3 Processing Element Nodes

Processing element nodes perform all program instructions and store system data. After describing processing element (PE) numbering, this section describes physical nodes, and the components of a processing element node.

PE Numbering

Depending on the context, a PE is identified by one of three types of numbers: a physical number, a logical number, or a virtual number. All three types of numbers consist of a PE bit, which identifies if the PE is PE 0 or PE 1 in a node, 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 where the PE is physically located in a system. 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 indicate where the PE is located in the cabinet.

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 where in the logical system of nodes a PE is located.

The logical nodes form a three-dimensional matrix of nodes. For example, Figure 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, if 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. Information is then rewritten into the routing tag look-up table of each node.

The routing tag look-up table contains information each node uses to create the routing tag in the header of a packet. Because the logical node number may correspond to any of the physical nodes, hardware in the nodes cannot use the logical node number to route data from one node to another.

Each PE node in the CRAY T3D system uses a look-up table to obtain the routing tag. Circuitry in the node enters the logical node number into the routing tag look-up table. The routing tag look-up table then provides the routing tag for a packet (refer to Figure 1). The routing tag steers the packet from the physical source node to the physical destination node.

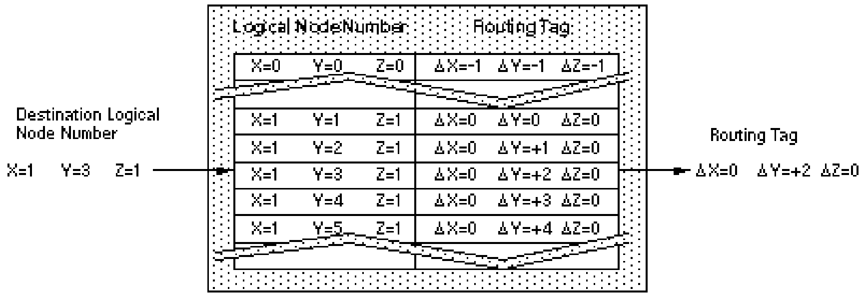

Figure 1. Routing Tag Look-up Table for Logical Node X=1, Y=1, Z=1<

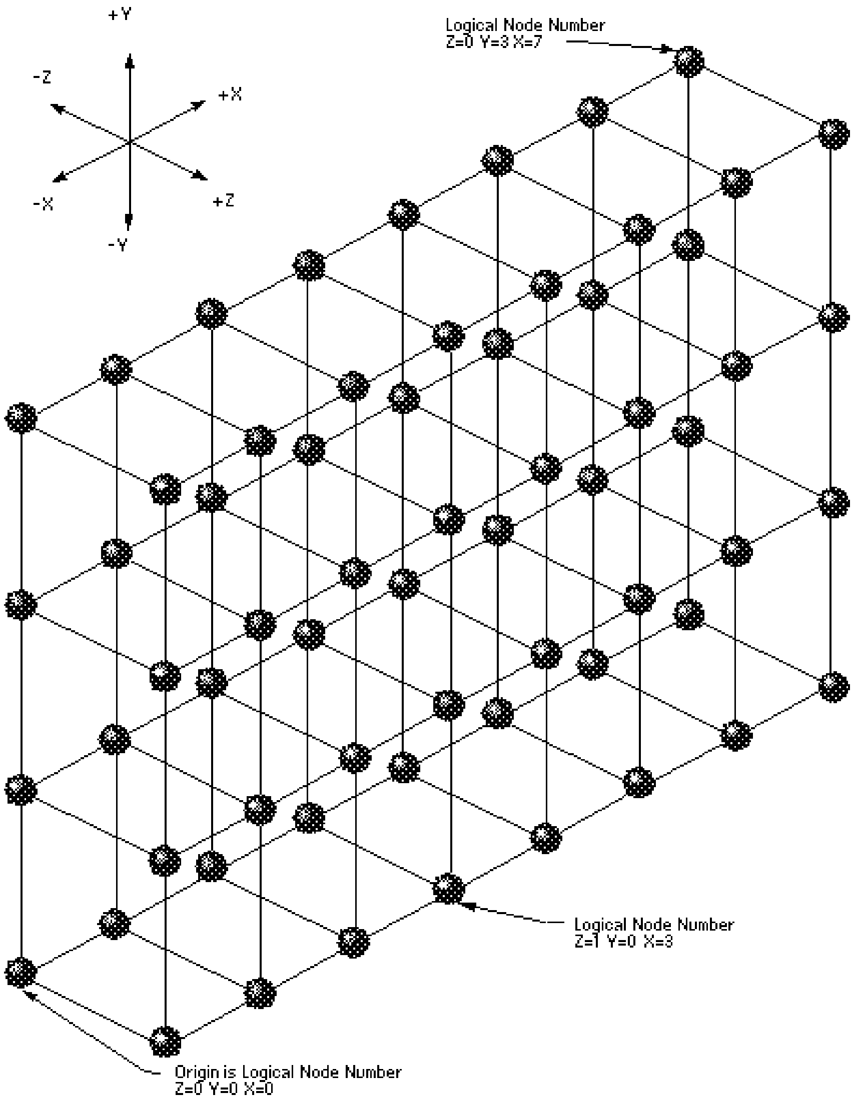

Figure 2. Logical Node Numbers

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 barrier synchronization is provided later in this section.) 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. 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 if the PE is PE 0 or PE 1 in the node.

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 one of 10 shapes.

Table 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).

Eight-node Array Shape 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

Table 1. Eight-node Partition Shapes

Figure 3 shows three of the eight-node partition shapes in a 128-PE CRAY T3D system.

