ADVANCED COMPUTER ARCHITECTURE

UNIT-1

PARALLEL COMPUTER MODEL

- The state of computing
- Multiprocessors and Multicomputer
- Multivector and SIMD computers

THE STATE OF COMPUTING:

- Modern computer are equipped with powerful hardware facilities driven by extensive software package.
- To access state of the art computing, we first review historical milestones in the development of computers.
- Basic hardware and software factors are identified in analyzing the performance of computers.

COMPUTER DEVELOPMENT MILESTONES:

Prior to 1945, computers were made with mechanical or electronically parts. The earliest mechanical computer can be traced back to 500 BC in the form to the abacus used in china.

COMPUTER GENARATIONS:

- Over the past several decades, electronic computers have gone through roughly five generation of developed of development. Each of the first three generation lasted about 10 years.
- The division of generation is marked primarily by major changes in hardware and software technologies. Most features introduced in earlier generation have been passed to later generations.

FIVE GENERATION OF ELECTRONIC COMPUTERS:

Generation	Technology and Architecture	Software and Applications	Representative Systems
First (1945–54)	Vacuum tubes and relay memories, CPU driven by PC and accumulator, fixed-point arithmetic.	Machine/assembly languages, single user, no subroutine linkage, programmed I/O using CPU.	ENIAC, Princeton IAS, IBM 701.
Second (1955–64)	Discrete transistors and core memories, floating-point arithmetic, I/O processors, multiplexed memory access.	HLL used with compilers, subroutine libraries, batch processing monitor.	IBM 7090, CDC 1604, Univac LARC.
Third (1965–74)	Integrated circuits (SSI/-MSI), microprogramming, pipelining, cache, and lookahead processors.	Multiprogramming and time-sharing OS, multiuser applications.	IBM 360/370, CDC 6600, TI-ASC, PDP-8.
Fourth (1975–90)	LSI/VLSI and semiconductor memory, multiprocessors, vector supercomputers, multicomputers.	Multiprocessor OS, languages, compilers, and environments for parallel processing.	VAX 9000, Cray X-MP, IBM 3090, BBN TC2000.
Fifth (1991–present)	ULSI/VHSIC processors, memory, and switches, high-density packaging, scalable architectures.	Massively parallel processing, grand challenge applications, heterogeneous processing.	Fujitsu VPP500, Cray/MPP, TMC/CM-5, Intel Paragon.

PROGRESS IN HARDWARE:

- As far as hardware technology is concerned, the first generation (1945-1954) used vacuum tubes and relay memories interconnected by insulated wires.
- The second generations (1955-1964) to as marked by the use of discrete transistors, diodes and magnetic ferrite coerces, interconnected by printed circuits.

THE FIRST GENERATION:

From the architectural and software points of view, first generation computers were built with a single central processing unit (cup) with performed serial fixed point arithmetic using a program counter, branch instructions and an accumulator. The cup must be involved in all memory access and input/out put (I/O) operations.

THE SECOND GENERATIONS:

Index registers, floating-point arithmetic, multiplexed memory, and I/O processors were introduced with second generation computers, high level language (HLLS), such as Fortran , Algol and cabal were introduced along with compilers. Subroutines libraries and batch processing monitors.

THE THIRD GENERATIONS:

The third generations was represented by the IBM/360-370 series, the CDC 6600/7600 series, Texas instruments ASC(Advanced Scientific Computer) and digital equipments PDP-8 series from the mid 1960 S to the mid 1970 S.

THE FOURTH GENERATION:

Parallel computers in various architectures appeared in the fourth generation of computer using shared or distributed memory or optimal vector hardware. Multiprocessing os , special language and compilers were developed for parallelism. Software tools and environment were created for parallel processing or distributed computing.

THE FIFTH GENERATION:

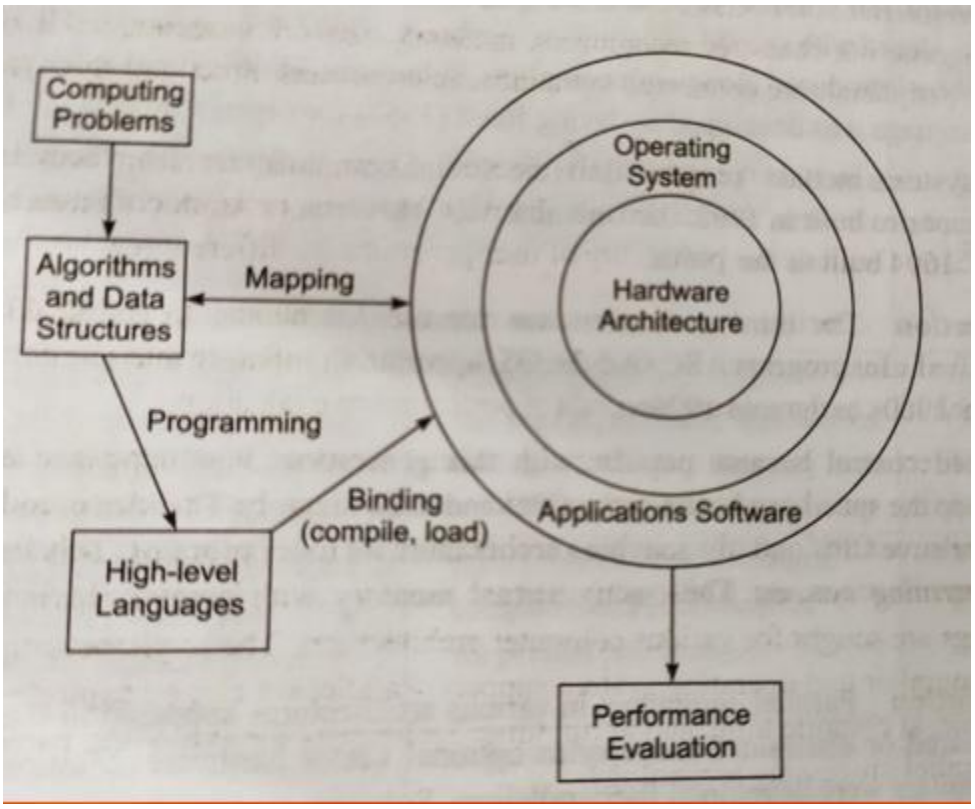
These systems emphasive superscalar processors cluster computers and massively parallel processing (MPP). Scalable and latency tolerant architecture are being adopted in MPP system using advanced VLSI technologies, high density packaging and optical technologies.

ELEMENTS OF MODERN COMPUTERS:

Hardware, software and programming elements of a modern computer system are briefly introduced below in the context of parallel processing.

COMPUTING PROBLEMS:

It has been long recognized that the concept of computer architecture is no longer restricted to the structure of the bare machine hardware. A modern computer is an integrated system consisting of machine hardware, an instruction set, system software, application programs and user interfaces.



For artificial intelligence (AI), the solutions demand logical interences and symbolic manipulation these computing problems have been labelled numerical computing, transaction processing, and logical reasoning. Some complex problems may demand a combination of these processing models.

ALGORITHMS AND DATA STRUCTURE:

Special algorithm and data structures are needed to specify the computations and communication involved in computing problems. Most numerical algorithms are deterministic, using regularly structured data. Symbolic processing may use heuristics or nondeterministic searches overlarge knowledge bases.

HARDWARE RESOURCES:

In addition, software interface programs are needed. These software interfaces include file transfer systems, editors, word processors, device drivers, interrupt handless, network communication programs,etc.., These programs greatly facilitated the portability of user program on different machine architecture.

OPERATING SYSTEM:

- AN effective operating manages the allocation and deallocation of resources using the execution of user programs. Beyond the OS, application software must be developed to benefit the users. Standard bench mark programs are needed for performance evaluation.
- The mapping of algorithmic and data structure onto the machine architecture includes processor scheduling, memory maps, interprocessors communication, etc., these activities are usually architecture dependent.

SYSTEM SOFTWARE SUPPORT:

- Software support is needed for the development of efficient programs in high level language. The source code written in a HLL must be first translated into object code by an optimizing compiler. A loader is used to initiate the program execution through the OS kernel.
- Resource binding demands the use of the compiler, assembler, loader and OS kernel to commit physical machine resources to program execution. Ideally, we need to develop a parallel programming environment with architecture independent languages compilers and software tools.

COMPILERS SUPPORT:

- These are three compiler upgrade approaches pre-processor, precompiled and parallelizing compiler. A pre-processor uses a sequential and a low level library of the target computer to implement high level parallel constructs.
- The efficiency of the binding process depends on the efficiency of the binding process depends on the effectiveness of the pre-processors, the precompiled, the parallelizing compiler, the loader and the OS support. This has been proven useful in enhancing the performance of parallel computers.

EVOLUTION OF COMPUTER ARCHITECTURE:

- The study of computer architecture involves both hardware organization and programming software requirements. As seen by an assembly language programmer, computer architecture is attracted by its instruction set, which include opcode, addressing modes, registers, virtual memory etc..,

- From the hardware implementation point of view, the abstract machine is organized with CPUs, caches, buses, microcode, pipeline, physical memory etc..,

LOOKAHEAD, PARALLELISM AND PIPELINING:

- Look ahead techniques were introduced to protect instructions order to overlap I/E operations and to enable functional parallelism. Functional parallelism was supported by two approaches.
- One is to use multiple functional units simultaneously and the other is to practice pipelining at various processing levels.

FLYNN'S CLASSIFICATION:

- Michael Flynn(1972) introduced a classification of various computer architectures based on notions of instructions and data streams.
- As conventional sequential machines is called SISD(Single instruction stream over a single data stream) computers. The parallel computers are reserved for MIMD (Multiple instructions stream over multiple data streams) machines.
- The SIMD and MISD models are more suitable for special-purpose computations. For this reasons MIMD is the most popular model, SIMD next and MISD the least popular model being application in commercial machines.

PARALLEL/ VECTOR COMPUTERS:

- Intrinsic parallel computers are those that executive programs in MIMD model. There are two major classes of parallel computers namely, shared-memory multiprocessors and message passing multicomputer.
- Memory-to-memory architecture supports the pipelined flow of vector operands directly is mostly machine- dependent. The Linda approach using tuple spaces offers an architecture-transparent communication level for parallel computers.

NEW CHALLENGES:

The technology of parallel processing is the outgrowth of several decades of research and industrial advances in microelectronics, printed circuits, high density packaging, advanced processors, memory system peripheral devices, communication channels, languages

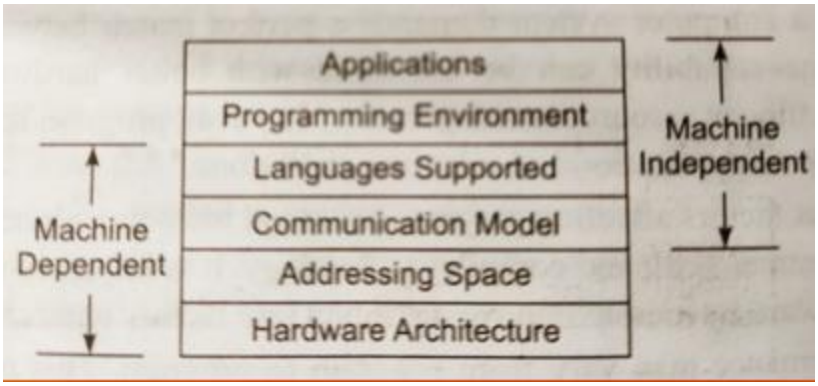
evolution, compiler sophistication, operating system, programming environments and application challenges.

SYSTEM ATTRIBUTES TO PERFORMANCE:

- There are also many other factors affecting program behaviour, including algorithm design, data structure language efficiency, programmer skills and compiler technology, It is impossible to achieve a perfect match between hardware and software by merely improving only a few factors without touching other factors.
- Consider the execution of a given program on a given computer. The simplest measure of program performance is the turnaround time, which includes disk and memory access, input and output activities compilation time, OS overhead and CPU from the memory pipelines and then back to the memory pipelines and then back to the memory. Register-to-register architecture use vector register to interface between the memory and functional pipelines.

DEVELOPMENT LAYERS:

A layered development of parallel computer is illustrated based on a classification by Lionel Ni (1990). Hardware configurations differ from machine to machine even those of the same model. The address space of a processor in a computer system varies among different architectures. It depends on the memory organization, which is machine dependent.



Programming languages such as Fortran, c, c++, Pascal, Ada, Lisp and others can be supported by most computers. However the communication models, shared variable versus message passing.

SYSTEM ATTRIBUTES:

The above five performance factors (IC , p , m , k , I) are influenced by four system attributes, instruction set architecture, compiler technology, cup implementation and control cache and memory hierarchy.

CLOCK RATE AND CPI:

The cup (or simply the processor) of today's digital computer is driven by a clock with a constant cycle time. The inverse of the cycle time is the clock rate. The size of a program is determined by its instruction count (Te). In terms of the number of machine instruction to be executed in the program.

PERFORMANCE FACTORS:

Let IC be the number of instructions in a given program, or the instruction count. The cup time (T in seconds/program) needed to execute the program is estimated by finding the product of the contributing factors.

$$T=IC *CPI * J$$

The CPI of an instruction type can be divided into two component terms corresponding to the total processor cycles and memory cycles needed to complete the execution of the instruction. Depending on the instruction cycle may involve one to as many as four memory references.

$$T=IC (P + M+ K) * J$$

The based can also be written as $T=IE*10^{-6}/MIPS$. Based on the system attributes identified and the above derived expressions, we conclude by indicating the fact the MIPS rate of a given computer is directly proportional to the clock rate and inversely proportional to the CPI.

FLOATING POINT OPERATIONS PERSECOND:

Most complete intensive applications in science and engineering make heavy the use of floating point operations. With prefix mega (10^6), giga (10^9), tera (10^{12}) or peta(10^{15}) this is written as megaflops(m flops),gigaflops(g flops), teraflops or pet flops.

THROUGH PUT RATE:

Another important concept is related to how many programs a system can execute unit time called the system throughput Ws. In a multi programmed, system the system throughput is often lower than the CPU throughput WP defined by

$$WP = F/IC*CPI$$

Note that WP= (MIPS) * 10_6/IE. The unit for WP is also programs/ second the CPU throughput is a measure of how many programs can be executed per second, based only on the MIPS rate and average program length (IE). Usually Ws<Wp due to the additional system overheads caused by the I/O compiler and OS when multiple programs are interleaved for CPU executing by multiprogramming or time sharing operations.

PROGRAMMING ENVIRONMENTS:

- The programmability of a computer depends on the programming environment provided to the users. In fact the marketability of any new computer system depends on the creation of a user friendly environment in which programming becomes a productive understanding rather than a challenge.
- In fact, the original UNIX/OS kernel was designed to respond to one system call form the user process at a time. Successive system calls must be serialized through the kernel.

IMPLICIT PARALLISM:

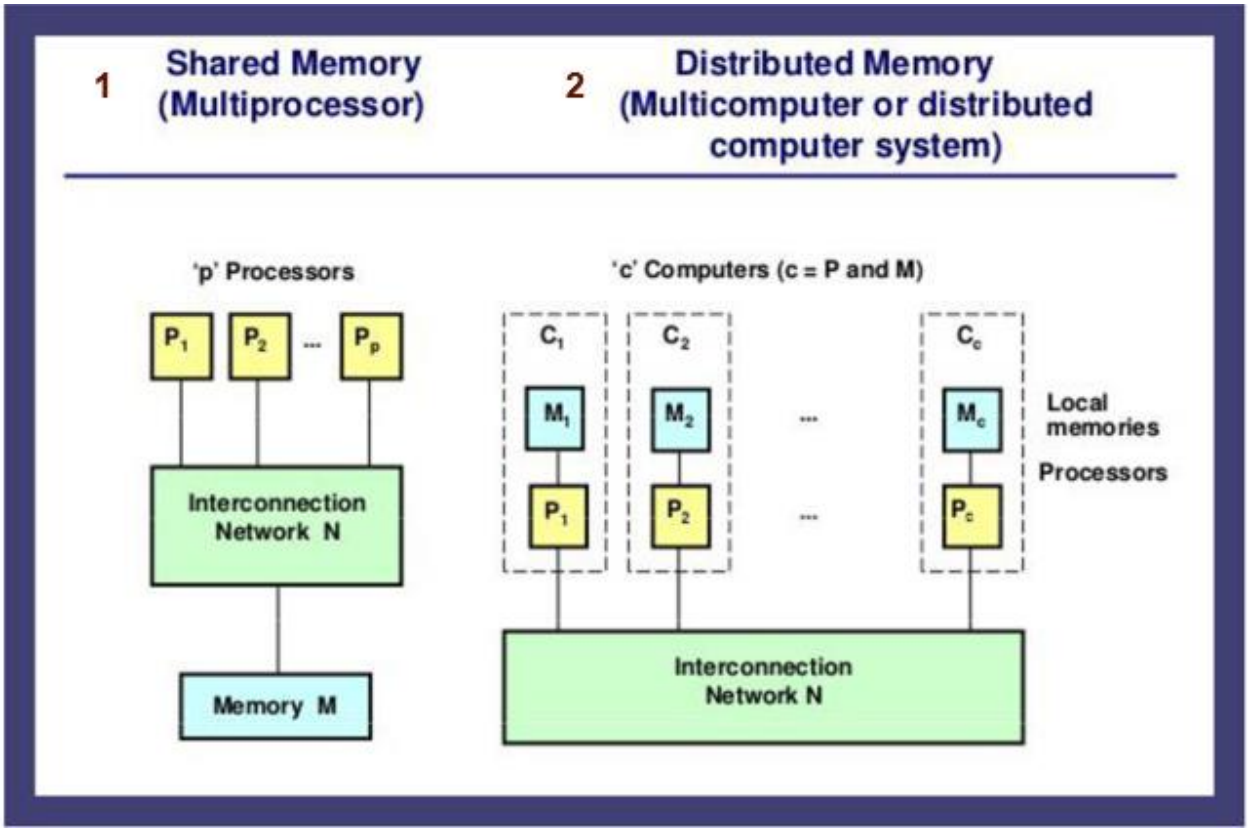
- An implicit approach uses a conventional language. Such as c, c++, Fortran or Pascal to write the source program with parallelizing compiler. Being implicit success relies heavily on the “intelligence” of a parallelizing compiler.
- This approach requires less effort on the part of the programmer. providing different levels of program abstraction validation, testing , debugging and lurning performance prediction and monitoring and lurning performance prediction and monitoring and visualization support to aid program development, performance measurement and graphics display and animation of compaction results.

MULTIPROCESSORS AND MULTICOMPUTER:

Two categories of parallel computer are architecturally modelled below. These physical models are distinguished by having a shared common memory or unshared distributed memories.

SHARED-MEMORY MULTIPROCESSORS:

- We describe below three shared-memory multiprocessors models.
- Uniform memory- access (UMA) Model
- Nonuniform memory- access (NUMA) Model
- Cache-only memory architecture (COMA) Models.



THE UMA MODELS:

In UMA multiprocessor model, the physical memory is uniformly shared by all the processors. All processors have equal access time to all memory words, which is why it is called uniform memory access.

EXPLICIT PARALLELISM:

- The second approach requires more effort by the programmer to development a source Program using parallel deducts of c, c++, Fortran or Pascal. Parallelism is explicitly specified in the user programs. This reduces the burden on the compiler to detect parallelism.
- Special software tools are needed to make an environment more friendly to user groups. Some of the tools are parallel extensions of conventional high-level languages. Others are integrated environments which include tools memories are converted to caches.
- Besides the UMA, NUMA and COMA specified above other variation exist multiprocessors.

REPRASENTATIVE MULTIPROCESSOR:

- Sever early commercially available multiprocessor are summarized. They represent four classes of multiprocessor. The sequent summitry s81 belonged to a class near-upper computer , the BBN TC-2000 represented the MPP class.
- Machine models are depicted. The shared memory is physically distributed to all processors called local memories. The collection of all local memory forms a global address space accessible by all processors.
- All clusters have equal access to the global memory. The cedar multiprocessor, built at the University of Illinois, had such a structure in which cluster was an Alliant FX 180 multiprocessors.

THE COMA MODEL:

- A multiprocessor using cache-only memory assumes the COMA model. Early examples of COMA machines include the Swedish Institute of computer science's data dittusion machine and tend all square research's KSR-1 machine.

- The COMA model is a special case of a NUMA machine, in which the distributed main. The boundary router may be connected to I/O and peripheral devices. Message passing between any two modes involves sequences of router and channel, mixed types of nodes are allowed in a heterogeneous multicomputer.
- The intermodal communication in a heterogeneous multicomputer. Is achieved through compatible data representation and message- passing protocols.

REPRESENTSTIVE MULTICOMPUTER:

Three early message-passing multicomputers are summarized with distributed processor/ memory nodes, such machines are better in achieving a scalable performance.

DISRIBUTED- MEMORY MULTICOMPUTERS:

A distributed-memory multicomputer system is modelled. The system consists of multiple computers often called nodes, interconnected by a message passing network. Each node is an autonomous computer considering of a processor. Local memory and sometimes attached disks or I/O peripherals.

MULTICOMPUTER GENERATION:

Modern provides point-to-point static connection among the nodes. All local memories are private and are accessible only by local processors. Suppose each line of code L2, L4 and L6 takes machine cycle to execute. The time required to execute the program control statements L1, L3, L5 and L7 is ignored to simplify the analysis, Assume that K cycles are needed for each interprocessor communication operation via the shared memory.

```
Do all      K=I.4
 Do         10= (K-1) * L+1, K* 1
            A (I) =B (I) + c (I)
            Continue
                  Sum (K) =0
                   Do 20=1.L
```

Sum (K) = Sum (K) + A ((K-1) * L+ J))

20 Continue

End all

The addition of each pair of partional sums requires K cycles through the shared Memory. An I-level binary adder tree can be constructed to merge all the partial sums, where I=log $_2$M. The adder tree takes I (K+1) cycles to merge the partial sums sequentially from the leaves to the root of the tree. Therefore, the multiprocessors requires 2L+1(k+1) =2N (m+ck+1) $\log_2$M cycles to produce the final sum.

THE NUMA MODELS:

- A NUMA multiprocessor is a shared-memory system in which the access time varies with the location of the memory word.
- Some computer manufactures have multiprocessor (MP) extensions of their uniprocessor (VP) product time. The UMA model is suitable for general purpose and time-sharing application by multiple users. To coordinate parallel events. Synchronization and communication among processors are done through using shared variable in the common memory.

APPROXIMATED PERFORMANCE OF A MULTIPROCESSOR:

This example exposes the reader to parallel program execution on a shared memory multiprocessor system. Consider the following fortran program written for sequential execution on a uniprocessor arrays A (I), B (I) and C (I) are assumed to have N elements.

L1: Do 10 I=1,

L2: A (I) =D (I) +C (I)

L3:10 Continue

L4: Sum=0

L5: Do 20 J=1, 2

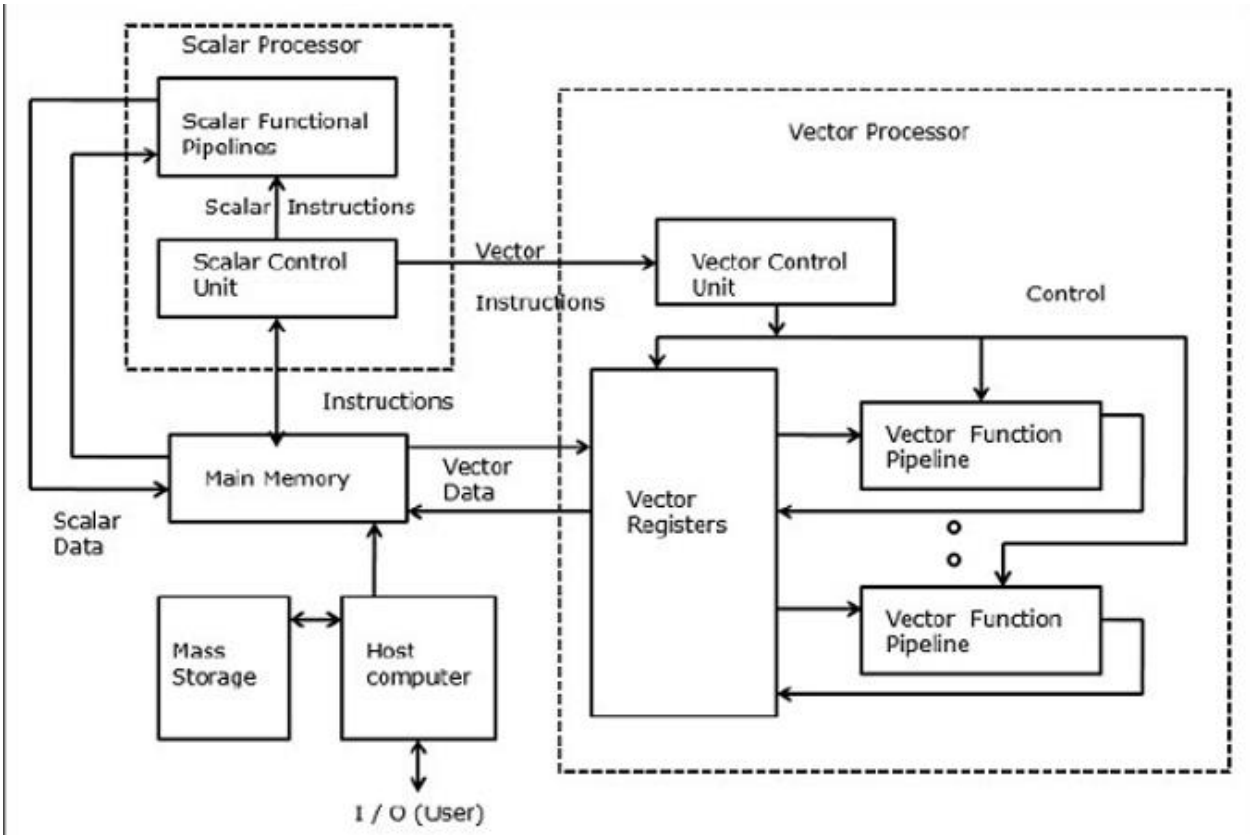
L6: Sum=Sum +A (J)

MULTIVECTOR AND SIMD COMPUTER:

We classify supercomputer either as pipelined vector machine using powerful processors equipped with vector hardware, or as SIMD computer emphasizing massive data parallelism.

