A Survey of Commercial Parallel Processors

Edward F. Gehringer
Janne Abullarade
Michael H. Gulyn

Computer Systems Laboratory
North Carolina State University
Raleigh, NC 27695-7911

Abstract

This paper compares eight commercial parallel processors along several dimensions. The processors include four shared-bus multiprocessors (the Encore Multimax, the Sequent Balance system, the Alliant FX series, and the ELXSI System 6400) and four network multiprocessors (the BBN Butterfly, the NCUBE, the Intel iPSC/2, and the FPS T Series). The paper contrasts the computers from the standpoint of interconnection structures, memory configurations, and interprocessor communication. Also, the shared-bus multiprocessors are compared in terms of cache-coherence strategies, and the network multiprocessors are compared in terms of node structure. Where possible, price and performance information has been included. The reader is cautioned that this survey is based largely on information submitted by manufacturers; the authors have not performed any independent evaluation.

Disclaimer

This report is condensed from a class project written by two of the authors under the direction of the third. The material has been drawn almost entirely from manufacturer-provided information available to the general public. In many areas, there is insufficient detail to make meaningful comparisons between systems. In addition, some of the data may be less up-to-date than others, although in almost all cases, the information was obtained from manufacturers in 1988. Some machines, for example, Ametek's 2010 and International Parallel Machines' IP-1, have been omitted due to lack of sufficient information. Despite these shortcomings, we hope this paper will be useful to computer architects interested in the current capabilities of commercial multiprocessors.

1. Overview

Over 30 companies have designed or marketed parallel processing computers. Many of the computers have not yet been configured with their maximum number of nodes. Manufacturers often boast that their computers offer supercomputer performance at one-fifth to one-tenth the cost. It is difficult to take these performance numbers at face value, because they are usually derived by multiplying the peak performance of one processing node by the number of nodes in the computer.[1] This does not account for inter-node communication,

[1] Manufacturers, of course, realize that a single number is not an adequate measure of a computer's performance. But, as two manufacturers mentioned in reviewing this paper, it is the customers that often insist on a single number to make a first-order comparison between systems. Only after that do more realistic measures come into play.

which limits the actual performance of the computer.

The multiprocessors discussed in this paper all have at least eight computational processors. They can be classified into two broad groups: shared-bus and network multiprocessors. We discuss four computers that use the bus architecture: the Encore Multimax, the Alliant FX/80, the Sequent Balance and Symmetry, and the ELXSI 6400. Among the network-based computers we consider, one of them, the BBN Butterfly™ Parallel Processor, uses a switching network for communication between the processing nodes. The other three network computers use the hypercube architecture: the Intel iPSC/2 line, the NCUBE/n, and the Floating Point Systems (now FPS Computing) T Series. Because of the limitations of bus contention, none of the bus-based computers has more than 32 nodes.[1] However, a future release by Encore promises a maximum of 128 processors. Among network-based machines, the BBN Butterfly can have up to 256 processors. NCUBE, whose processing node consists of only 7 chips, is selling the NCUBE/ten computer, which has 1024 nodes. Thinking Machines Corporation's Connection Machine [TR 8/88] can have up to 65K processors, but as the storage per processor is quite limited (no more than 64K *bits*/processor), it is not strictly comparable to the machines covered in this survey.

This paper is divided into sections that deal with the different components of a parallel computer: the interconnection structure, the memory structure, the node structure (for network-based computers), cache coherence (for bus-based architectures), communication and synchronization, performance, and price. In each section, shared-bus computers are considered first.

[1]For the ELXSI, the true limiting factor is that the bus runs on a 25 ns. cycle time, limiting the maximum length of the bus. That in turn limits the number of cards that can be physically attached to the bus, as it is necessary to have clearance for heat sinks, cables and such.

2. Interconnection structures

2.1. Shared-bus architectures

A problem faced by shared-bus systems is performance degradation caused by contention for the bus and for memory. Bus latency is the time for the bus and the shared memory to complete an average transaction. It consists of two parts, bus arbitration and usage, and memory-access time. Shared-bus systems use different approaches to reduce bus traffic. The Multimax system divides the global bus into three independent buses: one for addresses, one for data, and one for vectors. The Balance system has a one-bit data path called the System Link and Interrupt Controller (SLIC) Bus which connects all major components in the system and allows them to exchange interrupts, other low-level control signals, and error information independently of the system bus. The Alliant system divides the system bus into two data buses and an address bus. It also contains a concurrency control bus—a high-speed inter-element communication path that is independent of the program's data and instruction paths. The ELXSI Gigabus™ is a single 110-bit wide bus, clocked at 25 ns. and arbitrated by a Bus Control Unit. Memory data, memory addresses and control data are multiplexed on the Gigabus.

Encore's Multimax. The Multimax's Nanobus interconnects one System Control card, eight memory cards and eleven requestor cards (advanced dual-processor cards, Ethernet/Mass storage card, etc.). The Nanobus has an usable throughput of 100M bytes per second, which cannot be achieved by a single bus. Therefore, the Nanobus consists of three buses: the 32-bit address bus, the 64-bit data bus, and the interrupt-vector bus, which carries 14 lines to distribute interrupts through the system. See Figure 1.

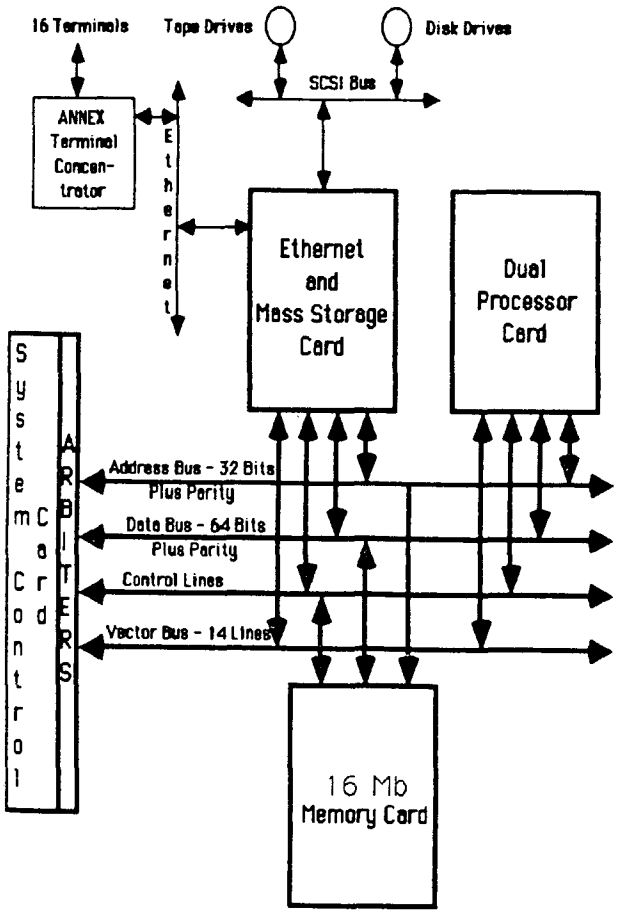

Figure 1: The Multimax System [Sc 11/86]

At least one of each of the following card types should be connected to the Nanobus:

- An Advanced Dual Processor card (APC), which contains two 32-bit National Semiconductor 32332 processors in the Multimax 320 (or 32032s in the Multimax 120), each with a private 64K-byte write-through cache. Optional Wytek floating-point co-processors have a peak floating-point rate of 1 DP megaflop. Each processor is connected to a memory-management unit (MMU). Earlier versions of the Encore also used an extended memory management unit (XMMU), but it is no longer needed with the 32332's 32-bit virtual address space. A new, much faster processor will be announced soon.
- The System Control Card (SCC), whose main element is a National 32016 microprocessor with 64K bytes of ROM and 128K bytes of RAM. The SCC performs bus arbitration, initializes the system, diagnoses all cards, monitors the environment and reports status whenever another card fails, and communicates with an operator and a remote console.
- A Shared memory Card (SMC), which has 16M bytes of interleaved memory with error detection and correction codes.
- An Ethernet/Mass Storage Card (EMC), which provides an Ethernet interface and a connection to storage devices via the Small Computer System Interconnect (SCSI). Each EMC card has a National 32032 to manage input/output transfers, diagnostics, and DMA engines.

The Nanobus is a pended (or "split-transaction") bus with an 80-ns. cycle. After the address of a read operation is transferred to memory, the bus is released. This action makes the bus available for other transactions while the requesting processor waits for the data read from memory. When memory is ready to send the data to the requestor, it has already released the bus. Therefore, all data transferred on the Nanobus contains a tag which identifies the requestor. The tag is generated by the requestor and consists of a 4-bit requestor ID [Sc 11/86, Bi 7/87] and 2 reserved bits. The reserved bits specify which processor within a module requested the data. The tag is latched by the addressed module and is returned with the requested data.

The Nanobus insures fair access to the address and data buses by using two arbiters: a centralized and distributed arbiter. Requests for the bus are send to the arbiters at the beginning of a clock cycle and grants are issued at the end of the same clock cycle.

The central arbiter is responsible for guaranteeing equal access for all requestors. It achieves this goal by using a round-robin scheme to rotate the highest priority among the modules [Sc 11/86, Bi 7/87].

A distributed arbiter is found on each module and consists of a state sequencer which determines whether the module may request the address bus. The individual distributed arbiters can modify the bus arbitration in certain situations. For example, if one of the modules is trying to access memory and is rejected several times over a short time interval, it asserts a special "priority" control line. The address-bus arbiter stops accepting any new bus requests into the round-robin queue until all the outstanding requests have been serviced [Bi 7/87].

The Sequent Balance. The Balance system is built around the system bus, which links the system's CPUs, memory and I/0 subsystems. The system bus operates at 10 MHz (1 cycle = 100 ns.), and can carry 64 bits of data multiplexed in time with 32-bit addresses. Currently, Balance systems only use 32-bit data and 28-bit addresses. However, the new Sequent Symmetry machine [LT 12/87] uses all 64 bits of the datapath. The system bus reads from and writes to memory data packets of 1, 2, 3, 4, and 8 bytes.

The bandwidth of the bus data path is optimized by the use of a split request/response protocol. In this protocol, memory read and memory write requests are stored in separate request pipelines (queues), and individual requests and responses are interleaved in sequential bus cycles. The bus is only busy only during the cycles needed to transfer information since requests and responses are serviced separately. Continuous use of 8-byte reads and 8-byte writes produce an effective data transfer rate of 26.67M bytes per second. If the full 64-bit bus were used, the bus would have an estimated data transfer rate of 64M bytes per second [Se 11/86, Th 2/88].

The system bus uses a centralized multilevel arbiter. The access of the bus is granted as follows. The first priority is given to an I/O controller responding to a read request, the second priority is given to a memory board responding to a read request, the third priority to Multibus adapter boards, SCSI/Ethernet/Diagnostic boards, and dual-channel disk controllers, and the last priority is given to the dual-processor boards. Among the two lowest-priority modules, arbitration is done by a round-robin scheduling discipline [Se 11/86].

The Balance system uses a distributed control-flow mechanism to avoid bus congestion. Each requestor keeps status information about the read and write queues and does not request access to the bus unless space is available in the corresponding queue [Th 2/88].

The following circuit boards are connected to the system bus (see Figure 2):

- One to fifteen dual-processor boards in the 21000 (one to six in the Balance 8000). Each board has two NSC Series 32000 CPUs. Each CPU has a 8K-byte memory cache, a floating-point acceleration coprocessor, an NS32082 Memory Management Unit, and a System Link and Interrupt Controller (SLIC) chip. Each CPU also has a small local memory to store frequently used kernel routines.

- One to four memory controller boards. Each board has 2M bytes of RAM and error checking correction (ECC) logic.

- Memory expansion boards. A 2M-byte or 6M-byte memory expansion board can be connected to a memory controller board to give a total of 4 or 8M bytes per controller.

- One to four Small Computer System Interface (SCSI)/Ethernet/Diagnostics (SCED) boards. Each board serves as host adapter on a SCSI bus and supervises system startup, diagnostics, and access to an Ethernet local area network.

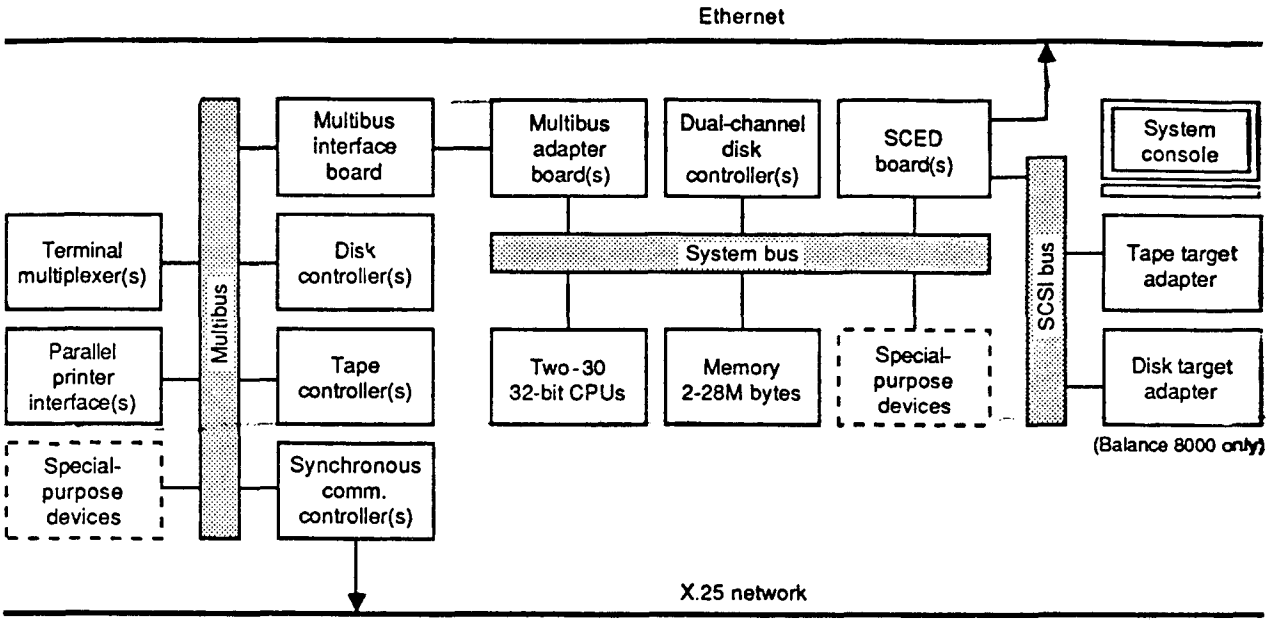

Figure 2: The Balance System Architecture [Th 2/88]

- A maximum of four MULTIBUS adapter boards which connect the system bus to a MULTIBUS system.
- A maximum of four dual-channel disk controllers (DCCs) per Balance 21000 system. A Balance 8000 system only has one DCC.

