Module -1 Question Bank

1. Explain the Generation of Electronic Computers?

Each of the first 3rd generation lasted about 10 years and the 4th generation was there for 15 years. The 5th generation has processors and memory devices with more than 1 billion transistors on a single silicon chip. Division into generations marked primarily by changes in hardware and software technologies.

Table 1.1 Five Generations 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.

First generation:

- Computers were built with a Single CPU which Performs serial fixed point arithmetic using a program counter, branch instruction, accumulator.
- CPU is used in all memory access and I/O operations.
- Machine / assembly language were used in first generation.

Second generation:

- Index registers, floating point arithmetic, multiplexed memory, I/O processors were included.
- High Level Language - Fortran, Algol, Cobol were introduced along with compilers, subroutine libraries and batch processing monitors.
- Register transfer language was developed by Irving Reed (1957) for Systematic design of digital computers in the second generation.

Third generation:

- Micro-programmed control was used.
- Pipelining and cache memory used to close up the speed gap between CPU and main memory.
- Multiprogramming was implemented to interleave CPU and I/O activities across multiple user programs.
- Time sharing OS using virtual memory with sharing or multiplexing of resources.

Fourth generation:

- Parallel computers with different architecture were introduced and 4^{th} generation computer make use of distributed or shared or optional vector hardware.
- Multiprocessing OS, special languages, compilers were created for parallelism and parallel processing or distributed computing software tools were developed.

Fifth generation:

- The system highlights the Superscalar processors, cluster computers, massively parallel processing (MPP).
- MPP system are adopted by
  - Scalable and latency tolerant architectures.
  - VSLI advanced technologies.
  - high density packaging.
  - Optical technologies.

2. Explain Elements of a modern computer system with neat diagram?

The hardware, software and programming elements of modern computer systems can be characterized by looking at a variety of factors in context of parallel computing these factors are:

- Computing problems
- Algorithms and data structures
- Hardware resources
- Operating systems
- System software support
- Compiler support

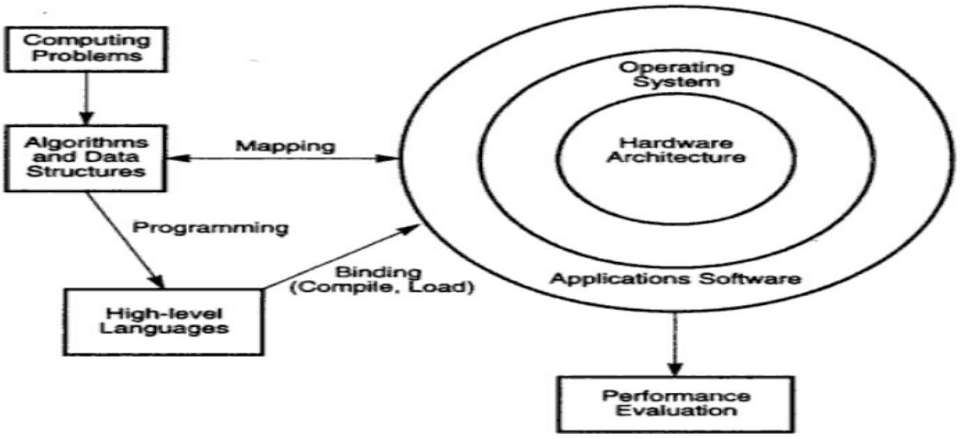


Figure 1.1 Elements of a modern computer system.

Computing Problems

- The use of a computer is driven by real-life problems demanding fast and accurate solutions. Depending on the nature of the problems, the solutions may require different computing resources.
- For numerical problems in science and technology, the solutions demand complex mathematical formulations and tedious integer or floating-point computations.
- For alpha numerical problems in business and government, the solutions demand accurate transactions, large database management, and information retrieval operations.
- For artificial intelligence (AI) problems, the solutions demand logic inferences and symbolic manipulations.
- These computing problems have been labeled numerical computing, transaction processing, and logical reasoning.
- Some complex problems may demand a combination of these processing modes.

Algorithms and data structures:

- Needed to specify the computations and communications in computing problems.
- Numeric problems are deterministic.
- Symbolic processing- heuristics or non deterministic searches over large databases.
- Problem formulation and the development of parallel algorithms often require interdisciplinary interactions among the theoreticians, experimentalists and computers programmers.

Hardware Resources:

- A modern computer system demonstrates its power through coordinated efforts by hardware resources, an operating system, and application software.
- Processors, memory, and peripheral devices form the hardware core of a computer system.

- Special hardware interfaces are often built into I/O devices, such as terminals, workstations, optical page scanners, magnetic ink character recognizers, modems, file servers, voice data entry, printers, and plotters.
- These peripherals are connected to mainframe computers directly or through local or wide-area networks

Operating System:

- An effective operating system manages the allocation and deallocation of resources during the execution of user programs.
- Beyond the OS, application software must be developed to benefit the users.
- Standard benchmark programs are needed for performance evaluation.
- Mapping is a bidirectional process matching algorithmic structure with hardware architecture, and vice versa.
- Efficient mapping will benefit the programmer and produce better source codes.
- The mapping of algorithmic and data structures onto the machine architecture includes processor scheduling, memory maps, interprocessor communications, etc. These activities are usually architecture-dependent.

System Software Support:

- Software support is needed for the development of efficient programs in high-level languages. The source code written in a HLL must be first translated into object code by an optimizing compiler.
- The compiler assigns variables to registers or to memory words and reserves functional units for operators.
- An assembler is used to translate the compiled object code into machine code which can be recognized by the machine hardware. A loader is used to initiate the program execution through the OS kernel.

Compiler support:

There are three compiler upgrade approaches:

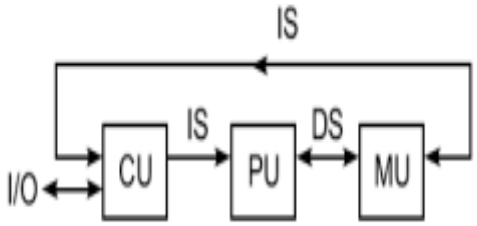
Preprocessor: A preprocessor uses a sequential compiler and a low-level library of the target computer to implement high-level parallel constructs.

Precompiler: The precompiler approach requires some program flow analysis, dependence checking, and limited optimizations toward parallelism detection.

Parallelizing Compiler: This approach demands a fully developed parallelizing or vectorizing compiler which can automatically detect parallelism in source code and transform sequential codes into parallel constructs.

