CRAY T3D™ Hardware Reference Manual

Volume 1 of 2

CMM-0602-0A0

Cray Research, Inc.

Any shipment to a country outside of the United States requires a letter of assurance from Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be directed to:

CRAY RESEARCH, INC.
Logistics
6251 South Prairie View Road
Chippewa Falls, WI 54729

Comments about this publication should be directed to:

CRAY RESEARCH, INC.
Hardware Publications and Training
890 Industrial Blvd.
Chippewa Falls, WI 54729

Record of Revision

Each time this manual is revised and reprinted, all changes issued against the previous version are incorporated into the new version, and the new version is assigned an alphabetic level which is indicated in the publication number on each page of the manual.

Changes to part of a page are indicated by a change bar in the margin directly opposite the change. A change bar in the footer indicates that most, if not all, of the page is new. If the manual is rewritten, the revision level changes but the manual does not contain change bars.

REVISION	DESCRIPTION
	October 1993. Original printing.
001	February 1994. This change packet adds control panel switch information and an index to the *CRAY T3D Hardware Reference Manual.*
0A0	May 1994. This revision to the two-volume hardware reference manual replaces the previous version. It adds various technical corrections and new information covering the control switch panel.

CONTENTS – VOLUMES 1 AND 2

VOLUME 1

VOLUME 2

PREFACE

This manual is written to assist Cray Research, Inc. (CRI) field engineers in maintaining the CRAY T3D system. It introduces the CRAY T3D product and describes its components.

The intended audience for this manual comprises CRI field engineers, product specialists, and systems test personnel. This manual is designed for classroom and reference use and is a CRI company proprietary document.

This manual is a two-volume set, the organization of which is described below.

> Volume 1 introduces and describes the CRAY T3D system architecture, interconnect network, processing element node, I/O gateway, system initialization, addressing, and operations.
>
> Volume 2 describes the block transfer engine, barrier synchronization, control and status, and LOSP and HISP channels. There is also a section that contains reference charts and diagrams, and a section that contains control panel switch information.

Any comments or suggestion for improvements to this publication may be submitted using one of the reader comment forms included in the front and back of each volume.

CONTENTS

1 ARCHITECTURE OVERVIEW

2 INTERCONNECT NETWORK

2 INTERCONNECT NETWORK (continued)

3 PROCESSING ELEMENT NODE

4 I/O GATEWAY

5 STAND-ALONE DIAGNOSTIC SYSTEM INITIALIZATION

6 ADDRESSING

6 ADDRESSING (continued)

7 OPERATIONS

FIGURES

FIGURES (continued)

FIGURES (continued)

FIGURES (continued)

FIGURES (continued)

TABLES

TABLES (continued)

1 ARCHITECTURE OVERVIEW

Cray Research is implementing a three-phase MPP program. Our goal is to reach a sustained performance on real customer code of one trillion floating-point operations per second. This manual describes the architecture, functions, and hardware of the first-phase MPP system, the CRAY T3D massively parallel system.

The CRAY T3D system contains hundreds or thousands of microprocessors, each accompanied by a local memory. The microprocessors and their associated local memory are located in the processing element nodes. These processing element nodes are linked by a three-dimensional interconnecting network, which facilitates rapid parallel processing.

The system is designed to support three styles of MPP programming: work sharing, data parallel, and message passing. Cray Research supports all three styles of programming so that customers may port a program from another vendor’s MPP system to a Cray Research MPP system with a minimum of effort. For more information on Cray Research’s MPP programming models, refer to the *CRAY T3D Software Overview Technical Note,* publication number SN-2505.

The CRAY T3D system connects to a host computer system. The host system provides support for applications running on the CRAY T3D system. All programs written for the CRAY T3D system are compiled on the host system but run on the CRAY T3D system.

The host system may be any Cray Research computer system that has an input/output subsystem model E (IOS-E). Host systems include the CRAY Y-MP E series computer systems, the CRAY Y-MP M90 series computer systems, and the CRAY C90 series computer systems.

The following subsections describe the basic functions and hardware of the CRAY T3D system.

Functional Description

The CRAY T3D system contains four types of components: processing element nodes, an interconnect network, I/O gateways, and a clock. Figure 1-1 shows a simplified model of the components.

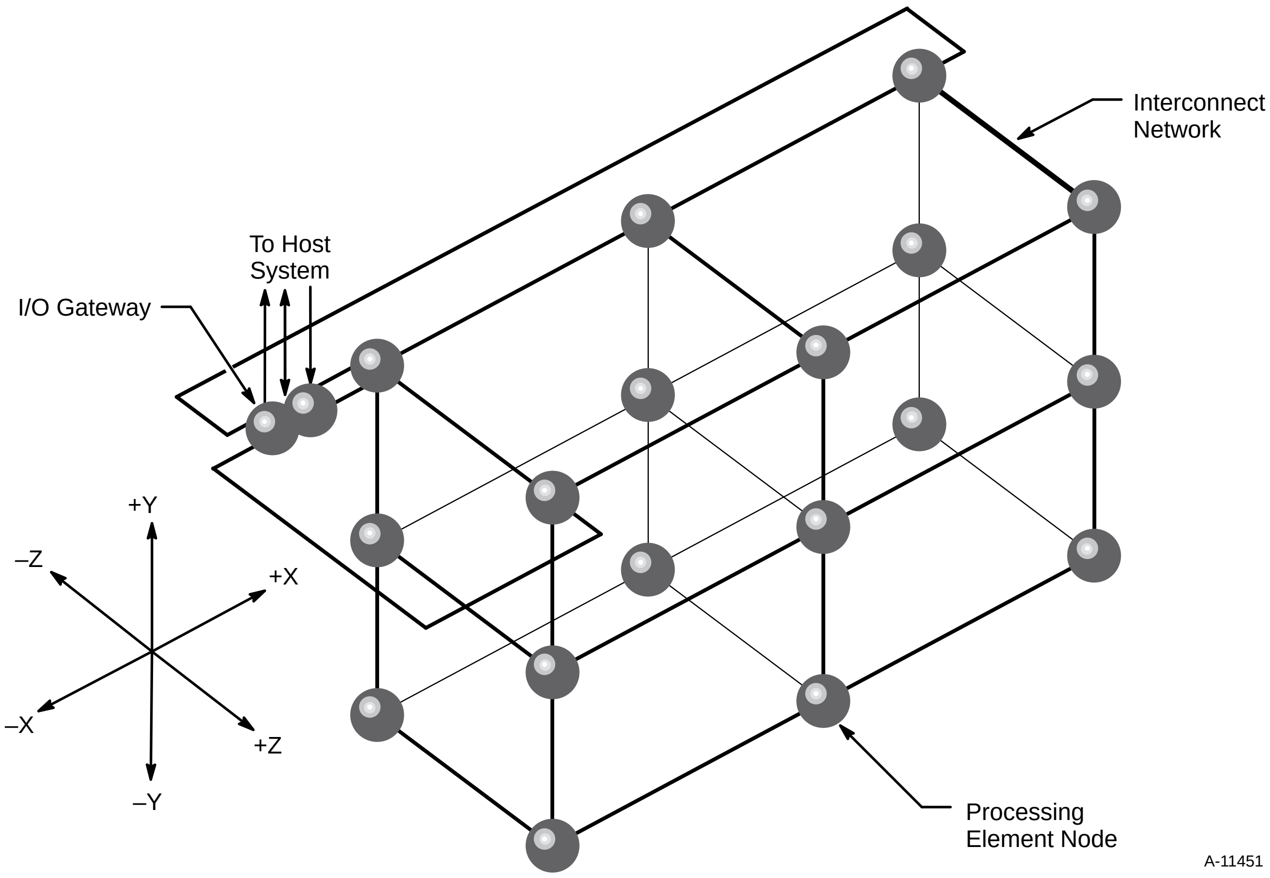


Figure 1-1. CRAY T3D System Components

The following subsections describe each component of the CRAY T3D system.

Processing Element Nodes

As previously stated, an MPP computer system contains hundreds or thousands of microprocessors, each accompanied by a local memory. The microprocessors and associated local memory are located in processing element nodes. Each processing element node contains two processing elements (PEs). In the CRAY T3D system, each PE contains a microprocessor, local memory, and support circuitry (refer to Figure 1-2).

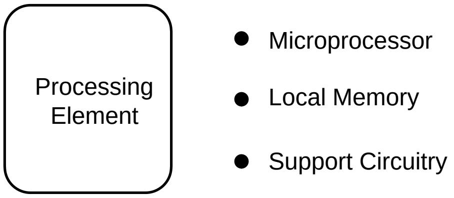


Figure 1-2. Processing Element Components

The microprocessor is a reduced instruction set computer (RISC) 64-bit microprocessor developed by Digital Equipment Corporation. The microprocessor performs arithmetic and logical operations on 64-bit integer and 64-bit floating-point registers. The microprocessor also contains an internal instruction cache memory and an internal data cache memory. Each stores 256 lines of data or instructions. Each line is four 64-bit words wide.

Local memory consists of dynamic random access memory (DRAM) that stores system data. A low-latency, high-bandwidth data path connects the microprocessor to local memory in a PE.

Together, the local memory of all PEs makes up the CRAY T3D system memory. The CRAY T3D system memory is physically distributed and logically shared. It is physically distributed because each PE contains local memory. The system memory is logically shared because the microprocessor in one PE can access the memory of another PE without involving the microprocessor of that PE.

The support circuitry extends the control and addressing functions of the microprocessor. This includes performing data transfers to or from local memory.

As mentioned, each processing element node contains two PEs. Each processing element node also contains a network interface and a block transfer engine (BLT). Refer to Figure 1-3. The following paragraphs briefly describe each of these components.

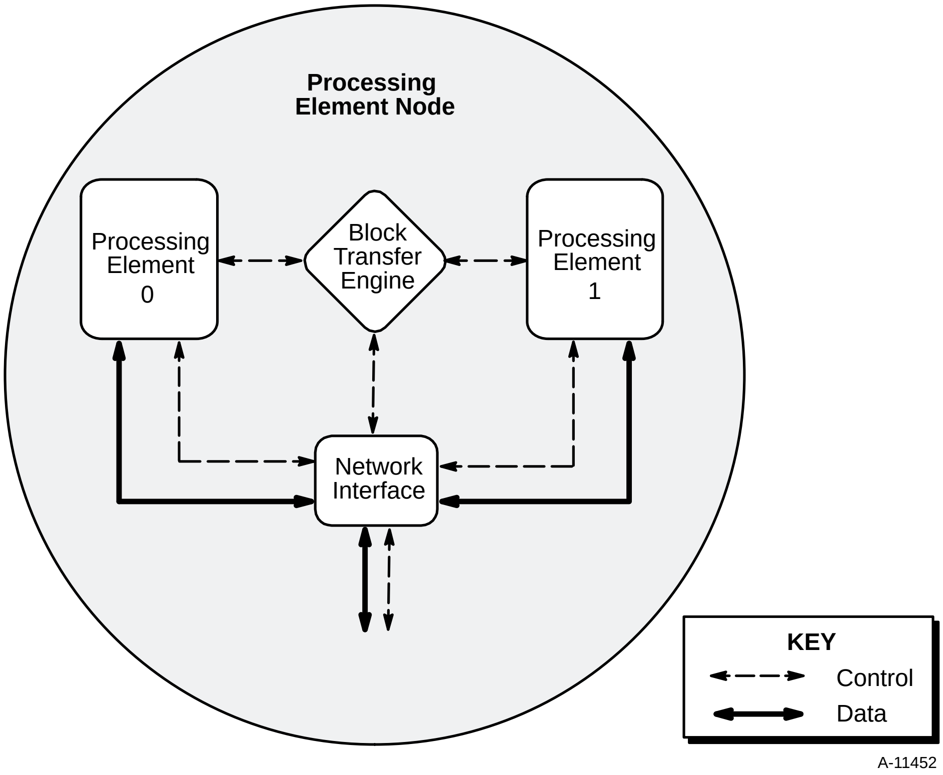


Figure 1-3. Processing Element Node

The two PEs in a processing element node are identical but function independently. The PEs share access to the BLT and network interface.

The network interface formats information before it is sent over the interconnect network to another processing element node or to the I/O gateway. The network interface also receives incoming information from another processing element node or from the I/O gateway and steers the information to PE 0 or PE 1 in the processing element node.

The BLT is an asynchronous direct memory access controller that redistributes system data. The BLT redistributes system data between the local memory in PE 0 or PE 1 and the memory in remote PEs. The BLT can redistribute up to 65,536 64-bit words of data (or 65,536 4-word lines of data) without interrupting the PE.

Interconnect Network

The interconnect network provides communication paths among the processing element nodes and the I/O gateways in the CRAY T3D system. The interconnect network forms a three-dimensional matrix of paths that connect the nodes in the X, Y, and Z dimensions (refer again to Figure 1-1).

The interconnect network is composed of communication links and network routers. The communication links transfer both data and control information between the network routers in the interconnect network. The network routers transfer the data and control information through the communication links in the interconnect network. Figure 1-4 shows how the components of the interconnect network connect to a processing element node.

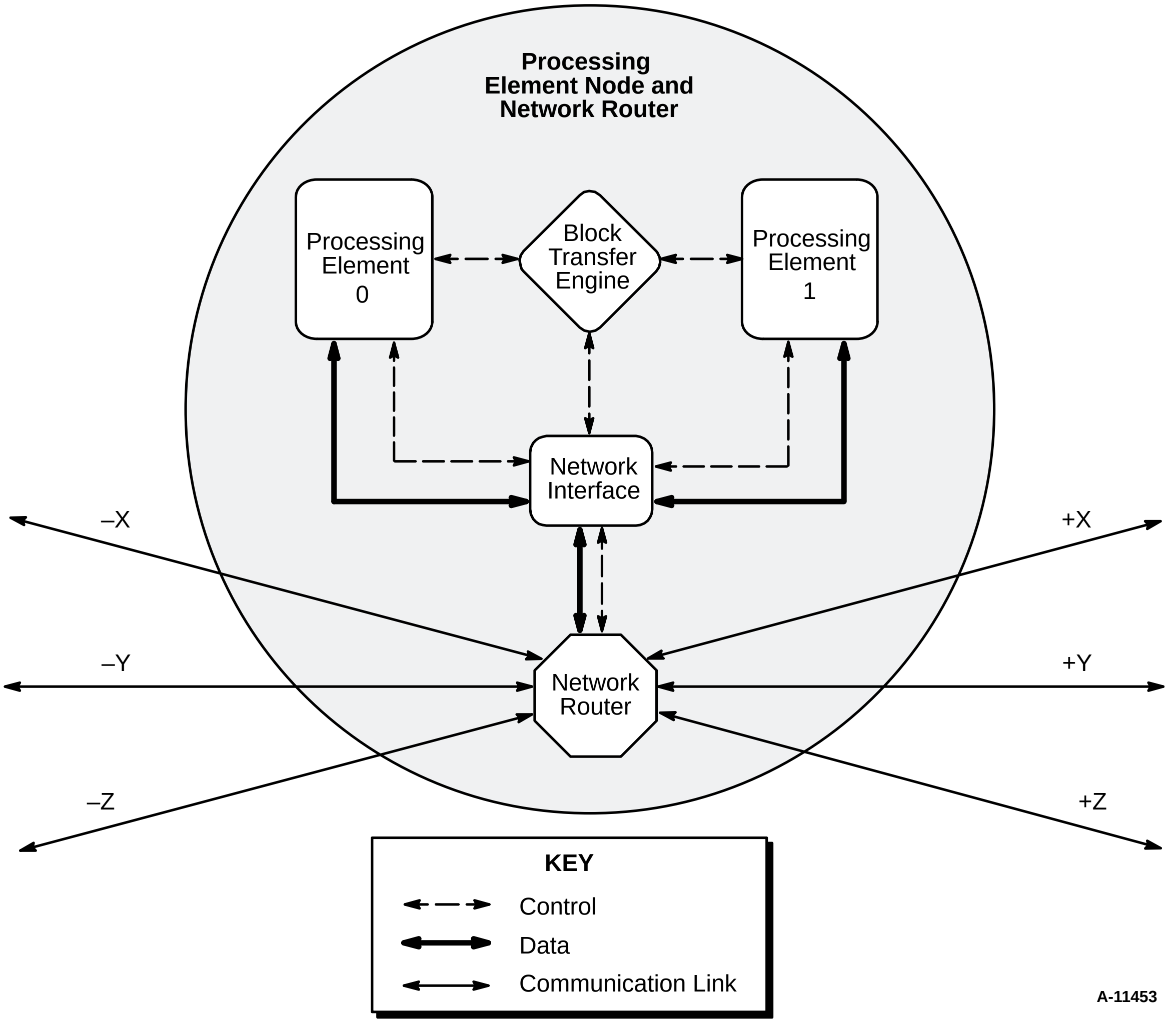


Figure 1-4. Interconnect Network Components

I/O Gateways

I/O gateways transfer system data and control information between the host system and the CRAY T3D system. The I/O gateways connect to the interconnect network through network routers that have communication links in the X and Z dimensions only. An I/O gateway can transfer information to any PE in the interconnect network.

An I/O gateway contains an input node, an output node, and low-speed (LOSP) circuitry. Figure 1-5 shows the components of an I/O gateway.

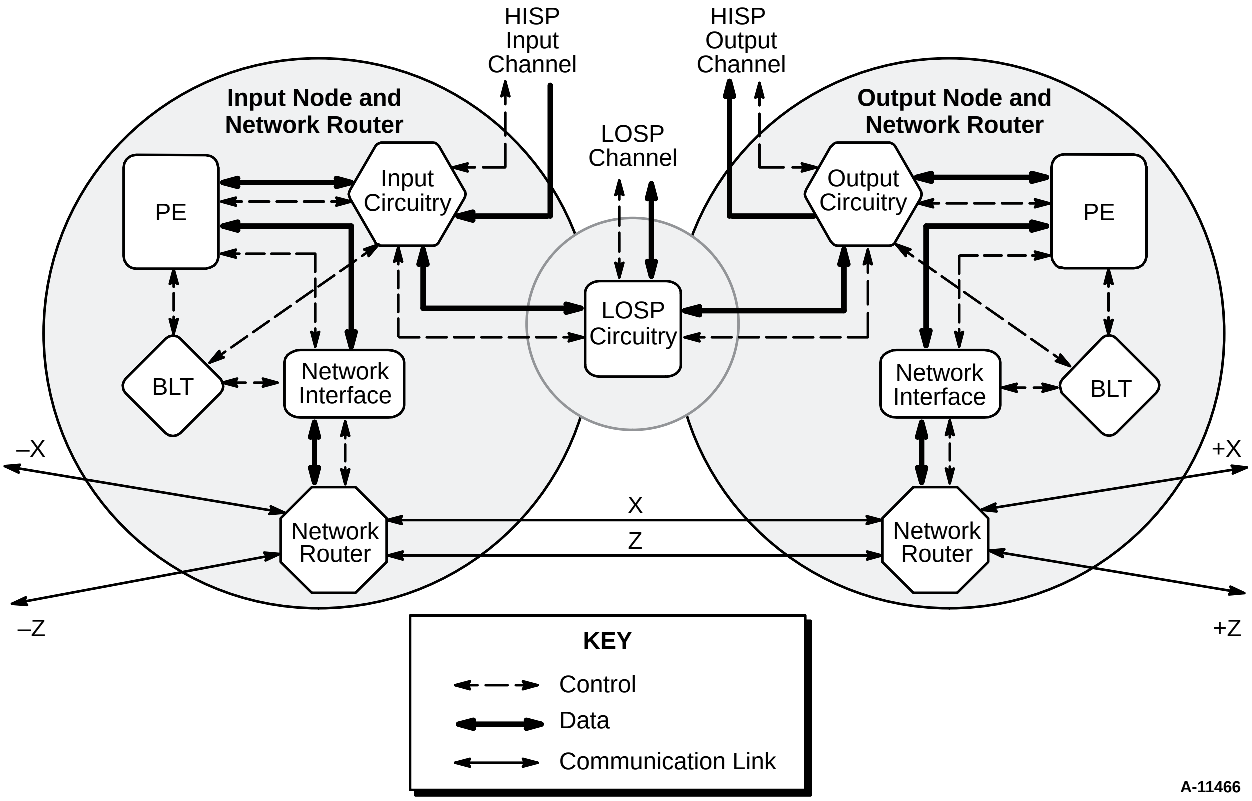


Figure 1-5. I/O Gateway

The input node contains one PE, a network interface, a block transfer engine, and high-speed (HISP) input circuitry. The BLT and network interface in the input node are identical to the BLT and network interface used in a processing element node.

The PE in the input node is designed to interface with the HISP input circuitry. Because of this characteristic, the PE in the input node does not contain the circuitry to perform all of the operations that a PE in a processing element node performs. The PE in an I/O gateway does not perform the following functions and operations. More information on these functions and operations is provided in Section 6, “Addressing,” and in Section 7, “Operations.”

- Incoming or outgoing atomic swap operations
- Data prefetch operations
- Read-ahead operations
- Data cache-line invalidation
- Virtual PE numbers and associated virtual range check

Instead, the circuitry is replaced with circuitry that interfaces with the HISP input circuitry. In addition, the PE in the input node contains half of the local memory that a PE in a processing element node contains.

The HISP input circuitry receives incoming system data from the host system over the HISP channel. After receiving the data, the HISP input circuitry, PE, and BLT in the input node transfer the data to the PEs in the CRAY T3D system.

Except for the HISP output circuitry, the output node is identical to the input node. The HISP output circuitry transmits outgoing system data to the host system over the HISP channel. After the PE, BLT, and HISP output circuitry in the output node receive data from the PEs in the CRAY T3D system, the HISP output circuitry transfers the data to the host system.

The LOSP circuitry transfers request and response information over the LOSP channel that connects the host system and the CRAY T3D system. This LOSP request and response information is used to control the transfer of system data over the HISP channel.

There are two types of I/O gateways: a master I/O gateway and a slave I/O gateway. The two types of I/O gateways correspond to the two types of components connected by a HISP channel. The master I/O gateway is the master component of a HISP channel and sends the address information to the host system during a HISP transfer. The slave I/O gateway is the slave component of a HISP channel and receives the address information from the host system during a HISP transfer.

Clock

The CRAY T3D system contains a central clock that provides a 6.60-ns clock signal. The clock signal is fanned out to all of the processing element nodes and I/O gateways in the system.

Hardware Description

The CRAY T3D system hardware uses current CRAY Y-MP technology. A system consists of processing element modules, I/O modules, a system/clock module, and a spare processing element module. All of the module types are single cold plate, CRAY Y-MP style modules. The following subsections describe the CRAY T3D modules.

Processing Element Module

A processing element module contains four processing element nodes, two per printed circuit board. Figure 1-6 shows the components of a processing element printed circuit board.

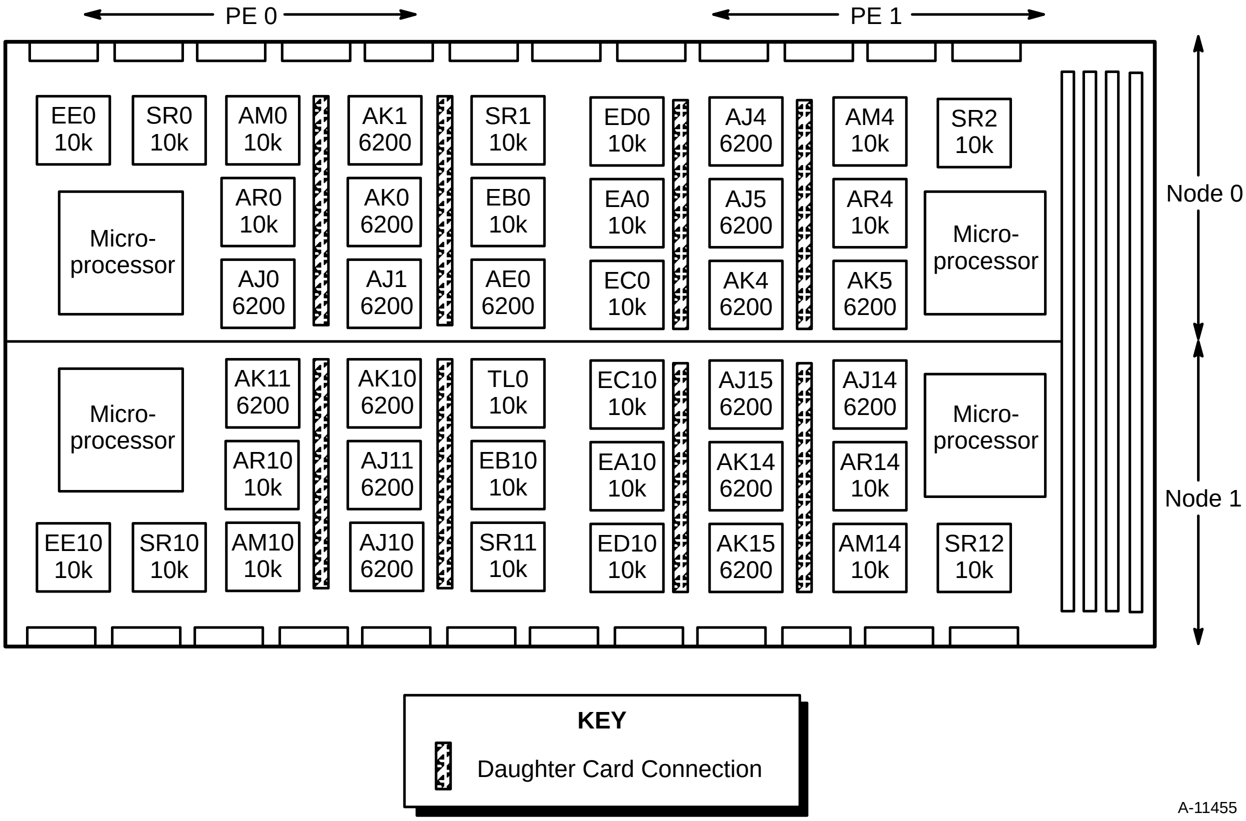


Figure 1-6. Processing Element Printed Circuit Board

Each processing element node contains two PEs. Each PE consists of a microprocessor, PE support circuitry, and local memory. Figure 1-7 highlights the microprocessor and PE support circuitry of PE 0 in node 0.

The PE support circuitry is made up of 6200- and 10k-gate array logic. This logic provides control and extends the address for the microprocessor. The remaining logic (E- and S-options) is shared by the two PEs within the processing element node.

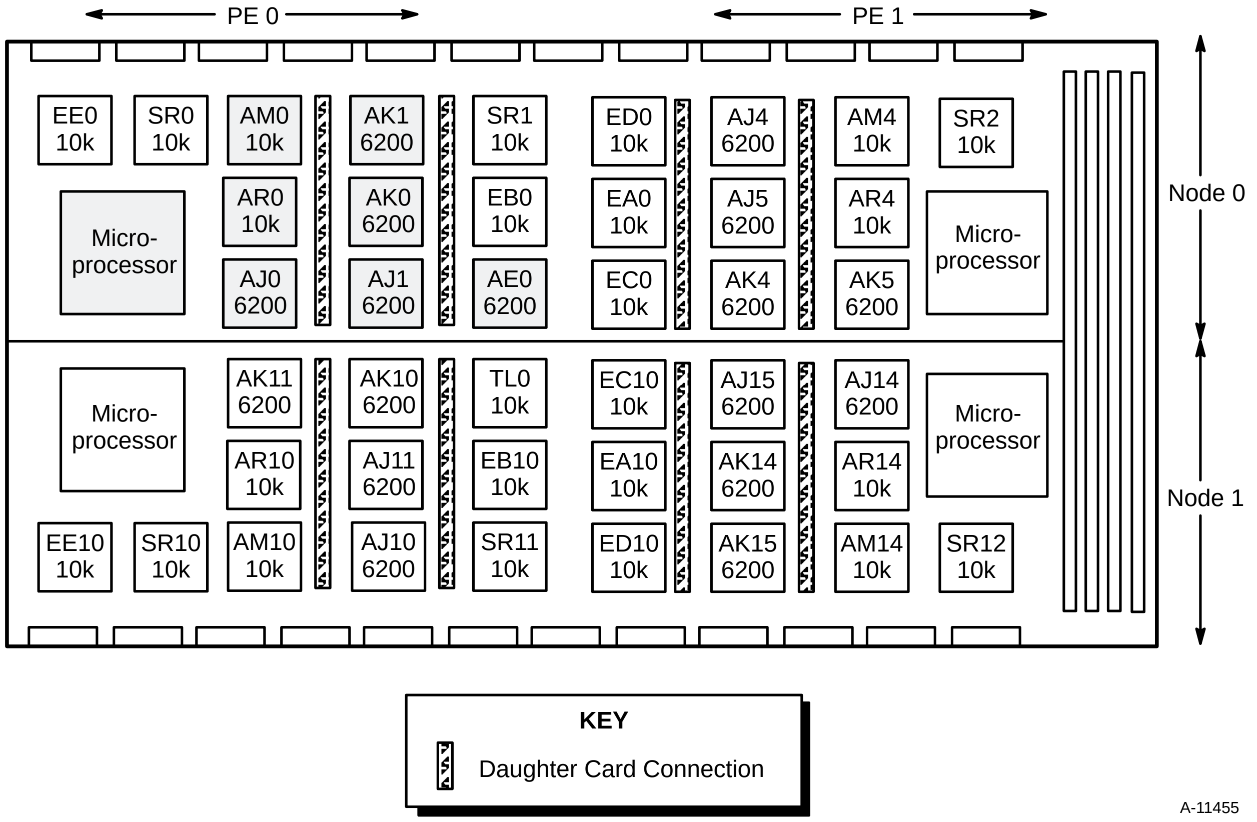


Figure 1-7. Components of PE 0 in Node 0

Local memory resides on a separate printed circuit board called a daughter card (refer to Figure 1-8). The daughter card attaches to the processing element module via a pin and socket type connection. The daughter cards are field replaceable units.

The two daughter cards of a PE contain either 2 million or 8 million words of storage. The memory is dynamic random access memory (DRAM) using 256k x 16 CMOS DRAM memory parts for 2 million words of storage and 1M x 16 CMOS DRAM memory parts for 8 million words of storage.

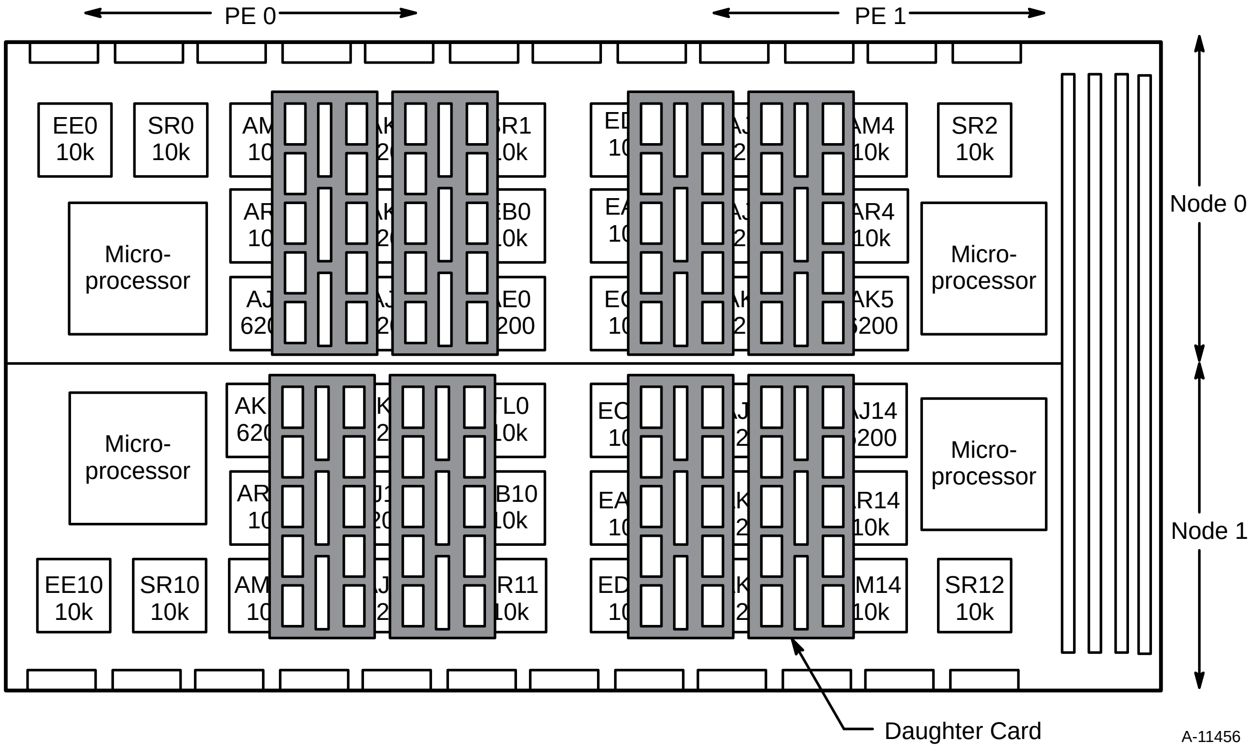


Figure 1-8. Processing Element Printed Circuit Board With Daughter Cards

Spare Processing Element Module

The spare processing element module is identical to the processing element module. Each processing element node on this module can be configured into the system to replace a faulty node.

I/O Module

An I/O module contains two I/O printed circuit boards. These two printed circuit boards can be two master I/O printed circuit boards, two slave I/O printed circuit boards, or one master I/O printed circuit board and one slave I/O printed circuit board. The master I/O printed circuit board controls the HISP transfers and is used when a HISP channel is connected to the mainframe. The slave I/O printed circuit board is used when a HISP is connected to an I/O cluster. In this case, the I/O cluster controls the HISP transfers.

Each I/O printed circuit board contains one I/O gateway. Each I/O gateway consists of an input node, an output node, and LOSP control circuitry. Figure 1-9 shows the hardware of a master I/O gateway or printed circuit board.

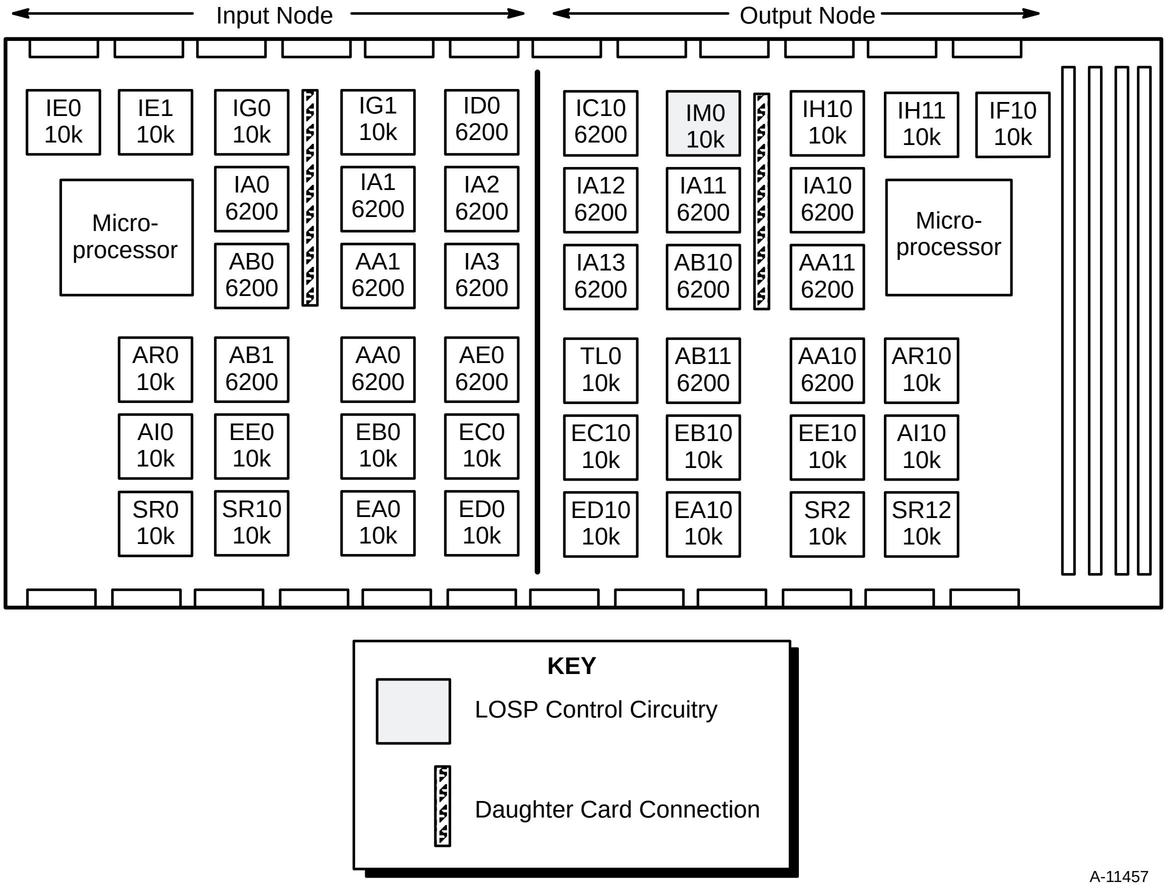


Figure 1-9. Master I/O Printed Circuit Board

System/Clock Module

The system/clock module contains clock circuitry and system fan-out circuitry. The clock circuitry resides on one printed circuit board (refer to Figure 1-10). This circuitry provides a 6.67-ns clock pulse that is fanned out to all of the modules within the CRAY T3D system. The system circuitry resides on the adjacent printed circuit board of this module (refer to Figure 1-11). The system circuitry contains logic that performs fanout of the clock signals, barrier signals, and heartbeat signals.

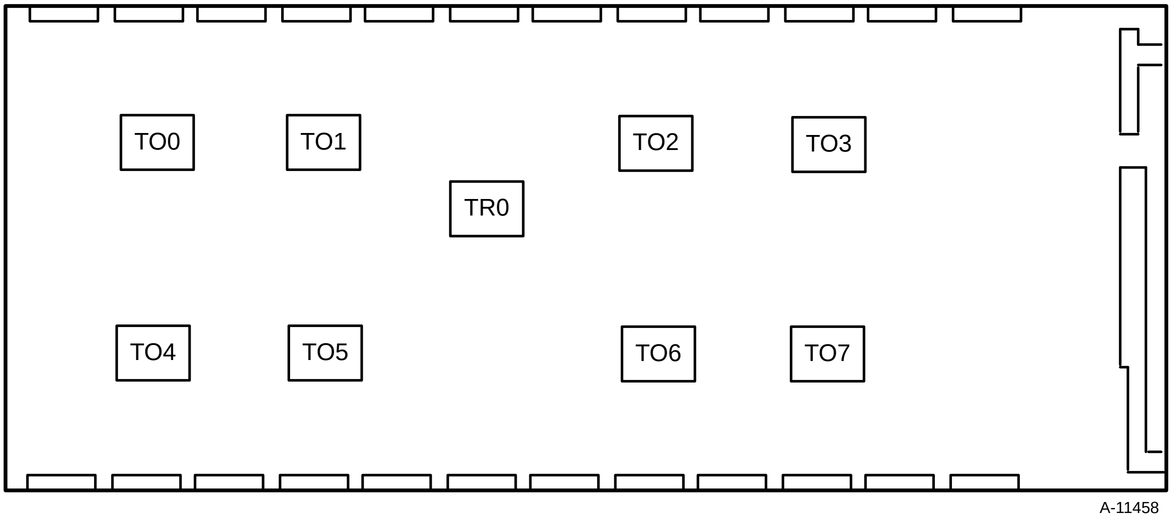


Figure 1-10. Clock Printed Circuit Board

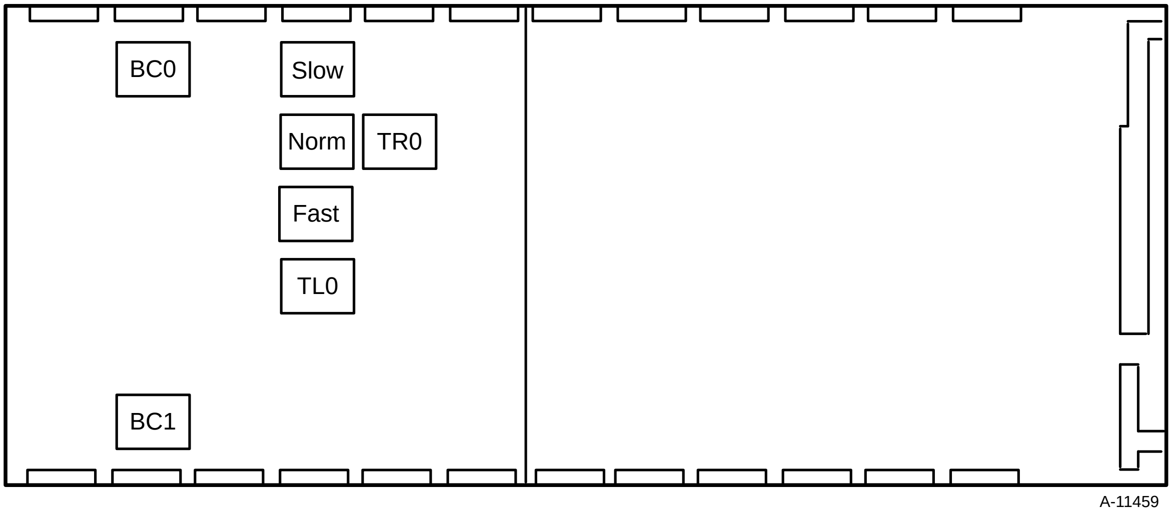


Figure 1-11. System Printed Circuit Board

System Cabinets

Together, the PE, I/O, system/clock, and spare modules make up the CRAY T3D system. Depending upon the size and configuration, the CRAY T3D system and host system are housed in a single cabinet or in multiple cabinets.

Single Cabinet

In a single cabinet configuration, the host system and the CRAY T3D system reside in the same chassis. Figure 1-12 shows a single cabinet that contains 256 PEs.

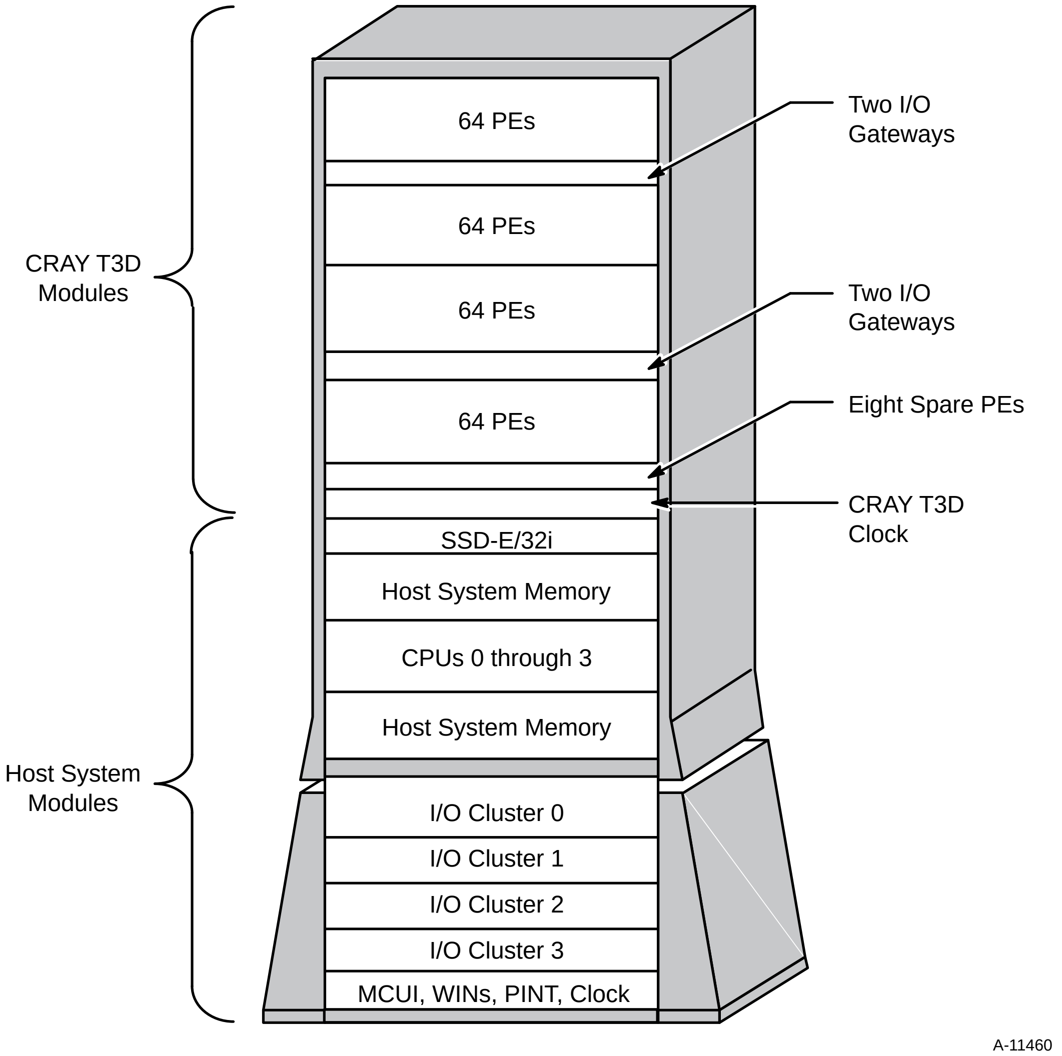


Figure 1-12. Single Cabinet 256-PE CRAY T3D System

Multiple Cabinet

In a multiple cabinet configuration, the CRAY T3D system resides in one to four separate chassis that are cabled to the host system chassis. Figure 1-13 shows an example of multiple cabinet chassis that contain 1,024 PEs.

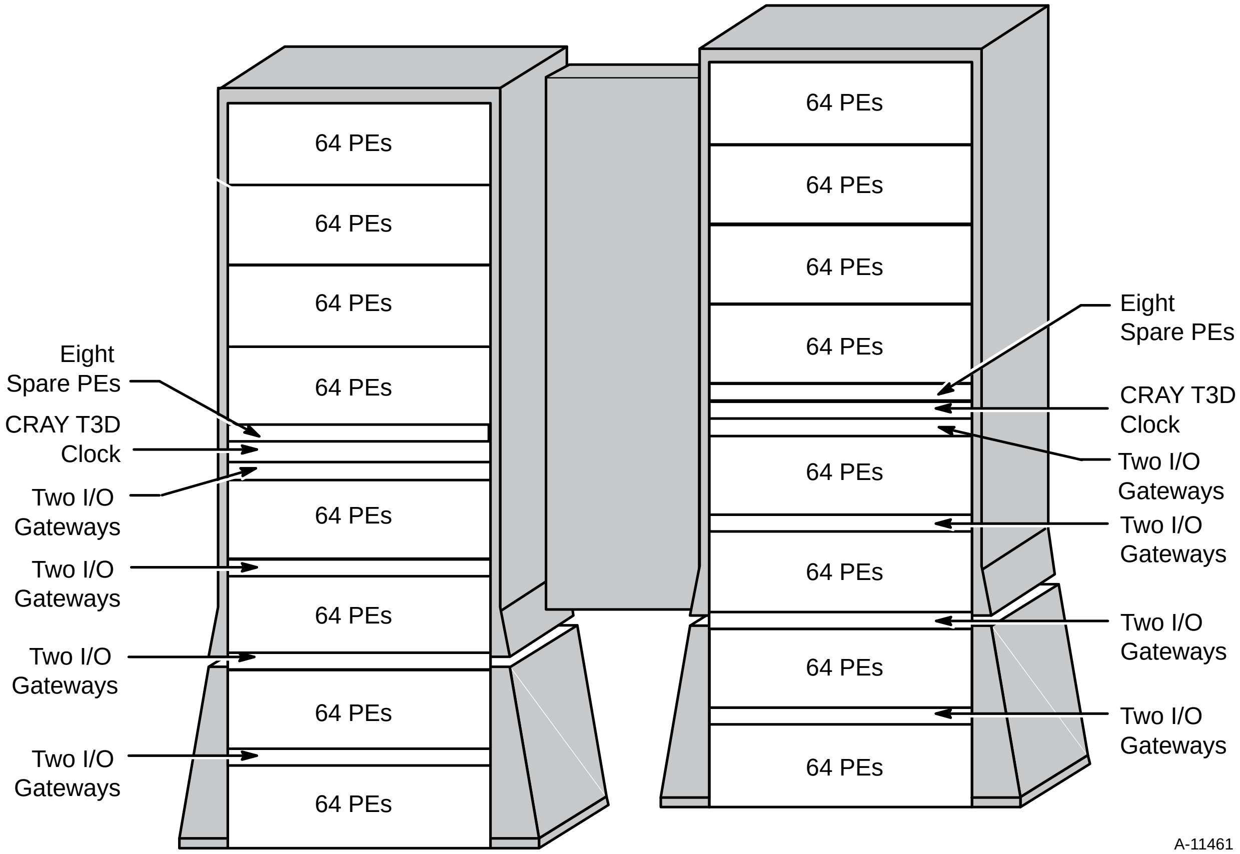


Figure 1-13. Multiple Cabinet 1024-PE CRAY T3D System

There are two types of multiple cabinet configurations: standard multiple cabinet systems and air-cooled multiple cabinet systems. More information on the different types of configurations is provided in the manual, “CRAY T3D Configurations.”

2 INTERCONNECT NETWORK

This section describes the functions and hardware of the interconnect network.

Functional Description

The interconnect network provides communication paths among the processing element nodes and the I/O gateways in the CRAY T3D system. The interconnect network is composed of communication links and network routers (refer to Figure 2-1).

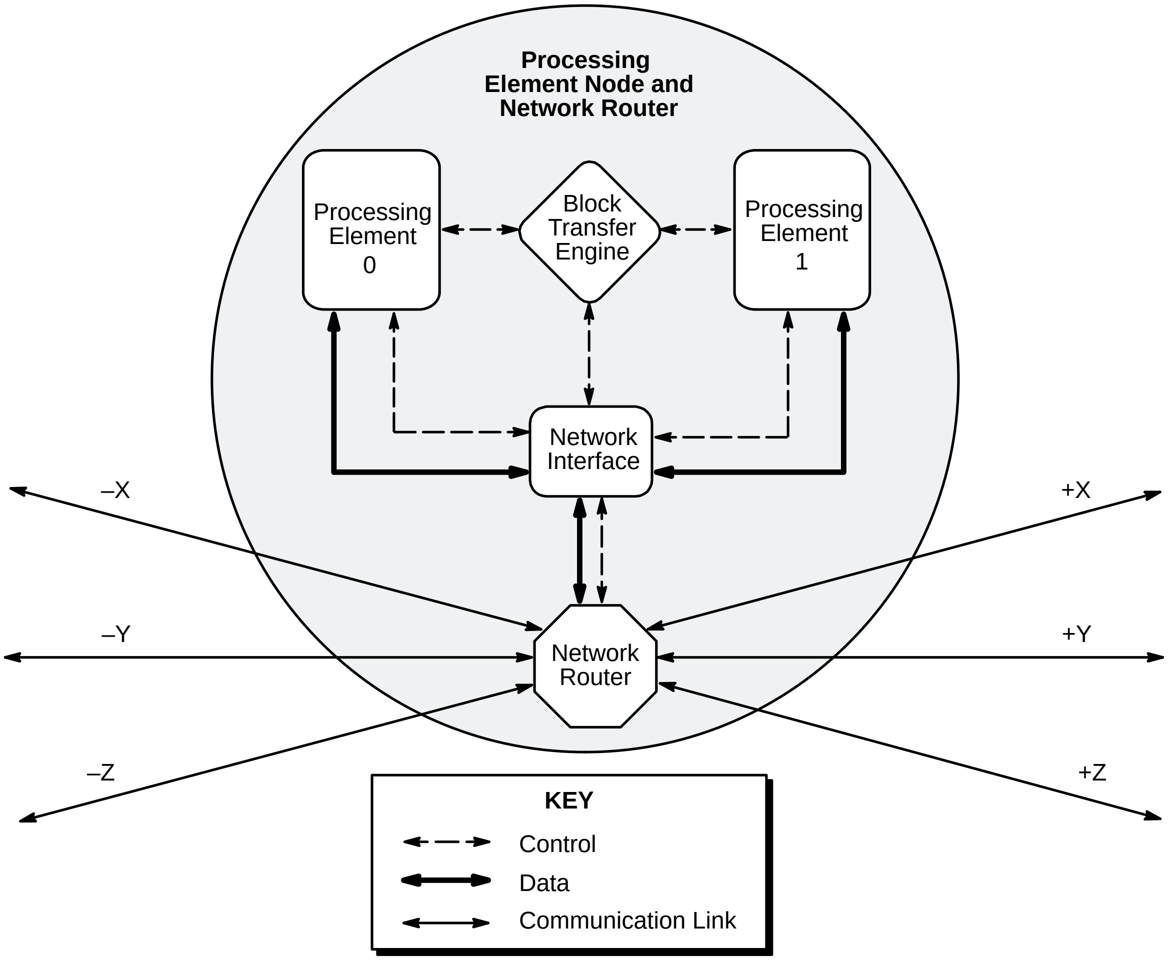


Figure 2-1. Interconnect Network Components

The following subsections describe the components and characteristics of the interconnect network.

Communication Links

Communication links transfer data and control information between the network routers in the interconnect network. Each communication link connects two nodes in one dimension (refer to Figure 2-2).

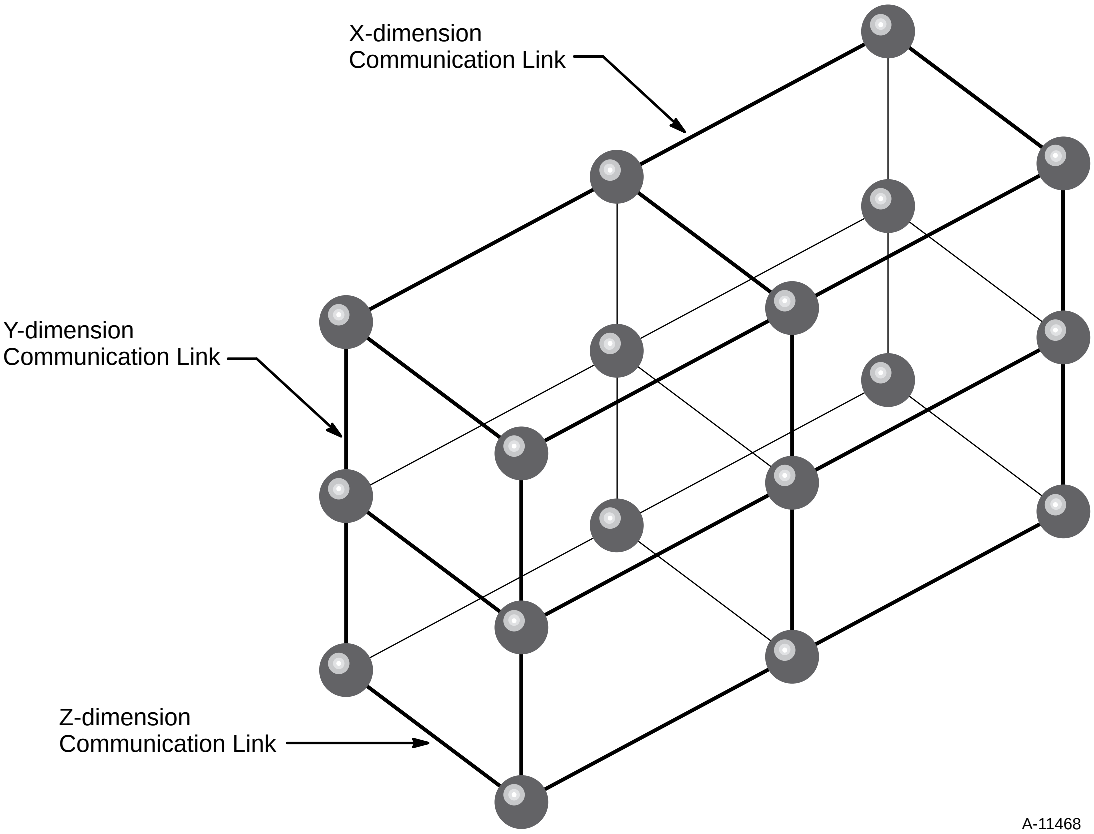

X-dimension Communication Link
Y-dimension Communication Link
Z-dimension Communication Link
A-11468

Figure 2-2. Communication Links

A communication link is actually two unidirectional channels. Each channel in the link contains Data, Physical Unit (Phit) Type, Virtual Channel Select, and Virtual Channel Acknowledge signals. Figure 2-3 shows the signals for both unidirectional channels in one communication link.

Data Signals

Each channel contains 16 data signals. Data signals carry two types of information: requests and responses. Requests contain information that requests a node to perform an activity. For example, a source node may send a request to a destination node in order to read data from memory in the destination node. This request is sent over one channel in the communication link.

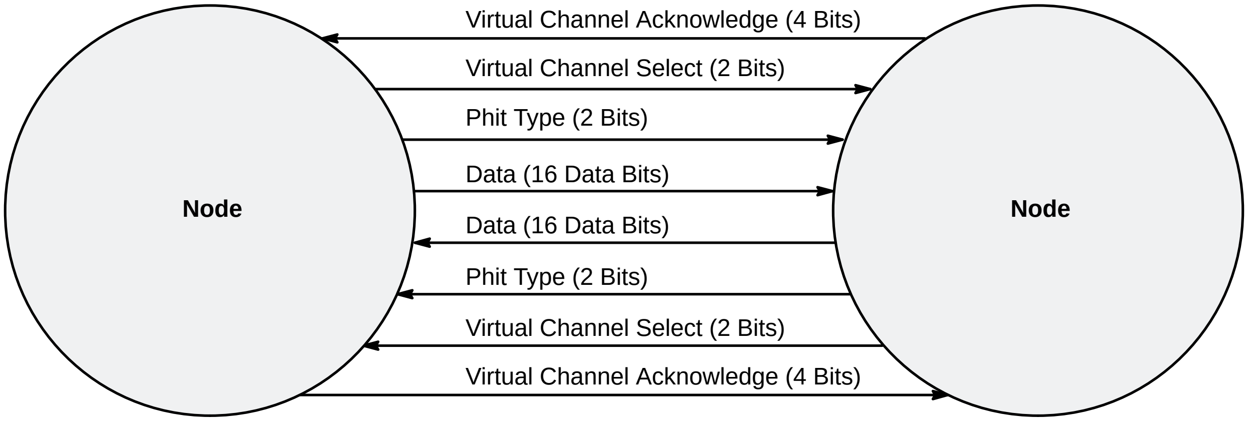

Node
Node
Virtual Channel Acknowledge (4 Bits)
Virtual Channel Select (2 Bits)
Phit Type (2 Bits)
Data (16 Data Bits)
Data (16 Data Bits)
Phit Type (2 Bits)
Virtual Channel Select (2 Bits)
Virtual Channel Acknowledge (4 Bits)

Figure 2-3. Communication Link Channel Signals

Responses contain information that is the result of an activity. For example, after receiving a request for read data, a destination node sends the response back to the source node. The response contains the read data.

Requests and responses must be logically separated. This is done by providing separate buffers for requests and responses. These buffers are used to create virtual channels. (More information on virtual channels is provided later in this section.)

Phit Type Bits

A phit is the amount of information that can be placed on a data channel in one clock period. In the CRAY T3D system, a phit is 16 bits. Multiple phits are transferred sequentially across the data channel in the form of a packet. (Information on packets is provided later in this section.)

Each channel contains 2 phit type bits that are controlled by the node sending information over the channel. These bits indicate what type of phit is on the data signals. Table 2-1 lists the definitions of the least significant bit (LSB) and most significant bit (MSB) of the phit type bits.

Table 2-1. Phit Type Bit Definitions

MSB	LSB	Data Signals Contain
0	0	No information
0	1	Packet routing tag phit
1	0	Packet phit
1	1	Last phit of packet

Virtual Channel Signals

The virtual channel signals control which virtual channel the data will use. A virtual channel is created when request information and response information transfer over the same physical communication link but are stored in separate buffers. The virtual channel signals include the virtual channel select bits and the virtual channel acknowledge bits. (More information on virtual channels is provided later in this section.)

There are two virtual channel select bits. These bits indicate which virtual channel buffer in the receiving node will store the information. Table 2-2 shows the definitions of the virtual channel select bits.

Table 2-2. Virtual Channel Select Bit Definitions

MSB	LSB	Definition	Name
0	0	Request buffer 0	Virtual channel 0
0	1	Request buffer 1	Virtual channel 1
1	0	Response buffer 0	Virtual channel 2
1	1	Response buffer 1	Virtual channel 3

The most significant bit of the virtual channel select bits indicates whether the information on the data signals is a request or a response. When set to 0, this bit indicates the information is a request. When set to 1, this bit indicates the information is a response.

The least significant bit of the virtual channel select bits indicates which of the two request buffers or two response buffers will store the information on the data signals. When set to 0, this bit indicates the information will be stored in buffer 0. When set to 1, this bit indicates the information will be stored in buffer 1.

There are 4 virtual channel acknowledge bits. Each virtual channel buffer controls one of the virtual channel acknowledge bits. For example, virtual channel buffer 2 controls bit 2 of the virtual channel acknowledge bits.

The node receiving information sets the appropriate virtual channel acknowledge bit to 1, empties the virtual channel buffer, and sends the information to another node or to a PE. Once the virtual channel buffer is empty, the first node resets the virtual channel acknowledge bit to 0. A 1 to 0 transition of the Acknowledgement signal notifies the source node that this portion of the transfer is complete.

Torus Interconnect Topology

The interconnect network is connected in a bidirectional torus. A torus contains communication links that connect the smallest numbered node in a dimension directly to the largest numbered node in the same dimension. This type of connection forms a ring where information can transfer from one node, through all of the nodes in the same dimension, and back to the original node.

Figure 2-4 shows a one-dimensional torus network in the X dimension. Information can transfer from node 00, through all of the nodes, and back to node 00 in a circular fashion. Each node has a communication link in both the positive and the negative directions of the X dimension.

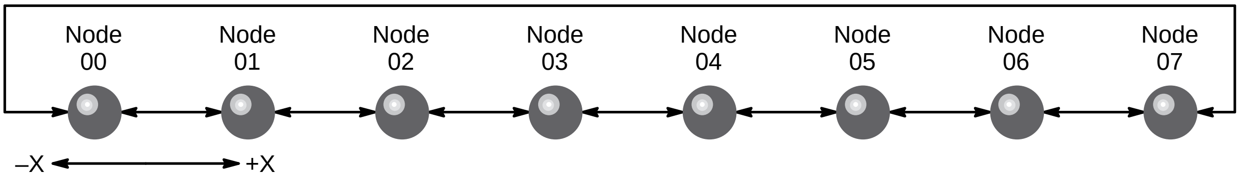


Figure 2-4. One-dimensional Torus Network

Torus networks offer several advantages for network communication. One advantage is speed of information transfers. For example, in Figure 2-4, node 07 can communicate directly with node 00 instead of sending information through all of the nodes in the X dimension.

Another advantage of the torus network is the ability to avoid bad communication links. For example, in Figure 2-4, if node 00 cannot transfer information directly to node 01 due to a bad communication link, node 00 can still communicate with node 01 by sending the information the long way around the network through the other nodes in the X dimension.

Figure 2-5 shows a two-dimensional torus network in the Y and X dimensions. Each node has communication links in both the positive and the negative directions of the Y and X dimensions.

Figure 2-6 shows a three-dimensional torus network in the Z, Y, and X dimensions. Each node has communication links in both the positive and the negative directions of the Z, Y, and X dimensions.

Several of the diagrams in this manual show three-dimensional network connections. For clarity, the communication link that completes the torus in each dimension is not shown in most diagrams. It is important to remember that although not shown in the diagrams, this communication link is always there.

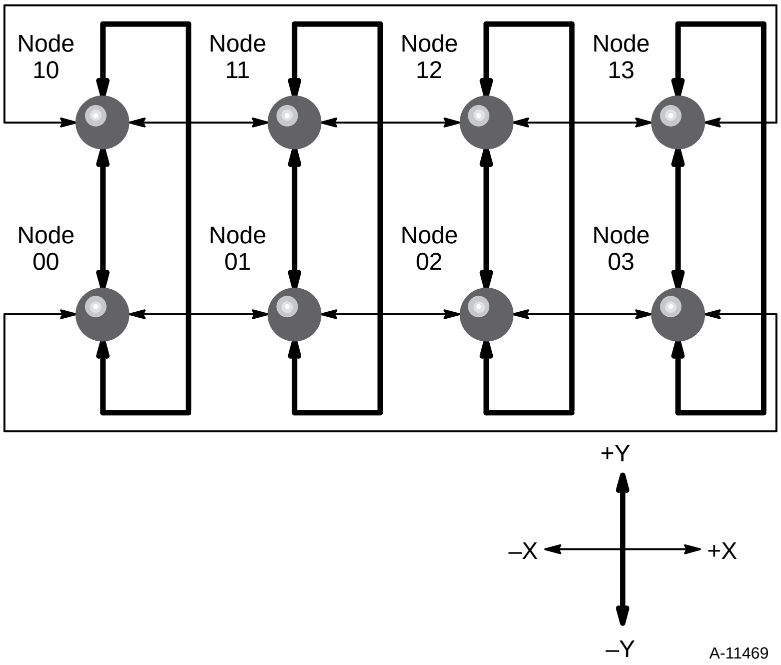


Figure 2-5. Two-dimensional Torus Network

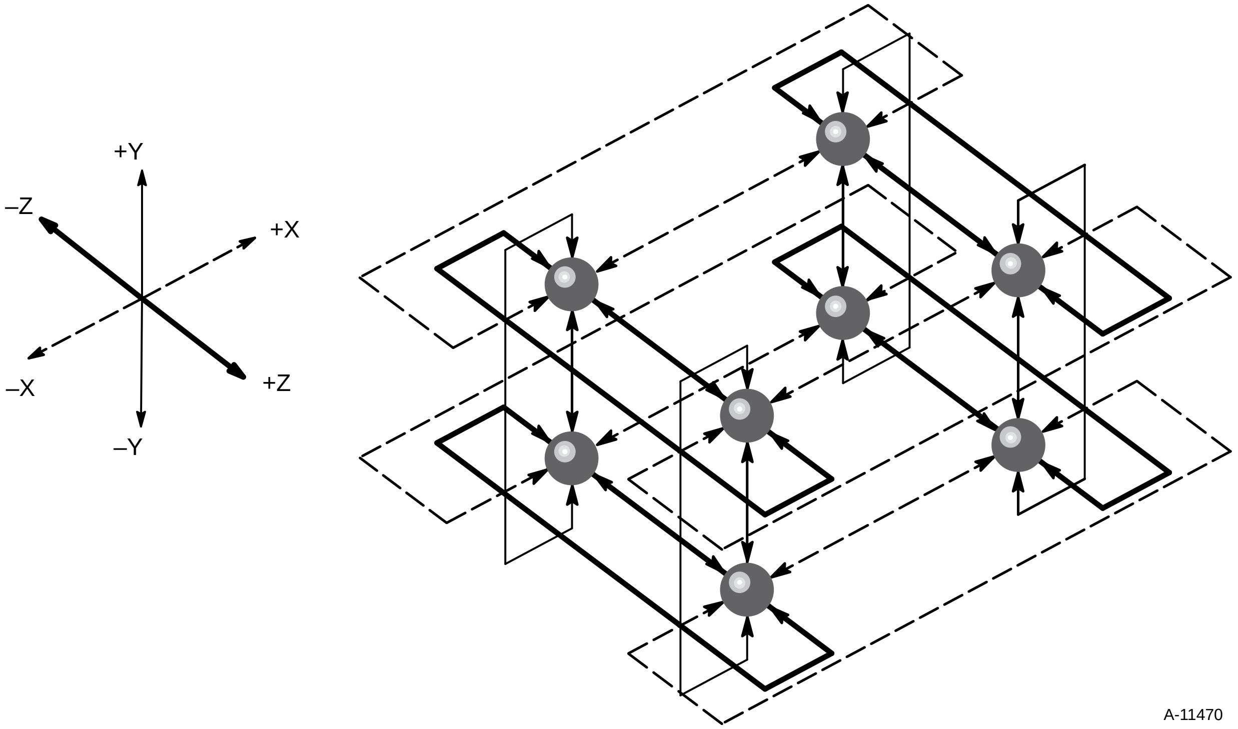


Figure 2-6. Three-dimensional Torus Network

Interleaving

The nodes in the interconnect network are interleaved. Interleaving is the physical placement of nodes so that the maximum wiring distance between nodes is minimized.

Figure 2-7 shows two one-dimensional torus networks. The eight nodes in the upper network are not interleaved. The eight nodes in the lower network are interleaved. In the interleaved network (also called a folded torus network), the physical length of the longest communication link is shorter than the physical length of the longest communication link in the noninterleaved network.

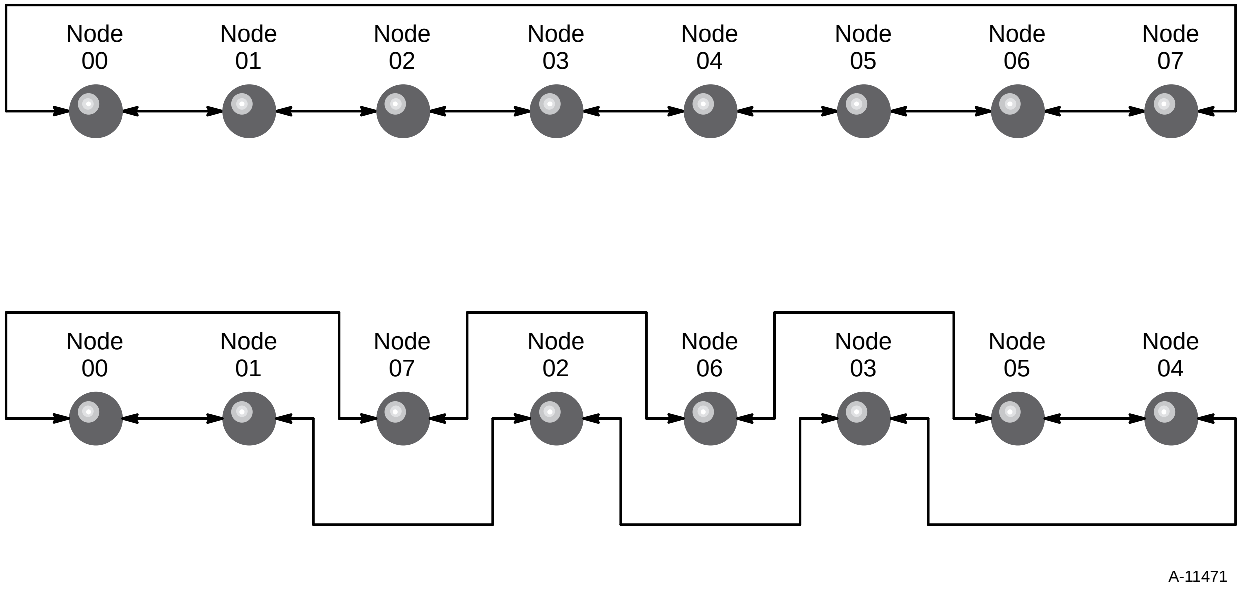


Figure 2-7. Interleaving

The X and Z dimensions of the network are interleaved. This minimizes the length of the physical communication links (wires) in the CRAY T3D system.

Several of the diagrams in this manual contain drawings of three-dimensional interconnect networks. For clarity, the communication links are shown logically and do not show the interleaving. It is important to remember that, although not shown, the nodes in the network are physically interleaved.

NOTE: The physical node numbers shown in Figure 2-7 do not reflect the actual numbering of physical nodes in a CRAY T3D system.

Dimension Order Routing

When a node sends information to another node, the information may travel through several communication links in the network. Each transfer over a communication link is hereafter referred to as a hop.

After information leaves a node, it travels through the network in the X dimension first, then through the Y dimension, and finally through the Z dimension. When finished moving through the communication links in the Z dimension, the information arrives at the destination node. This method of information travel is called dimension order routing.

For example, if node A shown in Figure 2-8 sends request information to node B, the information first travels one hop in the +X direction. Because the information does not need to travel any farther in the X dimension, it switches direction to the Y dimension.

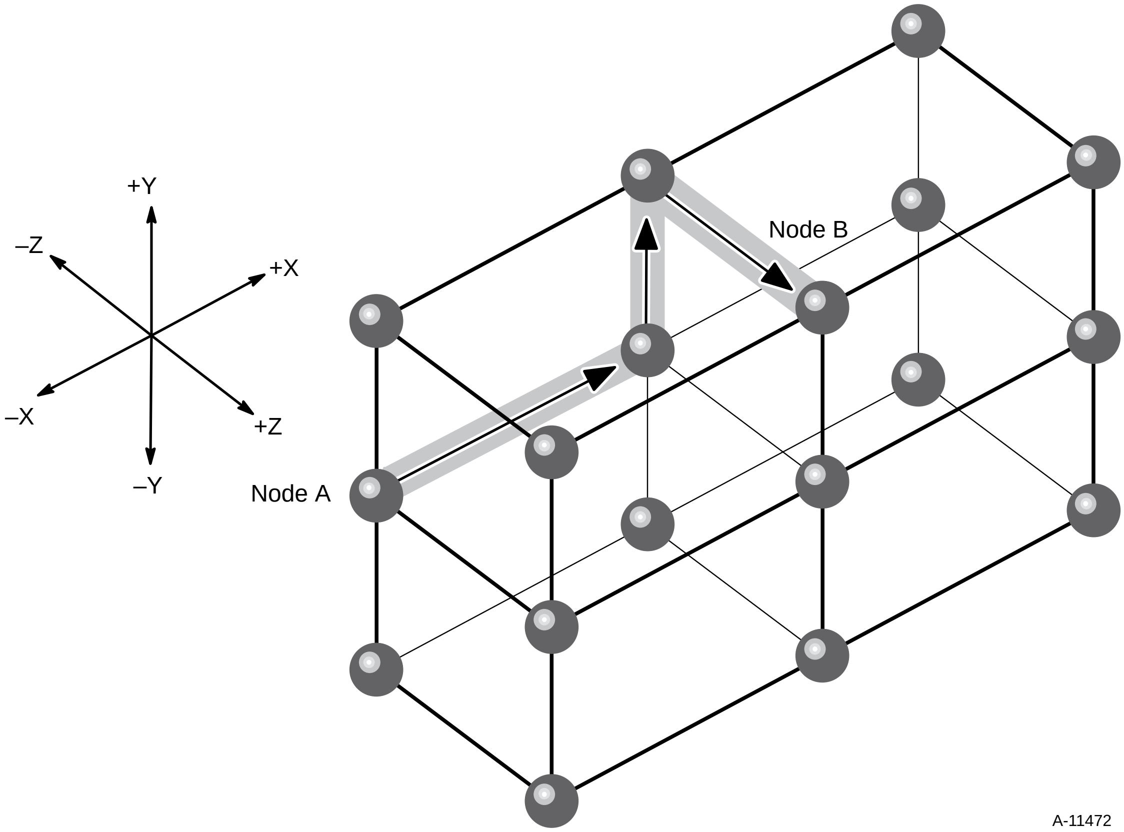


Figure 2-8. +X, +Y, and +Z Information Travel

After completing one hop in the +Y direction, the information switches direction to the Z dimension and completes one hop in the +Z direction. After completing one hop in the +Z direction, the request information arrives at node B.

Information does not always travel in the positive direction of a dimension. For example, if node B in Figure 2-9 sends response information to node A, the information completes one hop in the –X direction and then changes direction into the Y dimension.

The information completes one hop in the –Y direction before changing direction into the Z dimension. After completing one hop in the –Z direction, the response information arrives at node A.

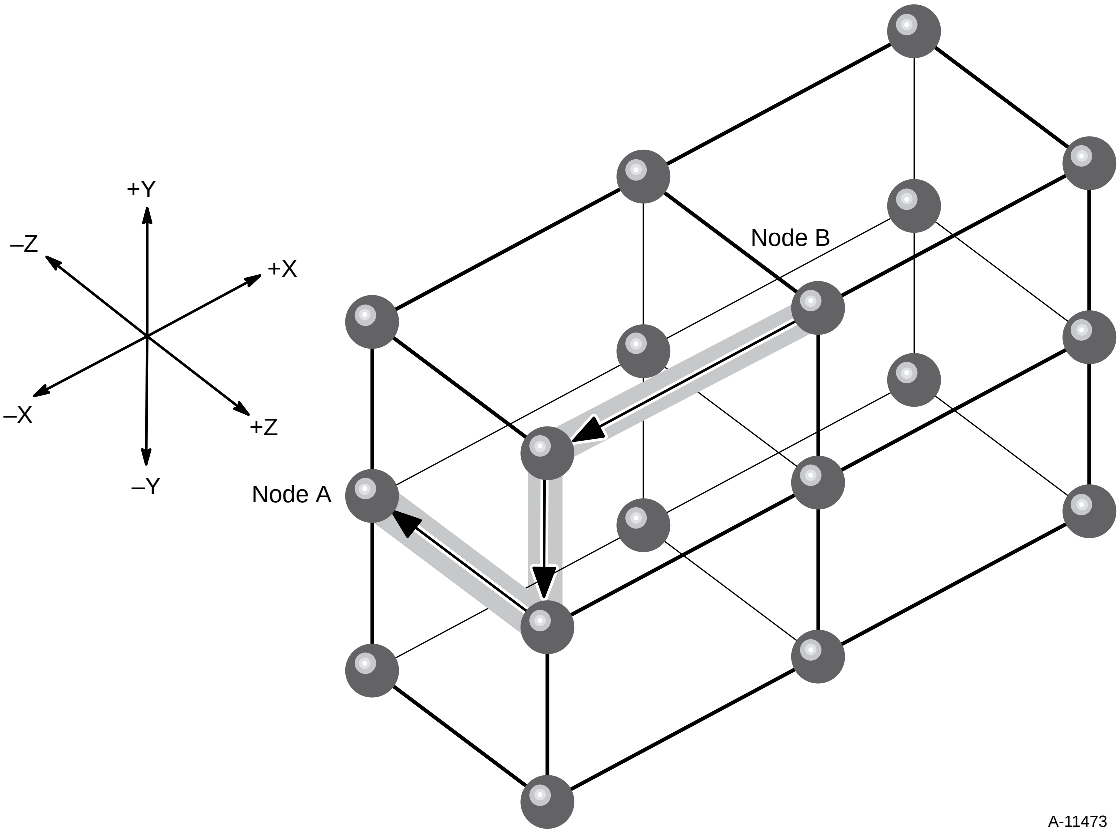


Figure 2-9. –X, –Y, and –Z Information Travel

Because information can travel in either the positive direction or the negative direction of a dimension, bad communication links can be avoided. For example, if node A in Figure 2-10 sends information to node B, the information completes one hop in the +X direction and then switches direction into the Y dimension.

Suppose, due to a bad communication link, the information cannot complete a hop in the +Y direction. Instead, the information may be routed so it completes two hops in the –Y direction and travels the long way around the torus in the Y dimension. After switching directions into the Z dimension, the information completes one hop in the +Z direction and arrives at node B.

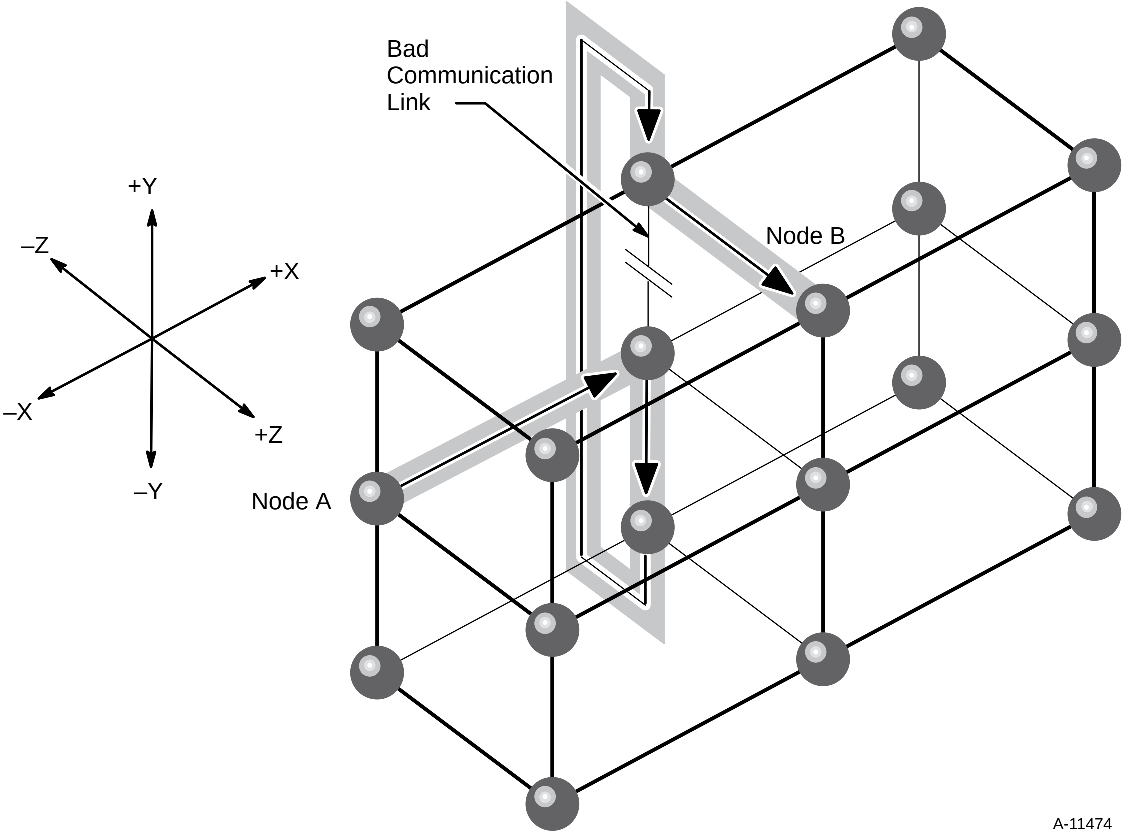


Figure 2-10. Avoiding a Bad Communication Link in the Y Dimension

Virtual Channels

A virtual channel (VC) is created when request information and response information travel over the same physical communication link but are stored in different buffers. The CRAY T3D system contains four VC buffers that buffer request information or response information as it travels through the interconnect network (refer to Table 2-3).

Table 2-3. Virtual Channel Buffers

Buffer Name	Definition
Virtual channel 0	Request buffer 0
Virtual channel 1	Request buffer 1
Virtual channel 2	Response buffer 0
Virtual channel 3	Response buffer 1

The VC buffers prevent two types of communication deadlock conditions from occurring in the interconnect network. The following paragraphs describe these conditions.

Without the VC buffers, a communication deadlock condition may occur if two nodes simultaneously transfer request or response information to each other. To prevent this condition from occurring, the CRAY T3D system contains two types of VC buffers: request buffers and response buffers. These buffers provide separate destination buffers for request and response information.

Also without the VC buffers, a communication deadlock condition may occur if all of the nodes in one dimension send request or response information to the next node in the dimension at the same time. For example, without the VC buffers, a deadlock condition may occur if all of the nodes in the X dimension send request information to the next node in the +X direction at the same time.

The buffers used when information travels through the network are determined by the dateline communication link. The dateline communication link is one communication link in each dimension that software designates as the dateline communication link.

When information traveling through a dimension uses the dateline communication link in that dimension, the information uses request buffer 1 or response buffer 1. If, when traveling through a dimension, the information never uses the dateline communication link in that dimension, the information uses request buffer 0 or response buffer 0.

For example, Figure 2-11 shows four nodes in the X dimension. Each node is transferring request information to the node two hops away in the +X direction. The dateline communication link is the communication link that connects nodes 1 and 2.

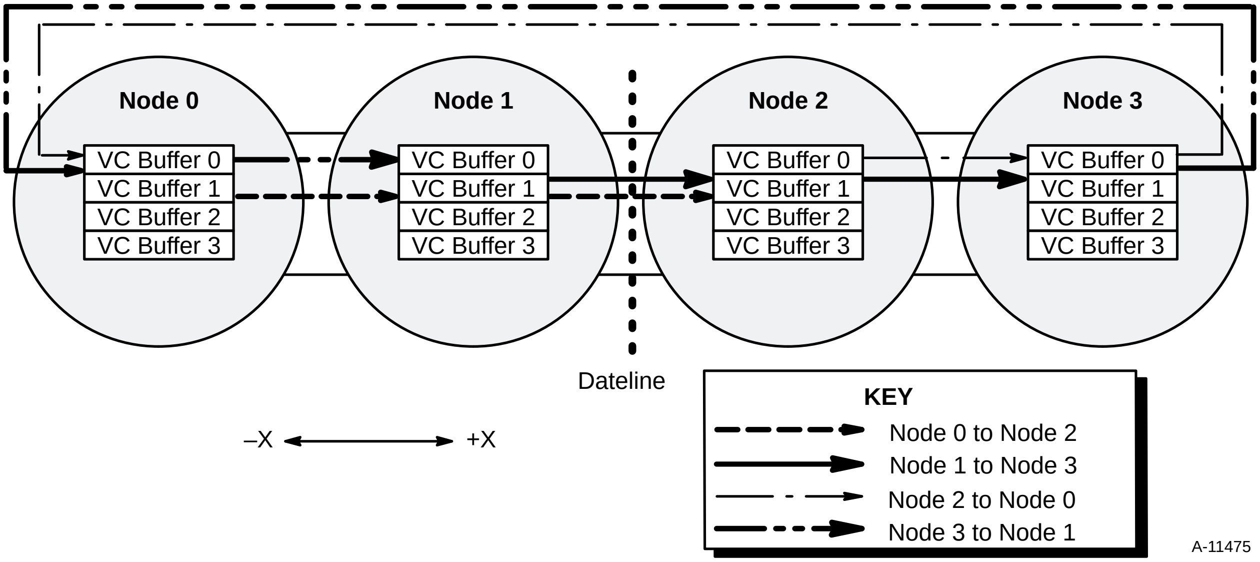


Figure 2-11. Dateline Communication Link

The request information that transfers from node 0 to node 2 and the request information that transfers from node 1 to node 3 will use the dateline communication link. Because of this characteristic, this request information uses VC buffer 1 (request buffer 1).

The request information that transfers from node 2 to node 0 and the request information that transfers from node 3 to node 1 will not use the dateline communication link. Because of this characteristic, this request information uses VC buffer 0 (request buffer 0).

After selecting a communication link to be the dateline communication link, software sets the X-dimension VC bit of each entry in the routing tag look-up tables to the appropriate value. The X-dimension VC bit is used as the least significant bit of the VC select bits in a communication link (refer again to Table 2-2). More information on the routing tag look-up tables is provided in Section 6, “Addressing.”

Packets

All information transferred over the data signals in a communication link (refer again to Figure 2-3) is in the form of a packet. A packet contains two parts: a header and a body (refer to Figure 2-12). The header and body have variable lengths and transfer over the communication link one 16-bit phit at a time.

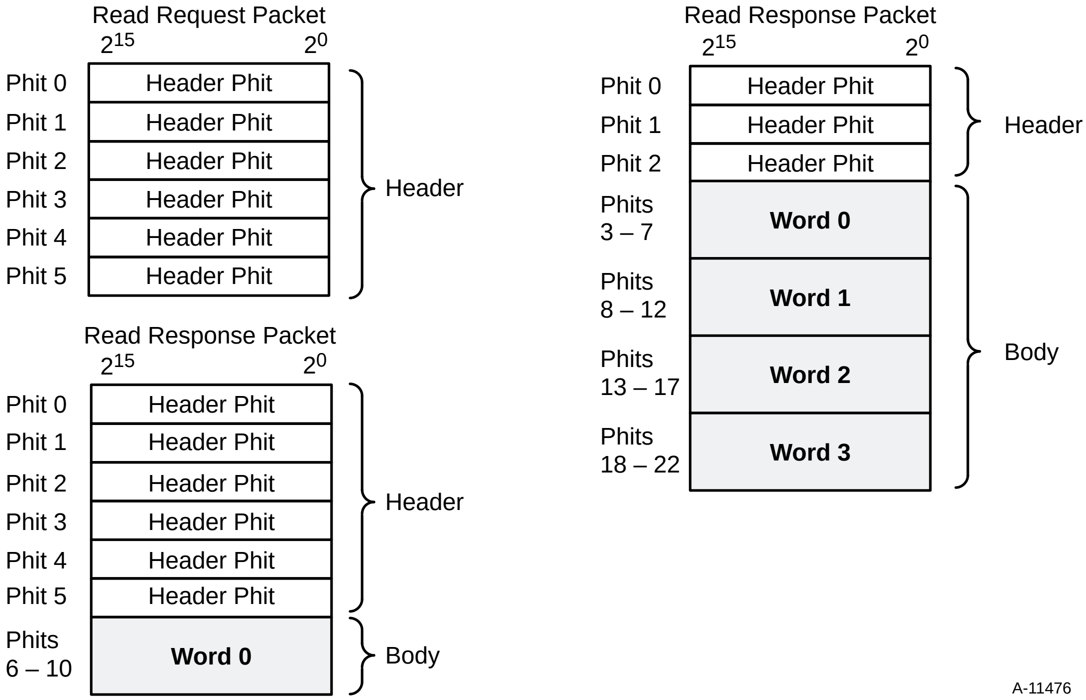


Figure 2-12. Generic Packet Formats

Every packet contains a header. The header always contains routing information, destination information, and control information. The routing information steers the packet through the network. The destination information indicates which PE will receive the packet. The control information instructs the PE that receives the packet to perform an operation. Optionally, the header may also contain memory address information and source information that indicates which PE created the packet.

A packet may or may not contain a body. The body of a packet contains one 64-bit word or four 64-bits words of system data. For example, the body of a read response packet contains 1 or 4 words of read data.

Network Routers

The network routers transfer packets through the communication links in the interconnect network. There are two types of network routers: processing element network routers and I/O gateway network routers.

The processing element network routers contain three components: an X-dimension switch, a Y-dimension switch, and a Z-dimension switch. Figure 2-13 shows the flow of packet information through a processing element network router.

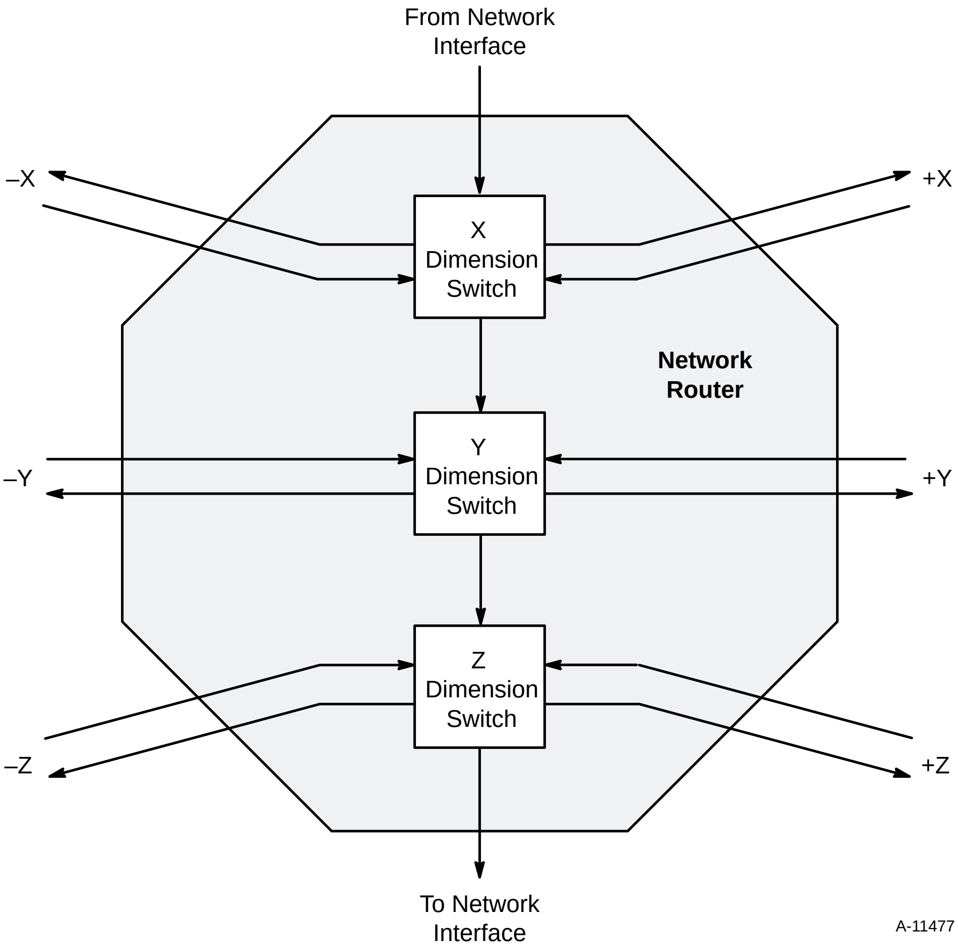


Figure 2-13. Processing Element Network Router

The X-dimension switch controls the flow of packets through the X-dimension communication links. Using both the routing information in the packet and information received from the channel control signals, the X-dimension switch steers packets from one X dimension. communication link to the other, or from one X-dimension communication link to the Y-dimension switch. Figure 2-14 shows the possible paths of packet information through the X-dimension switch.

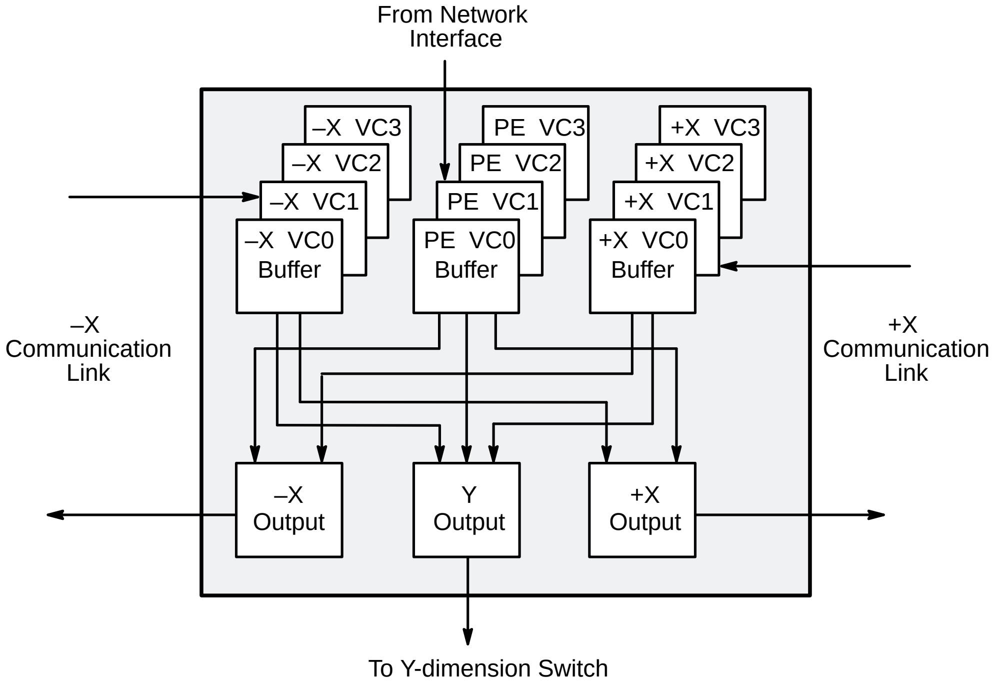


Figure 2-14. X-dimension Switch

As previously stated, each packet contains routing information. The packet routing information contains the two’s compliment of the number of hops the packet will make in each dimension and indicates the direction the packet will travel in each dimension.