The Balance system contains a 1-bit data path called the System Link and Interrupt Controller (SLIC) Bus which interconnects all the SLIC chip in the system. A SLIC chip is coupled with each processor, memory controller, I/0 controller and other major components in the system. The SLIC chips provide support for interrupt distribution, low-level mutual exclusion, and configuration and error control. One of the goals of the SLIC bus is to simplify the system bus and to allow the CPUs and other subsystems to exchange interrupts and other low-level control signals, and configuration and error information.

One of the most important tasks of the SLIC subsystem is the interrupt control and distribution. Every SLIC on the SLIC bus responds to interrupts directed to its SLIC ID number. Also, each SLIC coupled with a processor responds to a destination group ID number. When there is an interrupt in the system, all the SLICs in the specified processor group number arbitrate among themselves to decide which one will accept the interrupt. The SLIC which accepted the interrupt is masked out of the group, which means that this SLIC will not arbitrate for another interrupt until the current interrupt is completed. A SLIC arbitrates for an interrupt based on its local priority register. The idea is to have the idle processors and the processors running the least important tasks handle most of the interrupts [Se 11/86, Be 10/87, Th 2/88].

Alliant FX/80. The Alliant FX/80 is built around the memory bus, a synchronous memory access bus that consists of two 72-bit data paths (64 bits of data, 8 bits for single-bit error detection and correction and double-bit error detection), a 28-bit address bus, and a control bus. The data buses are bidirectional and are connected to memory modules, the computational

caches, and the interactive processor caches.

Data is always transferred as four eight-byte words, which is the size of a cache block. The first word is transferred on data bus A, and subsequent data alternates between data bus A and data bus B for four cycles. Another transfer request can be interleaved with the first transfer to allow a maximum of eight 8-byte words to be transferred in four cycles. The data-bus cycle time is 85 nanoseconds, and provides a maximum total bus bandwidth of 188M bytes per second. The data buses use a protocol called "convenient word first" to speed up accesses which may not be to the first word of a cache block [Al 10/86].

The Alliant FX/80 contains up to eight processors called computational elements (CEs) used for computational work. The CEs are connected to the memory bus through a computational processor cache (CPC). Up to eight computational elements are connected dynamically to up to four cache ports through a crossbar interconnection that provides a data bandwidth of 376M bytes per second, provided that the requested data is in the cache. In addition to the CEs, the FX/80 has up to twelve interactive processors (IPs) which execute interactive user jobs, input/output, and other operating system activities. The IPs are connected to the memory bus via the interactive processor caches (IPCs) and to peripheral devices via the VME bus (or optionally, a Multibus), which is IEEE 796 compatible. The FX/80 contains up to four IPCs, each of which connects one, two or three interactive processors. Figure 3 shows how the computational elements and the interactive elements are connected to the memory bus.

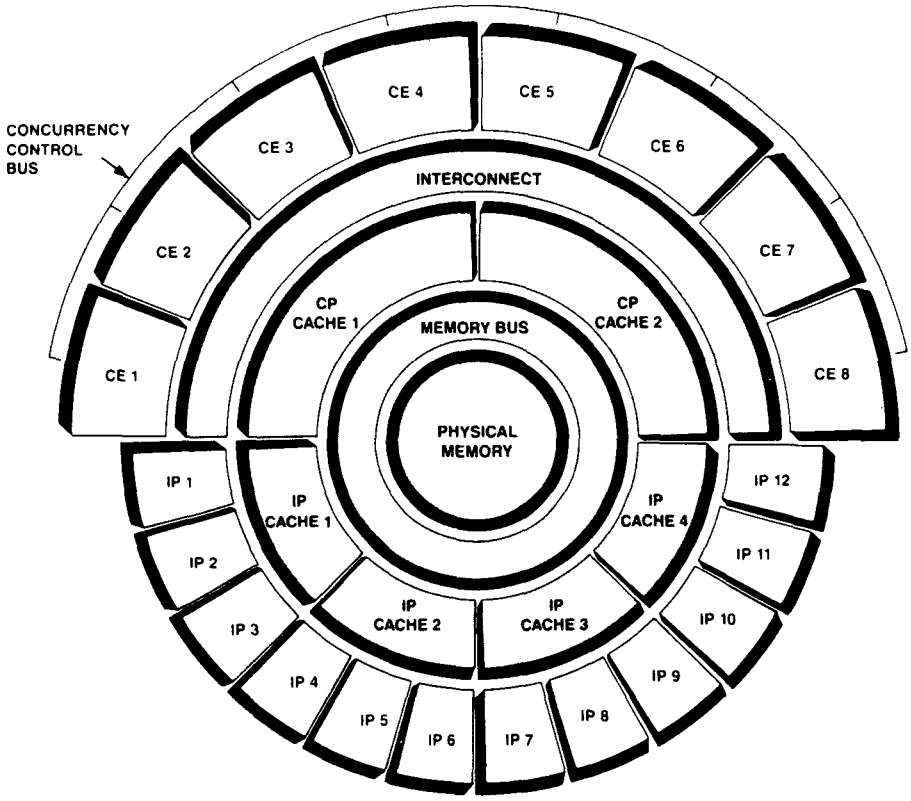

Figure 3: The Alliant FX/80 System Architecture [Al 10/86]

Each computational element is a microprogrammed pipelined processor with integrated floating point and vector instruction sets. The CE has four main functional systems: a pipelined instruction unit, a pipelined vector and floating point unit, a CE switch and a concurrency control unit [Pe 5/86, Al 10/86].

- The instruction unit is a five-level pipelined processor and includes the instruction cache, control section, instruction processor, and address translation unit. The instruction cache contains 64K bytes of 85-nanosecond RAM and is addressed with virtual addresses. The instruction cache has a copy of the current instruction stream and initiates a load when an instruction fetch operation results in a miss.

  The control section consists of an instruction parser, a microsequencer, and a RAM-based control store. The instruction parser decodes the opcodes from the data path and generates control-store microaddresses, checks for dependencies between instructions, and prevents a new instruction from executing if dependencies exist, and uses a branch-prediction unit to anticipate the flow of the program and prefetch instructions. The microsequencer and control-store memory array delay the execution of certain fields of a microword, decode delayed fields and control microtraps and unaligned memory references.

  The instruction processor consists of the address unit and the integer/logic unit. The address unit contains two identical 16-bit slices. The address unit contains the instruction buffer, which receives the output of the instruction caches, latches it and rotates 16-bit words to align the fields of the instruction.

  The address translation unit translates virtual addresses into physical addresses. It has a virtual address register and a translation cache.

- The floating point and vector unit contains eight 64-bit floating point data registers, each of which holds a 32-bit or a 64-bit floating-point number, eight vector registers each of which contains thirty-two 64-bit wide elements and floating-point status and control registers. These control registers support condition codes to handle the results of floating point comparison-and-test instructions, and various exception code. Floating-point operations are provided for arithmetic, conversion, testing and branching, and a hardware implementation of square root.

- The CE switch is a four-by-eight address and data switch, connecting up to eight CEs with the four cache ports on two CP caches. It consists of twenty-four 2,600-gate gate arrays, and provides a peak bandwidth of 376M bytes/second, when data is in the cache [St 6/87].

- The concurrency control unit (CCU) is a gate array which connects CEs via a concurrency control bus. The CCU is connected to the instruction unit of a computational element and to up to seven other CCUs, thereby controlling up to eight CEs running concurrently.

The Interactive Processor is an industry-standard Multibus card that contains a Motorola MC68020 microprocessor, an address-translation unit, an I/O map, and parity-protected RAM. Multibus devices are able to perform direct memory accesses via a direct memory access channel, operating through the I/O map. IPs are general-purpose processors; they can execute interactive applications such as editors and operating-system tasks such as paging, scheduling, and I/O. Up to twelve IPs can be used for these purposes, freeing the CEs to concentrate on computation. Each IP has 4M bytes of local dRAM for caching frequently used OS pages.

One such IP, the system interactive processor, is connected to the system console and remote diagnostic port. Some of its functions are to bootstrap the system, to

execute diagnostic software, to control the system diagnostic bus, and to handle other system bookkeeping tasks.

The ELXSI System 6400. The ELXSI System 6400 interconnection is the Gigabus (Figure 4). This bus is 110 bits wide, including 64 bits of data, 35 bits of control and 11 bits of parity. All functional units attach directly to the Gigabus, including up to 12 proprietary ECL CPUs, 8 memory controllers, 4 I/O processors, a Service Processor and a Bus Control Unit. The bus clocks at 40 MHz (25 ns.) for an aggregate data transfer rate of 320M bytes/s., exclusive of control and parity. Both interprocess communication and memory accesses occur over the Gigabus.

The bus is arbitrated by a request/grant mechanism based on slot priority. Every functional unit that wishes to transfer requests control from the Bus Control Unit. The BCU grants the bus to the highest priority functional unit for use on the next bus cycle. Someone else can be using the bus during this arbitration. On the cycle after the transfer the receiving functional unit acknowledges on a separate bus. Thus it is possible to sustain the 320M byte/s. data transfer rate of the Gigabus.

Eleven of the control bits are used for the management of the Gigabus itself. Included in these bits are the request for the bus, the grant of the bus and the acknowlegement bus, as well as power-fail warning, the bus clock signal and a variety of other housekeeping signals.

The remaining 88 bits make up the Bus Information Quantum (BIQ). The 24 bits of the BIQ control field identify the source functional unit, the destination functional unit, the BIQ (operation) type, whether this is the last BIQ of this operation, a tag field, and some housekeeping and escape functions. Five bits are currently allocated for the functional unit address, hence the limitation of 32 functional units on the Gigabus. The format of the other 64 bits of the BIQ is dependent on the BIQ type. The two classes of BIQs are data operations and message operations.

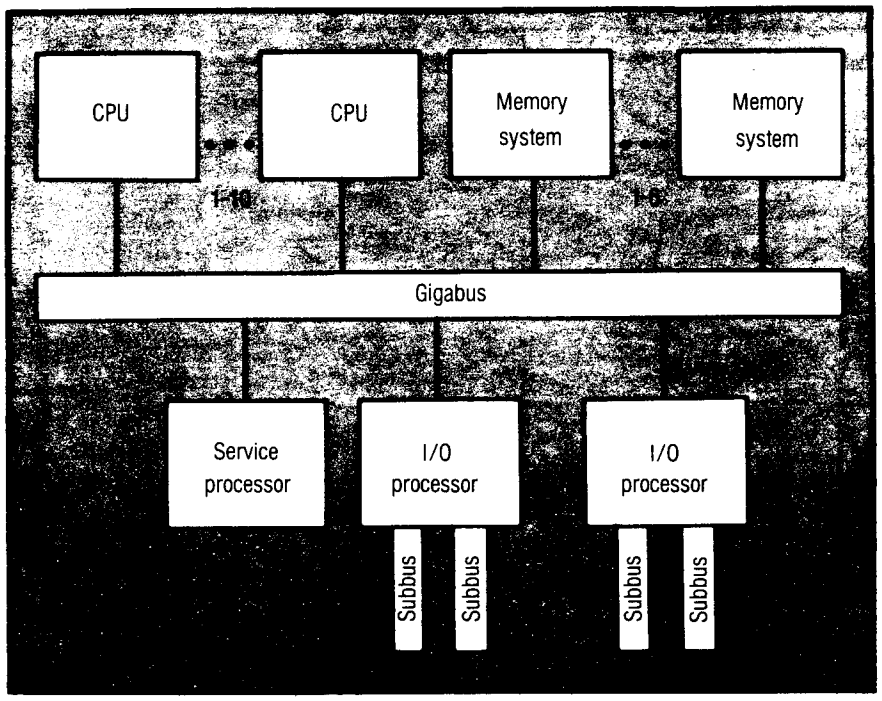

Figure 4: The ELXSI System 64000 [Ol 85]

- Data BIQs (that is, memory operations) currently include Read, Write, Exchange Or, Exchange And, Exchange Write, and Two Word Read. The exchange operations are atomic operations against memory, allowing the implementation of locks and semaphores.

- Message BIQs include Message Send, Small Message Send, Small Hardware Send, Received and Released Buffer, and Received and No Buffer Release. Thus the messaging protocols and the concept of the process are part of the machine architecture at its lowest levels.

Data BIQs include a process ID, an operation code and the physical memory address. If it is a Write request, then the data for memory follows in the next BIQ. For operations requiring a response, such as a Read, the memory controller will respond with one or two BIQs when the data is ready. The tag field in the control bits is used to relate responses to requests. It is possible for functional units to have multiple operations in progress at once. For example, filling a cache block might result in four outstanding two word reads. The responses might come back in any order, necessitating the use of the tag field to sort things out. The bus is not held between request and response.

