Design Methodology

System Design

- Computer – A system
- A collection of a large and complex components, that are connected to form a coherent entity with a specific function or purpose.

System Representation

- Two classes of objects are:
  - Components C
  - Set of Lines S that carry information
- System can be modelled by Graph G
  - C: Nodes of the graph
  - S: Edges of the graph
- The resulting graph is Block Diagram

Structure Vs Behavior

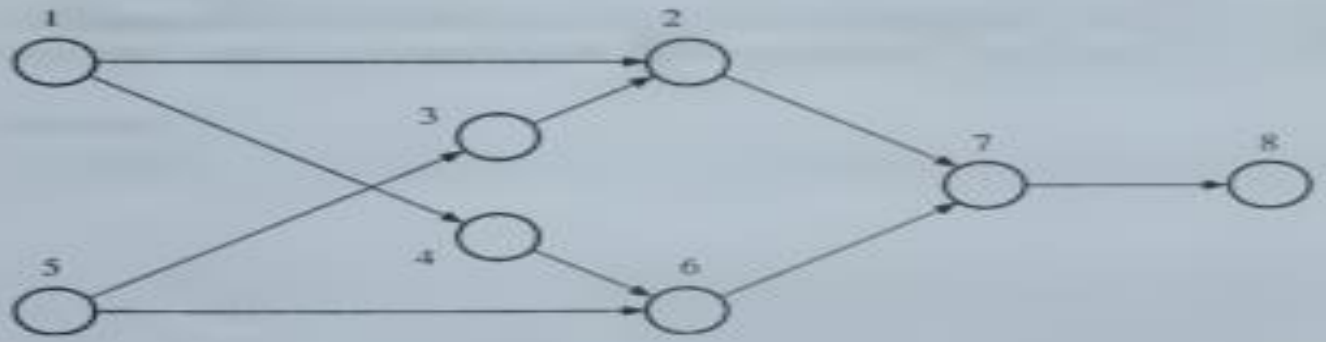
- Structure of a system is the abstract graph consisting of its block diagram with no functional information.
  - Names components and define their interconnection
- Behavioral description helps to determine for any given input ‘a’ to the system, the corresponding output f(a). It can be represented by:
  - Truth table
  - Mathematical equations(in the form of function)

$f(0,0) = 0$

$f(0,1) = 1$

$f(1,0) = 1$

$f(1,1) = 0$



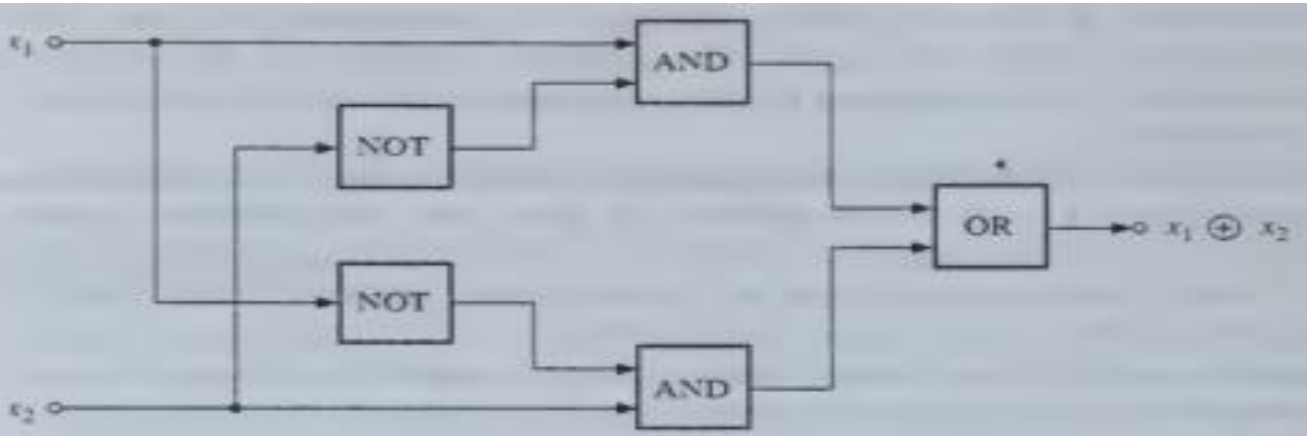


Figure 2.2
A block diagram representing an EXCLUSIVE-OR logic circuit.

Analysis Vs Synthesis

- Analysis- Given a system structure, the task of determining the behavior.
- Synthesis or Design- Determining a system structure that shows a given behavior.

Design Process

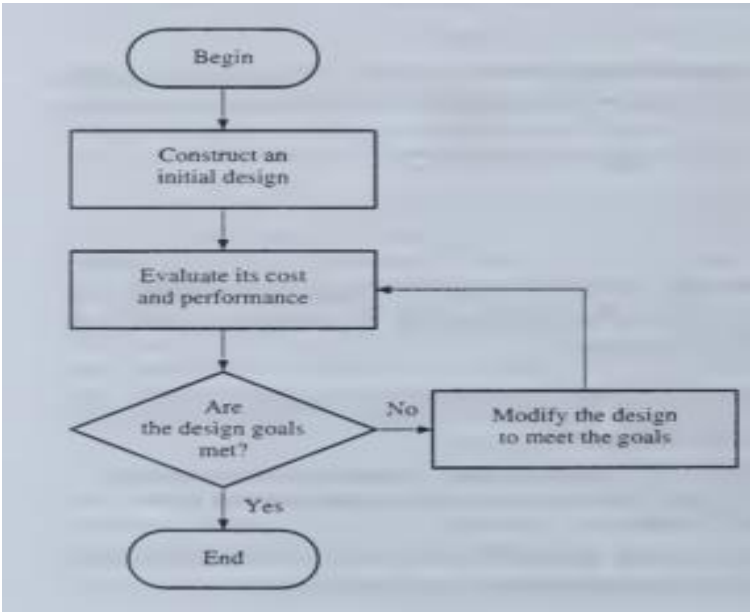
- Design Problem: Given a desired range of behavior and a set of available components, determine a structure formed from these components that achieves the desired behavior with acceptable cost and performance.

  The other constraints to satisfy are

  - High reliability
  - Low Power consumption
  - Compatibility with existing systems

Design Process

- Each Problem is broken down to smaller easier tasks involving various components.
- Smaller problems can be solved independently by different designers.
- Then it is combined to see whether it meets the relevant design objectives.
- Many iterations through the redesign is done to meet the necessary and satisfactory design.



Design Levels

- Three levels are recognized in computer design
  - The Processor Level also called the architecture, behavior or system level.
  - The Register Level, also called the register transfer level.
  - The gate level, also called the logic level

Level	Components	IC Density	Information Units	Time units
Gate	Logic gates, flipflops	SSI	Bits	10^{-12} to 10^{-9} s
Register	Registers, counters, combinational circuits, small sequential circuits	MSI	Words	10^{-9} to 10^{-6} s
Processor	CPUs, Memories, IO devices	VLSI	Block of words	10^{-3} to 10^{3}

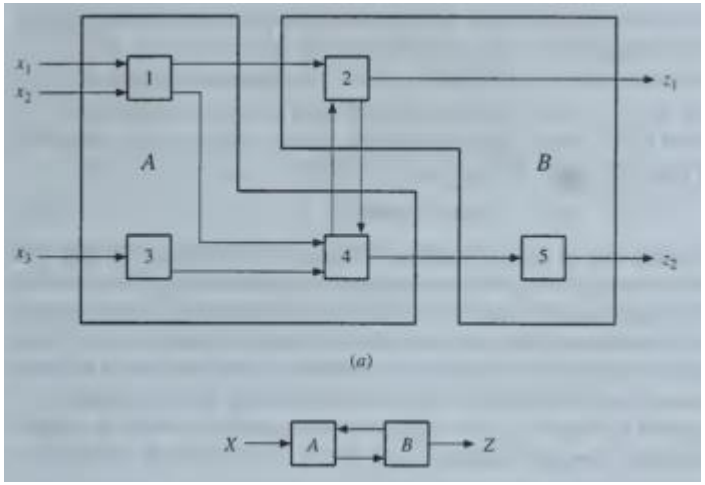
System hierarchy

- If a complex system is to be designed using small scale ICs or a single IC composed of standard cells, the design process consist of the following steps