Immediately after receiving the first phit of a packet header, the X-dimension switch reads the value stored in the X-dimension portion of the packet routing information. If the value is not 0, the X-dimension switch increments the value by 1 and sends the packet out on an X-dimension communication link. If the value is 0, the X-dimension switch sends the packet to the Y-dimension switch.

The X-dimension switch contains VC buffers to separate requests and responses. The channel control signals in the communication links control which VC buffer stores the packet information. Each buffer can store up to eight 16-bit parcels.

The Y- and Z-dimension switches function identically to the X-dimension switch. The Y- and Z-dimension switches transfer packets over the Y- and Z-dimension communication links, respectively.

The Y-dimension switch can receive packets from a negative Y-dimension switch, a positive Y-dimension switch, or an X-dimension switch. The Y-dimension switch can send packets to a negative Y-dimension switch, a positive Y-dimension switch, or a Z-dimension switch.

The Z-dimension switch can receive packets from a negative Z-dimension switch, a positive Z-dimension switch, or a Y-dimension switch. The Z-dimension switch can send packets to a negative Z-dimension switch, a positive Z-dimension switch, or the network interface.

The I/O gateway network routers operate similarly to the processing element node network routers; however, the I/O gateway network routers do not contain a Y-dimension switch. Figure 2-15 shows the components of the input node network router.

The two network routers for an I/O gateway are connected to each other. The +X and +Z communication links from the input node network router connect to the output node network router. The –X and –Z communication links from the output node network router connect to the input node network router.

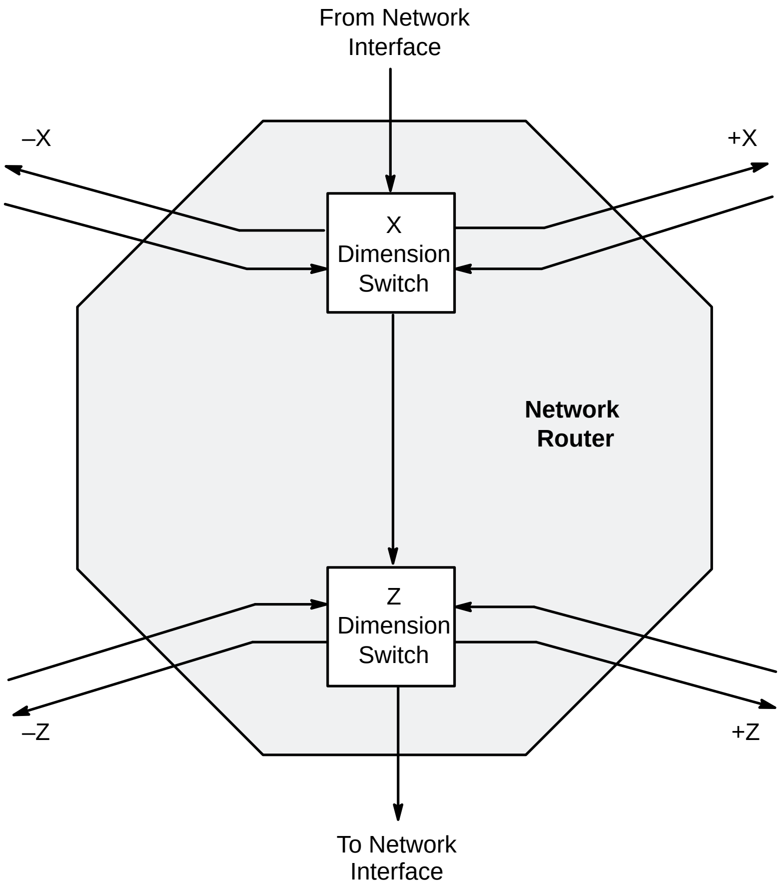


Figure 2-15. Input Node Network Router

Hardware Description

The hardware that makes up the interconnect network consists of options referred to as network router options. The following subsections describe the network router options.

SR Option - Network Router

The network router of a processing element node is made up of three network router options. The network router of an I/O gateway node is made up of only two network router options. These options are the SR options. Refer to Figure 2-16 and Figure 2-17.

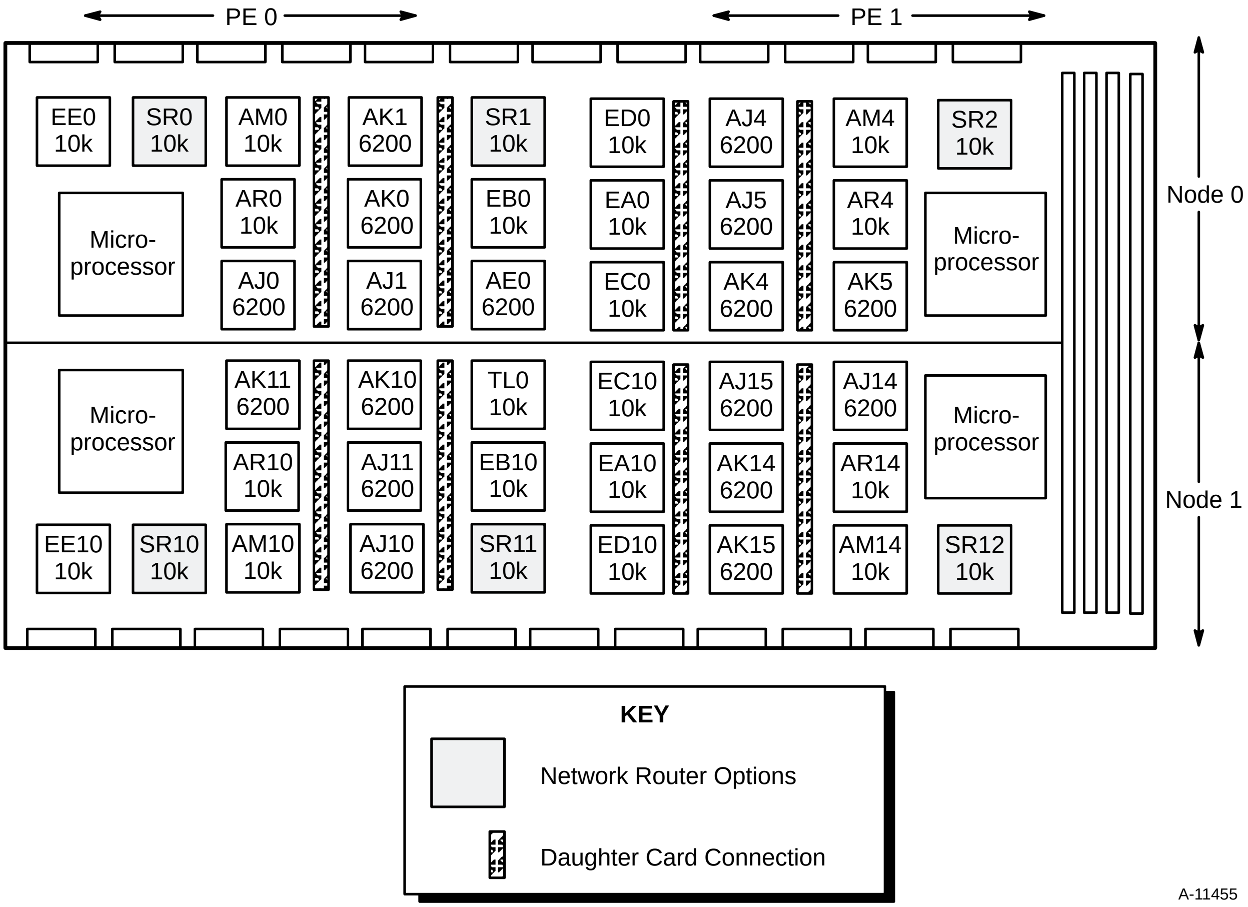


Figure 2-16. Processing Element Node Network Router Options

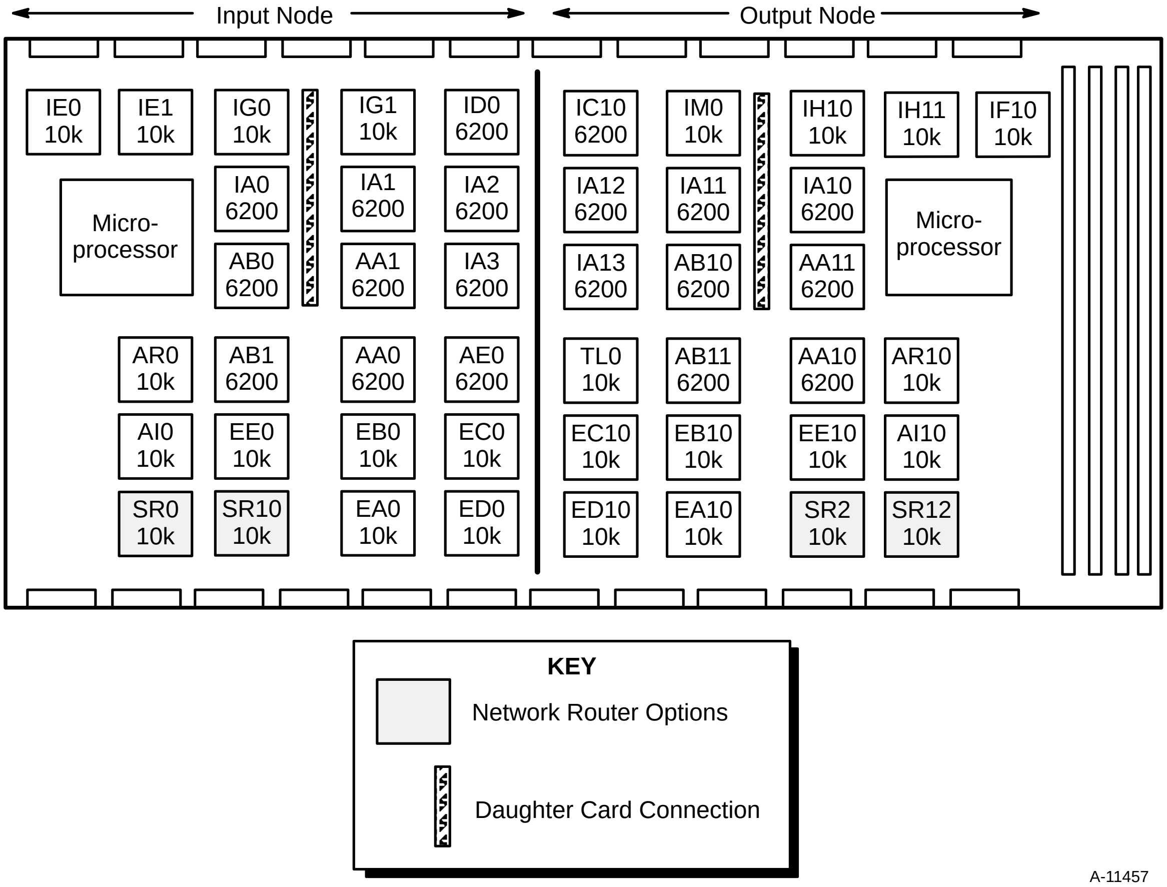


Figure 2-17. I/O Gateway Node Network Router Options

The SR option transfers request and response packets through the network. Of the three SR options in the processing element node 0 network router, the SR0 option handles the X dimension, the SR1 option handles the Y dimension, and the SR2 option handles the Z dimension. In the processing element node 1 network router, the SR10 option handles the X dimension, the SR11 option handles the Y dimension, and the SR12 option handles the Z dimension. In the input node network router of the I/O gateway, the SR0 option handles the X dimension and the SR2 option handles the Z dimension. And in the output node, the SR10 option handles the X dimension and the SR12 option handles the Z dimension.

The SR option handling the X dimension receives the request or response packet from the EA option (network interface option) in the same node or from an SR option handling the X dimension in a different node. This packet may come from the positive direction or the negative direction of the dimension.

The SR option contains three groups of buffers, one for each type of input (PE, positive, and negative). Within each group, there are four buffers that coincide with a VC. Refer to Figure 2-18. The VC select from the EA option or SR option dictates which buffer will store the packet.

After buffering the packet, the SR option determines whether the packet should be sent in the positive direction or the negative direction within the dimension or whether the packet should be sent to the next dimension. To determine this, the Δ X (2’s compliment of the number of hops) and the positive or negative select (the direction the packet will travel) of the routing tag are used. If the Δ X is 0, then the SR option switches the packet to the next dimension. If the Δ X is not 0, the SR option increments the 2’s compliment of the Δ X value and uses the positive or negative select to steer the packet in the correct direction. The SR options handling the Y and Z dimensions perform similar functions using the values of Δ Y and Δ Z. The packet has reached it’s destination when the values of Δ X, Δ Y, and Δ Z all equal 0.

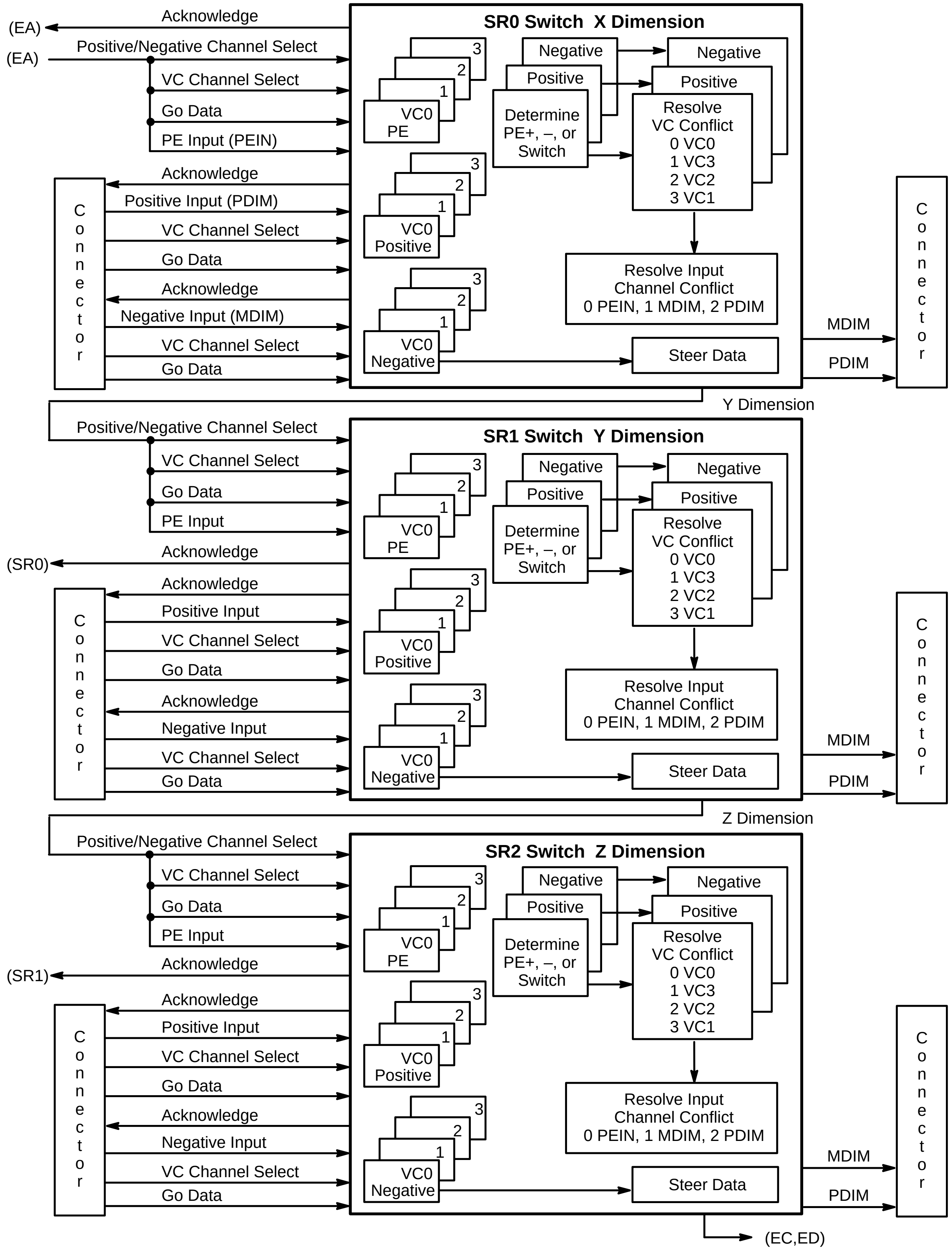


Figure 2-18. Network Router Options

When an SR option (source) sends 8 phits of a packet (hereafter referred to as a flit) to another SR option (destination), the source SR option will not send another flit until it receives an Acknowledge signal from the destination SR option. For example:

Option 1 sends a flit to Option 2. Refer to Figure 2-19.

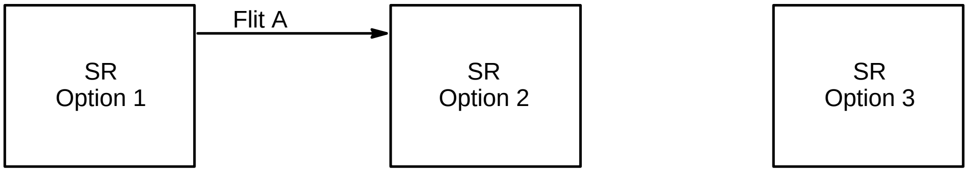


Figure 2-19. Initial Flit Transfer Between SR Options

Option 2 then sends the flit to Option 3. While this transfer is in progress, Option 2 sends an Acknowledge signal to Option 1, as illustrated in Figure 2-20. (Note that Option 3 completes similar transfers when it receives the flit.)

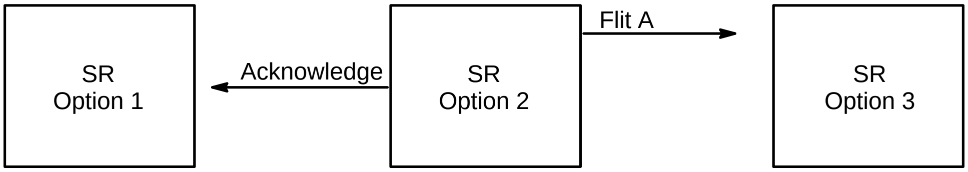


Figure 2-20. Acknowledge Signal

When Option 1 receives the Acknowledge signal, it sends another flit to Option 2, as illustrated in Figure 2-21.

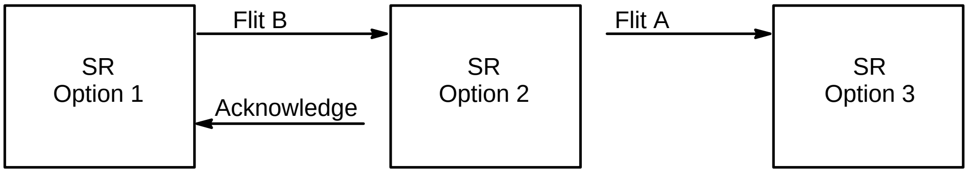


Figure 2-21. Subsequent Flit Transfer

This type of handshaking enables a continuous flow of flits between the SR options. In fact, an SR option can receive the first phit of a new flit at the same time that the SR option transfers the last phit of the previous flit to another option.

An SR option generates an Acknowledge signal when the SR option starts to send the flit to the next SR option. The SR option delays the Acknowledge signal for a fixed number of clock periods depending upon static select signals, which are forced in the wiremat. Table 2-4 lists the delay settings for these static select signals.

Table 2-4. Acknowledge Delay

Delay Setting	Clock Period †
11	1
10	1.5
01	2
00	3

† The clock period refers to the number of clock periods required to transfer a phit between the SR options.

The SR option delays the Acknowledge signal longer for a delay setting of 11 than it does for a delay setting of 10. This compensates for the extra time it takes the Acknowledge signal to travel between SR options. For example, in order for the two source SR options in Figure 2-22 to receive the Acknowledge signal at the same time, the destination SR option of the 1.5 CP transfer must send out the Acknowledge signal to the source SR option sooner than the destination SR option of the 1 CP transfer.

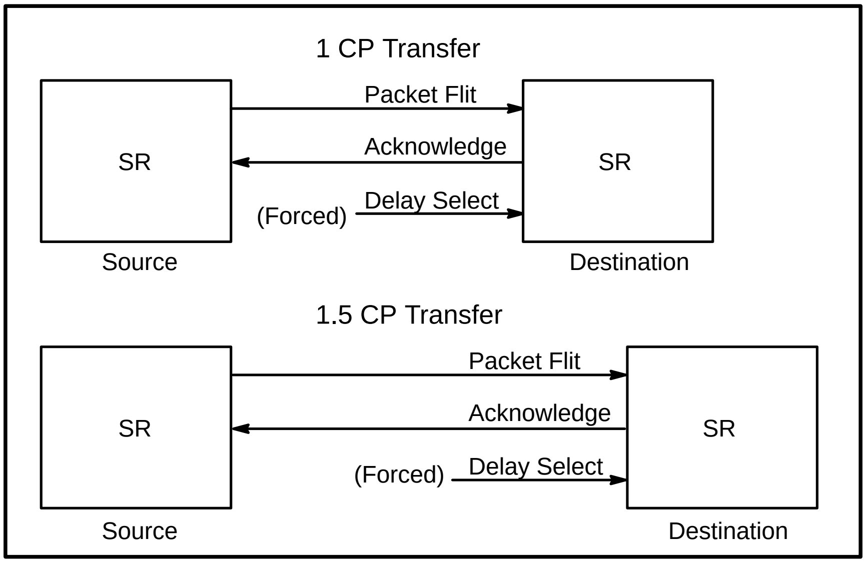


Figure 2-22. 1 CP Transfer and 1.5 CP Transfer Between SR Options

NOTE: When the wire length between the SR options is 1.5 CPs, the static select signals must be set to 10. A delay setting of 11 would degrade performance because the Acknowledge signal arrives at the source SR option 1 CP wire too late, causing an idle CP between flits.

When the wire length is 1 CP, the static select signals must be set to 11. A delay setting of 10 would cause the Acknowledge signal to arrive at the source SR option too soon. Because the source SR option receives the Acknowledge signal too soon, the source SR option will send the next flit to the destination SR option too soon, causing data corruption.

The SR options resolve conflicts that occur while transferring packets through the network. The following conflicts may occur:

- Simultaneous input conflict
- Simultaneous reference request conflict
- Virtual channel busy conflict
- Virtual channel reserved conflict

A simultaneous input conflict occurs when packets arrive simultaneously at the SR option using the processor element node input (PEIN), the positive input (PDIM), or the negative input (MDIM) and request the same output channel. For example, when two packets arrive simultaneously, one using the processing element node input and the other using the positive input, and both packets need to transfer in the negative direction of the dimension, a conflict will occur (refer to Figure 2-23).

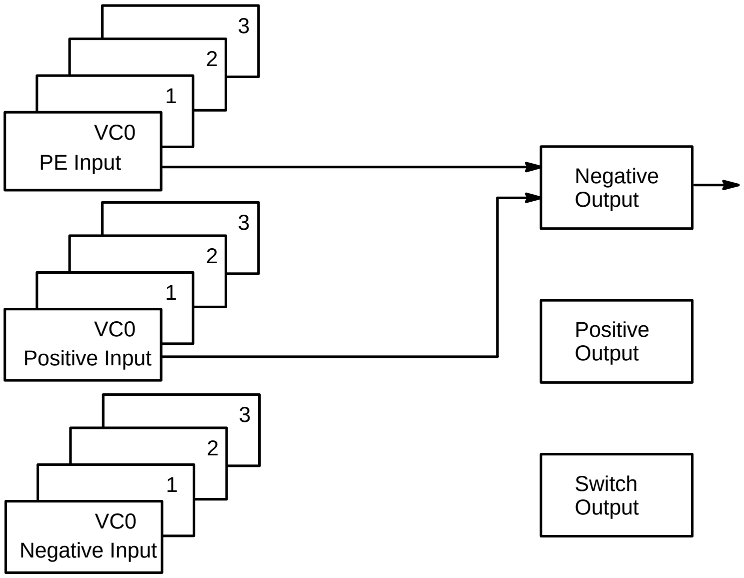


Figure 2-23. Simultaneous Input Conflict

The SR option resolves this type of conflict using the upper 2 bits of a conflict resolution counter. Table 2-5 shows the priority order of the three inputs at the different increments of the counter.

Table 2-5. Simultaneous Input Conflict Resolution

Count = 00xxx	Count = 01xxx	Count = 10xxx
PE input	Negative input	Positive input
Positive input	PE input	Negative input
Negative input	Positive input	PE input

A simultaneous reference request conflict occurs when an input (PEIN, PDIM, or MDIM) has two or more VCs requesting the same output channel. This type of conflict may occur after the resolution of a virtual channel conflict. For example, a packet arrives using the PEIN input and requests VC 1 for transfer into the negative direction. This packet comes in conflict with a VC that is busy. At the same time that the VC 1 busy is released, the PEIN input is used again by a packet that requests VC 3 for a transfer in the negative direction. The PEIN input now has two VCs that need the same output channel (refer to Figure 2-24).

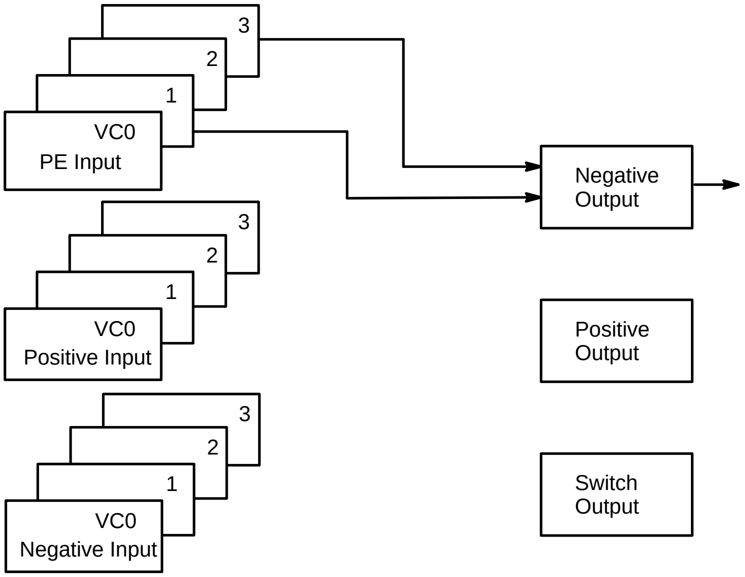


Figure 2-24. Simultaneous Reference Request Conflict

The SR option resolves this type of conflict using the lower 2 bits of the conflict resolution counter. Table 2-6 shows the priority order of the four VCs at the different increments of the counter.

Table 2-6. Simultaneous Reference Request Conflict Resolution

Count = xxx00	Count = xxx01	Count = xxx10	Count = xxx11
VC0	VC3	VC2	VC1
VC1	VC0	VC3	VC2
VC2	VC1	VC0	VC3
VC3	VC2	VC1	VC0

NOTE: The priority counter is disabled periodically by a random number generator in the EE option. This prevents an input or VC from dominating the access of the output channel.

A virtual channel busy conflict occurs when a packet tries to access a VC that is busy. The SR option sets the VC busy when it sends a packet to the next option in the interconnect network. The SR option holds the VC busy until it receives an Acknowledge signal from this option for the transfer.

A virtual channel reserved conflict occurs when a packet tries to access a VC that is being used to transfer a different packet. A VC is reserved by the input (PEIN, PDIM, or MDIM) for the packet that requested the VC first. It remains reserved until the entire packet is transferred through the node.

Once all conflicts are resolved, the SR option steers the packet toward the correct destination.

NOTE: The number of conflicts that occur for an interconnect network channel can be monitored by the microprocessor. For more information about how the conflicts are monitored, refer to the subsections “Network Performance Register” in Section 10, “Control and Status.”

3 PROCESSING ELEMENT NODE

Processing element nodes perform all program instructions and store system data. A processing element node is composed of four components: two processing elements (PEs), a network interface, and a block transfer engine (BLT). Refer to Figure 3-1. This section describes the functions and hardware of the components that make up a processing element node.

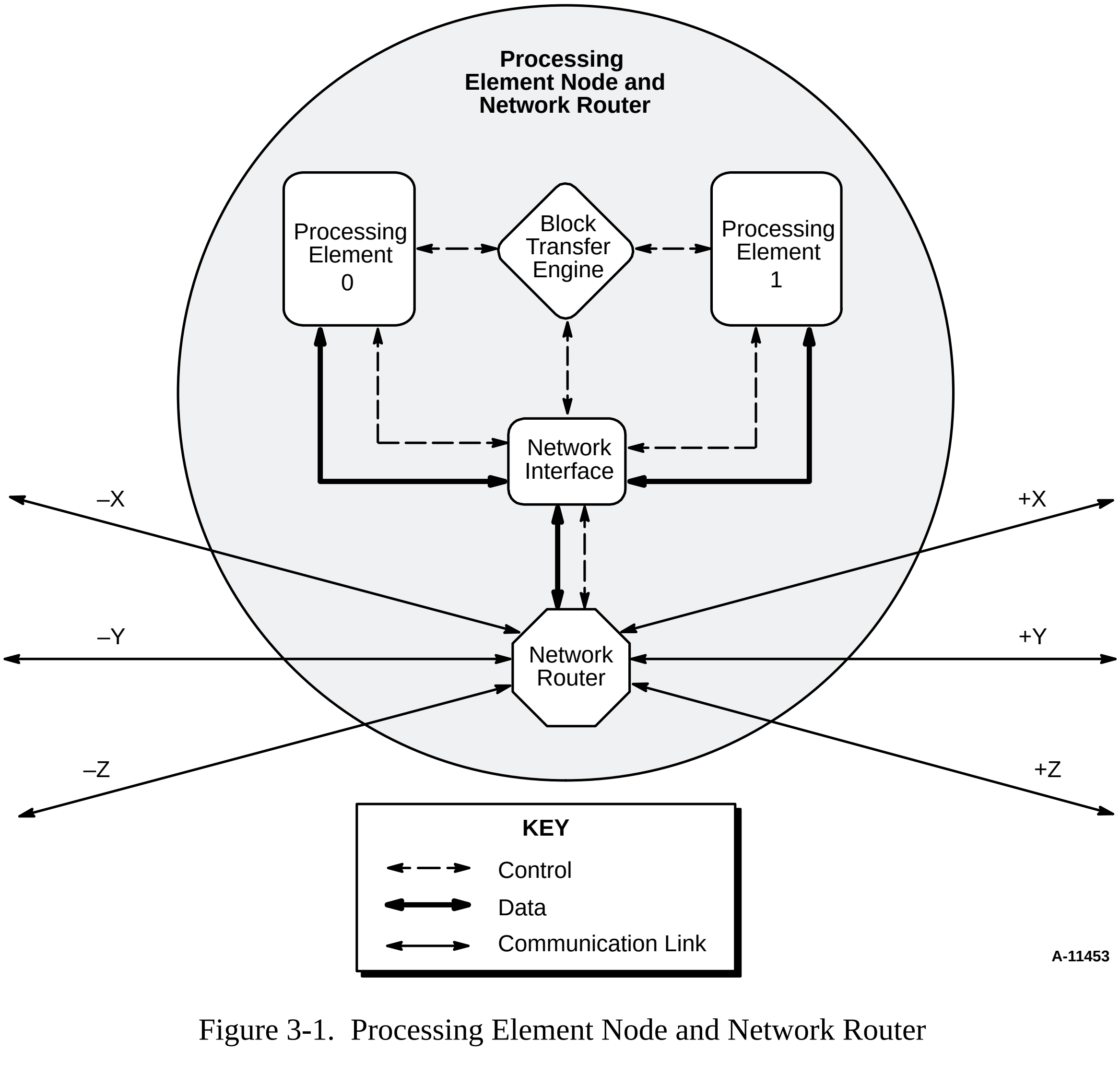


Figure 3-1. Processing Element Node and Network Router

Functional Description

Each processing element node contains two processing elements: PE 0 and PE 1. Each PE contains a microprocessor, local memory, and PE support circuitry. Refer to Figure 3-2.

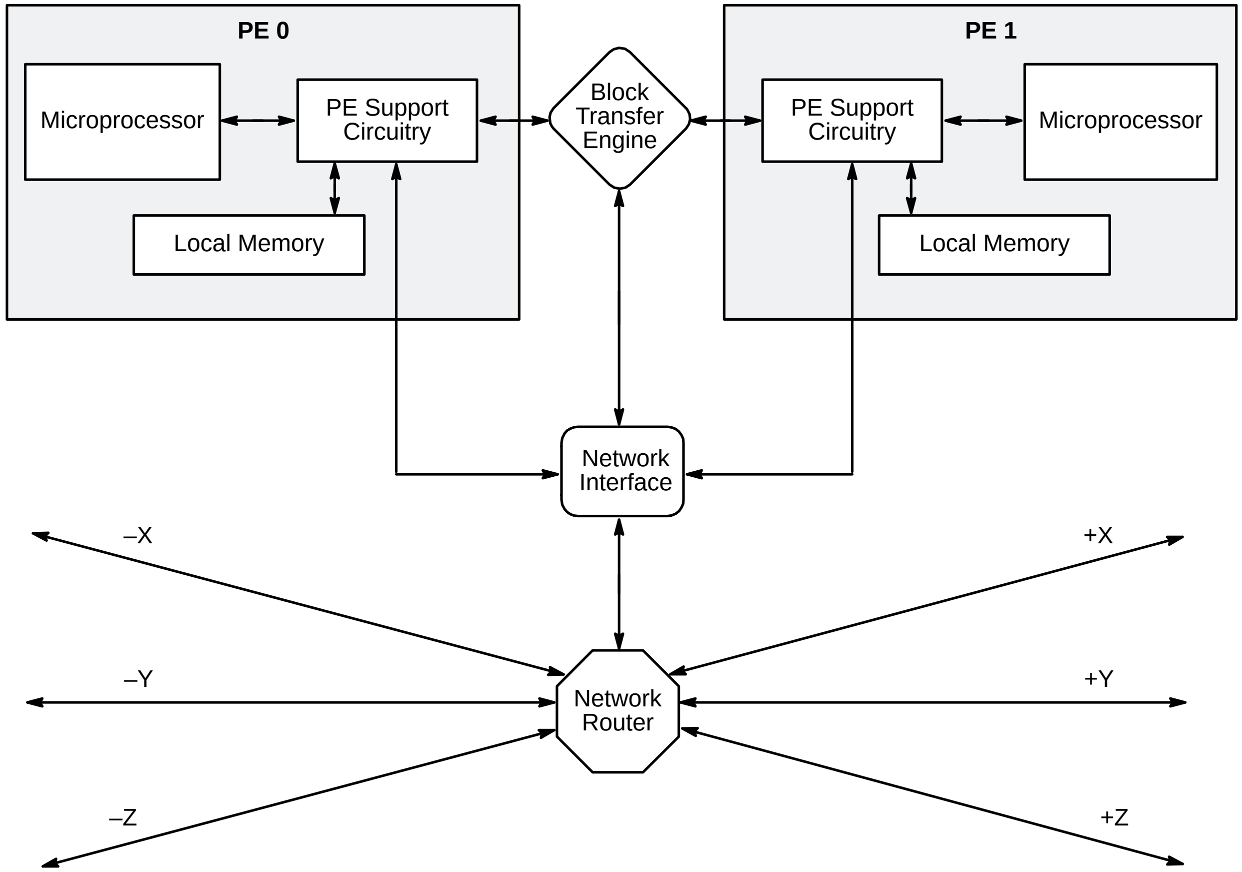


Figure 3-2. Processing Element Components

Microprocessor

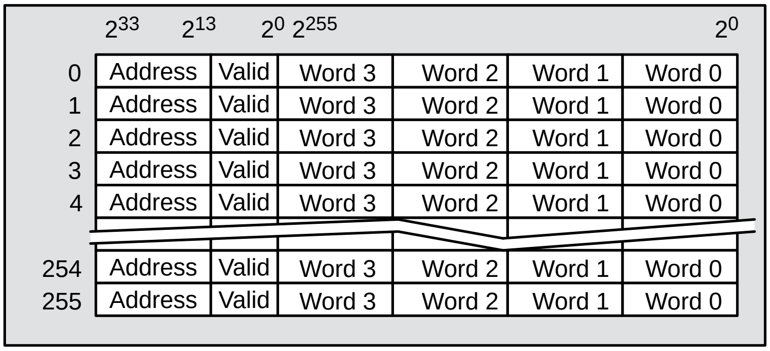
The microprocessor is a reduced instruction set computer (RISC) 64-bit microprocessor developed by Digital Equipment Corporation. Major components of the microprocessor include a data cache memory and a write buffer. The microprocessor also contains circuitry that enables the microprocessor to invalidate data cache memory lines (on command from PE support circuitry) and perform single error correction/double error detection (SECDED) on data words read from memory.

Data Cache

Data cache memory is a small, high-speed random access memory that temporarily stores frequently or recently accessed data. The data cache memory is internal to the microprocessor and stores 256 32-byte lines (four 64-bit words in a line) of data.

Format

Each line in the data cache contains three components: an address tag, a valid bit, and data. Figure 3-3 shows the format of the data cache memory.



0	Address	Valid	Word 3	Word 2	Word 1	Word 0
1	Address	Valid	Word 3	Word 2	Word 1	Word 0
2	Address	Valid	Word 3	Word 2	Word 1	Word 0
3	Address	Valid	Word 3	Word 2	Word 1	Word 0
4	Address	Valid	Word 3	Word 2	Word 1	Word 0
254	Address	Valid	Word 3	Word 2	Word 1	Word 0
255	Address	Valid	Word 3	Word 2	Word 1	Word 0

Figure 3-3. Data Cache Memory

The address tag contains bits 13 through 33 of the partial physical address for the data in the cache line. The address tag indicates the external memory location from which the data in the cache line originated.

The valid bit indicates whether the data in the cache line is valid. When set to 1, this bit indicates that the data is valid. When set to 0, this bit indicates that the data is not valid. Generally, the data in a cache line is considered valid when it matches the data stored in a corresponding external memory location.

Each line contains four 64-bit words of data. All 4 words in the cache line are either valid or invalid as indicated by the valid bit.

Operation

The data cache may be used or updated when the microprocessor issues a a load or store instruction. The following paragraphs describe how the data cache is used for both types of instructions.

When the microprocessor issues a load instruction, it converts the byte-oriented virtual address used in the instruction into a byte-oriented partial physical address. The partial physical address points to the first byte of data that is requested by the load instruction.

The microprocessor uses bits 5 through 12 of the partial physical address for the load instruction as an index that points to one of the 256 lines in the data cache. For example, if bits 5 through 12 of the partial physical address are set to 12, the microprocessor references cache line 12.

After referencing a cache line, the microprocessor compares bits 13 through 33 of the partial physical address to the address tag of the selected cache line. If they match, the cache line may contain the data requested by the load instruction. The microprocessor then examines the valid bit of the cache line. If the valid bit is set to 1, the microprocessor retrieves the valid data from the cache line and transfers the data to the register specified in the load instruction. If the valid bit is set to 0, the data in the cache line is invalid. In this case, the microprocessor must use the support circuitry to retrieve the requested data from external memory. When this occurs, it is called a data cache miss.

If bits 13 through 33 of the partial physical address and the address tag of the selected cache line do not match, the cache line does not contain the data requested by the load instruction. When this occurs, the microprocessor also encounters a data cache miss.

When a data cache miss occurs, the microprocessor presents the partial physical address to the support circuitry along with a READ_BLOCK cycle request code. After performing one of the read operations (designated by the function code in the DTB annex), the support circuitry sends data to the microprocessor and signals the microprocessor that the data should or should not be placed in the data cache (according to the function code type).

When the support circuitry signals the microprocessor that data should be placed in the data cache, the microprocessor stores the data in the data cache line referenced by bits 5 through 12 of the partial physical address for the data. The microprocessor then stores bits 13 through 33 of the partial physical address in the address tag of the cache line and sets the valid bit to 1 to indicate that the cache line contains valid data.

When the microprocessor issues a store instruction, it converts the byte-oriented virtual address used in the instruction into a byte-oriented partial physical address. The partial physical address points to the first byte of data that will be written in external memory.

After issuing the store instruction, the microprocessor compares the partial physical address with the address tag in a selected cache line. If they match and the valid bit is set to 1, the microprocessor updates the specified amount of data in the cache line before completing the store instruction. If the partial physical address and an address tag do not match, or if the valid bit is not set to 1, the microprocessor does not update a cache line.

Cache Line Invalidation

In the CRAY T3D system, data in the local memory of a PE may be modified by the microprocessor in the PE or by an external source (BLT or remote PE). In order to ensure that a PE’s data cache copy of local memory data remains consistent with the actual data in local memory, the support circuitry must have a method of informing the microprocessor when a line in the data cache may be invalid.

To signal the microprocessor when data may be invalid, the support circuitry uses the cache line invalidate pins of the microprocessor (refer to Figure 3-4). The cache line invalidate pins may be used to perform two types of cache line invalidations: nonfiltered and filtered.

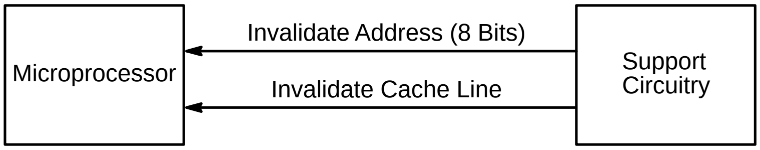


Figure 3-4. Cache Line Invalidate Pins

During a nonfiltered cache line invalidation, the support circuitry signals the microprocessor to invalidate a corresponding cache line when data is written into any location in local memory. There is no hardware support for maintaining data cache coherence with remote data (data in a remote PE’s memory).

To initiate a nonfiltered invalidation, the microprocessor sets the Enable Invalidates bit (bit 9) of the invalidate control register (INV_CR) to 1 and sets the enable invalidate filter bit (bit 8) of the INV_CR to 0. More information on the INV_CR is provided in Section 10, “Control and Status.”

When an external source writes data into local memory, the support circuitry places bits 5 through 12 of the address offset for the data on the invalidate address pins of the microprocessor. The support circuitry then sets the invalidate cache line pin to the microprocessor. After receiving the Invalidate Cache Line signal, the microprocessor sets the valid bit to 0 in the data cache line addressed by the Invalidate Address pins.

During a filtered cache line invalidation, only a single line in the data cache is enabled for invalidation. All other lines remain undisturbed during incoming external write operations.

To initiate a filtered invalidation, the microprocessor sets the Enable Invalidate bit (bit 9) of the INV_CR to 1 and sets the Enable Invalidate Filter bit (bit 8) of the INV_CR to 1. In addition to these bits, the microprocessor sets the Invalidate Filter bits (bits 0 through 7) of the INV_CR to the number of the only cache line that may subsequently be invalidated.

When an external source writes data into local memory, the support circuitry compares bits 5 through 12 of the address offset for the data to the Invalidate Filter bits of the INV_CR. If they match, the support circuitry places bits 5 through 12 of the address offset on the Invalidate Address pins of the microprocessor. The support circuitry then sets the Invalidate Cache Line pin to the microprocessor and writes data into local memory. After receiving the Invalidate Cache Line signal, the microprocessor sets the valid bit to 0 in the data cache line addressed by the Invalidate Address pins.

If bits 5 through 12 of the address offset and the Invalidate Filter bits of the INV_CR do not match, the support circuitry writes data into local memory as usual but does not signal the microprocessor to invalidate a cache line.

The filtered invalidation may be used to avoid a situation whereby the microprocessor ties up local memory with continuous read references when looping on a value in one memory location. Before issuing a cached read or cached atomic swap from local memory, the microprocessor sets the Invalidate Filter bits of the INV_CR so that they contain bits 5 through 12 of the address offset for the requested data.

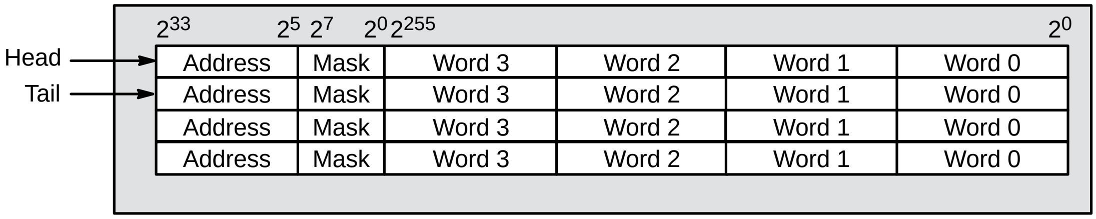
After the microprocessor receives the data and stores the data in the data cache, the microprocessor reads the data from the data cache during the loop rather than from local memory. When an external source changes data in the local memory location, the support circuitry signals the microprocessor to invalidate the cache line. The next time the microprocessor tries to read data from that line in the data cache, it encounters a data cache miss and must read the data from local memory to retrieve an updated copy of the data.

NOTE: Because bits 0 through 7 of the INV_CR correspond to only bits 5 through 12 of an address offset into local memory, a cache line may be invalidated by external writes to locations other than that associated with the specified address offset. For example, if bits 0 through 7 of the user control and status register were set to $5A_{16}$, the data cache line corresponding to 5A (line number 90) will be invalidated if a word is written into address offset $B40_{16}$ or address offset $2B40_{16}$.

Write Buffer

Each T3D microprocessor contains a write buffer. The write buffer temporarily stores write data before it transfers to the support circuitry. The write buffer receives write data at a high bandwidth and maximizes the rate at which data transfers between the microprocessor and support circuitry. The write buffer does this by attempting to merge 32- or 64-bit stores to the same line in a single cache-line sized write operation to external memory. The write buffer is internal to the microprocessor and stores up to four 32-byte lines (four 64-bit words in a line) of data.

Figure 3-5 shows the format of the write buffer. Each line in the write buffer contains three parts: an address, a mask, and data.



Address	Mask	Word 3	Word 2	Word 1	Word 0
Address	Mask	Word 3	Word 2	Word 1	Word 0
Address	Mask	Word 3	Word 2	Word 1	Word 0
Address	Mask	Word 3	Word 2	Word 1	Word 0

Figure 3-5. Write Buffer Format

The address is a 29-bit value that contains bits 5 through 33 of the partial physical address used for an instruction during a write function. The address is used to identify a line in the write buffer.

The mask is an 8-bit value that indicates which 32-bit halfwords are valid in a line of the write buffer. For example, if bits 0 and 1 of the mask are set to 1, the mask indicates that word 0 is valid in the write buffer line.

A line in the write buffer contains a maximum of 4 words of data. Not all of the data in a line may be valid. The validity of the data is indicated by the mask.

Write buffer circuitry in the microprocessor uses two pointers to control the write buffer: the head pointer and the tail pointer. The head pointer indicates the line in the write buffer that has contained valid data for the longest period of time. The tail pointer indicates the line in the write buffer that will be allocated next.

When the microprocessor issues a write-related instruction, the write buffer circuitry compares bits 5 through 33 of the partial physical address for that instruction with the address bits in each line of the write buffer. If it finds a match, the circuitry merges the new data into the corresponding line of the write buffer. The circuitry then sets the appropriate bits of the mask to indicate which new and old 32-bit halfwords in the write buffer line are valid.

If bits 5 through 33 of the partial physical address for an instruction and the address bits do not match, the write buffer circuitry writes the data into the write buffer line indicated by the tail pointer. The circuitry also sets the appropriate bits of the mask to 1 and writes bits 5 through 33 of the partial physical address into the address bits of that line.

The write buffer circuitry only merges store instruction data (for example, the store quadword instruction) in an attempt to fill a write buffer line before sending the data to the support circuitry. (The write buffer stores but does not merge memory barrier [MB], store conditional [STX_C], or fetch instructions.)

The write buffer circuitry attempts to flush the write buffer when any of the following conditions occur.

- At least 2 of the 4 lines in the write buffer contain valid data.
- One line in the write buffer contains valid data for 256 clock pulses, and the microprocessor has not issued a write-buffer related instruction within that time.
- One of the lines in the write buffer contains information for an MB or STX_C instruction. In this case, the write buffer circuitry sends all valid data to the support circuitry until the write buffer is empty.
- The microprocessor issues a load instruction that requires the use of the support circuitry (the requested data is not in the data cache) and the address for the load instruction matches the address of a write related instruction stored in the write buffer. In this case, the write buffer circuitry sends all the valid data to the support circuitry until the write buffer is empty. This ensures that a write related instruction accesses a memory location before the subsequent read instruction does.

When data and address information for the store instruction (possibly merged store instructions) leaves the write buffer, the microprocessor provides a cycle request code to the support circuitry. The microprocessor also provides the support circuitry with a valid partial physical address.

While the write buffer circuitry is sending a cycle request from the head line in the write buffer to the support circuitry, no new store instructions can be merged into the write buffer.

SECDED

The microprocessor may perform SECDED on data that it receives from the support circuitry during an operation. When software issues an operation where the microprocessor will eventually receive data, the software indicates whether the microprocessor should perform SECDED by setting bit 27 of the partial physical address to the appropriate value. (Also the type of operation indicates whether or not SECDED will be performed. For example, SECDED is not performed on fetch-and-increment register reads or prefetch fetch-and-increment register reads.)

When software sets bit 27 to 0, it signals the support circuitry that the microprocessor will perform SECDED on the data when it is returned. After the support circuitry retrieves data for the operation, the support circuitry sends the data to the microprocessor and signals the microprocessor that SECDED should be performed on the data.

When software sets bit 27 to 1, it signals the support circuitry that the microprocessor will not perform SECDED on the data. After the support circuitry retrieves data for the operation, the support circuitry sends the data to the microprocessor and signals the microprocessor that SECDED should not be performed on the data.

The data path between the microprocessor and the support circuitry is 128 bits wide. Along with each set of 128 data bits, the support circuitry sends 28 check bits to the microprocessor (7 check bits per 32-bit halfword). The microprocessor uses the check bits to perform SECDED; however, SECDED is not performed solely by hardware in the microprocessor.

Hardware in the microprocessor generates a new set of check bits for the data. (The original check bits were also generated by a microprocessor during a write operation or by the input node of an I/O gateway during a HISP input transfer.) Table 3-1 and Table 3-2 show the operation used to generate each of the check bits. For example, to generate check bit 6 for data bits 0 through 31, the hardware performs an XOR logical operation on data bits 24 through 31 and bits 0 through 7.

Table 3-1. Check Bits for Data Bits 16 through 31

Check Bit	Operation	Data Bits															
		31 – 28				27 – 24				23 – 20				19 – 16			
6	XOR	x	x	x	x	x	x	x	x								
5	XOR	x	x	x	x	x	x	x	x								
4	XOR	x	x							x	x	x	x	x	x		
3	XNOR			x	x	x				x	x	x				x	x
2	XNOR	x		x			x	x		x			x	x			x
1	XOR				x		x		x		x		x		x	x	x
0	XOR	x		x	x		x					x		x	x	x	

Table 3-2. Check Bits for Data Bits 0 through 15

Check Bit	Operation	Data Bits															
		15 – 12				11 – 8				7 – 4				3 – 0			
6	XOR									x	x	x	x	x	x	x	x
5	XOR	x	x	x	x	x	x	x	x								
4	XOR	x	x							x	x	x	x	x	x		
3	XNOR			x	x	x				x	x	x				x	x
2	XNOR	x		x			x	x		x			x	x			x
1	XOR				x		x		x		x		x		x	x	x
0	XOR		x			x		x	x	x	x		x				x

When the new set of check bits is equal to the original set of check bits, none of the data bits or original check bits changed value during the transfer from system memory to the microprocessor. When the new set of check bits is not equal to the original check bits, one or more of the data or check bits changed value after the microprocessor originally generated them.

When the hardware in the microprocessor detects that one bit changed value, the hardware corrects that bit and, if enabled, indicates a correctable read (CRD) interrupt. When more than one bit changed value, the hardware signals the microprocessor to halt program instructions and initiate a privileged architecture library (PAL) code machine check handler. The PAL code signals the operating system that an uncorrectable hardware error occurred.

Local Memory

Each PE contains local memory. Local memory consists of dynamic random access memory (DRAM) that stores system data. A low-latency, high-bandwidth data path connects the microprocessor to local memory in a PE.

System data is stored in a physically distributed, logically shared memory. Memory is physically distributed because each PE contains a local memory. Memory is logically shared because any microprocessor can access data in the local memory of any PE without involving that PE's microprocessor.

Figure 3-6 illustrates the physical distribution of memory in a 256-PE CRAY T3D system. The local memory in each PE stores a set number of 64-bit words (represented by the variable *m* in Figure 3-6). The size of local memory depends on the type of DRAM integrated circuits used in the system.

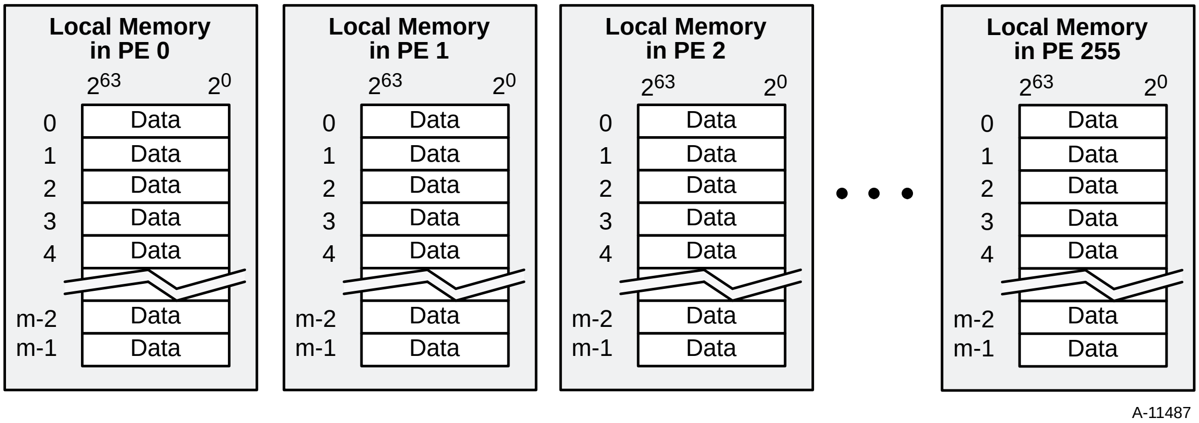


Figure 3-6. Physical Distribution of Memory

The total size of shared memory is the size of local memory in one PE multiplied by the total number of PEs in the system. For example, a CRAY T3D system with 512 PEs, each with 2 Mwords of local memory, has a total system memory of 1 Gword.

PE Support Circuitry

The PE support circuitry extends the control and addressing functions of the microprocessor. These functions include:

- Address interpretation
- Reads and writes
- Data prefetch
- Messaging
- Barrier synchronization
- Fetch and increment
- Status

More information on these functions is provided in Section 6, “Addressing,” and Section 7, “Operations.”

Hardware Description

The hardware of a processing element node resides on half of a printed circuit board (refer to Figure 3-7). This hardware consists of PE options, a BLT option, and network interface options.

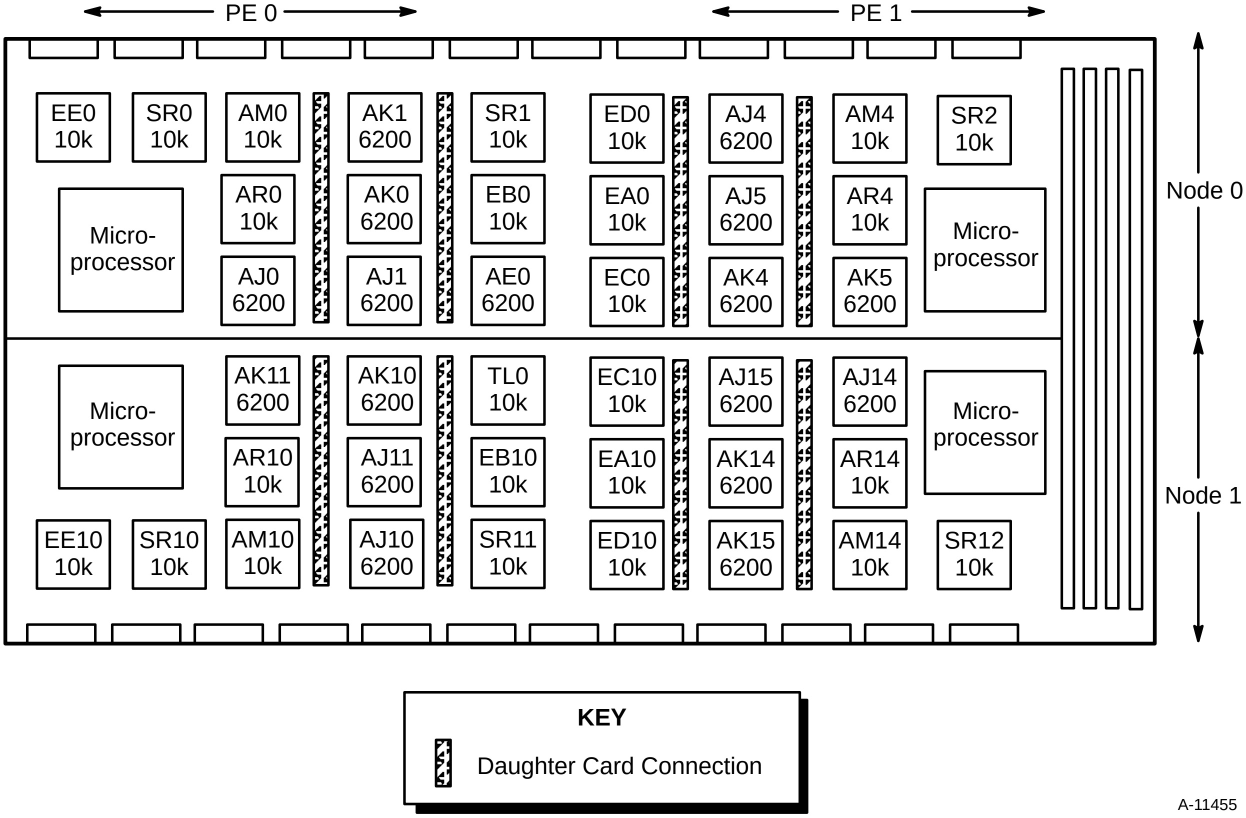


Figure 3-7. Processing Element Nodes 0 and 1

Each PE contains a microprocessor, local memory, and PE support circuitry. The following subsections describe the hardware used in these components.

Microprocessor

The microprocessor is self-contained in a package that is 2.4 in. x 2.4 in. as shown in Figure 3-8.

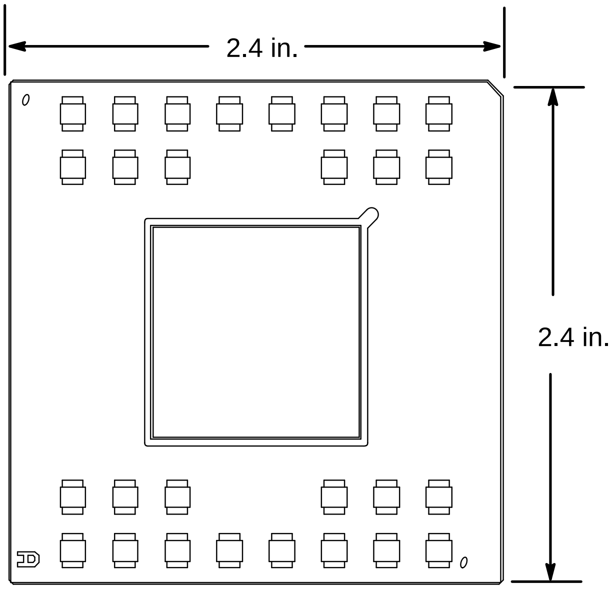


Figure 3-8. Microprocessor Package Dimension

Each microprocessor is a 431-pin grid array (PGA) integrated circuit. The microprocessor is mounted to the printed circuit board using a pin and socket connection (refer to Figure 3-9).

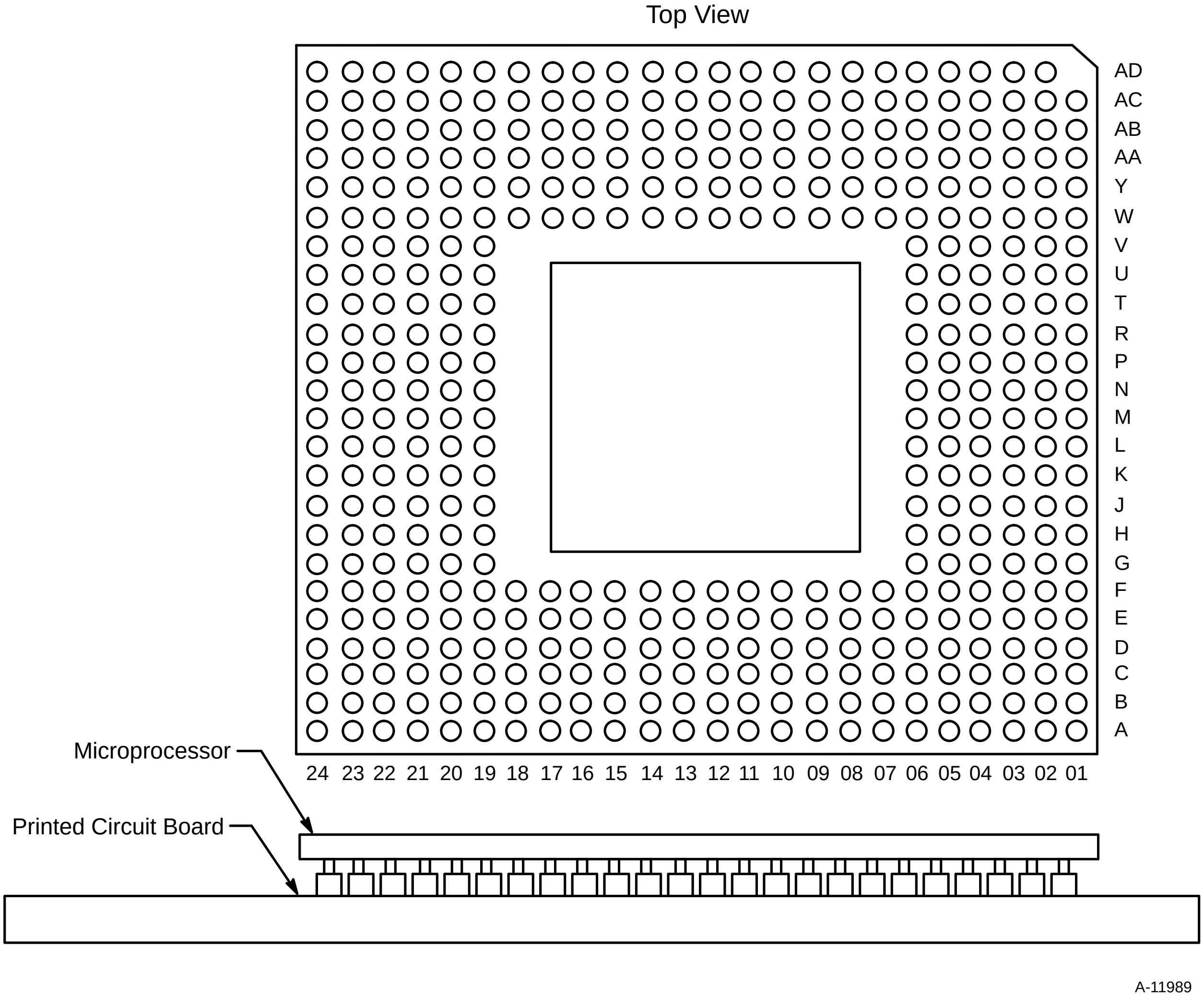


Figure 3-9. Microprocessor Connection to the Printed Circuit Board

The microprocessor communicates with the support circuitry using the signals identified in Figure 3-10. The following text describes each of these communication signals.

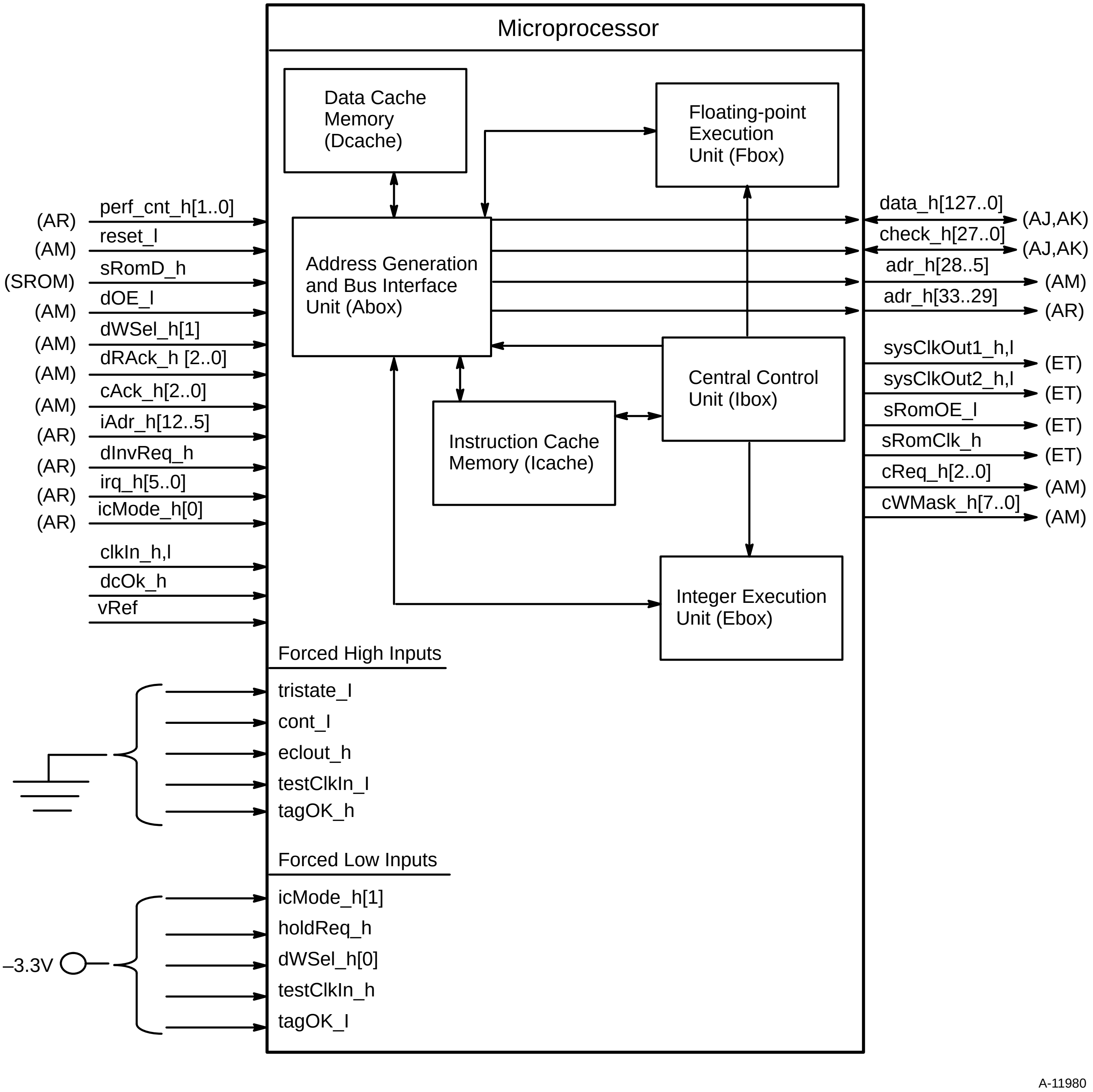


Figure 3-10. Input and Output Signals of the Microprocessor

- adr_h [bits 29 through 33] – The adr_h signals send the address bits 29 through 33 to the AR option. These bits are used to address the DTB annex.

- adr_h [bits 5 through 28] – The adr_h signals send the address offset to the AM option.

- cAck_h [bits 0 through 2] – The AM option sends the cAck_h signals to the microprocessor. The three bits are encoded to reflect the cycle acknowledge codes shown in Table 3-3.

Table 3-3. Cycle Acknowledge Codes

cAck_h Bits 2–0	Acknowledge Code
000	IDLE
001	HARD_ERROR
010	SOFT_ERROR
011	STxC_FAIL
100	OK

- check_h [bits 0 through 27] – The check_h signals are bidirectional signals that transfer the check bits between the microprocessor and the AJ, AK, and AR options.

- clkIn_h [bit 1] – The clkIn_h,1 signal inputs the clock signal.

- cReq_h [bits 0 through 2] – The cReq_h signals specify what type of request the microprocessor is making. These bits are sent to the AM option. The AM option decodes the cReq bits to determine the type of request, as shown in Table 3-4.

Table 3-4. Cycle Request

cReq_h Bits 2–0	Type of Request
000	IDLE
001	BARRIER
010	FETCH
011	FETCHM
100	READ_BLOCK
101	WRITE_BLOCK
110	LDxL
111	STxC

- cWMask_h [bits 0 through 7] – The cWMask_h signals specify address bits 2 through 4 during a READ_BLOCK request and to specify valid words of data during a WRITE_BLOCK request. During the READ_BLOCK, cWMask_h bits 1, 0, and 3 correspond to address bits 4, 3, and 2 respectively. During the WRITE_BLOCK, a bit of the cWMask_h indicates a valid 32-bit halfword within a cache line.

- data_h [bits 0 through 127] – The data_h signals are bidirectional signals that transfer data between the microprocessor and the AJ, AK, and AR options.

- dcOK_h – The dcOK_h signal switches clock sources between an on-chip ring oscillator (0) and an external clock oscillator (1).

- dInvReq_h – The dInvReq_h signal is sent to the microprocessor by the AM option. The microprocessor samples this signal every clock period. If this signal is a one, a location within the data cache will be invalidated. The invalid location is specified by the iAdr_h pins.

- dRAck_h [bits 0 through 2] – The AM option sends the dRAck_h signals to the microprocessor. The three bits are encoded to reflect the read data acknowledgements shown in Table 3-5.

Table 3-5. Read Data Acknowledgement

dRAck_h Bits 2–0	Acknowledgement Type
000	Idle
100	OK_NCACHE_NCHK
101	OK_NCACHE
110	OK_NCHK
111	OK

- dWsel_h [bit 1] – The AM option sends the dWsel_h signal to the microprocessor. The microprocessor uses this bit to determine which 2 words of the cache line should be placed on the data bus.

- dOE_l – The AM option sends the dOE_1 signal to the microprocessor. This signal enables the microprocessor to output data to the AJ, AK, and AR options.

- iAdr_h [bits 5 through 12] – The iAdr_h signals transfer the index of the invalid location in the data cache to the microprocessor. The iAdr_h signals are generated by the AR option.

- icMode_h [0] – The AR option sends the icMode_h signal to the microprocessor. This signal enables the microprocessor to read from the serial read-only memory (SROM) after a reset.

- irq_h [bits 0 through 5] – The AR option uses the irq_h signals to inform the microprocessor of any hardware interrupts. The interrupts that are reported to the microprocessor are the heartbeat interrupt, the barrier interrupt, the message interrupt, the BLT interrupt, the I/O interrupt, and the error interrupt. Refer to Table 3-6 for more information about the hardware interrupt signals.

Table 3-6. Hardware Interrupt Signals

Interrupt Request Pin	HIER Register Bit	HIRR Register Bit	Name	Cleared by
0	9	10	Heartbeat interrupt	Writing a 1 to bit 2 (clear heartbeat interrupt bit) in the status and control register (SCR)
1	10	11	Barrier interrupt	Reading the value of the barrier synchronization mask and interrupt (BSMI) register
2	11	12	Message interrupt	Writing to the message queue limit increment register (MQ_LIR)
3	12	5	BLT interrupt	Writing to the BLT control register (BLT_CR)
4	13	6	I/O interrupt	For information on clearing the I/O interrupt bit, refer to “Interrupt Flag Register” in Section 11, “LOSP and HISP Channels.”
5	14	7	Error interrupt	Reading the value stored in the system status register (SSR)

- perf_cnt_h [bits 0 and 1] – The perf_cnt_h signals inform the microprocessor's internal performance monitor of off-chip events. If a perf_cnt_h signal is set and selected as the source of its respective performance counter, the performance counter increments, acknowledging the event.

- reset_l – When the reset_1 signal is a 1, the microprocessor will reset its internal logic. The AM option sends this signal to the microprocessor.

- sRomClk_h – The sRomClk_h signal supplies a clock pulse to the static random access memory (SROM) on the processing element module and the I/O module. This clock pulse enables the SROM to output the next bit to the microprocessor.

- sRomD_h – The sRomD_h signal transfers data from the SROM to the microprocessor.

- sRomOE_l – The microprocessor sends the sRomOE_1 signal to the SROM. The sRomOE_1 signal serves as both an output enable to the SROM and a reset.

- sysClkOut1_h,l – The sysClkOut1–h,1 signal transfers the system clock from the microprocessor to the external logic.

- sysClkOut2_h,l – The sysClkOut2–h,1 signal transfers the system clock from the microprocessor to the external logic.

- vRef – The vRef signal is the analog reference voltage used by the input buffers of most signals.

Local Memory

Local memory consists of DRAM integrated circuits that are mounted on daughter-card printed circuit boards. The DRAM daughter cards plug into the processing element circuit board and reside on top of the integrated circuits used in the PEs (refer to Figure 3-11 and Figure 3-12).

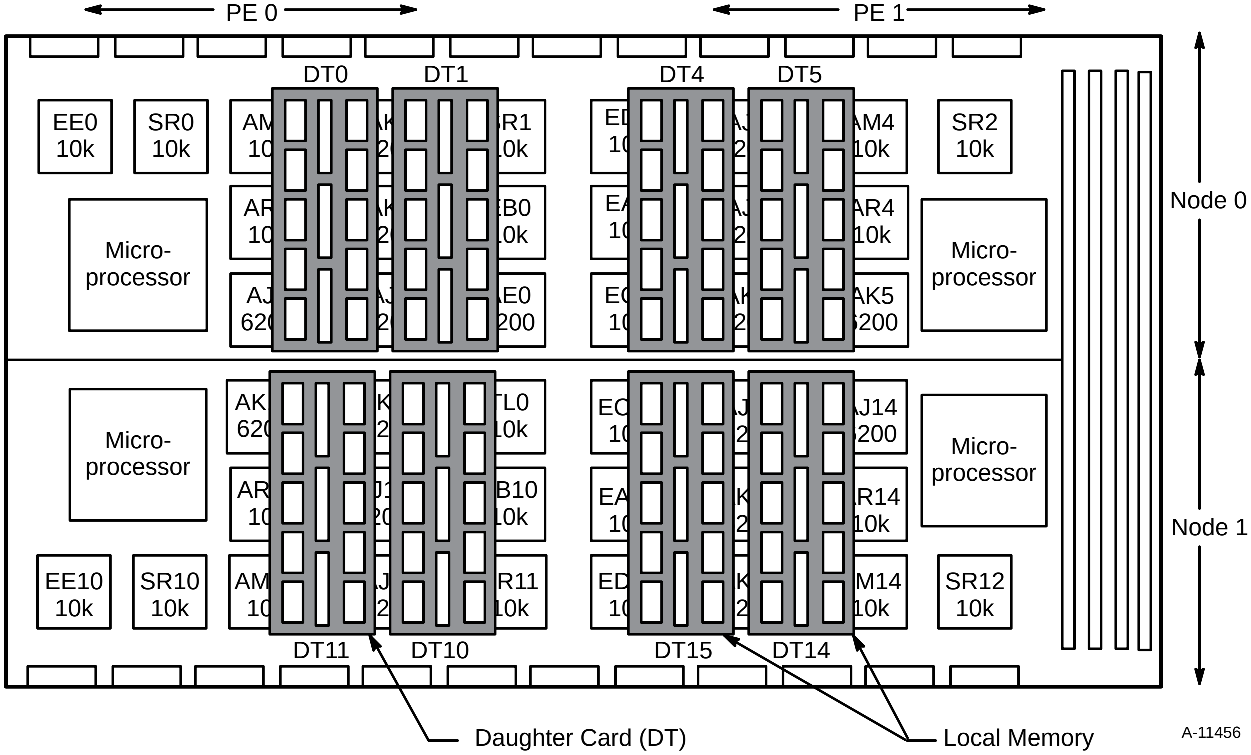


Figure 3-11. Processing Element Module with Daughter Cards

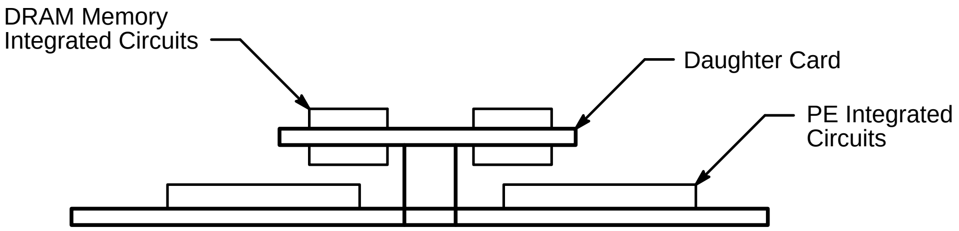


Figure 3-12. Side View of Processing Element Circuit Board

Each daughter card consists of one printed circuit board that houses memory chips on both the top and bottom of the printed circuit board. There are 20 memory chips that reside on 1 daughter card, 10 memory chips on each side.

Local memory consists of 2 daughter cards that contain 2 million or 8 million words of storage. For 2 million words of storage, 256K x 16 CMOS DRAM memory parts are used. For 8 million words of storage, 1M x 16 CMOS DRAM memory parts are used. Figure 3-13 shows the pinout for the 256K x 16 CMOS DRAM memory part.

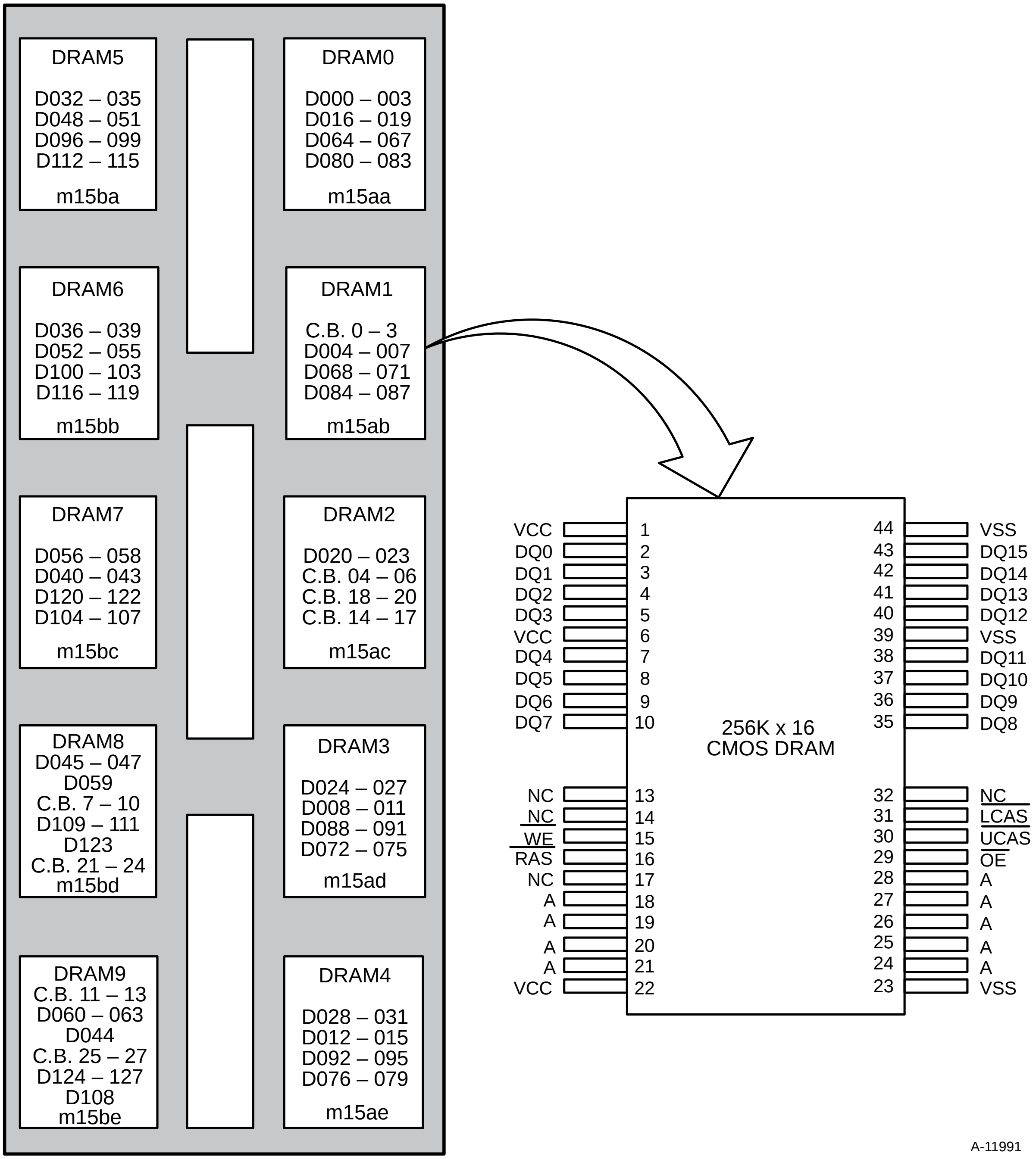
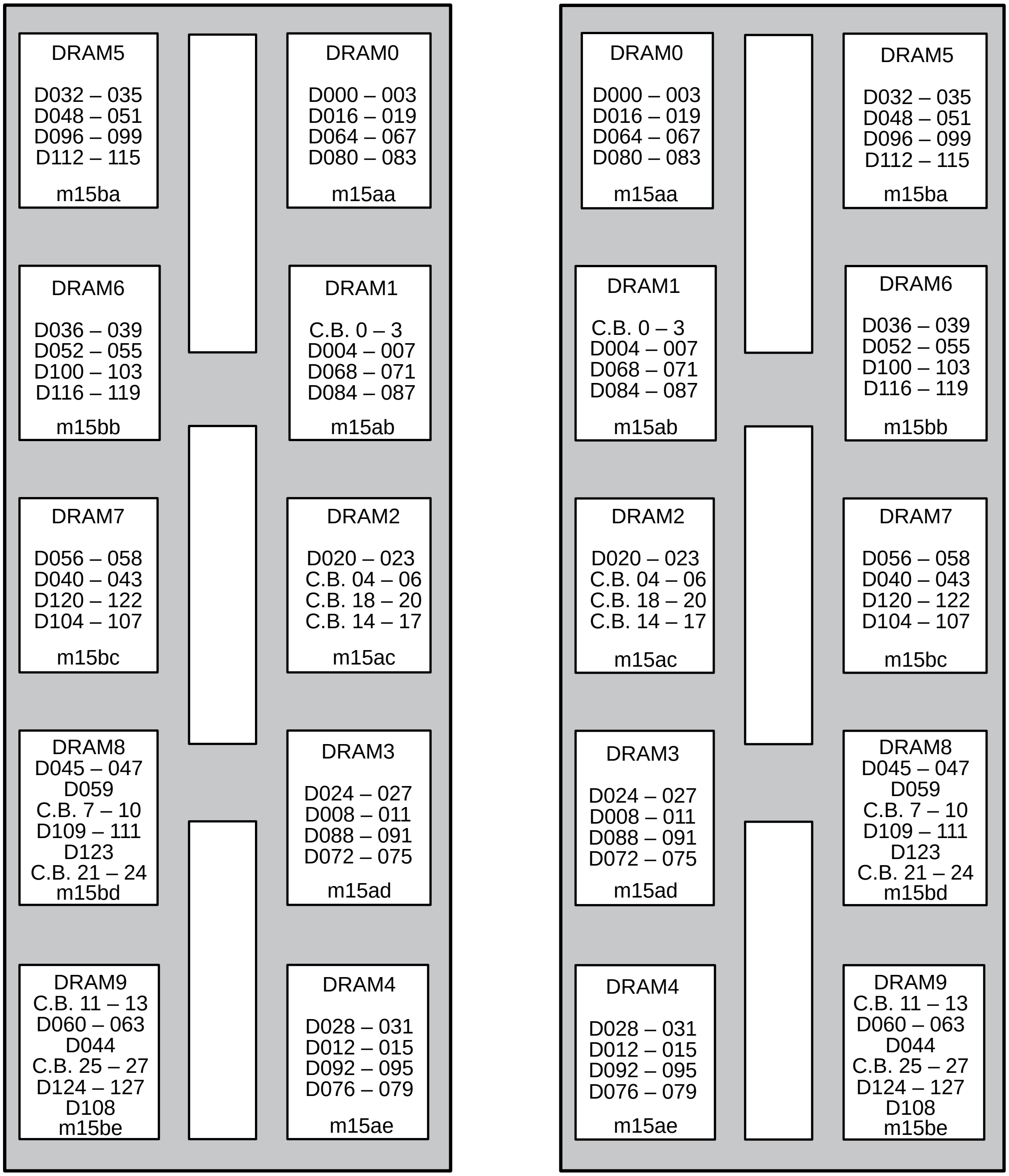


Figure 3-13. Pinout of 256k x 16 CMOS DRAM Memory Part

Each memory chip handles 16 bits of data. The data enters and leaves the memory chips using the bidirectional pins 2 through 5, 7 through 10, 35 through 38, and 40 through 43. These pins are used to output data during the time that the output enable (OE) (inverted) is a 1. The data pins are used to input data when the write enable (inverted) is a 1 and the OE (inverted) is a 0.

The data is distributed to the memory chips as shown in Figure 3-14 and Figure 3-15. The data is also distributed to the top and bottom halves of the daughter card concurrently. Figure 3-14 and Figure 3-15 show the data bit layout for both halves of the daughter cards. Note that the top half is referred to as Rank A and the bottom half is referred to as Rank B.

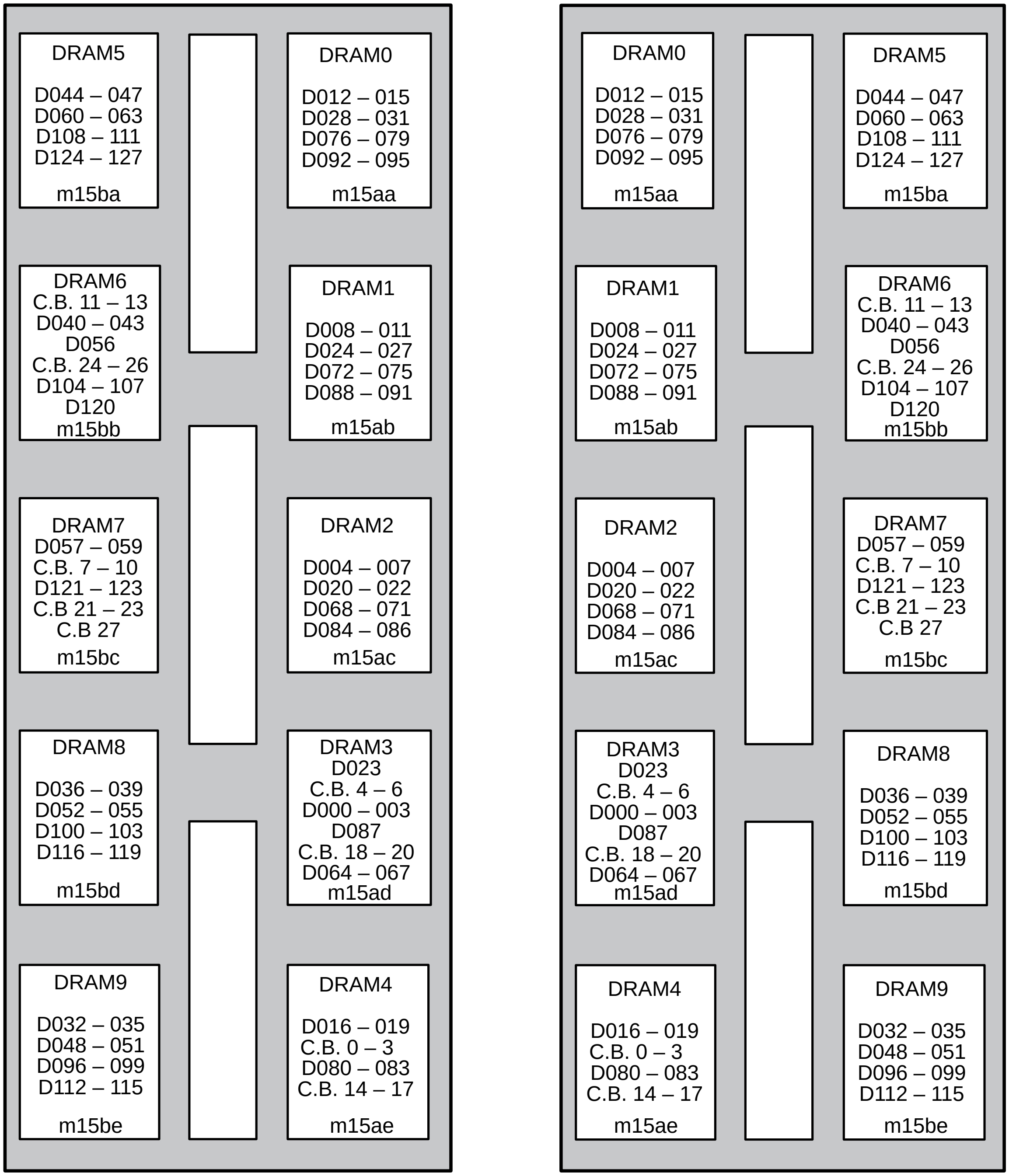
Each rank receives its own address, write enable (WE), RAS 0 – 1, and CAS 0 – 3 signals. The PE support logic generates the address and control signals that are presented to the memory chips. This logic also transfers the data between local memory, the microprocessor, and the network interface logic. Refer to Figure 3-16. A detailed description of the PE support logic is given in the next subsection.



Top Half of the Daughter Card
Rank A

Bottom Half of the Daughter Card
Rank B

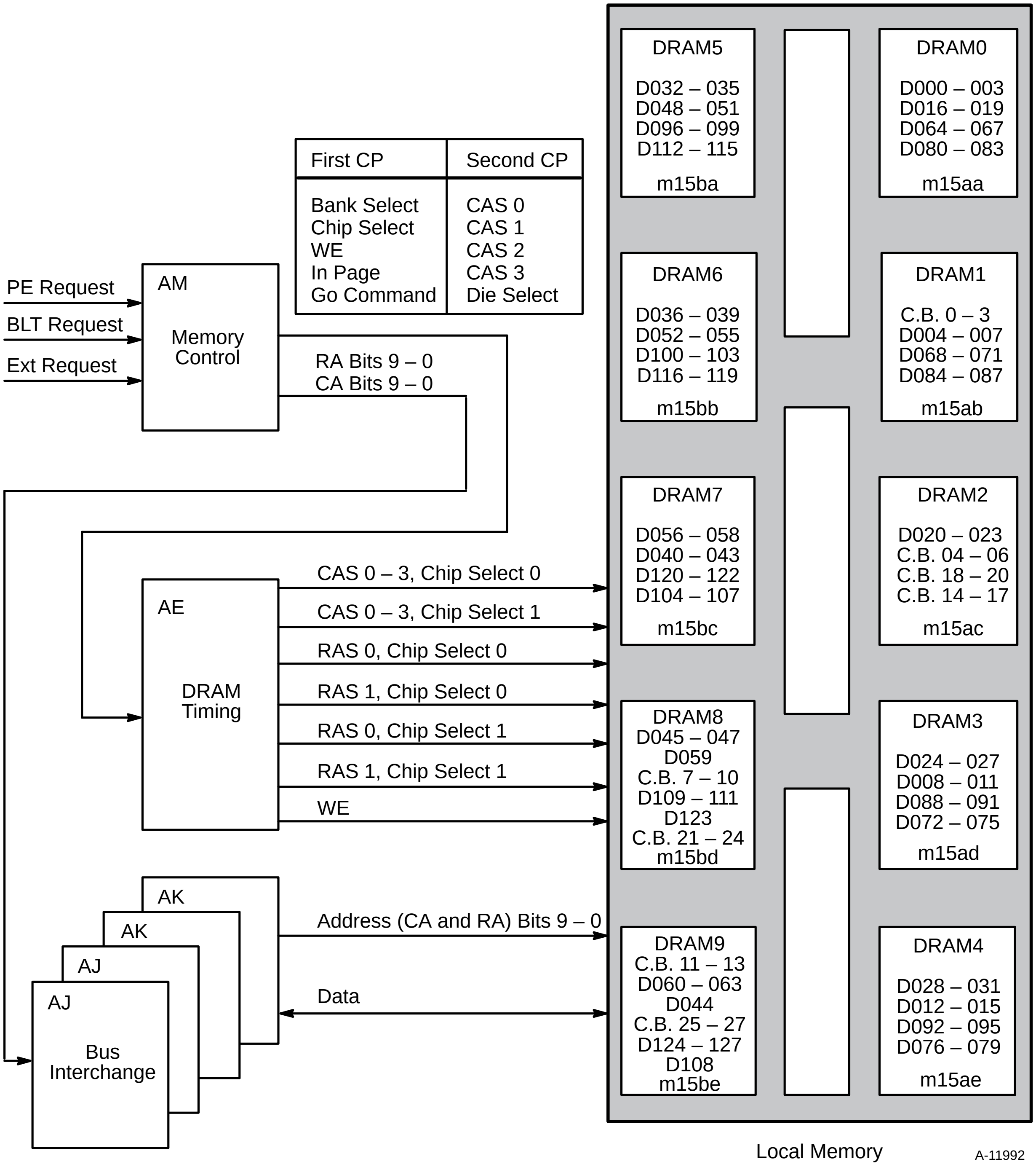
Figure 3-14. Data Bit Layout for PEM Daughter Cards 0, 1, 14, and 15



Top Half of the Daughter Card
Rank A

Bottom Half of the Daughter Card
Rank B

Figure 3-15. Data Bit Layout for PEM Daughter Cards 4, 5, 10, and 11



NOTE: The row address strobe (RAS) asserts when referencing a new page or doing a refresh. A new page reference occurs when a different page in the same die is referenced, when the other die in the rank is referenced, or after a refresh. The column address strobe (CAS) 0 through 3 asserts when referencing the same page or doing a refresh. The CAS mask indicates which of the 4 CAS signals will be asserted for the reference. The CAS mask points to valid halfwords within a cache line. The CAS asserts during the first clock period for a read and during the second clock period for a write. The CAS asserts before the RAS during a refresh. The write enable asserts during the first clock period of the write and continues to assert for the duration of the CAS.

Figure 3-16. PE Support Logic Distributing Data, Address, and Control to a Daughter Card

PE Support Circuitry

The PE support circuitry consists of several option types that extend the control and addressing functions of the microprocessor. These options are the AE option, the AJ option, the AK option, the AM option, and the AR option. Refer to Figure 3-17.