VECTOR SUPER COMPUTER:

A vector computer is often built on top of scalar processors. Program and data are first loaded into main memory through a host computer, all instruction are first decoded by the scalar control unit.



VECTOR PROCESSOR MODELS:

- Vector register are used to hold the vector operands, intermediate and final vector results.
- The vector functional pipeline retrieves operands from and put results into the vector operands from and put results into the vector register. All vector register are

programmable in user instruction. Each vector register is equipped with a component register used in successive pipeline cycles.

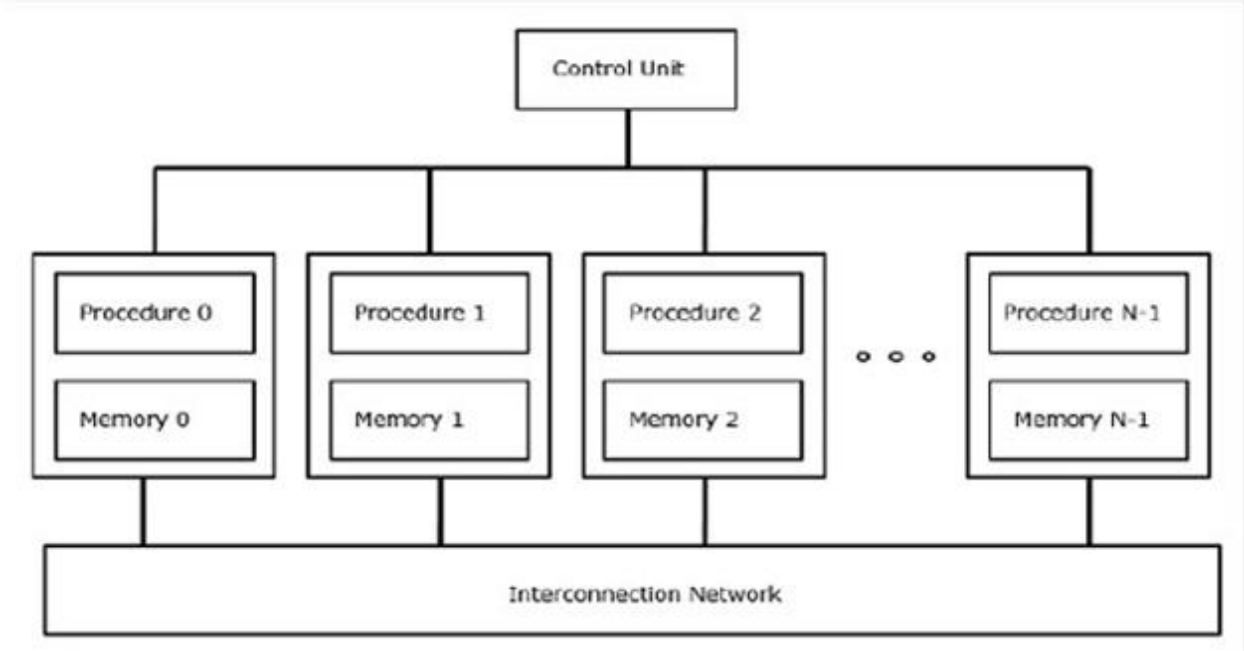
- The length of each vector register is usually fixed, say 64-bit component register in a vector register in a Cray series super computer.

REPRESENTATIVE SUPERCOMPUTERS:

Over a dozen pipelined vector computers have been manufactured, ranging from workstation to mini and supercomputers. The star dent 3000 multiprocessor equipped with vector pipelines the convex C3 series, the DEC UAX 9000, the IBM 390/VF, the Cray research Y-MP family, the NEC SX series the jujitsu VP 2000 , and the Hitachi S-810/20.

SIMD SUPPERCOMPUTERS:

An operational model of SIMD computers is presented the based on the work of H.J Siegel.



An operational model of an SIMD computer is specified by a 5-tuple

$$M = (N, C, I, M, R)$$

Where,

(1) N is the number of processing elements (PES) in the machine. For example, the Iliac 1v had 64 PES and the connection machine CM-2 had 65,536 PES.

(2) C is the set of instruction directly executed by the control unit (CU), including scalar and program flow control instruction.

(3) I is the set of instruction broadcast by the CU to all PEs for parallel execution. These include arithmetic, logic, data routing, masking and other local operations PE over data within that PE.

(4) M is the set of masking schemes, where each mask portions' the set of PES into enabled and disable subsets.

(5) R is the set of data-routing functions specifying various patterns to be set up in the interconnection network for inter-PE communication.

REPRESENTATIVE SIMD COMPUTERS:

- Three early commercial SIMD computers are summarized. The number of Peps in these systems from 4096 in the DAP610 to 16,384 in the masper MP-1 and 65,356 in theCM-2. Both the CM-2 and DAP 610 were fine gain, bit-slice SIMD computers with attached floating-point acceleration for blocks of PES.
- The CM-2 implementation 16 PES as a mesh on a single chip. Each 16-PE mesh placed at one vertex of a 12 dimensional hypercube. Thus $16*12^{12}=2^{16}=65,536$ PES formed the entire SIMD array.

UNIT -2

PROGRAM AND NETWORK PROPERTIES

- Conditions of parallelism
- Program partitioning and scheduling
- Program flow mechanisms
- System interconnect architectures

CONDITIONS OF PARALLELISM:

- The exploitation of parallelism has created a new dimension in computer science. In order to move parallel processing into the mainstream of computing.
- A theoretical treatment of parallelism is thus needed to build a basis for the above challenges. In practise parallelism appears in various forms in a computing environment.

DATA AND RESOURCE DEPENDENCES:

- The ability to execute several program segments in parallel requires each segment to be independent of the other segments.
- We use a dependence graph to describe the relations. The nodes of a dependence graph correspond to the program statements and the directed edges with different labels show the ordered relations among the statements.

DATA DEPENDENCE:

The ordering relationship between statements is indicated by the data dependence. Five types of data dependence are defined below.

- Flow dependence
- Antidependence
- Output dependence
- I/O dependence
- Unknown dependence

FLOW DEPENDENCE:

A statement S2 is flow dependent on statement s1 it an execution path exists from s1 to s2 and it at least one output of s1 feeds in as input s2. Flow dependence is depended as $s1 \rightarrow s2$.

ANTIDEPENDENCE:

Statement s2 is antidependence on statement s1 it s2 follows s1 in program order and it the output of s2 overlaps the input to s1.

OUTPUT DEPENDENCE:

Two statements are output dependent it they produce the same output variable $s1 \rightarrow s2$ indicates output dependence from s1 to s2.

I/O DEPENDENCE:

Read and write are I/O statement I/O dependence occurs not because the same variable is involved but because the same life is referenced by both I/O statements.

UNKNOWN DEPENDENCE:

1) The subscript of a variable is itself subscribed.
2) The subscript does not contain the loop index variables.
3) A variable appears more than once in with subscript having different coefficients of the loop variable.
4) The subscript is nonlinear in the loop index variable.

CONTROL DEPENDENCE:

This refers to the situation where the order of execution of statements cannot be determined before run time. For example, conditional statements will not be resolved until run time. In the following, we show one loop example with and another without control dependent iterations.

```
Do      201= 1, N
        A (1) = c (I)
        IF (A (I).L.T.O) A (1) =1
```

```
20        Continue
```

The following loop has control dependent iterations:

```
Do        101=1, N
          IF (A (I-1). EQ.O) A (I) =0
10        Continue
```

Control dependence often prohibits parallelism from being exploited.

RESOURCE DEPENDENCE:

This is different from data or control dependence, which demands the independent of the work to be done. Resource dependence is connected with the conflicts in using shared resources.

BERNSTEIN'S CONDITION:

- In 1966 Bernstein revealed a set of conditions based on which two process carexcault in parallel.
- Similarly, the output set consists of an output variable generated after execution of the process.
- Now, consider two process p1 and p2, I1 and I2, Q1 and Q2.

Formally, these conditions are stated follow:

$$I1 \cap O2 = \emptyset$$

$$I2 \cap O2 = \emptyset$$

$$01 \cap O2 = \emptyset$$

PROGRAM PARTITIONING AND SCHEDULING:

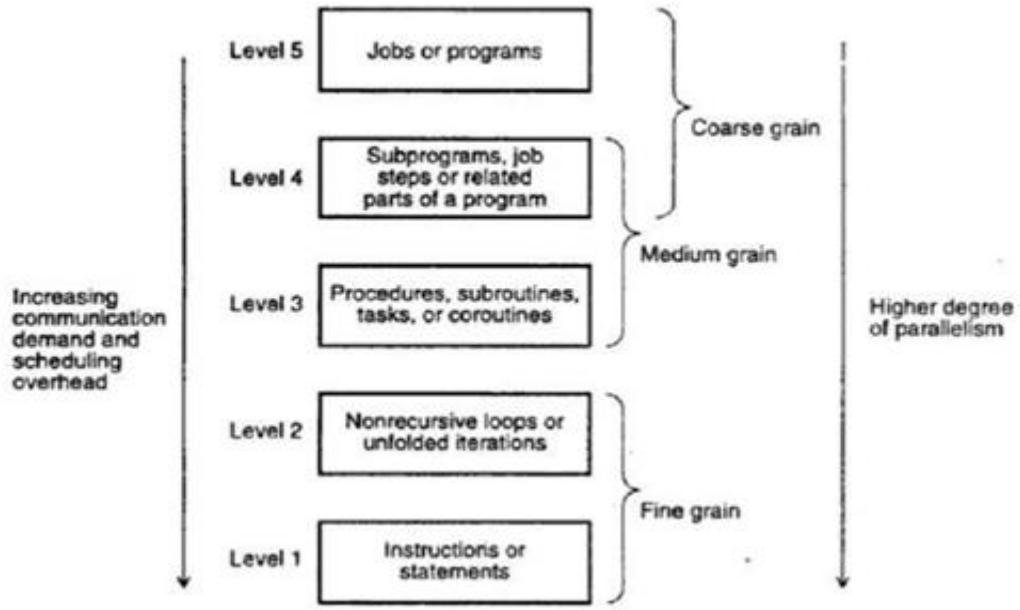
GRAIN SIZES AND LATENCY:

- Grain size or granularity is a measure of the amount of computations involved in a software process.
- The simplest measure is to count the number of instructions in a grain.

- Latency is a time measure of the communications overhead incurred between machine subsystems.

INSTRUCTION LEVEL:

At the lowest level. A typical grain consists less than 20 instructions called fine grain. Depending on individual program. Depending on individual program fine-grain parallelism at this level may range from two thousand. But level has shown single-instruction stream parallelism is greater than two, wall finds that the average parallelism at instruction level is ground fine environment.



LOOP LEVEL:

This corresponds to the interactive loop operations. A typing loop contains less than 500 instructions.

PROCEDURE LEVEL:

This level corresponds low medium- grain parallelism at the task procedural, subroutine continue level.

SUBPROGRAM LEVEL:

This corresponds to the level of job steps and related subprogram.

JOB LEVEL:

This corresponds of the parallel execution essentially into parallel computer. The grain size can be as high as millions of instructions in a single program.

COMMUNICATION LATENCY:

By balancing granularity and latency, one can achieve better performance of computer system. Various laterices are attributes to machine architecture, implementing technology and communication patterns involved.

GRAIN PACKING AND SCHEDULING:

Two fundamentals question to art in parallel program are.

1) How can is we partition a program in parallel branches.
2) What's the optimal size of concurrent grains computation?

This grain-size problem demands determine of both the number and the size of grain in a parallel program.

STATIC MULTIPROCESSORS SCHEDULING:

DODE DUPLICATION:

- In order to laminate the idle time and to further reduce the communications blacks among processor, one can duplicate some of the node in more one processor.
- Four major steps are involved in the grain determination and the process of scheduling optimization.

STEP 1: Construct a fine-grain program graph.

STEP 2: Schedule the fine-grain computation.

STEP3: Perform grain packaging to produce the course grains.

STEP4: Generate a parallel schedule based on the packed graph.

PROGRAM FLOW MECHANISMS:

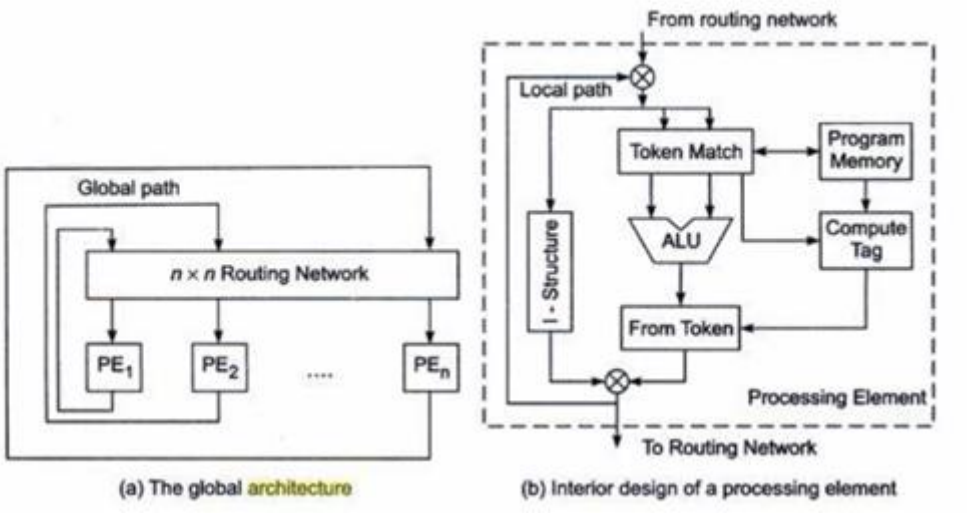
- Conventional computers are based on a control flow mechanism by which the order of program execution is explicitly stated in the user programs.
- Data flow computers emphasize a high degree of parallelism at the finr-grain instructional level.

CONTROL FLOW VERSUS DATA FLOW:

- Conventional V on Neumann computer use a program counter (PC) to sequence the execution of instructions flow in a program. This sequential execution style has been called control-driven. As program flow is explicitly controlled by programmers.
- In a dataflow computer, the execution of an instruction is driven by data availability instead of being guided by a program counter. The instruction in a data- driven program is not ordered in any way.
- Computational results (data tokens) are passed directly between instructions. The data generated by an instruction will be duplicated into many copies and forwarded directly to all needy instructions.

DATAFLOW ARCHITECTURE:

There are having been quite a few experimental dataflow computer projects arrived and his associates at MIT developed a tagged token architecture for building dataflow computers. The global architecture consists of N processing elements (PES) interconnected by an n×n roaling network.



Another synchronization mechanism called the 1-structure is provide within each PE. The 1-structure is tided memory unit for ever lapped usage of a data structure by both producer and consumer processes.

DEMAND DRIVEN MECHANISMS:

In a reduction machine, the computation is triggered by the demand for an operation's result. Consider the evaluation of a nested arithmetic expression $a = ((b + 1) \times c - (d \div e))$. The data driven computation seen above choose a bottom-up approach starting from the intermost operation b+1 and $d \div e$, then proceding to theX operation and finally to the outermost operation$-$.

A demon-driven computation chooses a top down approach, because operations are executed only with their results are required by another instruction.

REDUCTION MACHINE MODELS:

In a graph reduction model, the expression is represented as a directed graph. The graph is reduced by evaluation of branches or sub graphs can be reduced or evaluated in parallel upon demand.

COMPARISON OF FLOW MECHANISMS:

- The degree of explicit control decreases from control driven to demand driven.
- Furthermore, control tokens are used in control flow computers and reduction machines, respective the listed advantages and disadvantages of the reduction

machine models are based on research finding rather than on extensive operational experience. Models are based on research finding rather than on Extensive operational experience.

Machine Model	*Control Flow (control-driven)*	*Dataflow (data-driven)*	*Reduction (demand-driven)*
Basic Definition	Conventional computation; token of control indicates when a statement should be executed	Eager evaluation; statements are executed when all of their operands are available	Lazy evaluation; statements are executed only when their result is required for another computation
Advantages	Full control The most successful model for commercial products	Very high potential for parallelism	Only required instructions are executed
	Complex data and control structures are easily implemented	High throughput	High degree of parallelism
		Free from side effects	Easy manipulation of data structures
Disadvantages	In theory, less efficient than the other two	Time lost waiting for unneeded arguments	Does not support sharing of objects with changing local state
	Difficult in preventing run-time errors	High control overhead	Time needed to propagate demand tokens
		Difficult in manipulating data structures	

SYSTEM INTERCONNECT ARCHITECTURES

Static and dynamic networks for interconnecting computer subsystem or for constructing multiprocessors or multicomputer are introduced below. These networks can be used for internal connections. Among processors, modules and I/O adaptors in centralized system multicomputer nodes.

NETWORK PROPERTIES AND ROUTING:

The topology of an interconnection network can be either static or dynamic. Static networks are formed of point-to-point point direct connections which will not change during program execution. Dynamic networks include buses, crossbar switches, multistage networks and routers which are often used in shared memory multiprocessors.

- Node degree and network diameter
- Bisection width
- Data routing functions
- Permutations
- Perfect shuffle and exchange
- Hypercube routing functions
- Broadcast and multicast
- Network performance

NODE DEGREE AND NETWORK DIAMETER:

The number of edges (links or channels) incident on a node is called the node degree d. In the case of indirections channels, the number of channels into a node degree reflects the degree and that out of node degree reflects the degree and that out of node in the out degree.

BISECTION WIDTH:

When a given network is cut into two equal halves, the minimum number of edges along the cut is called the channels bisection width b. Then the wire bisection width is B=BW. This parameter B is fixed the channel width (in bits) W=B/b.

DATA ROUTING FUNCTIONS:

A data routing network is used for inter PE data exchange. This routing network can be static, such as the hypercube routing network used in the TMC/CM-2, dynamic such as the multistage network used in the IBM GFII.

Commonly seen data routing function among the PES include shifting, rotation permutation (one-to-one), broadcast (one-to-all), multicast (one-to-many), shuffle exchange etc..,

PERMUTATIONS:

- N objects, there are n! Permutation by which the n objects can be reordered. The set of all permutations from a permutation group with respect to composition operation.
- One can use cycle notation to specify a permutation function.

PERFECT SUFFLE AND EXCHANGE:

Perfect shuffle is a special permutation function suggested by Harold Stone (1971) for parallel processing application.

HYPERCUBE ROUTING FUNCTIONS:

A three dimensional binary cube network. Three routing function are defined by three bits in the node address.

For example, one can exchange the data between adjacent nodes which differ in the least significant bit C0.

BROADCAST AND MULTICAST:

Broadcast is a one-to-all mapping. This can be easily achieved in an SIMD computer using a broadcast bus extending from the array controller to all PES.

A message passing multicomputer also has mechanisms to broadcast messages. Multicast corresponds to a mapping from one PE to other PES (one-to-many).

NETWORK PERFORMANCE:

To summarize the above discussion the performance of an interconnection network is affected by the following factors.

1) **FUNCTIONALITY:**

This refers to how the network supports data routing, interrupt handling, synchronizaization, request/ message combining and coherence.

2) **NETWORK LATENCY:**

This refers to the worst case time delay for a unit message to be transferred through the network.

3) **BANDWIDTH:**

This refers to the maximum data transfer rate, in terms of Mbytes/s through Gbytes/s, transmitted through the network.

4) **HARDWARE COMPLEXITY:**

This refers to implementation costs such a those for wires, switches, connectors, arbitration and interface logic.

5) **SCALABILITY:**

This refers to the ability of a network to be modularity expandable with a scalable performance with increasing machine resources.

STATIC CONNECTION NETWORKS:

Static networks use direct links which are fixed once built. This type of network is more suitable for building computer where the communication patterns are predictable or implementable with static connections.

LINEAR ARRAY:

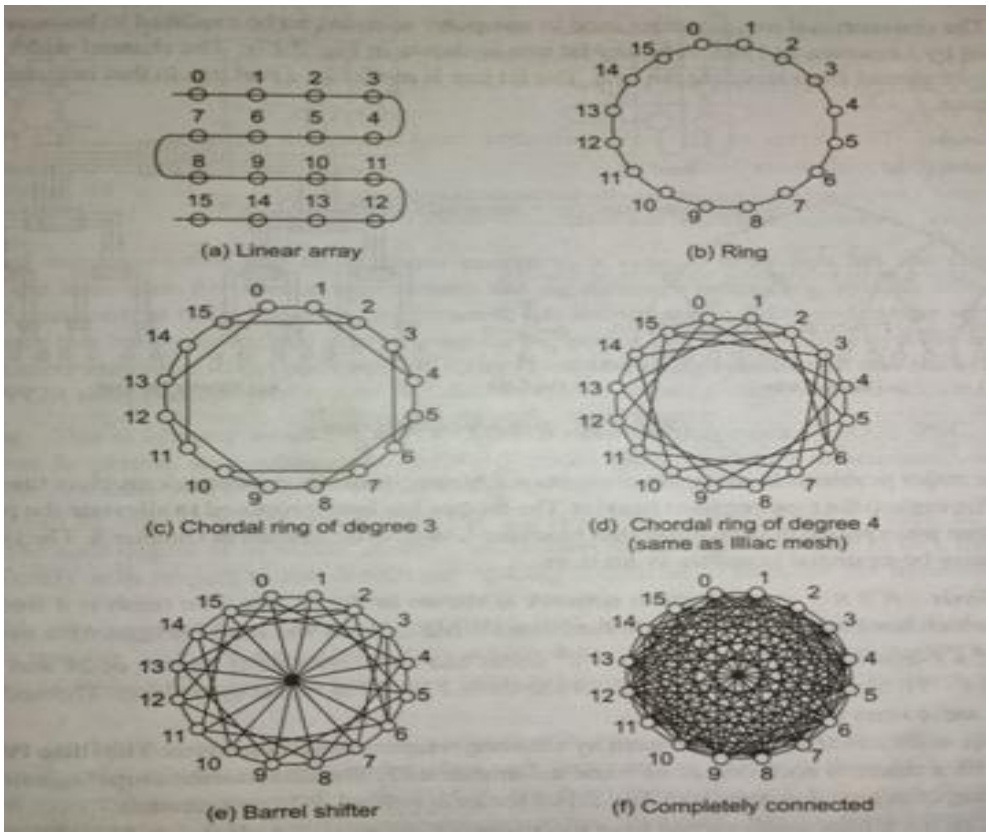
This a one-dimensional network in which N nodes are connected by N-1 links in a line. Internal nodes have degree 2, and the terminal nodes have degree 1, the diameter is N-1 which is rather long for large N. The bisection width b=1.

RING AND CHORDAL RING:

- A ring is obtaining by connection the two terminal nodes of a linear array with one extra link. A ring can be unidirectional or bidirectional.
- It is symmetric with a constant node degree of 2. The diameters is (N/2) for a bidirectional ring, and N for unidirectional ring.

BARREL SHIFTER:

The barrel shifter is obtained from the ring by adding extra links from each node to those nodes having a distance equal to an integer power of 2.

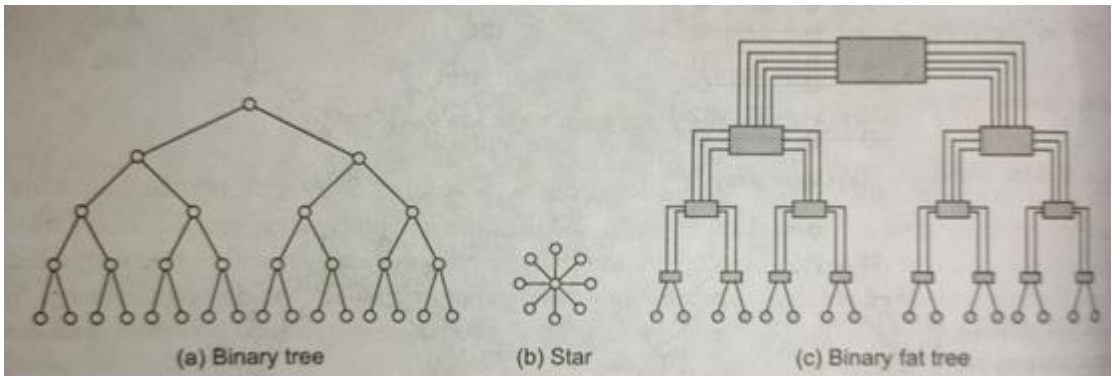


TREE AND STAR:

A binary tree of 31 nodes in five levels. In general, a k level, completely balances binary tree should have is 3 and the diameter is 2 (k-1) with a constant mode degree the binary tree is a scalable architecture.

FAT TREE:

The conventional tree structure used in computer science can be modified to become the fat tree as introduced by leadsperson in 1985 the channel width of a fat root the idea of binary fat trees can also be extended to multiway fat trees.



MESH AND TORUS:

- A 3*3 example mesh network is shown. The mesh is a frequently used architecture which has been implemented in the Iliac IV MPP DAP and Intel paragon with variations.
- In general a k-dimensional mesh $N=n^K$ nodes has an interior node degree of 2k and the network diameter is k (n-1) the node degrees at the boundary and corner nodes are 3 or 2.

