- Describe I/O subsystem architecture
- Discuss the architecture of I/O bus
- Identify PCI bus
- Explain microchannel architecture

2.1 INTRODUCTION

An implementation technique by which the execution of multiple instructions can be overlapped is called pipelining. This pipeline technique splits up the sequential process of an instruction cycle into sub-processes that operates concurrently in separate segments. The main advantage of pipelining is that it increases the instruction throughput, which is defined as the number of instructions completed per unit time. Thus, a program runs faster. In this lesson you will able to learn Microprocessor without Interlocked Pipeline Stages (MIPS), SPARC Micro Channel Architecture, I/O Subsystem Architecture, I/O Bus, PCI Bus.

2.2 PIPELINE ARCHITECTURE

In pipelining, several computations can run in distinct segments at the same time. A register is associated with each segment in the pipeline to provide isolation between each segment. Thus, each segment can operate on distinct data simultaneously. Pipelining is also called virtual parallelism as it provides an essence of parallelism only at the instruction level.

In pipelining, the CPU executes each instruction in a series of following small common steps:

1. Instruction Fetching
2. Instruction Decoding
3. Operand Address Calculation and Loading
4. Instruction Execution
5. Storing the result of the execution
6. Write back

The CPU while executing a sequence of instructions can pipeline these common steps. However, in a non-pipelined CPU, instructions are executed in strict sequence following the steps mentioned above.

To understand pipelining, let us discuss how an instruction flows through the data path in a five-segment pipeline. Consider a pipeline with five processing units, where each unit is assumed to take 1 cycle to finish its execution as described in the following steps:

(a) ***Instruction Fetch Cycle:*** In the first step, the address of the instruction to be fetched from memory into Instruction Register (IR) is stored in PC register.

(b) ***Instruction Decode Fetch Cycle:*** The instruction thus fetched is decoded and register is read into two temporary registers. Decoding and reading of registers is done in parallel.

(c) ***Effective Address Calculation Cycle:*** In this cycle, the addresses of the operands are being calculated and the effective addresses thus calculated are placed into ALU output register.

(d) ***Memory Access Completion Cycle:*** In this cycle, the address of the operand calculated during the prior cycle is used to access memory. In case of load and store instructions, either data returns from memory and is placed in the Load Memory Data (LMD) register or is written into memory. In case of branch instruction, the PC is replaced with the branch destination address in the ALU output register.

(e) ***Instruction Execution Cycle:*** In the last cycle, the result is written into the register file.

These steps of five-segment pipeline are shown in Figure 2.1

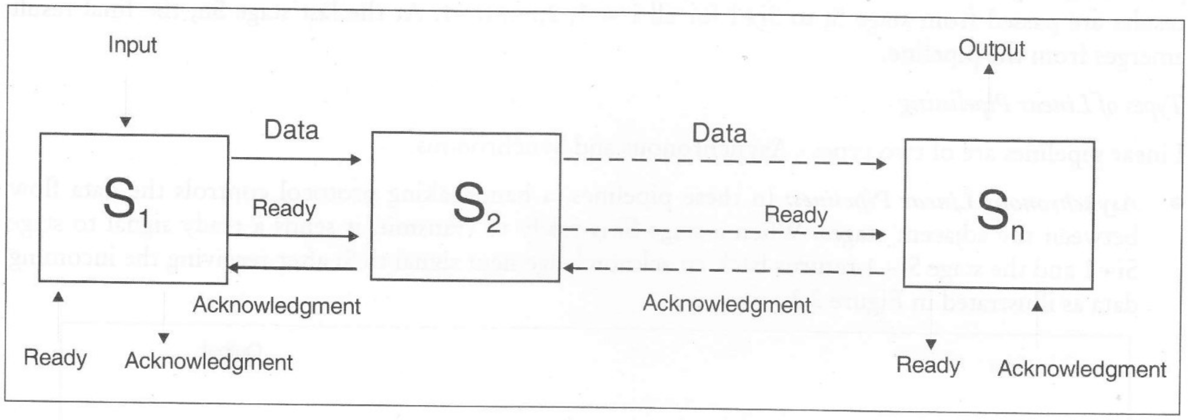


Figure 2.1: A Five-segment Pipeline

2.2.1 Types of Pipelining

Pipelines are of two types – Linear and Non-linear.

(a) ***Linear Pipelines:*** These pipelines perform only one pre-defined fixed functions at specific times in a forward direction from one stage to next stage. A linear pipeline can be visualised as a collection of processing segments, where each segment completes a part of an instruction. The result obtained from the processing in each segment is transferred to the next segment in the pipeline. As in these pipelines, repeated evaluations of the same function are performed with different data for some specified period of time, these pipelines are also called static pipelines.

(b) ***Non-linear pipelines:*** These pipelines can perform more than one operation at a time as they have the provision to be reconfigured to execute variable functions at different times. As these pipelines can execute different functions at different times, they are called dynamic pipelines. An example of a non-linear pipeline is a three-stage pipeline that performs subtraction and multiplication on different data at the same time as illustrated in Figure 2.2.

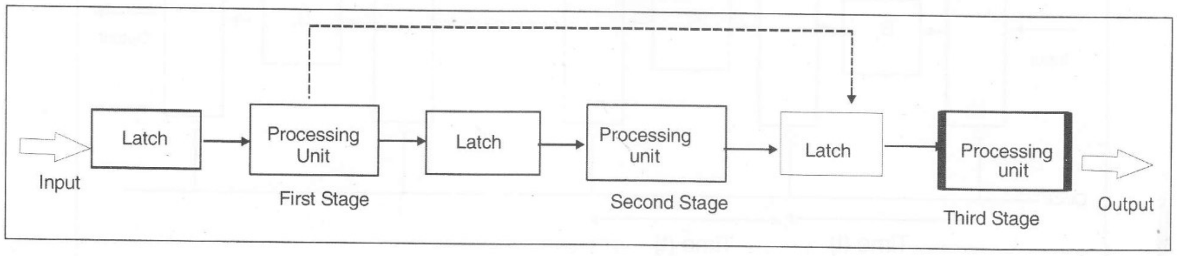


Figure 2.2: An Example of a Non-linear Pipeline

In this three-stage pipeline, the input data must go through stages 1, 2 and 3 to perform multiplication and through stages 1 and 3 only to perform subtraction. Therefore, dynamic pipelines require feed forward and feedback connections in addition to the streamline connections between the stages.

Linear Pipeline

In linear pipelining, only one pre-defined fixed functions is performed at specific times in a forward direction from one stage to next stage. So, a processor having linear pipelining is built with n processing stages. At the first stage S1, external inputs are fed into the pipeline and the processed results are passed from stage Si to Si+1 for all i = 1, 2,...., n –1. At the last stage Sn, the final result emerges from the pipeline.

Types of Linear Pipelining

Linear pipelines are of two types - Asynchronous and Synchronous.

- ***Asynchronous Linear Pipelines:*** In these pipelines, a handshaking protocol controls the data flow between the adjacent stages. When a stage Si is ready to transmit, it sends a ready signal to stage Si+1 and the stage Si+1 returns back an acknowledgement signal to Si after receiving the incoming data as illustrated in Figure 2.3.

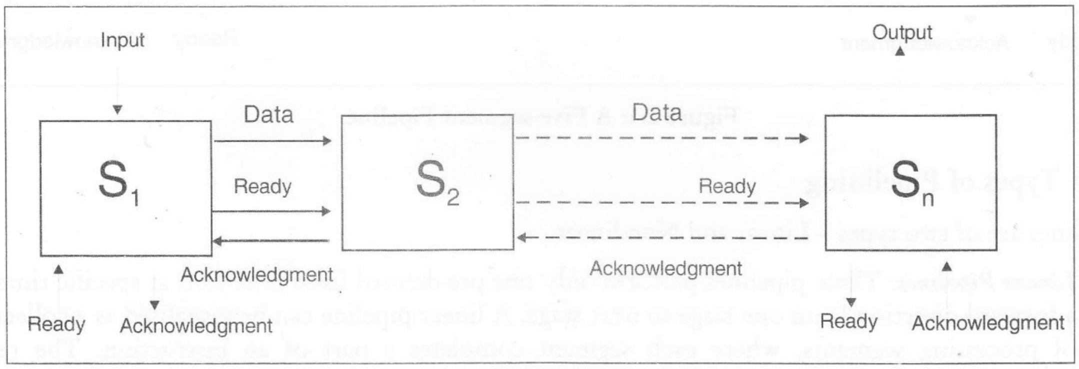


Figure 2.3: A Model of Asynchronous Linear Pipeline

- ***Synchronous Linear Pipelines:*** In these pipelines, clocked latches made with master-slave flip-flops are used to interface between stages. These latches can isolate inputs from outputs and control the timing signals. So, in synchronous linear pipelines, delays between stages are made almost equal in all stages as illustrated in Figure 2.4.

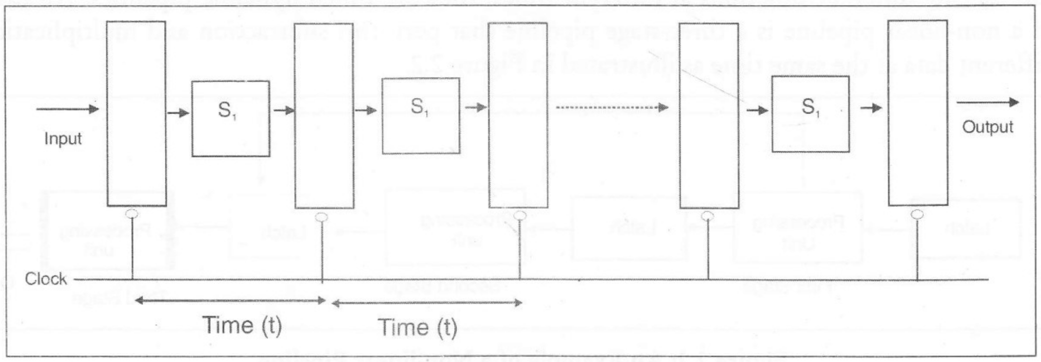


Figure 2.4: A Model of Synchronous Linear Pipeline

Non-linear Pipeline

In non-linear pipelining (dynamic pipelining), different functions can be executed by processors at different times as such pipelines have feed forward feedback connections along with the usual streamline connections. The scheduling of successive sub-functions in a non-linear pipeline becomes a non-trivial task due to the presence of these additional connections. The output from a dynamic pipeline is not necessarily comes out from the last stage of the pipeline as is generally happened in a linear pipeline.

