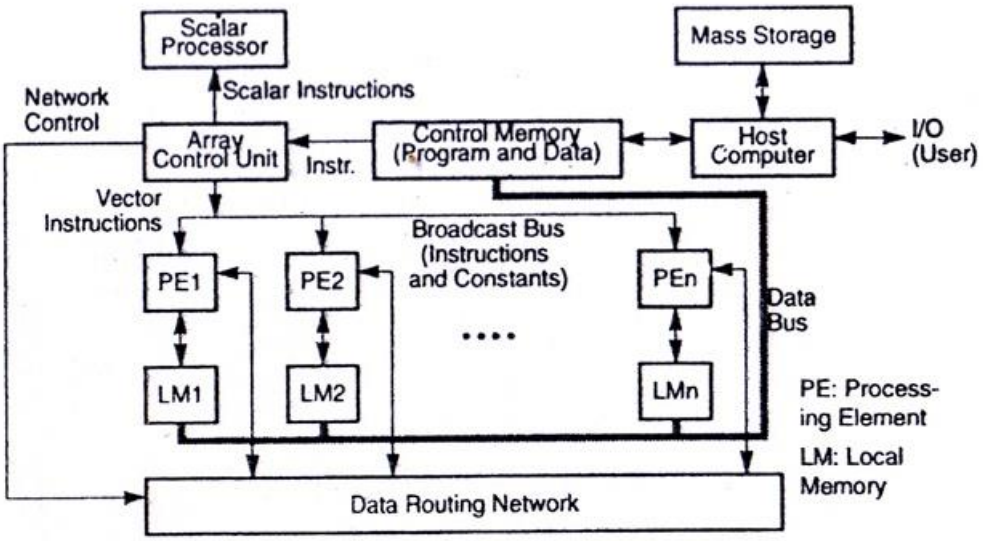
SIMD COMPUTER ORGANIZATIONS

This topic deals with interconnection networks, processing elements, memory and I/O structures.

Implementation Models

Two SIMD computer models are used based on the memory distribution and addressing scheme used. Most SIMD computers use single control unit and distributed memories, except for a few that use associative memories. The two models are Distributed memory models and Shared memory model. The instruction set of an SIMD computer is decoded by the array control unit. The processing elements in the SIMD array are passive ALUs executing instructions broadcast from the control unit. All PEs must operate in lockstep, synchronized by the same array controller.

Distributed Memory Model : Spatial parallelism is exploited in PEs in and SIMD computer. A distributed memory SIMD computer consists of an array of PEs which are controlled by the same array control unit. The diagram (a) shows this model.



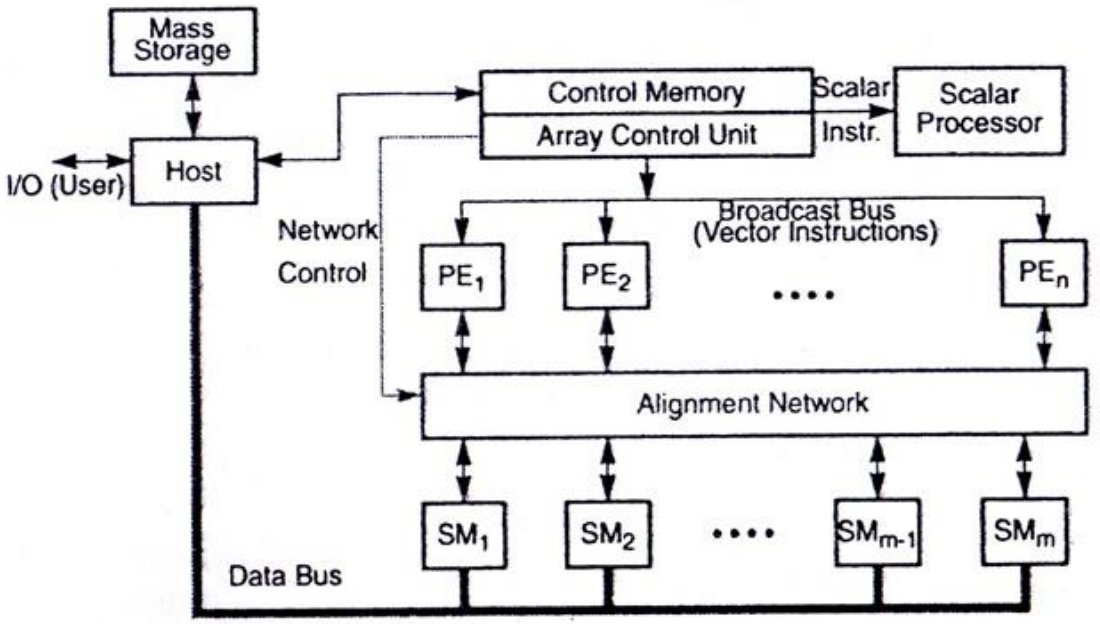
(a) Using distributed local memories (e.g., the Illiac IV)

The program and data are loaded into the control memory through the host computer. An instruction is sent to the control unit for decoding. If it is a scalar or program control operation, it will be directly executed by a scalar processor attached to the control unit. If the decoded instruction is a vector operation, it will be broadcast to all the PEs for parallel execution. The partitioned data sets are distributed to all the local memories attached to the PEs through a vector data bus. The PEs are interconnected by a data routing network which performs inter PE data communications such as shifting, permutation and other routing operations. The data routing network is under program control through the control unit.

The PEs are synchronized in hardware by the control unit. Also the same instruction is executed by the all PEs in the same cycle. Also the masking logic is provided to enable or disable any PE from an participation in a given instruction cycle. The Illiac IV was the early SIMD machine consist of 64 PEs with local memories interconnected by an 8 x 8 mesh with wraparound connection.

All SIMD machines built have been based on the distributed memory model. Various SIMD machines differ mainly in the data routing network chosen for inter PE communications.

Shared Memory Model



(b) Using shared-memory modules (e.g., the BSP)

The diagram shows shared memory model which shows the variation of the SIMD computer using shared memory between PEs. An alignment network is used as the inter PE memory communication network. This diagram is controlled by the control unit. The Burroughs Scientific Processor (BSP) had adapted this architecture , with n = 16 PEs updating m 17 shared memory modules through a 16 x 17 alignment network. The alignment network must properly set to avoid access conflicts. Most SIMD computers were built with distributed memories. Some SIMD computers uses bit slice PEs, like DAP610 and CM/200.

SIMD instructions SIMD computers can execute vector instructions for arithmetic, logic, data routing and masking operations over vector quantities. In bit slice SIMD machines the vectors are nothing but binary vectors. In word-parallel SIMD machines , the vector components are 4 or 8 byte numerical values.