SYSTOLIC ARRAYS:

In general static systolic arrays are pipeline with multidirectional flow of data streams the commercial machine Intel in app system was designed with a systolic architecture.

HYPER CUBES:

This is a binary n-cube architecture which has been implemented in the IPSc, n cube, and cm-2 systems. In general an n-cube consists of $n=2^n$ nodes spanning along n dimensions with two nodes per mention.

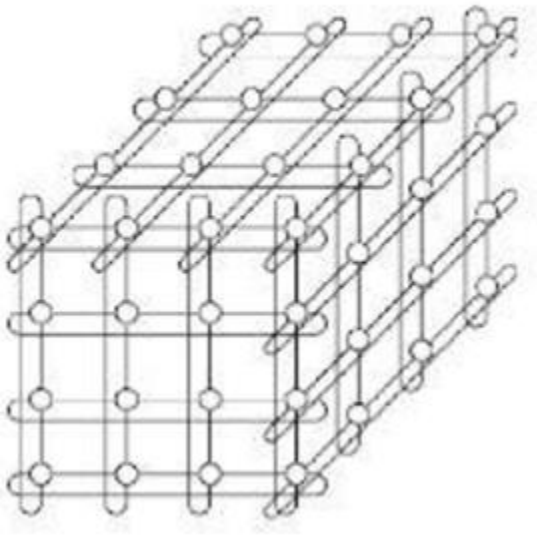
CUBE – CONNECTED CYCLES:

This architecture is modified from the hypercube. A 3- cube is modified to from 3-cube connected cycle. This idea is to cut off the corner nodes (vertices) of the 3-cube and replace each by a ring (cycle) of 3 nodes.

A k-cube can be thus transformed to a k-ccc with $k*2^k$ nodes.

K-ARG N-CUBE NETWORKS:

Rings, meshes, tori binary n-cubes (hyper cubes) and omega networks are topologically isomorphic to a family of k- ary n-cube networks.



The parameters n is the dimension of the cube and k is the radix or the number of nodes along each dimension.

NETWORK THROUGHPUT:

The network throughput is defined as the total number of message the network can handle per unit time one method of estimating throughput is to calculate the capacity of a network the total number of message that can be in the number network at once.

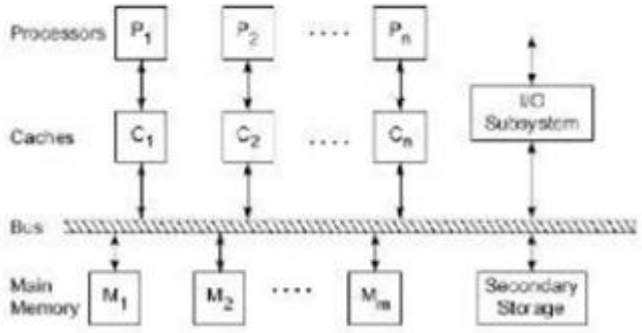
Low dimensional network reduce contention because having in more resource sharing and thus a better querying performance than having many low bandwidth channels. Consider a hyper cube with $N=2^n$ nodes.

DYNAMIC CONNECTION NETWORKS:

Multipurpose or general – purpose application, we many need to use dynamic connections which can implement all communication patterns based on program demands in increasing order of cost and performance, dynamic connection network include bus systems multistage inter connection networks (MIN), and crossbar switch networks.

DIGITAL BUSES:

A bus system in essentially a collection to wires and connector for data transaction among processors memory modules and peripheral devices to the bus the activate or master devices requests to address the memory.

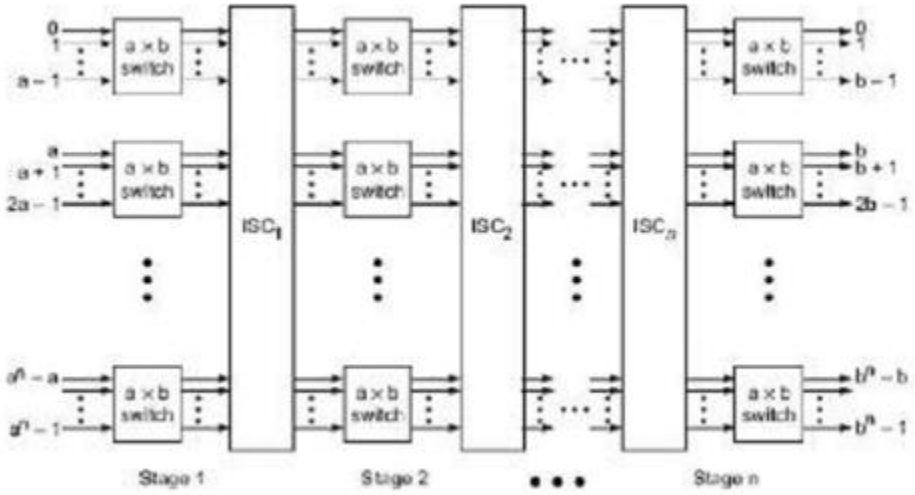


SWITCH MODULES:

An a*b switch module has a inputs and b outputs. A binary switch corresponds to a 2*2 switch module in which a=b=2 in theory a and b do not have to be equal how ever in practice a and b are often chosen as integer powers to 2; that is, $a=b=2^k$ for some K 1

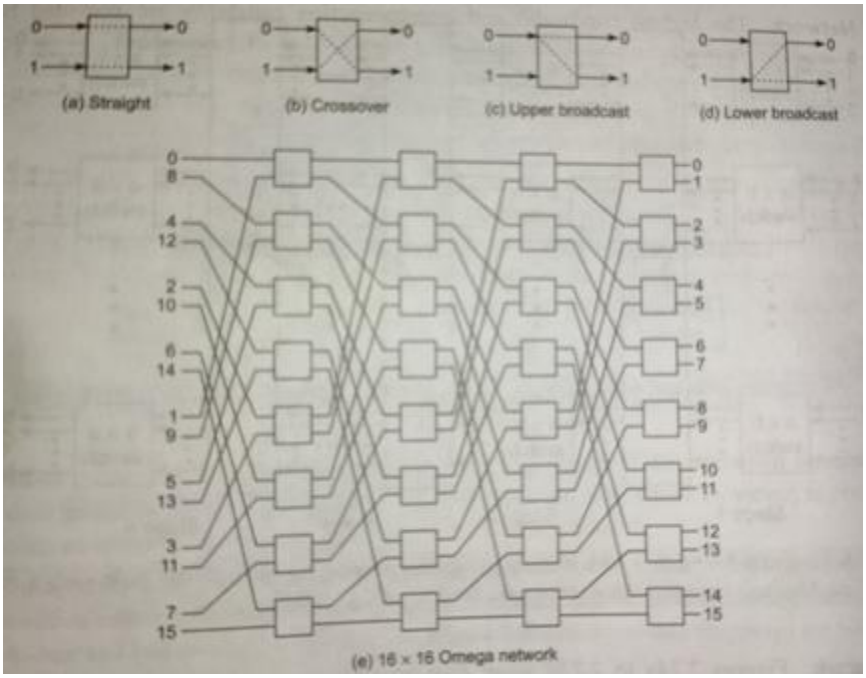
MULTISTAGE INTERCONNECTION NETWORK:

MINS have been used in both MIMD and SIMD computer a number of a*b switches are used in each stage. Fixed Inter stage connections are used between the switches in adjacent stages the switches are simplest module would be the 2*2 switches (a=b=2).



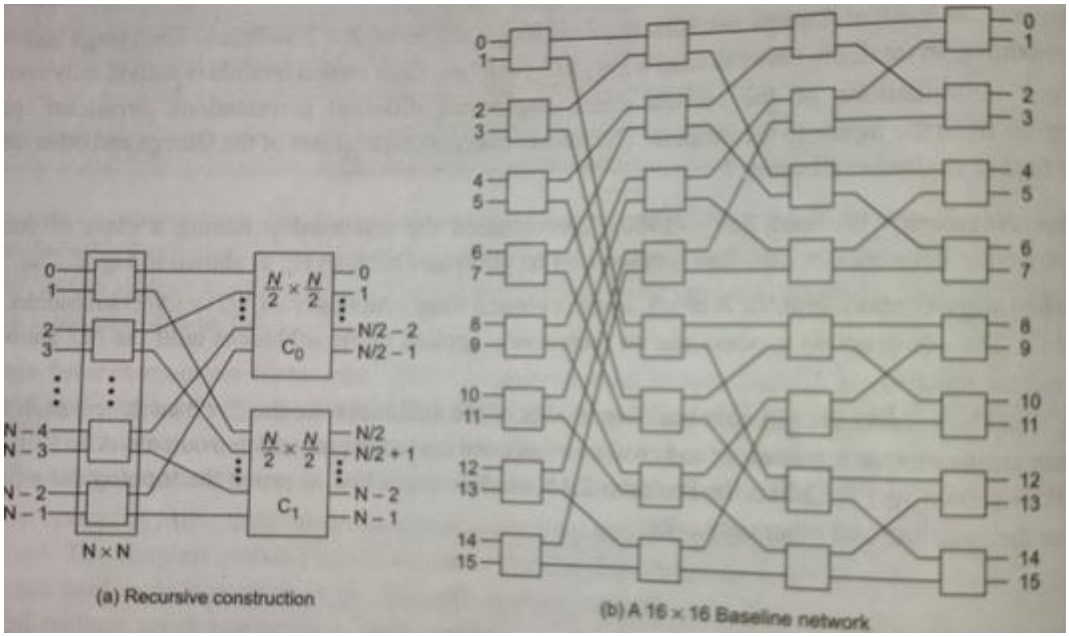
OMEGA NETWORK:

A 16*16 OMEGA NETWORK FOUR STAGES OF 2*2 switches are needed there are 16 inputs on the left and 16 outputs on the right the ISC pattern is the Perfect Shuffle over 16 objects.



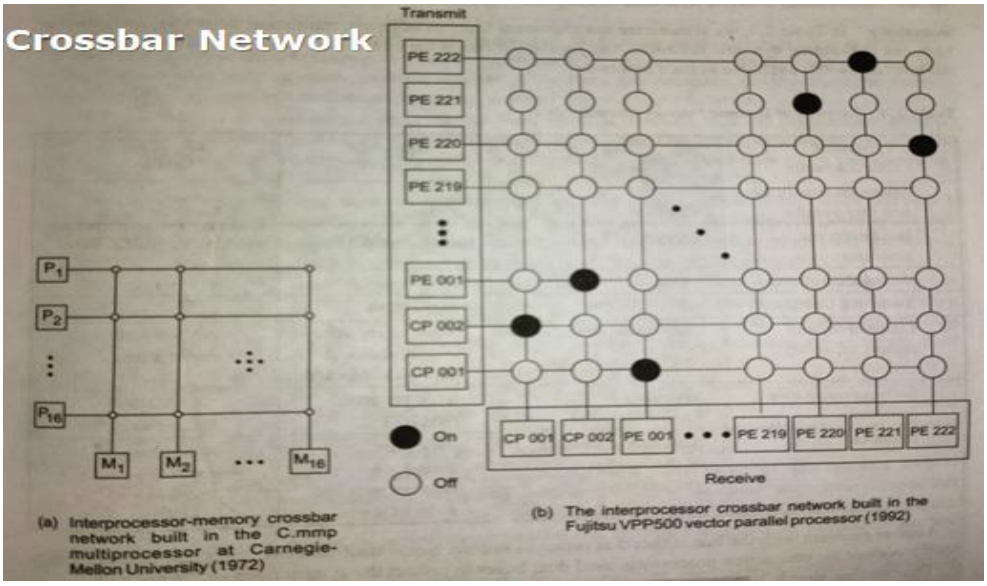
BASE LINE NETWORK:

- Wu and fang (1980) have studied the relationship among a class of multistage inter connection networks.
- The fruit stage contains on n*n.



CROSSBAR NETWORK:

- The highest bandwidth and inter connection capability are provided by crossbar networks. A crossbar network can be virtualized as a single stage switch network.
- To build a shared memory multi processor one can use a crossbar network between the processors and memory modules block and the second stage contains two (N/2)*(N/2) sub blocks labeled c_o and c_1 the construction process can be recursively applied to the sub blocks until the N/2 sub blocks of size 2*2 reached.



- Another type of crossbar network is for inter processor communication and is depicted. This large crossbar (224*224) was actually built in a vector parallel processor (vpp 500) by jujitsu inc. the PES are processors with attached memory. The inter processor crossbar provides permutation connection among the processors.
- Another problem with the bus is that it is prone to failure. Some fault tolerant system like the tandem tolerant system processing used dual buses to protect the system from single failures.
- The crossbar switch is the most be pensive one to built, due to the most hardware complicity increases as n^2 for a small.

UNIT-3

PROCESS AND MEMORY HIERACHY

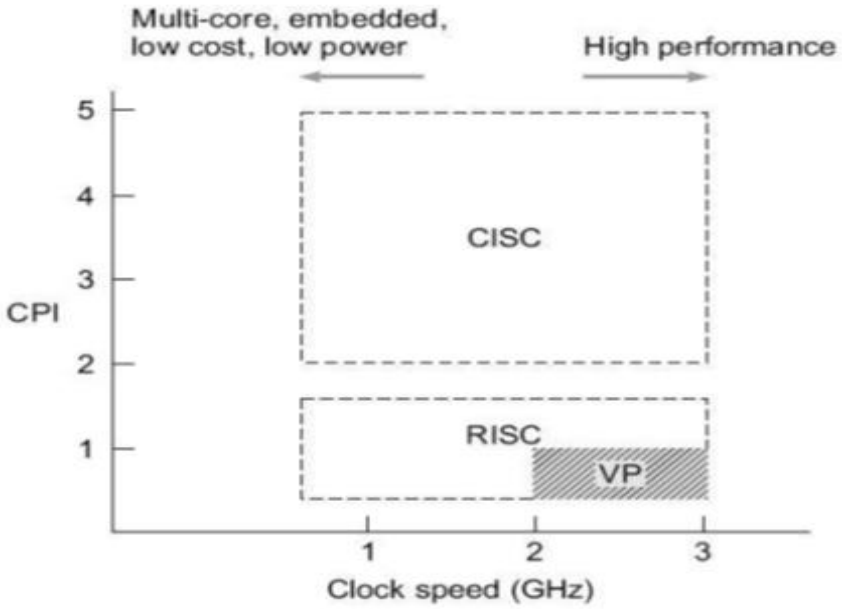
- **Advanced processor technology**
- **Super scalar and vector processors**
- **Linear pipeline processors**
- **Nonlinear pipeline processors**

ADVANCED PROCESSOR TECHNOLOGY:

- Architectural families of modern processors are introduced below, from processors used in workstation or multiprocessors to those designed for mainframes and supercomputer.
- The major processor families to be studied include the CISC, RISC, Superscalar, and VLIM, Super pipeline, vector and symbolic processors.

DESIGN SPACE OF PROCESSORS:

Various processor families can be mapped onto a coordinated space of clock rate versus cycles per instruction (CPI). As implementation technology evolutes reality, the clock rates of various processors have moved from low to higher speeds toward the right of the design space.



DESIGN SPACE

CISC:

- Complex – instruction – set computing.
- Clock rate to today's CISC processor range up to a few GHZ. CPI varies 1-20 upper part design space.

EX: Intel Pentium, IBM 390...

RISC:

- Reduced – instruction – set computing.
- With the use if pipelines it reduced the CPI.

EX: SPARC, Power series, MIPS, Alpha..

SUPER SCALAR PROCESSORS:

RISC subclass processor is super scalar processors, it allows multiple instructions to be issued simultaneously during each cycle. CPI of super scalar processor clock rate of super scalar processors matches that of scalar PISC processor.

VLIW:

- Very long instruction word.
- It has more function units than super scalar processors. CPI is lowered.

EX: Intel's I860 RISC processors

VP:

- Vector processor.
- CPI is very low. Lower right corner, If lower right corner is restricted by processor design cost and power.

INSTRUCTION PIPELINE:

The exception cycle of a typical instruction includes four phases:

- Fetch
- Decode
- Execute
- Write back

There instruction phases are often execute by an instruction pipeline as demonstrate.

PIPELINE OPERATION:

INSTRUCTION PIPELINE CYCLE:

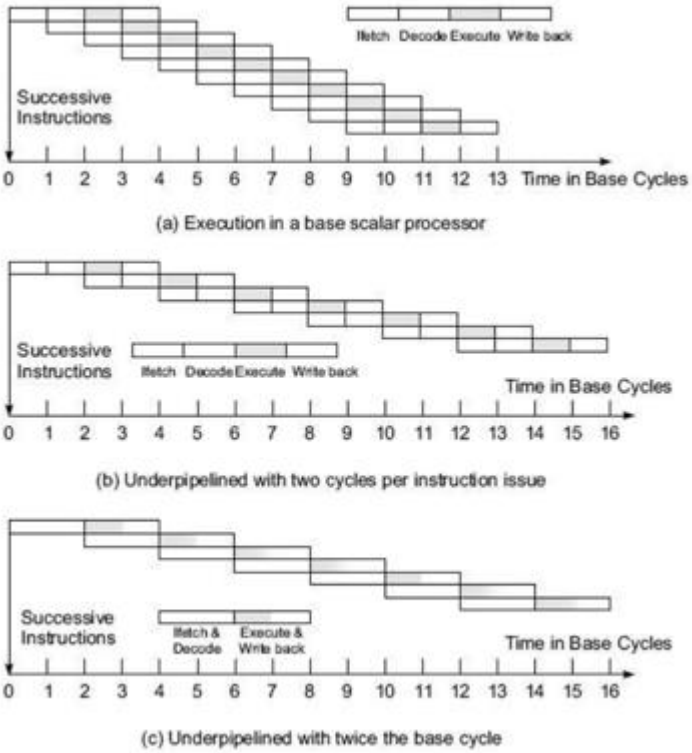
The clock period of the instruction pipeline.

INSTRUCTION ISSUE RATE:

The time required between the issuing of two adjacent instructions.

INSTRUCTION ISSUE RATE:

The number of instructions issued per cycle, also called the degree of a super scalar processor.

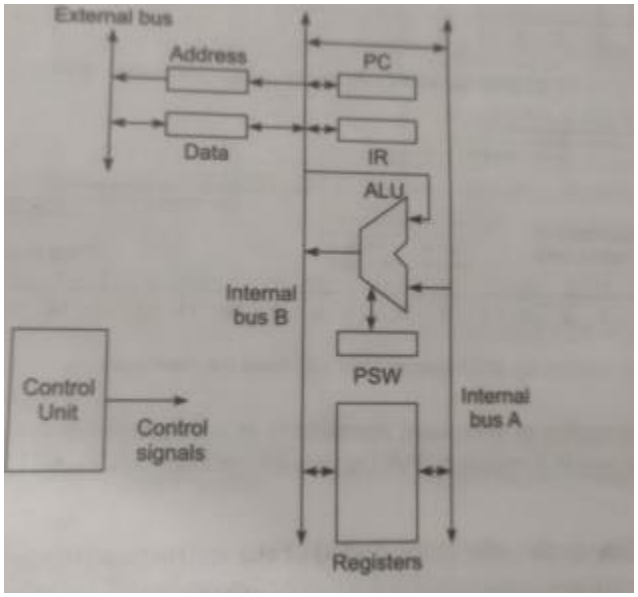


SIMPLE OPERATION LATENCY:

Simple operations make up the vast majority of instruction executed by the machine such as integer adds loads, stores, branches, moves, etc...

RESOURCE CONFLICTS:

- This refers to the instruction where two or more instructions demand use of the same functional unit at the same time.
- A base scalar processor is defined as a machine with one instruction issued per cycle one cycle latency for a simple operation and one cycle latency between instruction issues.
- The control unit generation control signals required for the fetch diode, ALU operation memory access and write result phases of instructions executions the control unit itself may employ hard writing logic or as was more common in order CISC style processors – micro coded logic.



INSTRUCTION SET ARCHITECTURE:

The instruction set of a computer specific the primitive commands or machine instruction that a programmer can use in an instruction set is attributed to the instruction formats date format opcode specification and flow control mechanisms used.

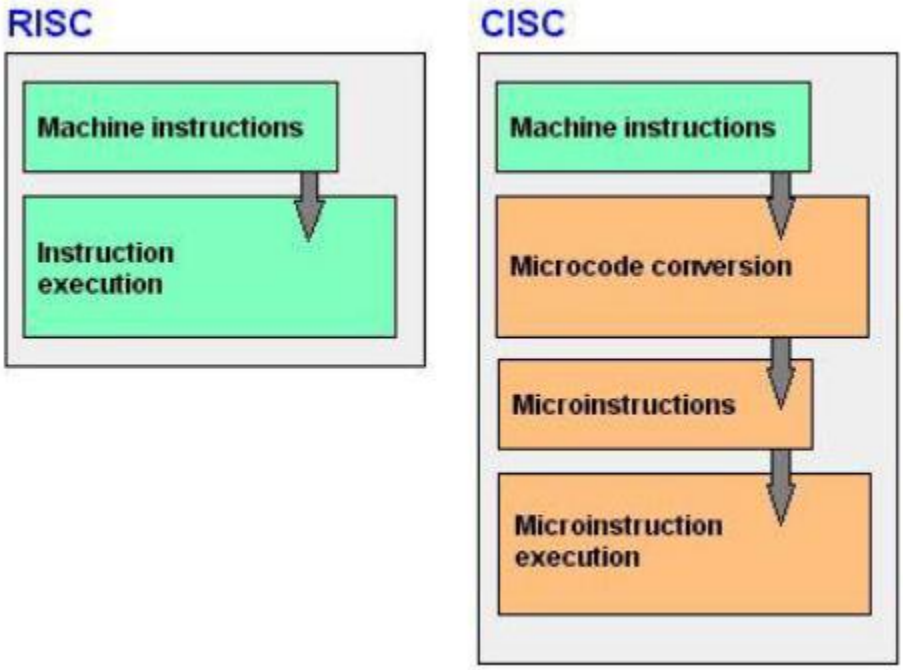
Complex instruction sets:

In the early days of computer history most computer feminism started with an instruction set which rather Simple the main reason for being simple than was the high cost of hardware.

REDUCED INSTRUCTION SETS:

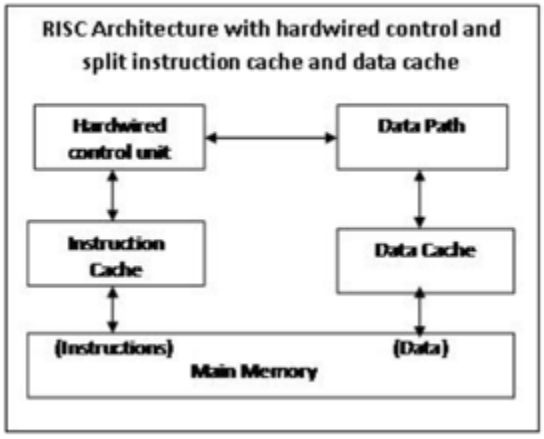
- After two decodes of using CISC processors. Computer designers began to revaluate the performance relationship between instruction set Architecture and available hardware/software technology.
- A RISC instructions set typically contains less than 100, instructions with a fixed instruction format (32 bits).
- The resulting benefits include a higher clock rate and a lower CPI, which lead to higher processor performance.

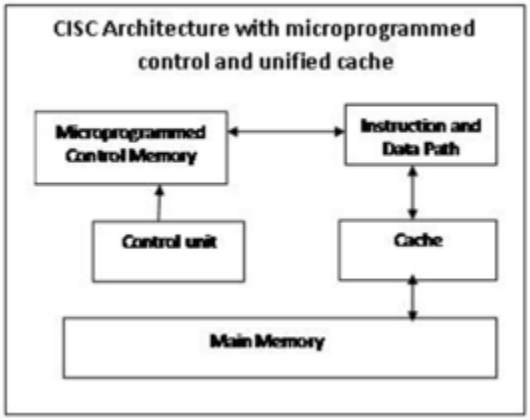
DIFFERENCE BETWEEN RISC AND CISC:



ARCHITECTURAL DISTINCTIONS:

- Hardware features built into CISC and RISC processors are compared below, some of the distinctions have since disappeared, however because processors are now designed with features from both types.
- The use of micro programmed control was found in traditional CISC, and hard wired control in most RISC. Thus control memory (ROM) was needed in earlier CISC processors, which showed down the instruction execution.





The large number of instruction used in a CISC and RISC processor are comparable. The result of using variable format instruction. Integer, floating, point and vector data and of using over a dozen different model. On the other hand most RISC processors use 32-bit instructions which are predominately register based.

Architectural Characteristic	*Complex Instruction Set Computer (CISC)*	*Reduced Instruction Set Computer (RISC)*
Instruction-set size and instruction formats	Large set of instructions with variable formats (16–64 bits per instruction).	Small set of instructions with fixed (32-bit) format and most register-based instructions.
Addressing modes	12–24.	Limited to 3–5.
General-purpose registers and cache design	8–24 GPRs, originally with a unified cache for instructions and data, recent designs also use split caches.	Large numbers (32–192) of GPRs with mostly split data cache and instruction cache.
CPI	CPI between 2 and 15.	One cycle for almost all instructions and an average CPI < 1.5.
CPU Control	Earlier microcoded using control memory (ROM), but modern CISC also uses hardwired control.	Hardwired without control memory.

CISC SCALAR PROCESSORS:

A scalar processor executes with data. The simplest scalar processor executes integer instruction using fixed point operation. More capable scalar processors execute both integer and floating point operations. A modern scalar processor may possess both an integer until and a floating penitent, or even multiple such units.