2.3 MIPS SERIES

MIPS(Microprocessor without Interlocked Pipeline Stages), is a RISC microprocessor architecture originated by MIPS Computer Systems Inc MIPS designs are used in SGI's computer product line, and have found broad application in embedded systems, Windows CE devices, and Cisco routers. The Nintendo 64 video game and Sony PlayStation2 consoles use MIPS processors. By the late 1990s it was estimated that one in three of all RISC chips produced were MIPS-based designs.

The oldest of these, MIPS 32/64 defines a control register set as well as the instruction set. Several "add-on" extensions are also available, including MIPS-3D which is a simple set of integer-based SIMD instruction set dedicated to common 3D tasks, MDMX which is a more extensive floating-point-based SIMD instruction set, MIPS16 which adds compression to the instruction stream to make programs take up less room, and the recent addition of MIPS MT, new multithreading additions to the system similar to Hyper Threading in the latest Intel lineup.

The sooner MIPS architectures were 32-bit implementations (generally 32 bit wide registers and data paths), later versions were 64-bit implementations. Five backward-compatible revisions of the MIPS instruction set exist, named MIPS I, MIPS II, MIPS III, MIPS IV and MIPS 32/64. Because the designers created such a clean instruction set (see Instructions), computer architecture courses in universities and technical schools often study the MIPS architecture. The design of the MIPS CPU family, together with SPARC, another early RISC architecture, greatly influenced later RISC designs like HP Precision Architecture and DEC Alpha.

2.4 MOTOROLA 88000

The 88000 was Motorola's attempt at a home-grown RISC (now often referred to as a *load-store*) design, started in the 1980s. Originally called the 78000 as homage to their famed 68000 series, the design went though a tortured development path (including the name change) before finally emerging in 1988. This was some two years after its competition in the form of the SPARC and MIPS, and the 88000 never managed to catch on.

Like the 68000 before it, the 88000 was considered to be a very "clean" design. It was a pure 32-bit system, using a true Harvard architecture with completely separate data and addresses busses (and caches), had a small but powerful command set, and—like all Motorola CPUs—did not use memory segmentation.

The first implementation of the 88000 design was in the 88100 CPU, which included an integrated FPU. Mated to this was the 88200 MMU and cache controller. Building a system out of this 1st generation 88000 required both chips and considerable wiring between them, driving up costs. This is likely another major reason for the 88000's limited success.

This was later addressed in the **88110**, which combined the two chips into a single package. An additional modification made at the behest of MITs *T project which resulted in the **88110MP**, including on-chip communications for use in multi-processor systems.

In the late 1980s several companies were actively watching the 88000 for future use, including NeXT and Apple Computer, but both gave up by the time the 88110 was available in 1990. The only widespread use would be in the Data General AViiON series, but they were one of the only remaining customers in 1995 when they went all-Intel. The only other known use was in the Japanese 4-processor **OMRON luna88k** machines, which were used for a short time on the Mach kernel project at Carnegie Mellon University.

There was also an attempt to popularize the system with the 88open group, similar to what Sun Microsystems was attempting with their SPARC design. It appears to have failed in any practical sense. In the early 1990s Motorola joined the AIM effort to create a new RISC design based on the IBM POWER design. They worked a few features of the 88000 into the new PowerPC design to offer their customer base some sort of upgrade path. At that point the 88000 was dumped as soon as possible.

2.5 SPARC MICRO CHANNEL ARCHITECTURE

SPARC (Scalable Processor Architecture) is defined as the ability to increase the amount of processing that can be done by adding more resources to a system. It differs from performance as it does not increase performance but rather maintains performance by providing higher throughput. In other words, performance refers to the system response time under a typical load whereas scalability refers to the ability of a system to increase that load without degrading response time.

Scalable Processor architecture is the ability of a computer system to run more than one processor. Almost all business applications are scalable. There are several ways to achieve scalability, such as using more powerful CPUs or adding additional CPUs. There are two different modes of scaling: Scale up and scale out. Scaling up is accomplished by adding additional resources to a single machine to allow an application to service more requests. The most common ways to do this are by adding memory (RAM) or to use a faster CPU. Scaling out is accomplished by adding servers to a server group to make applications scale by spreading the processing among multiple computers. Each scaling method requires an understanding of the bottlenecks and the applications before a particular method can be successfully utilized.

2.5.1 Scalability through Windows Server 2003

The hardest part of building the scalable architecture is to pick a scalable operating system. The Windows Server 2003 family includes a range of solutions that allows computer professionals to right-size their server operating system for their particular application requirements. There are different editions of the Windows Server 2003. The Standard Edition is designed for everyday business needs such as file and printer sharing secure Internet connectivity, centralized desktop application deployment, hosting Web and .NET Remoting applications, and rich collaborative applications. The Enterprise Edition is the platform of choice for applications, Web services, and infrastructure. It runs everything that the Windows Server 2003 Web Edition and Standard Edition do. It also scales up to 8 CPUs each with up to 32 GB RAM to deliver high reliability, performance, and superior business value. The Datacenter Edition is used for the business-critical and mission-critical applications that demand the highest levels of scalability and availability. It is the most powerful and functional server operating system Microsoft has ever offered. It supports up to 32-way Symmetric Multiprocessing (SMP) and provides both eight-node clustering and load balancing services as standard features.

2.5.2 Superscalar Architecture

Superscalar architecture is a strategy in designing a high-performance CPU by enabling it to fetch and execute several instructions at the same time. It is one of the techniques of Instruction-Level Parallelism (ILP). The other techniques of ILP are pipeline and super pipeline, which we will introduce in the later part of the guide.

The processor speed can be increased by the superscalar design. In this design, the instruction pipelines can be multiplied in the design of processor by using multiple functional units. Various instruction processing circuits can execute several instructions in the same processing phase at the same time. Both pipelining and superscalar design are instances of instruction-level parallelism. So, the result produced by them must be the same as those produced by serial execution of program instruction.

Superscalar Design

In a superscalar design, each of the scalar base pipeline stages is replicated so that two or more instructions at the same stage of pipeline can be processed simultaneously. In this design, multiple operations can concurrently run on separate multiple execution units, where each unit is usually pipelined. So, a superscalar processor contains one or more instruction pipelines that share a set of functional units (such as an integer add unit,functional units execute multiple instructions per clock cycle, which is called Instruction-level parallelism. Several instructions are simultaneously issued or dispatched to these different functional units by a process called machine parallelism.

In a superscalar design, an adequate control mechanism is required to handle multiple instructions executed simultaneously. Such mechanism preserves the execution order of dependent instructions to ensure valid result and to avoid factors that cause pipeline stalling.

2.6 I/O SUBSYSTEM ARCHITECTURE

A computer must have a system to get information from the outside world.and must be able to the communicate results to the external world. Programs and data should be entered into computer memory for processing and results obtained from computations must be recorded or displayed for the user. The most familiar method of entering information into a computer is using a typewriter like keyboard that allows a person to enter alphanumeric information directly. Every time a key is depressed, the terminal sends a binary coded character to the computer. When input information is transferred to the processor via a keyboard, the processor will be idle most of the time while waiting for the information to arrive. To use a computer efficiently, a large amount of programs and data must be prepared in advance and transmitted into a storage medium. The information in the disk is then transferred into computer memory at a rapid rate. Results of programs are also transferred into a high-speed storage, which can be transferred later to output device for results.

Devices are said to be connected online that are under the direct control of the computer. These devices are designed to read information into or out of the memory unit when the CPU gives a command. Input or output devices connected to the computer are also called peripherals. Among the most common peripherals are keyboards, display units and printers. Peripherals that provide auxiliary storage for the system are magnetic disks.

Other input and output devices are digital incremental plotters, optical and magnetic character readers, analog-to-digital converters etc. Not all input comes from people, and not all output is intended for people. Computers are used to control various processes in real time, such as machine tooling, assembly line procedures, and chemical and industrial processes. For such applications, a method must be provided for sensing status conditions in the process and sending control signals to the process being controlled.

I/O Processing

Input-output interface gives a method for transferring information between internal memory and I/O devices. Peripherals connected to a computer require special communication links for interfacing them with the central processing unit. The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.

The major differences are:

1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore a conversion of signal values may be required
2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU.
3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
4. The operating modes of peripherals are different from each other.

The above differences can be resolved by establishing communication between various peripheral devices, memory.

2.7 ARCHITECTURE OF I/O BUS

A communication link between the processor and several peripherals is represented the following figure 2.5. The I/O bus is made of data lines, address lines and control lines. The magnetic disk, printer and terminal are used in any general-purpose computer. The magnetic tape is used in computers for backup storage. Each peripheral device associated with it by interface unit: Each interface decodes the address and control received from the I/O bus, interprets them for the peripheral and provides signals for the peripheral controller. It also synchronizes the data flow and supervises the transfer between peripheral and processor. Each peripheral has its own controller that operates the particular electromechanical device. For example, the printer controller controls the paper motion, the print timing and the selection of printing characters. A controller may be housed separately or may be physically integrated with the peripheral. The I/O bus from the processor is attached to all peripheral interfaces. To communicate with a particular device, the processor places a device address on the address lines. Each interface attached to the I/O bus contains an address decoder that monitors the address lines. When the interface detects its own address, it activates the path between the bus lines and the device that it controls. All peripherals whose address does not correspond to the address in the bus are disabled by their interface.

