Chapter 8
Memory Units

Contents:

--

I. Introduction

The computer system essentially comprises three important parts – input device, central processing unit (CPU) and the output device. The CPU itself is made of three components namely, the arithmetic logic unit (ALU), memory unit, and the control unit.

In addition to these, auxiliary storage/secondary storage devices are used to store data and instructions on a long-term basis.

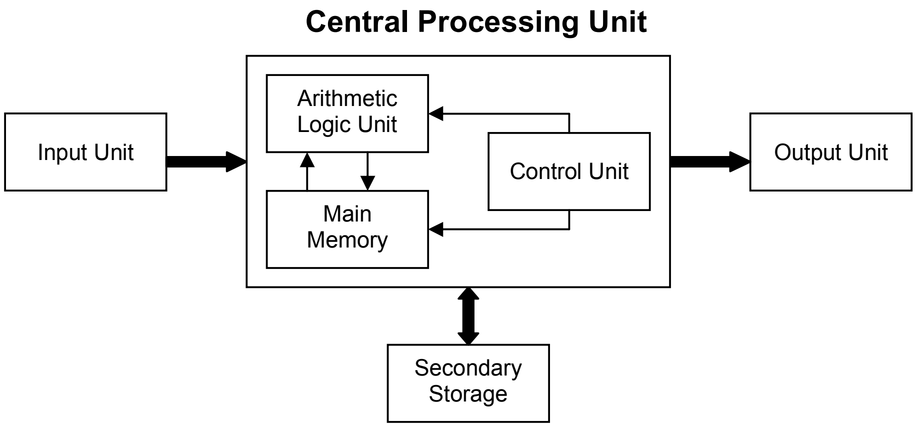


Figure 1 : Schematic Representation of a Computer

The objective of this chapter is to introduce the concept of Memory units of the computer which are shown in the above figure as main memory and secondary memory. The Memory unit is linked with other parts of the computer specifically as shown in the figure2.

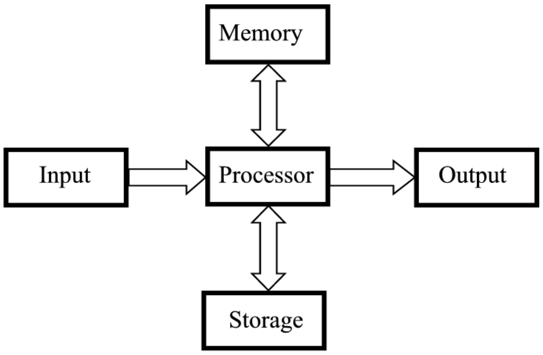


Figure 2: Linking Memory with other units

All storage devices are characterized with the following features:

- Speed
- Volatility
- Access method
- Portability
- Cost and capacity

1.1 Basic Units of Measurement

All information in the computer is handled using electrical components like the integrated circuits, semiconductors, all of which can recognize only two states – presence or absence of an electrical signal. Two symbols used to represent these two states are **0** and **1**, and are known as BITS (an abbreviation for **BI**nary Digi**TS**). **0** represents the absence of a signal, **1** represents the presence of a signal. A BIT is, therefore, the smallest unit of data in a computer and can either store a **0** or **1**.

Since a single bit can store only one of the two values, there can possibly be only four unique combinations:

00 01 10 11

Bits are, therefore, combined together into larger units in order to hold greater range of values.

BYTES are typically a sequence of eight bits put together to create a single computer alphabetical or numerical character. More often referred to in larger multiples, bytes may appear as **Kilobytes** (1,024 bytes), **Megabytes** (1,048,576 bytes), **GigaBytes** (1,073,741,824), **TeraBytes** (approx. 1,099,511,000,000 bytes), or **PetaBytes** (approx. 1,125,899,900,000,000 bytes).

Bytes are used to quantify the amount of data digitally stored (on disks, tapes) or transmitted (over the internet), and are also used to measure the memory and document size.

II. RAM,ROM,PROM,EPROM

The Term Computer Memory is defined as one or more sets of chips that store Data/program instructions, either temporarily or permanently. It is critical processing component in any computer. The PCs use several different types. They are :

- Main Memory / Primary Memory units
  - –Two most important are
    - RAM(Random Access Memory)
    - ROM(Read-only Memory)
  - –They work in different ways and perform distinct functions
  - –CPU Registers
  - –Cache Memory
- Secondary Memory/Auxiliary Memory
  Also termed as 'auxiliary' or 'backup' storage, it is typically used as a supplement to main storage. It is much cheaper than the main storage and stores large amount of data and instructions permanently. Hardware devices like magnetic tapes and disks fall under this category.