  1. Specify the processor level structure of the system

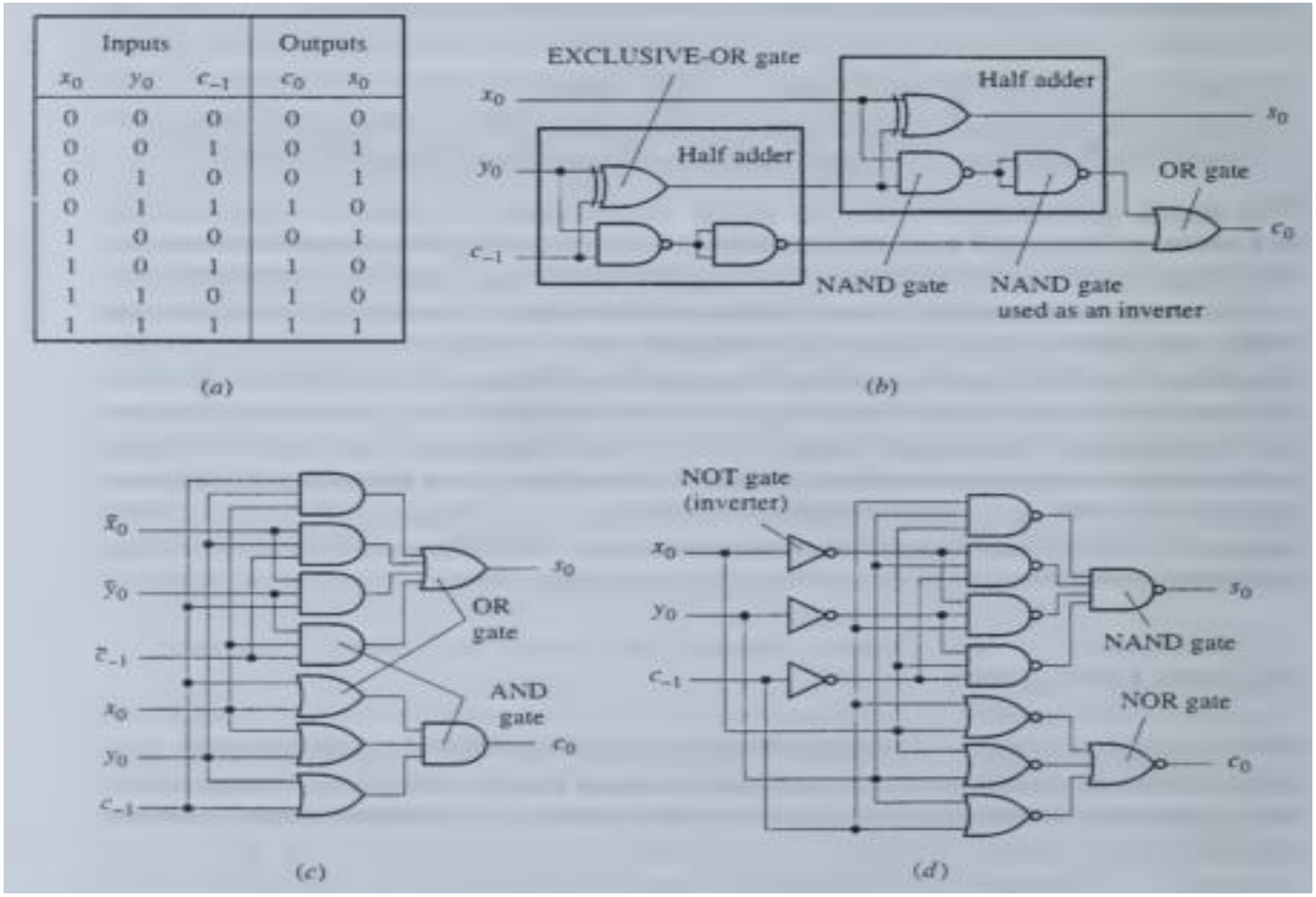
  2. Specify the register level structure of each component type identified in step 1.

  3. Specify the gate- level structure of each component type identified in step 2.



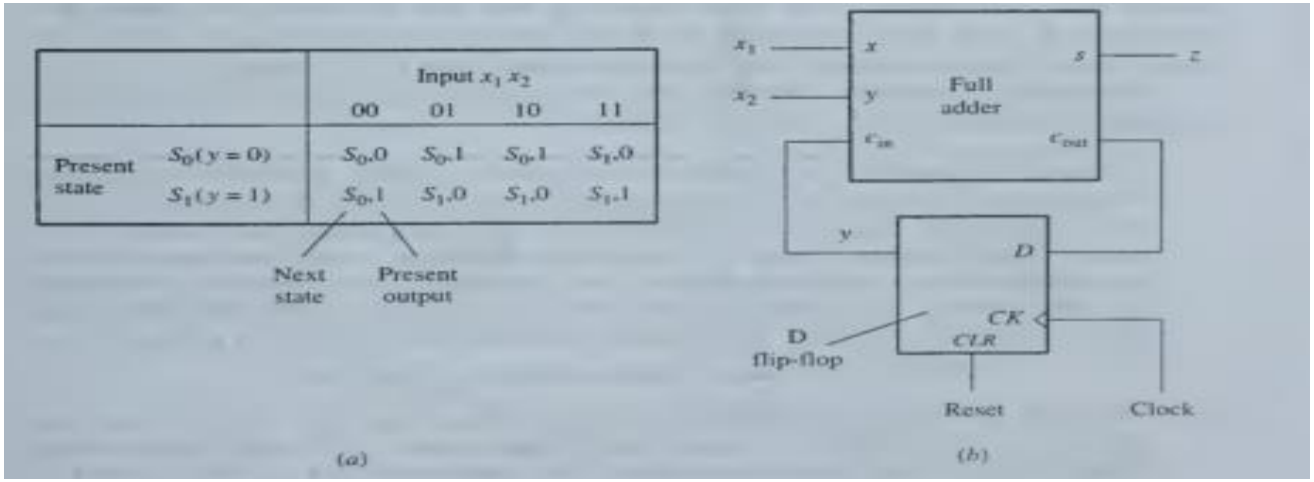
The Gate Level

Different ways to implement Full adder Ckt.



Inputs			Outputs	
x_0	y_0	c_{-1}	c_0	s_0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sequential Circuits



		Input $x_1 x_2$			
		00	01	10	11
Present state	$S_0 (y = 0)$	$S_0, 0$	$S_0, 1$	$S_0, 1$	$S_1, 0$
	$S_1 (y = 1)$	$S_0, 1$	$S_1, 0$	$S_1, 0$	$S_1, 1$

Next state

Present output

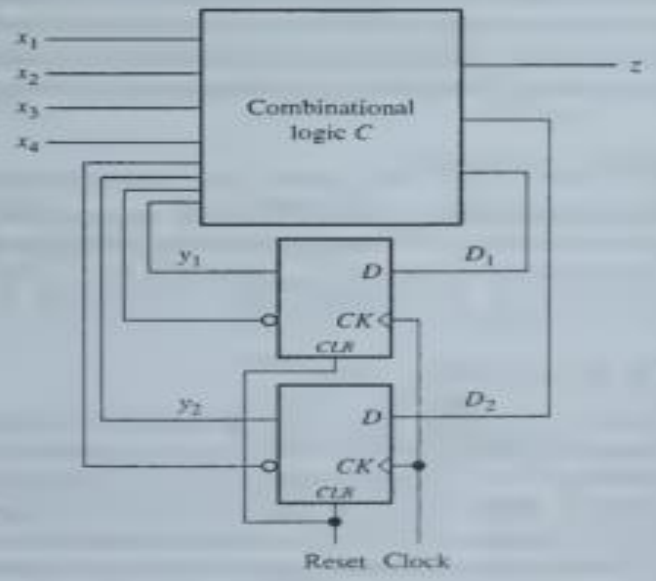
(a)

(b)

Design of a 4-bit-stream serial adder

		Present inputs $x_1x_2x_3x_4$ (decimal)															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Present state	S_0	$S_0,0$	$S_0,1$	$S_0,1$	$S_1,0$	$S_0,1$	$S_1,0$	$S_1,0$	$S_1,1$	$S_0,1$	$S_1,0$	$S_1,0$	$S_1,1$	$S_1,0$	$S_1,1$	$S_1,1$	$S_2,0$
	S_1	$S_0,1$	$S_1,0$	$S_1,0$	$S_1,1$	$S_1,0$	$S_1,1$	$S_1,1$	$S_2,0$	$S_1,0$	$S_1,1$	$S_1,1$	$S_2,0$	$S_1,1$	$S_2,0$	$S_2,0$	$S_2,1$
	S_2	$S_1,0$	$S_1,1$	$S_1,1$	$S_2,0$	$S_1,1$	$S_2,0$	$S_2,0$	$S_2,1$	$S_1,1$	$S_2,0$	$S_2,0$	$S_2,1$	$S_2,0$	$S_2,1$	$S_2,1$	$S_3,0$
	S_3	$S_1,1$	$S_2,0$	$S_2,0$	$S_2,1$	$S_2,0$	$S_2,1$	$S_2,1$	$S_3,0$	$S_2,0$	$S_2,1$	$S_2,1$	$S_3,0$	$S_2,1$	$S_3,0$	$S_3,0$	$S_3,1$

(a)



(b)

	Present inputs x_1 x_2 x_3 x_4	Present state y_1 y_2	Secondary outputs D_1 D_2	Primary output z
0	0 0 0 0	0 0	0 0	0
1	0 0 0 0	0 1	0 0	1
2	0 0 0 0	1 0	0 1	0
3	0 0 0 0	1 1	0 1	1
4	0 0 0 1	0 0	0 0	1
5	0 0 0 1	0 1	0 1	0
6	0 0 0 1	1 0	0 1	1
7	0 0 0 1	1 1	1 0	0
8	0 0 1 0	0 0	0 0	1
	...		...	
59	1 1 1 0	1 1	1 1	0
60	1 1 1 1	0 0	1 0	0
61	1 1 1 1	0 1	1 0	1
62	1 1 1 1	1 0	1 1	0
63	1 1 1 1	1 1	1 1	1

(c)

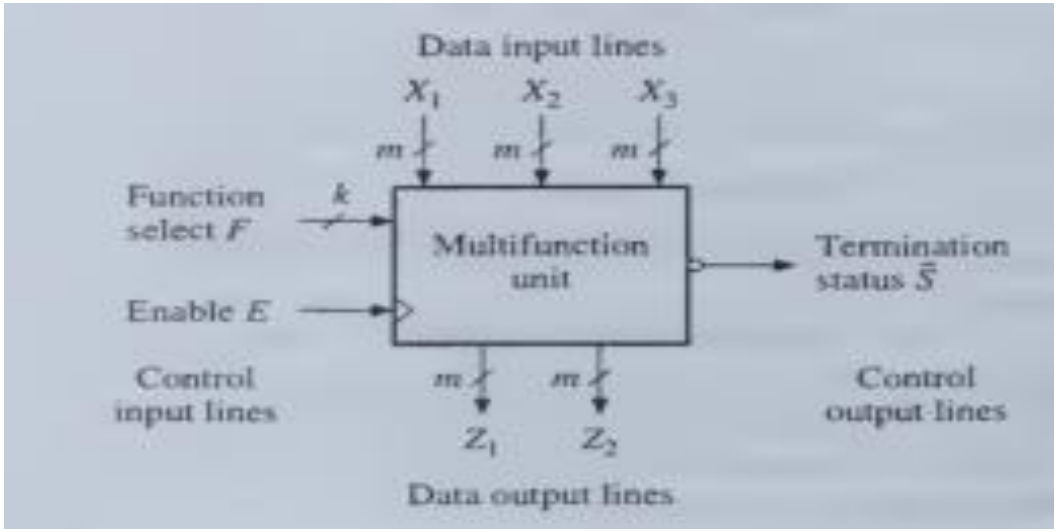
Figure 2.13
Four-bit-stream serial adder: (*a*) state table; (*b*) overall structure; (*c*) truth table for C.