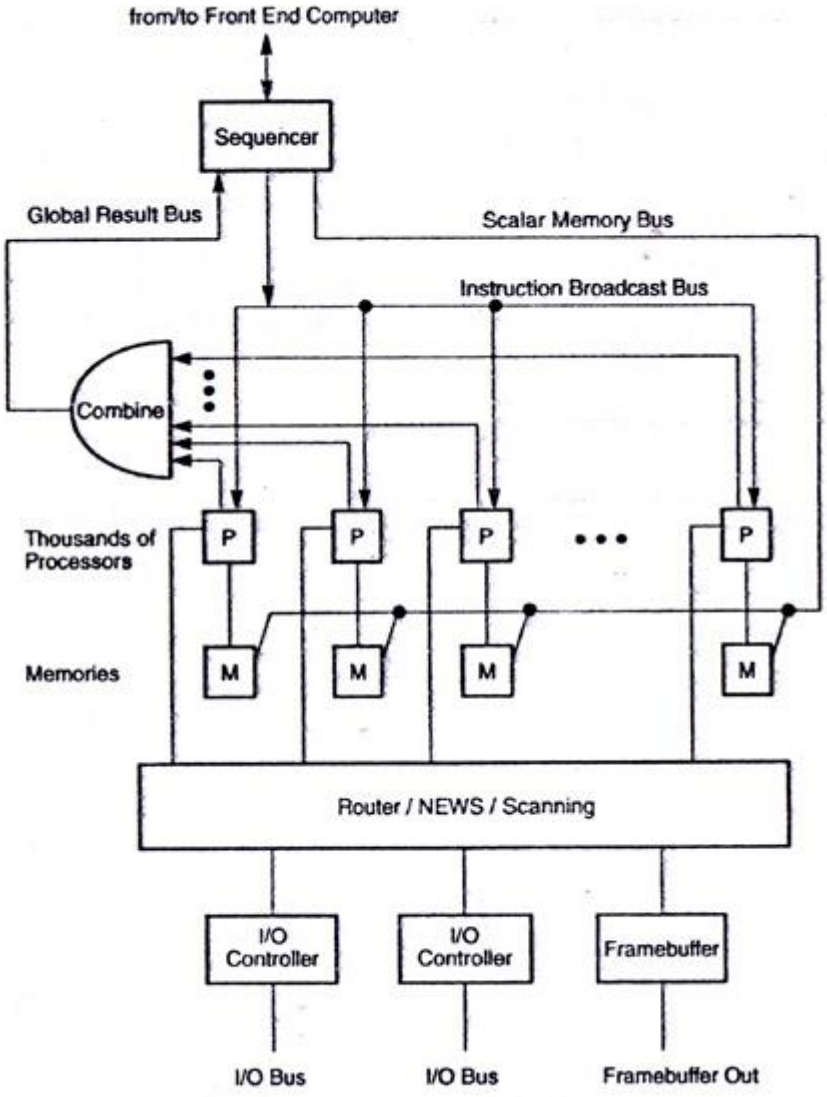
ALL SIMD instructions must use vector operands of equal length n, where n is the number of PEs. SIMD instructions are similar to those which is used in pipelined vector processors. The functions used in SIMD are permutations, broadcasts, multicasts, rotate and shift operations.

HOST and I/O All I/O activities are handled by the host computer in the SIMD organizations. A special control memory is used between the host and the array control unit. The data sets are distributed to the local memories or to the shared memory modules before starting of the program execution. The host manages the mass storage and graphics display of computational results.

The CM-2 Architecture

The CM stands for connection machine, the connection machine CM-2 produced by Thinking machines corporation was a fine grain MPP computer using thousands of bit slice PEs in parallel to achieve a peak processing speed of above 10 Gflops.

Program Execution Paradigm : All programs started execution on a front end, which is issued microinstructions to the back end processing array when data parallel operations were desired. The sequencer broke down these microinstructions and broadcast them to all data processors in the array. The data sets and results could be exchanged between the front end and processing array in one of three ways, they are broadcasting, global combining and scalar memory bus as in the following diagram

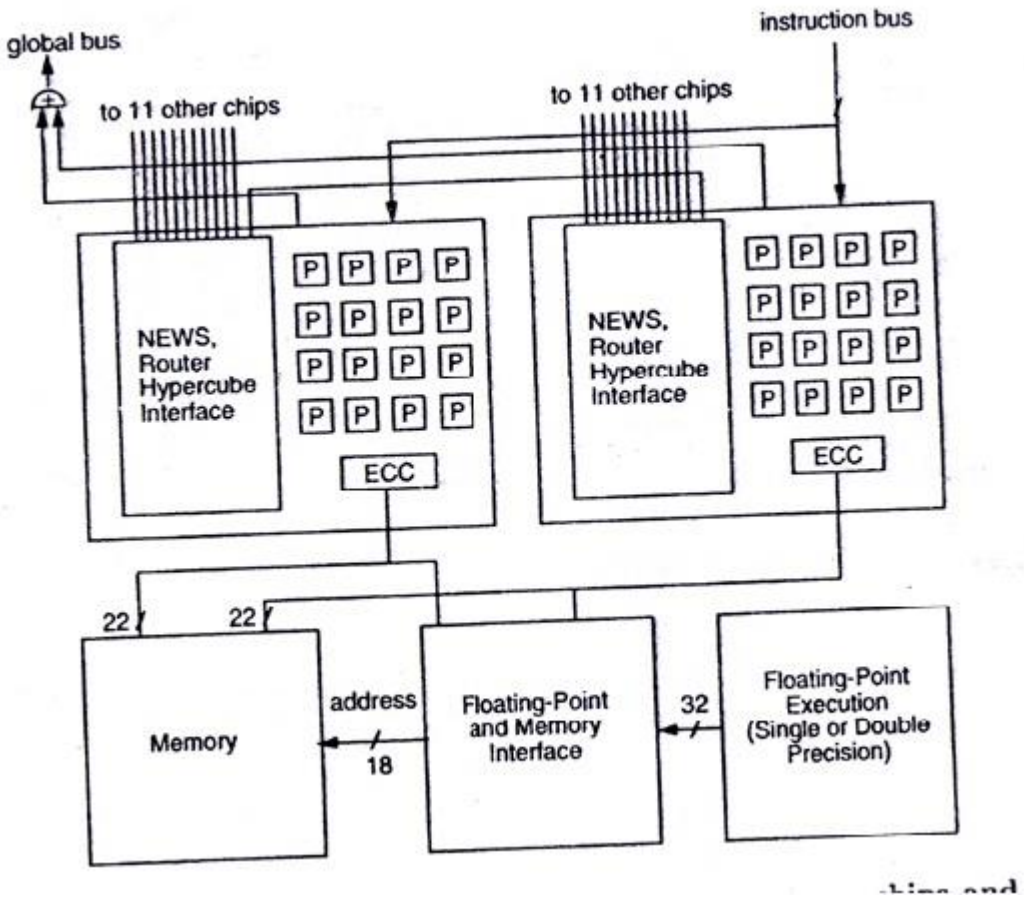

from/to Front End Computer
Sequencer
Global Result Bus
Scalar Memory Bus
Instruction Broadcast Bus
Combine
Thousands of Processors
P
Memories
M
Router / NEWS / Scanning
I/O Controller
I/O Controller
Framebuffer
I/O Bus
I/O Bus
Framebuffer Out

Global combining allowed the front end to obtain sum, largest value, logical OR etc.. of values, one from each processor. The scalar bus allowed the front end to read or to write one 32-bit value at a time from or to the memories attached to the data processors.

The Processing Array : The CM-2 was a back end machine for data parallel computation. The processing array contained from 4K to 64K bit slice data processors, all of which were controlled by a sequence as shown in the above diagram.

The sequencer decoded microinstructions from the front end and broadcast nanoinstructions to the processors in the array. All processors could access their memories simultaneously. All processors executed the broadcast instructions in a lockstep manner. The processors exchanged data among themselves in parallel through the router , NEWS grids, or a scanning mechanism. These network elements are also connected to I/O interfaces. A mass storage subsystem called the data vault was connected through the I/O for storing up to 60 Gbytes of data.

Processing Nodes: The diagram shows the CM-2 processor chips with memory and floating point chips.



Each data processing node contained 32 bit slice data processors, an optimal floating point accelerator , and interfaces for interprocessor communication. Each data processor was implemented with a 3-input and 2 output bit slice ALU and associated latches and a memory interface. This ALU could perform bit serial full adder and Boolean logic operations. The processor chips were paired in each node sharing a group of memory chips. Each processor chip contained 16 processors. The parallel instruction set, called Paris, included nanoinstructions for memory load and store, arithmetic and local, and control of the router, NEWS grid and hypercube interface, floating point, I/O ,and diagnostic operations.

The memory data path was 22 bits per processor chip. The 18-bit memory address allowed 2^{18} = 256 K memory words shared by 32 processors. The floating point handled 32-bit operations at a time.