Message BIQs contain a sending-process ID, an operation code, a destination process ID, the physical memory address of the message body (in most cases) and some information which helps the receiving functional unit to assess the software priority of this message. If appropriate, the functional unit will reschedule to allow the receiving process to execute immediately. There are provisions in the protocol to redirect messages in the event that a process has migrated to a different CPU. On heavily loaded systems, load-leveling process migrations occur every few seconds, so this is an important part of the Gigabus protocol.

Three different models of CPU are currently supported. The original CPU, introduced in 1984, is roughly 4 VAX MIPs and has a 16K-byte write-back cache. The 1986 CPU is a reworked version of the 1984 CPU with faster floating-point operations and branching and a 64K-byte cache. It generally performs at 7 VAX MIPs. The 1988 CPU is a completely new design. Pipelined, nearly all instructions execute in a single cycle (à la RISC). The cache is 1M byte and is split between code and data. Cache-to-CPU bandwidth is 640M bytes/s. This is the first ELXSI CPU to introduce vector instructions and also contains many innovations specific to ELXSI's real-time market. The CPU performs at roughly 25 VAX MIPs and will perform at 10 MFLOPs on the standard 100×100 64-bit Linpack benchmark. A notable feature of the ELXSI system is that all three generations of CPU can execute in the same frame. Thus upgrades consist of adding whatever is the current generation of CPU to the existing hardware, rather than discarding any existing hardware.

Each CPU contains sixteen complete process contexts called "register sets." The CPU always runs the highest-priority ready process that is in a register set. The combination of such a simple scheduling algorithm with hardware support results in extremely fast context switches, on the order of ten microseconds in the earlier two CPUs and around three microseconds in the current CPU. Extremely fast context switches are very important in the real-time marketplace.

At the architectural level, Input Output Processors (IOPs) are similar to CPUs. Specifically, I/O controllers generally have on-board CPUs which run the device driver. Communication between the operating system and the controller is via messages rather than the more traditional interrupts. During I/O operations the IOP supervises the DMA of data between the controller and main memory. These I/O operations occur over the Gigabus. CPUs and IOPs look the same from the perspective of a Memory Controller.

Each Memory Controller (MC) supports up to 256M bytes of memory. This memory

is organized into interleaved Memory Array Boards. The MC itself runs at 25 ns. and has deep buffering to allow it to service many requests simultaneously. There can be up to eight Memory Controllers per system, yielding a maximum physical memory size of 2 GB. Lifting the limit on the maximum amount of physical memory is mostly a software problem and is currently being investigated.

The Service Processor performs diagnostic and bootstrap functions. From the end user's perspective the most interesting function it performs is to monitor the health of the CPUs, deconfiguring them if they fail. This include migrating processes from the failed CPU.

2.2. Network Multiprocessors

The BBN Butterfly Parallel Processor has an interconnection network unique among the machines considered in this paper. Its "butterfly" switching network is an indirect binary n-cube packet-switching network, that connects from 2 to 256 processing nodes [Br 4/87]. According to BBN, the advantages of the Butterfly switch over other interconnection architectures include performance, cost, and flexibility.

The Butterfly switch connects different processing nodes, each consisting of a processor and memory. Figure 5 shows a Butterfly switch for a 64 processor system [BB 3/86]. The figure would more accurately represent the switch if it were in the shape of a cylinder, so that both the upper left input to the switch and the upper right output from the switch would be connected to the same processing node. As a result, any message going from one processor to any other processor is delayed by passing through log n stages.

The Butterfly is a shared-memory machine, while the hypercubes are message-passing machines. One Butterfly processor can reference memory in any other processor, through the switch. Instead of a switch, the other network processors provide hardware assistance for message passing. The Butterfly can do message passing too, but does it via software, using message buffers in shared memory.

The Butterfly switch is also more easily scalable than a single-bus interconnection. Bus architectures are limited to a few dozen processing elements; at this point, contention for the bus becomes so great that adding more processors will not

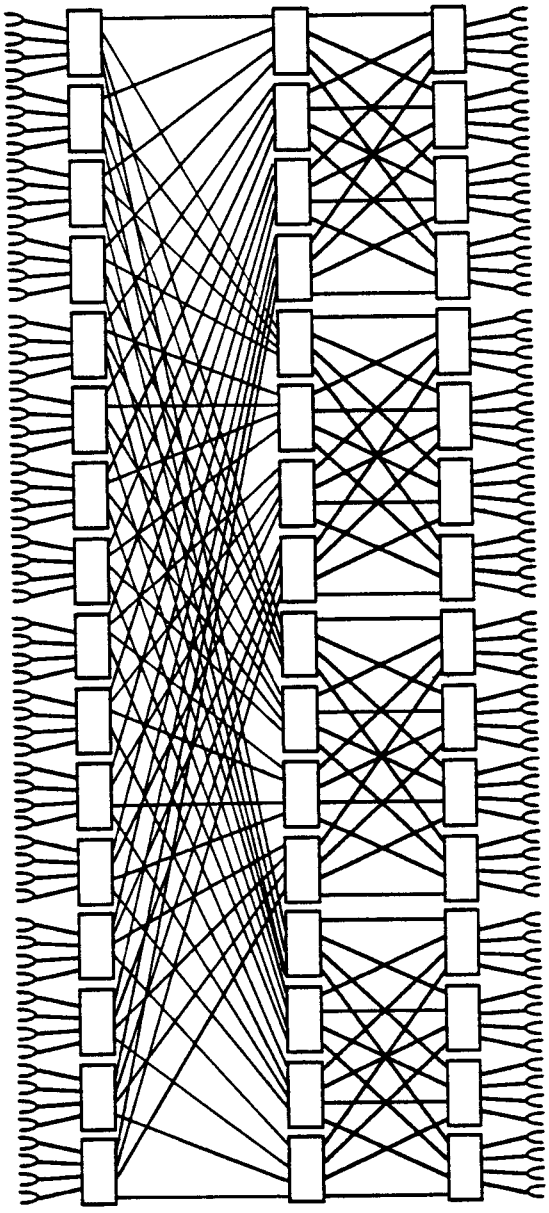

Figure 5: A 64-Node Butterfly Switch [BB 3/86]

improve performance. However, the bandwidth of the Butterfly switch grows with the number of processors, so a Butterfly can take advantage of a very large number of processors. The interconnection bandwidth can be increased further through the use of redundant switching nodes, which decrease contention on the Butterfly switch.

Compared with a standard hypercube interconnection, the Butterfly switch achieves lower latency. In first-generation hypercubes,[1] a message directed to a non-neighbor node must be forwarded by other processors along the way. This causes the other processors' applications to be interrupted, and therefore reduces performance. As a result, programmers try to program an application so that message passing need only be done between immediate neighbors [BB 87]. This often requires extra programmer effort. Also, since the processing nodes do not share memory, they require a copy of the program on each processing node. This necessitates larger memories at each node [BB 87].

A disadvantage of the Butterfly switch is that in order to avoid switch and memory contention, data and instructions must be distributed among the memory modules in a balanced way. This is facilitated by the Uniform System software library routines for memory management [Re 12/86]. Switch contention (but not memory contention) can also be reduced by placing redundant switches in the switching network. Redundant switches also improve fault tolerance. For clarity's sake, these redundant paths are not shown in the 64-node Butterfly switch in Figure 5 .

[1]The Intel iPSC/2's "direct-connect" routing [In 9/87] uses a multistage interconnection network which is isomorphic to a Butterfly interconnection. It eliminates the need to interrupt processors along the route. However, the Butterfly still achieves much lower latency than the iPSC/2. In a sense, this comparison is of apples (shared-memory references) and oranges (message passing). There is reason to believe that the Butterfly would not be faster at message passing than the iPSC/2. The lowest *process-to-process* message latency in the Butterfly is 1.80 ms. [SC 87] ; while for the iPSC/2 it is 0.58 ms. Both of these times are dominated by software.

The other network computers considered in this paper use the hypercube architecture, which was originally developed at Caltech. A hypercube is well suited to science and engineering applications because it contains many different topologies, such as the ring, various meshes, and FFT, that are encountered in these disciplines. The hypercube architecture is also being programmed for AI applications as they begin to require faster computing resources.

The three hypercube computers covered in this survey are the Intel iPSC/2, the NCUBE/n, and the FPS T Series. The largest Intel system sold to date contains only 128 nodes. The NCUBE architecture allows up to 1024 nodes, which is the size the NCUBE/ten (2^{10} nodes). The FPS T series has been discontinued by the manufacturer. It was constructed out of modules containing 8 processing nodes each. These modules can be aggregated into hypercubes of up to dimension 14, but the largest product was the FPS T200, which had 128 nodes.

All three hypercube computers have unshared local memory, and use message passing to communicate between nodes. The processing nodes of all three have floating-point accelerators. However, the processing-node boards differ significantly in size. While Intel and FPS use one or two large boards per node, NCUBE managed to put 64 processing nodes one board 16″ by 22″ [NC 88]. NCUBE provides only 512K of memory per node, though, while FPS provided 1M byte and Intel provides 4M bytes in each node.

3. Node Structure

A prominent feature of a network architecture is its node structure. A processing node usually consists of one control processor, one or more numerical coprocessors, local program and data memory, and hardware to communicate with other nodes. To minimize initial engineering costs and facilitate programming, all the processing nodes in commercial parallel architectures have the same basic design. Some processing nodes

can be enhanced by either adding daughter boards or connecting full-size enhancement boards.

The key issues designers face when designing a processing node include—

- Should custom VLSI or off-the-shelf components be used?
- How much numerical processing power is needed, and what processor should be used?
- How much memory should be provided per node?
- What kind of communication hardware should be provided?

All the processing nodes except NCUBE's were constructed from off-the-shelf components. NCUBE designed all of the processing node except the memory onto one VLSI chip. Off-the-shelf components allow the node to be configurable and reduce the initial engineering investment. Advantages of NCUBE's design include improved reliability, lower size and power consumption, and lower cost in mass production. NCUBE's decision to use one custom VLSI CPU is based on the belief that high performance in scientific applications can best be achieved with several hundreds of processing nodes [NC 88]. NCUBE has sold several NCUBE/tens, with 1024 nodes, the largest configuration covered in this survey.

Designed to perform numerical operations quickly, NCUBE's nodes have 32-bit processors and floating-point execution units that can process 64-bit floating-point quantities [Ha 10/86]. Intel and FPS also offered pipelined vector units to increase each node's performance to the minicomputer range.

All the hypercube nodes have only local memory, while the Butterfly Parallel Processor offers shared as well as local memory. A strictly local-memory architecture requires some data and code to be replicated, and therefore demands more memory than a shared-memory machine. Consequently, nodes in all the hypercubes except NCUBE's contain several megabytes of memory. Although the NCUBE nodes have only 512K, the 4M-byte host can run larger computations [Ha 10/86]. This is useful in the common case where the pre- or post-processing for a large numeric application requires much more memory than the individual processes of the parallel computation.

On first-generation hypercubes, "store-and-forward" message transmission caused intermediate processors to be interrupted and thus degraded performance. Some hypercubes now have hardware assistance for message forwarding. The iPSC/2 series nodes have routing hardware on the Direct-Connect routing board which receives and transmits messages independent of the processor [In 9/87]. The Direct-Connect routing board creates a connection from the source node to the destination node and then sends the message at 2.8M bytes/s. The Butterfly has a multistage routing network, removing the need for shared-memory references to traverse intermediate nodes [BB 3/86].

Butterfly Node Structure. Each Butterfly processing node (Figure 6 [BB 87]) is identical except for I/O options. Each node is capable of a peak performance of 2.5 MIPS. Its CPU is a 16 MHz 32-bit Motorola 68020 microprocessor, which can off-load floating point arithmetic to a Motorola 68881 floating-point coprocessor (IEEE 754 standard). Each node also has a 68851 paged memory- management unit (PMMU) and up to 4M bytes of memory [BB 87].

The Butterfly processing node has a separate communication processor to allow uninterrupted application processing. The communication processor is actually a microcoded coprocessor called the Processing Node Controller (PNC). The PNC, in combination with the 68851 PMMU, translates virtual addresses into physical addresses for both local and remote memory, so that a program need not distinguish between local and remote addresses [BB 3/86]. The PNC's microcode also provides operations that enhance the functionality of the MC68020 for parallel processing. For ex-

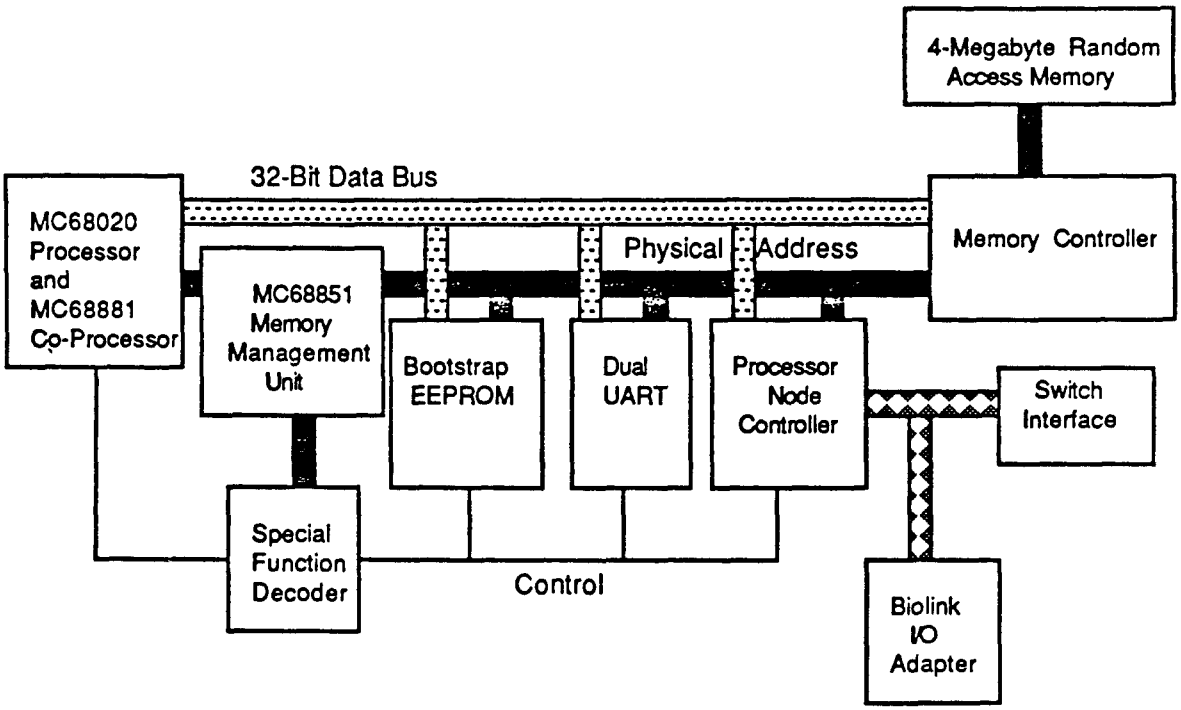

Figure 6: Block Diagram of Butterfly Node [BB 87]

ample, it has a test-and-set operation that guarantees "atomic" access to a memory location. The PNC can insure mutual exclusion because it handles all memory references made by its processor and by remote processors to its memory via the switch.