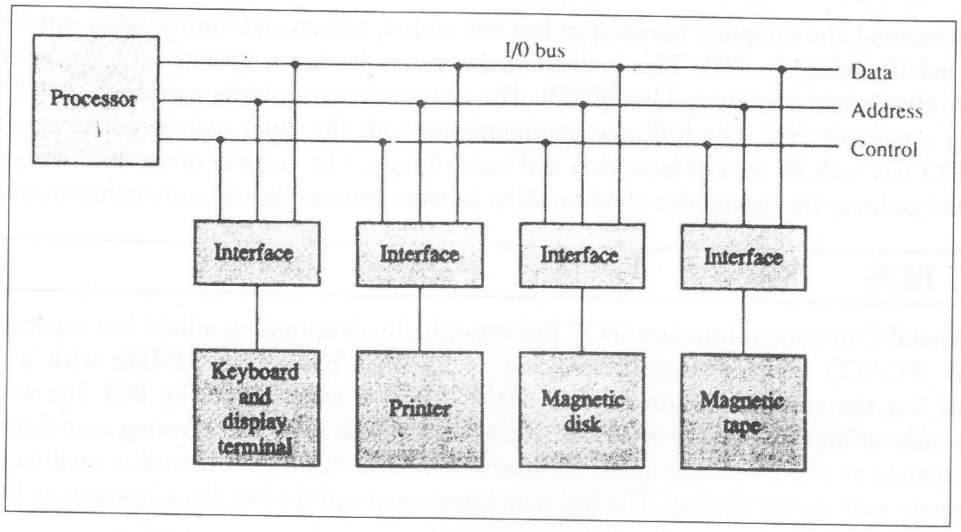


Figure 2.5: Connection of I/O Bus to Input-Output Devices

The address is made available in the address lines and the processor provides a function code in the control lines. The interface selected responds to the function code and proceeds to execute it. The function code is referred to as an I/O command and is in essence an instruction that is executed in the interface and is attached in the peripheral unit. The interpretation of the command depends on the peripheral that the processor is addressing. There are four types of commands that an interface may receive. They are classified as control, status, data output and data input.

We issue a control command to activate the peripheral. The particular control command issued depends on the peripheral. Each peripheral receives its own distinguished sequence of control commands, depending on its mode of operation.

We use a status command to test various status conditions in the interface and the peripheral. For example, the computer may wish to check the status of the peripheral before a transfer is initiated. During the transfer, one or more errors may occur which are detected by the interface.

A data output command causes the interface to respond by transferring data from the bus into one of its registers. Consider an example with a tape unit. The computer starts the tape moving by issuing a control command. Now the processor monitors the status of the tape by means of a status command.

By giving the data input command the interface receives an item of data from the peripheral and places it in its buffer register. The processor checks if data are available by means of a status command and then issues a data input command. The interface places the data on the data lines, and the processor accepts data.

I/O versus Memory Bus

To communicate with I/O, the processor must communicate with the memory unit. The memory bus contains data, address and read/write control lines. There are three ways to use computer buses to communicate with memory and I/O:

1. Use two separate buses, one for the memory and the other for I/O.
2. Use one common bus for both memory and I/O but have separate control lines for each.
3. Use one common bus for memory and I/O with common control lines.

In the first method, the computer has independent sets of data, address and control buses, one for accessing memory and the other for I/O. This is done in computers having a separate I/O Processor (IOP) in addition to the Central Processing Unit (CPU). The memory communicates with both the CPU and the IOP using a memory bus. The IOP also communicates with the input and output devices through a separate I/O bus with its own address, data and control lines. The purpose of the IOP is to provide an independent pathway for the transfer of information between external devices and internal memory.

2.8 PCI BUS

The Peripheral Component Interface 'PCI' Bus was initially developed as a local bus expansion for the ISA/EISA (PC/AT) bus. The first description of the PCI bus ran at 33MHz with a 32-bit bus (133MBps) but the current version runs at 66MHz with a 64-bit bus. The PCI bus works either synchronously or asynchronously with the motherboard clock rate. As operating asynchronously the bus will operate at any frequency up to the maximum (66MHz). Flow control is used to permit the bus to operate with slower devices. The bus is un-terminated and the bus clock operates at 133MHz.

PCI supports full device bus mastering, and offers bus arbitration facilities through the system chipset. PCI architecture allows bus mastering of multiple devices on the bus concurrently, with the arbitration circuitry working to ensure that no device on the bus (including the processor) locks out any other device. Though, in the event that no other device needs access to the bus, PCI will allow a bus master to transfer data at the maximum permissible rate. Note that, with some early motherboards it might be promising that not all of the available PCI bus slots will be capable of bus mastering. When in doubt it is wise to check with the motherboard manual.

The PCI standard forms part of the Plug and Play standard developed by Intel, Microsoft and a lot of other companies in which the PCI chipset handles the recognition of cards, working in conjunction with the BIOS and operating to automatically allocate resources for compatible peripheral cards.

The PCI bus uses its own internal interrupt system for dealing with requests from the cards on the bus. These interrupts are often called "#A", "#B", "#C" and "#D" to avoid puzzlement with the normal numbered system IRQs, though they are sometimes referred to by number.

PCI interrupts are mapped to the normal system interrupts (usually IRQ9 to IRQ12). This imposes a limit of four interrupts obtainable for PCI devices. Where more slots are provided (or where a USB controller is present) several PCI devices may be configured to share an IRQ.

2.9 MICRO CHANNEL ARCHITECTURE

The Micro Channel Architecture (MCA) bus (also called the *Micro Channel* bus; MCA stands for "Micro Channel Architecture") was IBM's effort to substitute the ISA bus with something "bigger and better". When the 80386DX was introduced in the mid-80s with its 32-bit data bus, IBM determined (much like it did with the AT) to generate a bus to match this width. MCA is 32 bits wide, and offers several important improvements over ISA. (One of MCA's disadvantages was rather poor DMA controller circuitry.)

The MCA bus has some pretty imposing features considering that it was introduced in 1987, a full seven years before the PCI bus made comparable features common on the PC. In some ways it was ahead of its time, because back then the ISA bus really wasn't a major performance limiting factor:

- ***32 Bit Bus Width:*** The MCA bus features a full 32 bit bus width, the same width as the VESA and PCI local buses. It had far better throughput to the ISA bus.
- ***Bus Mastering:*** The MCA bus hold bus mastering adapters for greater efficiency, including proper bus arbitration.
- ***Plug and Play:*** MCA routinely configured adapter cards, so there was no need to fiddle with jumpers. This was eight years before Windows 95 brought PnP into the mainstream!

MCA had a great deal of possibility. Unhappily, IBM made two decisions that would fate MCA to utter failure in the marketplace. First, they made MCA mismatched with ISA; this means ISA cards will not work at all in an MCA system, one of the few classes of PCs for which this is true. The PC market is *very* responsive to backwards-compatibility issues, as proof by the number of older standards that persist to this day (such as ISA!) Second, IBM decided to make the MCA bus proprietary. It in fact did this with ISA as well; though in 1981 IBM could afford to flex its muscles in this way, while by this time the clone makers were starting to come into their own and weren't interested in bending to IBM's wishes.

These two factors, joint with the increased cost of MCA systems, led to the end of the MCA bus. With the PS/2 now discontinued, MCA is dead on the PC platform, though it is still used by IBM on some of its RISC 6000 UNIX servers. It is one of the traditional examples in the field of computing of how non-technical issues often dominate over technical ones.

Check Your Progress

Fill in the blanks:

1. Pipelining is also called .... parallelism as it provides an essence of parallelism only at the instruction level.
2. In Asynchronous Linear Pipelines a .... protocol controls the data flow between the adjacent stages.
3. MIPS (*M*icroprocessor without *i*nterlocked *p*ipeline *s*tages), is a .... microprocessor architecture
4. Scalable Processor architecture is the ability of a computer system to run more than .... processor.
5. .... supports full device bus mastering, and provides bus arbitration facilities through the system chipset.

2.10 LET US SUM UP

The pipeline technique splits up the sequential process of an instruction cycle into sub-processes that operates concurrently in separate segments. Pipelines are of two types - Linear and Non-linear. Linear pipelines perform only one pre-defined fixed functions at specific times in a forward direction from one stage to next stage. These pipelines are also called static pipelines. Non-linear pipelines can perform more than one operation at a time as they have the provision to be reconfigured to execute variable functions at different times. They are also called dynamic pipelines. MIPS(Microprocessor without Interlocked Pipeline Stages), is a RISC microprocessor architecture developed by MIPS Computer Systems Inc MIPS designs are used in SGI's computer product line, and have found broad application in embedded systems, Windows CE devices, and Cisco routers. SPARC (Scalable Processor

Architecture) is defined as the ability to increase the amount of processing that can be done by adding more resources to a system. A computer must have a system to get information from the outside world and must be able to the communicate results to the external world. The I/O bus is made of data lines, address lines and control lines. The magnetic disk, printer and terminal are used in any general-purpose computer. The MCA bus has some pretty impressive features considering that it was introduced in 1987, a full seven years before the PCI bus made similar features common on the PC.

2.11 KEYWORDS

MIPS: Microprocessor without Interlocked Pipeline Stages

SPARC: Scalable Processor Architecture

PCT: Peripheral Component Interface

MCA: Micro Channel Architecture

RICS: Reduced Instruction Set Computer

IOP: Input/Output Processor

SMP: Symmetric multiprocessing

2.12 QUESTIONS FOR DISCUSSION

1. Explain the architecture of Pipeline.
2. What is the difference between SPARC Micro Channel Architecture and Micro Channel Architecture?
3. Explain the architecture of I/O Subsystem and I/O Bus.
4. Differentiate between PCI Bus and MIPS Series.

Check Your Progress: Model Answers

1. Virtual
2. Handshaking
3. RISC
4. One
5. PCI

2.13 SUGGESTED READINGS

Sajjan G. Shiva; *Computer Design and Architecture*; Marcel Dekker

Silvia Melitta Mueller, Wolfgang J. Paul; *Computer Architecture*; Springer

Joseph D. Dumas II; *Computer Architecture*; CRC Press

Nicholas P. Carter; *Schaum's Outline of Computer Architecture*; Mc. Graw-Hill Professional

LESSON

3

DATA FLOW ARCHITECTURE

CONTENTS

3.0 AIMS AND OBJECTIVES

After studying this lesson, you will be able to:

- Explain data flow architecture
- Discuss parallel architecture for control driven machine
- Identify pipeline hazards
- Define the cross bar switched system
- Explain multiprocessor with single and multiple-stage interconnections network
- Discuss switch lattice architecture

3.1 INTRODUCTION

Dataflow architecture directly contrasts the traditional Von Neumann architecture or control flow architecture. Dataflow architectures do not have a program counter. The execution of instructions in dataflow systems is solely determined based on the availability of input arguments to the instructions. Although dataflow architecture has not been used in any commercially successful computer hardware, it is very relevant in many software architectures such as database engine designs and parallel computing frameworks. There are two types of dependencies in software - Data Dependency and Resource Dependency. In data dependency, one module creates or modifies data that is used by another module while in resource dependency, an individual client accesses a sequence of resources to perform a particular task. Let us discuss these.