REPRESENTATION CISC PROCESSORS:

The VAX 8600 processors were built on a PC board. The i486 and M68040 were single chip microprocessors. These two processors families are still in use at present.

CISC MICROPROCESSORS FAMILIES:

Motorola produced its 8-bit microprocessor the MC68030 in 1974, then came the MC68030 and MC68040 in the Motorola MC680x0 family national semi conductions 32-bit microprocessors NS32532 was introduced in 1988.

RISC SCALAR PROCESSORS:

The RISC design gains its power by pushing some of the less frequently used operations into software. The reliance on a good compiler is much more demanding in RISC processors than in CISC processors. Instruction level parallelism is exploited by pipelining in both processors architecture.

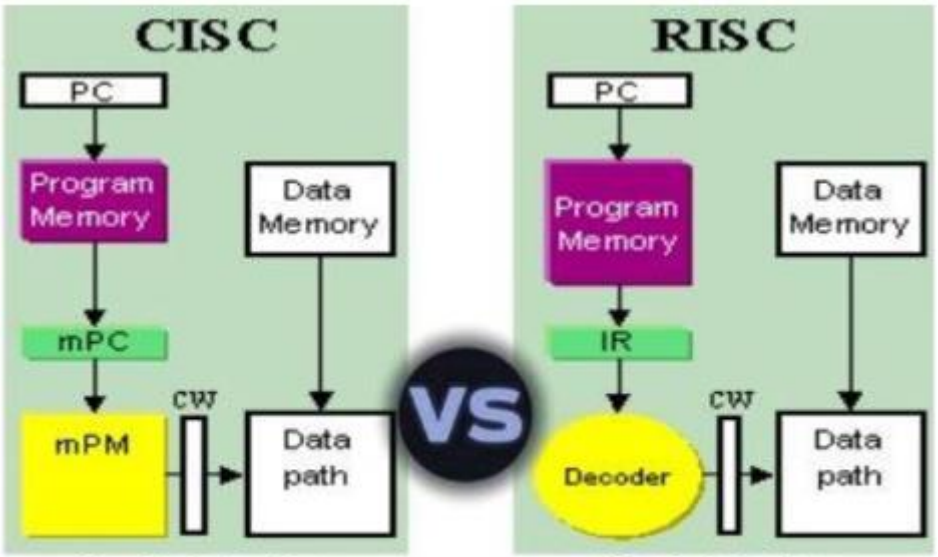
REPRESENTATIVE RISC PROCESSOR:

Four representatives RISC based processors from the year 1990. the sun SPARC Intel i860, Motorola M88100, and AMD 29000 are summarized. All of these processor use 32-bit instruction.

THE RISC IMPACTS:

The debate between RISC and CISC designers lasted for more than a decade. Based on one reported experiment, converting form a CISC program to an equivalent RISC program increases the code length by only 40%.

CISC AND RISC SCALAR PROCESSORS:



1. RISC stands for Reduced Instruction Set Computer.	1. CISC stands for Complex Instruction Set Computer.
2. RISC processors have simple instructions taking about one clock cycle. The average clock cycle per instruction (CPI) is 1.5	2. CSIC processor has complex instructions that take up multiple clocks for execution. The average clock cycle per instruction (CPI) is in the range of 2 and 15.
3. Performance is optimized with more focus on software	3. Performance is optimized with more focus on hardware.
4. It has no memory unit and uses a separate hardware to implement instructions..	4. It has a memory unit to implement complex instructions.
5. It has a hard-wired unit of programming.	5. It has a microprogramming unit.
6. The instruction set is reduced i.e. it has only a few instructions in the instruction set. Many of these instructions are very primitive.	6. The instruction set has a variety of different instructions that can be used for complex operations.
7. The instruction set has a variety of different instructions that can be used for complex operations.	7. CISC has many different addressing modes and can thus be used to represent higher-level programming language statements more efficiently.
8. Complex addressing modes are synthesized using the software.	8. CISC already supports complex addressing modes
9. Multiple register sets are present	9. Only has a single register set
10. RISC processors are highly pipelined	10. They are normally not pipelined or less pipelined
11. The complexity of RISC lies with the compiler that executes the program	11. The complexity lies in the microprogram
12. Execution time is very less	12. Execution time is very high

SUPERSCALAR AND VECTOR PROCESSORS:

A CISC or a RISC scalar processor can be improved with a super scalar or vector architected scalar processor those executing one instruction per cycle

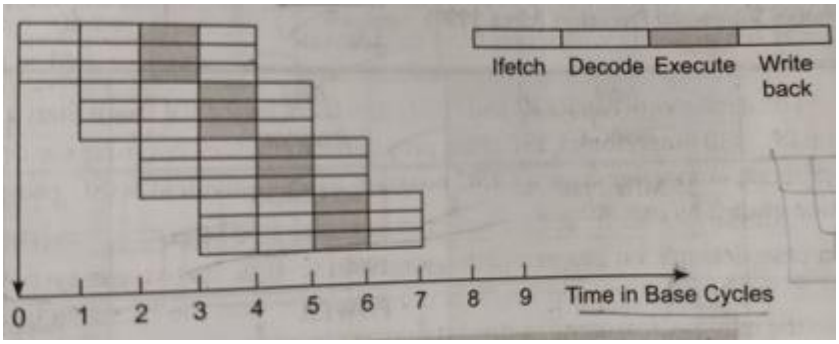
only instruction is issued per cycle. And only one completion of instruction is expected from pipeline per cycle.

SUPERSCALAR PROCESSORS:

Superscalar processors are designed to explicit more instruction level parallelism in user programs. Only independent instruction can be executed in parallel without causing a wait state.

PIPELINING IN SUPERSCALAR PROCESSORS:

The fundamental structure of a three issues pipeline, superscalar processors were originally developed as an alternative to vector processors, with a view to exploit higher degree of instruction level parallelism.



Superscalar processors of degree M can issues m instruction per cycle. In this scene, the base scalar processor, implemented either in RISC or CISC, has m=1. In order to fully utilize a superscalar processor of degree m, m instruction must be executable in parallel.

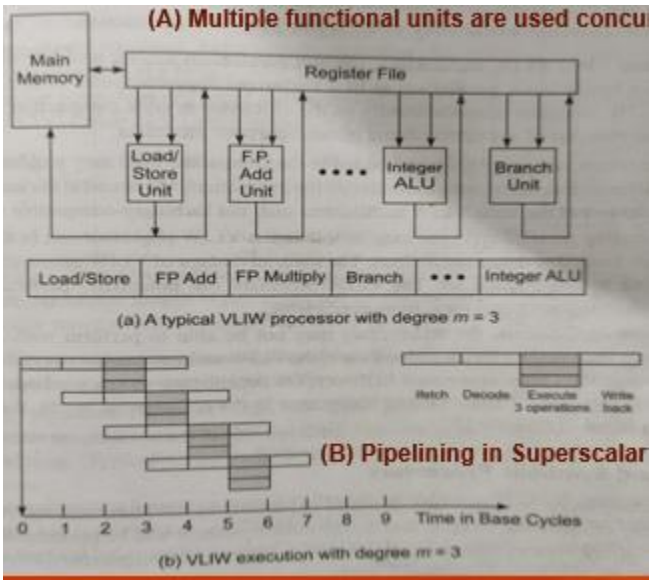
REPRESSENTATIVE SUPERSCALARE PROCESSOR:

- A number of commercially available processes have been implanted with the superscalar architecture. Notable arly ones include the IBM Rs16000, DEC Alpha 21064, and Intel i960 CA processors as summarized.

- Due to the reduced CPI and higher clock rates, used generally superscalar processors outperform scalar processors.

THE VLIW ARICHITECTURE:

- The VLIW architecture is generalized from low well established concepts. Horizontal micro coding and superscalar processing. A typical VLIW (Very Long Instruction Word). Machine has instruction words hundreds of bits in length.
- different fields of the long instruction word carry the op codes to be dispatched to different functional units this code compaction must be done by a compiler which can predict branch out comes using elaborate heuristics heuristic or run time statistics.



PIPELINING IN VLIN PROCESSORS:

- The execution of instruction by an idea VLIN processors in multiple operation VLIN machines behave much like superscalar machines with three difference.

- First the eliciting of VLIN instruction is easier than that of superscalar instructions.
- Second the code density of the superscalar machine is better when the available instruction level parallelism is less than that exploitable by the VLIN machine this is because the fixed VLIN format includes bits for non executable operations while the superscalar processors issues only executable instructions.
- Third a superscalar machine can be object mode compatible with a large family of non parallel machine.

VLIN OPPORTUNITIES:

- In a VLIN architecture random parallelism among scalar operations I exploited instead of regular or symhronous parallelism as in a vectorized super computer or in an SIMD computer
- In general purpose applications the architectural may not be able to perform well although the idea seems sounds in theory the dependence on trace scheduling compiling and code compaction has prevented it from gaining acceptance in the commercial world.

VECTOR AND SYMBOLIC PROCESSORS:

A vector processors can assume either a register to register vector processors like Cray super computer denote a vector register of length n as V_1a scalar register as s1 and memory array of length n as m (1:n) typical register based vector operations are listed below where a vector operator is denoted by a small circle “o”

V1	o	v2	- v3 (binary vector)
S1	o	v1	- v2 (scaling)
V1	o	v2	- s1 (binary reduction)

M (1: n) - v1 (vector load)

v1 - m (1: n) (vector store)

O v1 - v2 (unary vector)

O v1 - v1 (unary reduction)

Memory based vector operations are found in memory to memory vector processors such as those in the early super computer CDC cyber 205 listed below are a few examples.

m1 (1: n) o m2 (1: n) - m (1: n)

S1 o m1 (1: n) - m2 (1: n)

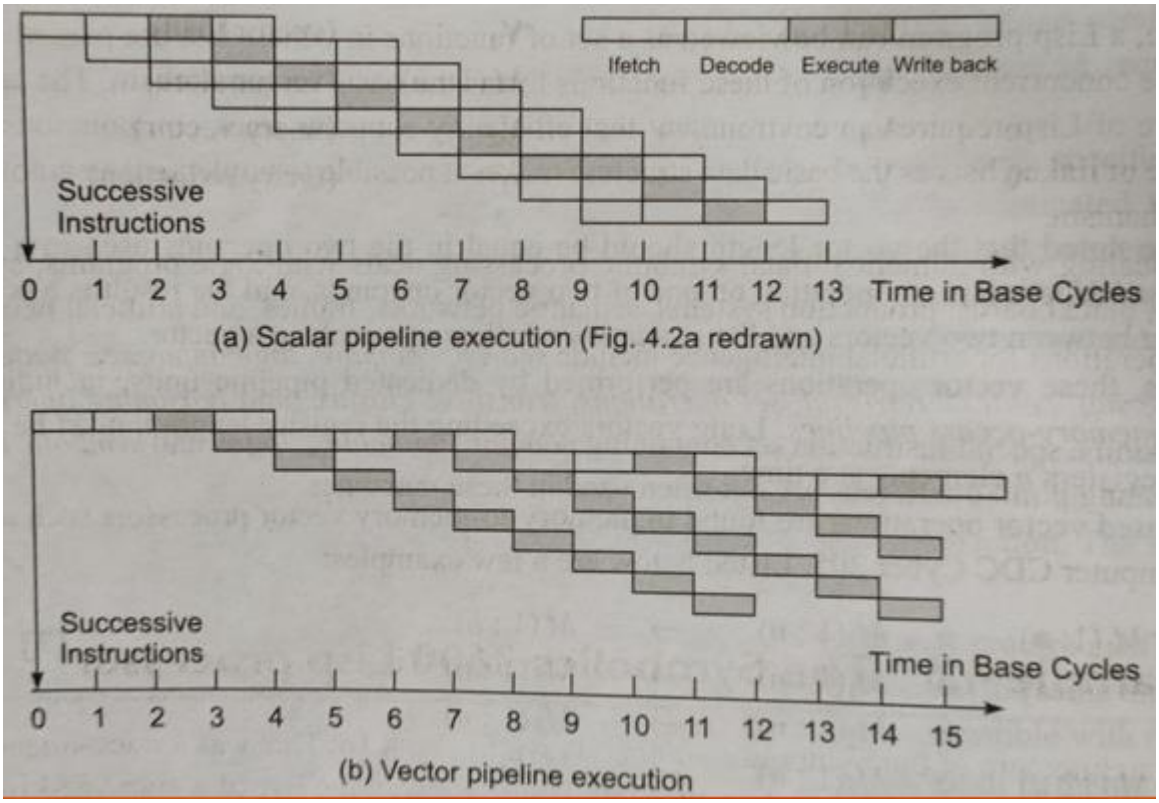
O m1 (1: n) - m2 (1: n)

m1 (1: n) o m2 (1: n) - m2 (1: n)

Where m1 (1: n) and m2 (1: n) are two vector of length n and m (k) denotes a scalar quantity stored in memory location k.

VECTOR PIPELINE:

Vector processors take advantage of unrolled loop level parallelism the vector pipelines can be attached to any scalar or super scalar processors.



SYMBOLIC PROCESSORS:

Symbolic processing has been applied in many areas including theorem proving pattern recognition expert systems knowledge engineering text retrial cognitive science and machine intelligence list processors or symbolic manipulations these characteristics.

LINEAR PIPELINE PROCESSORS:

Cascade of processing affixed function over a stream of data flowing one end to other in modern computer linear pipelines applied for

- Instruction Execution
- Arithmetic Execution
- Memory Access Operations

Depending on control of data flow in linear pipeline has two categories

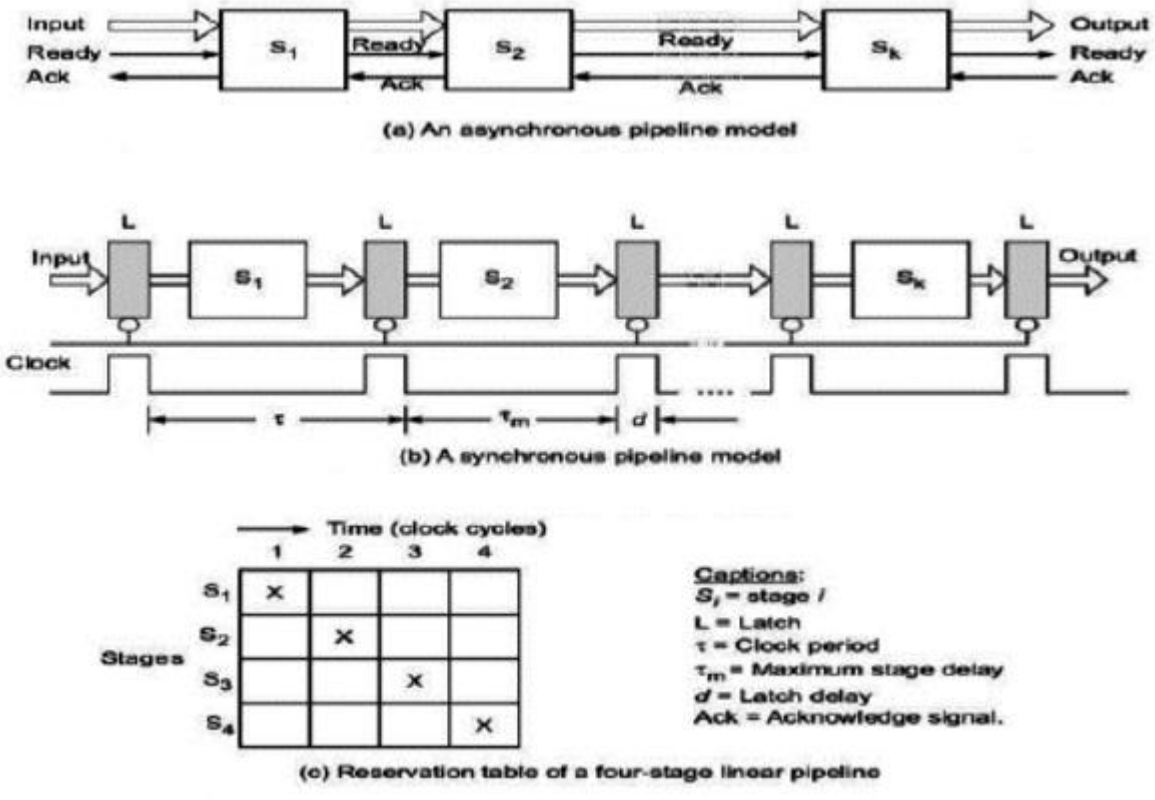
- Asynchronous Pipeline Model
- Synchronous Pipeline Model

ASYNCHRONOUS AND SYNCHRONOUS MODEL

A linear pipeline processors is constructed with k processing stages external inputs are fed into the pipeline at the first stage s1 the processed results are passed from s1 to stage si+1, for all i=1,2,……k-1 the result emerges from the pipeline at the last stage sk.

ASYNCHRONOUS MODEL:

Asynchronous pipeline are useful in designing communication channels in message passing multicomputer where pipelined worm whole routing through put rate different amounts of delay may be experienced in different stages.



SPEED UP EFFICIENCY AND THROUGH PUT:

Linear pipeline of k stages can process n tasks in k+ (n-1) clock cycles where k cycle is needed to complete. The execution of the very first task and the remaining n-1 tasks require n-1 cycles

$$T_k = [k+ (n-1)] J$$

Speed up factor:

The speedup factor of a k stage pipeline over an equivalent non pipeline is defined as

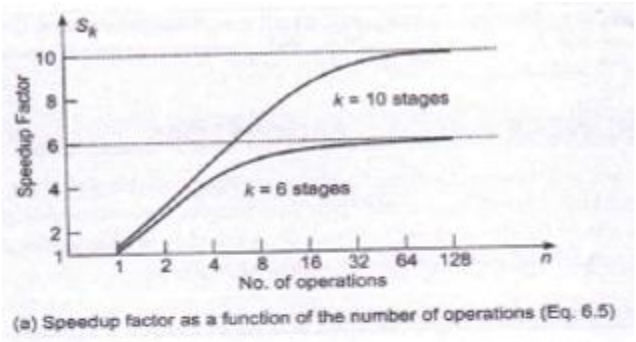
$$sk = \frac{T1}{tk} = \frac{nk\tau}{k\tau + (n-1)\tau} = \frac{nk}{k + (n-1)}$$

OPTIMAL NUMBER OF STAGES :

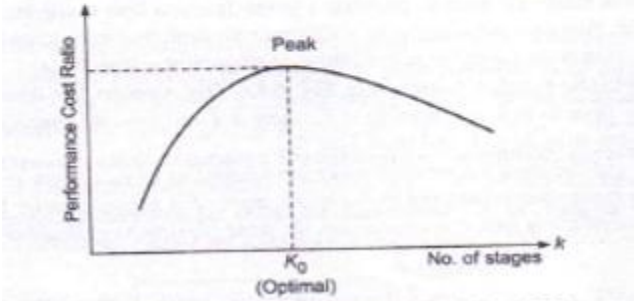
Let be the total time required for a no pipelined sequential program of a given function to execute the same program has a maximum through put of

$$f = \frac{1}{p} = \frac{1}{(\frac{t}{k} + d)}$$

$$PCR = \frac{f}{c + kh} = \frac{1}{\left(\frac{t}{k} + d\right)(c + kh)}$$



(a) Speedup factor as a function of the number of operations (Eq. 6.5)



(b) Optimal number of pipeline stages (Eqs. 6.6 and 6.7)

SYNCHRONOUS MODEL:

Synchronous pipeline are illustrated clocked latches are used to interface between stages the latches are mode with master slave flip flops which can isolate inputs from outputs upon the arrival to the next stage simultaneously.

CLOCKING AND TIMING CONTROL:

The clock cycle τ bet the time delay of the circuitry in stage si and d the time delay of the latch.

CLOCK CYCLE AND THROUGHPUT:

Denote the maximum stage delay as τmp and we can write τ as

$$\tau = \max\{\tau \mathrm{i}\}_1^k + d = \tau_m + \mathrm{d}$$

At the using edge of the clock pulse the data is latched to the master flip flops of each latch register in general $\tau_m \gg d$ by one to two orders of magnitude the pipeline frequency is defined as the inverse of the clock period.

$$f = \frac{1}{\tau}$$

CLOCK SKEWING:

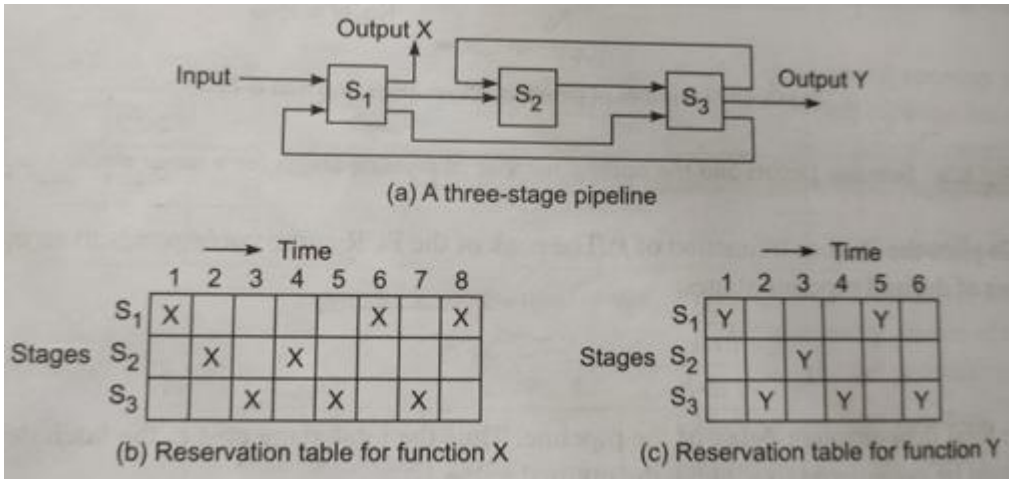
I delay we except the clock pulse to arrive at all stages (latches) at the same time let image be the time delay to the longest logic path within a tag and t_{min} that of the shortest logic path within a stage.

NONLINEAR PIPELINE PROCESSORS:

A dynamic pipeline can be reconfigured to perform variable function at different times.

RESERVATION AND LATENCY ANALYSIS:

These frees forward and feedback connection make the pipeline a nontrivial task with their connections the output of the pipeline is not necessary from the last stage.

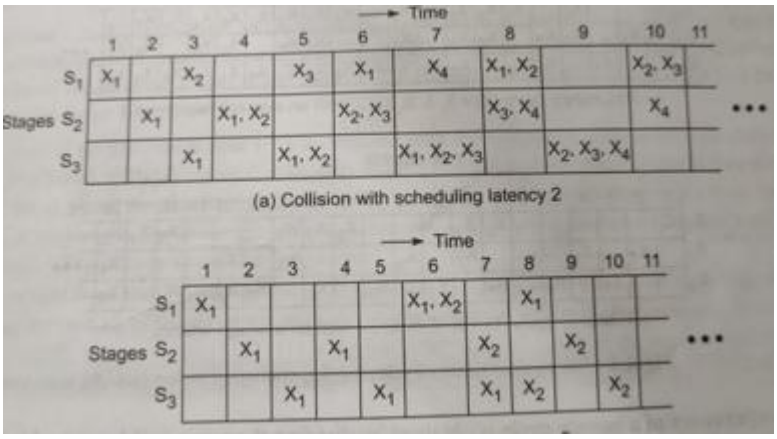


1. **Reservation tables:**

The reservation table for a static linear pipeline is trivial in the sense that data flow follows a linear streamline.

2. latency analysis:

- Collision
- Forbidden Latency
- Latency Sequence
- Latency Cycle
- Average Latency
- Constant Cycle



A collision or clash is a situation that occurs when two distinct pieces of data have the same has value checksum finger print or cryptograph digest.

LATENCY

It's the amount of delay (or times) it takes to send information from one point to the next latency in milliseconds or Ms.

COLLISION FREES SCHEDULING:

Shortest average latency between initiations without causing collisions.

- collision vector

- state diagrams
- greedy cycles

STATE DIAGRAM:

A state diagram is a type of diagram used in computer science and related fields to describe the behaviour of system.

GREEDY CYCLE:

A simply cycle is a greedy cycle if each latency contained in a cycle is the minimal latency (outgoing are) from a state in the cycle.

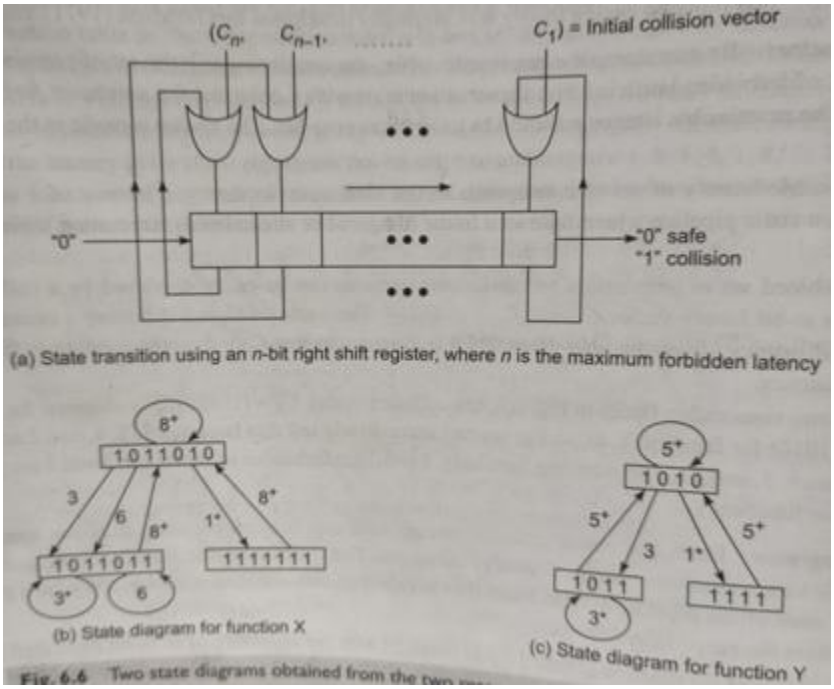
PIPELINE SCHEDULE OPTIMIZATIONS:

An optimization technique based on MAL given below. The idea is insert non compute delay stages into original pipeline.

