_di1	UNUSED	UNUSED	UNUSED
F	UNUSED	GND	d2p_d1_od1	VDD	UNUSED	GND	UNUSED	VDD
G	lssd_scan_out16	d2p_d0_od0	lssd_scan_in16	d2p_d1_od2	UNUSED	UNUSED	UNUSED	UNUSED
H	d2p_d1_od4	VDD	d2p_d0_od5	GND	d2p_d0_od4	VDDQ	d2p_d1_od0	GND
J	d2p_d0_od2	d2p_d1_od3	p2d_d6_oc3	d2p_d0_od6	UNUSED	d2p_d0_od3	d2p_d1_od5	UNUSED
K	d2p_d0_od1	GND	p2d_d6_oc4	VDDQ	p2d_d5_oc6	GND	p2d_d6_oc0	VDDQ
L	d2p_d0_od7	UNUSED	p2d_d5_oc2	p2d_d5_oc1	p2d_d4_oc0	p2d_d4_oc1	p2d_d5_oc5	p2d_d6_oc7
M	p2d_d6_oc5	VDD	p2d_d6_oc6	GND	lssd_scan_in15	VDD	lssd_scan_out15	GND
N	p2d_d3_oc0	p2d_d5_oc3	p2d_d5_oc4	UNUSED	p2d_d5_oc7	p2d_d4_oc5	p2d_d4_oc4	Vref
P	p2d_d3_oc7	VDDQ	p2d_d3_oc6	GND	lssd_scan_out14	VDD	lssd_scan_in14	GND
R	p2d_d2_oc5	p2d_d2_oc4	p2d_d3_oc1	p2d_d3_oc2	p2d_d4_oc3	p2d_d4_oc2	p2d_d4_oc7	p2d_d3_oc5
T	p2d_d1_oc3	GND	UNUSED	VDDQ	p2d_d4_oc6	GND	p2d_d2_oc3	VDDQ
U	p2d_d1_oc2	p2d_d0_oc1	p2d_d2_oc0	p2d_d2_oc6	p2d_d3_oc4	p2d_d2_oc2	p2d_d1_oc0	p2d_d0_oc5
V	p2d_d0_oc0	VDD	p2d_d2_oc7	GND	p2d_d1_oc1	VDDQ	p2d_d0_oc4	GND
W	lssd_scan_in13	p2d_d1_oc4	lssd_scan_out13	p2d_d0_oc2	p2d_d1_oc7	d2p_d2_id2	d2p_d2_id3	d2p_d1_id5
Y	p2d_d1_oc5	GND	p2d_d0_oc3	VDD	d2p_d2_id0	GND	d2p_d1_id4	VDD
AA	lssd_scan_in12	p2d_d0_oc6	lssd_scan_out12	d2p_d2_id1	lssd_scan_out11	d2p_d1_id6	d2p_d0_id0	UNUSED
AB	p2d_d0_oc7	VDDQ	d2p_d2_id4	GND	d2p_d1_id3	VDDQ	d2p_d0_id5	GND
AC	lssd_scan_in11	d2p_d2_id5	lssd_scan_in10	d2p_d1_id2	lssd_scan_in9	d2p_d0_id4	lssd_scan_in8	p2d_d0_id0
AD	d2p_d2_id6	GND	d2p_d1_id1	VDD	d2p_d0_id1	GND	p2d_d1_id3	VDDQ
AE	d2p_d2_id7	d2p_d1_id0	lssd_scan_out10	d2p_d1_id7	lssd_scan_out9	p2d_d1_id2	lssd_scan_out8	d2p_d0_id7

Table 27. Enhanced Port Processor Pinout (center)

	09	10	11	12	13	14	15	16	17
A	UNUSED	UNUSED	UNUSED	UNUSED	s2p_s1_en	UNUSED	UNUSED	p2s_s1_on	p2s_s1_o
B	UNUSED	GND	UNUSED	VDDQ	p2s_s0_c	VDD	UNUSED	GND	f2p_a0_on
C	s2p_s0_c	s2p_s0_cn	UNUSED	UNUSED	p2s_s0_cn	UNUSED	UNUSED	s2p_s1_c	s2p_s1_cn
D	s2p_s0_on	VDD	UNUSED	GND	s2p_s1_e	GND	UNUSED	VDD	p2s_s1_e
E	s2p_s0_e	UNUSED	p2s_s0_e	lssd_scan_ out19	test_lt	test_ri	p2s_s0_on	UNUSED	UNUSED
F	s2p_s0_en	GND	p2s_s0_en	VDDQ	UNUSED	VDDQ	p2s_s0_o	GND	UNUSED
G	UNUSED	UNUSED	UNUSED	lssd_scan_in19	UNUSED	jtag_trst_L	UNUSED	p2s_s1_cn	f2p_a0_en
H	UNUSED	VDDQ	UNUSED	GND	test_re	GND	s2p_s1_on	VDDQ	p2f_a0_c
J	Vref	Vref	UNUSED	UNUSED	s2p_s1_o	UNUSED	p2s_s1_c	p2f_a0_cn	f2p_a0_e
K	d2p_d1_od6	GND	UNUSED	VDD	UNUSED	VDD	UNUSED	GND	oob_ad1
L	p2d_d6_oc1	Vref	VDD	UNUSED	GND	UNUSED	VDD	oob_valid_L	oob_devsel3
M	d2p_d1_od7	VDD	Vref	GND	VDD	GND	oob_devsel0	VDD	UNUSED
N	p2d_d6_oc2	p2d_d5_oc0	GND	VDD	GND	VDD	GND	UNUSED	UNUSED
P	p2d_d3_oc3	VDD	p2d_d2_oc1	GND	VDD	GND	UNUSED	VDD	UNUSED
R	Vref	p2d_d1_oc6	VDD	p2d_d0_id6	GND	p2d_d3_ic6	VDD	Vref	p2d_d1_ic1
T	UNUSED	GND	Vref	VDD	p2d_d4_ic1	VDD	UNUSED	GND	Vref
U	p2d_d1_id1	d2p_d0_id2	p2d_d0_id5	p2d_d5_ic0	p2d_d6_ic4	p2d_d4_ic3	p2d_d3_ic7	p2d_d0_ic1	Vref
V	d2p_d0_id3	VDDQ	p2d_d6_ic3	GND	Vref	GND	p2d_d3_ic0	VDDQ	p2d_d0_ic0
W	p2d_d1_id0	p2d_d0_id4	p2d_d5_ic1	lssd_scan_ out7	p2d_d5_ic3	lssd_scan_in6	p2d_d4_ic2	UNUSED	p2d_d1_ic2
Y	p2d_d1_id7	GND	p2d_d5_ic5	VDDQ	p2d_d5_ic2	VDDQ	p2d_d5_ic6	GND	p2d_d2_ic4
AA	p2d_d1_id6	p2d_d6_ic2	p2d_d5_ic4	lssd_scan_in7	p2d_d4_ic0	lssd_scan_ out6	p2d_d5_ic7	p2d_d3_ic1	p2d_d2_ic5
AB	p2d_d0_id1	VDD	p2d_d6_ic5	GND	UNUSED	GND	p2d_d4_ic7	VDD	p2d_d2_ic2
AC	p2d_d0_id7	UNUSED	p2d_d6_ic6	p2d_d6_ic1	p2d_d4_ic4	p2d_d3_ic2	p2d_d4_ic6	p2d_d3_ic4	p2d_d3_ic5
AD	d2p_d0_id6	GND	p2d_d0_id3	VDDQ	p2d_d4_ic5	VDD	p2d_d2_ic0	GND	p2d_d1_ic5
AE	p2d_d1_id5	p2d_d1_id4	p2d_d0_id2	p2d_d6_ic0	p2d_d6_ic7	p2d_d3_ic3	p2d_d2_ic1	p2d_d2_ic7	p2d_d2_ic6

Table 28. Enhanced Port Processor Pinout (right side)

18	19	20	21	22	23	24	25	
f2p_a0_o	jtag_tms	UNUSED	ref_clkn	pwrup_reset_in_L	jtag_tdo	p2f_a0_on	UNUSED	**A**
VDDQ	UNUSED	GND	UNUSED	VDD	p2f_a0_o	GND	UNUSED	**B**
p2s_s1_en	jtag_tck	f2p_a0_cn	ref_clk	soc_inn	plltest_out	oob_ad7	plltest_in	**C**
GND	f2p_a0_c	VDDQ	soc_in	GND	oob_ad6	VDDQ	UNUSED	**D**
UNUSED	UNUSED	soc_out	jtag_tdi	UNUSED	plllock	UNUSED	VDDA	**E**
VDD	p2f_a0_en	GND	UNUSED	VDD	oob_ad3	GND	f2p_a1_c	**F**
p2f_a0_e	oob_ad5	oob_ad4	oob_ad0	oob_ad2	lssd_ce1_b	f2p_a1_cn	lssd_ce1_a	**G**
GND	f2p_a1_en	VDDQ	p2f_a1_c	GND	p2f_a1_e	VDD	f2p_a1_on	**H**
f2p_a1_e	p2f_a1_cn	p2f_a1_on	UNUSED	p2f_a1_en	f2p_b0_e	f2p_a1_o	oob_wait_L	**J**
VDDQ	p2f_a1_o	GND	f2p_b0_cn	VDDQ	f2p_b0_en	GND	oob_clk	**K**
UNUSED	f2p_b0_c	p2f_b0_cn	p2f_b0_c	f2p_b0_o	f2p_b0_on	oob_devsel1	oob_devsel2	**L**
GND	lssd_ce1_c1	VDD	lssd_ce1_c2	GND	oob_int_lo	VDD	oob_int_hi	**M**
UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	**N**
GND	UNUSED	VDD	lssd_ce1_c3	GND	UNUSED	VDDQ	UNUSED	**P**
UNUSED	p2f_b0_on	p2f_b0_e	p2f_b0_en	f2p_b1_en	f2p_b1_e	UNUSED	UNUSED	**R**
VDDQ	f2p_b1_cn	GND	p2f_b0_o	VDDQ	f2p_b1_o	GND	UNUSED	**T**
Vref	p2f_b1_e	f2p_b1_c	UNUSED	p2f_b1_c	f2p_b1_on	UNUSED	UNUSED	**U**
GND	UNUSED	VDDQ	p2f_b1_en	GND	p2f_b1_cn	VDD	UNUSED	**V**
p2d_d0_ic6	UNUSED	UNUSED	UNUSED	UNUSED	lssd_scan_in0	p2f_b1_o	lssd_scan_out0	**W**
VDD	p2d_d0_ic7	GND	UNUSED	VDD	UNUSED	GND	p2f_b1_on	**Y**
p2d_d1_ic3	p2d_d0_ic3	p2d_d0_ic5	lssd_scan_in2	UNUSED	lssd_scan_in1	UNUSED	lssd_scan_out1	**AA**
GND	p2d_d1_ic6	VDDQ	UNUSED	GND	UNUSED	VDDQ	UNUSED	**AB**
p2d_d2_ic3	lssd_scan_out5	p2d_d1_ic7	lssd_scan_out4	UNUSED	lssd_scan_out3	UNUSED	lssd_scan_out2	**AC**
VDDQ	UNUSED	GND	p2d_d0_ic2	VDD	UNUSED	GND	UNUSED	**AD**
p2d_d1_ic4	lssd_scan_in5	p2d_d1_ic0	lssd_scan_in4	p2d_d0_ic4	lssd_scan_in3	UNUSED	UNUSED	**AE**

Table 35. Enhanced Port Processor Alpha Pin List

Signal Name	Pin
ce0_io	A07
ce0_scan	C05
ce0_test	A05
ce0_tstm3	C07
d2p_d0_id0	AA07
d2p_d0_id1	AD05
d2p_d0_id2	U10
d2p_d0_id3	V09
d2p_d0_id4	AC06
d2p_d0_id5	AB07
d2p_d0_id6	AD09
d2p_d0_id7	AE08
d2p_d0_od0	G02
d2p_d0_od1	K01
d2p_d0_od2	J01
d2p_d0_od3	J06
d2p_d0_od4	H05
d2p_d0_od5	H03
d2p_d0_od6	J04
d2p_d0_od7	L01
d2p_d1_id0	AE02
d2p_d1_id1	AD03
d2p_d1_id2	AC04
d2p_d1_id3	AB05
d2p_d1_id4	Y07
d2p_d1_id5	W08
d2p_d1_id6	AA06
d2p_d1_id7	AE04
d2p_d1_od0	H07
d2p_d1_od1	F03
d2p_d1_od2	G04
d2p_d1_od3	J02
d2p_d1_od4	H01
d2p_d1_od5	J07
d2p_d1_od6	K09
d2p_d1_od7	M09
d2p_d2_id0	Y05
d2p_d2_id1	AA04
d2p_d2_id2	W06
d2p_d2_id3	W07
d2p_d2_id4	AB03
d2p_d2_id5	AC02
d2p_d2_id6	AD01
d2p_d2_id7	AE01
f2p_a0_c	D19
f2p_a0_cn	C20
f2p_a0_e	J17
f2p_a0_en	G17
f2p_a0_o	A18
f2p_a0_on	B17
f2p_a1_c	F25
f2p_a1_cn	G24
f2p_a1_e	J18
f2p_a1_en	H19
f2p_a1_o	J24
f2p_a1_on	H25
f2p_b0_c	L19
f2p_b0_cn	K21
f2p_b0_e	J23
f2p_b0_en	K23
f2p_b0_o	L22
f2p_b0_on	L23
f2p_b1_c	U20
f2p_b1_cn	T19
f2p_b1_e	R23
f2p_b1_en	R22
f2p_b1_o	T23
f2p_b1_on	U23
GND	B02
GND	B06
GND	B10
GND	B16
GND	B20
GND	B24
GND	D04
GND	D08
GND	D12
GND	D14
GND	D18
GND	D22
GND	F02
GND	F06
GND	F10
GND	F16
GND	F20
GND	F24
GND	H04
GND	H08
GND	H12
GND	H14
GND	H18
GND	H22
GND	K02
GND	K06
GND	K10
GND	K16
GND	K20
GND	K24
GND	L13
GND	M04
GND	M08
GND	M12
GND	M14
GND	M18
GND	M22
GND	N11
GND	N13
GND	N15
GND	P04
GND	P08
GND	P12
GND	P14
GND	P18
GND	P22
GND	R13
GND	T02
GND	T06
GND	T10
GND	T16
GND	T20
GND	T24
GND	V04
GND	V08
GND	V12
GND	V14
GND	V18
GND	V22
GND	Y02
GND	Y06
GND	Y10
GND	Y16
GND	Y20
GND	Y24
GND	AB04
GND	AB08
GND	AB12
GND	AB14
GND	AB18
GND	AB22
GND	AD02
GND	AD06
GND	AD10
GND	AD16
GND	AD20
GND	AD24
jtag_tck	C19
jtag_tdi	E21
jtag_tdo	A23
jtag_tms	A19
jtag_trst_L	G14
lssd_ce1_a	G25
lssd_ce1_b	G23
lssd_ce1_c1	M19
lssd_ce1_c2	M21
lssd_ce1_c3	P21
lssd_scan_in0	W23
lssd_scan_in1	AA23
lssd_scan_in10	AC03
lssd_scan_in11	AC01
lssd_scan_in12	AA01
lssd_scan_in13	W01
lssd_scan_in14	P07
lssd_scan_in15	M05
lssd_scan_in16	G03
lssd_scan_in17	E03
lssd_scan_in18	C03
lssd_scan_in19	G12
lssd_scan_in2	AA21
lssd_scan_in3	AE23
lssd_scan_in4	AE21
lssd_scan_in5	AE19
lssd_scan_in6	W14
lssd_scan_in7	AA12
lssd_scan_in8	AC07
lssd_scan_in9	AC05
lssd_scan_out0	W25
lssd_scan_out1	AA25
lssd_scan_out10	AE03
lssd_scan_out11	AA05
lssd_scan_out12	AA03
lssd_scan_out13	W03
lssd_scan_out14	P05
lssd_scan_out15	M07

Signal Name	Pin
lssd_scan_out16	G01
lssd_scan_out17	E01
lssd_scan_out18	C01
lssd_scan_out19	E12
lssd_scan_out2	AC25
lssd_scan_out3	AC23
lssd_scan_out4	AC21
lssd_scan_out5	AC19
lssd_scan_out6	AA14
lssd_scan_out7	W12
lssd_scan_out8	AE07
lssd_scan_out9	AE05
No Pin	A01
oob_ad0	G21
oob_ad1	K17
oob_ad2	G22
oob_ad3	F23
oob_ad4	G20
oob_ad5	G19
oob_ad6	D23
oob_ad7	C24
oob_clk	K25
oob_devsel0	M15
oob_devsel1	L24
oob_devsel2	L25
oob_devsel3	L17
oob_int_hi	M25
oob_int_lo	M23
oob_valid_L	L16
oob_wait_L	J25
p2d_d0_ic0	V17
p2d_d0_ic1	U16
p2d_d0_ic2	AD21
p2d_d0_ic3	AA19
p2d_d0_ic4	AE22
p2d_d0_ic5	AA20
p2d_d0_ic6	W18
p2d_d0_ic7	Y19
p2d_d0_id0	AC08
p2d_d0_id1	AB09
p2d_d0_id2	AE11
p2d_d0_id3	AD11
p2d_d0_id4	W10
p2d_d0_id5	U11
p2d_d0_id6	R12
p2d_d0_id7	AC09
p2d_d0_oc0	V01
p2d_d0_oc1	U02
p2d_d0_oc2	W04
p2d_d0_oc3	Y03
p2d_d0_oc4	V07
p2d_d0_oc5	U08
p2d_d0_oc6	AA02
p2d_d0_oc7	AB01
p2d_d1_ic0	AE20
p2d_d1_ic1	R17
p2d_d1_ic2	W17
p2d_d1_ic3	AA18
p2d_d1_ic4	AE18
p2d_d1_ic5	AD17
p2d_d1_ic6	AB19

Signal Name	Pin
p2d_d1_ic7	AC20
p2d_d1_id0	W09
p2d_d1_id1	U09
p2d_d1_id2	AE06
p2d_d1_id3	AD07
p2d_d1_id4	AE10
p2d_d1_id5	AE09
p2d_d1_id6	AA09
p2d_d1_id7	Y09
p2d_d1_oc0	U07
p2d_d1_oc1	V05
p2d_d1_oc2	U01
p2d_d1_oc3	T01
p2d_d1_oc4	W02
p2d_d1_oc5	Y01
p2d_d1_oc6	R10
p2d_d1_oc7	W05
p2d_d2_ic0	AD15
p2d_d2_ic1	AE15
p2d_d2_ic2	AB17
p2d_d2_ic3	AC18
p2d_d2_ic4	Y17
p2d_d2_ic5	AA17
p2d_d2_ic6	AE17
p2d_d2_ic7	AE16
p2d_d2_oc0	U03
p2d_d2_oc1	P11
p2d_d2_oc2	U06
p2d_d2_oc3	T07
p2d_d2_oc4	R02
p2d_d2_oc5	R01
p2d_d2_oc6	U04
p2d_d2_oc7	V03
p2d_d3_ic0	V15
p2d_d3_ic1	AA16
p2d_d3_ic2	AC14
p2d_d3_ic3	AE14
p2d_d3_ic4	AC16
p2d_d3_ic5	AC17
p2d_d3_ic6	R14
p2d_d3_ic7	U15
p2d_d3_oc0	N01
p2d_d3_oc1	R03
p2d_d3_oc2	R04
p2d_d3_oc3	P09
p2d_d3_oc4	U05
p2d_d3_oc5	R08
p2d_d3_oc6	P03
p2d_d3_oc7	P01
p2d_d4_ic0	AA13
p2d_d4_ic1	T13
p2d_d4_ic2	W15
p2d_d4_ic3	U14
p2d_d4_ic4	AC13
p2d_d4_ic5	AD13
p2d_d4_ic6	AC15
p2d_d4_ic7	AB15
p2d_d4_oc0	L05
p2d_d4_oc1	L06
p2d_d4_oc2	R06
p2d_d4_oc3	R05

Signal Name	Pin
p2d_d4_oc4	N07
p2d_d4_oc5	N06
p2d_d4_oc6	T05
p2d_d4_oc7	R07
p2d_d5_ic0	U12
p2d_d5_ic1	W11
p2d_d5_ic2	Y13
p2d_d5_ic3	W13
p2d_d5_ic4	AA11
p2d_d5_ic5	Y11
p2d_d5_ic6	Y15
p2d_d5_ic7	AA15
p2d_d5_oc0	N10
p2d_d5_oc1	L04
p2d_d5_oc2	L03
p2d_d5_oc3	N02
p2d_d5_oc4	N03
p2d_d5_oc5	L07
p2d_d5_oc6	K05
p2d_d5_oc7	N05
p2d_d6_ic0	AE12
p2d_d6_ic1	AC12
p2d_d6_ic2	AA10
p2d_d6_ic3	V11
p2d_d6_ic4	U13
p2d_d6_ic5	AB11
p2d_d6_ic6	AC11
p2d_d6_ic7	AE13
p2d_d6_oc0	K07
p2d_d6_oc1	L09
p2d_d6_oc2	N09
p2d_d6_oc3	J03
p2d_d6_oc4	K03
p2d_d6_oc5	M01
p2d_d6_oc6	M03
p2d_d6_oc7	L08
p2f_a0_c	H17
p2f_a0_cn	J16
p2f_a0_e	G18
p2f_a0_en	F19
p2f_a0_o	B23
p2f_a0_on	A24
p2f_a1_c	H21
p2f_a1_cn	J19
p2f_a1_e	H23
p2f_a1_en	J22
p2f_a1_o	K19
p2f_a1_on	J20
p2f_b0_c	L21
p2f_b0_cn	L20
p2f_b0_e	R20
p2f_b0_en	R21
p2f_b0_o	T21
p2f_b0_on	R19
p2f_b1_c	U22
p2f_b1_cn	V23
p2f_b1_e	U19
p2f_b1_en	V21
p2f_b1_o	W24
p2f_b1_on	Y25
p2s_s0_c	B13

Signal Name	Pin
p2s_s0_cn	C13
p2s_s0_e	E11
p2s_s0_en	F11
p2s_s0_o	F15
p2s_s0_on	E15
p2s_s1_c	J15
p2s_s1_cn	G16
p2s_s1_e	D17
p2s_s1_en	C18
p2s_s1_o	A17
p2s_s1_on	A16
plllock	E23
plltest_in	C25
plltest_out	C23
pwrup_reset_in_L	A22
ref_clk	C21
ref_clkn	A21
s2p_s0_c	C09
s2p_s0_cn	C10
s2p_s0_e	E09
s2p_s0_en	F09
s2p_s0_o	C08
s2p_s0_on	D09
s2p_s1_c	C16
s2p_s1_cn	C17
s2p_s1_e	D13
s2p_s1_en	A13
s2p_s1_o	J13
s2p_s1_on	H15
soc_in	D21
soc_inn	C22
soc_out	E20
test_di1	E05
test_di2	A03
test_lt	E13
test_re	H13
test_ri	E14
UNUSED	A02
UNUSED	A04
UNUSED	A06
UNUSED	A08
UNUSED	A09
UNUSED	A10
UNUSED	A11
UNUSED	A12
UNUSED	A14
UNUSED	A15
UNUSED	A20
UNUSED	A25
UNUSED	B01
UNUSED	B03
UNUSED	B05
UNUSED	B07
UNUSED	B09
UNUSED	B11
UNUSED	B15
UNUSED	B19
UNUSED	B21
UNUSED	B25
UNUSED	C02
UNUSED	C04

Signal Name	Pin
UNUSED	C06
UNUSED	C11
UNUSED	C12
UNUSED	C14
UNUSED	C15
UNUSED	D01
UNUSED	D03
UNUSED	D05
UNUSED	D07
UNUSED	D11
UNUSED	D15
UNUSED	D25
UNUSED	E02
UNUSED	E04
UNUSED	E06
UNUSED	E07
UNUSED	E08
UNUSED	E10
UNUSED	E16
UNUSED	E17
UNUSED	E18
UNUSED	E19
UNUSED	E22
UNUSED	E24
UNUSED	F01
UNUSED	F05
UNUSED	F07
UNUSED	F13
UNUSED	F17
UNUSED	F21
UNUSED	G05
UNUSED	G06
UNUSED	G07
UNUSED	G08
UNUSED	G09
UNUSED	G10
UNUSED	G11
UNUSED	G13
UNUSED	G15
UNUSED	H09
UNUSED	H11
UNUSED	J05
UNUSED	J08
UNUSED	J11
UNUSED	J12
UNUSED	J14
UNUSED	J21
UNUSED	K11
UNUSED	K13
UNUSED	K15
UNUSED	L02
UNUSED	L12
UNUSED	L14
UNUSED	L18
UNUSED	M17
UNUSED	N04
UNUSED	N16
UNUSED	N17
UNUSED	N18
UNUSED	N19
UNUSED	N20

Signal Name	Pin
UNUSED	N21
UNUSED	N22
UNUSED	N23
UNUSED	N24
UNUSED	N25
UNUSED	P15
UNUSED	P17
UNUSED	P19
UNUSED	P23
UNUSED	P25
UNUSED	R18
UNUSED	R24
UNUSED	R25
UNUSED	T03
UNUSED	T09
UNUSED	T15
UNUSED	T25
UNUSED	U21
UNUSED	U24
UNUSED	U25
UNUSED	V19
UNUSED	V25
UNUSED	W16
UNUSED	W19
UNUSED	W20
UNUSED	W21
UNUSED	W22
UNUSED	Y21
UNUSED	Y23
UNUSED	AA08
UNUSED	AA22
UNUSED	AA24
UNUSED	AB13
UNUSED	AB21
UNUSED	AB23
UNUSED	AB25
UNUSED	AC10
UNUSED	AC22
UNUSED	AC24
UNUSED	AD19
UNUSED	AD23
UNUSED	AD25
UNUSED	AE24
UNUSED	AE25
VDD	B04
VDD	B14
VDD	B22
VDD	D10
VDD	D16
VDD	F04
VDD	F08
VDD	F18
VDD	F22
VDD	H02
VDD	H24
VDD	K12
VDD	K14
VDD	L11
VDD	L15
VDD	M02
VDD	M06

Signal Name	Pin
VDD	M10
VDD	M13
VDD	M16
VDD	M20
VDD	M24
VDD	N12
VDD	N14
VDD	P06
VDD	P10
VDD	P13
VDD	P16
VDD	P20
VDD	R11
VDD	R15
VDD	T12
VDD	T14
VDD	V02
VDD	V24
VDD	Y04
VDD	Y08
VDD	Y18
VDD	Y22
VDD	AB10
VDD	AB16
VDD	AD04
VDD	AD14
VDD	AD22
VDDA	E25
VDDQ	D24
VDDQ	H20
VDDQ	K18
VDDQ	K22
VDDQ	P24
VDDQ	T18
VDDQ	T22
VDDQ	V20
VDDQ	AB24
VDDQ	V10
VDDQ	V16
VDDQ	Y12
VDDQ	Y14
VDDQ	AB06
VDDQ	AB20
VDDQ	AD08
VDDQ	AD12
VDDQ	AD18
VDDQ	D02
VDDQ	H06
VDDQ	K04
VDDQ	K08
VDDQ	P02
VDDQ	T04
VDDQ	T08
VDDQ	V06
VDDQ	AB02
VDDQ	B08
VDDQ	B12
VDDQ	B18
VDDQ	D06
VDDQ	D20
VDDQ	F12

Signal Name	Pin
VDDQ	F14
VDDQ	H10
VDDQ	H16
Vref	R16
Vref	V13
Vref	N08
Vref	L10
Vref	T17
Vref	U18
Vref	U17
Vref	T11
Vref	R09
Vref	M11
Vref	J10
Vref	J09

3.6.2 Package Dimensions

This section outlines the mechanical dimensions for the Enhanced Port Processor (EPP) device. The package is a 624 ceramic ball grid array (CBGA).

NOTE: Drawings are not to scale.

Figure 60. Enhanced Port Processor CBGA Package Dimensions - Top and Side Views

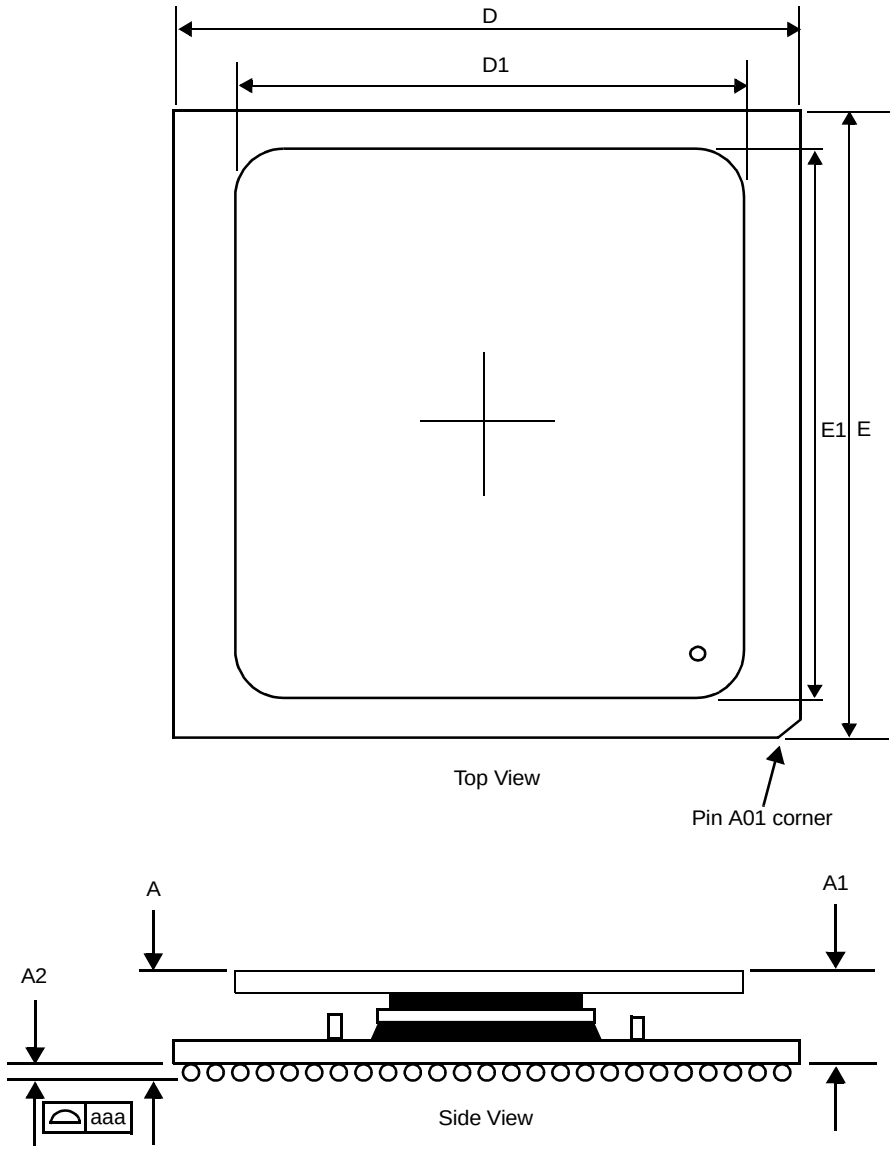


Figure 61. Enhanced Port Processor (EPP) CBGA Package Dimensions - Bottom View

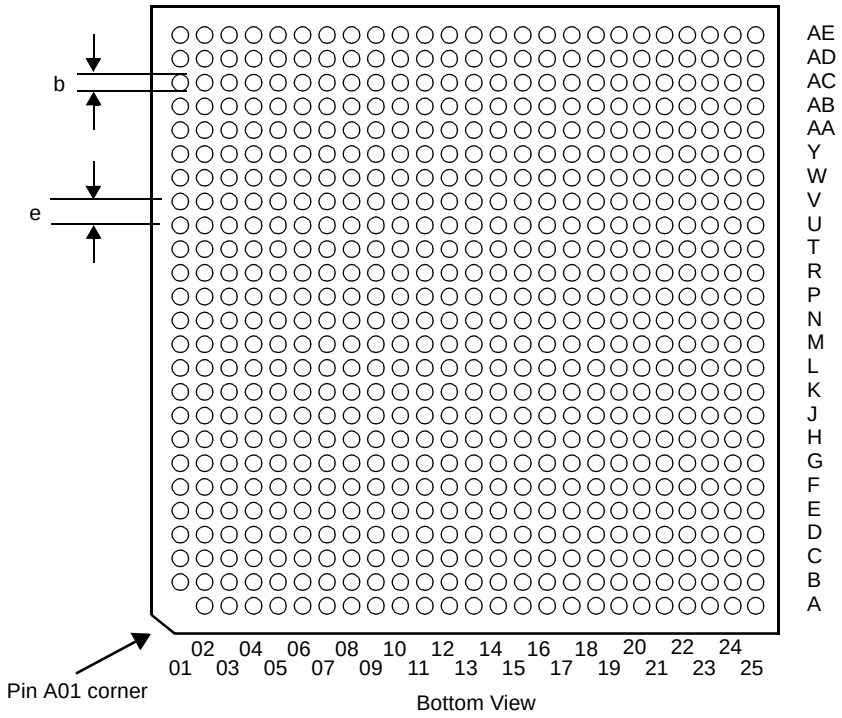


Table 36. Enhanced Port Processor CBGA Mechanical Specifications

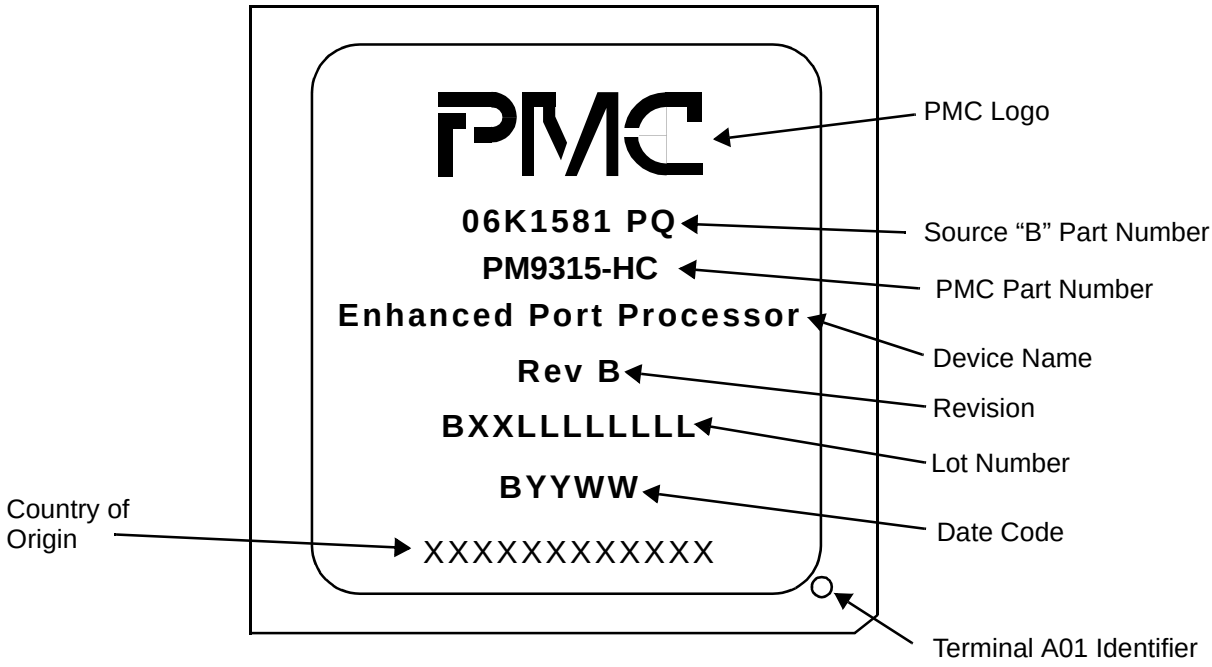
Symbol	e =1.27 mm			Units
	Minimum	Nominal	Maximum	
A		5.52		mm
A1		4.62		mm
A2		0.90		mm
aaa		0.15		mm
D		32.50		mm
D1		30.48		mm
E		32.50		mm
E1		30.48		mm
M		25 x 25		
N		624		
b	0.82		0.92	mm

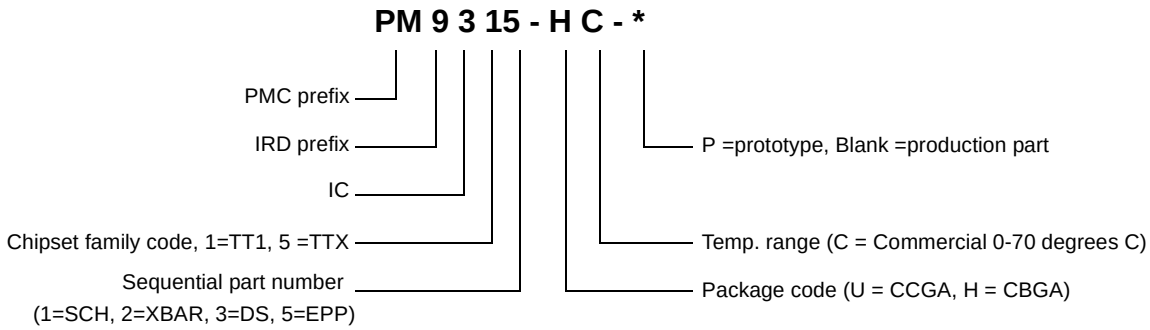
NOTE: Ball alloy is 90/10 PbSn

3.6.3 Part Number

The PMC-Sierra Part Number for the Enhanced Port Processor is:

Enhanced Port Processor	PM9315-HC





4 Crossbar

This chapter contains information on the Crossbar device, part number PM9312-UC, available from PMC-Sierra, Inc.