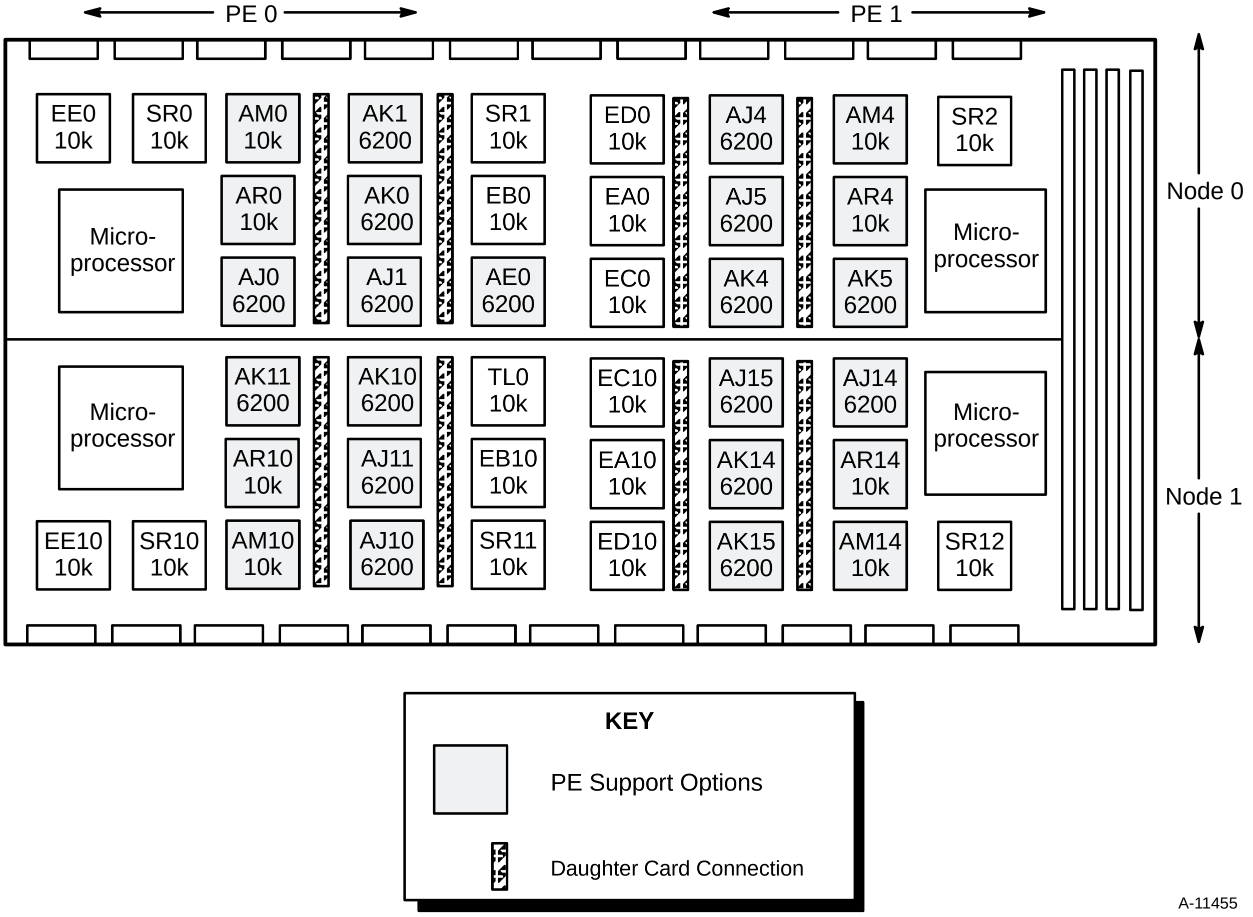


Figure 3-17. PE Support Circuitry

AE Option - CAS, RAS, and WE Generation and DRAM Timing

The AE option generates the CAS 0 – 3, RAS 0 – 1, and WE signals for the four PEs on a printed circuit board. Each node must initialize its portion of the AE option before it can make any memory references. Refer to Figure 3-18.

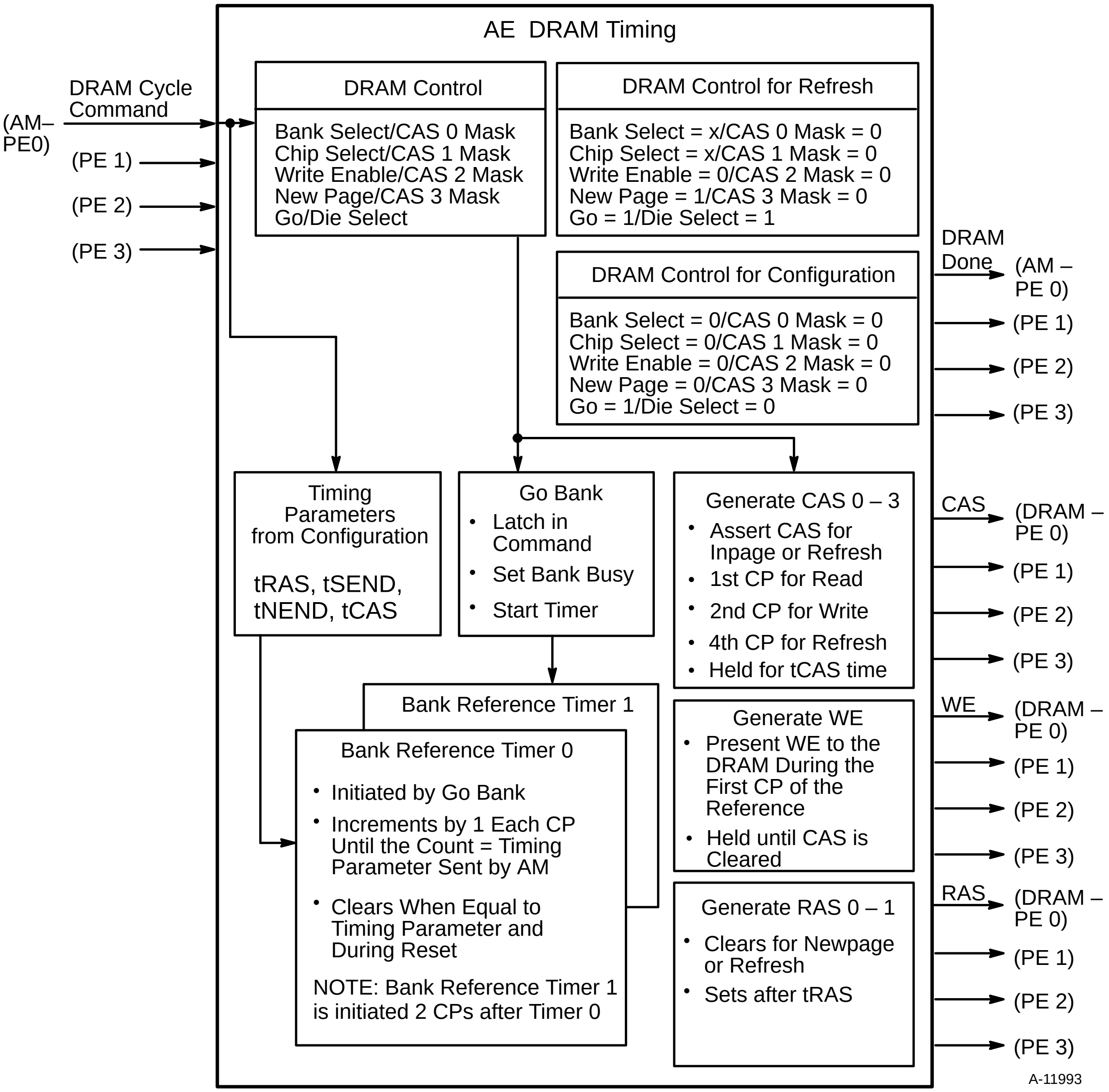


Figure 3-18. AE Option

The AE option generates the CAS 0 – 3, RAS 0 – 1, and WE signals by using the DRAM control commands it receives from the AM options. The AE option receives the DRAM control commands over two clock periods, as shown in Table 3-7.

Table 3-7. DRAM Control Command from the AM Option

1st Clock Period	2nd Clock Period
Bank Select	CAS 0 Mask
Chip Select	CAS 1 Mask
Write Enable	CAS 2 Mask
New Page	CAS 3 Mask
Go	Die Select

The CAS 0 – 3 signals enable the memory chips that are to be read from or written to. On the processing element module, each memory chip receives two CAS signals as shown in Figure 3-19. This arrangement enables a cache line of data to be read from or written to the two daughter cards without having to wait for a setup of a second column address. For example, the CAS 0 – 3 signals are presented to daughter card 0 to read or write word 0 and word 1. The same CAS 0 – 3 signals are presented to daughter card 1 to read or write word 2 and word 3. Daughter card 1 receives the CAS 0 – 3 signals two clock periods later than daughter card 0.

NOTE: The CAS 0 – 3 signals assert when there is an inpage reference or a refresh reference. An inpage cycle occurs when referencing a memory location with the same row address as the previous memory reference.

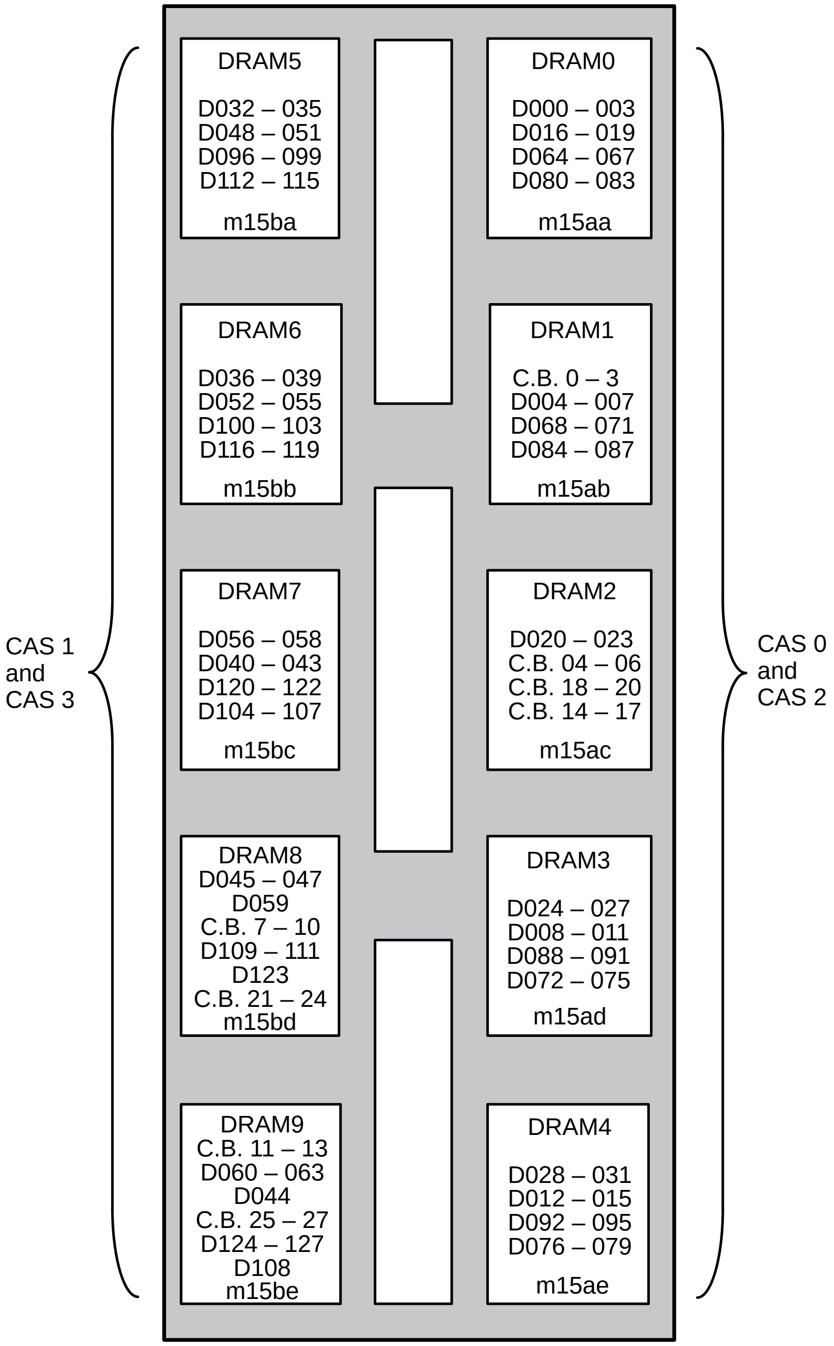


Figure 3-19. CAS 0 – 3 Designation on the PEM Daughter Card

The RAS 0 – 1 signals enable one of the two dies that may be present within the memory chips. If there is only one die within the memory chip, the RAS 1 signal is not used.

The appropriate RAS signal will assert according to the die select. The RAS signal will only clear at the beginning of a newpage cycle or a refresh cycle.

NOTE: A newpage cycle occurs when there is a reference to a different page in the same die, when there is a reference to the other die within the rank, or as part of any refresh operation.

The write enable enables the memory chips to input data using the bidirectional data pins. The write enable is asserted in the selected rank during write references from the first clock period after receiving the command until the bank CAS timing counter is reset to 0.

The AM option also configures the AE option with timing parameters for the four types of reference: refresh, newpage, samepage reads, and samepage writes. The AE option uses the timing parameters to supply the CAS 0 – 3, RAS 0 – 1, and WE to the memory chips for the appropriate amount of time.

To determine how long a signal should be asserted to the memory chips, the AE option compares the timing parameter data to the increment of the bank timing counters. The bank timing counters are initiated by the Go Bank signal. The counters increment by 1 each clock period until the value of the counter equals the timing parameter of the reference specified by the command. When this happens, the AE option stops asserting the signal to the memory chips and the bank timing counter is reset to 0.

AJ and AK Options - Data Bus Interface, Data Buffering, and Data Muxing

The AJ and AK options transfer data between local memory, the microprocessor, and the network interface logic. The AJ and AK options also distribute the address offset to local memory and contain the swaperand memory-mapped register. Refer to Figure 3-20. The swaperand register is described in Section 7, "Operations."

There are two AJ options and two AK options per PE. The AJ and AK options are essentially identical with the exception of the polarity of certain control signals sent by the AM option.

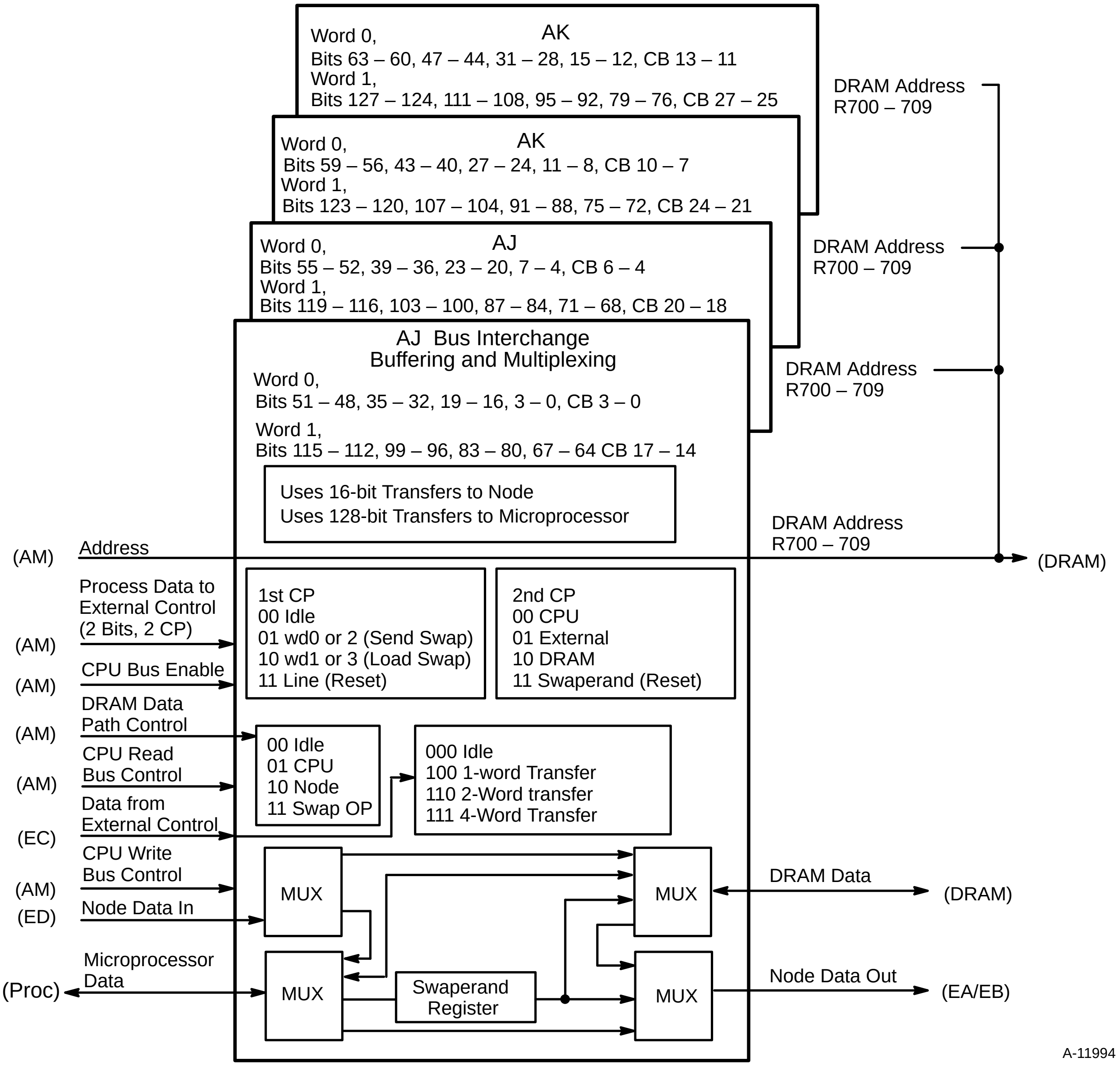


Figure 3-20. AJ and AK Options

The AJ and AK options receive control from the AM option that they use to steer the data and check bits to the correct destination. These control signals consist of DRAM data path control, CPU data path control, and external data path control.

The DRAM Write Data Path Control signal is a 2-bit code that specifies whether the microprocessor data, external data, or swaperand register data should be written to memory. The decode of the 2-bit code is shown in Table 3-8.

Table 3-8. DRAM Write Data Path Control

2-bit Code	Description
00	Idle
01	Microprocessor
10	External
11	Swaperand Register

Table 3-9 and Table 3-10 list the data bit distribution from the AJ and AK options to the daughter cards.

Table 3-9. Data Bit Distribution to Daughter Cards 0, 1, 14, and 15

Option	Data Bits	Memory Chips in Ranks A and B
AJ0	0 – 3, 16 – 19, 64 – 67, 80 – 83	DRAM 0
AJ1	4 – 7, 68 – 71, 84 – 87	DRAM 1
AJ0	C.B. 0 – 3	DRAM 1
AJ1	20 – 23, C.B. 4 – 6, C.B. 18 – 20	DRAM 2
AJ0	C.B. 14 – 17	DRAM 2
AK0	8 – 11, 24 – 27, 72 – 75, 88 – 91	DRAM 3
AK1	12 – 15, 28 – 31, 76 – 79, 92 – 95	DRAM 4
AJ0	32 – 35, 48 – 51, 96 – 99, 112 – 115	DRAM 5
AJ1	36 – 39, 52 – 55, 100 – 103, 116 – 119	DRAM 6
AK0	40 – 43, 56 – 57, 104 – 107, 120 – 122	DRAM 7
AK0	59, 123, C.B. 7 – 10, C.B. 21 – 24	DRAM 8

Table 3-9. Data Bit Distribution to Daughter Cards 0, 1, 14, and 15 (continued)

Option	Data Bits	Memory Chips in Ranks A and B
AK1	45 – 47, 109 – 111	DRAM 8
AK1	60 – 63, 124 – 127, C.B. 11 – 13, C.B. 25 – 27	DRAM 9

Table 3-10. Data Bit Distribution to Daughter Cards 4, 5, 10, and 11

Option	Data Bits	Memory Chips in Ranks A and B
AK1	12 – 15, 28 – 31, 76 – 79, 92 – 95	DRAM 0
Ak0	8 – 11, 24 – 27, 72 – 75, 88 – 91	DRAM 1
AJ1	4 – 7, 20 – 22, 68 – 71, 84 – 86	DRAM 2
AJ0	0 – 3, 64 – 67,	DRAM 3
AJ1	23, 87, C.B. 4 – 6, C.B. 18 – 20	DRAM 3
AJ0	16 – 19, 80 – 83, C.B. 0 – 3, C.B. 14 – 17	DRAM 4
AK1	44 – 47, 60 – 63, 108 – 111, 124 – 127	DRAM 5
AK0	40 – 43, 56, 104 – 107, 120, C.B 24	DRAM 6
AK1	C.B. 11 – 13, C.B. 25 – 26	DRAM 6
AK0	57 – 59, 121 – 123, C.B. 7 – 10, C.B. 21 – 23	DRAM 7
AK1	C.B. 27	DRAM 7
AJ1	36 – 39, 52 – 55, 100 – 103, 116 – 119	DRAM 8
AJ0	32 – 35, 48 – 51, 96 – 99, 112 – 115	DRAM 9

The DRAM Read Data Path Control signal selects 1 of the 2 words that are read out of memory to be delivered to the microprocessor or the external data staging (group of muxes). The AJ and AK options select the read data from the microprocessor data staging and the external data staging using the CPU Data Path Control and the External Data Path Control signals respectively. From the data staging, the AJ and AK options steer the read data to the correct destination.

NOTE: The read data is transferred to the network interface logic in 16-bit transfers, 4 bits from each AJ and AK option. The read data is transferred to the microprocessor 2 words at a time.

The CPU Data Path Control consists of three control signals: Load CPU Memory Read Stage, CPU Read Data Select, and Advance CPU Read Data (refer to Figure 3-21). The Load CPU Memory Read Stage signal loads the local memory read data into the first set of CPU memory readout staging latches. The Advance CPU Read Data signal enables the read data to be passed to the next set of latches in the read data staging. From these latches, the read data is passed to a multiplexer circuit (MUX). The CPU Read Data Select signal selects local data or external data from this MUX to be passed to the CPU bus.

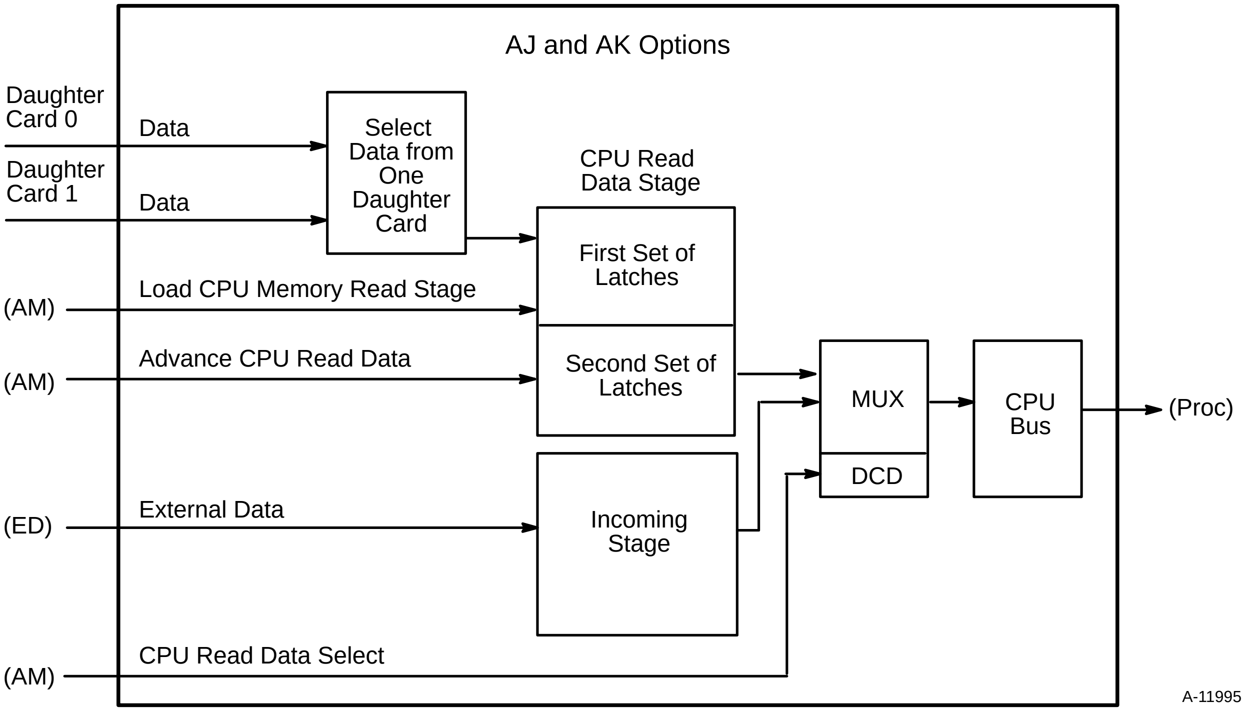


Figure 3-21. CPU Data Path Control

The External Outgoing Data Path Control signal is a 4-bit code sent to the AJ and AK options over 2 clock periods. During the first clock period, the 4-bit code specifies the amount of data to be transferred to the network interface logic. During the second clock period, the 4-bit code specifies the source of the data. The decode of this 4-bit code is shown in Table 3-11 and Table 3-12.

Table 3-11. External Outgoing Data Path Control (1st CP)

2-bit Code 1st CP	Description
00	Idle
01	Word 0 or 2 (send swap)
10	Word 1 or 3 (load swap)
11	Cache line or reset

Table 3-12. External Outgoing Data Path Control (2nd CP)

2-bit Code 2nd CP	Description
00	Microprocessor
01	External
10	DRAM
11	Swap or reset

NOTE: When this 2-bit code is set to 11 during both clock periods, the internal circuitry of the AJ and AK options is reset.

The External Incoming Data Path Control signal is sent by the EC option to control the transfer of data from the network interface (ED option) to the AJ and AK options. This control is a 3-bit code that is sent to the AJ and AK options 1 bit at a time over 3 clock periods. The decode of the 3-bit code is shown in Table 3-13.

Table 3-13. External Incoming Data Path Control

3-bit Code	Description
000	Idle
100	1-word transfer
110	2-word transfer
111	4-word transfer

AM Option - Processing Element Control

The AM option provides control for local and remote memory references. Refer to Figure 3-22. When the microprocessor initiates a request, the microprocessor sends the AM option a cycle request code and an address offset. The AM option also receives a PE number and a memory function code from the AR option. After receiving this information, the AM option determines whether the reference is local or remote.

The AM option determines whether a memory reference is local or remote by comparing the PE number from the AR option to the logical or virtual PE register. When the PE number matches the contents of the logical or virtual PE register, the reference is local. When the PE number does not match one of the registers, the reference is remote.

Local Reference

When the reference is local, the AM option analyzes the address offset to determine whether the reference is for local memory or for a memory-mapped register. If the address offset equals an address between 10000000_{16} and $1FFFFFFF_{16}$ (address bit 28 = 1), the reference is for a memory-mapped register. If the address equals an address between 00000000_{16} and $0FFFFFFF_{16}$ (address bit 28 = 0), the reference is for local memory.

When the reference is for a memory-mapped register, the AM option generates control to coordinate reading or writing of the register. For a write of a memory-mapped register in the network interface logic, the AM option instructs the AJ and AK options to steer the data from the microprocessor to the appropriate option in the network interface. For a write of a memory-mapped register in the AR option, the AM option sends control to the AR option to enable the AR option to input data from the microprocessor. The AM option also sends a command that informs the AR option which memory-mapped register to load.

For a write of a memory-mapped register in the AM option itself, the AM option sends the AR option control that enables the AR option to input data from the microprocessor. The AM option then instructs the AR option to send this data to the AM option. If the write is to a memory-mapped register in the AJ and AK options (for example, the swaperand register), the AM option sends the AJ and AK option the external outgoing data path control that indicates the swaperand register load.

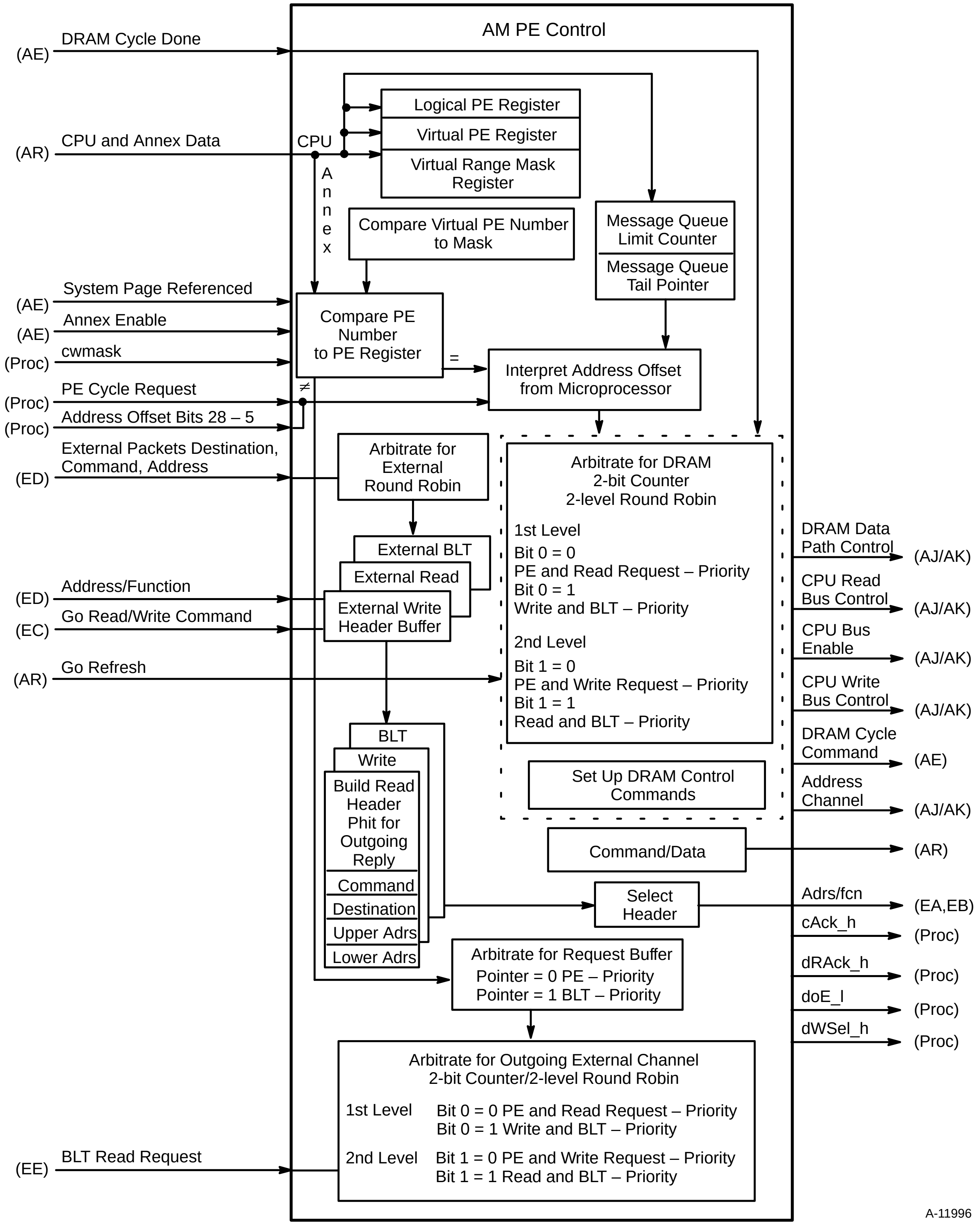

AM PE Control
(AE) DRAM Cycle Done
(AR) CPU and Annex Data
CPU
Annex
Logical PE Register
Virtual PE Register
Virtual Range Mask Register
Compare Virtual PE Number to Mask
Message Queue Limit Counter
Message Queue Tail Pointer
(AE) System Page Referenced
(AE) Annex Enable
(Proc) cwmask
(Proc) PE Cycle Request
(Proc) Address Offset Bits 28 – 5
Compare PE Number to PE Register
=
≠
Interpret Address Offset from Microprocessor
(ED) External Packets Destination, Command, Address
Arbitrate for External Round Robin
Arbitrate for DRAM 2-bit Counter 2-level Round Robin
1st Level
Bit 0 = 0
PE and Read Request – Priority
Bit 0 = 1
Write and BLT – Priority
2nd Level
Bit 1 = 0
PE and Write Request – Priority
Bit 1 = 1
Read and BLT – Priority
External BLT
External Read
External Write Header Buffer
(ED) Address/Function
(EC) Go Read/Write Command
(AR) Go Refresh
DRAM Data Path Control (AJ/AK)
CPU Read Bus Control (AJ/AK)
CPU Bus Enable (AJ/AK)
CPU Write Bus Control (AJ/AK)
DRAM Cycle Command (AE)
Address Channel (AJ/AK)
Set Up DRAM Control Commands
Command/Data
(AR)
BLT
Write
Build Read Header Phit for Outgoing Reply
Command
Destination
Upper Adrs
Lower Adrs
Select Header
Adrs/fcn (EA,EB)
cAck_h (Proc)
dRAck_h (Proc)
doE_l (Proc)
dWSel_h (Proc)
Arbitrate for Request Buffer
Pointer = 0 PE – Priority
Pointer = 1 BLT – Priority
Arbitrate for Outgoing External Channel 2-bit Counter/2-level Round Robin
1st Level Bit 0 = 0 PE and Read Request – Priority
Bit 0 = 1 Write and BLT – Priority
2nd Level Bit 1 = 0 PE and Write Request – Priority
Bit 1 = 1 Read and BLT – Priority
(EE) BLT Read Request
A-11996

Figure 3-22. AM Option

For a read of a memory-mapped register in the AR option, the AM option enables the data path between the microprocessor and the AR option. The AM option then sends control to the AR option to specify which memory-mapped register is to be read. When the microprocessor reads one of the memory-mapped registers of the AM option, the AM option again enables the data path between the AR option and the microprocessor, sends the data to the AR option, and instructs the AR option to pass this data to the microprocessor.

The AM option also checks for memory conflicts. Memory conflicts occur when more than one of the five types of references (refresh reference, PE reference, external read reference, external write reference, or BLT local read) request local memory at the same time. If a memory conflict occurs, the AM option arbitrates for local memory. The refresh is always given top priority. To arbitrate between the other references, the AM option uses a 2-bit counter to produce a 2-level round robin arbitration sequence. The first level uses bit 0 of the counter. When bit 0 equals 0, the PE and external read references have priority to reference local memory. When bit 0 equals 1, the BLT and external write references have priority. Bit 1 of the counter is used to determine the priority of the two remaining possible conflicts between PE and external read references or BLT and external write references. When bit 1 equals 0, the PE and external write references have priority to reference local memory. When bit 1 equals 1, the BLT and external read references have priority. Refer to Table 3-14.

Table 3-14. Arbitration for Local Memory

Level	Priority
1st Level	Bit 0 = 0 PE and Read Request – priority Bit 0 = 1 Write and BLT – priority
2nd Level	Bit 1 = 0 PE and Write Request – priority Bit 1 = 1 Read and BLT – priority

After the AM option determines which reference has priority to access memory, the AM generates DRAM control commands. These control commands are sent to the AE option over 2 clock periods. The Bank Select, Chip Select, Write Enable, New Page, and Go commands are sent during the first clock period and the CAS 0 Mask, CAS 1 Mask, CAS 2 Mask, CAS 3 Mask, and Die Select commands are sent during the second clock period. The AE option uses these commands to generate the

RAS 0 – 1, CAS 0 – 3, and Write Enable signals it sends to the DRAM. The AM option also sends the address offset to the AJ and AK options. The AJ and AK options send the address offset to local memory.

Remote References

The AM option also controls the setup for a remote reference. This setup involves validating the virtual PE number, creating header phits, and arbitrating for network interface request buffers.

Once the AM option determines the reference is remote, the AM option checks the validity of the virtual PE number by comparing the virtual PE number from the DTB annex to the virtual range mask register. If the virtual PE number is not valid, the AM option sets the Virtual PE number range error bit in the system status register and signals the AR option to set the error interrupt. If the virtual PE number is valid, the AM option builds a packet header that contains a command phit, a destination phit, an upper address phit, and a lower address phit.

The AM option generates the command phit by using the PE cycle request code, the cWMask, address bits 21 and 28, and the memory function code. Figure 3-23 shows the bit layout for the command phit.

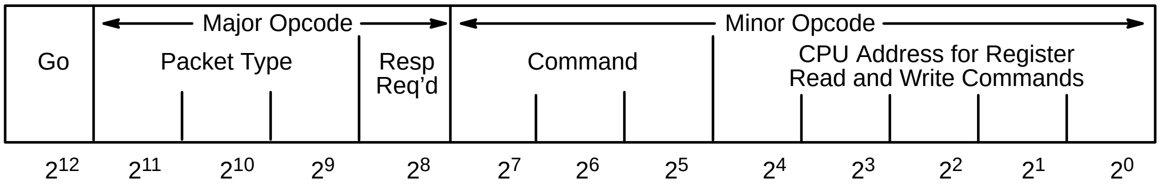


Figure 3-23. Command Phit

For the destination phit, the AM option uses the remote PE number from the AR option. The AM option also appends a bit to the remote PE number. This bit becomes bit 12 of the destination phit. The AM option sets this bit if the remote PE number is logical. Refer to Figure 3-24.

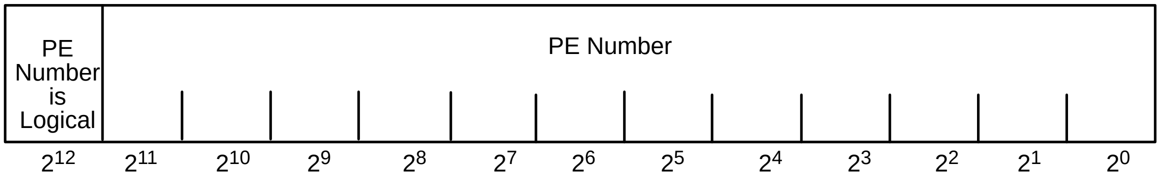


Figure 3-24. Destination Phit

For the upper and lower address phits, the AM option copies the address offset into the two address phits. The bit layout of the address phits is shown in Figure 3-25.

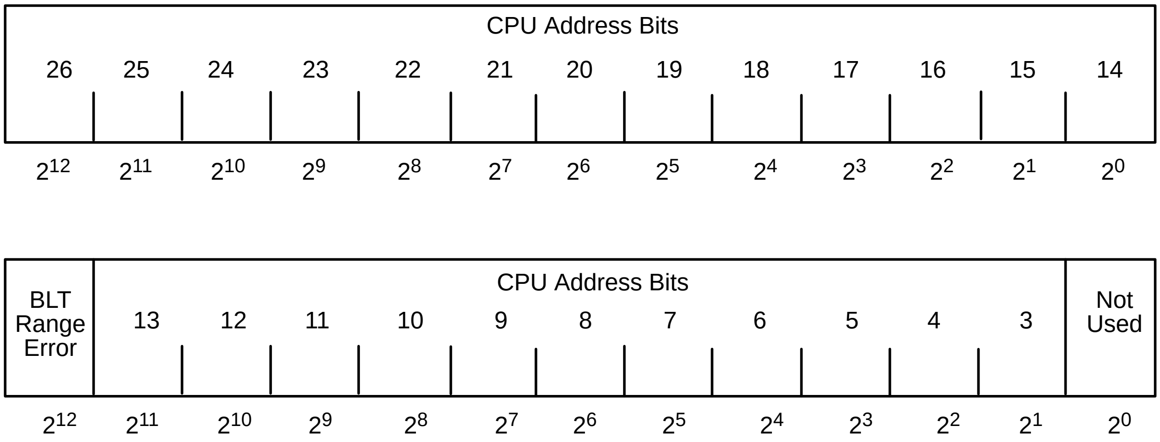


Figure 3-25. Address Phits

The AM option sends the command phit and the destination phit to the EB option and sends the two address phits to the EA option (via the same outgoing external channel) after it checks for request buffer conflicts and outgoing external channel conflicts. A request buffer conflict occurs when a microprocessor request and a BLT local read need the request buffer on the EA option at the same time. An outgoing external channel conflict occurs when a microprocessor request, a BLT request, and an external read response or an external write response conflict need the outgoing external channel at the same time. When a request buffer conflict or an outgoing external channel conflict occurs, the AM option arbitrates to determine which reference has priority to access the buffers.

The AM option arbitrates for the request buffer of the EA option using an arbitration pointer. When this arbitration pointer equals 0, the microprocessor request has priority to use the request buffer. When the arbitration pointer equals 1, the BLT request has priority. The arbitration pointer toggles each time the microprocessor or BLT requests to use a request buffer.

The AM option arbitrates for the outgoing external channel to the EA option using a 2-bit counter to produce a 2-level round robin arbitration sequence. The first level uses bit 0 of the counter. When bit 0 equals 0, the microprocessor request and external read response have priority to use the buffers. When bit 0 equals 1, the BLT request and external write response have priority. Bit 1 of the counter is used to determine the priority of the two remaining possible conflicts between a microprocessor request and an external read response or between a BLT request and an external write response. When bit 1 equals 0, the microprocessor request and external write response have priority to use the buffers. When bit 1 equals 1, the BLT request and external read response have priority. After each successful arbitration, the AM option increments the arbitration pointer, changing the outgoing external channel priority scheme.

The AM option also provides control for external requests. When an external request is received by the node, the node interface circuitry sends a copy of the request packet to the AM option in the destination PE. The AM option uses this information to determine what type of request is being made and to generate the appropriate control to satisfy the request. The AM option also uses this information to build the header portion of the response packet.

Memory-mapped Registers

The AM option contains several memory-mapped registers. Table 3-15 lists these memory-mapped registers along with each register address.

Table 3-15. Memory-mapped Registers in the AM Option

Address	Description	Name
10144000_{16}	User status register	USR
10448000_{16}	System status register	SSR
$1044C000_{16}$	DRAM Control register	DRAM_CR
10450000_{16}	Logical PE number register	L_WHOAMI
10454000_{16}	Virtual PE number register	V_WHOAMI
10458000_{16}	Virtual PE range mask register	VRT_RG
10460000_{16}	Message queue limit increment register	MQ_LIR
10464000_{16}	Message queue tail register	MQ_TP
10544000_{16}	Invalidate control register	INV_CR

AR Option - DTB Annex, Prefetch Queue, and Hardware Interrupts

The AR option handles all hardware interrupts. The AR option also contains the DTB annex and the prefetch queue. Refer to Figure 3-26.

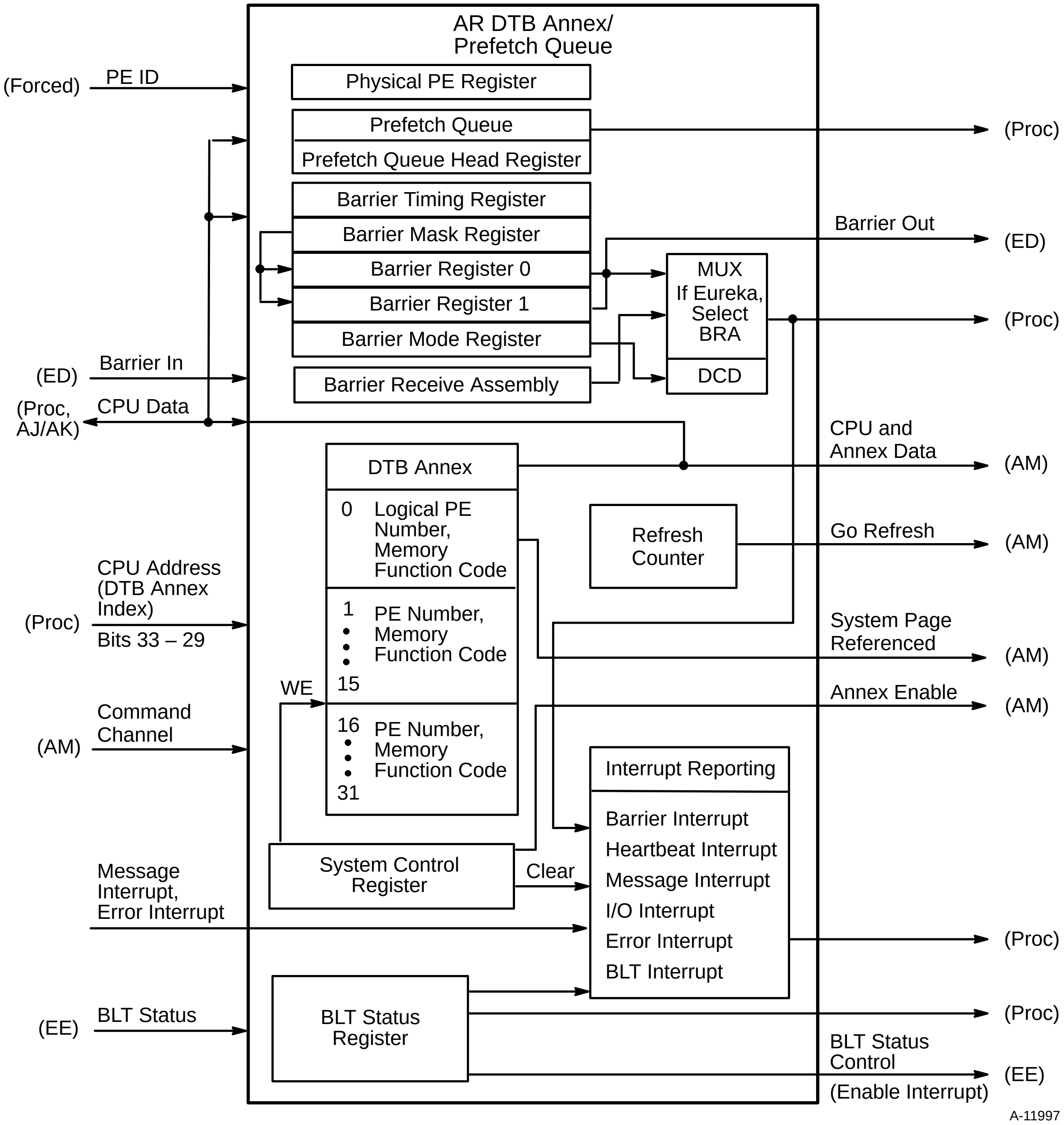


Figure 3-26. AR Option

Interrupt Handling

The AR option receives notification of all hardware interrupts. These interrupts include the heartbeat interrupt, the barrier interrupt, the message interrupt, the BLT interrupt, the I/O interrupt, and the error interrupt.

The AR option sets the heartbeat interrupt after it receives the heartbeat signal generated by the system printed circuit board. The AR option clears the heartbeat interrupt when bit 2 in the system control register is set by the microprocessor.

The AR option sets the barrier interrupt after it receives notification from the barrier synchronization logic (ED options) that a barrier or eureka synchronization operation is complete. The AR option clears the barrier interrupt after the microprocessor reads the contents of the barrier synchronization mask and interrupt register.

The AR option sets the message interrupt after receiving notification from the AM option that a message has arrived. The AR option clears the message interrupt after the microprocessor writes to the message queue limit increment register in the AM option.

The AR option sets the BLT interrupt after receiving notification from the EE option. The AR option clears the BLT interrupt after the microprocessor writes the BLT control register.

The AR options that reside on the I/O modules handle the I/O interrupt. For more information about the I/O interrupt, refer to Section 11, “LOSP and HISP.”

The AR option sets the error interrupt after receiving notification from the AM option that an error occurred. The AR option clears the error interrupt after the microprocessor reads the contents of the system status register.

DTB Annex

The DTB annex extends the addressing capabilities of the microprocessor. The DTB annex is a 32-entry RAM cell that is written by the microprocessor with a virtual or logical PE number and a memory function code. Refer to Figure 3-27.

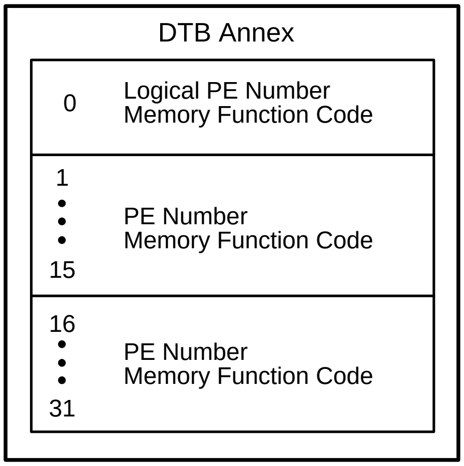


Figure 3-27. DTB Annex

The PE number is virtual when the entry is enabled to be written to by the user or operating system. The user or operating system can write to locations 1 through 15 of the DTB annex when bit 0 of the system control register is set to 1. The user or operating system can write to locations 16 through 31 of the DTB annex when bit 1 of the system control register is set to 1.

The PE number is logical when the entry is not enabled to be written to by the user or operating system. The user or operating system cannot write to locations 1 through 15 of the DTB annex when bit 0 of the system control register is set to 0. The user or operating system cannot write to locations 16 through 31 of the DTB annex when bit 1 of the system control register is set to 0.

The memory function code (bits 12 through 14 of the DTB annex entry) indicates the type of memory function that will be performed during an operation. Refer to Figure 3-28 and Table 3-16.

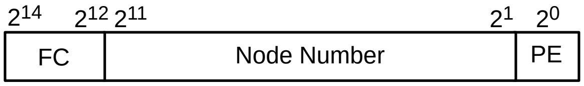


Figure 3-28. DTB Annex Entry

Table 3-16. Memory Function Codes

Bit 14	Bit 13	Bit 12	Function
0	0	0	Write or noncacheable read
0	0	1	Write or noncacheable atomic swap
0	1	0	Fetch-and-increment write or read
0	1	1	Reserved
1	0	0	Write or cached read
1	0	1	Write or cached atomic swap
1	1	0	Write or cached read ahead
1	1	1	Message or cached read

When written to, the AR option receives the DTB annex data from the microprocessor along with a DTB annex index. The DTB annex index consists of 5 bits that the AR option uses to address the locations within the annex. The same index is sent by the microprocessor when a location in the DTB annex is to be read.

After a location in the DTB annex is read, the AR option sends the contents of this location to the AM option. The AM option uses this information to determine the type of reference.

Prefetch Queue

The prefetch queue is a 78-bit x 16 random access memory (RAM) with associated “full” and “no-check” flags for each of the 16 locations. It is implemented using two 18-bit x 32 RAM macros plus additional latch and mux macros to create the full 78-bit (plus flags) RAM memory structure. It is used to temporarily store data when the microprocessor issues a prefetch instruction. The prefetch queue is a first-in, first-out (FIFO) memory that acts as an external memory pipeline. Refer to Figure 3-29.

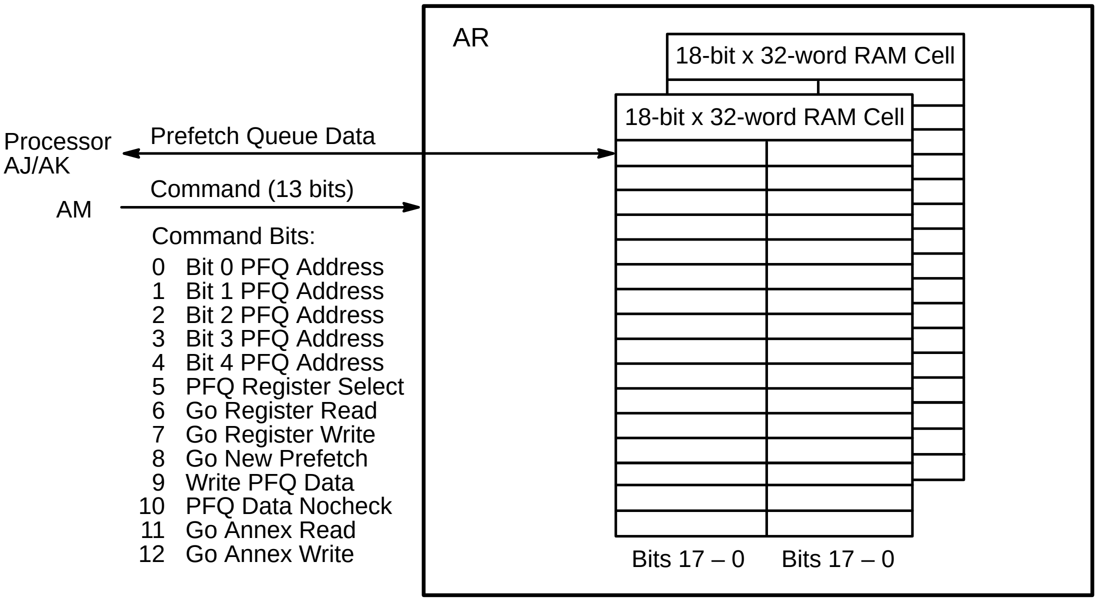


Figure 3-29. Prefetch Queue

For a prefetch instruction, data is read from a remote memory location and written into the prefetch queue. The data will remain in the prefetch queue until the microprocessor reads the data out of the prefetch queue.

Memory-mapped Registers

The AR option contains several memory-mapped registers. Table 3-17 lists these memory-mapped registers along with each register address.

Table 3-17. Memory-mapped Registers in the AR Option

Address	Description	Name
10400000_{16}	Physical PE number register	P_WHOAMI
10404000_{16}	Barrier timing and refresh register	BAR_TMG
10108000_{16}	Barrier synchronization register 0	BSR0
$1090C000_{16}$	Barrier synchronization register 1	BSR1
10410000_{16}	System control register	SCR
10414000_{16}	BLT status register	BLT_SR
10418000_{16}	Barrier mask and interrupt register	BSMI
$1041C000_{16}$	Barrier function register	BSFR
10180000_{16}	Prefetch queue register	PRE_FETCH

Network Interface

The hardware that makes up the network interface consists of four option types: the EA and EB options handle packet output, and the EC and ED options handle packet input. Refer to Figure 3-30.

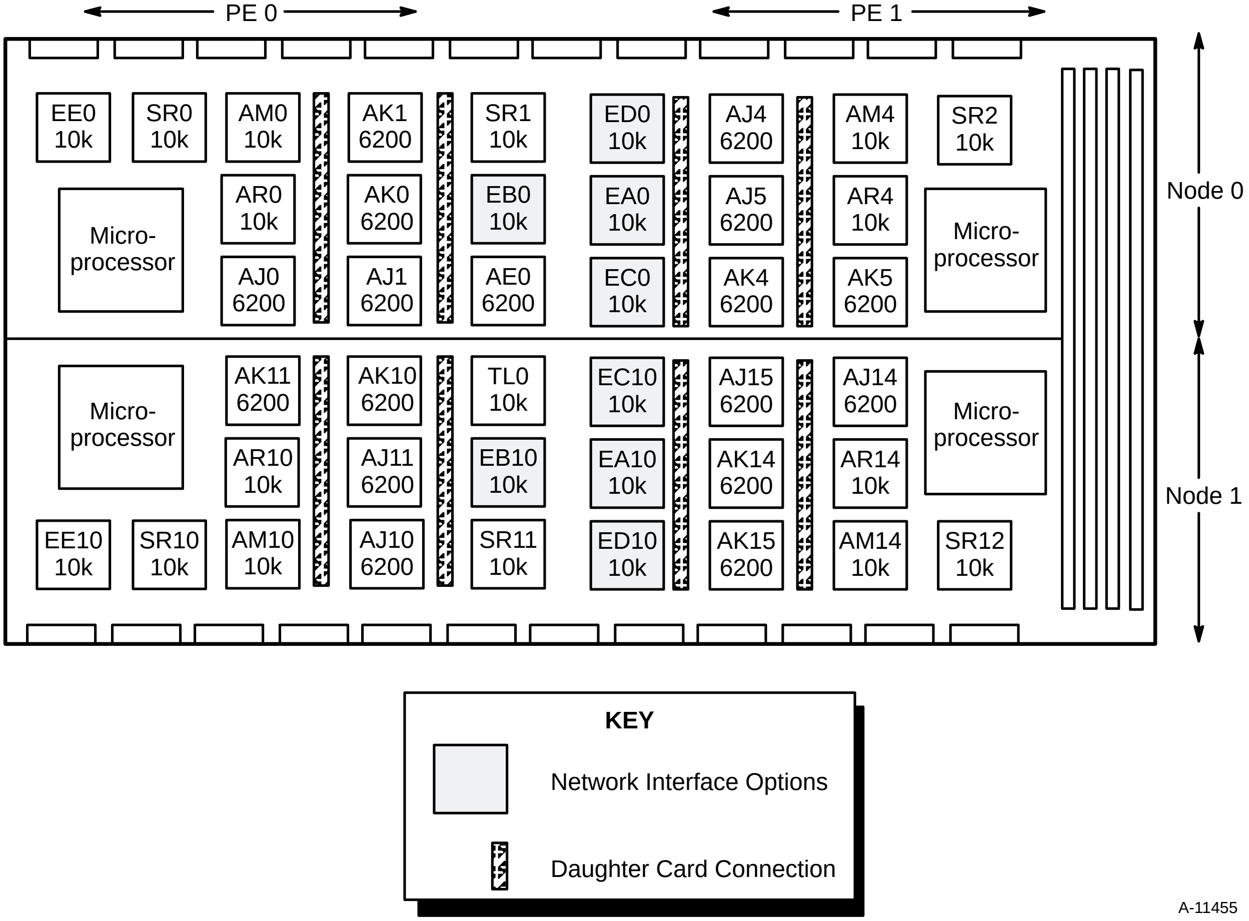


Figure 3-30. PE Network Interface Options

The following subsections describe the network interface options.

EA Option - Network Interface Output Buffers, Packet Assembly, and Parity Generation

The EA option contains logic that buffers request and response information before it is sent outside the node. The EA option also generates parity for this information and assembles it into packet format. Refer to Figure 3-31.

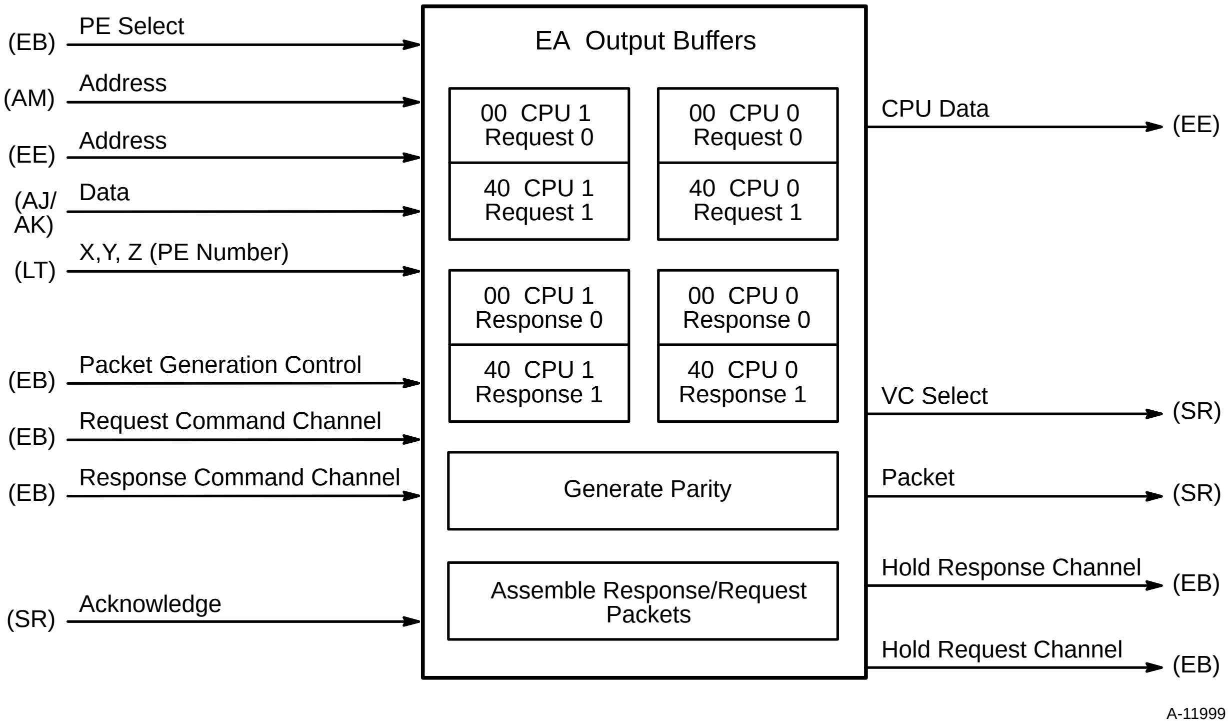


Figure 3-31. EA Option

Buffering Address and Data

The EA option contains eight buffers: two request buffers for PE 0, two request buffers for PE 1, two response buffers for PE 0, and two response buffers for PE 1. The EA option inputs the address from the AM option or the EE option and inputs the data from the AJ and AK options. The EA option writes this address and data into one of the buffers.

The EB option generates the following signals and sends them to the EA option to control the address and data input and the buffer writes.

- CPU Write Active – This signal instructs the EA option to start the address counter, which is used to generate successive buffer addresses. This signal is also used as the Write Enable signal for the buffers.
- CPU Clear Write Address – This signal instructs the EA option to clear the write address counter. The write address counter will remain cleared until the channel is activated again.
- CPU Buffer Select – This signal indicates to the EA option which buffer will be used.
- CPU Write Data Select – This signal enables the EA option to input request or response information from the AM option or the EE option.

Parity Generation

The EA option generates parity for all of the request information, with the exception of the routing tag and the destination PE number. The parity is used to detect any errors that occur while the packet is transferred through the network. The routing tag is not protected by parity because an error that occurs with this packet phit will cause a misroute of the packet. A misroute is detected when the destination node compares the destination phit to its source register.

Parity is generated for the address and data as they are written into the RAM cell or buffer; however, the parity that is generated for the data is not used because the data is protected by SECDED. Parity is used for the address; therefore, the EA option selects the parity bits from a MUX and outputs the parity to the SR option.

Because the command and source phits are not buffered, the EA option contains additional logic that generates parity for these three phits.

Packet Assembly

The EA option assembles the request or response information into a packet under the control of the EB option. The following signals enable the EA option to read the address and data from the buffer, input the other request or response information, generate parity for this information, assemble all of the information into a packet, and send the packet to the SR option.

- Request/Response Load Header – This signal selects the data from the routing tag look-up table. The EA option then sends the routing tag to the SR option.
- Request/Response Go Command – This signal selects the command, destination, and source from the request or response command channel of the EB option. The EA option generates parity for these signals (if parity is enabled) and sends them to the SR option.
- Request/Response Tail Pointer – This signal sets the Go Data signal that the EA option sends the SR option. This signifies that the EA has sent a complete packet to the SR option. Upon receiving this signal, the SR option begins routing the packet through the network.
- Request/Response Read Active – This signal activates the read address counter used to generate the read address that is presented to the buffer.
- Request/Response Buffer Select – Bit 0 of this signal selects either the two request buffers or the two response buffers. Bit 1 selects which of the two buffers will be read.
- Request/Response Enable Parity – This signal enables the EA option to generate parity for the command, destination, and source. It also selects the parity bits for the address and data that are generated by the buffer.

EB Option - Network Interface Output Buffer Control

The EB option contains the logic that controls packet generation. The EB option also contains logic that converts a virtual PE number to a logical PE number. Refer to Figure 3-32. This conversion is described in Section 6, “Addressing.”

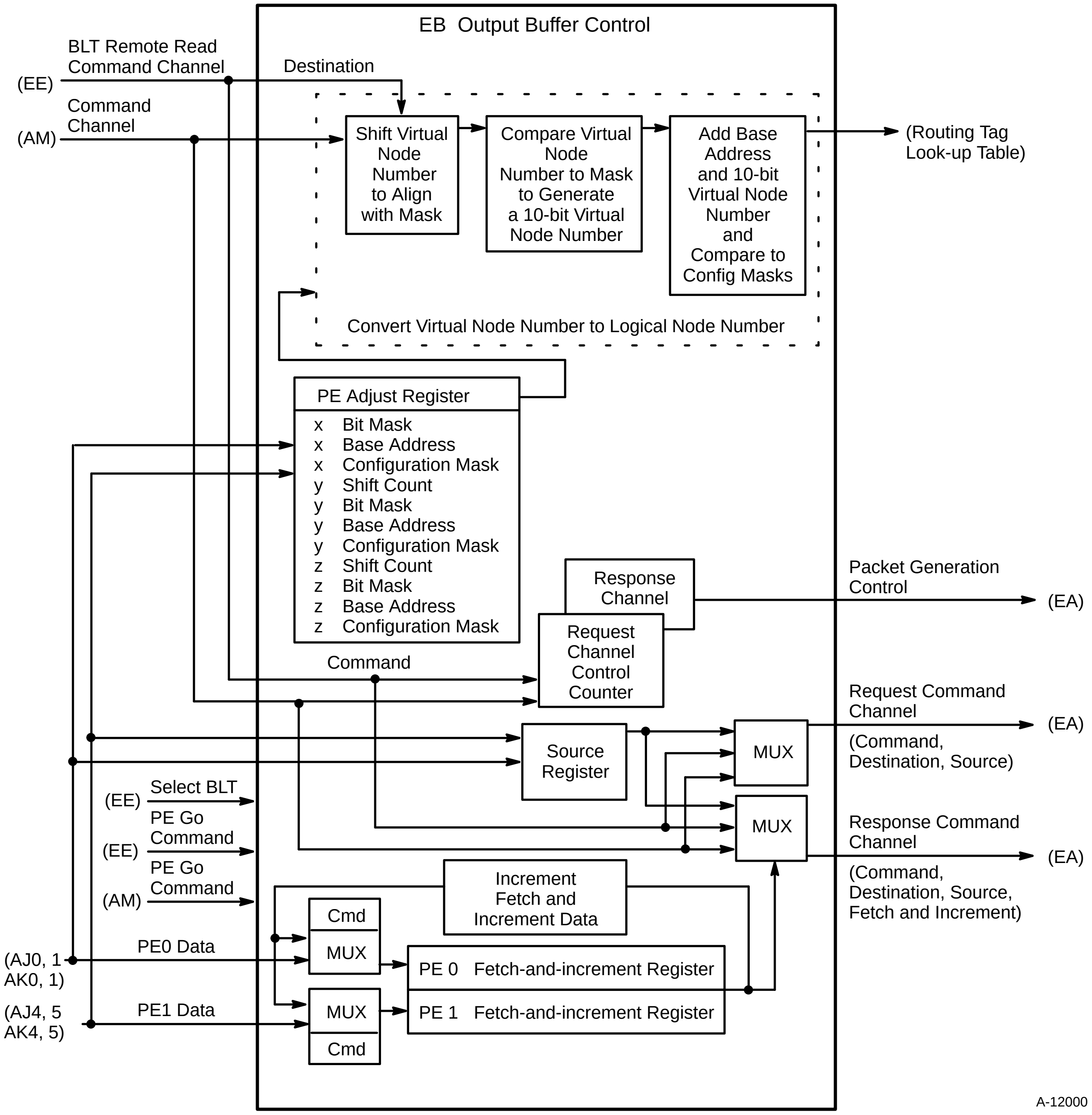

EB Output Buffer Control
BLT Remote Read Command Channel
(EE)
Command Channel
(AM)
Destination
Shift Virtual Node Number to Align with Mask
Compare Virtual Node Number to Mask to Generate a 10-bit Virtual Node Number
Add Base Address and 10-bit Virtual Node Number and Compare to Config Masks
(Routing Tag Look-up Table)
Convert Virtual Node Number to Logical Node Number
PE Adjust Register
x Bit Mask
x Base Address
x Configuration Mask
y Shift Count
y Bit Mask
y Base Address
y Configuration Mask
z Shift Count
z Bit Mask
z Base Address
z Configuration Mask
Response Channel
Request Channel Control Counter
Packet Generation Control
(EA)
Command
Request Command Channel
(EA)
(Command, Destination, Source)
Source Register
MUX
MUX
Response Command Channel
(EA)
(Command, Destination, Source, Fetch and Increment)
(EE)
Select BLT
(EE)
PE Go Command
(AM)
PE Go Command
Increment Fetch and Increment Data
Cmd
MUX
PE 0 Fetch-and-increment Register
PE 1 Fetch-and-increment Register
MUX
Cmd
(AJ0, 1 AK0, 1)
PE0 Data
(AJ4, 5 AK4, 5)
PE1 Data
A-12000

Figure 3-32. EB Option

Parity Generation

The EB option generates parity for the command and source phits of a response packet. The parity is used to detect any errors that occur while the packet is transferred through the network.

Packet Assembly

The EB option controls packet generation by generating signals that instruct the EA option when to input the request or response information, when and where to buffer this information, and when to output the information to the SR options.

The EB option generates the following control signals to control the input and buffering of the address and data:

- CPU Write Active – When the EB option receives the Go Command signal from the AM option or from the EE option (bit 12 of the command phit) and the command channel is not busy, the EB activates the channel by setting the CPU Write Active signal to 1.

- CPU Clear Write Address – The EB option generates this signal after all request information has been buffered.

- CPU Buffer Select – The EB option generates this 2-bit select code to inform the EA option of the buffer that will be used to store the request or response information. Bit 0 of the select code specifies which buffer (0 or 1) will be used for the PE. If the selected buffer is busy, the EB option will toggle the bit 0 to use the other buffer. Bit 1 specifies whether the request or the response buffer will be used.

- CPU Write Data Select – The CPU Write Data Select signal is partially generated by the EB option. The EB option generates the portion of the select command channel signal that tells the EA option to input the command channel data from the AM option. The select command channel signal is activated for the amount of time required to input the command channel information. This time is dependent upon the packet size.

The EB option also controls the input of other request or response information into the EA option, generating parity, and reading the address and data from the buffer. The EB option generates this control by way of a counter. The counter generates different control signals at different increments of the counter. The duration of the counter’s increment depends on the size of the packet. Bits 9 through 11 of the command sent by the AM option or the EE option specify the packet size. The following control signals are generated using this counter:

- Request/Response Load Header
- Request/Response Go Command
- Request/Response Tail Pointer
- Request/Response Read Active
- Request/Response Buffer Select
- Request/Response Enable Parity

For information on how the EA option uses these signals, refer to the “EA Option” subsection.

Memory-mapped Registers

The EB option contains several memory-mapped registers. Table 3-18 lists these memory-mapped registers along with each register address.

Table 3-18. Memory-mapped Registers in the EB Option

Address	Description	Name
$106B0000_{16}$	Network interface PE adjust register	LPE_XLATE
N/A	Fetch-and-increment registers	N/A
$106A0000_{16}$	Network interface source register	X_WHOAMI

EC Option - Network Interface Input Control and Packet Error Checking

The EC option contains control logic used to steer the incoming packet to the correct PE within the node. The EC option also checks this incoming packet for errors. Refer to Figure 3-33.

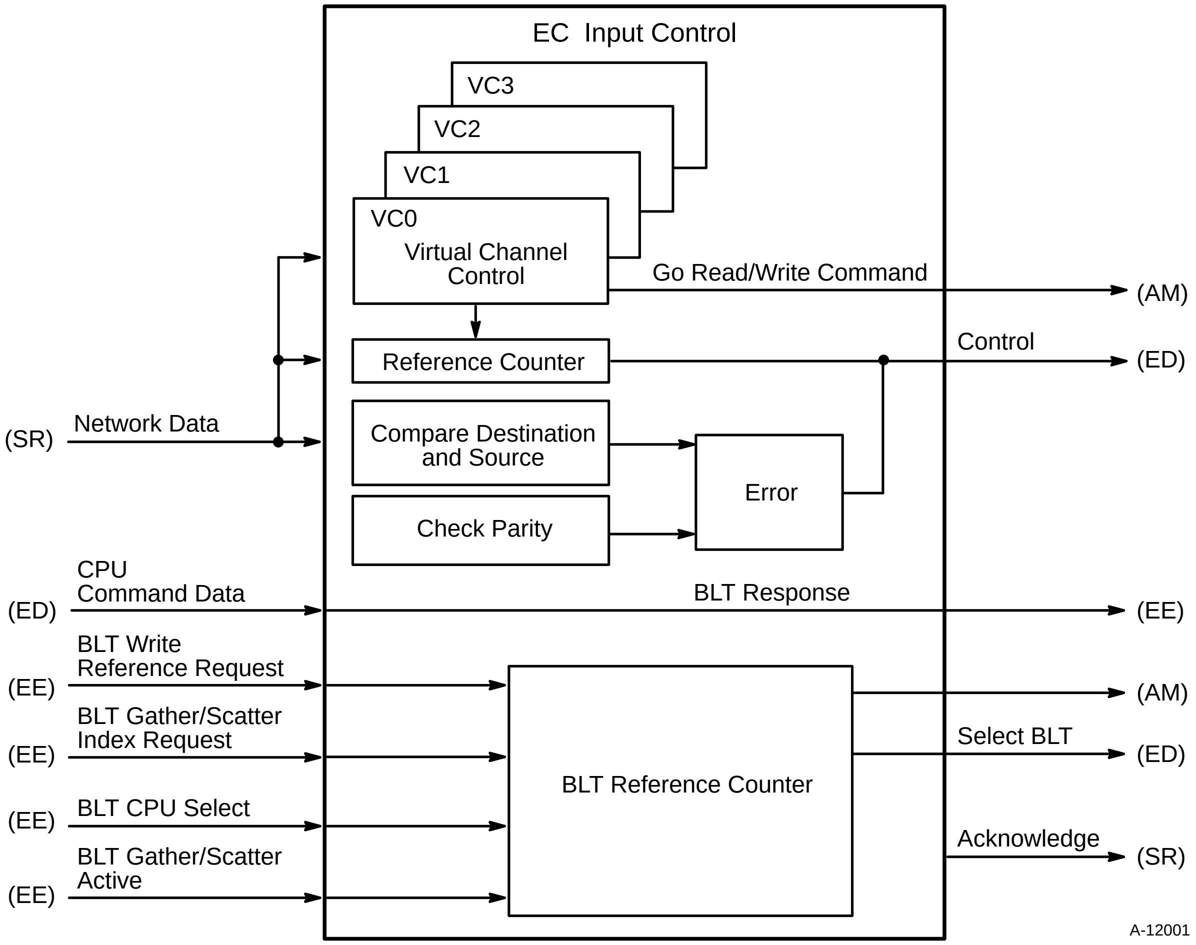


Figure 3-33. EC Option

Incoming Packet Control

The EC option receives the incoming response or request packet from the SR option. The EC option uses the VC Select and Go Data 0 – 1 signals from the SR option to determine when the entire packet has been received. After the entire packet has been buffered, the EC option generates control signals that are used by the ED option to read the packet out of the buffer and steer it to the correct PE within the node. These control signals are described in the following list:

NOTE: Due to potential network conflicts, the EC and ED options have to wait until an entire packet is buffered in the ED option before transferring the packet to the PE.

- CPU 0/1 Read Active – The EC option sets the Read Active signal after a complete packet is entered into the ED option’s buffer. The EC option determines that the packet transfer is complete using the Go Data 0 – 1 signals. The Go Data 0 – 1 signals indicate the tail reference of the packet when both signals are 1.

- CPU 0/1 Clear Read Address – The EC option generates the Clear Read Address signal by way of the reference counter. Once started, the counter increments for the amount of time it takes to read the packet out of the ED option’s buffer. After the complete packet is read out of the buffer, the EC option sets this signal to instruct the ED option to clear the read address counter.

- CPU 0/1 Hold Read Address – The EC option generates the Hold Read Address signal by way of the reference counter and a Write Reference Enable signal sent by the AM option. If the AM option clears the Write Reference Enable signal, the EC option suppresses the external writes by holding the increment of the read address counter.

- CPU 0/1 Select Destination – The EC option generates the Select Destination signal by way of the reference counter.

NOTE: The EC option determines which PE should receive the packet from bit 12 of the command phit.

Error Check

The EC option performs a parity check of all incoming packets to determine whether bits were picked or dropped during the transport through the network. The EC option also checks the packet to make sure the packet was routed to the correct processing element node.

The EC option performs the parity check by generating new parity bits for the command, destination, source, and address phits. The EC option compares the new parity bits to the old parity bits. If the parity bits differ, an error has occurred. The EC option notifies the ED option of the error. The ED option documents the error by setting bit 15 of the command phit.

The EC option checks for misrouted packets by comparing the node number of the destination PE phit from the packet to the node source register. If the two are equal, the packet was routed to the correct destination. If the destination PE number does not match the node source register, the packet was misrouted. When a misroute occurs, the EC option notifies the ED option of the error. The ED option indicates the error by setting bit 15 of the command phit.

Memory-mapped Registers

The EC option contains two memory-mapped registers. Table 3-19 lists these memory-mapped registers along with each register address.

Table 3-19. Memory-mapped Registers in the EC Option

Address	Description	Name
10680000_{16}	Index vector register	BLT_IVR
$106A0000_{16}$	Network interface source register	X_WHOAMI

ED Option - Network Interface Input Buffers

The ED option contains logic that buffers the incoming packets. Refer to Figure 3-34.

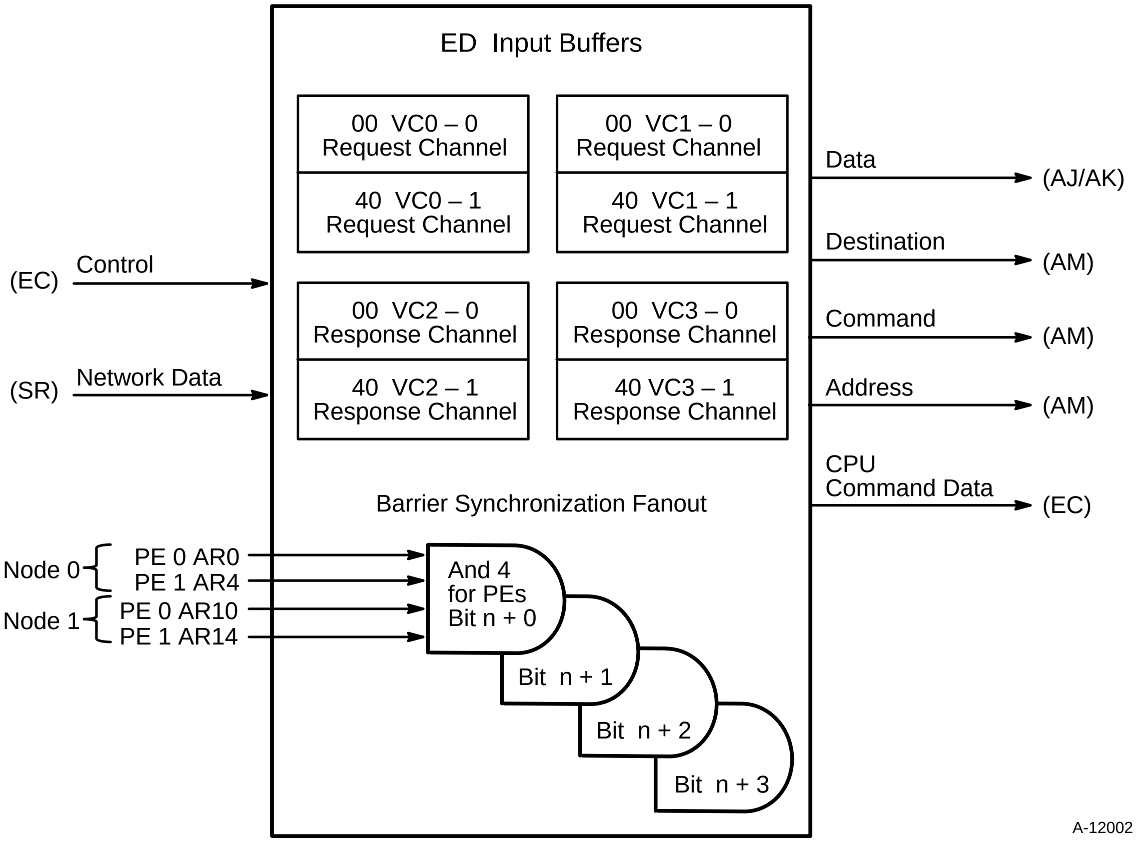

ED Input Buffers
00 VC0 – 0 Request Channel
40 VC0 – 1 Request Channel
00 VC1 – 0 Request Channel
40 VC1 – 1 Request Channel
00 VC2 – 0 Response Channel
40 VC2 – 1 Response Channel
00 VC3 – 0 Response Channel
40 VC3 – 1 Response Channel
(EC) Control
(SR) Network Data
Data (AJ/AK)
Destination (AM)
Command (AM)
Address (AM)
CPU Command Data (EC)
Barrier Synchronization Fanout
Node 0
PE 0 AR0
PE 1 AR4
Node 1
PE 0 AR10
PE 1 AR14
And 4 for PEs Bit n + 0
Bit n + 1
Bit n + 2
Bit n + 3
A-12002

Figure 3-34. ED Option

The ED option contains eight buffers, two for each virtual channel. The ED option uses the VC Select and Go Data 0 – 1 signals it receives from the SR option to determine which buffer should store the packet. The VC Select becomes the Write Enable for the designated buffer. The ED option also receives the VC0 – 3 Write Buffer Select signal from the EC option that becomes part of the write buffer address. The other write buffer address bits are derived from a write address counter. The counter is cleared when the Go Data 0 and Go Data 1 signals both equal 1, signifying the tail reference (end of packet).

The packet remains buffered until the ED option receives control from the EC option specifying which packet should be read out of the buffer and where the packet should be steered (PE 0 or PE 1). The following signals are generated by the EC option and used by the ED option to read the packet out of the buffer and steer it to the correct PE.

- CPU 0/1 Read Active – The ED option uses the Read Active signal to enable the increment of the read address counter.
- CPU 0/1 Clear Read Address – The ED option uses this signal to clear the read address counter. The counter will remain cleared until the read is activated again.
- CPU 0/1 Hold Read Address – The ED option uses this signal to stop and hold the increment of the read address counter.
- CPU 0/1 Select Destination – The ED option uses this signal to enable or disable the read address counter. If the Select Destination signal is a 1, the read address counter is disabled for that PE.

When the packet is misrouted or contains a network parity error, the EC option instructs the ED option to create an error message from the corrupted packet. This allows the entire packet to be written into the message queue so that it can be analyzed to determine what caused the error.

NOTE: Parity is regenerated for the packet information as it is written into the RAM cell or buffer. The ED option generates new parity for this information when the information is read out of the buffer. The ED option compares the parity bits that were generated when the information was written into the buffer to these new parity bits to verify the information was not corrupted while it was stored in the buffer.

When the information that is read out of the buffer is corrupted, the ED option signals the AM option that a parity error occurred. The AM option sets bit 0 (network buffer parity error) of the system status register and signals the AR option to set the error interrupt.

When the microprocessor acknowledges the error interrupt, the microprocessor must read the system status register to determine the cause of the error.

4 I/O GATEWAY

All input and output communication between the CRAY T3D system and the host system passes through the I/O gateways. There are two types of I/O gateways, the master I/O gateway and the slave I/O gateway. Each I/O gateway (master or slave) contains an input node, an output node, and LOSP circuitry (refer to Figure 4-1).

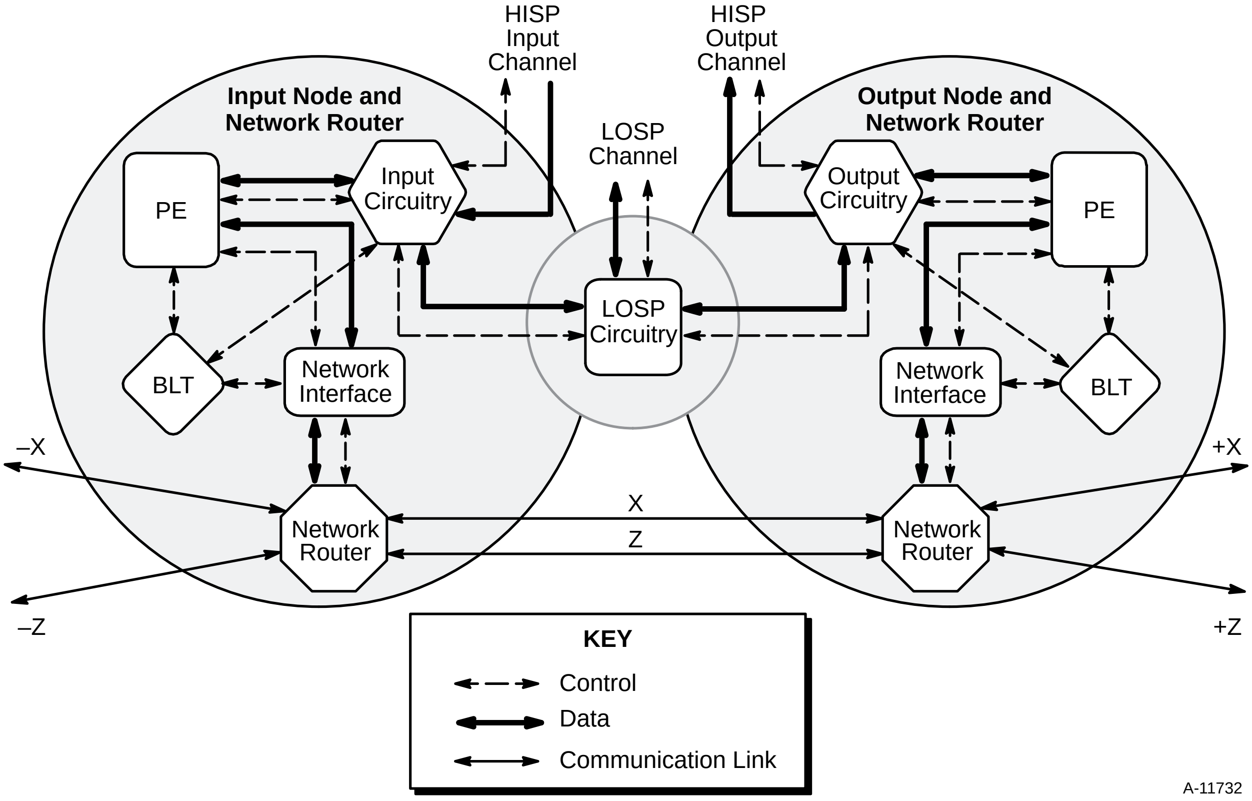


Figure 4-1. I/O Gateway

This section describes the function and hardware of the master and slave I/O gateway components.

Functional Description

As stated previously, there are two types of I/O gateways: the master I/O gateway and the slave I/O gateway. The I/O gateways are connected in the following cable configurations.

LOSP Channel

LOSP channels transfer request and response information between the CRAY T3D system and the host system or an input/output cluster (IOC). Either the CRAY T3D system or the host system (or IOC) can initiate a transfer of information over the LOSP channel.

Each LOSP channel is actually a pair of unidirectional channels. Figure 4-2 shows the signals used in a LOSP channel.

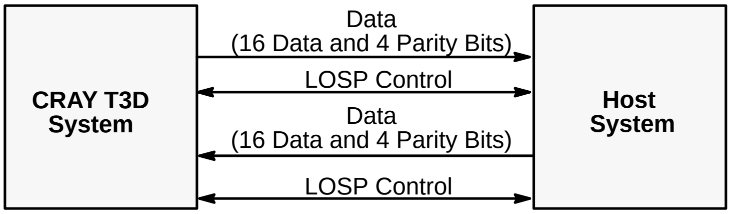

CRAY T3D System
Data (16 Data and 4 Parity Bits)
LOSP Control
Data (16 Data and 4 Parity Bits)
LOSP Control
Host System

Figure 4-2. LOSP Channel Signals

LOSP data is transferred over the LOSP channel in 16-bit parcels. The data contains information used to control the HISP channel. The 16-bits of data are protected by 4 parity bits that are used to check the data for errors.

The data transfer rate of a LOSP channel is 6 Mbytes/s in each direction. The LOSP channel uses control signals that indicate when data is on the channel, when data is received, and when a data transfer is finished.

When information transfers over the LOSP channel from the host system (or IOC) to the CRAY T3D system, the most significant bit of the first parcel transferred directs the information to the appropriate node. If this bit is 0, the information is for the output node. If this bit is 1, the information is for the input node.

When information transfers over the LOSP channel from the CRAY T3D system to the host system (or IOC), the input node and output node share the LOSP channel. The first node to request a transfer over the LOSP channel controls the channel until a disconnect is sent.

HISP Channel

HISP channels transfer system data between the CRAY T3D system and the host system (or IOC). The HISP channel connects two components: a master and a slave. The master controls the HISP channel by providing address information to the slave.

The data transfer rate of a HISP channel is 200 Mbytes/s in each direction; however, by modifying parameters in the I/O gateway memory-mapped registers, software may change the HISP channel transfer rate to 100 Mbytes/s. This enables the CRAY T3D system to operate with a host system that uses either 100 Mbytes/s or 200 Mbytes/s HISP channel protocol. Figure 4-3 shows the signals used in a HISP channel.

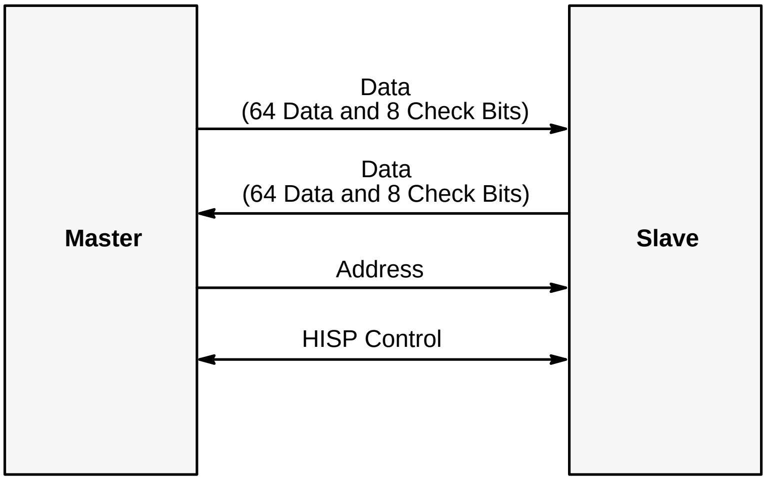


Figure 4-3. HISP Channel Signals

System data transfers between the master and slave in 64-bit words. The 64-bits of data are protected by 8 check bits used to check the data for errors and correct any single-bit errors.

Address and block length information is sent from the master to the slave. The address contains information on where data will be stored or read in the slave’s memory. The block length indicates the total number of words that will be transferred.

HISP protocol uses control signals that clear the HISP channel, control when a data transfer starts, and indicate when the last 64-bit word of data is transferring over the HISP channel. Error signals are also sent to indicate whether a data error occurred during the transfer.

Input Node

The input node receives data from the HISP channel, creates packets, and sends the packets to the processing element nodes in the CRAY T3D system using the interconnect network. The input node is composed of a network interface, a block transfer engine, a PE, and HISP input circuitry (refer to Figure 4-4).

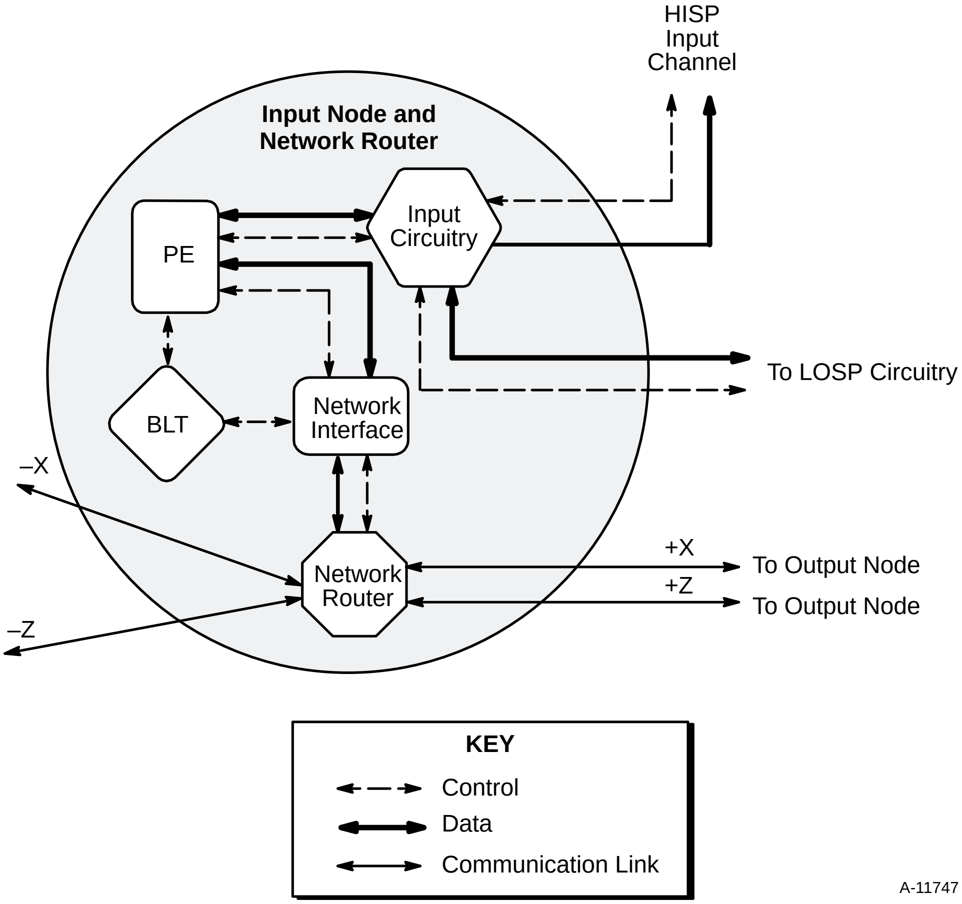


Figure 4-4. Input Node

Processing Element

The PE in the input node is designed to interface with the HISP input circuitry. Because of this characteristic, the PE in the input node does not contain the circuitry to perform all of the operations that a PE in a processing element node performs. Instead, the circuitry is replaced with circuitry that interfaces with the HISP input circuitry.

The PE in the input node of an I/O gateway does not perform the following functions and operations:

- Incoming or outgoing atomic swap operations
- Data prefetch operations
- Read-ahead operations
- Data cache-line invalidation
- Virtual PE numbers and associated virtual range check

More information on these functions and operations is provided in Section 6, “Addressing,” and in Section 7, “Operations.”

In addition, the PE in the input node contains half of the local memory that a PE in a processing element node contains. Because of this, the data and CAS 0 – 3 signals are distributed differently to the memory chips on the I/O module (IOM) daughter cards (refer to Figure 4-5). On the IOM daughter cards, there are 2.5 memory chips assigned to each individual CAS signal. For example, DRAM 0, DRAM 1, and half of DRAM 2 are enabled when CAS 0 is asserted. When CAS 0 asserts, data bits 0 through 22 of halfword 0 and data bits 0 through 15 of halfword 2 are read from or written to these three memory chips.

Network Interface

The network interface in the input node is identical to the network interface in the processing element node except that the network interface receives input from only one PE and the BLT. For more information on the network interface components, refer to “Network Interface” in Section 3.

Network Router

The network routers in the input node are identical to the network routers in the processing element node except there is not a network router for the Y dimension. This is because there are not enough connector pins to accommodate all three dimension network-router connections and the HISP connections. The absence of a Y dimension router does not prohibit the input node from accessing any node in the system. The data is sent out to the network using the X dimension and received using the Z dimension.