- It Modify Reservation Table.
- Improve The State Diagram.

TECHNIQUES ARE:

- Bounds On The MAL (Minimal Average Latency)
- Delay Insertion



COLLISION VECTOR:

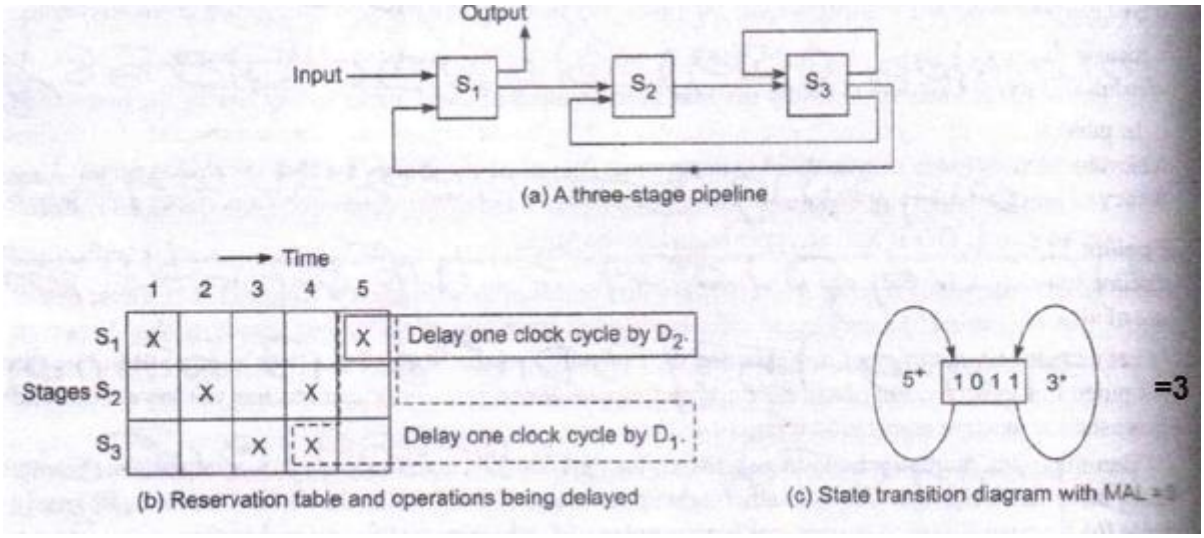
The collision vector is a method of analysing how often we can initiate a new operation into the pipeline and maintain synchronous flow without collisions.

- Pipeline Throughput (Initiation Rate)
- Pipeline Efficiency (State Unitization)
- Mal(Minimal Average Latency)

Lower bound by the maximum number of check marks in any row if reservation table

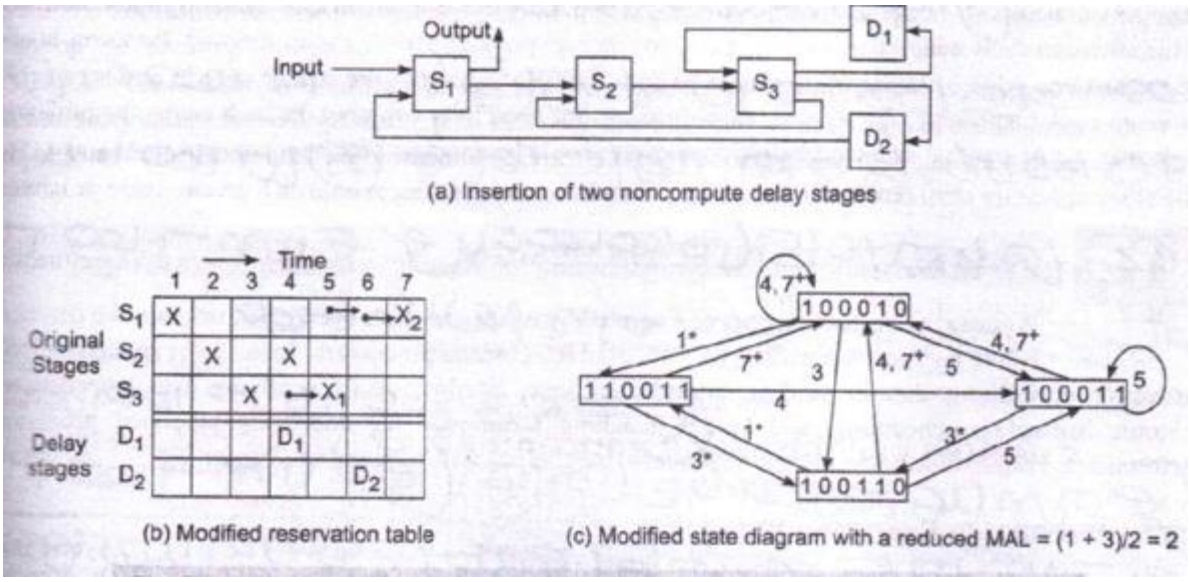
- Average Latency To Any Greedy Cycle In State Diagram

- Upper Bounded By The Number Of Is In Initial Collision Vector Plus1.



DELAY INSERTION:

- It's Used To Modify The Reservation Table .
- This Leads To a Modified State Diagram Which May Produce Greedy Cycle.
- Meeting the Lower Bound on the Mal.



UNIT- 4
MULTIPROCESSORS AND MULTICOMPUTER

- Multiprocessors System Interconnects
- Message Passing Mechanisms
- SIMD Computer Organization
- The Connection Machine CM – 5
- Fine Grain Multicomputer

MULTIPROCESSORS SYSTEM INTERCONNECTS:

Parallel processing demands the use of efficient system interconnects for fast communication among input/output and peripheral devices.

- Hierarchical buses
- Crossbar switches
- Multistage networks

NETWORK CHARACTERISTICS:

- Timing control
- Data transfer
- Control strategy

Each of the above type of networks can be designed with many choices. The choices are based on the topology timing protocol, switching method and control strategy.

TIMING CONTROL:

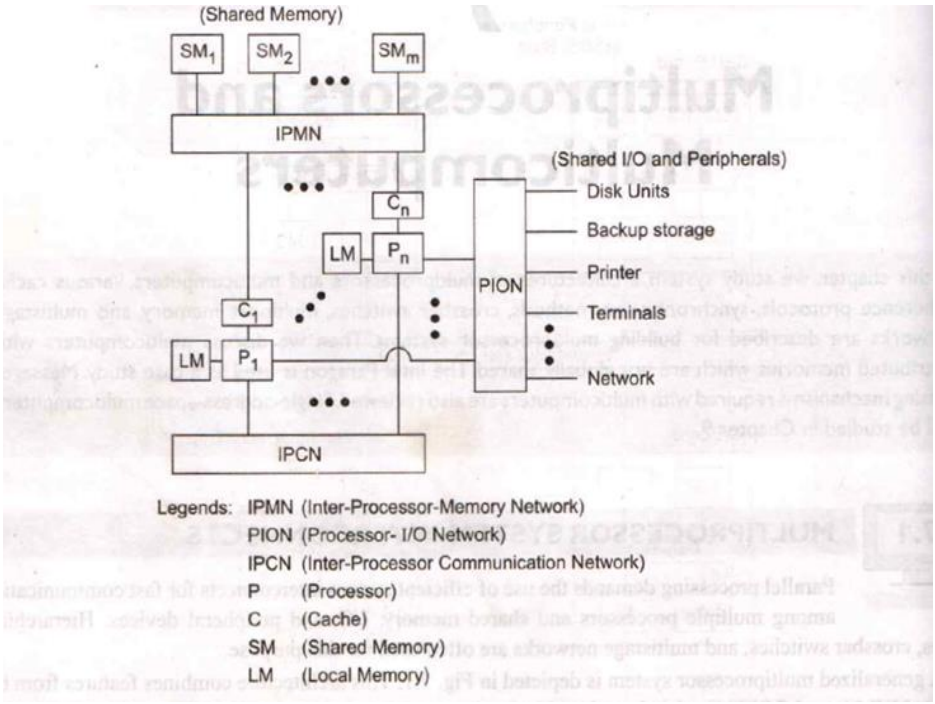
Timing, switching and control are three major operational characteristics are under program network.

DATA TRANSFER:

A network can transfer data using either circuit switching or packet switching.

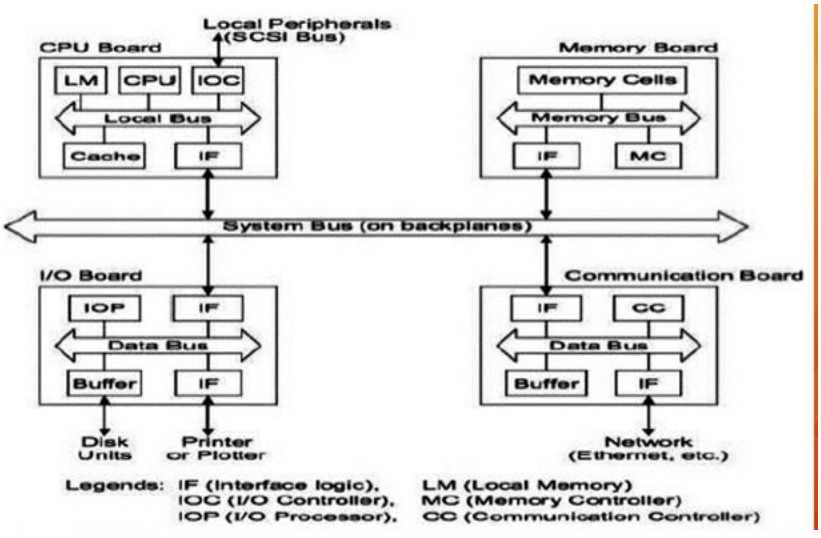
CONTROL STRATEGY:

Network control strategy is classified as centralized control, a global controller receives request from all devices, and request are handled by local devices independently.



HIERARCHICAL BUS SYSTEM:

Connecting various system and sub system components in a computer. Each bus is formed with a number of signal, control and power lines.



LOCAL BUS:

- Buses implemented within processors chip or on printed circuit boards are called local buses.
- A memory board uses a memory bus to connect the memory with the interface logic.

- An I/O or network interface chip or board uses a data bus. Each of these local buses consists of signal and utility lines.

BACKPLANE BUS:

A backplane bus standards have been developed over time such as the VME bus, multibus II and future bus as introduced to point – to – point switched interconnects have emerged as more efficient alternatives.

I/O BUS:

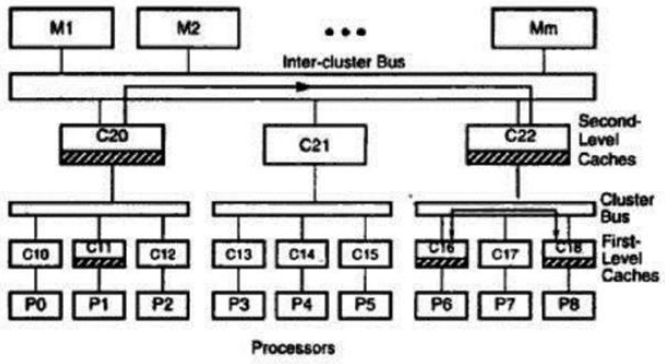
- Input/output devices are connected to a computer system through I/O bus. Such as SCSI (Small computer system interface) bus.
- This made coaxial cables.

EX:

Printer, disc and other devices.

HIERARCHICAL BUSES AND CACHES:

- Wilson (1987) proposed hierarchical cache/bus architecture is multi level tree structure in which the leaf nodes are processors and their private caches.
- An intercluster bus is used to provide communication among the cluster bus and the intercluster bus.
- Each second level cache must have a capacity that is at least an order of magnitude larger than the sum of the capacities of all first level caches beneath it.



CROSSBARS SWITCH MULTIPORT MEMEORY:

Switches network provide dynamic inter connects between the input and outputs. The multistage networks can be extended to large systems it the increased latency problem can be suitably addressed.

NETWORK STAGES:

- Single stage network
- Multistage network

SINGLE STAGE NETWORK (RECIRCULATING NETWORK)

Recirculate many times to reaching their destinations. Cheaper to build.

MULTISTAGE NETWORK (OMEGA NETWORK, FLIP NETWORK BASELINE NETWORK)

More than one stage of switch boxes. Network should be connected from any input to any output.

BLOCKING VERSUS NON – BLOCKING NETWORKS:

- Blocking
- Non – blocking

BLOCKING:

A multistage network is called blocking if the simultaneous interconnects of some multiple input – output pairs may result in conflicts in the use of switches or communication tasks.

NON – BLOCKING:

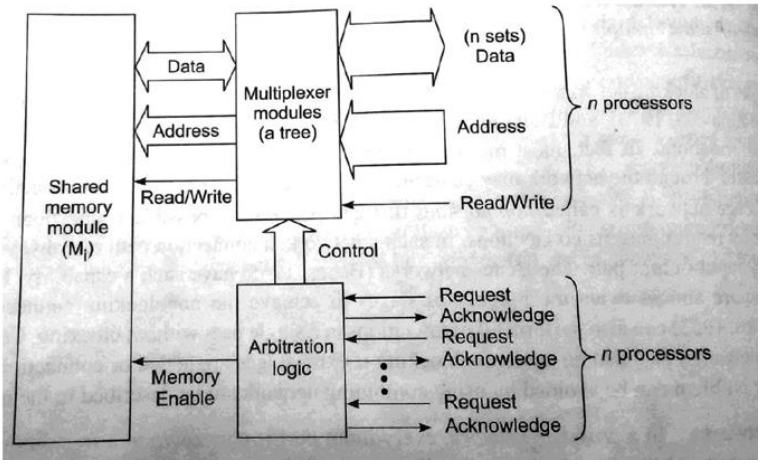
ALL possible connections inputs and outputs by rearranging its connections, connections path can always be established between any input and output pair.

CROSSBAR NETWORKS:

- In a crossbar networks, every input port is connected to the tree port through a cross point switch without blocking. A crossbar network is a single at the cross points.
- Once the data is read from the memory, its value is returned to the requesting processor along the same cross point switch. In general such a crossbar network requires the use of $n \times m$ crosspoint switches.

CROSSPOINT SWITCH DESIGN:

Out of n cross point switches in each column n×m crossbar mesh, only one can be connected at a time. To resolve the contention for each memory module, each cross point switch must be designed with extra hardware.

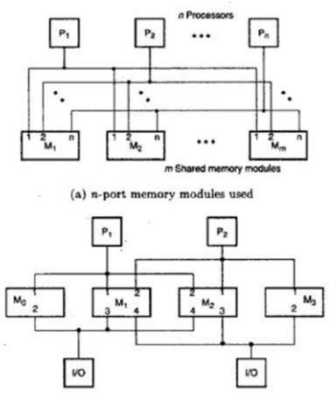


CROSSBAR LIMITATIONS:

A single processor can send many requests to multiple memory modules for an n×n crossbar network, at most n memory words can be delivered to at most n processors in each cycle.

MULTIPORT MEMORY:

- Because building a crossbar network into a large system is cost prohibitive, some mainframe multiprocessors used a multiple memory used a multiple memory organizations.
- Some of the processors are CPUs, some are I/O processors, and some are connected to dedicated processors.



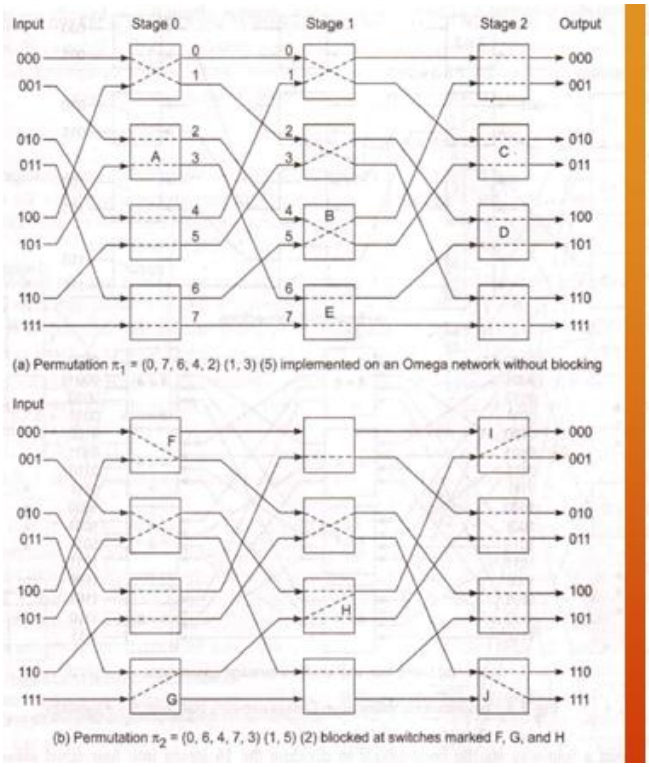
(a) n-port memory modules used

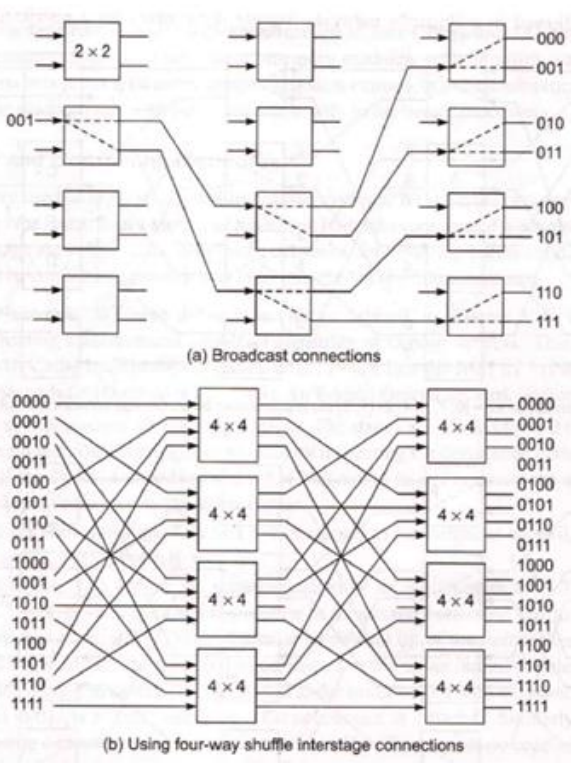
MULTISTAGE AND COMINING NETWORKS:

Multistage networks are used to build in large multiprocessors system.

ROUTING IN OMEGA NETWORKS:

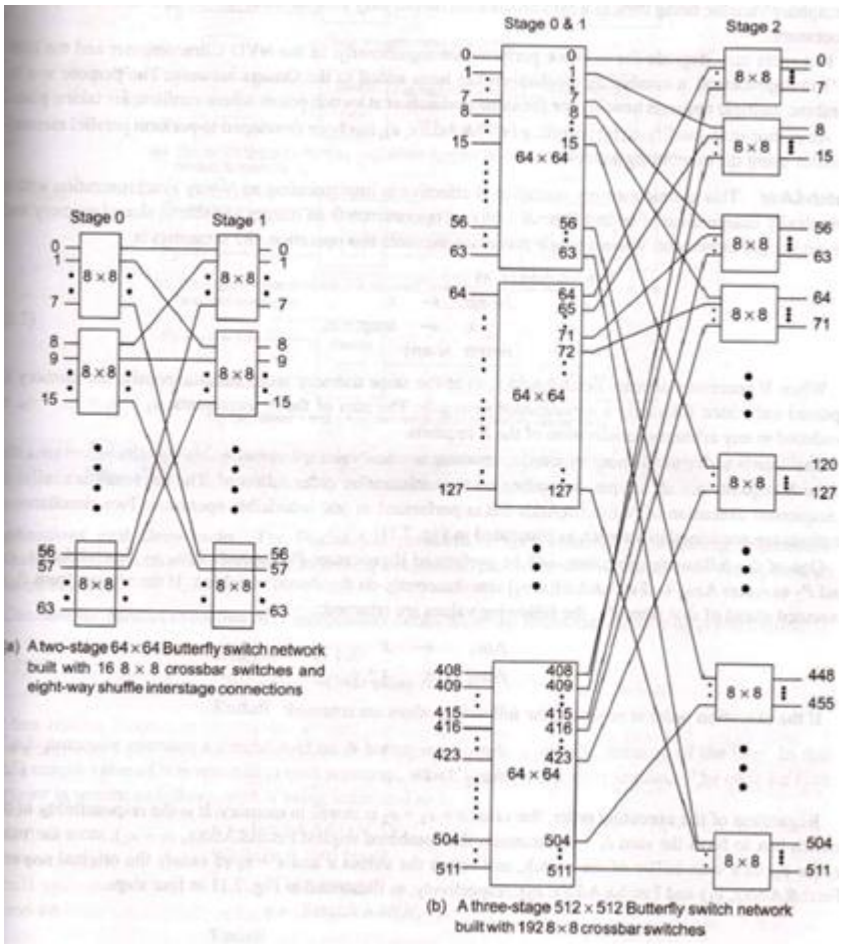
- This class of network was built into the Illinois cedar multiprocessor into the IBM RP3, and into the NYU ultra computer. An 8 input omega network.
- In general an n – input omega network has $\log_2 n$ stage. The stages are labeled from o to $\log_2 n - 1$ from the input end to the output end.





ROUTING IN BUTTERFLY NETWORK:

- This class of networks is constructed with crossbar switches as blocks the eight way suffle function is used to establish the interested connections between stage o and stage 1.
- A three stage butterfly network is constructed for 512 inputs again with 8×8 crossbar switches. Each of the 64×64 boxes in identical to the two stage butterfly network.



THE HOT – SPOT PROBLEM:

- When the network traffic is nonuiform a hotspot may appear cores ponding to a certain memory module being excessively accessed by many processors at the same time.
- An atomic read modify write primitive fetch Add(x, e) has been developed to perform parallel memory updates using the combining network.

FETCH AND ADD:

This atomic memory operation is effective in implementing an n – way synchronization with a complexity this operation, the semantic is,

$$fetch\ \&\ Add\ (x, e)$$

$$\{\ temp\ \leftarrow x;$$

$$x\ \leftarrow temp + e;$$

$$return\ temp\ \}$$

When n processor attempt fetch &Add (x, e) at the same memory word simultaneously, the memory is updated only once following a serialization principle. The sum of the N increments e1+e2+……+en is produced in any arbitrary serialization of the N requests.

One of the following operations will be performed it processors P_1 executes Ans_1←fetch &Add (x, e) and P1 executes Ans_2 ←fetch & Add (x, e2) simultaneously on the shared variable

$$Ans1 \leftarrow x$$

$$Ans2 \leftarrow x + e1$$

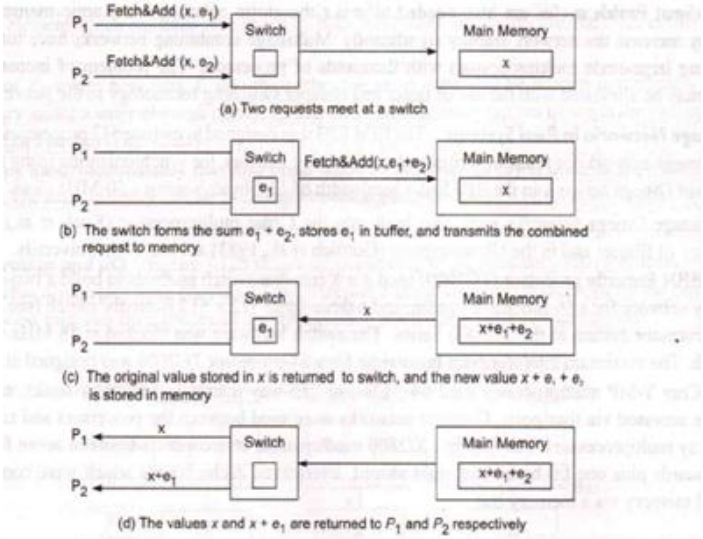
If the execution order is reserved the following values.

$$Ans\ 1 \leftarrow x + e2$$

$$Ans2 \leftarrow x$$

Regardless of the executing the value x+e1+e2 is stored in memory.

It is the responsibility of the switch box to form the sum e1+e2 transmit the combined request fetch & Add(x,e1,e2) store the value e1(or) e2 in a wait buffer of the switch and return the values x and x+e to satisfy the original request fetch & Add(x,e2) respectively four steps.



APPLICATION AND DRAWBACKS:

T he fetch & Add primitive is very effective in accessing sequentially allocated queue structure in parallel with indention code that operation different data sets,

$$Do\ all\ \ N = 1\ to\ 100$$

$$< code\ using\ N >$$

$$End\ all$$

The advantage of using a combining network to implement the fetch & Add operation is achieve at a increase in network cost.

MULTISTAGE NETWORKS IN REAL SYSTEM:

- The IBM was designed to include 512 processors using a high speed omega network for reads or writes and combining network for synchronization using fetch & Adds.
- A 128 port omega network in the RP3 had a bandwidth of 13 Gbytes/s using a 50 MHZ clock.