3. Write a short note on Flynn's Classification based on notions of instruction and data streams?

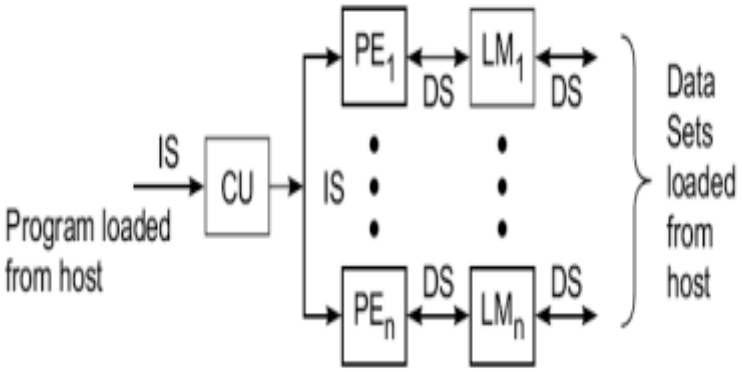
1. SISD (Single Instruction stream over a Single Data stream) computers



(a) SISD uniprocessor architecture

- Conventional sequential machines are called SISD computers. They are also called scalar processor i.e., one instruction at a time and each instruction have only one set of operands.
- Single instruction: only one instruction stream is being acted on by the CPU during any one clock cycle.
- Single data: only one data stream is being used as input during any one clock cycle deterministic execution.
- Instructions are executed sequentially. This is the oldest and until recently, the most prevalent form of computer.
- Examples: most PCs, single CPU workstations and mainframes

2. SIMD (Single Instruction stream over Multiple Data streams) machines



(b) SIMD architecture (with distributed memory)

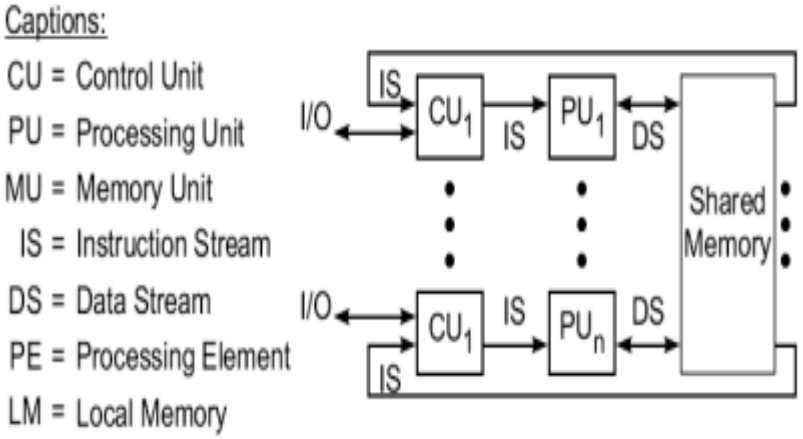
A type of parallel computer

- Single instruction: All processing units execute the same instruction issued by the control unit at any given clock cycle.

- Multiple data: Each processing unit can operate on a different data element. The processors are connected to shared memory or interconnection network providing multiple data to processing unit.
- This type of machine typically has an instruction dispatcher, a very high-bandwidth internal network, and a very large array of very small-capacity instruction units. Thus single instruction is executed by different processing unit on different set of data.
- Best suited for specialized problems characterized by a high degree of regularity, such as image processing and vector computation.
- Synchronous (lockstep) and deterministic execution.
- Two varieties: Processor Arrays e.g., Connection Machine CM-2, Maspar MP-1, MP-2 and Vector Pipelines processor e.g., IBM 9000, Cray C90, Fujitsu VP, NEC SX-2, Hitachi S820.

3. MIMD (Multiple Instruction streams over Multiple Data streams) machines.

- A single data stream is fed into multiple processing units.
- Each processing unit operates on the data independently via independent instruction streams.
- A single data stream is forwarded to different processing unit which are connected to different control unit and execute instruction given to it by control unit to which it is attached. Thus in these computers same data flow through a linear array of processors executing different instruction streams.
- This architecture is also known as Systolic Arrays for pipelined execution of specific instructions. Some conceivable uses might be:
  1. multiple frequency filters operating on a single signal stream
  2. multiple cryptography algorithms attempting to crack a single coded message.

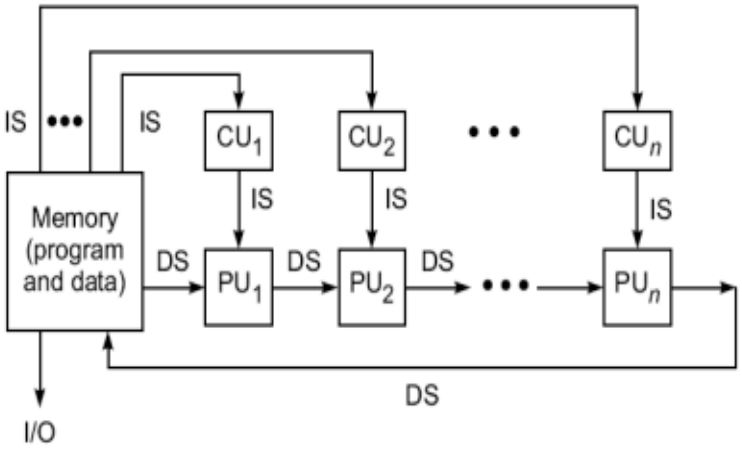


(c) MIMD architecture (with shared memory)

4. MISD (Multiple Instruction streams and a Single Data stream) machines

- Multiple Instructions: Every processor may be executing a different instruction stream.

- Multiple Data: Every processor may be working with a different data stream, multiple data stream is provided by shared memory. Can be categorized as loosely coupled or tightly coupled depending on sharing of data and control.
- Execution can be synchronous or asynchronous, deterministic or non-deterministic There are multiple processors each processing different tasks.
- Examples: most current supercomputers, networked parallel computer "grids" and multi-processor SMP computers - including some types of PCs.



(d) MISD architecture (the systolic array)

Fig. 1.3 Flynn's classification of computer architectures (Derived from Michael Flynn,

4. Explain the Implicit and Explicit Parallelism, with a neat diagram?

Implicit Parallelism

- An implicit approach uses a conventional language, such as C, Fortran, Lisp, or Pascal, to write the source program.
- The sequentially coded source program is translated into parallel object code by a parallelizing compiler.
- The compiler must be able to detect parallelism and assign target machine resources. This compiler approach has been applied in programming shared-memory multiprocessors.
- With parallelism being implicit, success relies heavily on the "intelligence" of a parallelizing compiler. This approach requires less effort on the part of the programmer.

Explicit Parallelism

- The second approach (Fig. 1.5b) requires more effort by the programmer to develop a source program using parallel dialects of C, Fortran, Lisp, or Pascal.
- Parallelism is explicitly specified in the user programs. This will significantly reduce the burden on the compiler to detect parallelism.

- Instead, the compiler needs to preserve parallelism and, where possible, assigns target machine resources.