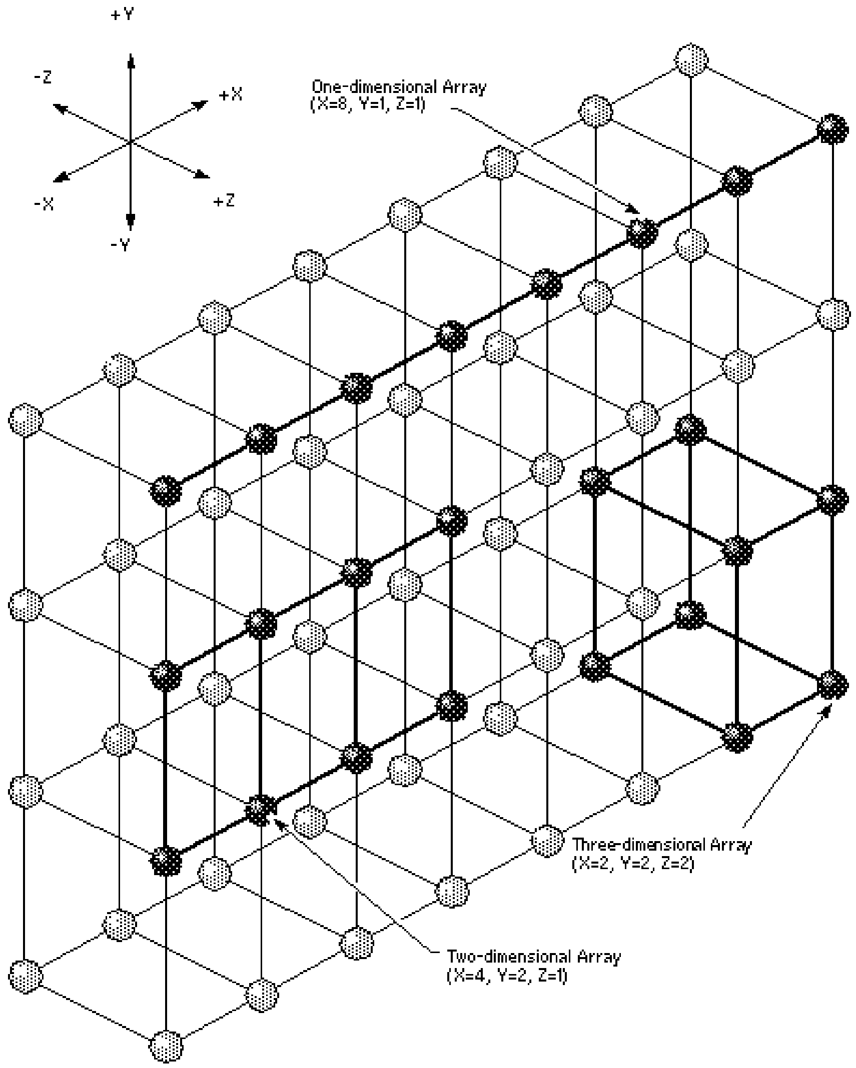

Figure 3. Three 8-node Partition Shapes in a 128-PE CRAY T3D System

As an example of virtual PE numbers, Figure 4 shows a two-dimensional, eight-node partition that contains 8 nodes. Each node in the partition is referred to by the three-bit virtual node numbers shown in Figure 4.

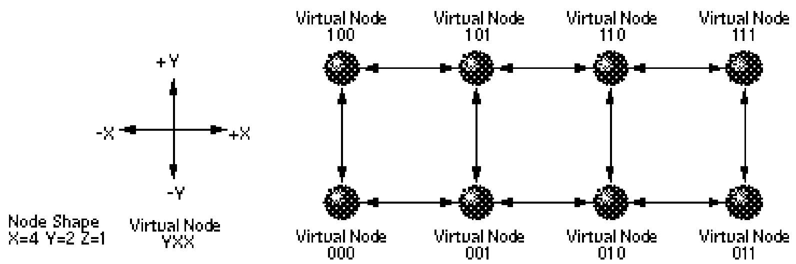

Figure 4. Virtual Node Numbers of a Two-dimensional Array

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 5 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 5 shows how virtual node Y=1 X=2 from Figure 4 may correspond to either logical node number Z=1 Y=2 X=2or logical node number Z=1 Y=3 X=6.