Block Transfer Engine

The BLT used in the input node of the I/O gateway is identical to the BLT used in the processing element node except that it receives information from only one PE. For more information on the BLT components, refer to Section 8, “Block Transfer Engine.”

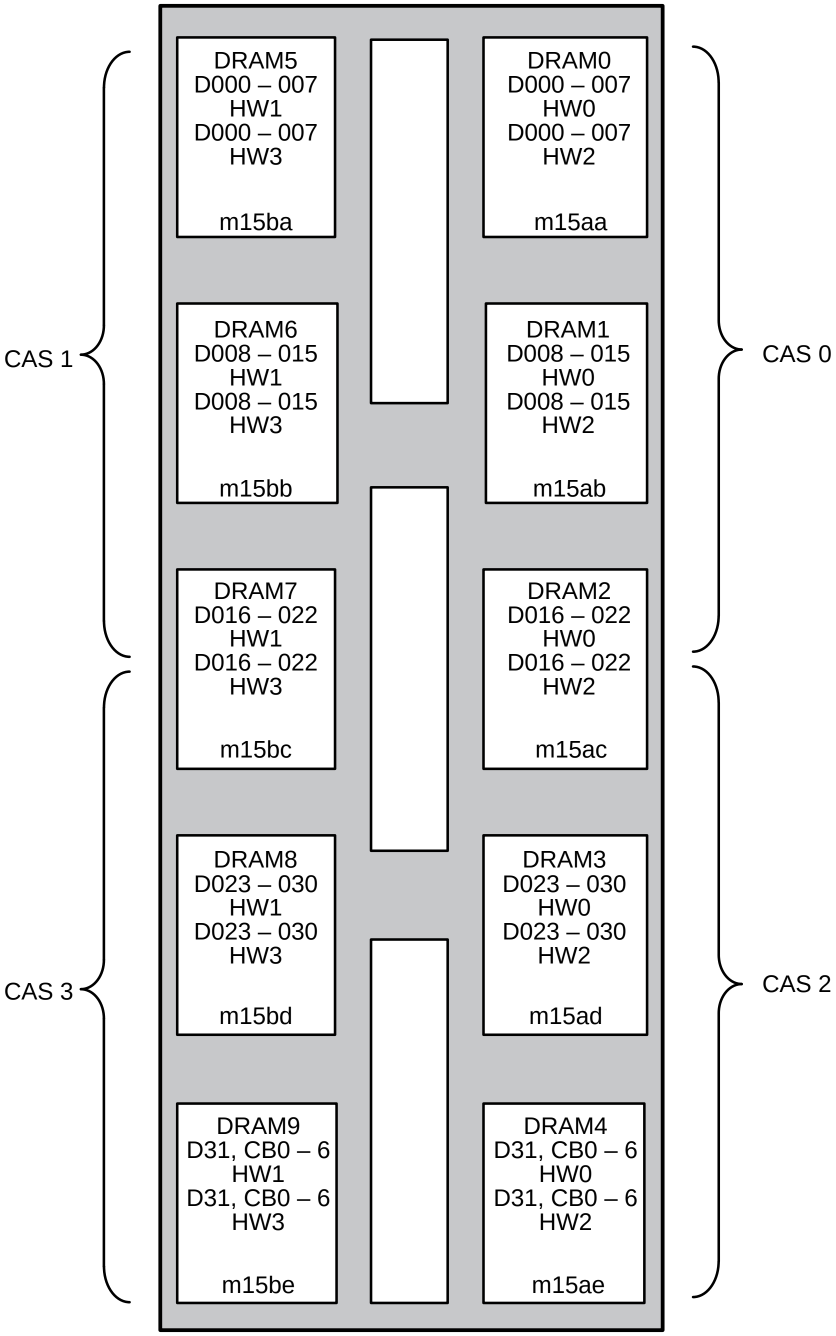
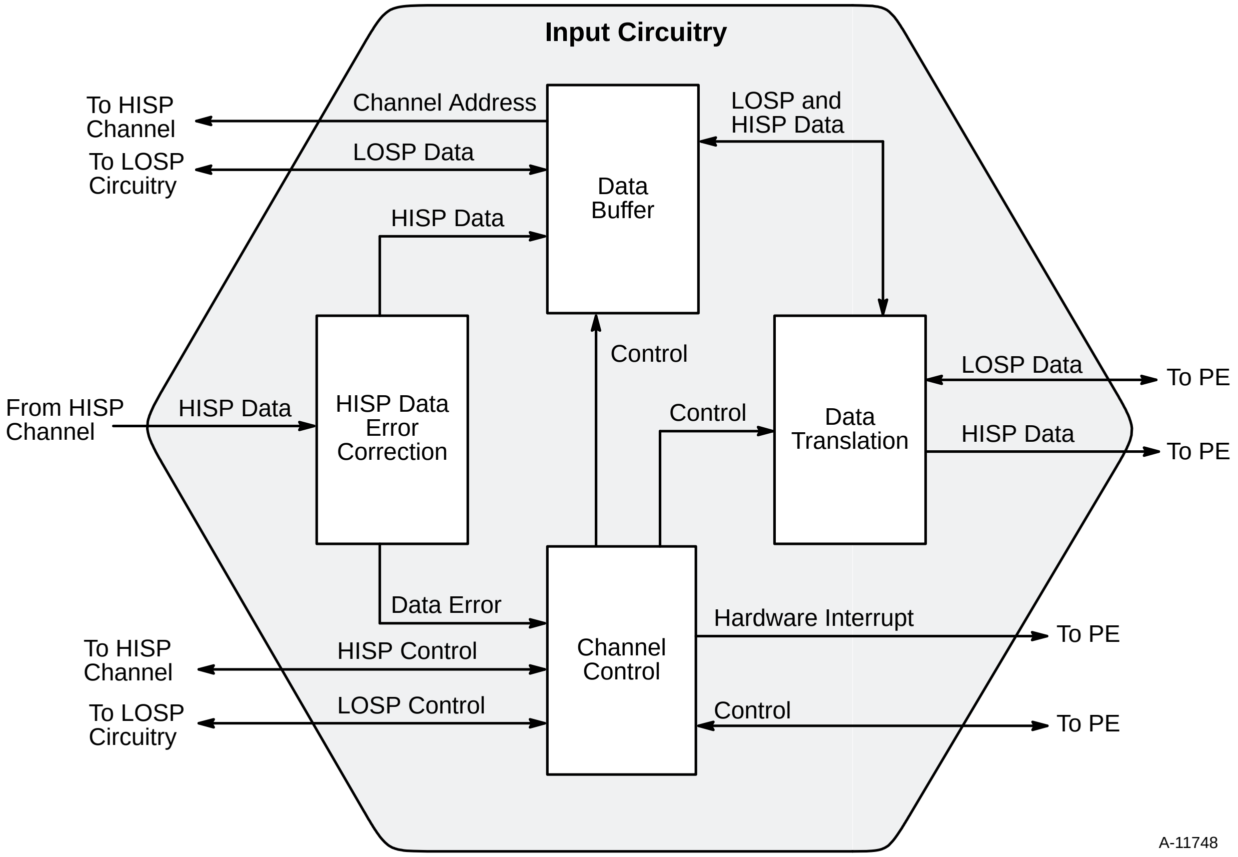


Figure 4-5. CAS 0 – 3 Designation on the IOM Daughter Card

Input Circuitry

The input circuitry receives system data and control directly from the HISP channel. The input circuitry contains HISP data error correction circuitry, a data buffer, a data translator, and channel control (refer to Figure 4-6). The following paragraphs describe each of these components.



On the master I/O gateway, the input circuitry sends the channel address to the HISP channel (as shown in this diagram). On the slave I/O gateway, the input circuitry receives the channel address from the HISP channel.

Figure 4-6. Input Circuitry

The error correction circuitry performs single error correction, double error detection (SECDED) on incoming HISP data. While receiving each 64-bit word of data and the 8 check bits, the error correction circuitry generates a new set of check bits for the data. If the new check bits do not match the original check bits, 1 or more of the data bits or original check bits changed value when transferred between the CRAY T3D system and the host system.

If only 1 bit changed value, the error correction circuitry changes the value of the bit back to the original value and signals the channel control circuitry that a single-bit error occurred. If 2 bits changed value, the error correction circuitry cannot correct the bits, but it does signal the channel control circuitry that a double-bit error occurred. If more than 2 bits changed values, the error correction circuitry attempts to correct the bits but will not be successful. After checking the HISP data for errors, the error correction circuitry sends the data to the data buffer.

The data buffer temporarily stores data from the HISP channel until it is transferred to the PE. The size of the buffer is 128 64-bit words of data.

The data buffer also transfers LOSP information between the LOSP circuitry and the PE. LOSP information is used to request and respond to transfers between the CRAY T3D system and the host system.

Data translation circuitry converts emitter coupled logic (ECL) signals from the input node circuitry into transistor transistor logic (TTL) signals that the PE uses. The data translation circuitry also converts TTL signals back to ECL signals.

The data translation circuitry generates new check bits that the microprocessor will use while performing SECDED. The data translation circuitry generates 7 check bits for each group of 32 bits that passes through this circuitry. After generating the check bits, the data translation circuitry sends the data and check bits to the PE.

The data translation circuitry also checks the LOSP data from the PE for SECDED errors. While receiving a group of 32 data bits and 7 check bits, the data translation circuitry generates a new set of check bits for the data. If the new check bits do not match the original check bits, 1 or more of the data bits or original check bits changed value when transferred from the PE to the input circuitry.

The channel control circuitry provides the control signals used in the LOSP and HISP channels. The channel control circuitry receives control signals from the PE and translates the signals into the host system channel control signals.

The channel control circuitry also controls the flow of error information through the input circuitry. It receives single-bit and double-bit error signals from the error correction circuitry and receives addressing errors from the data buffer. After receiving the error signals, the channel control circuitry sends a hardware interrupt to the microprocessor. The microprocessor then examines the status of the error.

Output Node

The output node receives system data from the processing element nodes through the interconnect network and sends the data to the host system over the HISP channel. The output node is composed of a network interface, a block transfer engine, a PE, and HISP output circuitry (refer to Figure 4-7).

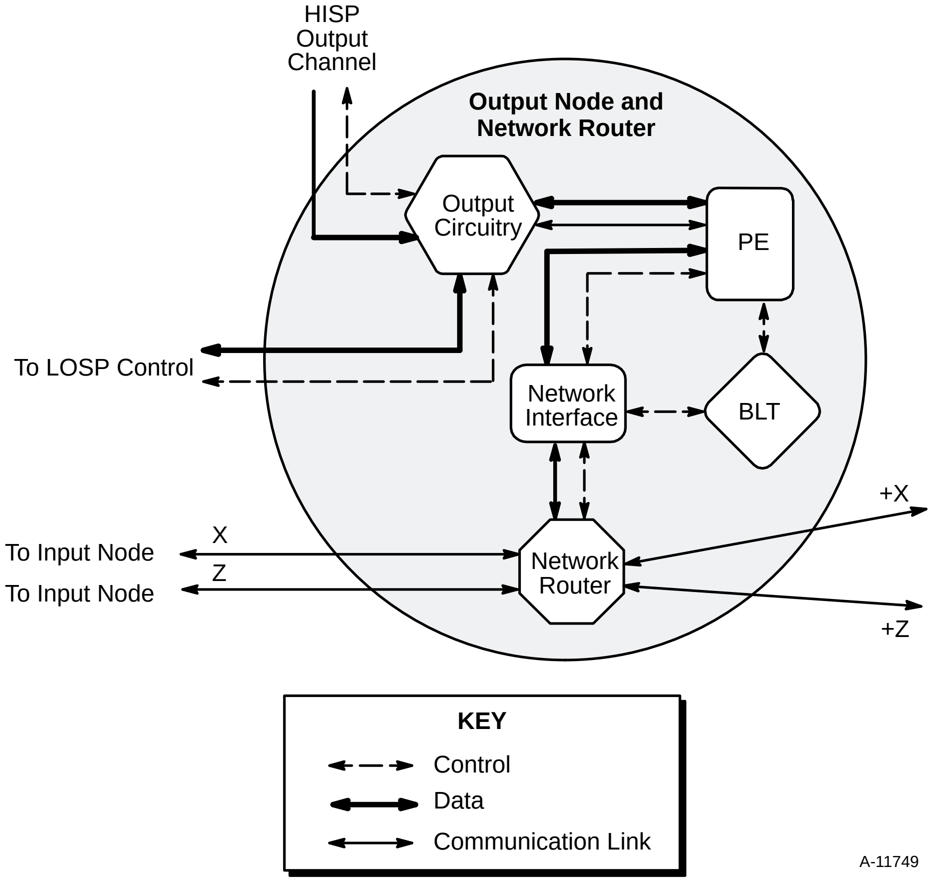


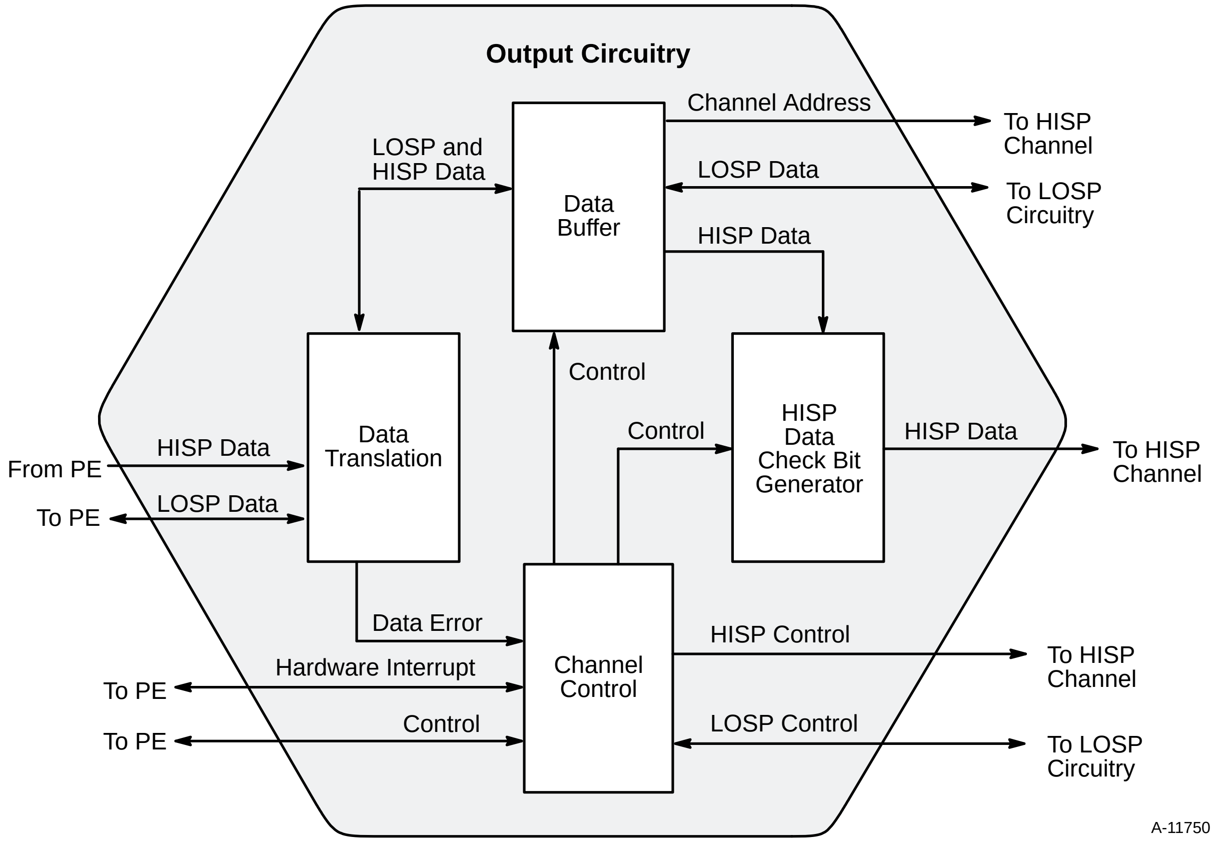
Figure 4-7. Output Node

Processing Element, Network Interface, Network Router, and Block Transfer Engine

Except for the HISP output circuitry, the output node contains the same components as the input node. These components function the same as the input node components.

Output Circuitry

The output circuitry transmits information to the HISP channel and transfers information over the LOSP channel. The output circuitry contains a data translator, a data buffer, a HISP data check-bit generator, and channel control (refer to Figure 4-8).



On the master I/O gateway, the output circuitry sends the channel address to the HISP channel (as shown in this diagram). On the slave I/O gateway, the output circuitry receives the channel address from the HISP channel.

Figure 4-8. Output Circuitry

Data translation circuitry converts TTL signals from the PE into ECL signals that the output circuitry uses. The data translation circuitry also converts signals back to TTL signals.

The data translation circuitry performs SECDED on incoming system data from the PE. While receiving a group of 32 data bits and 7 check bits, the data translation circuitry generates a new set of check bits for the data. If the new check bits do not match the original check bits, 1 or more of the data bits or original check bits changed value when transferred from the PE to the output circuitry.

If only 1 bit changed value, the data translation circuitry changes the value of the bit back to the original value and signals the channel control circuitry that a single-bit error occurred. If 2 bits changed value, the data translation circuitry cannot correct the bits, but it does signal the channel control circuitry that a double-bit error occurred. If more than 2 bits changed value, the error correction circuitry attempts to correct the bits but will not be successful. After checking the system data for errors, the data translation circuitry sends the data to the data buffer.

The data translation circuitry also generates new check bits (for LOSP data) that the microprocessor will use while performing SECDED. The data translation circuitry generates 7 check bits for each group of 32 bits that passes through this circuitry. After generating the check bits, the data translation circuitry sends the data and check bits to the PE.

The data buffer temporarily stores data from the PE until it is transferred to the HISP channel. The size of the buffer is 128 64-bit words of data.

The data buffer also buffers LOSP information between the LOSP circuitry and the interconnect network. LOSP information is used to request and respond to transfers between the CRAY T3D system and the host system.

The HISP data check bit generator generates check bits that are used by the host system while performing SECDED. As each 64-bit word of data passes through the check bit generator circuitry, the circuitry generates 8 check bits for the data. After generating the check bits, the check bit generator circuitry sends the data and check bits to the host system over the HISP channel.

The channel control circuitry provides the control signals used in the LOSP and HISP channels. The channel control circuitry receives control signals from the PE and translates the signals into the host system channel control signals.

The channel control circuitry also controls the flow of error information through the output circuitry. It receives single-bit and double-bit error signals from the data translation circuitry and receives addressing errors from the data buffer. After receiving the error signals, the channel control circuitry sends a hardware interrupt to the microprocessor. The microprocessor then examines the status of the error.

LOSP Circuitry

The LOSP control circuitry transfers request and response information over the LOSP channel that connects the host system and the CRAY T3D system (refer to Figure 4-9). The LOSP control circuitry receives data from the LOSP channel. The LOSP control circuitry buffers this data in an input buffer. When the data is read from the input buffer, the LOSP control circuitry checks the data for parity errors.

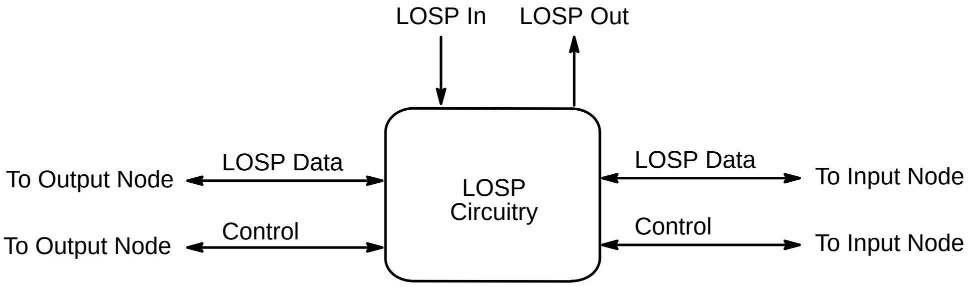


Figure 4-9. LOSP Circuitry

The LOSP circuitry contains an input buffer that stores 32 words. Although the input buffer contains 32 locations, the LOSP circuitry can input 33 words before interrupting the microprocessor in the input node or output node. The one additional word is buffered in circuitry within the input node or output node, depending upon the destination of the LOSP data.

The LOSP control circuitry determines which node the data is destined for by interpreting the most significant bit of the first word of the transfer. (The first word of the transfer is defined as the first word following a disconnect.) When the most significant bit is set to 1, the data is destined for the input node. When the most significant bit is set to 0, the data is destined for the output node.

The LOSP control circuitry checks the data for parity errors when the data is read out of the buffer. The LOSP circuitry checks for parity errors by generating a new set of parity bits. If the new set of parity bits does not match the original parity bits, one or more of the data or parity bits changed value while the data was transferring over the LOSP channel. When a parity error is detected, the LOSP circuitry reports the error to the node that is reading the data. After checking the LOSP data for errors, the LOSP circuitry sends the data to the input node or the output node.

NOTE: When there is a parity error, the data is sent to the input or output node unchanged.

The LOSP control circuitry also receives data from the input node and output node. The LOSP control circuitry buffers this data in an output buffer that stores up to 32 words. When the data is read from the output buffer, the LOSP control circuitry generates parity for the data.

The LOSP control circuitry generates 4 parity bits for each 16 bits of data it receives from the output buffer of the input node or output node. The host system uses these parity bits to check the data as it transfers over the LOSP channel. After generating the parity bits, the LOSP circuitry sends the LOSP data and parity bits over the LOSP-out channel to the host system.

Hardware Description

Each I/O gateway resides on one printed circuit board in the CRAY T3D system cabinet. Refer to Figure 4-10. The integrated circuits to the left of the bold line contain the components of the input node, with the exception of the AE option. The integrated circuits to the right of the line contain the components of the output node, with the exception of the IM and TL options. The IM option is the LOSP circuitry that is shared by the input and output nodes. The AE option contains DRAM timing circuitry, and the TL option contains clock fanout circuitry that is shared by the input and output nodes.

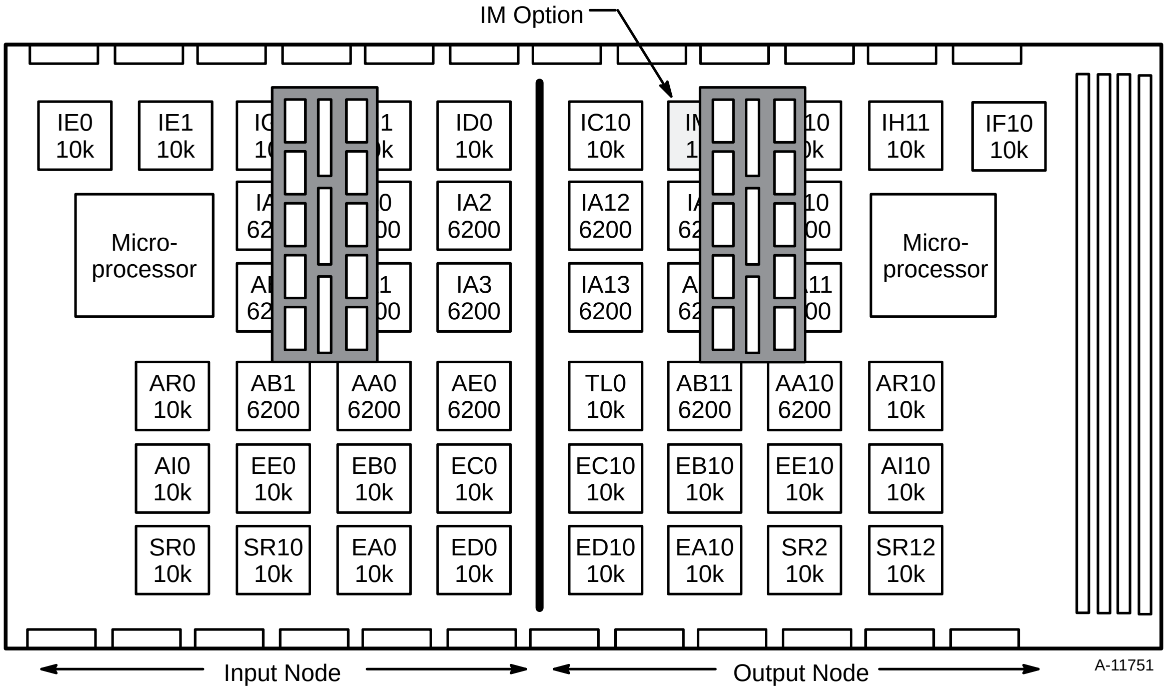


Figure 4-10. Master I/O Gateway

The following subsections describe the components of the input node, the output node, and the LOSP circuitry.

Input Node

The input node contains input circuitry, processing element node circuitry, and interconnect network circuitry. The input circuitry consists of options that translate TTL and ECL signals, control the LOSP and HISP channels, buffer the HISP data, and check for single and multiple bit errors. The processing element node logic consists of options that initiate the I/O transfer upon request, transfer blocks of data, and create packets. The network interconnect circuitry transfers the packets through the network.

Input Circuitry

The master input circuitry is made up of four option types: the IA option, the ID option, the IE option, and the IG option. Refer to Figure 4-11. The slave input circuitry is also made up of four option types. The four options are the IA option, the IE option, the IJ option, and the IT option.

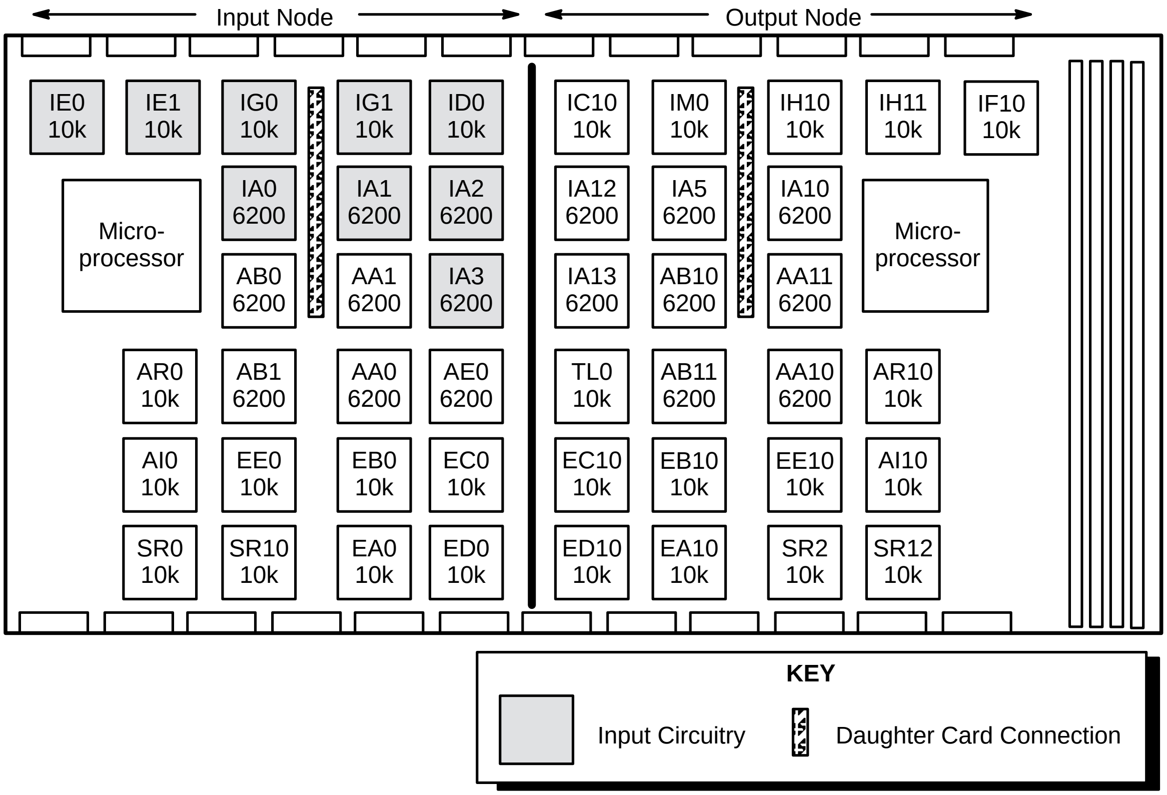


Figure 4-11. Master I/O Gateway, Input Options

IA Option - ECL to TTL Translation and HISP Data ECC Generation

The IA option generates error correction code (ECC) for the HISP data. The IA option also translates the ECL signals from the HISP channel to TTL signals before sending the data to the PE. Refer to the shaded area of Figure 4-12.

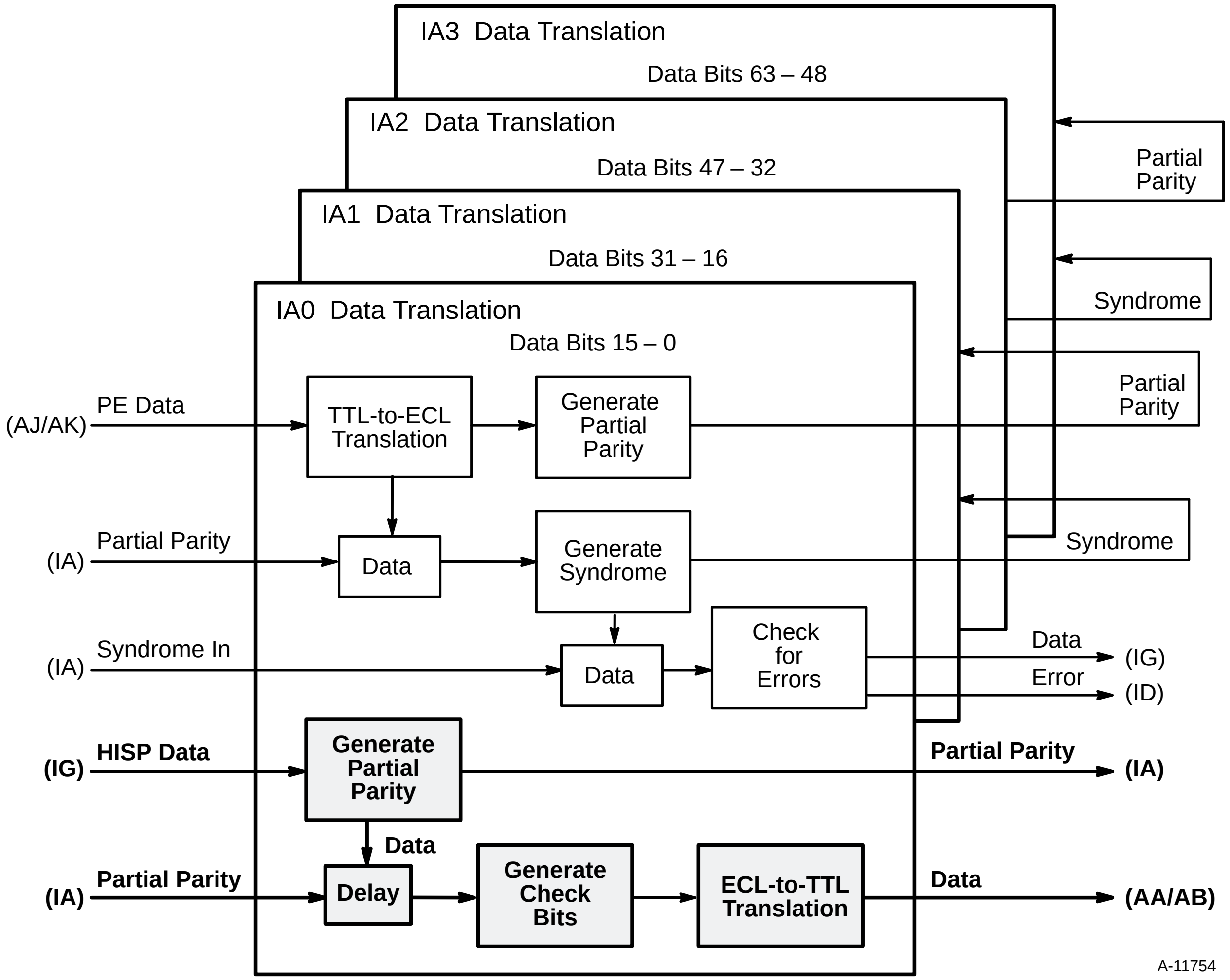


Figure 4-12. IA Option

The IA options in the input node receive HISP data from the IG option and generate parity for this data. Each IA option generates parity for the data bits it handles and communicates this partial parity to the other IA options within the input node (refer again to Figure 4-12). The IA options compare the partial parity they receive from the other IA options to the partial parity they generated for their data. This comparison yields check bits that are appended to the data as the HISP data is transferred to the PE.

After generating the check bits, the IA options convert the HISP data from ECL signals to TTL signals. This is necessary because the inputs of the AA and AB options within the PE that receive the HISP data from the IA options can only receive TTL signals. The AA and AB options will convert the HISP data back to ECL signals before the data is sent to the requesting PE.

ID Option - HISP Input Channel Control

The ID option controls the HISP and LOSP channels during the input of HISP data. Refer to Figure 4-13. This control consists of generating HISP and LOSP channel control, HISP and LOSP channel protocol, and ram-on-array (ROA) control.

The ID option generates the HISP and LOSP control using the address, control, and data it receives from the PE within the input node. The address and control are used to input other control information and to steer the data that the ID option receives from the microprocessor to an I/O memory-mapped register.

To input other control information and steer the data to the I/O memory-mapped registers, the ID option uses the control signal, Load Address Rank 0, to latch in the address. This address is used to generate 14 different decode signals, as shown in Figure 4-13. The following text describes the function of each decode signal.

The Decode 0 signal activates the LOSP and HISP configuration control. The configuration control enables the LOSP and HISP configuration information to be latched into the configuration memory-mapped registers.

The LOSP configuration information is used to set up the width and the delay count of the LOSP data resume pulse. The HISP configuration information controls the timing of the HISP address and data transfers. More information is provided on the LOSP and HISP configuration memory-mapped registers in Section 11, “LOSP and HISP Channels.”

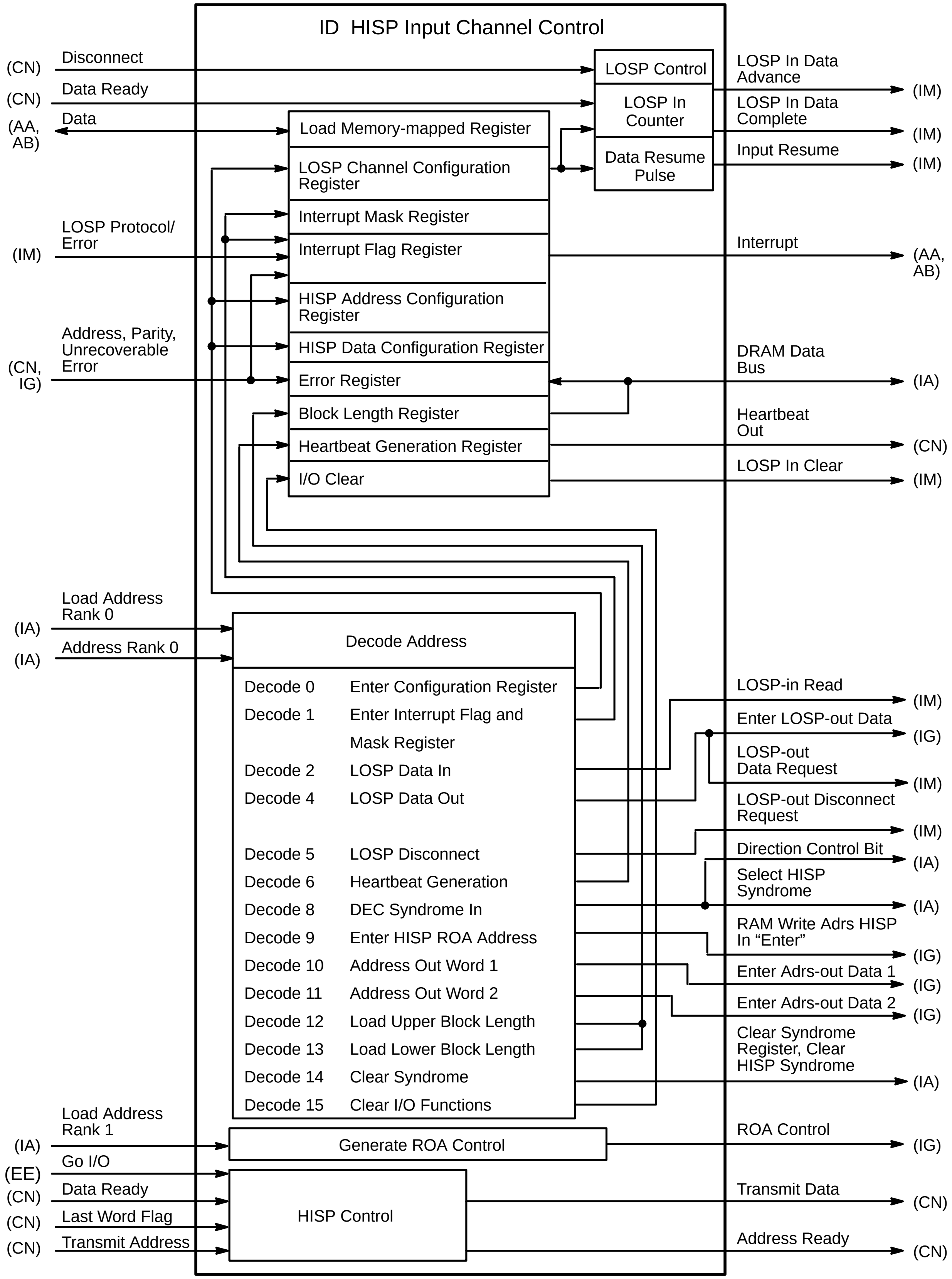


Figure 4-13. ID Option

The Decode 1 signal selects either the error information from the error register or the interrupt information from the interrupt flag register to be sent to the microprocessor. The error information consists of various error flags from the channels. The interrupt information consists of flags that indicate what caused an interrupt from the I/O channel. More information is provided on the error memory-mapped register and the interrupt flag memory-mapped register in Section 11, “LOSP and HISP Channels.”

The Decode 1 signal also enables the interrupt mask information to be latched into the interrupt mask register. The interrupt mask information masks out sources of interrupts. The bit layout for the interrupt mask register is the same as the interrupt flag register.

The Decode 2 signal activates the control signals LOSP In Data Read and Direction Control. The LOSP In Data Read signal (R61) is sent to the IM option (LOSP circuitry). This signal alerts the IM option that there is LOSP data to input. The IM option then controls the input of this data. The Direction Control signal (R2–3) is sent to the IA options. This signal controls the bidirectional pins of the IA option. When this signal is a 1, the IA option sends data to the PE. When it is a 0, the IA option inputs data from the PE.

The Decode 4 signal activates the control signals LOSP Data Out (R20) and LOSP Out Data Request (R50). The LOSP Data Out signal is sent to the IG options and the LOSP Out Data Request is sent to the IM option. The IG option uses the LOSP Data Out signal to input the PE data. The IM option uses the LOSP Out Data Request signal to generate LOSP Out Advance signals. The Advance signals are sent to the IG option so that the IG option can latch in the LOSP data from the PE.

The Decode 5 signal activates the control signal LOSP Out Disconnect Request (R51). The LOSP Out Disconnect Request signal is sent to the IM option. The IM option uses this signal to generate another LOSP Out Disconnect signal that is sent to the IC option. The IC option sends the disconnect signal to the external device.

The Decode 6 signal selects the Heartbeat signal from the heartbeat generation register. The Heartbeat signal is fanned out to the PEs within the system. The heartbeat signal can be used to establish the global time of day among the PEs, to detect a malfunctioning PE, to trigger a selective reset, or to trigger a global reset.

The Decode 8 signal, like Decode 2, activates the Direction Control signal that is sent to the IA option. This signal also activates the control signals Read DEC and HISP Syndrome Register (R10–11). The Read DEC and

HISP Syndrome signals are also sent to the IA option. The IA option uses the DEC and HISP Syndrome signals to select the DEC and HISP syndrome information that will be sent to the PE.

The Decode 9 signal activates the Enter ROA Address signal (R25). The Enter ROA Address signal is sent to the IG option. The IG option uses this signal to enable the input of the ROA address from the PE.

The Decode 10 signal activates the Enter HISP Address Word 1 signal (R21), which is then sent to the IG option. The IG option uses the Enter HISP Address Word 1 signal to enable the input of the HISP address.

The Decode 11 signal activates the Enter HISP Address Word 2 signal (R22). The Enter HISP Address Word 2 signal is sent to the IG option. The IG option uses this signal to enable the input of the upper HISP address.

The Decode 12 signal activates the Load Upper Block Length signal. The Load Upper Block Length signal selects the upper block length information from the input data and enables this information to be latched into the block length register. The upper block length information along with the lower block length information indicates how many blocks of data will be transferred.

The Decode 13 signal activates the Load Lower Block Length signal. The Load Lower Block Length signal selects the lower block length information from the input data and enables this information to be latched into the block length register. The lower block length information along with the upper block length information indicates how many blocks of data will be transferred.

The Decode 14 signal activates the Clear Syndrome Register signal (R8–9) and the Clear HISP Syndrome Register signal (R76). The Clear Syndrome Register signal is sent to the IA option, and the Clear HISP Syndrome Register signal is sent to the IE option. The IA and IE options use these signals to clear the ECC error output and multiple-bit error output.

The Decode 15 signal enables the I/O clear information to be latched into the I/O channel clear register. More information is provided on the I/O channel clear register in Section 11, “LOSP and HISP Channels.”

The ID option also receives the control signal, Load Address Rank 1, from the PE. The ID option uses this signal to generate ram-on-array (ROA) control. The ROA control signals are sent to the IG option.

The ID option provides the HISP protocol for address and incoming data. This HISP protocol consists of three signals: Address Ready, Transmit Data, and Clear Channel.

The Address Ready signal is first generated after the ID option receives the Transmit Address signal from the external device. Internal circuitry of the ID option generates another two or four Address Ready signals depending on whether the channel is configured for 100 MB/s or 200 MB/s. The Address Ready signal is sent to the external device.

The Transmit Data signal is generated after the ID option receives the Data Ready signal from the external device and the Go I/O signal from the EE option. The ID option continues to generate the Transmit Data signal to the external device until the block length is 0.

The Clear Channel signal is obtained from the I/O clear memory-mapped register. The microprocessor of the input node writes the clear information into the I/O clear memory-mapped register.

The ID option also provides the LOSP protocol for incoming data on the LOSP channel. This protocol consists of the Input Resume signal. The Input Resume signal is generated using the information from the LOSP-in configuration register. This information provides the timing parameters that indicate when the Input Resume signal should be sent to the external device.

IG Option - HISP Input Data Buffer

The IG option buffers data from the HISP channel until the BLT transfers the HISP data to the local memory of the destination node. The IG option also sends the HISP address to the HISP channel and provides a data path between the PE and the LOSP channel. Refer to Figure 4-14.

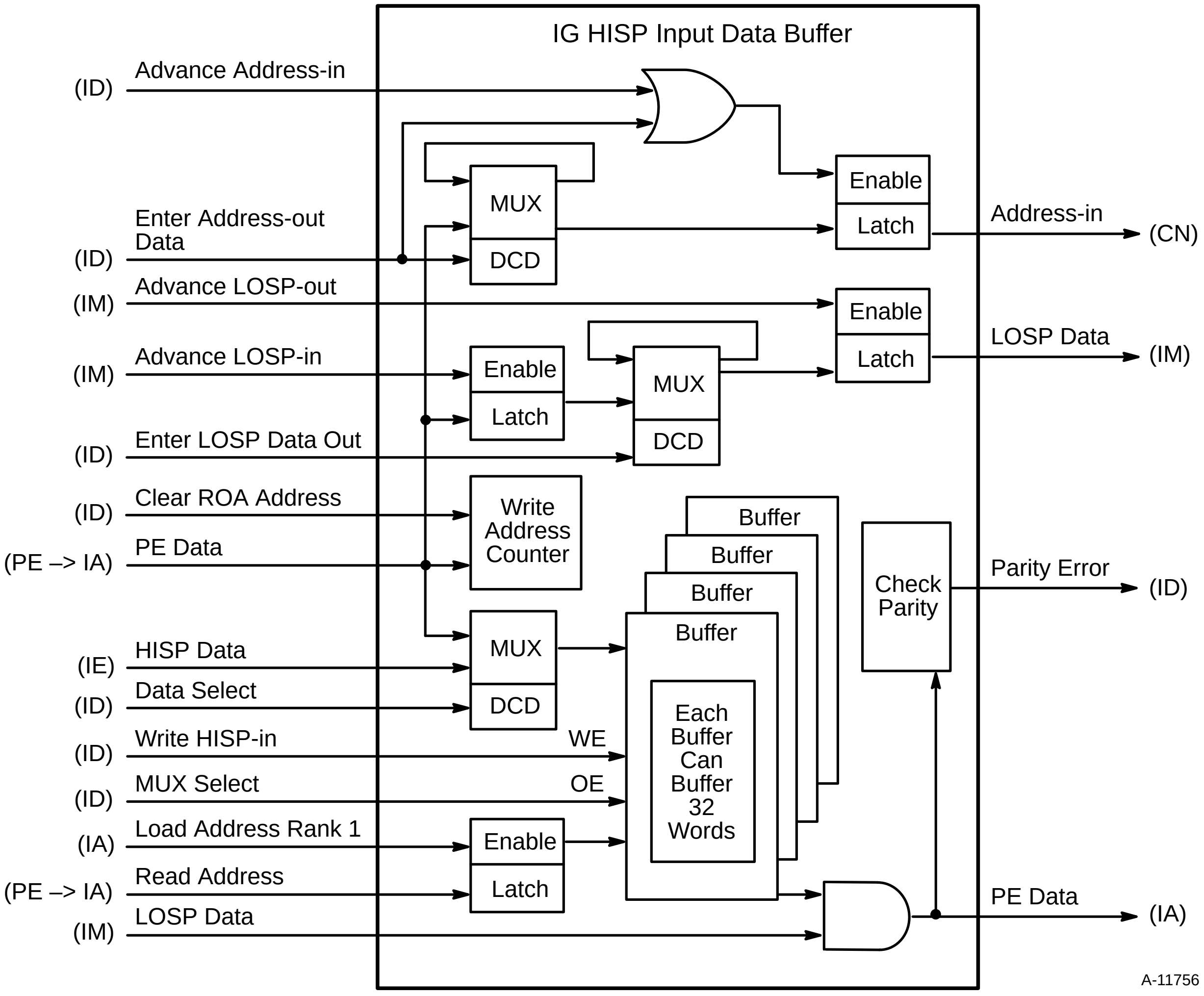


Figure 4-14. IG Option

The IG option contains four ram-on-array (ROA) buffers. Each buffer can store 32 64-bit words. This arrangement enables eight 16-word blocks of data from the HISP channel to be stored in the buffer before it is full.

The reading and writing of the buffer is controlled by the ID option. The ID option sends the IG option two control signals: Write HISP In and HISP In/Other Data Mux Select. The IG option uses the Write HISP In signal to generate the Write Enable signal for the buffer. The HISP In/Other Data Mux Select signal is used to generate the Output Enable signal for the buffer.

The read and write ROA address is sent to the IG option by the PE. The write address enters the IG option using the PE data input. The IG option uses this address as the first buffer address. The IG option generates the next successive write buffer address by incrementing the previous address by one. The IG option continues to increment the address by one until the ID option sends the Clear ROA Address signal. The read address enters the IG option using the inputs i230–234 and i241. The input signal, Load Address Rank 1, enables the IG option to latch in the read address. The IG option continues to input new read addresses until the Load Address Rank 1 signal is 0.

The PE sends the 16-bit or 12-bit HISP address to the IG option. The IG option sends this address to the HISP channel using two transfers. The Advance Address-in signal and Enter Address-out Data signal from the ID option control the two transfers.

The IG option also provides a data path between the PE and the LOSP channel. The IG option inputs 32 bits of data from the PE and sends this data to the LOSP channel using four transfers. The Advance LOSP-in signal and Advance LOSP-out signal from the IM option control the four transfers. The IG option also inputs LOSP data from the IM option and sends this data to the PE.

IE Option - HISP Error Correction

The IE option is a 10k gate array that contains circuitry that checks for errors. The IE option corrects any single-bit errors and notifies the ID option of any multiple-bit errors. Refer to Figure 4-15. This error correction feature can be disabled.

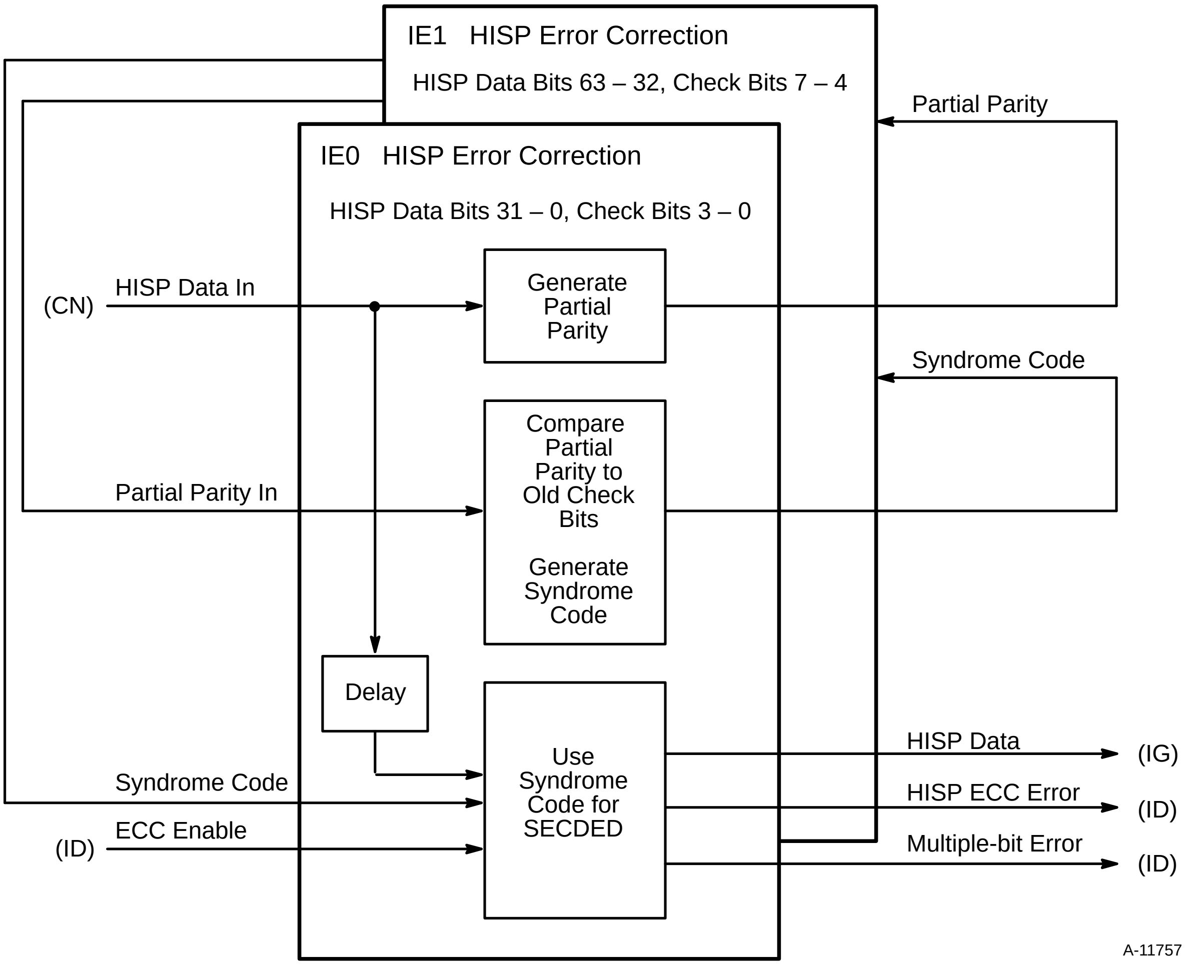

IE1 HISP Error Correction
HISP Data Bits 63 – 32, Check Bits 7 – 4
IE0 HISP Error Correction
HISP Data Bits 31 – 0, Check Bits 3 – 0
(CN)
HISP Data In
Generate Partial Parity
Partial Parity
Syndrome Code
Partial Parity In
Compare Partial Parity to Old Check Bits
Generate Syndrome Code
Delay
Use Syndrome Code for SECDED
Syndrome Code
(ID)
ECC Enable
HISP Data
(IG)
HISP ECC Error
(ID)
Multiple-bit Error
(ID)
A-11757

Figure 4-15. IE Option

The IE options check for errors by generating partial parity for the HISP data. This partial parity is communicated between the two IE options. Both IE options compare this partial parity to 4 check bits. This comparison yields a syndrome code. The syndrome code is communicated between the two IE options. The IE options analyze the syndrome code to determine whether there are any errors.

The IE options check for single-bit errors and multiple-bit errors. If the IE options detect a single-bit error, they use the syndrome code to determine the byte within the word and the bit within the byte that are in error. The incorrect bit is then corrected, and the HISP data is sent to the IG option. The IE options notify the ID option of the single-bit error by setting the HISP ECC Error signal to 1. If the IE options detect a multiple-bit error, the IE options set both the HISP ECC Error signal and the Multiple Bit Error signal to 1. These two signals are sent to the ID option, and the HISP data is sent to the IG option.

The single-bit error correction feature can be disabled on the IE option. The user can disable error correction by writing a 1 to bit 15 of the HISP address configuration register in the ID option. The ID option notifies the IE option to disable error correction by setting a signal called ECC Enable to 0.

Slave Input Circuitry

The slave input circuitry is also made up of four types of options. These options are similar to the options of the master input circuitry with a few exceptions. Table 4-1 lists the option types of both the master input circuitry and the slave input circuitry, and notes their differences.

Table 4-1. Differences Between Master and Slave Input Circuitry

Master Input Circuity	Slave Input Circuitry	Differences
ID Option	IT Option	The ID option generates control to send HISP address to the host system. The IT option generates control to receive the HISP address from an I/O cluster.
IG Option	IJ Option	The IG option receives the HISP address from the PE of the input node. The IG option sends this address to the external device. The IJ option receives the HISP address from an I/O cluster. The IJ option checks the parity of the HISP address and sends it to the PE of the input node.
IE Option	IE Option	None
IA Option	IA Option	None

Processing Element Node Circuitry

The processing element node circuitry in the input node consists of hardware that is similar, although not identical, to the hardware used for the regular processing element node. The following subsections describe the differences between the processing element node circuitry of the input node and the regular processing element node.

PE

The PE in the input node is designed to interface with the HISP input circuitry. Because of this characteristic, the PE in the input node does not contain all of the circuitry that a PE in a processing element node contains.

The microprocessor in the PE of the input node is identical to the microprocessor in the PE of the processing element node.

The local memory of the PE in the input node is identical to the local memory of the PE in the regular processing element node, except that it contains half the amount of memory storage. This memory resides on a daughter card that is attached to the I/O gateway printed circuit board; but because there is half the amount of memory storage, only one daughter card is needed.

The PE support circuitry of the input node is also similar to the PE support circuitry of the regular processing element node, with a few exceptions. Table 4-2 lists the option types of both PEs and notes their differences.

Table 4-2. Differences Between A-series Options

Input Node	Processing Element Node	Differences
AI	AM	The AI option does not support the following features: Prefetch operation Read ahead operation Swap operation DRAM page mode Buffered CPU cycle request CPU data cache invalidates on incoming writes V_WHOAMI and VRT_RNG registers Virtual PE range checking The AI option memory organization is also different. The AI option has a dual-bank organization where bank 0 is DRAM and bank 1 is I/O. The data path to local memory (bank 0) is 2 words wide. The data path for I/O (bank 1) is 1 word wide.
AA/AB	AJ/AK	The AJ and AK options have a different data path structure that supports the read ahead feature better. The AA and AB options data path for the upper word of bank 1 is removed because I/O does not need this path. The swap circuitry on the AA and AB options is not used.
AR	AR	None
AE	AE	None

Block Transfer Engine

The block transfer engine of the input node is identical to the block transfer engine of the processing element node.

Network Interface

The network interface of the input node is identical to the network interface of the processing element node.

Interconnect Network

The network routers of the input node are made up of the same option type as the network routers of the processing element node; however, one less option is involved. The SR option handling the Y dimension is not needed because the input node is only connected to the X and Z dimensions.

Output Node

The output node contains output circuitry, processing element node circuitry, and interconnect network circuitry. The output circuitry consists of options that translate TTL and ECL signals, control the LOSP and HISP channels, buffer the HISP data, and check for single and multiple bit errors. The processing element node logic consists of options that initiate the I/O transfer upon request, transfer blocks of data, and create packets. The network interconnect circuitry transfers the packets through the network.

Output Circuitry

The master output circuitry is made up of four option types: the IA option, the IC option, the IF option, and the IH option. Refer to Figure 4-16. The slave output circuitry is also made up of four option types. These option types are the IA option, the IF option, the IS option, and the IK option.

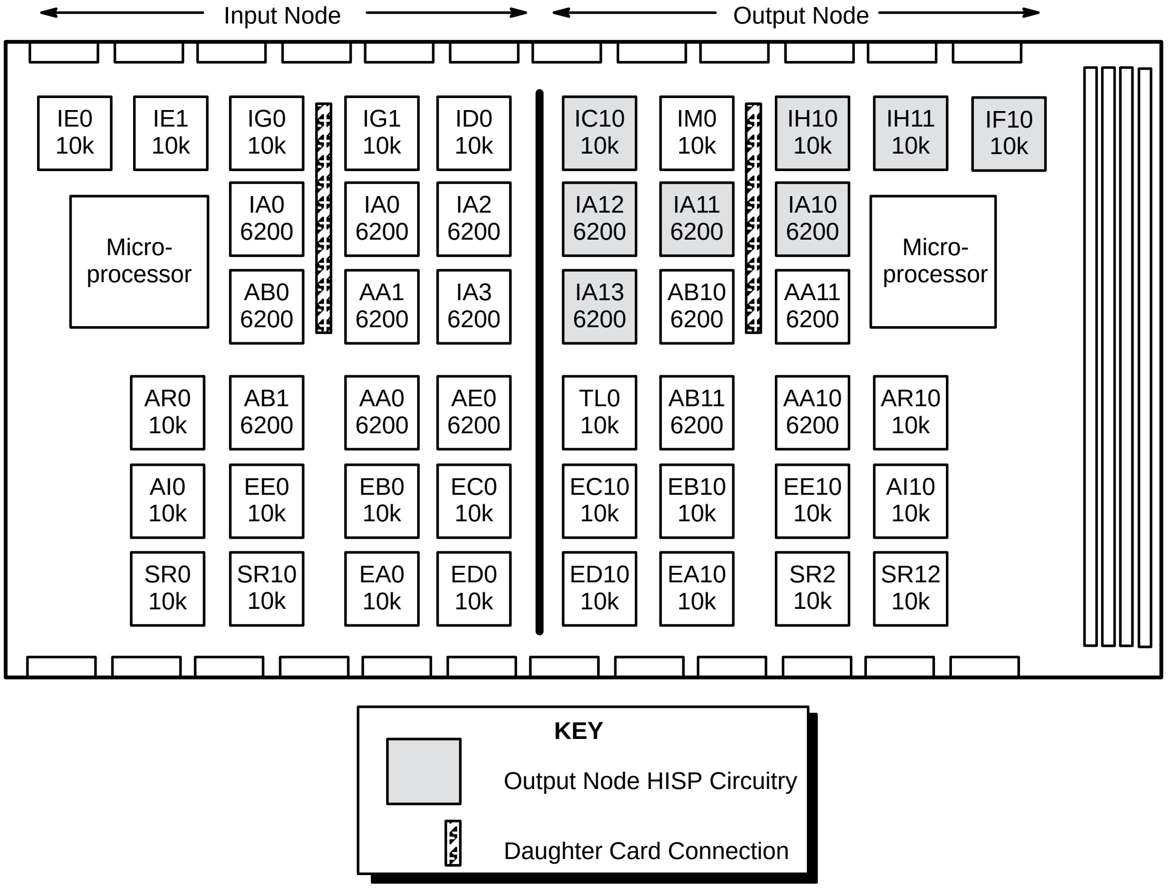


Figure 4-16. Master I/O Gateway, Output Options

IA Option - TTL to ECL Translation and HISP Error Check

The IA option in the output node translates TTL signals from the PE to ECL signals before sending the data to the HISP channel. The IA option also checks the PE data for errors. Refer to the shaded area of Figure 4-17.

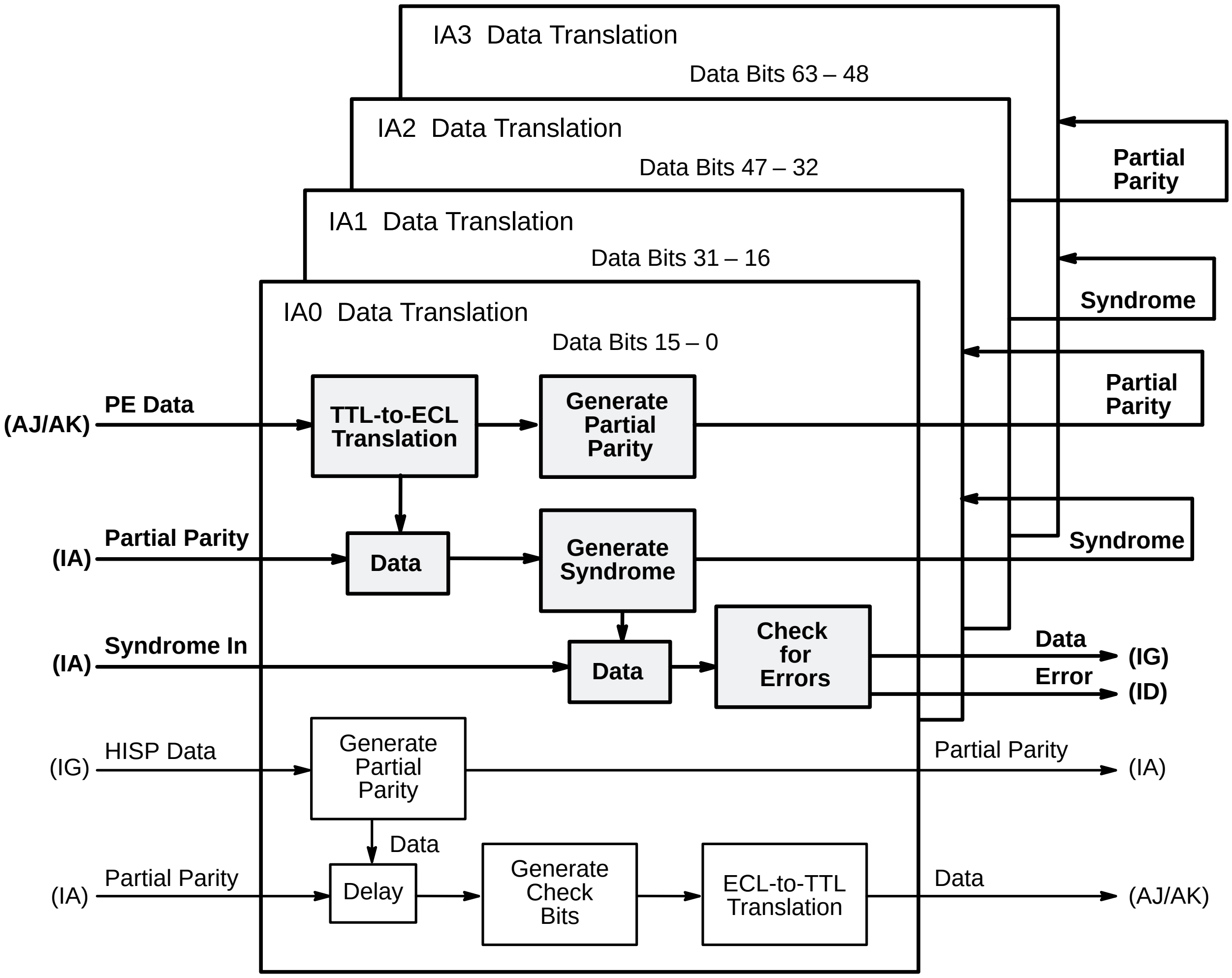


Figure 4-17. IA Option

After the IA options convert the PE data from TTL signals to ECL signals, the IA options check the data for errors. To do this, each IA option generates parity for the data bits it handles and transfers this partial parity to the other IA options (refer again to Figure 4-17). The IA options compare the partial parity to the check bits that accompanied the PE data. This compare yields a syndrome code that is used to determine whether there is a single- or multiple-bit error. When there is a single-bit error, the IA options correct this bit and notify the ID option of the error. When there is a multiple-bit error, the IA options also notify the ID option of the error, but the multiple-bit error cannot be corrected.

IC Option - HISP Output Channel Control

The IC option controls the HISP and LOSP channels during the output of HISP data. Refer to Figure 4-18. This control consists of generating HISP and LOSP channel control, HISP and LOSP channel protocol, and ram-on-array (ROA) control.

The IC option generates the HISP and LOSP control using the address, control, and data it receives from the PE of the output node. The address and control are used to input other control information and to steer the data received from the microprocessor to an I/O memory-mapped register.

To input other control information and steer the data to the I/O memory-mapped registers, the IC option uses the control signal, Load Address Rank 0, to latch in the address. This address is used to generate 13 different decode signals, as shown in Figure 4-18. The following text describes the function of each decode signal.

The Decode 0 signal activates the LOSP and HISP configuration control. The configuration control enables the LOSP and HISP configuration information to be latched into the configuration memory-mapped registers.

The LOSP configuration information is used to set up the width and the delay count of the LOSP data resume pulse. The HISP configuration information controls the timing of the HISP address and data transfers. More information is provided on the LOSP and HISP configuration memory-mapped registers in Section 11, “LOSP and HISP Channels.”

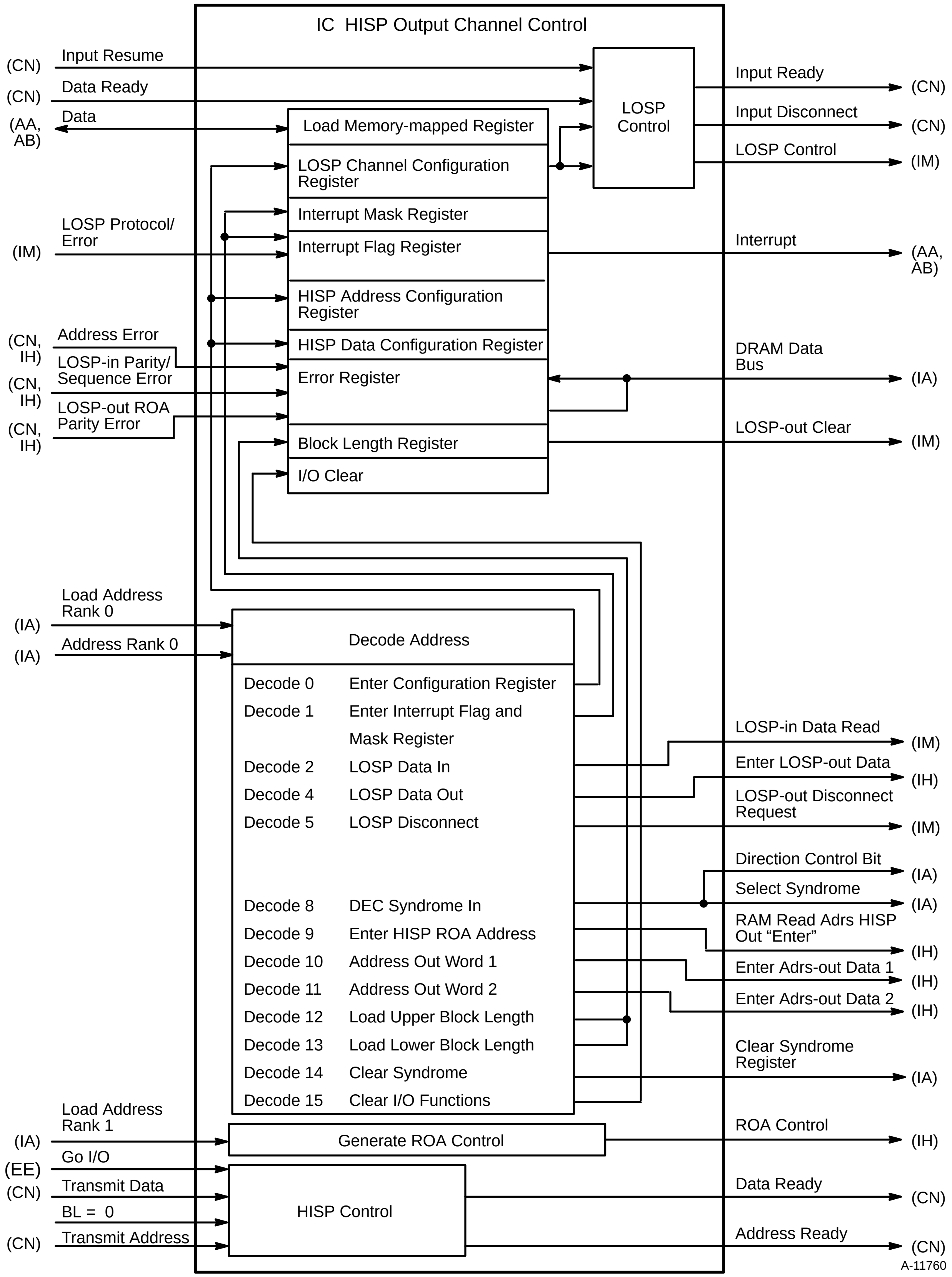

IC HISP Output Channel Control
(CN) Input Resume
(CN) Data Ready
(AA, AB) Data
LOSP Control
Load Memory-mapped Register
LOSP Channel Configuration Register
Interrupt Mask Register
Interrupt Flag Register
HISP Address Configuration Register
HISP Data Configuration Register
Error Register
Block Length Register
I/O Clear
(IM) LOSP Protocol/ Error
(CN, IH) Address Error
(CN, IH) LOSP-in Parity/ Sequence Error
(CN, IH) LOSP-out ROA Parity Error
Input Ready (CN)
Input Disconnect (CN)
LOSP Control (IM)
Interrupt (AA, AB)
DRAM Data Bus (IA)
LOSP-out Clear (IM)
(IA) Load Address Rank 0
(IA) Address Rank 0
Decode Address
Decode 0 Enter Configuration Register
Decode 1 Enter Interrupt Flag and Mask Register
Decode 2 LOSP Data In
Decode 4 LOSP Data Out
Decode 5 LOSP Disconnect
Decode 8 DEC Syndrome In
Decode 9 Enter HISP ROA Address
Decode 10 Address Out Word 1
Decode 11 Address Out Word 2
Decode 12 Load Upper Block Length
Decode 13 Load Lower Block Length
Decode 14 Clear Syndrome
Decode 15 Clear I/O Functions
LOSP-in Data Read (IM)
Enter LOSP-out Data (IH)
LOSP-out Disconnect Request (IM)
Direction Control Bit (IA)
Select Syndrome (IA)
RAM Read Adrs HISP Out “Enter” (IH)
Enter Adrs-out Data 1 (IH)
Enter Adrs-out Data 2 (IH)
Clear Syndrome Register (IA)
(IA) Load Address Rank 1
Generate ROA Control
ROA Control (IH)
(EE) Go I/O
(CN) Transmit Data
BL = 0
(CN) Transmit Address
HISP Control
Data Ready (CN)
Address Ready (CN)
A-11760

Figure 4-18. IC Option

The Decode 1 signal selects between the error information from the error register and the interrupt information from the interrupt flag register that will be sent to the microprocessor.

The Decode 1 signal also enables the interrupt mask information to be latched into the interrupt mask register. The interrupt mask information masks out sources of interrupts.

The Decode 2 signal activates the control signals LOSP In Data Read and Direction Control. The LOSP In Data Read signal (R61) is sent to the IM option (LOSP circuitry). This signal alerts the IM option that there is LOSP data to input. The IM option then controls the input of this data. The Direction Control signal (R2–3) is sent to the IA options. This signal controls the bidirectional pins of the IA option. When this signal is a 1, the IA option sends data to the PE. When it is a 0, the IA option inputs data from the PE.

The Decode 4 signal activates the control signal LOSP Data Out (R20). The LOSP Data Out signal is sent to the IH options so that each IH option can input data from its associated PE.

The Decode 5 signal activates the control signal LOSP Out Disconnect Request (R65). The LOSP Out Disconnect Request signal is sent to the IM option. The IM option uses this signal to generate another LOSP Out Disconnect signal, which is sent to the IC option. The IC option then sends the disconnect signal to the external device.

The Decode 6 signal selects the Heartbeat signal from the heartbeat generation register. The Heartbeat signal is fanned out to the PEs within the system. The heartbeat signal can be used to establish the global time of day among the PEs, to detect a malfunctioning PE, and to trigger either a selective or global reset.

The Decode 8 signal, like Decode 2, activates the Direction Control signal that is sent to the IA option. This signal also activates the control signals Read DEC and HISP Syndrome Register (R10–11). The Read DEC and HISP Syndrome Register signals are also sent to the IA option. The IA option uses the DEC and HISP Syndrome signals to select the DEC and HISP syndrome information that is sent to the PE.

The Decode 9 signal activates the Enter ROA Address signal (R25). The Enter ROA Address signal is sent to the IH option. The IH option uses this signal to enable the input of the ROA address from the PE.

The Decode 10 signal activates the Enter HISP Address Word 1 signal (R21). The Enter HISP Address Word 1 signal is sent to the IH option. The IH option uses this signal to enable the input of the HISP address.

The Decode 11 signal activates the Enter HISP Address Word 2 signal (R22). The Enter HISP Address Word 2 signal is sent to the IH option. The IH option uses this signal to enable the input of the upper HISP address.

The Decode 12 signal activates the Load Upper Block Length signal. The Load Upper Block Length signal selects the upper block length information from the input data and enables this information to be latched into the block length register. The upper block length information along with the lower block length information indicates how many blocks of data will be transferred.

The Decode 13 signal activates the Load Lower Block Length signal. The Load Lower Block Length signal selects the lower block length information from the input data and enables this information to be latched into the block length register. The lower block length information along with the upper block length information indicates how many blocks of data will be transferred.

The Decode 14 signal activates the Clear Syndrome Register signal (R8–9). The Clear Syndrome Register signal is sent to the IA options. The IA options use this signal to clear the ECC error output and multiple-bit error output.

The Decode 15 signal enables the I/O clear information to be latched into the I/O channel clear register. More information is provided on the I/O channel clear memory-mapped register in Section 11, “LOSP and HISP Channels.”

The IC option also receives the control signal, Load Address Rank 1, from the PE. The IC option uses this signal to generate ram-on-array (ROA) control. The ROA control signals are sent to the IH option.

The IC option provides the HISP protocol for address and outgoing data. This HISP protocol consists of five signals: Address Ready, Data Ready, Last Word Flag, Disable SECDED, and Clear Channel.

The Address Ready signal is first generated after the IC option receives the Transmit Address signal from the external device. Internal circuitry of the IC option generates another two or four Address Ready signals, depending on whether the channel is configured for 100 MB/s or 200 MB/s. The Address Ready signal is sent to the external device.

The Data Ready signal is generated after the IC option receives the Transmit Data signal from the external device and the output buffer is full.

The Last Word Flag is set to a 1 by the IC option after the block length is decremented to 0.

The Disable SECDED signal is obtained from the HISP address configuration memory-mapped register. The microprocessor writes the HISP address configuration information into the HISP address configuration memory-mapped register.

The Clear Channel signal is obtained from the I/O clear memory-mapped register. The microprocessor writes the clear information into the I/O clear memory-mapped register.

The IC option also provides the LOSP protocol for outgoing data on the LOSP channel. This protocol consists of the Input Ready signal and the Input Disconnect signal.

The Input Ready signal is generated after the IC option receives the LOSP Out Data Request signal from the IM option or after it receives the Input Resume signal from the external device. The Input Disconnect signal is generated after the IC option receives the LOSP Out Start Disconnect signal from the IM option.

IH Option - HISP Output Data Buffer

The IH option's primary function is to buffer data from a source node until it is instructed to send this data to the HISP channel. The IH option also sends the HISP address to the HISP channel and provides a data path between the PE and the LOSP channel. Refer to Figure 4-19.

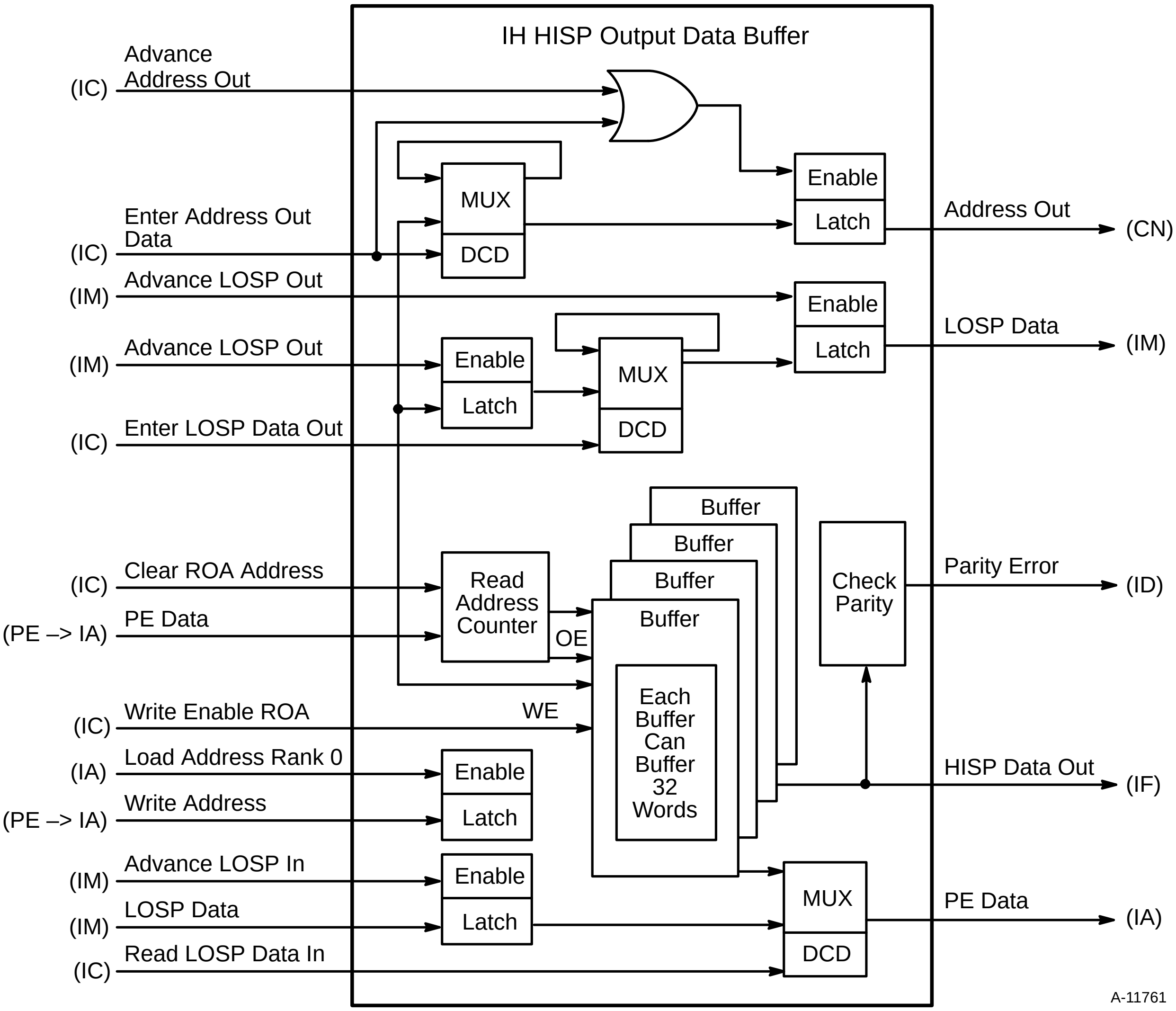


Figure 4-19. IH Option

The IH option contains four ram-on-array (ROA) buffers. Each buffer can store 32 64-bit words. This arrangement enables eight 16-word blocks of data from local memory to be stored in the buffer before it is full.

The IC option controls the reading and writing of the buffer. The IC option sends the Write Enable ROA signal and Clear ROA Address signal to the IH options. The IH options use the Write Enable ROA signal as the Write Enable for the buffer. The IH options use the Clear ROA Address signal to clear a counter that generates successive read buffer addresses.

The IH options receive the initial read and write addresses from the PE. The read address enters the IH option through the PE data input. The IH options use this address as the first buffer address. The IH options generate the next successive read buffer address by incrementing the previous address by 1. The IH options continue to increment the address by 1 until the IC option sends the Clear ROA Address signal. The write address enters the IH options through the inputs i230 through i234 and i241. The input signal, Load Address Rank 0, enables the IH option to latch in the write address. The IH options continue to input the read address until all of the data has been read out of the buffer.

To send the HISP address to the external device, the IH options input the 16-bit or 12-bit HISP address from the PE. The IH options take this address and send it to the HISP channel using two transfers. The Advance Address-in signal and the Enter Address-out Data signal from the IC option control the two transfers.

The IH options also provide a data path between the PE and the LOSP channel. The IH options input a word of data, 16 bits at a time, from the LOSP channel. This word of data is sent to the PE. The Advance LOSP-in signal and the Advance LOSP-out signal from the IM option control the four transfers needed to retrieve the word of data. The IH option also inputs LOSP data from the IM option and sends this data to the PE.

IF Option - HISP Output ECC Generation

The IF option is a 10k gate array that contains circuitry that generates error correction code or check bits for the HISP data. Refer to Figure 4-20.

The check bit generation can be disabled by the IC option. When disabled, the outputs for the check bits are all 0’s. The check bits may also be set to equal the checkbits generated from the previous word; this occurs when the ECC Diagnostic Mode signal is set.

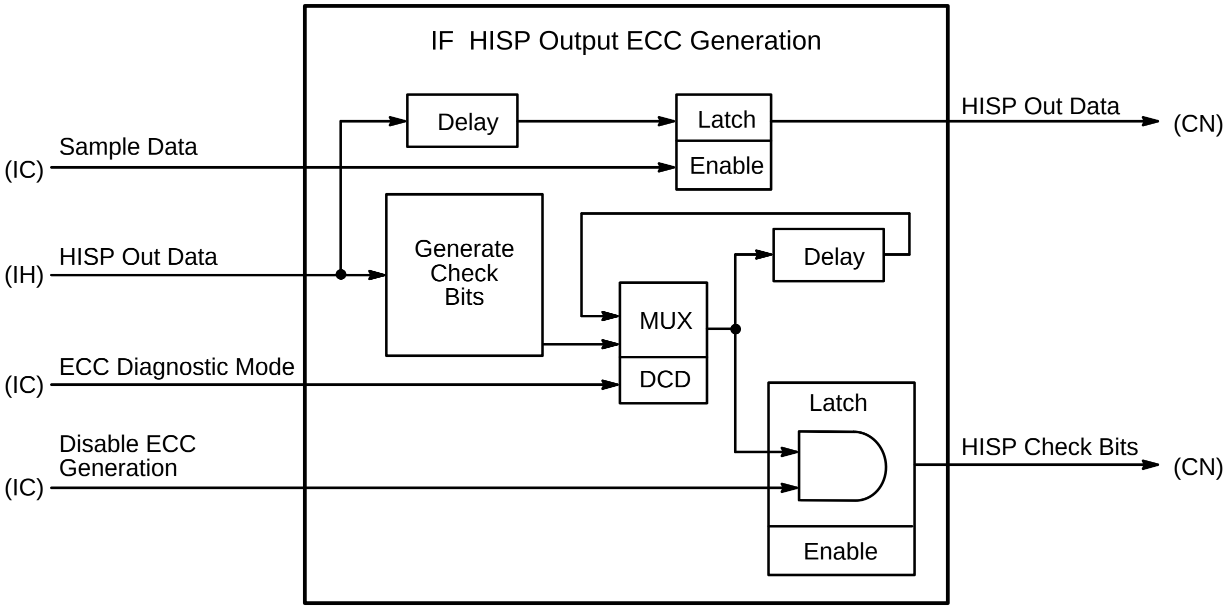


Figure 4-20. IF Option

Slave Output Circuitry

The slave output circuitry is also made up of four types of options. These options are similar to the options of the master output circuitry with a few exceptions. Table 4-3 lists the option types of both the master output circuitry and the slave output circuitry, and notes their differences.

Table 4-3. Differences Between Master and Slave Output Circuitry

Master Output Circuity	Slave Output Circuitry	Differences
IC Option	IS Option	The IC option generates control to send the HISP address to the host system. The IS option generates control to receive the HISP address from an I/O cluster.
IH Option	IK Option	The IH option receives the HISP address from the PE of the input node. The IG option sends this address to the external device. The IK option receives the HISP address from an I/O cluster. The IK option checks the parity of the HISP address and sends it to the PE of the input node.
IF Option	IF Option	None
IA Option	IA Option	None

Other Circuitry in the Output Node

The processing element node circuitry of the output node is identical to the processing element node circuitry of the input node. This is also true for the BLT, the network interface, and the network routers.

LOSP Circuitry

The LOSP circuitry controls the transfer of request and response information over the LOSP channel. The LOSP circuitry is contained in one option type, the IM option. Refer to Figure 4-21.

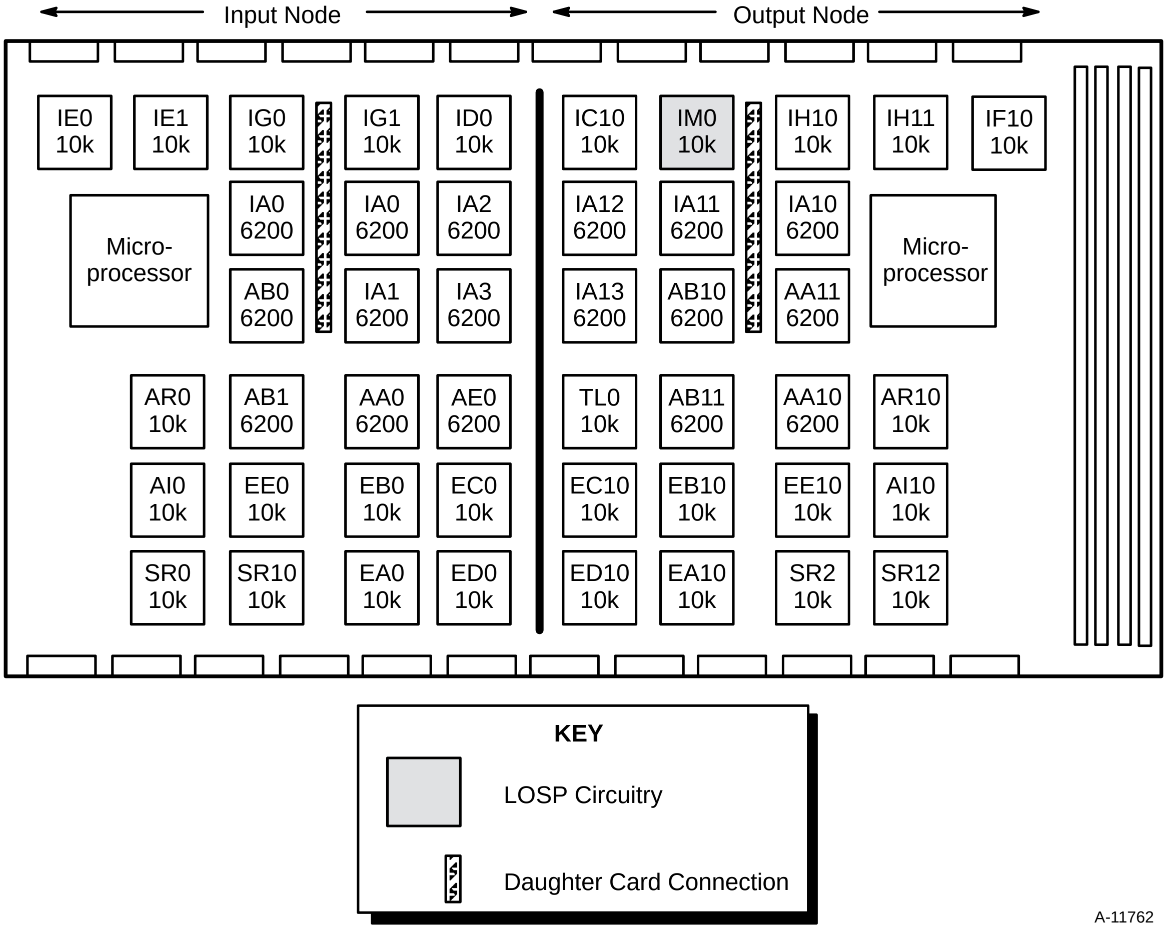


Figure 4-21. LOSP Circuitry

The IM option inputs LOSP data through the shared LOSP-in channel, buffers this data, and checks it for parity errors. The IM option also outputs LOSP data through the shared LOSP-out channels. The IM option receives this LOSP data from the input node or the output node. The IM option buffers this LOSP data and generates parity for it before it is sent to the external device. Refer to Figure 4-22.

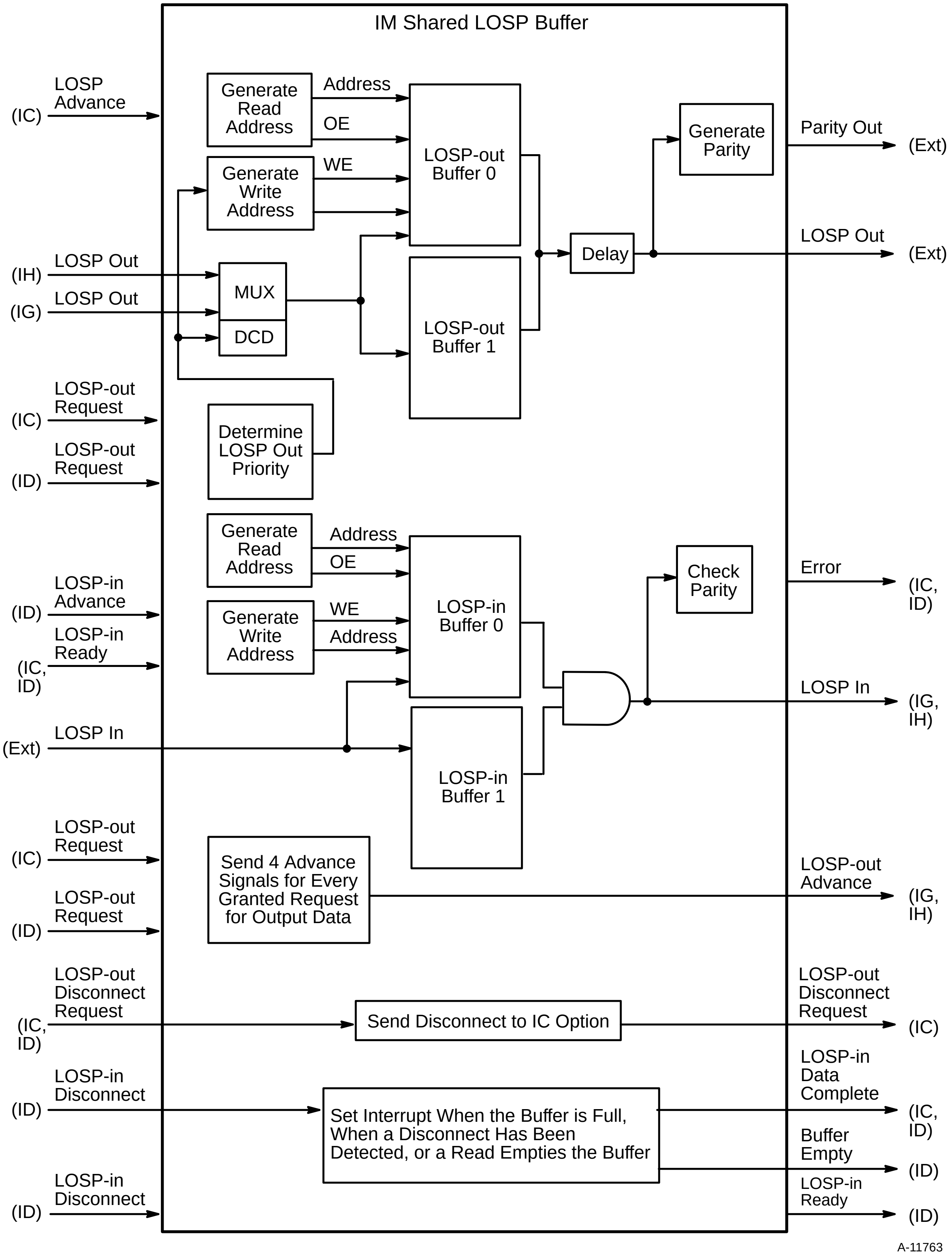


Figure 4-22. IM Option

The IM option inputs LOSP data through the shared LOSP-in channel and writes the data into LOSP-in buffer 0 or LOSP-in buffer 1. The ID option controls the writing of these buffers. The ID option sends the IM option a signal called LOSP In Sample and Advance. This signal enables the IM option to generate consecutive write buffer addresses.

The LOSP In Sample and Advance signal also activates the Write Enable for the buffers. This signal is ANDed with write buffer address bits 6 true and 6 false. When the ANDing with write buffer address bit 6 false produces a 1, the LOSP data is written into LOSP-in buffer 0. When the ANDing with write buffer address bit 6 true produces a 1, the LOSP data is written into LOSP-in buffer 1.

The IM option generates a read buffer address as long as the LOSP-in buffers are not empty and the buffers of the IG option are not full. This address is presented to the buffers, and the LOSP data is read out of one of the two buffers. The Output Enable, which in this case is read buffer address bit 6, selects the buffer from which the data will be read. When this bit is set to 0, the data is read out of LOSP-in buffer 0. When this bit is set to 1, the data is read out of LOSP-in buffer 1.

The data that is read out of the buffer is sent to the IG options in the input node and to the circuitry within the IM option that checks for parity errors. If a parity error is detected, the IM option notifies the ID option of the error.

The IM option receives LOSP-out data from the IG option of the input node or from the IH option of the output node. This data is written into one of two buffers, LOSP-out buffer 0 or LOSP-out buffer 1. Refer again to Figure 4-22. Because both nodes can make LOSP requests simultaneously, a conflict can occur. When this happens, the IM option services the output node first.

Once the IM option grants the request of the input node or the output node, the IM option generates a write buffer address, and the data is written into LOSP-out buffer 0 or LOSP-out buffer 1. Which buffer is used depends on the Write Enable. The Write Enable is generated by ANDing the LOSP Out Advance signal with write buffer address bits 6 true and 6 false. When the ANDing with write buffer address bit 6 false is set to 1, the node data is written into LOSP-out buffer 0. When the ANDing with bit 6 true is set to 1, the node data is written into LOSP-out buffer 1.

The IM option reads the data from the LOSP-out buffers after receiving the LOSP Out Advance signal from the IC option. The LOSP Out Advance signal enables the IM option to generate consecutive read buffer addresses along with the buffer output enable.

The IM option reads the LOSP data out of the buffer and sends it to the external device. The LOSP data is also sent to circuitry within the IM option that generates parity. The IM option generates parity for the LOSP data and sends the parity bits to the external device.