The Crossbar device is a multicast-capable, 32-port, reverse-routed Crossbar. Each Crossbar is connected to the matching logical Dataslice in every port. For example, XBAR0 connects to DS0. Each Crossbar switches an eight byte cell each frame time.

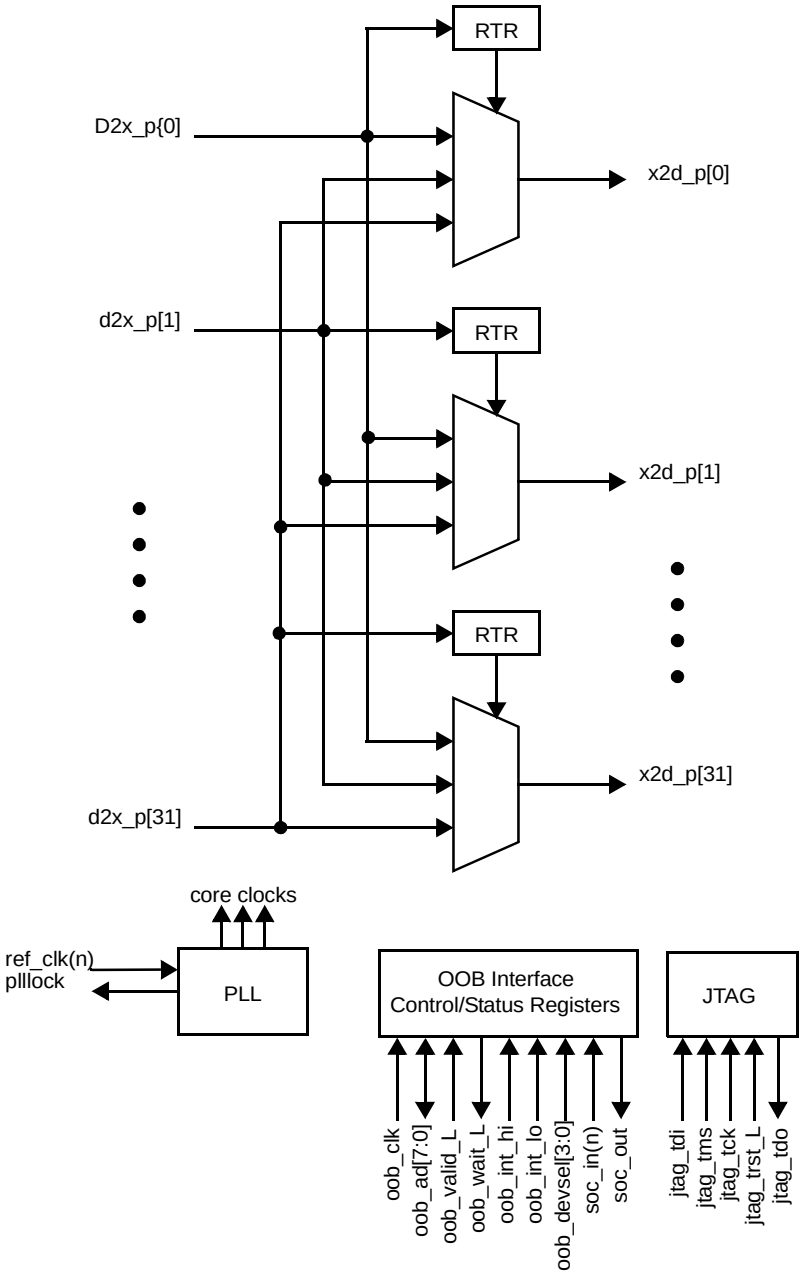
The device sets its crosspoints according to the reverse routing tags supplied by the Scheduler, via the EPPs and Dataslices. If the routing tag supplied is non-valid, the frame containing cell data being sent to this DS had a CRC error, or the frame containing the routing tag from this DS had a CRC error, then the VLD bit is *not* set on the frame to the DS.

4.1 CROSSBAR BLOCKS

Figure 62 shows the basic data flow through the Crossbar. The routing tag sent in the AIB frame from the Dataslice is registered in the routing tag register (RTR). The RTR selects one of the 32 incoming byte-wide busses and the output of the 32:1 byte-wide mux is driven to the transmitter side of the asymmetric serial link dumb end.

The total delay from d2x_p[31:0] to x2d_p[31:0] is four cell times.

Figure 62. The Basic Dataflow Through the Crossbar



4.1.1 OOB Interface and Control/Status Registers

All of the devices have an OOB interface. This interface allows a single local CPU to control and monitor all of the devices within a core fabric. Internally, each device provides registers that can be mapped into the CPU's address space. These registers are described in more detail in Section 4.4 "Crossbar Registers".

4.2 MODES OF OPERATION

The default mode of operation is for the Crossbar to act as a 32-port switch. In this mode, each ETT1 port has one connection to each of the Crossbars. If an 8- or 16-port switch is required, the Crossbar can be configured so that fewer Crossbar devices are required. In these modes, the term superport represents a system port.

In 8-port mode, the Crossbar considers four of its ports to be a single superport. For example, ports 0, 1, 2, and 3 make up superport 0, ports 28, 29, 30, and 31 are superport 7. A single superport is connected to only three Crossbars. For example, superport 0 has its first 4 Dataslices connected to ports 0, 1, 2, and 3 of Crossbar 0, its next 4 Dataslices connected to ports 0, 1, 2, and 3 of Crossbar 1, and its last 4 Dataslices connected to ports 0, 1, 2, and 3 of Crossbar 2.

In 16-port mode, the Crossbar considers two of its ports to be a single superport. Ports 0 & 1 make up superport 0, and ports 30 & 31 make up superport 15 for example. A single superport is connected to six Crossbars. In this case, superport 0 has its first 2 Dataslices connected to ports 0 and 1 of Crossbar 0, and its last Dataslices are connected to ports 0 and 1 of Crossbar 5.

The mode is set via the OOB bus. The Crossbar acts as if the superports make up a 'trunk', so that either two or four of the inputs are routed as a single port.

4.3 OOB ACCESS

The Crossbars can be written and read via the OOB bus. In addition to the device select addresses 0x7FF (all devices) and 0x7B (all Crossbars), each Crossbar can be addressed with an individual device select specified by {000100, CCCC, 000, C} where CCCCC is the Crossbar number in the range 0-31 (these correspond to device selects 0x200 to 0x2F1, not including all addresses within that range). The Registers section of this chapter illustrates the usage of the subaddress field to specify registers within the Crossbar.

4.4 CROSSBAR REGISTERS

The Crossbar device select low-order bits are hard wired on the board by four pins (oobdev_sel[3:0]) on the Crossbar package. See Section 1.7.1.3 “Individual Device Selects” on page 79 for more information.

4.4.1 Summary

The following table is a summary of information for all registers in the Crossbar. See the following Descriptions section for more information on individual registers.

Read and Clear means that reading the register causes it to be cleared (reset to zero)

Table 37. Crossbar Register Summary

Address	Symbol	Register	Access	Default Value
00000h	XSTS	Status	Read Only	40000000h
00004h	XMODERS	Control/Reset	Read/Write	00000004h
00008h	XIRLMSK	Low Priority Mask	Read/Write	00000000h
0000Ch	XIRHMSK	High Priority Mask	Read/Write	00000000h
00010h	XIR	General Interrupt Register	Read only	00000000h
00014h	XCRCMSK	CRC Error Interrupts Mask	Read/Write	00000000h
00018h	XCRC	CRC Error Interrupts	Read and Clear	00000000h
0001Ch	XRDYDMSK	RDY Inactive Interrupts Mask	Read/Write	00000000h
00020h	XRDYDN	Link RDY Inactive Interrupts	Read and Clear	00000000h
00024h	XRDYUMSK	Link RDY Active Interrupts Mask	Read/Write	00000000h
00028h	XRDYUP	Link RDY Active Interrupts	Read and Clear	00000000h
00030h	XAIBRS	AIB Reset	Read/Write	FFFFFFFFh
00034h	XAIBRDY	AIB Ready	Read Only	00000000h
0003Ch	XAIBEN	AIB Tx Enable	Read/Write	00000000h
00100h	XPLL	PLL Control/Status	Read/Write	0001447Ch

4.4.2 Crossbar Register Descriptions

Read and Clear means that reading the register causes it to be cleared (reset to zero).

All bits labeled as Reserved should be set to 0.

4.4.2.1 Status

Symbol: XSTS

Address Offset: 00000h
Default Value: 40000000h
Access: Read Only

The top 8 bits are device_id [3:0] and revision [3:0]. OOB interrupt bits are [1:0] (high, low), all other bits are reserved.

Bits	Description
31:28	**Device ID Number.** Identifies the specific device.
27:24	**Device Revision Number.**
23:2	Reserved.
1	**High Priority Interrupt.** 1 = There is an outstanding high priority interrupt. One of the bits in the General Interrupt Register is set, and is enabled via its corresponding high priority mask.
0	**Low Priority Interrupt.** 1 = There is an outstanding low priority interrupt. One of the bits in the General Interrupt Register is set, and is enabled via its corresponding low priority mask.

4.4.2.2 Control/Reset

Symbol: XMODERS

Address Offset: 00004h
Default Value: 00000004h
Access: Read/Write

This register configures the number of ports supported by this Crossbar. It also resets the entire device, equivalent to a hardware reset.

Bits	Description
31-3	Reserved.

Bits	Description (Continued)
2:1	**Port Mode.** These 2 bits select the number of ports the Crossbar is supporting. A zero value (00) corresponds to 8 ports, a one value (01) corresponds to 16 ports, and a two value (10) corresponds to 32 ports (the default). The number of slices per port are then given by 32/num_ports_selected. 10 = 32 ports x 1 slice 01 = 16 ports x 2 slices 00 = 8 ports x 4 slices
0	**Reset**. Writing a 1 to this location will reset the entire device. It is equivalent to a hardware reset. This register is cleared automatically after the device has been reset. Writing a 0 is not necessary. Soft reset takes 1mS to complete.

4.4.2.3 Low Priority Mask

Symbol: XIRLMSK

Address Offset: 00008h
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:3	Reserved.
2:0	**Low Priority Mask.** Mask bits for low priority interrupts. Each mask bit is set to 1 to enable a low priority interrupt when the corresponding bit in the General Interrupt Register is 1.

4.4.2.4 High Priority Mask

Symbol: XIRHMSK

Address Offset: 0000Ch
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:3	Reserved.
2:0	**High Priority Mask.** Mask bits for high priority interrupts. Each mask bit is set to 1 to enable a high priority interrupt when the corresponding bit in the General Interrupt Register is 1.

4.4.2.5 General Interrupt Register

Symbol: XIR

Address Offset: 00010h
Default Value: 00000000h
Access: Read only

Interrupt Register.

Bits	Description
31:3	Reserved.
2	**Ready Active.** This is the logical OR of the Ready Active Interrupt register, after it has been masked. This is set if a serial link goes from inactive to active and the corresponding mask bit is 1. This bit is not cleared when read. You must clear the Ready Active Interrupt register.
1	**Ready Inactive.** This is the logical OR of the Ready Inactive Interrupt register, after it has been masked. This is set if a serial link goes from active to inactive and the corresponding mask bit is 1. This bit is not cleared when read. You must clear the Ready Inactive Interrupt register.
0	**CRC Error.** This is the logical OR of the CRC register, after it has been masked. This is set if a CRC error occurs and the corresponding mask bit is 1. This bit is not cleared when read. You must clear the CRC Error Interrupts register.

4.4.2.6 CRC Error Interrupts Mask

Symbol: XCRCMSK

Address Offset: 00014h
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for per-port CRC errors

Bits	Description
31:0	**CRC Error Interrupts Mask.** Each bit is used to mask (enable) interrupts when the corresponding bit in the CRC Error Interrupts register is set to 1. The interrupt is enabled when the bit is 1.

4.4.2.7 CRC Error Interrupts

Symbol: XCRC

Address Offset: 00018h
Default Value: 00000000h
Access: Read and Clear

Interrupt Register for per-port CRC errors

Bits	Description
31:0	**CRC Error Interrupts.** Each bit indicates if a CRC error has occurred on the appropriate link (DS to XBAR) since the last time this register was read. Mask off unwanted (unused ports) bits via the CRC Error Interrupt Mask.

4.4.2.8 RDY Inactive Interrupts Mask

Symbol: XRDYDMSK

Address Offset: 0001Ch
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for per-port RDY down signals.

Bits	Description
31:0	**RDY Inactive Interrupts Mask.** Each bit is used to mask (enable) interrupts when the corresponding bit in the Link Ready Inactive register is set to 1. The interrupt is enabled when the bit is 1.

4.4.2.9 Link RDY Inactive Interrupts

Symbol: XRDYDN

Address Offset: 00020h
Default Value: 00000000h
Access: Read and Clear

Interrupt Register for per-port RDY down signals.

NOTE: The link may be up again (active) by the time the CPU reads this register. Mask off unwanted (unused ports) bits via the Link Ready Inactive Interrupts Mask. Reading this register clears all bits.

Bits	Description
31:0	**Link RDY Inactive Interrupts.** Each bit indicates if the Ready signal from a link has transitioned from Active to Inactive (the link has gone down for some reason).

4.4.2.10 Link RDY Active Interrupts Mask

Symbol: XRDYUMSK

Address Offset: 00024h
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for per-port RDY up signals.

Bits	Description
31:0	**Link RDY Active Interrupts Mask.** Each bit is used to mask (enable) interrupts when the corresponding bit in the Link Ready Active register is set to 1. The interrupt is enabled when the bit is 1.

4.4.2.11 Link RDY Active Interrupts

Symbol: XRDYUP

Address Offset: 00028h
Default Value: 00000000h
Access: Read and Clear

Interrupt Register for per-port RDY up signals.

NOTE: The link may be down again (inactive) by the time the CPU reads this register. Mask off unwanted (unused ports) bits via the Link Ready Active Interrupt Mask. This register is cleared to 0 when read.

Bits	Description
31:0	**Link RDY Active Interrupts.** Each bit indicates if the Ready signal from a link has transitioned from Inactive to Active (the link has come up).

4.4.2.12 AIB Reset

Symbol: XAIBRS

Address Offset: 00030h
Default Value: FFFFFFFFh
Access: Read/Write

Resets AIB link for each port.

Bits	Description
31:0	**AIB Reset.** Each bit is used to assert reset to the AIB link for each port. Reset is asserted when the bit is 1. If reset, the link will not operate or transition to ready. This register is set to FFFFFFFFh on power-up reset or if the reset bit in the control register is asserted.

4.4.2.13 AIB Ready

Symbol: XAIBRDY

Address Offset: 00034h
Default Value: 00000000h
Access: Read Only

Indicates the status of the AIB link.

Bits	Description
31:0	**AIB Ready.** Each bit indicates if the corresponding AIB link is ready (1 = active) or not (0 = inactive).

4.4.2.14 AIB Tx Enable

Symbol: XAIBEN

Address Offset: 0003Ch
Default Value: 00000000h
Access: Read/Write

Enables the transmitter of the corresponding AIB link.

Bits	Description
31:0	**AIB Tx Enable.** If the corresponding port is not physically present in the system then this bit should be set to zero to reduce unwanted electrical noise and to minimize unwanted effects when the port board is inserted.

4.4.2.15 PLL Control/Status

Symbol: XPLL

Address Offset: 00100h
Default Value: 0001447Ch
Access: Read/Write

Controls operation of the internal PLL (Phase locked loop). After a power on reset the PLL itself is held in reset. This is reflected in bit 16 of this register. The local CPU must reset this bit to 0 to enable operation of the device, and thus should write 0000447Ch to this register.

Bit	Description
31:17	**PLL status.** These bits reflect internal PLL operation status and should be ignored.
16	**Reset PLL.** When set to 1 the PLL is held reset. The supplied reference clock will be used as the internal clock. The serial links will not be operational. This bit will be 1 after power-up reset and should be deasserted for normal operation. PLL reset takes 10mS to complete.
15:0	**PLL control.**

4.5 CROSSBAR SIGNAL DESCRIPTIONS

This section describes the Crossbar signals.

The following notation is used to describe the signal type:

I Input pin

O Output pin

B Bidirectional Input/Output pin

The signal description also includes the type of buffer used for the particular signal:

PECL PECL are Pseudo-ECL (positive voltage ECL) compatible signals.

STI STI is a very high-speed asynchronous communications protocol used to connect devices that may be 1 to 15 meters apart.

HSTL All HSTL specifications are consistent with EIA/JEDEC Standard, EIA/JESD8-6 “High Speed Transceivers Logic (HSTL): A 1.5V Output Buffer Supply Voltage based Interface Standard for Digital Integrated Circuits”, dated 8/95. Refer to these specifications for more information**.**

CMOS The CMOS Buffers are either 3.3 V compatible or 2.5 V Low Voltage TTL compatible signals, as noted.

Table 38. Crossbar Signal Descriptions

Name	I/O	Type	Description
OOB Interface			
oob_ad[7:0]	B	CMOS	OOB Address bus
oob_clk	I	CMOS	OOB Clock
oob_devsel0	I	CMOS	OOB device select
oob_devsel1	I	CMOS	OOB device select
oob_devsel2	I	CMOS	OOB device select
oob_devsel3	I	CMOS	OOB device select
oob_int_hi	O	CMOS	OOB Interrupt
oob_int_lo	O	CMOS	OOB Interrupt
oob_valid_L	I	CMOS	OOB Control

Table 38. Crossbar Signal Descriptions (Continued)

Name	I/O	Type	Description
oob_wait_L	O	CMOS	OOB Control
AIB Interface			
d2x_p[31:0]_[c, cn, e, en, o, on]	I	STI	DS to Xbar cell data
x2d_p[31:0]_[c, cn, e, en, o, on]	O	STI	Xbar to DS cell data
Power Supply, Clock Source, Reset, and Diagnostics			
plllock	O	CMOS	Test output
pwruprst_L	I	CMOS	Synchronous, Active Low Reset
soc_out	O	CMOS	Buffered soc_in (NC)
VDDA		Isolated Supply	VDD (2.5 V) for PLL
VDD		Supply	VDD (2.5 V)
GND		Supply	GND (0 V)
ref_clk and ref_ clkn	I	PECL	System 200MHz clock (Differential)
soc_in and soc_ inn	I	PECL	System Start-of-cell (Differential)
JTAG Interface			
jtag_tck	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tdi	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tdo	O	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tms	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_trst_L	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!

Table 38. Crossbar Signal Descriptions (Continued)

Name	I/O	Type	Description
ASIC Manufacturing Test Interface			
ce0_io	I	CMOS	Test (GND)
ce0_scan	I	CMOS	Test (GND)
lssd_ce1_a	I	CMOS	Test (VDD)
lssd_ce1_b	I	CMOS	Test (VDD)
lssd_ce1_c1	I	CMOS	Test (VDD)
lssd_ce1_c2	I	CMOS	Test (VDD)
lssd_scan_ in[15:0]	I	CMOS	Test: Scan input (GND)
lssd_scan_ out[15:0]	O	CMOS	Test: Scan output
M_ackreg_L	O	CMOS	Test (NC)
M_pending_L	O	CMOS	Test (NC)
M_xreset_L	O	CMOS	Test (NC)
plltest_in	I	CMOS	Test (GND)
plltest_out	O	CMOS	Test output (NC)
test_di1	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_di2	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_lt	I	CMOS	Test (VDD)
test_ri	I	CMOS	Test (VDD)

4.6 PINOUT AND PACKAGE INFORMATION

4.6.1 Pinout Tables

Table 39. Crossbar Pinout (left side)

	01	02	03	04	05	06	07	08	09	10	11
A	No Pin	UNUSED	test_di2	x2d_p0_en	lssd_scan_out5	d2x_p0_on	lssd_scan_out7	x2d_p1_en	UNUSED	d2x_p1_on	UNUSED
B	UNUSED	GND	UNUSED	VDD	x2d_p0_e	GND	d2x_p0_o	VDD	x2d_p1_e	GND	d2x_p1_o
C	ref_clkn	UNUSED	plllock	UNUSED	lssd_scan_out6	soc_inn	lssd_scan_out8	x2d_p0_on	x2d_p1_o	x2d_p1_on	x2d_p2_o
D	x2d_p31_e	VDD	UNUSED	GND	UNUSED	VDD	pwruprst_L	GND	d2x_p0_c	VDD	d2x_p1_cn
E	VDDA	x2d_p31_en	plltest_in	UNUSED	ref_clk	UNUSED	soc_in	oob_valid_L	x2d_p0_o	d2x_p0_cn	d2x_p1_c
F	d2x_p31_o	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	d2x_p0_e	GND	x2d_p2_cn
G	UNUSED	d2x_p31_on	plltest_out	UNUSED	UNUSED	UNUSED	UNUSED	soc_out	x2d_p0_c	x2d_p1_cn	d2x_p1_e
H	x2d_p30_e	VDD	x2d_p31_o	GND	UNUSED	VDD	UNUSED	GND	oob_devsel3	VDD	x2d_p1_c
J	UNUSED	x2d_p30_en	x2d_p30_on	d2x_p31_cn	x2d_p31_on	d2x_p31_en	x2d_p31_cn	UNUSED	UNUSED	oob_clk	d2x_p0_en
K	d2x_p30_o	GND	x2d_p30_o	VDD	d2x_p31_c	GND	x2d_p30_c	VDD	UNUSED	GND	x2d_p0_cn
L	UNUSED	d2x_p30_on	x2d_p29_on	d2x_p30_c	d2x_p30_cn	x2d_p29_c	d2x_p30_en	x2d_p30_cn	d2x_p31_e	x2d_p31_c	UNUSED
M	x2d_p29_e	VDD	d2x_p29_cn	GND	x2d_p29_o	VDD	d2x_p29_e	GND	d2x_p30_e	VDD	UNUSED
N	UNUSED	x2d_p29_en	x2d_p28_o	x2d_p28_on	d2x_p29_c	d2x_p28_c	x2d_p28_e	d2x_p29_en	x2d_p29_cn	UNUSED	UNUSED
P	d2x_p29_o	GND	x2d_p28_c	VDD	x2d_p28_cn	GND	d2x_p28_cn	VDD	x2d_p28_en	GND	UNUSED
R	UNUSED	d2x_p29_on	d2x_p28_o	d2x_p28_on	UNUSED	UNUSED	UNUSED	d2x_p28_en	d2x_p28_e	UNUSED	UNUSED
T	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	lssd_scan_out2	GND	lssd_scan_out3	VDD	UNUSED
U	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
V	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	lssd_scan_out1	GND	lssd_scan_out0	VDD	UNUSED
W	UNUSED	x2d_p26_e	x2d_p27_en	x2d_p27_e	UNUSED	UNUSED	UNUSED	x2d_p27_o	x2d_p27_on	UNUSED	UNUSED
Y	x2d_p26_en	GND	d2x_p27_cn	VDD	d2x_p27_c	GND	x2d_p27_c	VDD	d2x_p27_o	GND	UNUSED
AA	UNUSED	d2x_p26_o	d2x_p27_en	d2x_p27_e	x2d_p26_cn	x2d_p27_cn	d2x_p27_on	x2d_p26_o	d2x_p26_c	UNUSED	UNUSED
AB	d2x_p26_on	VDD	x2d_p26_c	GND	d2x_p26_en	VDD	x2d_p26_on	GND	x2d_p25_on	VDD	UNUSED
AC	UNUSED	x2d_p25_e	d2x_p26_e	x2d_p25_cn	x2d_p25_c	d2x_p26_cn	x2d_p25_o	d2x_p25_c	x2d_p24_on	d2x_p24_cn	UNUSED
AD	x2d_p25_en	GND	d2x_p25_en	VDD	x2d_p24_cn	GND	d2x_p25_cn	VDD	UNUSED	GND	x2d_p23_c
AE	UNUSED	d2x_p25_o	d2x_p25_e	x2d_p24_c	d2x_p24_e	x2d_p24_o	d2x_p24_c	UNUSED	UNUSED	UNUSED	d2x_p23_e
AF	d2x_p25_on	VDD	d2x_p24_en	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	x2d_p22_cn
AG	UNUSED	x2d_p24_e	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	x2d_p23_cn	x2d_p22_c	d2x_p22_en
AH	x2d_p24_en	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	d2x_p23_en	GND	x2d_p21_c
AJ	UNUSED	d2x_p24_o	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	x2d_p23_on	d2x_p23_c	d2x_p22_cn
AK	d2x_p24_on	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	d2x_p23_cn	VDD	d2x_p22_c
AL	UNUSED	UNUSED	lssd_scan_in14	UNUSED	lssd_scan_in12	UNUSED	lssd_scan_in10	x2d_p23_o	x2d_p22_on	x2d_p22_o	x2d_p21_on
AM	UNUSED	GND	UNUSED	VDD	x2d_p23_en	GND	d2x_p23_on	VDD	x2d_p22_en	GND	d2x_p22_on
AN	UNUSED	UNUSED	lssd_scan_in15	x2d_p23_e	lssd_scan_in13	d2x_p23_o	lssd_scan_in11	x2d_p22_e	UNUSED	d2x_p22_o	UNUSED

Table 40. Crossbar Pinout (center)

	12	13	14	15	16	17	18	19	20	21	22
A	x2d_p2_en	UNUSED	d2x_p2_on	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	x2d_p5_e	UNUSED	d2x_p5_o
B	VDD	x2d_p2_e	GND	d2x_p2_o	VDD	UNUSED	VDD	x2d_p5_en	GND	d2x_p5_on	VDD
C	d2x_p2_c	x2d_p3_on	x2d_p3_cn	d2x_p3_on	UNUSED	UNUSED	UNUSED	x2d_p4_e	d2x_p4_c	d2x_p4_e	x2d_p5_cn
D	GND	x2d_p3_o	VDD	d2x_p3_o	GND	UNUSED	GND	x2d_p4_en	VDD	d2x_p4_en	GND
E	x2d_p2_on	d2x_p2_cn	x2d_p3_c	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	d2x_p4_cn	x2d_p5_c	d2x_p5_e
F	VDD	d2x_p3_cn	GND	oob_ad5	VDD	UNUSED	VDD	oob_ad2	GND	x2d_p4_c	VDD
G	d2x_p2_en	x2d_p3_en	d2x_p3_c	oob_ad4	lssd_scan_out10	UNUSED	lssd_scan_out11	oob_ad3	x2d_p4_cn	d2x_p4_o	x2d_p5_o
H	GND	d2x_p2_e	VDD	d2x_p3_e	GND	UNUSED	GND	x2d_p4_on	VDD	x2d_p5_on	GND
J	d2x_p1_en	x2d_p2_c	x2d_p3_e	d2x_p3_en	lssd_scan_out9	UNUSED	lssd_scan_out12	x2d_p4_o	d2x_p4_on	d2x_p5_cn	x2d_p6_o
K	VDD	oob_int_hi	GND	oob_ad7	VDD	UNUSED	VDD	oob_ad0	GND	UNUSED	VDD
L	oob_devsel2	UNUSED	oob_int_lo	UNUSED	oob_ad6	UNUSED	oob_ad1	UNUSED	UNUSED	UNUSED	M_xreset_L
M	GND	oob_devsel1	VDD	oob_wait_L	GND	UNUSED	GND	UNUSED	VDD	UNUSED	GND
N	UNUSED	VDD	oob_devsel0	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	VDD	UNUSED
P	VDD	UNUSED	GND	UNUSED	VDD	GND	VDD	UNUSED	GND	UNUSED	VDD
R	UNUSED	UNUSED	UNUSED	GND	UNUSED	UNUSED	UNUSED	GND	UNUSED	UNUSED	UNUSED
T	GND	UNUSED	VDD	UNUSED	GND	VDD	GND	UNUSED	VDD	UNUSED	GND
U	UNUSED	UNUSED	GND	UNUSED	VDD	GND	VDD	UNUSED	GND	UNUSED	UNUSED
V	GND	UNUSED	VDD	UNUSED	GND	VDD	GND	UNUSED	VDD	UNUSED	GND
W	UNUSED	UNUSED	UNUSED	GND	UNUSED	UNUSED	UNUSED	GND	UNUSED	UNUSED	UNUSED
Y	VDD	UNUSED	GND	UNUSED	VDD	GND	VDD	UNUSED	GND	UNUSED	VDD
AA	UNUSED	VDD	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	VDD	UNUSED
AB	GND	UNUSED	VDD	UNUSED	GND	UNUSED	GND	UNUSED	VDD	UNUSED	GND
AC	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
AD	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	VDD	UNUSED	GND	UNUSED	VDD
AE	d2x_p22_e	x2d_p21_cn	x2d_p20_en	d2x_p20_e	lssd_scan_in9	UNUSED	lssd_scan_in6	x2d_p19_on	d2x_p19_o	d2x_p18_c	x2d_p17_on
AF	GND	d2x_p21_en	VDD	d2x_p20_en	GND	UNUSED	GND	x2d_p19_o	VDD	x2d_p18_o	GND
AG	d2x_p21_e	x2d_p20_e	d2x_p20_cn	UNUSED	lssd_scan_in8	UNUSED	lssd_scan_in7	UNUSED	x2d_p19_c	d2x_p19_on	x2d_p18_on
AH	VDD	d2x_p20_c	GND	UNUSED	VDD	UNUSED	VDD	UNUSED	GND	x2d_p19_cn	VDD
AJ	x2d_p21_o	d2x_p21_c	x2d_p20_cn	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	d2x_p19_c	x2d_p18_cn	d2x_p18_en
AK	GND	x2d_p20_on	VDD	d2x_p20_on	GND	UNUSED	GND	x2d_p19_e	VDD	d2x_p19_e	GND
AL	d2x_p21_cn	x2d_p20_o	x2d_p20_c	d2x_p20_o	UNUSED	UNUSED	UNUSED	x2d_p19_en	d2x_p19_cn	d2x_p19_en	x2d_p18_c
AM	VDD	x2d_p21_en	GND	d2x_p21_on	VDD	UNUSED	VDD	x2d_p18_e	GND	d2x_p18_o	VDD
AN	x2d_p21_e	UNUSED	d2x_p21_o	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	x2d_p18_en	UNUSED	d2x_p18_on

Table 41. Crossbar Pinout (right side)