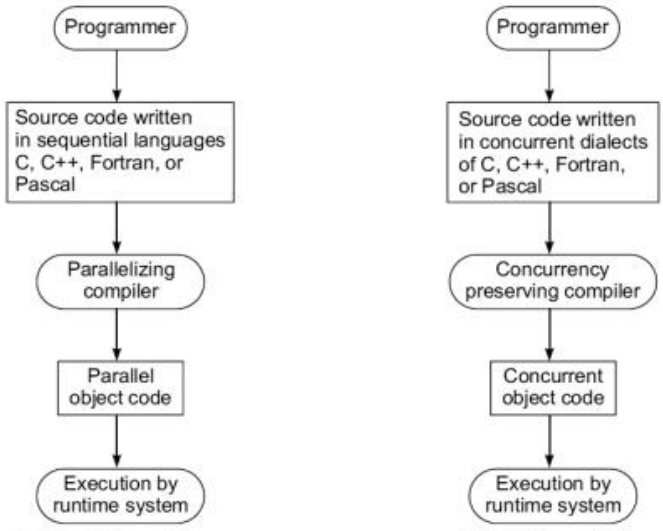


Fig. 1.5 Two approaches to parallel programming (Courtesy of Charles Seitz; adapted with permission from

5. Write a short note on:

a. Clock Rate and CPI **b. Performance factors**

c. MIPS Rate **d. Throughput Rate**

Clock Rate and CPI

- The CPU (or simply the processor) of today's digital computer is driven by a clock with a constant cycle time (τ in nanoseconds).
- The inverse of the cycle time is the clock rate ($f = 1/\tau$ in megahertz). The size of a program is determined by its instruction count (Ic), in terms of the number of machine instructions to be executed in the program.
- Different machine instructions may require different numbers of clock cycles to execute. Therefore, the cycles per instruction (CPI) becomes an important parameter for measuring the time needed to execute each instruction.
- For a given instruction set, can calculate an average CPI over all instruction types, provided to know their frequencies of appearance in the program.
- An accurate estimate of the average CPI requires a large amount of program code to be traced over a long period of time.
- Unless specifically focusing on a single instruction type, simply use the term CPI to mean the average value with respect to a given instruction set and a given program mix.

Performance Factors

- Let Ic be the number of instructions in a given program, or the instruction count. The CPU time (T in seconds/program) needed to execute the program is estimated by finding the product of three contributing factors: T = Ic x CPI x τ
- The execution of an instruction requires going through a cycle of events involving the instruction fetch, decode, operand(s) fetch, execution, and store results. In this cycle, only the instructions decode and execution phases are carried out in the CPU.
- The remaining three operations may be required to access the memory. Usually, a memory cycle is k times the processor cycle τ. The value of k depends on the speed of the memory technology and processor-memory interconnection scheme used.
- 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.

  $$T = Ic \times (p + m \times k) \times \tau$$

  - where p is the number of processor cycles needed for the instruction decode and execution,
  - m is the number of memory references needed,
  - k is the ratio between memory cycle and processor cycle,
  - Ic is the instruction count,
  - r is the processor cycle time.

MIPS Rate

- Let C be the total number of clock cycles needed to execute a given program. Then the CPU time in Equation can be estimated as T = C x τ = C/f. Furthermore, CPI = C/Ic and T = Ic x CPI x τ = Ic x CPI/f. The processor speed is often measured in terms of million instructions per second (MIPS), simply call it the MIPS rate of a given processor.
- It should be emphasized that the MIPS rate varies with respect to a number of factors, including the clock rate (f), the instruction count (Ic), and the CPI of a given machine, as defined below:

$$\text{MIPS rate} = \frac{I_c}{T \times 10^6} = \frac{f}{\text{CPI} \times 10^6} = \frac{f \times I_c}{C \times 10^6}$$

- The CPU time can also be written as T = Ic X 10-6 / MIPS. All four system attributes, instruction set, compiler, processor, and memory technologies, affect the MIPS rate, which varies also from program to program.

Throughput Rate

- Number of programs a system can execute per unit time, called the system throughput Ws (in programs/second).
- In a multi-programmed system, the system throughput is often lower than the CPU throughput Wp defined by:

$$W_p = \frac{f}{I_c \times CPI} \qquad (1.4)$$

- Note that Wp = (MIPS) X 106 /Ic from Eq. 1.3. The unit for Wp is programs/second. The CPU throughput is a measure of how many programs can be executed per second, based on the MIPS rate and average program length (Ic).
- The reason why Ws < Wp is due to the additional system overheads caused by the I/O, compiler and OS when multiple programs are interleaved for CPU execution by multiprogramming or timesharing operations.
- If the CPU is kept busy in a perfect program-interleaving fashion, then Ws = Wp. This will probably never happen, since the system overhead often causes an extra delay and the CPU may be left idle for some cycles.

6. Describe the three shared-memory multiprocessor models?

i) Uniform Memory-Access (UMA) model

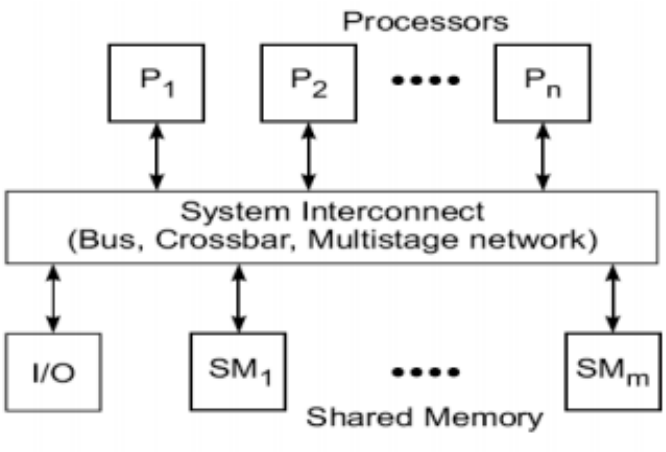


Fig. 1.6 The UMA multiprocessor model

- In a UMA multiprocessor model (Fig. 1.6), the physical memory is uniformly shared by all the processors. All processors have equal access time to all memory words, which is called uniform memory access.
- Each processor may use a private cache. Peripherals are also shared in some fashion. Multiprocessors are called tightly coupled systems due to the high degree of resource sharing. The system interconnect takes the form of a common bus, a crossbar switch, or a multistage network.