Register Level Design

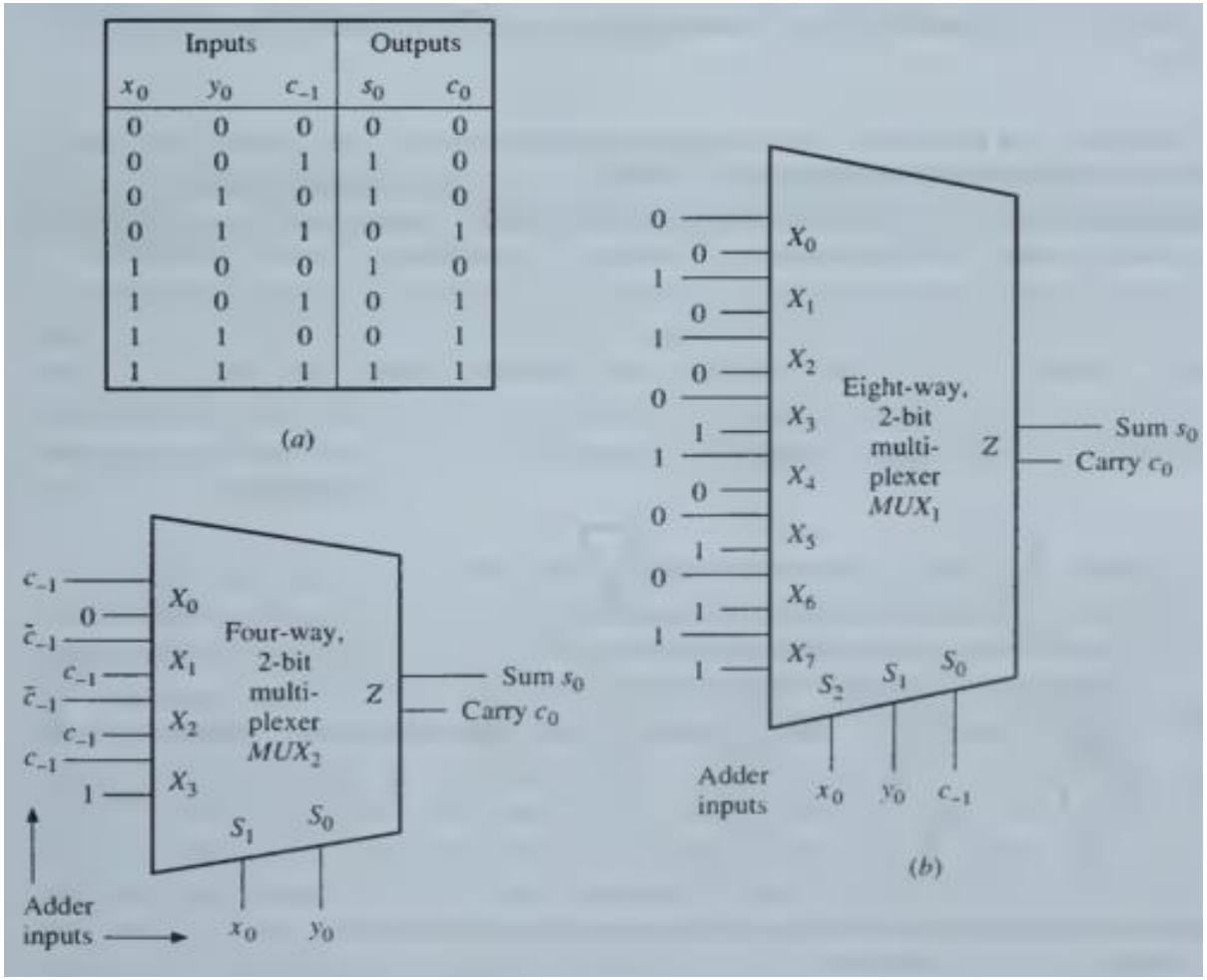
- Bits are group into words or vectors
- Both combinational and sequential circuits are used to process word

Type	Component	Functions
Combinational	Word gates.	Logical (Boolean) operations.
	Multiplexers.	Data routing; general combinational functions.
	Decoders and encoders.	Code checking and conversion.
	Adders.	Addition and subtraction.
	Arithmetic-logic units.	Numerical and logical operations.
	Programmable logic devices.	General combinational functions.
Sequential	(Parallel) registers.	Information storage.
	Shift registers.	Information storage; serial-parallel conversion.
	Counters.	Control/timing signal generation.
	Programmable logic devices.	General sequential functions.

- General Block representation of register level component



Example to implement a function using MUX



Inputs			Outputs	
x_0	y_0	c_{-1}	s_0	c_0
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Magnitude comparator using Adder

- Comparator is difficult to design at gate level
- A comparator for two n-bit numbers X and Y is given by

  X>Y => (X-Y)>0

  Now Y can be represented in 2's comp.

  i.e $(2^n-1)-Y'$

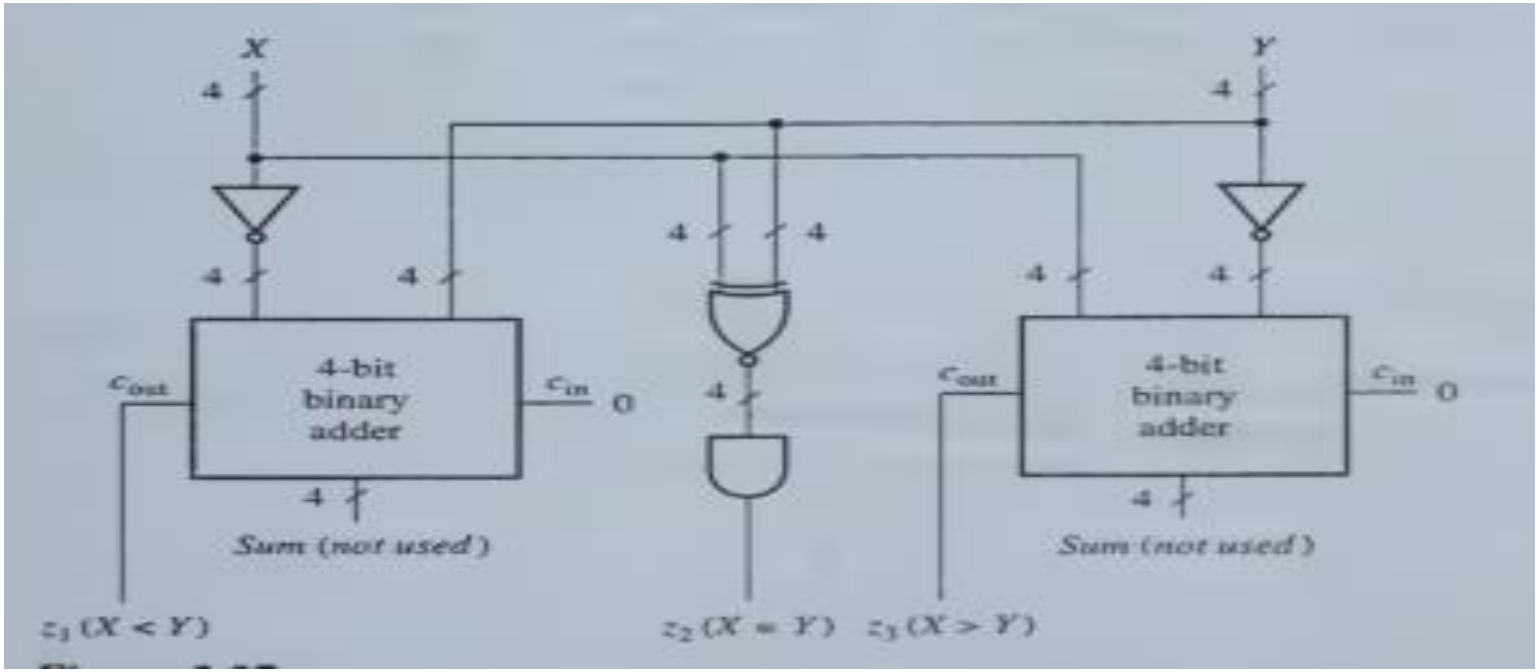
-> $X+Y' > (2^n-1)$

-> $X+Y' > 111\ldots1$

i.e If the inequality satisfies , then the C_{out} will be 1.