	23	24	25	26	27	28	29	30	31	32	33
A	UNUSED	x2d_p6_e	UNUSED	d2x_p6_o	lssd_scan_out14	x2d_p7_e	lssd_scan_out4	d2x_p7_o	jtag_tdo	UNUSED	UNUSED
B	x2d_p6_en	GND	d2x_p6_on	VDD	x2d_p7_en	GND	d2x_p7_on	VDD	UNUSED	GND	UNUSED
C	d2x_p5_en	d2x_p6_e	d2x_p6_en	d2x_p7_e	lssd_scan_out13	UNUSED	lssd_scan_out15	UNUSED	jtag_tms	UNUSED	jtag_tdi
D	x2d_p6_c	VDD	x2d_p7_cn	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	x2d_p8_en
E	x2d_p6_cn	x2d_p7_c	d2x_p7_en	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	jtag_tck	x2d_p8_e	UNUSED
F	d2x_p5_c	GND	x2d_p7_on	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	d2x_p8_on
G	x2d_p6_on	d2x_p6_c	d2x_p7_cn	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	jtag_trst_L	d2x_p8_o	UNUSED
H	d2x_p6_cn	VDD	M_pending_L	GND	UNUSED	VDD	UNUSED	GND	x2d_p8_on	VDD	x2d_p9_en
J	x2d_p7_o	M_ackreg_L	UNUSED	UNUSED	x2d_p8_c	d2x_p8_e	x2d_p8_o	d2x_p8_c	x2d_p9_o	x2d_p9_e	UNUSED
K	d2x_p7_c	GND	UNUSED	VDD	x2d_p9_cn	GND	d2x_p8_cn	VDD	x2d_p9_on	GND	d2x_p9_on
L	UNUSED	x2d_p8_cn	d2x_p8_en	x2d_p9_c	d2x_p9_e	x2d_p10_cn	d2x_p9_c	d2x_p9_cn	x2d_p10_o	d2x_p9_o	UNUSED
M	UNUSED	VDD	d2x_p9_en	GND	d2x_p10_en	VDD	x2d_p10_on	GND	d2x_p10_c	VDD	x2d_p10_en
N	UNUSED	UNUSED	x2d_p10_c	d2x_p10_e	x2d_p11_en	d2x_p11_cn	d2x_p10_cn	x2d_p11_o	x2d_p11_on	x2d_p10_e	UNUSED
P	UNUSED	GND	x2d_p11_e	VDD	d2x_p11_c	GND	x2d_p11_c	VDD	x2d_p11_cn	GND	d2x_p10_on
R	UNUSED	UNUSED	d2x_p11_en	d2x_p11_e	UNUSED	UNUSED	UNUSED	d2x_p11_o	d2x_p11_on	d2x_p10_o	UNUSED
T	UNUSED	VDD	ce0_io	GND	ce0_scan	VDD	UNUSED	GND	UNUSED	VDD	UNUSED
U	UNUSED	UNUSED	UNUSED	UNUSED	test_lt	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
V	UNUSED	VDD	test_ri	GND	test_di1	VDD	UNUSED	GND	UNUSED	VDD	UNUSED
W	UNUSED	UNUSED	x2d_p12_o	x2d_p12_on	UNUSED	UNUSED	UNUSED	x2d_p12_en	x2d_p12_e	x2d_p13_en	UNUSED
Y	UNUSED	GND	d2x_p12_on	VDD	x2d_p12_cn	GND	d2x_p12_cn	VDD	d2x_p12_c	GND	x2d_p13_e
AA	UNUSED	UNUSED	d2x_p13_cn	x2d_p13_on	d2x_p12_o	x2d_p12_c	x2d_p13_c	d2x_p12_en	d2x_p12_e	d2x_p13_on	UNUSED
AB	UNUSED	VDD	x2d_p14_o	GND	x2d_p13_o	VDD	d2x_p13_e	GND	x2d_p13_cn	VDD	d2x_p13_o
AC	UNUSED	d2x_p15_c	x2d_p15_o	d2x_p14_cn	x2d_p14_on	d2x_p13_c	x2d_p14_cn	x2d_p14_c	d2x_p13_en	x2d_p14_en	UNUSED
AD	d2x_p16_cn	GND	UNUSED	VDD	d2x_p14_c	GND	x2d_p15_c	VDD	d2x_p14_e	GND	x2d_p14_e
AE	x2d_p16_on	UNUSED	UNUSED	UNUSED	d2x_p15_cn	x2d_p15_on	d2x_p15_en	x2d_p15_cn	d2x_p14_en	d2x_p14_on	UNUSED
AF	d2x_p17_c	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	d2x_p15_e	VDD	d2x_p14_o
AG	x2d_p17_o	d2x_p17_cn	d2x_p16_c	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	lssd_ce1_a	x2d_p15_en	UNUSED
AH	d2x_p18_cn	GND	x2d_p16_o	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	x2d_p15_e
AJ	x2d_p17_c	x2d_p16_cn	d2x_p16_e	UNUSED	UNUSED	UNUSED	lssd_ce1_c1	UNUSED	lssd_ce1_b	d2x_p15_on	UNUSED
AK	x2d_p17_cn	VDD	x2d_p16_c	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	d2x_p15_o
AL	d2x_p18_e	d2x_p17_en	d2x_p17_e	d2x_p16_en	lssd_scan_in5	UNUSED	lssd_scan_in3	UNUSED	lssd_scan_in1	UNUSED	lssd_ce1_c2
AM	x2d_p17_e	GND	d2x_p17_o	VDD	x2d_p16_e	GND	d2x_p16_o	VDD	UNUSED	GND	UNUSED
AN	UNUSED	x2d_p17_en	UNUSED	d2x_p17_on	lssd_scan_in4	x2d_p16_en	lssd_scan_in2	d2x_p16_on	lssd_scan_in0	UNUSED	UNUSED

Table 33. Crossbar Alpha Pin List

Signal Name	Pin
ce0_io	T25
ce0_scan	T27
d2x_p0_c	D09
d2x_p0_cn	E10
d2x_p0_e	F09
d2x_p0_en	J11
d2x_p0_o	B07
d2x_p0_on	A06
d2x_p10_c	M31
d2x_p10_cn	N29
d2x_p10_e	N26
d2x_p10_en	M27
d2x_p10_o	R32
d2x_p10_on	P33
d2x_p11_c	P27
d2x_p11_cn	N28
d2x_p11_e	R26
d2x_p11_en	R25
d2x_p11_o	R30
d2x_p11_on	R31
d2x_p12_c	Y31
d2x_p12_cn	Y29
d2x_p12_e	AA31
d2x_p12_en	AA30
d2x_p12_o	AA27
d2x_p12_on	Y25
d2x_p13_c	AC28
d2x_p13_cn	AA25
d2x_p13_e	AB29
d2x_p13_en	AC31
d2x_p13_o	AB33
d2x_p13_on	AA32
d2x_p14_c	AD27
d2x_p14_cn	AC26
d2x_p14_e	AD31
d2x_p14_en	AE31
d2x_p14_o	AF33
d2x_p14_on	AE32
d2x_p15_c	AC24
d2x_p15_cn	AE27
d2x_p15_e	AF31
d2x_p15_en	AE29
d2x_p15_o	AK33
d2x_p15_on	AJ32
d2x_p16_c	AG25
d2x_p16_cn	AD23
d2x_p16_e	AJ25
d2x_p16_en	AL26
d2x_p16_o	AM29
d2x_p16_on	AN30
d2x_p17_c	AF23
d2x_p17_cn	AG24
d2x_p17_e	AL25
d2x_p17_en	AL24
d2x_p17_o	AM25
d2x_p17_on	AN26
d2x_p18_c	AE21
d2x_p18_cn	AH23
d2x_p18_e	AL23
d2x_p18_en	AJ22
d2x_p18_o	AM21
d2x_p18_on	AN22
d2x_p19_c	AJ20
d2x_p19_cn	AL20
d2x_p19_e	AK21
d2x_p19_en	AL21
d2x_p19_o	AE20
d2x_p19_on	AG21
d2x_p1_c	E11
d2x_p1_cn	D11
d2x_p1_e	G11
d2x_p1_en	J12
d2x_p1_o	B11
d2x_p1_on	A10
d2x_p20_c	AH13
d2x_p20_cn	AG14
d2x_p20_e	AE15
d2x_p20_en	AF15
d2x_p20_o	AL15
d2x_p20_on	AK15
d2x_p21_c	AJ13
d2x_p21_cn	AL12
d2x_p21_e	AG12
d2x_p21_en	AF13
d2x_p21_o	AN14
d2x_p21_on	AM15
d2x_p22_c	AK11
d2x_p22_cn	AJ11
d2x_p22_e	AE12
d2x_p22_en	AG11
d2x_p22_o	AN10
d2x_p22_on	AM11
d2x_p23_c	AJ10
d2x_p23_cn	AK09
d2x_p23_e	AE11
d2x_p23_en	AH09
d2x_p23_o	AN06
d2x_p23_on	AM07
d2x_p24_c	AE07
d2x_p24_cn	AC10
d2x_p24_e	AE05
d2x_p24_en	AF03
d2x_p24_o	AJ02
d2x_p24_on	AK01
d2x_p25_c	AC08
d2x_p25_cn	AD07
d2x_p25_e	AE03
d2x_p25_en	AD03
d2x_p25_o	AE02
d2x_p25_on	AF01
d2x_p26_c	AA09
d2x_p26_cn	AC06
d2x_p26_e	AC03
d2x_p26_en	AB05
d2x_p26_o	AA02
d2x_p26_on	AB01
d2x_p27_c	Y05
d2x_p27_cn	Y03
d2x_p27_e	AA04
d2x_p27_en	AA03
d2x_p27_o	Y09
d2x_p27_on	AA07
d2x_p28_c	N06
d2x_p28_cn	P07
d2x_p28_e	R09
d2x_p28_en	R08
d2x_p28_o	R03
d2x_p28_on	R04
d2x_p29_c	N05
d2x_p29_cn	M03
d2x_p29_e	M07
d2x_p29_en	N08
d2x_p29_o	P01
d2x_p29_on	R02
d2x_p2_c	C12
d2x_p2_cn	E13
d2x_p2_e	H13
d2x_p2_en	G12
d2x_p2_o	B15
d2x_p2_on	A14
d2x_p30_c	L04
d2x_p30_cn	L05
d2x_p30_e	M09
d2x_p30_en	L07
d2x_p30_o	K01
d2x_p30_on	L02
d2x_p31_c	K05
d2x_p31_cn	J04
d2x_p31_e	L09
d2x_p31_en	J06
d2x_p31_o	F01
d2x_p31_on	G02
d2x_p3_c	G14
d2x_p3_cn	F13
d2x_p3_e	H15
d2x_p3_en	J15
d2x_p3_o	D15
d2x_p3_on	C15
d2x_p4_c	C20
d2x_p4_cn	E20
d2x_p4_e	C21
d2x_p4_en	D21
d2x_p4_o	G21
d2x_p4_on	J20
d2x_p5_c	F23
d2x_p5_cn	J21
d2x_p5_e	E22
d2x_p5_en	C23
d2x_p5_o	A22
d2x_p5_on	B21
d2x_p6_c	G24
d2x_p6_cn	H23
d2x_p6_e	C24
d2x_p6_en	C25
d2x_p6_o	A26
d2x_p6_on	B25
d2x_p7_c	K23
d2x_p7_cn	G25
d2x_p7_e	C26
d2x_p7_en	E25
d2x_p7_o	A30
d2x_p7_on	B29
d2x_p8_c	J30

Signal Name	Pin
d2x_p8_cn	K29
d2x_p8_e	J28
d2x_p8_en	L25
d2x_p8_o	G32
d2x_p8_on	F33
d2x_p9_c	L29
d2x_p9_cn	L30
d2x_p9_e	L27
d2x_p9_en	M25
d2x_p9_o	L32
d2x_p9_on	K33
GND	B02
GND	B06
GND	B10
GND	B14
GND	B20
GND	B24
GND	B28
GND	B32
GND	D04
GND	D08
GND	D12
GND	D16
GND	D18
GND	D22
GND	D26
GND	D30
GND	F02
GND	F06
GND	F10
GND	F14
GND	F20
GND	F24
GND	F28
GND	F32
GND	H04
GND	H08
GND	H12
GND	H16
GND	H18
GND	H22
GND	H26
GND	H30
GND	K02
GND	K06
GND	K10
GND	K14
GND	K20
GND	K24
GND	K28
GND	K32
GND	M04
GND	M08
GND	M12
GND	M16
GND	M18
GND	M22
GND	M26
GND	M30
GND	P02
GND	P06

Signal Name	Pin
GND	P10
GND	P14
GND	P17
GND	P20
GND	P24
GND	P28
GND	P32
GND	R15
GND	R19
GND	T04
GND	T08
GND	T12
GND	T16
GND	T18
GND	T22
GND	T26
GND	T30
GND	U14
GND	U17
GND	U20
GND	V04
GND	V08
GND	V12
GND	V16
GND	V18
GND	V22
GND	V26
GND	V30
GND	W15
GND	W19
GND	Y02
GND	Y06
GND	Y10
GND	Y14
GND	Y17
GND	Y20
GND	Y24
GND	Y28
GND	Y32
GND	AB04
GND	AB08
GND	AB12
GND	AB16
GND	AB18
GND	AB22
GND	AB26
GND	AB30
GND	AD02
GND	AD06
GND	AD10
GND	AD14
GND	AD20
GND	AD24
GND	AD28
GND	AD32
GND	AF04
GND	AF08
GND	AF12
GND	AF16
GND	AF18
GND	AF22

Signal Name	Pin
GND	AF26
GND	AF30
GND	AH02
GND	AH06
GND	AH10
GND	AH14
GND	AH20
GND	AH24
GND	AH28
GND	AH32
GND	AK04
GND	AK08
GND	AK12
GND	AK16
GND	AK18
GND	AK22
GND	AK26
GND	AK30
GND	AM02
GND	AM06
GND	AM10
GND	AM14
GND	AM20
GND	AM24
GND	AM28
GND	AM32
jtag_tck	E31
jtag_tdi	C33
jtag_tdo	A31
jtag_tms	C31
jtag_trst_L	G31
lssd_ce1_a	AG31
lssd_ce1_b	AJ31
lssd_ce1_c1	AJ29
lssd_ce1_c2	AL33
lssd_scan_in0	AN31
lssd_scan_in1	AL31
lssd_scan_in10	AL07
lssd_scan_in11	AN07
lssd_scan_in12	AL05
lssd_scan_in13	AN05
lssd_scan_in14	AL03
lssd_scan_in15	AN03
lssd_scan_in2	AN29
lssd_scan_in3	AL29
lssd_scan_in4	AN27
lssd_scan_in5	AL27
lssd_scan_in6	AE18
lssd_scan_in7	AG18
lssd_scan_in8	AG16
lssd_scan_in9	AE16
lssd_scan_out0	V09
lssd_scan_out1	V07
lssd_scan_out10	G16
lssd_scan_out11	G18
lssd_scan_out12	J18
lssd_scan_out13	C27
lssd_scan_out14	A27
lssd_scan_out15	C29
lssd_scan_out2	T07
lssd_scan_out3	T09

Signal Name	Pin
lssd_scan_out4	A29
lssd_scan_out5	A05
lssd_scan_out6	C05
lssd_scan_out7	A07
lssd_scan_out8	C07
lssd_scan_out9	J16
M_ackreg_L	J24
M_pending_L	H25
M_xreset_L	L22
No Pin	A01
oob_ad0	K19
oob_ad1	L18
oob_ad2	F19
oob_ad3	G19
oob_ad4	G15
oob_ad5	F15
oob_ad6	L16
oob_ad7	K15
oob_clk	J10
oob_devsel0	N14
oob_devsel1	M13
oob_devsel2	L12
oob_devsel3	H09
oob_int_hi	K13
oob_int_lo	L14
oob_valid_L	E08
oob_wait_L	M15
plllock	C03
plltest_in	E03
plltest_out	G03
pwruprst_L	D07
ref_clk	E05
ref_clkn	C01
soc_in	E07
soc_inn	C06
soc_out	G08
test_di1	V27
test_di2	A03
test_lt	U27
test_ri	V25
UNUSED	A02
UNUSED	A09
UNUSED	A11
UNUSED	A13
UNUSED	A15
UNUSED	A16
UNUSED	A17
UNUSED	A18
UNUSED	A19
UNUSED	A21
UNUSED	A23
UNUSED	A25
UNUSED	A32
UNUSED	A33
UNUSED	B01
UNUSED	B03
UNUSED	B17
UNUSED	B31
UNUSED	B33
UNUSED	C02
UNUSED	C04

Signal Name	Pin
UNUSED	C16
UNUSED	C17
UNUSED	C18
UNUSED	C28
UNUSED	C30
UNUSED	C32
UNUSED	D03
UNUSED	D05
UNUSED	D17
UNUSED	D27
UNUSED	D29
UNUSED	D31
UNUSED	E04
UNUSED	E06
UNUSED	E15
UNUSED	E16
UNUSED	E17
UNUSED	E18
UNUSED	E19
UNUSED	E26
UNUSED	E27
UNUSED	E28
UNUSED	E29
UNUSED	E30
UNUSED	E33
UNUSED	F03
UNUSED	F05
UNUSED	F07
UNUSED	F17
UNUSED	F27
UNUSED	F29
UNUSED	F31
UNUSED	G01
UNUSED	G04
UNUSED	G05
UNUSED	G06
UNUSED	G07
UNUSED	G17
UNUSED	G26
UNUSED	G27
UNUSED	G28
UNUSED	G29
UNUSED	G30
UNUSED	G33
UNUSED	H05
UNUSED	H07
UNUSED	H17
UNUSED	H27
UNUSED	H29
UNUSED	J01
UNUSED	J08
UNUSED	J09
UNUSED	J17
UNUSED	J25
UNUSED	J26
UNUSED	J33
UNUSED	K09
UNUSED	K17
UNUSED	K21
UNUSED	K25
UNUSED	L01

Signal Name	Pin
UNUSED	L11
UNUSED	L13
UNUSED	L15
UNUSED	L17
UNUSED	L19
UNUSED	L20
UNUSED	L21
UNUSED	L23
UNUSED	L33
UNUSED	M11
UNUSED	M17
UNUSED	M19
UNUSED	M21
UNUSED	M23
UNUSED	N01
UNUSED	N10
UNUSED	N11
UNUSED	N12
UNUSED	N15
UNUSED	N16
UNUSED	N17
UNUSED	N18
UNUSED	N19
UNUSED	N20
UNUSED	N22
UNUSED	N23
UNUSED	N24
UNUSED	N33
UNUSED	P11
UNUSED	P13
UNUSED	P15
UNUSED	P19
UNUSED	P21
UNUSED	P23
UNUSED	R01
UNUSED	R05
UNUSED	R06
UNUSED	R07
UNUSED	R10
UNUSED	R11
UNUSED	R12
UNUSED	R13
UNUSED	R14
UNUSED	R16
UNUSED	R17
UNUSED	R18
UNUSED	R20
UNUSED	R21
UNUSED	R22
UNUSED	R23
UNUSED	R24
UNUSED	R27
UNUSED	R28
UNUSED	R29
UNUSED	R33
UNUSED	T01
UNUSED	T03
UNUSED	T05
UNUSED	T11
UNUSED	T13
UNUSED	T15

Signal Name	Pin
UNUSED	T19
UNUSED	T21
UNUSED	T23
UNUSED	T29
UNUSED	T31
UNUSED	T33
UNUSED	U01
UNUSED	U02
UNUSED	U03
UNUSED	U04
UNUSED	U05
UNUSED	U06
UNUSED	U07
UNUSED	U08
UNUSED	U09
UNUSED	U10
UNUSED	U11
UNUSED	U12
UNUSED	U13
UNUSED	U15
UNUSED	U19
UNUSED	U21
UNUSED	U22
UNUSED	U23
UNUSED	U24
UNUSED	U25
UNUSED	U26
UNUSED	U28
UNUSED	U29
UNUSED	U30
UNUSED	U31
UNUSED	U32
UNUSED	U33
UNUSED	V01
UNUSED	V03
UNUSED	V05
UNUSED	V11
UNUSED	V13
UNUSED	V15
UNUSED	V19
UNUSED	V21
UNUSED	V23
UNUSED	V29
UNUSED	V31
UNUSED	V33
UNUSED	W01
UNUSED	W05
UNUSED	W06
UNUSED	W07
UNUSED	W10
UNUSED	W11
UNUSED	W12
UNUSED	W13
UNUSED	W14
UNUSED	W16
UNUSED	W17
UNUSED	W18
UNUSED	W20
UNUSED	W21
UNUSED	W22
UNUSED	W23

Signal Name	Pin
UNUSED	W24
UNUSED	W27
UNUSED	W28
UNUSED	W29
UNUSED	W33
UNUSED	Y11
UNUSED	Y13
UNUSED	Y15
UNUSED	Y19
UNUSED	Y21
UNUSED	Y23
UNUSED	AA01
UNUSED	AA10
UNUSED	AA11
UNUSED	AA12
UNUSED	AA14
UNUSED	AA15
UNUSED	AA16
UNUSED	AA17
UNUSED	AA18
UNUSED	AA19
UNUSED	AA20
UNUSED	AA22
UNUSED	AA23
UNUSED	AA24
UNUSED	AA33
UNUSED	AB11
UNUSED	AB13
UNUSED	AB15
UNUSED	AB17
UNUSED	AB19
UNUSED	AB21
UNUSED	AB23
UNUSED	AC01
UNUSED	AC11
UNUSED	AC12
UNUSED	AC13
UNUSED	AC14
UNUSED	AC15
UNUSED	AC16
UNUSED	AC17
UNUSED	AC18
UNUSED	AC19
UNUSED	AC20
UNUSED	AC21
UNUSED	AC22
UNUSED	AC23
UNUSED	AC33
UNUSED	AD09
UNUSED	AD13
UNUSED	AD15
UNUSED	AD17
UNUSED	AD19
UNUSED	AD21
UNUSED	AD25
UNUSED	AE01
UNUSED	AE08
UNUSED	AE09
UNUSED	AE10
UNUSED	AE17
UNUSED	AE24

Signal Name	Pin
UNUSED	AE25
UNUSED	AE26
UNUSED	AE33
UNUSED	AF05
UNUSED	AF07
UNUSED	AF09
UNUSED	AF17
UNUSED	AF25
UNUSED	AF27
UNUSED	AF29
UNUSED	AG01
UNUSED	AG03
UNUSED	AG04
UNUSED	AG05
UNUSED	AG06
UNUSED	AG07
UNUSED	AG08
UNUSED	AG15
UNUSED	AG17
UNUSED	AG19
UNUSED	AG26
UNUSED	AG27
UNUSED	AG28
UNUSED	AG29
UNUSED	AG30
UNUSED	AG33
UNUSED	AH03
UNUSED	AH05
UNUSED	AH07
UNUSED	AH15
UNUSED	AH17
UNUSED	AH19
UNUSED	AH27
UNUSED	AH29
UNUSED	AH31
UNUSED	AJ01
UNUSED	AJ03
UNUSED	AJ04
UNUSED	AJ05
UNUSED	AJ06
UNUSED	AJ07
UNUSED	AJ08
UNUSED	AJ15
UNUSED	AJ16
UNUSED	AJ17
UNUSED	AJ18
UNUSED	AJ19
UNUSED	AJ26
UNUSED	AJ27
UNUSED	AJ28
UNUSED	AJ30
UNUSED	AJ33
UNUSED	AK03
UNUSED	AK05
UNUSED	AK07
UNUSED	AK17
UNUSED	AK27
UNUSED	AK29
UNUSED	AK31
UNUSED	AL01
UNUSED	AL02

Signal Name	Pin
UNUSED	AL04
UNUSED	AL06
UNUSED	AL16
UNUSED	AL17
UNUSED	AL18
UNUSED	AL28
UNUSED	AL30
UNUSED	AL32
UNUSED	AM01
UNUSED	AM03
UNUSED	AM17
UNUSED	AM31
UNUSED	AM33
UNUSED	AN01
UNUSED	AN02
UNUSED	AN09
UNUSED	AN11
UNUSED	AN13
UNUSED	AN15
UNUSED	AN16
UNUSED	AN17
UNUSED	AN18
UNUSED	AN19
UNUSED	AN21
UNUSED	AN23
UNUSED	AN25
UNUSED	AN32
UNUSED	AN33
VDD	B04
VDD	B12
VDD	B18
VDD	B26
VDD	D10
VDD	D20
VDD	D28
VDD	D32
VDD	F04
VDD	F08
VDD	F16
VDD	F22
VDD	H02
VDD	H14
VDD	H24
VDD	H28
VDD	K08
VDD	K12
VDD	K18
VDD	K30
VDD	M06
VDD	M24
VDD	M32
VDD	N13
VDD	N21
VDD	P04
VDD	P16
VDD	P18
VDD	P26
VDD	T02
VDD	T10
VDD	T14
VDD	T17

Signal Name	Pin
VDD	T20
VDD	T28
VDD	U16
VDD	U18
VDD	V06
VDD	V14
VDD	V17
VDD	V20
VDD	V24
VDD	V32
VDD	Y08
VDD	Y16
VDD	Y18
VDD	Y30
VDD	AA13
VDD	AA21
VDD	AB02
VDD	AB10
VDD	AB28
VDD	AD04
VDD	AD16
VDD	AD22
VDD	AD26
VDD	AF06
VDD	AF10
VDD	AF20
VDD	AF32
VDD	AH12
VDD	AH18
VDD	AH26
VDD	AH30
VDD	AK02
VDD	AK06
VDD	AK14
VDD	AK24
VDD	AM08
VDD	AM16
VDD	AM22
VDD	AM30
VDD	F30
VDD	H32
VDD	K26
VDD	M28
VDD	P22
VDD	P30
VDD	T24
VDD	T32
VDD	V28
VDD	Y22
VDD	Y26
VDD	AB24
VDD	AB32
VDD	AD30
VDD	AF28
VDD	AK32
VDD	AB14
VDD	AB20
VDD	AD12
VDD	AD18
VDD	AF14
VDD	AF24

Signal Name	Pin
VDD	AH08
VDD	AH16
VDD	AH22
VDD	AK10
VDD	AK20
VDD	AK28
VDD	AM04
VDD	AM12
VDD	AM18
VDD	AM26
VDD	D02
VDD	H06
VDD	K04
VDD	M02
VDD	M10
VDD	P08
VDD	P12
VDD	T06
VDD	V02
VDD	V10
VDD	Y04
VDD	Y12
VDD	AB06
VDD	AD08
VDD	AF02
VDD	AH04
VDD	B08
VDD	B16
VDD	B22
VDD	B30
VDD	D06
VDD	D14
VDD	D24
VDD	F12
VDD	F18
VDD	F26
VDD	H10
VDD	H20
VDD	K16
VDD	K22
VDD	M14
VDD	M20
VDDA	E01
x2d_p0_c	G09
x2d_p0_cn	K11
x2d_p0_e	B05
x2d_p0_en	A04
x2d_p0_o	E09
x2d_p0_on	C08
x2d_p10_c	N25
x2d_p10_cn	L28
x2d_p10_e	N32
x2d_p10_en	M33
x2d_p10_o	L31
x2d_p10_on	M29
x2d_p11_c	P29
x2d_p11_cn	P31
x2d_p11_e	P25
x2d_p11_en	N27
x2d_p11_o	N30
x2d_p11_on	N31

Signal Name	Pin
x2d_p12_c	AA28
x2d_p12_cn	Y27
x2d_p12_e	W31
x2d_p12_en	W30
x2d_p12_o	W25
x2d_p12_on	W26
x2d_p13_c	AA29
x2d_p13_cn	AB31
x2d_p13_e	Y33
x2d_p13_en	W32
x2d_p13_o	AB27
x2d_p13_on	AA26
x2d_p14_c	AC30
x2d_p14_cn	AC29
x2d_p14_e	AD33
x2d_p14_en	AC32
x2d_p14_o	AB25
x2d_p14_on	AC27
x2d_p15_c	AD29
x2d_p15_cn	AE30
x2d_p15_e	AH33
x2d_p15_en	AG32
x2d_p15_o	AC25
x2d_p15_on	AE28
x2d_p16_c	AK25
x2d_p16_cn	AJ24
x2d_p16_e	AM27
x2d_p16_en	AN28
x2d_p16_o	AH25
x2d_p16_on	AE23
x2d_p17_c	AJ23
x2d_p17_cn	AK23
x2d_p17_e	AM23
x2d_p17_en	AN24
x2d_p17_o	AG23
x2d_p17_on	AE22
x2d_p18_c	AL22
x2d_p18_cn	AJ21
x2d_p18_e	AM19
x2d_p18_en	AN20
x2d_p18_o	AF21
x2d_p18_on	AG22
x2d_p19_c	AG20
x2d_p19_cn	AH21
x2d_p19_e	AK19
x2d_p19_en	AL19
x2d_p19_o	AF19
x2d_p19_on	AE19
x2d_p1_c	H11
x2d_p1_cn	G10
x2d_p1_e	B09
x2d_p1_en	A08
x2d_p1_o	C09
x2d_p1_on	C10
x2d_p20_c	AL14
x2d_p20_cn	AJ14
x2d_p20_e	AG13
x2d_p20_en	AE14
x2d_p20_o	AL13
x2d_p20_on	AK13
x2d_p21_c	AH11

Signal Name	Pin
x2d_p21_cn	AE13
x2d_p21_e	AN12
x2d_p21_en	AM13
x2d_p21_o	AJ12
x2d_p21_on	AL11
x2d_p22_c	AG10
x2d_p22_cn	AF11
x2d_p22_e	AN08
x2d_p22_en	AM09
x2d_p22_o	AL10
x2d_p22_on	AL09
x2d_p23_c	AD11
x2d_p23_cn	AG09
x2d_p23_e	AN04
x2d_p23_en	AM05
x2d_p23_o	AL08
x2d_p23_on	AJ09
x2d_p24_c	AE04
x2d_p24_cn	AD05
x2d_p24_e	AG02
x2d_p24_en	AH01
x2d_p24_o	AE06
x2d_p24_on	AC09
x2d_p25_c	AC05
x2d_p25_cn	AC04
x2d_p25_e	AC02
x2d_p25_en	AD01
x2d_p25_o	AC07
x2d_p25_on	AB09
x2d_p26_c	AB03
x2d_p26_cn	AA05
x2d_p26_e	W02
x2d_p26_en	Y01
x2d_p26_o	AA08
x2d_p26_on	AB07
x2d_p27_c	Y07
x2d_p27_cn	AA06
x2d_p27_e	W04
x2d_p27_en	W03
x2d_p27_o	W08
x2d_p27_on	W09
x2d_p28_c	P03
x2d_p28_cn	P05
x2d_p28_e	N07
x2d_p28_en	P09
x2d_p28_o	N03
x2d_p28_on	N04
x2d_p29_c	L06
x2d_p29_cn	N09
x2d_p29_e	M01
x2d_p29_en	N02
x2d_p29_o	M05
x2d_p29_on	L03
x2d_p2_c	J13
x2d_p2_cn	F11
x2d_p2_e	B13
x2d_p2_en	A12
x2d_p2_o	C11
x2d_p2_on	E12
x2d_p30_c	K07
x2d_p30_cn	L08

Signal Name	Pin
x2d_p30_e	H01
x2d_p30_en	J02
x2d_p30_o	K03
x2d_p30_on	J03
x2d_p31_c	L10
x2d_p31_cn	J07
x2d_p31_e	D01
x2d_p31_en	E02
x2d_p31_o	H03
x2d_p31_on	J05
x2d_p3_c	E14
x2d_p3_cn	C14
x2d_p3_e	J14
x2d_p3_en	G13
x2d_p3_o	D13
x2d_p3_on	C13
x2d_p4_c	F21
x2d_p4_cn	G20
x2d_p4_e	C19
x2d_p4_en	D19
x2d_p4_o	J19
x2d_p4_on	H19
x2d_p5_c	E21
x2d_p5_cn	C22
x2d_p5_e	A20
x2d_p5_en	B19
x2d_p5_o	G22
x2d_p5_on	H21
x2d_p6_c	D23
x2d_p6_cn	E23
x2d_p6_e	A24
x2d_p6_en	B23
x2d_p6_o	J22
x2d_p6_on	G23
x2d_p7_c	E24
x2d_p7_cn	D25
x2d_p7_e	A28
x2d_p7_en	B27
x2d_p7_o	J23
x2d_p7_on	F25
x2d_p8_c	J27
x2d_p8_cn	L24
x2d_p8_e	E32
x2d_p8_en	D33
x2d_p8_o	J29
x2d_p8_on	H31
x2d_p9_c	L26
x2d_p9_cn	K27
x2d_p9_e	J32
x2d_p9_en	H33
x2d_p9_o	J31
x2d_p9_on	K31

4.6.2 Package Dimensions

This section outlines the mechanical dimensions for the Crossbar device. The package is a 1088 ceramic column grid array (CCGA).

NOTE: Drawings are not to scale.