- Most computer manufacturers have multiprocessor (MP) extensions of their uniprocessor (UP) product line. The UMA model is suitable for general-purpose and timesharing applications by multiple users.
- It can be used to speed up the execution of a single large program in time-critical applications. To coordinate parallel events, synchronization and communication among processors are done through using shared variables in the common memory.
- When all processors have equal access to all peripheral devices, the system is called a symmetric multiprocessor. In this case, all the processors are equally capable of running the executive programs, such as the OS kernel and I/O service routines.

ii. Non uniform-Memory-Access (NUMA) model

- A NUMA multiprocessor is a shared-memory system in which the access time varies with the location of the memory word. Two NUMA machine models are depicted in Fig. 1.7.

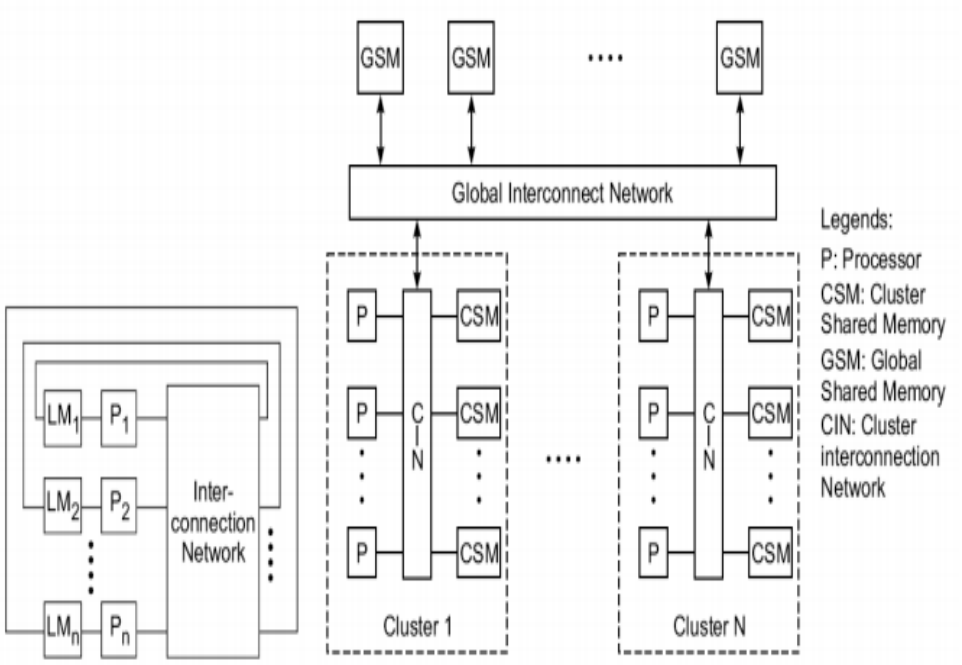


Fig. 1.7 Two NUMA models for multiprocessor systems

- The shared memory is physically distributed to all processors, called local memories. The collection of all local memories forms a global address space accessible by all processors. It is faster to access a local memory with a local processor.
- The access of remote memory attached to other processors takes longer due to the added delay through the interconnection network. The BBN TC-2000 Butterfly multiprocessor assumes the configuration shown in Fig. 1.7a.
- It is faster to access local memory with a local processor.

- The access of remote memory attached to other processor takes longer due to the added delay through interconnection network.
- Besides distributed memories, globally shared memory can be added to a multiprocessor system.
- There are 3 memory access patterns:
  - Fastest local memory access
  - Global memory access
  - Slowest is access of remote memory
- The processors are divided in to several clusters. Each clusters is itself an UMA or a NUMA multiprocessor. The clusters are connected to global shared memory modules.
- The entire system is considered a NUMA multiprocessor.
- All processors belonging to the same cluster are allowed to uniformly access the cluster shared memory modules. All clusters have equal access to the global memory.
- Access time in the cluster memory is shorter than that to the global memory.

iii. Cache-Only Memory Architecture (COMA) model

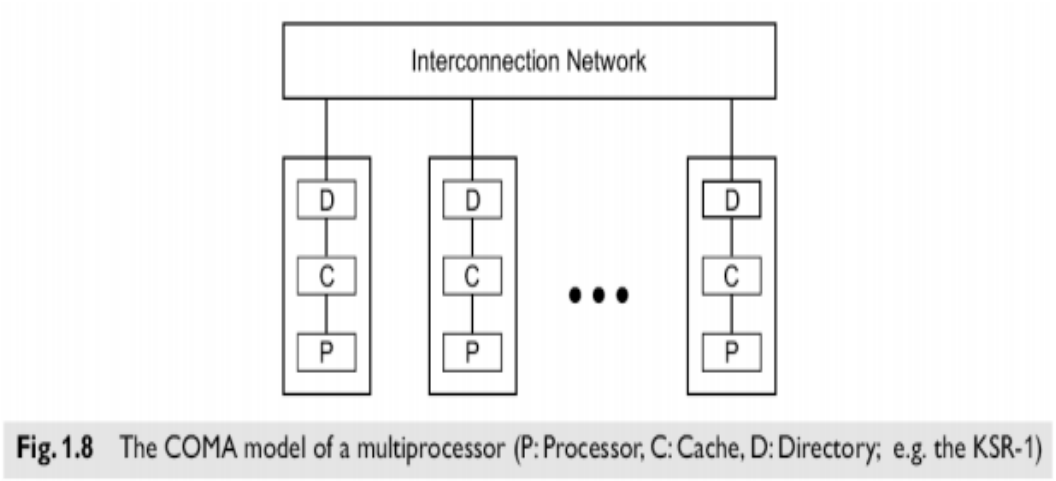


Fig. 1.8 The COMA model of a multiprocessor (P: Processor, C: Cache, D: Directory; e.g. the KSR-1)

- A multiprocessor using cache-only memory assumes the COMA model.
- The COMA model is a special case of a NUMA machine, in which the distributed main memories are converted to caches. There is no memory hierarchy at each processor node. All the caches form a global address space. Remote cache access is assisted by the distributed cache directories (D in Fig. 1.8).
- Depending on the interconnection network used, sometimes hierarchical directories may be used to help locate copies of cache blocks. Initial data placement is not critical because data will eventually migrate to where it will be used.
- Besides the UMA, COMA, NUMA models, other variations exists multiprocessors. Cache coherent non uniform memory access (CC-NUMA) [distributed shared memory and cache directories].
- Cache coherent COMA machine is one in which all cache copies must be kept consistent.