5 STAND-ALONE DIAGNOSTIC SYSTEM INITIALIZATION

While running diagnostics, the CRAY T3D system is initialized in two stages. The first stage involves master clearing the CRAY T3D system, and loading and executing boot code from a serial read-only memory (SROM). The second stage involves loading and executing the stand-alone diagnostic privileged architecture library (SADPAL) code.

NOTE: While reading through the following subsections, refer to Table 5-2 beginning on page 5-12.

Deadstart Sequence

The CRAY T3D system connects to a Cray Research mainframe or input/output subsystem model E (IOS-E) cluster using a 100-Mbyte/s or 200-Mbyte/s high speed (HISP) channel and a 6-Mbyte/s low speed (LOSP) channel. It can also connect to a maintenance workstation (MWS) using the LOSP channel. The Cray Research mainframe, IOS-E, or MWS will be referred to as the host system for the CRAY T3D system throughout this section.

The MWS-based diagnostic control program that runs on the host system, `mppms`, sets up the configuration for the CRAY T3D system. This involves determining and designating the initial barrier configuration and calculating each node’s routing table.

After the configuration setup is complete, `mppms` initiates the deadstart sequence. It does this by sending a Master Clear signal to the deadstart I/O gateway using the LOSP channel. The deadstart I/O gateway uses this Master Clear signal to generate a 4-clock period (CP) Global Reset System Heartbeat signal. This signal is fanned out to each PE within the CRAY T3D system. Within each PE, the Global Reset Heartbeat signal is sent to the AR option.

The Global Reset Heartbeat signal causes the AR options to enable the PE reset and the network reset. The AR option sends the PE Reset signal to the AM option and sends the Network Reset signal to the EC option. The

EC option fans the Network Reset signal out to the EA option, the EB option, the ED option, the EE option, the SR options, and the TL (clock) option. All of these options use this signal to master clear their circuitry. The AM option uses the PE Reset signal to master clear its own circuitry, and it sends this signal to the microprocessor.

The PE Reset signal causes the Reset_l signal on the microprocessor to set. This forces the rest of the PE support circuitry into a reset state. It also causes the microprocessor to read the boot code from the SROM. (The SROM chips are mounted on the printed circuit board near the microprocessor.) This code is loaded into the instruction cache of the microprocessor. Once the code is loaded, the microprocessor executes the boot code.

The boot code consists of three code segments. The first code segment performs a portion of the initialization and tests for the deadstart node. The second code segment executes on the deadstart node only, and the third code segment executes on all nodes except the deadstart node.

First Segment of the Boot Code

During the first segment, the microprocessors execute boot code that initializes several internal microprocessor registers as follows:

- The process cycle counter is enabled and set to all 1’s.
- The interrupt masks are disabled, and all pending interrupts are cleared.
- All internal microprocessor buffers are purged and then enabled.
- The error correction of the bus interface unit is enabled, and the level-two cache is disabled.
- The super page mode and machine check of the address unit are enabled.
- The super page mode, floating point, branch prediction, and dual issue mode of the instruction unit are enabled.

The first segment boot code also instructs the microprocessor to initialize processing element node circuitry as follows:

- The physical PE number (P_WHOAMI) register is copied into the virtual and logical PE number (V_WHOAMI and L_WHOAMI) registers located in the AM option.
- The virtual PE range mask register on the AM option is cleared.
- The logical PE translation parameters are loaded into the network interface PE adjust registers located in the ED options. These parameters are set so that a PE adjustment does not take place.
- All entries in the DTB annex are set to match the local PE number.
- The barrier synchronization logic is initialized with all bits enabled and in eureka mode.
- The DRAM configuration registers are written with parameter information.
- The routing table entries are cleared.

The final portion of the first segment boot code instructs the microprocessor to read the P_WHOAMI register to determine whether the processing element node is an I/O gateway node. When the node is a processing element node, the nondeadstart code (third code segment) is loaded into the instruction cache of the microprocessor and executed.

When the node is an I/O gateway node, the microprocessor reads the interrupt flag register (IO_IFLAG) to determine whether the Master Clear signal is active on the channel. When the Master Clear is not active, the nondeadstart code is loaded into the instruction cache of the microprocessor and executed.

When the Master Clear is enabled, the output node of the I/O gateway sends a master clear acknowledge packet back to the host system over the LOSP channel. After the acknowledgement is sent, the input node and the output node try to read a packet from the LOSP channel. The node that is able to read the packet is the deadstart node. (The host system determines which node receives the packet.) When the node is the deadstart node, the deadstart code (second code segment) is executed. The other node of this I/O gateway executes the nondeadstart code.

Second Code Segment

The microprocessor in the deadstart node executes the second code segment, also referred to as the deadstart code. This code enables the deadstart node to send a response packet to `mppms` indicating that it is ready to input data through the LOSP channel. The microprocessor polls the IO_IFLAG register to determine when the host system sent data over the LOSP channel. When the LOSP input data available bit of the IO_IFLAG register is set to 1, the deadstart node reads data from the LOSP input buffer.

This LOSP data consists of a command and a header. The microprocessor must interpret the command as either a READ_MEM command, a WRITE_MEM command, a GO_LOOP command, or a LOAD_PAL command or the microprocessor will reject the data. The microprocessor also rejects the data when there is a parity error.

The READ_MEM command and the WRITE_MEM command enable the host system to read and write to any location in the local memory of the deadstart node. Because these commands are part of the SROM code, memory will not be disturbed when issuing these commands. The READ_MEM command is used when there is a severe system crash (system cannot be deadstarted). Using this command, the user can read the memory of the deadstart node to help isolate the problem. The WRITE_MEM command is used for debugging when developing software. The user can write a specific data pattern to the memory of the deadstart node and then read this memory to see whether memory was patterned correctly.

The GO_LOOP command initiates the level 0 diagnostic test. The level 0 diagnostics can test areas such as local memory and the LOSP channel.

The LOAD_PAL command enables the deadstart node to input the SADPAL code from the LOSP channel. The microprocessor of the deadstart node instructs the LOSP input circuitry and PE support circuitry to load the SADPAL code into local memory, beginning at location 0. After the SADPAL code is loaded into the local memory of the deadstart node, the microprocessor sends a response packet to `mppms`, exits the boot code, and enters the SADPAL code at the deadstart entry point.

Third Code Segment

The third code segment, also referred to as the nondeadstart code, runs in all nondeadstart nodes. This code segment begins with a loop instructing the microprocessors to read the data (configuration message) from the barrier register.

NOTE: The deadstart node uses the barrier register to distribute parameter and routing information to the other PEs within the system. For more information about how the barrier register is used, refer to the “Deadstart Entry Point” subsection later in this section.

The microprocessor compares the first three digits (12 bits) from the barrier register with the contents of the P_WHOAMI register. When they do not match, the microprocessor ignores the rest of the data associated with that configuration message unless the three digits are set to fff. The microprocessor interprets this number as an end-of-message marker, which causes the microprocessor to enter a tight-scope loop. This will occur on the nodes that are not configured into the system.

When the first three digits and the contents of the P_WHOAMI register match, the nondeadstart code instructs the microprocessor to input the rest of the message and write it into local memory and its data cache.

The last portion of the message consists of a check symbol that indicates the end of the message and whether any errors have occurred. When an error occurs, the entire message is rejected. Refer to Table 5-1 for a description of a configuration message.

Table 5-1. Description of the Configuration Message

Digit (4 Bits)	Description
2 – 0	Address of the destination PE.
3	Value to be loaded into the NODE_SCR register to partition the barrier distribution tree.
5 – 4	Value to be loaded into the BAR_TMG register to time the barrier distribution tree.
7 – 6	A command code. A value of zero indicates the deadstart sequence. A nonzero value selects a diagnostic loop for execution and aborts the deadstart process.
15 – 8	The routing tag entry for the deadstart I/O gateway. Digits 10 through 8 contain a logical node number that is applied as the address to the routing tag look-up table. Digits 15 through 11 contain the actual routing tag.
23 – 16	The routing tag entry for source node of the SADPAL code image. Digits 18 through 16 contain the logical node number that is applied as the address to the routing tag look-up table. Digits 23 through 19 contain the actual routing tag.
31 – 24	The routing tag entry for a regular node. Note: This field is optional. Also, the number of fields containing routing tag entries can range from 0 through n.
35 – 32	Reserved.
39 – 36	Value to be loaded into the L_WHOAMI register and the X_WHOAMI register.
55 – 40	Value to be loaded into the DRAM configuration register.

The configuration message now resides in local memory and the data cache. The nondeadstart code instructs the microprocessor to read this information out of the data cache and load this information into the appropriate registers. For example, digits 8 through 23 will be loaded into the routing tag look-up table.

NOTE: The microprocessor in the odd PE of the node reads the configuration message out of local memory and then loads the information into the appropriate registers. This is because the configuration message in the data cache was corrupted.

Next, the nondeadstart code instructs the microprocessors to decode the command (digits 6 and 7). When the command byte is 0 (indicates deadstart sequence), the microprocessor reads the SADPAL code from the local memory of the source PAL node. Each PE stores this copy of the SADPAL code in its local memory. Once the SADPAL code is written into memory, the microprocessor exits the SROM code and enters the SADPAL code at the initialization entry point.

NOTE: The source PAL node can be any node(s) in the system. The source PAL node is assigned by the system administrator.

When the command byte is a nonzero value, the microprocessor reads level 0 diagnostic code from the SROM. This diagnostic code tests areas such as local memory and the LOSP channel.

NOTE: The deadstart node can not execute the level 0 diagnostic code.

Stand-alone Diagnostic Privileged Architecture Library Code

The SADPAL code consists of two entry points: the deadstart entry point and the initialization entry point.

Deadstart Entry Point

The deadstart node enters the SADPAL code at the deadstart entry point. The deadstart entry point in the SADPAL code instructs the microprocessor to poll the IO_IFLAG register. When the LOSP input data available bit is set to 1, the deadstart node reads a command data structure from the LOSP input buffer. When the microprocessor does not interpret this new command as a LOAD_ROUTE command, the microprocessor rejects the command data structure. The microprocessor also rejects the command data structure when there are any parity errors or premature disconnects.

When the command is LOAD_ROUTE and there are no other errors, the microprocessor instructs the input circuitry and PE support circuitry to load the data portion of the command data structure into local memory. This data contains routing tag information for the deadstart node to all other nodes. From local memory, the deadstart node transfers this routing tag information to its routing tag look-up table.

Next, the SADPAL code enables the deadstart node to send a response packet to `mppms` and poll the IO_IFLAG register. When the LOSP input data available bit is set to 1, the deadstart node reads next data structure from the LOSP input buffer. This data structure consists of configuration information for the other PEs in the system. The `mppms` diagnostic control program indicates the start of this configuration information by sending a LOAD_CNFIG command across the LOSP channel.

Once the microprocessor receives the LOAD_CNFIG command, it instructs the input circuitry and PE support circuitry to input the LOSP data and write it into local memory. The deadstart node will distribute this information to the other PEs in the system using the barrier hardware.

NOTE: The barrier hardware was designed to synchronize the PEs, not to distribute data; therefore, the SADPAL code is written to handle this special case.

The SADPAL code instructs the deadstart node to divide the LOSP data into 4-bit transfers and append a validation bit to each 4-bit group of data. Next, the deadstart node generates two copies of the 5-bit group. Refer to Figure 5-1. This triplication provides a way for the destination PE to check the data for errors.

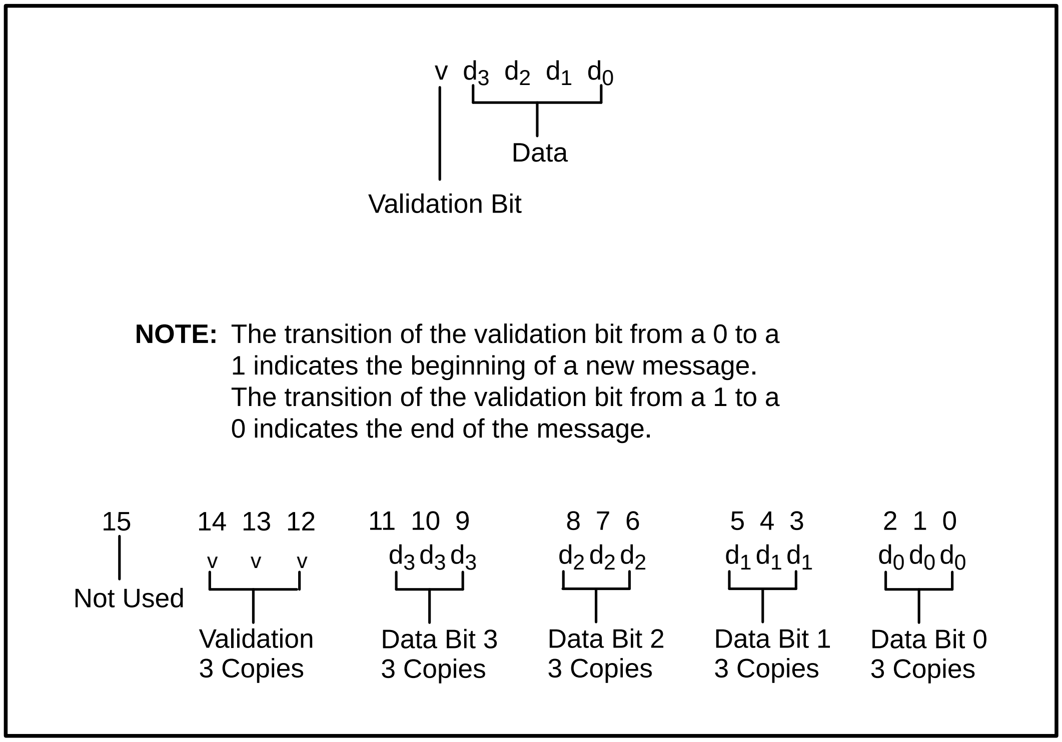


Figure 5-1. Bit Layout of Routing Tag Information for the Barrier Channel

The deadstart node places the 15 bits in the barrier register of the deadstart node (1 bit of the barrier register is not used). The deadstart node sends the 15 bits to the system module in 4-bit transfers. From the system module, the bits are fanned out to the other PEs within the system.

Once a configuration message is sent to each PE in the system, the SADPAL instructs the deadstart node to scan the diagnostic monitor communication area (DMCA) memory locations, send a response packet to `mppms`, and poll the IO_IFLAG register.

NOTE: There is one DMCA memory location for each PE. Normally, the deadstart node scans the DMCA memory locations until all PEs have responded to their DMCA memory location. When all PEs have responded, the deadstart node sends a response to `mppms` and then polls the IO_IFLAG register. At this time, the deadstart node scans the DMCA memory locations but it does not wait for a response from the PEs. This is because the PEs do not respond to the DMCA memory locations until after they receive the LOAD_ROUTE command.

When the LOSP input data available bit is set to a 1, the deadstart node reads another command data structure from the LOSP input buffer. The microprocessor rejects the command data structure when there are parity errors or premature disconnects.

When the command is LOAD_ROUTE and there are no other errors, the microprocessor instructs the input circuitry and PE support circuitry to load the data portion of the command data structure into local memory. This data contains the system route file. When there is an entry in the system route file that is designated for the deadstart node, the deadstart node transfers its portion of this information to its routing tag look-up table.

The other PEs within the system also need to receive the system route file. To indicate that this data is available for them to read, the deadstart node sets up the appropriate DMCA memory locations (indicated by a mask in the command packet).

Next, the deadstart node scans the DMCA memory locations until all appropriate PEs have responded that they have read the system route file. Once all PEs have responded, the deadstart node sends a response packet to `mppms` and polls the IO_IFLAG register.

NOTE: At this point the deadstart node has completed the system initialization. The remaining text describes how the deadstart node loads a diagnostic test, initiates the execution of the diagnostic test, and halts the execution of the diagnostic test.

When the LOSP input data available bit is set to a 1, the deadstart node reads another command data structure from the LOSP input buffer. When the command is LOAD_TEST and there are no other errors, the microprocessor instructs the input circuitry and PE support circuitry to load the data portion of the command data structure into local memory. This data contains the diagnostic code.

To indicate to the other PEs in the system that the diagnostic code is available for them to read, the deadstart node sets up the appropriate DMCA memory locations (indicated by a mask in the command packet).

Next, the deadstart node scans the DMCA memory locations until all appropriate PEs have responded that they have read the diagnostic code. Once all PEs have responded, the deadstart node sends a response packet to `mppms` and polls the IO_IFLAG register.

When the LOSP input data available bit is set to a 1, the deadstart node reads another command data structure from the LOSP input buffer. When the command is PE_RESUME and there are no other errors, the microprocessor instructs the input circuitry and PE support circuitry to load the data portion of the command data structure into local memory.

To indicate to the other PEs in the system that they should begin executing the diagnostic code, the deadstart node sets up the appropriate DMCA memory locations (indicated by a mask in the command packet).

Next, the deadstart node scans the DMCA memory locations until all appropriate PEs have responded that they are executing the diagnostic code. Once all PEs have responded, the deadstart node sends a response packet to `mppms` and polls the IO_IFLAG register.

When the LOSP input data available bit is set to a 1, the deadstart node reads another command data structure from the LOSP input buffer. When the command is PE_STOP and there are no other errors, the microprocessor instructs the input circuitry and PE support circuitry to load the data portion of the command data structure into local memory.

To indicate to the other PEs in the system that they should stop executing the diagnostic code, the deadstart node sets up the appropriate DMCA memory locations (indicated by a mask in the command packet).

Next, the deadstart node scans the DMCA memory locations until all appropriate PEs have responded that they have stopped executing the diagnostic code. Once all PEs have responded, the deadstart node sends a response packet to `mppms` and polls the IO_IFLAG register.

Initialization Entry Point

While the deadstart node executes the deadstart entry code, the microprocessors within the nondeadstart nodes enter the SADPAL code at the initialization entry point. This code enables the microprocessors to respond to their DMCA memory location and poll this DMCA memory location for the next command.

The next command that the nondeadstart nodes receive is the LOAD_ROUTE command. This command enables the nondeadstart nodes to read the system route file from the local memory of the deadstart node and to write the system route file into their routing tag look-up tables.

Once the nondeadstart nodes have loaded the system route file into their routing tag look-up tables, the nondeadstart nodes respond to their DMCA memory locations. After sending the response, the nondeadstart nodes poll their DMCA memory location for the next command.

NOTE: At this point the nondeadstart nodes have completed the system initialization. The remaining text describes how the nondeadstart nodes load a diagnostic test and initiate the execution of the diagnostic test.

To load a diagnostic test, the next command that the nondeadstart nodes receive is the LOAD_TEST command. This command enables the nondeadstart nodes to read the diagnostic code from the local memory of the deadstart node and to write the diagnostic code into their local memory.

Once the nondeadstart nodes have read the diagnostic code, the nondeadstart nodes respond to their DMCA memory location. After sending the response, the nondeadstart nodes poll their DMCA memory location for the next command.

To initiate the execution of the diagnostic test, the next command that the nondeadstart nodes receive is the PE_RESUME command. This command enables the nondeadstart nodes to read the diagnostic code out of local memory, write it into the instruction cache of the microprocessors, and begin executing the diagnostic code.

Once the nondeadstart nodes are executing the diagnostic code, the nondeadstart nodes respond to their DMCA memory location. After sending the response, the nondeadstart nodes poll their DMCA memory location for the next command.

Table 5-2. System Initialization

<table>
<tr><th>mppms</th><th>Deadstart Node</th><th>Nondeadstart Node</th></tr>
<tr><td>Asserts a Master Clear signal on the LOSP channel.</td><td colspan="2"></td></tr>
<tr><td></td><td>Receives Master Clear signal and generates a Global Reset Heartbeat signal (4 CP).
Fans this signal out to the other processing elements (PEs) in the CRAY T3D system and to the AR option of its own support circuitry.</td><td></td></tr>
<tr><td></td><td colspan="2">The AR option in the support circuitry of each PE receives the Global Reset Heartbeat signal.
The AR option sends a PE Reset signal to the AM option, which causes the PE to reset.
The AM option sets the Reset_l to a 1 and sends it to the microprocessor. This resets the microprocessor.
The AR option sets bit 5 (node enable bit) of the SCR to 0. This resets the network interface.
When the Reset_l signal is set to 0, the microprocessor automatically reads boot code from the SROM. This code is loaded into the instruction cache of the microprocessor and executed.</td></tr>
<tr><td></td><td colspan="2">SROM Code, Segment 1:
Initializes the registers internal to the microprocessor and the PE support circuitry.
<b>Note:</b> The even numbered PE initializes the shared logic.
Determine whether the PE is in the deadstart node.</td></tr>
</table>

Table 5-2. System Initialization (continued)

`mppms`	Deadstart Node	Nondeadstart Node
	SROM Code, Segment 2: Returns a response packet to `mppms` indicating it is ready for LOSP data. Polls the IO_IFLAG register for LOSP input data available.	SROM Code, Segment 3: Waits for a configuration message to come across the barrier channel.
Sends the LOAD_PAL command packet and PAL code.		
	SROM Code, Segment 2: Receives the LOAD_PAL command packet. Writes SADPAL code into local memory. Sends response packet to `mppms`. Jumps to the deadstart node entry point of the SADPAL code. Polls the IO_IFLAG register for LOSP input data available.	SROM Code, Segment 3: Waits for a configuration message to come across the barrier channel.
Sends the LOAD_ROUTE command packet and deadstart node to PE routes.		
	SADPAL Code: Receives the LOAD_ROUTE command packet. Writes the Deadstart node to PE routes into local memory. Sends a response packet to `mppms`. Writes the deadstart node to PE routes to the deadstart node’s routing tag look-up table. Polls the IO_IFLAG register for LOSP input data available.	SROM Code, Segment 3: Waits for a configuration message to come across the barrier channel.

Table 5-2. System Initialization (continued)

`mppms`	Deadstart Node	Nondeadstart Node
Sends the LOAD_CONFIG command packet and configuration messages.		
	SADPAL Code: Receives the LOAD_CNFIG command packet. Writes configuration messages into local memory. Sends a response packet to `mppms`. Transmits the configuration messages over the barrier channel.	SROM Code, Segment 3: Polls the barrier channel for a configuration message until it receives a message that is addressed to the local PE. Writes the configuration message into local memory and the data cache. Reads the configuration out of local memory or the data cache and writes it into the appropriate registers. Reads the SADPAL code from the local memory of the deadstart node. Writes the SADPAL to local memory. Jumps to the initialization entry point in the SADPAL code.
	SADPAL Code: Scans DMCA. Sends a response packet to `mppms`. Polls the IO_IFLAG register for LOSP input data available.	SADPAL Code: Polls DMCA.

Table 5-2. System Initialization (continued)

`mppms`	Deadstart Node	Nondeadstart Node
Sends the LOAD_ROUTE command packet and system routes.		
	SADPAL Code: Receives the LOAD_ROUTE command packet. Writes system routes into local memory. Sets the DMCA for the appropriate PEs (indicated by a mask in the command packet) to indicate the load route. Scans DMCA until the appropriate PEs have responded. Sends a response packet to `mppms`. Polls the IO_IFLAG register for LOSP input data available.	SADPAL Code: Detects load route in the DMCA. Reads system route file from the local memory of the deadstart node and writes the system route file into the routing tag look-up table. Responds to DMCA. Polls DMCA.
NOTE: At this point, the system initialization is complete. The remainder of this table indicates how a diagnostic test is loaded into the PEs and how the execution of the diagnostic test is started.		

Table 5-2. System Initialization (continued)

`mppms`	Deadstart Node	Nondeadstart Node
Sends the LOAD_TEST command packet and diagnostic test code.		
	SADPAL Code: Receives the LOAD_TEST command packet. Writes the diagnostic test code into local memory. Sets the DMCA for the appropriate PEs to indicate load test. Scans DMCA until the appropriate PEs have responded. Sends a response packet to `mppms`. Polls the IO_IFLAG register for LOSP input data available.	SADPAL Code: Detects load test in the DMCA. Reads diagnostic test code from the local memory of the deadstart node and writes it into its local memory. Responds to DMCA. Polls DMCA.

Table 5-2. System Initialization (continued)

mppms	Deadstart Node	Nondeadstart Node
Sends the PE_RESUME command packet.		
	SADPAL Code: Receives the PE_RESUME command packet. Writes the command packet into local memory. Sets the DMCA for the appropriate PEs to indicate PE resume. Scans DMCA until the appropriate PEs have responded. Sends a response packet to mppms. Polls the IO_IFLAG register for LOSP input data available.	SADPAL Code: Detects PE resume in the DMCA. Reads diagnostic test code from local memory and writes it into the instruction cache of the microprocessor. Executes the diagnostic test code. Responds to DMCA. Polls DMCA.

6 ADDRESSING

Because the CRAY T3D system has a physically distributed, logically shared memory, the address generated by the microprocessor in a PE cannot be used to directly reference physical memory. This section describes the components and the process used to reference system data and registers in the CRAY T3D system.

Functional Description

The PEs within the CRAY T3D system can be identified using a virtual PE number, a logical PE number, and a physical PE number. The CRAY T3D system interprets the virtual PE as a PE within a user partition. A logical PE is a PE that is configured into the system (this excludes disabled spare nodes, I/O nodes, and faulty nodes). A physical PE is any PE within the system.

Addresses used in the CRAY T3D system can also be identified as virtual, logical, and physical. Each type of address contains information that the CRAY T3D system uses to address local memory, reference memory-mapped registers, and route packets through the network.

PE Numbering

Depending on the context, a PE is identified by one of three types of numbers: a physical PE number, a logical PE number, or a virtual PE number. All three types of numbers consist of a PE bit and a field containing the node number or node coordinates.

Physical PE Number

Every PE in the CRAY T3D system is assigned a unique number that indicates the PE. This number is the physical PE number.

The support circuitry in each PE contains a register called the physical PE register. When a circuit board is placed in the system cabinet, hardware automatically sets the bits of the physical PE register to a unique number.

The physical PE number contains two parts: the physical node number and the PE bit. The physical node number indicates which physical node the PE is located in. The PE bit indicates PE 0 or PE 1 in that node. More information on the exact bit format and use of the physical PE number is provided in “Register Mapping” in this section.

Logical PE Number

Not all of the physical PEs in a CRAY T3D system are part of the logical configuration of a CRAY T3D system. For example, a 512-PE CRAY T3D system contains 520 physical PEs (not including PEs in the I/O gateways). Of these 520 PEs, 512 PEs are used in the logical system and 8 PEs (in 4 spare PE nodes) are used as spare PEs.

Each physical PE used in a logical system is assigned a unique logical PE number. The logical PE number identifies the location of a PE in the logical system of nodes.

The logical PE number contains two parts: the logical node number and the PE bit. The logical node number indicates which logical node the PE resides in (processing element node, input node, or output node). The PE bit indicates whether the PE is PE 0 or PE 1. Figure 6-1 shows a simplified format of the logical PE number. More information on the exact bit format of the logical PE number is provided in “Register Mapping” in this section.

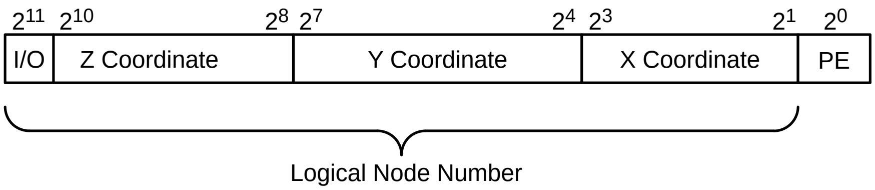


Figure 6-1. Logical PE number

The logical nodes form a three-dimensional matrix of nodes. For example, Figure 6-2 shows the logical PE nodes for a 128-PE CRAY T3D system. Although the system actually contains 68 physical PE nodes, only 64 of the nodes are used in the logical system. The remaining 4 spare physical nodes are physically connected to the interconnect network but are not given logical node numbers.

This type of configuration enables a spare node to logically replace a failing node. When this occurs, the spare node obtains a logical number and the failing node does not receive a new logical node number.

For example, when logical node Z=0, Y=2, X=3 fails to operate properly, the physical node assigned to this number may be removed from the logical system. A spare node is then assigned the logical node number Z=0, Y=2, X=3 and the failing node does not receive a logical node number.

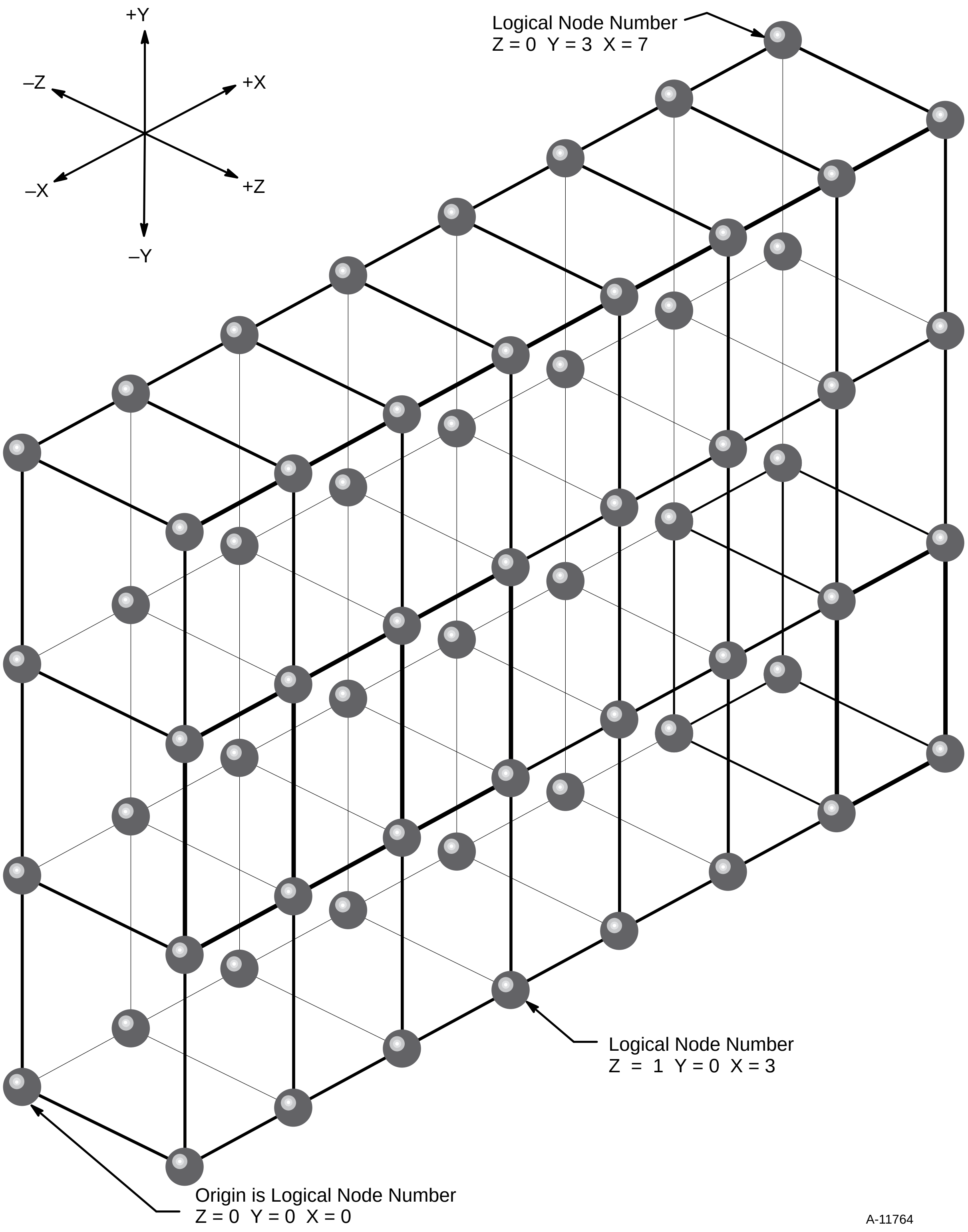


Figure 6-2. Logical Node Numbers

Virtual PE Number

When an MPP application initiates, the support software running on the host system determines the resources needed for the application and creates a partition for the application to run in. A partition is a group of PEs and a portion of the barrier synchronization resources that are assigned to one application. (More information on the barrier synchronization is provided in Section 9, “Barrier Synchronization.”) The application uses virtual PE numbers to reference the PEs in a partition.

There are two types of partitions: an operating system partition and a hardware partition. In an operating system partition, when the application transfers data between PEs, the operating system must be involved with the transfer. (When multiple references to the same PE are made, the operating system is involved with the first transfer to the remote PE. Subsequent transfers to the same PE do not involve the operating system.) The operating system converts the virtual PE numbers used by the application into logical PE numbers.

In a hardware partition, when the application transfers data between PEs, the operating system is not involved with the transfer. Hardware in each PE node converts the virtual PE numbers used by the application into logical PE numbers.

The virtual PE number contains two parts: the virtual node number and the PE bit. The virtual node number ranges from 1 to 10 bits and indicates which processing element node in a hardware partition the PE resides in. The PE bit indicates when the PE is PE 0 or PE 1 (refer to Figure 6-3).

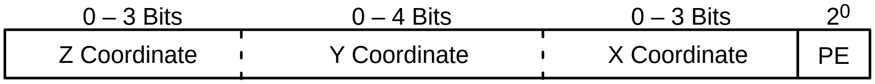


Figure 6-3. Virtual PE Number

The virtual node number has 0 to 3 bits assigned to the X dimension, 0 to 4 bits assigned to the Y dimension, and 0 to 3 bits assigned to the Z dimension. By assigning bits of the virtual node number to the appropriate dimensions, software arranges the virtual nodes into one of several shapes. For example, a three-bit virtual node number indicates there are eight nodes in the hardware partition. These nodes may be arranged in 1 of 10 shapes.

Table 6-1 lists the possible node shapes for a three-bit virtual node number. For each shape, the number of nodes in each dimension is limited to powers of two (1, 2, 4, 8, 16, etc.).

Table 6-1. Eight-node Partition Shapes

Eight-node Array Shaped in X, Y, Z	Description
8, 1, 1	One-dimensional array in the X dimension
1, 8, 1	One-dimensional array in the Y dimension
1, 1, 8	One-dimensional array in the Z dimension
1, 2, 4	Two-dimensional array in the Y-Z plane
1, 4, 2	Two-dimensional array in the Y-Z plane
2, 1, 4	Two-dimensional array in the X-Z plane
4, 1, 2	Two-dimensional array in the X-Z plane
2, 4, 1	Two-dimensional array in the X-Y plane
4, 2, 1	Two-dimensional array in the X-Y plane
2, 2, 2	Three-dimensional array

Figure 6-4 shows three of the eight-node partition shapes in a 128-PE CRAY T3D system.

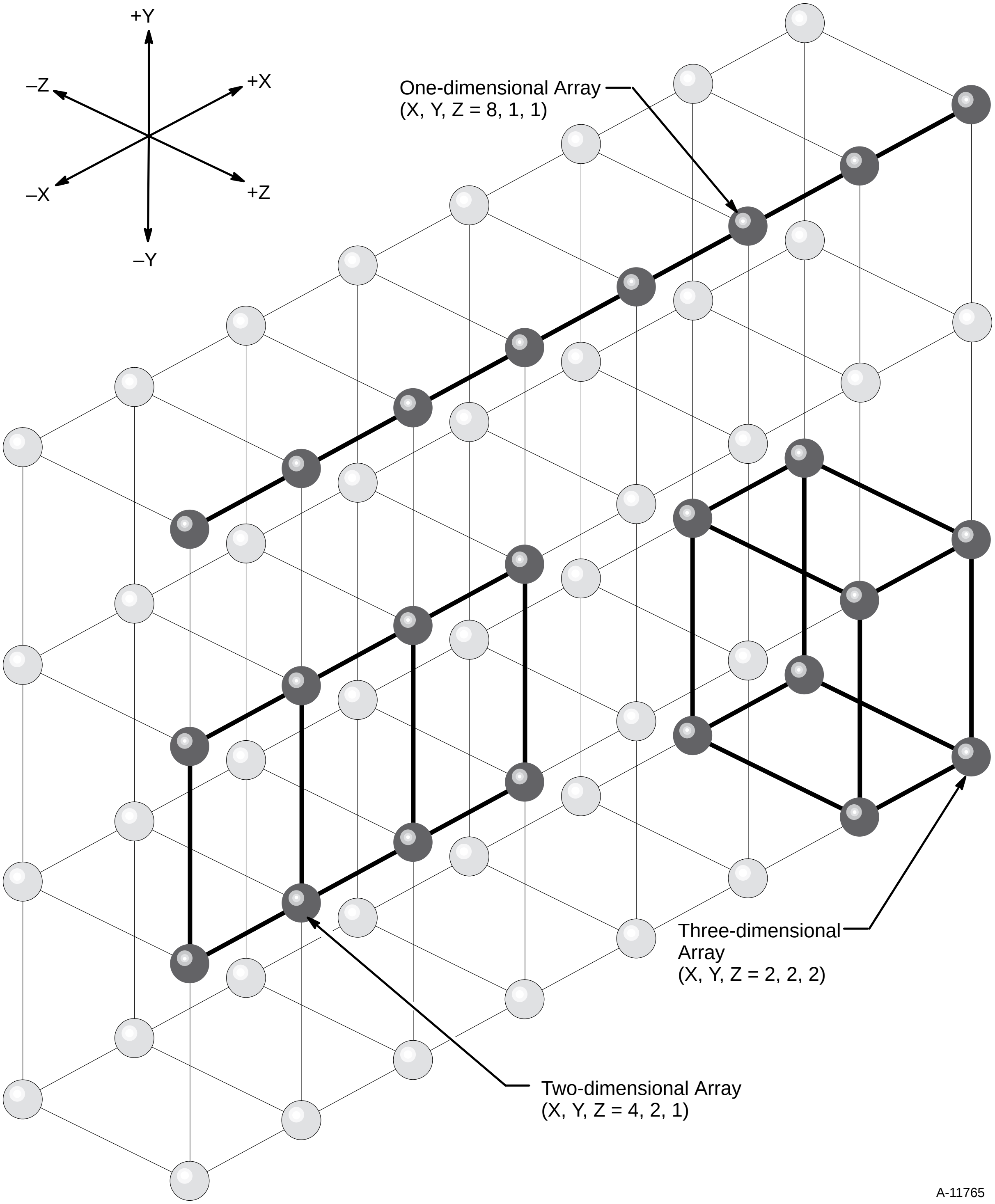


Figure 6-4. Three 8-node Partition Shapes in a 128-PE System

As an example of virtual PE numbers, Figure 6-5 shows a two-dimensional, eight-node partition that contains 16 PEs. Each node in the partition is referred to by the three-bit virtual node numbers shown in Figure 6-5.

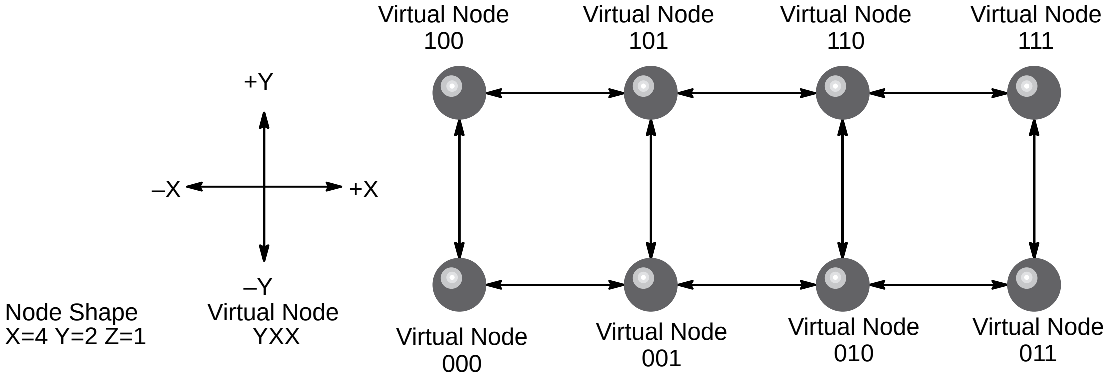


Figure 6-5. Virtual Node Numbers of a Two-dimensional Array

This two-dimensional array of eight nodes may actually correspond to one of many eight-node two-dimensional arrays in the logical system. For example, Figure 6-6 shows two examples of how this two-dimensional array may be placed in the logical system of nodes in a 128-PE CRAY T3D system.

A virtual node number does not always correspond to the same logical node number. For example, Figure 6-6 shows how virtual node Y=1 X=2 from Figure 6-5 may correspond to either logical node number Z=1 Y=2 X=2 or logical node number Z=1 Y=3 X=6.

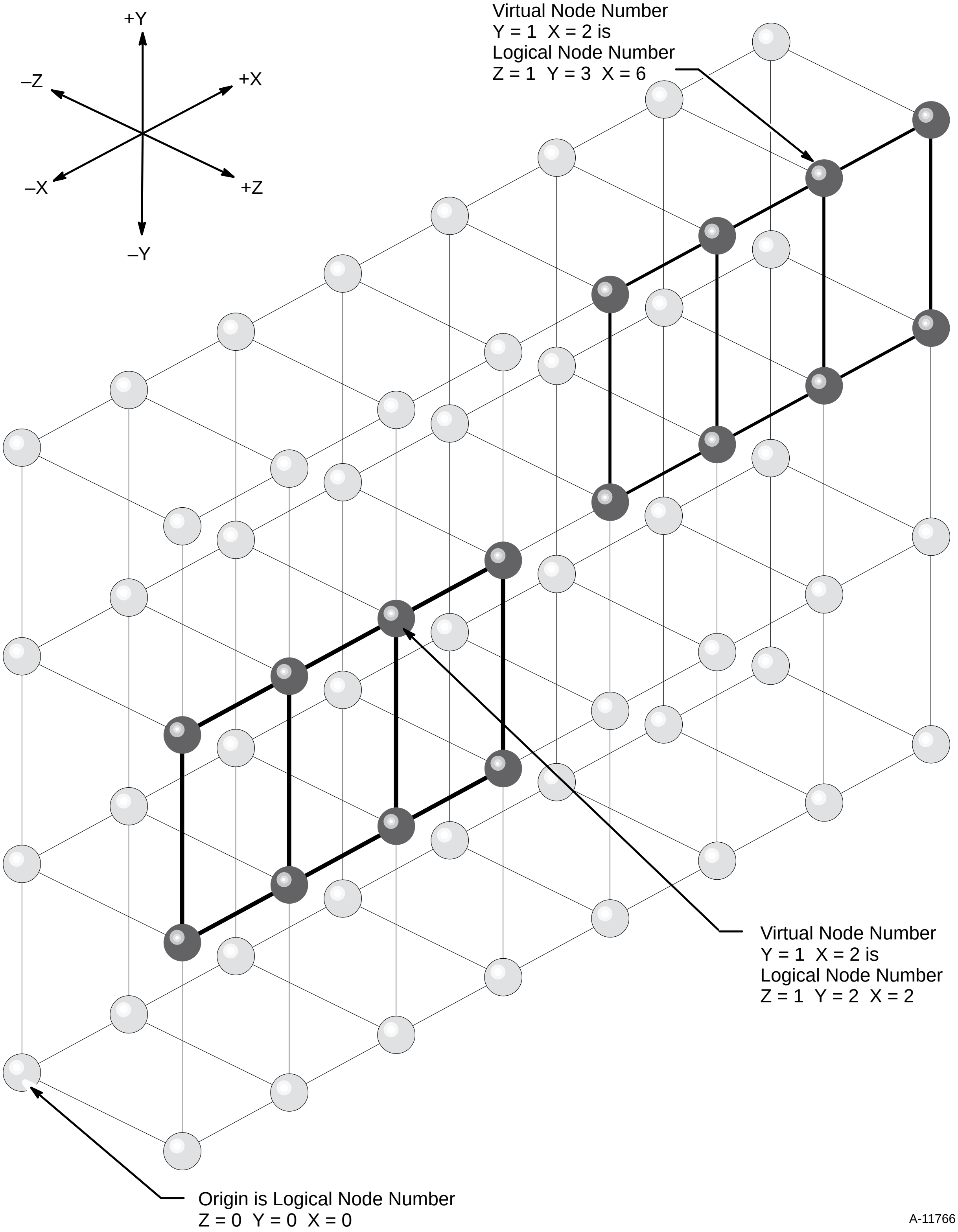


Figure 6-6. Virtual and Logical Node Numbers

Microprocessor Address Interpretation

Application programs running in each PE generate a virtual address that the hardware in each microprocessor converts into a partial physical address. The following subsections describe both the virtual address and the partial physical address.

Virtual Address

The virtual address is a byte oriented address generated by a program compiler. The virtual address may contain three parts: a segment, a virtual index, and a virtual address offset. Figure 6-7 shows the bit format of the virtual address. Although the length is 64 bits, only 41 bits of the address are actually used.

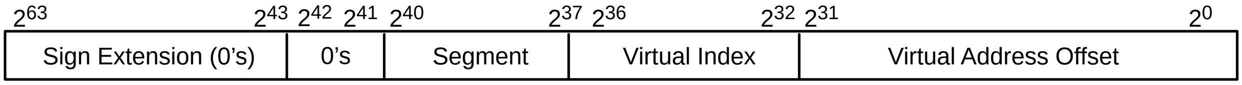


Figure 6-7. Virtual Address Format

The microprocessor operating system software specifies how the microprocessor hardware maps the virtual address into the partial physical address. Because this process is defined by software, the current format of the virtual address may be different than the format shown in Figure 6-7. This sample virtual address is included as an example of how the microprocessor interprets a virtual address.

Segment

The segment is a 4-bit number that divides the virtual address space into as many as 16 segments. Each segment identifies a unique type of memory. For example, when the segment number is set to 4, information may be written to or read from memory-mapped registers in the support circuitry of a PE or in other components of a PE node (refer to Figure 6-8).

Some segments are designated as private and some segments are designated as shared. A private segment refers to information that is stored in the local memory of a PE and is used only by the microprocessor in the PE. Only the microprocessor in the PE can write to or read from a segment designated as private.

A shared segment refers to information that may be stored in any PE in the system and may be used by any microprocessor in a partition. The shared segments are located in the same area of local memory in all of the PEs within a partition.

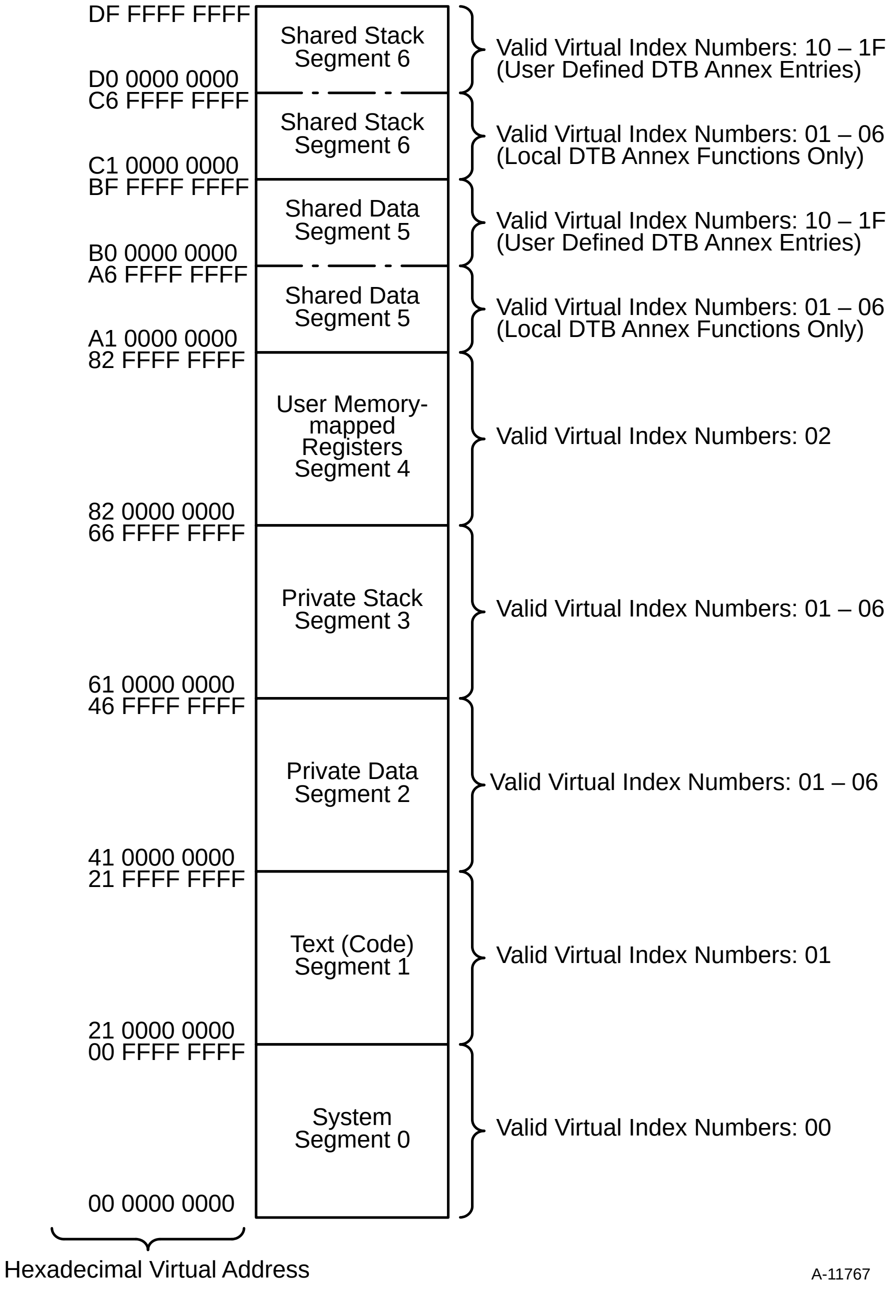

DF FFFF FFFF
Shared Stack Segment 6
Valid Virtual Index Numbers: 10 – 1F (User Defined DTB Annex Entries)
D0 0000 0000
C6 FFFF FFFF
Shared Stack Segment 6
Valid Virtual Index Numbers: 01 – 06 (Local DTB Annex Functions Only)
C1 0000 0000
BF FFFF FFFF
Shared Data Segment 5
Valid Virtual Index Numbers: 10 – 1F (User Defined DTB Annex Entries)
B0 0000 0000
A6 FFFF FFFF
Shared Data Segment 5
Valid Virtual Index Numbers: 01 – 06 (Local DTB Annex Functions Only)
A1 0000 0000
82 FFFF FFFF
User Memory-mapped Registers Segment 4
Valid Virtual Index Numbers: 02
82 0000 0000
66 FFFF FFFF
Private Stack Segment 3
Valid Virtual Index Numbers: 01 – 06
61 0000 0000
46 FFFF FFFF
Private Data Segment 2
Valid Virtual Index Numbers: 01 – 06
41 0000 0000
21 FFFF FFFF
Text (Code) Segment 1
Valid Virtual Index Numbers: 01
21 0000 0000
00 FFFF FFFF
System Segment 0
Valid Virtual Index Numbers: 00
00 0000 0000
Hexadecimal Virtual Address
A-11767

Figure 6-8. Virtual Address Space

Virtual Index

The virtual index is a 5-bit number that references one of the entries in the data translation buffer (DTB) annex. The DTB annex is a 32-entry table external to the microprocessor (in the support circuitry of a PE) that contains information used to expand the addressing range of the microprocessor to include all of system memory. More information on the DTB annex is provided later in this section.

Virtual Address Offset

The virtual address offset is a byte oriented address that points to a byte within a segment of the virtual address space. The lower 32 bits of the virtual address define the offset. The virtual address offset may range from 0 to 4 giga-bytes.

Partial Physical Address

After receiving the virtual address from an application program, hardware in the microprocessor converts the virtual address into a partial physical address. The partial physical address is an address that the microprocessor places on the address pins of the microprocessor.

In the CRAY T3D system, the address pins of the microprocessor do not directly address a physical memory. Instead, support circuitry in the PE interprets the partial physical address and routes data between the microprocessor and either local memory, memory-mapped registers, or memory in a remote PE.

Figure 6-9 shows how a microprocessor maps a virtual address into a partial physical address.

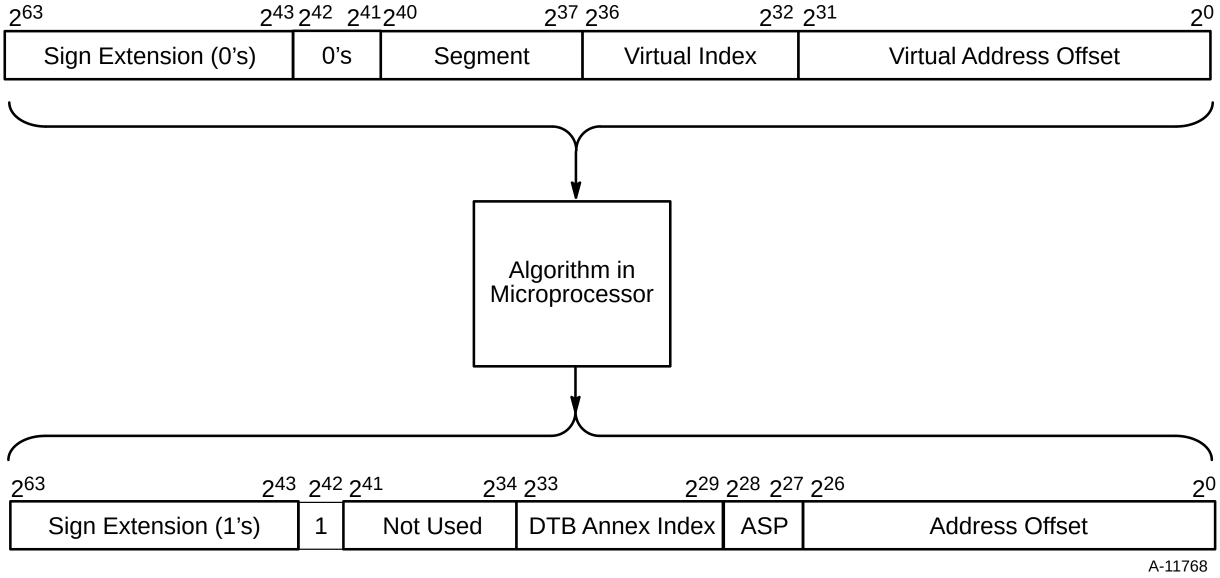


Figure 6-9. Virtual to Partial Physical Address Conversion

The partial physical address contains three parts: the DTB annex index, the address space partition (ASP), and the address offset. Bits 34 through 63 of the partial physical address are not placed on output pins of the microprocessor and are not used.

DTB Annex Index

The DTB annex index is a 5-bit number that references one of the 32 entries in the physical DTB annex (refer to Figure 6-10).

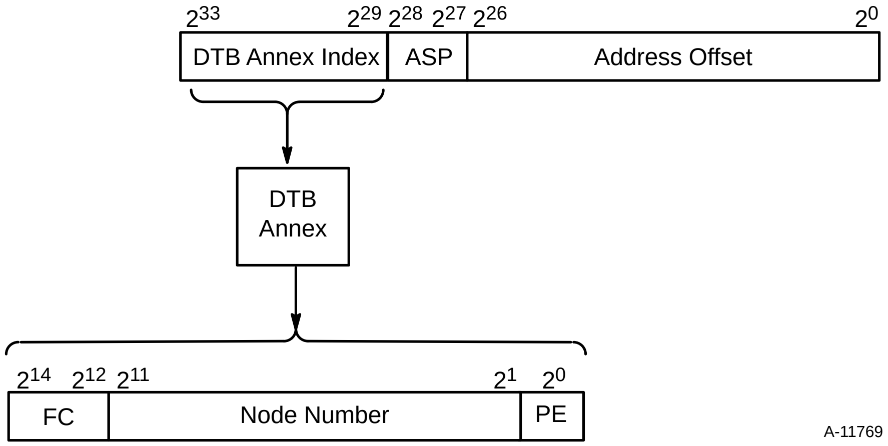


Figure 6-10. DTB Annex

Each entry in the DTB annex contains two parts: a PE number and a memory function code (FC). Figure 6-11 shows the format of a DTB annex entry.

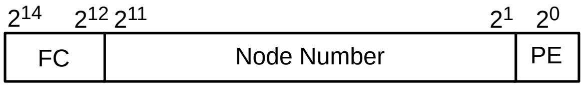


Figure 6-11. DTB Annex Entry

The PE number contains a node number and PE bit. The node number can be a virtual node number or a logical node number. The PE bit indicates the PE in which the memory function will occur.

The function code is a 3-bit number that indicates the memory reference function that the PE will perform. Table 6-2 lists the function codes and the corresponding memory functions. More information on each memory function is provided in Section 7, “Operations.”

Table 6-2. Memory Function Codes

Function Code Bits 14 – 12	Memory Function
000	Write or noncacheable read
001	Write or noncacheable atomic swap
010	Fetch-and-increment write or read
011	Reserved (may cause an error interrupt)
100	Write or cached read
101	Write or cached atomic swap
110	Write or cached read ahead
111	Message or cached read

As stated, each entry in the DTB annex describes a memory reference function that the PE will perform and indicates the PE in which the memory function will occur. Two bits of the system control register (SCR) determine whether entries 1 through 31 contain virtual or logical PE numbers (entry 0 is always interpreted as a logical PE number).

When bit 0 or bit 1 of the SCR is set to 1, the user or the operating system may write to the corresponding entries of the DTB annex. In this case, the support circuitry interprets the entries of the DTB annex as virtual PE numbers and function codes. The support circuitry also reads bit 11 of the PE number in the DTB annex entry as a 0.

NOTE: For bit 0, the corresponding entries of the DTB annex are entries 1 through 15. For bit 1, the corresponding entries are 16 through 31.

When bit 0 or bit 1 of the SCR is set to 0, the corresponding entries of the DTB annex may not be written to. In this case, the support circuitry interprets the entries of the DTB annex as logical PE numbers and function codes. The support circuitry also reads bit 11 of the PE number in the DTB annex entry as the value it is set to in the DTB annex entry.

The support circuitry always interprets entry 0 of the DTB annex as a logical PE number and a function code that is defined by the operating system. In order for the support circuitry to read the correct value of bit 11 of DTB annex entry 0, bit 0 of the SCR must be set to 0. (This only affects PE numbers where bit 11 of the PE number may be set to 1.)

Table 6-3 shows the entries for a sample DTB annex. In this example, bit 1 of the SCR is set to 1 and bit 0 is set to 0. Entry 0 is reserved for use by the operating system.

Table 6-3. Sample DTB Annex Entry Assignments

Virtual Index	DTB Annex Function	Function Code	PE Number	Valid Segments	User Write Permission
0	System use	Not applicable	Not applicable (NA)	NA	No
1	Write or cached read	100	Local logical PE number	1-6	No
2	Write or read	000	Local logical PE number	2-6	No
3	Atomic swap	001	Local logical PE number	2, 3, 5, 6	No
4	Fetch and increment	010	Local logical PE number	2, 3, 5, 6	No
5	Cached atomic swap	101	Local logical PE number	2, 3, 5, 6	No
6	Message write (loopback)	111	Local logical PE number	2, 3, 5, 6	No
7	Reserved for future use	Reserved	Reserved for future use	Reserved	No
8 – 15	Reserved for future use	Reserved	Reserved for future use	Reserved	No
16 – 31	User defined	User defined	Virtual PE number	5, 6	Yes

After receiving the DTB annex index from the microprocessor, the DTB annex sends the contents of the appropriate DTB annex entry to the PE support circuitry. The support circuitry then interprets the DTB annex information to determine what type of memory function to perform and where to perform the memory function.

Address Space Partition Bits

The address space partition bits divide the address space for bits 0 through 26 of the address offset into three regions (refer to Table 6-4). Bit 28 divides the partial physical address space into local memory and memory-mapped register space. Bit 27 further divides the local memory space into local memory with SECDED protection and local memory without SECDED protection. Figure 6-12 shows a map of the partial physical address space for a PE.

Table 6-4. Address Space Partition Bit Definitions

ASP Bits 28 and 27	Region
00	Local memory with SECDED protection
01	Local memory without SECDED protection
10	Memory-mapped register space
11	

The address space for local memory with SECDED protection provides up to 128 Mbytes (16 Mwords) of memory; however, not all of the address space is used. For example, CRAY T3D systems with local memory sizes of 2 Mwords (16 Mbytes) per PE use only the 23 least significant bits of the address offset. The 16 Mwords of address space are provided for future expansion of system memory.

The address space for local memory without SECDED protection provides up to 16 Mwords of memory; however, not all of this address space is used. The information stored in this section of local memory address space is from the headers of messages. More information on messages is provided in Section 7, “Operations.”

The address space for memory-mapped registers contains the addresses for information that is stored or read from memory-mapped registers. Memory mapped-registers are registers that are located in the support circuitry of a PE or other components of a PE node but are addressed by the local microprocessor as if they were located in a local memory.

For example, when the microprocessor stores information in a register that controls the block transfer engine (BLT), the microprocessor performs a write operation to the address that points to the BLT register. After receiving the data and address information, the support circuitry in the PE routes the data to the appropriate register instead of storing the information in local DRAM memory.

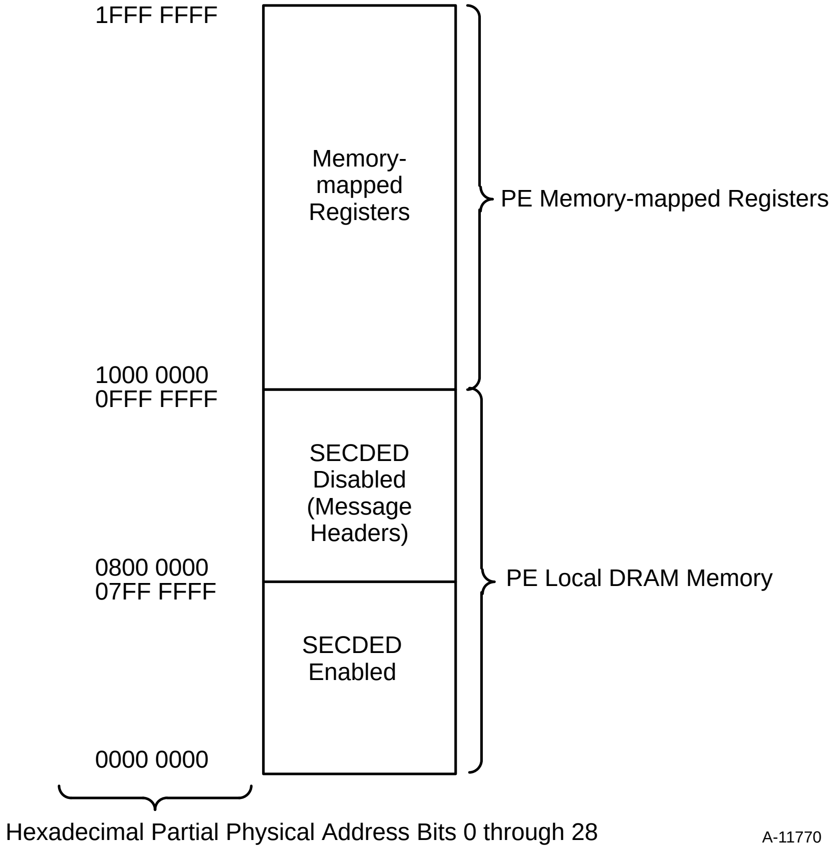


Figure 6-12. Partial Physical Address Space

Address Offset

The address offset is a byte oriented address that points to a byte in the local memory of a PE; however, the support circuitry in the PE may not use the last 2 bits of the address offset portion of the partial physical address.

The support circuitry interprets bits 2 through 26 of the address offset as a halfword-oriented address that points to a 32-bit halfword in the local memory of a PE. Because the support circuitry may use only bits 2 through 26 of the address offset, the term “address offset” also refers to a 25-bit address offset that references a 32-bit halfword in local memory.

PE Support Circuitry Address Interpretation

When the microprocessor requests that a memory function be performed, the microprocessor and DTB annex present the support circuitry with address information. The support circuitry may interpret this information as an address for a virtual PE or as an address for a logical PE.

Virtual PE Address Interpretation

When the entries of the DTB annex may be written to, the support circuitry interprets the address information as a virtual PE address. The support circuitry receives four types of information from the microprocessor and DTB annex: a memory function code, a virtual PE number, the ASP bits, and an address offset (refer to Figure 6-13).

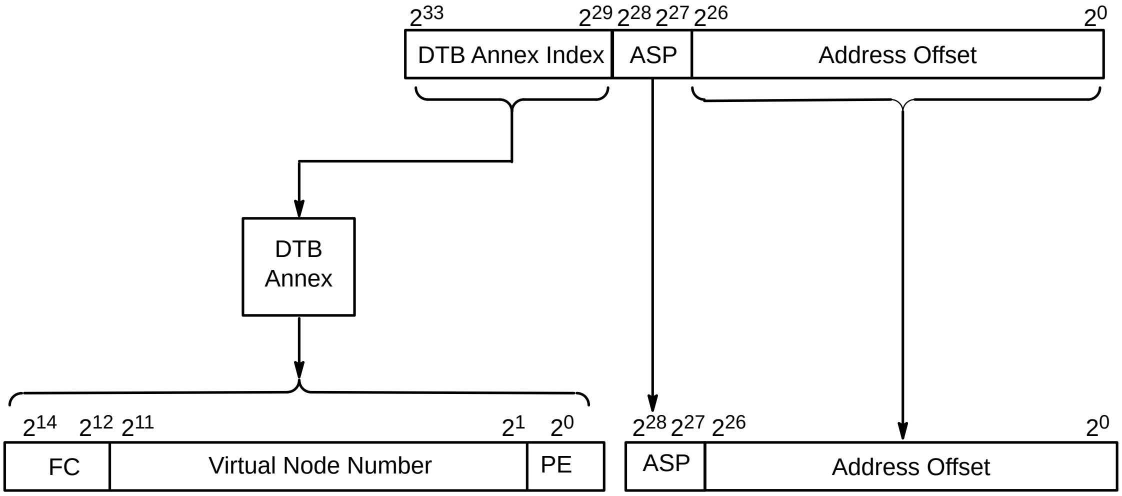


Figure 6-13. Virtual PE Address Information

After receiving the virtual PE number from the DTB annex, the support circuitry compares the value of the virtual PE number to the value stored in the virtual PE (V_WHOAMI) register. The V_WHOAMI register contains the virtual PE number and is loaded by the operating system when the partitioning of the CRAY T3D system changes.

When the virtual PE number from the DTB annex matches the PE number stored in the V_WHOAMI register, the microprocessor is requesting that a memory function be performed in the local PE. After interpreting the function code bits that select which memory function to perform, the support circuitry transfers information between the microprocessor and local memory or between the microprocessor and a memory-mapped register (the location is specified by the ASP bits).

When the virtual PE number from the DTB annex does not match the PE number stored in the V_WHOAMI register, the microprocessor is requesting that a memory function be performed in a remote PE. When the microprocessor requests that a memory function be performed in a remote PE, the support circuitry checks the value of the virtual PE number to ensure that it is less than the highest numbered virtual PE in the hardware partition.

The support circuitry checks the value of the virtual PE number it received from the DTB annex by comparing the PE number to the virtual PE range mask stored in the virtual PE range mask (VRT_RG) register. This register is loaded by the operating system when the partitioning of the CRAY T3D system changes. Bit 0 of the VRT_RG register is not used. Bits 1 through 10 of the VRT_RG register contain a contiguous field of 0’s in the legal bit positions for the size of the user partition, and the remaining bits are set to 1’s (refer to Table 6-5).

Table 6-5. Virtual PE Range Mask Values

2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Number of Nodes in Partition	Number of PEs in Partition
1	1	1	1	1	1	1	1	1	1	x	1	2
1	1	1	1	1	1	1	1	1	0	x	2	4
1	1	1	1	1	1	1	1	0	0	x	4	8
1	1	1	1	1	1	1	0	0	0	x	8	16
1	1	1	1	1	1	0	0	0	0	x	16	32
1	1	1	1	1	0	0	0	0	0	x	32	64
1	1	1	1	0	0	0	0	0	0	x	64	128
1	1	1	0	0	0	0	0	0	0	x	128	256
1	1	0	0	0	0	0	0	0	0	x	256	512
1	0	0	0	0	0	0	0	0	0	x	512	1,024
0	0	0	0	0	0	0	0	0	0	x	1,024	2,048

When the same bit of the virtual PE number and the virtual PE range mask are set to 1 (logical AND function), a virtual PE range error occurs. For more information on the virtual PE number range error, refer to “System Status Register” in Section 10, “Control and Status.”

After checking the value of the virtual PE number, the support circuitry in the PE sends address and control information to the network interface. The network interface uses this information to create packets.

Logical PE Address Interpretation

When the entries of the DTB annex may not be written to, the support circuitry interprets the address information as a logical PE address. The support circuitry receives four types of information from the microprocessor and DTB annex: a memory function code, a logical PE number, the ASP bits, and an address offset (refer to Figure 6-14).

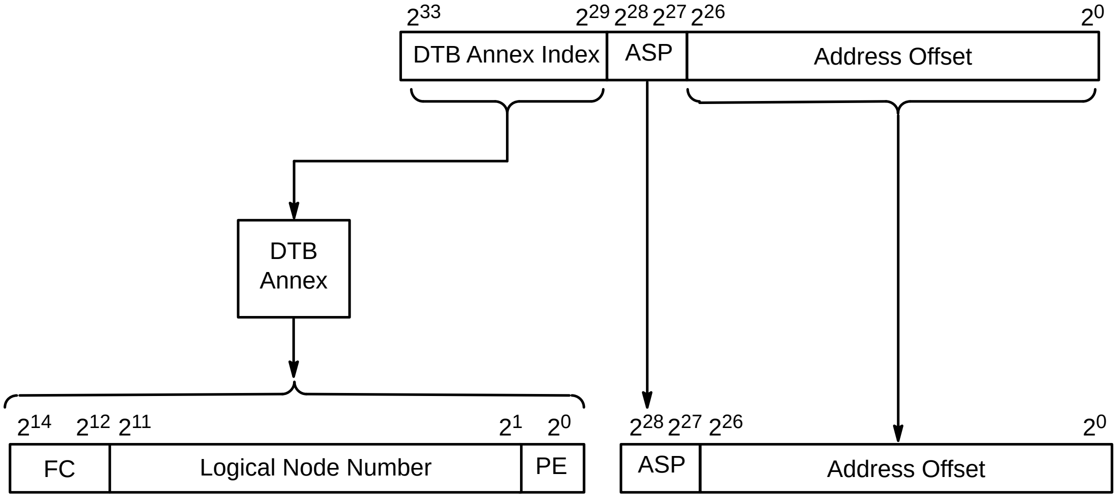


Figure 6-14. Logical PE Address Information

After receiving the logical PE number from the DTB annex, the support circuitry compares the value of the logical PE number to the value stored in the logical PE (L_WHOAMI) register. The L_WHOAMI register contains the logical PE number of the PE and is loaded by the operating system when the CRAY T3D system is initialized.

When the logical PE number from the DTB annex matches the PE number stored in the L_WHOAMI register, the microprocessor is requesting that a memory function be performed in the local PE. After interpreting the function code bits that select which memory function to perform, the support circuitry transfers information between the microprocessor and local memory or between the microprocessor and a memory-mapped register (the location is specified by the ASP bits).

When the logical PE number from the DTB annex does not match the PE number stored in the L_WHOAMI register, the microprocessor is requesting that a memory function be performed in a remote PE. When this occurs, the support circuitry in the PE does not perform a range check on the logical PE number. The support circuitry sends address and control information directly to the network interface for use in a packet.

Network Interface Address Interpretation

The network interface receives address information and data from PE 0, PE 1, or the BLT in a processing element node (refer to Figure 6-15). The address information may be virtual PE address information or logical PE address information. When the information is virtual PE address information, the network interface must convert the virtual PE information into logical PE information before using the information to create a packet.

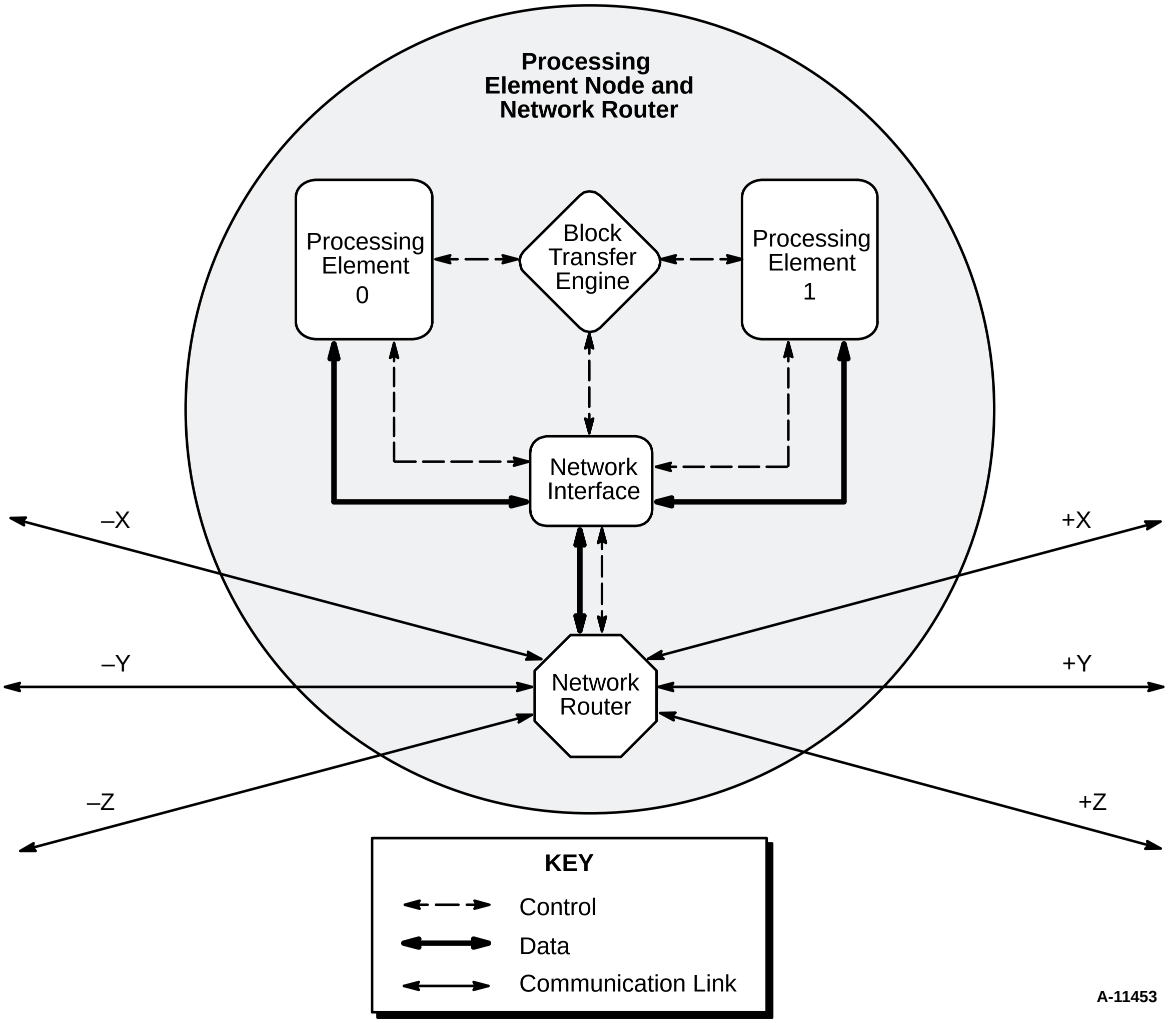


Figure 6-15. Network Interface Connections

Converting a Virtual PE Number Into a Logical PE Number

As was shown in Figure 6-6, the virtual PE number may correspond to different logical PE numbers. Because of this characteristic, the network interface cannot directly use a virtual PE number to determine which logical PE a memory function will be directed to. The network interface must convert the virtual PE number into a corresponding logical PE number.

For example, Figure 6-17 shows three 8-node partitions (partitions A, B, and C) in a 128-PE CRAY T3D system. In each partition, a virtual PE number refers to a PE in the partition. The virtual PE number contains a virtual node number and PE bit. The virtual node number in all three partitions is a 3-bit number.

All 3 bits of the virtual node number for partition A are assigned to the X dimension (refer again to Figure 6-17). In partition B, bits 0 and 1 of the virtual node number are assigned to the X dimension and bit 2 is assigned to the Y dimension. In partition C, bit 0 of the virtual node number is assigned to the X dimension, bit 1 is assigned to the Y dimension, and bit 2 is assigned to the Z dimension.

The network interface uses a three-step process to convert the virtual PE number into a logical PE number. First, the network interface reformats the virtual node number (which may be a 1- to 10-bit number) into a 10-bit virtual node number (refer to Figure 6-16).

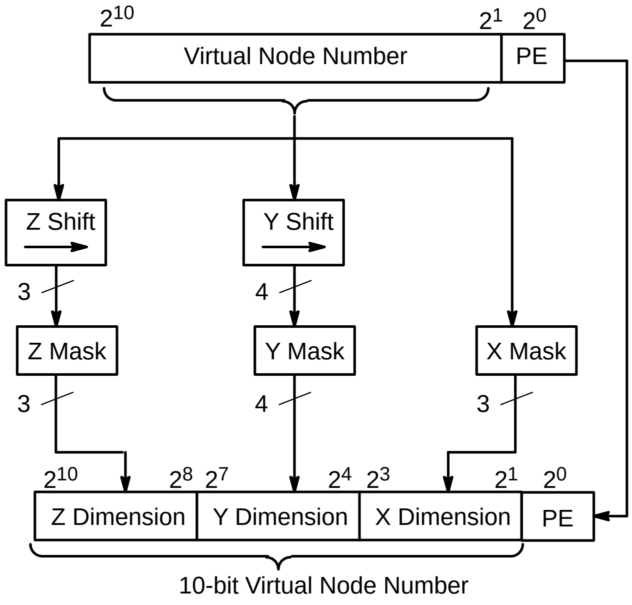


Figure 6-16. Reformatting a Virtual Node Number to a 10-bit Virtual Node Number

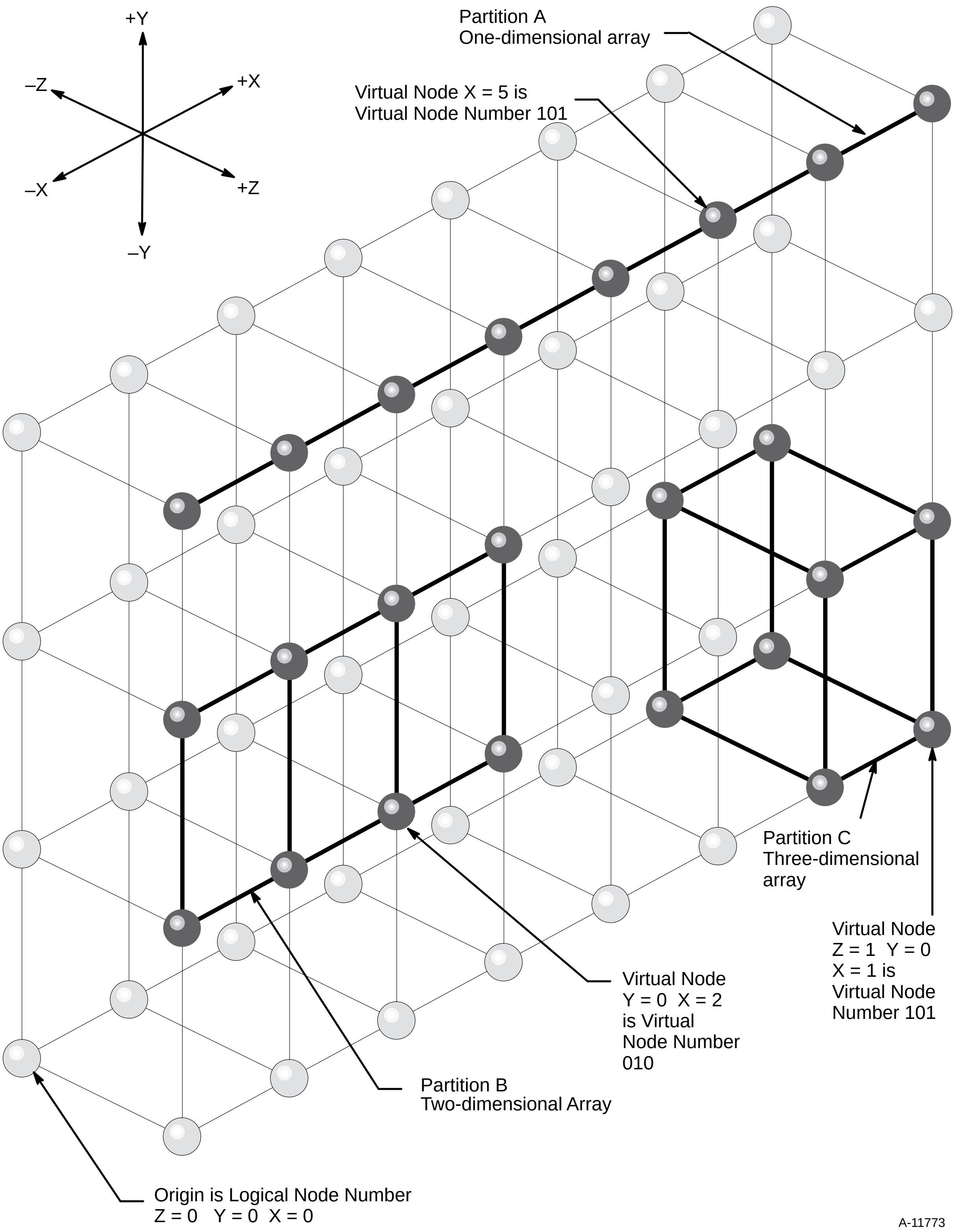


Figure 6-17. Three 8-node Partition Shapes and Virtual Node Numbers

The network interface compares the 3 least significant bits of the reformatted virtual node number to the X mask. The X mask is a contiguous, right-justified field of 1's that indicates how many bits of the virtual node number are assigned to the X dimension. When a bit in the X mask is set to 1, the corresponding bit of the virtual node number is valid. When a bit in the X mask is set to 0, the corresponding bit of the virtual node number is not used for the X dimension.

The network interface then shifts the reformatted virtual node number 0 to 3 bits to the right until the first bit that is assigned to the Y dimension is in the 0 bit location. The network interface compares the 4 least significant bits of the shifted virtual node number to the Y mask. When a bit in the Y mask is set to 1, the corresponding bit of the shifted virtual node number is valid. When a bit in the Y mask is set to 0, the corresponding bit of the shifted virtual node number is not used for the Y dimension.

The network interface then shifts the original reformatted virtual node number 0 to 7 bits to the right until the first bit that is assigned to the Z dimension is in the 0 bit location. The network interface compares the 3 least significant bits of the shifted virtual node number to the Z mask. When a bit in the Z mask is set to 1, the corresponding bit of the shifted virtual node number is valid. When a bit in the Z mask is set to 0, the corresponding bit of the shifted virtual node number is not used for the Z dimension.

After reformatting the virtual node number into a 10-bit virtual node number, the network interface adds a base node address to the 10-bit virtual node number (refer to Figure 6-18).

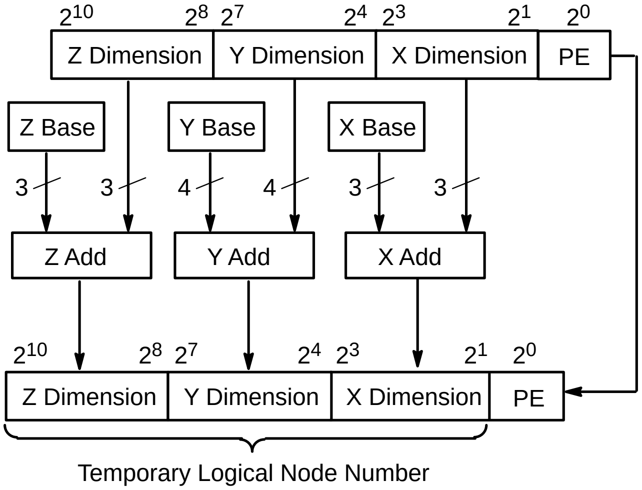


Figure 6-18. Adding the Base Node Address

The base node address is the logical node number of the smallest numbered virtual PE node in a partition. For example, Table 6-6 lists the base node addresses for the three partitions shown in Figure 6-17. Table 6-6 also lists the shift and mask values for each partition.

Table 6-6. Partitions A, B, and C Translation Parameters

Partition	Parameter	Z Dimension	Y Dimension	X Dimension
A	Base	1	3	0
	Shift	0	0	Not Applicable
	Mask	0000	0000	111
B	Base	1	1	0
	Shift	0	2	Not Applicable
	Mask	0000	0001	011
C	Base	0	0	6
	Shift	2	1	Not Applicable
	Mask	0001	0001	001

After adding the base node address to the 10-bit virtual node number, the network interface compares the two most significant bits of the X-, Y-, and Z-dimension numbers to the X-, Y-, and Z-dimension configuration masks (refer to Figure 6-19). When a bit of the X-, Y-, or Z-dimension configuration mask is set to 0, the corresponding bit in the X-, Y-, or Z-dimension number is valid. When a bit of the X-, Y-, or Z-dimension configuration mask is set to 1, the corresponding bit in the X-, Y-, or Z-dimension number is set to 0.

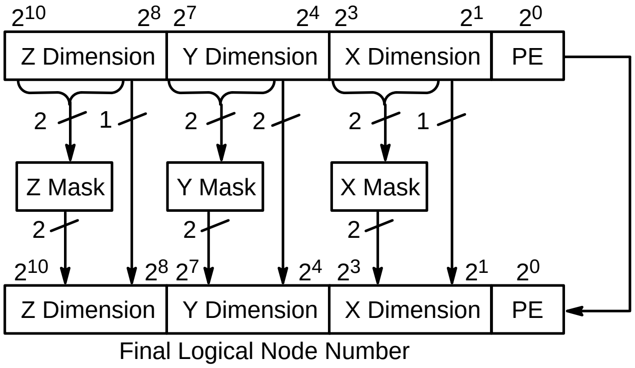


Figure 6-19. Configuration Masks

The configuration masks prevent the final logical node number from becoming larger than the number of logical nodes in a system. For example, Figure 6-21 shows a two-dimensional, four-node partition in a 128-PE CRAY T3D system. When the network interface converts the virtual node number 11 (Y=1 X=1) into a logical node number, the network interface first reformats the virtual node number into a 10-bit virtual node number.

After reformatting the virtual node number into a 10-bit virtual node number, the network interface adds the base node address to the virtual node number (refer to Figure 6-20). In this example, the Y-dimension portion of the temporary logical node number is four. This value is larger than the largest numbered node in the Y dimension (refer again to Figure 6-21).

The network interface then compares the temporary logical node number to the configuration masks to obtain the final logical node number of Z=1 Y=0 X=7 (refer again to Figure 6-20 and Figure 6-21). For more information on the configuration mask values for the different configurations of the CRAY T3D system, refer to “Register Mapping” in this section.

	Z	Y	X	
	000	0001	001	10-bit Virtual Node Number
+	001	0011	110	Base Node Address
	001	0100	111	Temporary Logical Node Number
	11	11	00	Configuration Masks
	001	0000	111	Final Logical Node Number

Figure 6-20. Configuration Mask Example

The base, shift, mask, and configuration mask values for all three dimensions are stored in a register called the network interface PE adjust (LPE_XLATE) register. This register is loaded by the operating system when the partitioning of the CRAY T3D system changes.

By changing the values of the base, shift, and mask numbers in the LPE_XLATE register, the operating system defines the partitions in the logical CRAY T3D system. More information on the LPE_XLATE register is provided in “Register Mapping” in this section.

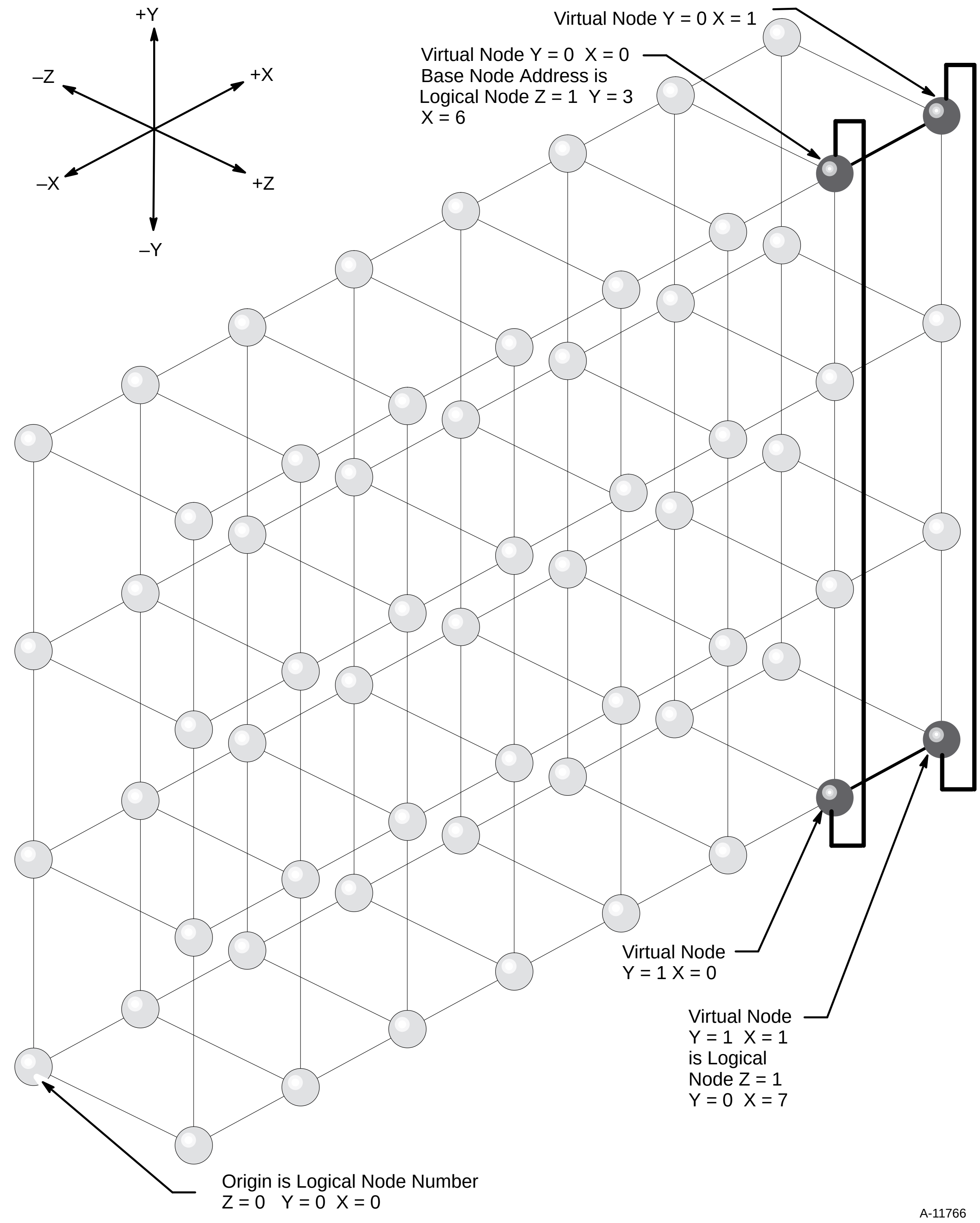


Figure 6-21. Two-dimensional 4-node Partition and Virtual Node Numbers

Converting the Logical PE Number Into a Routing Tag

The network interface receives a logical PE number in one of two ways: directly from a PE or the BLT in the processing element node, or by converting a virtual PE number into a logical PE number. As was previously described, the logical PE number may not always correspond to the same physical PE in the CRAY T3D system.

The logical PE number contains two parts: the logical node number and the PE bit. Because the logical node number does not always correspond to the same physical node, the network interface cannot use the logical node number to calculate how to route information to the correct physical node.

The network interface in each node uses a look-up table to obtain the routing tag. Circuitry in the network interface enters the logical node number into the routing tag look-up table. The routing tag look-up table then provides the routing tag that corresponds to that logical node (refer to Figure 6-22).

The routing tag contains information that indicates the path a packet must follow through the interconnect network to get from the physical source node to the physical destination node. This information includes how many hops to complete in each dimension, what direction to travel in each dimension, and which virtual channel (VC) to use in each dimension. For more information on virtual channels, refer again to “Virtual Channels” in Section 2, “Interconnect Network.”

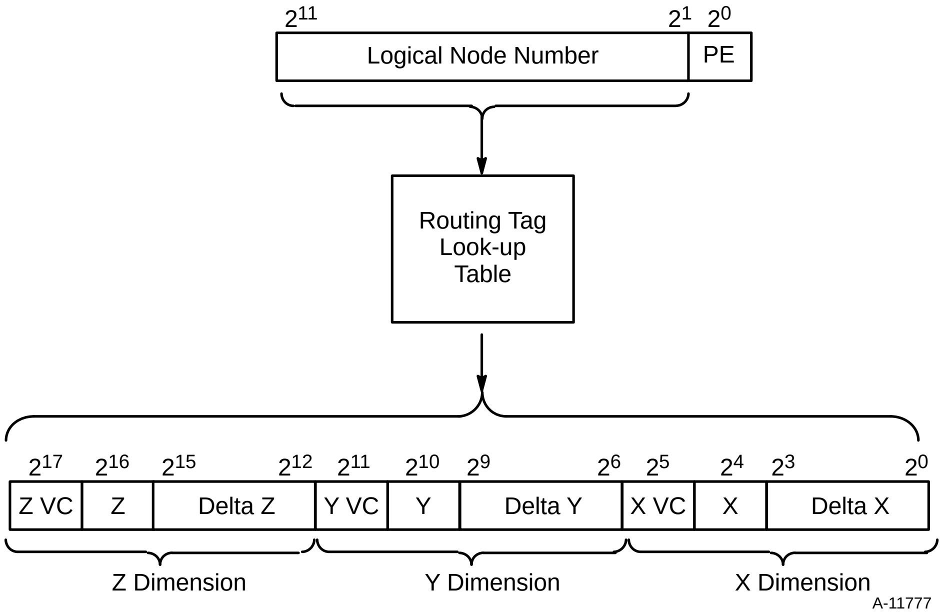


Figure 6-22. Logical Node Number to Routing Tag Conversion

The routing tags for each node in the CRAY T3D system are generated by a program that runs in the host system. The routing tag calculation program uses a configuration file that contains information on each node and communication link in the CRAY T3D system.

When a communication link or node in the CRAY T3D system is not fully functional, the system administrator modifies the configuration file in the host system. The administrator then runs the routing tag calculation program. The program generates new routing tags that steer information around a bad node or communication link.

When the CRAY T3D system is booted, the routing tags are written into the routing tag look-up table of each node in the system. More information on the addresses and format of the routing tag look-up table is provided in “Register Mapping” in this section.