Figure 63. Crossbar CCGA Package Dimensions - Top and Side Views

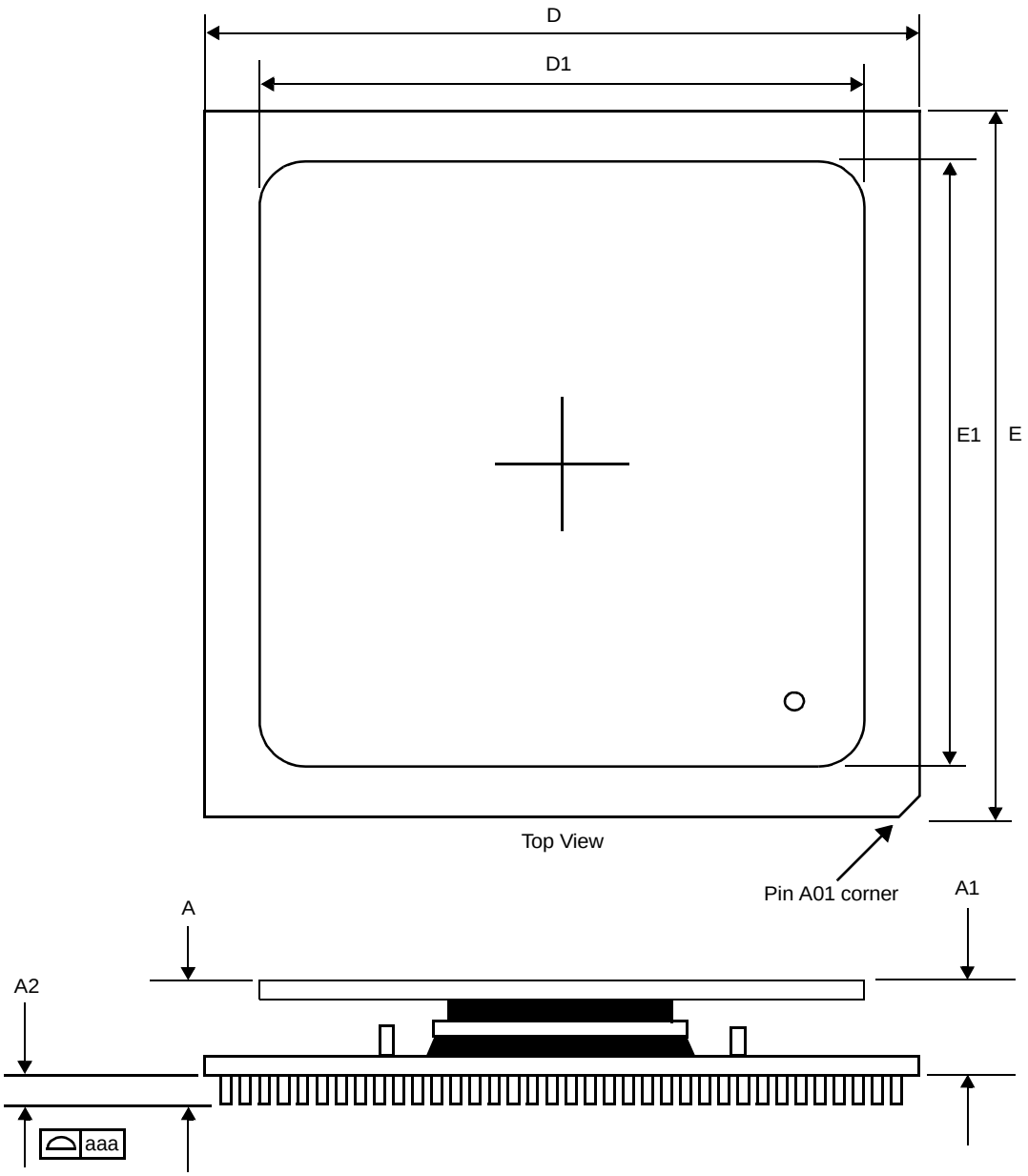


Figure 64. Crossbar CCGA package Dimensions - Bottom View

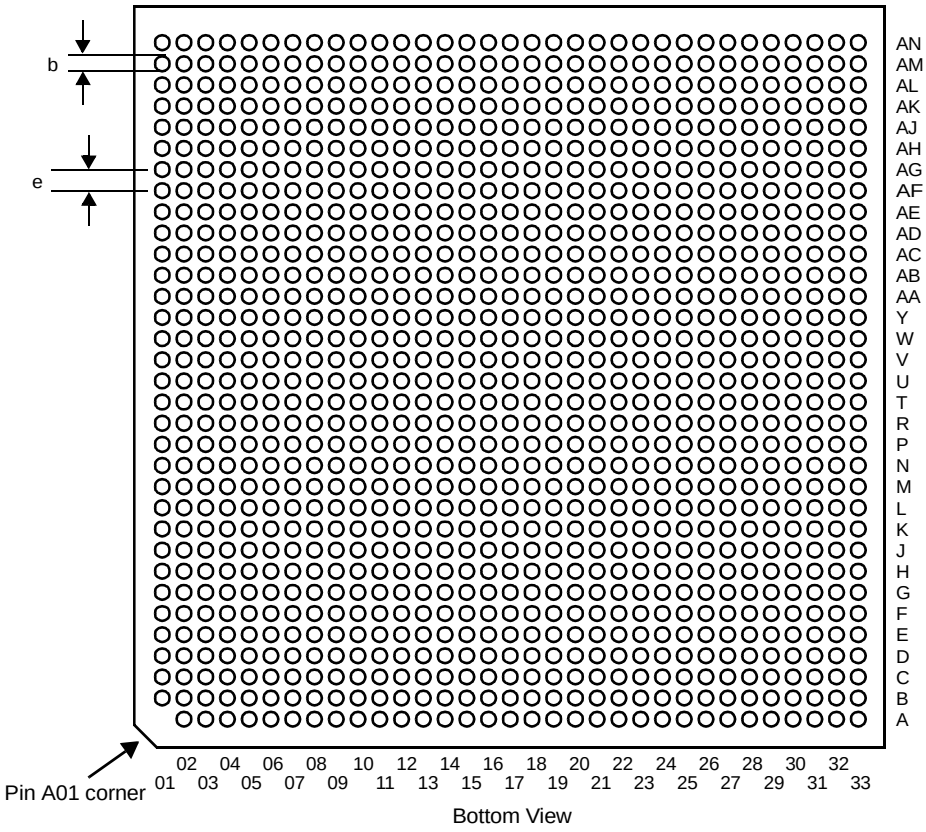


Table 34. Crossbar CCGA Mechanical Specifications

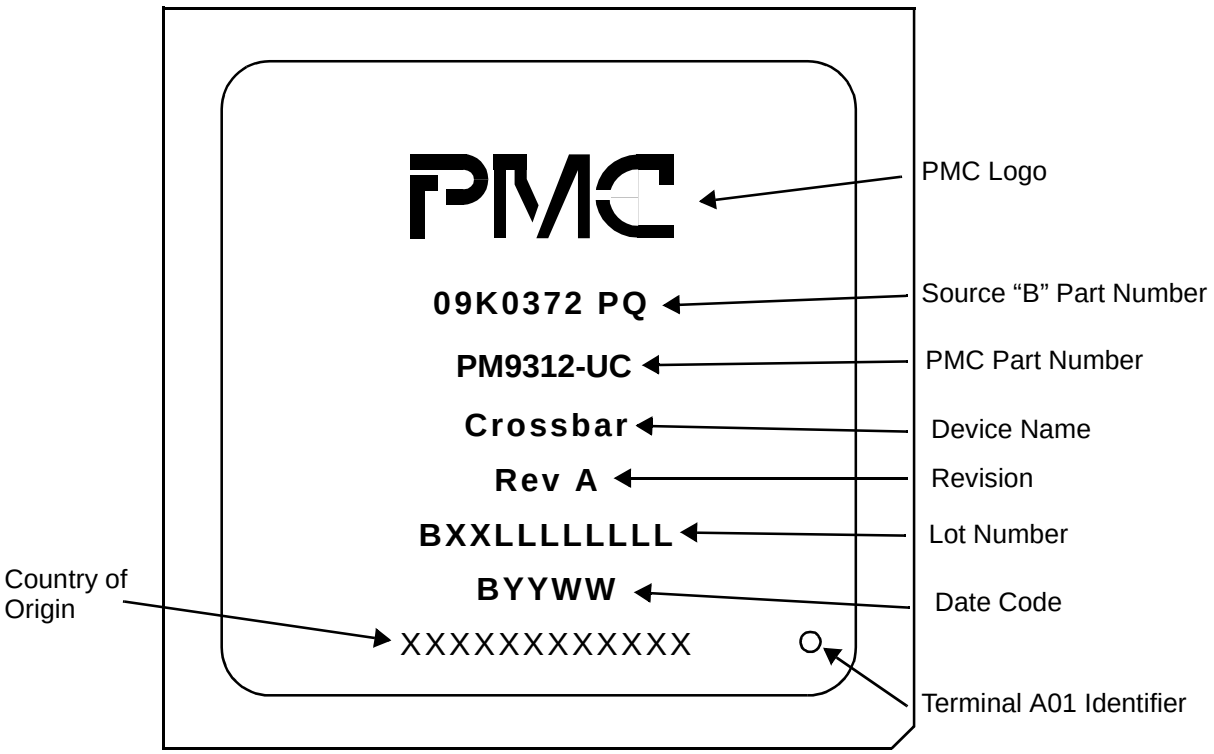
Symbol	e = 1.27 mm pitch			Units
	Minimum	Nominal	Maximum	
A		6.83		mm
A1		4.62		mm
A2		2.21		mm
aaa		0.15		mm
D		42.50		mm
D1		40.64		mm
E		42.50		mm
E1		40.64		mm
M		33 x 33		
N		1088		
b	0.48		0.52	mm

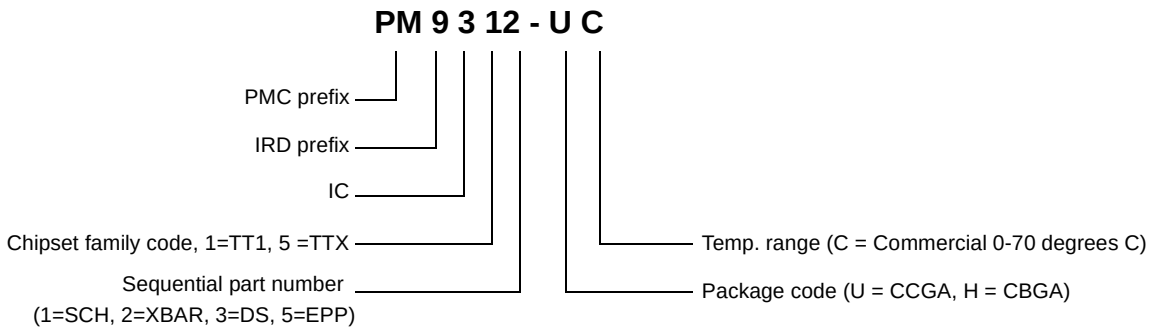
NOTE: Column alloy is 90/10 PbSn

4.6.3 Part Number

The PMC-Sierra Part Number for the Crossbar is:

Crossbar	PM9312-UC





5 Scheduler

This chapter contains information on the Scheduler device, part number PM9311-UC, available from PMC-Sierra, Inc.

The Scheduler communicates directly with all of the fabric ports. Its purpose is to arbitrate the flow of cells through the fabric from the input ports, through the fabric, to the required output ports. The Scheduler has information about all cells that are waiting in queues at the input ports. The output ports tell the Scheduler when they can accept new cells from the input ports. At every cell time, the Scheduler uses all of this information to arbitrate between all of the inputs to establish the set of cells that can be transferred through the crossbar fabric to the output queues. The effectiveness of the Scheduler at performing this arbitration determines the throughput of the fabric under any given cell load.

The Scheduler must have accurate, up to date information in order to arbitrate efficiently and fairly. Consequently, the EPPs send new state information to the Scheduler at every cell time.

5.1 BLOCK STRUCTURE

The block diagram of the Scheduler is shown in Figure 65 on page 255.

5.1.1 Port Block

The Scheduler can arbitrate between as many as 32 ports of OC-192c or 128 ports of OC-48c. The state associated with each port is maintained in a port block. Each port block contains a serial link, a port control unit, and some queue information memory. The serial link provides a full duplex, 1.6 Gbit/s, communications channel to the appropriate EPP. This channel is used to send new request and backpressure information to the Scheduler, and to send grant and routing tag information to the EPP.

The port control unit provides a set of registers that indicate the current status of many elements of the port block, and that can also be used to control the behaviour of the port block. It is described further in Section 5.2 “Port State”.

The port block also has some local memory which contains the number of outstanding requests for every queue associated with this port.

5.1.2 Arbiter Block

The arbiter receives information from each of the port blocks. This information indicates which of the virtual output queues in each port have cells waiting to be forwarded to their appropriate destination port. At every cell time, the arbiter combines these inputs to decide which ports can forward a cell at some fixed time in the future. The resulting set of grants is then passed back to the port blocks.

The arbiter has a number of control registers which are used primarily for diagnostics. Except for initial configuration after reset, the arbiter does not require any intervention from the OOB CPU.

5.1.3 OOB Interface and Control/Status Registers

All of the devices have an OOB interface. This interface allows a single local CPU to control and monitor all of the devices within a core fabric. Internally, each device provides registers that can be mapped into the CPU's address space. These registers are described in more detail in Section 5.4 "Scheduler Registers".

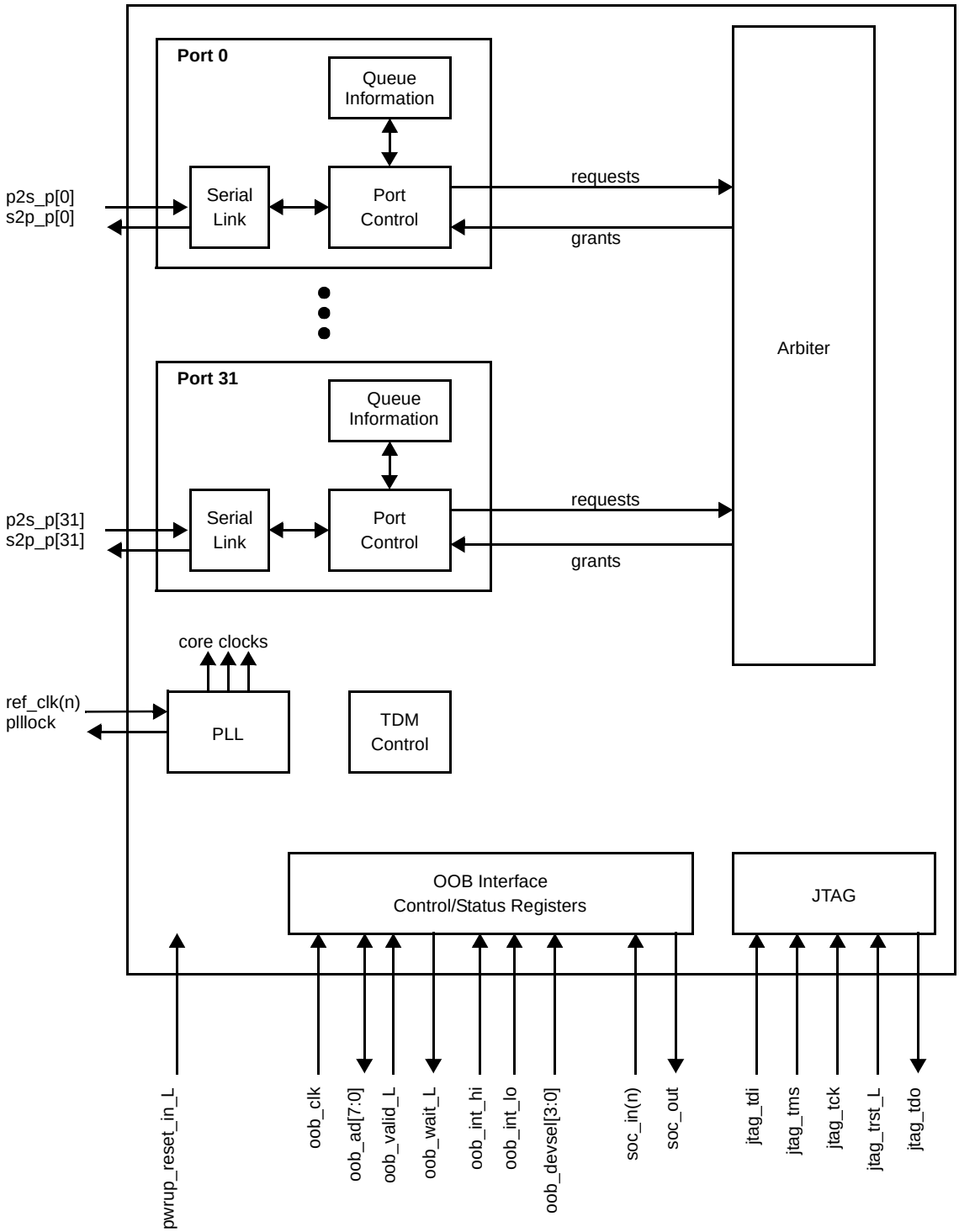
5.1.4 TDM Service

At every cell time the Scheduler may receive a TDM request from each port that intends to transfer a TDM cell (according to the TDM tables in the Port Processor). If a TDM request is received then the Scheduler will return a TDM Grant to that port, and that port will not be used in the arbitration of the best-effort traffic for that cell time.

The Scheduler may also receive notification that a port is expecting to receive a TDM cell. This notification identifies the port that will source the TDM cell. If the Scheduler receives a TDM request from the designated source port then the destination port will receive an appropriate routing tag. However, if the source port has not made a TDM request then the destination port will be eligible to receive a best-effort cell.

The Scheduler is the point of synchronization for the TDM service provided by the core fabric. The TDM control block generates synchronization signals at regular intervals. The period of these signals is determined either by internal registers, or by signals received from an EPP. The Scheduler sends the synchronization pulses to all 32 EPPs at the same time to indicate the start of a new TDM frame. See Section 1.2.5 "TDM Service" on page 22 for more information on the TDM service.

Figure 65. Scheduler Block Diagram

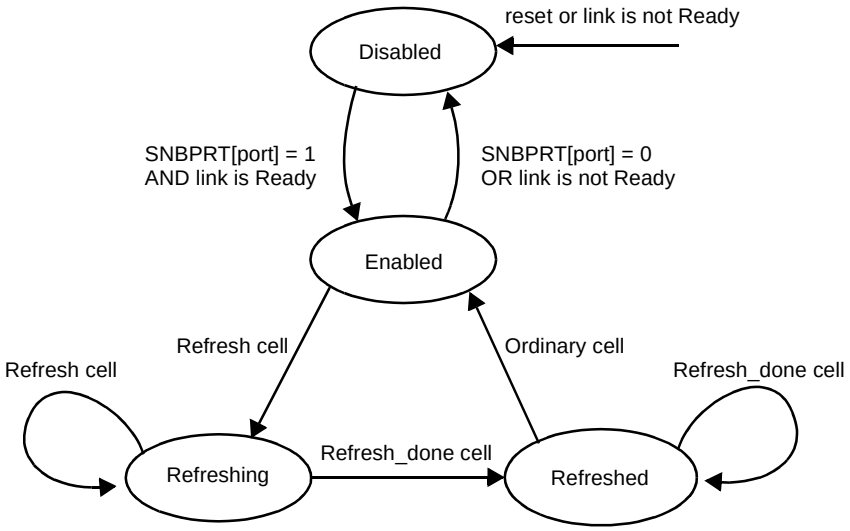


5.2 PORT STATE

The Scheduler controls all cell flow within the fabric for both normal, best-effort cells and TDM cells. The Scheduler can enable and disable cell flow on a port-by-port and class-of-service basis. When a new port is added to the system, it can be tested offline without affecting existing traffic flows.

Each Scheduler port has a very simple state machine, as shown in Figure 66. The port's current state determines the type of cells it can, and cannot, forward through the fabric.

Figure 66. Port State Machine



After reset, the port is in the *disabled* state. No cells will flow to or from this port when it is disabled.

Once the enable non-TDM traffic register for this port is set to 1, then the port is *enabled*. While the port is enabled, it receives requests from the EPP and issues requests to the arbiter. It receives grants from the arbiter and passes those grants to the EPP.

It also receives routing tags that it passes on to the EPP. At any time if errors are received on the appropriate serial link then the port will be returned to the disabled state and must be reset before it can be enabled.

The CPU initiates a Scheduler refresh by issuing a "Go Refresh" command to the EPP. The EPP then sends 384 Refresh cells to the Scheduler. The Scheduler port transitions to the *refreshing* state as soon as it receives the first of these Refresh cells. Once the EPP has sent all of the Refresh cells it then sends Refresh_Done cells, causing the Scheduler port to transition to the *refreshed* state. The CPU then terminates the refresh operation by issuing a "Go Scheduler" command to the EPP. This causes the EPP to send ordinary cells. As soon as the Scheduler port receives an ordinary cell it will transition back to the *enabled* state.

TDM traffic operates in two of the three active states *enabled* and *refreshing*, provided the corresponding bit in the enable_TDM_traffic register is set to 1. TDM traffic does not operate if the port is disabled. TDM traffic and non-TDM traffic are enabled independently.

NOTE: TDM traffic does *not* operate while the Scheduler is in the *refreshed* state. So the CPU should try to minimize the time that the Scheduler is in the *refreshed* state if TDM traffic is used.

Non-TDM traffic only operates in the *enabled* state. When the port is in *refreshing* or *refreshed* states, non-TDM traffic does not operate for this port: requests are not made and backpressure is asserted to prevent other ports from sending to this port.

Table 35. Non-TDM Traffic

Scheduler Port State	Enable non-TDM Register	Unicast Backpressure	Multicast Backpressure
Disabled	don't care	Asserted	De-asserted
Enabled	1 (non-TDM enabled)	Not used in ETT1	As per information from EPP
Enabled	0 (non-TDM disabled)	Asserted	Asserted
Refreshing	Don't care	Asserted	Asserted
Refreshed	Don't care	Asserted	Asserted

Table 35 shows whether non-TDM traffic can flow, based on the port state.

NOTE: Multicast backpressure is not asserted if the port is disabled. This is to prevent head of line blocking in the case where one port is sending a multicast cell to a port which is disabled.

See Section 1.3.4 "Multicast Traffic (subport mode)" on page 46 for more on the multicast blocking capability.

5.3 FAULT TOLERANCE

The Scheduler reacts to link faults in different ways, depending upon the mode it is in. The mode is set by the CRC_Action bit in the control register. This bit is set only if the Scheduler is the primary scheduler in a dual scheduler system, otherwise it is not set.

In both modes, if a Scheduler port detects a CRC error on the incoming link from the EPP, or the link goes down (the Ready line transitions from active to inactive for at least one clock cycle), then the port does the following:

- The Scheduler port is placed into the disabled state

  NOTE: The Enable Port register will not reflect this change.

- The appropriate register (CRC or Link Ready Inactive) is updated.

This port asserts backpressure to all other ports. Once the CPU determines that this port can continue operation, then a refresh operation must be performed on this port and then the port can be enabled.

If the CRC_Action bit is set, then in addition to the above actions, the Scheduler also asserts reset to all of its serial links to all EPPs. This turns off all of the links, forcing all of the EPPs to switch to the secondary Scheduler. All of the primary Scheduler's ports are set to the disabled state and the enable_port register is set to 0.

The Scheduler must be reset by the CPU and have its queue information refreshed before it is re-enabled.

5.4 SCHEDULER REGISTERS

The Scheduler device select low-order bits are hard wired on the board by four pins (oobdev_sel[3:0]) on the Scheduler package. See Section 1.7.1.3 “Individual Device Selects” on page 79 for more information.

5.4.1 Summary

The following table is a summary of information for all registers in the Scheduler. See the following Descriptions section for more information on individual registers.

Read and Clear means that reading the register causes it to be cleared (reset to zero)

Table 36. Scheduler Register Summary

Address	Access	Register	Symbol	Default Value
00000h	Read Only	Status	SSTS	00000000h
00004h	Read/Write	Control and Reset	SCTRLRS	00000130h
00008h	Read/Write	Low Priority Mask	SIRLMSK	00000000h
0000Ch	Read/Write	High Priority Mask	SIRHMSK	00000000h
00010h	Read and Clear	Interrupt Register	SIR	00000000h
00014h	Read/Write	CRC Error Interrupt Mask	SCRCMSK	00000000h
00018h	Read and Clear	CRC Errors	SCRC	00000000h
0001Ch	Read/Write	Link Ready Inactive Interrupt Mask	SRDYDMSK	00000000h
00020h	Read and Clear	Link Ready Inactive	SRDYDN	00000000h
00024h	Read/Write	Link Ready Active Interrupt Mask	SRDYUMSK	00000000h
00028h	Read and Clear	Link Ready Active	SRDYUP	00000000h
00030h	Read/Write	AIB Reset	SAIBRS	FFFFFFFFh
00034h	Read Only	AIB Ready	SAIBRDY	00000000h
00038h	Read/Write	Enable Port	SENBPRT	00000000h
0003C	Read/Write	AIB Tx Enable	SPORTEN	00000000h
00040h	Read/Write	Flow Control Crossbar Sync	FCCSYN	00000000h

Table 36. Scheduler Register Summary

Address	Access	Register	Symbol	Default Value
00080h	Read/Write	Enable non-TDM Traffic	SNTDMEN	00000000h
00084h	Read/Write	Enable TDM Traffic	STDMEN	00000000h
00088h	Read/Write	Reset Port	SPORTRS	00000000h
0008Ch	Read/Write	Freeze Port	SPORTFRZ	00000000h
00090h	Read Only	Port is Refreshing	SRFRSNG	00000000h
00094h	Read Only	Port is Refreshed	SRFRS	00000000h
00098h	Read Only	TDM Status	STDMSTS	00000000h
0009Ch	Read/Write	TDM Control	STDMCTRL	00000000h
00100h	Read/Write	PLL control/status	SPLL	0001447Ch

5.4.2 Scheduler Register Descriptions

Read and Clear means that reading the register causes it to be cleared (reset to zero).

All bits labled as Reserved should be set to 0.

5.4.2.1 Status

Symbol: SSTS

Address Offset: 00000h
Default Value: 30000000h
Access: Read Only

Status register.

Bits	Description
31:28	**Chip ID Number.** Identifies the specific device.
27:24	**Chip Revision Number.**
23:2	Reserved.
1	**High Priority Interrupt.** 1 = an outstanding high priority interrupt. One of the bits in the Interrupt Register is set, and is enabled via its corresponding high priority mask.
0	**Low Priority Interrupt.** 1 = an outstanding low priority interrupt. One of the bits in the Interrupt Register is set, and is enabled via its corresponding low priority mask.

5.4.2.2 Control and Reset

Symbol: SCTRLRS

Address Offset: 00004h
Default Value: 00000130h
Access: Read/Write

Control and Reset register.

Bits	Description
31-9	Reserved.
8-4	**BP_FIFO.** Specifies the depth of the backpressure FIFO. This is an internal FIFO and users should set these bits to a value of 14h after reset. The default value is 13h
3	Reserved.

Bits	Description (Continued)
2	**CRC Action.** This specifies what action is to be taken when an enabled port (port enable register bit is 1) encounters a CRC error from its attached Port Processor. If set to 0 then the port with the error is shut down and the Scheduler continues to arbitrate among the remaining ports. If set to 1 then the Scheduler will reset all serial links, effectively disabling all 32 ports. This bit should only be set to 1 for the primary Scheduler within a core with redundant Schedulers.
1	**Subport mode**. If set to 1 then the Scheduler operates in subport mode, in which each port is considered to be four subports with only one priority. This bit is not used in the ETT1 Scheduler and must be set to 0.
0	**Reset**. Writing a 1 to this location will reset the entire device. It is equivalent to a hardware reset. This register is cleared automatically when the chip is reset. This bit will always read as 0 and it is not necessary to write a 0 having just written a 1. Soft reset takes 1mS to complete.

5.4.2.3 Low Priority Mask

Symbol: SIRLMSK

Address Offset: 00008h
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:5	Reserved.
4:0	**Low Priority Mask**. Mask bits for low priority interrupts. Each mask bit is set to 1 to enable a low priority interrupt when the corresponding bit in the Status Register is 1.

5.4.2.4 High Priority Mask

Symbol: SIRHMSK

Address Offset: 0000Ch
Default Value: 00000000h
Access: Read/Write

Interrupt Mask for interrupts.

Bits	Description
31:5	Reserved.

Bits	Description
4:0	**High Priority Mask**. Mask bits for high priority interrupts. Each mask bit is set to 1 to enable a high priority interrupt when the corresponding bit in the Status Register is 1.

5.4.2.5 Interrupt Register

Symbol: SIR

Address Offset: 00010h
Default Value: 00000000h
Access: Read and Clear

Interrupt Register.

Bits	Description
31:5	Reserved.
4	**All Ports Refreshed.** During a refresh operation, this bit will be set when all enabled ports have been refreshed. See also Refreshed Status register. This bit is cleared on reading this register.
3	**TDM Sync Lost.** This bit is set if the TDM feature has been configured, and no Sync signal is received for three consecutive sync periods. This would suggest that a failure has occurred in either the Port Processor or its attached linecard. This bit is cleared on reading this register.
2	**Ready Inactive.** This is the logical OR of the Link Ready Inactive Interrupt register, after it has been masked by the Link Ready Inactive Interrupt Mask register. This bit is set if a serial link goes from active to inactive and the corresponding mask bit is 1. This bit is not cleared when read - the user must clear the Link Ready Inactive Interrupt register.
1	**Ready Active.** This is the logical OR of the Ready Active interrupt register, after it has been masked. This is set if a serial link goes from inactive to active and the corresponding mask bit is 1. This bit is not cleared when read - the user must clear the Ready Active interrupt register.
0	**CRC Error.** This is the logical OR of the CRC interrupt register, after it has been masked. This is set if a CRC error occurs and the corresponding mask bit is 1. This bit is not cleared when read - you must clear the CRC register.

5.4.2.6 CRC Error Interrupt Mask

Symbol: SCRCMSK

Address Offset: 00014h
Default Value: 00000000h
Access: Read/Write

Interrupt mask for CRC errors.

Bits	Description
31:0	**CRC Error Interrupt Mask.** Each bit is used to mask (enable) interrupts when the corresponding bit in the CRC Error register is set to 1. The interrupt is enabled when the bit is 1.

5.4.2.7 CRC Errors

Symbol: SCRC

Address Offset: 00018h
Default Value: 00000000h
Access: Read and Clear

CRC Interrupt Error Register

Bits	Description
31:0	**CRC Errors.** Each bit indicates if a CRC error has occurred on the appropriate link (Port Processor to Scheduler) since the last time this register was read. Mask off unwanted (unused ports) bits via the CRC Error Interrupt Mask.

5.4.2.8 Link Ready Inactive Interrupt Mask

Symbol: SRDYDMSK

Address Offset: 0001Ch
Default Value: 00000000h
Access: Read/Write

Enables interrupts when the corresponding bit in the Link Ready Inactive register is set to 1.

Bits	Description
31:0	**Link Ready Inactive Interrupt Mask.** Each bit is used to mask (enable) interrupts when the corresponding bit in the Link Ready Inactive register is set to 1. The interrupt is enabled when the bit is 1.

5.4.2.9 Link Ready Inactive

Symbol: SRDYDN

Address Offset: 00020h
Default Value: 00000000h
Access: Read and Clear

Each bit indicates if the Ready signal from a link has transitioned from Active to Inactive (the link has gone down for some reason).

NOTE: The link may be up again (active) by the time the CPU reads this register. Mask off unwanted (unused ports) bits via the Link Ready Inactive Interrupt Mask. Reading this register will clear all bits.

Bits	Description
31:0	**Link Ready Inactive.** Each bit indicates if the Ready signal from a link has transitioned from Active to Inactive.

5.4.2.10 Link Ready Active Interrupt Mask

Symbol: SRDYUMSK

Address Offset: 00024h
Default Value: 00000000h
Access: Read/Write

Enables interrupts when the corresponding bit in the Link Ready Active register is set to 1.

Bits	Description
31:0	**Link Ready Active Interrupt Mask.** Each bit is used to mask (enable) interrupts when the corresponding bit in the Link Ready Active register is set to 1. The interrupt is enabled when the bit is 1.

5.4.2.11 Link Ready Active

Symbol: SRDYUP

Address Offset: 00028h
Default Value: 00000000h
Access: Read and Clear

Each bit indicates if the Ready signal from a link has transitioned from Inactive to Active (the link has come up).

NOTE: The link may be down again (inactive) by the time the CPU reads this register. Mask off unwanted (unused ports) bits via the Link Ready Active Interrupt Mask. Reading this register will clear all bits.

Bits	Description
31:0	**Link Ready Active.** Each bit indicates if the Ready signal from a link has transitioned from Inactive to Active.

5.4.2.12 AIB Reset

Symbol: SAIBRS

Address Offset: 00030h
Default Value: FFFFFFFFh
Access: Read/Write

Used to assert reset to the AIB link for each port.

Bits	Description
31:0	**AIB Reset.** Each bit is used to assert reset to the AIB link for each port. Reset is asserted when the bit is 1. If reset the link will not operate or transition to ready. This register is set to 0xFFFF_FFFF on power-up reset or if the reset bit in the Control Register is asserted.

5.4.2.13 AIB Ready

Symbol: SAIBRDY

Address Offset: 00034h
Default Value: 00000000h
Access: Read Only

Indicates if the corresponding AIB link is ready.

Bits	Description
31:0	**AIB Ready.** Each bit indicates if the corresponding AIB link is ready (1 = active) or not (0 = inactive).

5.4.2.14 Enable Port

Symbol: SENBPRT

Address Offset: 00038h
Default Value: 00000000h
Access: Read/Write

This register is used to enable a scheduler port to carry traffic. After reset this register is 0 - all scheduler ports are disabled. While a scheduler port is disabled it will not respond to requests from the attached port card, nor will it issue grants.

Bits	Description
31:0	**Enable Port.** Each bit indicates if the corresponding port should be enabled.

NOTE: Reading this register only reflects the last value written to it. A CRC error, for example, might cause a Scheduler port to become inactive, but that change will not be reflected in this register. Similarly, if a port is active, then setting the port to inactive by writing to the Reset Port register will not be reflected in this register even though the port will be inactive. Whenever a port is reset the appropriate bit in this register should also be cleared.

5.4.2.15 AIB Tx Enable

Symbol: SPORTEN

Address Offset: 0003Ch
Default Value: 00000000h
Access: Read/Write

This is used to enable the transmitter of the corresponding AIB link.

Bits	Description
31:0	**AIB Tx Enable.** If the corresponding port is not physically present in the system then the appropriate bit should be set to zero to reduce unwanted electrical noise and to minimize unwanted effects when the port board is inserted

5.4.2.16 Flow Control Crossbar Sync

Symbol: FCCSYN

Address Offset: 00040h
Default Value: 00000000h
Access: Read/Write

NOTE: This register is only used in conjunction with the Enhanced Port Processor. It is not used with the Port Processor.

The counter is reloaded at reset or whenever a refresh_go occurs. Thus, the counter can be synchronized between two Scheduler devices. To force the counter to reload, set the Run/Stop bit to 0, write a new count value to the Synch_period, and then set the run/stop bit to 1. Dual Schedulers should always be refreshed (synchronized) after this register has been modified.

Bits	Description
31:16	**Synch_period**. This value denotes the period at which OC48 synch pulses will be issued. Setting this to the recommended value of 32 means that the OC48_synch bit will be set to 1 for every one in 32 frames sent to the Port Processor.
15:1	Reserved.
0	**Run/Stop.** This bit enables the OC-48 Synch counter. When 0 the counter does not run and is continuously reloaded with the value in the synch_period field. When this bit is set to 1 the counter counts down by one for each cell time. When the counter reaches the value one a synchronization pulse is issued in the scheduler to Port Processor frame on all ports (a bit is set in the frame).

5.4.2.17 Enable non-TDM Traffic

Symbol: SNTDMEN

Address Offset: 00080h
Default Value: 00000000h
Access: Read/Write

Indicates if the corresponding port can send and receive non-TDM traffic.

Bits	Description
31:0	**Enable non-TDM Traffic.** Each bit indicates if the corresponding port can send and receive non-TDM traffic (i.e. best-effort cells of priority 0,1,2 or 3). This bit should be de-asserted (set to 0) before telling the EPP to refresh the Scheduler's state.

5.4.2.18 Enable TDM Traffic

Symbol: STDMEN

Address Offset: 00084h
Default Value: 00000000h
Access: Read/Write

Indicates if the corresponding port can send and receive TDM traffic.

Bits	Description
31:0	**Enable TDM Traffic.** Each bit indicates if the corresponding port can send and receive TDM traffic (bit set to 1). Each bit should be 0 if the appropriate port does not send or receive TDM traffic.

5.4.2.19 Reset Port

Symbol: SPORTRS

Address Offset: 00088h
Default Value: 00000000h
Access: Read/Write

Resets each port.

Bits	Description
31:0	**Reset Port.** Each bit indicates if the corresponding port is held in reset (1). By asserting and then de-asserting this bit for a port, the internal state associated with that port is reset so that the port is in the *Disabled* state (see Section 5.2 on page 256). This register should not be left asserted. Do not use this register to keep ports in the inactive state. Instead, use the SENBPRT register to enable or disable ports.

5.4.2.20 Freeze Port

Symbol: SPORTFRZ

Address Offset: 0008Ch
Default Value: 00000000h
Access: Read/Write

Freezes Scheduler ports.

Bits	Description
31:1	Reserved.
0	**Freeze Port.** This is used for diagnostic purposes only! Asserting this bit will cause all ports to halt operation and to send a freeze command to their attached Port Processor. De-asserting this bit will allow traffic flow to be resumed. Caution: Cell loss may occur when modifying this bit.

5.4.2.21 Port is Refreshing

Symbol: SRFRSNG

Address Offset: 00090h
Default Value: 00000000h
Access: Read Only

Refreshing status.

Bits	Description
31:0	**Port is Refreshing.** During a refresh operation the port transitions from *active* to *refreshing* when it receives a refresh cell from the Port Processor. This bit is 1 if the port is currently refreshing (has received a refresh cell) but has not yet completed the refresh operation.

5.4.2.22 Port is Refreshed

Symbol: SRFRS

Address Offset: 00094h
Default Value: 00000000h
Access: Read Only

Refreshed status.

Bits	Description
31:0	**Port is Refreshed.** During a refresh operation the port transitions from *active* to *refreshing* when it receives a refresh cell from the Port Processor. Once all refresh cells have been received the port is considered to be “refreshed”. This bit is 1 if the port is currently refreshed - i.e. it has been refreshed but has not yet restarted normal operation. The refreshed bit in the Interrupt Register is set when all enabled ports have this bit set to one, and at least one port is enabled.

5.4.2.23 TDM Status

Symbol: STDMSTS

Address Offset: 00098h
Default Value: 00000000h
Access: Read Only

Indicates TDM status.

Bits	Description
31:17	Reserved.
16	Indicates the TDM frame (0 or 1) that was received in the most recent cell from the current port that is providing the suggested TDM sync.
15:11	Reserved.
10:0	The current period minus one at which TDM sync signals will be issued to the Port Processors. So if the current TDM Synch is being generated every 512 cell times then this field will be read as 511.