7. Explain the concept of Vector Supercomputers?

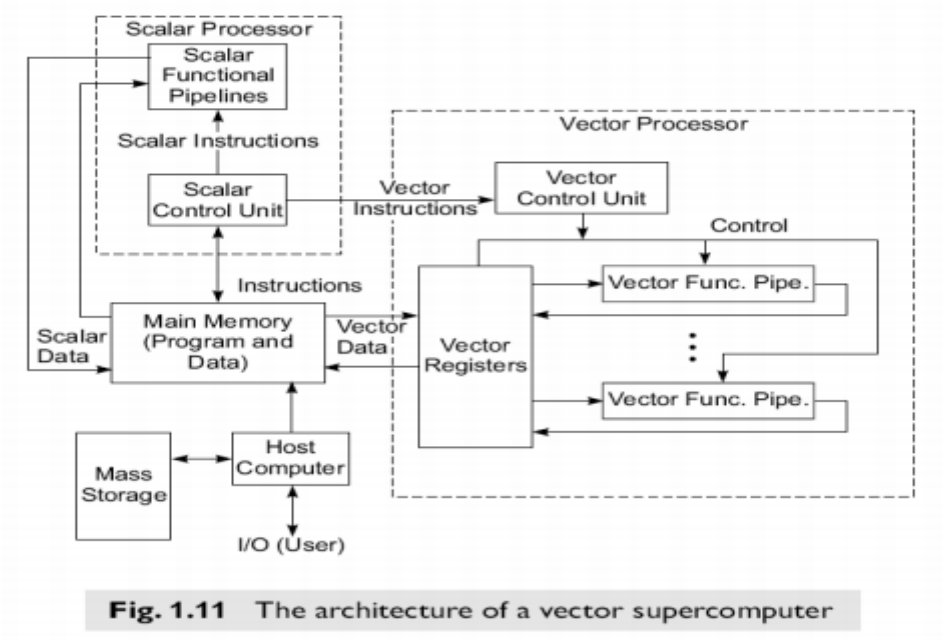


Fig. 1.11 The architecture of a vector supercomputer

- A vector computer is often built on top of a scalar processor. As shown in Fig. 1.11, the vector processor is attached to the scalar processor as an optional feature.
- Program and data are first loaded into the main memory through a host computer. All instructions are first decoded by the scalar control unit. If the decoded instruction is a scalar operation or a program control operation, it will be directly executed by the scalar processor using the scalar functional pipelines.
- If the instruction is decoded as a vector operation, it will be sent to the vector control unit. This control unit will supervise the flow of vector data between the main memory and vector functional pipelines.
- The vector data flow is coordinated by the control unit. A number of vector functional pipelines may be built into a vector processor.

Vector Processor Models

- Figure l. l l shows a register-to-register architecture.
- Vector registers are used to hold the vector operands, intermediate and final vector results. The vector functional pipelines retrieve operands from and put results into the vector registers.
- All vector registers are programmable in user instructions. Each vector register is equipped with a component counter which keeps track of the component registers used in successive pipeline cycles.
- The length of each vector register is usually fixed, say, sixty-four 64-bit component registers in a vector register in a Cray Series supercomputer. Other machines, like the Fujitsu VP2000 Series, use reconfigurable vector registers to dynamically match the register length with that of the vector operands.

8. Explain SIMD Supercomputers?

SIMD computers have a single instruction stream over multiple data streams. 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 Illiac IV had 64 PEs and the Connection Machine CM-2 had 65,536 PEs.

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

3. I s the set of instructions broadcast by the CU to all PEs for parallel execution. These include arithmetic, logic, data routing, masking, and other local operations executed by each active PE over data within that PE.

4. M is the set of masking schemes, where each mask partitions the set of PEs into enabled and disabled subsets. R is the set of data-routing functions, specifying various patterns to be set up in the interconnection network for inter-PE communications.

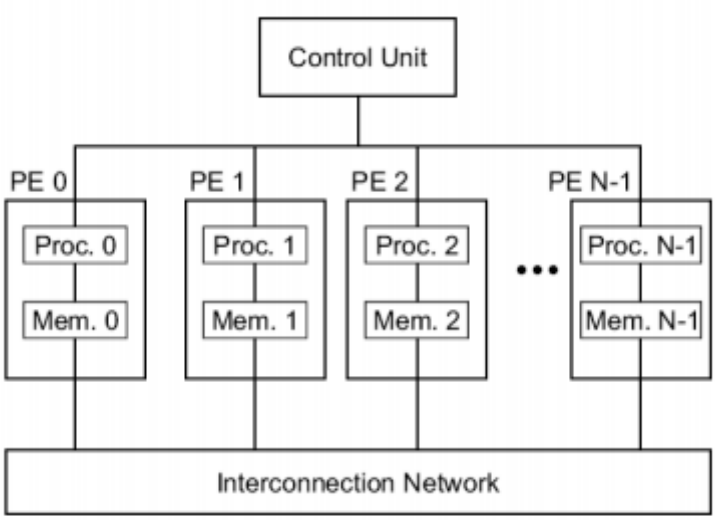


Fig. 1.12 Operational model of SIMD computers

9. Write short notes on:

a. PRAM Models **b. VLSI Complexity Model**

a. PRAM Models

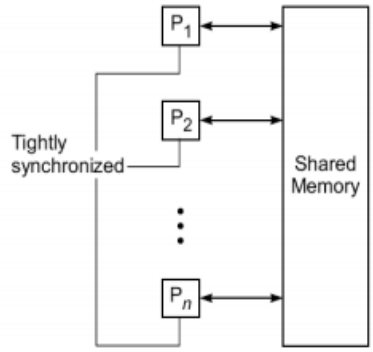


Fig. 1.14 PRAM model of a multiprocessor system with shared memory, on which all *n* processors operate in lockstep in memory access and program execution operations. Each processor can access any memory location in unit time

An n-processor PRAM (Fig. 1.14) has a globally addressable memory. The shared memory can be distributed among the processors or centralized in one place. The n processors operate on a synchronized read-memory, compute and write-memory cycle. With shared memory, the model must specify how concurrent read and concurrent write of memory are handled. Four memory-update options are possible:

Exclusive Read (ER) — this allows at mast one processor to read from any memory location in each cycle, a rather restrictive policy.

Exclusive Write (EW) — this allows at most one processor to write into a memory location at a time.

Concurrent Read (CR) — this allows multiple processors to read the same information from the same memory cell in the same cycle.

Concurrent Write (CW) — this allows simultaneous writes to the same memory location. In order to avoid confusion, some policy must be set up to resolve the write conflicts.

PRAM Variants: There are 4 variants of the PRAM model, depending on how the memory reads and writes are handled.

EREW – PRAM Model (Exclusive Read, Exclusive Write): This model forbids more than one processor from reading or writing the same memory cell simultaneously. This is the most restrictive PRAM model proposed.

CREW – PRAM Model (Concurrent Read, Exclusive Write): The write conflicts are avoided by mutual exclusion. Concurrent reads to the same memory location arc allowed.

ERCW – PRAM Model (Exclusive Read, Concurrent Write): This allows exclusive read or concurrent writes to the same memory location.