The status of C_{out} shows whether X>Y

The Same thing holds good to find whether X<Y.
For X=Y : Each bit of Xi is XNORed with Yi



Buses

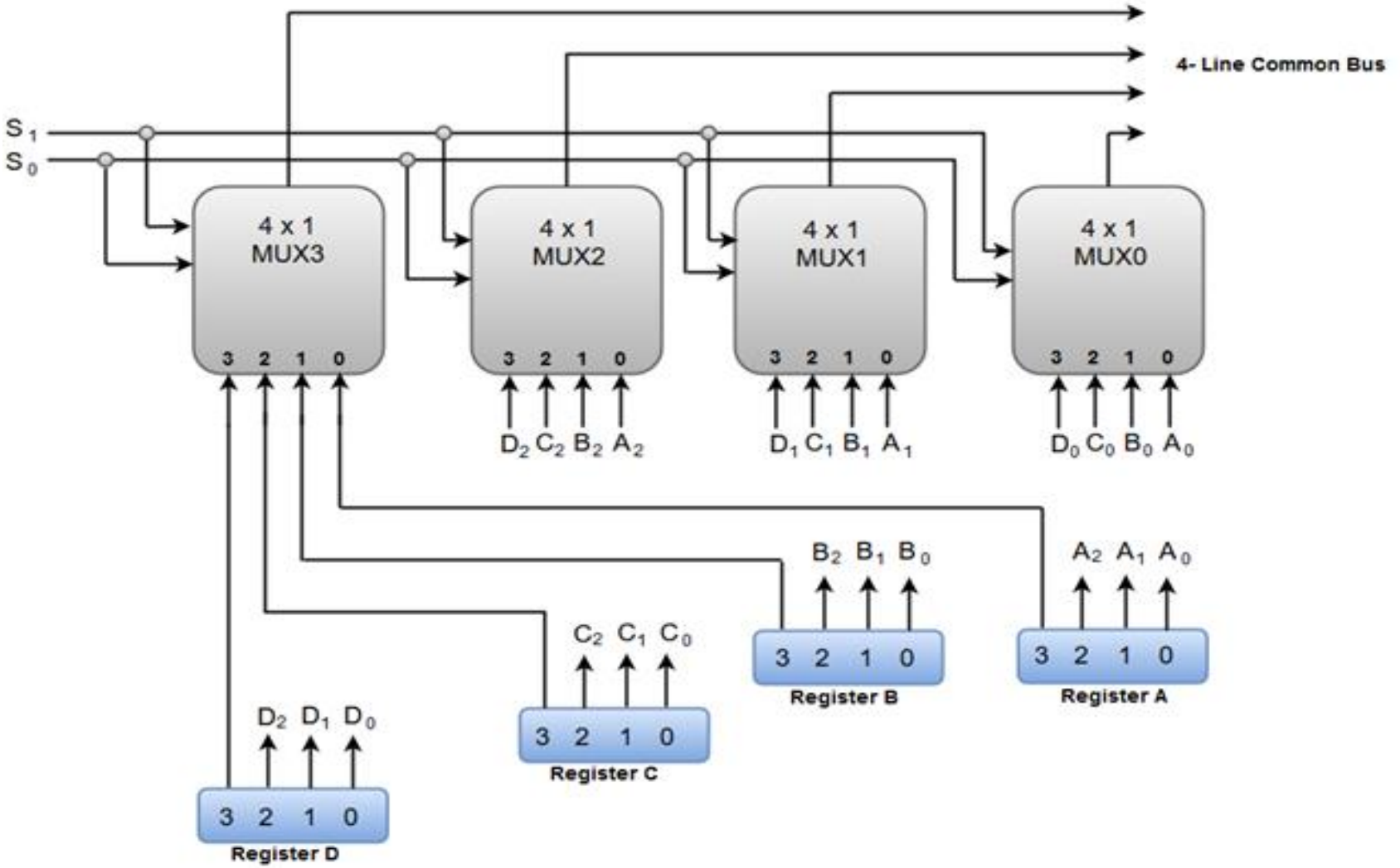
- They are a set of lines(wires) designed to transfer all bits of a word from a specified source to a specified destination on the same or a different IC.
- A bus consists of a set of common lines, one for each bit of register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during a particular register transfer.
- They can be unidirectional or bidirectional
- If used over longer distances, they require signal amplification circuit.
- To reduce cost bus can be shared(shared bus) or else it can be a Dedicated bus.

BUSES(Contd)

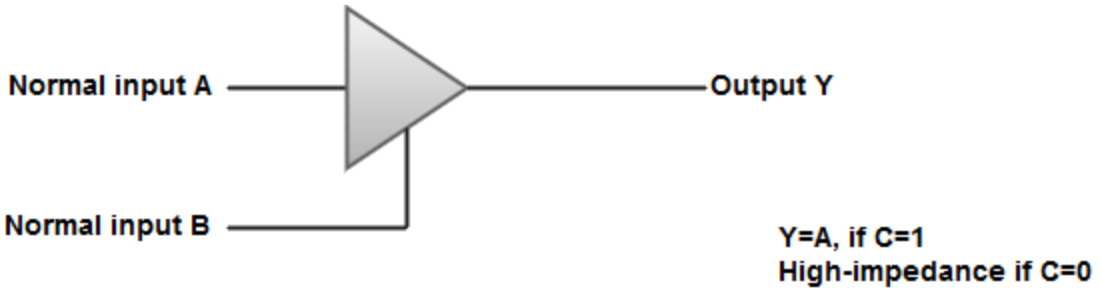
- The following block diagram shows a Bus system for four registers. It is constructed with the help of four 4 * 1 Multiplexers each having four data inputs (0 through 3) and two selection inputs (S1 and S2).
- The two selection lines S1 and S2 are connected to the selection inputs of all four multiplexers. The selection lines choose the four bits of one register and transfer them into the four-line common bus.

S1	S0	Register Selected
0	0	A
0	1	B
1	0	C
1	1	D

Bus System for 4 Registers:

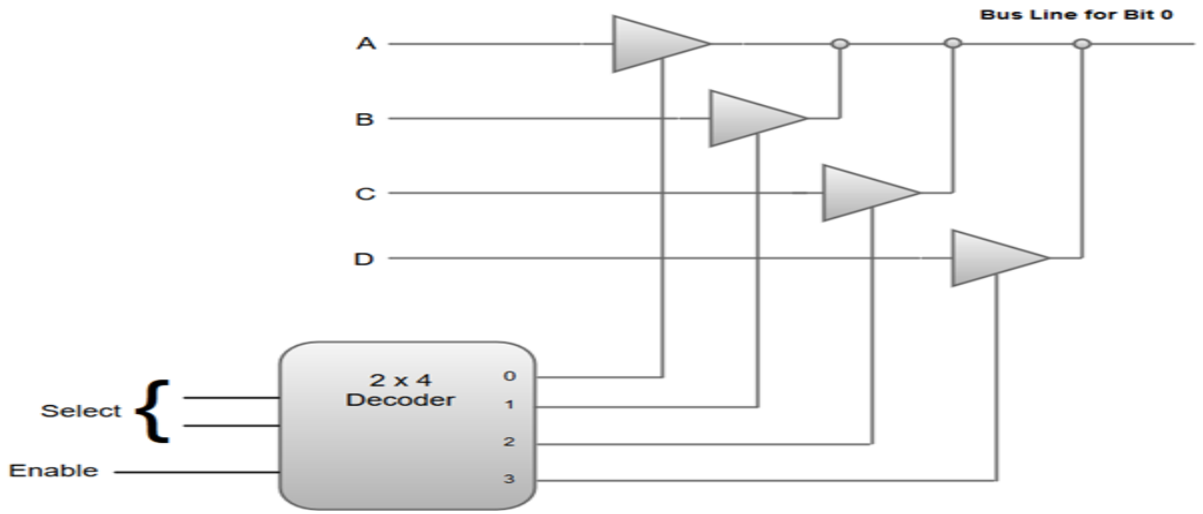


- A bus system can also be constructed using **three-state gates** instead of multiplexers.
- The **three state gates** can be considered as a digital circuit that has three gates, two of which are signals equivalent to logic 1 and 0 as in a conventional gate. However, the third gate exhibits a high-impedance state.