5.4.2.24 TDM Control

Symbol: STDMCTRL

Address Offset: 0009Ch

Default Value: 00000000h
Access: Read/Write

TDM Control Register

Bits	Description
31	**Enable TDM Sync.** Set this bit to 1 to enable TDM sync to be sent out to all the Port Processors. Default is 0 (TDM sync is off). This bit only controls the propagation of TDM Synch out of the Scheduler, and does not affect the internal TDM PLL.
30	**Load Initial Period.** Set this bit to 1 and then 0 to cause the Scheduler to use the Initial Period field in this register as the current TDM sync period. (Only used if the Scheduler generates TDM Synch itself)
29:27	Reserved.
26:16	**Initial Period.** This can be used in conjunction with the Load Initial Period bit to set an initial TDM period. Only use this if there is not a currently valid port supplying TDM sync and TDM frame. (Only used if the Scheduler generates TDM Synch itself). This value is the initial period minus one at which TDM sync signals will be issued to the Enhanced Port Processors
15:11	Reserved.
10:8	**FIFO Depth.** Leave this value set to zero.
7	**Transmitted TDM Frame** (0 or 1). This is the value of the TDM Table select that is sent to the PPs when bit 6 is 0. (Only used if the Scheduler generates TDM Synch itself)
6	**Current Port Valid.** Indicates that the Current Port field is valid, i.e. there is a port that is currently sending correct TDM sync and frame information to the Scheduler. This bit is 0 if the Scheduler generates TDM Synch itself.
5	Reserved.
4:0	**Current Port.** Specifies the port which is currently sending suggested TDM sync signal and TDM frame (which is echoed back out to all Port Processors). This field is irrelevant unless bit 6 is set. If these bits are modified then bit 6 should be set to 0 and then back to 1.

5.4.2.25 PLL Control/Status

Symbol: SPLL

Address Offset: 00100h
Default Value: 0001447Ch
Access: Read/Write

Controls operation of the internal PLL (Phase locked loop). After a power on reset the PLL itself is held in reset. This is reflected in bit 16 of this register. The local CPU must reset this bit to 0 to enable operation of the device, and thus should write 0000447Ch to this register.

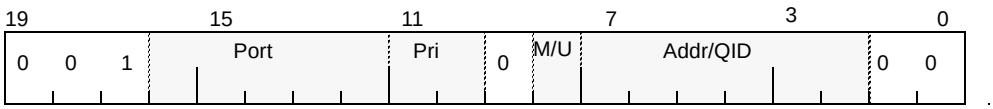
Bit	Description
31:17	**PLL status.** These bits reflect internal PLL operation status and should be ignored.
16	**Reset PLL.** When set to 1 the PLL is held reset. The supplied reference clock will be used as the internal clock. The serial links will not be operational. This bit will be 1 after power-up reset and should be deasserted for normal operation. PLL reset takes 10mS to complete.
15:0	**PLL control.**

5.4.3 Input Queue Memory (IQM)

The Scheduler has internal memory which stores the number of waiting cells at every input port. This memory can be read by the CPU. Caution: while the memory can also be modified, doing so may result in cell loss. CPU access to this memory is provided for diagnostic purposes only. After reset the memory is cleared to zero.

Each Scheduler port (0:31) has four IQM units, one for each best-effort priority. Each IQM is organized as two independent memories: the first is 64 words of 32 bits and stores multicast fanout information. The second is 32 words of 7 bits and stores the number of waiting cells at each unicast virtual output queue.

The individual entries in each IQM are addressed by the CPU as shown below:



Port is the port number 0:31.

Pri is the selected best effort priority 0:3.

M/U is 1 for the multicast fanout memory and 0 for the unicast request count memory.

Addr/QID is the multicast fanout address or a unicast VOQ identifier (only 0:31).

So, for example, unicast VOQ 10 contains the number of requests (cells) that are waiting to go from this port to output port 10.

The multicast fanout addresses are organized as a circular list. However the head of the list cannot be determined by the CPU except immediately after a refresh operation, at which time the second entry in the multicast fanout queue is at location zero. The first entry is held in a private register and is not visible to the CPU.

Although the IQM should not be modified during normal operation, it is prudent to verify the integrity of the memory after the Scheduler is reset.

5.5 SCHEDULER SIGNAL DESCRIPTIONS

This section describes the Scheduler signals.

The following notation is used to describe the signal type:

I Input pin

O Output pin

B Bidirectional Input/Output pin

The signal description also includes the type of buffer used for the particular signal:

PECL PECL are Pseudo-ECL (positive voltage ECL) compatible signals.

STI STI is a very high-speed asynchronous communications protocol used to connect devices that may be 1 to 15 meters apart.

HSTL All HSTL specifications are consistent with EIA/JEDEC Standard, EIA/JESD8-6 “High Speed Transceivers Logic (HSTL): A 1.5V Output Buffer Supply Voltage based Interface Standard for Digital Integrated Circuits”, dated 8/95. Refer to these specifications for more information**.**

CMOS The CMOS Buffers are either 3.3 V compatible or 2.5 V Low Voltage TTL compatible signals, as noted.

Table 37. Scheduler Signal Descriptions

Name	I/O	Type	Description
OOB Interface			
oob_ad[7:0]	B	CMOS	OOB Address bus
oob_clk	I	CMOS	OOB CLock
oob_devsel0	I	CMOS	OOB device select
oob_devsel1	I	CMOS	OOB device select
oob_devsel2	I	CMOS	OOB device select
oob_devsel3	I	CMOS	OOB device select
oob_int_hi	O	CMOS	OOB Interrupt
oob_int_lo	O	CMOS	OOB Interrupt

Table 37. Scheduler Signal Descriptions (Continued)

Name	I/O	Type	Description
oob_valid_L	I	CMOS	OOB Control
oob_wait_L	O	CMOS	OOB Control
AIB Interface			
p2s_p[31:0]_[c, cn, e, en, o, on]	I	STI	Port Processor to Scheduler AIB
s2p_p[31:0]_[c, cn, e, en, o, on]	O	STI	Scheduler to Port Processor AIB
Power Supply, Clock Source, Reset, And Diagnostics			
plllock	O	CMOS	Indicates if the PLL has locked (1)
pwrup_reset_in_L	I	CMOS	Synchronous, Active Low Reset
soc_out	O	CMOS	Buffered soc_in (NC)
VDDA		Isolated Supply	VDD (2.5 V) for PLL
VDD		Supply	VDD (2.5 V)
GND		Supply	GND (0 V)
ref_clk and ref_clkn	I	PECL	System 200MHz clock (differential)
soc_in and soc_inn	I	PECL	System Start-of-cell (differential)
JTAG Interface			
jtag_tck	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tdi	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tdo	O	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_tms	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
jtag_trst_L	I	CMOS_ 2.5_only	1149.1 JTAG 2.5V ONLY!
ASIC Manufacturing Test Interface			

Table 37. Scheduler Signal Descriptions (Continued)

Name	I/O	Type	Description
ce0_io	I	CMOS	Test (GND)
ce0_scan	I	CMOS	Test (GND)
ce0_testm3	I	CMOS	Test (GND)
lssd_ce1_a	I	CMOS	Test (VDD)
lssd_ce1_b	I	CMOS	Test (VDD)
lssd_ce1_c1	I	CMOS	Test (VDD)
lssd_ce1_c2	I	CMOS	Test (VDD)
lssd_ce1_c3	I	CMOS	Test (VDD)
lssd_scan_in[15:0]	I	CMOS	Scan input (GND)
lssd_scan_out[15:0]	O	CMOS	Scan output (NC)
mon[15:0]	O	CMOS	Test (NC)
plltest_in	I	CMOS	Test (GND)
plltest_out	O	CMOS	Test output (NC)
test_di1	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_di2	I	CMOS	Test (VDD) Should be driven to GND during reset. All outputs are tristated when low.
test_lt	I	CMOS	Test (VDD)
test_ri	I	CMOS	Test (VDD)

5.6 PINOUT AND PACKAGE INFORMATION

5.6.1 Pinout Tables

Table 38. Scheduler Pinout (left side)

	01	02	03	04	05	06	07	08	09	10	11
A	No Pin	pwrup_ reset_in_L	test_di2	mon4	ref_clk	p2s_p0_cn	jtag_trst_L	s2p_p0_en	p2s_p0_o	s2p_p1_e	s2p_p2_en
B	oob_ad0	GND	soc_out	VDD	mon5	GND	s2p_p1_o	VDD	p2s_p0_on	GND	p2s_p1_o
C	jtag_tms	oob_ad1	jtag_tck	soc_in	ref_clkn	s2p_p0_o	ce0_testm3	p2s_p1_c	s2p_p0_e	s2p_p1_en	p2s_p1_on
D	oob_wait_L	VDD	oob_ad2	GND	soc_inn	VDD	s2p_p0_on	GND	p2s_p1_cn	VDD	p2s_p2_cn
E	VDDA	oob_int_lo	plltest_out	oob_ad3	plllock	mon2	mon3	p2s_p0_c	s2p_p1_on	s2p_p2_o	p2s_p2_c
F	p2s_p16_c	GND	s2p_p16_on	VDD	oob_ad6	GND	mon1	VDD	s2p_p1_c	GND	s2p_p2_on
G	UNUSED	s2p_p17_on	plltest_in	s2p_p16_o	oob_ad7	oob_ad4	mon0	mon7	p2s_p0_e	p2s_p1_en	p2s_p2_e
H	s2p_p16_e	VDD	p2s_p17_cn	GND	p2s_p16_cn	VDD	oob_ad5	GND	mon6	VDD	s2p_p2_cn
J	p2s_p16_on	p2s_p16_o	s2p_p16_en	p2s_p17_c	s2p_p17_o	s2p_p17_c	p2s_p16_e	oob_clk	mon9	s2p_p0_cn	s2p_p1_cn
K	s2p_p17_en	GND	s2p_p17_e	VDD	s2p_p18_on	GND	s2p_p17_cn	VDD	oob_int_hi	GND	s2p_p0_c
L	s2p_p18_e	p2s_p17_on	p2s_p17_o	p2s_p18_c	p2s_p18_cn	s2p_p18_o	s2p_p18_cn	p2s_p17_en	p2s_p16_en	s2p_p16_c	mon8
M	p2s_p18_o	VDD	s2p_p18_en	GND	s2p_p19_on	VDD	s2p_p19_cn	GND	s2p_p18_c	VDD	s2p_p16_cn
N	s2p_p19_en	s2p_p19_e	p2s_p18_on	p2s_p19_c	p2s_p19_cn	s2p_p19_o	p2s_p19_en	p2s_p19_e	s2p_p19_c	p2s_p18_e	p2s_p17_e
P	p2s_p19_o	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	p2s_p18_en
R	UNUSED	p2s_p19_on	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
T	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	lssd_scan_ in15	GND	lssd_scan_ out15	VDD	UNUSED
U	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	oob_ devsel0	UNUSED	UNUSED
V	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	lssd_scan_ out14	GND	lssd_scan_ in14	VDD	UNUSED
W	UNUSED	s2p_p20_e	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
Y	s2p_p20_en	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	s2p_p21_o
AA	p2s_p20_o	p2s_p20_on	s2p_p21_e	s2p_p20_cn	s2p_p20_c	p2s_p20_en	s2p_p20_o	s2p_p20_on	p2s_p20_cn	s2p_p21_on	s2p_p22_on
AB	s2p_p21_en	VDD	p2s_p21_o	GND	p2s_p20_e	VDD	p2s_p20_c	GND	p2s_p21_cn	VDD	p2s_p23_c
AC	p2s_p21_on	s2p_p22_e	s2p_p22_en	s2p_p21_cn	s2p_p21_c	p2s_p21_en	p2s_p21_c	s2p_p22_o	s2p_p23_o	p2s_p23_cn	UNUSED
AD	p2s_p22_o	GND	p2s_p22_on	VDD	p2s_p21_e	GND	p2s_p22_c	VDD	UNUSED	GND	s2p_p24_cn
AE	s2p_p23_e	s2p_p23_en	p2s_p23_o	s2p_p22_cn	p2s_p22_e	p2s_p22_cn	s2p_p23_on	UNUSED	UNUSED	s2p_p24_c	s2p_p25_c
AF	p2s_p23_on	VDD	s2p_p22_c	GND	s2p_p23_c	VDD	UNUSED	GND	UNUSED	VDD	s2p_p26_c
AG	UNUSED	p2s_p22_en	lssd_scan_ out13	p2s_p23_en	UNUSED	UNUSED	UNUSED	UNUSED	p2s_p24_en	p2s_p25_e	p2s_p26_en
AH	s2p_p23_cn	GND	p2s_p23_e	VDD	UNUSED	GND	UNUSED	VDD	s2p_p25_cn	GND	s2p_p27_cn
AJ	UNUSED	UNUSED	lssd_scan_ in13	UNUSED	lssd_scan_ out12	UNUSED	UNUSED	p2s_p24_c	s2p_p25_o	s2p_p26_o	p2s_p26_cn
AK	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	p2s_p24_cn	GND	s2p_p26_on	VDD	p2s_p26_c
AL	lssd_scan_ in12	UNUSED	lssd_scan_ in11	UNUSED	lssd_scan_ in10	s2p_p24_o	lssd_scan_ in9	p2s_p25_c	s2p_p24_en	s2p_p25_e	p2s_p25_o
AM	UNUSED	GND	UNUSED	VDD	s2p_p24_on	GND	p2s_p25_cn	VDD	p2s_p24_o	GND	p2s_p25_on
AN	UNUSED	UNUSED	lssd_scan_ out11	UNUSED	lssd_scan_ out10	s2p_p25_on	lssd_scan_ out9	s2p_p24_e	p2s_p24_on	s2p_p25_en	s2p_p26_e

Table 39. Scheduler Pinout (center)

	12	13	14	15	16	17	18	19	20	21	22
A	p2s_p2_on	s2p_p3_e	p2s_p3_cn	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p4_e	p2s_p4_on	s2p_p5_e
B	VDD	s2p_p3_en	GND	p2s_p3_c	VDD	UNUSED	VDD	s2p_p4_en	GND	p2s_p4_o	VDD
C	s2p_p2_e	p2s_p2_o	p2s_p3_en	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p5_en	p2s_p5_on
D	GND	s2p_p3_cn	VDD	UNUSED	GND	UNUSED	GND	UNUSED	VDD	s2p_p4_c	GND
E	s2p_p3_o	s2p_p3_c	p2s_p3_e	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p4_cn	p2s_p4_en
F	VDD	s2p_p3_on	GND	UNUSED	VDD	UNUSED	VDD	UNUSED	GND	p2s_p4_e	VDD
G	p2s_p3_on	UNUSED	UNUSED	UNUSED	ce0_io	UNUSED	test_di1	UNUSED	UNUSED	UNUSED	s2p_p4_on
H	GND	p2s_p3_o	VDD	UNUSED	GND	UNUSED	GND	UNUSED	VDD	UNUSED	GND
J	s2p_p2_c	mon15	UNUSED	UNUSED	ce0_scan	UNUSED	test_ri	UNUSED	UNUSED	s2p_p4_o	p2s_p5_cn
K	VDD	p2s_p2_en	GND	UNUSED	VDD	UNUSED	VDD	UNUSED	GND	s2p_p5_o	VDD
L	p2s_p0_en	p2s_p1_e	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	p2s_p4_c	s2p_p6_on	s2p_p7_o
M	GND	mon11	VDD	mon12	GND	UNUSED	GND	UNUSED	VDD	UNUSED	GND
N	oob_valid_L	VDD	mon10	mon13	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	VDD	UNUSED
P	VDD	oob_devsel1	GND	mon14	VDD	GND	VDD	UNUSED	GND	UNUSED	VDD
R	oob_devsel2	UNUSED	oob_devsel3	GND	UNUSED	UNUSED	UNUSED	GND	UNUSED	UNUSED	UNUSED
T	GND	UNUSED	VDD	UNUSED	GND	VDD	GND	UNUSED	VDD	UNUSED	GND
U	UNUSED	UNUSED	GND	UNUSED	VDD	GND	VDD	UNUSED	GND	UNUSED	UNUSED
V	GND	UNUSED	VDD	UNUSED	GND	VDD	GND	UNUSED	VDD	UNUSED	GND
W	UNUSED	UNUSED	UNUSED	GND	UNUSED	UNUSED	UNUSED	GND	UNUSED	UNUSED	UNUSED
Y	VDD	UNUSED	GND	UNUSED	VDD	GND	VDD	UNUSED	GND	UNUSED	VDD
AA	UNUSED	VDD	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	VDD	UNUSED
AB	GND	UNUSED	VDD	UNUSED	GND	UNUSED	GND	UNUSED	VDD	UNUSED	GND
AC	p2s_p24_e	p2s_p25_en	s2p_p27_c	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	p2s_p28_cn	s2p_p30_o	s2p_p31_on
AD	VDD	p2s_p26_e	GND	UNUSED	VDD	UNUSED	VDD	UNUSED	GND	s2p_p29_on	VDD
AE	s2p_p26_cn	p2s_p27_e	UNUSED	UNUSED	lssd_scan_out8	UNUSED	lssd_scan_in7	UNUSED	UNUSED	s2p_p28_on	p2s_p29_c
AF	GND	UNUSED	VDD	UNUSED	GND	UNUSED	GND	UNUSED	VDD	UNUSED	GND
AG	p2s_p27_en	UNUSED	UNUSED	UNUSED	lssd_scan_in8	UNUSED	lssd_scan_out7	UNUSED	UNUSED	UNUSED	s2p_p28_o
AH	VDD	s2p_p27_o	GND	UNUSED	VDD	UNUSED	VDD	UNUSED	GND	p2s_p28_en	VDD
AJ	s2p_p27_on	p2s_p27_cn	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p28_c	p2s_p28_e
AK	GND	p2s_p27_c	VDD	UNUSED	GND	UNUSED	GND	UNUSED	VDD	s2p_p28_cn	GND
AL	s2p_p26_en	p2s_p26_on	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p29_e	p2s_p29_o
AM	VDD	s2p_p27_e	GND	p2s_p27_on	VDD	UNUSED	VDD	s2p_p28_e	GND	p2s_p28_on	VDD
AN	p2s_p26_o	s2p_p27_en	p2s_p27_o	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p28_en	p2s_p28_o	s2p_p29_en

Table 40. Scheduler Pinout (right side)

	23	24	25	26	27	28	29	30	31	32	33
A	p2s_p5_o	p2s_p6_on	s2p_p7_en	p2s_p7_o	lssd_ce1_a	p2s_p6_en	lssd_ce1_c1	UNUSED	lssd_ce1_c3	UNUSED	UNUSED
B	s2p_p6_en	GND	s2p_p7_e	VDD	s2p_p6_cn	GND	p2s_p7_en	VDD	UNUSED	GND	UNUSED
C	s2p_p6_e	p2s_p6_o	p2s_p7_on	s2p_p6_c	test_lt	p2s_p7_e	lssd_ce1_b	UNUSED	jtag_tdo	UNUSED	jtag_tdi
D	s2p_p5_c	VDD	p2s_p5_en	GND	s2p_p7_cn	VDD	UNUSED	GND	UNUSED	VDD	UNUSED
E	s2p_p5_cn	p2s_p5_e	p2s_p6_e	s2p_p7_c	UNUSED	UNUSED	lssd_ce1_c2	UNUSED	UNUSED	UNUSED	UNUSED
F	p2s_p4_cn	GND	p2s_p6_cn	VDD	UNUSED	GND	UNUSED	VDD	s2p_p8_o	GND	p2s_p8_cn
G	s2p_p5_on	s2p_p6_o	s2p_p7_on	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p8_on	UNUSED	s2p_p9_on	UNUSED
H	p2s_p5_c	VDD	UNUSED	GND	UNUSED	VDD	p2s_p8_c	GND	p2s_p9_c	VDD	s2p_p8_en
J	p2s_p6_c	p2s_p7_c	UNUSED	UNUSED	p2s_p8_en	s2p_p9_cn	s2p_p9_o	p2s_p9_cn	s2p_p8_e	p2s_p8_on	p2s_p8_o
K	p2s_p7_cn	GND	UNUSED	VDD	s2p_p9_c	GND	s2p_p10_o	VDD	s2p_p9_en	GND	s2p_p9_e
L	UNUSED	s2p_p8_cn	p2s_p8_e	p2s_p9_e	s2p_p10_c	s2p_p10_on	p2s_p10_c	p2s_p10_cn	p2s_p9_on	p2s_p9_o	s2p_p10_en
M	s2p_p8_c	VDD	s2p_p10_cn	GND	s2p_p11_c	VDD	s2p_p11_o	GND	s2p_p10_e	VDD	p2s_p10_on
N	p2s_p9_en	p2s_p10_en	s2p_p11_cn	p2s_p11_en	p2s_p11_e	s2p_p11_on	p2s_p11_c	p2s_p11_cn	p2s_p10_o	s2p_p11_en	s2p_p11_e
P	p2s_p10_e	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	p2s_p11_on
R	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	p2s_p11_o	UNUSED
T	UNUSED	VDD	lssd_scan_in0	GND	lssd_scan_out0	VDD	UNUSED	GND	UNUSED	VDD	UNUSED
U	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED
V	UNUSED	VDD	lssd_scan_out1	GND	lssd_scan_in1	VDD	UNUSED	GND	UNUSED	VDD	UNUSED
W	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	UNUSED	s2p_p12_en	UNUSED
Y	s2p_p13_on	GND	UNUSED	VDD	UNUSED	GND	UNUSED	VDD	UNUSED	GND	s2p_p12_e
AA	s2p_p14_o	s2p_p13_o	p2s_p12_c	s2p_p12_o	s2p_p12_on	p2s_p12_e	s2p_p12_cn	s2p_p12_c	s2p_p13_en	p2s_p12_o	p2s_p12_on
AB	p2s_p15_cn	VDD	p2s_p13_c	GND	p2s_p12_cn	VDD	p2s_p12_en	GND	p2s_p13_on	VDD	s2p_p13_e
AC	UNUSED	p2s_p15_c	s2p_p15_on	s2p_p14_on	p2s_p13_cn	p2s_p13_e	s2p_p13_cn	s2p_p13_c	s2p_p14_e	s2p_p14_en	p2s_p13_o
AD	p2s_p31_c	GND	UNUSED	VDD	p2s_p14_cn	GND	p2s_p13_en	VDD	p2s_p14_o	GND	p2s_p14_on
AE	p2s_p30_cn	p2s_p31_cn	UNUSED	UNUSED	s2p_p15_o	p2s_p14_c	p2s_p14_e	s2p_p14_c	p2s_p15_on	s2p_p15_e	s2p_p15_en
AF	p2s_p29_cn	VDD	UNUSED	GND	UNUSED	VDD	s2p_p15_cn	GND	s2p_p14_cn	VDD	p2s_p15_o
AG	s2p_p29_o	s2p_p30_on	s2p_p31_o	UNUSED	UNUSED	UNUSED	UNUSED	p2s_p15_e	lssd_scan_in2	p2s_p14_en	UNUSED
AH	p2s_p28_c	GND	p2s_p30_c	VDD	UNUSED	GND	UNUSED	VDD	p2s_p15_en	GND	s2p_p15_c
AJ	s2p_p29_c	p2s_p29_en	p2s_p30_en	s2p_p31_cn	UNUSED	UNUSED	lssd_scan_in3	UNUSED	lssd_scan_out2	UNUSED	UNUSED
AK	s2p_p29_cn	VDD	p2s_p29_e	GND	s2p_p31_c	VDD	UNUSED	GND	UNUSED	VDD	UNUSED
AL	s2p_p30_en	p2s_p30_on	p2s_p31_o	s2p_p30_cn	lssd_scan_out6	p2s_p31_en	lssd_scan_out5	UNUSED	lssd_scan_out4	UNUSED	lssd_scan_out3
AM	s2p_p30_e	GND	s2p_p31_en	VDD	s2p_p30_c	GND	p2s_p31_e	VDD	UNUSED	GND	UNUSED
AN	p2s_p29_on	p2s_p30_o	s2p_p31_e	p2s_p31_on	lssd_scan_in6	p2s_p30_e	lssd_scan_in5	UNUSED	lssd_scan_in4	UNUSED	UNUSED

Table 41. Scheduler Alpha Pin List

Signal Name	
ce0_io	G16
ce0_scan	J16
ce0_testm3	C07
GND	B02
GND	B06
GND	B10
GND	B14
GND	B20
GND	B24
GND	B28
GND	B32
GND	D04
GND	D08
GND	D12
GND	D16
GND	D18
GND	D22
GND	D26
GND	D30
GND	F02
GND	F06
GND	F10
GND	F14
GND	F20
GND	F24
GND	F28
GND	F32
GND	H04
GND	H08
GND	H12
GND	H16
GND	H18
GND	H22
GND	H26
GND	H30
GND	K02
GND	K06
GND	K10
GND	K14
GND	K20
GND	K24
GND	K28
GND	K32
GND	M04
GND	M08
GND	M12
GND	M16
GND	M18
GND	M22
GND	M26
GND	M30
GND	P02
GND	P06
GND	P10
GND	P14
GND	P17
GND	P20
GND	P24
GND	P28
GND	P32
GND	R15

Signal Name	
GND	R19
GND	T04
GND	T08
GND	T12
GND	T16
GND	T18
GND	T22
GND	T26
GND	T30
GND	U14
GND	U17
GND	U20
GND	V04
GND	V08
GND	V12
GND	V16
GND	V18
GND	V22
GND	V26
GND	V30
GND	W15
GND	W19
GND	Y02
GND	Y06
GND	Y10
GND	Y14
GND	Y17
GND	Y20
GND	Y24
GND	Y28
GND	Y32
GND	AB04
GND	AB08
GND	AB12
GND	AB16
GND	AB18
GND	AB22
GND	AB26
GND	AB30
GND	AD02
GND	AD06
GND	AD10
GND	AD14
GND	AD20
GND	AD24
GND	AD28
GND	AD32
GND	AF04
GND	AF08
GND	AF12
GND	AF16
GND	AF18
GND	AF22
GND	AF26
GND	AF30
GND	AH02
GND	AH06
GND	AH10
GND	AH14
GND	AH20
GND	AH24

Signal Name	
GND	AH28
GND	AH32
GND	AK04
GND	AK08
GND	AK12
GND	AK16
GND	AK18
GND	AK22
GND	AK26
GND	AK30
GND	AM02
GND	AM06
GND	AM10
GND	AM14
GND	AM20
GND	AM24
GND	AM28
GND	AM32
jtag_tck	C03
jtag_tdi	C33
jtag_tdo	C31
jtag_tms	C01
jtag_trst_L	A07
lssd_ce1_a	A27
lssd_ce1_b	C29
lssd_ce1_c1	A29
lssd_ce1_c2	E29
lssd_ce1_c3	A31
lssd_scan_in0	T25
lssd_scan_in1	V27
lssd_scan_in10	AL05
lssd_scan_in11	AL03
lssd_scan_in12	AL01
lssd_scan_in13	AJ03
lssd_scan_in14	V09
lssd_scan_in15	T07
lssd_scan_in2	AG31
lssd_scan_in3	AJ29
lssd_scan_in4	AN31
lssd_scan_in5	AN29
lssd_scan_in6	AN27
lssd_scan_in7	AE18
lssd_scan_in8	AG16
lssd_scan_in9	AL07
lssd_scan_out0	T27
lssd_scan_out1	V25
lssd_scan_out10	AN05
lssd_scan_out11	AN03
lssd_scan_out12	AJ05
lssd_scan_out13	AG03
lssd_scan_out14	V07
lssd_scan_out15	T09
lssd_scan_out2	AJ31
lssd_scan_out3	AL33
lssd_scan_out4	AL31
lssd_scan_out5	AL29
lssd_scan_out6	AL27
lssd_scan_out7	AG18
lssd_scan_out8	AE16
lssd_scan_out9	AN07
mon0	G07

Signal Name	
mon1	F07
mon10	N14
mon11	M13
mon12	M15
mon13	N15
mon14	P15
mon15	J13
mon2	E06
mon3	E07
mon4	A04
mon5	B05
mon6	H09
mon7	G08
mon8	L11
mon9	J09
No Pin	A01
oob_ad0	B01
oob_ad1	C02
oob_ad2	D03
oob_ad3	E04
oob_ad4	G06
oob_ad5	H07
oob_ad6	F05
oob_ad7	G05
oob_clk	J08
oob_devsel0	U09
oob_devsel1	P13
oob_devsel2	R12
oob_devsel3	R14
oob_int_hi	K09
oob_int_lo	E02
oob_valid_L	N12
oob_wait_L	D01
p2s_p0_c	E08
p2s_p0_cn	A06
p2s_p0_e	G09
p2s_p0_en	L12
p2s_p0_o	A09
p2s_p0_on	B09
p2s_p10_c	L29
p2s_p10_cn	L30
p2s_p10_e	P23
p2s_p10_en	N24
p2s_p10_o	N31
p2s_p10_on	M33
p2s_p11_c	N29
p2s_p11_cn	N30
p2s_p11_e	N27
p2s_p11_en	N26
p2s_p11_o	R32
p2s_p11_on	P33
p2s_p12_c	AA25
p2s_p12_cn	AB27
p2s_p12_e	AA28
p2s_p12_en	AB29
p2s_p12_o	AA32
p2s_p12_on	AA33
p2s_p13_c	AB25
p2s_p13_cn	AC27
p2s_p13_e	AC28
p2s_p13_en	AD29

Signal Name	
p2s_p13_o	AC33
p2s_p13_on	AB31
p2s_p14_c	AE28
p2s_p14_cn	AD27
p2s_p14_e	AE29
p2s_p14_en	AG32
p2s_p14_o	AD31
p2s_p14_on	AD33
p2s_p15_c	AC24
p2s_p15_cn	AB23
p2s_p15_e	AG30
p2s_p15_en	AH31
p2s_p15_o	AF33
p2s_p15_on	AE31
p2s_p16_c	F01
p2s_p16_cn	H05
p2s_p16_e	J07
p2s_p16_en	L09
p2s_p16_o	J02
p2s_p16_on	J01
p2s_p17_c	J04
p2s_p17_cn	H03
p2s_p17_e	N11
p2s_p17_en	L08
p2s_p17_o	L03
p2s_p17_on	L02
p2s_p18_c	L04
p2s_p18_cn	L05
p2s_p18_e	N10
p2s_p18_en	P11
p2s_p18_o	M01
p2s_p18_on	N03
p2s_p19_c	N04
p2s_p19_cn	N05
p2s_p19_e	N08
p2s_p19_en	N07
p2s_p19_o	P01
p2s_p19_on	R02
p2s_p1_c	C08
p2s_p1_cn	D09
p2s_p1_e	L13
p2s_p1_en	G10
p2s_p1_o	B11
p2s_p1_on	C11
p2s_p20_c	AB07
p2s_p20_cn	AA09
p2s_p20_e	AB05
p2s_p20_en	AA06
p2s_p20_o	AA01
p2s_p20_on	AA02
p2s_p21_c	AC07
p2s_p21_cn	AB09
p2s_p21_e	AD05
p2s_p21_en	AC06
p2s_p21_o	AB03
p2s_p21_on	AC01
p2s_p22_c	AD07
p2s_p22_cn	AE06
p2s_p22_e	AE05
p2s_p22_en	AG02
p2s_p22_o	AD01

Signal Name	
p2s_p22_on	AD03
p2s_p23_c	AB11
p2s_p23_cn	AC10
p2s_p23_e	AH03
p2s_p23_en	AG04
p2s_p23_o	AE03
p2s_p23_on	AF01
p2s_p24_c	AJ08
p2s_p24_cn	AK07
p2s_p24_e	AC12
p2s_p24_en	AG09
p2s_p24_o	AM09
p2s_p24_on	AN09
p2s_p25_c	AL08
p2s_p25_cn	AM07
p2s_p25_e	AG10
p2s_p25_en	AC13
p2s_p25_o	AL11
p2s_p25_on	AM11
p2s_p26_c	AK11
p2s_p26_cn	AJ11
p2s_p26_e	AD13
p2s_p26_en	AG11
p2s_p26_o	AN12
p2s_p26_on	AL13
p2s_p27_c	AK13
p2s_p27_cn	AJ13
p2s_p27_e	AE13
p2s_p27_en	AG12
p2s_p27_o	AN14
p2s_p27_on	AM15
p2s_p28_c	AH23
p2s_p28_cn	AC20
p2s_p28_e	AJ22
p2s_p28_en	AH21
p2s_p28_o	AN21
p2s_p28_on	AM21
p2s_p29_c	AE22
p2s_p29_cn	AF23
p2s_p29_e	AK25
p2s_p29_en	AJ24
p2s_p29_o	AL22
p2s_p29_on	AN23
p2s_p2_c	E11
p2s_p2_cn	D11
p2s_p2_e	G11
p2s_p2_en	K13
p2s_p2_o	C13
p2s_p2_on	A12
p2s_p30_c	AH25
p2s_p30_cn	AE23
p2s_p30_e	AN28
p2s_p30_en	AJ25
p2s_p30_o	AN24
p2s_p30_on	AL24
p2s_p31_c	AD23
p2s_p31_cn	AE24
p2s_p31_e	AM29
p2s_p31_en	AL28
p2s_p31_o	AL25
p2s_p31_on	AN26

Signal Name	
p2s_p3_c	B15
p2s_p3_cn	A14
p2s_p3_e	E14
p2s_p3_en	C14
p2s_p3_o	H13
p2s_p3_on	G12
p2s_p4_c	L20
p2s_p4_cn	F23
p2s_p4_e	F21
p2s_p4_en	E22
p2s_p4_o	B21
p2s_p4_on	A21
p2s_p5_c	H23
p2s_p5_cn	J22
p2s_p5_e	E24
p2s_p5_en	D25
p2s_p5_o	A23
p2s_p5_on	C22
p2s_p6_c	J23
p2s_p6_cn	F25
p2s_p6_e	E25
p2s_p6_en	A28
p2s_p6_o	C24
p2s_p6_on	A24
p2s_p7_c	J24
p2s_p7_cn	K23
p2s_p7_e	C28
p2s_p7_en	B29
p2s_p7_o	A26
p2s_p7_on	C25
p2s_p8_c	H29
p2s_p8_cn	F33
p2s_p8_e	L25
p2s_p8_en	J27
p2s_p8_o	J33
p2s_p8_on	J32
p2s_p9_c	H31
p2s_p9_cn	J30
p2s_p9_e	L26
p2s_p9_en	N23
p2s_p9_o	L32
p2s_p9_on	L31
plllock	E05
plltest_in	G03
plltest_out	E03
pwrup_reset_in_L	A02
ref_clk	A05
ref_clkn	C05
s2p_p0_c	K11
s2p_p0_cn	J10
s2p_p0_e	C09
s2p_p0_en	A08
s2p_p0_o	C06
s2p_p0_on	D07
s2p_p10_c	L27
s2p_p10_cn	M25
s2p_p10_e	M31
s2p_p10_en	L33
s2p_p10_o	K29
s2p_p10_on	L28
s2p_p11_c	M27