Multiprocessor system interconnect is one of the most important components of a multiprocessor machine. In this lesson, we will discuss the various types of multiprocessor system interconnects viz. Bus Network, Star Network, Ring Network, Mesh Network, Hypercube Network, Tree Network, Hierarchical common bus systems, Crossbar networks, Multiport networks and Multistage networks.

When multiple processors with separate caches share a common memory, it is necessary to keep the caches in a state of coherence. In this lesson, we will also discuss the problems of cache coherence and the various mechanisms to solve those problems.

3.2 DATAFLOW ARCHITECTURE

In a traditional computer design, the processor executes instructions, which are stored in memory in particular sequences. In each processor, the instruction executions are in serial order and therefore are slow. There are four possible ways of executing instructions:

- *Control-flow Method:* In this mechanism, an instruction is executed when the previous one in a defined sequence has been executed. This is the traditional way.
- *Demand-driven Method:* In this mechanism, an instruction is executed when the results of the instruction are required by other instruction.
- *Pattern-driven Method:* In this mechanism, an instruction is executed when particular data patterns appear.
- *Dataflow Method:* In dataflow method, an instruction is executed when the operands required become available.

Dataflow architecture is a computer architecture that directly contrasts the traditional control flow architecture (von Neumann architecture). It does not have a program counter and the execution of instructions is solely determined based on the availability of input arguments to the instructions. The dataflow architecture is very relevant in many software architectures today including parallel computing frameworks. This architecture was proposed in the 1970s and early 1980s by Jack Dennis of MIT.

3.2.1 Dataflow Programming

Software written using dataflow architecture consists of a collection of independent components running in parallel that communicate via data channels.

In a dataflow model, a node is a computational component, and an arrow is a buffered data channel. A control algorithm is divided into nodes first. Each concurrently executing node is a self-contained software part with well-defined functionality. Data channels provide the sole mechanism by which nodes can interact and communicate with each other by ensuring lower coupling and greater reusability. Data channels can also be implemented transparently between processors to carry messages between components that are physically distributed. In the dataflow architecture, a control application is composed of function bodies and data channels, and the connections between function bodies and data channels are described in a dataflow graph. Consequently, designing a control application mainly involves constructing such a dataflow graph by selecting function bodies from the design library and connecting them together. Additional user-defined or application-specific function bodies are also easily supported. A model of dataflow programming is shown in Figure 3.1.

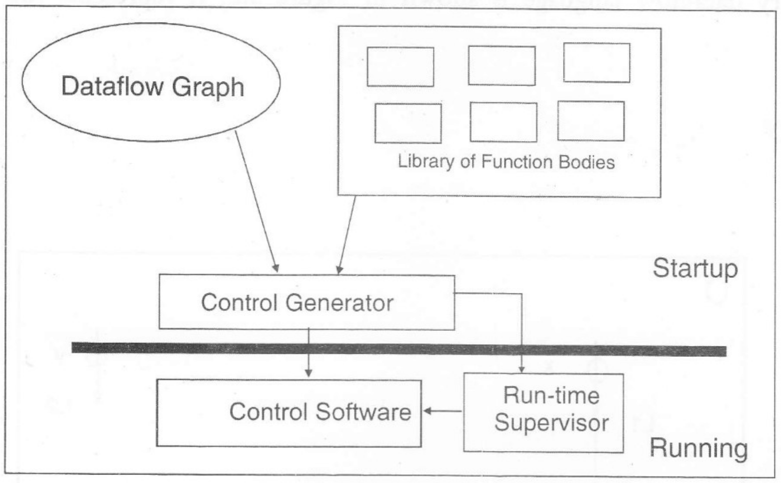


Figure 3.1: A Model of Dataflow Programming

3.2.2 Dataflow Graph

Data flow computational model uses directed graph to describe a computation. This graph is called dataflow graph or data dependency graph. This graph consists of nodes and edges (arcs). Nodes represent operations and edges represent data paths. Dataflow is a distributed model of computation as there is no single locus of control.

Dataflow graph is asynchronous as execution of a node starts when matching data is available at a node's input ports. In the original dataflow models, data tokens are consumed when the node executes. Some models were extended with "Sticky tokens", the tokens that stay much like a constant input and match with tokens arriving on other inputs. Nodes can have varying granularity, from instructions to functions.

Once a node is activated and the nodal operation is performed, this is called "Fired Results", which are passed along the arc to waiting node. This process is repeated until all of the nodes are fired and the final result is created. More than one node can also be fired simultaneously. Arithmetic operators and conditional operators act as nodes.

Example

To understand the dataflow graph, let us design an elementary processor to utilize the elementary data-flow language as its base language. A program in the elementary data-flow language is a directed graph in which the nodes are operators or links. Arcs along which values may travel connect these nodes. An operator of the schema is enabled when tokens are present on all input arcs. The enabled operator may fire at any time, removing the tokens on its input arcs, computing a value from the operands associated with the input tokens, and associating that value with a result token placed on its output arc. A result may be sent to more than one destination by means of a link which removes a token on its input arc and places tokens on its output arcs bearing copies of the input value. An operator or a link cannot fire unless there is no token present on any output arc of that operator or link. An example of a program in the elementary data-flow language is shown in Figure 3.2. It represents the following simple instructions:

input x, y

p := (x*(x+y))+y

q := (x+y)/p

output p,q

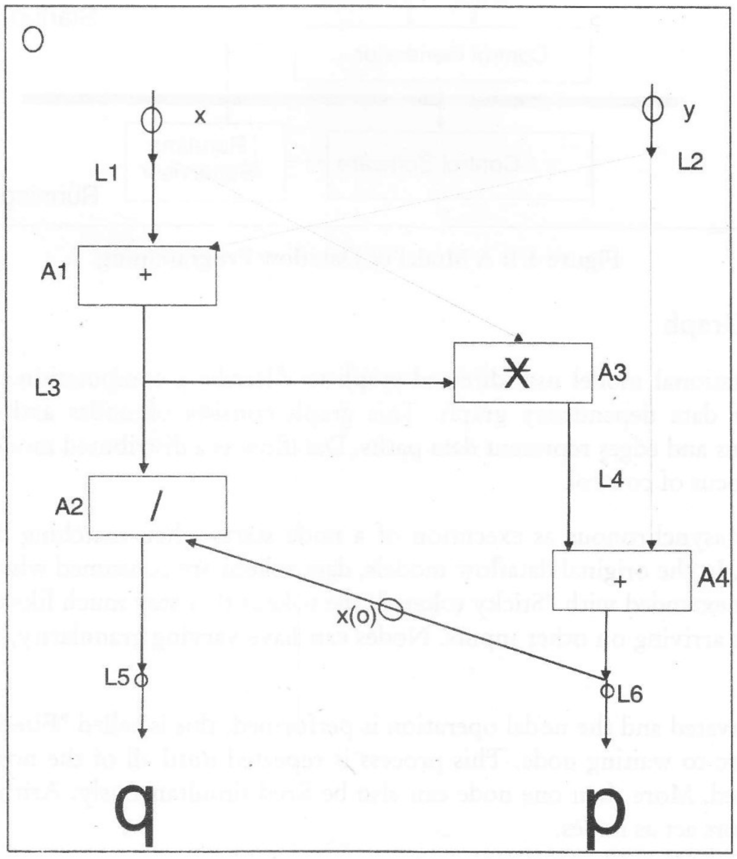


Figure 3.2: An Example of a Program in the Elementary Data-flow Language

In the dataflow graph shown in Figure 3.2, the rectangular boxes are operators. Each arithmetic operator is reflected in a corresponding operator in the program. The small dots are links while the large dots represent tokens holding values for the initial configuration of the program. The program runs in the following steps:

1. Links L1 and L2 are initially enabled.
2. The firing of LI makes copies of the value x available to operators A1 and A3; firing L2 presents the value y to operators AI and A4.
3. Once Li and L2 have fired, operator A1 is enabled since it will have a token on each of its input arcs.
4. After A1 has fired, link L3 will become enabled.
5. The firing of L3 will enable the concurrent firing of operators A2 and A3, and so on.

The various symbols used in a dataflow graph are shown in Figure 3.2.

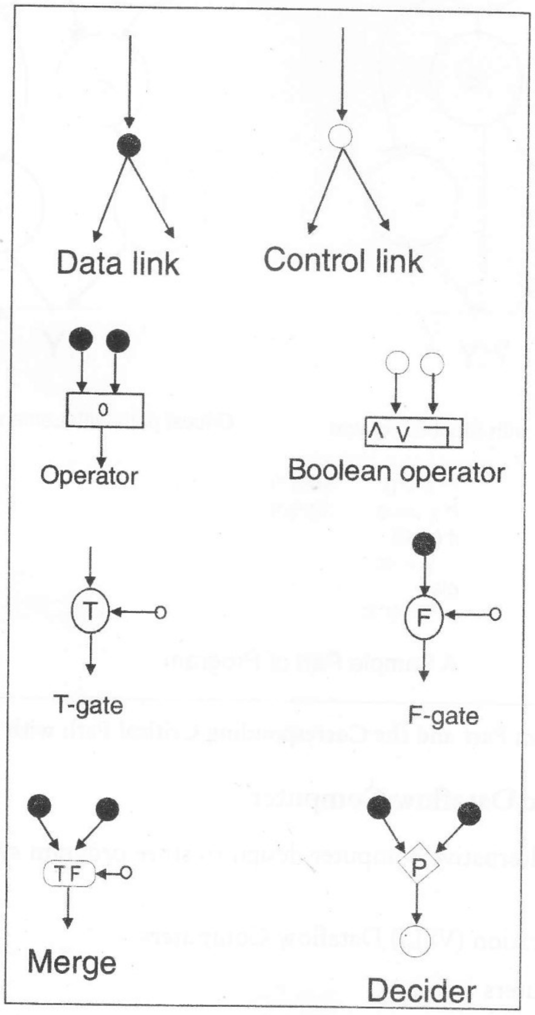
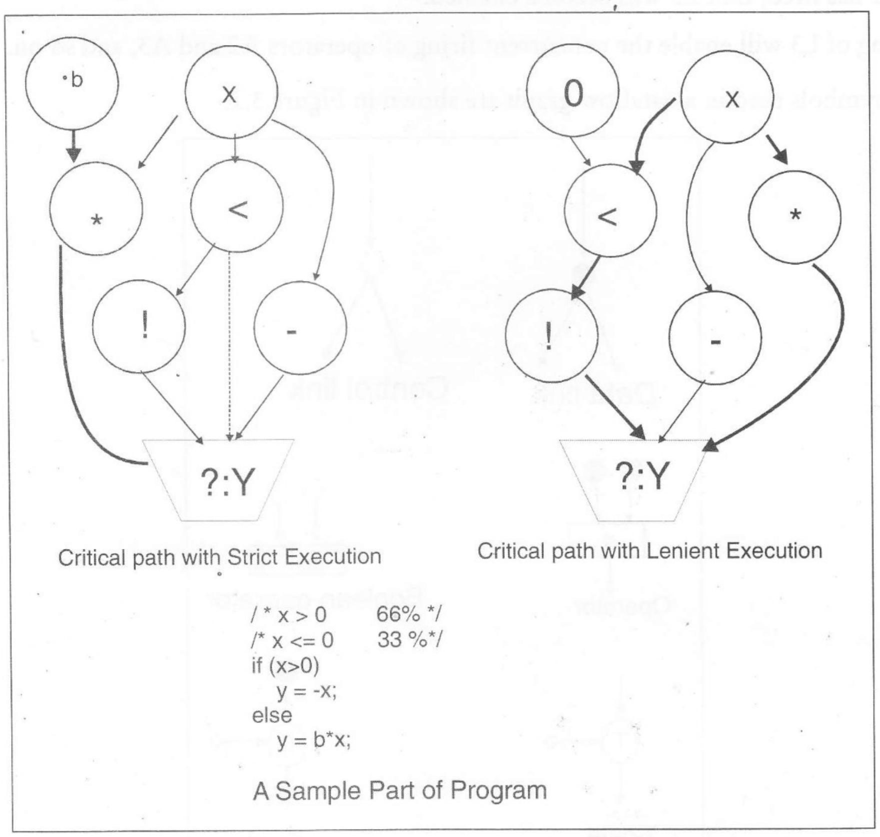