Hypercube Routers: Special hardware was built on each processor chip for data routing among the processors. The router nodes on all data processor chips were wired together to form a Boolean n-cube. A full configuration of CM-2 had 4096 router nodes on processor chips interconnected as a 12 dimensional hypercube. Each router was connected to 12 other router nodes.

The NEWS Grid: Within each processor chip the 16 physical processors could be arranged as an 8 x 2 , 1 x 16, 4 x 4, 4 x 2 x 2 , or 2 x 2 x 2 x 2 grid. Sixty four virtual processors could be assigned to each physical processor. These 64 virtual processors could be rearranged to form a 8 x 8 grid with in the chip. The NEWS grid was based on the fact that each processor has a north, east, west and south neighbor in the various grid configurations. Also a subset of the hypercube wires could be chosen to connect the 2^{12} nodes as a two dimensional grid of any shape, 64 x 64 being one of the possible grid configurations.

Scanning and Spread Mechanisms: Besides dynamic configuration in NEWS grids through the hypercube routers, the CM-2 had been built with special hardware support for scanning or spreading across NEWS grids. Scanning on NEWS grids combined communication and computation. The operation cold simultaneously scan in every row of a grid along a particular dimension for the partial sum of that row , the largest or smaller value or bitwise OR,AND or exclusive OR.

I/O and Data Vault : The connection Machine emphasized massive parallelism in computing as well as in visualization of computational results. High speed I/O channels were available from 2 to 16 channels for data and/or image I/O operations. Peripheral devices attached to I/O channels included the data vault, CM-IOP system and VME bus.

Applications of CM-2:

(i) Used in Document retrieval using relevance feedback
(ii) Memory based reasoning in the medical diagnostic system called QUACK for simulating the diagnosis of a disease.
(iii) SPICE like VLSI circuit analysis and layout, fluid dynamics
(iv) Signal/image/vision processing and integration

(v) Neural network
(vi) Context free parsing
(vii) Ray tracing graphics.

The MasPar MP-1 Architecture

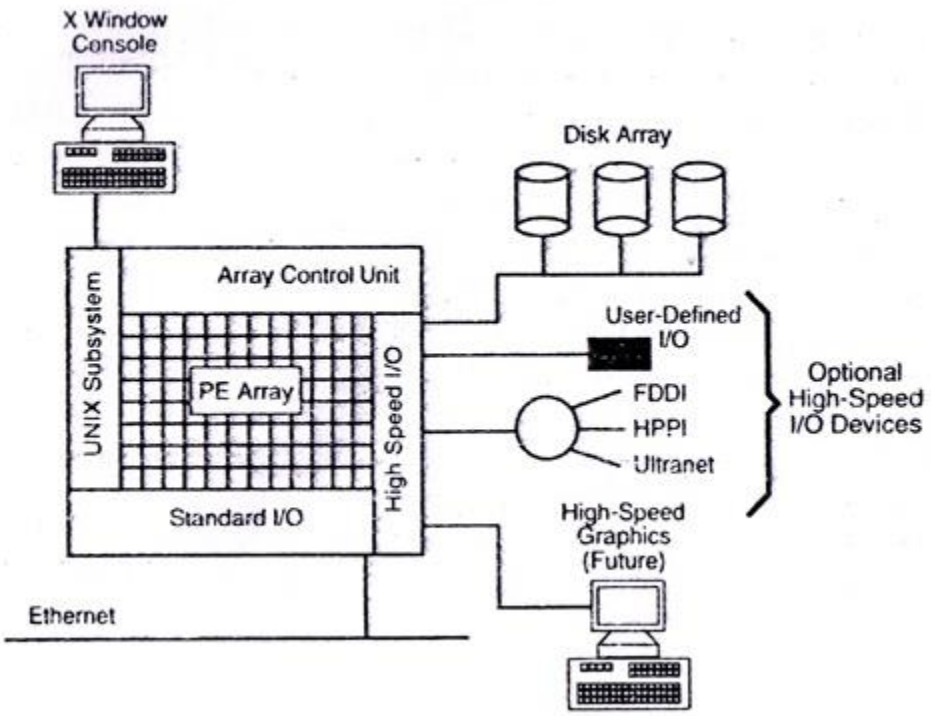
It is a medium grain SIMD computer, different from the CM-2. It provide interprocessor communication mechanisms.

The MasPar MP-1: The MP-1 architecture consists of four subsystems, they are (i) the PE array, (ii) the array control unit (ACU),(iii) a UNIX subsystem with standard I/O, and (iv) high speed I/O subsystem as in the diagram. The MP-1 family included the configurations with 1024, 4096 and up to 16,384 processors. The peak performance of the 16K processor configuration was 26,000 MIPS in 32-bit RISC integer operations. The system also had a peak floating point capability of 1.5 Gflops in single precision and 650 Mflops in double precision operation.

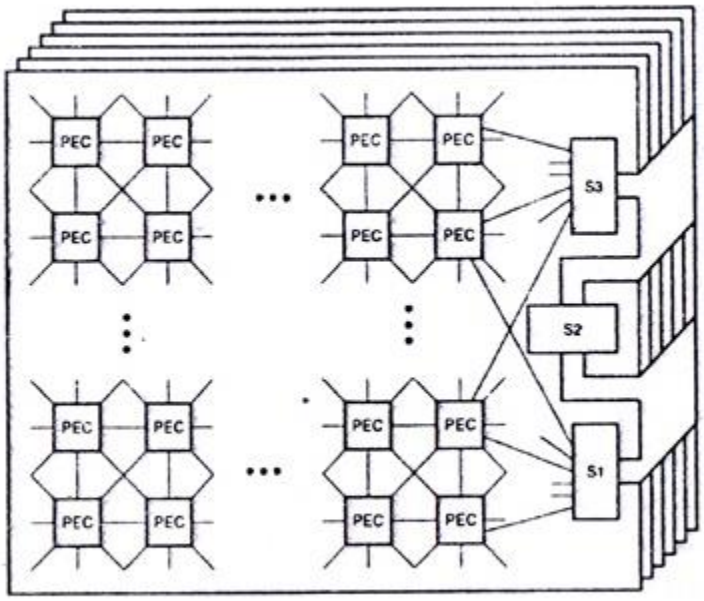
Array Control Unit: The ACU was 14-MIPS scalar RISC processor using a demand paging instruction memory. The ACU fetched and decoded MP-1 instructions, computed addresses and scalar data values, issued control signals to the PE array, and monitored the status of thePE array.

The ACU was microcoded to achieve horizontal control of the PE array. Most scalar ACU instructions executed in one 70 ns clock. The whole ACU was implemented on the PC board.

An implemented functional unit, called a memory machine, was used in parallel with the ACU. The memory machine performed PE array load and store operations, while the ACU broadcast arithmetic, logic and routing instructions to the PEs for parallel execution.