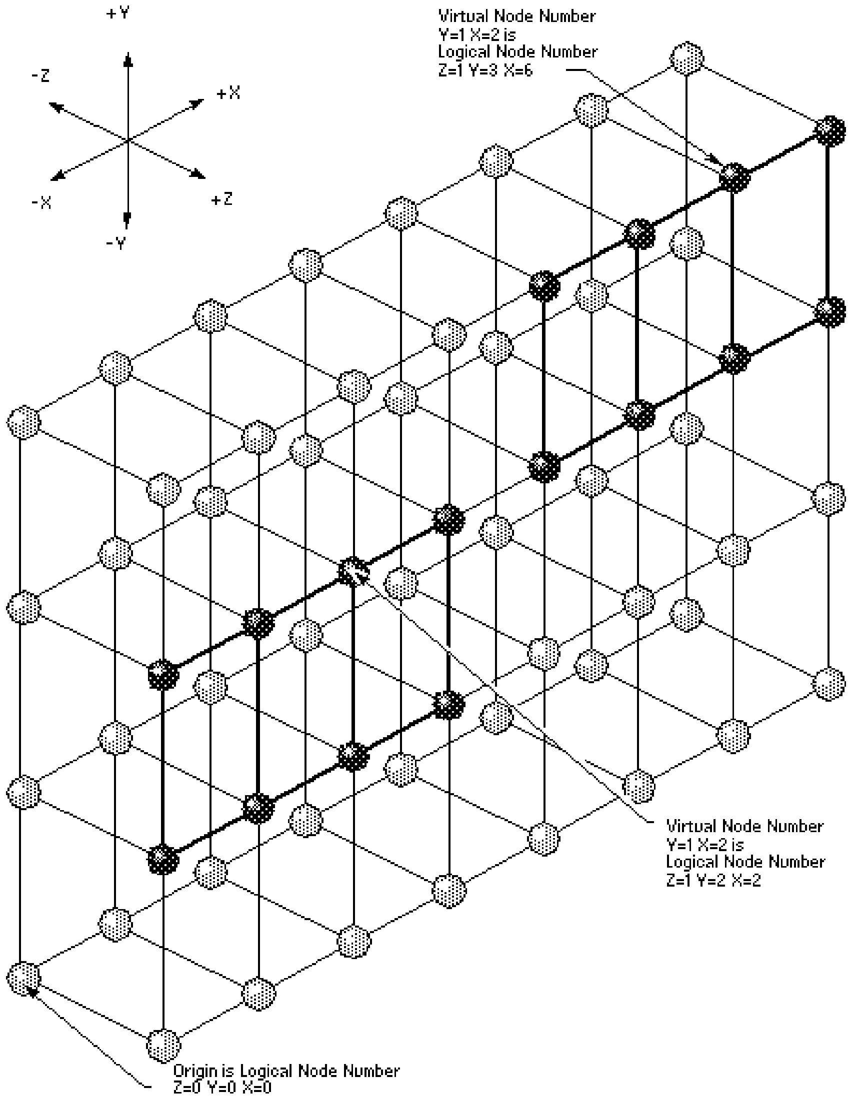

Figure 5. Virtual and Logical Node Numbers

Physical Nodes

Physically, each processing element node resides on half of a circuit board in the CRAY T3D system cabinet (refer to Figure 6). The integrated circuits above the dashed line are used for the components of one processing element node. The integrated circuits below the dashed line are used for the components of another processing element node.

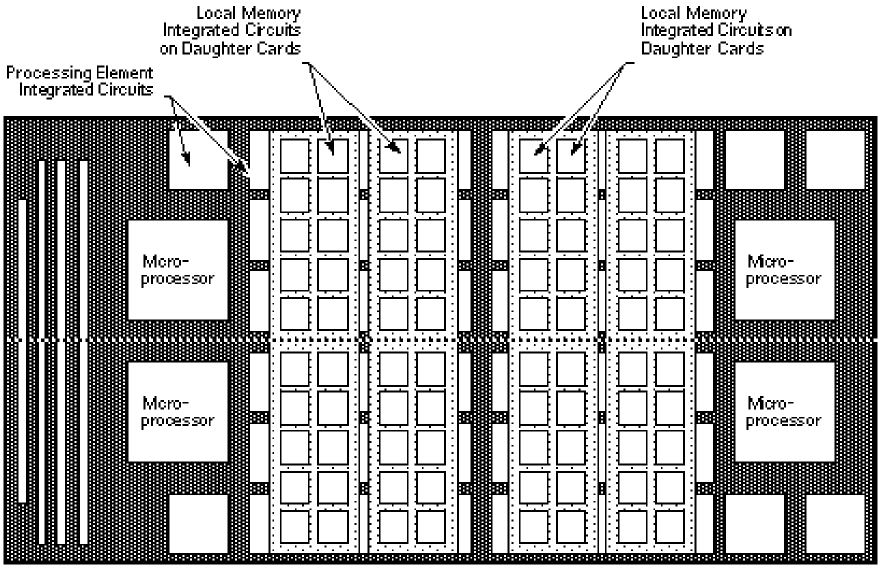

Figure 6. Processing Element Node Circuit Board

Components

A processing element node is composed of four components: two PEs, a BLT, and a network interface (refer to Figure 7).

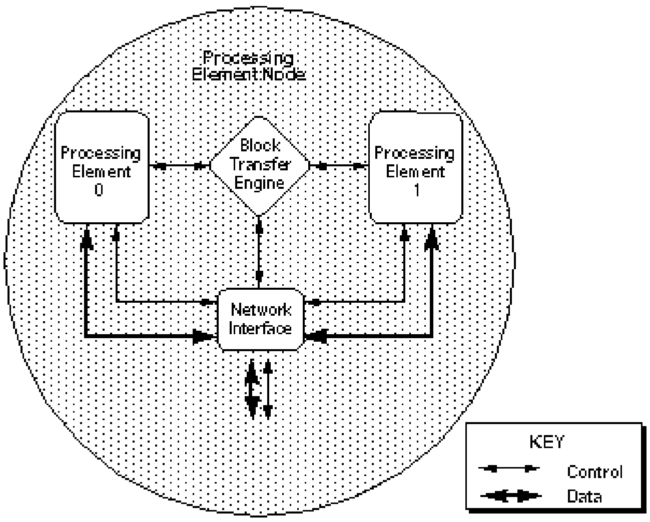

Figure 7. Processing Element Node

Figure 8 is a functional block diagram of the components in a processing element node. The following subsections describe these components.