Computer's memory can be classified into two types – RAM and ROM.

RAM or Random Access Memory is the central storage unit in a computer system. It is the place in a computer where the operating system, application programs and the data in current use are kept temporarily so that they can be accessed by the computer's processor. The more RAM a computer has, the more data a computer can manipulate.

Random access memory, also called the Read/Write memory, is the temporary memory of a computer. It is said to be 'volatile' since its contents are accessible only as long as the computer is on. The contents of RAM are cleared once the computer is turned off.

ROM or Read Only Memory is a special type of memory which can only be read and contents of which are not lost even when the computer is switched off. It typically contains manufacturer's instructions. Among other things, ROM also stores an initial program called the 'bootstrap loader' whose function is to start the computer software operating, once the power is turned on.

Read-only memories can be **manufacturer-programmed** or **user-programmed**. While manufacturer-programmed ROMs have data burnt into the circuitry, user-programmed ROMs can have the user load and then store read-only programs. PROM or Programmable ROM is the name given to such ROMs.

Information once stored on the ROM or PROM chip cannot be altered. However, another type of memory called EPROM (Erasable PROM) allows a user to erase the information stored on the chip and reprogram it with new information. EEPROM (Electrically EPROM) and UVEPROM (Ultra Violet EPROM) are two types of EPROM's.

Storage Vs. Memory

RAM is volatile memory having a limited storage capacity. Secondary/auxiliary storage is storage other than the RAM. These include devices that are peripheral and are connected and controlled by the computer to enable permanent storage of programs and data.

Magnetic medium was found to be fairly inexpensive and long lasting medium and, therefore, became the preferred choice for auxiliary storage. Floppy disks and hard disks fall under this category. The newer forms of storage devices are **optical storage**

devices like CDs, DVDs, Pen drive, Zip drive etc.

To Summaries:

The memory is specifically meaning the RAM. This keeps the information for a shorter period of time (usually volatile), is faster and more expensive.
By Storage we mean the Hard disk. Here the information is retained longer (non-volatile),It's Slower and Cheaper

III.Auxiliary Storage Devices-Magnetic Tape, Floppy Disk, Hard Disk.

The Magnetic Storage Exploits duality of magnetism and electricity. It converts electrical signals into magnetic charges, captures magnetic charge on a storage medium and then later regenerates electrical current from stored magnetic charge. Polarity of magnetic charge represents bit values zero and one.

Magnetic Disk

The Magnetic Disk is Flat, circular platter with metallic coating that is rotated beneath read/write heads. It is a Random access device; read/write head can be moved to any location on the platter.

Floppy Disk

These are small removable disks that are plastic coated with magnetic recording material. Floppy disks are typically 3.5″ in size (diameter) and can hold 1.44 MB of data. This portable storage device is a rewritable media and can be reused a number of times.
Floppy disks are commonly used to move files between different computers. The main disadvantage of floppy disks is that they can be damaged easily and, therefore, are not very reliable. The following figure shows an example of the floppy disk. Figure 3 shows a picture of the floppy d

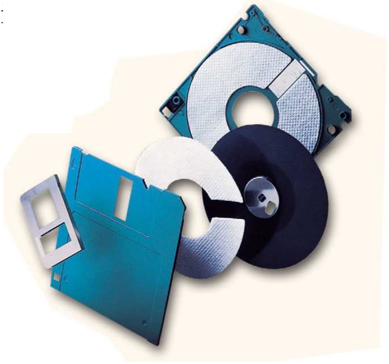


Figure 3: Floppy Disk

HARD DISK

Another form of auxiliary storage is a hard disk. A hard disk consists of one or more rigid metal plates coated with a metal oxide material that allows data to be magnetically recorded on the surface of the platters. The hard disk platters spin at a high rate of speed, typically 5400 to 7200 revolutions per minute (RPM).Storage capacities of hard disks for personal computers range from 10 GB to 120 GB (one billion bytes are called a gigabyte).

Figure 4: Hard Disk

IV.Optical Disks: CD-R Drive, CD-RW disks, DVD, Blue ray Discs

Optical Mass Storage Devices Store bit values as variations in light reflection. They have higher area density & longer data life than magnetic storage. They are also Standardized and relatively inexpensive. Their Uses: read-only storage with low performance requirements, applications with high capacity requirements & where portability in a standardized format is needed.