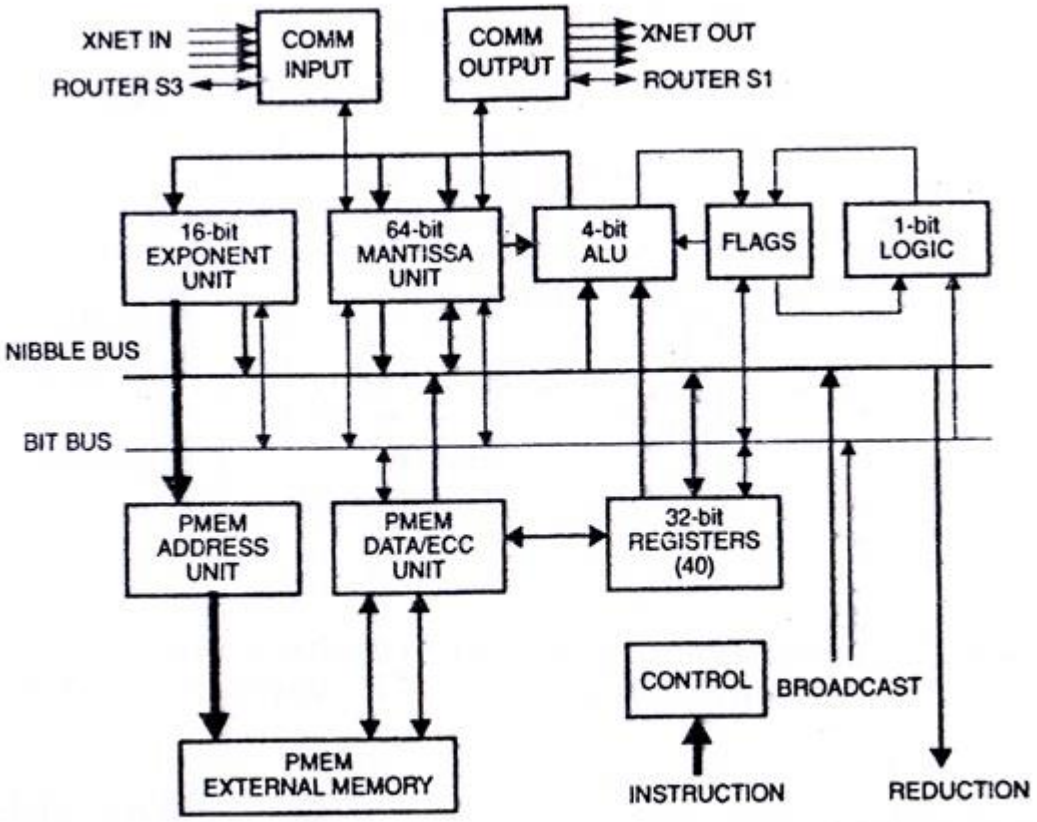


(a) MP-1 System Block Diagram



(b) Array of PE clusters

The PE array: Each processor board has 1024 PEs and associated memory arranged as 64 PE clusters(PEC) with 16 PEs per cluster. The following diagram (b) shows the inter PEC connections on each processor board. Each PEC chip was connected to eight neighbors via the X-Net mesh and global multistage crossbar router network, labeled S1,S2, S3 as in the following diagram.

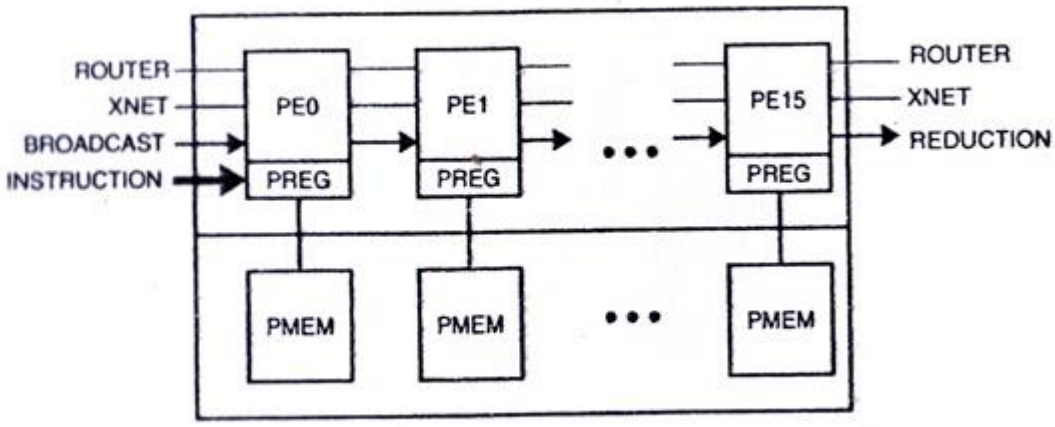


(b) Processor element and memory

Each PE cluster in the following diagram (a) was composed of 16 PEs and 16 processor memories(PEMs). The PEs were logically arranged as a 4 x 4 array for the X-Net two dimensional mesh interconnections. The 16 PEs in a cluster shared an access port to the multistage crossbar router. Interprocessor communications were carried out through three mechanisms they are ,

(I) ACU-PE array communications
(II) X-Net nearest neighbor communications
(III) Global crossbar router communications

The first mechanism supported ACU instruction / data broadcasts to all PEs in the array simultaneously and performed global reductions on parallel data to recover scalar values from the array.



(a) A PE cluster

X-Net Mesh Interconnect : The X-net interconnect directly connected each PE with its eight neighbors in the two dimensional mesh. Each PE had four connections at its diagonal corners, forming an X pattern similar to the BLITZEN X grid network. A tri stage node at each X intersection permitted communication with any of eight neighbors using only four wires per PE.

The connections to the PE array edges were wrapped around to form a 2-D torus. The torus structure is symmetric and facilitates several important matrix algorithms and can use a one dimensional ring with two X-Net steps.

Multistage Crossbar Interconnect: The network provided the global communication between all PEs and formed the basis for the MP-1 I/O system. The three router stages implemented the function of a 1024 x 1024 crossbar switch. The router chips were used on each processor board. Each PE cluster shared an originating port connected to router stage Sq and a target port connected to router stage S3. Connections were established from an originating PE through the stages S1,S2 and Se and then to the target PE. The router supported up to 1024 simultaneous connections with an aggregate bandwidth of 1.3 Gbytes/s.

Processor Elements an Memory : The PE design had mostly data path logic and no restriction fetch or decode logic. This design is included in the above diagram (b). Both integer and floating point computations executed in each PE with a register based RISC architecture. Load and store instructions moved data between the PEM and the register set.

Each PE(processing element) had forty 32-bit registers available to the programmer and eight 32-bit registers for system use. The registers were bit and byte addressable. Each PE had a 4-bit integer ALU, a 1-bit logic unit, a 64-bit mantissa unit, a 16-bit exponent unit and a flag unit. The NIBBLE bus was four bits wide and the BIT bus was one bit wide. The PEM could be directly or indirectly addressed with a maximum aggregated memory bandwidth of 12 Gbytes/s.

Most data movement with each PE occurred on the NIBBLE bus and the BIT bus. Different functional units within the PE could be simultaneously active during each microstep.

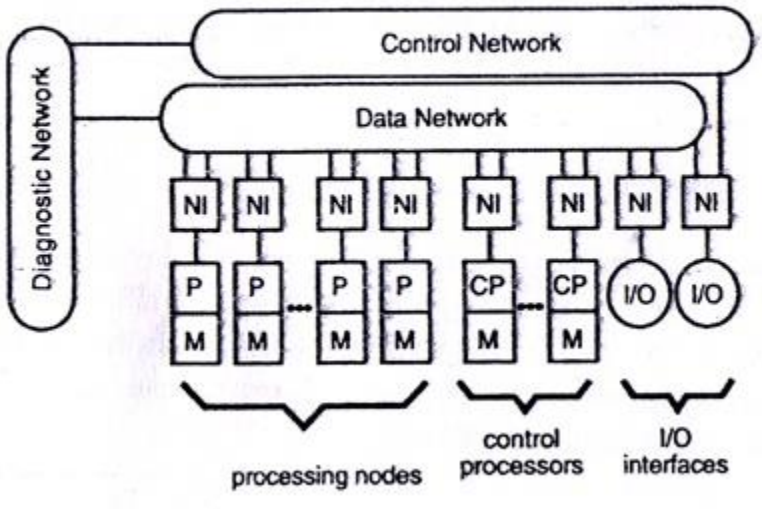
Parallel Disk Arrays: Another feature of Masspar-I is the massive parallel I/O architecture. The PE array communicated with a parallel disk array through the high speed I/PO subsystem, which is using the 1.3 Gbytes/s global router network. The disk array provided up to 17.3 Gbytes of formatted capacity with a 9-Mbytes/s sustained disk I/O rate. The parallel disk array was a necessity to support data parallel computation and provide file system transparency and multilevel fault tolerance.