CRCW – PRAM Model (Concurrent Read, Concurrent Write): This model allows either concurrent reads or concurrent writes to the same memory location.

b. VLSI Complexity Model

Parallel computers rely on the use of VLSI chips to fabricate the major components such as processor arrays memory arrays and large scale switching networks. The rapid advent of very large scale integrated (VSLI) technology now computer architects are trying to implement parallel algorithms directly in hardware.

An AT2 model is an example for two dimension VLSI chips.

- S- problem size involved in the computation. For certain computations, there exists a lower bound f(s) such that A X T^2 >= O(f(s)). Chip area A is a measure of the chip's complexity.
- Latency T is the time required from when inputs are applied until all outputs are produces for a single problem instance.
- The chip is represented by the base area in the 2 horizontal dimensions. The vertical dimension corresponds to time. Therefore, 3-D solid represents the history of the computation performed by the chip.

Memory bound on the chip area:

- There are many computations which are memory bound, due to the need to process large data sets. To implement this type of computation on silicon, one is limited by how densely information (bit cells) can be placed on the chip.
- The amount of information processed by the chi can be visualized as information flow upward across the chip area. Each bit can flow through a unit area of the horizontal chip slice. Thus, the chip area bounds the amount of memory bits stored on the chip

I/O bound on volume AT:

- The volume of the rectangular cube is represented by the product AT. As the information flows through the chip for a period of time T, the number of input bits cannot exceed the volume AT.
- The area A corresponds to data in to and out of the entire surface of the silicon chip. The areal measure sets the maximum I/O limit.
- The height T of the volume can be visualized as a number of snapshots on the chips as computing time elapses. The volume represents the amount of information flowing through the chip during the entire course of the computation.

Bisection communication bound ((√AT)):

- A communication limited lower bound on the bisection area ((√AT)). The bisection is represented by the vertical slice cutting across the shorter dimension of the chip area.
- The distance of this dimension is √(A) for a square chip. The height of the cross section is T. The bisection area represents the maximum amount of information exchange between the two halved of the chip circuit during the time period.
- The cross section area √(AT) limits the communication bandwidth of a computation.

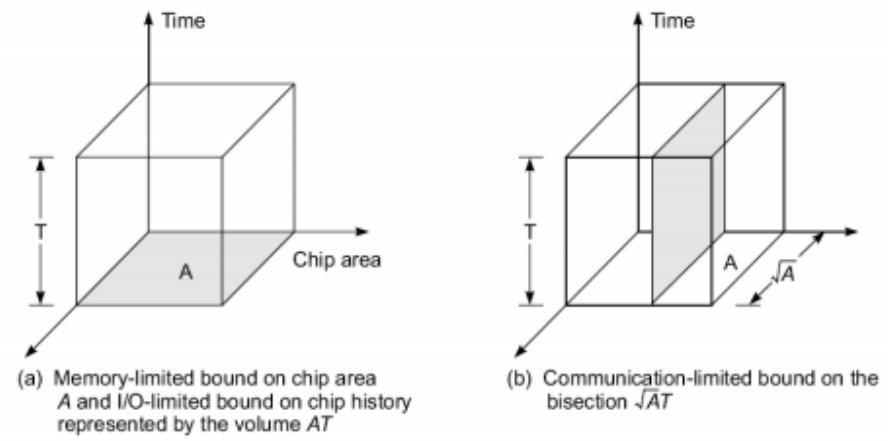


Fig. 1.15 The AT^2 complexity model of two-dimensional VLSI chips

10. With the example explain scheduling of the fine-grain and coarse-grain programs?

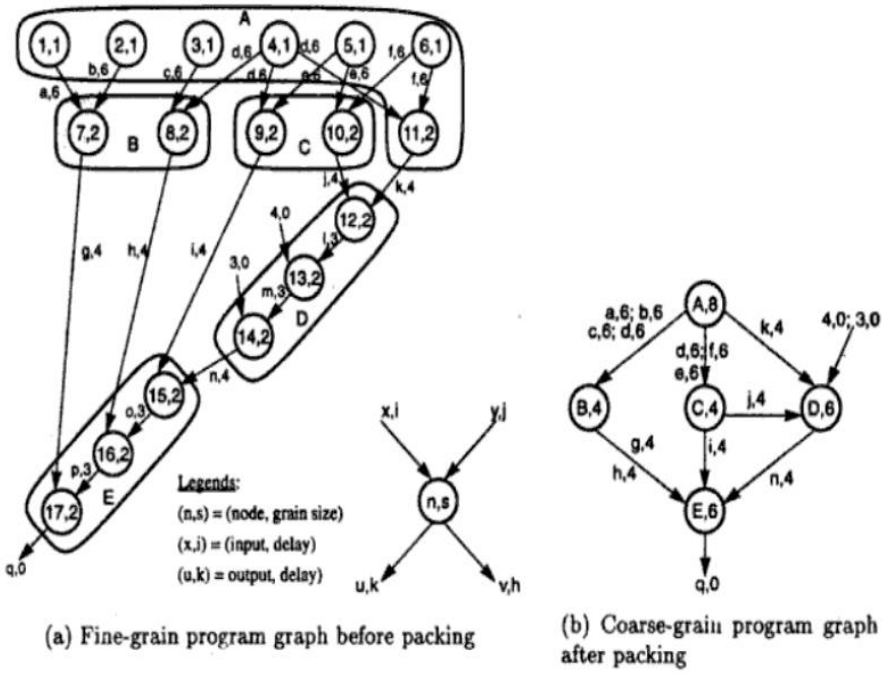
- The grain size problem demands determination of both the number and the size of grains / microtasks in a parallel program.
- There exist tradeoffs between parallelism and scheduling/ synchronization overhead.
- The time complexity involves both computation and communication overhead.
- Program partitioning involves

– Algorithm designer
– Programmer
– Compiler
– OS support

Example: Program graph before and after grain packing

- Program graph shows the structure of a program.
- Each node in the program graph corresponds to a computational unit in the program.
- The grain size is measured by the number of basic machine cycles (including both processor and memory cycles) needed to execute all the operations within the node. N –node size (n,s) and S-grain size of the node.
- The grain size reflects the number of computations involved in a program segment.
- Fine grain nodes have a smaller grain size. Coarse grain nodes have larger grain size.

- Edge label (v,d) between 2 nodes specifies the output variable v from the source node or the input variable to the destination node and the communication delay d between them.
- This delay includes all the path delays and memory latency involved.



(a) Fine-grain program graph before packing

(b) Coarse-grain program graph after packing

Figure 2.6 A program graph before and after grain packing in Example 2.4. (Modified from Kruatrachue and Lewis, *IEEE Software*, Jan. 1988)

There are 17 nodes in the fine grain program graph (fig 2.6a) and 5 in the coarse grain program graph (fig 2.6b). The coarse grain node is obtained by combining grouping multiple fine grain nodes.