Example of the Optical Drives

- CD's (Compact Disk)

Their storage:

~ 700 MB storage

Their Types:

- CD-ROM (read only)
- CD-R: (record) to a CD
- CD-RW: can write and erase CD to reuse it (re-writable)
- DVD(Digital Video Disk)

CD:

Compact Disk (CD) is portable disk having data storage capacity between 650-700 MB. It can hold large amount of information such as music, full-motion videos, and text etc. It contains digital information that can be read, but cannot be rewritten. Separate drives exist for reading and writing CDs.

Since it is a very reliable storage media, it is very often used as a medium for distributing large amount of information to large number of users. In fact today most of the software is distributed through CDs.

DVD

Digital Versatile Disk (DVD) is similar to a CD but has larger storage capacity and enormous clarity. Depending upon the disk type it can store several Gigabytes of data (as opposed to around 650MB of a CD). DVDs are primarily used to store music or movies and can be played back on your television or the computer too. They are not rewritable media. Its also termed DVD (Digital Video Disk)

DVD-ROM

- Over 4 GB storage (varies with format)
- DVD- ROM (read only)
- Many recordable formats (e.g., DVD-R, DVD-RW; ..)
- Are more highly compact than a CD.
- Special laser is needed to read them

Blu-ray Technology

The name is derived from the blue-violet laser used to read and write data. It was developed by the Blu-ray Disc Association with more than 180 members. Some companies with the technology are Dell, Sony, LG.The Data capacity is very large because Blu-ray uses a blue laser(405 nanometers) instead of a red laser(650 nanometers) this allows the data tracks on the disc to be very compact. This allows for more than twice as small pits as on a DVD. Because of the greatly compact data Blu-ray can hold almost 5 times more data than a single layer DVD. Close to 25 GB!.Just like a DVD Blu-ray can also be recorded in Dual-Layer format. This allows the disk to hold up to 50 GB!!

The Variations in the formats are as follows:

- BD-ROM (read-only) - for pre-recorded content
- BD-R (recordable) - for PC data storage
- BD-RW (rewritable) - for PC data storage
- BD-RE (rewritable) - for HDTV recording

Summary:

The chapter has introduced the concept of memory units and its categories as the main and the second memory and their characteristic features. It discusses the features of RAM, ROM, PROM, and EPROM. Auxiliary Storage Devices like -Magnetic Tape, Hard Disk, Floppy Disk are also discussed .The chapter concludes with an introduction to the Optical Disks such as CD-R Drive,CD-RW disks,DVD,Blue ray Discs

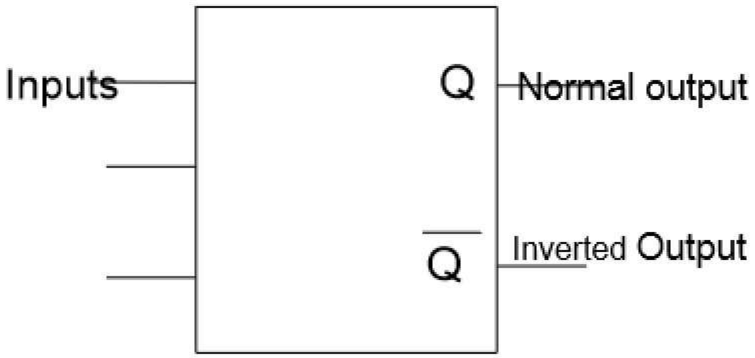
Design Engineering

FLIP-FLOPS

Flip Flop (Sequential Circuits)

- What is Flip flop?
- Answer: In digital circuits, the flip-flop, is a kind of bi-stable multivibrator.
- It is a Sequential Circuits / an electronic circuit which has two stable states and thereby is capable of serving as one bit of memory , bit 1 or bit 0.

Introduction – Flip Flop



They have two stable conditions and can be switched from one to the other by appropriate inputs. These stable conditions are usually called the states of the circuit.

- They are 1 (HIGH) or 0 (LOW).
- Whenever we refer to the state of flip flop, we refer to the state of its normal output (Q).
- More complicated Flip flop use a clock as the control input. These clocked flip-flops are used whenever the input and output signals must occur within a particular sequence.