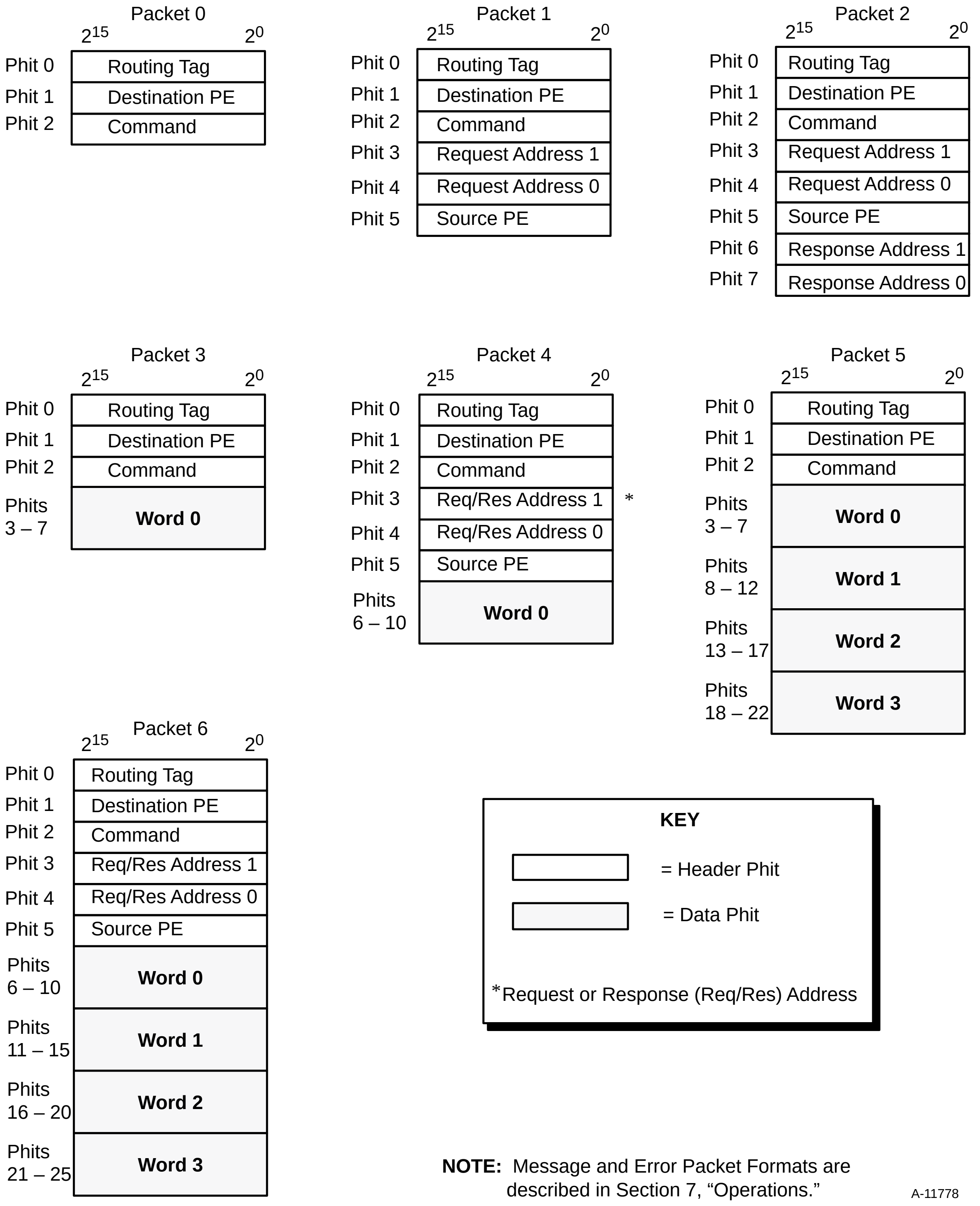
Creating a Packet

The network interface generates information and uses the information it receives from a PE or the BLT to create packets. Table 6-7 lists the types of packets the network interface creates during various functions.

Table 6-7. Functions and Packet Types

Function	Request Packet Type	Response Packet Type
PE generated noncacheable read	1	3
PE generated write (single-word body)	4	0
PE generated cached read	1	5
PE generated write (4-word body)	6	0
BLT generated single-word read	2	4
BLT generated single-word write	4	0
BLT generated block read	2	6
BLT generated block write	6	0
Prefetch read	1	3
Fetch-and-increment read	1	3
Fetch-and-increment register write	4	0
Noncacheable atomic swap	4	3

There are a total of 8 packet types (0 through 7). Figure 6-23 shows the format of packet types 0 through 6. The format of packet type 7, used for message packets, is shown in Section 7, “Operations.” Each packet contains a header and may contain data. The following subsections describe each phit of the header and data portions of a packet. (Message and error packets are described in Section 7, “Operations.”)



NOTE: Message and Error Packet Formats are described in Section 7, “Operations.”

Figure 6-23. Packet Types

Header

Every packet contains a header. The header may consist of the following types of phits: a routing tag phit, a destination phit, a command phit, request address phits, a source phit, or response address phits. The following paragraphs describe each type of phit.

Every packet contains a routing tag phit. After receiving the routing tag from the network interface, the X-dimension switch in the network router examines bits 0 through 3 of the routing tag. When bits 0 through 3 are equal to zero, the X-dimension switch sends the routing tag information to the Y-dimension switch in the network interface. As the X-dimension switch sends information to the Y-dimension switch, the bits of the routing tag are rearranged on the circuit board so that bits 0 through 3 contain the ∆Y bits.

The bit format of the routing tag phit is rearranged on the circuit board each time information transfers between the X- and Y-dimension switches or between the Y- and Z-dimension switches in the network interface. The bits of the routing tag phit are rearranged so the same 10K options can be used for the X-, Y-, and Z-dimension switches.

When bits 0 through 3 of the original routing tag are not equal to zero, the X-dimension switch checks the value of the ±X bit. This bit indicates when the packet should be sent on the +X or –X communication link. The X-dimension switch then checks the value of the X VC bit to determine which virtual channel to use on the communication link. (For more information about virtual channels, refer again to Section 2, “Interconnect Network” and the subsection, “Routing Tag Look-up Table Low-order Bits Registers” later in this section.) After checking the value of these 2 bits, the X-dimension switch increments the value of the ∆X portion of the routing tag phit by one and sends the phit to the network interface of the next node in the +X or –X direction. (The ∆X portion is incremented because it contains the two’s compliment of the number of hops to complete in the X dimension.)

Every packet contains a destination phit. The destination phit contains the destination PE number (refer to Figure 6-24). The destination PE number is the logical PE number of the PE that will receive the packet. The network interface obtains the destination PE number from bits 0 through 11 of the destination logical PE number that was read from the DTB annex or that was converted from the virtual PE number read from the DTB annex.

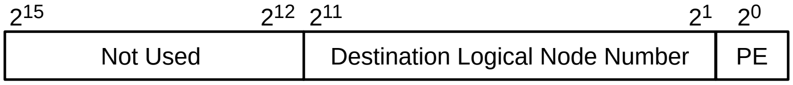


Figure 6-24. Destination Phit

Every packet also contains a command phit with information that indicates to the destination PE what type of function to perform. The network interface formats the command information after it receives the information from the support circuitry in a PE or the BLT.

Bits 0 through 11 of the command phit contain a command field (refer to Figure 6-25). Bit 12 indicates when the packet will go to PE 0 or PE 1 in the destination PE node. Bits 13 and 14 of the command phit contain an even parity bit for bits 8 through 12 and 0 through 7, respectively. When set to 1, bit 15 indicates the packet is an error message created by the network interface from an incorrectly routed packet.

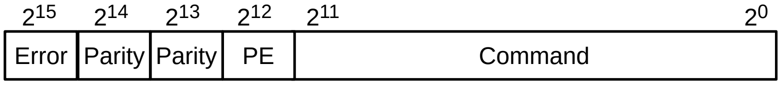


Figure 6-25. Command Phit

The command field contains three parts: a packet type field, a request or response bit, and a command (refer to Figure 6-26). The packet type field indicates what type of packet the command phit is in (refer again to Figure 6-23). When set to 1, the request or response bit indicates the packet is a request packet. When set to 0, the request or response bit indicates the packet is a response packet. The command indicates what function to perform. More information on the command fields for each packet is provided in Section 7, “Operations.”

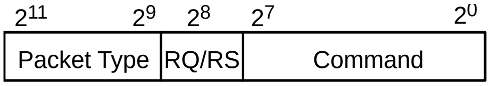


Figure 6-26. Command Field

All the phits in a packet, except the routing tag, destination, and data phits, are protected by parity bits. As the network interface receives each phit of a packet (except the routing tag, destination, and data phits), the network interface generates 2 new parity bits.

When the new parity bits are not the same as the parity bits in the phit, an error occurred during the transfer. The network interface then sets bit 1 of the system status register (SSR) to 1, converts the packet into an error message, and sends the error message to PE 0 or PE 1 (identified in the command phit). More information on the system status register is provided in Section 10, “Control and Status.” More information on messages is provided in Section 7, “Operations.”

Only request packets contain request address phits. The request address points to one 64-bit word or the first word in a 4-word block of data in the memory of a PE. The network interface obtains the request address from the address offset generated by a PE or the BLT. Figure 6-27 shows the format of the request address phits.

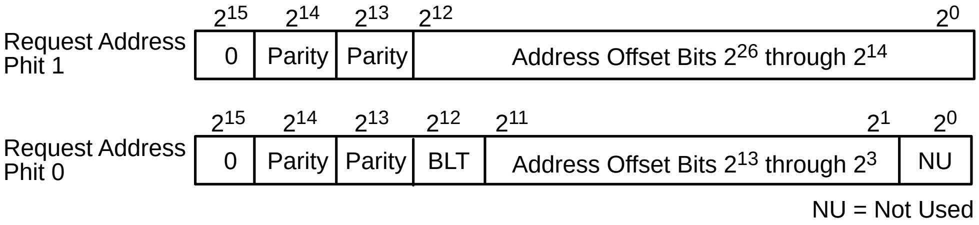


Figure 6-27. Request Address Phits

The BLT error bit (bit 12) is used to indicate one of three BLT errors in request packet types 4 or 6. When the BLT issues a write request (packet type 4 or 6) and the BLT error bit is set to 1, the BLT aborts the transfer.

Every request packet also contains a source phit. The source phit contains the source node number, which is the logical node number of the node creating the request packet. The source phit also contains a PE bit that indicates which PE in the source node created the packet. Figure 6-28 shows the format of the source phit.

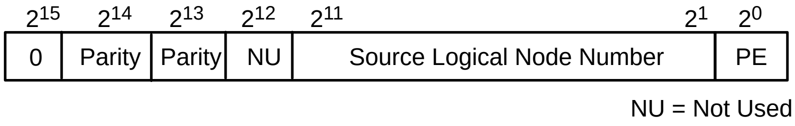


Figure 6-28. Source Phit

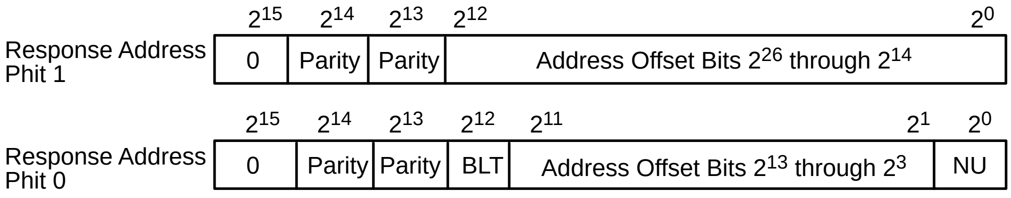
The node that receives the request packet uses the source node number as the destination node number in a response packet. The network interface generates the source node number by reading the node number stored in a register called the network interface source (X_WHOAMI) register.

The network interface also uses the X_WHOAMI register to check if a packet was routed to the correct node. After receiving a packet, the network interface compares the node number stored in the X_WHOAMI register with the destination node number in the packet. When the numbers are equal, the packet arrived at the correct destination node. When the numbers are not equal, an error occurred and the packet arrived at the wrong node.

When the network interface receives a packet that was supposed to transfer to another node, hardware in the network interface sets bit 1 (network packet error bit) of the system status register to indicate that a packet misrouted. More information on the system status register is provided in Section 10, “Status and Control.”

The network interface then converts the packet into an error message and sends the message to PE 0 or PE 1 in the node that received the packet (according to the PE bit that is stored in the command field of the packet). The microprocessor can then examine the message to try and determine the cause of the error.

A request or response packet may also contain a response address. The response address points to a location in local memory of the source PE where 1 word or 4 words of data from a read response packet will be stored. Figure 6-29 shows the format of the response address phits.



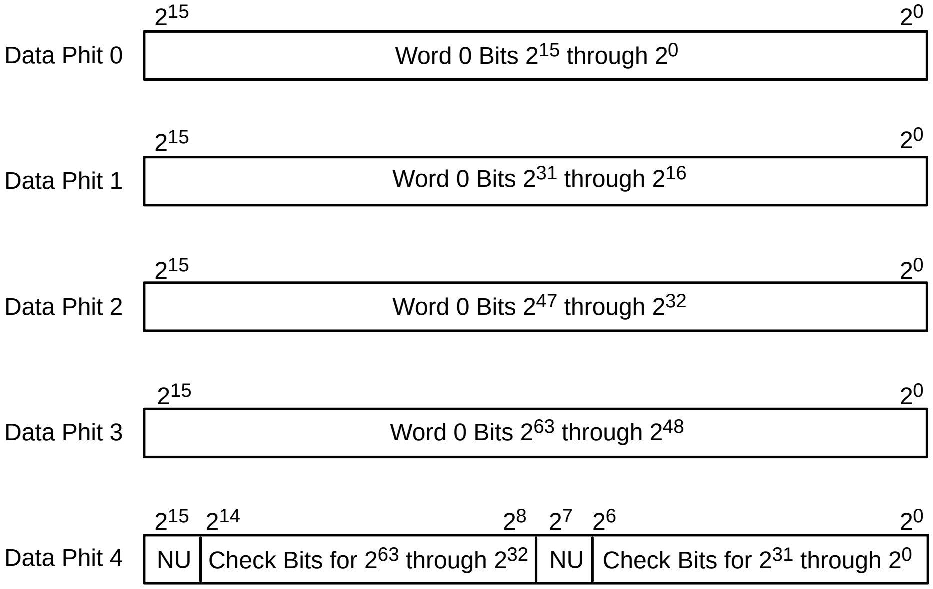
NOTE: NU = Not Used

Figure 6-29. Response Address Phits

The BLT error bit (bit 12) is used to indicate a BLT range error in request packet type 2 or response packet type 4 or 6. When the BLT issues a read request (packet type 2) and the BLT error bit is set to 1 in response address phit 0, the PE that receives the packet performs the memory read and creates a response packet type 4 or 6. The PE also sets the BLT error bit to 1 in response address phit 0 of the response packet. The PE that receives the response packet then does not perform the memory write.

Data

Both request and response packets may contain data. Data is one 64-bit word or four 64-bit words that will be used by the PE receiving the packet. The network interface receives data for a packet from a PE in the processing element node. Seven check bits (used by the microprocessor to check the data for errors) are sent along with each 32-bit halfword. More information on how the microprocessor checks and corrects data errors is provided in Section 3, “Processing Element Node.” Figure 6-30 shows the format of the data phits for word 0 in a packet.


2^{15}
2^{0}
Data Phit 0
Word 0 Bits 2^{15} through 2^{0}
2^{15}
2^{0}
Data Phit 1
Word 0 Bits 2^{31} through 2^{16}
2^{15}
2^{0}
Data Phit 2
Word 0 Bits 2^{47} through 2^{32}
2^{15}
2^{0}
Data Phit 3
Word 0 Bits 2^{63} through 2^{48}
2^{15} 2^{14} 2^{8} 2^{7} 2^{6} 2^{0}
Data Phit 4
NU
Check Bits for 2^{63} through 2^{32}
NU
Check Bits for 2^{31} through 2^{0}

NOTE: NU = Not Used

Figure 6-30. Data Phits

Register Mapping

The CRAY T3D system uses several memory-mapped registers when interpreting addressing information. These registers are used to identify where a PE is located, check the value of virtual and logical PE numbers, and create the routing tag.

The following subsections describe the addressing and bit assignments for the memory-mapped registers (hereafter referred to as registers) used for addressing in the CRAY T3D system. Each subsection also provides a brief summary of the function of the register.

Table 6-8 is a summary of the registers and their names. Table 6-8 also lists the partial physical address of each register as it appears on the address pins of the microprocessor.

Table 6-8. Address Register

Address	Register Name	Direction	Description
10400000_{16}	P_WHOAMI	Read	Physical PE number
10450000_{16}	L_WHOAMI	Write	Logical PE number
10454000_{16}	V_WHOAMI	Write	Virtual PE number
10458000_{16}	VRT_RG	Write	Virtual PE range
$106A0000_{16}$	X_WHOAMI	Write	Network interface source
$106B0000_{16}$	LPE_XLATE	Write	Network interface PE adjust
$106F0000_{16}$	ROUTE_LO	Write	Routing tag look-up table low order bits
$107F0000_{16}$	ROUTE_HI	Write	Routing tag look-up table high order bits

NOTE: Because of multiplexed data paths, when one PE in a processing element node is modifying the contents of the X_WHOAMI, LPE_XLATE, ROUTE_LO, or ROUTE_HI register, the other PE in the node must not attempt to modify any of the shared registers at the same time. The shared registers include the X_WHOAMI, LPE_XLATE, ROUTE_LO, ROUTE_HI, NET_ENA, NET_PFM, NODE_CSR, and BLT registers.

Because the virtual address is defined by software, the addresses for each of the registers are given according to the partial physical address as they appear on the pins of the microprocessor.

Physical PE Number Register
Address 10400000_{16}

The physical PE number register (P_WHOAMI) is a 12-bit, read-only, system privileged register that contains the physical PE number. The physical PE number is a unique number assigned to each PE in the system.

Figure 6-31 shows the bit assignments for the P_WHOAMI register address as they appear on the address pins of the microprocessor.

HEX	1	0				4				0				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x = Don’t Care

Figure 6-31. P_WHOAMI Register Address Bit Assignments

The operating system uses the values stored in the P_WHOAMI registers to load the logical PE number registers of the CRAY T3D system when the system is deadstarted. This register provides a unique temporary logical PE number for each physical PE in the system and allows the operating system to transfer deadstart information to all the PEs in the system. After the system is deadstarted, the logical PE number (L_WHOAMI) registers are loaded with the appropriate logical PE numbers.

When a processing element module (PEM) or an I/O module (IOM) is inserted into the CRAY T3D cabinet, hardware on the module and signals from the cabinet wire mat set the value of the bits in the P_WHOAMI register. These values provide a unique physical PE number for each PE in the system.

Table 6-9 shows the bit format of the P_WHOAMI register, and the following subsections describe each bit of the register.

Table 6-9. P_WHOAMI Register Format

Bits	Name
0	PE number
11 – 1	Physical node number
63 – 12	These bits are not used

PE Number
Bit 0

The PE number bit indicates whether the PE is PE 0 or PE 1 in a processing element node. When set to 1, the PE is PE 1; and when set to 0, the PE is PE 0. When the PE is in an input node or an output node, this bit is set to 0.

NOTE: Throughout this manual, whenever a physical PE number is referenced, it is represented as a three-digit hexadecimal number. For example, a PE may have a physical PE number of $10B_{16}$.

Physical Node Number
Bits 1 through 11

These bits indicate which physical node the PE is located in.

NOTE: Throughout this manual, whenever a physical node number is referenced, it is also represented as a three-digit hexadecimal number. This number is equivalent to the number read from a P_WHOAMI register with bit 0 set to 0. For example, physical node $10A_{16}$ contains the physical PEs $10A_{16}$ and $10B_{16}$.

Bits 4 through 7 of the physical node number indicate the position of the physical node on a printed circuit board (PCB). The following text describes each of these bits.

The PCB node number bit (bit 4) indicates which node on a printed circuit board the PE resides in. On a PEM printed circuit board, this bit indicates whether the PE resides in PCB node 0 or PCB node 1. On an IOM printed circuit board, this bit indicates whether the PE resides in the input node (node 0) or output node (node 1).

The printed circuit board number bit (bit 5) indicates which printed circuit board on a PEM or an IOM the PE resides on. Figure 6-32 shows the location of printed circuit boards 0 and 1. The printed circuit boards are also referred to as board A (0) and board B (1).

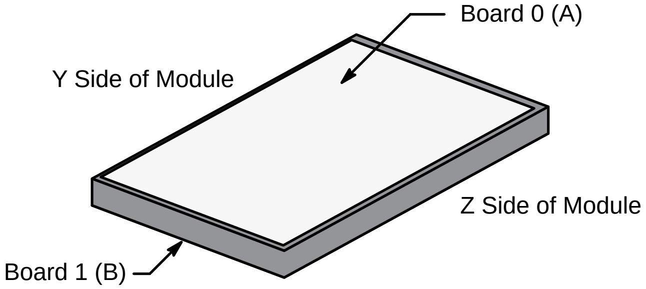


Figure 6-32. Printed Circuit Board Numbers

The chassis ID bits (bits 6 and 7) indicate which cabinet in a multiple cabinet configuration the PEM or IOM resides in. Figure 6-33 shows the cabinet numbering for multiple cabinet systems.

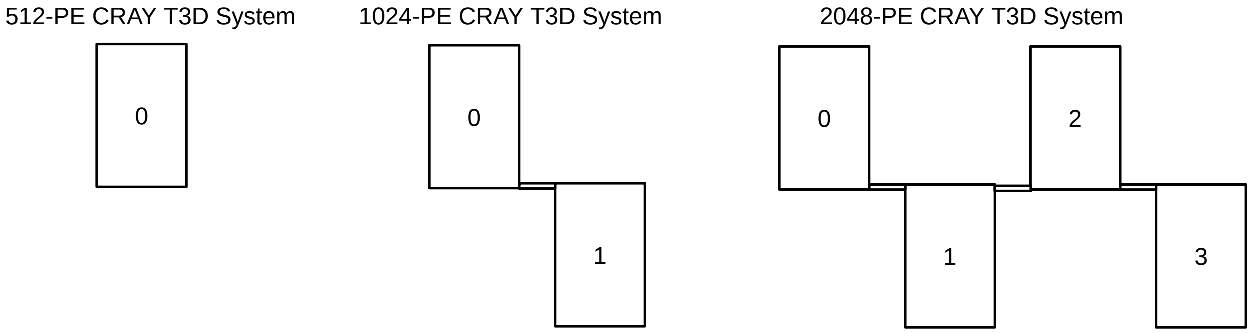


Figure 6-33. Multiple Cabinet Configuration Cabinet Numbers

Logical PE Number Register
Address 10450000_{16}

The logical PE number register (L_WHOAMI) is a 12-bit, write-only, system privileged register that contains the logical PE number. The logical PE number is a unique number that is assigned by the operating system to each physical PE used in the logical CRAY T3D system. The support circuitry uses the L_WHOAMI register to check whether a memory function is to be performed in the local PE or a remote PE.

Figure 6-34 shows the bit assignments for the L_WHOAMI register address as they appear on the address pins of the microprocessor.

HEX	1	0				4				5				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	0	0	0	1	0	1	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x = Don’t Care

Figure 6-34. L_WHOAMI Register Address Bit Format

Table 6-10 shows the bit format of the L_WHOAMI register and describes each bit in the register.

Table 6-10. L_WHOAMI Register Format

Bits	Name	Description
0	PE number	This bit indicates whether the PE is PE 0 or PE 1 in the logical node. This bit is set to 0 when the PE is in an input or output node.
3 – 1	X coordinate	These bits contain the X-dimension portion of the logical node number.
7 – 4	Y coordinate	These bits contain the Y-dimension portion of the logical node number.
10 – 8	Z coordinate	These bits contain the Z-dimension portion of the logical node number.
11	I/O node	When set to 1, this bit indicates that the PE is in an input node or an output node.
63 – 12	Not used	These bits are not used.

NOTE: When writing a PE number into the L_WHOAMI register, bit 0 of the system control register should be set to 0.

Virtual PE Number Register
Address 10454000_{16}

The virtual PE number register (V_WHOAMI), is an 11-bit, write-only, system privileged register that contains the virtual PE number. The virtual PE number is a number assigned by the operating system to a logical PE in a partition. The support circuitry uses the V_WHOAMI register to check whether a memory function is to be performed in the local PE or a remote PE.

Figure 6-35 shows the bit assignments for the V_WHOAMI register address as they appear on the address pins of the microprocessor.

HEX	1	0				4				5				4				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	0	0	0	1	0	1	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 6-35. V_WHOAMI Register Address Bit Format

Table 6-11 shows the bit format of the V_WHOAMI register and describes each bit in the register.

Table 6-11. V_WHOAMI Register Format

Bits	Name	Description
0	PE number	This bit indicates whether the PE is PE 0 or PE 1 in the virtual node.
10 – 1	Virtual node number	These bits contain the virtual node number. The virtual node number may range from 1 to 10 bits.
63 – 11	Not used	These bits are not used.

Virtual PE Range Mask Register
Address 10458000_{16}

The virtual PE range mask register (VRT_RG) is a 10-bit, write-only, system privileged register that contains the virtual PE range mask. The virtual PE range mask contains a contiguous field of 0's in the legal bit positions for the size of the user partition. The remaining bits are set to 1's.

Figure 6-36 shows the bit assignments for the VRT_RG register address as they appear on the address pins of the microprocessor.

HEX	1	0				4				5				8				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	0	0	0	1	0	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x = Don't Care

Figure 6-36. VRT_RG Register Address Bit Assignments

Table 6-12 shows the bit format of the VRT_RG register and describes each bit. For examples of virtual range masks, refer again to Table 6-5.

Table 6-12. VRT_RG Register Format

Bits	Name	Description
0	Not used	This bit is not used.
10 – 1	Virtual PE range mask	These bits contain the virtual PE range mask.
63 – 11	Not used	These bits are not used.

Network Interface Source Register Address $106A0000_{16}$

The network interface source register (X_WHOAMI) is an 11-bit, write-only, system privileged register that contains the source logical node number. The network interface uses the source logical node number to indicate which logical node created a packet (source phit).

Figure 6-37 shows the bit assignments for the X_WHOAMI register address as they appear on the address pins of the microprocessor.

HEX	1	0				6				A				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	1	0	1	0	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 6-37. X_WHOAMI Register Address Bit Assignments

Table 6-13 shows the bit format of the X_WHOAMI register and describes each bit in the register.

Table 6-13. X_WHOAMI Register Format

Bits	Name	Description
0	Not used	This bit is not used.
3 – 1	X coordinate	These bits contain the X-dimension portion of the source logical node number.
7 – 4	Y coordinate	These bits contain the Y-dimension portion of the source logical node number.
10 – 8	Z coordinate	These bits contain the Z-dimension portion of the source logical node number.
11	I/O	When set to 1, this bit indicates that the source logical node is an input node or an output node.
63 – 12	Not used	These bits are not used.

The network interface also uses the X_WHOAMI register to check the value of the destination logical node number in a packet header. After receiving a packet, the network interface compares the node number stored in the X_WHOAMI register with the destination node number in the packet. When the numbers are equal, the packet arrived at the correct destination node. When the numbers are not equal, a network packet error occurred and the packet arrived at the wrong node.

Network Interface PE Adjust Register
Address $106B0000_{16}$

The network interface PE adjust register (LPE_XLATE) is a 31-bit, write-only, system privileged register. The network interface PE adjust register contains the shift, base, mask, and configuration mask values used to translate a virtual node number into a logical node number (refer again to Figure 6-16 through Figure 6-19).

Figure 6-38 shows the bit assignments for the LPE_XLATE register address as they appear on the address pins of the microprocessor.

HEX	1	0				6				B				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	1	0	1	0	1	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Figure 6-38. LPE_XLATE Register Address Bit Assignments

Table 6-14 shows the bit format of the LPE_XLATE register and describes each bit of the register. Table 6-15 lists the values of the configuration masks for each configuration of the CRAY T3D system.

Table 6-14. LPE_XLATE Register Format

Bits	Name	Description
1 – 0	Y shift	These bits contain the number of bits the virtual address must be shifted right to place the first bit assigned to the Y dimension in the bit 0 location.
4 – 2	Z shift	These bits contain the number of bits the virtual address must be shifted right to place the first bit assigned to the Z dimension in the bit 0 location.
7 – 5	X mask	These bits contain a right-justified field of 1’s that indicate how many bits of the virtual address are assigned to the X dimension. The other bits are set to 0’s.
11 – 8	Y mask	These bits contain a right-justified field of 1’s that indicate how many bits of the virtual address are assigned to the Y dimension. The other bits are set to 0’s.
14 – 12	Z mask	These bits contain a right-justified field of 1’s that indicate how many bits of the virtual address are assigned to the Z dimension. The other bits are set to 0’s.

Table 6-14. LPE_XLATE Register Format (continued)

Bits	Name	Description
17 – 15	X base	These bits contain the X-dimension portion of the logical node number corresponding to the smallest numbered virtual node in a partition.
21 – 18	Y base	These bits contain theY-dimension portion of the logical node number corresponding to the smallest numbered virtual node in a partition.
24 – 22	Z base	These bits contain the Z-dimension portion of the logical node number corresponding to the smallest numbered virtual node in a partition.
26 – 25	X-configuration mask	These bits contain the X-dimension configuration mask. For more information on this mask, refer to Table 6-15.
28 – 27	Y-configuration mask	These bits contain the Y-dimension configuration mask. For more information on this mask, refer to Table 6-15.
30 – 29	Z-configuration mask	These bits contain the Z-dimension configuration mask. For more information on this mask, refer to Table 6-15.
63 – 31	Not used	These bits are not used.

Table 6-15. Configuration Mask Values

System	Z-Configuration Mask Bits 30 and 29	Y-Configuration Mask Bits 28 and 27	X-Configuration Mask Bits 26 and 25
128-PE CRAY T3D multiple-cabinet system	11	11	00
256-PE CRAY T3D multiple-cabinet system	10	11	00
512-PE CRAY T3D multiple-cabinet system	00	11	00
1024-PE CRAY T3D multiple-cabinet system	00	10	00
2048-PE CRAY T3D multiple-cabinet system	00	00	00
128-PE CRAY T3D single-cabinet system	10	11	10
256-PE CRAY T3D single-cabinet system	00	11	10

Routing Tag Look-up Table Low-order Bits Registers Address $106F0000_{16}$

The routing tag look-up table low-order bits registers (ROUTE_LO) are 12-bit, write-only, system privileged registers that each contain the 12 low-order bits of a routing tag. Figure 6-39 shows the bit assignments for a ROUTE_LO register address as they appear on the address pins of the microprocessor.

HEX	1	0				6				F				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	1	0	1	1	1	1	a	a	a	a	a	a	a	a	a	a	a	x	x	x	x	x

x = Don’t Care a = Look-up Table Address

Figure 6-39. ROUTE_LO Register Address Bit Assignments

Bits 5 through 15 of the address indicate one of the 2,048 ROUTE_LO registers. Each ROUTE_LO register corresponds to one destination logical node number. For example, the address for the ROUTE_LO register that corresponds to the destination logical node number Z=1, Y=5, X=2 has bits 5 through 15 set to 15 = 0 (I/O bit), 14 through 12 = 001 (Z dimension), 11 through 8 = 0101 (Y dimension), and bits 7 through 5 set to 010 (X dimension).

Table 6-16 shows the bit format of each ROUTE_LO register and describes each bit in the register. Figure 6-40 shows eight nodes in the X dimension of a CRAY T3D system. For clarity, the communication links that connect to the other nodes in the Z and Y dimensions are not shown. This figure is used as a reference for the following examples.

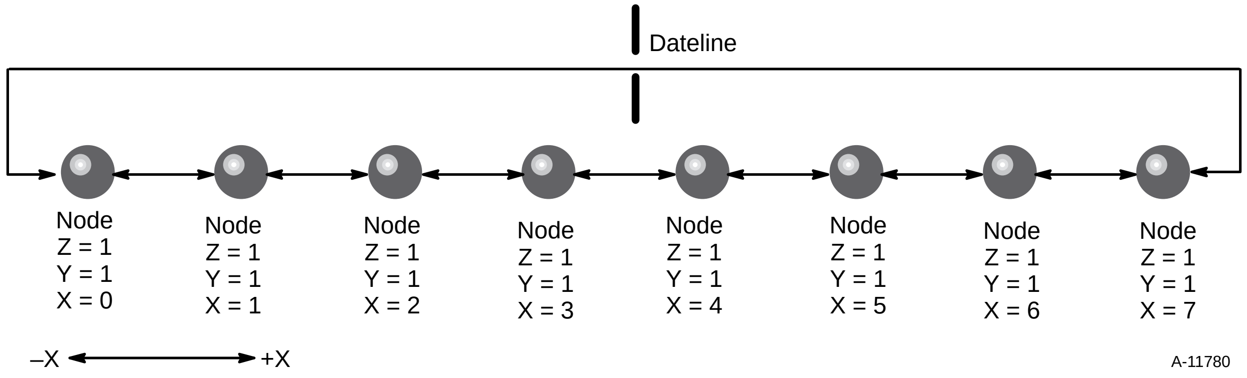


Figure 6-40. Dateline Crossing

Table 6-16. ROUTE_LO Register Format

Bits	Name	Description
3 – 0	ΔX	These bits contain the two’s complement of the number of hops the packet will make in the X dimension.
4	±X	When set to 0, this bit indicates the packet will travel in the +X direction. When set to 1, this bit indicates the packet will travel in the –X direction.
5	X VC select bit 0	When set to 0, this bit indicates a request packet will use VC 0 and a response packet will use VC 2 in the X dimension. When set to 1, this bit indicates a request packet will use VC 1 and a response packet will use VC 3 in the X dimension.
9 – 6	ΔY	These bits contain the two’s compliment of the number of hops the packet will make in the Y dimension.
10	±Y	When set to 0, this bit indicates the packet will travel in the +Y direction. When set to 1, this bit indicates the packet will travel in the –Y direction.
11	Y VC select bit 0	When set to 0, this bit indicates a request packet will use VC 0 and a response packet will use VC 2 in the Y dimension. When set to 1, this bit indicates a request packet will use VC 1 and a response packet will use VC 3 in the Y dimension.
15 – 12	Zero	These bits are not used and should be set to 0’s.
31 – 16	Repeated data	Bits 31 through 16 must be set to the same value as bits 15 through 0.
47 – 32	Repeated data	Bits 47 through 32 must be set to the same value as bits 15 through 0.
63 – 48	Repeated data	Bits 63 through 48 must be set to the same value as bits 15 through 0.

The following paragraphs provide examples of ROUTE_LO register values for the routing tag look-up table in logical node Z=1, Y=1, X=2 in Figure 6-40. These examples assume the logical PE numbers are set to match how the physical nodes are connected. Also, these examples assume that the dateline for the X dimension is the communication link that connects the smallest numbered node to the largest numbered node. For more information on the dateline, refer again to “Virtual Channels” in Section 2, “Interconnect Network.”

The value of the ROUTE_LO register for the destination node number Z=1, Y=1, X=4 is ∆X=E, ±X=0, X VC=0, ∆Y=0, ±Y=0, Y VC=0. The ∆X value is set to E_{16} because E is the two’s compliment of 2. The packet will complete two hops in the X dimension to travel from node Z=1, Y=1, X=2 to node Z=1, Y=1, X=4. The ±X value is set to 0 because the packet will travel in the positive X direction. The X VC select bit 0 is set to 0 because the packet does not cross the dateline in the X dimension when it travels through the network.

The value of the ROUTE_LO register for the destination node number Z=1, Y=1, X=7 is ∆X=D, ±X=1, X VC=1, ∆Y=0, ±Y=0, Y VC=0. The ∆X value is set to D_{16} because D is the two’s compliment of 3. The packet will complete three hops in the X dimension to travel from node Z=1, Y=1, X=2 to node Z=1, Y=1, X=7. The ±X value is set to 1 because the packet will travel in the negative X direction. The X VC select bit 0 is set to 1 because the packet does cross the dateline in the X dimension when it travels through the network.

The eight nodes shown in Figure 6-40 do not connect to an I/O gateway or a spare processing element node in the X dimension. When a dimension contains an I/O gateway, a spare processing element node, or both an I/O gateway and a spare processing element node, the routing tags must contain values that reflect the number of hops a packet makes when traveling through that dimension.

Figure 6-41 shows eight processing element nodes, an I/O gateway, and a spare processing element node in the X dimension of a CRAY T3D system. For clarity, the communication links that connect to the other nodes in the Z and Y dimensions are not shown.

The following paragraphs provide examples of ROUTE_LO register values for the routing tag look-up table in logical node Z=0, Y=1, X=3 in Figure 6-41. These examples assume the logical PE numbers are set to match how the physical nodes are connected. Also, these examples assume that the dateline for the X dimension is the communication link that connects the smallest numbered node to the largest numbered node.

The value of the ROUTE_LO register for the destination node number Z=0, Y=1, X=0 is ∆X=B, ±X=1, X VC=0, ∆Y=0, ±Y=0, Y VC=0. The ∆X value is set to B because B is the two’s compliment of 5. Because the I/O gateway contains two nodes (input node and output node) the packet will complete five hops in the X dimension to travel from node Z=0, Y=1, X=3 to node Z=0, Y=1, X=0. The ±X value is set to 1 because the packet will travel in the negative X direction. The X VC select bit 0 is set to 0 because the packet does not cross the dateline in the X dimension when it travels through the network.

The value of the ROUTE_LO register for the destination node number Z=0, Y=1, X=5 is ∆X=D, ±X=0, X VC=0, ∆Y=0, ±Y=0, Y VC=0. The ∆X value is set to D because D is the two's compliment of 3. Because of the spare processing element node, the packet completes three hops in the X dimension when traveling from node Z=0, Y=1, X=3 to node Z=0, Y=1, X=5. The ±X value is set to 0 because the packet will travel in the positive X direction. The X VC select bit 0 is set to 0 because the packet does not cross the dateline in the X dimension when it travels through the network.

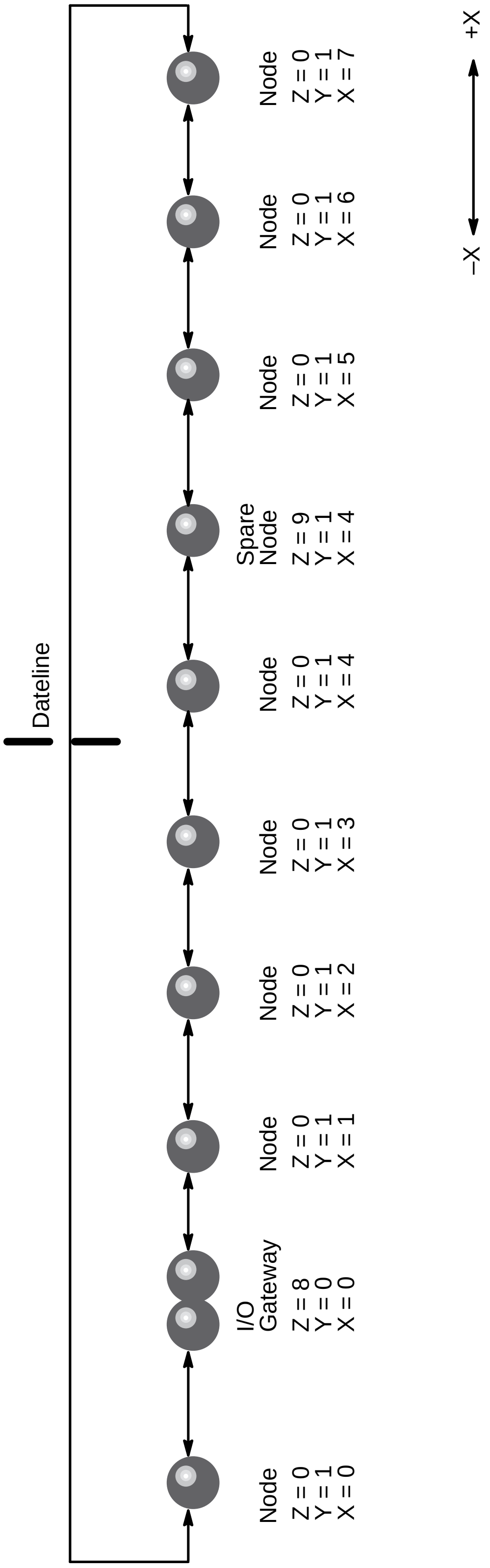


Figure 6-41. I/O Gateway and Spare PE Node Locations

Routing Tag Look-up Table High-order Bits Registers Address $107F0000_{16}$

The routing tag look-up table high-order bits registers (ROUTE_HI) are 6-bit, write-only, system privileged registers that each contain the 6 high-order bits of a routing tag. Figure 6-42 shows the bit assignments for a ROUTE_HI register address as they appear on the address pins of the microprocessor.

Bits 5 through 15 of the address indicate one of the 2,048 ROUTE_HI registers. Each ROUTE_HI register corresponds to one destination logical node number. For example, when bits 5 through 15 of the address are set to 15 = 0 (I/O bit), 14 through 12 = 001 (Z dimension), 11 through 8 = 0101 (Y dimension), and bits 7 through 5 set to 010 (X dimension), the ROUTE_HI register corresponds to a destination logical node number of Z=1, Y=5, X=2.

HEX	1	0				7				F				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	1	1	1	1	1	1	a	a	a	a	a	a	a	a	a	a	a	x	x	x	x	x

NOTE: x = Don't Care a = Look-up Table Address

Figure 6-42. ROUTE _HI Registers Address Bit Assignments

Table 6-17 shows the bit format of each ROUTE_HI register and describes each bit of the register.

Table 6-17. ROUTE_HI Register Format

Bits	Name	Description
3 – 0	ΔZ	These bits contain the two's complement of the number of hops the packet will make in the Z dimension.
4	±Z	When set to 0, this bit indicates the packet will travel in the +Z direction. When set to 1, this bit indicates the packet will travel in the –Z direction.
5	Z VC select bit 0	When set to 0, this bit indicates a request packet will use VC0 and a response packet will use VC1 in the Z dimension. When set to 1, this bit indicates a request packet will use VC 1 and a response packet will use VC 3 in the Z-dimension.
15 – 6	0	These bits are not used and should be set to 0's.

Table 6-17. ROUTE_HI Register Format (continued)

Bits	Name	Description
31 – 16	Repeated data	Bits 31 through 16 must be set to the same value as bits 15 through 0.
47 – 32	Repeated data	Bits 47 through 32 must be set to the same value as bits 15 through 0.
63 – 48	Repeated data	Bits 63 through 48 must be set to the same value as bits 15 through 0.

Hardware Description

The following subsections describe how the hardware interprets the partial physical address to address local memory, to reference memory-mapped registers, and to route packets through the interconnect network.

Partial Physical Address

The partial physical address consists of the DTB annex index, the address space partition, and the address offset. The microprocessor sends the DTB annex index to the AR option. The address space partition and the address offset are sent to the AM option.

DTB Annex Index

The AR option uses the DTB annex index to address the DTB annex. This index consists of 5 bits that specify which 1 of the 32 locations within the DTB annex will be written to or read from. For more information on the DTB annex, refer again to “AR Option” in Section 3, “Processing Element Node.”

Address Space Partition

The AM option uses the address space partition bit 28 to determine whether local memory or a memory-mapped register is being referenced. When bit 28 is set to 0, local memory is being referenced. When bit 28 is set to 1, a memory-mapped register is being referenced.

When local memory is being referenced, the AM option uses address space partition bit 27 to determine whether SECDED will be performed on the data. When referencing local memory with SECDED protection, the AM option instructs the microprocessor to perform SECDED by setting bit 0 of the Read Data Acknowledge signal to 1. When referencing local memory without SECDED protection, the AM option instructs the microprocessor not to perform SECDED by setting bit 0 of the Read Data Acknowledge signal to 0 (refer to Table 6-18).

Table 6-18. Read Data Acknowledgement

Bits 2 – 0	Acknowledgement Type
000	Idle
100	OK_NCACHE_NCHK
101	OK_NCACHE
110	OK_NCHK
111	OK

Address Offset

During a reference to local memory, the AM option uses the address offset as the memory address that points to a location in local memory. The AM option sends the address offset to the AJ and AK options. The AJ and AK options present the address offset to the DRAM memory chips.

During a reference to a memory-mapped register, the AM option interprets the address offset to determine which memory-mapped register is being referenced. Once the memory-mapped register is identified, the AM option generates control to coordinate reading and writing of this register. For more information on how the AM option controls the memory-mapped register references, refer again to “AM Option” in Section 3, “Processing Element Node.”

PE Support Circuitry Address Interpretation

The address information that is sent to the AM option may be a virtual address or a logical address. It is the responsibility of the AM option to determine the type of address.

Virtual PE Address Interpretation

The AM option interprets the address as a virtual PE address when the system page bit from the AR option is set to 1 and the DTB annex write access is disabled. When the PE address is virtual, the AM option sets bit 12 of the destination phit to 0. This indicates to the EB option that the PE number (derived from the virtual PE address) is virtual and should be converted into a logical PE number.

Logical PE Address Interpretation

The AM option interprets the address as a logical PE address when the system page bit from the AR option is set to 0 and the DTB annex write access is enabled. When the address offset is logical, the AM option sets bit 12 of the destination phit to 1. This indicates to the EB option that the PE number is logical.

Network Interface Address Interpretation

The AM option sends the virtual or logical PE number to the network interface options (EA and EB). The AM option also appends an additional bit (bit 12) to the PE number to indicate whether the PE number is logical or virtual. When this bit is set to 1, the EB option interprets the PE number as logical. When this bit is set to 0, the EB option interprets the PE number as virtual and converts the virtual PE number into a logical PE number. The logical PE number is then converted into a routing tag.

Converting a Virtual PE Number into a Logical PE Number

The EB option uses the parameter information from the PE adjust memory-mapped register to convert a virtual PE number into a logical PE number (refer to Figure 6-43). This register is loaded with parameter information during system initialization. The PE adjust parameter information consists of the following bits:

- X mask bits 0 through 2
- X base node address bits 0 through 2
- X configuration mask bits 0 and 1
- Y shift count bits 0 and 1
- Y base node address bits 0 through 3
- Y mask bits 0 through 3
- Y configuration mask bits 0 and 1
- Z shift count bits 0 through 2
- Z base node address bits 0 through 2
- Z mask bits 0 through 2
- Z configuration mask bits 0 and 1

For more information on converting a virtual PE number to a logical PE number, refer again to “Converting a Virtual PE Number to a Logical PE Number” from the functional description portion of this section.

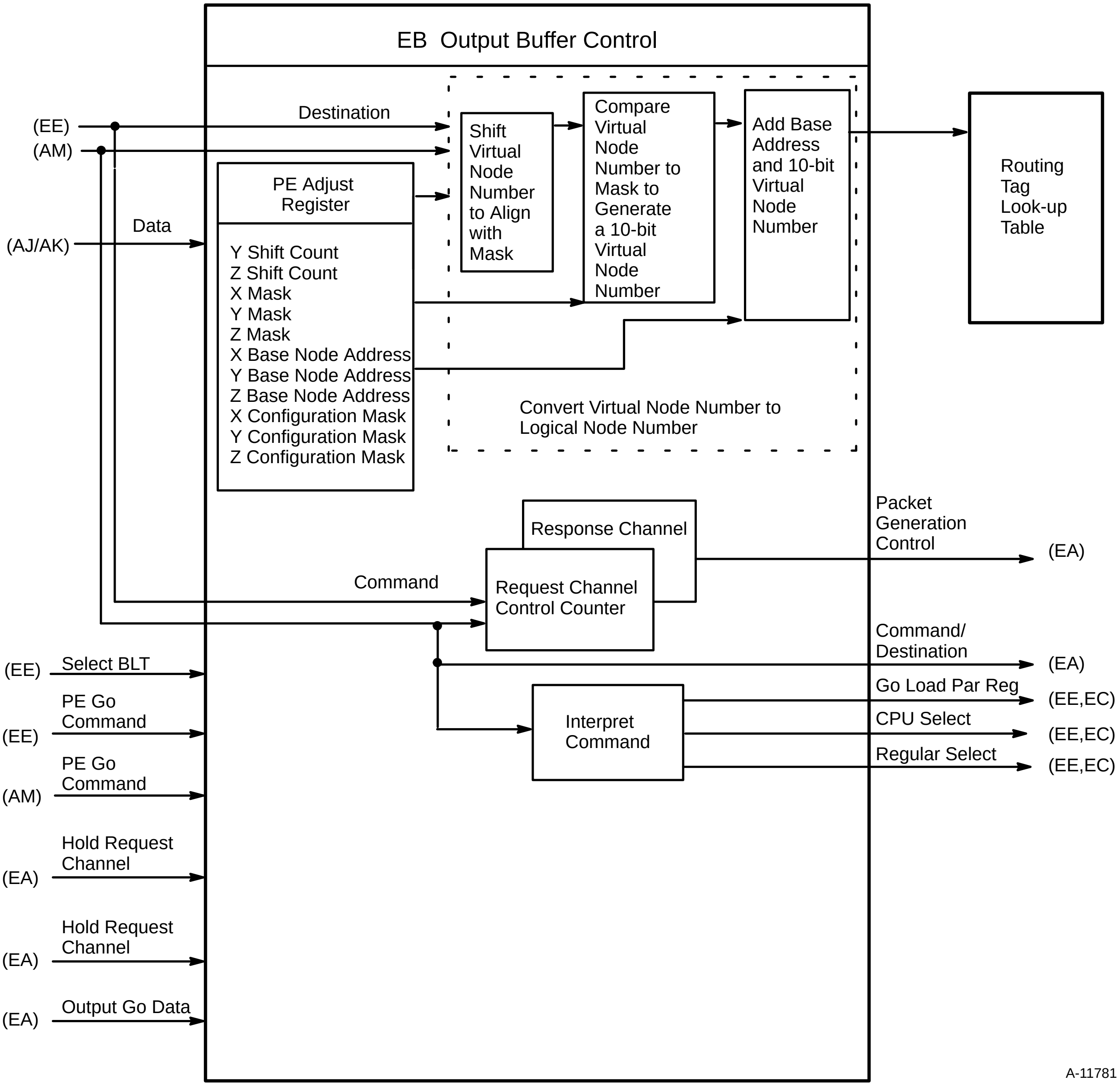


Figure 6-43. Virtual PE Number Conversion Circuitry in the EB Option

Converting a Logical PE Number into a Routing Tag

Once the EB option has the logical PE number (either from the AM or the conversion circuitry within the EB option), the EB option sends the logical PE number to the routing tag look-up table. The routing tag look-up table is used to store the routing tag for each logical PE number within the system.

To retrieve a routing tag for a logical PE number, the logical PE number is used to address a location within the routing tag look-up table. Each location within the routing tag look-up table contains a routing tag. These routing tags indicate the number of hops required in each dimension to transfer a packet from the source node to a destination node. This information is generated by a software program and loaded into the routing tag look-up table during system initialization.

NOTE: When a node is faulty, the routing tag look-up table is rewritten to delete the routing tag of the faulty node and replace it with the routing tag of a spare processing element node.

From the routing tag look-up table, the routing tag is sent to the EA option where it becomes part of an outgoing packet.

Creating a Packet

The routing tag and the other request information (command, destination PE number, and address) are assembled into a packet on the AM option before being transferred through the interconnect network. There are two classifications of packets: request packets and response packets. The following subsections describe the hardware involved with creating a request packet and a response packet.

Request Packet

The AM option formats the local address, remote address, command, and destination PE number needed for a request packet and forwards it to the EA and EB options. The AM option sends the local and remote addresses to the EA option and sends the command and destination PE number to the EB option (refer to Figure 6-44). During a remote write reference, the AM option also instructs the AJ and AK options to steer the data from the microprocessor to the EA option.

The EA option buffers the address and data in one of the request buffers. Parity is generated for the address as it is written into the buffer.

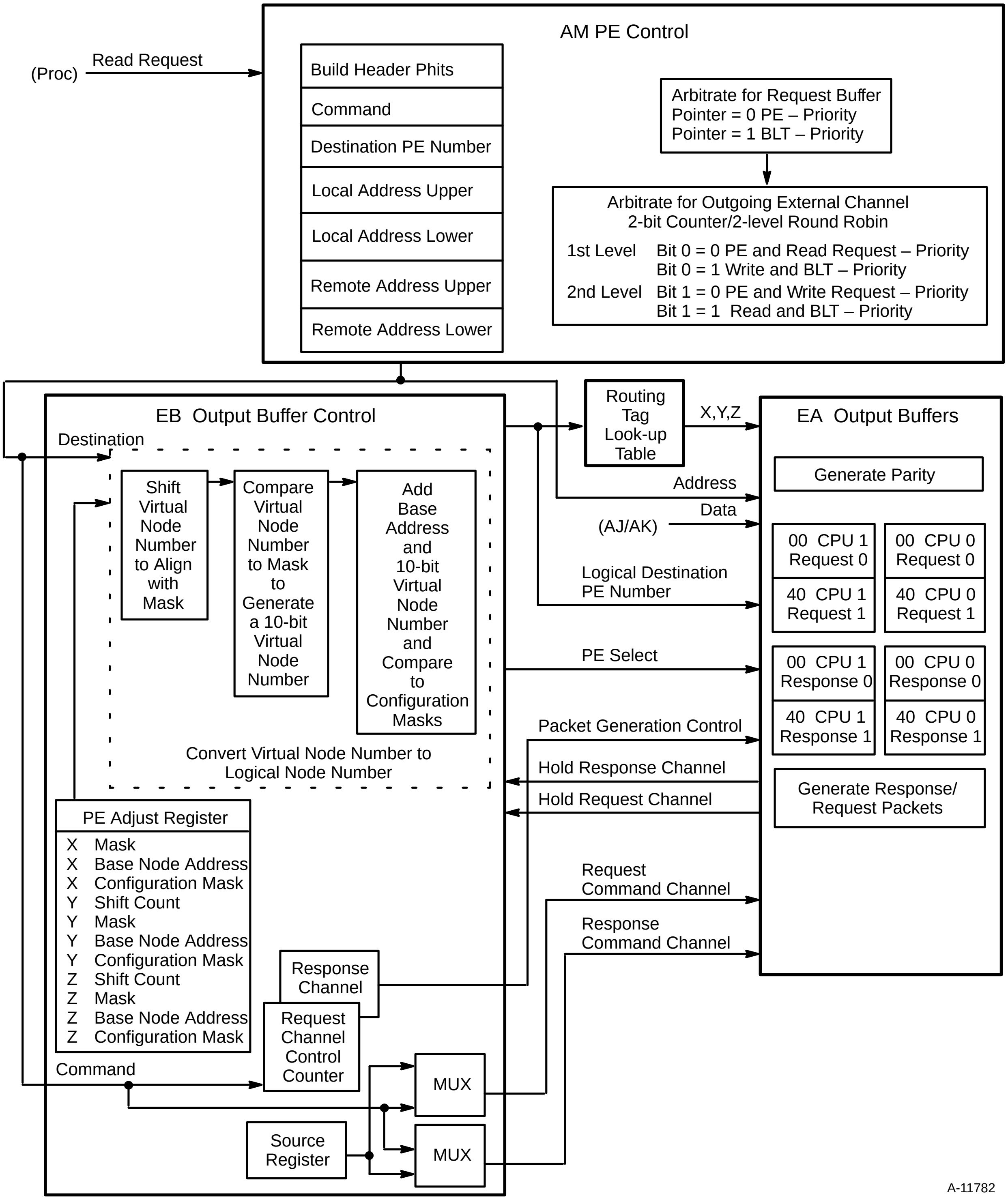


Figure 6-44. Communication Between Options While Creating a Request Packet

The EB option inputs the command and destination PE number and analyzes this information. From the command, the EB option derives the information needed to generate signals that control the packet assembly on the EA option (refer again to Figure 6-44). The EB option also determines whether the destination PE number is virtual or logical using bit 12 of the destination PE number. When the destination PE number is virtual, the EB option converts it into a logical PE number.

The EB option sends the command, destination PE number, and packet generation control to the EA option. The EB option also sends the contents of its source PE number register to the EA option and the logical PE number to the routing tag look-up table.

The logical PE number is used to address a location in the routing tag look-up table. This location contains the routing tag used to route the request packet through the network. This routing tag is sent to the EA option.

At this point, the EA option has the local and remote addresses, the command, the destination PE number, the source PE number, and the routing tag. All of this information will be assembled into a request packet. The EA option uses the packet generation control signals sent by the EB option to read the addresses from the request buffer; generate parity for the command, destination PE number, and the source PE number; and assemble this information into a packet (refer to Figure 6-45). For more information on the packet generation control signals, refer again to “EA Option” and “EB Option” in Section 3, “Processing Element Node.”

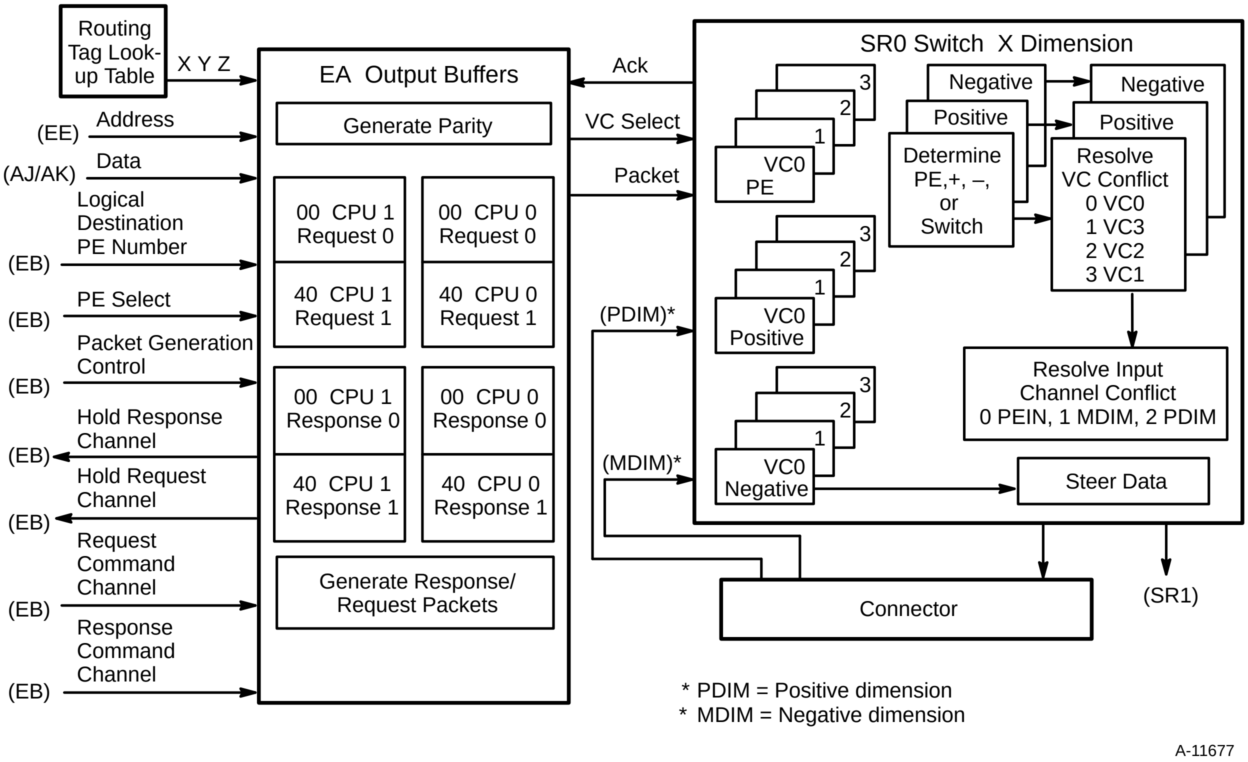


Figure 6-45. Request Packet Assembly

Under the control of the EB option, the EA option sends the request packet to the SR option that handles the X dimension of the interconnect network. From this option, the request packet is transferred through the network to the destination node.

Response packet

The AM option also formats the command, destination PE number, and address needed for a response packet. The AM option derives this information from an associated incoming request packet. (For example, the source PE number from a request packet becomes the destination PE number for the response packet.) The command and destination PE number are sent to the EB option, and the local address (if present) is sent to the EA option.

During a remote read reference, the AM option also instructs the AJ and AK options to send the data to the EA option. The EA option buffers the address and data in a response buffer. The EA option also receives the command, the destination PE number, and the source PE number information from the EB option (refer to Figure 6-46).

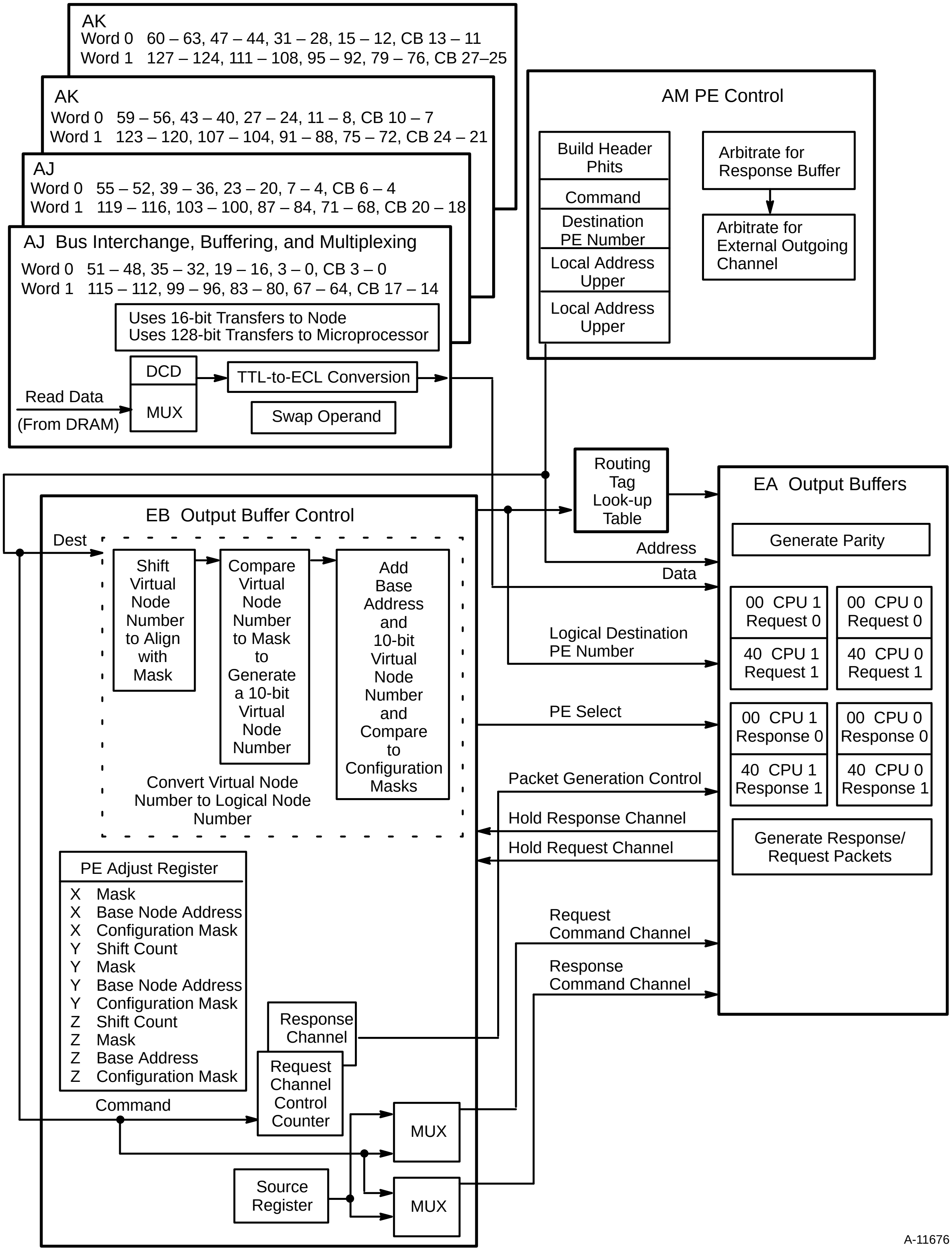

AK
Word 0 60 – 63, 47 – 44, 31 – 28, 15 – 12, CB 13 – 11
Word 1 127 – 124, 111 – 108, 95 – 92, 79 – 76, CB 27–25
AK
Word 0 59 – 56, 43 – 40, 27 – 24, 11 – 8, CB 10 – 7
Word 1 123 – 120, 107 – 104, 91 – 88, 75 – 72, CB 24 – 21
AJ
Word 0 55 – 52, 39 – 36, 23 – 20, 7 – 4, CB 6 – 4
Word 1 119 – 116, 103 – 100, 87 – 84, 71 – 68, CB 20 – 18
AJ Bus Interchange, Buffering, and Multiplexing
Word 0 51 – 48, 35 – 32, 19 – 16, 3 – 0, CB 3 – 0
Word 1 115 – 112, 99 – 96, 83 – 80, 67 – 64, CB 17 – 14
Uses 16-bit Transfers to Node
Uses 128-bit Transfers to Microprocessor
DCD
TTL-to-ECL Conversion
Read Data
(From DRAM)
MUX
Swap Operand
AM PE Control
Build Header Phits
Command
Destination PE Number
Local Address Upper
Local Address Upper
Arbitrate for Response Buffer
Arbitrate for External Outgoing Channel
Routing Tag Look-up Table
EA Output Buffers
Generate Parity
00 CPU 1 Request 0
00 CPU 0 Request 0
40 CPU 1 Request 1
40 CPU 0 Request 1
00 CPU 1 Response 0
00 CPU 0 Response 0
40 CPU 1 Response 1
40 CPU 0 Response 1
Generate Response/ Request Packets
Address
Data
Logical Destination PE Number
PE Select
Packet Generation Control
Hold Response Channel
Hold Request Channel
Request Command Channel
Response Command Channel
EB Output Buffer Control
Dest
Shift Virtual Node Number to Align with Mask
Compare Virtual Node Number to Mask to Generate a 10-bit Virtual Node Number
Add Base Address and 10-bit Virtual Node Number and Compare to Configuration Masks
Convert Virtual Node Number to Logical Node Number
PE Adjust Register
X Mask
X Base Node Address
X Configuration Mask
Y Shift Count
Y Mask
Y Base Node Address
Y Configuration Mask
Z Shift Count
Z Mask
Z Base Address
Z Configuration Mask
Response Channel
Request Channel Control Counter
Command
MUX
Source Register
MUX
A-11676

Figure 6-46. Collecting Information for the Response Packet

The EB option instructs the EA option to read the address and data from the response buffer; input the command, destination PE number, and the source PE number; and assemble this information into a response packet. The EA option sends the response packet to the SR option that handles the X dimension of the interconnect network (refer to Figure 6-47). From this option, the response packet is transferred through the network, back to the source node.

NOTE: For a response, the EB option generates parity for the command and the source PE number.

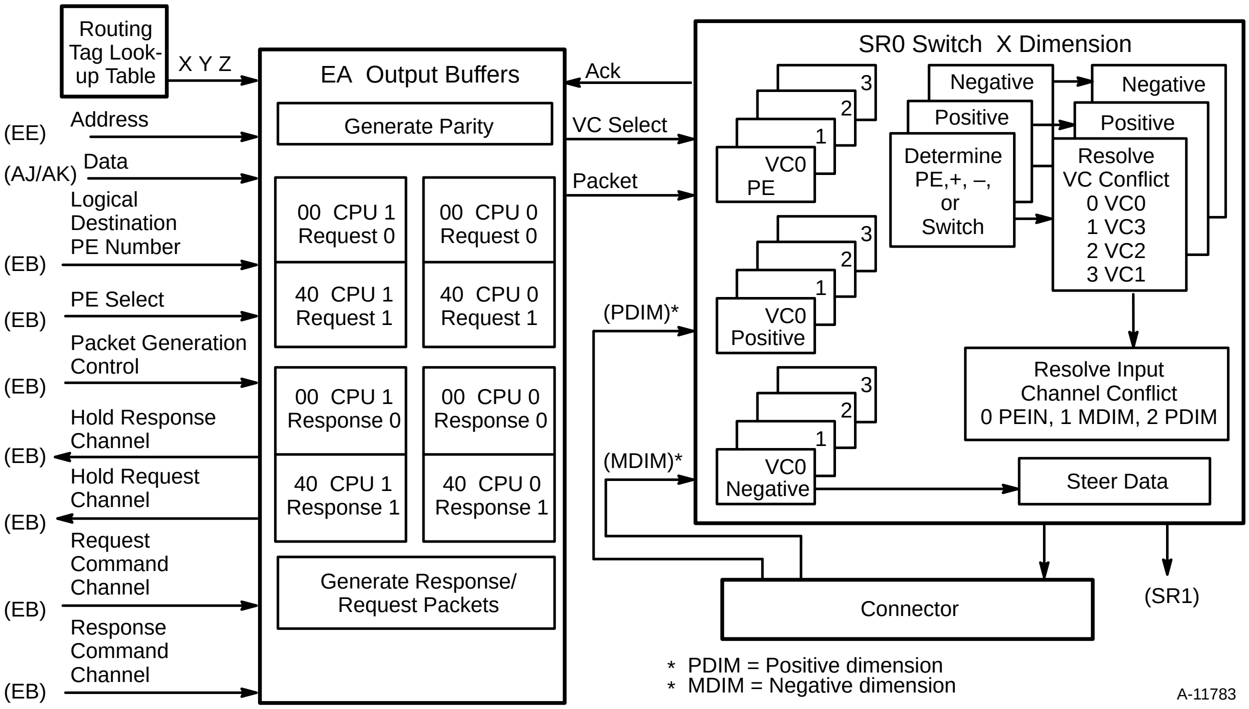

Routing Tag Look-up Table
X Y Z
EA Output Buffers
(EE) Address
(AJ/AK) Data
(EB) Logical Destination PE Number
(EB) PE Select
(EB) Packet Generation Control
(EB) Hold Response Channel
(EB) Hold Request Channel
(EB) Request Command Channel
(EB) Response Command Channel
Generate Parity
00 CPU 1 Request 0
00 CPU 0 Request 0
40 CPU 1 Request 1
40 CPU 0 Request 1
00 CPU 1 Response 0
00 CPU 0 Response 0
40 CPU 1 Response 1
40 CPU 0 Response 1
Generate Response/ Request Packets
Ack
VC Select
Packet
(PDIM)*
(MDIM)*
SR0 Switch X Dimension
3
2
1
VC0 PE
VC0 Positive
VC0 Negative
Negative
Positive
Determine PE,+, –, or Switch
Negative
Positive
Resolve VC Conflict 0 VC0 1 VC3 2 VC2 3 VC1
Resolve Input Channel Conflict 0 PEIN, 1 MDIM, 2 PDIM
Steer Data
Connector
(SR1)
* PDIM = Positive dimension
* MDIM = Negative dimension
A-11783

Figure 6-47. Response Packet Assembly

Register Locations

There are several memory-mapped registers associated with address interpretation. Table 6-19 lists these memory-mapped registers along with each register address, the option where the memory-mapped register resides, and the boolean terms within the option.

Table 6-19. Address Memory-mapped Registers

Address	Name	Description	Option	Terms
10400000_{16}	P_WHOAMI	Physical PE number register	AR option	ppe0 – 11
10450000_{16}	L_WHOAMI	Logical PE number register	AM option	lpe00 – 11
10454000_{16}	V_WHOAMI	Virtual PE number register	AM option	vpe1 – 10
10458000_{16}	VRT_RG	Virtual PE range mask register	AM option	vrm01 – 10
$106A0000_{16}$	X_WHOAMI	Network interface source register	EB option EC option	a80 – 90 j0 – 10
$106B0000_{16}$	LPE_XLATE	Network interface PE adjust register	EB option	l100 – 130
$106F0000_{16}$	ROUTE_LO	Routing tag look-up table low-order bits register	Routing tag look-up table	q01 – 04
$107F0000_{16}$	ROUTE_HI	Routing tag look-up table high-order bits register	Routing tag look-up table	q01 – 04

7 Operations

After receiving control, address, and data information from the microprocessor, the support circuitry performs a specific operation. The operation performed is determined by the cycle request code generated by the microprocessor, the function code stored in the DTB annex, and the value of bits 27 and 28 of the partial physical address. As was described in Section 6, “Addressing,” when bit 28 of the partial physical address is set to 1, the support circuitry transfers information with a memory-mapped register (regardless of the function code in the DTB annex). This section provides a functional and hardware description of the operations performed when bit 28 is set to 0.

Functional Description

The PE support circuitry interprets a cycle request from the microprocessor and a function code from the DTB annex to determine the type of request being made by the microprocessor. Once the type of request is known, the PE support circuitry generates the necessary control signals to carry out the request.

Cycle Requests and Function Codes

The microprocessor presents cycle request codes to the support circuitry by using the cycle request pins. Table 7-1 is a quick reference table that lists all of the cycle request codes. More information on the cycle request codes is provided in Section 10, “Control and Status.”

Table 7-1. Cycle Request Codes

Cycle Request Pins $2^2 – 2^0$	Cycle
000	IDLE
001	BARRIER
010	FETCH
011	FETCHM
100	READ_BLOCK

Table 7-1. Cycle Request Codes (continued)

Cycle Request Pins $2^2 – 2^0$	Cycle
101	WRITE_BLOCK
110	LDxL
111	STxC

Each cycle request code corresponds to a specific group of microprocessor instructions; however, the microprocessor may not immediately present the cycle request code to the support circuitry when the microprocessor issues an instruction. In certain cases, the microprocessor may also present cycle request codes to the support circuitry in a different order than the microprocessor issued the instructions. As an example, the following paragraphs describe how the WRITE_BLOCK cycle request code for a store instruction may be presented to the support circuitry several clock periods after the microprocessor actually issued the store instruction.

When the microprocessor issues a store instruction, hardware in the microprocessor places data and address information for the store instruction in the write buffer of the microprocessor. The hardware merges store instructions which modify portions of the same cache line in an attempt to assemble a cache-line amount (hereafter referred to as a block) of data before sending the data to the support circuitry. The following factors determine when the data and address information leaves the write buffer.

- At least two of the four lines in the write buffer contain valid data.
- One line in the write buffer contains valid data for 256 clock pulses and the microprocessor has not issued a write-buffer related instruction within that time.
- One of the lines in the write buffer contains information for a memory barrier (MB) or store integer register into memory conditional (STX_C) instruction. In this case, the write buffer circuitry sends all the valid data to the support circuitry until the write buffer is empty.

- The microprocessor issues a load instruction that requires the use of the support circuitry (the requested data is not in the data cache) and the address for the load instruction matches the address for a write related instruction stored in the write buffer. In this case, the write buffer circuitry sends all the valid data to the support circuitry until the write buffer is empty. This ensures that a write related instruction accesses a memory location before the subsequent read instruction does.

When data and address information for the store instruction (possibly merged store instructions) leaves the write buffer, the microprocessor provides a WRITE_BLOCK cycle request code to the support circuitry.

The microprocessor provides the support circuitry with a valid partial physical address along with the cycle request code (refer to Figure 7-1). For more information on the partial physical address, refer again to Section 6, “Addressing.”

The DTB annex index portion of the partial physical address points to an entry in the DTB annex. This entry contains a function code and a PE number. The function code indicates to the support circuitry what type of memory function to perform during an operation.

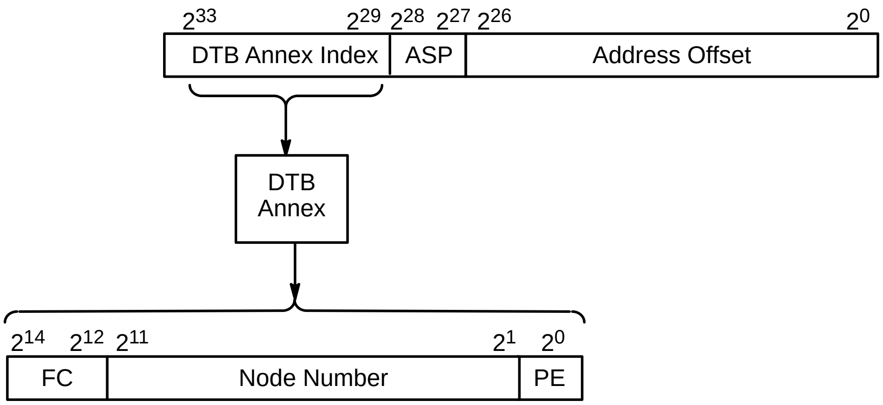


Figure 7-1. Partial Physical Address and DTB Annex Entry

Table 7-2 is a quick reference table for the function codes. Each function code indicates a different type of memory function performed during an operation. For example,when bit 14 of the function code is set to 1, data from a read operation is sent to the microprocessor, and the support circuitry instructs the microprocessor to update the data cache. When bit 14 of the function code is set to 0, data from a read operation is sent to the microprocessor, and the microprocessor is instructed not to update the data cache.

Each function code corresponds to a memory function but does not necessarily correspond to a unique operation. The names listed in Table 7-2 are only provided to give each function code a unique identification.

Table 7-2. Function Codes

Bit 14	Bit 13	Bit 12	Function
0	0	0	Write or noncacheable read
0	0	1	Write or noncacheable atomic swap
0	1	0	Fetch-and-increment write or read
0	1	1	Reserved (may cause an error interrupt)
1	0	0	Write or cached read
1	0	1	Write or cached atomic swap
1	1	0	Write or cached read ahead
1	1	1	Message or cached read

Table 7-3 lists all the combinations of function code (FC) and cycle requests and shows the operations that the support circuitry performs for each combination. To the microprocessor, the support circuitry performs the same operations for a FETCHM cycle request code as it does for a FETCH; however, the support circuitry may perform a different function during a FETCHM than it does during a FETCH. More information on the FETCH and FETCHM cycle requests is provided in “FETCH Versus FETCHM” later in this section.

Table 7-3. Cycle Request Types, Function Codes, and Operations

	Cycle Request Code						
FC	BARRIER	LDxL	STxC	READ_BLOCK	WRITE_BLOCK	FETCH	FETCHM
000	Memory barrier	DTB annex read	DTB annex write	Noncacheable read	Write	Prefetch read	Prefetch read
001	Memory barrier	DTB annex read	DTB annex write	Noncacheable atomic swap	Write	Prefetch atomic swap	Prefetch atomic swap
010	Memory barrier	DTB annex read	DTB annex write	Fetch-and-increment read	Fetch-and-increment write	Prefetch fetch-and-increment read	Prefetch fetch-and-increment read
011	Memory barrier	DTB annex read	DTB annex write	Error interrupt	Error interrupt	Error interrupt	Error interrupt
100	Memory barrier	DTB annex read	DTB annex write	Cached read	Write	Prefetch read	Prefetch read
101	Memory barrier	DTB annex read	DTB annex write	Cached atomic swap	Write	Prefetch atomic swap	Prefetch atomic swap
110	Memory barrier	DTB annex read	DTB annex write	Cached read ahead	Write	Prefetch read	Prefetch read
111	Memory barrier	DTB annex read	DTB annex write	Cached read	Message write	Prefetch read	Prefetch read

The operations listed in Table 7-3 are classified into seven types of operations:

- Memory barrier
- DTB annex read or write
- Reads
- Writes
- Fetch-and-increment
- Messaging facility
- Prefetch

Memory Barrier

The memory barrier operation is performed by the support circuitry after the microprocessor issues a memory barrier (MB) instruction. The MB instruction is used to guarantee that load or store instructions issued after the MB instruction will not access memory until after any load or store instructions issued before the MB instruction have accessed memory.

Because the microprocessor empties the write buffer when issuing an MB instruction, the MB instruction is also used to control when write information is presented to the support circuitry. More information on the write buffer is provided in Section 3, “Processing Element Node.”

When the microprocessor presents a barrier cycle request code to the support circuitry, the support circuitry performs a memory barrier operation regardless of what function code is set in the DTB annex or what address is on the pins of the microprocessor.

DTB Annex Read or Write

When the microprocessor issues a load from memory to register locked (LDQ_L) or a store integer register data into memory conditional (STQ_C) instruction, the microprocessor exchanges information with the DTB annex. (The LDL_C and STL_C instructions may produce unreliable results and should not be used.) These instructions are not implemented in the CRAY T3D system as they are described in the *Alpha Architecture Handbook* or the *DECChip™ 21064-AA RISC Microprocessor Preliminary Data Sheet*.

The LDQ_L instruction is used to read an entry of the DTB annex. The address used with an LDQ_L instruction must be cache-line aligned (bits 0 through 4 of the byte-oriented address set to 0, and bits 29 through 33 set to the appropriate entry in the DTB annex). When the microprocessor issues an LDQ_L instruction, the corresponding line in the data cache is invalidated even though the returned data is not stored in the data cache.

NOTE: This characteristic occurs only when the support circuitry contains an 8AM option, revision 9J or later.

The STQ_C instruction is used to store a new PE number and function code in the DTB annex. The address used with a STQ_C instruction must also be cache-line aligned, with bits 29 through 33 set to the appropriate entry in the DTB annex.

Bits 0 and 1 of the system control register (SCR) enable or disable the ability to write to an entry in the DTB annex. More information on the SCR is provided in Section 10, “Control and Status.” More information on the data cache is provided in Section 3, “Processing Element Node.”

Reads

Read operations transfer data from system memory to a register in the microprocessor. Depending on the type of read operation, the microprocessor may also update information in the data cache. Except when the function code is set to 2 or 3, the support circuitry performs a read operation when the microprocessor sets the cycle request code to READ_BLOCK.

The microprocessor may initiate a read operation when it issues a load instruction. When the microprocessor issues a load instruction, it converts the byte-oriented virtual address used in the instruction into a byte-oriented partial physical address. The partial physical address points to the first byte of data requested by the load instruction.

The microprocessor then compares the partial physical address to the address tag in a line of the data cache. When it finds a match, the microprocessor reads the requested amount of data from the line in the data cache. When it does not find a match, hardware in the microprocessor presents the partial physical address to the support circuitry along with a READ_BLOCK cycle request code.

The partial physical address contains a DTB annex index. The support circuitry retrieves the function code and PE number from the DTB annex entry addressed by the DTB annex index. The value of the function code read from the DTB annex determines which read operation the support circuitry performs.

There are two main types of read operations:

- noncacheable reads
- cached reads

Bit 14 of the function code in the DTB annex indicates which main type of read operation the support circuitry will perform. When set to 0, the support circuitry will perform a noncacheable read operation. When set to 1, the support circuitry will perform a cached read operation. The following subsections describe the types of noncacheable and cached read operations.

Noncacheable Reads

When the support circuitry receives a READ_BLOCK cycle request code and a partial physical address, the support circuitry retrieves the function code and PE number from the DTB annex. When bit 14 of the function code is set to 0 (except for function code 2 and 3), the support circuitry performs one of the noncacheable memory read operations. When the function code is set to 2, the support circuitry performs a fetch-and-increment register read operation, which is always noncacheable. More information on the fetch-and-increment register read operation is provided in “Fetch-and-increment” later in this section.

The noncacheable memory read operations signal the support circuitry to transfer data from system memory to the microprocessor. As the support circuitry transfers the data to the microprocessor, the support circuitry signals the microprocessor not to place the data in a line of the data cache (i.e. the microprocessor does not update the data cache). The microprocessor simply places the requested data in the corresponding register in the microprocessor and discards the remainder of the returned block of data. The requested data may be a 32-bit halfword or a 64-bit word.

As the microprocessor receives data from the support circuitry, the microprocessor performs single-error correction/double-error detection (SECDED). More information on how the microprocessor performs SECDED is provided in Section 3, “Processing Element Node.”

There are two types of noncacheable memory read operations: a normal noncacheable read and a noncacheable atomic swap. To the microprocessor, both noncacheable read operations look identical; however, the support circuitry performs different memory functions during each operation.

Normal Noncacheable Read

When the function code read from the DTB annex is set to 0, the support circuitry performs the normal noncacheable read (hereafter referred to as a noncacheable read) operation. During the noncacheable read operation, the support circuitry first checks the value of the PE number read from the DTB annex.

When the PE number is set to the local PE, the support circuitry retrieves data from local memory. The support circuitry then sends the data to the microprocessor and signals the microprocessor that it should not update the data cache.

When the PE number is set to a remote PE, the support circuitry creates a single-word read request packet and sends the packet to the remote PE. After receiving the request packet, the support circuitry in the remote PE retrieves data from memory. The support circuitry then creates a single-word read response packet and sends it to the PE that requested the noncacheable read operation.

After receiving the single-word read response packet, the support circuitry in the PE that requested the noncacheable read operation sends the data to the microprocessor and signals the microprocessor not to update the data cache.

Noncacheable Atomic Swap Read

When the function code read from the DTB annex is set to 1, the support circuitry performs a noncacheable atomic swap read (hereafter referred to as an atomic swap) operation. An atomic swap operation reads a 64-bit word from a location in system memory and then writes a 64-bit word into the same location in an indivisible operation.

NOTE: Avoid issuing a write operation when the function code in the DTB annex is set to 1.

Before initiating the atomic swap operation, the microprocessor should have loaded a 64-bit word in a memory-mapped register called the swaperand register. More information on the swaperand register is provided in “Register Mapping” later in this section.

During the atomic swap operation, the support circuitry first checks the value of the PE number read from the DTB annex. When the PE number is set to the local PE, the support circuitry retrieves a 64-bit word from local memory. Immediately after reading the word, the support circuitry transfers the word from the swaperand register to the same local memory location. The support circuitry then sends the word read from local memory to the microprocessor and signals the microprocessor not to update the data cache.

When the PE number is set to a remote PE, the support circuitry creates a single-word request packet that contains a copy of the value read from the swaperand register and sends the packet to the remote PE. After receiving the request packet, the support circuitry in the remote PE reads a word of data from memory. Immediately after reading the word, the support circuitry transfers the swaperand value from the atomic swap request packet to the same local memory location. The support circuitry in the remote PE then creates an atomic swap response packet containing the read data and sends the packet to the PE that requested the atomic swap operation.

After receiving the atomic swap response packet, the support circuitry in the PE that requested the atomic swap operation delivers the word of data from the response packet to the microprocessor and signals the microprocessor not to update the data cache.

The atomic swap operation is used to synchronize functions between PEs. For example, the following paragraphs describe how the atomic swap operation may be used to lock and unlock a memory location.

When a memory location contains the number 5 and a programmer instructs two PEs to each add the number 2 to this memory location, the final value in the memory location should be 9. To ensure that the final value is correct, a programmer may use atomic swap operations to lock and unlock the memory location.

Before performing the first atomic swap, the microprocessor in one of the PEs (PE A) loads an invalid number into its swaperand register. In this example, the value –99 is not a valid number.

NOTE: After system deadstart, when an atomic swap is initiated without storing an initial value in the swaperand register, memory errors may be caused by uninitialized check bits in the swaperand register.

After loading the swaperand register with –99, the microprocessor in PE A performs an atomic swap operation with the memory location that contains the number 5 (refer to Figure 7-2). After the atomic swap, the microprocessor in PE A receives the number 5, and the memory location contains –99.

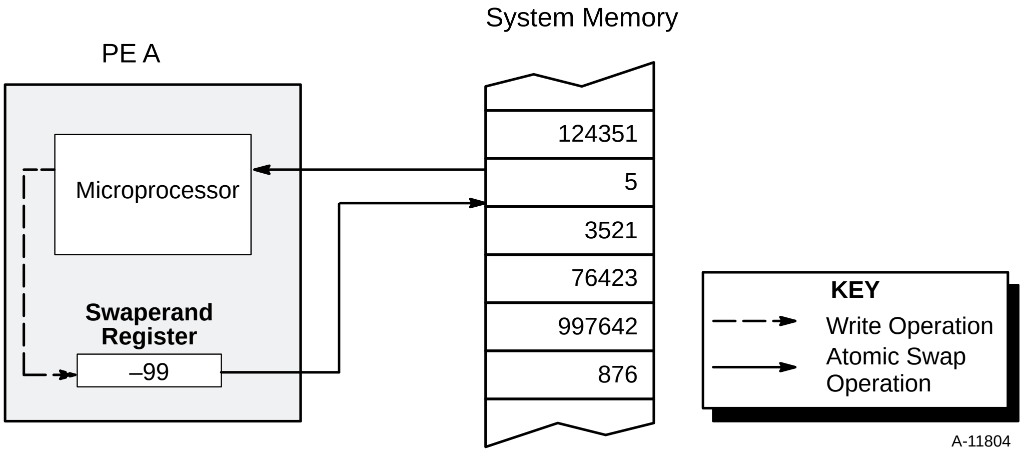


Figure 7-2. PE A Atomic Swap Operation

The microprocessor in the other PE (PE B) now loads its swaperand register with –99 and performs an atomic swap operation with the same memory location that PE A used. After the atomic swap, the microprocessor in PE B receives a value of –99, and the memory location contains –99. This indicates to the microprocessor in PE B that another PE is using the memory location. The software in PE B continues to perform atomic swap operations until the value returned is not –99.

The value stored in the swaperand register does not change unless the microprocessor writes a new value into the swaperand register. Because of this characteristic, the microprocessor in PE B can repeatedly perform the atomic swap operation without writing to the swaperand register before issuing each atomic swap operation.

After adding 2 to the number 5, the microprocessor in PE A writes the number 7 into the same memory location as before. This function overwrites the –99 value and unlocks the memory location.

When the microprocessor in PE B performs another atomic swap operation (refer to Figure 7-3), it receives the number 7 and the memory location contains –99. This function locks the memory location again.

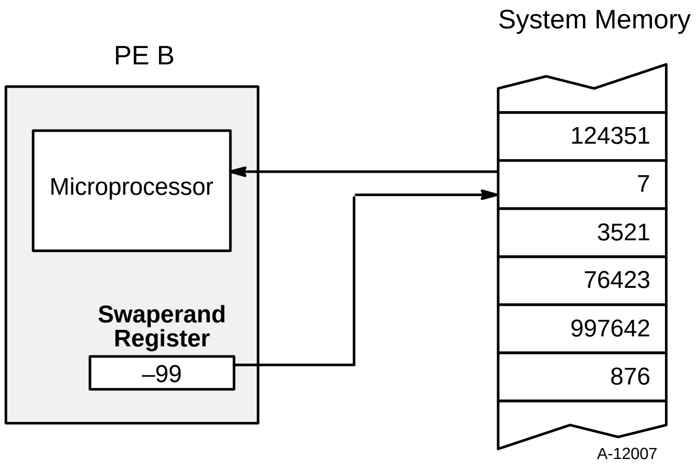


Figure 7-3. PE B Atomic Swap Operation

After adding 2 to the number 7, the microprocessor in PE B writes the number 9 into the same memory location as before. This function overwrites the –99 value, which unlocks the memory location. The memory location now contains the correct final value.

Cached Reads

When the support circuitry receives a READ_BLOCK cycle request code and a partial physical address, the support circuitry retrieves the function code and PE number from the DTB annex. When bit 14 of the function code is set to 1, the support circuitry performs one of the cached read operations.

During a cached read operation, the support circuitry transfers a block of data from system memory to the microprocessor. As the support circuitry does this, it signals the microprocessor to place the block of data in the data cache (update the data cache).

The microprocessor places the data in the cache line addressed by bits 5 through 12 of the partial physical address and completes the associated register load operation. As the microprocessor is receiving data from the support circuitry, it performs SECDED. More information on how the microprocessor performs SECDED is provided in Section 3, “Processing Element Node.”

There are three types of cached read operations: a normal cached read, a cached read ahead, and a cached atomic swap. To the microprocessor, all three cached read operations look identical; however, the support circuitry performs different memory functions during each operation.

Normal cached Read

When the function code read from the DTB annex is set to 4 or 7, the support circuitry performs a normal cached read operation (hereafter referred to as a cached read). During the cached read operation, the support circuitry first checks the value of the PE number read from the DTB annex.

When the PE number is set to the local PE, the support circuitry retrieves a block of data from local memory. The support circuitry then sends the block of data to the microprocessor and signals the microprocessor to update the data cache.

When the PE number is set to a remote PE, the support circuitry creates a request packet (requesting 4 words) and sends the packet to the remote PE. After receiving the request packet, the support circuitry in the remote PE retrieves a block of data from memory. The support circuitry in the remote PE then creates a 4-word read response packet and sends the packet to the PE that requested the cached read operation.

After receiving the 4-word read response packet, the support circuitry in the PE that requested the cached read operation sends the block of data to the microprocessor and signals the microprocessor to update the data cache.

Cached Read Ahead

When the function code read from the DTB annex is set to 6 and the PE number is set to the local PE, the support circuitry performs the cached read ahead operation. When the PE number is set to a remote PE, the support circuitry does not perform the cached read ahead operation. Instead, the support circuitry performs a cached read operation.

The support circuitry contains a local memory read stage. After the support circuitry performs a cached read ahead operation, the local memory read stage buffers a block of data (or instruction fetches) read from local memory. When the microprocessor issues any type of cached or noncacheable read operation (except atomic swaps) with an address that matches the buffered block of data, data transfers from the local memory read stage to the microprocessor. This action prevents the support circuitry from having to access DRAM memory to retrieve the block of data and decreases the latency of the read operation.

During a cached read ahead operation, the support circuitry retrieves a block of data from local memory (or the local memory read stage). The support circuitry then sends the block of data to the microprocessor and signals the microprocessor to update the data cache.

Immediately after sending the data to the microprocessor, the support circuitry retrieves the next sequential block of data from local memory and buffers the data in the local memory read stage of the support circuitry. The data buffered in the support circuitry remains in the support circuitry until the `MB` instruction is issued or a data or instruction load operation from a different local memory address occurs.

Because of how data is buffered in the local memory read stage, it is possible for an external source (PE or BLT) to change the contents of a local memory location while the previous content of that location is buffered in the local memory read stage of the support circuitry.

A read ahead operation for instruction fetches may also be enabled by writing a 1 to bit 11 of the DRAM control register (DRAM_CR). More information on the DRAM_CR is provided in Section 10, “Control and Status.” When the instruction fetch read ahead operation and the data read ahead operation are both enabled, system performance could be negatively affected under certain conditions.

Cached Atomic Swap Read

When the function code read from the DTB annex is set to 5, the support circuitry performs the cached atomic swap read (hereafter referred to as a cached atomic swap) operation. Like a noncacheable atomic swap operation, the cached atomic swap operation reads a 64-bit word from a location in system memory and then writes a 64-bit word into the same location in an indivisible operation; however, during a cached atomic swap the support circuitry reads a block of data from local memory that contains the word to be swapped and signals the microprocessor to update the data cache.

NOTE: Avoid issuing a write operation when the function code in the DTB annex is set to 5.

Before initiating the cached atomic swap operation, the microprocessor should have loaded a 64-bit word in the swaperand register. More information on the swaperand register is provided in “Register Mapping” later in this section.

During the cached atomic swap operation, the support circuitry reads a block of data from local memory. This block of data contains the 64-bit word that will be swapped.

Immediately after reading the block of data, the support circuitry transfers the word from the swaperand register to the appropriate word in the block of local memory. The support circuitry then sends the block read from local memory to the microprocessor and signals the microprocessor to update the data cache.

Although the cached atomic swap operation may be performed with a remote PE, the cache-line invalidate filter cannot be used to ensure a data cache line is valid compared to a remote memory location. Because of this characteristic, the cached atomic swap operation should, in general, only be performed in the local PE.

The cached atomic swap operation is typically used in conjunction with the cache-line invalidate filter mechanism. This feature enables the microprocessor to continuously poll a location in local memory without unnecessarily consuming local memory bandwidth. More information on the data cache and the cache-line invalidate filter is provided in Section 10, “Control and Status,” and in Section 3, “Processing Element Node.”