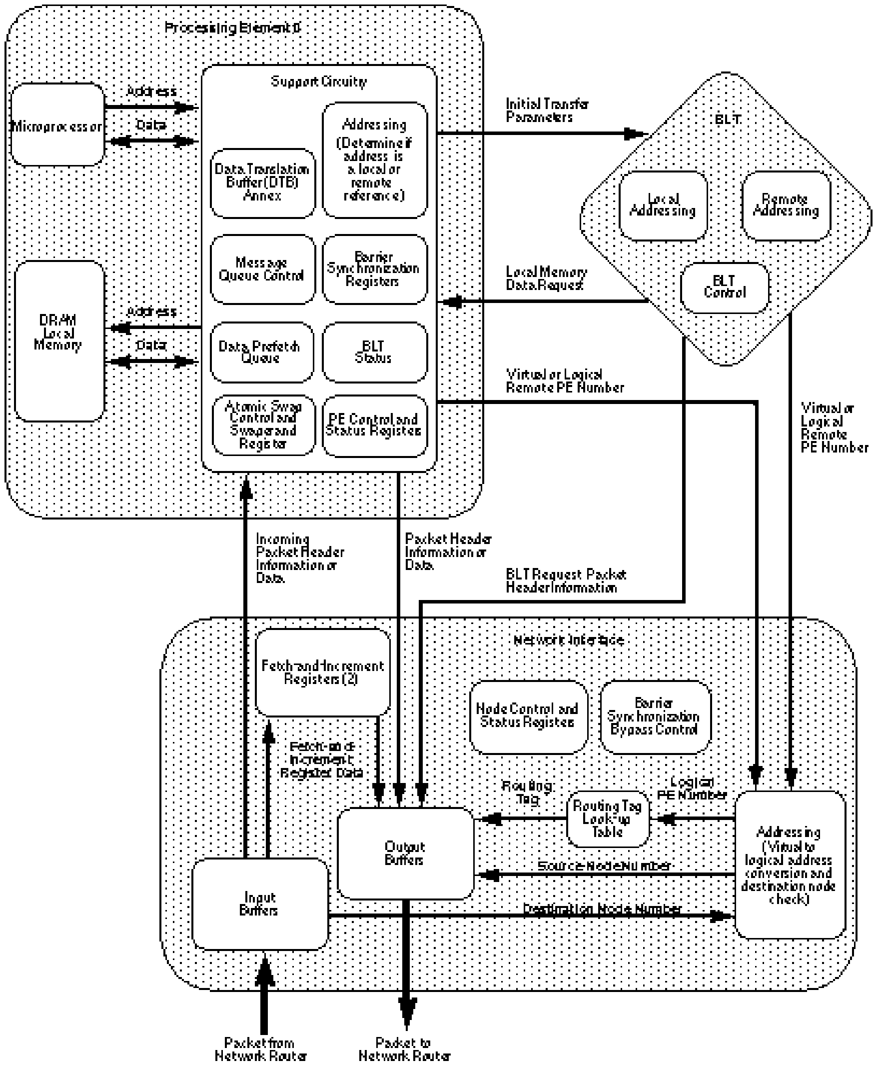

Figure 8. Processing Element Node Functional Block Diagram

Processing Elements

Each processing element node contains two PEs: PE 0 and PE 1. Each PE contains a microprocessor, local memory, and support circuitry.

Microprocessor

The microprocessor is a reduced instruction set computer (RISC) 64-bit microprocessor developed by Digital Equipment Corporation. The microprocessor comprises a central control unit, an integer execution unit, a floating-point execution unit, an address generation and bus interface unit, data cache memory, and an instruction cache memory (refer to Figure 9). The following paragraphs describe each of these components.

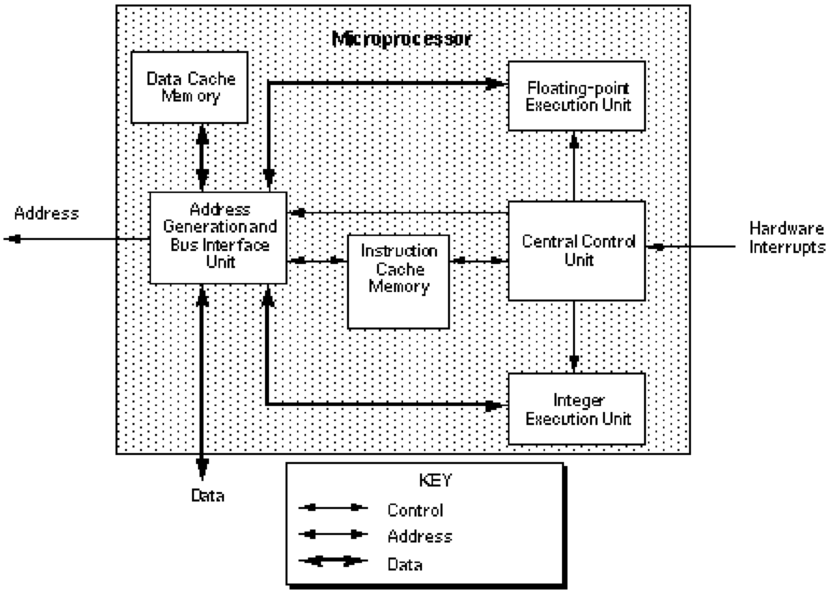

Figure 9. Microprocessor

The central control unit issues instructions to the integer execution unit, floating-point execution unit, and the address generation and bus interface unit. The central control unit also receives hardware interrupt signals from external PE circuitry. These interrupts include barrier synchronization interrupts, messaging interrupts, BLT interrupts, and error interrupts.

The integer execution unit performs integer operations on 64-bit integer registers. Integer operations include arithmetic, compare, logical, and shift operations. There are a total of 32 integer registers.

The floating-point execution unit performs floating-point operations on 64-bit floating-point registers. These operations include IEEE arithmetic instructions, plus instructions for performing conversions between floating-point and integer quantities. There are 32 floating-point registers.

The address generation and bus interface unit generates memory addresses and steers data and control information. The address generation circuitry converts the virtual address it receives from the compiler into address information used by the support circuitry in the PE. The bus interface circuitry responds to read or write instructions for the data cache, instruction cache, and data bus.

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 (refer to Figure 10).

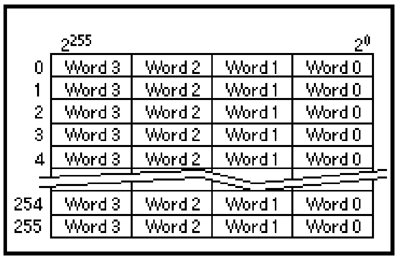

Figure 10. Data Cache and Instruction Cache Memory Organization

The instruction cache memory operates similarly to the data cache memory, but stores instructions. The instruction cache memory is also internal to the microprocessor and stores 256 32-byte lines of data.