MESSAGE PASSING MECHANISMS:

- Message passing in a multi computer network demands special hardware and software support. We introduced the concept of virtual channels.
- Deterministic and adaptive routing algorithms (deadlock free message).
- Operation using virtual channels or virtual subnets and greedy routing algorithms.
    - One – to – one(Unicast routing operations)
    - One - to – many(Multicast routing operations)
    - One – to –all(Broadcast routing operations)

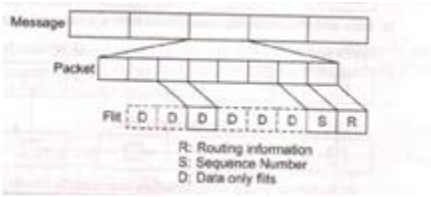
MESSAGE ROUTING SCHEMES:

- Store and forward
- Wormbole routing schemes

In two generation of multicomputer.

MESSAGE FORMATS:

- A message is the logical unit for internodes communication. It is often assembled from an asbitary number of fixed length packets, thus it may have a variable length.
- A packet is the basic unit containing the destination address for routing purpose.



PACKET:

Basic unit containing destination address packet length range 64 to 512 bits.

FLIT:

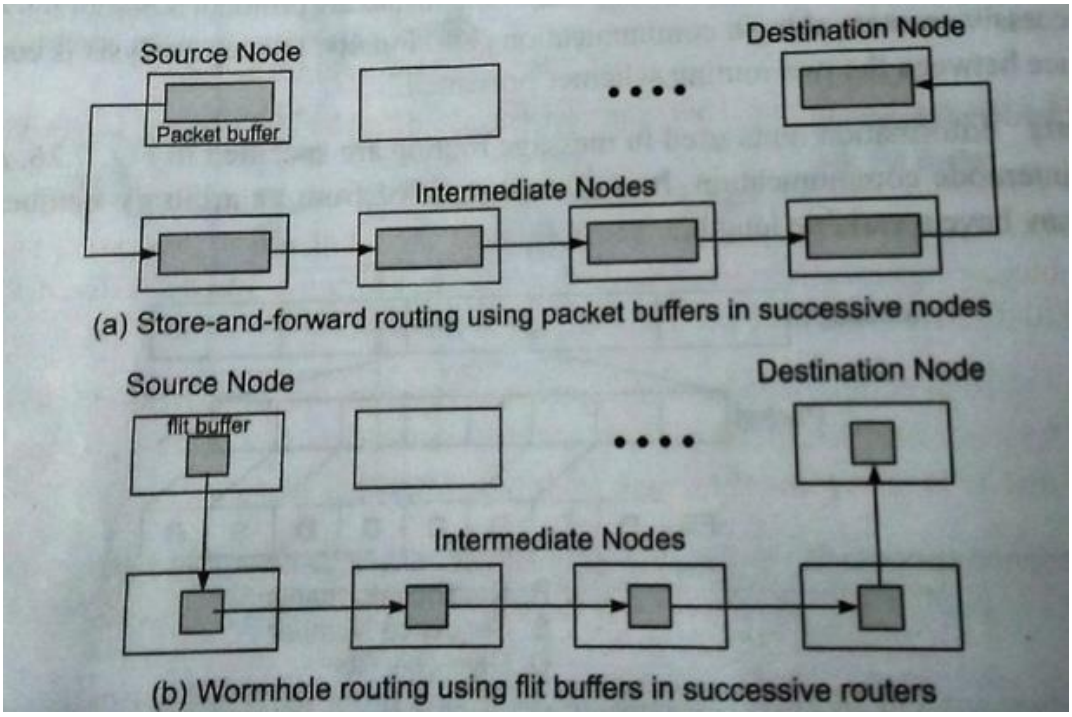
Sub divided packets routing information in header flits remaining flits containing data elements.

STARE AND FORWARD ROUTING:

- Packets are the basic unit information flow in a store and forward network.
- Source node intermediate node destination node.
- First stored in the buffer.

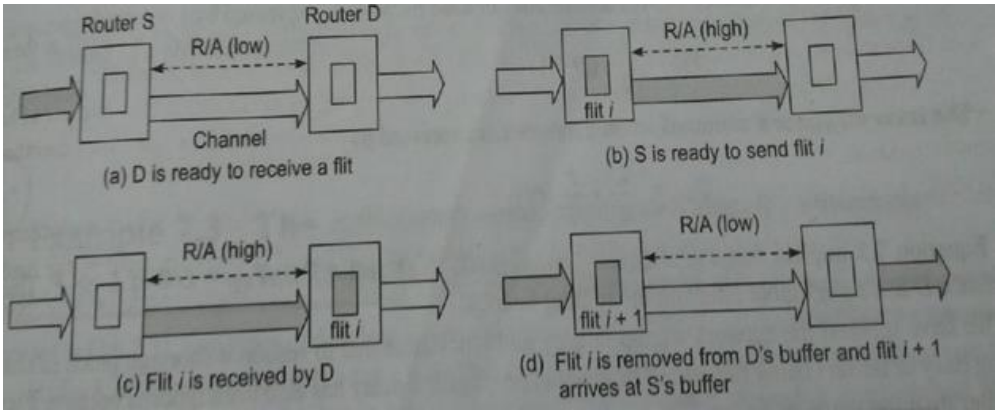
WORMBOLE ROUTING:

- Subdividing the packet into smaller flits latter generation of multicomputer implement the wormhole routing scheme.
- All the flits in the same packet were transmitted in order as inseparable companion in a pipelined fashion.



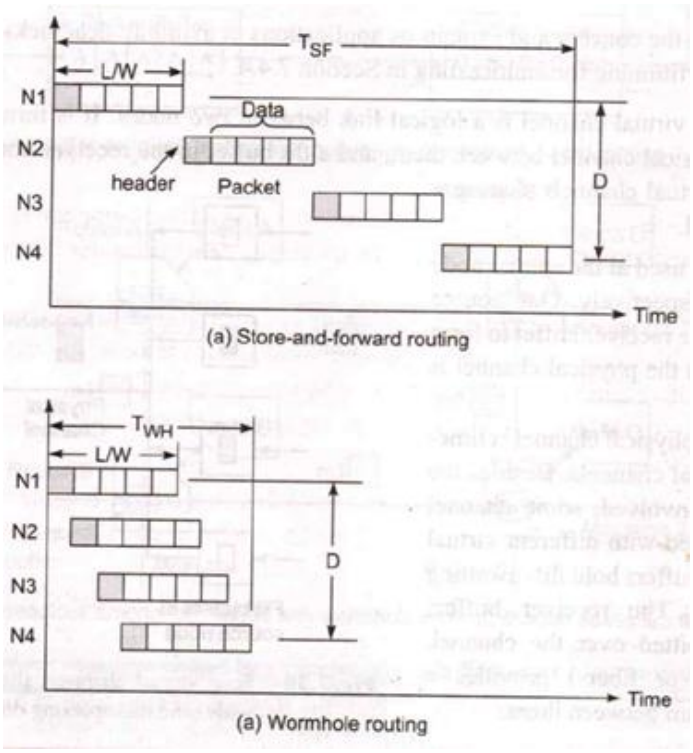
ASYNCHRONOUS PIPELINE:

The pipelining of successive flits in a packet is done asynchronously using a stand shaking protocol. Along the path an I bit ready/ request(R/A) line is used between adjacent routers.



LATENCY ANALYSIS:

A time comparison between store and forward and wormhole routed networks is given L is the packet length (in bits), W the channel bandwidth (in bits/s),D the distance (number of nodes averse minus 1) and flit length.



The communication latency TSF store and forward network is expressed by

$$TSF = L/W\ (D+1)$$

The latency TWH for a wormhole routed network is expressed by.

$$TWH = 2/W + F/W \times D$$

Equation 7.5 implies that TSF is directly proportional to D. In εq TWH = $L/_W\, i + L \gg F$.

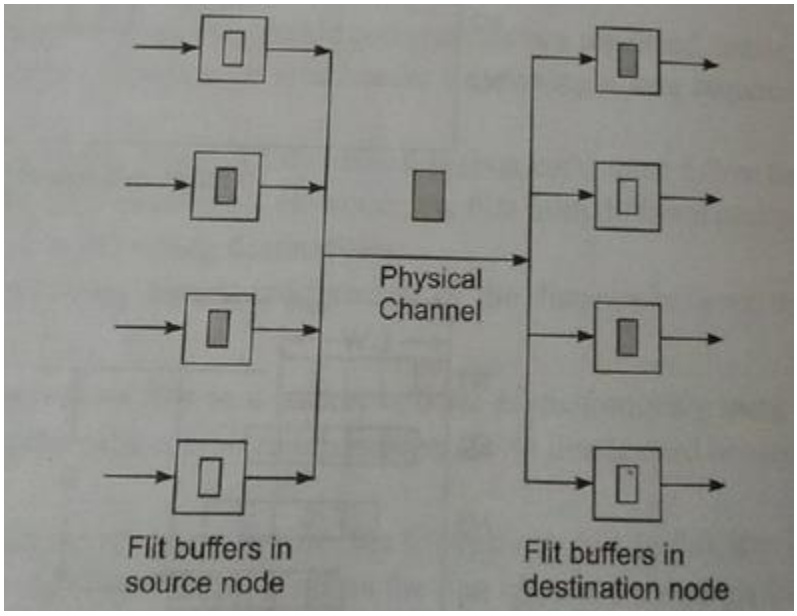
Thus the distance D has a religible effect on the routing latency.

DEADLOCK AND VIRTUAL CHANNELS:

The communication channels between nodes wormhole routed multicomputer network are actually shared by many possible source and destination pairs.

VIRTUAL CHANNELS:

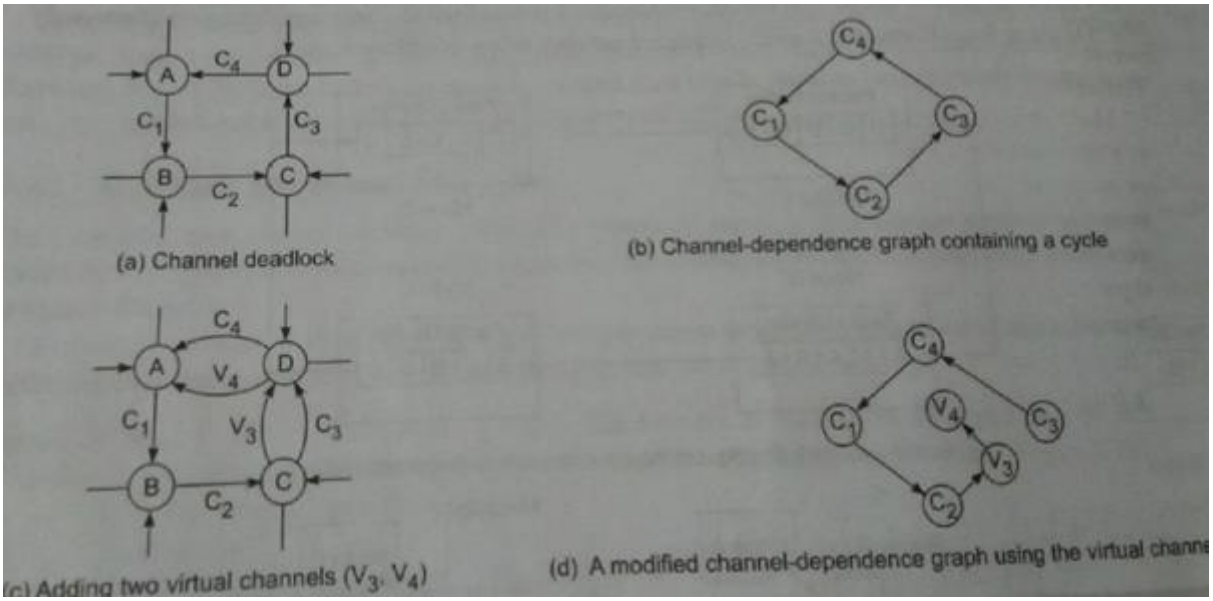
A virtual channel is a logical link between two nodes. It is formed by a flit buffer in the source node, a physical channel between them, and a flit buffer in the receiver node. The concept of four virtual channels sharing a single physical channel.



Four flit buffers are used at the source node and receiver node, respectively. In other words the physical channel is time shared by all virtual channels. The channel (wires or fibers) provides a communication medium between them.

DEADLOCK AVOIDANCE:

By adding two virtual channels v3 and v4. A Modified channel dependence graph is obtained by using the virtual channels v3 and v4, after the use channel c2 instead of reusing c3 and c4.

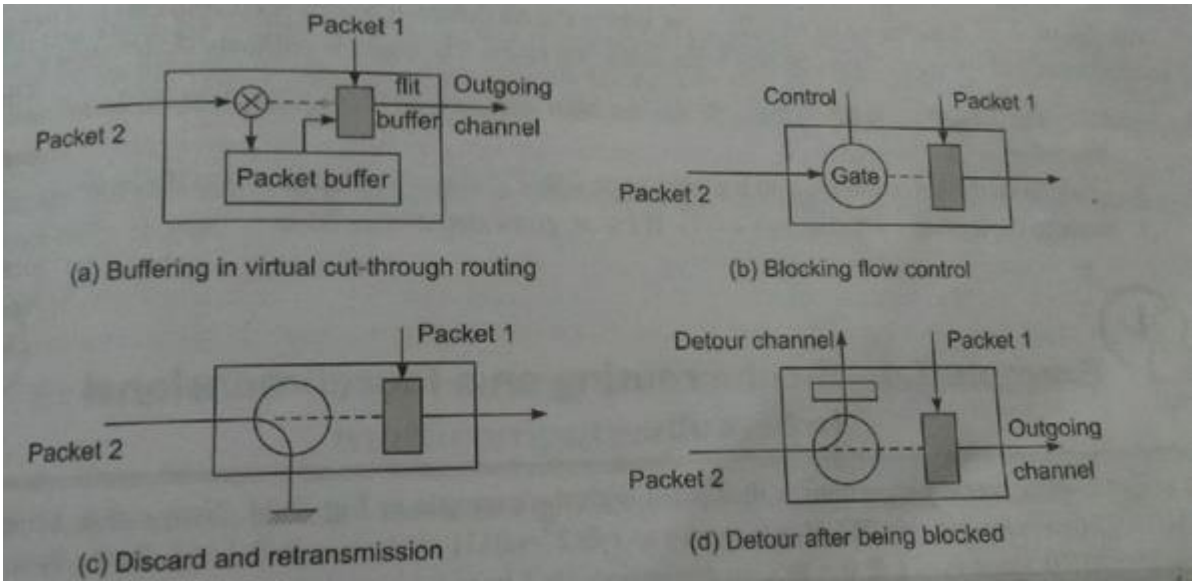


FLOW CONTROL STRALEGIES:

- In this section, we examine various strategy developed to control smooth network traffic flow without reusing congestion or deadlock situation.
- Based on these policies, we describe below deterministic and adaptive routing algorithms developed for one –to –one communication.

PACKET COLLISION RESOLUTION:

- In order to move a flit between adjacent nodes in a pipeline of channels, three elements must be present (1) the source buffer holding the flit, (2) the channel being allocated and (3), the receiver buffer accepting the flit.
- Pure wormhole routing uses a blocking policy in case of packet collision the second packet is being blocked from advancing, however it is not being abandon.
- The fourth policy is called detour the blocked packet is routed to a detour channel. The blocking policy is called detour implement, but may result in the idling resources allocated to the blocked packet.

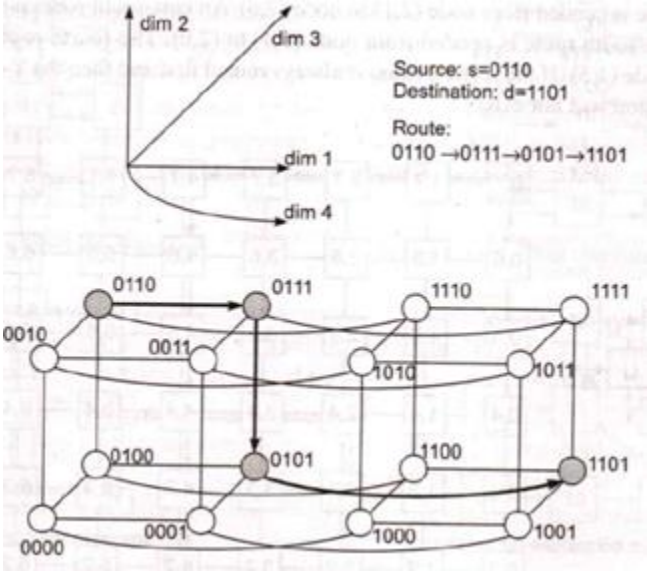


DIMENSION ORDER ROUTING:

Packet routing can be conducted deterministically or adaptively. In deterministic routing, the communication path is completely determined by the source and destination addresses.

E – CUBE ROUTING ON HYPER CUBE:

- Consider an n – cube with $n=2^n$bn-1, bn-2……b1b0. Thus the source code is s=Sn+1……s1s0 and the destination node is d=dn-1……..d1d0.
- We denote the n dimensions as I=1, 2……..n where the i^{th} dimension corresponds to the (i-1) s bit in the node address.



X – Y ROUTING ON A 2D MESH:

It is like a mesh connected networks X – Y routing format.

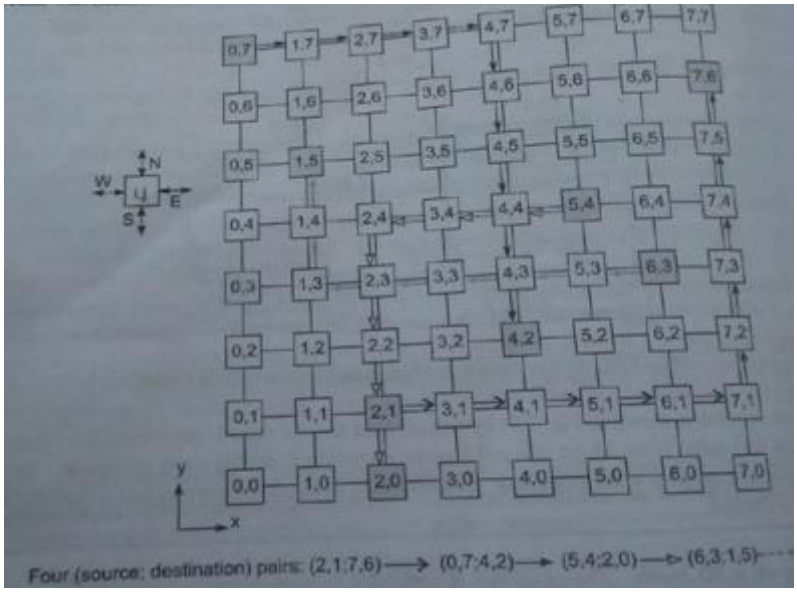
Any source node to any designation node along with the x – y axis. Four possible routing.

i) East – north

ii) East – south

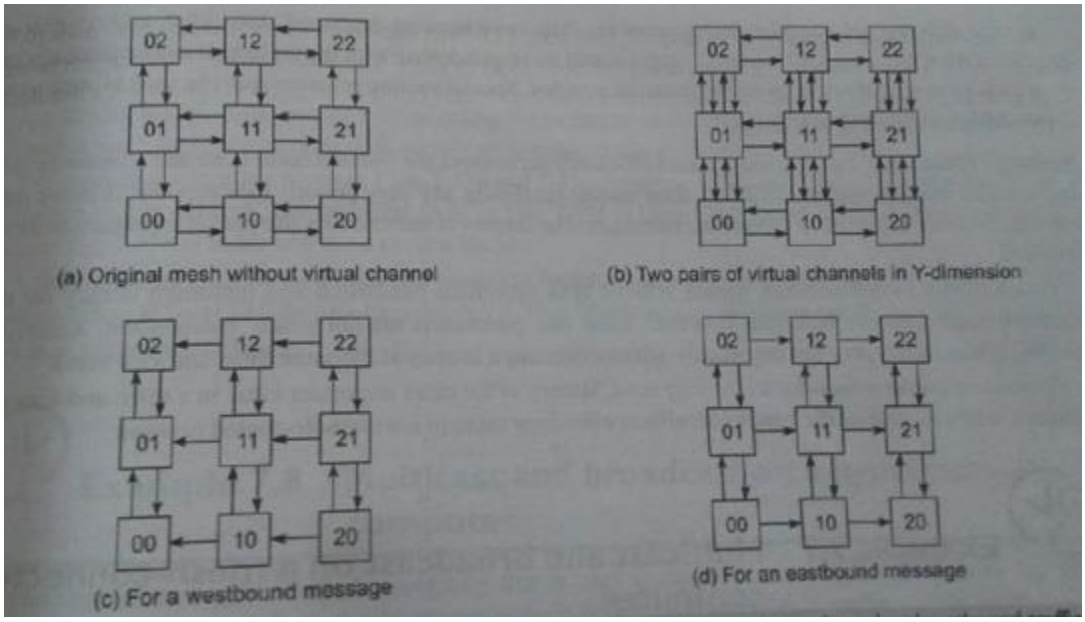
iii) West – North

iv) West – south



ADAPTIVE ROUTING:

The main purpose of using adaptive routing is to achieve efficiency and avoid deadlock. The concept of virtual channels makes adaptive routing economical and feasible to implement.



MULTICAST ROUTING ALGORITHMS:

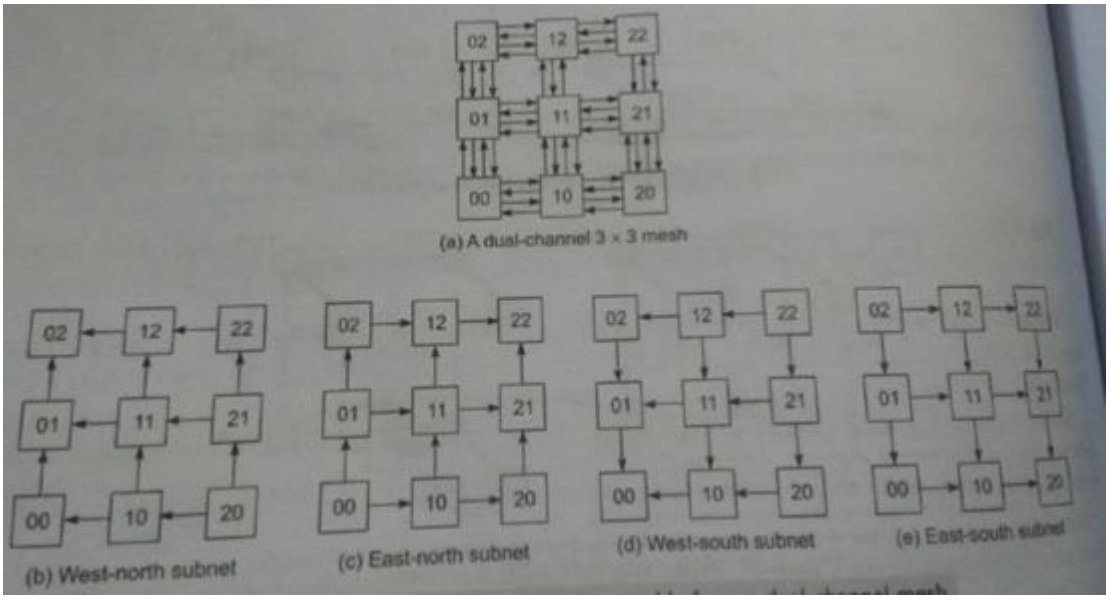
Various communication patterns are specified below. Routing efficiency is defined.

COMMUNICATION PATTERNS:

- A multicast pattern corresponds to one – to – many communications.
- A broadcast pattern corresponds to the case of one – to – all communication.

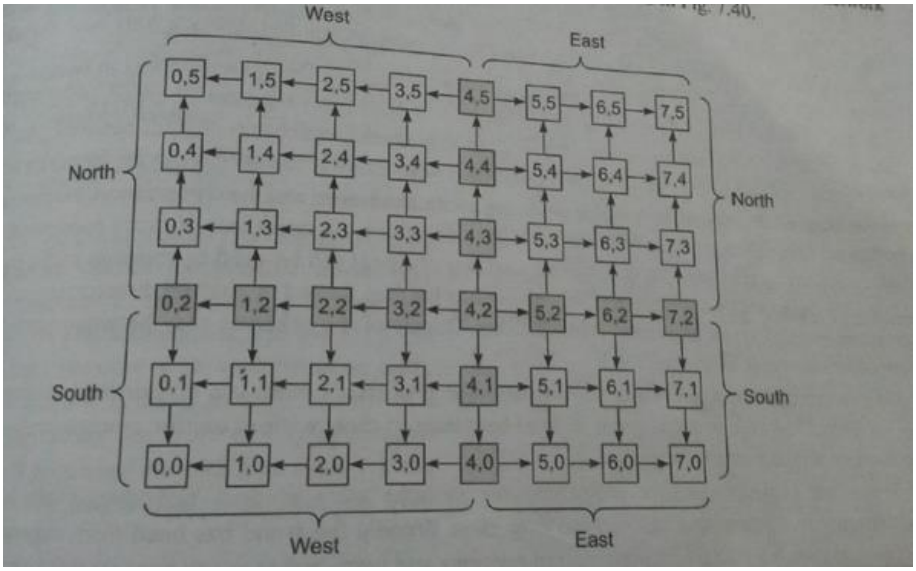
VIRTUAL NETWORK:

Used to reduce the network traffic.



NETWORK PARTIOING:

The concept of virtual networks leads to the partitioning of a given physical network into logical sub networks for multicast communication.