Signal Name	
s2p_p11_cn	N25
s2p_p11_e	N33
s2p_p11_en	N32
s2p_p11_o	M29
s2p_p11_on	N28
s2p_p12_c	AA30
s2p_p12_cn	AA29
s2p_p12_e	Y33
s2p_p12_en	W32
s2p_p12_o	AA26
s2p_p12_on	AA27
s2p_p13_c	AC30
s2p_p13_cn	AC29
s2p_p13_e	AB33
s2p_p13_en	AA31
s2p_p13_o	AA24
s2p_p13_on	Y23
s2p_p14_c	AE30
s2p_p14_cn	AF31
s2p_p14_e	AC31
s2p_p14_en	AC32
s2p_p14_o	AA23
s2p_p14_on	AC26
s2p_p15_c	AH33
s2p_p15_cn	AF29
s2p_p15_e	AE32
s2p_p15_en	AE33
s2p_p15_o	AE27
s2p_p15_on	AC25
s2p_p16_c	L10
s2p_p16_cn	M11
s2p_p16_e	H01
s2p_p16_en	J03
s2p_p16_o	G04
s2p_p16_on	F03
s2p_p17_c	J06
s2p_p17_cn	K07
s2p_p17_e	K03
s2p_p17_en	K01
s2p_p17_o	J05
s2p_p17_on	G02
s2p_p18_c	M09
s2p_p18_cn	L07
s2p_p18_e	L01
s2p_p18_en	M03
s2p_p18_o	L06
s2p_p18_on	K05
s2p_p19_c	N09
s2p_p19_cn	M07
s2p_p19_e	N02
s2p_p19_en	N01
s2p_p19_o	N06
s2p_p19_on	M05
s2p_p1_c	F09
s2p_p1_cn	J11
s2p_p1_e	A10
s2p_p1_en	C10
s2p_p1_o	B07
s2p_p1_on	E09
s2p_p20_c	AA05
s2p_p20_cn	AA04

Signal Name	
s2p_p20_e	W02
s2p_p20_en	Y01
s2p_p20_o	AA07
s2p_p20_on	AA08
s2p_p21_c	AC05
s2p_p21_cn	AC04
s2p_p21_e	AA03
s2p_p21_en	AB01
s2p_p21_o	Y11
s2p_p21_on	AA10
s2p_p22_c	AF03
s2p_p22_cn	AE04
s2p_p22_e	AC02
s2p_p22_en	AC03
s2p_p22_o	AC08
s2p_p22_on	AA11
s2p_p23_c	AF05
s2p_p23_cn	AH01
s2p_p23_e	AE01
s2p_p23_en	AE02
s2p_p23_o	AC09
s2p_p23_on	AE07
s2p_p24_c	AE10
s2p_p24_cn	AD11
s2p_p24_e	AN08
s2p_p24_en	AL09
s2p_p24_o	AL06
s2p_p24_on	AM05
s2p_p25_c	AE11
s2p_p25_cn	AH09
s2p_p25_e	AL10
s2p_p25_en	AN10
s2p_p25_o	AJ09
s2p_p25_on	AN06
s2p_p26_c	AF11
s2p_p26_cn	AE12
s2p_p26_e	AN11
s2p_p26_en	AL12
s2p_p26_o	AJ10
s2p_p26_on	AK09
s2p_p27_c	AC14
s2p_p27_cn	AH11
s2p_p27_e	AM13
s2p_p27_en	AN13
s2p_p27_o	AH13
s2p_p27_on	AJ12
s2p_p28_c	AJ21
s2p_p28_cn	AK21
s2p_p28_e	AM19
s2p_p28_en	AN20
s2p_p28_o	AG22
s2p_p28_on	AE21
s2p_p29_c	AJ23
s2p_p29_cn	AK23
s2p_p29_e	AL21
s2p_p29_en	AN22
s2p_p29_o	AG23
s2p_p29_on	AD21
s2p_p2_c	J12
s2p_p2_cn	H11
s2p_p2_e	C12

Signal Name	
s2p_p2_en	A11
s2p_p2_o	E10
s2p_p2_on	F11
s2p_p30_c	AM27
s2p_p30_cn	AL26
s2p_p30_e	AM23
s2p_p30_en	AL23
s2p_p30_o	AC21
s2p_p30_on	AG24
s2p_p31_c	AK27
s2p_p31_cn	AJ26
s2p_p31_e	AN25
s2p_p31_en	AM25
s2p_p31_o	AG25
s2p_p31_on	AC22
s2p_p3_c	E13
s2p_p3_cn	D13
s2p_p3_e	A13
s2p_p3_en	B13
s2p_p3_o	E12
s2p_p3_on	F13
s2p_p4_c	D21
s2p_p4_cn	E21
s2p_p4_e	A20
s2p_p4_en	B19
s2p_p4_o	J21
s2p_p4_on	G22
s2p_p5_c	D23
s2p_p5_cn	E23
s2p_p5_e	A22
s2p_p5_en	C21
s2p_p5_o	K21
s2p_p5_on	G23
s2p_p6_c	C26
s2p_p6_cn	B27
s2p_p6_e	C23
s2p_p6_en	B23
s2p_p6_o	G24
s2p_p6_on	L21
s2p_p7_c	E26
s2p_p7_cn	D27
s2p_p7_e	B25
s2p_p7_en	A25
s2p_p7_o	L22
s2p_p7_on	G25
s2p_p8_c	M23
s2p_p8_cn	L24
s2p_p8_e	J31
s2p_p8_en	H33
s2p_p8_o	F31
s2p_p8_on	G30
s2p_p9_c	K27
s2p_p9_cn	J28
s2p_p9_e	K33
s2p_p9_en	K31
s2p_p9_o	J29
s2p_p9_on	G32
soc_in	C04
soc_inn	D05
soc_out	B03
test_di1	G18

Signal Name	
test_di2	A03
test_lt	C27
test_ri	J18
UNUSED	A15
UNUSED	A16
UNUSED	A17
UNUSED	A18
UNUSED	A19
UNUSED	A30
UNUSED	A32
UNUSED	A33
UNUSED	B17
UNUSED	B31
UNUSED	B33
UNUSED	C15
UNUSED	C16
UNUSED	C17
UNUSED	C18
UNUSED	C19
UNUSED	C20
UNUSED	C30
UNUSED	C32
UNUSED	D15
UNUSED	D17
UNUSED	D19
UNUSED	D29
UNUSED	D31
UNUSED	D33
UNUSED	E15
UNUSED	E16
UNUSED	E17
UNUSED	E18
UNUSED	E19
UNUSED	E20
UNUSED	E27
UNUSED	E28
UNUSED	E30
UNUSED	E31
UNUSED	E32
UNUSED	E33
UNUSED	F15
UNUSED	F17
UNUSED	F19
UNUSED	F27
UNUSED	F29
UNUSED	G01
UNUSED	G13
UNUSED	G14
UNUSED	G15
UNUSED	G17
UNUSED	G19
UNUSED	G20
UNUSED	G21
UNUSED	G26
UNUSED	G27
UNUSED	G28
UNUSED	G29
UNUSED	G31
UNUSED	G33
UNUSED	H15
UNUSED	H17

Signal Name	
UNUSED	H19
UNUSED	H21
UNUSED	H25
UNUSED	H27
UNUSED	J14
UNUSED	J15
UNUSED	J17
UNUSED	J19
UNUSED	J20
UNUSED	J25
UNUSED	J26
UNUSED	K15
UNUSED	K17
UNUSED	K19
UNUSED	K25
UNUSED	L14
UNUSED	L15
UNUSED	L16
UNUSED	L17
UNUSED	L18
UNUSED	L19
UNUSED	L23
UNUSED	M17
UNUSED	M19
UNUSED	M21
UNUSED	N16
UNUSED	N17
UNUSED	N18
UNUSED	N19
UNUSED	N20
UNUSED	N22
UNUSED	P03
UNUSED	P05
UNUSED	P07
UNUSED	P09
UNUSED	P19
UNUSED	P21
UNUSED	P25
UNUSED	P27
UNUSED	P29
UNUSED	P31
UNUSED	R01
UNUSED	R03
UNUSED	R04
UNUSED	R05
UNUSED	R06
UNUSED	R07
UNUSED	R08
UNUSED	R09
UNUSED	R10
UNUSED	R11
UNUSED	R13
UNUSED	R16
UNUSED	R17
UNUSED	R18
UNUSED	R20
UNUSED	R21
UNUSED	R22
UNUSED	R23
UNUSED	R24
UNUSED	R25

Signal Name	
UNUSED	R26
UNUSED	R27
UNUSED	R28
UNUSED	R29
UNUSED	R30
UNUSED	R31
UNUSED	R33
UNUSED	T01
UNUSED	T03
UNUSED	T05
UNUSED	T11
UNUSED	T13
UNUSED	T15
UNUSED	T19
UNUSED	T21
UNUSED	T23
UNUSED	T29
UNUSED	T31
UNUSED	T33
UNUSED	U01
UNUSED	U02
UNUSED	U03
UNUSED	U04
UNUSED	U05
UNUSED	U06
UNUSED	U07
UNUSED	U08
UNUSED	U10
UNUSED	U11
UNUSED	U12
UNUSED	U13
UNUSED	U15
UNUSED	U19
UNUSED	U21
UNUSED	U22
UNUSED	U23
UNUSED	U24
UNUSED	U25
UNUSED	U26
UNUSED	U27
UNUSED	U28
UNUSED	U29
UNUSED	U30
UNUSED	U31
UNUSED	U32
UNUSED	U33
UNUSED	V01
UNUSED	V03
UNUSED	V05
UNUSED	V11
UNUSED	V13
UNUSED	V15
UNUSED	V19
UNUSED	V21
UNUSED	V23
UNUSED	V29
UNUSED	V31
UNUSED	V33
UNUSED	W01
UNUSED	W03
UNUSED	W04

Signal Name	
UNUSED	W05
UNUSED	W06
UNUSED	W07
UNUSED	W08
UNUSED	W09
UNUSED	W10
UNUSED	W11
UNUSED	W12
UNUSED	W13
UNUSED	W14
UNUSED	W16
UNUSED	W17
UNUSED	W18
UNUSED	W20
UNUSED	W21
UNUSED	W22
UNUSED	W23
UNUSED	W24
UNUSED	W25
UNUSED	W26
UNUSED	W27
UNUSED	W28
UNUSED	W29
UNUSED	W30
UNUSED	W31
UNUSED	W33
UNUSED	Y03
UNUSED	Y05
UNUSED	Y07
UNUSED	Y09
UNUSED	Y13
UNUSED	Y15
UNUSED	Y19
UNUSED	Y21
UNUSED	Y25
UNUSED	Y27
UNUSED	Y29
UNUSED	Y31
UNUSED	AA12
UNUSED	AA14
UNUSED	AA15
UNUSED	AA16
UNUSED	AA17
UNUSED	AA18
UNUSED	AA19
UNUSED	AA20
UNUSED	AA22
UNUSED	AB13
UNUSED	AB15
UNUSED	AB17
UNUSED	AB19
UNUSED	AB21
UNUSED	AC11
UNUSED	AC15
UNUSED	AC16
UNUSED	AC17
UNUSED	AC18
UNUSED	AC19
UNUSED	AC23
UNUSED	AD09
UNUSED	AD15

Signal Name	
UNUSED	AD17
UNUSED	AD19
UNUSED	AD25
UNUSED	AE08
UNUSED	AE09
UNUSED	AE14
UNUSED	AE15
UNUSED	AE17
UNUSED	AE19
UNUSED	AE20
UNUSED	AE25
UNUSED	AE26
UNUSED	AF07
UNUSED	AF09
UNUSED	AF13
UNUSED	AF15
UNUSED	AF17
UNUSED	AF19
UNUSED	AF21
UNUSED	AF25
UNUSED	AF27
UNUSED	AG01
UNUSED	AG05
UNUSED	AG06
UNUSED	AG07
UNUSED	AG08
UNUSED	AG13
UNUSED	AG14
UNUSED	AG15
UNUSED	AG17
UNUSED	AG19
UNUSED	AG20
UNUSED	AG21
UNUSED	AG26
UNUSED	AG27
UNUSED	AG28
UNUSED	AG29
UNUSED	AG33
UNUSED	AH05
UNUSED	AH07
UNUSED	AH15
UNUSED	AH17
UNUSED	AH19
UNUSED	AH27
UNUSED	AH29
UNUSED	AJ01
UNUSED	AJ02
UNUSED	AJ04
UNUSED	AJ06
UNUSED	AJ07
UNUSED	AJ14
UNUSED	AJ15
UNUSED	AJ16
UNUSED	AJ17
UNUSED	AJ18
UNUSED	AJ19
UNUSED	AJ20
UNUSED	AJ27
UNUSED	AJ28
UNUSED	AJ30
UNUSED	AJ32

Signal Name	
UNUSED	AJ33
UNUSED	AK01
UNUSED	AK03
UNUSED	AK05
UNUSED	AK15
UNUSED	AK17
UNUSED	AK19
UNUSED	AK29
UNUSED	AK31
UNUSED	AK33
UNUSED	AL02
UNUSED	AL04
UNUSED	AL14
UNUSED	AL15
UNUSED	AL16
UNUSED	AL17
UNUSED	AL18
UNUSED	AL19
UNUSED	AL20
UNUSED	AL30
UNUSED	AL32
UNUSED	AM01
UNUSED	AM03
UNUSED	AM17
UNUSED	AM31
UNUSED	AM33
UNUSED	AN01
UNUSED	AN02
UNUSED	AN04
UNUSED	AN15
UNUSED	AN16
UNUSED	AN17
UNUSED	AN18
UNUSED	AN19
UNUSED	AN30
UNUSED	AN32
UNUSED	AN33
VDD	B04
VDD	B12
VDD	B18
VDD	B26
VDD	D10
VDD	D20
VDD	D28
VDD	D32
VDD	F04
VDD	F08
VDD	F16
VDD	F22
VDD	H02
VDD	H14
VDD	H24
VDD	H28
VDD	K08
VDD	K12
VDD	K18
VDD	K30
VDD	M06
VDD	M24
VDD	M32
VDD	N13

Signal Name	
VDD	N21
VDD	P04
VDD	P16
VDD	P18
VDD	P26
VDD	T02
VDD	T10
VDD	T14
VDD	T17
VDD	T20
VDD	T28
VDD	U16
VDD	U18
VDD	V06
VDD	V14
VDD	V17
VDD	V20
VDD	V24
VDD	V32
VDD	Y08
VDD	Y16
VDD	Y18
VDD	Y30
VDD	AA13
VDD	AA21
VDD	AB02
VDD	AB10
VDD	AB28
VDD	AD04
VDD	AD16
VDD	AD22
VDD	AD26
VDD	AF06
VDD	AF10
VDD	AF20
VDD	AF32
VDD	AH12
VDD	AH18
VDD	AH26
VDD	AH30
VDD	AK02
VDD	AK06
VDD	AK14
VDD	AK24
VDD	AM08
VDD	AM16
VDD	AM22
VDD	AM30
VDD	F30
VDD	H32
VDD	K26
VDD	M28
VDD	P22
VDD	P30
VDD	T24
VDD	T32
VDD	V28
VDD	Y22
VDD	Y26
VDD	AB24
VDD	AB32

Signal Name	
VDD	AD30
VDD	AF28
VDD	AK32
VDD	AB14
VDD	AB20
VDD	AD12
VDD	AD18
VDD	AF14
VDD	AF24
VDD	AH08
VDD	AH16
VDD	AH22
VDD	AK10
VDD	AK20
VDD	AK28
VDD	AM04
VDD	AM12
VDD	AM18
VDD	AM26
VDD	D02
VDD	H06
VDD	K04
VDD	M02
VDD	M10
VDD	P08
VDD	P12
VDD	T06
VDD	V02
VDD	V10
VDD	Y04
VDD	Y12
VDD	AB06
VDD	AD08
VDD	AF02
VDD	AH04
VDD	B08
VDD	B16
VDD	B22
VDD	B30
VDD	D06
VDD	D14
VDD	D24
VDD	F12
VDD	F18
VDD	F26
VDD	H10
VDD	H20
VDD	K16
VDD	K22
VDD	M14
VDD	M20
VDDA	E01

5.6.2 Package Dimensions

This section outlines the mechanical dimensions for the Scheduler device. The package is a 1088 ceramic column grid array (CCGA).

NOTE: Drawings are not to scale.

Figure 67. Scheduler CCGA package Dimensions - Top and Side Views

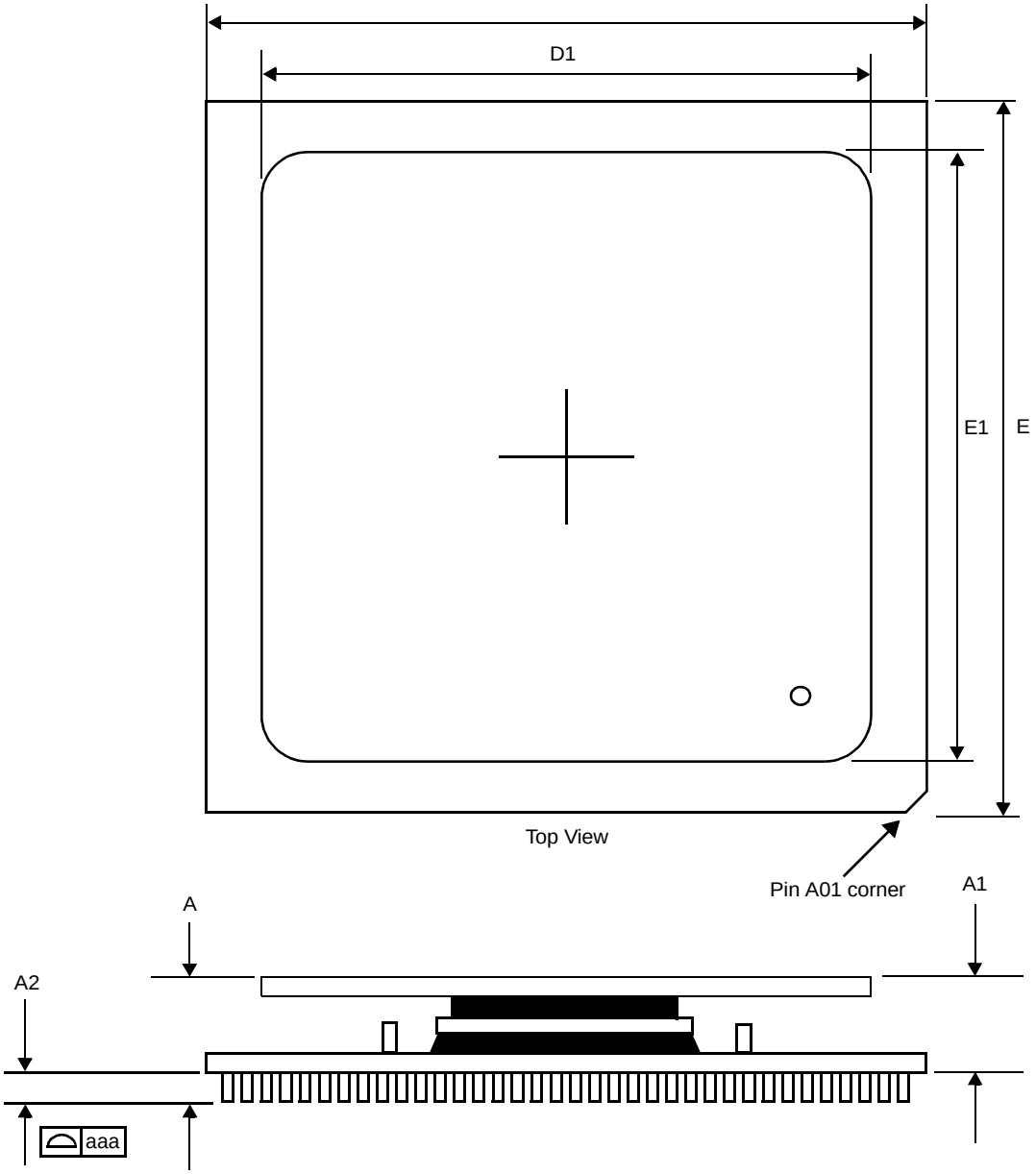


Figure 68. Scheduler CCGA package Dimensions - Bottom View

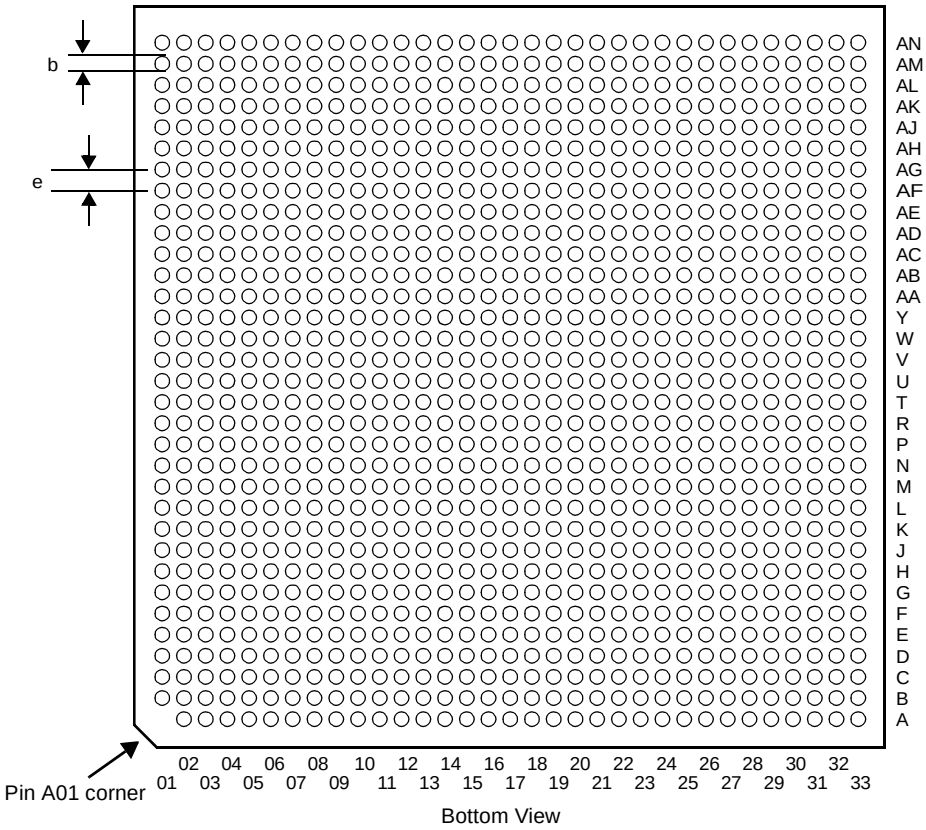


Table 42. Scheduler CCGA Mechanical Specifications

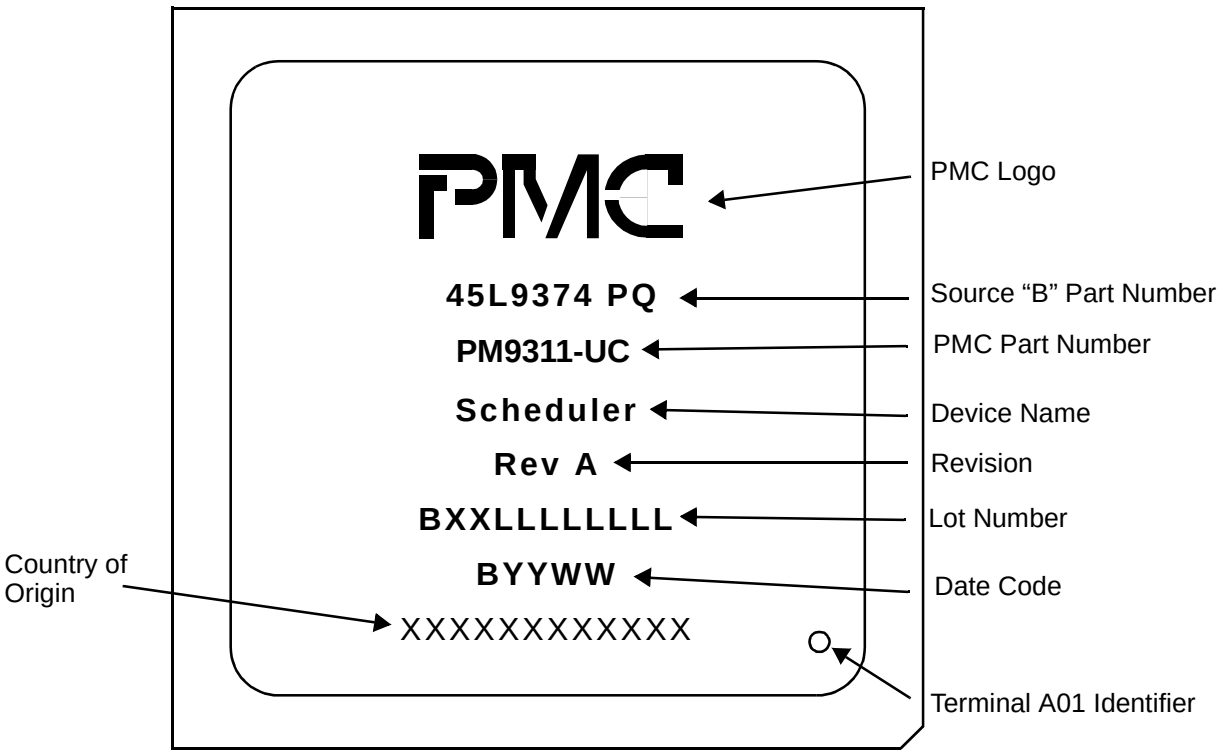
Symbol	e = 1.27 mm pitch			Units
	Minimum	Nominal	Maximum	
A		6.83		mm
A1		4.62		mm
A2		2.21		mm
aaa		0.15		mm
D		42.50		mm
D1		40.64		mm
E		42.50		mm
E1		40.64		mm
M		33 x 33		
N		1088		
b	0.48		0.52	mm

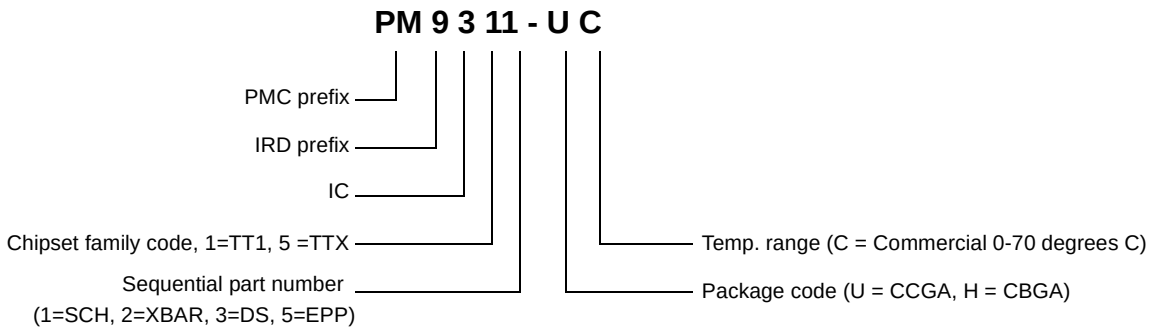
NOTE: Column alloy is 90/10 PbSn

5.6.3 Part Number

The PMC-Sierra Part Number for the Scheduler is:

Scheduler	PM9311-UC





6 Characteristics

6.1 SIGNAL ASSOCIATIONS

Table 43. Signal Associations

Symbol	Device	Associated Signals
3.3V-tolerant 2.5V CMOS (OOB and Test Interfaces)		
Vol1, Voh1 Ioh1, Iol1, Zout1, tval1, tsus1, tr1, tf1	DS	mon[15:0], oob_ad[7:0], oob_int_hi[1:0], oob_int_lo[1:0], oob_wait_L, plllock, lssd_scan_out[15:0], plltest_out
	EPP	oob_ad[7:0], oob_int_hi, oob_int_lo, oob_wait_L, soc_out, lssd_scan_out[15:0], plltest_out
	XBAR	M_ackreg_L, M_pending_L, M_xreset_L, oob_ad[7:0], oob_int_hi, oob_int_lo, soc_out, plllock, lssd_scan_out[15:0], plltest_out
	SCHED	mon[15:0], oob_ad[7:0], oob_int_hi, oob_int_lo, soc_out, plllock, lssd_scan_out[15:0], plltest_out
Vih1, Vil1, tsu1, th1	DS	oob_ad[7:0], oob_clk, oob_devsel0, oob_devsel1, oob_devsel2, oob_valid_L, pwrup_reset_L, crdcrcen0, crdcrcen1, crden0, crden1, ibpen0, ibpen1, ce0_io, ce0_scan, ce0_tstm3, lssd_ce1_a, lssd_ce1_b, lssd_ce1_c1, lssd_ce1_c2,, lssd_ce1_c3, lssd_scan_in[15:0], plltest_in, test_di1, test_di2, test_lt, test_re, test_ri
	EPP	oob_ad[7:0], oob_clk, oob_valid_L, pwrup_reset_in_L, ce0_io, ce0_scan, ce0_test, ce0_tstm3, lssd_ce1_a, lssd_ce1, plltest_in_b, lssd_ce1_c1, lssd_ce1_c2, lssd_ce1_c3, lssd_scan_in[15:0], test_di1, test_di2, test_lt, test_re, test_ri
	XBAR	oob_ad[7:0], oob_clk, oob_valid_L, oob_devsel0, oob_devsel1, oob_devsel2, oob_devsel3, oob_valid_L, pwruprst_L, ce0_io, ce0_scan, lssd_ce1_a, lssd_ce1_b, lssd_ce1_c1, lssd_ce1_c2, lssd_scan_in[15:0], plltest_in, test_di1, test_di2, test_lt, test_ri
	SCHED	oob_ad[7:0], oob_clk, oob_valid_L, oob_devsel0, oob_devsel1, oob_devsel2, oob_devsel3, oob_valid_L, pwrup_reset_in_L, ce0_io, ce0_scan, ce0_testm3, lssd_ce1_a, lssd_ce1_b. lssd_ce1_c1, lssd_ce1_c2, lssd_ce1_c3, lssd_scan_in[15:0], plltest_in, test_di1, test_di2, test_lt, test_ri
tper1, tph1, tpl1	DS	oob_clk
	EPP	oob_clk
	XBAR	oob_clk
	SCHED	oob_clk

Table 43. Signal Associations (Continued)

Symbol	Device	Associated Signals
2.5V CMOS (JTAG Interface)		
Vol2, Voh2,Ioh2, Iol2, Zout2, tval2, tsus2, tr2, tf2	DS	jtag_tdo
	EPP	jtag_tdo
	XBAR	jtag_tdo
	SCHED	jtag_tdo
Vih2, Vil2, tsu2, th2	DS	jtag_tdi, jtag_tms, jtag_trst_L
	EPP	jtag_tdi, jtag_tms, jtag_trst_L
	XBAR	jtag_tdi, jtag_tms, jtag_trst_L
	SCHED	jtag_tdi, jtag_tms, jtag_trst_L
tper2, tph2, tpl2	DS	jtag_tck
	EPP	jtag_tck
	XBAR	jtag_tck
	SCHED	jtag_tck
3.3V-tolerant 2.5V CMOS (Serdes Interface)		
Vol3, Voh3, Ioh3, Iol3, Zout3, tval3, tsus3, tr3, tf3	DS	fotx0_txd[9:0], pin_diode0_reset_L, vcsel0_reset_L, xcvr0_loopback, fotx1_ txd[9:0], pin_diode1_reset_L,, vcsel1_reset_L, xcvr1_loopback, tx_clkin, tx_clkout, fotx_clk_out
Vih3, Vil3, tsu3, th3	DS	forx0_clk, forx0_rxd[9:0], pin_diode0_sig_det, vcsel0_laser_act, xcvr0_com_det, xcvr0_sig_det, forx1_clk, forx1_rxd[9:0], pin_diode1_sig_det, vcsel1_laser_act, xcvr1_com_det, xcvr1_sig_det, fotx_clk_in
tper3, tph3, tpl3	DS	forx0_clk, forx1_clk, fotx_clk_in
	EPP	forx0_clk, forx1_clk, fotx_clk_in
	XBAR	forx0_clk, forx1_clk, fotx_clk_in
	SCHED	forx0_clk, forx1_clk, fotx_clk_in

Table 43. Signal Associations (Continued)

Symbol	Device	Associated Signals
200 Mbps HSTL (EPP/DS Interface)		
Voh4, Vol4, tval4, tsus4, tr4, tf4	DS (HSTL Class 1)	d2p_d0_id[7:0], d2p_d0_od[7:0], d2p_d1_id[7:0], d2p_d1_od[7:0]
	EPP (HSTL Class 1)	p2d_d0_id[7:0], p2d_d1_id[7:0], p2d_d2_id[7:0], p2d_d[3:0]_ic[7:0], p2d_d[3:0]_oc[7:0]
Vih4, Vil4, Zterm4, tsu4, th4	DS	p2d_ic[7:0], p2d_oc[7:0], p2d_d0_id[7:0], p2d_d1_id[7:0]
	EPP	d2p_d0_id[7:0], d2p_d0_od[7:0], d2p_d1_id[7:0], d2p_d1_od[7:0], d2p_d2_id[7:0], d2p_d2_od[7:0]
800 Mbps STI (AIB Interfaces)		
Vocomm5, Vodiff5, Zout5, tval5, tsus5, tr5, th5	DS	d2x_d0a_[c, cn, e, en, o, on], d2x_d0b_[c, cn, e, en, o, on], d2x_d1a_[c, cn, e, en, o, on], d2x_d1b_[c, cn, e, en, o, on]
	EPP	p2s_s0_[c cn, e, en, o, on], p2s_s1_[c cn, e, en, o, on]
	XBAR	x2d_p[31:0]_[c, cn, e, en, o, on]
	SCHED	s2p_p[31:0]_[c, cn, e, en, o, on]
Vicomm5, Vidiff5	DS	x2d_d0a_[c, cn, e, en, o, on], x2d_d0b_[c, cn, e, en, o, on], x2d_d1a_[c, cn, e, en, o, on], x2d_d1b_[c, cn, e, en, o, on]
	EPP	s2p_s0_[c cn, e, en, o, on], s2p_s1_[c cn, e, en, o, on]
	XBAR	d2x_p[31:0]_[c, cn, e, en, o, on]
	SCHED	s2p_p[31:0]_[c, cn, e, en, o, on]
tper5, tph5, tpl5	DS	x2d_d0a_[c,cn], x2d_d0b_[c,cn], x2d_d1a_[c,cn], x2d_d1b_[c,cn]
	EPP	s2p_s0_[c,cn], s2p_s1_[c,cn]
	XBAR	d2x_p[31:0]_[c,cn]
	SCHED	s2p_p[31:0]_[c,cn]

Table 43. Signal Associations (Continued)

Symbol	Device	Associated Signals
PECL		
Vicomm6, Vidiff6	DS	ref_clk, ref_clkn, soc_in, soc_inn
	EPP	ref_clk, ref_clkn, soc_in, soc_inn
	XBAR	ref_clk, ref_clkn, soc_in, soc_inn
	SCHED	ref_clk, ref_clkn, soc_in, soc_inn
tper6, tph6, tpl6	DS	ref_clk, ref_clkn
	EPP	ref_clk, ref_clkn
	XBAR	ref_clk, ref_clkn
	SCHED	ref_clk, ref_clkn

6.2 ABSOLUTE MAXIMUM RATINGS

Table 44. Absolute Maximum Ratings

Symbol	Parameter	Conditions	Min	Typ	Max	Units
VDD	Supply Voltage		2.4		2.6	V
Tship	Shipping	Dry Pack	-40		60	^{o}C
Tstg	Storage Temp	60%RH	0		30	
Tlead	Lead Temp	Soldering, 60s (IR or Vapor) (Phase Reflow)			220	
	ESD Protection		1			kV

Table 45. Absolute Maximum Ratings for 3.3V-tolerant 2.5V CMOS (OOB & Serdes Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DCin	Input Voltage		-0.6		3.9	V
DCout	Output Voltage					

Table 46. Absolute Maximum Ratings for 2.5V CMOS (JTAG Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DCin	Input Voltage		-0.6		VDD+0.6	V
DCout	Output Voltage					

Table 47. Absolute Maximum Ratings for 200 Mbps HSTL (EPP/DS Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DCin	Input Voltage		-0.6		VDD+0.6	V
DCout	Output Voltage					
VDDQin	HSTL Output Supply V		1.5	1.6	1.7	

Table 48. Absolute Maximum Ratings for 800 Mbps STI (AIB Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DCin	Input Voltage		-0.6		VDD+0.6	V
DCout	Output Voltage					

Table 49. Absolute Maximum Ratings for 2.5V 200 Mbps PECL (CLK and SOC Signals)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DCin	Input Voltage		-0.6		VDD+0.6	V
DCout	Output Voltage					

6.3 RECOMMENDED OPERATING CONDITIONS

Table 50. Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Units
VDD	Supply Voltage		2.4		2.6	V
VDDQ	HSTL Output Supply Voltage		1.5	1.6	1.7	
Fhssch, Fhsxbar	Mechanically Isolated Heat Sink Applied Force				23	lbs
Fhsepp	Mechanically Isolated Heat Sink Applied Force				13	lbs
Mhsds, Mhsepp	Attached Heat Sink Mass				30	g
Mhsxbar, Mhssch	Attached Heat Sink Mass				50	g
Tj	Junc Oper Temp		10		85	^{o}C
Θjcepp	EPP Heat Transfer	Incident air velocity = 2 m/s			0.36	^{o}C/W
Θjcds	DS Heat Transfer					
Θjcxbar	XBAR Heat Transfer				0.36	
Θjcsch	SCH Heat Transfer				0.26	
PDepp6	EPP Power Diss	6 DS Loads ref_clk =200mHz	8.68	9.62	10.60	W
PDepp7	EPP Power Diss	7 DS Loads ref_clk =200mHz	8.68	9.62	10.60	
PDds01	DS Power Diss	EPP Header Data Connections ref_clk =200mHz	3.82	4.24	4.67	
PDds45	DS Power Diss	No Header Connections ref_clk =200mHz	3.65	4.04	4.45	
PDxbar	XBAR Power Diss	ref_clk =200mHz	9.52	10.55	11.63	
PDsch	SCH Power Diss		19.04	21.10	23.26	

NOTE: Power dissipation is based on 2.5 V +/-5% supply. Power is dissipated within the package, and can be used for heat transfer calculations. Power dissipation for 1.5V HSTL output is primarily external to the device package, through the termination resistors, see Table 51.