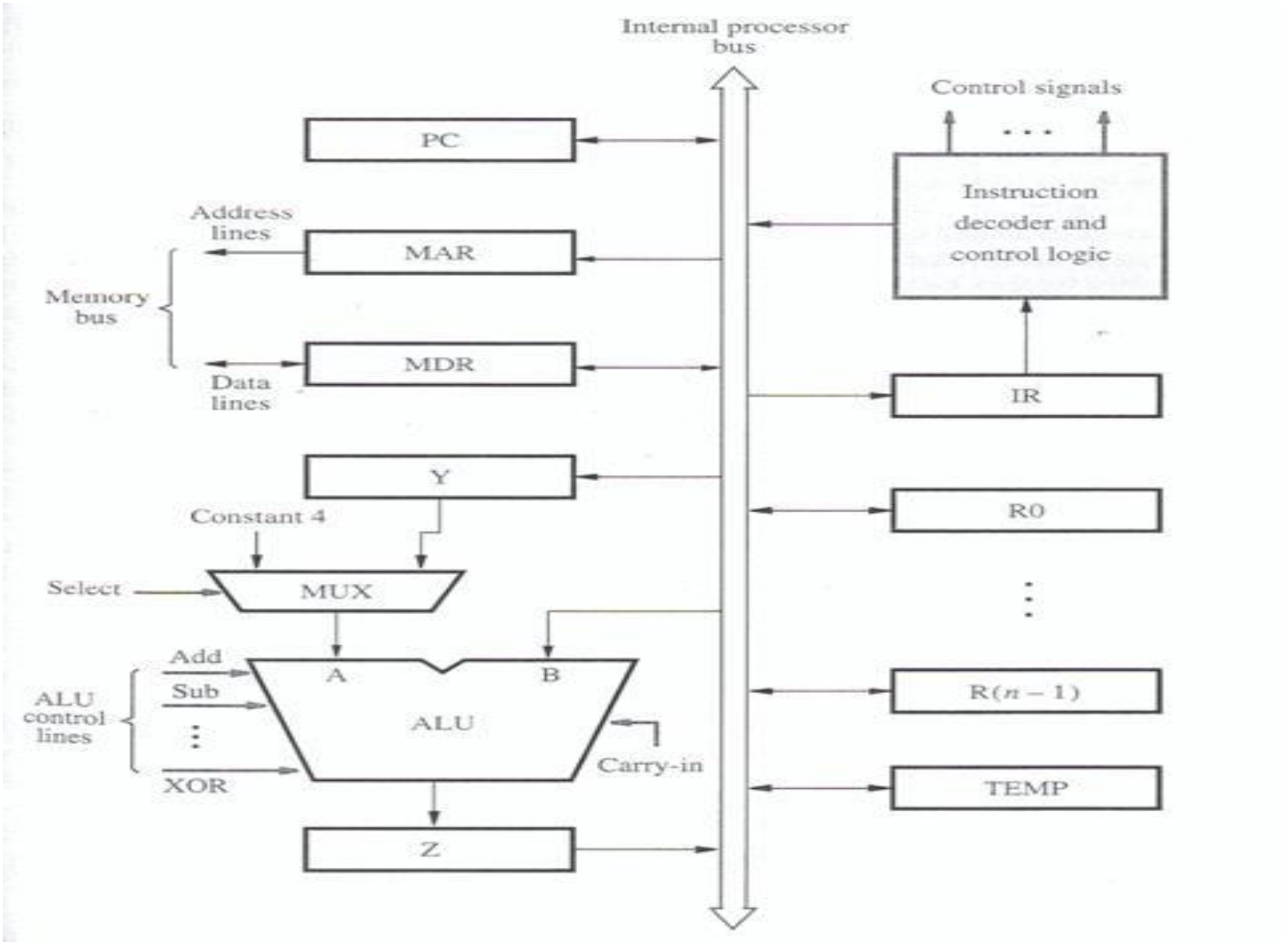


- The outputs generated by the four buffers are connected to form a single bus line.
- Only one buffer can be in active state at a given point of time.
- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.
- A 2 * 4 decoder ensures that no more than one control input is active at any given point of time

Bus line with three state buffer:

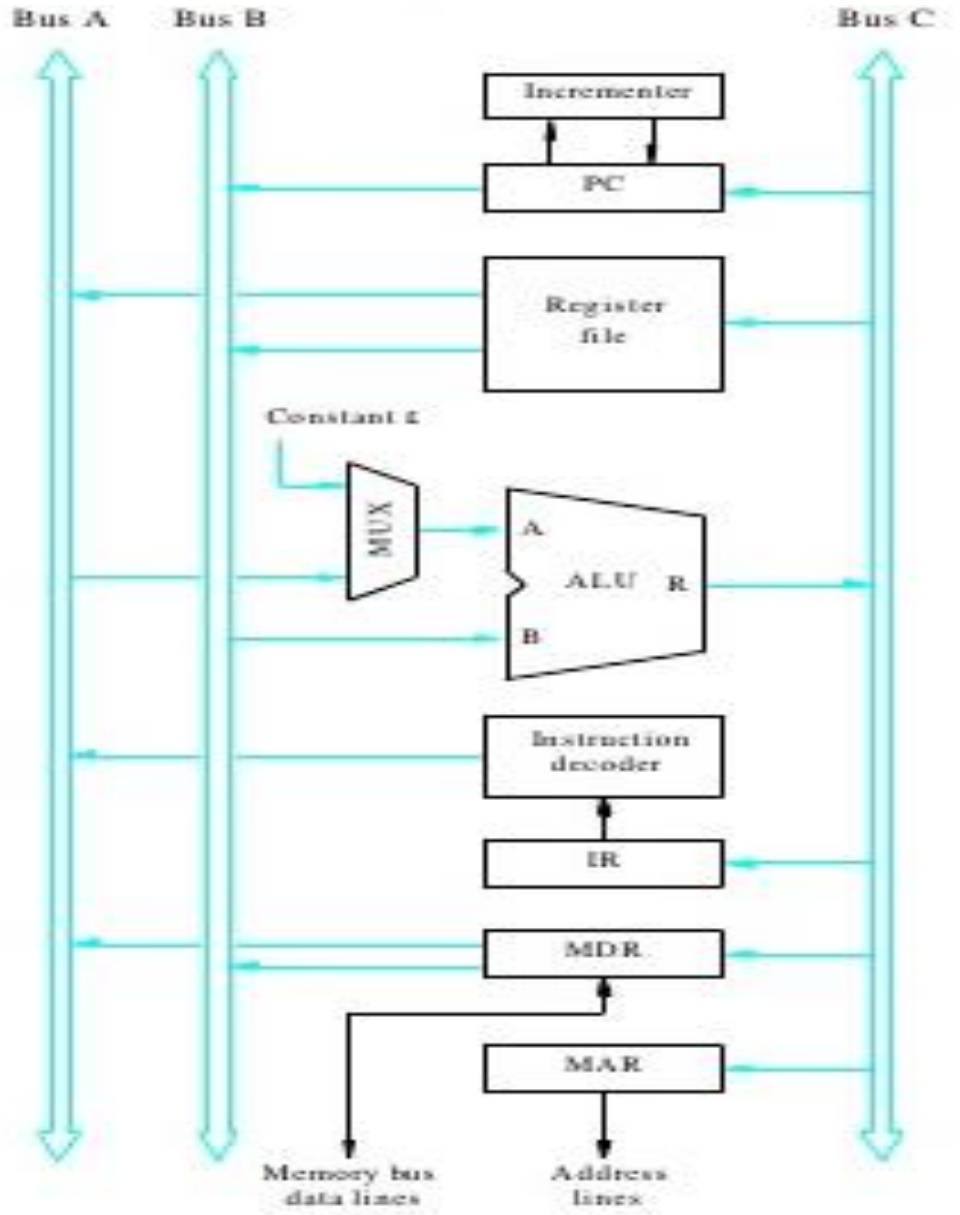


Single Bus Structure



Multiple Bus Structure

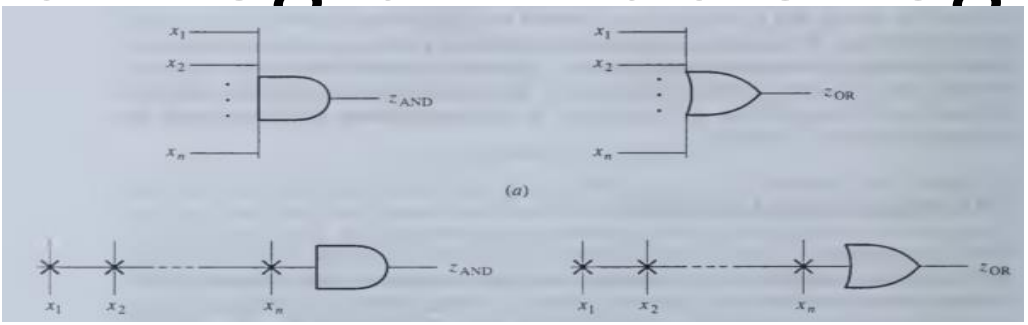
- Most of the modern processor provide multiple internal path that enable several data transfer to take place parallelly

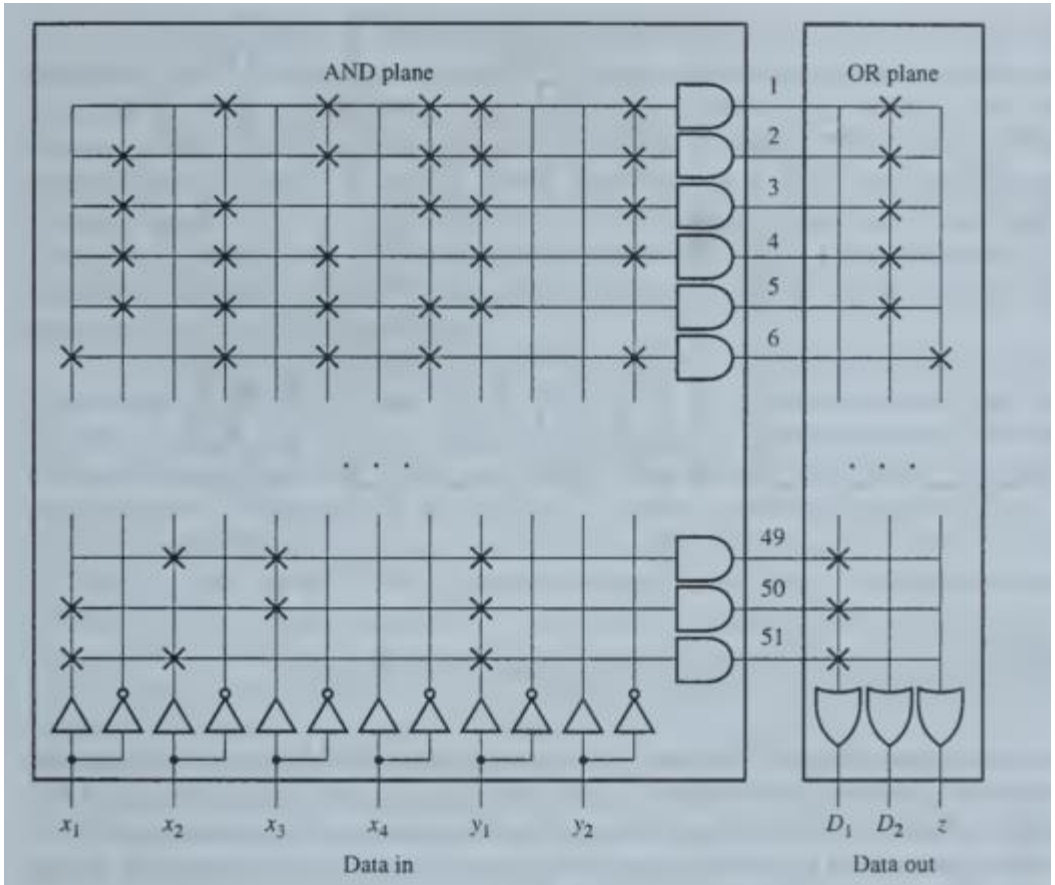

Bus A
Bus B
Bus C
Incrementer
PC
Register file
Constant 4
MUX
A
ALU
R
B
Instruction decoder
IR
MDR
MAR
Memory bus data lines
Address lines

CONNECTING REGISTRS

- In a digital system with many registers, it is impractical to have data and control lines to directly allow each register to be loaded with the contents of every possible other registers
- To completely connect n registers → n(n-1) lines $O(n^2)$ cost
  - This is not a realistic approach to use in a large digital system
- Instead, take a different approach
  - Have one centralized set of circuits for data transfer – the bus
  - Have control circuits to select which register is the source, and which is the destination

PLD and Programmable Logic Array





Register Level Design

- A register level system consists of a set of registers linked by combinational data transfer and data processing circuits.
- Each operation is implemented by one or more elementary register transfer steps of the form:
  - *cond*: z:= f(x1,x2,x3 …xk);

  Where f is a function to be performed or an instruction to be executed in one clock cycle.