NCUBE Node Structure. The NCUBE processing node is generic for all models of NCUBE parallel processor and consists of seven chips, six DRAMs, and the custom processor. Designed over a 2-year period exclusively for parallel processing, the custom processor contains 160,000 HMOS transistors. It contains a 32-bit integer ALU and shifter, 16 general-purpose registers, a 64-bit IEEE standard floating-point unit, and an instruction cache [Ju 6/86]. A 64-bit floating point multiply operation takes 2.0 μs. at 10 MHz. This compares with 9.5 μs. for the same operation on an Intel 80287 math coprocessor running at 10 MHz. A large part of this speedup "can be attributed to the fact that the floating-point hardware has been fully integrated with the rest of the processor" [Ju 6/86]. This eliminates the time-consuming interface protocol between the processor and the floating-point unit. The custom processor has a total of 11 bidirectional serial DMA channels. Ten bidirectional channels are for connections to neighboring processing nodes and one is for system I/O [Pa 5/86]. (This is what limits the NCUBE hypercube to 2^{10} nodes.) Each channel has a peak transfer rate of 8M bits/s. in each direction (2M *bytes*/sec overall). Message passing is done via DMA, allowing the processor to resume computing while a message is being transferred. When a DMA transfer is completed, the DMA channel interrupts the processor, so that no processor polling is required to determine which DMA channel just finished [Pa 5/86].

iPSC/2 Node Structure. While NCUBE used a custom processor, Intel took the

opposite approach and constructed a processing node out of off-the-shelf components to make their processing nodes configurable and upgradable [In 10/87]. The computational processor is Intel's most powerful 32-bit microprocessor, the 80386. The companion 80387 numeric coprocessor provides 64-bit IEEE-compatible scalar floating-point precision at 0.3 MFLOP. Using the instruction timing to rate floating-point speed, the iPSC/2-SX takes 0.81 μs. to perform a 64-bit multiply. Four DMA channels are found on each processing board for transfer of large blocks of data [In 88]. Two channels allow for data transfer to other processing nodes through the routing hardware, one channel for each direction. The other two channels are used by software for transfers within the processing node.

Flexibility is achieved in memory configuration, routing hardware, and enhancement options. The memory for the processing node is located on a daughter board which is laid on top of the processing node board. This gives the processing node the ability to expand its memory as technology makes memory cheaper and more compact. The processing node can hold up to 16M bytes of DRAM. Since the Direct-Connect hardware for routing is also on a daughter board, the customer can upgrade the routing capabilities of the node without replacing the entire node. Presently, the only routing daughter board available is the Direct-Connect board.

Each node also provides an iLBX (local bus) interface so that optional boards can be connected directly to the processing node to enhance the node. Presently, Intel has two enhancement boards, the Vector Extension (VX) and Scalar Extension (SX) numerical accelerators. The VX numerical accelerator is a full size board which resides beside the processing node board and is connected directly to the processing node through a ribbon cable. The relationship between the processing node and the VX numerical accelerator is that of tightly coupled (shared memory) processors where the memory located on the VX board can be accessed and used by the processing node as if it were its own memory [In 88]. Its pipelined floating-point processor consists of a floating point adder and multiplier. The addition is done by an Analog Devices Inc. 3220 floating-point multiplier chip, and the multiplication is done by an Analog Devices 3210 floating-point adder chip [El 4/86]. The board also performs 32-bit fixed point and logical functions. The hardware is optimized for matrix multiplication, and floating-point operations are compatible with the IEEE-754 floating-point standard. Theoretically, double-precision floating-point calculations are sped up by two orders of magnitude over the non-VX version of iPSC/2 computer.

The SX numerical accelerator approximately triples the power of the 80387 coprocessor. The SX uses a 1167 Weitek fast floating-point unit and is most effective in heavily scalar computations and computations with short vectors. The SX is not a full-size board, but a daughter board to the node board.

FPS Node Structure. The FPS T Series processing node is called the Vector node and is claimed by the manufacturer to be a high-performance vector computer which has a peak performance of 12 MFLOPS [FP 11/87a]. Figure 7 shows a block diagram of the node structure. The node controller is the T414 Transputer, an integrated microcomputer, which can operate at least 3 times the speed of the Motorola 68020 [Fr 8/86]. The vector processor unit operates at 8 MHz, but to achieve peak performance, data must be aligned on 128-doubleword boundaries [Re –]. The node can contains 1M byte of memory [FP 11/87a]. Data within the node can be transferred either by the control processor or by the vector processor [Re –]. The vector processor is much faster than the control processor at transferring data within the processor, but requires data alignment [Re –].

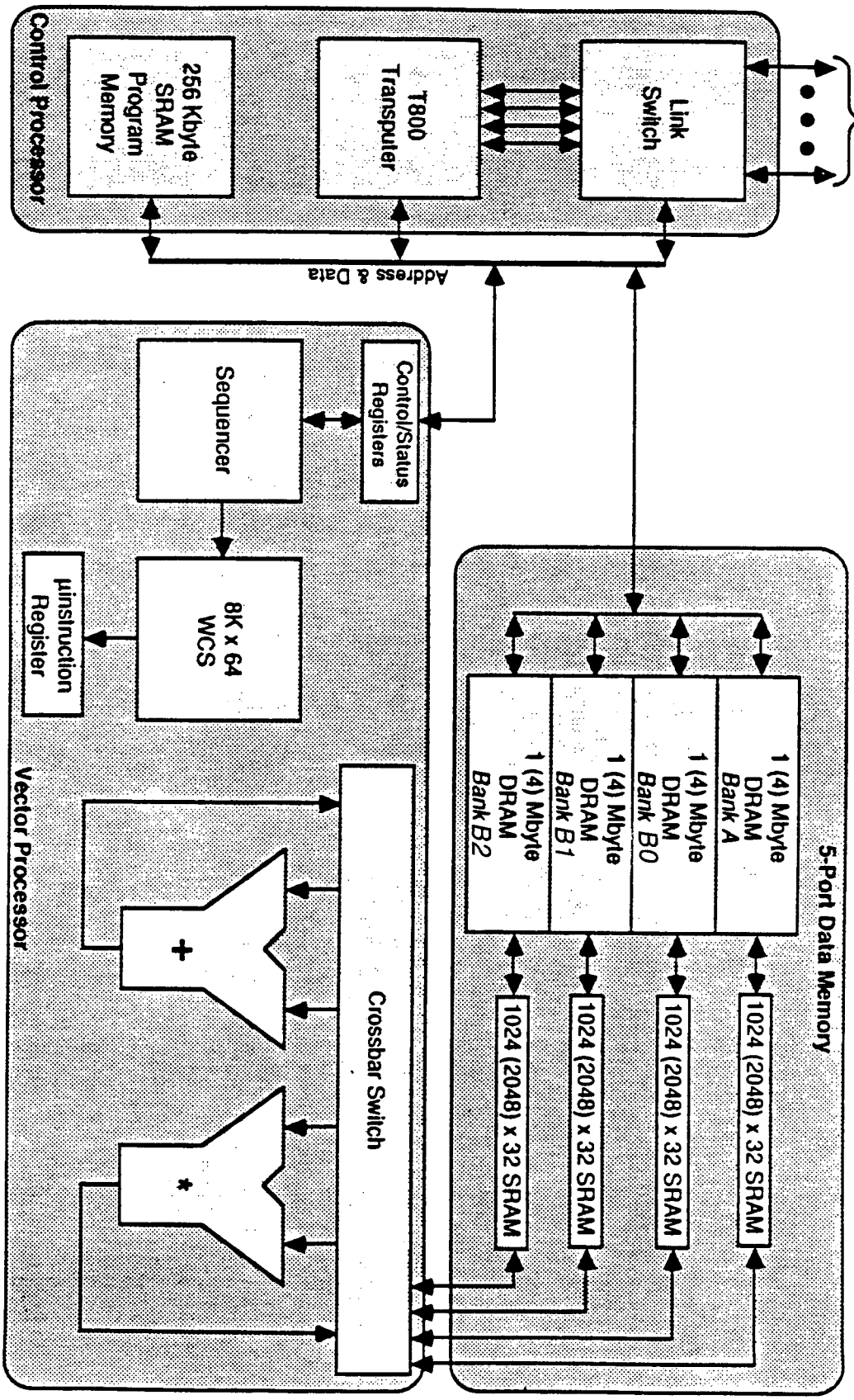

Figure 7: FPS T Series Node Block Diagram [FP 11/87a]

4. Memory structures

4.1. Shared-bus architectures

One problem faced by designers of tightly coupled multiprocessor systems is minimizing the interference between processors sharing main memory. In a shared-bus architecture, contention for the bus and memory can slow down the entire system. Even with a fast bus, reasonably priced dynamic RAM memories are not fast enough to satisfy all processors' requests. A solution to this problem is to place a cache built from fast static RAM between the processor and main memory. The cache intercepts many of the CPU's requests for main memory. Consequently, bus traffic is reduced and more processors can share memory with less contention.

The Multimax, the Balance system and the ELXSI System 6400 have a distributed cache architecture, that is, a cache on each processor. The Alliant system has two types of caches: the computational processor cache (CPC), and the interactive processor cache (IPC). A CPC can be shared by up to four computational elements, and a IPC cache can be shared by up to three interactive processors. Each processor in the Balance system has 8K bytes of local DRAM to store a copy of the most commonly used operating system's kernel to reduce bus traffic further. The ELXSI's register sets serve as a context cache, with the same effect.

Encore's Multimax. The Multimax memory subsystem is made up of several cards, each of which has its own controller. Each controller has four rows of RAM chips. The Multimax system can contain up to eight shared-memory cards. Each memory card has sixteen megabytes of memory organized into two independent banks with four-way interleaving across card pairs, and two-way interleaving across banks on each card. The memory access rate can be as high as 100M bytes per second. In the Multimax system, two processors share a common 32-kilobyte cache.

The memory subsystem and the Nanobus function as a single synchronous unit. The memory cards derive their internal clock signals from the Nanobus, and consequently they run synchronously with the bus [Sc 11/86]. The memory subsystem and the Nanobus have a pipelined interface which improves the speed of the bus operations. Each Nanobus memory controller can accept the next request from the bus while it is still processing the previous one. The new request is buffered, allowing the bus-transfer timing to be as fast as the time needed to load a register [Bi 7/87].

Sequent Balance System. The Balance system can contain up to four memory modules. Each module contains a memory controller board with 2M bytes of RAM and, optionally, a memory expansion board of 2M or 6M bytes. To support the high bandwidth of the system bus, a memory module can respond to a read request in 3 cycles (300 ns), and to a four- or eight-byte request in 2 cycles. If there is a pair of memory modules of equal size, alternate eight-byte address blocks can be interleaved between the two modules. The interleaving of address blocks allows two or more modules to process write or read requests in the same memory area concurrently.

The memory subsystem is pipelined and contains a request queue, a response queue, an error-checking and correction controller, and a system link and interrupt controller. The pipelined operation allows a single memory controller to support a memory transfer rate of eight bytes per 300 nanoseconds [Se 11/86, Th 2/88].

The 8K bytes of local DRAM in each processor hold a copy of the operating system (DYNIX) kernel's interrupt and trap vectors, the module table, the first-level page table, and other frequently used kernel routines. Each processor also has a 8K-byte, 2-way set-associative cache. The data in the cache is organized into 512 rows, each of which has two eight-byte blocks. Bits 3-11 of the data address determine the row in which the data is stored in the cache [Se 11/86, Th 2/88].

Alliant FX/80. The global physical memory in Alliant systems uses 256K dynamic RAMs and is field expandable in 8M-byte increments to a maximum of 64M bytes. Each memory module is four-way interleaved and can supply the full bus bandwidth of 188M bytes per second for sequential read accesses and 80% of this bandwidth for write accesses. The memory modules check the address-bus parity.

Each CPC (see above) is a high-speed global memory buffer for the computational elements. A CPC module can serve as a two-way interleaved 256K-byte cache for up to four computational elements. If eight computational elements are used, two CPCs provide a four-way interleaved 512K-byte cache with a maximum bandwidth of 376M bytes per second. The CPC supports multiple cache accesses in parallel. If one processor has to stop due to a cache miss, the CPC continues to service the requests of the other processors connected to it [Al 10/86]. One of the advantages of the CPC architecture, as opposed to a distributed cache architecture is that data can be passed very quickly among the processors connected to a CPC. This is particularly useful when processing data-dependent loops.

Each IPC is a 32K-byte module which can connect a maximum of three IPs to the memory bus at a bandwidth of 94M bytes per second. The IPCs are capable of providing each IP a continuous bandwidth of 5M bytes per second, for a maximum bandwidth of 60M bytes per second when the system contains 12 IPs. The IPCs block and unblock global memory, maintain maximum bus efficiency, increase interactive processor performance by speeding program code fetches and I/O transfers.