Introduction: Types Of Flip Flop

1. SR Flip Flop
   a.SR Flip Flop Active Low = NAND gates
   b. SR Flip Flop Active High = NOR gates
2. Clocked SR Flip Flop
3. JK Flip Flop
4. JK Flip Flop With Pre-set And Clear
5. T Flip Flop
6. D Flip Flop
7. Master-Slave Edge-Triggered Flip-Flop

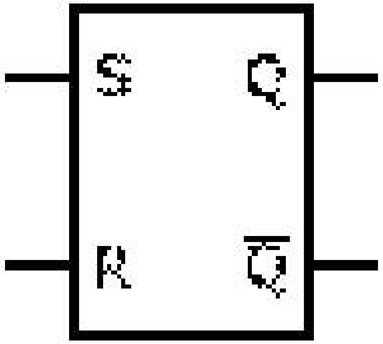
The Used of Flip Flop

- For Memory circuits
- For Logic Control Devices
- For Counter Devices
- For Register Devices

SR Flip Flop

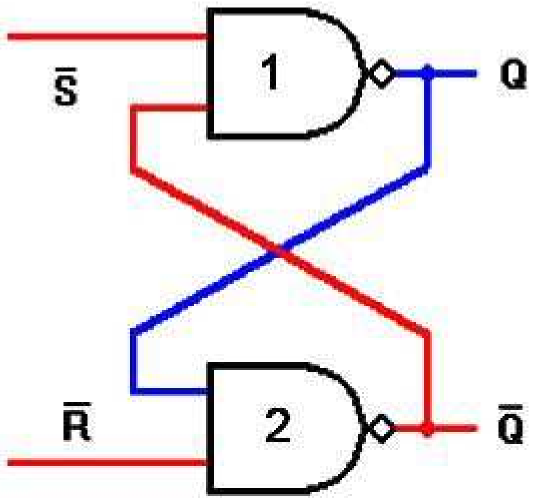
- The most basic Flip Flop is called SR Flip Flop.
- The basic RS flip flop is an asynchronous device.
- In asynchronous device, the outputs is immediately changed anytime one or more of the inputs change just as in combinational logic circuits.
- It does not operate in step with a clock or timing.
- These basic Flip Flop circuit can be constructed using two NAND gates latch or two NOR gates latch.
- SR Flip Flop Active Low SR Flip Flop Active Low SR Flip Flop Active Low SR Flip Flop Active Low = NAND gates
- SR Flip Flop Active High SR Flip Flop Active High SR Flip Flop Active High SR Flip Flop Active High = NOR gates

SR Flip Flop



- The SR Flip Flop has two inputs, SET (S) and RESET (R).
- The SR Flip Flop has two outputs, Q and ‾
- The Q output is considered the normal output and is the one most used.
- The other output ‾ is simply the compliment of output Q.

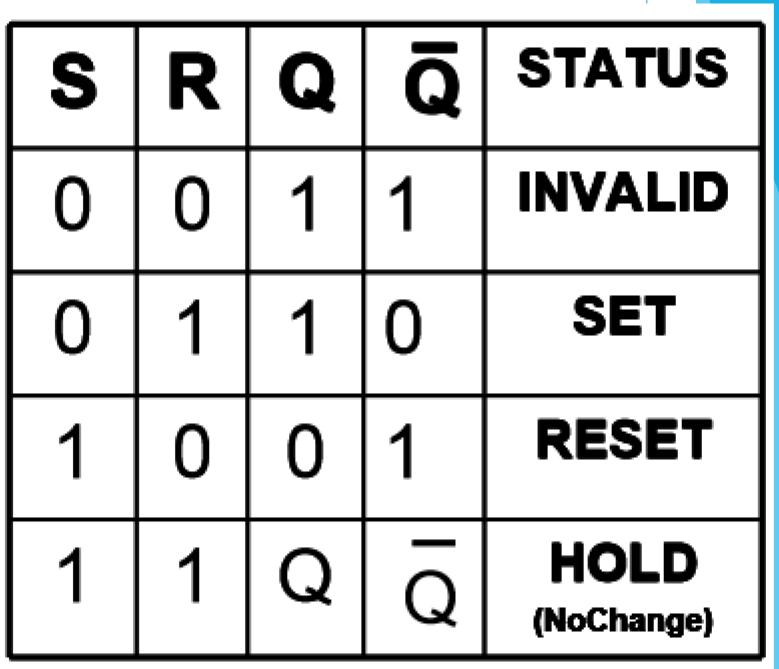
SR Flip Flop - NAND GATE LATCH



- The NAND gate version has two inputs, SET (S) and RESET (R).
- Two outputs, Q as normal output and ‾ as inverted output and feedback mechanism.
- The feedback mechanism is required to form a sequential circuit by connecting the output of NAND-1 to the input of NAND-2 and vice versa.
- The circuit outputs depends on the inputs and also on the outputs.