  The prefix cond denotes a control condition that must be specified.

Register Transfer Language

- The operations executed on data stored in register are called microoperation
- A microoperation is an elementary operation performed on the information stored in one or more register.
- Examples of microperations are shift, load, count etc.

Register Transfer Language(contd)

- The internal hardware organization of a digital computer is best defined by specifying:

1. The set of registers it contains and their function.
2. The sequence of microoperations performed on the binary information stored in the registers.
3. The control that initiates the sequence of microoperations.

- Rather than specifying a digital system in words, a specific notation is used, *register transfer language*

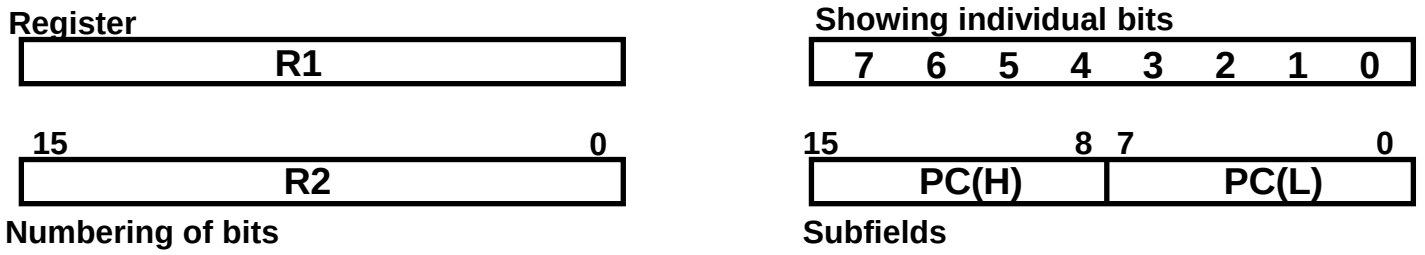
- For any function of the computer, the register transfer language can be used to describe the (sequence of) microoperations

- Register transfer language
  - A symbolic language
  - A convenient tool for describing the internal organization of digital computers
  - Can also be used to facilitate the design process of digital systems.

DESIGNATION OF REGISTERS

- **Designation of a register**
  - **a register**
  - **portion of a register**
  - **a bit of a register**

- **Common ways of drawing the block diagram of a register**



Register Transfer Language(Contd)

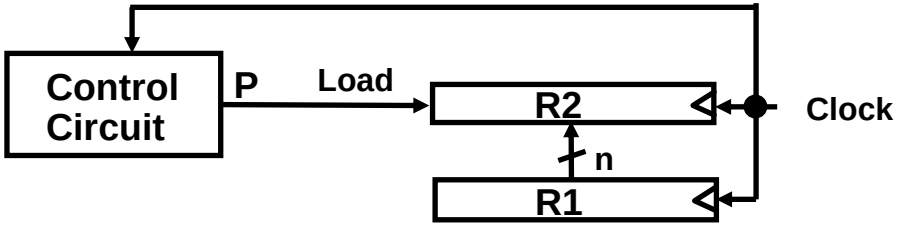
- Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register.
  - For example, the register that holds an address for the memory unit is usually called a memory address register and is designated by the name MAR.
  - R2 <-- R1
  - If (P = 1) then (R2 <-- R1) can be written as P: R2 <-- R1
  - A comma is used to separate two or more operations that are executed at the same time. The statement
  - T: R2 <- R1, R1 <- R2

HARDWARE IMPLEMENTATION OF CONTROLLED TRANSFERS

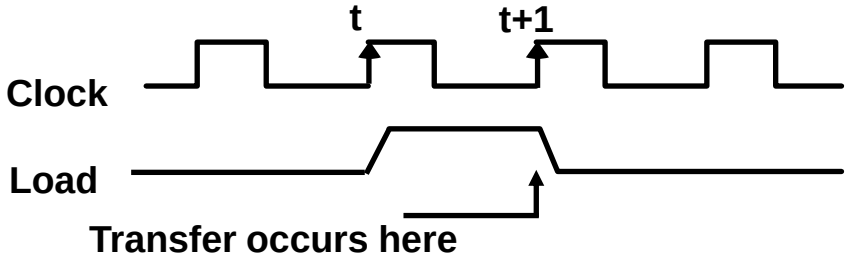
Implementation of controlled transfer

P: R2 ← R1

Block diagram



Timing diagram



- **The same clock controls the circuits that generate the control function and the destination register**
- **Registers are assumed to use *positive-edge-triggered* flip-flops**

BASIC SYMBOLS FOR REGISTER TRANSFERS

Symbols	Description	Examples
Capital letters & numerals	Denotes a register	MAR, R2
Parentheses ()	Denotes a part of a register	R2(0-7), R2(L)
Arrow ←	Denotes transfer of information	R2 ← R1
Colon :	Denotes termination of control function	P:
Comma ,	Separates two micro-operations	A ← B, B ← A

^ is AND ∧ is OR if it occurs in condition,
+ is "plus" if it occurs in register-transfer statement

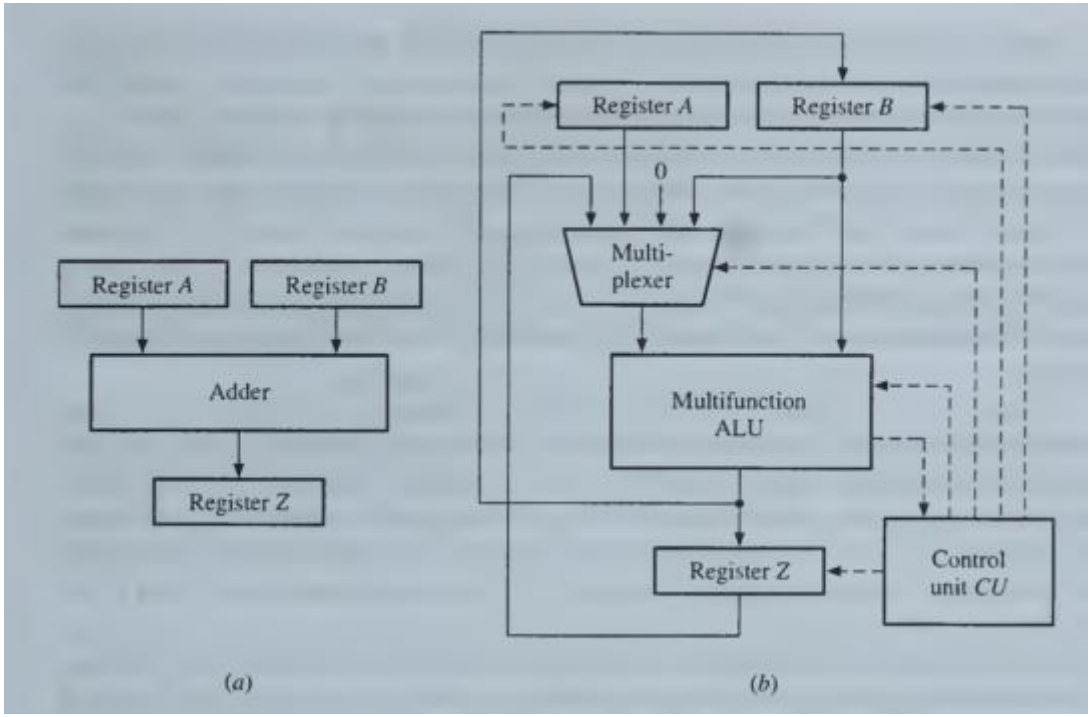
RTL SYNTAX

- Bits are numbered from the rightmost bit 0 (least significant) to leftmost bit n-1 (most significant)
- R1(0 - 3) denotes bits R13, R12, R11, and R10
- R1 $\leftarrow$ M[AR] denotes that register R1 gets the data from memory “pointed to” by the contents of the address register
  - A register transfer language statement has a one-to-one correlation with hardware connections

Register Level Design(Contd)

- The first figure performs a single action, Z:=A+B
- The Second figure shows a more complicated system that can perform several different operations. Such a multifunction system is generally partitioned into a data-processing part, called a datapath, and a controlling part, the control unit, which is responsible for selecting and controlling the actions of the datapath.
  - Z:=A+ B; B:=A-B;
- Less obvious operations that can be performed are the simple data transfer
- Z:=B, which is implemented as Z:=0+B;
- the clear operation B:=0 is implemented by B:=B-B
- The negative operation is B:=0-B

  Some double operations can also be performed in one clock cycle B :=Z+B, Z:=Z+B;


Register A
Register B
Adder
Register Z
(a)
Register A
Register B
0
Multi-
plexer
Multifunction
ALU
Register Z
Control
unit CU
(b)

Design techniques

- General approach to the design problem
  1. Define the behavior by a set of sequences of register transfer operation, such that each operation can be implemented directly using the available design components. This constitutes an algorithm AL to be executed.
  2. Analyze AL to determine the type of components and number of each type are required for the datapath DP
  3. Construct a block diagram for DP using the components identified in step 2. Make the connections between the components so that all data paths implied by AL are present and the given performance-cost constraints are met.