THE CONNECTION MACHINE CM-5

The connection machine CM-5 was the most innovative effort of Thinking Machines corporation toward this end.

A Synchronized MIMD Machine : The CM-5 is a universal architecture, which combines the advantages of both SIMD and MIMD machines. Traditionally , supercomputer programmers were forced to choose between MIMD and SIMD computers. An MIMD machine is a good at independent branching but bad at synchronization and communication. The CM-5 was designed with a synchronized MIMD structure to support both styles of parallel computation.

The Building blocks: The CM-5 architecture is as shown in the following diagram. The machine was designed to contain from 32 to 16,384 processing nodes, each of which could have a 32-MHz SPARC processor, 32-Mbytes of memory and a 128 Mflops vector processing unit capable of performing 64-bit floating point and integer operations. The CM-5 system used a number of control processors , which were Sun Microsystems workstation computers. The number of control processors, varying with different configurations, ranged from one to several tens. Each control processor was configured with memory and disk based on the needs.



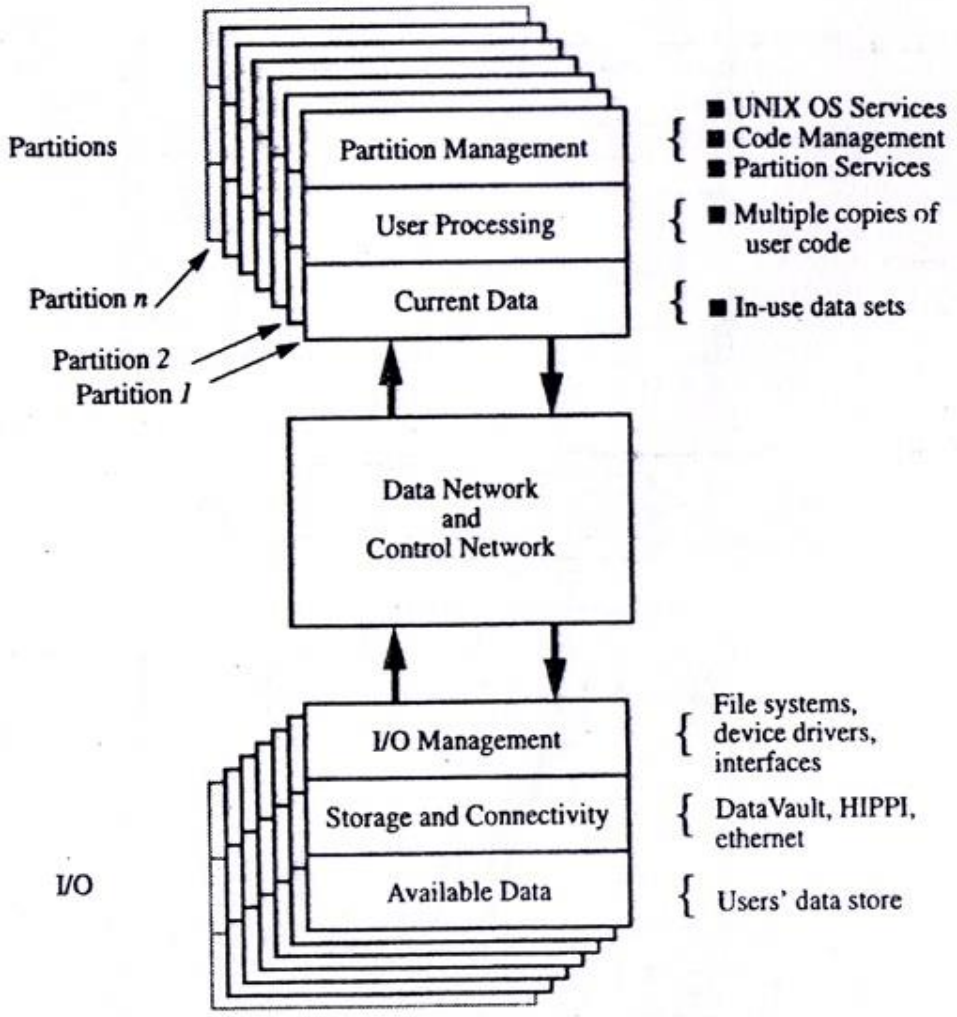
Input and output were provided through high bandwidth I/O interfaces to graphics devices, mass secondary storage such as data vault, and high performance networks. Additional low-speed I/O was provided by Ethernet connections to the control processors. The largest configuration was occupy the space of 30m x 30m and designed for a peak performance of 1 Tflops.

The Network Functons : The building blocks were interconnected by three networks, they are data network, a control network and diagnostic network. The data network provided high performance, point to point data communications between the processing nodes. The control network provided cooperative operations, including broadcast, synchronization, and scans.

The diagnostic network allowed “back door” access to all system hardware to test system integrity and to detect and isolate errors. The data and control networks were connected to processing nodes, control processors and I/O channels through network interfaces. The CM-5 architecture was considered universal because it was optimized for data parallel processing of large and complex problems. The data and control networks were designed to have good scalability, making the machine size limited by the cost but not by any architectural or engineering constraint. The networks depend on specific types of processors. The network interfaces were designed to provide an abstract view of the networks.

The system operations

The system operated one or more user partitions. Each partition consisted of a control processor , a collection of processing nodes, and dedicated portions of the data and control networks. The following diagram shows the distributed control of CM-5 architecture obtained through the dynamic use of the two interprocessor communication networks.



Partitioning of resources was managed by a system executive. The control processor assigned to each partition behaved like a partition manager. Each user process executed on a single partition but could exchange data with processes on other partitions. Since all partitions utilized UNIX time-sharing and security features, each allowed multiple users to access the partition.

Access to system functions was classified as either privileged or nonprivileged. Privileged system functions included access to data and control networks. These accesses could be executed directly by user code without system calls. The OS kernel overhead could be eliminated in network communication within a user task.

Some control processors in the CM-5 were assigned to manage the I/O devices and interfaces. This organization allowed a process on any partition to access any I/O device, and ensured that access to one

device does not impede access to other devices. The system operations, is defined in the above diagram. These operations are divided into user oriented partitions, I/O services based on system calls, dynamic control of the data and control networks and system management and diagnostics.

The two networks could download the user code from a control processor to the processing nodes, pass I/O requests, transfer messages of all sorts between control processors, and transfer data between nodes and I/O devices. The I/O capacity could be scaled with increasing numbers of processing nodes or of control partitions.

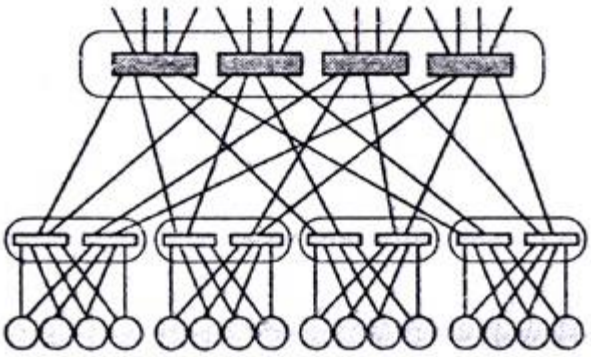
THE CM-5 NETWORK ARCHITECTURE

The data network was based on the fat tree was designed by Leiserson in 1985. It is applied to the CM-5 construction.

FAT TREES

A fat tree is more like a real tree in that it becomes thicker as it requires more leaves. Processing nodes, control processors, and I/O channels are located at the leaves of a fat tree. A binary fat tree was the example. The internal nodes are switches, unlike an ordinary binary tree, the channel capacities of a fate tree increase as we ascend from the leaves to root.