Figure 3.3: The Various Symbols used in a Dataflow Graph

Dynamic Critical Path

The dynamic critical path of dataflow graph is simultaneously a function of program dependences, runtime execution path, hardware resources and dynamic scheduling. All critical events must be last arrival events. Such an event is the last one, which enable data to be latched. Events correspond to signal transitions on the edges of the dataflow graphs. Most often, the last-arrival event is the last input to reach an operation. However, for lenient operations the last arrival event is the input that enables the computation of the output. In lenient execution, all forward branches are executed simultaneously. In a typical execution, multiple critical events may correspond to the same hardware structure. A sample program part and the corresponding critical path with strict and lenient execution are shown in Figure 3.4. In strict execution, the multiplier is on the critical path while in lenient execution, the multiplier is critical only when its result is used by latter computations.


b
x
*
<
!
-
?:Y
Critical path with Strict Execution
0
x
<
*
!
-
?:Y
Critical path with Lenient Execution
A Sample Part of Program

```
/ * x > 0       66% */
/* x <= 0       33 %*/
if (x>0)
    y = -x;
else
    y = b*x;
```

Figure 3.4: A Sample Program Part and the Corresponding Critical Path with Strict and Lenient Execution

3.2.3 Static and Dynamic Dataflow Computer

Data flow computer is an alternative computer design to store program systems. The various kinds of data flow systems are:

(a) Very Large Scale Integration (VSLI) Dataflow Computers

(b) Static Dataflow Computers

(c) Dynamic Dataflow Computers

VLSI Dataflow Computers

In Very Large Scale Integration (VSLI) Dataflow Computers, the dataflow technique is applied to very large scale integrated arrays of interconnected cells. A cell consists of a processor, a small stored program and I/O communication channel. Each cell performs primitive dataflow operations on data received via direct links from neighboring cells.

The array is connected to a host system, which downloads the programs to cell. A cell will be fired when all operands for one of its stored instruction are received. A typical operation may require one or two input operands from neighboring cells and produce one output operand to a neighboring cell.

Static Dataflow Computers

Designs that use conventional memory addresses as data dependency tags are called static dataflow computers. The static dataflow mechanism was the first one to receive attention for hardware realization at MIT. In static dataflow computers, nodes are fired when all input tokens are released and the previous output tokens have been consumed. Input tokens are then removed and new output tokens are generated. Static dataflow computers allow pipeline computations and loops but not code sharing and recursion. There is a handshaking acknowledgement mechanism, which can take the form of special control tokens set from processors once they respond to a fired node.

The processing elements receive the operation packets as the following form:

Opcode Operands	Destinations

The resulting packet has the following form:

Value	Destination

In static dataflow computers, numbers are never referred to by address in memory, as they are not stored in a globally accessible memory. The operands can only be affected by one selected node at a time. However, the complex data structures, or even simple arrays could not reasonably be carried in the instruction and hence cannot be handled in static dataflow mechanism. The static dataflow computers do not allow multiple instances of the same routines to be executed simultaneously because the simple tags could not differentiate between them.

Dynamic Dataflow Computers

Designs that use Content-addressable Memory (CAM) are called dynamic dataflow computers. They use tags in memory to facilitate parallelism. In dynamic dataflow architecture, the dataflow graph being executed is not fixed and can be altered through such actions as code sharing and recursion. Tags could be attached to the packets to identify tokens with particular computations. In dynamic dataflow, a node fires when all input tokens with the same tag appear. More than one token is allowed on each arc and previous output tokens need not be consumed before the node can be fired again.

The dynamic dataflow computers need storage for the unmatched tokens. The First-In-First-Out (FIFO) token queue for storing the tokens is not suitable. However, no acknowledgement mechanism is required in these systems.

3.3 PARALLEL ARCHITECTURE

Parallel architecture is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently ("in parallel"). As power consumption (and consequently heat generation) by computers has become a concern in recent years, parallel computing has become the dominant paradigm in computer architecture, mainly in the form of multicore processors. There are several different forms of parallel computing: bit-level, instruction level, data, and task parallelism. Parallelism has been employed for many years, mainly in high-performance computing, but interest in it has grown lately due to the physical constraints preventing frequency scaling.

Parallel computers can be classified according to the level at which the hardware supports parallelism—with multi-core and multi-processor computers having multiple processing elements within a single machine, while clusters, MPPs, and grids use multiple computers to work on the same task. Specialized parallel computer architectures are sometimes used alongside traditional processors, for accelerating specific tasks.

Parallel computer programs are not easy to write than sequential ones, because concurrency introduces several new classes of potential software bugs, of which race conditions are the most common. Communication and synchronization between the different subtasks are typically one of the greatest obstacles to getting good parallel program performance.

3.4 PIPELINE HAZARDS

Hazards are the situation that stops the next instruction in the instruction stream from being executing during its designated clock cycle. Hazards reduce the performance from the ideal speedup gained by pipelining.

There are three classes of hazards:

- ***Structural Hazards:*** They occur from resource conflicts when the hardware cannot support all possible combinations of instructions in simultaneous overlapped execution.
- ***Data Hazards:*** They occur when an instruction depends on the result of a previous instruction in a way that is exposed by the overlapping of instructions in the pipeline.
- ***Control Hazard:*** They occur from the pipelining of branches and other instructions that change the PC.

Hazards in pipelines can make it essential to stall the pipeline. The processor can stall on different events:

- A ***cache miss.*** A cache miss stalls all the instructions on pipeline both before and after the instruction causing the miss.
- A ***hazard in pipeline.*** Removing a hazard often requires that some instructions in the pipeline to be allowed to proceed while others are delayed. When the instruction is stalled, all the instructions issued *later* than the stalled instruction are also stalled. Instructions issued *earlier* than the stalled instruction must carry on, since otherwise the hazard will never clear.

3.5 MULTIPROCESSOR WITH SINGLE AND MULTI-STAGE INTER CONNECTION NETWORK

System interconnection network establishes communications between the computer resources for flowing the information either between any pair of modules within the system or from one module to other modules outside the system. In order to solve a problem, the processing elements have to cooperate and to exchange their computed data over the network. There can be two types of interconnection network:

1. Static Network
2. Dynamic Network

3.5.1 Static Network

The static networks are formed of point-to-point direct connections, which are fixed once built and do not change during program execution. They are used for fixed connections among subsystems of a centralised system. They can also be used for building distributed systems such as multiprocessors and multicomputers. The SIMD machines where the communication patterns are predefined or predictable also use static networks. The common topologies used in static networks are:

1. ***Bus Network:*** The bus network consists of a single communication channel to which each node is connected. Each node has a device that sends messages along the bus in either direction as shown in Figure 3.5. A given message contains data, error-checking code, the address of the node that sends the message, and the address of the node that receives the message. When a message passes through each node, that node checks the message for its address. If the node finds its address in a message, it reads the data along with error checking code and sends a message to the sender that data has been received.

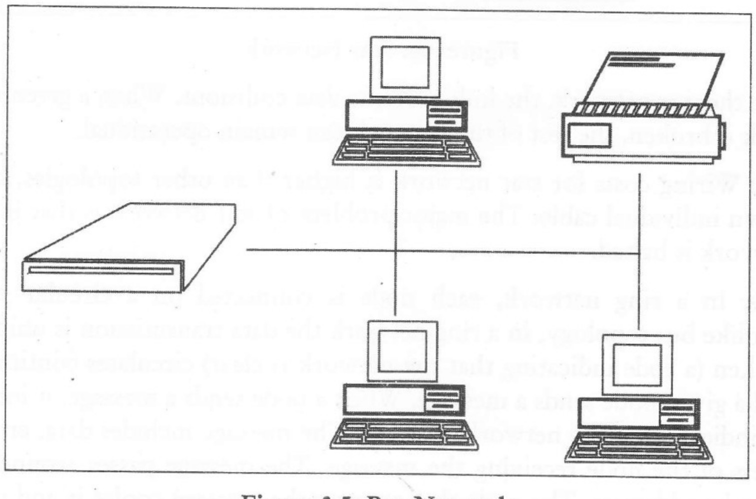


Figure 3.5: Bus Network

Advantages: The bus network is simple in layout and ease in connectivity. So, it is easier to find errors in locating cable faults in bus network. The bus network is ideal for one to many data transmissions.

Disadvantages: A problem occurs with a bus network when two or more nodes send messages at the same time. This creates an interference pattern, and when one of the node on the network detects this pattern, it stops all transmissions. Another problem with a bus network is that a broken connection along the bus brings the whole network down.