Writes

Write operations transfer data from the write buffer in the microprocessor to system memory. Except when the function code is set to 2, 3, or 7, the support circuitry performs a normal memory write operation (hereafter referred to as a write) when the microprocessor sets the cycle request code to WRITE_BLOCK.

The microprocessor may initiate a write operation when it issues a store instruction. When the microprocessor issues a store instruction, it converts the byte-oriented virtual address used in the instruction into a byte-oriented partial physical address. The partial physical address points to the first byte of data that will be written in system memory.

After issuing the store instruction, hardware in the microprocessor compares the partial physical address with the address tag in each line of the data cache. When the hardware finds a match, the hardware updates the specified amount of data in the data cache line. When the hardware does not find a match, the hardware does not update the data cache.

The hardware then stores the data and address information for the store instruction in a new line of the write buffer or merges the data with an existing line in the write buffer. The hardware then updates the mask bits for the write buffer line to indicate which 32-bit halfwords in the write buffer line are valid. More information on the write buffer is provided in Section 3, “Processing Element Node.”

As was previously described, several conditions determine when information from a store instruction leaves the write buffer. When the information does leave the write buffer, the microprocessor provides a WRITE_BLOCK cycle request code, eight 32-bit halfwords of data from the write buffer line, newly generated check bits for the data, the write buffer line mask bits, and the partial physical address to the support circuitry.

When the support circuitry receives a WRITE_BLOCK cycle request code and partial physical address, the support circuitry first retrieves the function code and PE number from the DTB annex. When the function code is set to 0, 1, 4, 5, or 6, the support circuitry performs a write operation.

During a write operation, the support circuitry uses bits 5 through 26 of the partial physical address as an address offset that points to a block of data in the memory of a PE. The support circuitry also uses the mask bits to determine which of the eight 32-bit halfwords it received from the microprocessor are valid and should be written to system memory.

The support circuitry then checks the value of the PE number read from the DTB annex. When the PE number is set to the local PE, the write data is for local memory. When the PE number is set to a remote PE, the write data is for a remote PE.

When the write data is for local memory, the support circuitry receives the eight 32-bit halfwords and mask bits from the microprocessor. The support circuitry then writes the valid halfwords into the corresponding local memory locations.

When the write data is for a remote PE, the support circuitry creates one of two write request packets. One of the write request packets has a 4-word body. The body of the packet contains all 8 halfwords received from the microprocessor, and the command phit in the header of the packet contains the mask bits.

The other type of write request packet has a 1-word body. The body of the packet contains 2 halfwords received from the microprocessor, and the command phit in the header of the packet contains 2 mask bits. The position of the valid data in a write buffer line determines which 2 mask bits are used.

After creating a write request packet, the support circuitry increments a counter (called the outstanding write request counter) that counts the number of write request packets created and sent to remote PEs. The support circuitry then sends the write request packet to the remote PE. The support circuitry in the remote PE then writes the valid data into memory and creates a write response packet. The support circuitry in the remote PE sends the response packet to the PE that requested the write operation.

After receiving a write response packet, the support circuitry in the PE that requested the write operation decrements the outstanding write request counter. This action completes a write operation.

The microprocessor can check for outstanding write requests by reading bit 10 (remote writes are outstanding bit) of the user control and status register (USR). When set to 1, this bit indicates that one or more remote write operations have been started, but the support circuitry has not received the corresponding write response packets. When set to 0, this bit indicates that all remote write operations have completed. (Outstanding remote write operations do not include the message writes but do include fetch-and-increment writes).

When the microprocessor sends data for a write operation to the support circuitry, it also transfers new check bits for the data. More information on the check bits and on how the microprocessor performs SECDED is provided in Section 3, “Processing Element Node.”

Fetch-and-increment

The fetch-and-increment operation transfers a 32-bit value between the microprocessor and a fetch-and-increment register. Each processing element node contains two fetch-and-increment registers; however, the fetch-and-increment registers function independently of the PEs and may be used by any PE in a partition. A PE can read from or write to any fetch-and-increment register.

NOTE: A problem may occur when software requests a fetch-and-increment register write immediately after requesting a fetch-and-increment register read from the same fetch-and-increment register (or when software requests a read immediately after requesting a write). Due to the way in which hardware handles network buffer conflicts, the write (or read) request information could pass the read (or write) request information in the network.

To avoid this problem, software should not issue a fetch-and-increment register write (or read) for a register until the previously issued fetch-and-increment register read (or write) for that register completes.

Fetch-and-increment Register Read

The fetch-and-increment read operation transfers one 32-bit value from a fetch-and-increment register to the microprocessor. Immediately after the value is transferred out of the fetch-and-increment register, hardware automatically increments the value stored in the fetch-and-increment register by one.

The microprocessor initiates a fetch-and-increment read operation by issuing an LDQ instruction with a partial physical address that points to an entry in the DTB annex. The microprocessor presents the partial physical address to the support circuitry along with a READ_BLOCK cycle request.

When the support circuitry receives a READ_BLOCK cycle request and partial physical address, the support circuitry retrieves a function code and PE number from the DTB annex. When the function code is set to 2, the support circuitry performs a fetch-and-increment read operation.

During a fetch-and-increment read operation, the support circuitry does not use the address offset portion of the partial physical address; however, the address offset must be cache-line aligned and is recommended to be set to 0. Also, the ASP bits must be set so that bit 28 is 0 and bit 27 is 1 or 0.

When the function code is set to 2 and the support circuitry receives a READ_BLOCK cycle request, the support circuitry interprets the PE number portion of an entry in the DTB annex as a fetch-and-increment register address (refer to Figure 7-4).

The fetch-and-increment register address contains two parts: a node number and a fetch-and-increment (F&I) register bit (refer again to Figure 7-4). The node number is a logical or virtual node number that indicates the node in which the fetch-and-increment register is located. The F&I bit indicates when the fetch-and-increment register is register 0 or register 1.

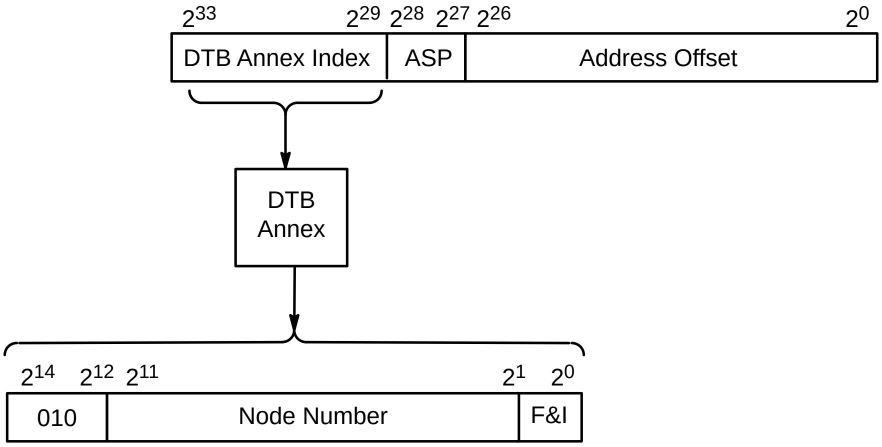


Figure 7-4. Fetch-and-increment Register Addressing

Because the fetch-and-increment registers function independently of the PEs, the support circuitry always interprets the fetch-and-increment register address as pointing to a remote node. The support circuitry in the node creates a fetch-and-increment read request packet and sends the packet to the destination node.

After receiving the fetch-and-increment read request packet, circuitry in the destination node retrieves the value stored in the fetch-and-increment register. Immediately after retrieving the value, circuitry in the destination node increments the value stored in the fetch-and-increment register by 1. The circuitry in the destination node then creates a fetch-and-increment read response packet and sends the response packet to the PE that requested the fetch-and-increment read function.

The body of the fetch-and-increment read response packet contains the 32-bit value that was read from the fetch-and-increment register and a 32-bit copy of that value. Figure 7-5 shows the actual format of the fetch-and-increment register.

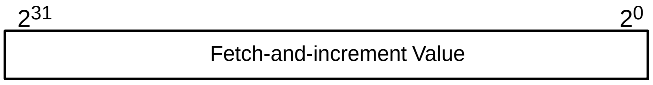


Figure 7-5. Fetch-and-increment Register Format

After receiving the fetch-and-increment read response packet, the support circuitry in the PE that requested the fetch-and-increment read sends the two values from the response packet to the microprocessor. The microprocessor then uses the two values to ensure that the body of the packet was not corrupted while traveling through the interconnect network.

Because the value stored in the fetch-and-increment register changes every time the register is read, the value read from the fetch-and-increment register is not protected by SECDED check bits. Instead, software must compare the two values that the microprocessor received from a fetch-and-increment response packet.

When the values are the same, the body portion of the packet was not corrupted while traveling through the network and the fetch-and-increment value is valid. When the values are not the same, the body portion of the packet was corrupted while traveling through the network and the fetch-and-increment value cannot be used.

Fetch-and-increment Register Write

The fetch-and-increment register write operation transfers one 32-bit value from the microprocessor to a fetch-and-increment register. The fetch-and-increment register write (hereafter referred to as a fetch-and-increment write) operation is used to initialize the value stored in the fetch-and-increment register.

To initiate a fetch-and-increment write operation, the microprocessor issues an STQ instruction with a partial physical address that points to the appropriate entry in the DTB annex. The microprocessor presents the partial physical address to the support circuitry along with a WRITE_BLOCK cycle request code.

When the support circuitry receives a WRITE_BLOCK cycle request and partial physical address, the support circuitry retrieves a function code and PE number from the DTB annex. When the function code is set to 2, the support circuitry performs a fetch-and-increment write operation.

NOTE: Always issue a memory barrier (MB) instruction after issuing a fetch-and-increment register write operation.

During a fetch-and-increment write operation, the support circuitry does not use the address offset portion of the partial physical address; however, the address offset must be cache-line aligned and is recommended to be set to 0. Also, the ASP bits must be set so that bit 28 is 0 and bit 27 is 1 or 0.

When the function code is set to 2 and the support circuitry receives a WRITE_BLOCK cycle request, the support circuitry interprets the PE number portion of an entry in the DTB annex as a fetch-and-increment register address (refer again to Figure 7-4).

Because the support circuitry always interprets a fetch-and-increment register address as pointing to a remote node, the support circuitry creates a fetch-and-increment write request packet. The request packet contains a 1-word body. The body contains the 32-bit value for the fetch-and-increment register and a 32-bit value that is not used and may be undefined.

After creating a fetch-and-increment write request packet, the support circuitry increments the counter that counts the number of write request packets created and sent to remote PEs. The support circuitry then sends the request packet to the remote PE.

After receiving the fetch-and-increment write request packet, circuitry in the destination node stores the 32-bit value in the fetch-and-increment register. The circuitry then creates a fetch-and-increment write response packet. The circuitry in the destination node sends the fetch-and-increment write response packet to the PE that requested the fetch-and-increment write operation.

After the support circuitry in the PE that requested the fetch-and-increment write receives the fetch-and-increment write response packet, the support circuitry decrements the outstanding write request counter. This action completes the fetch-and-increment write operation.

The value stored to a fetch-and-increment register can be checked by following the fetch-and-increment write operation immediately with a fetch-and-increment read operation. The value returned should be the same as the value that was written. The value stored in the fetch-and-increment register following the read will be one greater than the value that was written.

Messaging Facility

The messaging facility transfers a special packet (packet type 7), called a message, from one PE to another PE. After receiving a message, the support circuitry in a PE interrupts the microprocessor and places the message in a message queue. The microprocessor may then read the message from the message queue.

Writing a Message

The microprocessor initiates a message write operation using store instructions. In order to provide data for a complete message, the microprocessor must fill a write buffer line with data.

When the write buffer transfers the data and address information for the message to the support circuitry, the microprocessor provides the support circuitry with a WRITE_BLOCK cycle request code and a partial physical address. After receiving the WRITE_BLOCK cycle request and partial physical address, the support circuitry retrieves a function code and PE number from an entry in the DTB annex. When the function code is set to 7, the support circuitry performs a message write operation.

During a message write operation, bit 28 of the partial physical address must be set to 0. Also, bits 0 through 26 of the address offset are not used as address information (refer to Figure 7-6). Instead, bits 5 through 12 of the address offset may contain a software defined code. (When sending a message to a PE in an I/O gateway, bits 24 and 26 of the address offset must be set to 0 to prevent the message from being stored in the HISP buffer of the destination PE.)

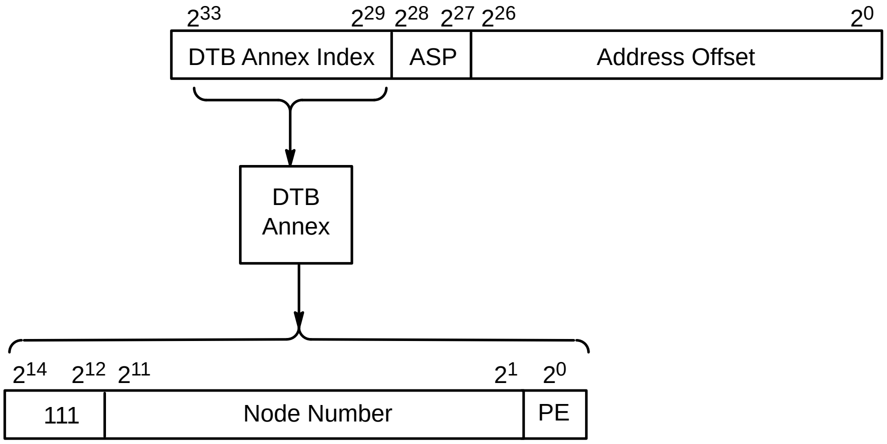


Figure 7-6. Message Write Address Information

After receiving the message information, the support circuitry creates a message packet and sends the packet to the destination PE (even when this is the local PE). Figure 7-7 shows the format of the message packet.

As previously stated, in order to provide data for a complete message, the microprocessor must fill a write buffer line with data. When the microprocessor receives an interrupt while filling a write buffer line, the microprocessor stops storing data into the write buffer, and the write buffer sends all valid data to the support circuitry. Because of this characteristic, data from a write buffer line that contained part of a message would be sent to the support circuitry. Messages that were broken up due to the write buffer emptying prematurely also use a packet with a 4-word body; however, not all of the words in the body of the message packet contain valid data. A mask in the command phit of the message packet indicates which words are valid.

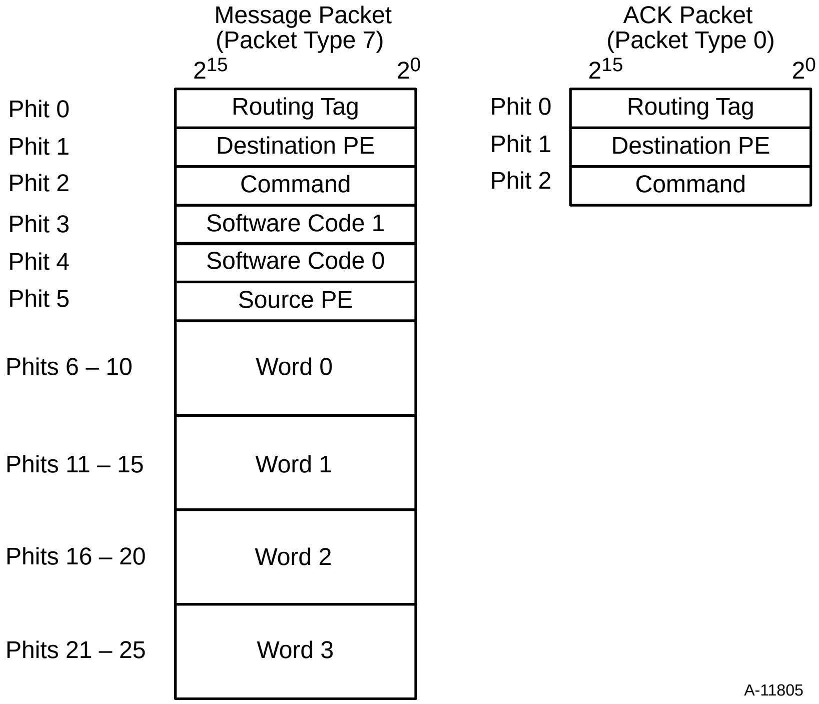


Figure 7-7. Messaging Facility Packet Formats

Table 7-4 shows the format of the command field in the command phit for the message packet. Bits 0 through 7 of the command field contain a mask (m) that indicates which 32-bit halfwords in the body of the message packet contain valid data.

Table 7-4. Command Fields for Message, ACK, and NACK Packets

Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Packet Type
1	1	1	1	m	m	m	m	m	m	m	m	Message
0	0	0	0	1	0	0	x	x	x	x	x	ACK
1	1	1	0	m	m	m	m	m	m	m	m	NACK

NOTE: x = Don’t Care

After receiving the message packet, the support circuitry in the destination PE attempts to store the message in the message queue. When the message queue can accept the message, the support circuitry stores the message in the queue and sets the message hardware interrupt for the microprocessor.

The support circuitry in the destination PE then creates a message acknowledge (ACK) packet and sends the packet to the PE that created the message. Figure 7-7 also shows the format of an ACK packet, and Table 7-4 shows the format of the command field in an ACK packet.

When the message queue in the destination PE cannot accept the message, the support circuitry returns the message to the requesting PE by creating a no-acknowledge (NACK) packet. A NACK packet contains the same information as the message packet; however, the NACK packet has the destination node and source phits exchanged and bit 8 of the command field set to 0 (refer to Figure 7-8 and again to Table 7-4).

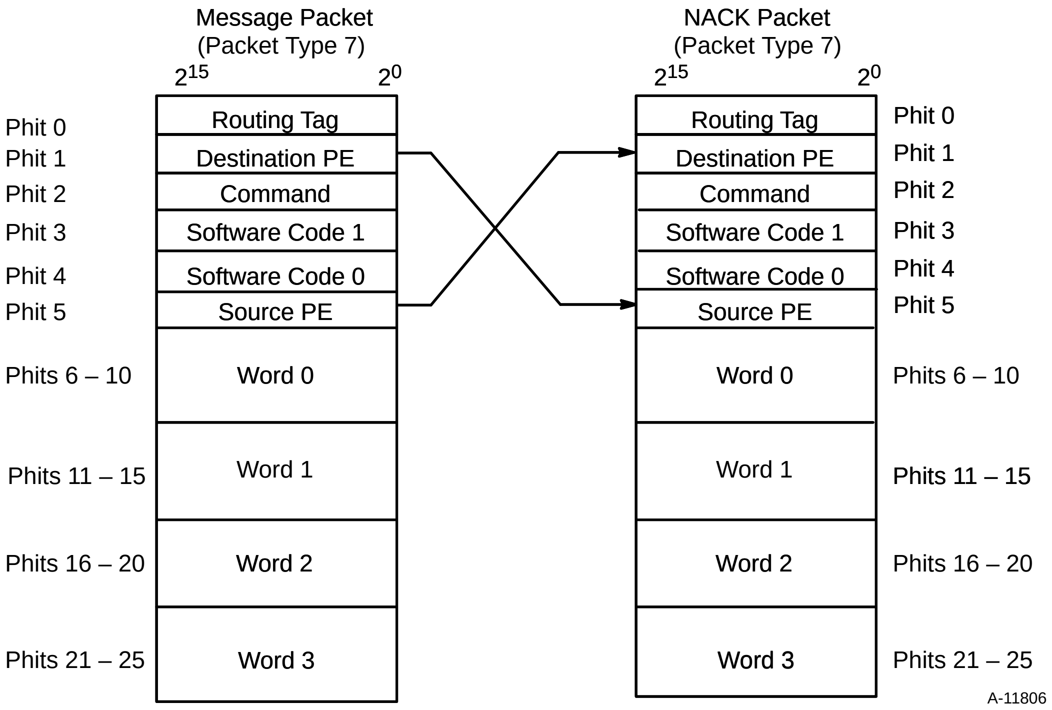


Figure 7-8. NACK Packet Format

After receiving a NACK packet, the support circuitry in the PE that created the message stores the NACK in its own message queue. The message queue always contains room for NACK packets and never rejects a NACK packet.

The microprocessor that sent the message can then read the NACK from the message queue. Because the NACK contains all of the information that the original message contained, the microprocessor can reconstruct and resend the original message. This process may repeat until the message is accepted and the support circuitry receives an ACK packet (instead of a NACK), which completes the message write. Because of this feature, message delivery is guaranteed regardless of the amount of system message traffic.

Message Queue

After receiving a message packet, the support circuitry in a PE places the message in a designated location in local memory called the message queue. The message queue stores up to 4,080 message, NACK, or error packets and contains 16 reserved locations for a small amount of overflow (total of 256K bytes of information). The support circuitry places message packets in the message queue in the order they are received. The base address offset of the message queue is $40000_{16,}$ and the limit of the message queue is $7FFFF_{16}$.

Figure 7-9 shows the format of a message packet in the message queue. The body of the packet is stored in memory locations n through n + 3. Several copies of the header phits of the packet are stored in memory locations n + 4 through n + 7.

Message Packet

Memory Address	2^{63} 2^{48}	2^{47} 2^{32}	2^{31} 2^{16}	2^{15} 2^{0}
n + 7	Software Code 0	Software Code 1	Command	Destination PE
n + 6	Source PE	Software Code 0	Software Code 1	Command
n + 5	Software Code 0	Software Code 1	Command	Destination PE
n + 4	Source PE	Software Code 0	Software Code 1	Command
n + 3	Word 3			
n + 2	Word 2			
n + 1	Word 1			
n + 0	Word 0			

A-11807

Figure 7-9. Format of a Message Packet in the Message Queue

Although the parity bits in the header phits of the message packet enable the network interface to check when an error occurred during a transfer, the microprocessor does not use the parity bits to check when an error occurred while reading the header phits from the message queue. To enable the microprocessor to detect errors when reading header phits from the message queue, the support circuitry places several copies of the header phits in the message queue.

To check for errors in the header phits, the message handling software compares the word of header phit information stored at location n+4 with the word of header phit information stored at location n+6. Likewise, the message handling software compares the word of header phit information stored at location n+5 with the word of header phit information stored at location n+7 (refer to Figure 7-9).

When the word in location n+4 matches the word in location n+6 and the word in location n+5 matches the word in location n+7, the header phits of the packet were not corrupted when transferring from the message queue to the microprocessor. When the words do not match, one or more of the header phit bits changed value while being transferred from the message queue to the microprocessor.

When the message handling software detects an error, the software must determine which header phit contains the error. To do this, the software examines each header phit in the word read from the message queue and generates a new set of parity bits for the header phit. When the new parity bits match the parity bits read from the message queue, the header phit is not corrupted. When the new parity bits do not match the parity bits read from the message queue, the header phit is corrupted. For more information on the bit format of each header phit, refer again to “Creating a Packet” in Section 6, “Addressing.”

When the microprocessor reads the header phit words from the message queue, the microprocessor signals the support circuitry that it does not want to perform SECDED on the words. The microprocessor does this by setting bit 27 of the partial physical address to 1. The support circuitry then sends the header phit words to the microprocessor and signals the microprocessor that SECDED should not be performed on the header information.

When the microprocessor reads the body of the message from the message queue, the microprocessor signals the support circuitry that it wants to perform SECDED on the words. The microprocessor does this by setting bit 27 of the partial physical address to 0. The support circuitry then sends the body of the message to the microprocessor and signals the microprocessor that SECDED should be performed on the data. More information on how the microprocessor performs SECDED is provided in Section 3, “Processing Element Node.”

Control

The message queue is actually a first-in, first-out (FIFO) circular buffer. The support circuitry signals the message queue to perform functions when a message is written, when a message is accepted, when a NACK is received, and when an ACK is received.

When a message is written, the support circuitry reserves a location in the message queue. By reserving a location in the queue, the support circuitry guarantees that there will be room in the message queue in the event of receiving a NACK.

When a message is received, the support circuitry stores the message in a location of the message queue. Likewise, when a NACK is received, the support circuitry stores the NACK in a location of the message queue. This location was reserved when the original message was written.

When an ACK is received, the support circuitry releases the location in the message queue reserved for a NACK. The reserved location is no longer needed after receiving an ACK.

The support circuitry contains a hardware counter and two memory-mapped registers used to control the message queue. The hardware counter, called the limit counter, is a 13-bit counter that indicates how many message slots in the message queue do not contain information or are not reserved for possible NACK packets. The limit counter cannot be read but can be reset to 4079 when software writes any value to the message queue tail pointer (MQ_TP) memory-mapped register.

The MQ_TP register is actually a 12-bit hardware counter that contains the tail pointer. The tail pointer is a value that represents the slot in the message queue where the next message that arrives will be stored. The actual memory offset represented by the tail pointer is calculated by multiplying the tail pointer by 64 and adding the result to the message queue base address. The value of the MQ_TP may be read and may be reset to 0. To reset the MQ_TP register to 0, software must write any value to the MQ_TP register.

The message queue limit increment register (MQ_LIR) is a memory-mapped register that software uses to manipulate the limit counter. When software writes any value to the MQ_LIR, the value of the limit counter increments by one. When software reads the value of the MQ_LIR, bit 12 of the register contains the sign bit of the limit counter and the remaining bits are not defined.

In addition to these hardware registers and counters, the messaging facility software must maintain a 12-bit head pointer. The head pointer indicates the slot location of the next message in the message queue that will be read by the microprocessor.

The support circuitry also controls the message and error hardware interrupts. The message interrupt indicates to the microprocessor that one or more message, NACK, or error packets have been stored in the message queue. The error interrupt indicates to the microprocessor that one or more errors have occurred. More information on the message and error hardware interrupts is provided in the following subsections and in "Hardware Interrupt Pins" in Section 10, "Control and Status."

The following sections describe how the registers, counters, and signals are used to control the messaging facility.

Writing a Message

When the microprocessor signals the support circuitry that the microprocessor is writing a message, the support circuitry checks the value of the message queue limit counter. When the limit counter is greater than or equal to zero, the support circuitry decrements the limit counter by one. The support circuitry decrements the value of the limit counter to reserve a location in the message queue in the event of receiving a NACK response packet.

Figure 7-10 shows a sample message queue before and after sending a message. After decrementing the limit counter, the support circuitry sends a message packet to the destination PE.

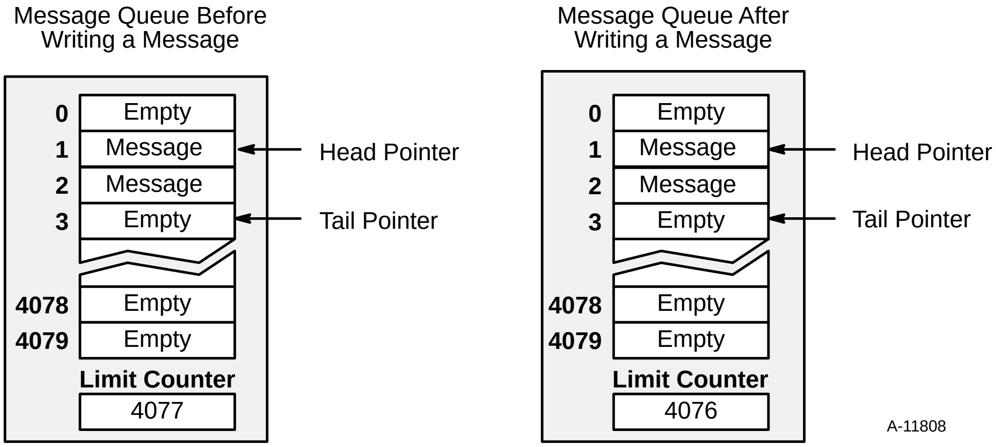


Figure 7-10. Writing a Message

When the limit counter is less than zero, the support circuitry indicates that a message queue oversubscribed condition exists. To indicate that a message queue oversubscribed condition exists, the support circuitry sets the message queue oversubscribed bit (bit 2) in the system status register (SSR) to 1 and sets the Error hardware interrupt to the microprocessor.

Although the message queue oversubscribed condition exists, the support circuitry still decrements the limit counter by one and sends a message packet to the destination PE. When a message queue oversubscribed condition occurs, the operating system should immediately dispose of unread messages stored in the message queue to provide more space in the message queue.

Receiving a Message

When the support circuitry in a destination PE receives a message packet, the support circuitry first checks the value of the limit counter in the message queue. When the value of the limit counter is greater than zero, the support circuitry accepts the message. When the value of the limit counter is less than or equal to zero, the message queue is full and the support circuitry rejects the message.

When the support circuitry accepts a message, it stores the message in the message queue at the location pointed to by the tail pointer. After storing the message in the queue, the support circuitry increments the tail pointer, decrements the limit counter, and sets the message hardware interrupt to the microprocessor. Figure 7-11 shows a sample message queue before and after receiving a message.

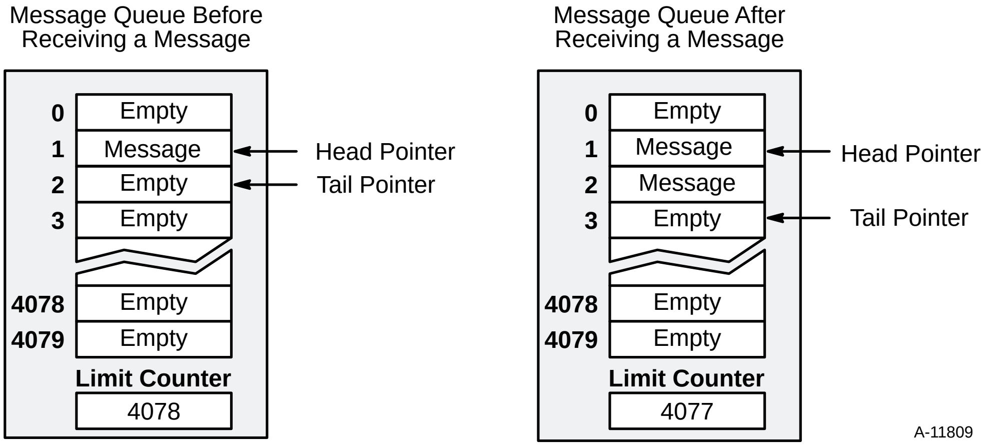


Figure 7-11. Receiving a Message

The tail pointer only increments. Because of this characteristic, after the tail pointer reaches the last entry in the message queue, the tail pointer automatically wraps to the first entry of the message queue when the support circuitry accepts another message. Figure 7-12 shows a sample message queue before and after the tail pointer wraps to the first entry of the message queue. When the head pointer reaches the last entry in the message queue, software must reset the value of the head pointer to the beginning of the message queue.

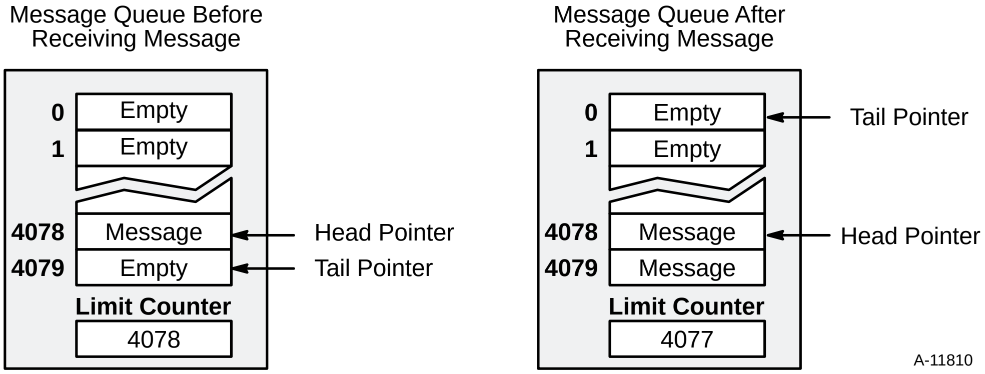


Figure 7-12. Wrapping Around

After accepting the message, the support circuitry creates an ACK packet. The support circuitry sends the message ACK packet to the PE that sent the original message.

After receiving the ACK packet, the support circuitry in the PE that sent the message increments the value of the limit counter to deallocate the slot on the message queue reserved for a possible NACK. For example, Figure 7-13 shows a sample message queue before and after receiving an ACK packet.

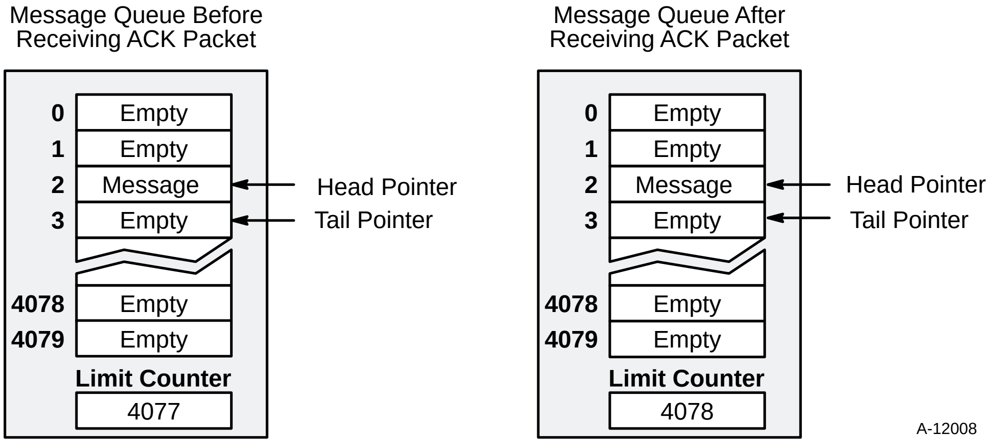


Figure 7-13. Receiving an Acknowledgement

By incrementing the value of the limit counter, the support circuitry frees up space in the message queue for sending or receiving another message. This action completes a successful message transfer from one PE to another.

As previously stated, when the support circuitry receives a message packet, the support circuitry first checks the value of the limit counter in the message queue. When the value of the limit counter is less than or equal to zero, the support circuitry rejects the message.

When the support circuitry rejects a message, it converts the message into a NACK packet. After converting the packet, the support circuitry sends the NACK packet to the PE that sent the original message.

After receiving the NACK packet, the support circuitry in the PE that sent the message stores the NACK packet in the message queue at the location pointed to by the tail pointer. After storing the NACK packet in the queue, the support circuitry increments the tail pointer and sets the message hardware interrupt to the microprocessor. Figure 7-14 shows a sample message queue before and after receiving a NACK packet.

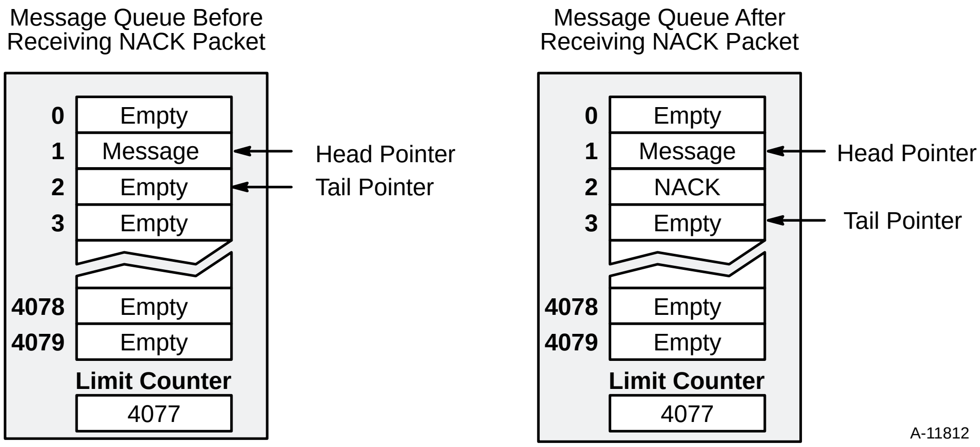


Figure 7-14. Receiving a NACK Packet

When the microprocessor sent the original message, the support circuitry decremented the value of the limit counter. Because the limit counter was decremented at that time, the support circuitry does not decrement the limit counter when the support circuitry receives a NACK packet.

After the support circuitry sets the message interrupt to the microprocessor, the microprocessor reads the NACK information from the message queue. The microprocessor then sends the message again to the destination PE.

Reading the Message

After the support circuitry stores a message packet, NACK packet, or error message in the message queue, the support circuitry sets the message hardware interrupt to the microprocessor. This signals the microprocessor that one or more message facility packets have arrived and are stored in the message queue.

The microprocessor may then read the message, NACK, or error message packet from the message queue and interpret the software code, encoded commands, or addresses to decipher the action requested by the message. After reading a packet from the message queue, the microprocessor increments the software-controlled head pointer and increments the limit counter by writing any value to the MQ_LIR register. When the microprocessor increments the limit counter, the support circuitry clears the message hardware interrupt. Figure 7-15 shows a sample message queue before and after the microprocessor reads a message or NACK packet from the queue.

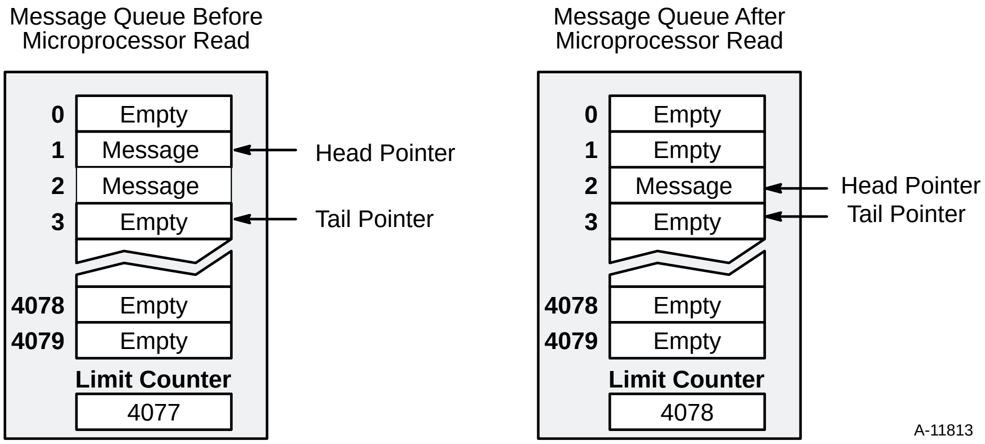


Figure 7-15. Microprocessor Read

Normally, after the microprocessor receives the message hardware interrupt, the microprocessor continues to read messages from the message queue until the value of the head pointer equals the value of the tail pointer. This indicates to the microprocessor that there are no more messages or NACK packets in the message queue.

To avoid missing a message arrival, the microprocessor should increment the limit counter (to clear the message hardware interrupt) before comparing the value of the tail pointer and the head pointer. The limit counter is incremented by writing any value to the MQ_LIR register.

Error Messages

In addition to message packets and NACK packets, the support circuitry may receive error “messages” and store the error messages in the message queue. The network interface generates error messages when a network packet error occurs.

There are two types of network packet errors: a misrouted packet error and a packet parity error. When the network interface receives a packet and the value of the destination phit does not match the value stored in the network interface source (X_WHOAMI) register, a misrouted packet error occurs. When the network interface detects a parity error in the header phit of a packet, a packet parity error occurs.

When either network packet error occurs, the network interface turns the packet it received into an error message by setting bit 15 of the command phit to 1. (For more information on the command phit of a packet, refer again to “Creating a Packet” in Section 6, “Addressing.”) The network interface then sends the error message to the appropriate PE in the node.

After receiving an error message, the support circuitry performs the following actions:

- Sets the network packet error bit (bit 1) of the SSR to 1
- Sets the error hardware interrupt to the microprocessor
- Stores the error message in the message queue
- Sets the message hardware interrupt to the microprocessor.
- Does not create a message ACK packet even when the original packet was a message

After receiving the first error message, the support circuitry discards all subsequent error messages until the SSR register is read to clear the existing error conditions.

The network interface may create an error message out of any of the eight packet types. Figure 7-16, Figure 7-17, and Figure 7-18 show the format of error messages for packet types 0 through 6 as they are stored in the message queue. The format of error messages for packet type 7 (message and NACK packets) is the same as shown in Figure 7-9; however, the error bit (bit 15) is not set in the command phit of words n+5 and n+7.

Packet Type 0

Memory Address	2^{63} – 2^{48}	2^{47} – 2^{32}	2^{31} – 2^{16}	2^{15} – 2^{0}
n + 7	Not Valid Data	Not Valid Data	Command	Destination PE
n + 6	Not Valid Data	Not Valid Data	Not Valid Data	Command
n + 5	Not Valid Data	Not Valid Data	Command	Destination PE
n + 4	Not Valid Data	Not Valid Data	Not Valid Data	Command
n + 3	Word 3 (Does not contain valid data)			
n + 2	Word 2 (Does not contain valid data)			
n + 1	Word 1 (Does not contain valid data)			
n + 0	Word 0 (Does not contain valid data)			

Packet Type 1

Memory Address	2^{63} – 2^{48}	2^{47} – 2^{32}	2^{31} – 2^{16}	2^{15} – 2^{0}
n + 7	Request Address 0	Request Address 1	Command	Destination PE
n + 6	Source PE	Request Address 0	Request Address 1	Command
n + 5	Request Address 0	Request Address 1	Command	Destination PE
n + 4	Source PE	Request Address 0	Request Address 1	Command
n + 3	Word 3 (Does not contain valid data)			
n + 2	Word 2 (Does not contain valid data)			
n + 1	Word 1 (Does not contain valid data)			
n + 0	Word 0 (Does not contain valid data)			

Packet Type 2

Memory Address	2^{63} – 2^{48}	2^{47} – 2^{32}	2^{31} – 2^{16}	2^{15} – 2^{0}
n + 7	Request Address 0	Request Address 1	Command	Destination PE
n + 6	Source PE	Request Address 0	Request Address 1	Command
n + 5	Request Address 0	Request Address 1	Command	Destination PE
n + 4	Source PE	Request Address 0	Request Address 1	Command
n + 3	Not Valid Data	Not Valid Data	Response Address 0	Response Address 1
n + 2	Not Valid Data	Not Valid Data	Response Address 0	Response Address 1
n + 1	Not Valid Data	Not Valid Data	Response Address 0	Response Address 1
n + 0	Not Valid Data	Not Valid Data	Response Address 0	Response Address 1

The error bit (bit 15) is not set in these command phits.

A-11814

Figure 7-16. Error Messages for Packet Types 0 through 2 in the Message Queue

Packet Type 3

		2^{63} 2^{48}	2^{47} 2^{32}	2^{31} 2^{16}	2^{15} 2^{0}
	n + 7	Not Valid Data	Not Valid Data	Command	Destination PE
	n + 6	Not Valid Data	Not Valid Data	Not Valid Data	Command
Memory Address	n + 5	Not Valid Data	Not Valid Data	Command	Destination PE
	n + 4	Not Valid Data	Not Valid Data	Not Valid Data	Command
	n + 3	Word 0			
	n + 2	Word 0			
	n + 1	Word 0			
	n + 0	Word 0			

Packet Type 4

		2^{63} 2^{48}	2^{47} 2^{32}	2^{31} 2^{16}	2^{15} 2^{0}
	n + 7	Req/Res Address 0	Req/Res Address 1	Command	Destination PE
	n + 6	Source PE	Req/Res Address 0	Req/Res Address 1	Command
Memory Address	n + 5	Req/Res Address 0	Req/Res Address 1	Command	Destination PE
	n + 4	Source PE	Req/Res Address 0	Req/Res Address 1	Command
	n + 3	Word 0			
	n + 2	Word 0			
	n + 1	Word 0			
	n + 0	Word 0			

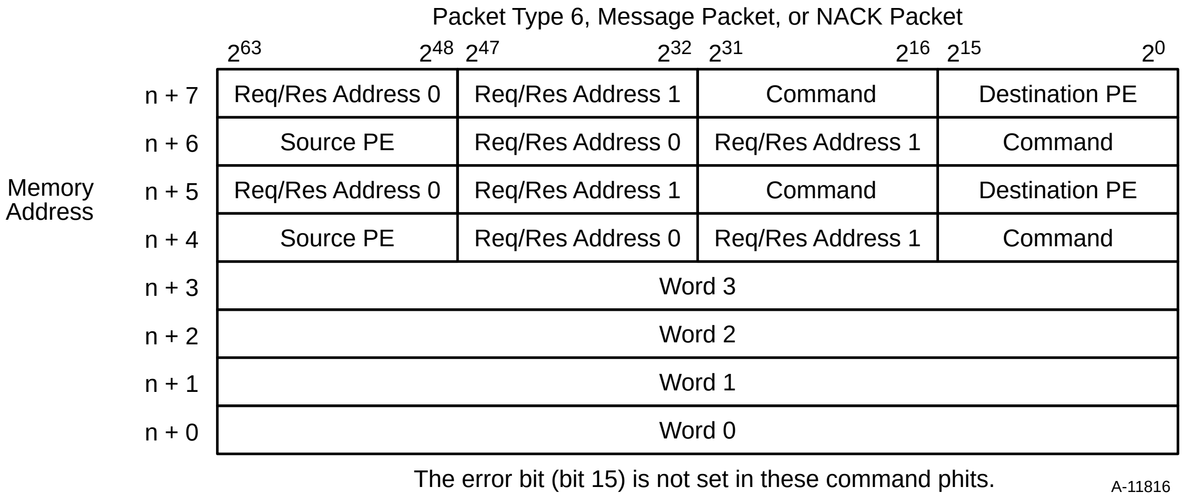
Packet Type 5

		2^{63} 2^{48}	2^{47} 2^{32}	2^{31} 2^{16}	2^{15} 2^{0}
	n + 7	Not Valid Data	Not Valid Data	Command	Destination PE
	n + 6	Not Valid Data	Not Valid Data	Not Valid Data	Command
Memory Address	n + 5	Not Valid Data	Not Valid Data	Command	Destination PE
	n + 4	Not Valid Data	Not Valid Data	Not Valid Data	Command
	n + 3	Word 3			
	n + 2	Word 2			
	n + 1	Word 1			
	n + 0	Word 0			

The error bit (bit 15) is not set in these command phits.

A-11815

Figure 7-17. Error Message for Packet Types 3 through 5 in the Message Queue



Packet Type 6, Message Packet, or NACK Packet

Memory Address	2^{63} – 2^{48}	2^{47} – 2^{32}	2^{31} – 2^{16}	2^{15} – 2^{0}
n + 7	Req/Res Address 0	Req/Res Address 1	Command	Destination PE
n + 6	Source PE	Req/Res Address 0	Req/Res Address 1	Command
n + 5	Req/Res Address 0	Req/Res Address 1	Command	Destination PE
n + 4	Source PE	Req/Res Address 0	Req/Res Address 1	Command
n + 3	Word 3			
n + 2	Word 2			
n + 1	Word 1			
n + 0	Word 0			

The error bit (bit 15) is not set in these command phits.

Figure 7-18. Error Messages for Packet Type 6 in the Message Queue

To determine what type of request or response packet the error message was created from, software must read the value of the command field in the command phit. Table 7-5 lists the command fields in each type of packet. For more information on the command phit, refer again to Section 6, “Addressing.”

In Table 7-5, bits 9 through 11 of the command field indicate the type of packet in which the command phit is located. (For more information on the packet types, refer again to Section 6, “Addressing.”) When set to 1, bit 8 of the command phit indicates the packet is a request. When set to 0, bit 8 of the command phit indicates the packet is a response.

Bits 0 through 7 of the command phit contain an opcode that signals the support circuitry what type of operation to perform. In some cases, the opcode contains mask bits (m) that indicate which 32-bit halfwords in the body of the packet contain valid data. When the packet is a data prefetch operation packet, the command field contains 4 bits of address (a) that indicate which entry of the data prefetch queue will receive the word of data from the body of the packet.

When reading the body of an error message, bit 27 of the partial physical address should be set to 1. This indicates that the microprocessor does not want to perform SECDED on the body of the error message.

Table 7-5. Packet Command Fields

Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Packet Name
0	0	0	0	0	0	1	x	x	x	x	x	PE write response (includes fetch-and-increment writes)
0	0	0	0	0	1	0	x	x	x	x	x	BLT write response
0	0	0	0	1	0	0	x	x	x	x	x	ACK
0	0	1	1	0	0	0	x	x	x	x	x	PE single-word read request
0	0	1	1	0	1	0	x	x	x	x	x	Fetch-and-increment read request
0	0	1	1	0	1	1	x	a	a	a	a	Prefetch fetch-and-increment read request
0	0	1	1	1	0	0	x	x	x	x	x	PE block read request
0	0	1	1	1	1	1	x	a	a	a	a	Prefetch read request
0	1	0	1	0	0	0	x	x	x	x	x	BLT single-word read request
0	1	0	1	1	0	0	x	x	x	x	x	BLT block read request
0	1	0	1	1	0	1	x	x	x	x	x	BLT I/O block read request
0	1	1	0	0	0	0	x	x	x	x	x	PE single-word read response
0	1	1	0	0	1	0	x	x	x	x	x	Fetch-and-increment read response
0	1	1	0	0	1	1	x	a	a	a	a	Prefetch fetch-and-increment read response
0	1	1	0	1	0	0	x	x	x	x	x	Atomic swap response
0	1	1	0	1	0	1	x	a	a	a	a	Prefetch atomic swap response
0	1	1	0	1	1	1	x	a	a	a	a	Prefetch read response
1	0	0	0	0	0	0	x	x	x	x	x	BLT single-word read response
1	0	0	1	0	0	1	x	x	x	m	m	PE write request (1-word body)
1	0	0	1	0	1	0	x	x	x	x	x	BLT single-word write request
1	0	0	1	1	0	0	x	x	x	x	x	Atomic swap request
1	0	0	1	1	0	1	x	a	a	a	a	Prefetch atomic swap request
1	0	0	1	1	1	0	x	x	x	x	x	Cached atomic swap request
1	0	0	1	1	1	1	x	x	x	x	x	Fetch-and-increment write request
1	0	1	0	1	0	0	x	x	x	x	x	PE block read response
1	0	1	0	1	1	0	x	x	x	x	x	Cached atomic swap response
1	1	0	0	1	0	0	x	x	x	x	x	BLT block read response
1	1	0	0	1	0	1	x	x	x	x	x	BLT I/O block read response
1	1	0	1	0	0	0	0	0	0	0	0	BLT block write request
1	1	0	1	m	m	m	m	m	m	m	m	PE write request (4-word body)

Table 7-5. Packet Command Fields (continued)

Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Packet Name
1	1	1	0	m	m	m	m	m	m	m	m	NACK
1	1	1	1	m	m	m	m	m	m	m	m	Message

Message Queue Fully Subscribed Conditions

Because of the way the write buffer works in the microprocessor, the microprocessor may be notified of a message queue oversubscribed condition when there are still up to four messages in the write buffer. When the microprocessor receives the error interrupt that indicates the oversubscribed condition, the microprocessor empties the write buffer.

As the support circuitry receives each of the four message writes from the microprocessor, the support circuitry decrements the message queue limit counter in preparation for possible NACK packets and sends the messages to the destination PEs. In this case, the value of the limit counter continues to go negative with each message sent. The negative value indicates that the message queue is fully subscribed and any NACK packet received will be stored in one of the 16 locations reserved for overflow packets.

In addition to possible NACK packets from messages that were stored in the write buffer, the support circuitry may receive an error message at any time. When the support circuitry receives an error message and the message queue is fully subscribed, the support circuitry still stores the error message in the message queue and decrements the limit counter by one. Because the limit counter is already negative, this also uses one of the 16 slots in the message queue reserved for overflow packets.

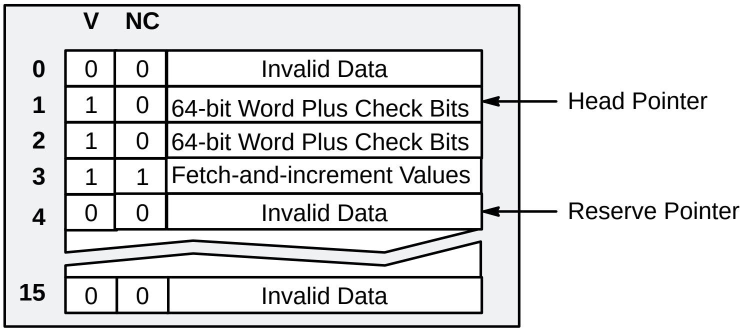
Another condition, which is highly unlikely, occurs when a PE issues 4,080 message writes and has not yet received any ACK packets. In this case the message queue is fully subscribed, but empty. When this occurs, software must periodically read the value of the MQ_LIR register and examine bit 12 of the register. When this bit is set to 1, the value of the limit counter is negative and no more message writes should be issued. When this bit is set to 0, the value of the limit counter is positive and message writes may be issued.

Data Prefetch

When the microprocessor issues a noncacheable read, atomic swap, or fetch-and-increment register read operation, the microprocessor stalls until the support circuitry retrieves a word of data from system memory and sends the word to the microprocessor. One way to hide the latency that this stall represents is to use the data prefetch queue in the support circuitry.

Data Prefetch Queue

The data prefetch queue is a circular queue that can store up to sixteen 64-bit words. Each entry in the prefetch queue contains a valid bit (V) and a no-check bit (NC); each entry may also contain either a 64-bit word of data plus check bits or the two 32-bit values read from a fetch-and-increment register (refer to Figure 7-19).



	V	NC	
0	0	0	Invalid Data
1	1	0	64-bit Word Plus Check Bits
2	1	0	64-bit Word Plus Check Bits
3	1	1	Fetch-and-increment Values
4	0	0	Invalid Data
15	0	0	Invalid Data

Figure 7-19. Data Prefetch Queue

When set to 1, the valid bit indicates that the entry of the prefetch queue contains valid data. When set to 0, the valid bit indicates that the entry of the prefetch queue does not contain valid data.

When set to 1, the no-check bit indicates the microprocessor should not perform SECDED on the data read from that entry of the prefetch queue. When set to 0, the no-check bit indicates the microprocessor should perform SECDED on the data read from that entry of the prefetch queue.

The 64-bit word in an entry of the prefetch queue may be a word read from memory or two values read from the fetch-and-increment register. When the word is read from memory, the no-check bit is set to 0. When the word is read from the fetch-and-increment register, the no-check bit is set to 1.

The support circuitry uses two pointers to control the prefetch queue: the reserve pointer and the head pointer. The reserve pointer indicates the next entry in the prefetch queue that will be reserved for a word of data. The head pointer indicates the entry in the prefetch queue that will be delivered to the microprocessor.

When the microprocessor requests that data be stored in the prefetch queue, the support circuitry first reads the number of the entry indicated by the reserve pointer. The support circuitry uses this queue entry number in the command phit of a prefetch request packet. The support circuitry then increments the reserve pointer.

For example, in Figure 7-20, the support circuitry read the entry number of 3, used this number in the command phit of a prefetch request packet, and then incremented the reserve pointer. The destination PE, after receiving the prefetch request packet, creates a prefetch response packet and sends the response packet to the PE that requested the prefetch operation. The command phit of the response packet contains the prefetch queue entry number.

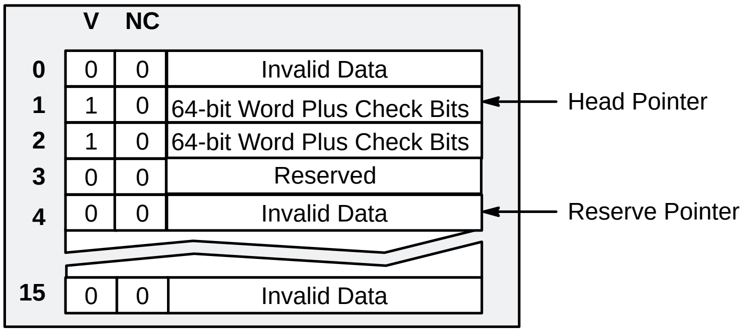
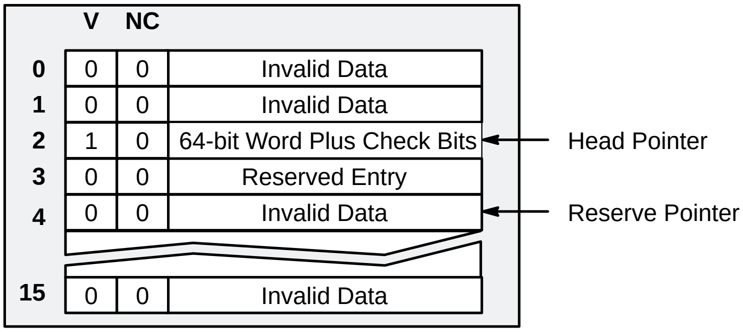


Figure 7-20. Reserving an Entry in the Prefetch Queue

After receiving a prefetch response packet, the support circuitry stores the 64-bit word from the body of the response packet in the location indicated by the queue entry number in the command phit. This method of storing words in the prefetch queue ensures that the words are stored in the queue in the same order that the microprocessor requested them, even when the support circuitry receives response packets out of order.

The microprocessor reads data from the prefetch queue one word at a time. After delivering a word from the head entry of the prefetch queue to the microprocessor, the support circuitry sets the valid bit for that entry to 0 in order to mark it as invalid. The support circuitry then increments the head pointer.

For example, in Figure 7-21, the support circuitry delivered the word of data stored in queue entry 1 to the microprocessor. After delivering the data, the support circuitry set the valid bit to 0 to mark entry 1 as invalid and then incremented the head pointer.



	V	NC	
0	0	0	Invalid Data
1	0	0	Invalid Data
2	1	0	64-bit Word Plus Check Bits
3	0	0	Reserved Entry
4	0	0	Invalid Data
15	0	0	Invalid Data

Figure 7-21. Reading a Word from the Prefetch Queue

Data Prefetch Requests

To store data in the prefetch queue, the microprocessor issues a `FETCH` or `FETCH_M` instruction instead of a load instruction. After issuing the instruction, hardware in the microprocessor stores information for the `FETCH` or `FETCH_M` instruction in the write buffer. When the information leaves the write buffer, the microprocessor presents a FETCH or FETCHM cycle request code and a partial physical address to the support circuitry.

After receiving this information from the microprocessor, the support circuitry retrieves a function code and PE number from the DTB annex. The value of the function code determines which one of three data prefetch requests the support circuitry performs.

When the support circuitry performs a data prefetch request, the support circuitry always interprets the PE number stored in the DTB annex as a remote PE. Because of this characteristic, the support circuitry always creates a prefetch request packet and sends the packet to the destination PE, even when the destination PE is the PE that requested the prefetch.

Prefetch Memory Read

When the function code is set to 0, 4, 6, or 7, the support circuitry performs a normal prefetch memory read request. During a prefetch memory read request, the support circuitry creates a prefetch read request packet and sends it to a remote PE. The remote PE reads a word from memory, creates a prefetch read response packet, and sends the response packet to the PE that requested the prefetch memory read.

After receiving a word of information from a prefetch read response packet, the support circuitry places the word in the appropriate entry of the data prefetch queue. The support circuitry then marks that entry in the prefetch queue as valid and sets the no-check bit for that entry to 0.

Prefetch Fetch-and-increment Register Read

When the function code is set to 2, the support circuitry performs a prefetch fetch-and-increment register read request. During this prefetch request, the support circuitry creates a prefetch fetch-and-increment request packet and sends the packet to the destination PE node. Circuitry in the destination PE node reads the value of the fetch-and-increment register and then increments the value of the register by one (the same action done during a fetch-and-increment register read operation). The circuitry then creates a prefetch fetch-and-increment response packet.

After receiving a 64-bit word of information (32-bit fetch-and-increment value plus a copy of that value) from a prefetch fetch-and-increment response packet, the support circuitry in the PE that requested the prefetch places the word in the appropriate entry of the data prefetch queue. The support circuitry then marks that entry in the prefetch queue as valid and sets the no-check bit for that entry to 1.

Prefetch Atomic Swap

When the function code is set to 1 or 5, the support circuitry performs a prefetch atomic swap request. During this prefetch request, the support circuitry creates a prefetch atomic swap request packet that contains the value of the PE’s swaperand register and sends the request packet to a remote PE. The support circuitry in the remote PE reads a word from memory and then writes the swaperand word from the prefetch atomic swap request packet into the same memory location. The support circuitry in the remote PE then creates a prefetch atomic swap response packet and sends the packet to the PE that requested the prefetch atomic swap.

After receiving a word of information from a prefetch atomic swap response packet, the support circuitry in the PE that requested the prefetch places the word of data in the appropriate entry in the data prefetch queue. The support circuitry then marks that entry of the prefetch queue as valid and sets the no-check bit for that entry to 0.

Reading Data from the Prefetch Queue

To read a word of data from the prefetch queue, the microprocessor reads a memory-mapped register called the prefetch queue head (PRE_FETCH) register. After receiving the read PRE_FETCH register information from the microprocessor, the support circuitry delivers the word at the head of the prefetch queue to the microprocessor. While doing this, the support circuitry signals the microprocessor not to perform SECDED when the no-check bit for the entry is set to 1. The support circuitry then marks that entry of the prefetch queue as invalid and increments the head pointer.

To read the next word in the prefetch queue, the microprocessor again reads the PRE_FETCH register. After receiving the read PRE_FETCH information, the support circuitry delivers the second word (which is now the head of the prefetch queue) to the microprocessor. The support circuitry then marks that entry of the prefetch queue as invalid and increments the head pointer.

The microprocessor may read the PRE_FETCH register before the support circuitry receives the prefetch response packet that corresponds to the entry pointed to by the head pointer. In this case, the head pointer points to an entry that is reserved and does not contain valid data. When this occurs, the microprocessor stalls until the support circuitry receives the associated prefetch response packet and stores the requested word in the prefetch queue.

Error Conditions

Two error conditions may be encountered when using the prefetch queue: a prefetch queue overrun error and a prefetch queue underrun error. A prefetch queue overrun error occurs when the microprocessor has issued 16 prior prefetch requests without reading data from the prefetch queue and then issues another prefetch request. When this occurs, the support circuitry sets the prefetch queue overrun bit (bit 3) of the system status register (SSR) to 1 and sets the error interrupt. More information on the SSR and error interrupt is provided in Section 10, “Control and Status.”

A prefetch queue underrun error occurs when none of the entries of the prefetch queue contain valid data, or when none are reserved, and the microprocessor tries to read a word of data from the prefetch queue before issuing a prefetch request. When this occurs, the support circuitry sets the prefetch queue underrun bit (bit 4) of the SSR to 1 and sets the error interrupt.

NOTE: Because of the way the write buffer functions in the microprocessor, the microprocessor cycle request associated with a `FETCH` or `FETCH_M` instruction may leave the microprocessor after a subsequently issued read of the PRE_FETCH register leaves. This action can result in a prefetch queue underrun error. Two methods may be used to avoid this error.

One method is to issue an `MB` instruction after the FETCH or FETCH_M instruction and before issuing the associated PRE_FETCH register read instruction. This forces the FETCH or FETCH_M request out of the write buffer and to the support circuitry ahead of the PRE_FETCH register read request.

Another method is to issue more prefetch (FETCH or FETCH_M) requests than the maximum number that can be stored in the write buffer (four FETCH or FETCH_M instructions). For example, issue at least five prefetch requests before reading the PRE_FETCH register. This guarantees that the first FETCH request has left the microprocessor. Then, if more prefetch requests must be issued, make sure at least five FETCH requests have been issued before issuing each PRE_FETCH read request.

FETCH Versus FETCH_M

Software may issue a data prefetch request using the FETCH or FETCH_M instruction. The support circuitry performs the same function when it receives a FETCH cycle request code as it performs when it receives a FETCHM cycle request code, except when performing a data prefetch operation from local memory. (Prefetch fetch-and-increment operations and remote data prefetch operations function the same with FETCH or FETCHM.)

When the support circuitry receives a FETCH cycle request, the support circuitry prevents subsequent local memory data operations from completing until the data prefetch response packet has been generated. When the support circuitry receives a FETCHM cycle request, the support circuitry does not prevent subsequent local memory operations from completing.

FETCH

When the support circuitry receives a FETCH cycle request, the support circuitry prevents additional local-memory read, local-memory write, or local-memory data prefetch operations from occurring until the support circuitry generates a prefetch response packet for the original local-memory data prefetch operation. This ensures the correct ordering of local-memory data prefetches, reads, and writes.

For example, after receiving a FETCH cycle request code and a function code of 0, the support circuitry creates a prefetch read request packet and sends it to the destination PE. When the destination PE is the local PE, the packet travels through the network interface and network router in the processing element node and then back to the support circuitry in the PE.

While the prefetch request packet travels through the network, the support circuitry does not allow local-memory read, write, or data prefetch operations that are requested by the microprocessor to complete. The support circuitry does allow local-memory instruction prefetch operations and all prefetch fetch-and-increment register read operations to complete. The support circuitry also allows all remote operations to complete.

After receiving the prefetch request packet from the network interface, the support circuitry reads a word of data from local memory and creates a prefetch read response packet. The support circuitry then sends the response packet to the source PE. As soon as the support circuitry creates the prefetch response packet, the support circuitry allows requests from the microprocessor for local-memory reads, writes, or data prefetches to complete.

Because the source PE is the local PE, the prefetch response packet also travels through the network interface and network router and back to the support circuitry. The support circuitry then stores the prefetch data in the designated entry of the prefetch queue.

FETCHM

When the support circuitry receives a FETCHM cycle request code from the microprocessor, the support circuitry does allow requests from the microprocessor for local-memory reads, writes, or data prefetches to complete. In this case, the support circuitry does not wait for a data prefetch response packet to be generated.

For example, after receiving a FETCHM cycle request code and a function code of 0, the support circuitry creates a prefetch read request packet and sends it to the destination PE. When the destination PE is the local PE, the packet travels through the network interface and network router in the processing element node and back to the support circuitry in the PE. While the request packet travels through the network, requests from the microprocessor for local-memory read, write, or data prefetch operations will be allowed to complete.

After receiving the prefetch request packet from the network interface, the support circuitry reads a word of data from local memory and creates a prefetch read response packet. The support circuitry sends the response packet to the source PE. Because the source PE is the local PE, the response packet also travels through the network interface and network router and back to the support circuitry. The support circuitry then stores the prefetch data in the designated entry of the prefetch queue.

Although the `FETCH_M` instruction may be used to reduce the latency of multiple local-memory data prefetch, read, and write operations, read and write ordering problems may occur when the `FETCH_M` instruction is used. For example, a read and write ordering problem may occur when the microprocessor requests a local-memory data prefetch read operation and then requests a local-memory write operation to the same local memory location.

When performing the local-memory data prefetch read operation, the support circuitry creates a prefetch read request packet and sends the packet through the network interface and network router. Because the support circuitry does not use the network during a local-memory write operation, the support circuitry may complete the local-memory write operation while the prefetch read request packet travels through the network.

After receiving the prefetch read request packet, the support circuitry reads the modified data from local memory and creates a prefetch read response packet. Because the local-memory write and local-memory data prefetch read operations completed out of order, the prefetched data does not reflect the contents of the local memory location when the microprocessor issued the local-memory data prefetch operation.

Register Mapping

The CRAY T3D system may use the fetch-and-increment register or memory-mapped registers when performing various operations. These registers are used to store a word of data prior to an atomic swap, store the fetch-and-increment value, control messaging, or read data from the data prefetch queue.

The following subsections describe the addressing and bit assignments for the registers used for operations in the CRAY T3D system. Each subsection also provides a brief summary of the function of each register.

Table 7-6 is a summary of the registers and their names. Table 7-6 also lists the partial physical address of each register as it appears on the address pins of the microprocessor.

Table 7-6. Memory Function Registers

Address	Register Name	Direction	Description
10300000_{16}	SWAP	Write	The swaperand register
DTB annex entry	Fetch-and-increment	Read and write	The fetch-and-increment registers
10460000_{16}	MQ_LIR	Read and Write	Message queue increment limit counter register
10464000_{16}	MQ_TP	Read and write	Message queue tail pointer register
10180000_{16}	PRE_FETCH	Read	Data prefetch queue head register

Because the virtual address is defined by software, the addresses for each of the registers are given according to the partial physical address as it appears on the address pins of the microprocessor.

Swaperand Register Address 10300000_{16}

The swaperand register (SWAP) is a 64-bit, write-only, general access register. The swaperand register contains 1 word of data that will be written to a memory location during an atomic swap or cached atomic swap operation.

Each PE contains a swaperand register. Figure 7-22 shows the bit assignments for the swaperand register address as they appear on the address pins of the microprocessor.

HEX	1	0				3				0				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	x	1	1	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x = Don’t Care

Figure 7-22. Swaperand Register Address Bit Format

Table 7-7 shows the bit format of the swaperand register and describes the bits in the register.

Table 7-7. Swaperand Register Format

Bits	Name	Description
63 – 0	Swaperand	These bits contain a 64-bit word of data that will be written into a memory location during an atomic swap or cached atomic swap operation.

Because the swaperand register is a write-only register, the microprocessor cannot directly read the contents. To read the contents of the swaperand register, the microprocessor must perform an atomic swap with a location in memory and then read the word of data from that memory location.

Fetch-and-increment Register

Each processing element node contains two 32-bit fetch-and-increment registers. A fetch-and-increment register (F&I) is a hardware register that is incremented each time information is read from the fetch-and-increment register. Although each processing element node contains two fetch-and-increment registers, the registers function independently of the PEs and may be referenced by any PE in a partition.

The microprocessor addresses a fetch-and-increment register by placing the function code of 2 and a 12-bit fetch-and-increment register address in a line of the DTB annex. The fetch-and-increment register address contains two parts: a node number and a fetch-and-increment (F&I) register bit (refer to Figure 7-23).

The node number is a logical or virtual node number that indicates the node in which the fetch-and-increment register is located. The F&I register bit indicates which fetch-and-increment register within the node will be referenced.

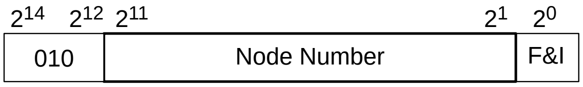


Figure 7-23. Addressing a Fetch-and-increment Register

Table 7-8 shows the bit format of a fetch-and-increment register.

Table 7-8. Fetch-and-increment Register Format

Bits	Name	Description
31 – 0	Fetch-and-increment value	These bits contain the value of the fetch-and-increment register.

Message Queue Tail Pointer Address 10464000_{16}

The message queue tail pointer (MQ_TP) register is a 12-bit, readable and writable, system privileged register. The MQ_TP is a hardware counter that represents the slot in the message queue where the next message that arrives will be stored. The actual memory offset represented by the tail pointer is calculated by multiplying the tail pointer by 64 and adding the result to the message queue base address (40000_{16}).

When the MQ_TP is read, it returns the current value of the MQ_TP. When any value is written to the MQ_TP, the support circuitry resets the value of the MQ_TP to 0 and resets the message queue limit counter to 4,079.

Figure 7-24 shows the bit assignments for the MQ_TP register address as they appear on the address pins of the microprocessor.

HEX	1	0				4				6				4				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	0	0	0	1	1	0	0	1	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x = Don’t Care

Figure 7-24. Message Queue Tail Pointer Register Address Bit Format

Table 7-9 shows the bit format of the MQ_TP register and describes the bits in the register.

Table 7-9. MQ_TP Register Format

Bits	Name	Description
11 – 0	Message queue tail pointer	These bits represent the slot in the message queue where the next message that arrives will be stored.
63 – 12	Not used	These bits are not used.

Message Queue Limit Increment Register Address 10460000_{16}

The message queue limit increment register (MQ_LIR) is a readable and writable, system privileged register. Software must use the MQ_LIR to manipulate the limit counter.

When any value is written to the MQ_LIR, the value of the message queue limit counter increments by one. When a value is read from the MQ_LIR, bit 12 of the register contains the sign bit of the limit counter, and the remaining bits are not defined.

Figure 7-25 shows the bit assignments for the MQ_LIR register address as they appear on the address pins of the microprocessor.

HEX	1	0				4				6				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	1	0	0	0	1	1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x = Don’t Care

Figure 7-25. Message Queue Increment Limit Counter Register Address Bit Format

When the MQ_LIR is written to, the value of the bits does not matter. Table 7-10 shows the bit format of the MQ_LIR register when it is read and describes the bits in the register.

Table 7-10. MQ_LIR Register Read Format

Bits	Name	Description
11 – 0	Not used	These bits are not used, and their value is undefined.
12	Sign bit of limit counter	This bit contains the value of the sign bit for the message queue limit counter. When set to 0, the limit counter is greater than or equal to zero. When set to 1, the limit counter is less than zero.
63 – 13	Not used	These bits are not used, and their value is undefined.

Prefetch Queue Head Register Address 10180000_{16}

The prefetch queue head (PRE_FETCH) register is a read only, general access register. When the microprocessor reads a word of data from the PRE_FETCH register, the support circuitry transfers 1 word from the head of the data prefetch queue to the microprocessor.

Figure 7-26 shows the bit assignments for the PRE_FETCH register address as they appear on the address pins of the microprocessor.

HEX	1	0				1				8				0				0				0				0			
Binary	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
	1	x	x	x	x	x	x	0	1	1	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

x = Don't Care

Figure 7-26. Prefetch Queue Head Register Address Bit Format

Table 7-11 shows the bit format of the PRE_FETCH register when it is read and describes the bits in the register.

Table 7-11. PRE_FETCH Register Read Format

Bits	Name	Description
63 – 0	Word of data	These bits contain the word of data read from the head of the prefetch queue.

Hardware Characteristics

The following subsections describe characteristics such as timing and signal flow through the CRAY T3D hardware that may have an effect on software.

Support Circuitry Data Paths

The support circuitry contains several paths for read and write data to transfer between local memory, the microprocessor, and the network interface. Figure 7-27 shows the data paths through a PE. Read data is data that is read from system memory. Write data is data that is written into system memory.

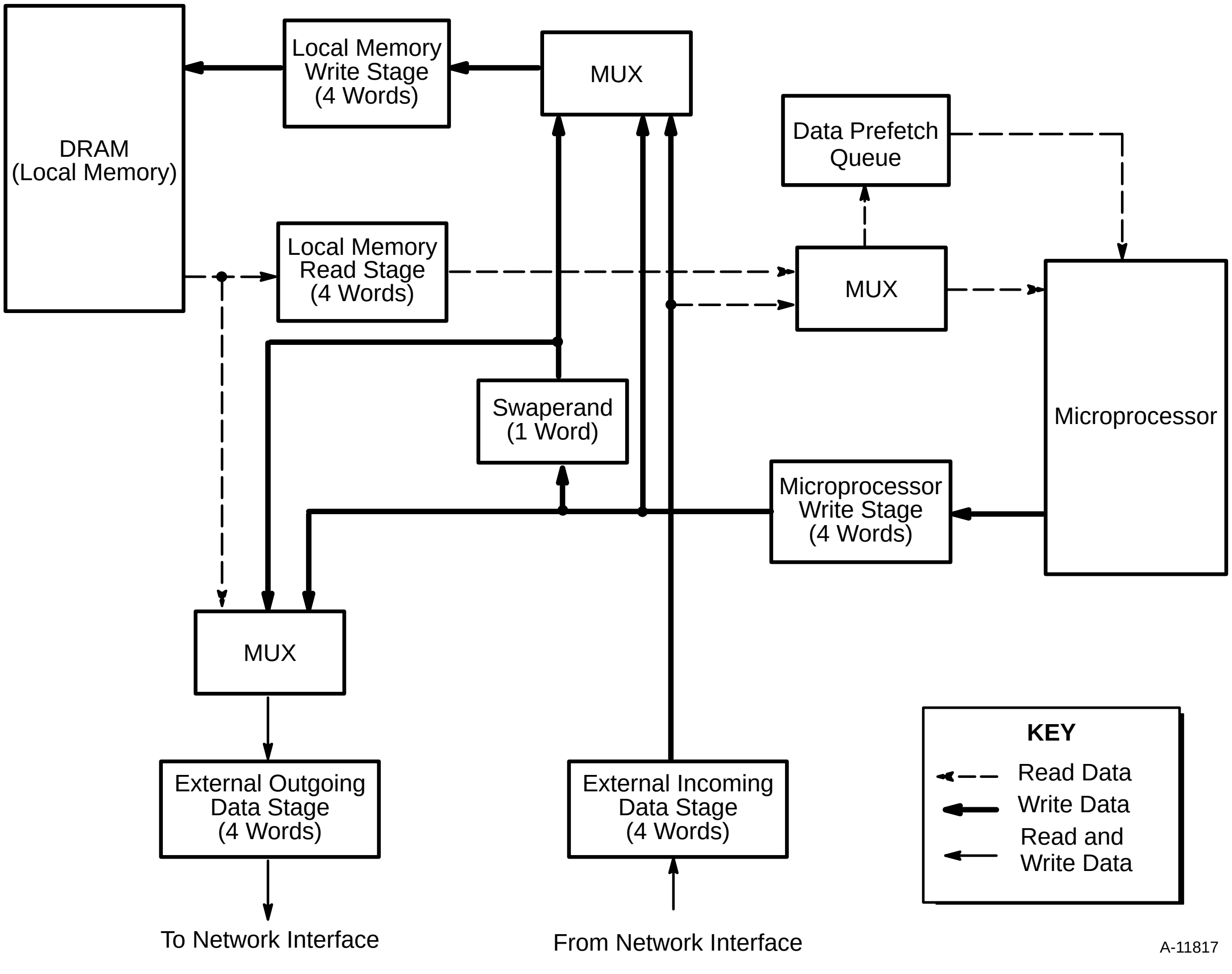

Local Memory Write Stage (4 Words)
MUX
DRAM (Local Memory)
Data Prefetch Queue
Local Memory Read Stage (4 Words)
MUX
Swaperand (1 Word)
Microprocessor
Microprocessor Write Stage (4 Words)
MUX
KEY
Read Data
Write Data
Read and Write Data
External Outgoing Data Stage (4 Words)
External Incoming Data Stage (4 Words)
To Network Interface
From Network Interface
A-11817

Figure 7-27. Data Paths through a PE

Read Data

The support circuitry contains paths for local and remote read data. Local read data transfers from local memory, through the local memory read stage, and to the microprocessor (refer to Figure 7-27). The local memory read stage is used during a cached read ahead operation. After the support circuitry performs a cached read ahead operation, the local memory read stage buffers a block of data (or instruction fetches) read from local memory.

Outgoing remote read data transfers from local memory, through the external outgoing data stage, and to the network interface. Incoming remote read data transfers from the network interface, through the external incoming data stage, and to the microprocessor.

The data prefetch queue receives prefetch read data from the MUX (refer again to Figure 7-27). The data prefetch queue sends prefetch read data to the microprocessor.

Write Data

The support circuitry also contains paths for local and remote write data. Normally, local write data transfers from the microprocessor, through the microprocessor write stage, through the local memory write stage, and to local memory (refer to Figure 7-27). Before a local atomic swap operation, 1 word of local write data transfers from the microprocessor, through the microprocessor write stage, and to the swaperand register. During the atomic swap operation, 1 word of local write data transfers from the swaperand register, through the local memory write stage, and to local memory.

Normally, outgoing remote write data transfers from the microprocessor, through the microprocessor write stage, through the external outgoing data stage, and to the network interface. Before a remote atomic swap operation, 1 word of outgoing remote write data transfers from the microprocessor, through the microprocessor write stage, and to the swaperand register. During the atomic swap operation, the word of remote write data transfers from the swaperand register, through the external outgoing data stage, and to the network interface.

Incoming remote write data transfers from the network interface, through the external incoming data stage, through the local memory write stage, and to local memory.

Support Circuitry Packet-header Paths

The support circuitry contains several paths for packet header information to transfer between local memory, the microprocessor, and the network interface. Figure 7-28 shows the packet-header paths through a PE.

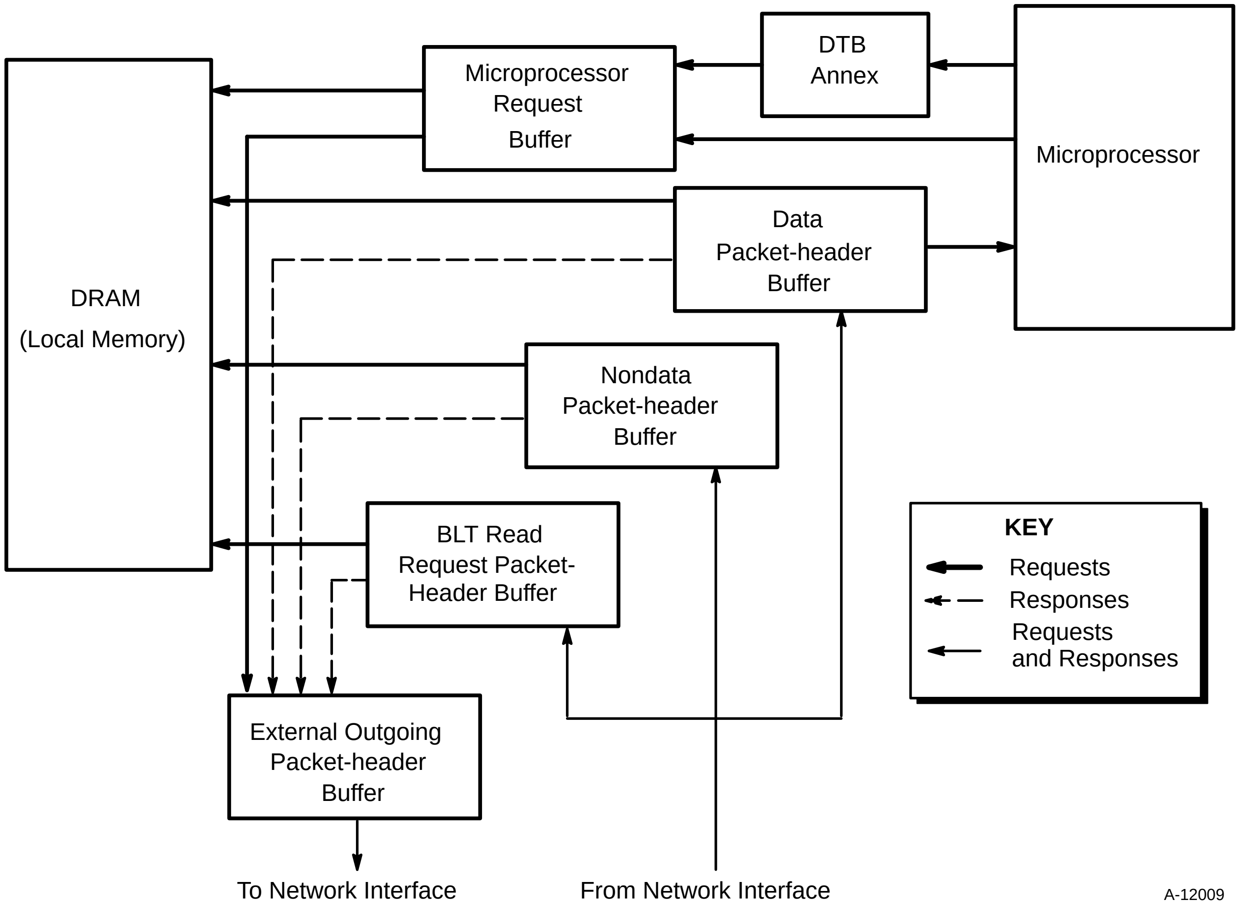

DRAM
(Local Memory)
Microprocessor Request Buffer
DTB Annex
Microprocessor
Data Packet-header Buffer
Nondata Packet-header Buffer
BLT Read Request Packet-Header Buffer
External Outgoing Packet-header Buffer
KEY
Requests
Responses
Requests and Responses
To Network Interface
From Network Interface
A-12009

Figure 7-28. Packet-header Paths through a PE

Request Packet-header Information

The support circuitry receives request header information from two sources: the microprocessor or the network interface. The microprocessor generates request address and command information that is used for a local or remote operation. The network interface receives request packets and transfers the header information to the support circuitry.

Microprocessor

The microprocessor provides the partial physical address and a cycle request code to the support circuitry. The DTB annex in the support circuitry then generates a PE number and a function code. Using the address, cycle request code, PE number, and function code, the support circuitry generates request address and command information and temporarily stores the information in the microprocessor request buffer.

When applicable, the support circuitry generates the corresponding cycle acknowledge code to signal the microprocessor that it may continue issuing other instructions. For example, during a write request, the write request address and command information is temporarily stored in the microprocessor request buffer, and the support circuitry provides an OK cycle acknowledge code to the microprocessor. This action enables the microprocessor to continue issuing other instructions while the support circuitry completes the write operation.

When the request information stored in the microprocessor request buffer is a local operation, the support circuitry transfers the address and command information to local memory. When the request information is for a remote operation, the support circuitry transfers the address and command information from the microprocessor request buffer, to the external outgoing packet-header buffer, and then to the network interface.

Network Interface

After receiving an incoming packet, the network interface sends address and command information from the header of the packet to the destination PE (and, if applicable, sends the packet data to the PE). The packet header information is temporarily stored in one of three buffers in the support circuitry: the data packet-header buffer (also referred to as the write-header buffer), the nondata packet-header buffer (also referred to as the read-header buffer), or the BLT read request packet-header buffer. (For information on where packet data is buffered, refer to “Support Circuitry Data Paths” in this section.

The data packet-header buffer temporarily stores address and command information from the header of a packet that contained data. When the data is for the microprocessor (for example, a read response), the support circuitry transfers the address and command information from the data packet-header buffer to the microprocessor. When the data is for local memory (for example, a write request), the support circuitry transfers the address and command information from the data packet-header to local memory.

The nondata packet-header buffer temporarily stores information from the header of a packet that did not contain data. When applicable, the support circuitry transfers the address and command information from the nondata packet-header buffer to local memory. For example, the support circuitry transfers address and command information to local memory if the packet was a read request packet, but does not transfer information to local memory if the packet was a write response packet.

The BLT read request packet-header buffer temporarily stores information from the header of a BLT read request packet. The support circuitry transfers address and command information from the BLT read request packet-header buffer to local memory.

Response Packet-header Information

The support circuitry generates response packet-header information using information that was stored in the data packet-header buffer, nondata packet-header buffer, or BLT read request packet-header information (that was stored in the data packet-header buffer) to generate write response packet-header information.

After generating response packet-header information, the support circuitry transfers the header information to the external outgoing packet-header buffer and then to the network interface.

NOTE: Although in most cases the support circuitry uses the information stored in a packet-header buffer to generate a response packet, in some cases the generated packet is actually another outgoing request packet. For example, when the BLT transfers data from a local PE to a remote PE, the BLT creates a read request packet to read data from the local PE. The read request transfers from the BLT to the local PE support circuitry where it is stored in the BLT read request packet-header buffer.

The support circuitry then transfers the read request to local memory. As the local memory read completes, the support circuitry uses the information stored in the BLT read request packet-header buffer to create a BLT remote write request packet header. The support circuitry then transfers the header information to the external outgoing packet-header buffer and to the network interface.

Possible Data Cache Coherence Problems

When reading from or writing to the same memory location, a data cache coherency problem may occur if a different DTB annex entry is used to perform the read than is used to perform the write. For example, the following sequence of events may cause a data cache coherency problem.

1. Perform a cached read operation from a memory location.

2. Perform a write operation to the same memory location but use a different DTB annex entry.

3. Try to perform another cached read operation from the same memory location; however, because the data in the data cache is still valid, the data returned will be the data in the data cache instead of the data that was written to the memory location in Step 2.

When different DTB annex entries are used to reference the same memory location, the address presented to the data cache for the tag compare is different. Because of this characteristic, the microprocessor store operation does not hit in the data cache and the data cache is not updated with the store data.

Performing read and write operations using the same DTB annex entry solves this problem. Then, before switching to another DTB annex entry, the memory barrier instruction should be issued to flush the write buffer.

Operating Programming Restrictions

When performing operations that modify the state of the support circuitry, some programming procedures must be followed to ensure correct results from the operations. The following list describes these procedures.

- Avoid issuing a write operation when the function code in the DTB annex is set to 1 (atomic swap) or 5 (cached atomic swap). When a write operation must be issued with the function code set to 1 or 5, issue a memory barrier (MB) instruction after the write operation and before issuing an atomic swap or cached atomic swap operation.

- After issuing a fetch-and-increment register write operation, issue an MB instruction.

- Avoid issuing a write operation to the prefetch queue head register (PRE_FETCH). When a write operation to the PRE_FETCH register must be issued, issue an MB instruction after the write operation.

  NOTE: A write operation to the PRE_FETCH register is not a hardware-supported operation but is possible to issue.

- When transferring data with memory-mapped registers, avoid using the store longword (STL) and load longword (LDL) instructions.

- Avoid using the load longword from memory to register locked (LDL_L) and store longword integer register data into memory conditional (STL_C) instructions.

Repeatedly Reading a Remote Memory Location

When software code requests that several PEs (more than 64) repeatedly read the same remote memory location, the code will be more effective when a small delay is placed in the code after each read request.

Hardware Description

The following subsections describe how the PE support circuitry implements the CRAY T3D operations.

Cycle Request and Function Codes

The AM option uses the cycle request code and function code to determine what type of operation is requested by the microprocessor. Refer again to Table 7-1 through Table 7-3 for lists of the cycle request codes, the memory function codes, and their associated operations.

Memory Barrier

When the microprocessor issues a memory barrier instruction, it generates a cycle request code of 1. The microprocessor sends this cycle request code to the AM option. After receiving a memory barrier instruction, the AM option ensures that all pending load and store instructions access memory before issuing any new instructions.

DTB Annex Read or Write

The microprocessor can read from and write to the DTB annex within the PE. The microprocessor initiates the DTB Annex read operation by generating a cycle request code of 6. This cycle request code is sent to the AM option. The AM option interprets this code and signals the AR option to read data from the DTB annex. The location that is read is specified by partial physical address bits 29 through 33 (sent to the AR option by the microprocessor). The AR option steers the data from the DTB annex to the microprocessor. The AM option also sends a 3-bit cycle acknowledge to the microprocessor. For the DTB annex read, the cycle acknowledge equals 3 (STx_C Fail).

The microprocessor initiates the DTB annex write operation by generating a cycle request code of 7. This cycle request code is sent to the AM option. The AM option interprets this code and verifies that the DTB annex write is enabled. To verify whether the DTB annex write is enabled, the AM option receives bit 0 or bit 1 of the system control register from the AR option. When bit 0 is set to a 1, the microprocessor can write to locations 1 through 15 of the DTB annex. When bit 1 is set to a 1, the microprocessor can write to locations 16 through 31 of the DTB annex.

When the DTB annex write is enabled, the AM option signals the AR option to write the data from the microprocessor to the DTB annex. The location that is written is specified by partial physical address bits 29 through 33 (sent to the AR option by the microprocessor). When the DTB write is disabled, the AM option aborts the request. When the abort occurs, the AM option signals the AR option to set the error interrupt. When the DTB annex write is complete, the AM option responds to the microprocessor with a cycle acknowledge of 3 (STx_C Fail).

Reads

The microprocessor initiates a read when it experiences a data cache miss. A data cache miss occurs when the microprocessor needs data that is not located in its internal data cache. There are two type of reads: noncacheable reads and cached reads.

Noncacheable Read

For the noncacheable read operation, the PE support circuitry reads from local or remote memory and sends this data to the microprocessor. This type of read operation is significant because the PE support circuitry signals the microprocessor not to place the data in the data cache. Instead, the microprocessor sends the data directly to the register where the data is needed.

There are two types of noncacheable reads: a normal noncacheable read and a noncacheable atomic swap read. Because the PE support circuitry performs different memory functions for each operation, examples of both are given.

NOTE: Before the microprocessor can issue a read operation, the microprocessor may have to write to the DTB annex with the associated function code and PE number.

Normal Noncacheable Read

For the normal noncacheable read operation, the microprocessor sends the address offset, the cycle request code, and the DTB annex index to the PE support circuitry. The address offset and the cycle request code are sent to the AM option. The DTB annex index is sent to the AR option. Refer to Figure 7-29.

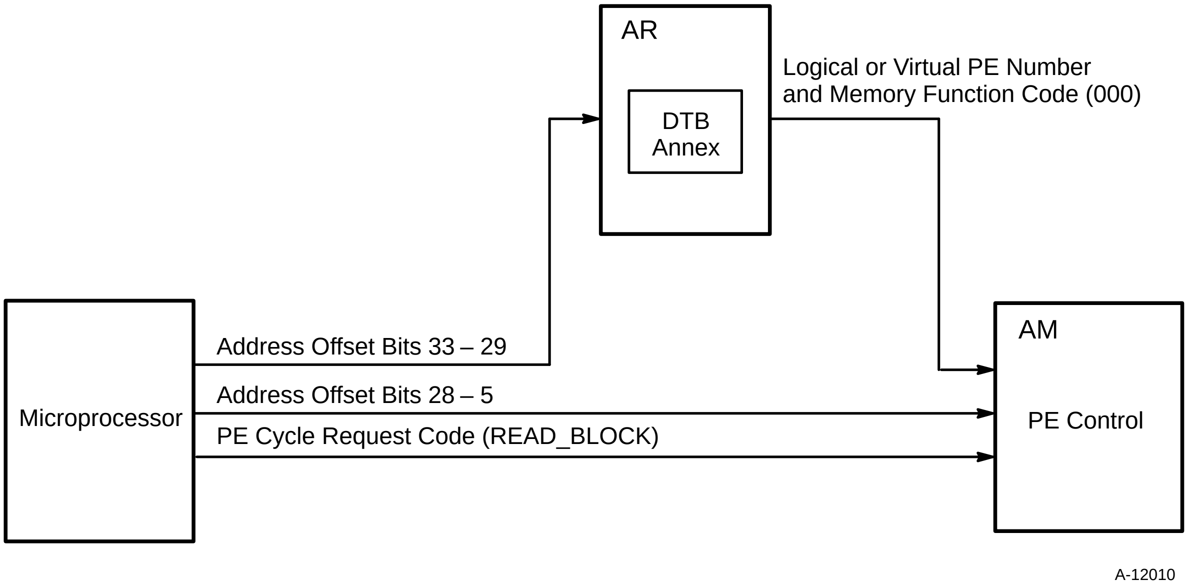

AR
DTB Annex
Logical or Virtual PE Number and Memory Function Code (000)
AM
PE Control
Microprocessor
Address Offset Bits 33 – 29
Address Offset Bits 28 – 5
PE Cycle Request Code (READ_BLOCK)
A-12010

Figure 7-29. Control and Address for a Normal Noncacheable Read

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

The AM option determines whether a memory reference is local or remote by comparing the PE number from the AR option to the logical or virtual PE register (refer to Figure 7-30). When the PE number matches the contents of the logical or virtual PE register, the reference is local. When the PE number does not match the contents of the logical or virtual PE register, the reference is remote.

When the reference is local, the AM option analyzes the address offset to determine whether the reference is for local memory or for a memory-mapped register. When the address offset equals an address between 10000000_{16} and $1FFFFFFF_{16}$ (address bit 28 = 1), the reference is for a memory-mapped register. When the address equals an address between 00000000_{16} and $0FFFFFFF_{16}$ (address bit 28 = 0), the reference is for local memory. For the normal noncacheable read operation, the address falls within the range for a local memory reference.