The hierarchical nature of fat tree can be exploited to give each user partition a dedicated sub tree. Which cannot be interfered with by any other partition message traffic. The CM-5 data network was actually implemented with a 4-ary fat tree as shown in the following diagram .



To implement the partitions , one could allocate different subtrees to handle different partitions. The size of the subtrees varied with different partition demands. The I/O channels were assigned to another subtree, which was not devoted to any user partition. The I/O subtree was accessed as shared system resource . All leaf nodes had unique physical addresses.

The Data Network : To route a message from one processor node to another. The message ws sent up the tree to the least common ancestor(predecessor) of the two processors and then down to the destination.

In the 4-ary fate tree implementation in the above diagram of the data network, each connection provided a link to another chip with a raw bandwidth of 20 Mbytes/s in each direction. By selecting at each level of the tree whether two or four parent links are used, the bandwidths between nodes in the fat tree could be adjusted.

Each processor had two connections to the data network, corresponding to a raw bandwidth of 4 Mbytes/s in and out of each leaf node. In the first two levels, each router chip used only two parent connections to the next higher level, which yields a bandwidth of 160 Mbytes/s out a subtree with 16 leaf nodes. The bandwidth continued to scale linearly up to 16,384 nodes, the largest CM-5 configuration planned.

As a message went to tree , it would have several choices as to which parent connection to take. The decision was resolved by pseudo randomly selecting from among those links that were unobstructed by other messages. After reaching the least common ancestor of the source and destination nodes, the message took a single available path of links down the destination.

The data network chips were driven by a 40 MHz clock. The first two levels were routed through backplanes. The wires on higher levels were routed through cables, which could be either 9 or 26 feet in length. Faulty processor nodes or connection links could be mapped out of the system and quarantined. This allowed the system to remain functional while servicing and testing the mapped out portion. The data network was acyclic from input to output.

The Control Network: The architecture of the control network was that of a complete binary tree with all system components at the leaves. Each user partition was assigned to subtree of the network. Processing nodes were located at leaves of the subtree, and a control processor was mapped into the partition at an additional leaf. The control processor executed scalar part of the code, while the processing nodes executed the data parallel part.

Unlike the variable length messages transmitted by the data network, control network packets that had a fixed length of 65 bits. There are three major types of operations on the control network, they are (i) broadcasting, (ii) Combining and (iii) Global operations. These operations provided interprocessor communications.

The control network provided the mechanisms allowing data parallel code to be executed efficiently. It is supported MIMD execution for general purpose applications. The binary tree architecture made the control network simpler to implement than the fat tree used in the data network. It also has a additionl switching capability.

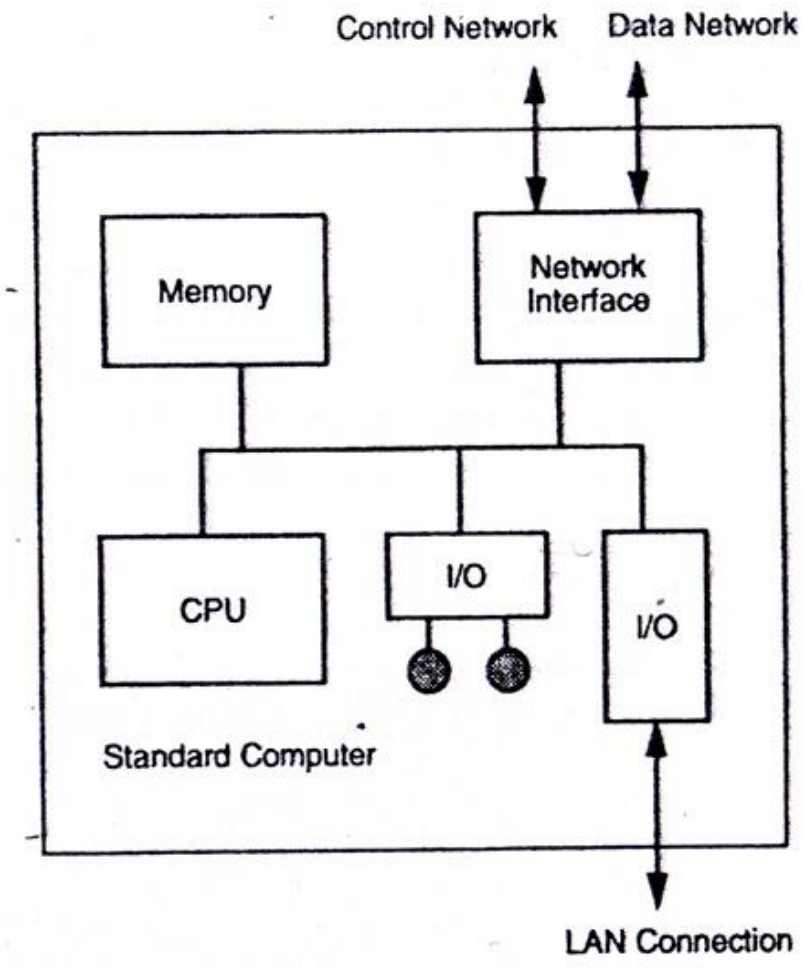
The Diagnostic Network: This network was needed for upgrading system availability. Built in testability was achieved. This network was organized as a binary tree for its simplicity in addressing. One or more diagnostic processors were at the root. The leaves were pods(case), and each pod was a physical system, such as a board or a backplane.

The diagnostic network allowed groups of pods to be addressed according to a hypercube address scheme. A special diagnostic interface was design to form an in system check of the integrity of all CM-5 chips that supported the JTAG(Joint Test Action Group) standard and all networks. It provides scan access to all chips which support JTAG standard. The network itself was completely testable and diagnosable.

Control Processors and Processing Nodes

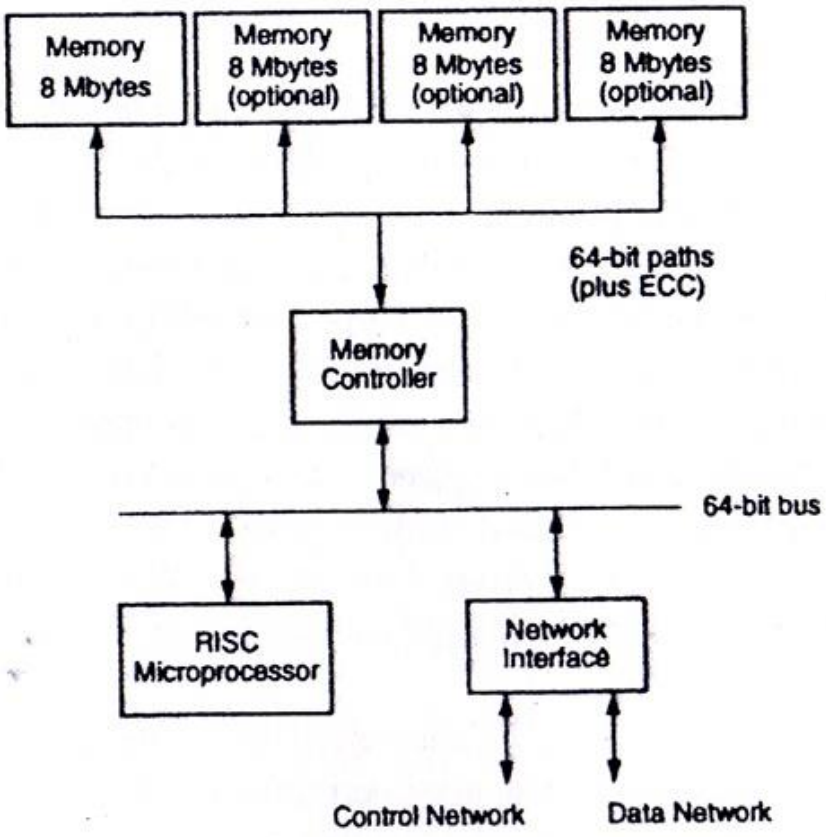
Control Processors

The following diagram shows the basic control processor consists of a RISC microprocessor , memory subsystem , I/O with local disks and Ethernet connections along with CM-5 network interface. This was equivalent to a standard off the shelf workstation class computer system. The network interface connected the control processor to the rest of the system through the control network and the data network.