2. ***Star Network:*** Star network is an improvement over the bus topology. In a star network, each node is connected via its own path to a central hub which acts as a switching station as shown in Figure 3.6. The hub reads the addresses of messages sent by the nodes and routes the messages accordingly.

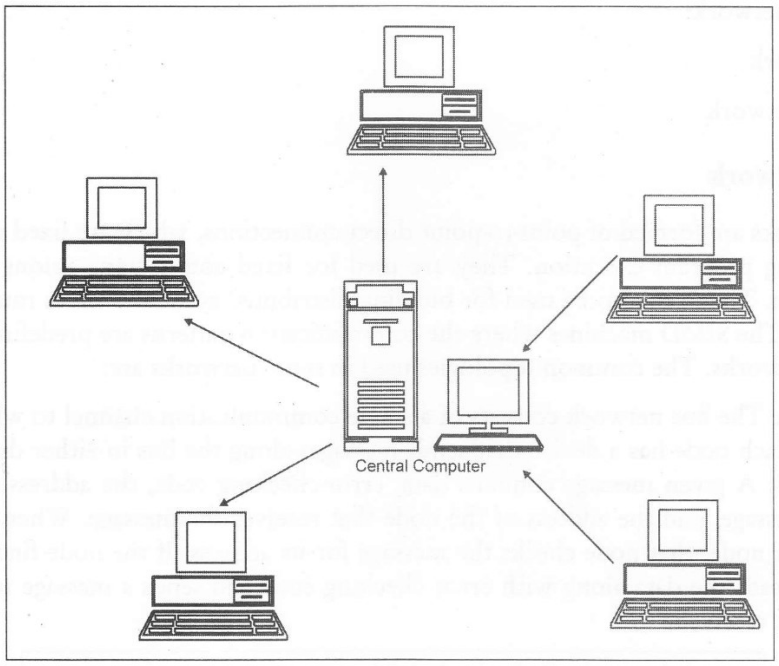


Figure 3.6: Star Network

Advantages: In the star network, the hub prevents data collisions. When a given node's connection to the network is broken, the rest of the network can remain operational.

Disadvantages: Wiring costs for star network is higher than other topologies, because each node requires its own individual cable. The major problem of star network is that in case the hub fails the entire network is halted.

3. ***Ring Network:*** In a ring network, each node is connected on a circular path as shown in Figure 3.7. Unlike bus topology, in a ring network the data transmission is unidirectional. In ring network, a token (a code indicating that the network is clear) circulates continuously around the network until a given node sends a message. When a node sends a message, it intercepts the token, changes it to indicate that the network is in use. The message includes data, error checking code, and the address of the node receiving the message. The message passes around the circular path until it reaches its addresses. The node that receives the message copies it and then passes it back along the path until it reaches the sender. The sender then removes the message from the network and changes the token back to its original state.

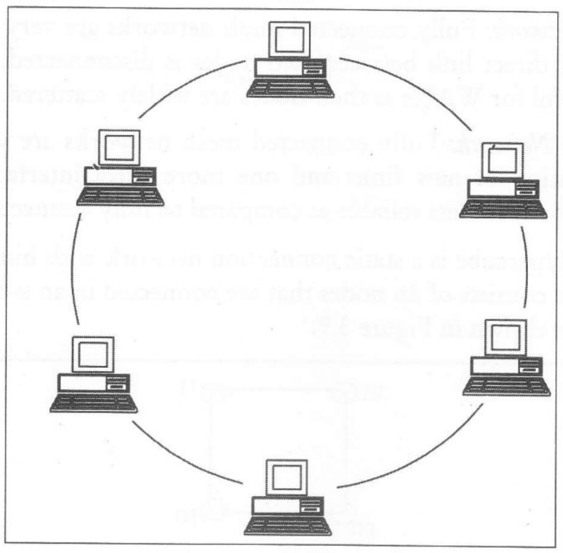


Figure 3.7: Ring Network

Advantages: The ring network does not have any routing problem as every node on the ring receives the data. As every generated packet eventually returns to the sender node, so there is no need of acknowledgment of successful data transmission. Ring network also avoids data collisions.

Disadvantages: Like bus network, the ring network will go down if a single connection is broken.

4. ***Mesh Network:*** In the mesh network, the nodes are connected randomly using the communications links. The mesh network is either fully connected or connected with partial links. In fully connected mesh topology, each node is connected to every other node as shown in Figure 3:8. So, there is no need of any routing protocol as nodes are directly connected. However, in partially connected mesh topology, each node is not connected to every other node. So, a router protocol or procedure is used to transfer information among widely scattered nodes, which are not directly connected.

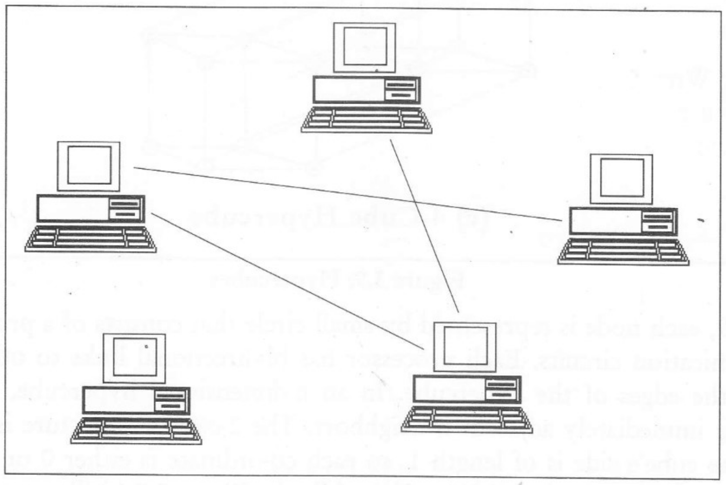
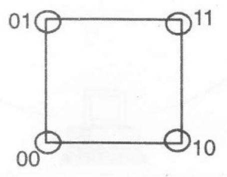


Figure 3.8: Mesh Network

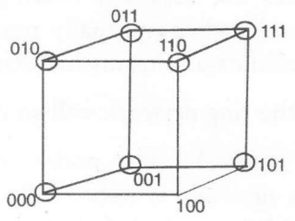
Advantages of Mesh Network: Fully connected mesh networks are very reliable, as alternate paths are always available if direct link between two nodes is disconnected. Partially connected mesh networks are very useful for WANs as their nodes are widely scattered.

Disadvantages of Mesh Network: Fully connected mesh networks are very expensive as adding a node requires installation of new links and one more extra interface in each node. Partially connected mesh networks are less reliable as compared to fully connected mesh networks.

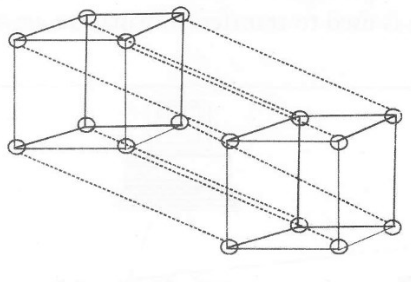
5. ***Hypercube Network:*** Hypercube is a static connection network with binary n-cube architecture. In a hypercube, an n-cube consists of 2n nodes that are connected in an n-dimensional cube with two nodes per dimension as shown in Figure 3.9.



(a) 2-Cube Hypercube



(b) 3-Cube Hypercube



(c) 4-Cube Hypercube

Figure 3.9: Hypercubes

In Figure 3.9, each node is represented by small circle that consists of a processor, local memory and communication circuits. Each processor has bi-directional links to other processors. These links form the edges of the hypercube. In an n-dimensional hypercube, each node is directly connected to immediately adjacent n neighbors. The 2-cube architecture is shown in Figure 3.9 (a). Here, the cube's side is of length 1, so each co-ordinate is either 0 or 1. The Figure 3.9 (b) illustrates a 3-cube hypercube with 8 nodes while the Figure 3.9 (c) illustrates 4-cube architecture, formed by interconnecting two 3-cubes architectures.

6. ***Tree Network:*** The basic tree network is the binary tree. It has two nodes (called sons) for the root node (called father). The interior nodes have three connections – two sons, and one father, whereas all the leaves just have one father as illustrated in Figure 3.10.

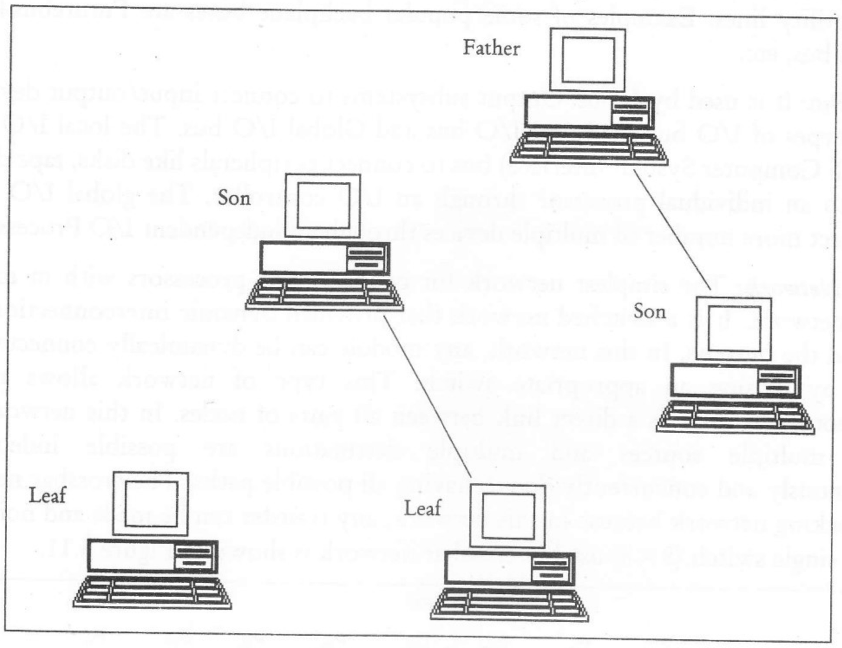


Figure 3.10: Tree Network

3.5.2 Dynamic Network and Switch Lattice Architecture

The dynamic networks are formed of dynamic connections that provide all possible communication patterns depending on the requirements of program execution. They are mainly implemented using switching elements, which can be dynamically configured to satisfy the communication demand during the execution of user programs. These networks include:

1. ***Hierarchical Common Bus System:*** The hierarchical common bus system is the simplest and most economical type of interconnection network. It is a digital bus system that consists of a hierarchy of different common buses. The structure and interfaces used in a multiprocessing system are almost the same as for a single-processor system that uses a bus interconnection. The processors, memory modules and various input/output units are directly attached to this bus system. Each bus is formed with control, address and data lines. However, different types of buses are used to perform different interconnection functions. The hierarchical bus system consists of the following types of buses:

   (a) *Local Buses:* They are implemented on PCBs (Printed Circuit Boards) and each processor is provided with a local bus. The local bus is connected to a local memory unit, which contains part of the shared address space. This system configuration provides common communication path among the major components. It also removes most of the routine memory traffic from the main system bus that can now devote more for inter-processor communication. The local bus is generally designed with standard buses like the PCI local bus and VESA Local Bus (VLB). The local bus can also support local I/O subsystems.