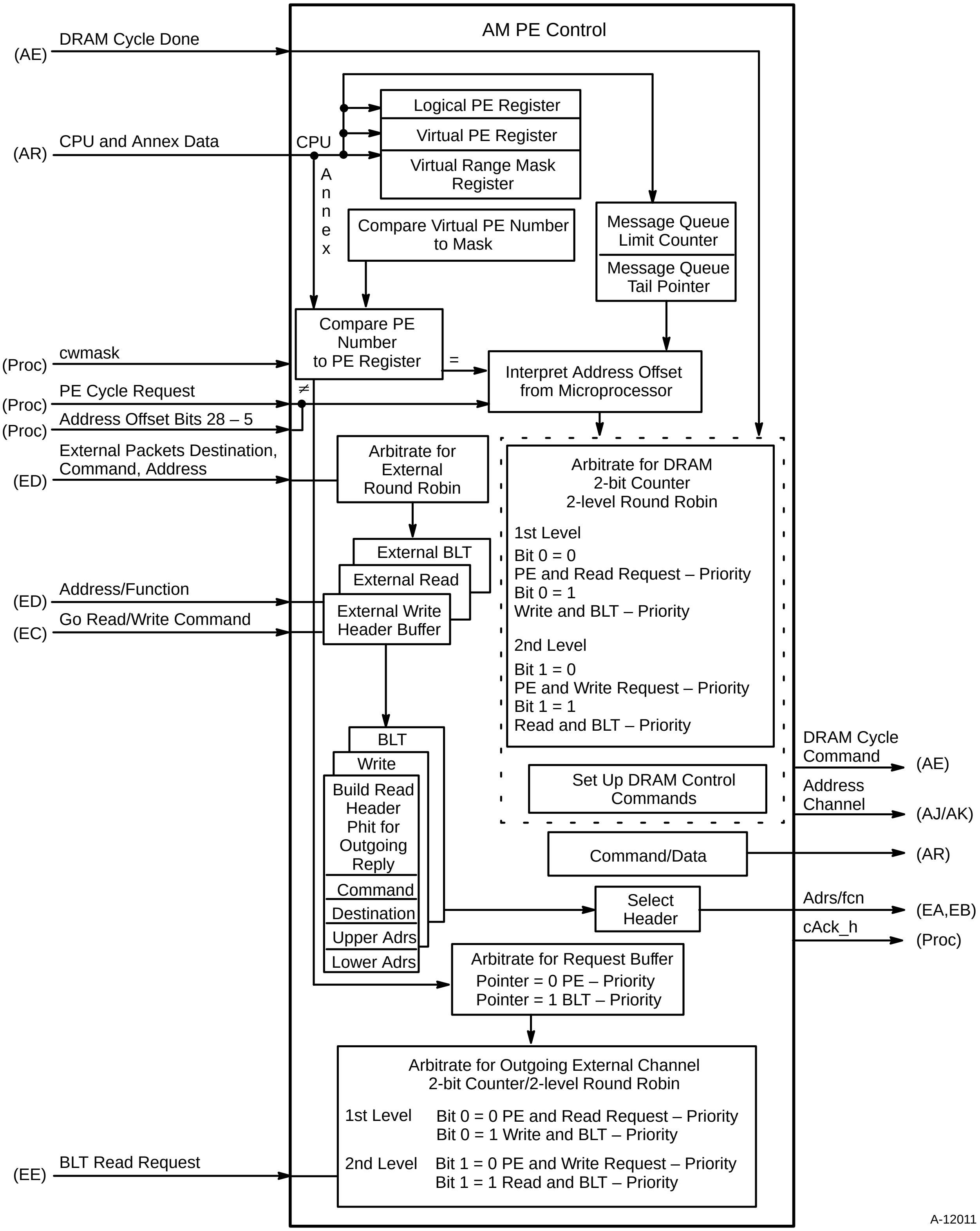

AM PE Control
(AE) DRAM Cycle Done
(AR) CPU and Annex Data
CPU
Annex
Logical PE Register
Virtual PE Register
Virtual Range Mask Register
Compare Virtual PE Number to Mask
Message Queue Limit Counter
Message Queue Tail Pointer
Compare PE Number to PE Register
=
≠
(Proc) cwmask
(Proc) PE Cycle Request
(Proc) Address Offset Bits 28 – 5
Interpret Address Offset from Microprocessor
(ED) External Packets Destination, Command, Address
Arbitrate for External Round Robin
Arbitrate for DRAM
2-bit Counter
2-level Round Robin
1st Level
Bit 0 = 0
PE and Read Request – Priority
Bit 0 = 1
Write and BLT – Priority
2nd Level
Bit 1 = 0
PE and Write Request – Priority
Bit 1 = 1
Read and BLT – Priority
External BLT
External Read
External Write Header Buffer
(ED) Address/Function
(EC) Go Read/Write Command
BLT
Write
Build Read Header Phit for Outgoing Reply
Command
Destination
Upper Adrs
Lower Adrs
Set Up DRAM Control Commands
DRAM Cycle Command (AE)
Address Channel (AJ/AK)
Command/Data
(AR)
Select Header
Adrs/fcn (EA,EB)
cAck_h (Proc)
Arbitrate for Request Buffer
Pointer = 0 PE – Priority
Pointer = 1 BLT – Priority
Arbitrate for Outgoing External Channel
2-bit Counter/2-level Round Robin
1st Level Bit 0 = 0 PE and Read Request – Priority
Bit 0 = 1 Write and BLT – Priority
2nd Level Bit 1 = 0 PE and Write Request – Priority
Bit 1 = 1 Read and BLT – Priority
(EE) BLT Read Request
A-12011

Figure 7-30. AM Option

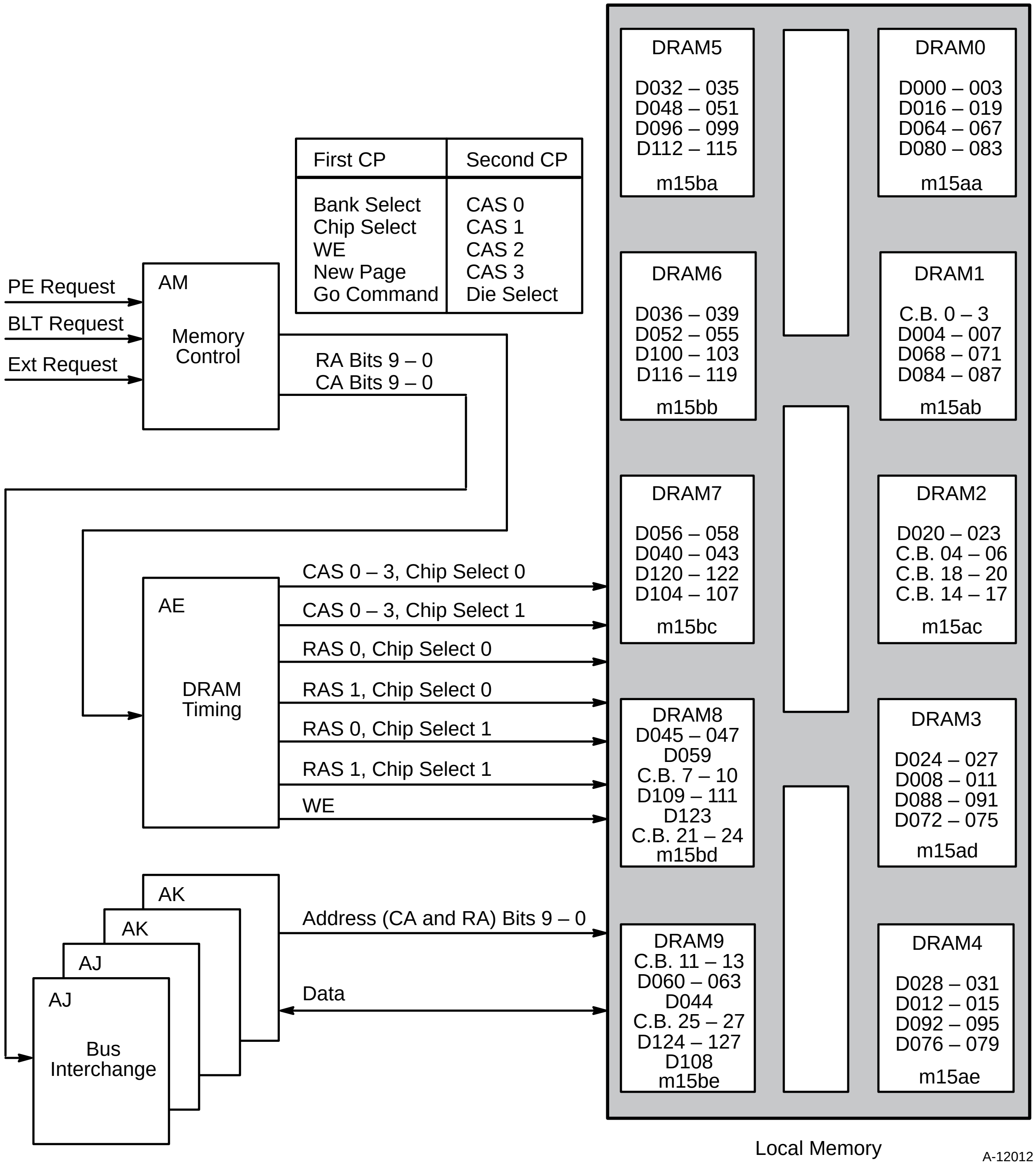
The AM option also checks for memory conflicts. Memory conflicts occur when more than one of the five types of references (refresh reference, PE reference, external read reference, external write reference, or BLT local read reference) request local memory at the same time. When a memory conflict occurs, the AM option arbitrates for local memory. The refresh is always given top priority. To arbitrate between the other references, the AM option uses a 2-bit counter to produce a 2-level round robin arbitration sequence. For more information on the 2-level round robin arbitration sequence, refer again to “AM Option” in Section 3, “Processing Element Node.”

When the normal noncacheable read operation has priority to access memory, the AM generates DRAM control commands. These control commands are sent to the AE option over 2 clock periods (refer to Figure 7-31). The Bank Select, Chip Select, Write Enable, New Page, and Go commands are sent during the first clock period; and the CAS 0 Mask, CAS 1 Mask, CAS 2 Mask, CAS 3 Mask, and Die Select commands are sent during the second clock period.

The AE option uses these DRAM commands to generate the CAS 0 – 3 and RAS 0 – 1 signals. The CAS 0 – 3 signals enable data to be read from the appropriate memory chips on the daughter cards. The RAS 0 – 1 signals enable data to be read from one of the two dies that may be present within the memory chips. When there is only one die within the memory chip, the RAS 1 signal is not used.

The AM option also supplies the AE option with timing parameter data. The AE option uses the timing parameter data to determine how long to assert the RAS and CAS signals to the memory chips. The AE option makes this determination by comparing the timing parameter data to the value of a bank timing counter. When the timing parameter data equals the value of the counter (the counter increments by one each clock period), the AE starts asserting the signal and resets the bank timing counter to zero. The AE option also compares the timing parameter data to the bank timing counter to determine when to stop asserting the signals.

The AM option also sends the address offset to the AJ and AK options. The AJ and AK options send this address to the DRAM. After the DRAM is addressed, the AJ and AK options read the data from the location.



First CP	Second CP
Bank Select	CAS 0
Chip Select	CAS 1
WE	CAS 2
New Page	CAS 3
Go Command	Die Select

NOTE: The row address strobe (RAS) asserts when referencing a new page or doing a refresh. A new page reference occurs when a different page in the same die is referenced, when the other die in the rank is referenced, or after a refresh. The column address strobe (CAS) 0 through 3 asserts when referencing the same page or doing a refresh. The CAS mask indicates which of the 4 CAS signals will be asserted for the reference. The CAS mask points to valid halfwords within a cache line. The CAS asserts during the first clock period for a read and during the second clock period for a write. The CAS asserts before the RAS during a refresh. The write enable asserts during the first clock period of the write and continues to assert for the duration of the CAS.

Figure 7-31. Distributing Address and Control to a Daughter Card

The AJ and AK options receive the DRAM Data Path Control and CPU Data Path Control signals from the AM option (refer to Figure 7-32). The DRAM Data Path Control signal enables the AJ and AK options to input the data from memory 1 word at a time. The CPU Path Control signal enables the AJ and AK options to steer the data to the microprocessor.

While the AJ and AK options are transferring the data to the microprocessor, the AM option signals the microprocessor using the read data acknowledge to input the data, to perform SECDED, and not to place it in its internal cache.

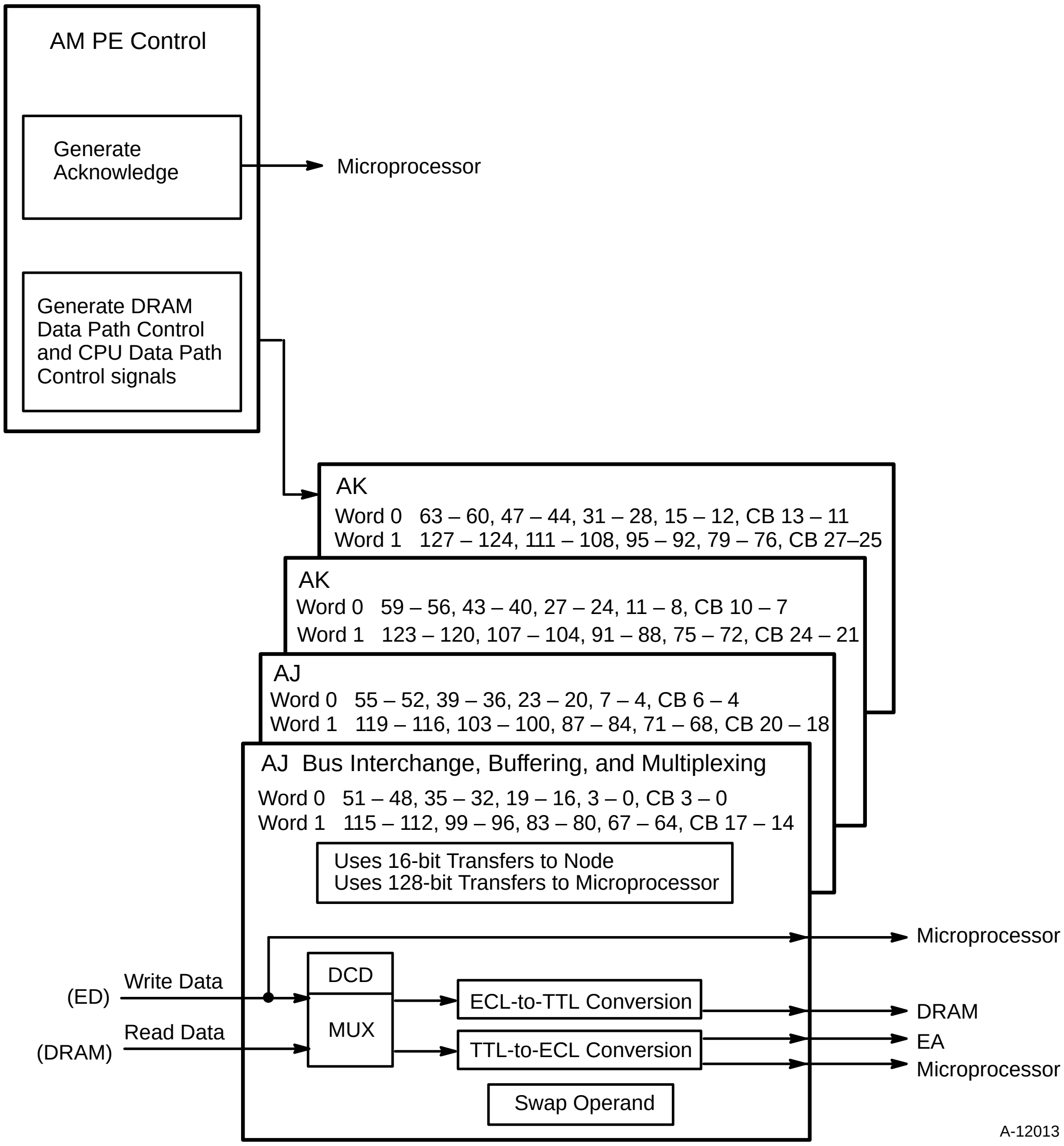

AM PE Control
Generate Acknowledge
Microprocessor
Generate DRAM Data Path Control and CPU Data Path Control signals
AK
Word 0 63 – 60, 47 – 44, 31 – 28, 15 – 12, CB 13 – 11
Word 1 127 – 124, 111 – 108, 95 – 92, 79 – 76, CB 27–25
AK
Word 0 59 – 56, 43 – 40, 27 – 24, 11 – 8, CB 10 – 7
Word 1 123 – 120, 107 – 104, 91 – 88, 75 – 72, CB 24 – 21
AJ
Word 0 55 – 52, 39 – 36, 23 – 20, 7 – 4, CB 6 – 4
Word 1 119 – 116, 103 – 100, 87 – 84, 71 – 68, CB 20 – 18
AJ Bus Interchange, Buffering, and Multiplexing
Word 0 51 – 48, 35 – 32, 19 – 16, 3 – 0, CB 3 – 0
Word 1 115 – 112, 99 – 96, 83 – 80, 67 – 64, CB 17 – 14
Uses 16-bit Transfers to Node
Uses 128-bit Transfers to Microprocessor
Microprocessor
Write Data
(ED)
Read Data
(DRAM)
DCD
MUX
ECL-to-TTL Conversion
TTL-to-ECL Conversion
Swap Operand
DRAM
EA
Microprocessor
A-12013

Figure 7-32. Reading Data from Memory and Sending it to the Microprocessor

When the noncacheable read operation is remote, the AM option checks the validity of the virtual PE number, creates the header phits, and arbitrates for the network interface request buffers and the outgoing channel (refer to Figure 7-33). For more information on how the AM option does this, refer to “AM Option” in Section 3, “Processing Element Node.”

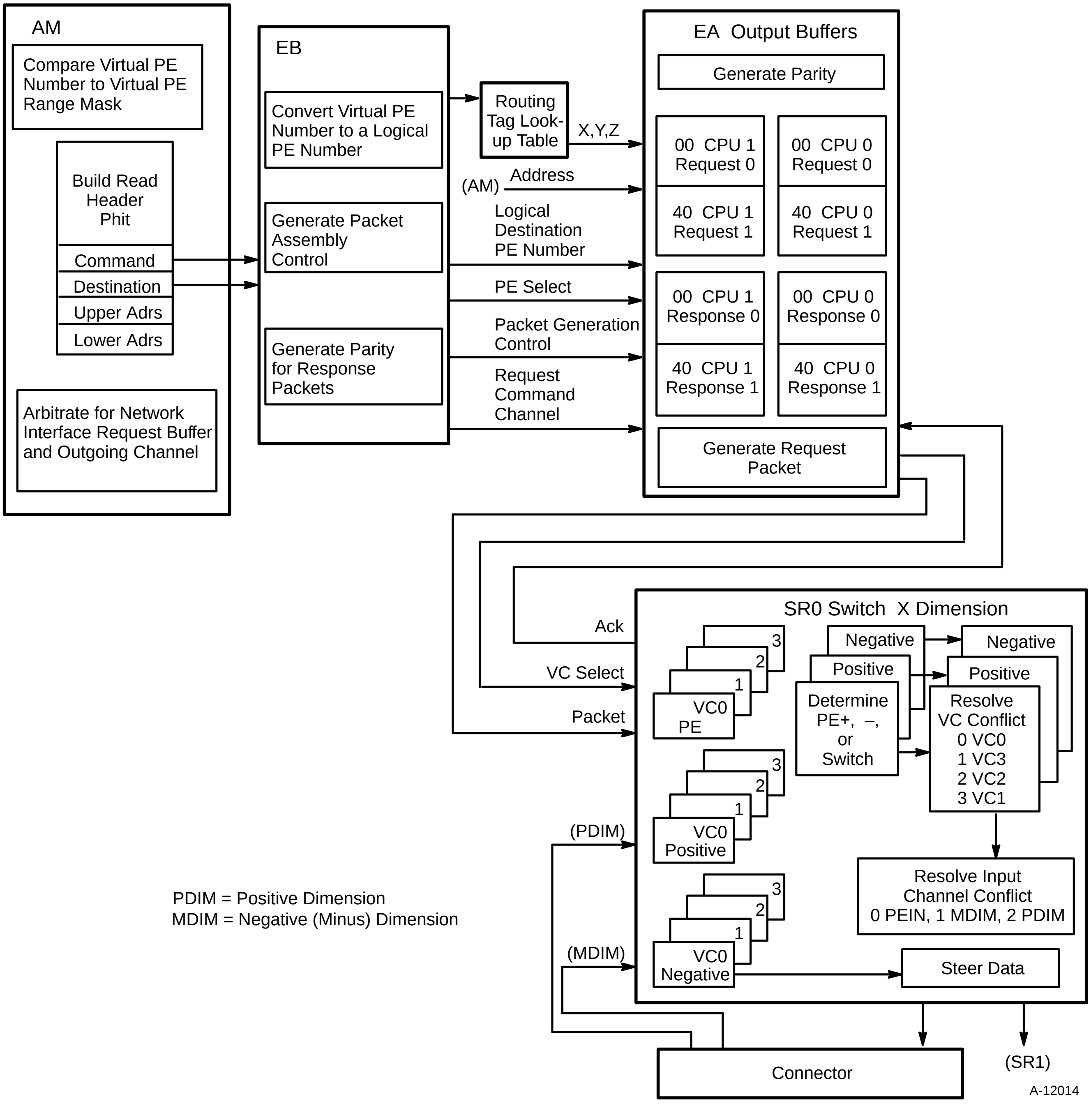

AM
Compare Virtual PE Number to Virtual PE Range Mask
Build Read Header Phit
Command
Destination
Upper Adrs
Lower Adrs
Arbitrate for Network Interface Request Buffer and Outgoing Channel
EB
Convert Virtual PE Number to a Logical PE Number
Generate Packet Assembly Control
Generate Parity for Response Packets
Routing Tag Look-up Table
X,Y,Z
(AM)
Address
Logical Destination PE Number
PE Select
Packet Generation Control
Request Command Channel
EA Output Buffers
Generate Parity
00 CPU 1 Request 0
00 CPU 0 Request 0
40 CPU 1 Request 1
40 CPU 0 Request 1
00 CPU 1 Response 0
00 CPU 0 Response 0
40 CPU 1 Response 1
40 CPU 0 Response 1
Generate Request Packet
SR0 Switch X Dimension
Ack
VC Select
Packet
3
2
1
VC0 PE
VC0 Positive
VC0 Negative
(PDIM)
(MDIM)
Negative
Positive
Determine PE+, –, or Switch
Negative
Positive
Resolve VC Conflict 0 VC0 1 VC3 2 VC2 3 VC1
Resolve Input Channel Conflict 0 PEIN, 1 MDIM, 2 PDIM
Steer Data
Connector
(SR1)
PDIM = Positive Dimension
MDIM = Negative (Minus) Dimension
A-12014

Figure 7-33. Generating a Remote Request

When the normal noncacheable read request has priority to access the buffer, the AM option sends the header phits to the EA and EB options of the network interface. The command and destination PE number are sent to the EB option and the address is sent to the EA option in the order shown in Figure 7-34.

<table>
<tr><th></th><th>R612</th><th>R611</th><th>R610</th><th>R609</th><th>R608</th><th>R607</th><th>R606</th><th>R605</th><th>R604</th><th>R603</th><th>R602</th><th>R601</th><th>R600</th></tr>
<tr><td>CP1</td><td>Go</td><td colspan="3">Packet Type</td><td>Resp Req’d</td><td colspan="3">Command Code</td><td colspan="5">Prefetch Queue Address</td></tr>
<tr><td>CP2</td><td>PE Adjust Disable</td><td colspan="12">PE Number</td></tr>
<tr><td>CP3</td><td>Die Select</td><td colspan="10">Row Address</td><td></td><td>Chip Select</td></tr>
<tr><td>CP4</td><td>BLT Range Error</td><td colspan="10">Column Address</td><td></td><td></td></tr>
</table>

Figure 7-34. Command Channel From the AM Option to the EA and EB Options

The EA and EB options assemble the header phits into a request packet and send it to the SR option that handles the X dimension of the interconnect network. From the SR option, the packet is transferred through the network to the destination node.

When the request packet reaches the destination node, the EC and ED options input the request packet (refer to Figure 7-35). When the entire packet is buffered in the ED option, the EC option checks the packet for destination and network parity errors and signals the ED option to read the packet from the buffer.

The ED option reads the packet from the buffer, checks the packet for network buffer parity errors, and sends the command, remote address, and source PE number to the AM option in the destination PE.

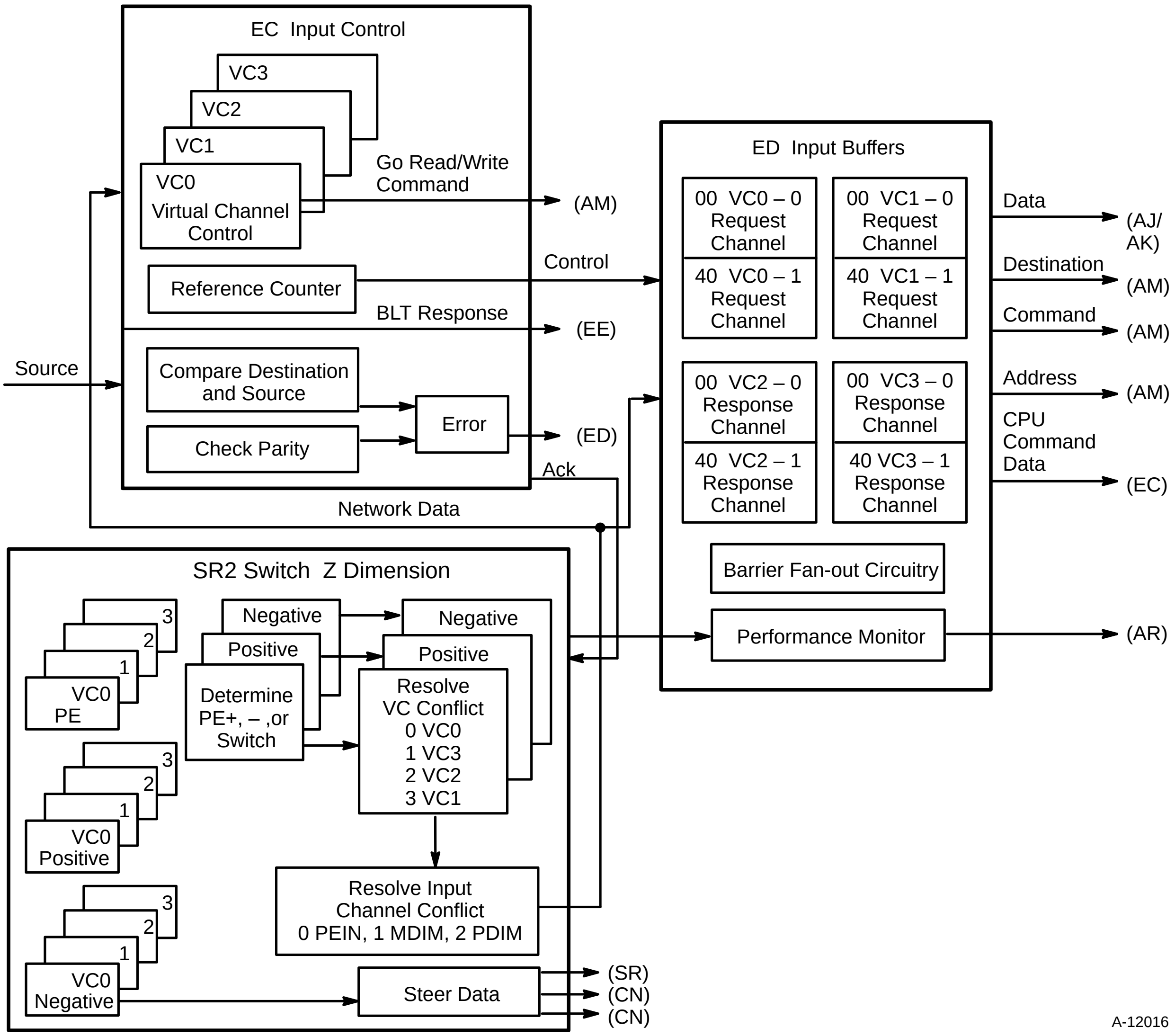

EC Input Control
VC3
VC2
VC1
VC0
Virtual Channel Control
Go Read/Write Command
(AM)
Reference Counter
Control
BLT Response
(EE)
Source
Compare Destination and Source
Check Parity
Error
(ED)
Ack
Network Data
ED Input Buffers
00 VC0 – 0 Request Channel
40 VC0 – 1 Request Channel
00 VC1 – 0 Request Channel
40 VC1 – 1 Request Channel
00 VC2 – 0 Response Channel
40 VC2 – 1 Response Channel
00 VC3 – 0 Response Channel
40 VC3 – 1 Response Channel
Barrier Fan-out Circuitry
Performance Monitor
Data
(AJ/ AK)
Destination
(AM)
Command
(AM)
Address
(AM)
CPU Command Data
(EC)
(AR)
SR2 Switch Z Dimension
3
2
1
VC0 PE
VC0 Positive
VC0 Negative
Negative
Positive
Determine PE+, – ,or Switch
Negative
Positive
Resolve VC Conflict 0 VC0 1 VC3 2 VC2 3 VC1
Resolve Input Channel Conflict 0 PEIN, 1 MDIM, 2 PDIM
Steer Data
(SR)
(CN)
(CN)
A-12016

Figure 7-35. Input of the Request Packet

From the command, the address, and the source PE number, the AM option yields DRAM control signals and response header phits. The AM option sends the DRAM control signals to the AE option and the address to the AJ and AK options (refer to Figure 7-36).

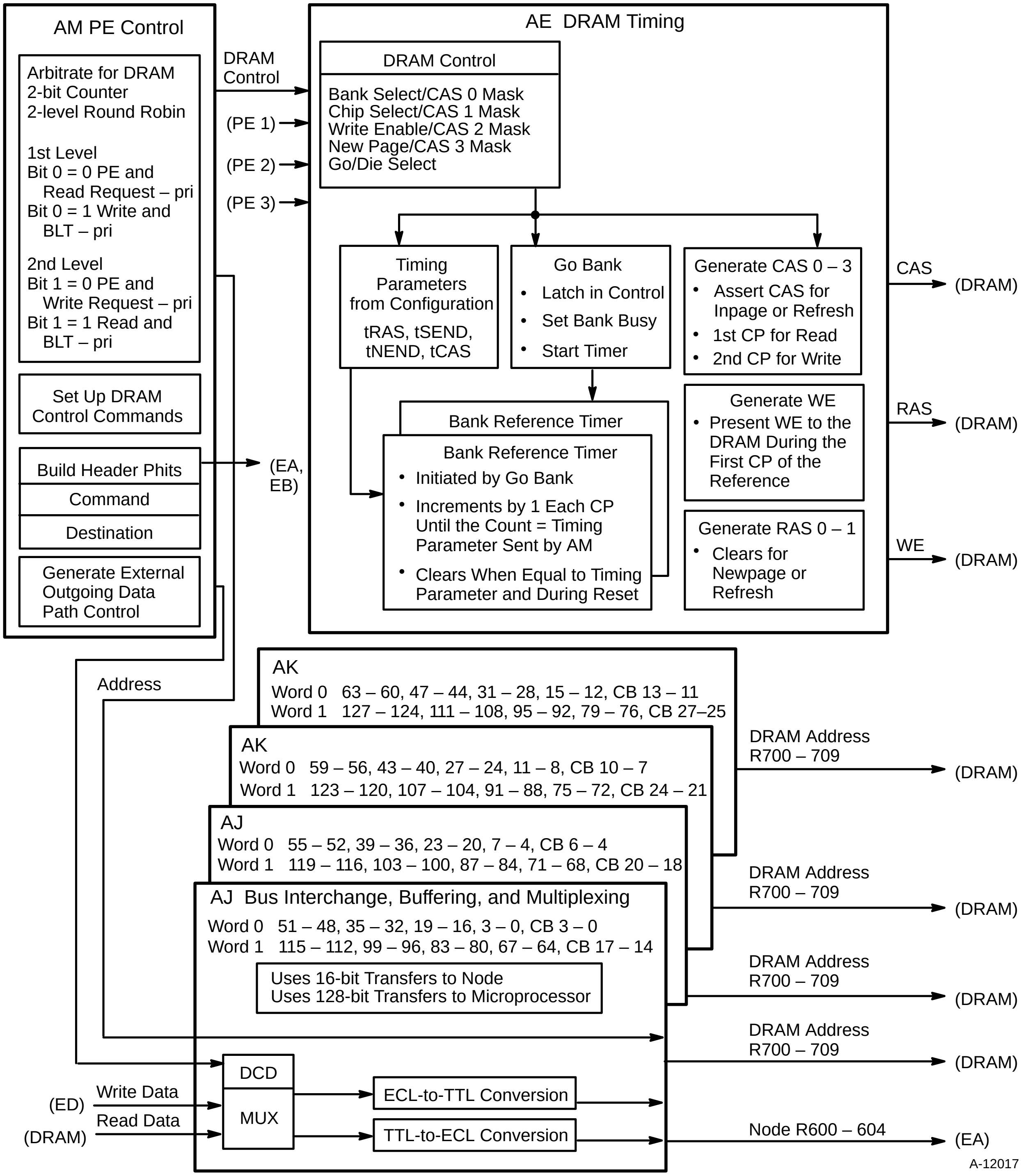

AM PE Control
Arbitrate for DRAM
2-bit Counter
2-level Round Robin
1st Level
Bit 0 = 0 PE and Read Request – pri
Bit 0 = 1 Write and BLT – pri
2nd Level
Bit 1 = 0 PE and Write Request – pri
Bit 1 = 1 Read and BLT – pri
Set Up DRAM Control Commands
Build Header Phits
Command
Destination
Generate External Outgoing Data Path Control
DRAM Control
(PE 1)
(PE 2)
(PE 3)
(EA, EB)
Address
AE DRAM Timing
DRAM Control
Bank Select/CAS 0 Mask
Chip Select/CAS 1 Mask
Write Enable/CAS 2 Mask
New Page/CAS 3 Mask
Go/Die Select
Timing Parameters from Configuration
tRAS, tSEND, tNEND, tCAS
Go Bank
• Latch in Control
• Set Bank Busy
• Start Timer
Generate CAS 0 – 3
• Assert CAS for Inpage or Refresh
• 1st CP for Read
• 2nd CP for Write
Bank Reference Timer
Bank Reference Timer
• Initiated by Go Bank
• Increments by 1 Each CP Until the Count = Timing Parameter Sent by AM
• Clears When Equal to Timing Parameter and During Reset
Generate WE
• Present WE to the DRAM During the First CP of the Reference
Generate RAS 0 – 1
• Clears for Newpage or Refresh
CAS
(DRAM)
RAS
(DRAM)
WE
(DRAM)
AK
Word 0 63 – 60, 47 – 44, 31 – 28, 15 – 12, CB 13 – 11
Word 1 127 – 124, 111 – 108, 95 – 92, 79 – 76, CB 27–25
AK
Word 0 59 – 56, 43 – 40, 27 – 24, 11 – 8, CB 10 – 7
Word 1 123 – 120, 107 – 104, 91 – 88, 75 – 72, CB 24 – 21
AJ
Word 0 55 – 52, 39 – 36, 23 – 20, 7 – 4, CB 6 – 4
Word 1 119 – 116, 103 – 100, 87 – 84, 71 – 68, CB 20 – 18
AJ Bus Interchange, Buffering, and Multiplexing
Word 0 51 – 48, 35 – 32, 19 – 16, 3 – 0, CB 3 – 0
Word 1 115 – 112, 99 – 96, 83 – 80, 67 – 64, CB 17 – 14
Uses 16-bit Transfers to Node
Uses 128-bit Transfers to Microprocessor
DCD
MUX
ECL-to-TTL Conversion
TTL-to-ECL Conversion
(ED)
Write Data
Read Data
(DRAM)
DRAM Address R700 – 709
(DRAM)
DRAM Address R700 – 709
(DRAM)
DRAM Address R700 – 709
(DRAM)
DRAM Address R700 – 709
(DRAM)
Node R600 – 604
(EA)
A-12017

Figure 7-36. Generation of DRAM Control and Response Header Phits

The AJ and AK options, along with the AE option, address the DRAM. The AE option supplies the RAS and CAS signals to the DRAM. The AJ and AK options supply the internal address. After the DRAM is addressed, the AJ and AK options read the data from this memory location.

The AM option also generates a response command and destination PE number using the command and source PE number phits of the request packet. This information is sent to the EB option.

The AM option sends the External Outgoing Data Path Control signal to the AJ and AK options. This signal enables the AJ and AK options to steer the read data to the EA and EB options of the network interface.

The EA and EB options assemble this information into a response packet (refer again to Figure 7-33). The EA option sends the response packet to the SR option that handles the X dimension of the interconnect network. From this option, the response packet is transferred through the network, back to the source node.

When the response packet reaches the source node, the EC and ED options input the response packet (refer again to Figure 7-35). As soon as the entire packet is buffered in the ED option, the EC option checks the packet for destination and network parity errors, and signals the ED option to read the packet from the buffer. The command, remote address, and source PE number of the packet are sent to the AM option. The data is sent to the AJ and AK options.

The AJ and AK options receive two control signals: one from the EC option and one from the AM option (refer to Figure 7-37). From the EC option, the AJ and AK options receive the External Incoming Data Path Control signal. This signal indicates the amount of data the ED option will send to the AJ and AK options. From the AM option, the AJ and AK options receive the CPU Read Data Select signal. This signal enables the AJ and AK options to select the external data and send it to the microprocessor.

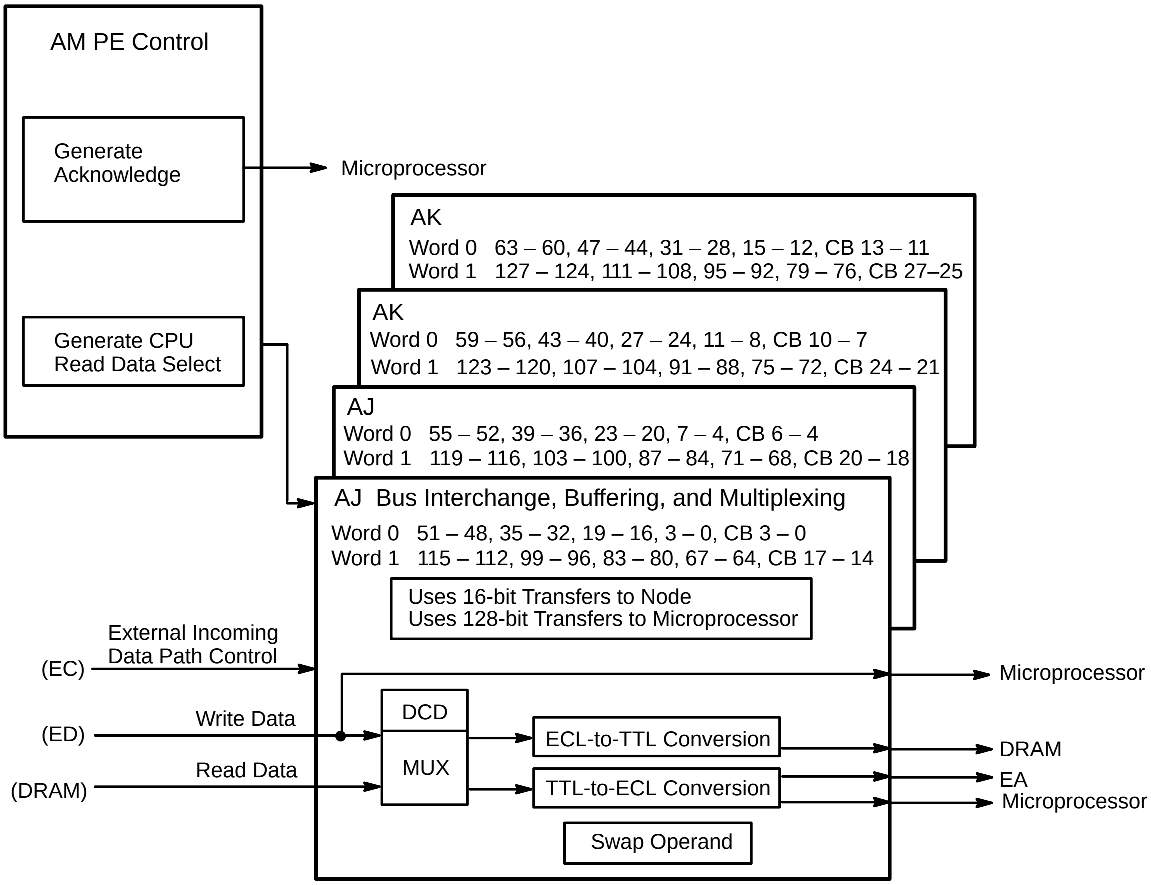

AM PE Control
Generate Acknowledge
Microprocessor
Generate CPU Read Data Select
AK
Word 0 63 – 60, 47 – 44, 31 – 28, 15 – 12, CB 13 – 11
Word 1 127 – 124, 111 – 108, 95 – 92, 79 – 76, CB 27–25
AK
Word 0 59 – 56, 43 – 40, 27 – 24, 11 – 8, CB 10 – 7
Word 1 123 – 120, 107 – 104, 91 – 88, 75 – 72, CB 24 – 21
AJ
Word 0 55 – 52, 39 – 36, 23 – 20, 7 – 4, CB 6 – 4
Word 1 119 – 116, 103 – 100, 87 – 84, 71 – 68, CB 20 – 18
AJ Bus Interchange, Buffering, and Multiplexing
Word 0 51 – 48, 35 – 32, 19 – 16, 3 – 0, CB 3 – 0
Word 1 115 – 112, 99 – 96, 83 – 80, 67 – 64, CB 17 – 14
Uses 16-bit Transfers to Node
Uses 128-bit Transfers to Microprocessor
External Incoming Data Path Control
(EC)
Write Data
(ED)
Read Data
(DRAM)
DCD
MUX
ECL-to-TTL Conversion
TTL-to-ECL Conversion
Swap Operand
Microprocessor
DRAM
EA
Microprocessor

Figure 7-37. Steering Data from the Network Interface to the Microprocessor

The AM option also sends a read data acknowledge (5; OK_NCACHE) to the microprocessor. This acknowledge instructs the microprocessor to input the data, to perform SECDED, and not to place the data in its internal data cache.

Noncacheable Atomic Swap Read

The noncacheable atomic swap read retrieves a 64-bit word from a location in system memory and sends it to the requesting microprocessor (refer to Figure 7-38). This location in system memory is then written with the contents of the swaperand register. The swaperand register is located in the AJ and AK options.

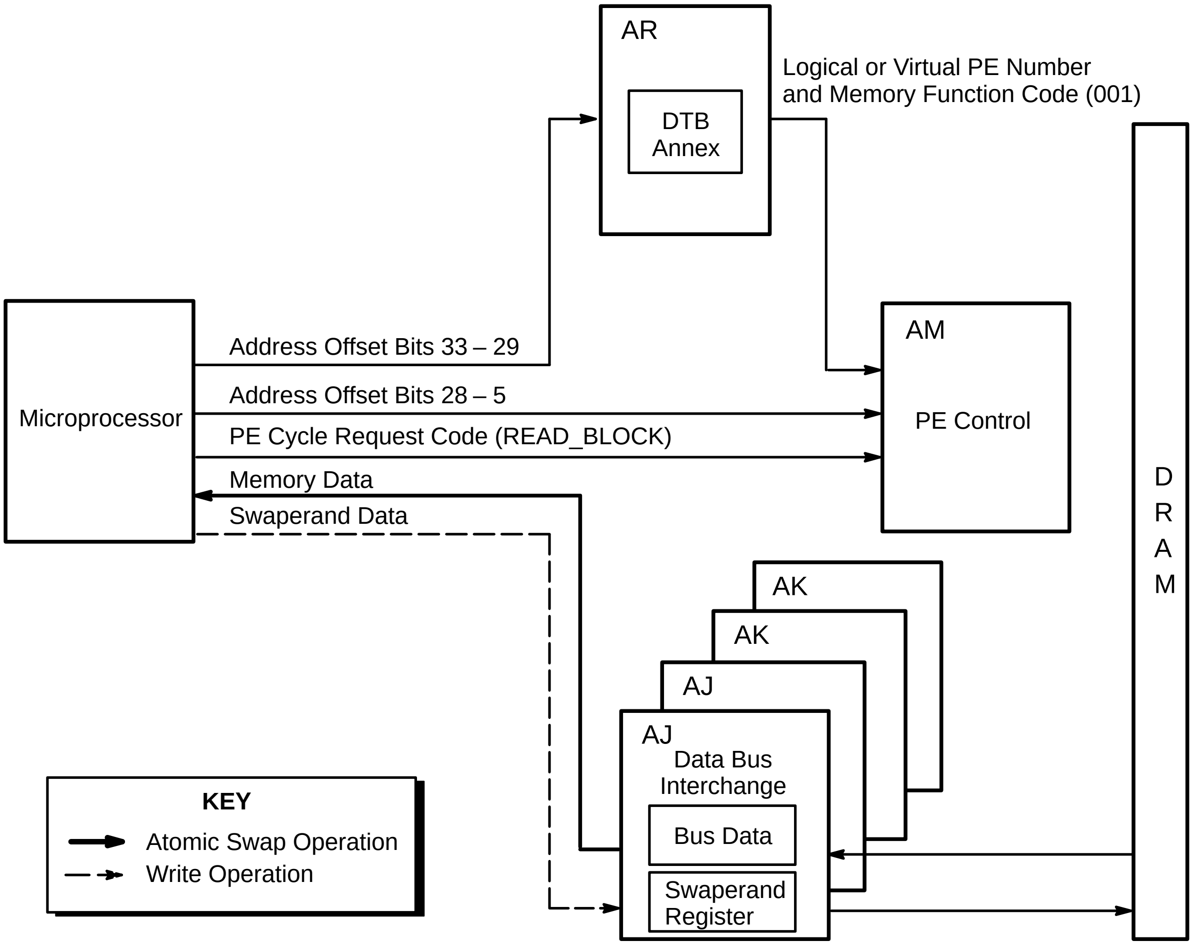


Figure 7-38. Noncacheable Atomic Swap Read from Local Memory

Before the microprocessor can initiate a noncacheable atomic swap read, the microprocessor must load 64 bits of data into the swaperand register. To do this, the microprocessor sends the address offset, the cycle request code, and the DTB annex index to the PE support circuitry. The address offset and the cycle request code are sent to the AM option. The DTB annex index is sent to the AR option.

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

The AM option determines whether the memory reference is local or remote by comparing the PE number from the AR option to the logical or virtual PE register. For a memory-mapped register load, the reference is local so the PE number will match the logical or virtual PE register.

Next, the AM option analyzes the address offset to determine whether the reference is for local memory or for a memory-mapped register. To load the swaperand register, the address offset equals 10300000_{16}.

Once the AM option has determined that the request is for a swaperand register load, the AM option generates the Data To External Control signal that is sent to the AJ and AK options. The Data To External Control signal is a 4-bit code that is sent to the AJ and AK options over 2 clock periods. To load the swaperand register, the Data To External Control signal equals a 2 (10_2) during the first clock period and a 3 (11_2) during the second clock period. When the 2-bits during the second clock period indicate the swap, the AJ and AK options activate the Go Load Swap signal. This signal, along with the 2-bit code from the first clock period, activates the Load Swaperand Register signal. This signal enables the AJ and AK options to load the 64-bits of data from the microprocessor into the swaperand register.

Once the swaperand register is loaded with data, the microprocessor initiates the noncacheable atomic swap read by supplying the PE support circuitry with the address offset, the cycle request code, and the DTB annex index. The address offset and the cycle request code are sent to the AM option. The DTB annex index is sent to the AR option.

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

For the noncacheable atomic swap read, the cycle request code equals 4 (READ_BLOCK) and the memory function code equals 1 (noncacheable atomic swap read). The PE number from the DTB annex may equal the virtual or logical PE number of the local PE or it may equal the virtual or logical PE number of a remote PE. The address offset points to a location in system memory. The contents of this location will be sent to the microprocessor and replaced with the contents of the swaperand register.

To perform the read of the memory location, the AM option performs the same operation as indicated by the example for the noncacheable read operation.

Immediately following the read, the AM option instructs the AJ and AK options to write the contents of the swaperand register to the same memory location where the data was read. To steer the contents of the swaperand to a location in local memory, the AM option generates the DRAM control signals, the address and the DRAM Write Data Control signal. The DRAM control signals are sent to the AE option, and the DRAM Write Control signal is sent to the AJ and AK options along with the address.

The AE, AJ, and AK options use the DRAM control signals and the address to address local memory. Next, the AJ and AK options use the DRAM Write Data Control signal to select the swaperand data and write this data to the DRAM.

When swapping with a remote memory location, the AM option generates the Data To External Control signal and sends this signal to the AJ and AK options. The AJ and AK options use this signal to select the swaperand data and steer this data to the network interface options. The EA option in the network interface buffers the swaperand data in a request buffer.

The AM option also supplies the EA and EB options with the request packet header. The command and destination PE number are sent to the EB option, and the address is sent to the EA option. The address is also buffered in the request buffer.

The EA and EB options assemble this information into a request packet and send the packet through the interconnect network to the destination node. The EC and ED options in the network interface receive the packet. The ED option buffers the packet, and the EC checks it for destination and network parity errors.

Under the control of the EC option, the ED option reads the request packet from the buffer. The ED option checks for network buffer parity errors and sends the command, address, and source PE number to the AM option. The ED option sends the data to the AJ and AK options. From the command, the addresses, and the source PE number, the AM option yields DRAM control signals and response header phits. The AM option sends the DRAM control signals to the AE option and the address to the AJ and AK options.

The AJ and AK options, along with the AE option, address the DRAM. The AE option supplies the RAS, CAS, and WE signals to the DRAM. The AJ and AK options supply the internal address. After the DRAM is addressed, the AJ and AK options deliver the write data to the DRAM.

The AM option generates a new command and destination PE number using the command and source PE number phits of the request packet. This information is sent to the EB option. The EB option generates parity for these two phits and sends them to the EA option. The EA option assembles this information into a response packet and sends it through the interconnect network to the source PE.

When the packet reaches the source PE (from the network interface), the AM option analyzes the response. Once the AM option determines that this is a response to the noncacheable atomic swap operation, the AM option sends the cycle acknowledge signal (4; OK) to the microprocessor, signalling the completion of the noncacheable atomic swap read.

Cached Reads

For the cached read operation, the PE support circuitry reads from local or remote memory and sends this data to the microprocessor. This type of read operation is significant because the PE support circuitry signals the microprocessor to place the data in its internal cache.

There are three types of cached reads: a normal cached read, a cached read ahead, and a cached atomic swap. Because the PE support circuitry performs a different memory function for the cached read ahead operation, an example of this operation is given. An explanation of the differences between the normal noncacheable read and the normal cached read and between the noncacheable atomic swap read and the cached atomic swap read is also given.

Normal Cached Read

The CRAY T3D hardware implements the normal cached read operation in the same way it does the noncacheable read operation, with the exception of the memory function code and the read data acknowledgement. For the normal cached read operation, the memory function code equals 4 or 7. The read data acknowledgement is equal to 7. The AM option sends this acknowledgement to the microprocessor to instruct the microprocessor to input the data, perform SECDED, and place the data in the cache.

Cached Read Ahead

For the cached read ahead operation, the PE support circuitry reads a block of data from local memory and sends this data to the microprocessor (refer to Figure 7-39). Immediately after sending the data to the microprocessor, the PE support circuitry retrieves the next consecutive block of data. This block of data is held in the PE support circuitry. When the microprocessor issues another load instruction with the same address as the block of data held in the PE support circuitry, the PE support circuitry steers the data to the microprocessor.

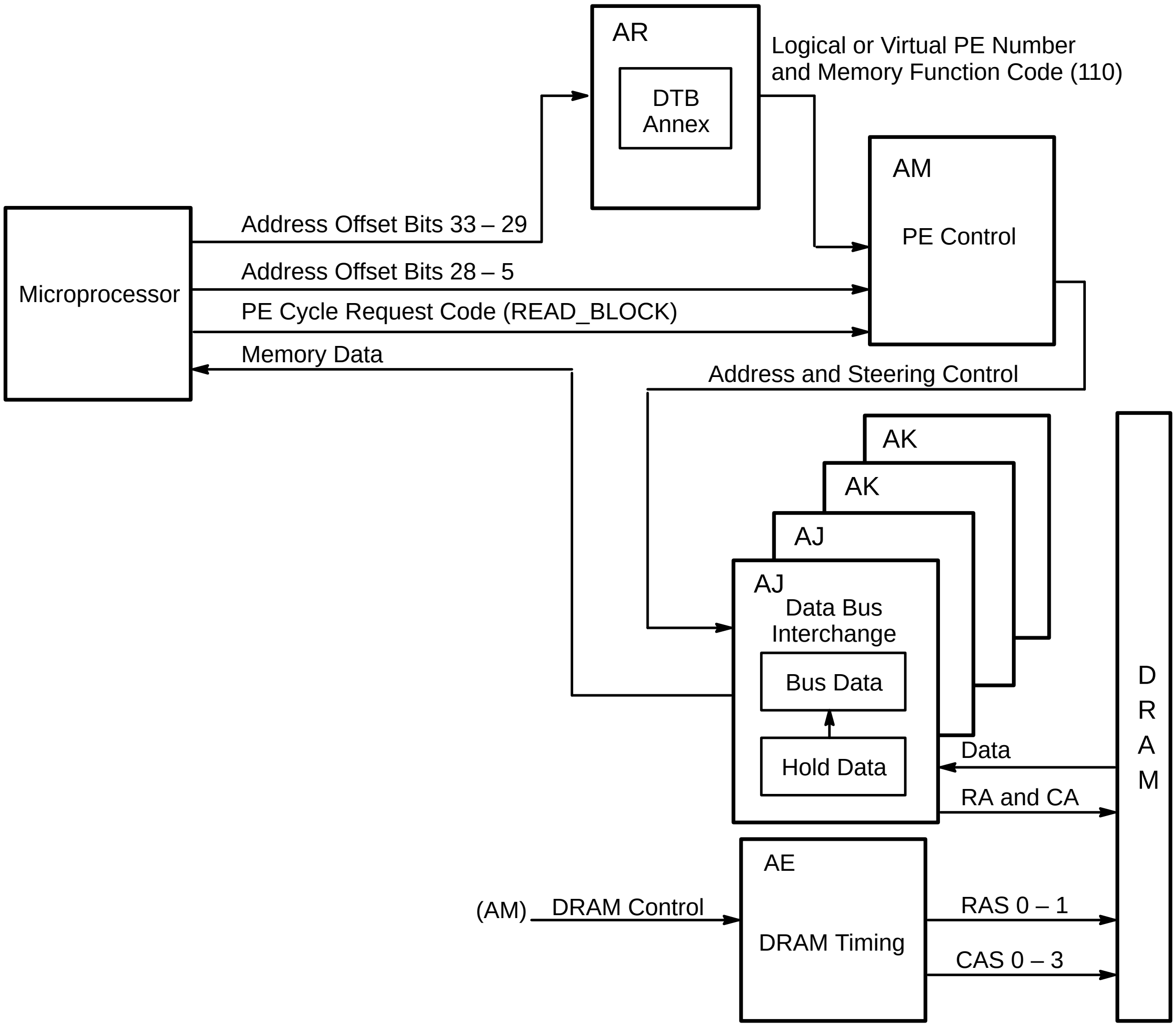

AR
DTB Annex
Logical or Virtual PE Number and Memory Function Code (110)
AM
PE Control
Microprocessor
Address Offset Bits 33 – 29
Address Offset Bits 28 – 5
PE Cycle Request Code (READ_BLOCK)
Memory Data
Address and Steering Control
AK
AK
AJ
AJ
Data Bus Interchange
Bus Data
Hold Data
Data
RA and CA
D R A M
AE
DRAM Timing
(AM)
DRAM Control
RAS 0 – 1
CAS 0 – 3

Figure 7-39. Cached Read Ahead Operation

The microprocessor initiates the cached read ahead by supplying the PE support circuitry with a cycle request code set to 4 and a function code set to 6.

NOTE: When the PE number from the DTB annex equals the virtual or logical PE number of the local PE, the AM option interprets the operation as a cached read ahead operation. When the PE number from the DTB annex does not equal the virtual or logical PE number of the local PE (remote PE number), the AM option interprets a normal cached read.

For a cached read ahead operation, the AM option arbitrates for the DRAM and generates DRAM control signals needed to retrieve a block of data from local memory. The DRAM control is sent to the AE option and the address is sent to the AJ and AK options. The AE, AJ, and AK options address the DRAM, and the AJ and AK options read a block of data from the DRAM. The AM option sends the AJ and AK options the control that instructs the AJ and AK options to steer the data to the microprocessor. The AM option then sends an acknowledgement to the microprocessor, instructing the microprocessor to input the data, perform SECDED, and place the data in the cache.

Immediately after sending the data to the microprocessor, the AM option arbitrates for the DRAM and generates DRAM control to retrieve the next consecutive block of data. Again the DRAM control is sent to the AE option, and the address is sent to the AJ and AK options. The AE, AJ, and AK options address the DRAM, and the AJ and AK options read a block of data out of the DRAM.

The AJ and AK options buffer this data in a local memory read stage until the AM option sends control that instructs the AJ and AK options to steer the data to the microprocessor or to clear out the buffers. The AM option sends the control to steer the data to the microprocessor after the microprocessor issues another load instruction with the same address as the data in the local memory read stage. The AM option signals the AJ and AK options to clear the buffers when the microprocessor issues a memory barrier instruction or a load instruction with an address that does not match the data in the local memory read stage.

Cached Atomic Swap

The CRAY T3D hardware implements the cached atomic swap operation in the same way it does the noncacheable swap operation, with the exception of the memory function code and the read data acknowledgement. For the cached atomic swap operation, the memory function code is set to 5 (write or cached atomic swap). The read data acknowledgement is set to 7 (OK). The AM option sends this acknowledgement to the microprocessor to instruct the microprocessor to input the data, perform SECDED, and place the data in the data cache.

Writes

The microprocessor initiates the write operation when it needs to transfer data from the internal write buffer to system memory. The PE support circuitry interprets a normal write operation when the cycle request code is set to 5 (WRITE_BLOCK) and a function code is set to 0, 1, 4, 5, or 6.

For the write operation, the microprocessor sends the address offset, the cycle request code, the DTB annex index, and the data to the PE support circuitry. The address offset and the cycle request code is sent to the AM option. The DTB annex index is sent to the AR option. And the data is sent to the AJ and AK options. Refer to Figure 7-40.

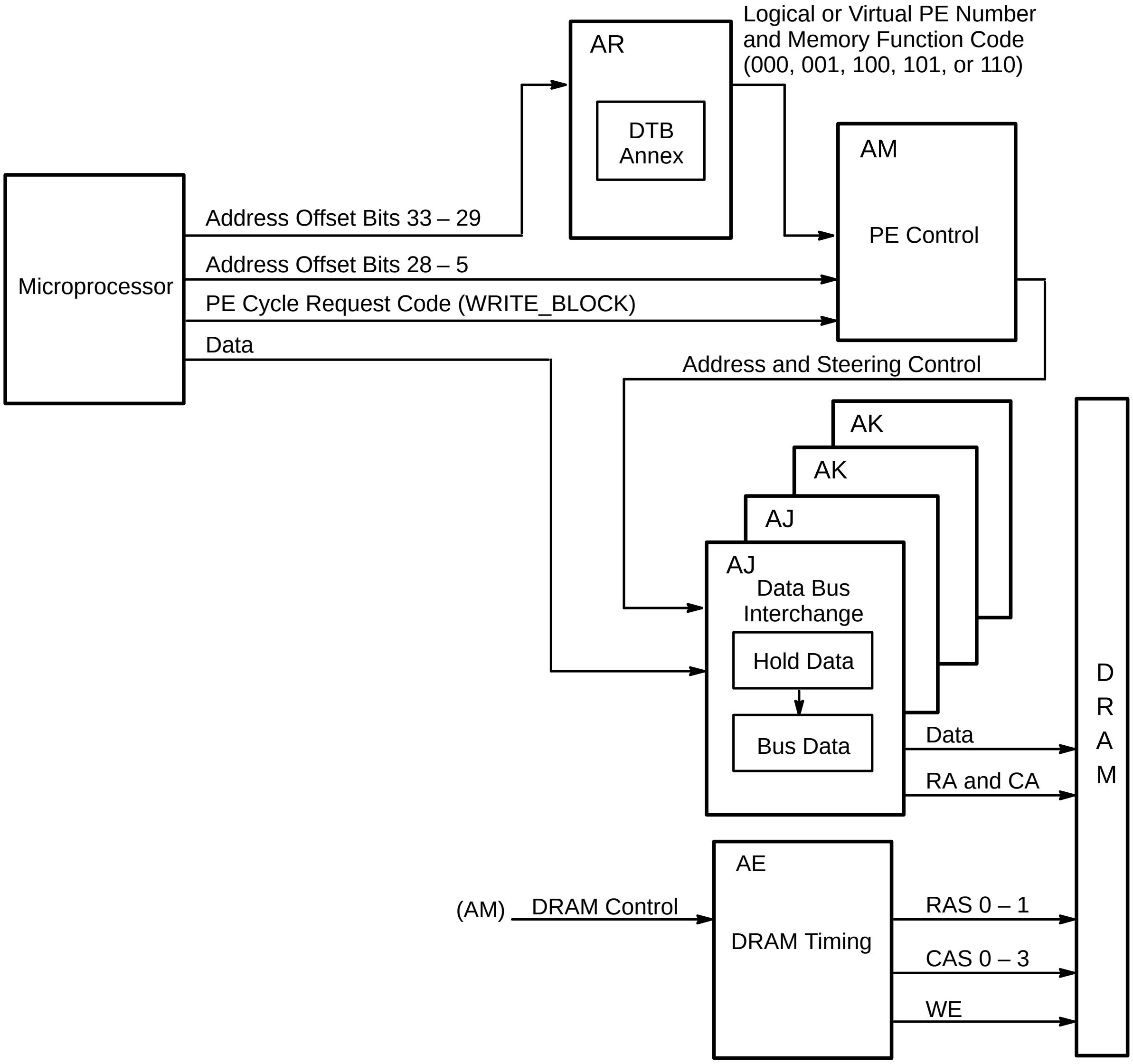

AR
DTB Annex
Logical or Virtual PE Number and Memory Function Code (000, 001, 100, 101, or 110)
AM
PE Control
Microprocessor
Address Offset Bits 33 – 29
Address Offset Bits 28 – 5
PE Cycle Request Code (WRITE_BLOCK)
Data
Address and Steering Control
AK
AK
AJ
AJ
Data Bus Interchange
Hold Data
Bus Data
Data
RA and CA
D R A M
AE
DRAM Timing
(AM)
DRAM Control
RAS 0 – 1
CAS 0 – 3
WE

Figure 7-40. Local Write Operation

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

The AM option determines whether a memory reference is local or remote by comparing the PE number from the AR option to the logical or virtual PE register. When the PE number matches the contents of the logical or virtual PE register, the reference is local. When the PE number does not match the contents of the logical or virtual PE register, the reference is remote.

When the reference is local, the AM option analyzes the address offset to determine whether the reference is for local memory or for a memory-mapped register. When the address offset equals an address between 10000000_{16} and $1FFFFFFF_{16}$ (address bit 28 = 1), the reference is for a memory-mapped register. When the address equals an address between 00000000_{16} and $0FFFFFFF_{16}$ (address bit 28 = 0), the reference is for local memory. For the write operation, the address falls within the range for a local memory reference.

The AM option also checks for memory conflicts. Memory conflicts occur when more than one of the five types of references (refresh reference, PE reference, external read reference, external write reference, or BLT local read reference) request local memory at the same time. When a memory conflict occurs, the AM option arbitrates for local memory. The refresh is always given top priority. To arbitrate between the other references, the AM option uses a 2-bit counter to produce a 2-level round robin arbitration sequence. For more information on the 2-level round robin arbitration sequence, refer again to “AM Option” in Section 3, “Processing Element Node.”

When the write operation has priority to access memory, the AM generates DRAM control commands. These control commands are sent to the AE option over 2 clock periods. The Bank Select, Chip Select, Write Enable, New Page, and Go commands are sent during the first clock period; and the CAS 0 Mask, CAS 1 Mask, CAS 2 Mask, CAS 3 Mask, and Die Select commands are sent during the second clock period.

The AE option uses these DRAM commands to generate the CAS 0 – 3, RAS 0 – 1, and WE signals. The CAS 0 – 3 signals enable data to be written into the appropriate memory chips on the daughter cards. The RAS 0 – 1 signals enable data to be written into one of the two dies that may be present within the memory chips. When there is only one die within the memory chip, the RAS 1 signal is not used. The WE signal enables the memory chips to input the data.

The AM option also supplies the AE option with timing parameter data. The AE option uses the timing parameter data to determine how long to assert the RAS, CAS, and WE signals to the memory chips. The AE option makes this determination by comparing the timing parameter data to the value of a bank timing counter. When the timing parameter data equals the value of the counter (increments by one each clock period), the AE starts asserting the signal and resets the bank timing counter to zero. The AE option also compares the timing parameter data to the bank timing counter to determine when to stop asserting the signals.

The AM option also sends the address offset to the AJ and AK options. The AJ and AK options send this address to the DRAM. After the DRAM is addressed, the AJ and AK options write the data to this location. The AM option signals the microprocessor that the write is complete by setting the cycle acknowledgement to 4 (OK).

When the write operation is remote, the AM option checks the validity of the virtual PE number, creates the header phits, and arbitrates for the network interface request buffers and the outgoing channel. For more information on how the AM option does this, refer to “AM Option” in Section 3, “Processing Element Node.”

When the write request has priority to access the buffer, the AM option sends the header phits to the EA and EB options of the network interface. The command and destination PE number are sent to the EB option and the address is sent to the EA option (refer again to Figure 7-34).

The AM option also sends the External Outgoing Data Path Control signal to the AJ and AK options. This signal enables the AJ and AK options to steer the data to the EA and EB options. Figure 7-41 shows the order of this data transfer.

	AK1				AK0				AJ1				AJ0			
	R603	R602	R601	R600	R603	R602	R601	R600	R603	R602	R601	R600	R603	R602	R601	R600
								Data Bits								
CP5	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								Data Bits								
CP6	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
								Data Bits								
CP7	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
								Data Bits								
CP8	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
							Check Bits									
CP9		13	12	11	10	9	8	7		6	5	4	3	2	1	0

Figure 7-41. Data Channel From the AJ and AK Options to the EA and EB Options

The EA option generates parity for the header phits, assembles this information into a request packet, and sends it to the SR option that handles the X dimension of the interconnect network. From the SR option, the packet is transferred through the network to the destination node.

When the request packet reaches the destination node, the EC and ED options input the request packet. When the entire packet is buffered in the ED option, the EC option checks the packet for destination and network parity errors, and signals the ED option to read the packet from the buffer.

The ED option reads the packet from the buffer, checks the packet for network buffer parity errors, and sends the command, remote address, and source PE number to the AM option in the destination PE.

From the command, the addresses, and the source PE number, the AM option yields DRAM control signals and response header phits. The AM option sends the DRAM control signals to the AE option and the address to the AJ and AK options.

The AJ and AK options, along with the AE option, address the DRAM. The AE option supplies the RAS, CAS, and WE signals to the DRAM. The AJ and AK options supply the internal address. After the DRAM is addressed, the AJ and AK options write the data to this memory location.

The AM option also generates a new command and destination PE number using the command and source PE number phits of the request packet. This information is sent to the EB option. The EB option generates parity for the command and source PE number and sends this information to the EA option.

The EA and EB options assemble this information into a response packet. The EA option sends the response packet to the SR option that handles the X dimension of the interconnect network. From this option, the response packet is transferred through the network, back to the source node.

When the response packet reaches the source node, the EC and ED options input the response packet. As soon as the entire packet is buffered in the ED option, the EC option checks for destination and network parity errors, and signals the ED option to read the packet from the buffer. The ED option checks the packet for network buffer parity errors and sends the header phits of the packet to the AM option.

The AM option interprets the information from the header phits as the response to the write. The AM option signals the microprocessor that the write operation is complete by setting the cycle acknowledgement to a 4 (OK).

Fetch-and-increment

The fetch-and-increment operation transfers 32 bits of data between the microprocessor and a fetch-and-increment register. Immediately following the read of a fetch-and-increment register, the contents of the fetch-and increment register are incremented by one.

The fetch-and-increment registers are located in the EB options. There are two fetch-and-increment registers within each node (one per PE). These fetch-and-increment registers can be read or written by any microprocessor within the system.

Fetch-and-increment Register Read

For the fetch-and-increment read operation, the microprocessor sends the address offset, the cycle request code, and the DTB annex index to the PE support circuitry. The address offset and the cycle request code are sent to the AM option. The DTB annex index is sent to the AR option.

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number (actually an F&I register address) and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

The AM option interprets the cycle request code (4; READ_BLOCK) and the memory function code (2) to determine the type of request. For the fetch-and-increment read operation, the AM option ignores the address offset and uses the PE number from the DTB annex as the fetch-and-increment register address.

NOTE: The fetch-and-increment register address contains two parts: a node number and an F&I register bit. The node number is a logical or virtual node number and indicates the node in which the fetch-and-increment register is located. The F&I bit indicates whether the fetch-and-increment register is register 0 or register 1.

The AM option treats all fetch-and-increment read operations as remote references; therefore, the AM option uses the information from the DTB annex and the microprocessor to build the header phits for a request packet. The AM option also arbitrates for a request buffer in the EA option and the external outgoing channel.

When the fetch-and-increment read operation has access to a request buffer and the outgoing channel, the AM option sends the command and the destination PE number to the EB option and sends the address to the EA option.

The EA and EB options assemble this information into a request packet and send the packet through the interconnect network to the destination node. The EC and ED options in the network interface of the destination node receive the packet. The ED buffers the packet, and the EC option checks it for errors.

Under the control of the EC option, the ED option reads the request packet from the buffer. The ED option checks the packet for network buffer parity errors and sends the request information to PE 0 or PE 1 depending on the F&I bit. When the F&I bit is set to 0, the ED option sends the packet to PE 0. When the F&I bit is set to 1, the ED option sends the packet to PE 1. Within the designated PE, the AM option receives the command, F&I register address, and source PE number. From the command, the address, and the source PE number, the AM option generates response header phits. The AM option sends the response header phits to the EB option.

When the EB option interprets the response command as a fetch-and-increment read, the EB reads the data from fetch-and-increment register 0 or 1, depending upon which PE generated the response (refer to Figure 7-42). When PE 0 generates the fetch-and-increment response, the EB option reads from fetch-and-increment register 0. When PE 1 generates the fetch-and-increment response, the EB option reads from fetch-and-increment register 1. The EB option sends this data to the EA option. The EB option also keeps a copy of the fetch-and-increment data, increments this data by one, and writes the result back into the fetch-and-increment register.

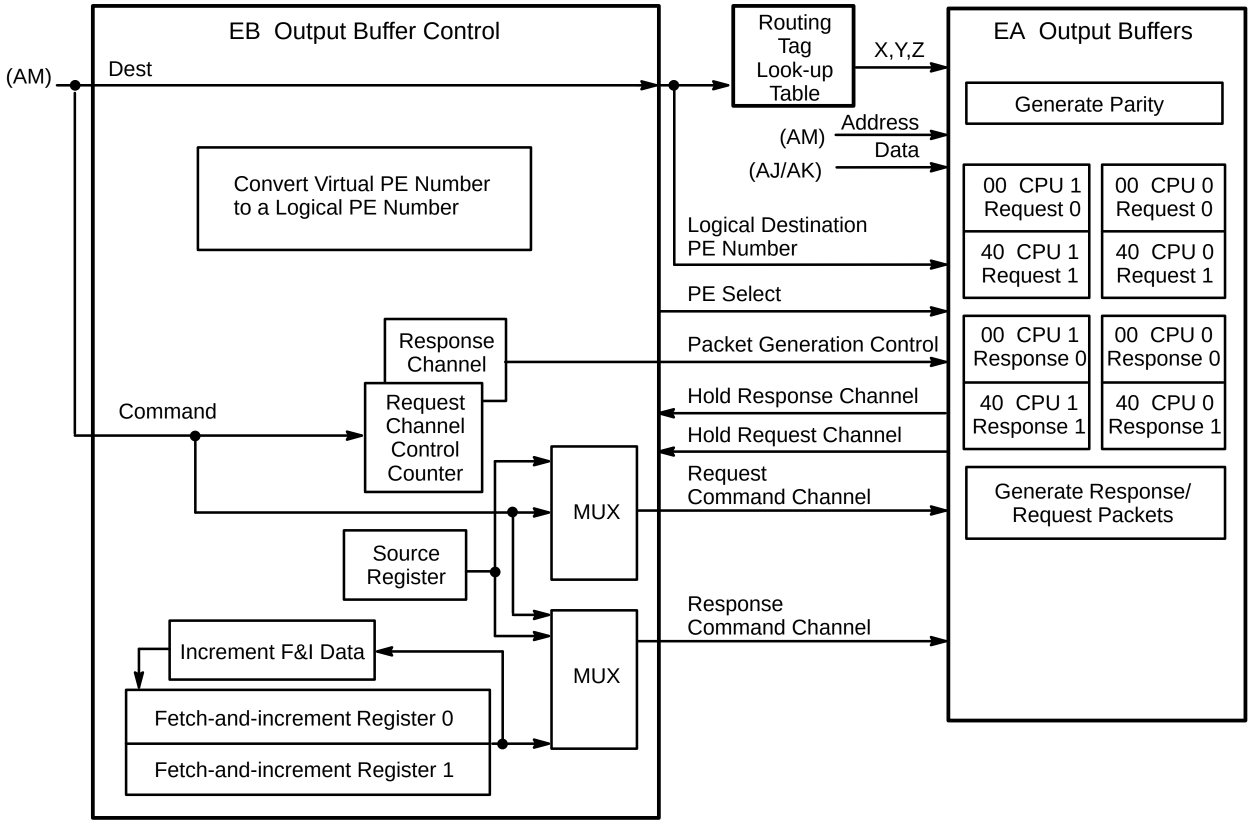


Figure 7-42. Reading From the Fetch-and-increment Register

In addition to the fetch-and-increment data, the EB option sends the command, the destination PE number, and the source PE number to the EA option. The EB option then instructs the EA option to assemble this information into a response packet. The EA option sends the response packet to the SR option that handles the X dimension of the interconnect network. From this option, the response packet is transferred through the network, back to the source node.

When the response packet reaches the source node, the EC and ED options input the response packet. The ED option buffers the entire response packet, except for the routing tag phit.

When the entire packet is buffered in the ED option, the EC option sends the ED option a PE Active signal, which enables the ED option to read the packet from the buffer. The command and source PE number of the packet are sent to the AM option. The data is sent to the AJ and AK options.

The AM option sends the DRAM Read Data Path Control and CPU Read Data Select signals to the AJ and AK options. These signals enable the AJ and AK options to select external data and send it to the microprocessor.

The AM option also sends a read data acknowledge (6; OK_NCHK) to the microprocessor. This acknowledge instructs the microprocessor to input the data and not perform SECDED. Because the value stored in the fetch-and-increment register changes every time the register is read, the value read from the fetch-and-increment register is not protected by SECDED check bits. Instead, software must compare the two values the microprocessor received from a fetch-and-increment response packet.

When the values are the same, the body portion of the packet was not corrupted while traveling through the network, and the fetch-and-increment value is valid. When the values are not the same, the body portion of the packet was corrupted while traveling through the network, and the fetch-and-increment value cannot be used.

Fetch-and-increment Register Write

For the fetch-and-increment write operation, the microprocessor sends the address offset, the cycle request code, and the DTB annex index to the PE support circuitry. The address offset and the cycle request code are sent to the AM option. The DTB annex index is sent to the AR option. The microprocessor also sends the fetch-and-increment data (32 bits) to the AJ and AK options.

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number (actually an F&I register address) and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

The AM option interprets the cycle request code (5; WRITE_BLOCK) and the memory function code (2) to determine the type of request. For the fetch-and-increment read operation, the AM option ignores the address offset and uses the PE number from the DTB annex as the fetch-and-increment register address.

The AM option treats all fetch-and-increment write operations as remote references; therefore, the AM option uses the information from the DTB annex and the microprocessor to build the header phits for a request packet. The AM option also arbitrates for a request buffer in the EA option and the external outgoing channel.

Once the fetch-and-increment write operation has access to a request buffer and the outgoing channel, the AM option sends the command and the destination PE number to the EB option. The AM option also instructs the AJ and AK options to send the fetch-and-increment data to the EA and EB options.

The EB option inputs the command and destination PE number and generates the packet generation control. The EB option sends the packet generation control to the EA option, along with the command, the destination PE number, and the source PE number.

The EA option uses the packet generation control to read the data from the buffer, assemble the request packet, and send the request packet to the SR option that handles the X dimension of the interconnect network. From the SR option, the response packet is transferred through the network to the destination node.

After arriving at the destination node, the ED option buffers the incoming packet. When the entire packet is buffered in the ED option, the EC option checks the packet for destination and network parity errors and signals the ED option to read the packet from the buffer. The ED option checks the packet for network buffer parity errors and sends the packet to PE 0 or PE 1, depending on the F&I bit. When the F&I bit is set to 0, the ED option sends the packet to PE 0. When the F&I bit is set to 1, the ED option sends the packet to PE 1. Within the designated PE, the AM option receives the command, destination PE number, and source PE number. The AJ and AK options receive the data.

The AM option generates a new command and destination PE number using the command and source PE number phits of the request packet. This information is sent to the EB option. The AM option also instructs the AJ and AK options to send the data to the EA and EB options.

When the EB option interprets the response command as a fetch-and-increment write, the EB writes the data from the AJ and AK options to fetch-and-increment register 0 or 1, depending upon which PE generated the response (refer to Figure 7-43). When PE 0 generates the fetch-and-increment response, the EB option writes to fetch-and-increment register 0. When PE 1 generates the fetch-and-increment response, the EB option writes to fetch-and-increment register 1.

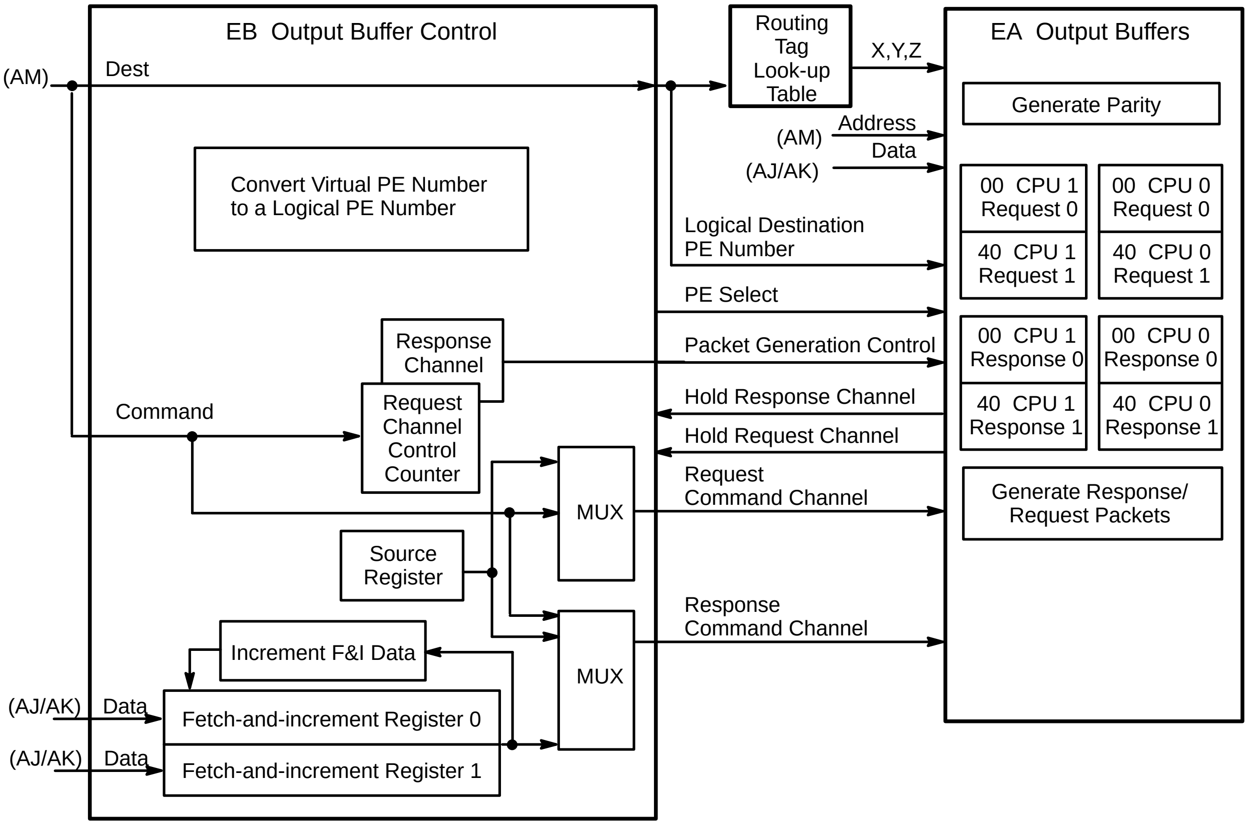


Figure 7-43. Writing to the Fetch-and-increment Register

After the fetch-and-increment write is complete, the AM option generates a second set of response header phits. The new command header phit reflects the response or acknowledgement of the fetch-and-increment write. This information is sent to the EB option.

The EB option generates parity for the response command and source PE number and sends this information to the EA option. The EB option instructs the EA option to assemble this information into a response packet. The EA option sends the response packet to the SR option that handles the X dimension of the interconnect network. From this option, the response packet is transferred through the network, back to the source node.

When the response packet reaches the source node, the EC and ED options input the response packet. The ED option buffers the entire response packet except for the routing tag phit. As soon as the entire packet is buffered in the ED option, the EC option signals the ED option to read the packet from the buffer. The command and source PE number of the packet are sent to the AM option.

Using this information, the AM option determines that this is a response to the fetch-and-increment write. The AM option sends a cycle acknowledge (4; OK) to the microprocessor to inform the microprocessor that the fetch-and-increment write is complete.

Messaging Facility

The messaging facility transfers information from one PE to another PE. After the message reaches the destination PE, the PE support circuitry writes the message into the message queue and interrupts the microprocessor. The microprocessor may then read the message from the message queue.

Message Queue and Message Queue Pointers

The message queue is an area of local memory that is designated for information that arrives at the PE in the form of a message. The reading and writing of the message queue are controlled by a tail pointer, a head pointer, and a limit counter. The message queue tail pointer register contains the tail pointer. This register is located in the AM option. The tail pointer points to the next available location in the message queue. The head pointer is a software pointer that points to the next message in the message queue to be read by the microprocessor. The limit counter is a 13-bit counter that indicates how many slots within the message queue contain invalid data. The AM option increments this counter each time the microprocessor writes to the message queue limit increment register. This register is also located in the AM option. For more information on the message queue pointers and counter, refer again to “Message Queue” earlier in this section.

Writing a Message

When writing a message, the microprocessor sends the address offset, the cycle request code, and the DTB annex index to the PE support circuitry. The address offset and the cycle request code are sent to the AM option. The DTB annex index is sent to the AR option. The microprocessor also sends the data for the message to the AJ and AK options.

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

The AM option interprets the cycle request code (5; WRITE_BLOCK) and the memory function code (7) to determine the type of request.

The AM option treats all messages as remote references; therefore, the AM option uses the information from the DTB annex and the microprocessor to build the header phits for a request packet. The AM option also arbitrates for a request buffer in the EA option and the external outgoing channel.

Once the message has access to a request buffer and the outgoing channel, the AM option sends the command and the destination PE number to the EB option. The AM option also instructs the AJ and AK options to send the data to the EA and EB options.

The EB option inputs the command and destination PE number and generates the packet generation control. The EB option sends the packet generation control to the EA option along with the command, the destination PE number, and the source PE number.

The EA option uses the packet generation control to read the data from the buffer, assemble the message, and send the message to the SR option that handles the X dimension of the interconnect network. From the SR option, the message is transferred through the network to the destination node.

After arriving at the destination node, the ED option buffers the incoming packets. When the entire packet is buffered in the ED option, the EC option signals the ED option to read the packet from the buffer. The command phit, upper address phit, lower address phit, and source PE number phit of the message are sent to the AM option. The data (4 words even when the data is not valid) is sent to the AJ and AK options.

The ED also sends the header phits of the message packet to the AJ and AK options. After the ED option has sent the command, address, and source PE number phits to the AM option and the data to the AJ and AK options, the EC option signals the ED option to reset the read address counter to 0. This, along with the Read Active signal, enables the ED option to read the destination PE number phit, command phit, upper address phit, lower address phit, and source PE number phit from the buffer and send it to the AJ and AK options.

NOTE: The AJ and AK options triplicate the header phits before writing this information into the message queue (refer again to Figure 7-16 through Figure 7-18). This triplication enables the microprocessor to determine the validity of the header phits. The AJ and AK options also triplicate the data when the message packet contains only 1 word of data.

From the command, the AM option determines that the packet is a message. The AM option uses the contents of the message queue tail pointer register to calculate the address of the message queue where this message will be stored. The AM option multiplies the contents of the tail pointer by 64 and adds this result to the message queue base address.

The AM option also arbitrates for the DRAM and generates DRAM control. This control is sent to the AE option, and the address (from the tail pointer) is sent to the AJ and AK options.

The AJ and AK options, along with the AE option, address the DRAM. The AE option supplies the RAS and CAS signals to the DRAM. The AJ and AK options supply the internal address. After the DRAM is addressed, the AJ and AK options write the entire message into the message queue area of memory.

NOTE: After the packet is written into the message queue, the AM option signals the AR option to set the message interrupt.

The AM option generates a new command and destination PE number using the command and source PE number phits of the request packet. This information is sent to the EB option. The EB option generates parity for this information and sends it to the EA option. The EA option assembles this information into a message acknowledge packet and sends the packet through the interconnect network to the source node.

After the AM option of the source PE receives the message response, the AM option sends the cycle acknowledgement (4; OK) to the microprocessor, signalling the message write was successful.

Reading a Message

The microprocessor reads a message out of the message queue by initiating a normal cached read. The address offset supplied to the PE support circuitry during this read comes from the head pointer. After reading the message out of the message queue, the microprocessor increments both the head pointer and the limit pointer.

Data Prefetch

The data prefetch operation is used to hide the latency of read operations. To invoke the prefetch operation, the microprocessor is programmed to issue a FETCH or FETCHM instruction several instructions ahead of when the data is actually needed. The PE support circuitry fetches this data and places it into a buffer called the prefetch queue. When the microprocessor needs the data, the microprocessor reads the data from the prefetch queue, which is faster than reading data from local or remote memory.

NOTE: When a FETCH instruction is outstanding and the PE support circuitry receives a write request for the same memory location as for the FETCH instruction, the AM option stalls the write request until the FETCH instruction is complete. This protects the prefetch data from being overwritten by new data. The AM option will not stall a write request when there is an outstanding FETCHM instruction.

The data prefetch queue is located in the AR option (refer to Figure 7-44). This is a circular queue that can store up to sixteen 64-bit words. For more information on the prefetch queue, refer again to “AR Option” in Section 3, “Processing Element Node.”

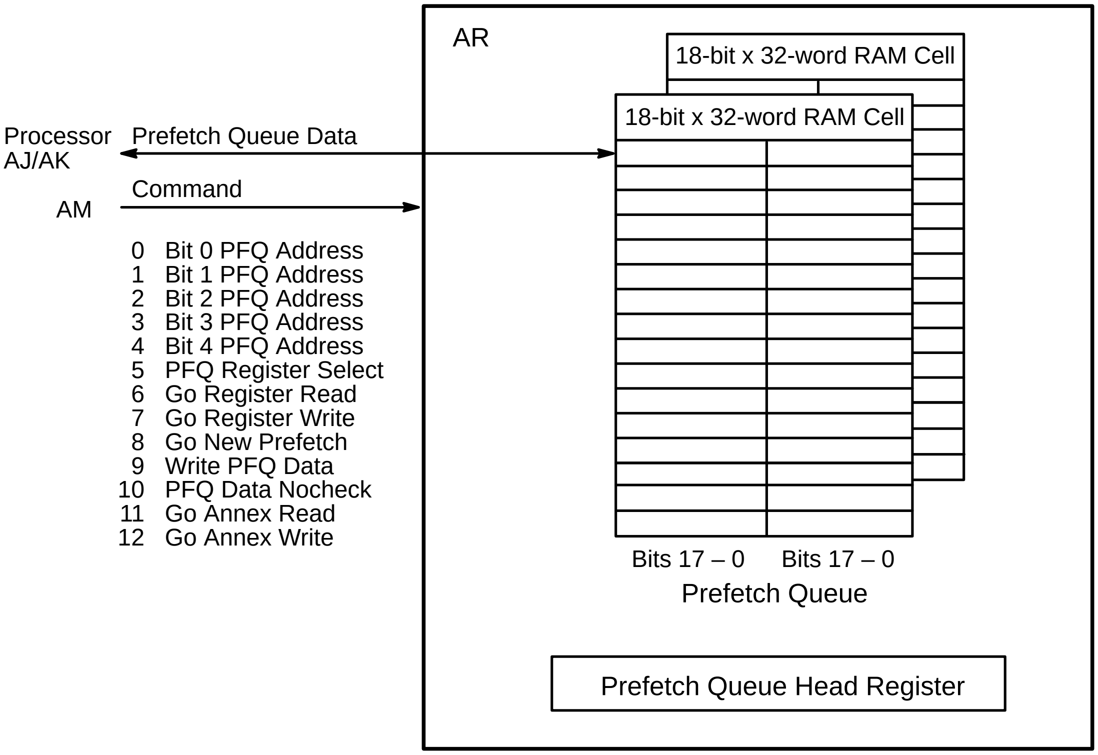


Figure 7-44. Prefetch Queue

Prefetch Operations

For a prefetch operation, the microprocessor sends the address offset, the cycle request code, and the DTB annex index to the PE support circuitry. The address offset and the cycle request code are sent to the AM option. The DTB annex index is sent to the AR option.

The AR option uses the DTB annex index to address the DTB annex. From this address, the AR option retrieves the PE number and memory function code needed for this request. The AR option sends the memory function code and the PE number to the AM option.

Normally, the AM option determines whether a memory reference is local or remote by comparing the PE number from the AR option to the logical or virtual PE register. For a prefetch operation, the AM option always interprets the PE number from the DTB annex as a remote PE.

Next, the AM option checks the validity of the virtual PE number, creates the header phits, and arbitrates for the network interface request buffers. This is done in the same way as for other remote references, with the exception of generating the command phit.

The AM option generates the command phit by using the PE cycle request code and the memory function code. For a prefetch operation, the PE cycle request code is set to 2 or 3. The memory function code indicates which prefetch operation is requested. When the function code is set to 0, 4, 6, or 7, the request is a normal prefetch memory read. When the function code is set to 2, the request is a prefetch fetch-and-increment register read. When the function code is 1 or 5, the request is a prefetch atomic swap.

For a prefetch queue operation, the command also contains the address where the data will be written into the prefetch queue. The AM option generates this address using a prefetch queue write address counter (also referred to as the reserve pointer). The value of the prefetch write address counter becomes bits 0 through 4 of the request command. After using the value of the counter, the AM option increments the counter by one.

The AM option sends the header phits to the EA and EB options where this information is assembled into a packet. This packet is transferred through the network to the destination node.

The EC and ED options of the destination node input the request packet. The EC option determines which PE gets the packet, checks the packet for errors, and instructs the ED option to send the header phits to the AM option in the destination PE.

The AM option uses this information to set up for the memory read and to build the response header phits. The AM option instructs the AE, AJ, and AK options to address the appropriate location in memory and read the data from this location. The AM option also generates a response command from the request command and generates the destination PE number from the source PE number of the request.

This response information, along with the read data, is sent to the EA and EB options where the data is assembled into a response packet. The response packet is then sent back to the source node through the interconnect network.

The EC and ED options of the source node input the response packet. The EC option determines which PE will receive the packet, checks the packet for errors, and instructs the ED option to send the header phits to the AM option and send the data to the AJ and AK options.

The AM option decodes the response command to determine the type of response. For a prefetch response, the AM option generates control that is sent to the AJ and AK options. This control instructs the AJ and AK options to steer the data to the AR option. (The data is sent to the AR option using the same path used to send data to the microprocessor.)

The AM option also generates a command that is sent to the AR option. This command instructs the AR option to capture the data and place it in the prefetch queue. This command consists of 13 bits, as shown in Figure 7-45.

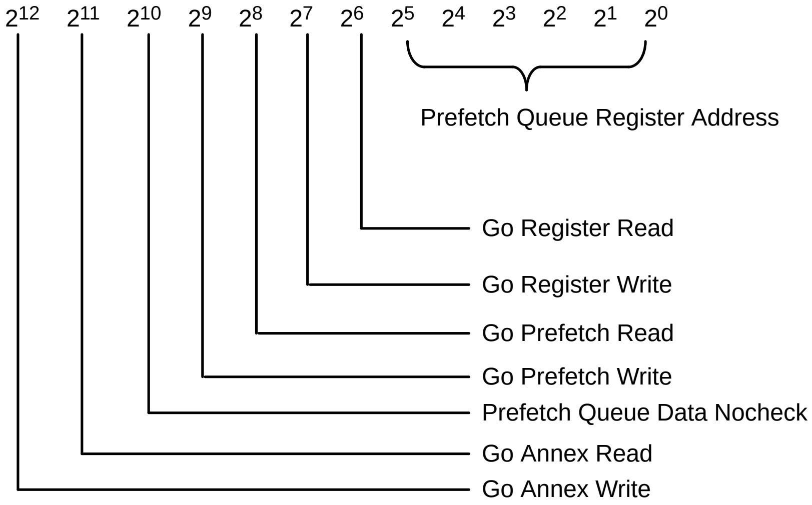


Figure 7-45. Command from AM Option

For a prefetch write operation, the AR option uses bits 0 through 5 and bits 9 and 10 of this command. Bits 0 through 5 of this command indicate the prefetch queue address. Bit 9 indicates the prefetch queue write, and bit 10 indicates whether the data is protected by SECDED.

NOTE: Data from the fetch-and-increment register is not protected by SECDED.

Reading Data from the Prefetch Queue

The microprocessor reads data out of the prefetch queue by initiating a read of the memory-mapped register called the prefetch queue head register. The AM option interprets this register read request and generates a command that is sent to the AR option. This command consists of 13 bits (refer again to Figure 7-45).

To read from the prefetch queue, the AM option uses the value from a prefetch read address counter for bits 0 through 5 of the command. The AR option uses this address to read the data out of the prefetch queue.

After the AR option reads the data out of the prefetch queue, the AR option holds the data and sends the valid bit and the no-check bit to the AM option. The AM option verifies that the data is valid and sets bit 8 of the 13-bit command to 1. This enables the AR option to select the prefetch queue data and send it to the microprocessor. The AM option also determines whether or not the microprocessor should perform SECDED by analyzing the no-check bit. When this bit is set to 1, the AM option instructs the microprocessor not to perform SECDED by sending the microprocessor a read acknowledgement set to 6 (OK_NCHK). When this bit is set to 0, the AM option instructs the microprocessor to perform SECDED by sending the microprocessor a read acknowledgement set to 7 (OK).