The microprocessor also performs error detection and correction. After receiving 128 data bits and 28 check bits from the PE circuitry, the microprocessor generates a new set of check bits. If the new set of check bits is not identical to the original set of check bits, 1 or more of the system data or original check bits changed value during the data transfer.

If only 1 bit changed value, hardware in the microprocessor corrects the value of the incorrect bit. If more than 1 bit changed value, the microprocessor is interrupted.

Each microprocessor is a 431-pin pin grid array (PGA) integrated circuit (refer again to Figure 6). Table 2 lists the specifications for the microprocessor.

Characteristic	Specification
Number of microprocessors per processing element node	2 (one in each PE)
Microprocessor type	Reduced instruction set computer (RISC) 64-bit microprocessor
Bi-directional data bus	128 data bits plus 28 check bits
Data error protection	Single error correction/double error detection (SECDED)
Address bus	34 bits of address information
Clock speed	6.67 ns
Issue rate	2 instructions per clock period maximum
Internal data cache memory size	8 Kbytes (256 32-byte lines)
Internal instruction cache memory size	8 Kbytes (256 32-byte lines)
Latency of data cache to CPU transfers	3 clock periods
Bandwidth of data cache to CPU transfers	64 bits per clock period
Floating-point instruction unit	Supports the Institute of Electrical and Electronic Engineers (IEEE) floating point arithmetic instructions plus instructions for converting between floating-point and integer quantities
Total number of floating-point registers	32
Floating-point register size	64 bits
Integer instruction unit	Integer arithmetic, compare, logical, and shift instructions
Total number of integer registers	32
Integer register size	64 bits
Integrated circuit type	Complimentary metal-oxide semiconductor (CMOS)
Integrated circuit size	14.1 mm X 16.8 mm
Total number of pins	431
Number of signal pins	291
Typical power dissipation	23 W

Table 2. Microprocessor Specifications

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 the microprocessor in that PE.

Figure 11 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 11). The size of local memory depends on the type of DRAM integrated circuits used in the system.

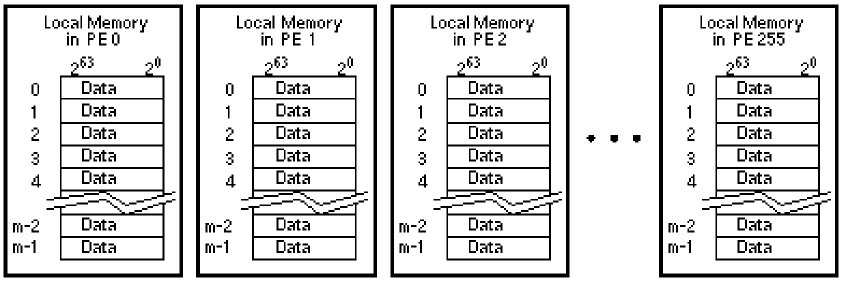

Figure 11. Physical Distribution of Memory

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 (8 Gbytes).

Each microprocessor uses memory addressing that references any word in shared memory. This address, the virtual address, is initially generated by the program compiler. The virtual address is converted into a logical node number, PE number, and address offset by the microprocessor and other components in the processing element node.

Local memory comprises DRAM integrated circuits that are mounted on daughter-card printed circuit boards. The DRAM daughter cards plug into the PE circuit board and reside on top of the integrated circuits used in the PEs (refer again to Figure 6 and to Figure 12)

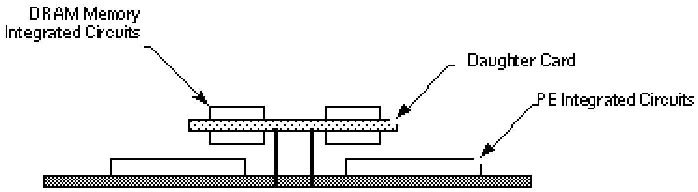

Figure 12. Side View of PE Circuit Board

Table 3 lists the specifications for local memory.

Characteristic	Specification
Local memory size per PE	2 Mwords (4 Mbit DRAMs) or 8 Mwords (16 Mbit DRAMs)
Total memory per processing element node	4 Mwords (4 Mbit DRAMs) or 16 Mwords (16 Mbit DRAMs)
Total logically shared memory in a CRAY T3D system	Local memory size per PE multiplied by the total number of PEs in the system

Table 3. Local Memory Specifications

Support Circuitry

The 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

Address Interpretation

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

The support circuitry uses part of the address generated by the microprocessor as an index into a 32-entry table called the DTB annex. Each entry in the DTB annex contains a virtual or logical PE number and a function code. The PE number is the number of the destination PE. The function code indicates what type of memory function the support circuitry will perform.

The support circuitry compares the PE number received from the DTB annex with the number of the PE that contains the support circuitry. If they match, the microprocessor is addressing local memory. If they do not match, the microprocessor is addressing memory in another PE.

When the microprocessor requests a data transfer with local memory, the support circuitry uses the address from the microprocessor as an address offset for data in local memory. The support circuitry then transfers data between the microprocessor and local memory.

When the microprocessor requests a data transfer with remote memory, the support circuitry sends the remote PE number along with the address offset and control information to the network interface for use in the header of a request packet.

When the microprocessor addresses a register in the support circuitry, the support circuitry routes the information to the appropriate register. In some cases, the support circuitry also performs a function related to the register.

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.

After receiving address and cycle request information from the microprocessor, the support circuitry retrieves a function code and PE number from an entry in the DTB annex. The value of the function code determines which read operation the support circuitry performs. There are two main types of read operations: noncacheable reads and cached reads.

During noncacheable reads, the microprocessor does not place a copy of the read data in the data cache. During cached reads, the microprocessor does place a copy of the read data in the data cache. Cached reads may be used to reduce the latency of subsequent read operations to specified memory addresses. If the data in the data cache is valid, the microprocessor reads data from the data cache instead of from system memory.