(b) *Backplane Bus:* It is a printed circuit on which many connectors are attached. The connectors are used to mount various functional plug-in boards. Motherboard can also be called a backplane bus. The system bus is constructed on the backplane bus with shared signal paths and utility lines. Examples of some popular backplane buses are Futurebus+, Multibus II, VME bus, etc.

(c) *I/O Bus:* It is used by Input/Output subsystems to connect input/output devices. There are two types of I/O buses - Local I/O bus and Global I/O bus. The local I/O bus uses SCSI (Small Computer System Interface) bus to connect peripherals like disks, tape drives, printers, etc. to an individual processor through an I/O controller. The global I/O bus is used to connect more number of multiple devices through an independent I/O Processor (lOP).

2. ***Crossbar Network:*** The simplest network for connecting n processors with m memories is the crossbar network. It is a switched network that provided dynamic interconnections between the inputs and the outputs. In this network, any module can be dynamically connected to any other module, by closing an appropriate switch. This type of network allows many transfers simultaneously as there is a direct link between all pairs of nodes. In this network, all transfers between multiple sources and multiple destinations are possible independently and asynchronously and concurrently due to having all possible paths. The crossbar network is called a non-blocking network because in this network, any transfer can be made and none is prevented. A simple single switch (8 × 8) used in crossbar network is shown in Figure 3.11.

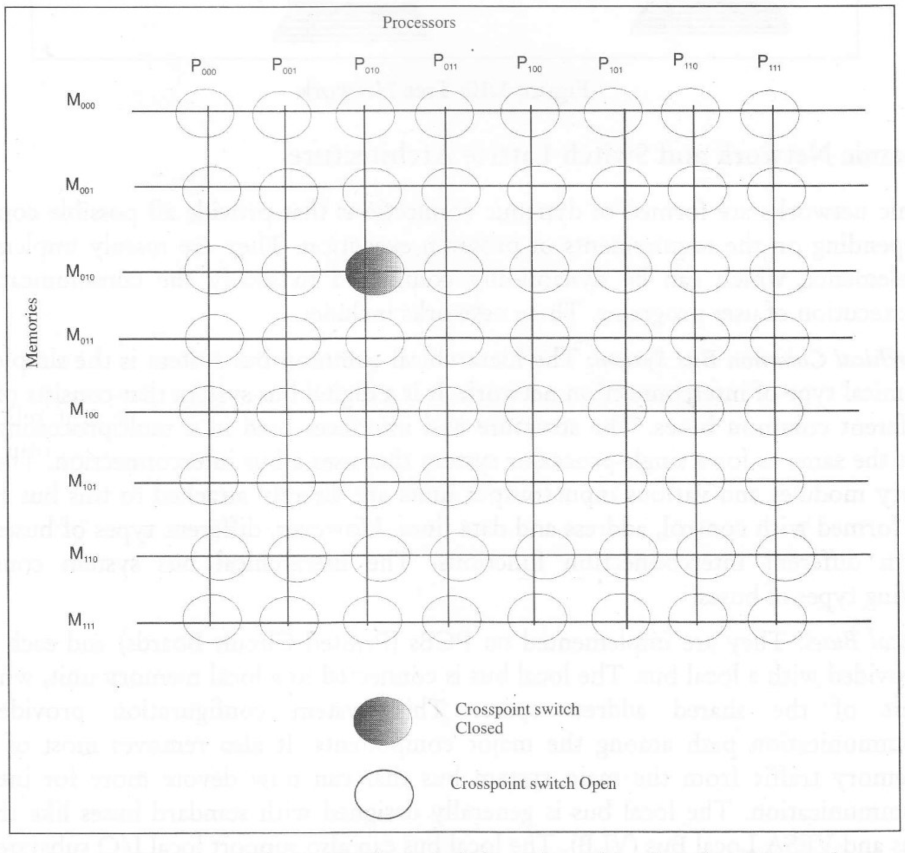


Figure 3.11: A Simple Single Switch (8 × 8) used in Crossbar Network

The crossbar switch is placed at each intersection of a horizontal and vertical line called a cross point. It can be electrically opened or closed. As n number of modules is interconnected in a crossbar network, the number of cross points becomes n2, the total number of switches needed becomes too large as n increases. This result in increase in the cost of hardware required for networking. Therefore, the crossbar networks are profitably used only in small or medium-sized systems.

3. ***Multiport Network:*** The multiport network uses a timeshared contention bus that allows main memory modules to have direct and independent access with each processor as shown in Figure 3.12.

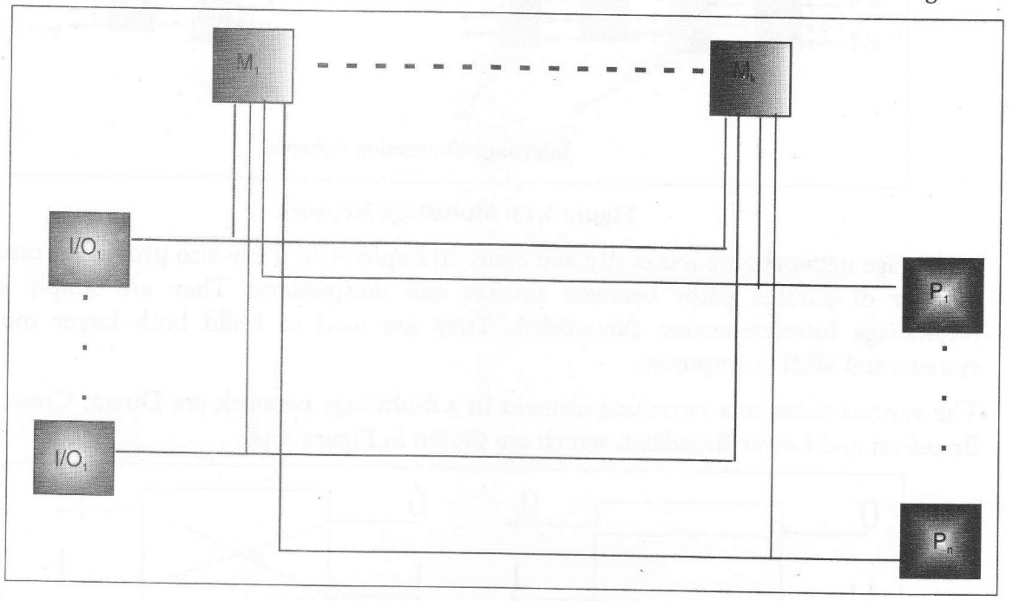


Figure 3.12: Multiport Network

In a multiport network, added logic is associated with each memory module to directly access ports. This network resolves conflicts by assigning permanently designated priorities to each memory port. As the physical and electrical interfaces at each such port are almost identical to a single-port memory module, neither the processor nor the I/O modules require any major modification to accommodate multiport memory.

Many mainframe multiprocessors use a multiport memory organization as it relieves the cross point switches from its additional burden of handling all arbitrations and conflict resolutions. The multiport network offers an improved performance as its memory system provides a dedicated path for each processor to each memory module. It is also possible to configure portions of memory as private to one or more processors/I/O modules. So, the multiport network also includes security features against unauthorized access.

4. ***Multistage Networks:*** Both bus-based and crossbar systems are single-stage switching networks. Multistage network is implemented by multiple stages of switches to set up paths between sources and destinations as shown in Figure 3.13.

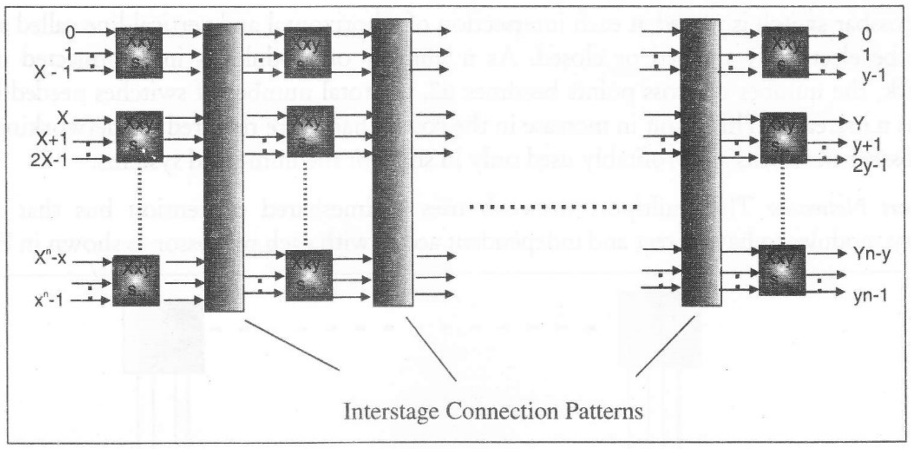


Figure 3.13: Multistage Network

Multistage networks are less costly and easier to implement. They also provide a reasonably large number of parallel paths between sources and destinations. They are simply called MIN (Multistage Interconnection Networks). They are used to build both larger multiprocessor systems and SIMD computers.

The various states of a switching element in a multistage network are Direct, Crossover, Upper Broadcast and Lower Broadcast, which are shown in Figure 3.14.