Begin

1.	$a := 1$	10.	$j := e \times f$
2.	$b := 2$	11.	$k := d \times f$
3.	$c := 3$	12.	$l := j \times k$
4.	$d := 4$	13.	$m := 4 \times l$
5.	$e := 5$	14.	$n := 3 \times m$
6.	$f := 6$	15.	$o := n \times i$
7.	$g := a \times b$	16.	$p := o \times h$
8.	$h := c \times d$	17.	$q := p \times q$
9.	$i := d \times e$		

End

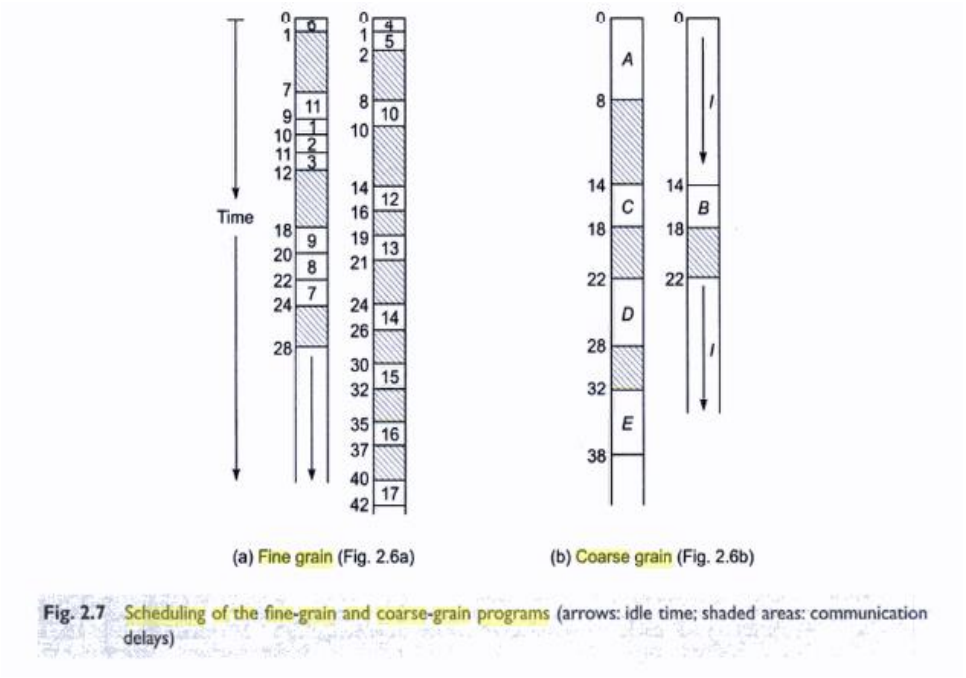
- Nodes 1,2,3,4,5,6- memory reference /data fetch operations.
- Each takes one cycle to address and 6 cycles to fetch from memory.

- All remaining (7 to 17) are CPU operations, each requiring 2 cycles to complete
- After packing, the coarse grain nodes have larger grain sizes.
- (A,8) –combining nodes(1,1) (2,1)(3,1)(4,1)(5,1)(6,1)(11,2)
- Grain size-8

A=1+1+1=1+1+1+1+1+2=8

Grain packing:

- Applying fine grain to achieve higher degree of parallelism.
- Combinating / packing – (fine grain nodes +coarse grain node)- eliminate unnecessary communication delays / reduce overall scheduling overhead.
- Fine grain operations within a single coarse grain node are assigned to the same processor for execution.
- The fine grain schedule is longer (42 time units) because more communication delays (shaded area). The coarse grain schedule is shorter (38 time unit) because communication delays among nodes 12, 13 and 14 within the same node D are eliminated after grain pulling.



(a) Fine grain (Fig. 2.6a) (b) Coarse grain (Fig. 2.6b)

Fig. 2.7 Scheduling of the fine-grain and coarse-grain programs (arrows: idle time; shaded areas: communication delays)

11. Compare control-driven, data-driven and demand-driven:

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	

12. With the neat diagram explain the levels of parallelism in program execution on modern computers?

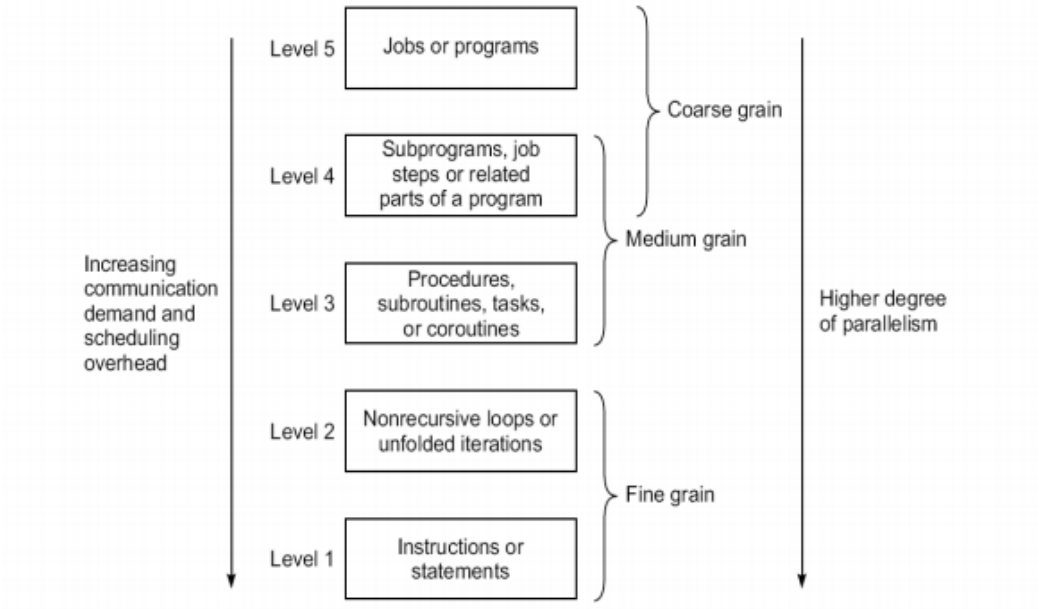


Fig. 2.5 Levels of parallelism in program execution on modern computers (Reprinted from Hwang, *Proc. IEEE*, October 1987)

Instruction Level Parallelism:

- This fine-grained, or smallest granularity level typically involves less than 20 instructions per grain.
- The number of candidates for parallel execution varies from 2 to thousands, with about five instructions or statements (on the average) being the average level of parallelism.
- Advantages: There are usually many candidates for parallel execution. Compilers can usually do a reasonable job of finding this parallelism.