During noncacheable memory read operations, as the support circuitry transfers the read data to the microprocessor, the support circuitry signals the microprocessor not to update a line in the data cache. There are two types of noncacheable read operations: normal noncacheable read and noncacheable atomic swap read.

Normal noncacheable read operations transfer data from memory to a register in the microprocessor without updating the data cache. Noncacheable atomic swap read operations transfer a 64-bit word from memory to the microprocessor and then transfer another 64-bit word into the same memory location in an indivisible operation.

Before initiating an atomic swap operation, the microprocessor loads a 64-bit word into a register called the swaperand register. During the atomic swap operation, the support circuitry transfers a word from memory to the microprocessor, then transfers the word from the swaperand register to the same memory location.

During cached read operations, as the support circuitry transfers the read data to the microprocessor, the support circuitry signals the microprocessor to update a line in the data cache. There are three types of cached read operations: a normal cached read, a cached atomic swap read, and a cached read ahead.

Normal cached reads and cached atomic swap reads operate the same as a noncacheable normal read and noncacheable atomic swap read except, when transferring data to the microprocessor, the support circuitry signals the microprocessor to update a line in the data cache.

Cached read aheads are used to hide the latency of local memory reads. The following paragraphs describe a cached read ahead.

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 four-word 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 for 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 a memory-barrier instruction is issued or a data or instruction read operation from a different local memory address occurs.

Writes

Write operations transfer data from the microprocessor to system memory. To initiate a write operation, the microprocessor provides the support circuitry with an address and with cycle request information. The microprocessor may then continue issuing program instructions while the support circuitry completes the write operation.

After receiving the address and cycle request information from the microprocessor, the support circuitry retrieves a function code and PE number from an entry in the DTB annex. The support circuitry then checks the value of the PE number read from the DTB annex. If the PE number is set to the local PE, the support circuitry writes up to four 64-bit words into local memory. If the PE number is set to a remote PE, the support circuitry creates a write request packet that contains up to 4 words of data and sends the request packet to the remote PE.

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. 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.

Data Prefetch

When requested by the microprocessor, the support circuitry performs a data prefetch operation. A data prefetch operation transfers one 64-bit word of data from memory in a remote PE to the data prefetch queue, which is located in the local PE support circuitry.

The microprocessor initiates a data prefetch operation when it encounters a prefetch instruction in a program. A programmer may place the prefetch instruction several instructions before the instruction that actually uses the prefetch data.

When issuing the prefetch instruction, the microprocessor signals the support circuitry that the next data transfer is a prefetch operation. After the microprocessor issues the prefetch instruction, the microprocessor continues with other program instructions.

The support circuitry assembles information for a prefetch read request packet and sends the information to the remote PE over the interconnect network. After receiving the request packet, the support circuitry in the destination PE creates a prefetch read response packet that contains the word of data and indicates that the data is a prefetch response. The support circuitry in the remote PE then sends the response packet to the local PE over the interconnect network.

The support circuitry in the local PE receives the prefetch response packet and stores the word of data in the data prefetch queue. When the microprocessor issues the instruction that uses the data, the microprocessor reads the data from the data prefetch queue instead of creating a read request packet and waiting for a response.

The data prefetch queue stores a maximum of 16 words. The microprocessor can issue up to 16 data prefetch instructions before reading the data out of the prefetch queue 1 word at a time.

Messaging

The support circuitry also controls the messaging facility. The messaging facility transfers a special packet, 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.

The message queue is located in a reserved portion of local memory. The message queue stores up to 4,080 message packets and includes 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 that they are received.

To create a message, the microprocessor fills one of its internal write buffer lines with 4 words of data. The microprocessor then transfers the data from the write buffer to the support circuitry. During the transfer, the microprocessor also provides the support circuitry with an address and with cycle request information.

After receiving the address and cycle request information from the microprocessor, the support circuitry retrieves a function code and PE number from an entry in the DTB annex. The function code indicates that the support circuitry should perform a message write. The support circuitry then creates a message packet and sends the packet to the destination PE.

After receiving the message packet, the support circuitry in the destination PE attempts to store the message in the message queue. If 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 packet and sends the packet to the PE that created the message.

If the message queue in the destination PE cannot accept the message (full message queue), the support circuitry returns the message to the requesting PE by creating a no-acknowledge (NACK) packet. After receiving the NACK, the requesting PE can resend the message. Because of this feature, message delivery is guaranteed regardless of the amount of system message traffic.

In addition to message packets, the support circuitry may receive error "messages" and store the error messages in the message queue. The network interface generates error messages if it receives a misrouted packet or if it receives a packet that contains parity errors. If a network error occurs, the network interface turns the packet it received into an error message and sends the error message to the appropriate PE in the node.

Barrier Synchronization

The support circuitry also controls barrier synchronization operations. There are two types of barrier synchronization operations: barriers and eurekas.

A barrier is a point in program instructions where a microprocessor must wait until all other microprocessors associated with the barrier have finished their part of the program instructions. A programmer may use a barrier to ensure that all of the microprocessors associated with a distributed, parallel loop in a program finish the instructions for the loop before continuing with other program instructions.

The support circuitry in each PE contains two 8-bit registers called barrier register 0 and barrier register

1. Each bit in the barrier registers is connected to a separate barrier synchronization circuit. For example, Figure 13 shows the barrier synchronization circuit for bit 2 of barrier register 0 in a simplified CRAY T3D system.

All of the barrier synchronization circuits function identically and independently. The following paragraphs describe the operation of the barrier synchronization circuit connected to bit 2 of barrier register 0. Before the barrier synchronization process begins, bit 2 of barrier register 0 in each PE is reset to a logical 0.

Each barrier synchronization circuit in the CRAY T3D system is actually an AND-tree and fan-out tree circuit (refer again to Figure 13). The AND-tree circuit receives an input from all of the PEs. The fan-out tree circuit sends a copy of the final AND gate output to all of the PEs.