ELXSI System 6400. In the ELXSI system, each memory module on the bus contains 256 megabytes. A maximum of eight memory modules can be connected to the 25-ns. bus, providing up to 2 gigabytes of physical memory in a system. All memory accesses on the ELXSI are virtual; that is, they are relative to a process-specific data structure (the process's page-map table) which bears no particular resemblence to physical memory. Physical addresses are known only to the Memory Manager, which is responsible for the page-map tables, and to the hardware. There are no addressing modes that allow an instruction to access particular locations in physical memory. Thus, a page of data must be in a process's page map table if it is to reference it. Operating system processes are restricted to their address spaces in exactly the same fashion as user processes.

Processes can share memory if shared page is in the page-map table of each process. When created, a process does not share memory with any other process; hence there can be no concurrency control problems due to shared memory. Specific system calls allow two processes to agree to share some portion of their address space. Multiple pieces can be shared, and there is no requirement that they be shared at the same virtual address in each process. The Memory Manager arranges this by placing the same physical memory addresses in each process's page-map table.

Shared memory-management facilities include system calls and instructions to control the residency of data in the cache. Pages of data can be marked as non-cacheable. References to data in such pages always result in a cache fault. The data will always be returned from main memory and be written back to main memory. Such references can be on arbitrary byte boundaries [Ol 5/86, Sh 2/87].

It is the programmer's responsiblity to deal with concurrency control. A variety of hardware and software facilities are provided to assist in this. However, concurrency control is generally expensive, as it requires the use of the bus and physical main memory, defeating the performance advantages of the cache. ELXSI's philosophy, like RISC principles, maintains that hardware should do the simple things and software should avoid doing expensive things. Because accesses to cache are so much faster than accesses to main memory, ELXSI usually encourages users to consider organizing their programs to minimize the number of synchronization

points. However, the system will support even the simplest shared memory models.

4.2. Network Multiprocessors

BBN Butterfly. Each Butterfly processing node has 4M bytes of memory. This memory accepts both local and remote reads, but all reads must pass through the Processing Node Controller. Therefore, if only message passing is used and no remote memory reads/writes are made, the memory is considered unshared and local. On the other hand, if remote memory reads/writes are used, then the memory is considered shared. The time required for a local memory reference on any size Butterfly Parallel Processor is about 500 ns., while the time for a remote memory reference is 5 μs.

NCUBE. Each NCUBE node has a 16-bit data path to 512K bytes of local DRAM. which corrects all single-bit errors and detects any double-bit errors [NC 88]. The program and data for the node are located in the 512K bytes of memory.

NCUBE processors are found on processor boards and I/O boards. A processor board has 64 processors, each with 512K bytes of unshared memory. An I/O board has 16 processors, which also have 512K bytes each; but in addition, a host I/O board has 2M bytes for the use of an Intel 80286 *host* processor. The host processor can access all 4M bytes, and so the memory on a host board is, in a sense, shared memory.

Intel iPSC/2. All memory in the iPSC/2 series computer is local to the processing nodes. Presently, processing nodes can be equipped with 1M to 16M bytes of memory [In 88]. Each node also has a 64K SRAM zero-wait-state cache.

FPS T Series. All the memory in the FPS T Series was local memory. The memory structure of the node is shown in Figure 7. The Transputer contains 1K byte of 50 ns. SRAM [FP 11/87a]. Because the FPS vector processor could only move data on 128-byte doubleword boundaries [Re –], the location of data within the node was much more critical for the FPS T Series than other parallel computers [Fr 8/86]. A unique feature of the FPS T Series was that each module of 8 nodes (arranged in a hypercube) had a separate system board. This system board interfaced the modules to the operating system and also had its own 256M-byte hard disk [Re –, FP 11/87a]. This allowed fast program and data input for any size hypercube.

5. Cache coherence

Caches improve performance in a shared-bus multiprocessor system by decreasing memory access times and diminishing bus traffic. A private cache can be provided for each processor (distributed cache architecture) or a single cache can service a group of processors (shared cache architecture).

However, the use of caches may create a cache coherence problem if multiple copies of main memory locations can exist in different caches. If a CPU changes the value of a shared memory location, it is necessary for all caches that contain that location to either invalidate or update it to contain the modified value. There are two basic types of memory-update policies: write-through and write-back. In a write-through policy, every time a cache's copy of a location is modified by its processor, the new data is immediately transmitted to main memory. In a write-back policy, a cache only updates the local cache copy and delays sending the modified copy to main memory until it is removed from the cache.

The ELXSI System 6400 and the Alliant FX/80 and FX/1 use a write-back policy. While new products by Sequent (the Symmetry) and Encore (the ultramax project's future machine) use a write-back scheme, Sequent's Balance system uses a non-allocating write-through policy, and the current Multimax also uses write-through. Each cache in the Balance system contains a write buffer to keep the processor from waiting for the completion of a bus write cycle. When a processor needs to send modified data to memory, it temporarily stores the data and address in the buffer.

The buffer completes the write operation to main memory independently. For successive writes, the second write waits until the first one is completed.

The Multimax, the Alliant systems and the Balance system contain hardware at each cache to monitor all bus transactions for coherence purposes. This "bus watcher" examines every transaction on the system bus. When a write cycle is detected, it checks the address to see if the location being modified is present in the local cache. If so, the bus watcher makes the local cache invalidate its copy of the data. In the Balance system, if a processor attempts to read invalidated data, it gets a cache miss and fetches the valid copy from main memory [Th 2/88]. Sequent's Symmetry processor [LT 12/87] uses a cache-coherence protocol similar to the Illinois cache coherence scheme [PP 84]. In the Multimax and Alliant systems, if a cache *x* requests some data from main memory and a modified copy of the data is in cache *y*, then cache *y* intercepts the read operation requested by cache *x* and sends a copy of its modified data to both cache *x* and main memory. If cache *x* has requested a writable copy of the data, cache *y* will invalidate its local copy.

In the ELXSI System 6400, processes usually interact via messages, not via shared data. Processes can share read-only data and cannot modify another process's space except by using special system services. Therefore, there are no bus watchers, since a cache does not need to know the contents of the other caches. The caches update main memory when a modified location in the cache is replaced by a new location [Ol 85]. Since the operating system is written entirely without shared memory, the operating system data can be cache resident. In busy systems, this significantly improves the overall performance of the operating system.

Encore Computer Corporation's ultramax project is presently working on a new system (not yet named) which contains up to 128 processors and 2048M bytes of physical memory. Such large configurations are achieved by tightly coupling Multimax systems through a hierarchical cache structure, as is shown in Figures 8 and 9. In Figure 8, references to remote memories go through M_C2, and then are picked off by S_T at the appropriate cluster and sent down to the referenced memory card. Besides the caches that connect each CPU with the Nanobus, another cache level buffers each group of processors from the rest of the system. The second-level caches have to be an order of magnitude larger than the sum of the sizes of all the first-level caches that are connected to them. The idea is that any memory locations for which there are copies in the first-level caches will also have copies in the associated second-level cache [Wi 6/87]. Therefore, the second-level cache (and its associated routing switch) acts as multicache coherence monitor for the first-level caches connected to it.

An example of how multicache coherence control is done in a hierarchical shared bus multiprocessor using the Goodman cache coherence control is given by Drew Wilson in [Wi 6/87]. Other shared-bus cache coherence protocols can also be modified to be implemented in a hierarchical multiprocessor system.

Although the ultramax programming model presents main memory as a large address space residing on the global bus,

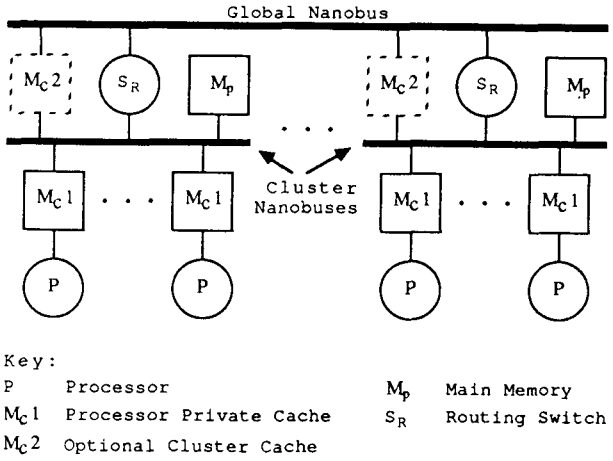

Figure 8: Hierarchical Cache Structure [Wi 7/87]

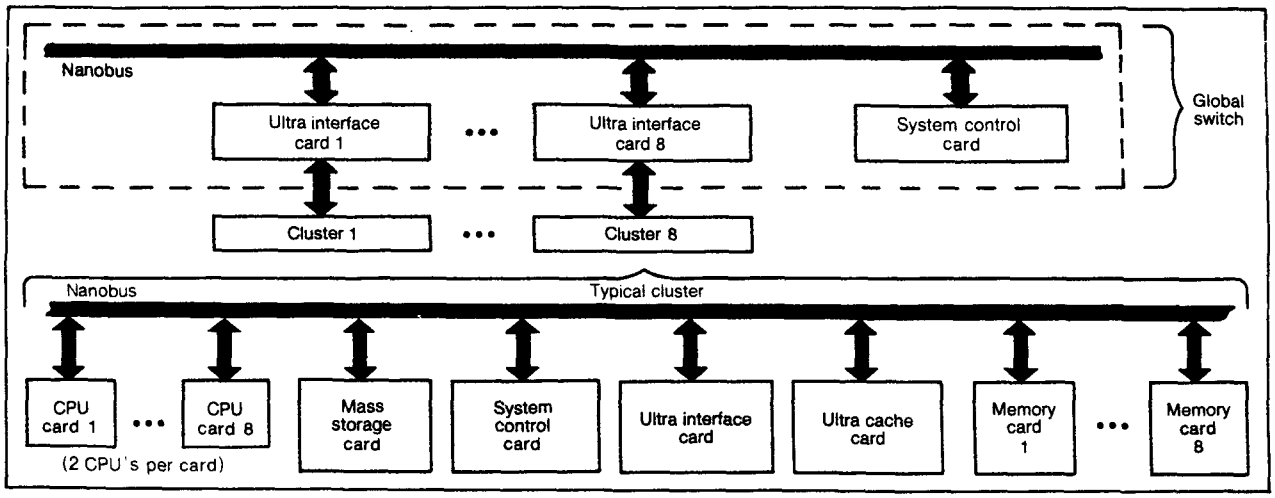

Figure 9: Encore's New Architecture [Wi 7/87]

the memory is physically distributed among groups of processors connected to a Nanobus. Data and code are kept largely local to a particular Nanobus and its processors in order to decrease the global data traffic at the higher levels of the system. Remote requests for local memory are transferred via the local shared bus, using a special adapter board to maintain coherence. This architecture is called cluster architecture since each Nanobus in the system forms a multiprocessor cluster with access to a bank of cluster local memory [Wi 6/87].

The clustered design is implemented by adding two new modules to the commercial Multimax system, the Ultra Interface Card (UIC) and the Ultra Cache Card (UCC). A UIC is connected to each Nanobus and communicates with a corresponding UIC installed in the global bus. The global bus is another Nanobus backplane without processors or memory cards connected to it. The UCC is installed next to the UIC in each local Nanobus and is used to locally cache data from remote Nanobuses [Bi 7/87].

Wilson did some simulation experiments and constructed an analytical model to analyze the performance of medium- and large-scale hierarchically clustered multiprocessors [Wi 6/87]. The results of the simulations showed that a hierarchical multiprocessor computer structure achieves good speedup if suitable parallel algorithms are used. The analytical model indicated that using 64M byte cluster caches and 13 MIPS processors, which will be available in the near future, it will be possible to construct a shared memory multiprocessor of over 1000 MIPS.

6. Processor Communication and Synchronization

Synchronization is necessary for multiprocessor systems to execute parallel operations and to insure that no more that one process has write access to a sharable resource at the same time. Most multiprocessor systems use locks and semaphores for this purpose. In some systems such as Encore and the Balance, synchronization is done by the hardware, while in the ELXSI System 6400 it is done by software. Interprocessor communication is necessary in several situations:

- to tell a processor to remove itself from service,
- to initiate low-priority interrupt services, or
- to deliver a software signal.

In the Alliant and Balance systems, processes communicate via interrupts; in the ELXSI System 6400 they use messages.

6.1. Shared-bus architectures

Encore's Multimax. The Multimax uses semaphores to synchronize processors' accesses to shared resources. The Nanobus memories implement atomic test-and-set operations at the memory-chip level. The memory chip executes the atomic test-and-set function without locking the bus or the memory controller, and also without restricting the software in the number, location, or use of semaphores, each of which occupy a byte [Sc 11/86].

The Sequent Balance System. The Balance System synchronizes processors by using the System Link and Interrupt Controller (SLIC). The SLIC performs this function with a set of 64 single-bit gates. These gates are logically equivalent to a test-and-set primitive. The CPUs use the SLIC *Request Gate* and *Release Gate* operations to insure that only one CPU at a time has access to a shared data structure. A gate is acquired via a single atomic operation. When a gate is acquired by a CPU, a SLIC first verifies that the gate is free, then sends a *Request Gate* command on the SLIC bus. The successful transmission of *Request Gate* insures exclusive ownership of the gate [Be 10/87, Th 2/88].

At any time, each gate is either in a free or in an occupied state. When a process needs to use a gate, it loops requesting the gate until it acquires it. Since each SLIC keeps track of the status of each gate, the busy-waiting is done checking the status register of the local SLIC, not across the SLIC bus.