Loop-level Parallelism:

- Typical loop has less than 500 instructions. If a loop operation is independent between iterations, it can be handled by a pipeline or by a SIMD machine.
- Most optimized program construct to execute on a parallel or vector machine. Recursive loops are difficult to handle. Vector processing is exploited at the loop level by a vectorizing compiler. It is also considered a fine grain of computation

Procedure-level Parallelism:

- The level corresponds to medium grain parallelism at the task, procedural, subroutine and co routine level.
- Detection of parallelism is difficult than finer grain level. Multitasking belongs to this category.
- Efforts by programmer may be needed to restructure a program at this level.
- Complier assistance is also needed.

Subprogram-level Parallelism:

- Corresponds to the level of job steps and related subprograms.
- Multiprogramming on a uniprocessor or on a multiprocessor is conducted at this level.
- Parallelism is exploited by algorithm designers or programmer rather than by compilers.

Job or Program-Level Parallelism:

- Corresponds to the execution of essentially independent jobs on a parallel computer.
- Job level parallelism is handled by the program loader and by the OS.
- Time sharing /space sharing multiprocessors explore this level of parallelism.
- Fine grain parallelism.

– Instruction/ loop level (assisted by parallelizing / vectroizing complier)

- Medium grain parallelism

– Task /job step (significant roles for the programmer is compiler)

- Coarse grain parallelism

– At the program level relies on effective OS.

– Efficiency of the algorithm.

13. Define the following terms for various system interconnect architectures

a) Node degree and network diameter **b) Bisection Width**

c) Hypercube routing functions **d) Broadcast and multicast**

d) Network performance

Node degree and network diameter:

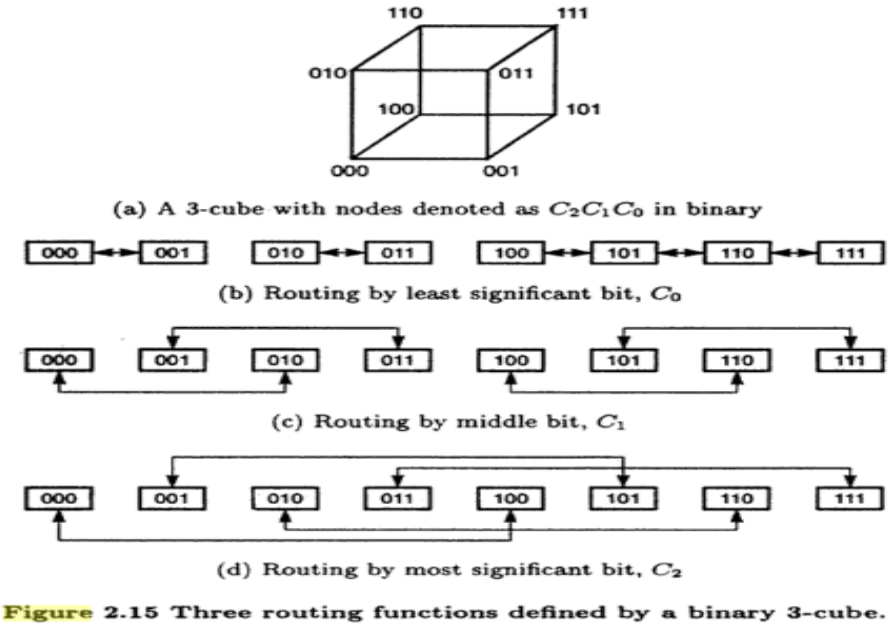
- Node degree d- number of edges / links/channels incident on a node.
- In case of unidirectional channels
  - Indegree- number of channels in to a node
  - Outdegree-number of channels out of a node
  - Node degree- sum of 2.
- Node degree reflects the number of I/O ports required per node[cost]. Node degree must be kept small to reduce cost. Constant node degree helps to achieve modularity for scalable system. Diameter d- of a network is the maximum shortest oath between any 2 nodes.
  - Path length: number of links traversed.
  - Network diameter- maximum number of distinct hops between any 2 nodes. Network diameter should be small.

Bisection Width:

- When a given network is cut into two equal halves, the minimum number of edges (channels) along the cut is called the bisection width b. In the case of a communication network, each edge may correspond to a channel with w bit wires. The wire bisection width is B = bw. This parameter B reflects the wiring density of the network. When the B is fixed, the channel width (in bits) w = B/b.

Hypercube routing functions:

- It's a 3-D binary cube network. 3-d routing functions are defined by 3 bits in the node address.
- One can exchange the data between adjacent nodes which differ in the least significant bit C0.
- 2 other routing patterns can be obtained by checking the middle bit C1 and most significant bit C2.
- These data exchange functions can be used in routing messages in a hypercube multicomputer



(a) A 3-cube with nodes denoted as $C_2C_1C_0$ in binary

(b) Routing by least significant bit, C_0

(c) Routing by middle bit, C_1

(d) Routing by most significant bit, C_2

Figure 2.15 Three routing functions defined by a binary 3-cube.

Broadcast and multicast:

- Broadcast
  - One to all mapping.
  - Easily achieved in an SIMD computer using a broadcast bus extending from the array controller to all PE's.
  - Message passing multicomputer has mechanisms to broadcast messages.
- Multicast
  - Mapping from one PE to other PEs (one to many).
  - Has to be implemented with matching of destination codes in the network.

Network performance:

i) Functionality: how network supports

- Data routing
- Interrupt handling
- Synchronization
- Request/message passing combining
- Cohernce/quality

ii) Network latency: the worst case time delay for a unit message to be transferred through the network.

iii) Bandwidth: maximum data transfer rate in terms of Mbytes or Gbytes transmitted through the network.

iv) Hardware complexity: Implementation cost such as for wires, switches , connectors, arbitration, interface logic

v) Scalability: The ability of a network to be modularly expandable with a scalable performance with increasing machine resources.

14. Explain with a neat diagram, Multistage Networks?

Multistage Networks:

- Crossbars have excellent performance scalability but poor cost scalability. Buses have excellent cost scalability, but poor performance scalability.
- A multistage interconnection network strikes a compromise between theses extremes. A multistage network connects a number of processors to a number of memory banks, via a number of switches organized in layers.
- In general, any multistage network is comprised of a collection of a x b switch modules and fixed network modules. The a x b switch modules are used to provide variable permutation or other reordering of the inputs, which are then further reordered by the fixed network modules.
- A multistage network connects a number of processors to a number of memory banks, via a number of switches organized in layers.
- Multistage connection network are designed with the use of small elementary crossbar switches connected in multiple layers.
- The elementary crossbar switches can implement 4 types of connections: straight, crossed upper broadcast and lower broadcast. All elementary switches are controlled simultaneously.
- The network like this is an alternative for crossbar switches if we have to switch a large number of connections, over 100. The extension cost for such a network is relatively low.