Table 51. Additional Power Design Requirements

Symbol	Parameter	Conditions	Min	Typ	Max	Units
PDhstl1	Termination Power per HSTL Class 1 output	ref_clk =200mHz		22.2[a]		mW

a. These power figures are not included as part of the heat calculation numbers. The 1.6V supply current would be based on the HSTL termination resistance scheme used in Figure 80 of the Characteristics section. This is dependent on the termination scheme used in the system and the tolerance of the resistors used for termination.

6.4 DC ELECTRICAL CHARACTERISTICS

Table 52. DC Electrical Characteristics for 3.3V-tolerant 2.5V CMOS (OOB Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Voh1	Output High V		2.0		VDD	V
Vol1	Output Low V		0		0.4	
Ioh1	Output High I	Voh@2.0V, VDD@2.3V Tj=85°C	7[a]			mA
Iol1	Output Low I	Vol@0.4V VDD@2.3V Tj=85°C	9[a]			
Zout1	Output Impedance			50		Ohm
Vih1	Input High V		1.7			V
Vil1	Input Low V		0		0.7	

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

Table 53. DC Electrical Characteristics for 2.5V CMOS (JTAG Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Voh2	Output High V		2.0		VDD	V
Vol2	Output Low V		0		0.4	
Ioh2	Output High I	Voh@2.0V, VDD@2.3V Tj=85°C	7[a]			mA
Iol2	Output Low I	Vol@0.4V VDD@2.3V Tj=85 °C	9[a]			
Zout2	Output Impedance			50		Ohm
Vih2	Input High V		1.7		VDD	V
Vil2	Input Low V		0		0.7	

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

Table 54. DC Electrical Characteristics for 3.3V-tolerant 2.5V CMOS (Serdes)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Voh3	Output High V		1.9		VDD	V
Vol3	Output Low V		0		0.4	
Ioh3	Output High I	Voh@2.0V, VDD@2.3V Tj=85^{o}C	7[a]			mA
Iol3	Output Low I	Vol@0.4V VDD@2.3V Tj=85^{o}C	9[a]			
Zout3	Output Impedance			50		Ohm
Vih3	Input High V		1.7			V
Vil3	Input Low V		0		0.7	

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

Table 55. DC Electrical Characteristics for 200 Mbps HSTL (EPP/DS Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
VDDQ[a]	HSTL Supply V		1.5	1.6	1.7	V
Vref	HSTL Ref V		0.7	0.75	0.8	
Voh4	Output High V (Class 2)	Ioh=16mA@1.0V	VDDQ -0.4			
	Output High V (Class 1)	Ioh=8mA@1.0V				
Vol4	Output Low V (Class 2)	Iol=16mA@0.4V			0.4	
	Output Low V (Class 1)	Iol=8mA@0.4V				
Vih4	Input High V		VREF +0.05		VDDQ +.0v	
Vil4	Input Low V		-0.5		VREF -0.05	
Zterm4 (HSTL Class 1)	Load Term	R1=R2=100 Ohm Req=50 Ohm		100		Ohm

a. VDDQ is recommended to be 1.6 V to provide additional noise margin above 0.75 V for HSTL signals.

Table 56. DC Electrical Characteristics for 800 Mbps STI (AIB Interfaces)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Vocomm5	Output Comm Mode		1.05		1.45	V
Vodiff5	Output Diff Swing		0.8		1.2	
Zout5	Output Impedance			20		Ohm
Vicomm5	Input Comm Mode			1.25		V
Vidiff5	Input Diff Swing		0.4			
Zterm5	Term Impedance	100 ohm Line-to-Line in Receiver		100		Ohm

Table 57. DC Electrical Characteristics for 2.5V 200 Mbps PECL (CLK and SOC Signals)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Vicomm6	Input Comm Mode			1.3		V
Vidiff6	Input Diff Swing			0.4		

6.5 AC ELECTRICAL CHARACTERISTICS

Table 58. AC Electrical Characteristics for 3.3V-tolerant 2.5V CMOS (OOB Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
tsu1	Input Setup	Trf=2ns Referenced to OOB_clock.	7.6			ns
th1	Input Hold		2.4			
tval1	Output Valid	R1=50 Ohm, C1=1 pF Referenced to OOB_clock.			12[a]	
tsus1	Output Sustain		1.7			
tr1, tf1	Output Rise/Fall Time				1.0[a]	
tper1	Clock Period	Defines OOB_clock.		40		
tph1	Clock Pulse High			20		
tpl1	Clock Pulse Low					

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

Table 59. AC Electrical Characteristics for 2.5V CMOS (JTAG Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
tsu2	Input Setup	Trf=2ns Referenced to jtag_clock.	24[a]			ns
th2	Input Hold		13[a]			
tval2	Output Valid	R1=50 Ohm, C1=1 pF Referenced to jtag_clock.			37[a]	
tsus2	Output Sustain		20[a]			
tr2, tf2	Output Rise/Fall Time				1.0[a]	
tper2	Clock Period	Defines jtag_clock.		100		
tph2	Clock Pulse High		45[a]			
tpl2	Clock Pulse Low					

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

Table 60. AC Electrical Characteristics for 3.3V-tolerant 2.5V CMOS (Serdes Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
tsu3	Input Setup	Trf=500 ps Referenced to forx0_clk, forx1_clk	453			ps
th3	Input Hold		970			
tval3	Output Valid	R1=50 Ohm, C1=1 pF Referenced to fotx_clk_out,	2.45[a]			ns
tsus3	Output Sustain		2.87			
tr3, tf3	Output Rise/Fall Time				0.7[a]	
tper3	Clock Period	Defines fotx_clk_in, forx0_clk, forx1_clk		6.67		
tph3	Clock Pulse High			3.33		
tpl3	Clock Pulse Low					

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

Table 61. Reference Clock for 200 Mbps HSTL (EPP/DS Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
tsu4	Input Setup	Trf=700 ps Referenced to ref_clk, ref_clkn	530			ps
th4	Input Hold		500			
tval4	Output Valid	Class1: R1=R2=100 Ohm, C1=1 pF Referenced to ref_clk, ref_clkn			2200	
tsus4	Output Sustain		800			
tr4, tf4	Output Rise/Fall Time				520[a]	

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

NOTE: Does not include Static Phase Offset or Jitter (See Table 64 on page 308)

Table 62. AC Electrical Characteristics for 800 Mbps STI (AIB Interface)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
tsu5	Input Setup	Trf=750 ps Referenced to s2p_s[1:0]_[c,cn] x2d_d[1:0][a,b]_[c,cn] d2x_p[31:0]_[c,cn] s2p_p[31:0]_[c,cn]	190			ps
th5	Input Hold		180			
tval5	Output Valid	R1=100 Ohm, C1=C2=1 pF Referenced to s2p_s[1:0]_[c,cn] x2d_d[1:0][a,b]_[c,cn] d2x_p[31:0]_[c,cn] s2p_p[31:0]_[c,cn]	382[a]			
tsus5	Output Sustain		432			
tr5	Output Rise Time	Swing = 0.8V Vlo = 0.761V Vhi = 1.59V R1=100 Ohm, C1=C2=1 pF Referenced to s2p_s[1:0]_[c,cn] x2d_d[1:0][a,b]_[c,cn] d2x_p[31:0]_[c,cn] s2p_p[31:0]_[c,cn]			645	
tf5	Output Fall Time				386	
tper5	Clock Period	Defines s2p_s[1:0]_[c,cn] x2d_d[1:0][a,b]_[c,cn] d2x_p[31:0]_[c,cn] s2p_p[31:0]_[c,cn]		2.5		ns
tph5	Clock Pulse High		1.10		1.38	
tpl5	Clock Pulse Low					

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

NOTE: Does not include Static Phase Offset or Jitter (See Table 64 on page 308)

Table 63. AC Electrical Characteristics for 2.5V 200Mbps PECL (CLK Signals)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
tsu6	Input Setup	Trf=600 ps Referenced to ref_clk, ref_clkn	1.3			ns
th6	Input Hold		0.3[a]			
tper6	Clock Period	Defines ref_clk, ref_clkn		5		
tph6	Clock Pulse High			2.5		
tpl6	Clock Pulse Low					

a.. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters

Table 64. Jitter and Static Phase Offset for PLL

Symbol	Parameter	Conditions	Min	Typ	Max	Units
tccj	PLL Jitter - Cycle to Cycle				100[a]	ps
tltj	PLL Long-term Jitter or Device to Device				400[a]	
tpos	PLL Static Phase Offset				200[a]	

a. This parameter is not tested. It is provided here as a reasonable guide to design, based on expected process parameters.

6.6 TIMING DIAGRAMS

Figure 69. Setup and Hold for 3.3V-tolerant 2.5V CMOS and 2.5V CMOS (OOB Interface)

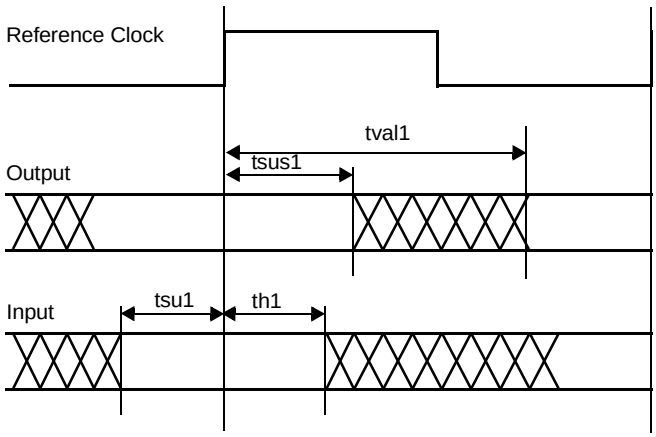


Figure 70. Rise and Fall Times for 3.3V-tolerant 2.5V CMOS and 2.5V CMOS (OOB Interface)

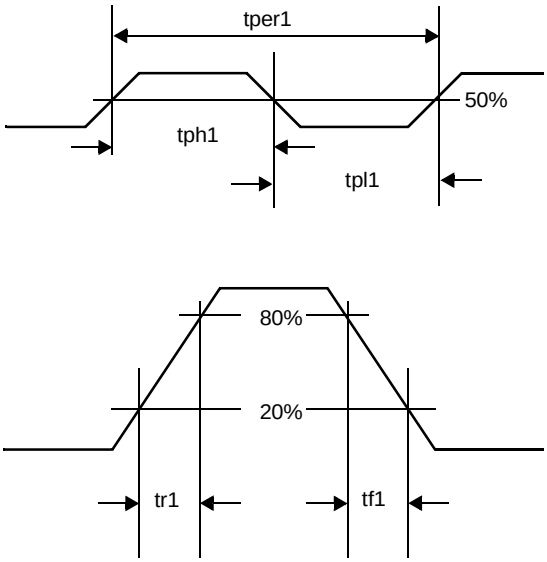


Figure 71. OOB Test Loading

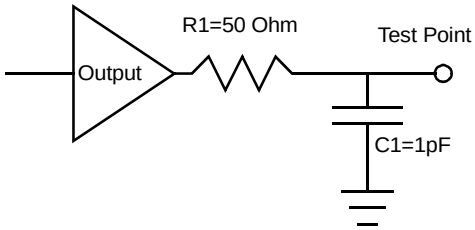


Figure 72. Setup and Hold for 3.3V-tolerant 2.5V CMOS and 2.5V CMOS (JTAG Interface)

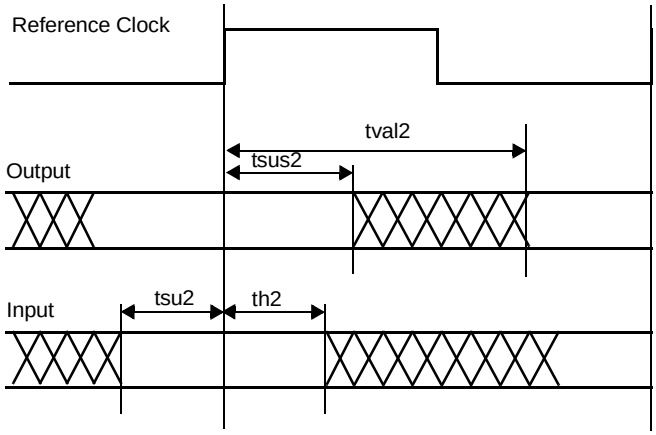


Figure 73. Rise and Fall Times for 3.3V-tolerant 2.5V CMOS and 2.5V CMOS (JTAG Interface)

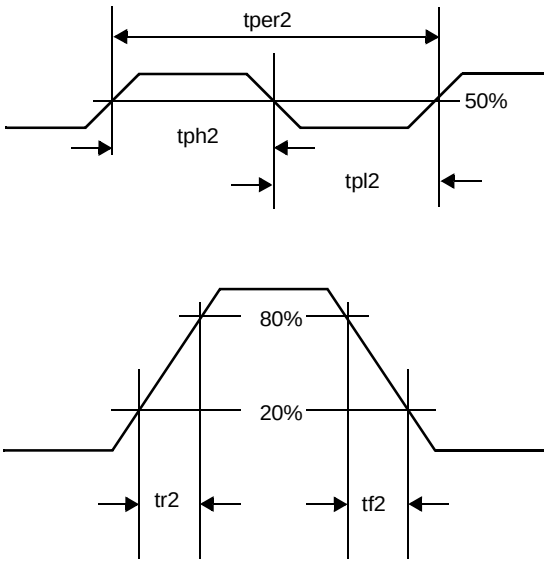


Figure 74. JTAG Test Loading

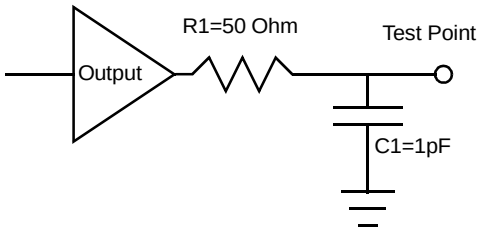


Figure 75. Setup and Hold for 3.3V-tolerant 2.5V CMOS and 2.5V CMOS (Serdes Interface)

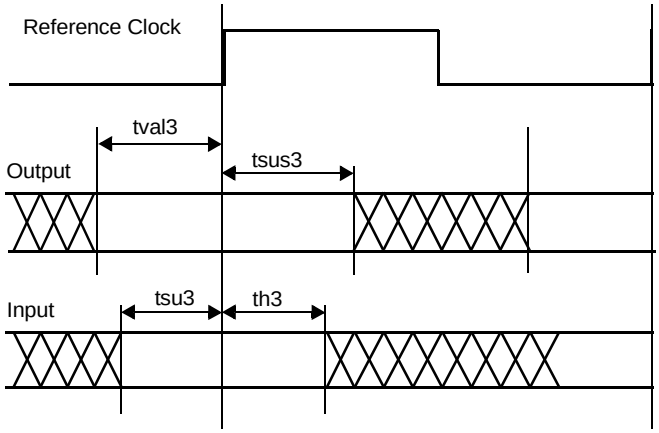


Figure 76. Rise and Fall Times for 3.3V-tolerant 2.5V CMOS and 2.5V CMOS (Serdes Interface)

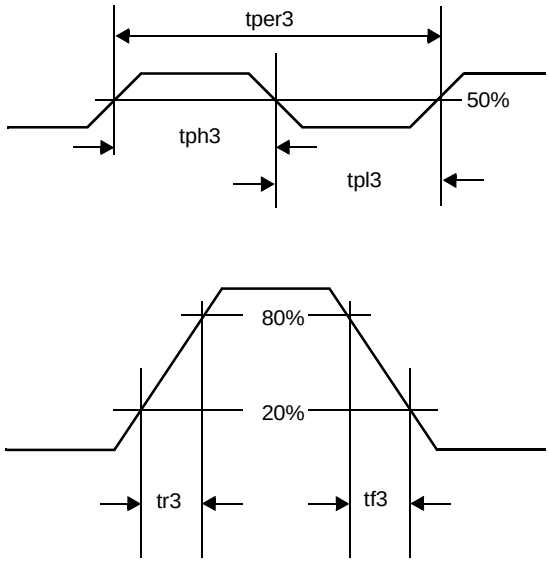


Figure 77. Serdes Test Loading

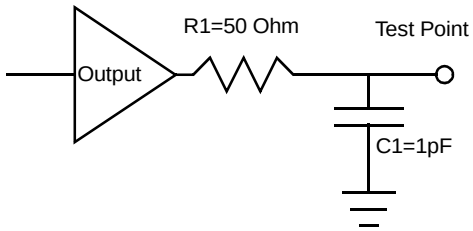


Figure 78. Setup and Hold for 200 Mbps HSTL (EPP/DS Interface)

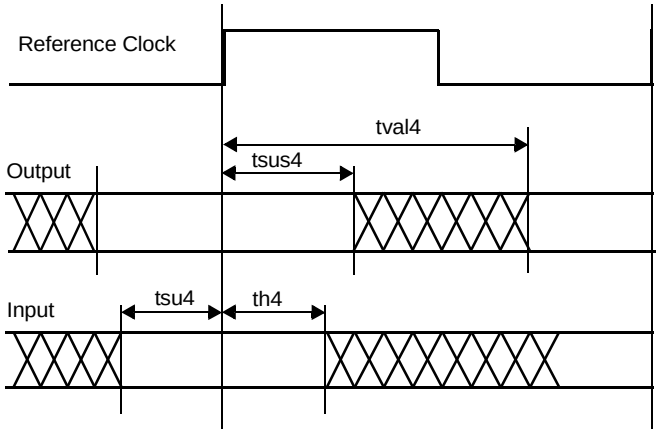


Figure 79. Rise and Fall Times for 200 Mbps HSTL (EPP/DS Interface)

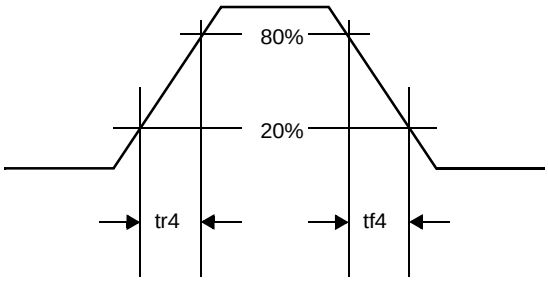


Figure 80. Class 1 HSTL Test Loading

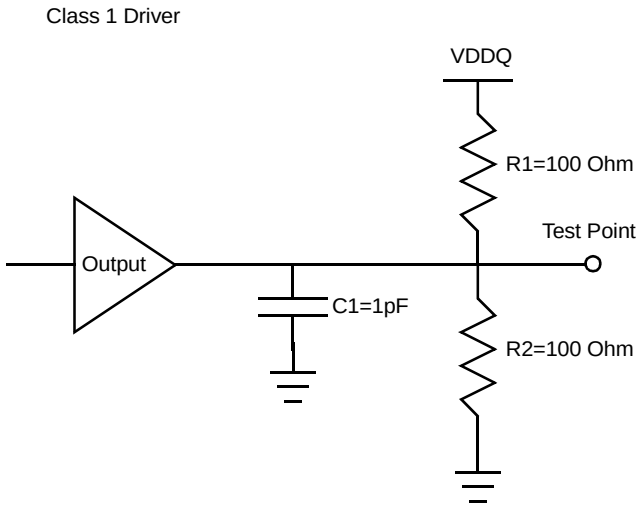


Figure 81. Setup and Hold for 800 Mbps STI (AIB Interfaces)

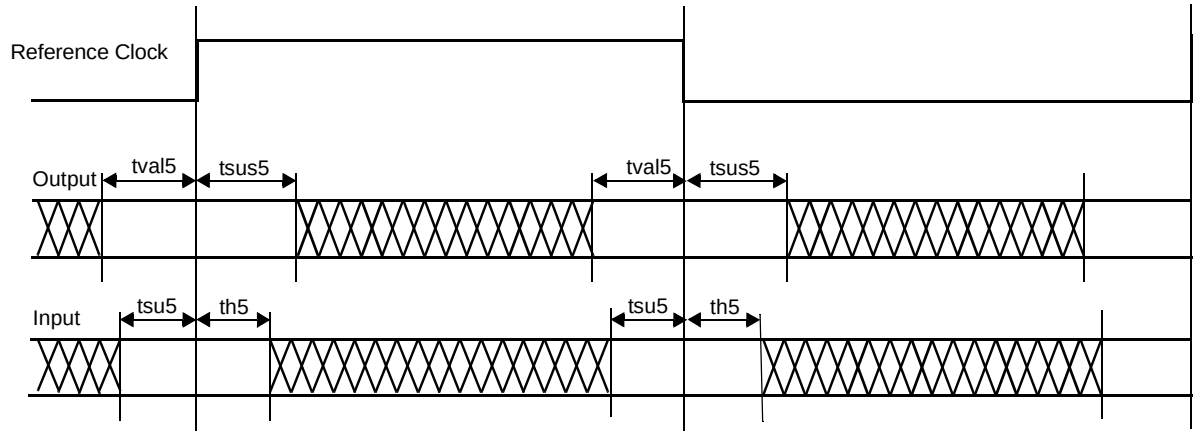


Figure 82. Rise and Fall Times for 800 Mbps STI (AIB Interfaces))

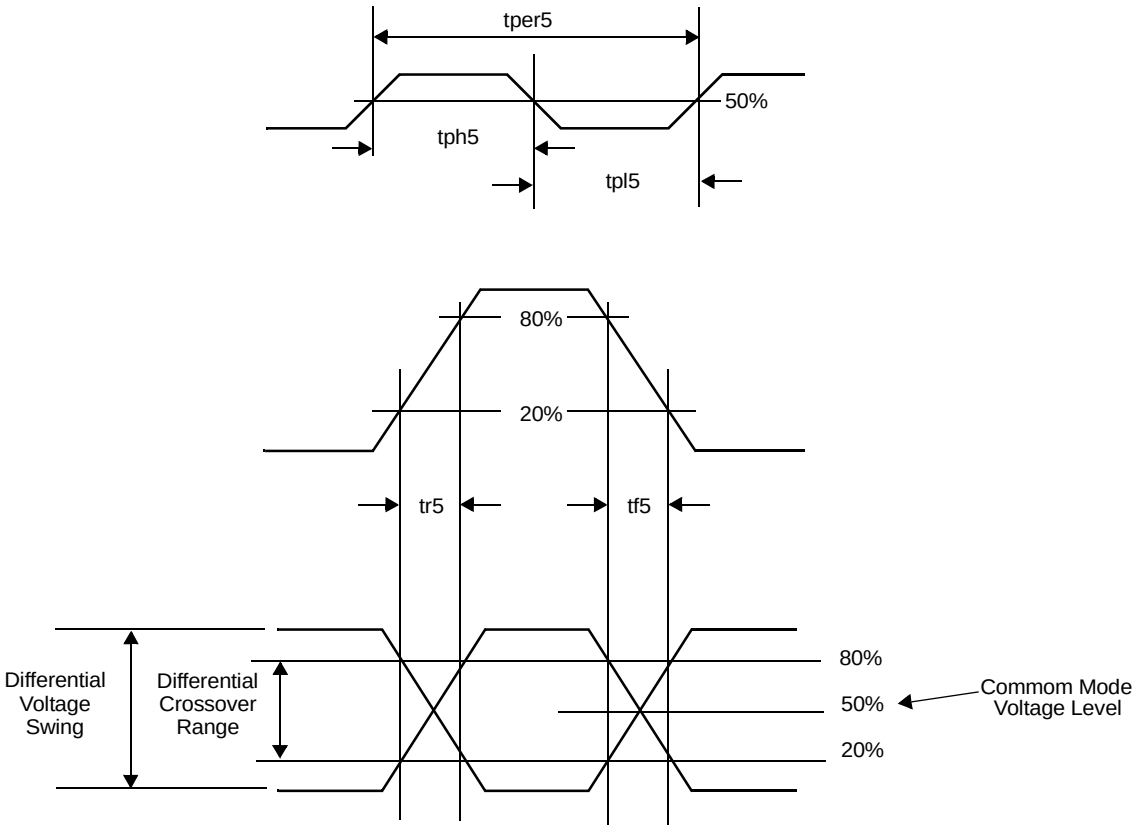


Figure 83. AIB Test Loading

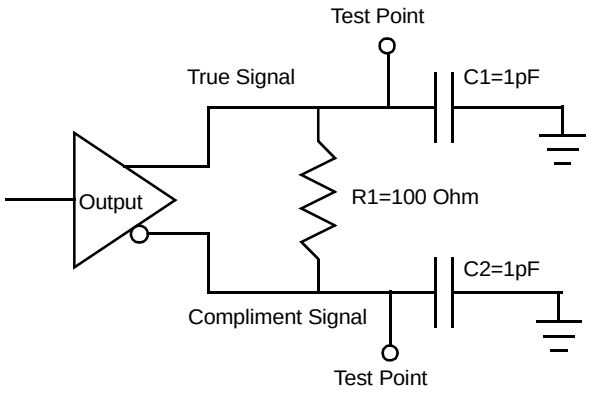


Figure 84. Setup and Hold for 2.5V 200 Mbps PECL (CLK & SOC Signals)

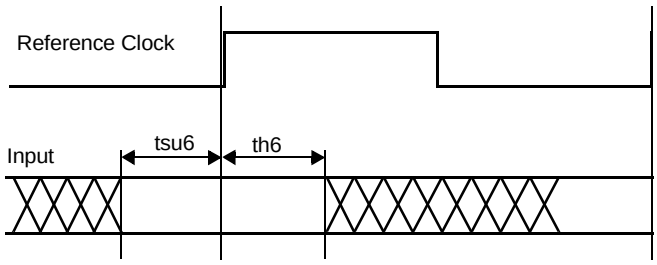
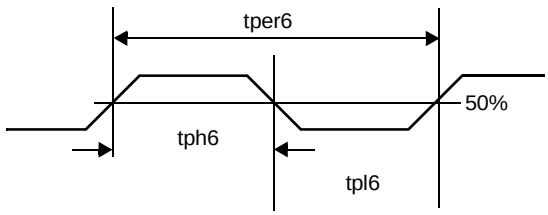


Figure 85. Clock Period and Duty Cycle for 2.5V 200 Mbps PECL (CLK Signal)



Appendix A. General Packet/Frame Formats

A.1 FRAME FORMATS - DATASLICE TO AND FROM ENHANCED PORT PROCESSOR

Table 65 and Table 66 show the format of the Request/Grant from iDS and Data from oDS frames, Table 67 shows the format of the Grant/Data to iDS frames, and Table 68 shows the format of the Control to xDS frames.

Table 65. Request/Data from iDS Frame Format

Field	Size (bits)	Meaning
Reserved	1	
VLD	1	1 if frame contains valid cell
Code Error	1	1 if frame contained 8b/10b error when received
Token Request	1	1 if DS requests token to send data to EPP
Reserved	4	
Cell Data	48	Cell Data (includes LCS header)
Reserved	8	

Table 66. Data from oDS Frame Format

Field	Size (bits)	Meaning
Reserved	1	
VLD	1	1 if frame is valid (used on oDS/oEPP only)
Reserved	6	
Cell Data	48	Cell Data (includes LCS header)
Reserved	8	

Table 67. Grant/Data to iDS Frame Format

Field	Size (bits)	Meaning
Reserved	8	

Table 67. Grant/Data to iDS Frame Format

Field	Size (bits)	Meaning
Cell Data	48	Cell Data (includes LCS header with CRC over Tag only)
Credit CRC or Reserved	8	CRC over the credit portion of the LCS header on DS1, reserved on DS0, DS2 --DS11

Table 68. Control to xDS Link Format

Field	Size (bits)	Meaning
Reserved	1	
Read Zero/ Token	1	Output Side: 1 if DS is to send all-0 cell into oFIFO or Crossbar Input Side: 1 if DS now has a token to use for sending cells into the EPP
Rtag VLD	1	1 if the routing tag is valid
RTag	5	Routing Tag (only used by iEPP)
Read Req	1	1 if read is to be performed from iQM or oQM
Read Addr.	23	Read Address of cell to be read from iQM or oQM
Write Req.	1	1 if write is to be performed to iQM or oQM
Write Addr.	23	Write Address of cell to be written into iQM or oQM
Reserved	8	

A.2 FRAME FORMATS - DATASLICE TO AND FROM CROSSBAR

Table 69 shows the frame format from the Dataslices to the Crossbar; Table 70 shows the frame format used from the Crossbar to the Dataslices.

Table 69. Dataslice to Crossbar Frame Format

Field	Size (bits)	Meaning
synched	1	AIB synched
Cell VLD	1	1 if the cell is valid
RTagVLD	1	Routing Tag Valid
RTag	5	Routing Tag
Cell Data	48	Cell Data
CRC	8	CRC computed over previous 56 bits

Table 70. Crossbar to Dataslice Frame Format

Field	Size (bits)	Meaning
synched	1	AIB synched
VLD	1	1 if cell data is valid
Reserved	6	
Cell Data	48	Cell Data
CRC	8	CRC computed over previous 56 bits

A.3 FRAME FORMATS - ENHANCED PORT PROCESSOR TO AND FROM SCHEDULER

Table 71 shows the frame format from the Scheduler to the PP. Table 72 shows the frame format from the EPP to the Scheduler. When the “Request Valid” is set to 0 but the “M/U” bit is set to 1, the frame is a special control frame for the Scheduler.

Table 71. Scheduler to EPP Frame Format

Field	Size (bits)	Meaning
synched	1	AIB Synched
Grant VLD	1	1 if the grant is valid
Grant DQ	1	1 if the grant should cause the cell to be dequeued
Reserved	4	
TDM Grant	1	1 if the grant is for a TDM cell
Grant M/U	1	1 if the grant is multicast, 0 if unicast
Grant Priority	2	The priority of the grant
Grant Port	5	The output port of the grant
Reserved	2	
RTag VLD	1	1 if the routing tag is valid
RTag Port	5	The input port of the routing tag
Freeze	1	1 if the EPP should be frozen, 0 if not.
Flow Control Crossbar Sync	1	Flow Control Crossbar Sync bit

Table 71. Scheduler to EPP Frame Format (Continued)

Field	Size (bits)	Meaning
Reserved	6	
Sync	1	if set, PP should set outgoing LCS TDM Sync bit
TDM Table Sel	1	if 0, use TDM table 1, if 1, use TDM table 2.
Reserved	22	
CRC	8	CRC computed over previous 56 bits

Table 72. EPP to Scheduler Frame Format

Field	Size (bits)	Meaning
synched	1	AIB Synched
BP VLD	1	1 if the QID being read off the BP FIFO should be backpressured
UBP VLD	1	1 if the unbackpressure information is valid
UBP	8	QID (cast, priority, input port [if unicast, input port ignored if multicast]) for which backpressure is deasserted.
Input TDM Req.	1	Asserted when input has TDM cell to send.
Output TDM Req.	1	Asserted when output is ready to receive TDM cell
TDM Port	5	Input port from which the output is ready to receive TDM cell (valid only if Output TDM asserted)
Suggested TDM Sync	1	1 if this port is currently sending a suggested TDM Sync.
TDM Table Select	1	TDM Frame to use next time the Sync is set (only used if this port happens to be chosen as Master by the Scheduler)
Request VLD	1	1 if the best-effort request to follow is valid.
M/U and control msg.	1	If Request VLD is set then this bit indicates if the request is multicast (set to 1) or unicast (set to 0). If Request VLD is not set then this bit indicates if this frame is a control message to the Scheduler (set to 1) or just a frame with no request (set to 0).
Priority	2	Priority of request
Fanout	32	Multicast fanout if request is multicast, top 5 bits used for output port and bottom 27 bits reserved if the request is unicast.
CRC	8	CRC computed over previous 58 bits

Appendix B. Common Pinout Configuration

B.1 JTAG INTERFACE

The ETT1 Chip Set supports the following JTAG public instructions in conformance with the IEEE Std 1149.1 specification:

- IDCODE = b100
- BYPASS = b111
- EXTEST = b000
- SAMPLE/PRELOAD = b010

As part of that standard, the following pins are used on each of the ETT1 Chip Set devices.

jtag_tck, jtag_tdi, jtag_tdo, jtag_tms, jtag_trst_L

The id codes used for the ETT1 Chip Set are as follows:

Dataslice	14367049h
Enhanced Port Processor	14581049h
Crossbar	14372049h
Scheduler	14374049h

B.2 RESERVED MANUFACTURING TEST PINS

Each device in the ETT1 Chip Set has a number of manufacturing test pins. These pins are reserved for use during the manufacturing process. For functional operation, input test pins must be driven by a voltage source, while output test pins should be left unconnected. Depending on conventions used during PCB assembly, test inputs can be tied with a 1k ohm resistor to the appropriate VDD or GND supply. Such a convention allows for onboard test operation for unusual diagnostic purposes. Test outputs should still be soldered to the PCB for proper thermal and mechanical operation, and these outputs are often brought to the solder (bottom) side of the board through vias for standard test fixtures.