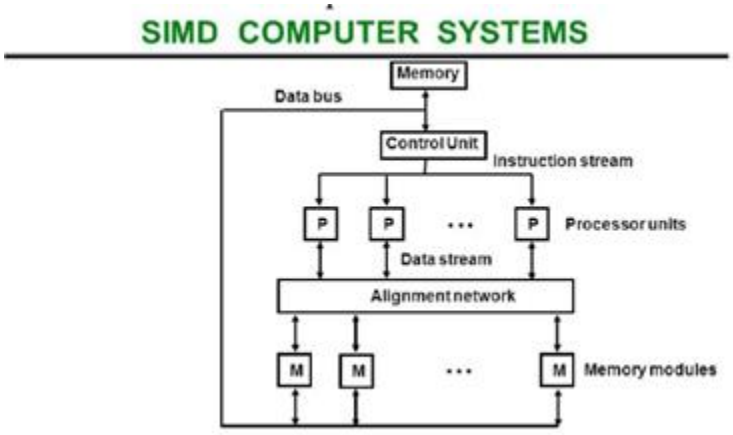
SIMD COMPUTER ORGANIZATIONS:

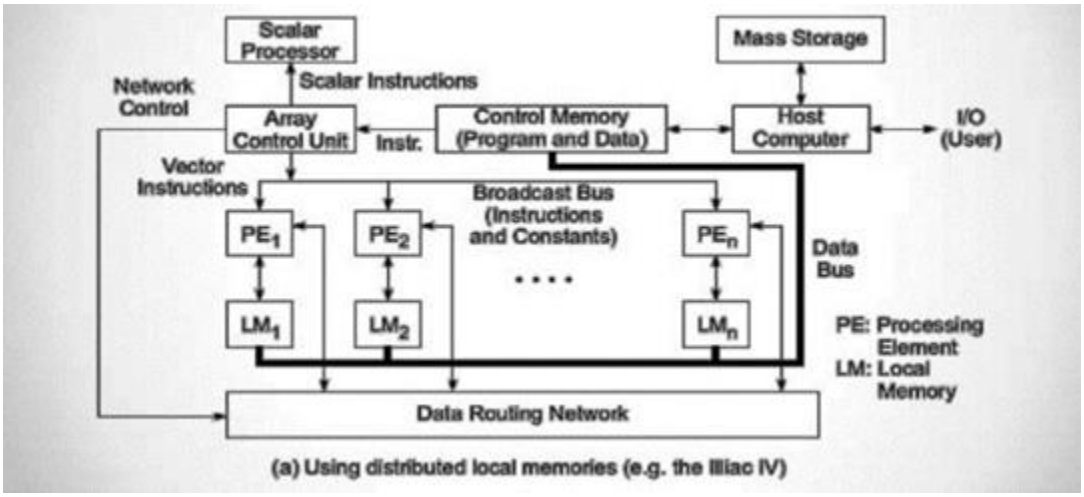
SIMD computers having a single instruction stream over multiple data streams. Vector processing can also be carried out SIMD computers.

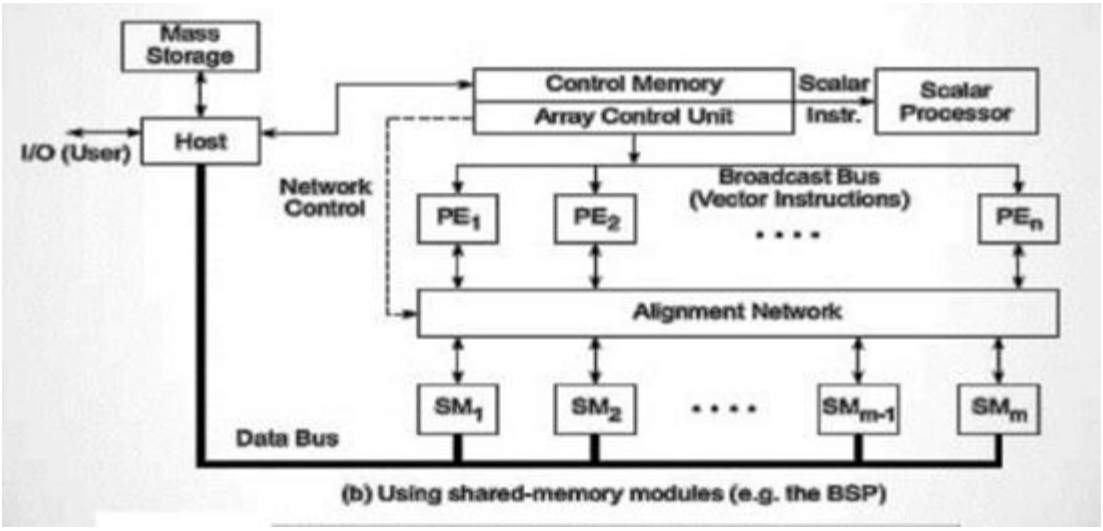
- Implementation modes
- The CM -2 architecture
- The masper MP-1 architecture

IMPLEMENDATION MODELS:

Two SIMD computer models are described below based on the memory distribution and addressing scheme based.







DIASTRIBUTED MEMORY MODEL:

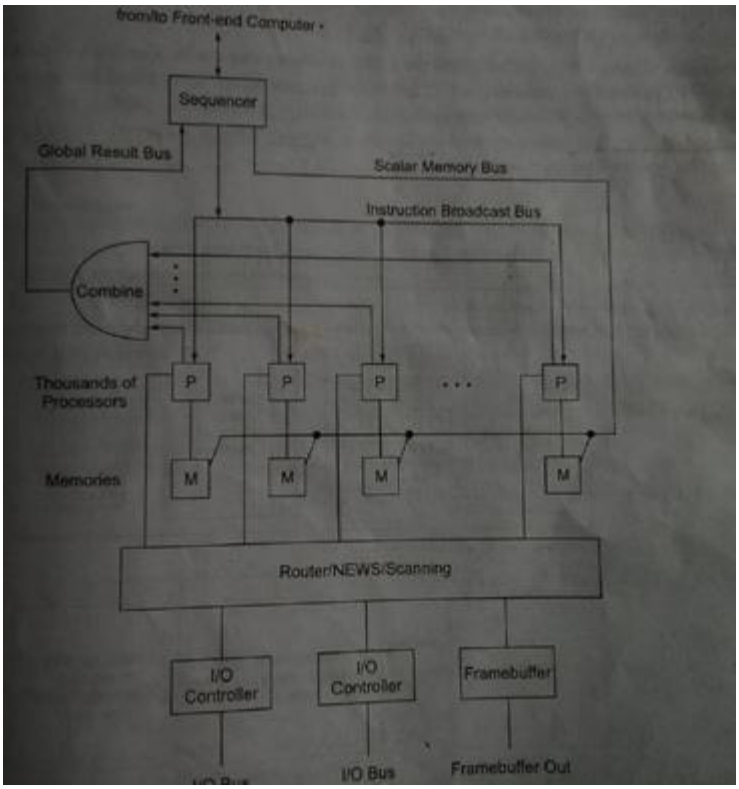
Spatial parallelism is exploited among the PES in an SIMD computer. A distributed memory SIMD computer consists of an array of PES which are controlled by the same array control unit.

THE CM – 2 ARCHITECTURE:

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 issued microinstruction to the back end processing array when data parallel operations were desired.



PROCESSING NODES:

CM -2 processors chips with memory and floating point chips. Each data processing code contained 32 bit slice data processors, an optimal floating point accelerator and interfaces for interprocessor communication.

THE MASSPAR MP – 1 ARCHITECTURE:

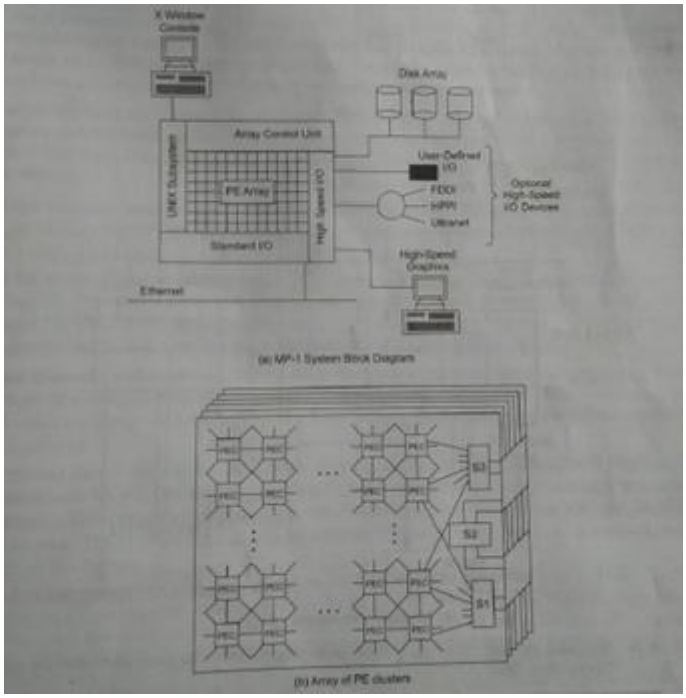
This was a medium grain SIMD computer quid different from the MP - 2 parallel architecture and MP – 1 hardware design are described below.

THE MASSPER MP – 1 ARCHITECTURE:

The MP – 1 architecture consisted of four subsystem, the PE array, the array control unit, a UNIX subsystem with standard I/O and a high speed I/O subsystem as depicted.

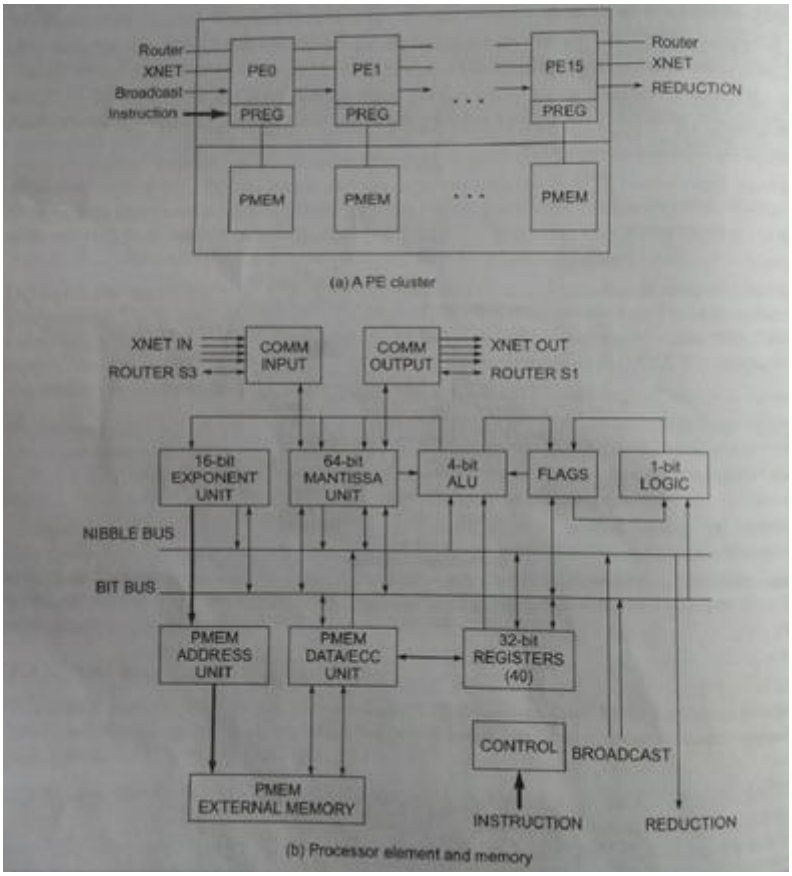
ARRAY CONTROL UNIT:

The ACU was a 14 MIPS scalar RISC processor using a demand paging instruction memory.



THE PE ARRAY:

Each processor board had 1024 PES and associated memory arranged as 64PE chip was connected to eight neighbors via the X- net mesh and a global multistage crossbar routing network labled s1, s2 and



Each PE cluster was composed of 16PES and 16 processors memories (PEMS). Interprocessor communication were carried but via three mechanisms.

1) ACU – PE array communication
2) N – net nearest neighbor communication
3) Global crossbar router communication

X – NET MESH INTERCONNECTS:

The X- net interconnect directly connected each PE with its eight neighbor in the two dimensional mesh. The connections to the PE array edges were wrapped around to from a 2 – D torus.

MULTISTAGE CROSSBAR INTERCONNECTS:

The network provided global communication between all PES and formed the basics for MP- 1 I/O system. Each PE cluster shared an originating port connected to router stage s1 and a target part connected to router stage s3.

PROCESSOR ELEMENTS AND MEMORY:

- The PE design had mostly data path logic and instruction fetch or decode logic. Both integer and floating point computational executed in each PE with register based RISC architecture.
- Most data movement with each PE occurred on the NIBBCE bus and the BIT bus.

PARALLEL DISK ARRAYS:

- Another feature worthily of mention is the massively parallel I/O architecture implemented in the MP – 1.
- The disk array provides up to 17.3 Gbytes of formatted capacity with a 9 – Mbytes/s sustained disk I/O rate.

THE CONNECTION MACHINE CM – 5:

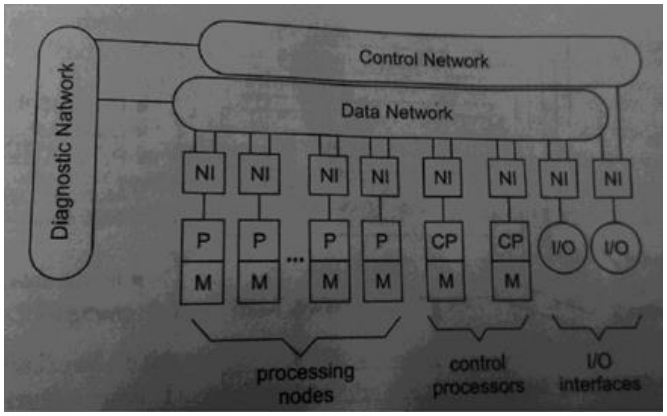
The grand challenge application drives the development of present and future MPP system to achieve higher and higher performance goals.

A SYNCHRONIZED MIMD MACHINE:

The CM -2 and its predecessor were criticized for having a rigid SIMD architecture, limiting general purpose application.

THE BULDING BLOCKS:

The machine was designed to contain from 32 to 16, 384 processing nodes each of which could have a 32 – MHz SPARC processor, 32 M bytes of memory and a 128 –M Flops vector processing unit capable of performing 64 – bit floating point operations.

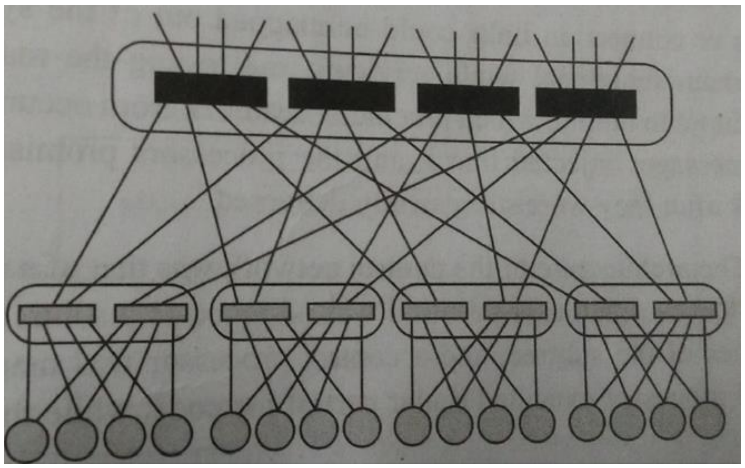


THE CM- 5 NETWORK ARCHITECTURE:

The data network was based on the fat – tree concept introduced by leiserson (1985), we explain how it is applied in CM – 5 construction.

FAT TREES:

A fat tree is more like a real tree in that it becomes thinker as it acquires more leaves. Processing nodes control processors and I/O channels are located are to leaves of a fat tree.



THE DATA NETWORK:

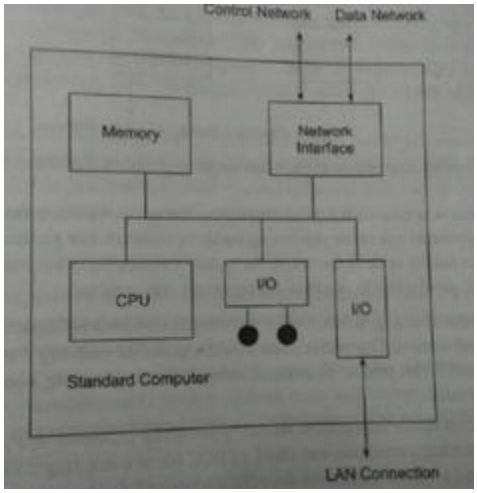
To route a message from one processor node to another , the message was sent up the tree to the least common ancestor of the two processors and then down to the destination.

CONTROL PROCESSSORS AND PROCESSING NODES:

The functional architecture of the control processing nodes is described in this subsection.

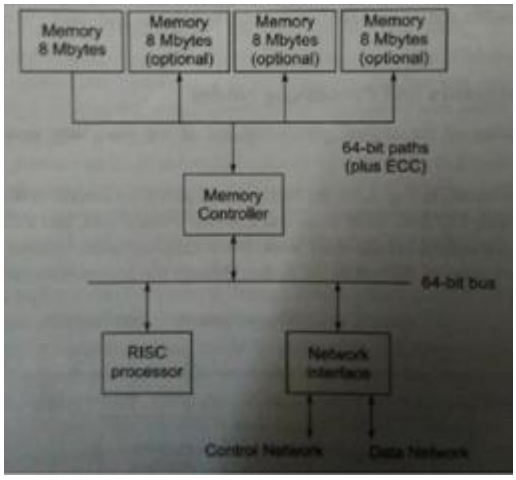
CONTROL PROCESSOR:

The basic control processor consisted of a RISC microprocessor (CPU), memory subsystem, I/O with local disks and Ethernet connections and a CM -5 network interface.



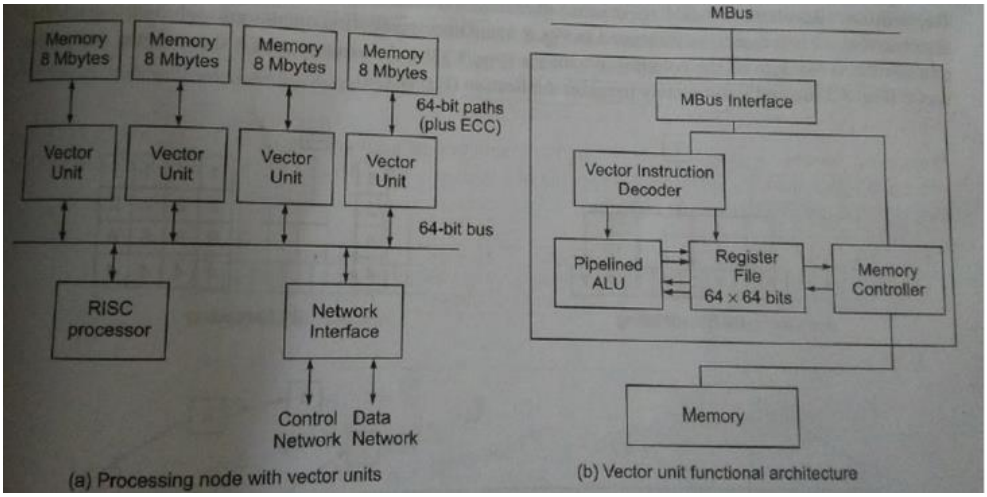
PROCESSING NODES:

It was a SPARC based processor with a memory subsystem, consisting of a memory controller and 8.16 or 32 Mbytes of DRAM memory.



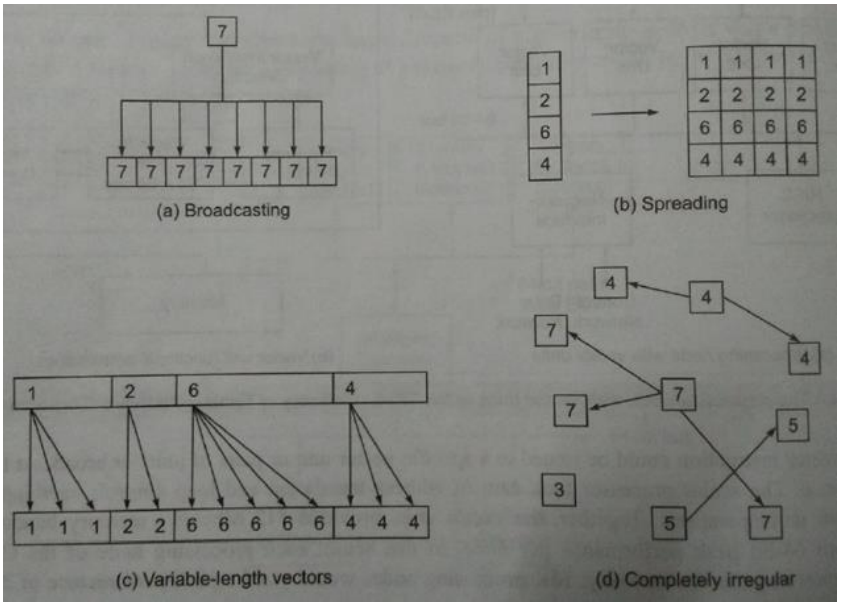
VECTOR UNITS:

Vector units could be added between the memory bank and the system bus as an optional feature.



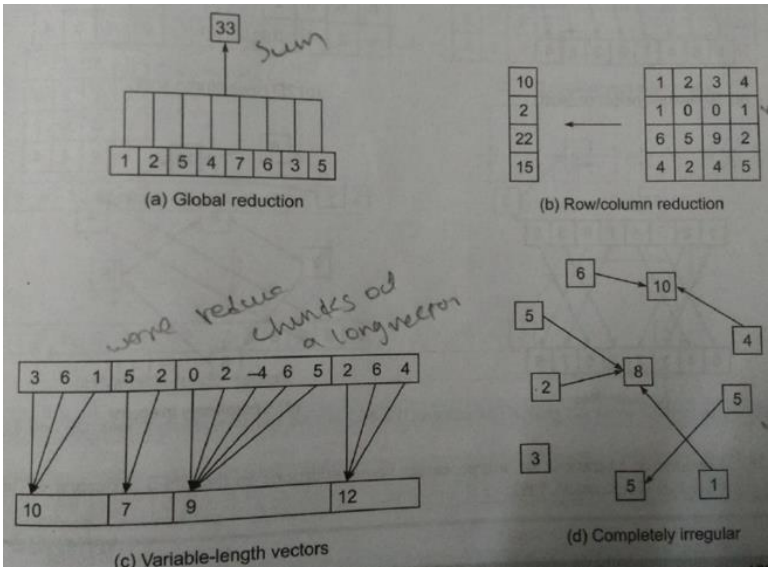
INTER PROCESSOR COMMUNICATION:

We have described the high speed scanning and spreading mechanisms built into the CM – 2. CM -2 in the CM – 5 these mechanism were designed to be further upgrade into four categories of interprocessors communication, application, reduction, permutation, parallel prefix.



REDUCTION:

Vector reduction was implemented on the CM – 2 by fast scanning, and on the CM – 5 the mechanism was further generalized as the opposite of replication.



FINE GRAIN MULTICOMPUTERS:

Message passing multicomputer are used to execute medium grain programs with approximately 10ms tasks size as in the IPSC.

FINE GRAIN PARALLELISM:

This comparison leads to the rationales for developing fine grain multicomputer.

LATENCY ANALYSIS:

- The computing granularity and communication and latency of leading early examples of multiprocessors, data parallel computers, and medium and fine grain multicomputer are summarized in table.
- The communication latency TC measure the data or message transfer time on a system interconnects. The sum TC + TS gives the total time required for IPC.

FINE GRAIN PARALLELISM:

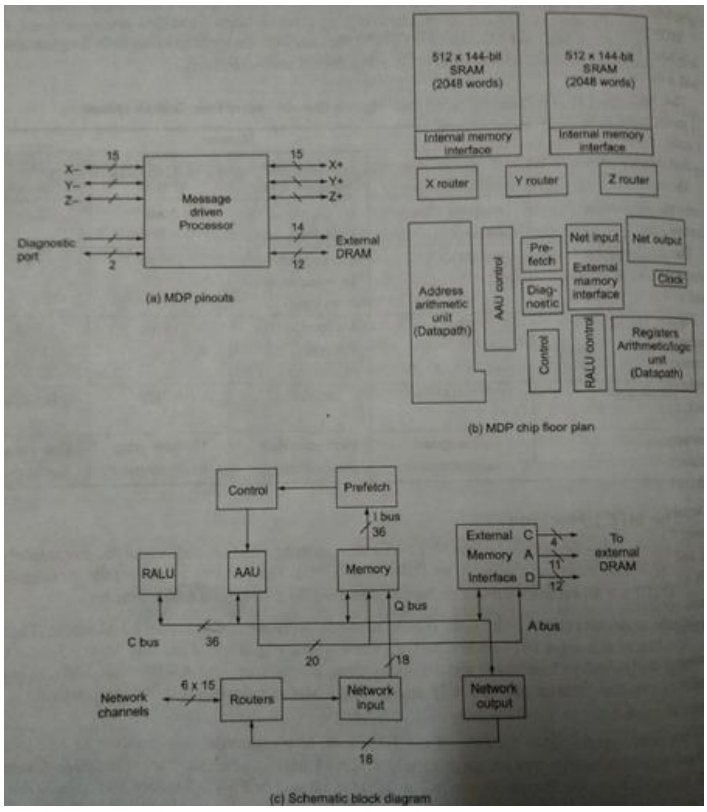
The grain size T_g is measured by the execution time of a typical program, including both computing time and communication time involved.

THE MIT J- MACHINE:

The architecture and building block of the MIT J- machine, its instruction set and system design consideration we described below based on the paper by dally et al (1992).