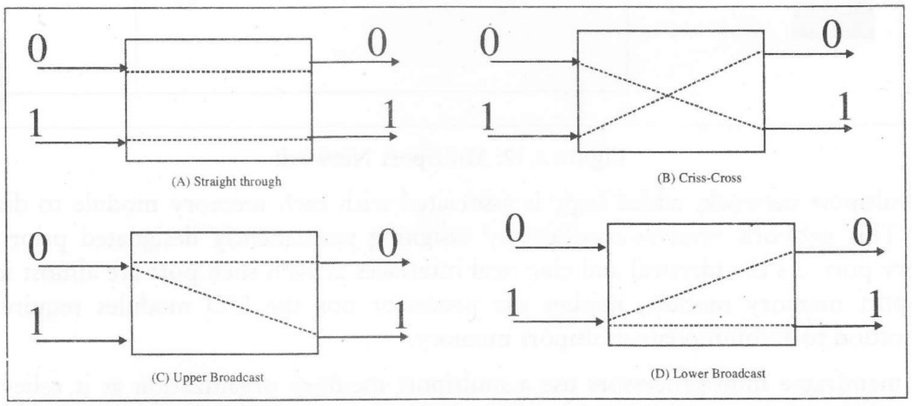


Figure 3.14: States of Switching Element in Multiport Network

5. ***Omega Network:*** Many kinds of multistage networks can be constructed, but the most economical among them is the Omega network. The Omega network can be used to broadcast data from one source to many destinations. It can be constructed by any of four possible connections of 2 x 2 switches by setting the control signals of the switching elements in various ways such as Processor to Processor or Processor to Memory connections. This can be established depending on the following factors:

   - Number of stages
   - Fixed connections linking the stages
   - Setting of the switching elements

   For example, an 8 × 8 omega network, built in 3 stages with 2 × 2 switches is shown in Figure 3.15.

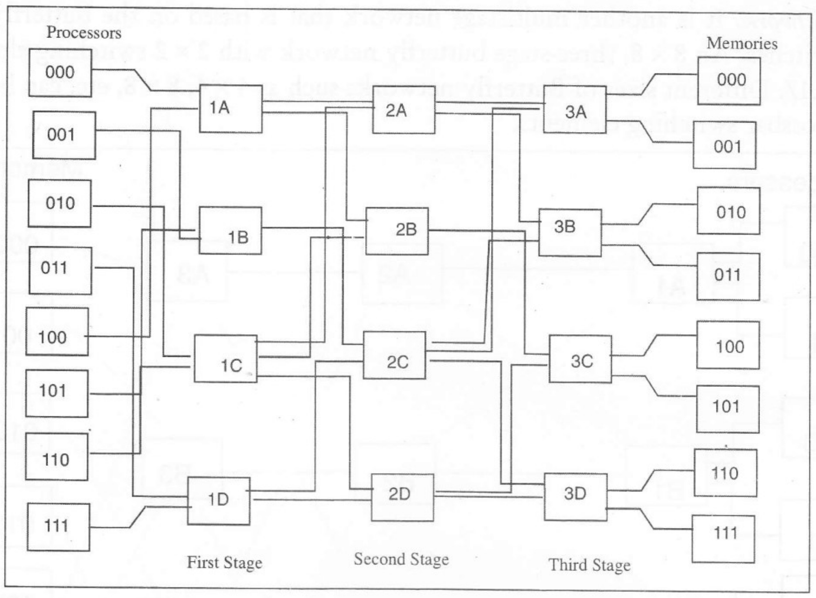


Figure 3.15: An 8 × 8 Omega Network, Built in 3 Stages with 2 × 2 Switches

In an 8 × 8 omega network, eight processors are connected to eight memories using 12 switching elements. They are arranged into three stages in order to provide dynamic connections among eight processors and eight memories. In general terms, for n processors and n memories, log2 n stages are needed with n/2 switching element per stage.

6. ***Benes Network:*** Benes network is a modification of omega network. In Omega network, there is exactly one path from any processor to any memory. However, Benes network has more than one path and many alternatives as illustrated in Figure 3.16. Blocking problem is one of the major drawbacks of omega network. It can be solved by adding more stages using more hardware. Benes network offer a flexible communication, but is expensive due to more number of wires and switches.

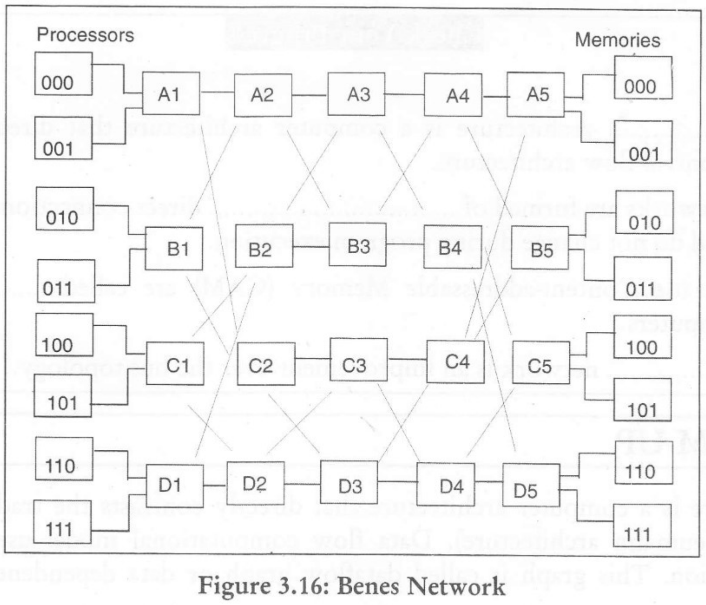


Figure 3.16: Benes Network

7. ***Butterfly Network:*** It is another multistage network that is based on the butterfly connection of crossbar switches. An 8 × 8, three-stage butterfly network with 2 × 2 switching elements is shown in Figure 3.17. Different sizes of Butterfly networks such as 4 × 4, 8 × 8, etc. can be built based on different crossbar switching elements.

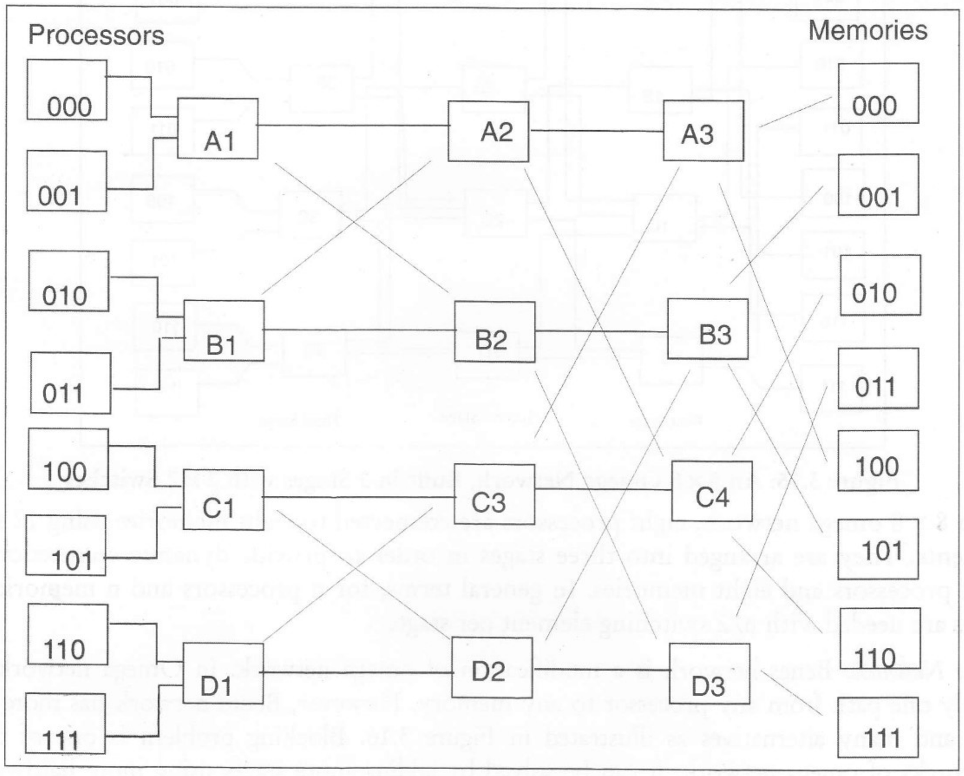


Figure 3.17: A 3-stage Butterfly Network

Check Your Progress

Fill in the blanks:

1. .... architecture is a computer architecture that directly contrasts the traditional control flow architecture.
2. The static networks are formed of .... direct connections, which are fixed once built and do not change during program execution.
3. Designs that use Content-addressable Memory (CAM) are called .... dataflow computers.
4. .... network is an improvement over the bus topology.

3.6 LET US SUM UP

Dataflow architecture is a computer architecture that directly contrasts the traditional control flow architecture (von Neumann architecture). Data flow computational model uses directed graph to describe a computation. This graph is called dataflow graph or data dependency graph. Data flow computer is an alternative computer design to store program systems. The various kinds of data flow systems are: Very Large Scale Integration (VSLI) Dataflow Computers, Static Dataflow Computers and Dynamic Dataflow Computers. In Very Large Scale Integration (VSLI) Dataflow Computers, the dataflow technique is applied to very large scale integrated arrays of interconnected cells. Designs that use conventional memory addresses as data dependency tags are called static dataflow computers. Designs that use Content-addressable Memory (CAM) are called dynamic dataflow computers. System interconnection network establishes communications between the computer resources for flowing the information either between any pair of modules within the system or from one module to other modules outside the system. In order to solve a problem, the processing elements have to cooperate and to exchange their computed data over the network.

3.7 KEYWORDS

CAM: Content-addressable memory

PCBs: Printed Circuit Boards

FIFO: First-In-First-Out

VSLI: Very Large Scale Integration

VLB: VESA Local Bus

3.8 QUESTIONS FOR DISCUSSION

1. Explain a model of data flow programming.
2. What is the difference between static and data flow computer?
3. How static network can be broadly classified?
4. What is cross-bass switched system?
5. Explain Switched Lattice Architecture with the example of multi-stage and multi-port network.

Check Your Progress: Model Answers

1. Dataflow
2. Point-to-point
3. Dynamic
4. Star

3.9 SUGGESTED READINGS

Sajjan G. Shiva; *Computer Design and Architecture*; Marcel Dekker

Silvia Melitta Mueller, Wolfgang J. Paul; *Computer Architecture*; Springer

Joseph D. Dumas II; *Computer Architecture*; CRC Press

Nicholas P. Carter; *Schaum's Outline of Computer Architecture*; Mc. Graw-Hill Professional