For correct functional operation of the ETT1 devices, the following test inputs should be tied to VDD through a resistor:

lssd_ce1_a, lssd_ce1_b, lssd_ce1_c1, lssd_ce1_c2, lssd_ce1_c3,

test_ri, test_lt,

lssd_scan_in[0:15],

plltest_in

The following test inputs should be tied to GND through a resistor:

ce0_io, ce0_scan, ce0_test, ce0_tstm3,

test_re

The following test outputs should be left floating:

lssd_scan_out[0:15],

mon[0:15],

plltest_out

The following test inputs must be driven low during reset and high during operation. These can be connected to the same reset signal used for the pwrup_reset_in_L input.

NOTE: These inputs can also driven low during board assembly tests to tristate all outputs. Testing of board continuity can done during ICT.

test_di1, test_di2

The following test output could be left floating. Optionally, it may be used to monitor that the internal PLL has locked to the ref_clk/ref_clkn inputs by checking the signal level during ETT1 operation.

plllock

B.3 POWER SUPPLY CONNECTIONS

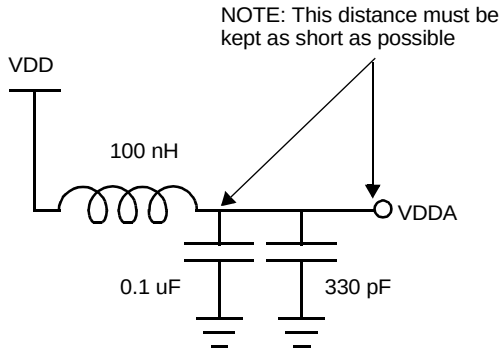
All supply inputs labeled VDD must be driven by a common 2.5 V (nominal) supply.

All supply inputs labeled GND must be tied to a common (0 V) ground.

All pins labeled UNUSED must not be tied to a power or ground plane.

The supply input labeled VDDA must be connected to the common 2.5 V supply through a noise isolation circuit. One possible method is shown in Figure 86.

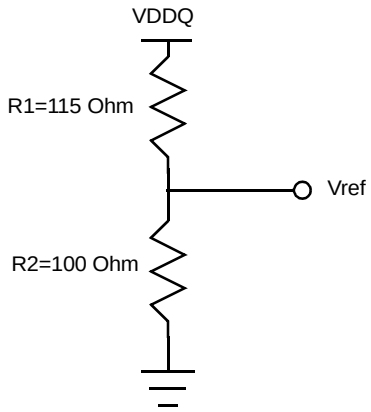
Figure 86. Example Noise Isolation Circuit



All supply inputs labeled VDDQ must be driven by a common 1.6 V (nominal) supply.

All pins labeled Vref must be connected to a common voltage reference plane that is 0.75 V and is derived from VDDQ.This voltage reference plane (Vref) must be supplied by a 115 ohm / 100 ohm voltage divider. One possible method is shown in Figure 87.

Figure 87. Example Voltage Divider



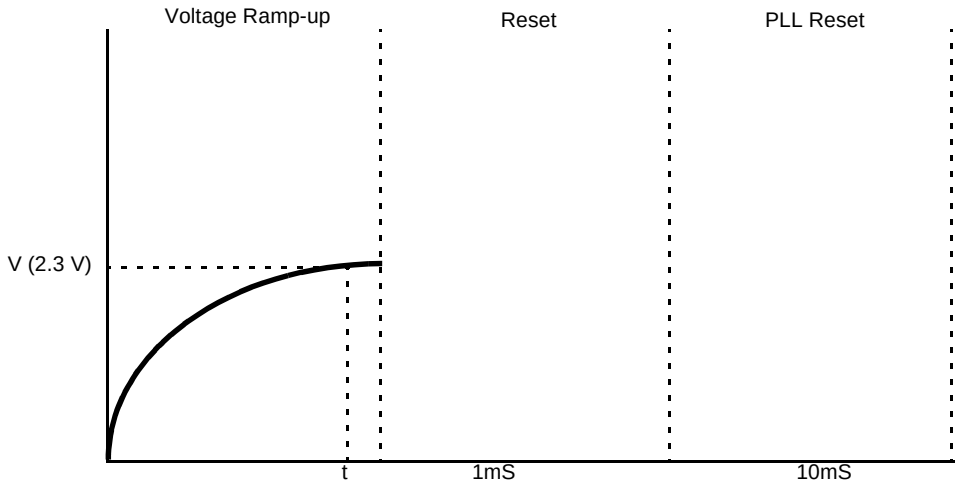
B.4 EXAMPLE POWER UP SEQUENCE

The power up sequence, shown in Figure 88, consists of three parts before the device is fully operational:

- voltage ramp-up
- hard reset
  - Note : pwruprst_L is a synchronous (to ref_clk and soc_in), active low signal
- PLL reset

The voltage ramp-up time is the time needed for the power to the device to stabilize above 2.3 V.

Figure 88. Voltage Ramp-up



Note: This example is design dependent
Note: pwruprst_L is a synchronous, active low signal

Appendix C Interfacing Details for ETT1

C.1 COMPENSATING FOR ROUND TRIP DELAY BETWEEN LINECARD AND ETT1(LCS)

A key benefit of the LCS™ protocol is the physical separation of the ETT1 fabric and the linecards. This separation introduces some latency in the communication channel between the linecard and ETT1. The credit mechanism used in LCS ensures that this latency does not adversely impact the throughput of the system. However, the linecard must compensate for the latency in the communication channel so that the credit mechanism can function correctly. This section describes the operations that the linecard should perform.

C.1.1 Preventing Underflow of the VOQ

The linecard must respond to grants from the ETT1Chip Set within a certain period of time in order to sustain maximum throughput. In this section we explain how much time the linecard has in which to respond to grants.

Consider the case where the virtual output queue in the port card in Figure 89 is full, with 64 cells waiting to be forwarded to the appropriate output. Now, assume that the ETT1 Scheduler starts to forward those cells, one per cell time. When the first cell is forwarded, a grant is sent to the linecard, indicating that another cell can be sent to that output. If that cell does not arrive before the VOQ is empty, then the throughput for that flow cannot be equal to the link rate. Therefore, the new cell forwarded by the linecard must arrive at the EPP/DS before the VOQ is emptied.

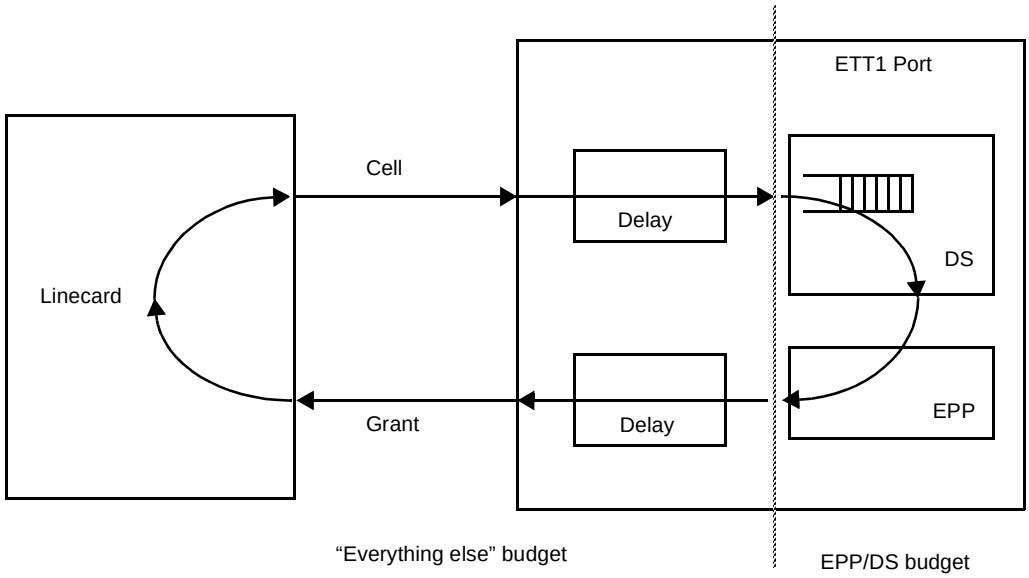
The round trip time for the whole process must be less than or equal to the number of cells in the VOQ; 64 cells for an OC-192c flow and 16 cells for an OC-48c flow. Figure 89 shows this time divided between the EPP/DS devices and “everything else”.

The EPP/DS uses 22 cells plus the programmed_token_delay. Refer to Section 3.4.2.39 “Internal Delay Matching Adjustments” on page 188. Subtract the total (22 cells + programmed_token_delay) from 64 to get the budget for “everything else” (OC-192c cells).

“Everything else” includes any delay on the port card (Serdes), a mux device, the propagation delay of the signal down the fiber, the Serdes at the linecard and the linecard device delay.

To calculate the time available to the linecard device, subtract the other elements from the “for everything else” budget. The fiber delay is approximately 15 cell times (7.5 cells in each direction), assuming a 70m link.

Figure 89. The Time Available to the Linecard



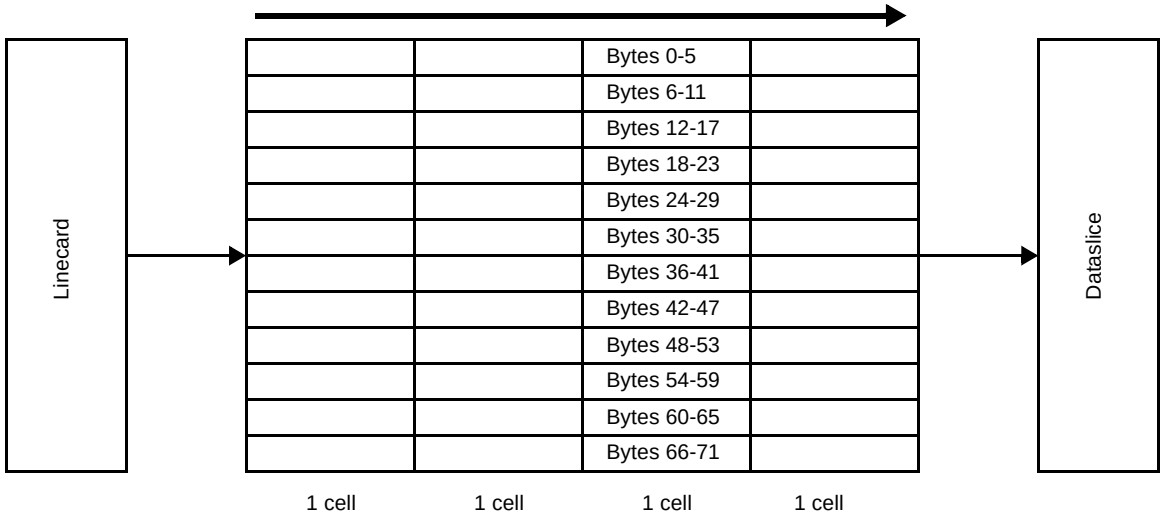
The time available to the linecard is a function of the round trip minus the time used by the EPP/DS devices.

C.2 THE 8B/10B INTERFACE TO THE DATASLICE

This section describes the external interface to the Dataslice. It explains how each physical link should be configured, how the links are combined to carry cell traffic, and what errors may occur and what impact those errors may have.

The Dataslice uses 12 physical *links* in order to provide a 10 Gbit/s aggregate *channel* rate. The 72 byte cells carried across the channel are striped across the 12 individual links. Figure 90 shows this striping in one direction.

Figure 90. Cells are Striped Across the 12 Physical Links



The link technology used is the same as that used by Gigabit Ethernet (GE), except that it operates at 1.5 Gbaud (150MHz) instead of 1.25 Gbaud (125MHz). While the Dataslice can be programmed to use any arbitrary 8b/10b code, the ETT1system has been focused on the use of the GE standard and thus the 8b/10b code developed originally by IBM. The following discussion assumes the use of standard Serdes components; however an alternative 8b/10b scheme could be used if suitable physical level components are available.

C.2.1 Link

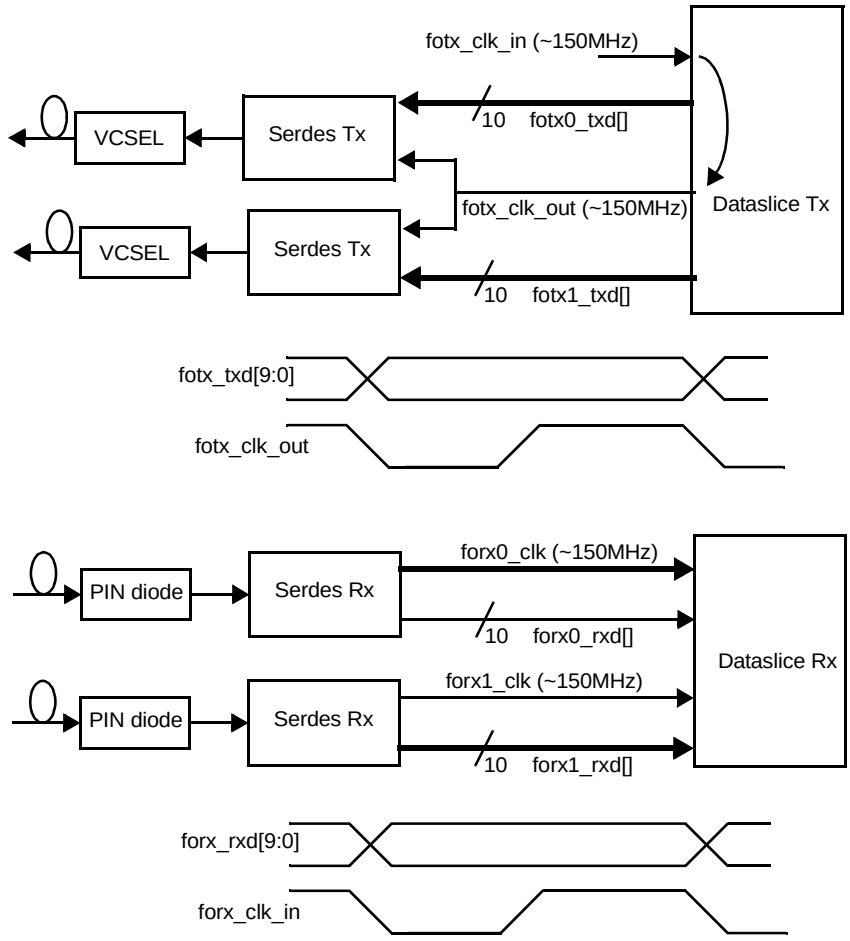
Before looking at how the channel operates, we first consider a single link in its various aspects: physical interface, initialization, and errors.

C.2.1.1 Link Physical Interface

Each logical Dataslice device connects to a Serdes device via 10-bit wide transmit and receive interfaces, operating at 150MHz in a source synchronous mode - i.e. the data is accompanied by a suitable clock. The Serdes transmits a serialized bit stream to a 1.5Gbaud VCSEL and receives a serialized bit stream from a 1.5Gbaud PIN diode.

The ETT1 port card provides a 150MHz oscillator that is used as a reference clock for the Dataslice parallel bus transmitters. In the egress direction this reference clock is provided to the Dataslice which adjusts the phase of the reference so that it is aligned in mid-bit time of the transmitted word. In the ingress direction the Serdes device recovers a nominal 150MHz receive clock from the incoming serial stream and provides it to the Dataslice receiver. The Dataslice expects to sample the incoming 10-bit word using the receive clock; the data is then re-synchronized internally within the Dataslice. Figure 91 illustrates the two directions.

Figure 91. The 10-bit Data Paths



C.2.1.2 Link Initialization

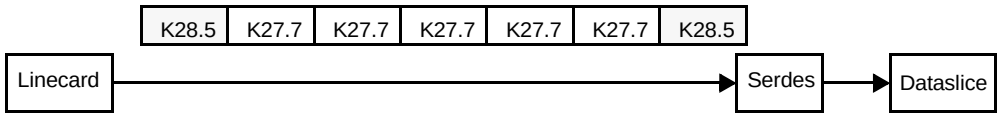
At power-on, every individual link needs to be initialized so that the receiver can lock onto the incoming stream and can identify both word and cell boundaries. To this end, the transmitter must send a particular sequence of 8b/10b control words. Now, the Dataslice assumes that it will be supplied with a word clock (forx_clk) and so only needs to determine a cell boundary. Remember that each Dataslice only sees six words of each cell. So the Dataslice looks for one occurrence of control word 0 followed by five occurrences of some other control word (e.g. 1). The Dataslice has its own 10b->8b decoder and so it can map an arbitrary 8b/10b control word to its own decoded control word.

Assuming the use of 8b/10b Serdes, then we suggest the following mapping:

Table 73. Suggested 8b/10b Decode Map Within the Dataslice

8b/10b code	Dataslice code
K28.5	Control 0
K27.7	Control 1 through 254
K29.7	Control 255

Figure 92. Initial Sequence Expected at Port Card



The initial control word sequence decoded by the Dataslice is expected to be {0,1,1,1,1,1,0,1,1...}

Once the Dataslice has seen five consecutive occurrences of the expected sequence it then locks its cell timing so that the K28.5 word corresponds to the first word in the cell.

The linecard may instead send {K28.5, K29.7, K29.7, K29.7, K29.7, K29.7}. This simply tells the Dataslice that the *linecard's receiver* has locked onto the Dataslice's transmitted signal. This is useful because it tells the ETT1 CPU that the link is up and locked in both directions.

The transmit side of the Dataslice sends the same sequence: it starts by sending {K28.5, K27.7, K27.7, K27.7, K27.7, K27.7}. When the *receive* side of the Dataslice has achieved cell frame lock then the transmit side changes to send {K28.5, K29.7, K29.7, K29.7, K29.7, K29.7} instead.

The sequence {K28.5, K27.7, K27.7, K27.7, K27.7, K27.7} is called Idle-not-ready.

The sequence {K28.5, K29.7, K29.7, K29.7, K29.7, K29.7} is called Idle-ready.

Table 74 shows the state machine operated by the transmit side of the Dataslice.

Table 74. Dataslice Egress Truth Table

8B/10B Encoder Enable	Output Fifo Status	8B/10B Ingress Link Ready	10B Transmit Data (TxD)
0	don't care	don't care	Null (10'b0)
1	0 (FIFO empty)	0	K28.5, K27.7
1	0	1	K28.5, K29.7
1	1 (FIFO not empty)	don't care	Dxx.x
1	1	don't care	Dxx.x

If the encoder is not enabled then all zeros are sent.

If the Dataslice output FIFO is empty then the Dataslice sends either Idle-ready or Idle-not-ready, depending on the state of the receiver.

Otherwise the Dataslice will send the data cell at the head of its transmit FIFO.

Why K28.5, K27.7 etc? The Dataslice doesn't care what the actual 10b control words are. We have chosen K28.5 as it is the 8b/10b comma character which the Serdes receiver will try to lock to in order to determine the word boundary in the bit stream. The choice of K27.7 and K29.7 is somewhat arbitrary but they comply with the 8b/10b transmission rules. (Indeed any control characters can be used provided that the Dataslice decoder table is configured appropriately.)

To summarize, the steps required to bring up the ingress and egress linecard to Dataslice links are:

- Program the 8b/10b codec tables on each Dataslice;
- Transmit Idle-not-ready cells from the linecard to each Dataslice;
- Enable the 8b/10b codecs on each Dataslice causing the Dataslices to start sending Idle-not-ready cells to the linecard;
- Each Dataslice will transition to sending Idle-ready cells when it has acquired cell synchronization from the linecard;
- Likewise the linecard should transition to sending Idle-ready cells when it has acquired cell synchronization from each Dataslice.

Eventually both the linecard and the Dataslice will be receiving the Idle-ready sequence, indicating that both ends of the link have locked up. The linecard transmitter should then switch to sending data cells.

The Dataslice has a receive FIFO which stores the incoming words only if they are decoded as a data word. Idle-not-ready and Idle-ready are not stored in the FIFO. A data cell is a sequence of six 8b/10b data words (Dxx.x). These data cells are stored in the receive FIFO.

The Dataslice reads from the receive FIFO at a nominal 150MHz which is derived from a local oscillator. This local oscillator will operate at a slightly different frequency to that used on the linecard transmitter. To avoid overruns in the case where the linecard is at a slightly faster frequency than the Dataslice, the linecard must send Idle cells at regular periods. The period is determined by the maximum difference in frequency that is possible. Assuming +/-200 ppm oscillators, and allowing for storage of one extra cell in the FIFO, then 1 in 2,500 cells must be an Idle cell to avoid overrun.

The oscillator on the ETT1 port card might be faster than that used on the linecard. Therefore, the Port Processor contains a register that will determine the period between Idle cells sent from the ETT1 port, to avoid overrunning the receive FIFO in the linecard.

C.2.1.3 Link Errors

If the link has no errors then the Dataslice will always see valid sequences of either six control words or six data words. In reality, errors will occur. The system designer must be aware of the implications of the different errors.

- If a single word is corrupted to map to an illegal 10b word then the Dataslice will set DIR bit 23 and assert an interrupt signal to the OOB bus. If the error occurs at one of the Dataslices that deals with the LCS header or magic packets (i.e., Dataslices 0, 1, or 2) then the error is communicated to the Enhanced Port Processor which will ignore that cell.

- If a word is corrupted to look like another valid word, then this will *not* be detected, unless it corrupts the LCS header in which case the header CRC should detect it. If the user payload is corrupted then the cell will still be carried through the switch to the egress linecard.

- The Serdes recognizes the bit sequence 0011111xx as a comma character for byte alignment. This bit sequence occurs in the K28.5 comma character and it is used for link synchronization as described above. An error can potentially cause the 0011111xx bit sequence to appear anywhere in the serial bit stream, causing the Serdes to re-align to the corrupted 0011111xx comma and thus lose byte synchronization. This condition will cause a sequence of 8B/10B errors to be encountered by the 8B/10B decoder and the Dataslice will raise the DIR23 decoder error interrupt. The Serdes should re-acquire lock when the linecard next sends an Idle cell. To prevent fiber optics bit errors from throwing the Serdes out of byte synchronization the Serdes byte sync enable pin can be connected to a pin on the Dataslice that is accessible through the OOB. Once the Serdes has locked onto the right byte boundary software can make sure that the Serdes will no longer look for the 0011111xx comma character.

- A subtle yet serious error can occur if an idle cell is corrupted in such a way that the comma character as programmed in the Dataslice 8B/10B decoder lookup table appears anywhere other than at the start of the cell or several data words in the same cell are corrupted into valid control words. This can cause the entire cell to appear as a cell of the other type (data -> Idle or Idle -> data). While improbable, this can have serious consequences that are discussed in the section on Channel errors.

C.2.2 Channel

A channel consists of 12[1]links and thus 12 logical Dataslices, each slice dealing with six bytes of the 72 byte cell.

C.2.2.1 Synchronization

Each of the 12 channels carry a part (6-bytes) of the same LCS cell. Although the 12 channels are all operating at the same frequency, their phase may be offset with respect to each other. In other words, each Dataslice may receive its 6-byte portion of the cell at a slightly different time from its neighboring Dataslices on the same switch port. So that the 12 parts of a cell remain intact, and are all switched at the same time,

1. The EPP can support fourteen logical slices (i.e. seven Dataslice chips)

we need to realign the output from the 12 channels. This re-alignment (or synchronization) is achieved as follows.

The Dataslices provide a mechanism that guarantees that the slices of the same cell are removed from the input FIFOs synchronously in the core ETT1 clock domain[1]. This synchronization is achieved by only removing cells from the input FIFO when all 12 Dataslices have received their 6 bytes of a given cell. To accomplish this, upon each new cell arrival, a token is passed from Dataslice 0 to the EPP, which in turn broadcasts the token back to every Dataslice after an additional programmed token delay. . Refer to Section 3.4.2.39 “Internal Delay Matching Adjustments” on page 188. Each token that is granted by the EPP allows for one cell to be removed from the input FIFO of each Dataslice and sent to the EPP.

The token delay mechanism compensates for a certain amount of skew/delay measured from arrival at Dataslice 0 to arrival at Dataslices 1 through 11. Additional depth in the input FIFO, beyond the depth required to support the token delay, compensates for skew from arrival at Dataslices 1 through 11 to arrival at Dataslice 0. Table 75 shows the allowable skew in each direction for each possible value of the programmed token delay, in units of 150MHz clock cycle times.

Table 75. Programmed Token Delay vs. Inter-link Skew

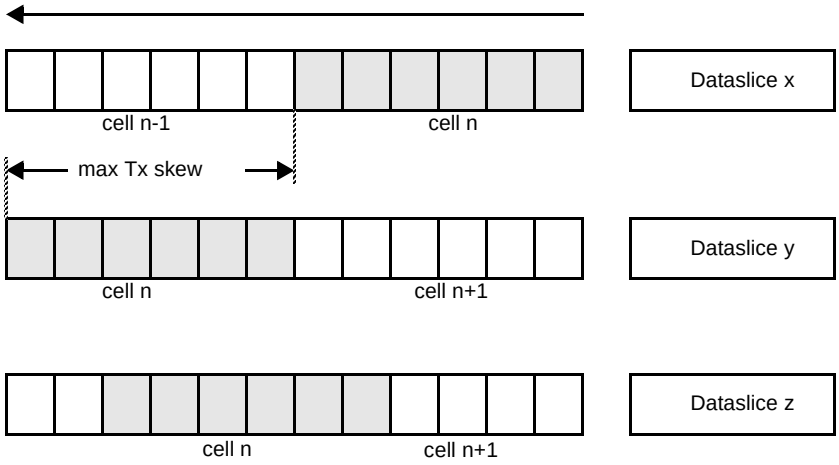
Programmed Token Delay	Max Skew DS0 to DS1 through 11 (150 MHz Clock Cycles)	Max Skew DS1 through 11 to DS0 (150 MHz Clock Cycles)
0	11	19
1	17	13
2	23	7
3	29	1

Possible sources of skew include linecard launch timing, fiber, connector, serializer/deserializer, mux chip. The worst-case skew depends on the system design. Therefore the system designer must determine the worst-case skew in each direction relative to Dataslice 0, and select a programmed token delay value that will provide adequate protection against both worst cases. The smallest sufficient token delay should be chosen, because the programmed token delay adds to the (E)PP/DS contribution to the RTT and decreases the budget for the rest of the system (see C.1.2).

In the transmit direction from the Dataslice to the linecard there is no synchronization. Each Dataslice makes its own decision, at every cell time, as to whether or not it has a cell to send. An Idle cell is transmitted whenever the output FIFOs is empty. The Dataslice egress start-of-cell synchronization is determined independently by each Dataslice based on the asynchronous reset pulse. Further, the empty to non-empty transition of the output FIFOs can be skewed by up to one clock cycle in the 150MHz transmit clock domain. Therefore, it is possible for transmit cells to be skewed by up to one cell time or six 150MHz transmit clock cycles at the Dataslice egress, as shown in Figure 93. Consequently, the linecard must resynchronize the incoming slices to form the complete cell.

1. All ETT1 devices operate from the same core clock of 200MHz.

Figure 93. Cells May Be Skewed Between Dataslices



C.2.2.2 Channel Errors

Each link in the channel can, of course, suffer bit errors as discussed in the section on link errors above. However, the channel can incur a particular type of error which the system designer must be aware of.

As mentioned above, it is possible for three or more data words in a data cell to be corrupted to look like control words. The Dataslice decides whether a cell is a data cell or Idle cell based on the majority of words seen. It is important to understand that each Dataslice decides *independently* whether a received cell is an idle cell or a data cell. This decision is based on a majority vote among the six characters received in a cell time: Kxx.x characters identify idle cells and Dxx.x characters identify data cells. It is very unlikely but nevertheless possible that a burst of bit errors on the fiber corrupt more than half of the characters in a cell time in such a way that an idle cell is mistaken for a data cell and vice-versa. Because idle cells are not written into the input FIFO of the Dataslice this condition causes every subsequent cell to be misaligned as it is read from the input FIFO and passed on to the port processor.

There are cases where the corruption of control words is recognized by the Dataslice and signalled to the linecard by dropping the ingress ready status (DIR#9 and revert to sending idle-not-ready cells). In other cases, this condition is only recognized by the port processor when the LCS header is misaligned and in that case LCS CRC error interrupts will be raised. If the misalignment occurs in the cell payload, the linecard must recognize this condition either by checking CRCs on a cell or packet level or by means of a keep-alive watchdog traffic stream (for example, a loopback TDM connection). The recommended recovery actions from this rare condition require the port to be taken down and brought back up again.

C.3 MANAGING CELL RATES (SETTING THE IDLE COUNT REGISTER)

To avoid cell loss due to buffer overrun it is necessary that the linecard does not send cells at a rate that exceeds the ETT1 port cards ability to accept cells. Note: by buffer overrun we refer to the shallow, rate-matching FIFOs within the various devices; we do not mean the ingress or egress queues. The same is true for the ETT1 port card: it should not send cells at a rate that will cause cell loss at the linecard.

The Enhanced Port Processor ensures this by periodically inserting an Idle cell in the egress cell stream (to the linecard). Clearly, if the linecard can accept a cell rate that is fundamentally greater than the maximum rate at which cells are sent, then there is no problem.

In practice, however, the rate at which cells are transmitted and the rate at which cells can be accepted are determined by the clocks at each end. These clocks have some inherent imprecision, usually expressed as some number of parts per million. This means that the true rate at which cells can be transmitted or accepted might be slightly greater or smaller than the nominal rate. The remainder of this section is an example to explain this in more detail.

Example: We assume that the ETT1 port and Linecard operate at a nominal frequency of 200MHz with an imprecision of +/-200ppm. One (OC-192c) cell time is eight clocks or 40ns, corresponding to a nominal rate of 25M cells/s +/-5000 cells/s.

1. Linecard to ETT1 Port.

   Assuming a slow clock at ETT1, then the ETT1 port may only be able to accept 24,995,000 cells/s. The linecard must not exceed this rate. So, if the linecard has maximum frequency clock (200.04MHz) then it could send 25,005,000 cells/s. So, to avoid overrunning the ETT1 port the linecard must send 10,000 Idle cells per second to bring its effective cell rate down to that of the ETT1 port.

2. ETT1 Port to Linecard.

   In this direction the worst case is where the ETT1 port has the highest frequency clock and the linecard has the lowest frequency. So the ETT1 port must insert 10,000 Idle cells per second. This would be done by programming the value 2,499 into the Idle Count Register in the Enhanced Port Processor. This causes the Enhanced Port Processor to insert one Idle cell for every 2,499 cells transmitted.

C.4 AVOIDING DELAYS DUE TO IDLE CELLS

Figure 94 shows a simple view of a linecard. A grant is received from the ETT1 port and the linecard must immediately respond with the corresponding cell body. However, if the Idle counter expires just at that moment, then the linecard cannot send the cell body; it must send an Idle cell instead. This behavior can cause a cell body to be delayed, thus diminishing the effective round-trip time available to the linecard, as well as introducing extra queuing into the linecard.

Figure 94. The Idle Counter Can Delay an Outbound Cell

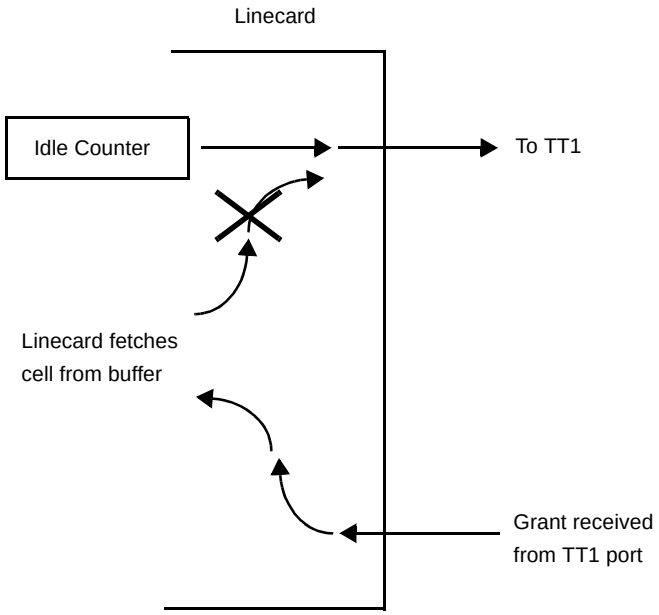
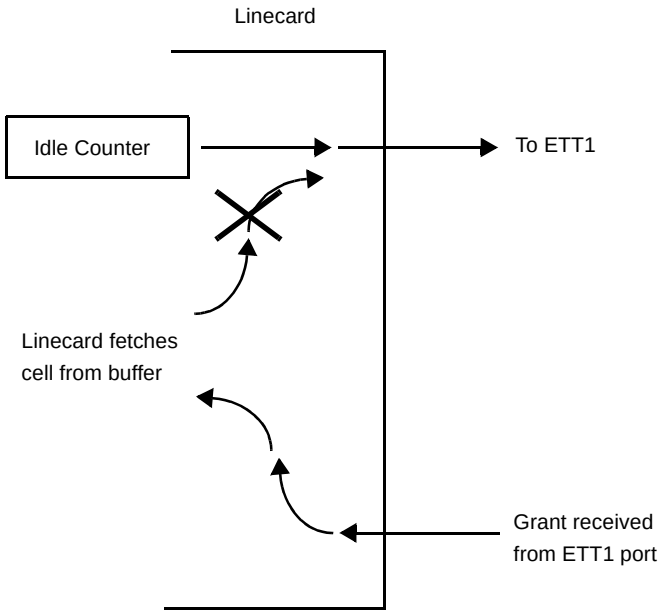


Figure 95. The Idle Counter Can Delay an Outbound Cell



To avoid this, we suggest that the linecard implement the following protocol:

1. Whenever the Idle counter expires a flag F is set to 1 and the Idle counter starts again.

2. Whenever the linecard receives an LCS cell that does *not* have a grant *and* F = 1 then reset F to 0 and send an Idle cell.

This very simple algorithm ensures that an Idle cell can be sent whenever the linecard does not have to respond to a grant. The only requirement here is that the ETT1 port must send a minimum rate of non-grant cells. This can be achieved in one of two ways:

1. Program the Idle Count register in the EPP to send Idles slightly more frequently than the linecard.
2. Program the NoGrant register in the EPP to restrict the rate of grants coming from the EPP.

NOTE: The maximum number of cells between Idles sent by the linecard, if it follows this algorithm, is then set by the following equation:

max_non_idle = linecard_idle_count + min(EPP_idle_count, EPP_NoGrant_count)

This means that the just setting the Idle count at the linecard is not sufficient to ensure a minimum Idle cell rate; the rate at which non-grant cells are received must be included in the equation.

The only further complication is that the linecard also sends out requests for new cells. However, it is assumed that requests (new cells) arrive at the linecard at a slower rate than they can be forwarded to the ETT1 fabric due to speedup within the fabric. Given this, then the linecard just needs to provide a one-deep queue for outbound requests, as shown in Figure 96. Given the higher rate at which requests are sent, then the request FIFO will never overflow.

Figure 96. One-deep Queue for Outbound Requests