The DYNIX operating system uses three kinds of mutual exclusion primitives built from gates: the direct use of the gates, spin locks, and counting semaphores. The direct use of gates is the fastest, but this is only employed in the most time-critical sections, since there are only 64 gates. A lock is a byte in memory which can be in one of two states: locked or unlocked. A single gate can synchronize accesses to many locks, and therefore, there can be an unlimited number of locks. When a processor needs to access a shared data structure, it has to wait until the associated lock is unlocked. Then it changes the lock state to locked, to indicate other CPUs that the data structure is in use. Locks are used to protect data structures for a short amount of time, because a CPU can not perform any other operations while another CPU is using a lock that it needs. The Counting Semaphores are the highest level mutual exclusion primitive and are used to block waiting for an event, or to guard a very long critical region. When a process tries to acquire a semaphore which is being used, the process is placed on a waiting queue, associated with the semaphore, and the CPU is freed to be used by another process. When a semaphore is released, the first processor in the waiting list is given access to the critical region.

Communication between processors is implemented with programmed interrupts via the SLIC. A processor can send another processor a normal maskable interrupt, a non-maskable interrupt, or a software interrupt.

Alliant FX/80. Synchronization between processors in the Alliant FX/80 is provided by the Concentrix operating system, which uses priority-level locking and a hierarchy of global test-and-set locks. Global test-and-set locking is used to manage multiprocessor interactions. Each global lock consists of an access location, a processor tag, a priority identifier, and a recursion counter. The access location is used for atomic test-and-set operations by processors trying to access a lock. The processor tag holds the identification number of the processor which currently owns the lock. The priority identifier is used to implement a linear ordering of the types of sharable resources for deadlock avoidance. The recursion count keeps track of how many times a lock has been acquired recursively.

Concentrix keeps track of the lock states of processes by using a lock-stack per process. Locks are pushed into the stack as they are acquired by processes and popped from the stack as they are released by processes. Each process's lock stack is stored

in its kernel user-area and is preserved through process sleeps. The locks owned by a process are released when it goes to sleep and are re-acquired when it wakes up.

Processes in the Alliant architecture communicate via a cross-processor-interrupt (CPI) facility. The CPI allows any processor in the system to interrupt any other process in the system. A CPI can be directed to a single processor or to a group of processors through a selective broadcast. Concentrix uses the CPI facility to activate remote procedure calls (RPC) on other processors, to initiate remote asynchronous system traps on other processors, and to synchronize the processors in the system. Remote procedure calls are mainly used to activate routines that modify remote processor or device states. RPCs can be asynchronous or synchronous and are implemented using a global mailbox facility to pass arguments. Asynchronous RPCs suspend the calling processor until all target processors have sent a software interrupt to perform the RPC. Synchronous RPCs suspend the calling processor until all the target processors have executed the RPC [Te –].

The most powerful feature of the Alliant architecture is the ability to use multiple computational elements concurrently in the execution of a user application in a way that is transparent to the user. This is called *Alliant Concurrency* and is controlled at execution time by the Concurrency Control Unit, an 8000-gate CMOS array that connects the processors through a concurrency-control bus. The concurrency-control bus provides an independent communication path between the CCUs. Concentrix coordinates the CE (computational element) complex during the execution of concurrent processes.

Alliant concurrency uses the program loop control as the source of parallel instruction streams. At compilation time, the FX/Fortran compiler inserts special concurrency control instructions whenever it detects loops which can be executed in parallel by multiple CEs.

Initially, code is executed serially by one computational element while the others wait. Concurrent execution starts when the active CE reaches a concurrency control instruction inserted by the FX/Fortran compiler. At that point, the complex assigns to each CE a value for the loop index according to a global counter. Concurrency-control data is transmitted via the concurrency-control bus. When a CE completes an iteration of the loop, the complex assigns it another index value, and it restarts the loop. When all the iterations are finished, the CEs go idle except for the last CE executing a loop iteration, which continues executing sequentially [Pe 5/86, Al 10/86].

The following example shows how the Alliant executes a do loop containing data dependencies [Pe 5/86]:

```
        X(0)  = 0
        DO 12 I = 0, N
        Y(I) = SIN(A(I))*COS(B(I)) + C(I)
        Y(I) = (Y(I)-C(I))/B(I)
12      X(I+1) = X(I)+Y(I)
```

CE0	CE1	CE2
sequential code	idle	idle
Y(0) = SIN(A(0)) etc.	Y(1) = SIN(A(1) etc.)	Y(2) = SIN(A(2) etc.)
Y(0) = (Y(0) etc.)	Y(1) = (Y(1) etc.)	Y(2) = (Y(2) etc.)
X(1) = X(0) + Y(0)	stall	stall
Y(3) = SIN(A(3) etc.	X(2) = X(1) + Y(1)	stall
Y(3) = Y(3) etc.	Y(4) = SIN(A(4) etc.)	X(3) = X(2) + Y(2)
X(4) = X(3) + Y(3)	Y(4) = (Y(4) etc.	Y(5) = SIN(A(5) etc.)
...	X(5) = X(4) + Y(4)	Y(5) = (Y(5) etc.)
	...	X(6) = X(5) + Y(5)
		...

Figure 10: A DO Loop Containing Dependencies as Executed on the Alliant

The above code runs in parallel over multiple computational elements on the Alliant as shown in Figure 10.

The instruction streams of the different CEs are offset by initially stalling the streams on CE1 and CE2 until `X(I)` is calculated by the CE executing the previous iteration. Besides synchronizing loops with data dependencies, Alliant concurrency can execute loops that contain conditional branches, loop exits, and potential feedback.

In order to manage a process with multiple instruction streams, Concentrix replicates a number of kernel data structures, such as kernel stacks and processor-control blocks. The number of replicated kernel structures depends on the number of CEs needed to execute the code streams. Sequential processes only need one kernel stack, while concurrent processes on a four-CE complex need four kernel stacks.

ELXSI System 6400. The System 6400 uses a message-based system to assure that only one process at a time has access to a particular instance of shared resource. When a resource is allocated to a processor, all state information relative to that resource is kept in the process's private address space, and is accessible to another process only via requests in message form. The system does not have supervisor mode, since it does not need supervisor calls to allow a processor to write or directly read state information of a resource belonging to another process.

The operating system provides spin locks, binary and counting semaphores, and Sleep and Wakeup services to synchronize multiple processes. These functions insure an atomic exchange between a register and a memory location in the requesting process's address space. The Sleep and Wakeup services queue and dequeue processes, allowing a CPU to service another process while a semaphore is locked.

As noted above, ELXSI's System 6400 processors communicate via messages. In data communications terminology, ELXSI processes are interconnected via explicit virtual circuits [Ol 5/86]. This means that a point-to-point connection has to be established before two processes can communicate. Once the connection is established, many messages can be transmitted in either direction. Each connection coming out of a process is called a link, and points to a queue of messages called a "funnel." Many links may point into a funnel. The messages on a determined funnel are sent to the consumer in FIFO order . The consumer is able a to receive messages from a specific funnel, a set of funnels, or all funnels. There are a variety of instructions to send and receive messages and links and to manipulate the communication structure [Ol 85].

I/O is also done through messages. When a message is sent to the I/O subsystem, the receiving process is an I/O controller. The microcode is in charge of routing messages, including keeping track of the migration of a process to another CPU or the transmission of a message to an I/O controller. The links are capabilities which must be created by the process that performs the operation and must be sent to the process which wants to execute the operation [Ol 85, Ol 5/86, Sh 2/87].

The operating system consists of multiple object-manager processes which communicate only through messages. Functions such as memory management, process and CPU scheduling, and facilities to allow a process to create other processes, are performed by the "System Foundation." The System Foundation consists of a series of processes which provide the mentioned services to all processes and also to all operating systems which run on the computer [Ol 85, Ol 5/86, Sh 2/87].

6.2. Communication in Network Architectures

A shared-bus architecture has no need to make routing decisions; all processor-memory references simply traverse the shared bus. Network multiprocessors use

various strategies to route messages from source to destination.

BBN Butterfly. The Butterfly switch operates much like a packet-switching network. Figure 11 [BB 3/86] shows how a packet is sent through the switch. The message is being sent from processing node 5 to processing node 14. As the message passes through each switching node, two bits of the address packet are used to decide which of the node's four outputs the message should be directed to. A switching node can send messages through all four outputs at the same time. Should two messages require the same output, however, one of the messages will be delayed by the switching node until the other message has been sent. An example of a remote memory read operation follows:

> "When the MC68000 makes a read reference, its local Processor Node Controller gains control and uses its memory management hardware to transform the supplied virtual address into a physical address, which corresponds to memory on another Processor Node. To read the referenced location, the PNC sends a packet addressed to the remote Processor Node through the switch requesting the contents of that physical memory address. The remote PNC receives the packet, reads the referenced memory location, and sends a reply packet containing the value through the switch back to the source Processor Node. When it receives the reply, the source PNC satisfies the MC68000's read request with the value obtained from the reply." [BB 3/86]

After this document was released, BBN replaced the MC68000 by the MC68020. The round-trip time for a remote memory reference is about six microseconds for any size Butterfly switching network [BB 87]. Each path through the switch has a peak performance of 32M bit/s. The Butterfly switch can also transfer blocks at the full 32M bit/s. rate of the switch.

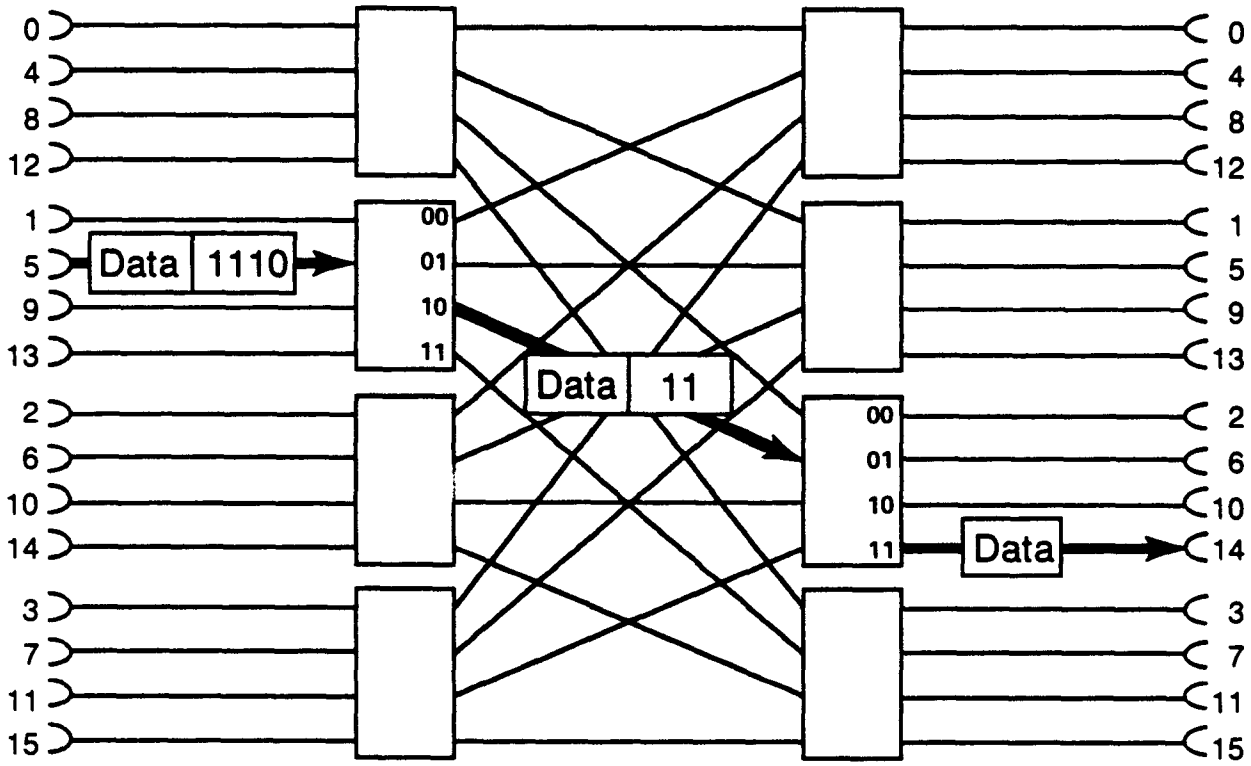

Figure 11: Operation of Butterfly Switch [BB 3/86]

NCUBE. In a hypercube, each node must be able to communicate bidirectionally with several other nodes (the number of such nodes equals the dimension of the hypercube) and pass along messages whose destinations are at other nodes. As mentioned in Section 3, each NCUBE processing node has 11 bidirectional synchronous bit-serial DMA channels—connected to ten neighboring nodes and one I/O board. The internode channels include parity checking and run at 8 MHz for a rate of 1M byte/s in each direction, full duplex [Pa 5/86]. The 11 DMA channels are on the custom VLSI processor chip. Each channel has two write-only 32-bit registers: one for the address of the message buffer and one for the number of bytes left to be sent or received. Each channel can interrupt the processor upon completion, or the processor can poll each channel's ready flag. If the destination node is more than one node away, the node's system kernel (VERTEX) routes the message by forwarding it.

Intel iPSC. A major shortcoming in the original iPSC series was the time required to transmit message whose destination was several nodes removed from its source. The iPSC/2 addresses this problem with a communication daughter board called a Direct-Connect routing module. This routing hardware creates a communication path to the destination node at a cost of "a few microseconds per node in the path" [In 88]. Once the communication path is created, the message is transferred at a rate of 2.8M bytes/s. without the interaction of either the source, destination, or intermediate node processors. The typical time for a message transmit and acknowledge receive is about 290 μs. Intel claims that its iPSC/2 series can send short messages three times faster and long messages ten times faster than the iPSC/1 [In 10/87]. Because message delays are all about the same, process and data placement has little influence on program speed, which "makes programming easier."