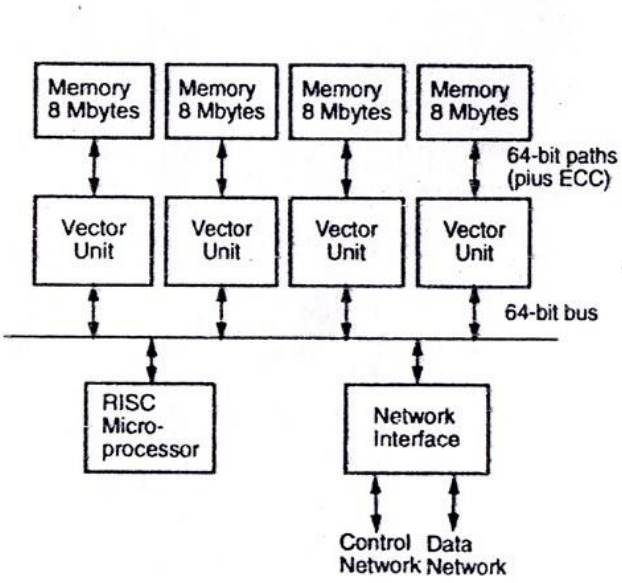


Each control processor ran CMOST, a UNIX based OS with extensions for managing the parallel processing resources of the CM-5. Some control processors managed computational resources in user partitions. Others were used to manage I/O resources. Control processors specialized in managerial functions rather than computational functions.

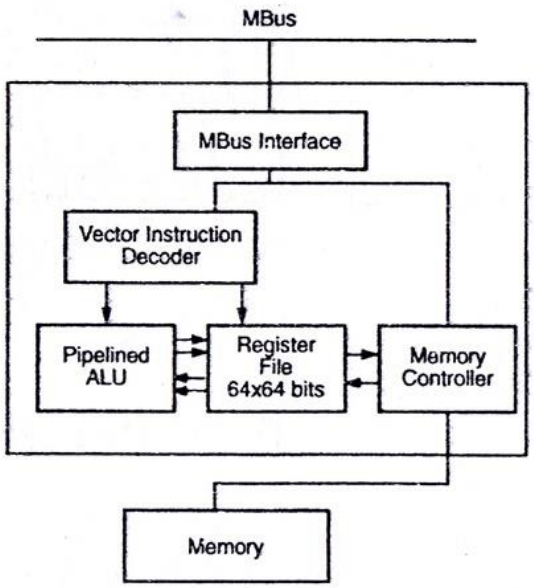
Processing Nodes The following diagram shows the basic structure of a processing node. It was a SPARC based processors with a memory subsystem consisting of a memory controller and a 8,16 or 32 Mbytes of DRAM memory. The internal bus was 64-bits wide. The SPARC processor was chosen for its multiwindow feature to facilitate fast context switching. This was very crucial to the dynamic use of the processing nodes in different user partitions at different times. The network interface connected the node to the rest of the system through the control and data networks.



Vector Units : As illustrated in the following diagram , the vector units could be added between the memory bank and the system bus as an optional feature. The vector units would replace the memory controller . Each vector unit had a dedicated 72-bit so its attached memory bank, providing a peak memory bandwidth of 128 Mbytes/s per vector unit. The vector unit executed vector instructions issued by the scalar processor and performed all functions of a memory controller, including generation and check ECC bits. From the following diagram (b) each vector unit instruction decoder , a pipelined ALU, and sixty four 64-bit registers like a conventional vector processor.



(a) Processing node with vector units



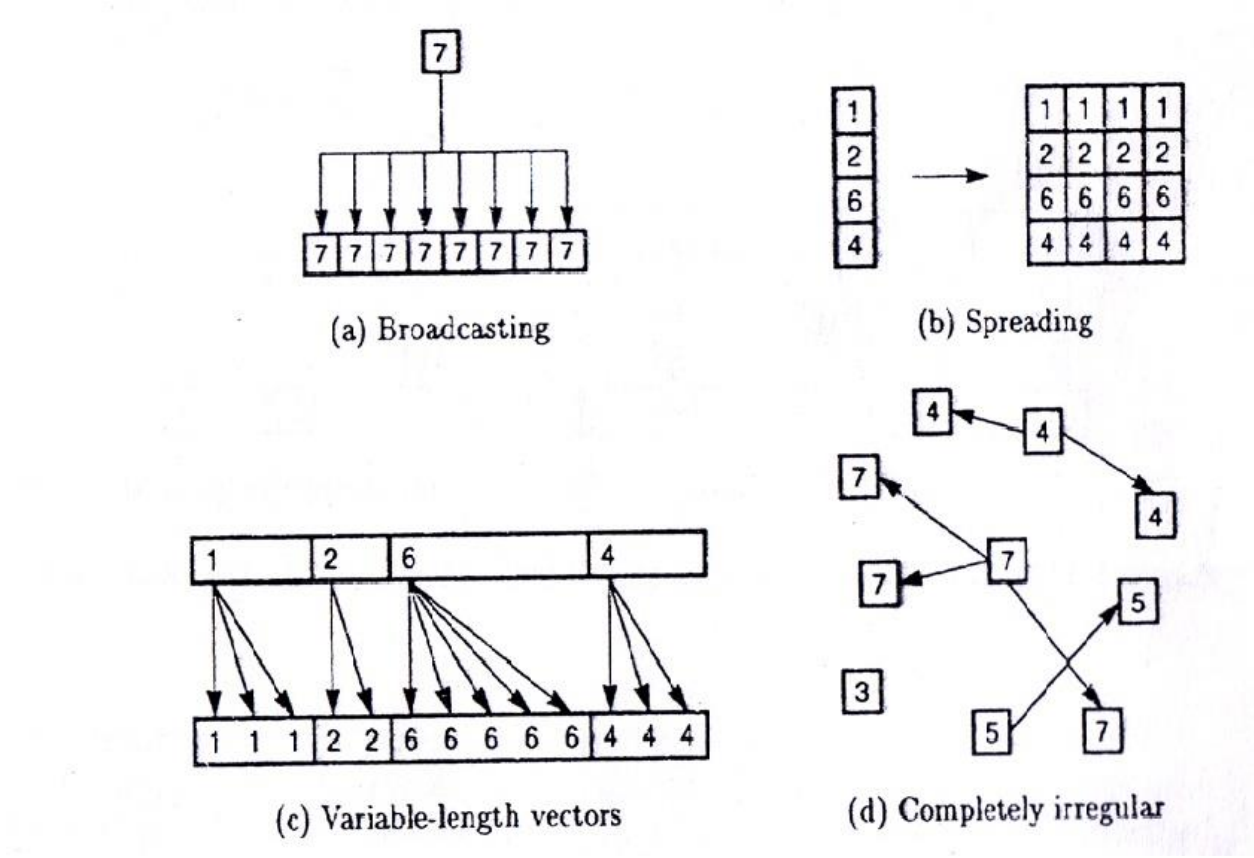
(b) Vector unit functional architecture

Each vector instruction could be issued to a specific vector unit or pairs of units or broadcast to all four units at once. The scalar processor took care of address translation and loop control, overlapping them with vector unit operations. The vector units provided 512 Mbytes/s memory bandwidth and 128 Mflops 64-bit peak performance per node. The CM-5 is a supercomputer.