The first layer of the AND tree contains four AND gates. Each AND gate receives signals from two PEs. For example, one AND gate receives signals from bit 2 of barrier register 0 in PE 0 and bit 2 of barrier register 0 in PE 1. When all of the microprocessors set bit 2 of barrier register 0 to 1, the output of each of the four AND gates is 1.

The second layer of the AND tree contains two AND gates. Each AND gate receives signals from two of the AND gates in the first layer of the AND tree. When the output of all the AND gates in the first layer of the AND tree is 1, the output of both the AND gates in the second layer of the AND tree is 1.

The third layer of the AND tree contains the final AND gate. This AND gate receives signals from both AND gates in the second layer of the AND tree. When the output of both AND gates in the second layer of the AND tree is 1, the output of the final AND gate is 1. The output of the final AND gate connects to the fan-out tree circuit.

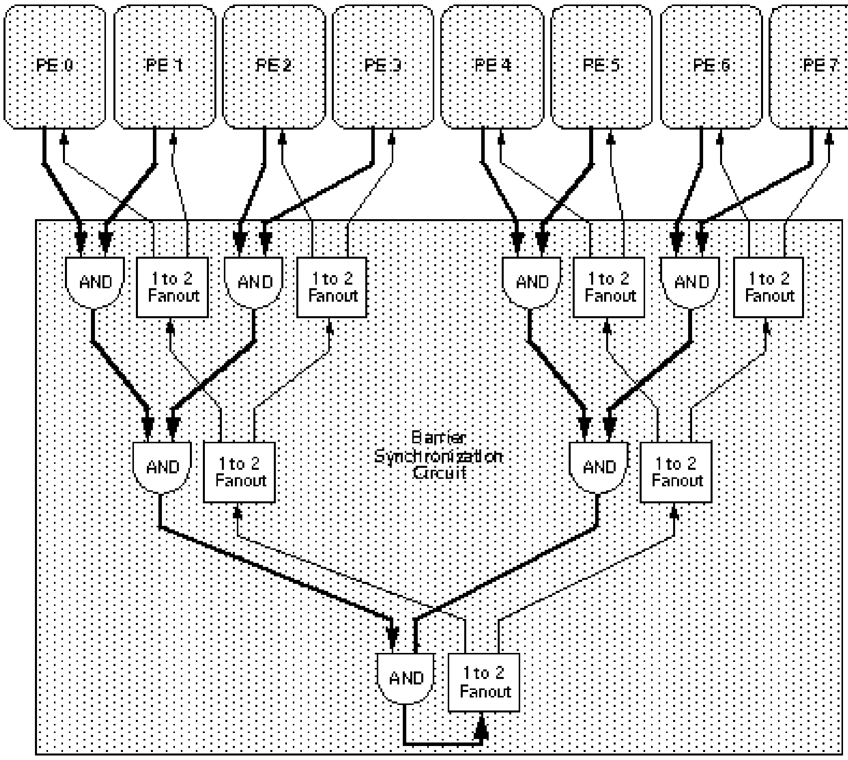

Figure 13. Simplified Barrier Synchronization Circuit

The first fan-out block in the fan-out tree receives a 1 from the final AND gate. After creating two copies of the 1, the first fan-out block sends the 1's to two fan-out blocks in the second layer of the fan-out tree.

The two fan-out blocks in the second layer of the fan-out tree each create two copies of the 1. The two fan-out blocks in the second layer of the fan-out tree then send the 1's to four fan-out blocks in the third layer of the fan-out tree.

The four fan-out blocks in the third layer of the fan-out tree each create two copies of the 1. The fan-out blocks in the third layer of the fan-out tree then send the 1's to the support circuitry in each of the eight PEs.

The microprocessor monitors the barrier synchronization circuit using one of two methods. In the first method, after the microprocessor sets bit 2 of barrier register 0 to 1, the microprocessor enters a loop that continuously checks the value of bit 2 of barrier register 0. After receiving a 1 from the fan-out circuitry, the support circuitry resets bit 2 of barrier register 0 to 0. Because the microprocessor constantly checks the value of bit 2 of barrier register 0, the microprocessor continues with program instructions as soon as bit 2 of barrier register 0 is reset to 0.

In the second method, after the microprocessor sets bit 2 of barrier register 0 to 1, the microprocessor enables a hardware interrupt. The microprocessor may then issue program instructions that are not associated with the barrier. After receiving a 1 from the fan-out circuitry, the support circuitry resets bit 2 of barrier register 0 to 0 and sets the hardware interrupt. This interrupt indicates to the microprocessor that all of the microprocessors have reached the barrier.

Each of the barrier synchronization circuits may also be used for eureka synchronization. Eureka synchronization uses a point in program instructions where a microprocessor is informed when the first microprocessor associated with the eureka has finished its part of the program instructions. Eureka synchronization functions like a global logical OR operation.

Eureka synchronization has several uses, including database searches. Using eureka synchronization, a programmer can stop a database search as soon as any microprocessor finds the data rather than waiting for all of the microprocessors to exhaust the search.

When used for eureka synchronization, a barrier synchronization circuit operates differently than when used for barrier synchronization. Before the eureka synchronization begins, each microprocessor associated with the eureka sets the appropriate bit of one of the barrier registers to a 1. For example, each microprocessor may set bit 2 of barrier register 0 to 1.

As soon as bit 2 of barrier register 0 in each PE is set to 1, the eureka synchronization begins (this event is usually controlled with a separate barrier synchronization operation). When a microprocessor finishes the program instructions associated with the eureka, the microprocessor resets bit 2 of barrier register 0 to 0. Because all of the inputs to the barrier synchronization circuit are not 1, the output of the final AND gate resets to 0.

When the output of the final AND gate is 0, the fan-out circuitry sends a 0 to the support circuitry in each PE. The support circuitry then resets bit 2 of barrier register 0 and if enabled, sets the microprocessor hardware interrupt. This signals the microprocessor that the eureka synchronization is complete.