FPS T Series. The FPS T Series node had 15 channels, of which 14 were allotted for communicating with neighboring nodes, and one was connected to the system board. The 15 channels were multiplexed from the four synchronous I/O channels of the Transputer by a switching network (see Figure 7); each Transputer channel could connect to any node channel. To change one connection in the switch (i.e., to point one of the four Transputer I/O channels to a different node's I/O channel) required 1 ms. [Re –]. As a result, communication between nodes was time consuming, and an internode data transfer of 256 words required 21 times the time needed for a vector add operation.

Communication between T Series nodes was coordinated by the control processor (Section 3) rather than by a separate communication processor. Therefore, messages to non-adjacent nodes were divided into packets and forwarded from one Transputer to another. Each Transputer along the path had to be interrupted to forward each packet; however, given the Transputer's very fast context-switch time of 2.5–6.25 μs., the store-and-forward overhead was not extremely large.

7. Power and performance

Encore's Multimax. The original Multimax offered 1.5 to 15 MIPS. The current Multimax 320 can contain from 2 to 20 National 32332 processors, providing 4 to 40 MIPS of computing power. An APC-based Multimax with Wytek floating-point accelerator can achieve 1.57 DP Whetstones per processor.

Sequent's Balance System. The Balance 8000 system can contain two to twelve National Semiconductor 32032 processors with an aggregate performance of 1.5 to 8.4 MIPS. The Balance 21000 system contains four to thirty NS32032 processors with an aggregate performance of 3 to 21 MIPS.

The Balance system was measured by comparing each CPU with other microprocessors using the Dhrystone benchmark. The results show that the performance of each CPU in a Balance system is about 1.4 times the performance of the VAX 11/750 for a single-stream CPU-bound integer appli-

cation, or approximately 0.8 MIPS [Th 2/88].

Another measure of Balance system performance is the speedup of a single application program using multiple processors. In one of the applications, Linpack, the parallelization of a standard floating-point program, resulted in improvement by a factor of 27.4 with a 30-processor system [Th 2/88].

Studies have also been done for the Balance 8000 cache and bus protocols. The single-thread cache performance demonstrated that the cache can achieve a hit rate of 95% in integer applications. This rating results from the high locality in the applications and the 8-byte line size, which allows implicit prefetching of instructions and 32-bit data. The double-precision floating-point applications attained a cache hit rate of only 85%, because accessing 64-bit-word data breaks the implicit prefetching strategy [Th 2/88].

A study of Multibus utilization for multithread applications showed that the bus utilization is under 25% for an eight-processor system. The multiuser benchmark showed that the bus was less of a limiting factor, and that potentially the number of processors used in the Balance 21000 can be increased with a write-back cache-coherence protocol if the I/O capability is extended [Be 10/87].

Alliant FX/series. The Alliant architecture supports up to twenty processors, in two categories: computational elements and interactive processors. The computational elements are 14.9M-Whetstone single-precision (32-bit) and 13.9M-Whetstone double-precision (64-bit) Motorola 68020 general-purpose microprocessors. In vector mode, each CE executes floating-point instructions at a peak rate of 23.6 MFLOPS in single precision and 11.8 MFLOPS in double precision. The interactive processor module is a VME card which contains a Motorola microprocessor, a virtual-memory address-translation unit, an I/O map, local parity-protected RAM, power-up EPROMs, and two serial ports.

By the Whetstone benchmark, the Alliant CE is about five times the VAX 8600 or twelve times a VAX 11/780 in double precision.

The following application shows the advantages that detached CEs have in computer environments where the application mix is dynamic and consists of a large number of codes. While multiple copies of a NASA fluid dynamics code, FLO22, were being run in the background, on a FX/80 system with eight CEs, the time to perform an additional copy was measured. With up to seven copies running in the background, the eighth job takes only slightly longer than on an unloaded system [TM –].

ELXSI System 6400. The System 6400 contains 1 to 12 CPUs which are implemented with ECL gate-array technology. On a single 6420 CPU (the medium-speed processor), the Livermore kernels execute at 1.1 megaflops (harmonic mean of 24 kernels, vector length 167, 64-bit floating point), and on a 12-processor system they execute at 13 megaflops. The Cray XMP-1 achieved 8.1 megaflops on the same benchmark. The company recently announced their high-performance 6460 CPU, which will perform from three to six times faster than the 6420 CPU, depending on the application. Each 6460 CPU will run at 25 times the speed of the VAX 11/780 on run-of-the-mill scalar code. In floating-point intensive applications the 6460 CPU will be even faster than that.

John Sanguinetti ran a series of experiments to find out whether the ELXSI System 6400 could achieve, or maybe exceed, linear performance improvement. The workloads he used in the experiments were multiple-job workloads, like the ones used in general-purpose scientific applications. The results of the experiments showed that as CPUs are added, the power of the machine grows linearly [Sa 9/86].

Studies showed that the message system is fast compared to many software-controlled interprocess communication mechanisms. To send a message from one process to another, including process switch

and receipt by the second processor, takes approximately 115 ns. plus 450 μs. per byte on the 6420 CPU [Ol 5/86]. (The 6460 CPU should reduce these times by about half.) The message-system overhead in three real time-sharing workloads was measured to be 2.5, 0.3 and 4.7 percent of the total CPU cycles. The time included some of the tasks performed by the kernel in a typical time-sharing operating system . Experiments done with different configurations indicate that the proportion of CPU cycles consumed by the message system is not affected by the number of CPUs. The effective bandwidth of the message system is around 2M bytes per second, which is not adequate for bulk data transfer. Therefore, the I/O controllers use DMA for block transfers but still use messages to indicate the completion of a task [Ol 5/86].

Jack J. Dongarra compared the performance of about 100 computer systems while solving dense systems of linear equations using the Linpack software in a Fortran environment [Do 3/88]. The following data was obtained using Linpack to solve a system of linear equations of order 100. In the tables below, "BLAS" means "basic linear-algebra subprograms," "coded BLAS" refers to the use of assembly language coding of the BLAS, and "rolled BLAS" refers to a Fortran version with single statement and simple loops [Do 3/88].

"Ratio" is the number of times faster or slower a particular computer configuration is when compared to the CRAY-1S using a Fortran coding for the BLAS in full precision [Do 3/88].

- Solving a system of linear equations with LINPACK in full precision using all Fortran:

Computer	OS/Compiler	Ratio	MFLOPS	Time (secs.)
CRAY-1S	CFT (Rolled BLAS)	1	12	0.056
Alliant FX/80 (8 CEs)	FX Fortran v3.1.33 (Rolled BLAS)	1.4	8.5	0.0805
Alliant FX/1 (1CE)	FX Fortran v3.1.33 (Rolled BLAS)	7.5	1.6	0.572
ELXSI 6420	Unix 5.3, f77-Oskm	6.9	1.8	0.385
ELXSI 6420	Fortran 5.14, opt=10	9.2	1.3	0.516
Encore Multimax (w/FPA)	f77	52	0.24	2.9*
Sequent Balance 8000	DYNIX Fortran 2.4.4	208	0.059	11.7

- Solving a system of linear equations with LINPACK in Full Precision using Coded BLAS:

Computer	OS/Compiler	Ratio	MFLOPS	Time (secs.)
Cray-1S	CFT (Coded BLAS)	0.54	23	0.030
Alliant FX/80 (8 CEs)[1]	FX Fortran v3.1.33 (Coded BLAS)	1.1	10.9	0.0631
Sequent Balance 8000	DYNIX Fortran 2.4.4 (Coded BLAS)	185	0.066	10.4

[1]The Dongarra benchmarks measured the Alliant FX/8; Alliant has updated the measurements for the FX/80.

- Solving a system of Linear Equations with LINPACK in Half Precision using all Fortran:

Computer	OS/Compiler	Ratio	MFLOPS	Time (secs.)
Alliant FX/80 (8 CEs)	FX Fortran v3.1.33 (Rolled BLAS)	1.2	10.6	0.0649
Alliant FX/1 (1CE)	FX Fortran v3.1.33 (Rolled BLAS)	8.0	1.5	0.465
ELXSI 6420	Fortran 5.14, opt=10 (Coded BLAS)	6.1	2.0	0.342
Encore Multimax (w/FPA)	f77	36	0.34	2.0*
Sequent Balance 8000	DYNIX Fortran 2.4.4	162	0.075	9.10

- Solving a System of Linear Equations with LINPACK in Half Precision using coded BLAS:

Computer	OS/Compiler	Ratio	MFLOPS	Time (secs.)
Alliant FX/80 (8 CEs)	FX Fortran v3.1.33 (Rolled BLAS)	.86	14.0	0.070
Alliant FX/1 (1 CE)	FX Fortran v3.1.33 (Rolled BLAS)	7.0	1.7	0.340
ELXSI	FTN MOD 2 (Coded BLAS)	7.5	1.6	0.418
Sequent Balance 8000	DYNIX Fortran 2.4.4 (Coded BLAS)	148	0.083	8.31

Further results can be found in [Do 3/88]; note especially Table 7 in that paper.

Butterfly Parallel Processor. The Butterfly uses 16-MHz Motorola 68020/68881 processors with floating-point hardware. Each node has a peak performance of 2.5 MIPS; thus the largest configuration of 256 nodes has a peak performance in excess of 600 MIPS. The results of several application programs show that a 256-node Butterfly Parallel Processor can achieve a speedup of 190 to 230 over the performance of a single node [Re 12/86].

NCUBE. NCUBE's processing node and I/O processor both consist of the same custom 160,000 transistor one-chip CPU. The CPU operates at 8 MHz and executes non-math instructions at 2 MIPS, single-precision operations at 0.5 MFLOPS, and double-precision operations at 0.3 MFLOPS [Ju 6/86]. For the 1024-node NCUBE/ten, this means a peak performance of 500 MFLOPS or 2000 MIPS. In benchmark tests running Fortran Dhrystone code, the NCUBE VLSI processor (8 MHz) was compared with the Intel 80286/80287 (8 MHz) and the VAX-11/780 with the floating-point accelerator [Ha 10/86]. The results in Dhrystones/s. were 1249 for NCUBE, 510 for 80286/80287, and 741 for 11/780. The NCUBE processor did well in Benchmark tests using Fortran Whetstones, too. The Whetstone code simulated scientific applications with many double-precision floating-point operations. The results in kWhetstones/s are 476 for NCUBE, 101 for the 286, and 426 for 11/780. Unfortunately, benchmarks for entire systems are not available.

Intel iPSC/2. The control processor of the iPSC/2 series processing node is the 32-bit 80386 processor [In 10/87]. This processor, along with a new node operating system called NX/2, provides 3 to 5 times the performance of the original iPSC. The 80387 also provides a five-fold performance increase over its predecessor, the 80287, used on the first generation iPSC. The peak performance for a 64-node Basic System (without numerical accelerators) is 16 MFLOPS for 32-bit precision and 13 MFLOPS for 64-bit precision [In 88]. The peak performance for a 64 node VX system (vector accelerator) is 1280 MFLOPS for 32-bit precision and 424 MFLOPS for 64-bit precision [In 88]. For both systems, the peak instruction execution rate is 256 MIPS. The iPSC/2 performs 8064 Dhrystones (1.1) per second; 1,331K Whetstones/sec. with the 80387 option, and 2,192K Whetstones/sec. with the SX (1167) option. The 2D wave equation is solved on a 64-node iPSC/2-VX at 454 megaflops with 32-bit precision.

* Per processor.

Company/product	Price	Number of processors	Processor type	Memory	Architecture	Peak performance	Applications
Alliant FX series	$59,900–$1 million	1–8	custom CMOS	shared	bus	188 MFLOPS 119 MIPS	Scientific/ engineering
BBN Butterfly	$11,000 per node	2–256	68020 with custom coprocessor	shared	switch network	600 MIPS	Scientific/ engineering, AI
ELXSI 6400	$295,000	1–12	custom	shared & local cache	bus	7–156 MIPS	Aerospace/defense simulation; real time
Encore Multimax	Starts at $100,000	2–20	32332	shared & local	bus	40 MIPS	Technical R&D; software development; commercial
Floating Point T-Series		8–16,384	Transputer	local	hypercube	16 MFLOPS & 7.5 MIPS/processor	High-end scientific/ engineering; AI
Intel iPSC/2-VX	$165,000–$3,000,000	16, 32, 64, 128	80386 & 80387; 1167 & 3210 & 3320	local	hypercube	2560 MFLOPS (32-bit) 848 MFLOPS (64 bits) 512 MIPS	High-end scientific/ engineering; AI
NCube	$10,000–$1.5 million	4–1024	custom	shared & local	hypercube	500 MFLOPS, 2000 MIPS	Scientific/engineering; database; AI
Sequent Balance 2100	$49,000 $500,000	2–30	32032	shared & local	bus	2.8–21 MIPS	Scientific/engineering; commercial; database

Table 1: Comparison of Various Multiprocessors [HT 2/87]

The Linpack benchmark uses the Gaussian Elimination algorithm for matrix factorization and requires high communication between nodes and long messages [In 88]. Running this benchmark, the 32-node iPSC/2 VX achieved a performance of 55 MFLOPS, the 64-node iPSC/2 VX achieved a performance of 86 MFLOPS, while the Cray XMP achieved a performance of 100 MFLOPS. A 2-dimensional Fast Fourier Transform, was run on the 32 node and 64 node iPSC/2 VX computers. The FFT requires row and column data exchanges; thus each node must communicate with other nodes at about the same time. Running this benchmark on a 1024×1024 matrix, the 32-node iPSC/2 VX achieved a performance of 154 MFLOPS, the 64-node iPSC/2 VX achieved a performance of 158 MFLOPS, while the Cray XMP achieved a performance of 100.3 MFLOPS[In 88].

FPS T Series. Two processors were used in FPS T Series nodes, the Transputer (control processor) and a vector processor. The Transputer operates at 16 MHz and "as a 32-bit processor ... is at least 3 times the speed of the Motorola 68020" [Fr 8/86]. The vector processor was a Weitek VLSI floating-point chip which operates at 8 MHz [Fr 8/86, FP 11/87a]. The two chips combined to give the processing node a peak performance of 12 MFLOPS and 8 MIPS [FP 11/87a].