THE J – MACHINE ARCHITECTURE:

The K – ary n – cube networks were applied in the MIT – 5 machines. The initial prototype J – machine used a 1024 node network (8×8×16), which was a reduced 16 – ary 3 – cube with 8 nodes along the x and y dimensions and 16 nodes along the z – dimensions.



INSTRUCTION SET ARCHITECTURE:

The MDP extended a conventional microprocessor instruction set architecture with instruction to support parallel processing.

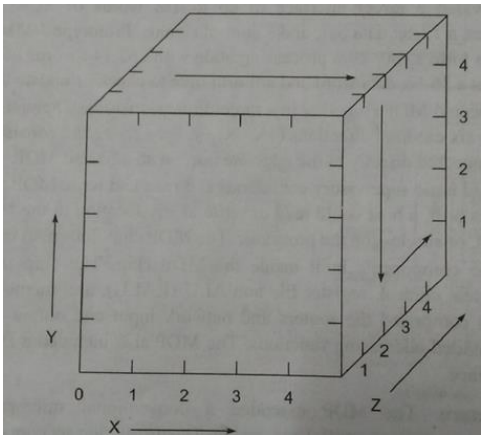
COMMUNICATION SUPPORT:

The MDP provide hardware support for end – to – end message delivery, buffering and task scheduling.

SEND R0, 0: Send net address (priority 0)

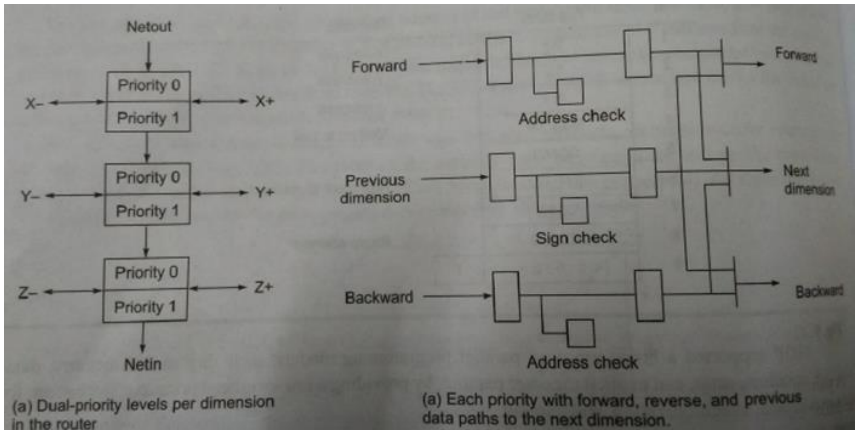
SEND R1, R2, 0: Header and receiver (priority 0)

SEND R3 (3, A3), 0: Selector and communication and message.



THE ROUTER DESIGN:

- The routers formed the switches in a J – machine network and delivered message their destinations.
- Each router contained two separate virtual networks with different priorities that shared the same physical channels.

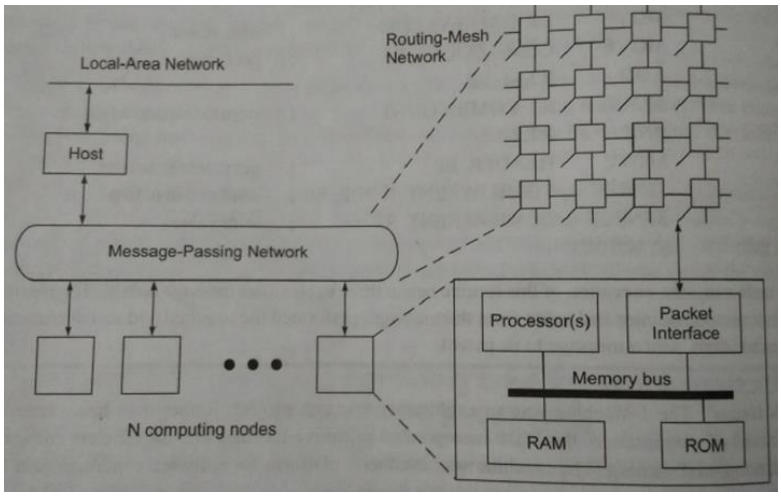


THE CALTECHN MOSAIC C:

The Caltech mosaic c was an experimental fine grain multicomputer that exployed single chip and advanced packaging technology to demonstrate the performance/cost advantages of fine grain multicomputer architecture.

FROM COSMIC CUBE TO MOSAIC C:

The programs in microelectronics over the preceding decode was such that mosaic nodes were=60 times faster, used=20 times less power, were=100 times smaller and were=25 times less expensive to manufacture than cosmic cube nodes.



MOSAIC C NODES:

The mosaic multicomputer node was a single 9.25 mm×10.00 mm chip fabricated in 1.2 μm feature size two level mental CMOS process. At 5-v operation, the synchronous parts of the chip separated with large margins at a 30 – MHZ clock rate and the chip dissipated=0.5 w.

MOSAIC C 8×8 BOARDS:

Sixty four mosaic chips were packaged by type automated bonding (TAB) in an 8×8 array on a circuit board. Host interface boards were also used to connect the mosaic arrays and workstation.

APPLICATION AND FUTURE TRENDS:

The mosaic may be taken as the origin of two scaling tracks.

1) Single chip nodes are a technologically attractive point in the design space in multicomputer.
2) It was also forecasts that constant node, computers centralized would allow a mosaic 8×8 board to be implemented as 9 single chip, with about 20 times the performance per node within 10 years.

UNIT -5

SOFTWARE FOR PARALLEL PROGRAMMING

- **Parallel programming models**
- **Parallel languages and compiler**
- **Dependence analysis of data**

PARALLEL PROGRAMMING MODEL:

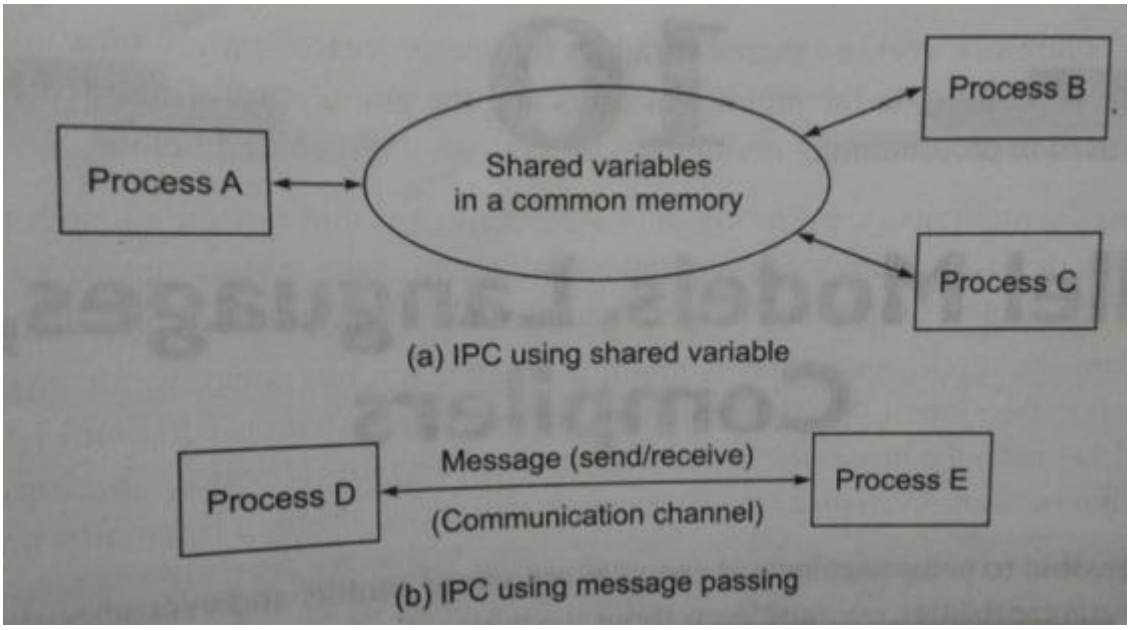
A programming model is a collection of program abstraction providing a programmer a simplified and set of transparent view of the computer hardware/software system. Five models are characterized below for these computers that exploit parallelism with different execution paradigms.

SHARED VARIABLE MODEL:

- In all programming systems, we consider processor active resources and memory and I/O devices passive resources.
- A program is a collection of processes. Parallelism depends on how interprocess communication (IPC) is implemented. Fundamental issues in parallel programming are centered around the specification creation, suspension, reaction, termination and synchronization or different processes.

SHARED VARIABLE COMMUNICATION:

Multiprocessor programming is based on the use of shared variables in a common memory for IPC.



Fine grain MIMD parallelism is exploited in tightly coupled multiprocessors. Interprocessor unconditionally, depending on the mechanisms used.

CRITICAL SECTION:

A critical seduction (CS) is a code segment accessing shared variables, which must be executed by only one process at a time and which one started must be completed without interruption.

- MUTUAL EXECLUSION – At most one process executing the CS at a time.
- NO DEADLOCK IN WAITING – No circular wait by two or more processes trying to enter the CS.
- NONPREEMPTION – No interrupt until completion, once entered the CS.
- EVANTUAL ENTRY – A process attempting to enter its CS will eventually succeed.

PROTECTED ACCESS:

- The main problem associated with the use of a CS is avoiding race conditions where concurrent process executing in different orders produce different results.
- Shared variable programming requires special atomic operations for IPC, new language constructs for expressing parallelism.

MULTIPROGRAMMING:

A multiprogrammed multiprocessor allows multiple programs to run concurrently through time sharing of all the processor in the system, multiple programs are inter leaved in their CPU and I/O activities.

MULTIPROCESSING:

In other words, we define MIMD multiprocessing with fine grain instruction level parallelism. MIMD multiprocessing exploits course grain procedure level parallelism. This is quite different from the operations implemented on a message passing system.

MULTITHREADING:

- The traditional UNIX/OS has a single threaded kernel in which only one process can receive OS kernel service at a time.
- The concept of multithreading is an extension of the concepts of multitasking and multiprocessing.

PARTIONING AND REPLICATION:

- The goal of parallel processing is to exploit parallelism as much as possible with the lowest overhead.
- Program replication refers to duplication of the same program code for parallel execution on multiple processors over different data sets.

SHEDULING AND SYNCHRONIZATION:

- Dynamic scheduling catches the run time conditions. However dynamic scheduling requires task context switching, preemption and much more OS support.
- In a conventional (IPC) is conducted at the process level. This mutual exclusion property is enforced with the use of locks, semaphores and monitors to be described.

CACHE COHERENCE AND PROTECTION:

- Besides maintaining data coherence in a memory hierarchy, multi processors must assume data consistency between private caches and the shared memory.

- These coherence control operations require special bus or network protocols for implemented.

MESSAGE PASSING MODELS:

Two processes D and E residing at different processor nodes may communicate with each other by passing message through a direct or indirect network.

- Synchronous message passing
- Asynchronous message passing
- Distributing the computation

SYNCHRONOUS MESSAGE PASSING:

- Synchronous message passing must synchronize the sender process and the receiver process in time and space, just like a telephone call using circuit switched lines.
- In a synchronous paradigm, the passing of a message must synchronize the sending process and the receiving process in time and space.

ASYNCHRONOUS MESSAGE PASSING:

- Asynchronous communication does not require that message sending and receiving by synchronized in time and space.
- Nonblocking can be achieved by asynchronous message passing in which two processors do not have to be synchronized either in timer or in space.
- Asynchronous communication requires to use of buffers to hold the messages along the path of the connecting channels.

DISTRIBUTING THE COMPUTATIONS:

Program replication and data distribution are used in multicomputer. The processors in a multicomputer (or a NORMA machine) are loosely coupled in the sense that they do not share memory.

DATA PARALLEL MODEL:

- Data parallel programs require the use of pre – distributed data structure makes a big difference in data parallel programming.
- It is applied to fine grain problems using regular grids.

- Hardware synchronization is enforced by the control unit to carry out the lock step execution of SIMD program.
    - Data Parallelism
    - Array Language Extensions
    - Compiler Support

DATA PARALLELISM:

- A latter SIMD computer, the connection machine CM -2 offered bit slice fine grain data parallelism using 16, 384 PES concurrently in a single array configuration. This demanded a lower degree of array segmentation and thus offered higher flexibility in programming.
- Instead by, inter PE communication are directly controlled by hardware.

ARRAY LANGUAGE EXTENSIONS:

Array extensions in data parallel languages are represented by high level data types. The array syntax enables the removed of some nested loops in the code and should reflect the architecture of the array processor.

EX:

Array preprocessing languages are CFD for Iliac IV, DAP Fortran for the AMT/distributed array processor. C* for the TMC/ connection machine.

COMPILER SUPPORT:

Compiler optimized control of SIMD machine hardware allows the programmer to drive the PE array transparently the compiler must separate the program into scalar and components with the OS environments.

OBJECT ORIENTED MODEL:

In this model, objects are dynamically created ands manipulated processing is performed by sending and receiving message among objects.

- Concurrent oop
- An Actor Model
- Parallelism Coop

CONCURRENT OOP:

- The popularity of object oriented programming (OOP) is attributed to three application demands.
- As a matter of fact, program abstraction leads to program modularity and software reusability as is commonly experienced with OOP.
- The development of concurrent object oriented programming (COOP) provides alternative models for multiprocessors or a multicomputer.

AN ACTOR MODEL:

Actors are self contained, iterative, independencecomponenys of a computing system that communicate by asynchronous message passing.

1) CREATE – Creating an actor from a behavior description and a set of parameters.
2) SEND – TO – Sending a message n to another actor.
3) BECOME – An actor replacing its own behavior by a new behavior.

PARALLELISM COOP:

Three common patterns of parallelism have been practice of COOP.

- FIRST, pipeline concurrency involves the overlapped enumeration of successive solution and testing of pipeline.
- SECOND, divide and conquer concurrency involve the concurrent elaboration of different subprogram to produce a solution to the overall problems.
- THIRD, pattern is called cooperative problem solving.

FUNCTIONAL AND LOGIC MODELS:

Two language oriented programming models for parallel processing are described below. We reveal opportunities for parallelism in these potential in AI applications.

- Functional Programming Model
- Logic Programming Model

FUNCTIONAL PROGRAMMING MODEL:

- A functional programming language emphasizes the functionality of a program and should not produce side effects after execution.
- The clack of side effects opens up much more opportunity for parallelism. Precedence restrictions occur only as a result of function application. All single assignment and dataflow language are functional in nature.

LOGIC PROGRAMMING MODEL:

- Based on predicated logic, logic programming is suitable for knowledge processing dealing with large database.
- Concurrent prolog, developed by Shapiro (1986), and parlog introduced by Clark (1987) are two parallel logic programming language.
- In many ways, the FGCS project was a marriage of parallel processing hardware and AI software.

PARALLEL LANGUAGES AND COMPILERS:

The environment for parallel computers is much more demanding than that for sequential computers. Users should not have no spend a lot of time programming hardware details.

LANGUAGE FEATURES FOR PARALLELISM:

Chang and smith (1990) classified the language feature for parallel programming into six categories according to functionality.

OPTIMAZATION FEATURES:

- AUTOMATED PARALLELIZER – Examples are: Express C automated parallizer and the Alliant FX Fortran compiler.
- SEMI AUTOMATED PARALLELIZER - Needs compiler directives or programmer's interaction such as DINO.
- ITERACTIVE RESTRUCTURE SUPPORT - Static analyzer, run time statistics, dataflow graph and code translate for restructuring Fortran code, such as the MIMD from pacific sierra.

AVAILABILITY FEATURES:

- SCALABILITY – The language is scalable to the number of processors available and independent of hardware topology.
- COMPABILTY – The language is compatible with an established sequential language.
- PORTABILITY – The language is portable to shared memory multiprocessors, message passing multicomputer or both.

SYNCHRONIZATION/ COMMUNICATION FEATURES:

- Single assignment languages
- Shared variables(locks) for IPC
- Logically shared memory

- Send and receive for message passing
- Remote procedure call
- Data flow languages

CONTROLL OF PARALLELISM:

- Coarse medium of mine grain
- Explicit versus implicit parallelism
- Global parallelism in the entire program
- Loop parallelism in iterations
- Task split parallelism
- Shared task queue
- Divide and conquer paradigm
- Shared abstract data types.

DATA PARALLELISM FEATURES:

- RUN TIME AUTOMATIC DECOMPOSITION - Data are automatically distributed with no user intervention as in express.
- MAPPING SPECIFICATION – Provides a facility for users to specify communication patterns or how data and process are mapped onto the hardware as in DIWO.
- VIRTUAL PROCESS SUPPORT – The compiler maps the virtual processors dynamically or statically onto the physical processors.
- DIRECT ACCESS TO SHARED DATA – Shared data can be directly accessed without monitor control as in Linda.
- SPMD SUPPORT – SPMD programming as in DIWO and hyper tasking.

PROCESS MANAGEMENT FEATURES:

- DYNAMIC PROCESS CREATION AT RUN TIME.
- LIGHTWEIGHT PROCESS – Compared to UNIX process.
- REPLICATED WORKS – Same program on every node with different data.
- PARTITIONED NETWORKS – Each processor or node might have more than one process and all processor nodes.
- AUTOMATIC LOAD BALANCING – The worked is dynamically migrated among busy and idle node to achive the same amount of at various processor nodes.

PARALLEL LANGUAGE CONSTRUCTS:

FORTRAN 90 ARRAY NOTATIONS:

A multidimensional data array is represented by an array name indexed by a sequence of subscript triples, one for each dimension. Triples for different dimensions are separated by common.

E1: e2: e3

E1: e2

E1: *: e3

E1: *

E1

*

PARALLEL FLOW CONTROL:

- The conventional Fortran DO loop declares that all scalar instruction within the (DO, End do) pair are executed sequentially and so are the successive iterations.
- When the successive iterations of a loop depend on each other, we use the(DO across, End across) pair to declare parallelism with loop carried dependences.

```
Do across I= 2, N
    Do J = 2, N
S1:     A (I, J) = (A (I (J-1) + A (I, J+1))/2
    End do
    End across
```

Another program construct is the (Cobegin, Coend) pair. Synchronizations among concurrent processes created within the pair are implied.

Cobegin

P1

P2

.

.

Pn

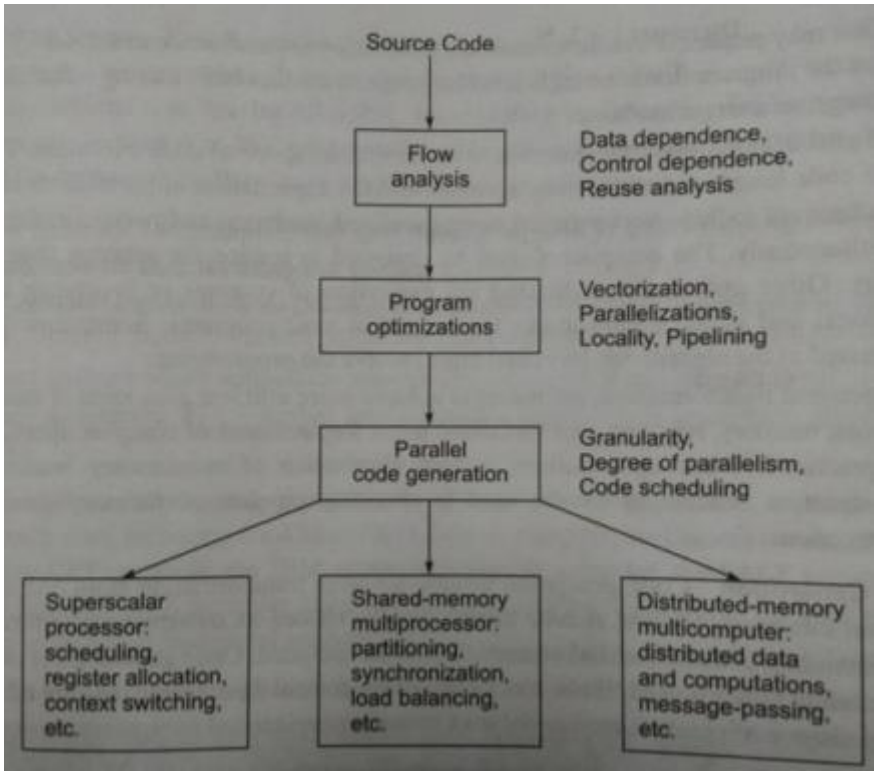
Coend

Causes process P1, P2, …….Pn to start simultaneously and to proceed concurrently until they have all ended.

The Join Q command recombines the two processes into one process.

OPTIMIZING COMPILERS FOR PARALLELISM:

Because high level languages are used almost exclusively to write programs today, compilers have become a necessity in modern computers.



FLOW ANALYSIS:

- This phase reveals the program flow patterns in order to determine data and control dependences in the source code.
- Generally speaking, instruction level parallelism is exploited in superscalar or VLSI processors.

PROGRAM OPTIMIZATIONS:

- This refers to the transformation of user programs in order to explore the hardware capabilities as possible.
- Transformation can be conducted at the loop level, locating level, or perfecting level with the ultimate goal of reaching global optimization.

Parallel code generation is very different for different computer classes.

- Different program use different pass unit and thus go though different sequence of transformation.

- Paraphrase is retagetable to produce code for different classes of parallel vector computers.

The PFC and Parascope:

- PFC (Allen and Jenney , 1984) performed syntax analysis including the following four steps:
- Interprocedural flow analysis using call graph.
- Standard transformations such as do-loop normalization subscript categorization deletion of dead etc….
- Dependence analysis which applied the seperability GCD and Bannered tests jointly.
- Vector code generation PFC further implemented a parallel code generation algorithm.

Commercial compilers:

Optimizing compilers have also been developed in a number of commercial parallel/ vector computer, including the alliant FX/F Fortran compiler the convex parallel/vectorizing compiler the cray CFT compiler and Intel IPSC-VX compiler.

Dependence analysis of data arrays:

Iteration space and dependence analysis:

Flow dependence antidependence and output dependence were defined for scalar data precise and efficient dependence for scalar data precise and efficient dependence tests are essential to the effectiveness of a parallelizing complier.

Dependence Testing:

Calculating data dependence far array is complicated by the fact. That two array references may not access the same memory location.

Do i1 = L_1, U_1

Do i_2 = L_2, U_2

……..

Do in = Ln, Un

S1: A (t (i_1,……in)…….t_m (i_1 ……in))……

S2: ……= A (g_1 (i_1 ……in),……..gm (i……in))…

End do

……

End do

End do

Dependence Equation:

Let α and β be vector of n integer index within the range of the upper and lower bounds of the n loops.

$$fi(\alpha)=gi(\beta)\ \forall i,\ 1 \leq i \leq m$$

Distance and direction vector:

Suppose there exists a data dependence for α=(α1, α2, ……..αn) and β(β1, β2,……..βn)

$$di = \begin{cases} < if\ \propto i\ < \beta i \\ = if\ \alpha i\ =\ \beta i \\ > if\ \alpha;\ > \beta i \end{cases}$$

For example, consider the following loop nest

Do I = L1, U1

Do j = L2, U2

Do k = L3, U3

A (i+1, j, k-1) = A (I, j, k) +c

End Do

End DO

End DO

The distance and direction vectors for the dependence between interations along three dimensions of the array A are (1, 0, -1) and (<, =, >) respectively.

Subscript Separability and partitioning:

Subscript categories:

- A subscript is said to be zero index variable (ZIV) if the subscript position contains no index no index in either references.
- Any subscript with more than one index is said to be multiple index variable (MIV).

Subscript separability:

If all the subscripts are separable we may compute the direction vector for each subscript independently and merge the direction vector on a positional baris with full precision.

Subscript portioning:

In the following loop nest the first subscript yields the direction vector (<) for the i=loop the second subscript yields the direction vector (=) for the J=loop.

```
Do I = L1, U1

Do J = L2, U2

A (i+1, 5) =A (I, N) +c

End do

End do
```

Thus a merge yields the following set of direction vector for both dimensions,

{(<,<) , (<,=) , (<,>)}

Categorized dependence tests:

The testing algorithm:

1. Partition the subscripts into separable and minimal coupled groups using the following algorithm.

Subscript partitioning algorithm (go++, and Tseng).

Input: A pair of m-dimensional algorithm array references containing subscript s1…. .Sm enclosed in n loops with indices I1……In.

Output: A set of portions P1……Pn n≤n, each containing a separable or minimal coupled group

Test categories:

A subscript expression is linear if it has the form a1 i1 + a2 i2 + ……. +an in Te where $i_{k\ is}$ the index for the loop at nesting level k, expressions.

The ZIV test:

The ZIV test is a dependence test performance on two loops invariant expressions if the difference simplifies to a none zero constant, we have proved independence.

The SIV test:

An SIV subscript for index I is said to be strong if it has from,

(Ai+c1, ai+c2), i.e.

$$d = i^1 - i\ = \frac{c1 - c2}{a}$$

Dependence exists it and only it d is an integrated$|d| \leq u - l$, where u and l are the loop under and lower bounds,

$$Direction = \begin{cases} < if\ d < 0 \\ = if\ d\ = 0 \\ = if\ d\ = 0 \end{cases}$$

Were Zero SIV test:

The care in which a_1=0 or a_2 =0 is called a weak – zero SIV subscript it a_2 = 0 the dependence equation reduces to

$$i = \frac{c2 - c1}{a1}$$

Weak crossing SIV test:

All subscribe where a_2 = a, are weak crossing SIV there subscribe typically occur as part of choler decomposition.

$$i = \frac{c2 - c}{2a}$$

The MIV tests:

SIV tests can be extended to handle complex iteration spaces where loop bounds may be functions of other loop includes.

EX:

Triangular or trapezoidal loops.