Interprocessor Communications:

There is a high speed and spreading mechanisms in CM-2. In the CM-5 these mechanisms were designed to be further upgraded into four categories of interprocessor communication : replication, reduction, permutation, parallel prefix. These operations could be applied to regular or irregular data sets including vectors, linked lists, completely irregular patterns. The role of control network is also identified in these operations.

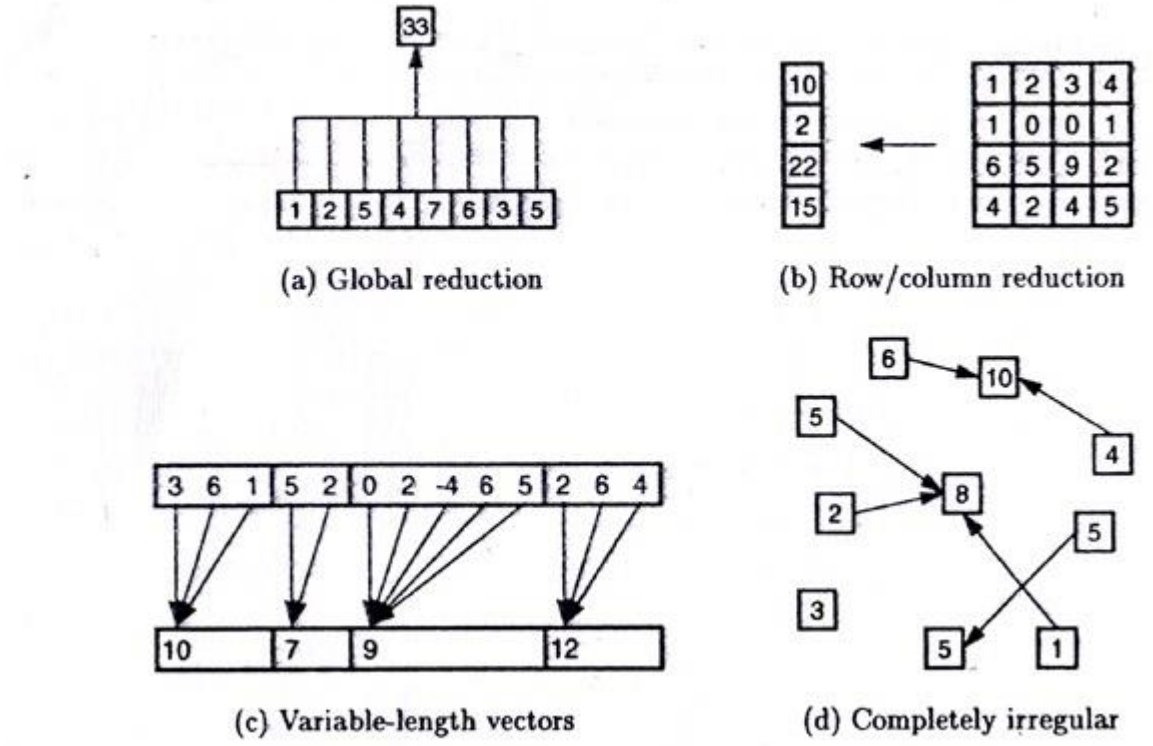
Replication: When we recall the broadcast operation, where a single value may be replicated to as many copies and distributed to all processors as in the following diagram(a), other duplication operations include the spreading of a column vector into all the columns of the matrix as in the figure (b) , the expansion of a short vector into a long vector as in diagram (c) and a completely irregular duplication as in diagram (d). The replication plays a fundamental role in matrix arithmetic and vector processing, particularly on a data parallel machine. Replication is carried out through the control network in four kinds of broadcasting schemes, they are user broadcast, supervisor broadcast, interrupt broadcast, utility broadcast. These operations can be used to download code and to distribute data to implement better synchronization and to configure partitions through the Operating Systems.



Reduction: Vector reduction was implemented on the CM-2 by fast scanning, and on the CM-5, this mechanism as further generalized as the opposite of replication. It is illustrated in the following diagram, global reduce produces the sum of vector components as in diagram (a). Also the row/column reductions produce the sums per each row or column of a matrix as in diagram (b).

Variable length vectors were reduced in chunks of a long vector in diagram ©. The same idea was applied to a completely irregular as in diagram (d). In general reduction functions include the maximum, the minimum, the average, the dot product, the sum , logical AND, logical OR. Fast scanning and combining are necessities in implementing these operations.

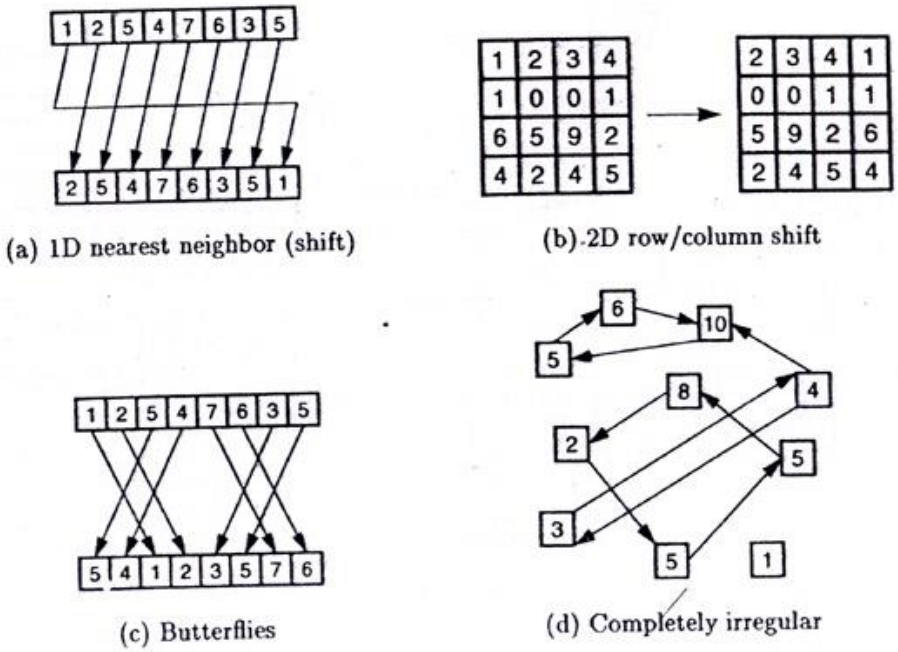
Four types of combining operations reduction, forward scan, backward scan and router done, were xsupported by the control network. Router done refers to the detection of completion of a message routing cycle, based on Kirchoff's current law, in that the network interfaces keep track of the number of messages entering andleaving the data network.



(a) Global reduction (b) Row/column reduction

(c) Variable-length vectors (d) Completely irregular

Permutation: Data parallel computing relies on permutation for fast exchange of data among processing nodes. The following diagram shows four cases of permutations performed on CM-5. These permutation operations are often needed in matrix transpose , reversing vector, shifting a multidimensional grid, and FFT butterfly operations.



(a) 1D nearest neighbor (shift) (b) 2D row/column shift

(c) Butterflies (d) Completely irregular

Parallel Prefix:

This is a kind of combining operation supported by the control network . A parallel prefix operation delivers to the ith processor the result of applying one of the five reduction operators to the values in the preceding i – 1 processors , in the linear order given by the data address. The idea illustrated in the following diagram with four examples. The following diagram (a) shows the one dimensional sum prefix, in which for example the fourth output 12 is the sum of the first four input elements . The two dimensional row/column sum prefix (diagram (b)) can be similarly performed using the forward scanning mechanism.

The figure (c) computes the one dimensional prefix sum on sections of a long vector independently. The diagram (d) shows the forward scanning along linked lists to produce the prefix sums as outputs.