FPS lists benchmarks for processing done on the T100 [FP 11/87b]. The T100 contains 64 processing nodes and had a peak performance of 1168 MFLOPS. Benchmarks were programmed in three languages: Occam, C, and Fortran. For 3D N-body simulation, the results for the three languages respectively were: 268, 269, and 171 MFLOPS. For 2D convolution using a SD mesh topology, the results were: 579, 607, and 381 MFLOPS. Several other benchmarks are listed in [FP 11/87b].

8. Price

Price information was not available on all computers. Relative prices for many computers are shown in Table 1. Some specific price calculations are discussed below:

Alliant FX Series. The FX/1 has a base price of $59,900, including one CE, one IP, 32M bytes of main memory, one disk drive and a cart tape drive. The FX/4 has a base price of $99,900 and includes one CE, one IP, 32M bytes of main memory, a 256K-byte CE cache, one IP cache, one disk drive, and cart tape. The FX/80 has a base price of $299,000, and includes sixteen terminal lines, two CEs, two IPs, 32M bytes of main memory, one disk drive and a 50 ips. tridensity tape drive. All systems include operating system, Fortran, and parallel scientific and mathematical libraries.

Butterfly Parallel Processor. The base cost of the Butterfly Parallel Processor is approximately $11,000 per node, and the computer can be expanded one node at at time (except for the switching network). This price includes switch hardware and cabinetry, but is exclusive of I/O devices. A 128-node Butterfly GP1000 including 500 MBytes of disk storage and UNIX™ software with a peak performance of approximately 300 MIPS would cost $1.8 million. The cost per MIP of such a system is $6000.

ELXSI System 6400. Prices start at $295,000 for a complete packaged entry system, including peripherals and software. A packaged configuration with ten 25-MIP CPUs, 512M bytes of memory, 5G-byte disk, a couple of operating systems and other software, plus the usual peripherals, has a list price of $3,999,000. ELXSI offers a "very attractive" grant program for Universities interested in doing research in cooperation with ELXSI.

NCUBE. The price for a complete NCUBE/ten system is $1.5 million. Assuming a peak performance of 500 MFLOPS and 2000 MIPS, the performance prices are $3000/MFLOP and $750/MIP. An application article mentioned the price of a NCUBE/six (64 nodes) as around $200,000 [Ma 2/87]. Assuming a peak performance of 31 MFLOPS and 125 MIPS, the

performance prices for the NCUBE/six are $6500/MFLOPS and $1600/MIPS [HT 2/87].

Intel iPSC/2. The least expensive iPSC/2 system is the 16-node Basic System with 1M byte of memory per node at a total cost of $165,000 [In 88]. A 32-node version of the iPSC/2 VX system with 8M bytes of memory on each node costs $796,000, and the 64-node version costs $1,572,000. Each development station costs $20,000, and an iPSC/2 simulator runs $495. All system and development stations include extensive software packages. Using the peak performance MFLOPS, the 32-node VX system costs $4700/MFLOP (64-bit precision) and the 64-node system costs $5700/MFLOP (64-bit precision). Using the 2-D FFT benchmark performance results, the 32-node iPSC/2 VX system costs $5200/MFLOP and the 64-node version costs $9900/MFLOP [In 88].

Sequent Symmetry. An entry-level configuration includes two Intel 80386 processors, 8M bytes of ECC RAM, 150M byte SCSI disk, a cartridge tape drive, 16 asynchronous ports, the DYNIX operating system and C compiler, Ethernet interface and TCP/IP software, and sells for $89,500. A typical large system includes 30 Intel 80386 processors, 80M bytes RAM, 4.3 gigabytes of SMD disk storage, a 6250 bpi tape drive, 64 asynchronous ports, the DYNIX operating system and C compiler, Ethernet interface and TCP/IP software, with a price of $870,000. Sequent Balance systems range in price from $49,500–$500,000.

FPS T Series. The FPS T series machines are no longer being sold.

9. Summary

This paper has focused on eight commercial parallel processors, four shared-bus machines and four network multiprocessors. The shared-bus architectures studied in this paper were the Encore Multimax, the Sequent Balance System, the Alliant FX series, and the ELXSI System 6400.

One problem with a shared-bus architecture is the degradation of the performance as multiple processors compete for access to the bus and memory space. Different approaches have been taken by multiprocessor systems to reduce the traffic on the system bus. The Multimax system divides the global bus into three independent buses: one for addresses, one for data and one for vectors. The Balance system has a one-bit data path called the System Link and Interrupt Controller Bus which interconnects all major components in the system and allows them to exchange interrupts and other low-level control signals, and error information independently of the system bus. Its processors have a private memory which holds the most commonly used kernel routines to further reduce bus traffic. The Alliant system divides the system bus into two data buses, and an address bus. It also contains a concurrency-control bus which allows the computational elements to exchange data when performing concurrent operations. The ELXSI System 6400 caches non-shared data at each processor; since operating-system processes do not share memory, and user processes tend to use shared memory sparingly due to its expense, bus traffic is reasonably low.

Memory access time can be reduced by the use of cache memories. They can considerably reduce the bus traffic, because they allow most memory references to be satisfied without a bus transaction. However, the use of caches may create a cache coherence problem if multiple copies of main memory locations can be stored in different caches. The Multimax system, the ELXSI System 6400 and the Alliant system use a write-back update policy, and the Balance system uses a write-through policy.

So far, no commercially available shared-bus computer has had more than 30 processors, owing to bus-bandwidth limitations. Encore is presently testing a new shared-bus system with up to 128 processors. This is done by tightly coupling Multimax systems through a hierarchical cache structure.

Among network architectures, this paper has discussed the BBN Butterfly Parallel processor which uses a "butterfly" switching network for interprocessor communication. The Butterfly switch allows communication via message passing or shared memory. "Shared-memory" communication is transparent to the program because all data access to the memory of other nodes is performed by a control processor. Some other advantages of the Butterfly switch are that it is expandible by one node at a time, and that it provides redundant paths to decrease contention. One disadvantage of the Butterfly switch is that data must be distributed uniformly over the processing nodes to decrease contention. In many applications, the Butterfly switching network has achieved more than 80% of its peak performance. The cost of the Butterfly computer is higher per node than most other parallel processing computers, but the performance and flexibility may offset the price.

The other network architectures discussed have been three hypercubes: the NCUBE, the Intel iPSC/2, and the FPS T Series. Although hypercube interconnections are well suited to many scientific and engineering problems, current hypercubes restrict the computer to unshared memory and therefore message passing. The major problem a hypercube design must solve is transferring messages between nodes which are not neighbors. Both the NCUBE nodes and the FPS nodes use the "store-and-forward" approach. The Intel iPSC/2, however, has a Direct-Connect routing board which creates a channel between any two nodes in the hypercube, and then transfers the message packet at 2.8M bytes/s. This hardware resulted in a ten-fold speedup in the transfer of long messages.

To be attractive to engineering and scientific applications, the hypercubes include special numerical computation hardware. The Intel iPSC/2 offers a scalar numerical accelerator board which increases the power of the floating point coprocessor three to five times. Intel offers a pipelined vector processor (and the FPS nodes included one) to boost the performance of one node to several MFLOPS.

Acknowledgments

This paper would have been far less accurate and up-to-date without the assistance of our contacts at the manufacturers: Carl Howe at BBN, Ike Bunn of Alliant Computer Systems, Bill Richardson of NCUBE, Len Shar and Robert Olson of ELXSI, Brian Whitney of FPS, Justin Rattner of Intel Scientific Computers, Drew Wilson of Encore Computer Corporation, and Shreekant "Ticky" Thakkar of Sequent Computer Systems. Their cooperation under our tight time constraints is gratefully acknowledged.

References

[Al 10/86] "FX/Series Product Summary," Alliant Computer Systems, Acton, MA, October, 1986.

[Al 85] Almasi, G. S., "Overview of Parallel Processing," *Parallel Computing*, 1985, pp. 191-203.

[BB 3/86] BBN Laboratories, "Butterfly Parallel Processor Overview," BBN Report no. 6148, BBN Laboratories, Inc., Version 1, March 6, 1986.

[BB 87] BBN Advanced Computers, Inc. "The Butterfly GP 1000 Parallel Processor," BBN Advanced Computers, Inc.

[Be 10/87] Beck, Bob, Kasten, B., and Thakkar, S.,"VLSI assist for a multiprocessor,"*Proceedings of the Second International Conference on Architectural Support for Programming Languages and Operating Systems*, October 1987, pp. 10-20.

[Bi 7/87] Billig, Richard R. "A fast backplane cluster heralds a 1000-MIPS computer," *Electronic Design*, July, 1987, pp. 81-86.

[Br 4/87] Brooks, Eugene D. III, "A Butterfly processor-memory interconnection for a vector processing environment," *Parallel Computing*, April, 1987, pp. 103-110.

[Do 3/88] Dongarra, Jack J., "Performance of various computers using standard linear equations software in a Fortran environment," *ACM Computer Architecture News*, March 1988, pp. 47-69.

[El 4/86] "Vector processing boosts hypercube's performance," *Electronics*, April, 1986, pp. 30-31.

[FP 11/87a] Floating Point Systems, "The FPST series—a parallel vector supercomputer," FPS, Beaverton, OR, Nov., 1987.

[FP 11/87b] FPS, "Analysis of Fortran and C performance on the FPS T Series parallel supercomputer," FPS, Inc. Beaverton, OR, Nov. 1987.

[Fr 8/86] Frenkel, Karen A., "Evaluating two massively parallel machines," *Communications of the ACM*, August, 1986, pp. 752-758.

[Ha 10/86] Hayes, John P., "A microprocessor-based hypercube supercomputer," *IEEE Micro*, October 1986 pp. 6-17.

[HT 2/87] *High Technology Business* magazine, "Representative parallel/multiprocessing computers," Feb. 1987.

[In 9/87] "Direct-Connect™ routing solves node communications challenge," *iSCurrents*, Fall–Winter 1987, Intel Scientific Computers, Beaverton, OR, pp. 5–6.

[In 10/87] Intel Scientific Computers, "The INTEL iPSC/2 System product-information", Intel Scientific Computers, Beaverton, OR.

[In 88] Intel Scientific Computers, " iPSC (brochure), Intel Scientific Computers, Beaverton, OR.

[LT 12/87] Lovett, Tom and Thakkar, Shreekant, "The Symmetry multiprocessor system," Sequent Computer Systems, Inc., to be published.

[Ju 6/86] Jurasek, David, "Microprocessor design in custom VLSI", VLSI Systems Design, June 1986, pp. 26-30.

[Ma 2/87] Martin, W.R. "Monte Carlo Photon transport on shared memory and distributed memory parallel processors", University of Michigan, Feb. 1987.

[NC 88] NCUBE Corporation, "NCUBE/10, an overview" (brochure), NCUBE, Beaverton, OR.

[Ol 85] Olson, Robert, "Parallel Processing in a message-based operating system," *IEEE Software*, [1985, pp. 39-49.

[Ol 5/86] Olson, Robert, "Real-time response on a message-based multiprocessor," *IEEE Software*, May 1986, pp. 28-35.

[Pa 5/86] Palmer, John F., "The VLSI parallel computer," *Proc. COMPCON*, May 1986, pp. 397-401.

[Pe 5/86] Perron, Robert, "The architecture of the Alliant FX/8 computer," *Proc. COMPCON*, May, 1986, pp. 390-396.

[PP 84] Papmarcos, M. and Patel, J., "A low overhead coherence solution for multiprocessors with private cache memories," *Proceedings of the 11th International Symposium on Computer Architecture*, June 1984, pp. 348–354.

[Re 12/86] Rettberg, Randall, "Contention is no obstacle to shared-memory multiprocessing," *Communications of the ACM*, December, 1986, pp 1202-1212.

[Re –] Reeves, Anthony P., "Parallel Pascal and the FPS Hypercube supercomputer."

[Sa 9/86] Sanguinetti, John, "Performance of a message-based multiprocessor," *IEEE Computer*, September, 1986, pp. 47-55.

[Sc 11/86] Schanin, David, "The design and development of a very high speed system bus—The Encore Multimax Nanobus," *Proc. FJCC*, November 1986, pp. 410-418.

[SC 87] Scott, Michael and Cox, Alan, "An empirical study of message-passing overhead," *Proceedings of the 7th International Conference on Distributed Computing Systems*, Berlin, Sept. 21–25, 1987, pp. 536–543.

[Se 11/86] "Balance Technical Summary," Sequent Computer Systems, Inc., Nov. 1986.

[Sh 2/87] Shar, Leonard E., "Designing a multiple processor environment," *Proc. COMPCON*, February, 1987, pp. 110-113.

[St 6/87] Steinberg, Jeffrey A., "A new twist: vectors in parallel," *Digital Review*, June 29, 1987, pp. 63-66.

[Te –] Test, Jack A., "Multiprocessor Management in the Concentrix operating system," Alliant Computer Systems Corporation, Acton, MA, pp. 35-43.

[Th 2/88] Thakkar, Shreekant, "The Balance Multiprocessor System," *IEEE Micro*, Feb. 1988, pp. 57-69.

[TM –] Test, Jack A., Myszewski, M., Swift, Richard C., "The Alliant FX/Series: Automatic parallelism in a multiprocessor mini-supercomputer."

[TR 8/88] Tucker, Lewis W. and Robertson, George G., "Architecture and applications of the Connection Machine," *IEEE Computer*, 21:8, August 1988, pp. 26–38.

[Wi 6/87] Wilson, Andrew W. Jr. , "Hierarchical Cache/Bus Architecture for shared Memory Multiprocessors," *Proceedings of the 14th International Symposium on Computer Architecture*, June, 1987, pp. 244-252.