Approach to the design problem(Cntd)

4. Analyze AL and DP to identify the control signals needed. Introduce into DP the logic or control points necessary to apply these signals.
5. Design a control unit CU for DP that meets all the requirements of AL.
6. Verify, typically by computer simulation, that the final design operates correctly and meets all performance-cost goals.

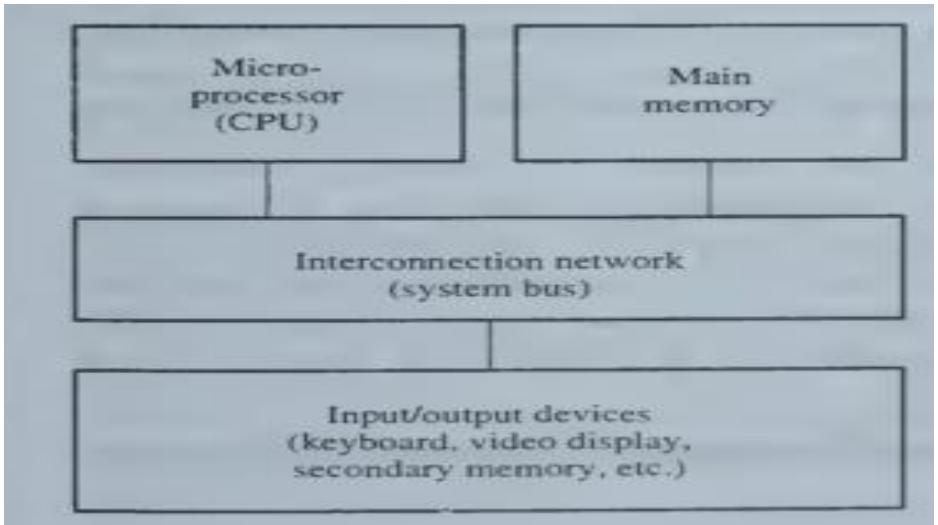
- The identification of the data-processing components in step 2 is straight-forward, but complications arise when the possibility of sharing components exists.
- For example, c: A:=A + B, C:=C + D;
- they can be done in parallel if two independent adders are provided.
- However, costs can be lowered by sharing a single adder and performing the two additions sequentially,
  - Thus: c(t0) : A :=A + B;
    - c(t0+1) : C:=C+ D;

The Processor Level

- Highest level in the hierarchy
- Processing of blocks of information such as programs and data files.
- It is a process, a little design theory at this level of abstraction.

Processor-Level Components

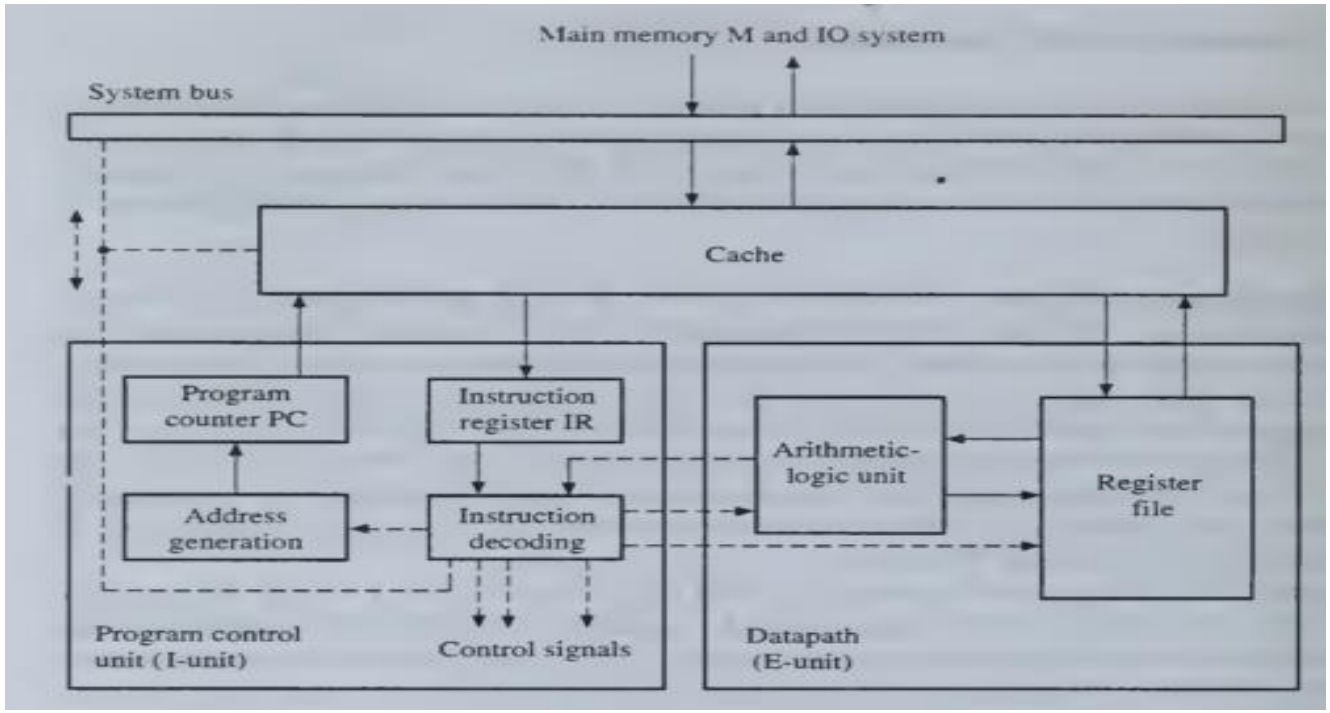
- The component at the processor level fall into four main groups:
  - processors
  - memories
  - IO devices
  - interconnection networks



CPU

- CPU to be a general-purpose, instruction-set processor that has overall responsibility for program interpretation and execution in a computer system.
- The qualifier general-purpose CPUs distinguishes from other, more specialized processors, such as IO processors (IOPs), whose functions are restricted.
- Most contemporary CPUs are microprocessors, implying that their physical implementation is a single VLSI chip.

•The essential internal organization of a CPU at the register level.
• The CPU contains the Logic needed to execute its particular instruction set and is divided into datapath and control units.
• The control part (the I-unit) generates the addresses of instructions and data stored in external memory.
•The datapath (E-unit) has the arithmetic-logic circuits that execute most instructions; it also has a set of registers for temporary data storage.



- The CPU is a synchronous sequential circuit whose clock period is the computer's basic unit of time.
- In one clock cycle the CPU can perform a register-transfer operation, such as fetching an instruction word from M via the system bus and loading it into the instruction register IR.
- This operation can be expressed formally by
  - IR:=M(PC);
- The I-unit then issues the sequence of control signals that enables execution of the instruction in question.
- The entire process of fetching, decoding, and executing an instruction constitutes the CPU's **instruction cycle.**

Memory

- CPUs and other instruction-set processors operate in conjunction with external memories that store the programs and data required by the processors .
- Numerous memory technologies exist, and they vary greatly in cost and performance. The cost of a memory device generally increases rapidly with its speed of operation.
- The memory part of a computer can be divided into several major subsystems:

  1. Main memory M, consisting of relatively fast storage ICs connected directly to, and controlled by, the CPU.

2. Secondary memory, consisting of less expensive devices that have very high storage capacity. These devices often involve mechanical motion and so are much slower than M. They are generally connected indirectly (via M) to the CPU and form part of the computer‘s IO system.

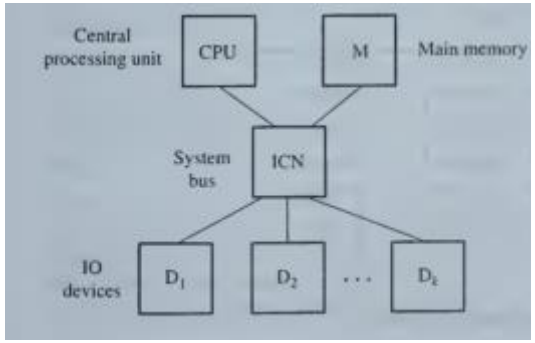
3. Many computers have a third type of memory called a cache, which is positioned between the CPU and main memory. The cache is intended to further reduce the average time taken by the CPU to access the memory system. Some or all of the cache may be integrated on the same IC chip as the CPU itself.

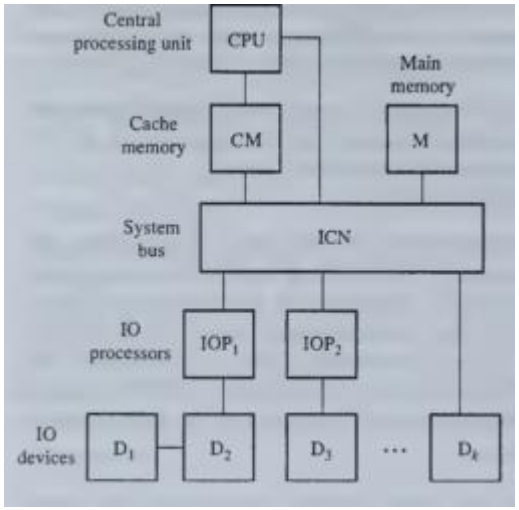
IO Devices

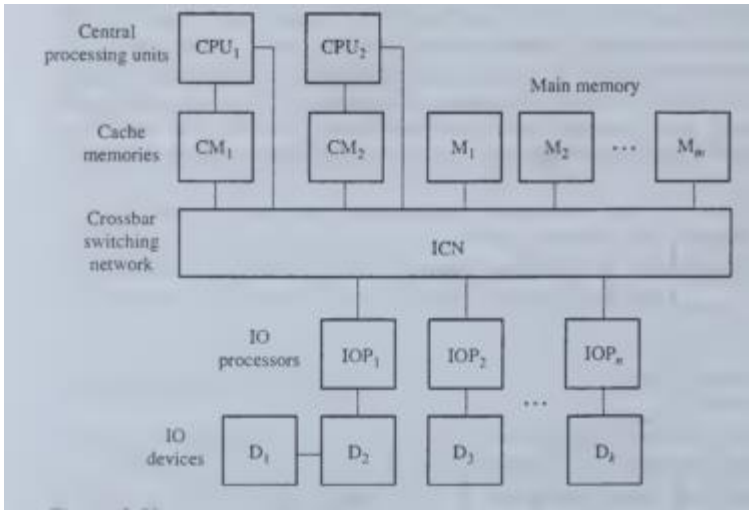
- IO Devices are the means by which a computer communicates with the outside world.
- A primary function of IO devices is to act as data transducers, that is, to convert information from one physical representation to another.