Each barrier synchronization circuit receives an input from all of the PEs in the CRAY T3D system; however, not all of the PEs in the system must participate in the same barrier or eureka operation. The support software running in the host system may divide a barrier synchronization circuit into smaller circuits that each receive inputs from a limited number of PEs. This is done so the smaller barrier synchronization circuits more closely match user partitions.

Each AND gate in the AND tree is paired with a fan-out block in the fan-out tree. An AND gate and fan-out block pair is called a bypass point (refer again to Figure 13). The support software can redirect the output of an AND gate in a bypass point so that the output of the AND gate connects to the fan-out block in the bypass point. For example, Figure 14 shows a bypass point when the output of the AND gate is not redirected to the fan-out block and when the output of the AND gate is redirected to the fan-out block.

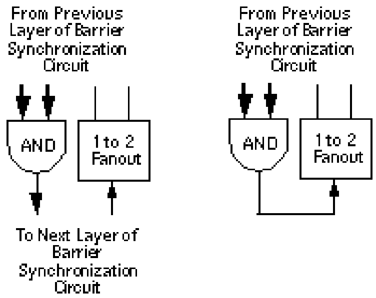

Figure 14. Bypass Points

By redirecting the output of AND gates in different level bypass points, the support software may divide a physical barrier synchronization circuit into a combination of barrier partitions. Because the number and shape of the PEs in a barrier partition may not exactly match the number and shape of the PEs in a user partition, the barrier synchronization circuit may be further partitioned by software using a barrier mask register.

Using the barrier mask register and the bypass points, the support software may match a barrier partition to a user PE node partition so that all the PEs in a user partition use the same barrier synchronization circuitry. For more information on user partitions, refer again to "Virtual PE Number" at the beginning of this section.

Fetch and Increment

The support circuitry also performs read or write operations to the fetch-and-increment registers. A fetch-and-increment register is a special register where after reading information from a fetch-and-increment register, hardware in the PE node automatically increments the contents of the register by one (refer to Figure 15).

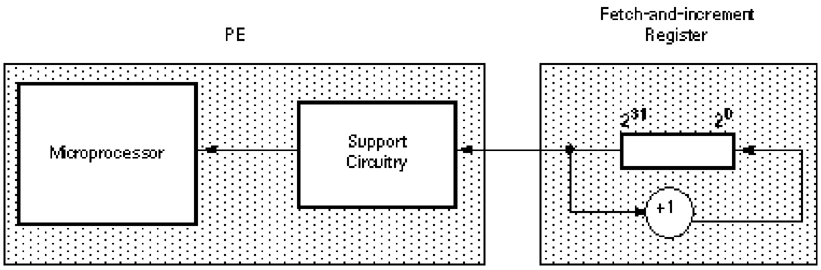

Figure 15. Fetch-and-increment

Each processing element node contains two fetch-and-increment registers, which are each 32 bits in size. This size is large enough to contain a loop index value or an address offset value.

Although each processing element node contains two fetch-and-increment registers, the registers function independently of the PEs. Any PE may use any of the fetch-and-increment registers in a partition.

The fetch-and-increment register may be used to dynamically distribute independent iterations of a program loop to more than one microprocessor. For example, four independent iterations of a program loop may be distributed among four different microprocessors.

Status

Each PE contains registers that indicate the status of PE operations. The status information includes error information and outstanding request information. The status information is used by the operating system.

BLT

The block transfer engine (BLT) is an asynchronous direct memory access device that redistributes system data. The BLT redistributes system data between globally addressable system memory and local memory in either of the PEs in a processing element node. The BLT can create up to 65,536 packets that contain one 64-bit word of data or up to 65,536 packets that contain four 64-bit words of data without interruption from the PE.

The BLT performs four types of data transfer operations: constant stride read, constant stride write, gather, and scatter. A constant stride read operation transfers data from fixed increment address locations in system memory to fixed increment address locations in local memory. A constant stride write operation transfers data from fixed increment address locations in local memory to fixed increment address locations in system memory.

A gather operation transfers data from nonsequential memory locations in system memory to fixed increment address locations in local memory. A scatter operation transfers data from fixed increment address locations in local memory to nonsequential address locations in system memory.

The BLT receives initial transfer parameters from one of the PEs in a processing element node and then functions independently from the PEs. The BLT contains three main components: system addressing, local addressing, and control.

The system addressing portion of the BLT generates addresses that are used to reference a location in system memory (which includes local or remote memory). For example, the system addressing circuitry may generate addresses that point to a word of read data in a remote PE during a BLT constant stride read or gather operation.

The local addressing portion of the BLT generates addresses that are used to reference a location in the local PE. For example, the local addressing circuitry may generate addresses that point to the location in local memory where data from a BLT read response packet will be stored.

The control portion of the BLT controls the BLT transfer. In addition, the control circuitry provides an interrupt to the PE when an error occurs, when the BLT is free to start a transfer, and when the BLT transfer is complete.

Network Interface

The network interface assembles outgoing request and response packets and steers incoming request and response packets to the correct PE in the node. The network interface also contains the fetch-and-increment registers and barrier synchronization bypass point control.

When assembling an outgoing request or response packet, the network interface receives packet header information from PE 0, PE 1, or the BLT. In addition, the network interface may receive data from a PE and receives a virtual or logical PE number from a PE or the BLT.

If the PE number is a virtual PE number, the network interface converts the virtual PE number into a logical PE number. The network interface then converts the logical PE number into a routing tag that is used in the outgoing packet.

When receiving an incoming packet, the network interface checks the destination node number in the packet header. If the destination node number is the same as the number of the node that the network interface is in, the packet arrived at the correct node. The network interface then sends the packet to the destination PE. If the destination node number is not correct, the network interface converts the packet into an error message and sends the error message to one of the PEs in the node.