Interconnection network

- Processor-level components communicate by word-oriented buses.
- In systems with many components, communication may be controlled by a subsystem called an interconnection network; terms such as
  - switching network, communications controller, and bus
- The function of the interconnection network is to establish dynamic communication paths among the components via the buses
- For cost reasons, the paths are usually shared.


Central processing unit
CPU
M
Main memory
System bus
ICN
IO devices
D_1
D_2
D_k


Central processing unit
CPU
Main memory
Cache memory
CM
M
System bus
ICN
IO processors
IOP_1
IOP_2
IO devices
D_1
D_2
D_3
D_k


Central processing units
CPU_1
CPU_2
Main memory
Cache memories
CM_1
CM_2
M_1
M_2
M_m
Crossbar switching network
ICN
IO processors
IOP_1
IOP_2
IOP_n
IO devices
D_1
D_2
D_3
D_k

- Processor-level components is generally asynchronous in that the components cannot be synchronized directly by a common clock signal.
- This synchronization problem can be due to several causes.
  - A high degree of independence exists among the components.

    For example- CPUs and IOPs execute different types of programs and interact relatively infrequently and at unpredictable times.
  - Component operating speeds vary over a wide range. CPUs operate from 1 to 10 times faster than main-memory devices, while main-memory speeds can be many orders of magnitude faster than IO-device speeds.
  - The physical distance separating the components can be too large to permit synchronous transmission of information between them.

Processor Level Design

- Processor-level design is less amenable to formal analysis than is design at the register level.
- The design approach at this level is to take a prototype design of known performance and modify it where necessary to accommodate new technologies or meet new performance requirements.

Processor Level Design(Contd)

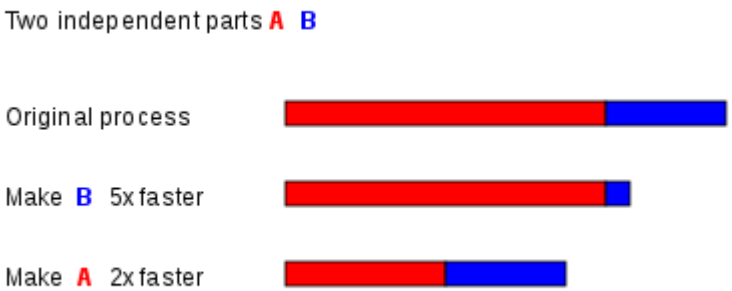
- The performance specifications usually take the following form:
  - The computer should be capable of executing *a* instructions.
  - The computer should be able to support *c* memory or IO devices.
  - The computer should be compatible with computers of type *e*.
  - The total cost of the system should not exceed *f*

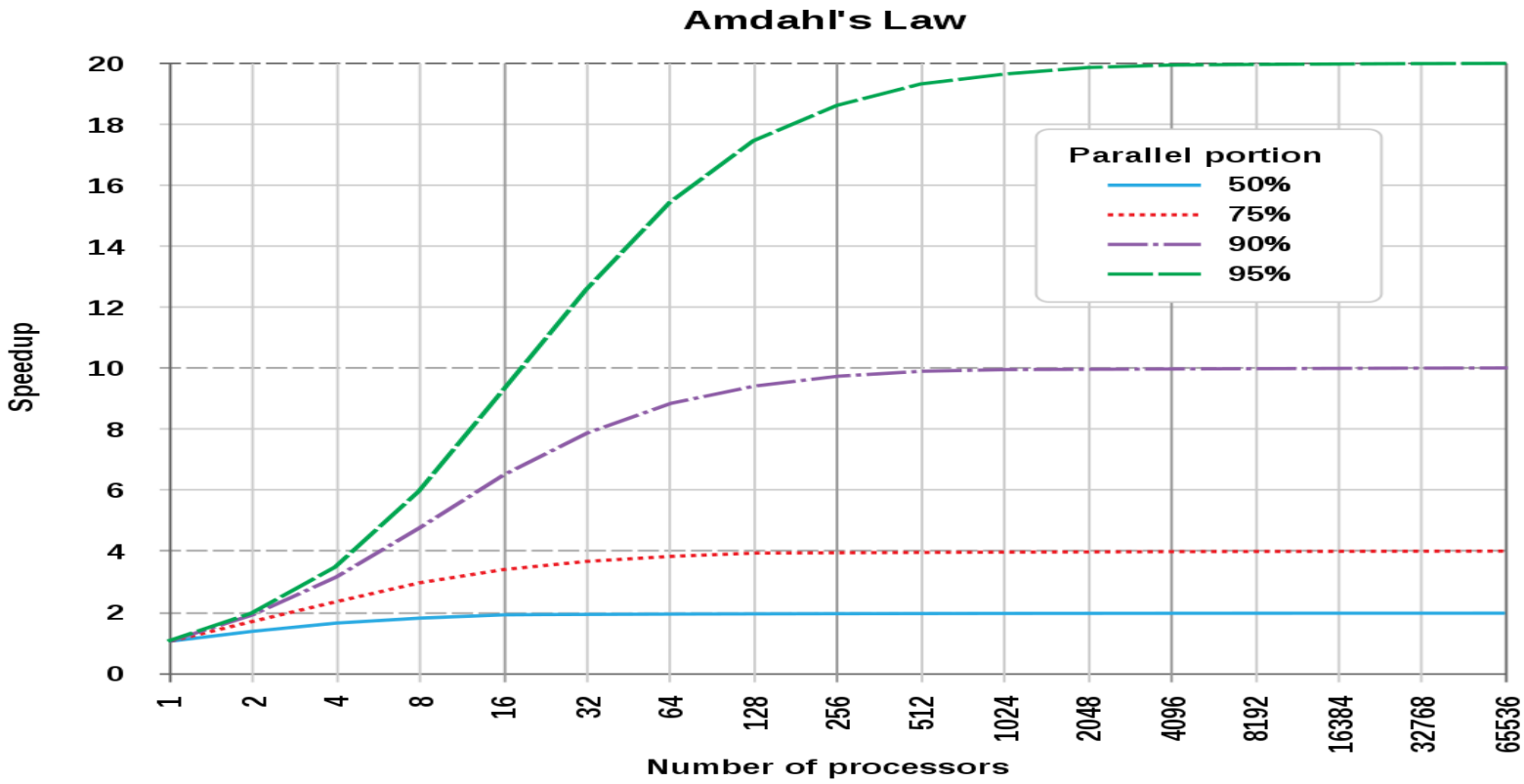
Prototype structures

- The design process involves two major steps:
  - Select a prototype design and adapt it to satisfy the given performance constraints
  - Determine the performance of the proposed system. Modify if unsatisfactory .Repeat the steps until an acceptable design is obtained.
- The need to remain compatible with the existing hardware and software standards also influence the design

Amdahl's Law

- Designers look for ways to improve system performance by advances in technology or change in design
    - For Ex: Use of Parallel Processors, Cache memories, speedup in memory/IO access time etc.
- A speedup in one aspect of the technology or design does not result in a corresponding improvement in performance.
- This limitation is succinctly expressed by Amdahl's law.


Two independent parts A B
Original process
Make B 5x faster
Make A 2x faster


Amdahl's Law
Speedup
Number of processors
Parallel portion
50%
75%
90%
95%
0
2
4
6
8
10
12
14
16
18
20
1
2
4
8
16
32
64
128
256
512
1024
2048
4096
8192
16384
32768
65536

- Consider a program running on a single processor such that a fraction (1 - *f) of the execution time involves code that is* inherently sequential, and a fraction *f that involves code that is infinitely parallelizable* with no scheduling overhead.
- Let *T be the total execution time of the program* using a single processor. Then the speedup using a parallel processor with *N processors* that fully exploits the parallel portion of the program is as follows:

- Speedup = Time to execute program on a single processor
  Time to execute program on *N parallel processors*

$$= \frac{T(1-f)+Tf}{T(1-f)+\frac{Tf}{N}} = \frac{1}{(1-f)+\frac{f}{N}}$$

Two important conclusions can be drawn:

1. When *f is small, the use of parallel processors has little effect.*

2. As *N approaches infinity, speedup is bound by* 1/ (1 - *f),* so that there are diminishing returns for using more processors.

$$\text{Speedup} = \frac{\text{Performance after enhancement}}{\text{Performance before enhancement}} = \frac{\text{Execution time before enhancement}}{\text{Execution time after enhancement}}$$

Example

- Suppose that a task makes extensive use of floating-point operations, with 40% of the time consumed by floating-point operations. With a new hardware design, the floating-point module is sped up by a factor of *K. Then* the overall speedup is as follows:

Speedup $= 1/(1 - f) + (f/SU_f)$
$= 1/((0.6)+(0.4/k))$

Thus, independent of *K, the maximum speedup is 1.67.*

Amdahl's law – Example 2

Suppose that we are considering an enhancement that runs 10 times faster than the original machine but is usable only 40% of the time. What is the overall speedup gained by incorporating the enhancement?

Se = 10

F = 40/100 = 0.4

So = 1 / ((1 - F) + F/Se)

= 1/ (0.6 + 0.4/10)

= 1/0.64 = 1.56.