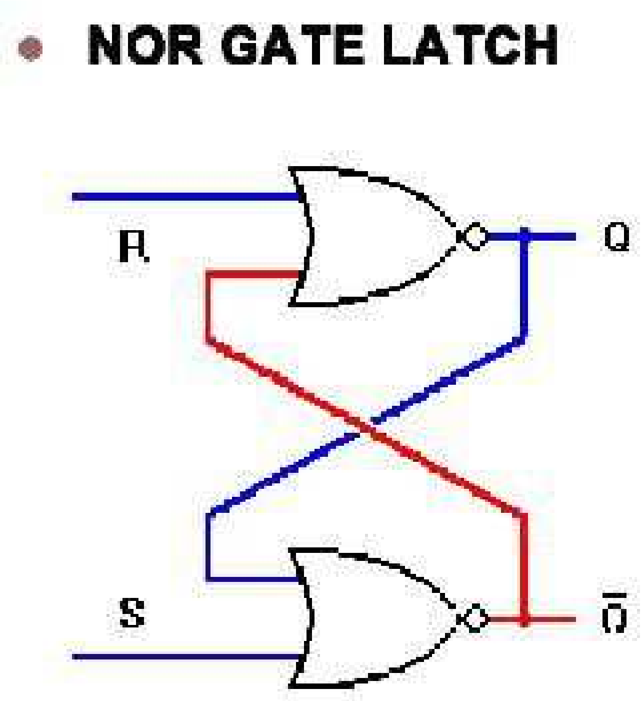
SR Flip Flop - NAND GATE LATCH

- From the description of the NAND gate latch operation, it shows that the SET and RESET inputs are active LOW.
- The SET input will set Q = 1 when SET is 0 (LOW).RESET input will reset Q = 0 when RESET is 0 (LOW)
- In the prohibited/INVALID state both outputs are 1. This condition is not used on the RS flip -flop. The set condition means setting the output Q to 1.
- Likewise, the reset condition means resetting (clearing) the output Q to 0. The last row shows the disabled, or hold , condition of the RS flip-flop. The outputs remain as they were before the hold condition existed. There is no change in the outputs from the previous states



S	R	Q	$\bar{Q}$	STATUS
0	0	1	1	INVALID
0	1	1	0	SET
1	0	0	1	RESET
1	1	Q	$\bar{Q}$	HOLD (NoChange)

SR Flip Flop - NOR GATE LATCH NOR GATE LATCH NOR GATE LATCH NOR GATE LATCH

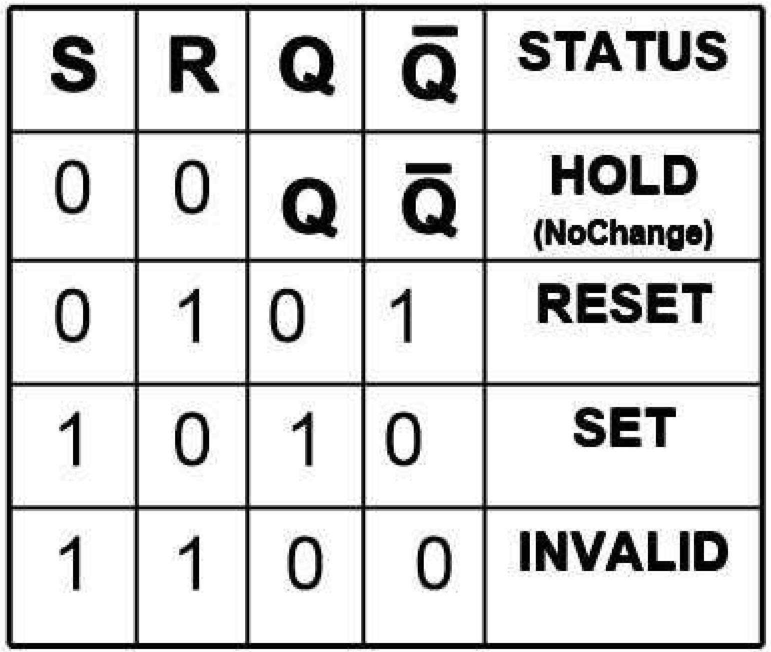


- The latch circuit can also be constructed using two NOR gates latch.
- The construction is similar to the NAND latch except that the normal output Q and inverted output ¯ have reversed positions.

SR Flip Flop - NOR GATE LATCH

- S = 1, R = 0; This will set Q to 1, it works in SET mode operation.
- S = 1, R = 1; This condition tries to set and reset the NOR gate latch at the same time, and it produces Q = ¯ = 0 This is an unexpected condition and are not used.
- Since the two outputs should be inverse of each other. If the inputs are returned to 1 simultaneously, the output states are unpredictable.
- This input condition should not be used and when circuits are constructed, the design should make this condition SET=RESET = 1 never arises

SR Flip Flop - NOR GATE LATCH

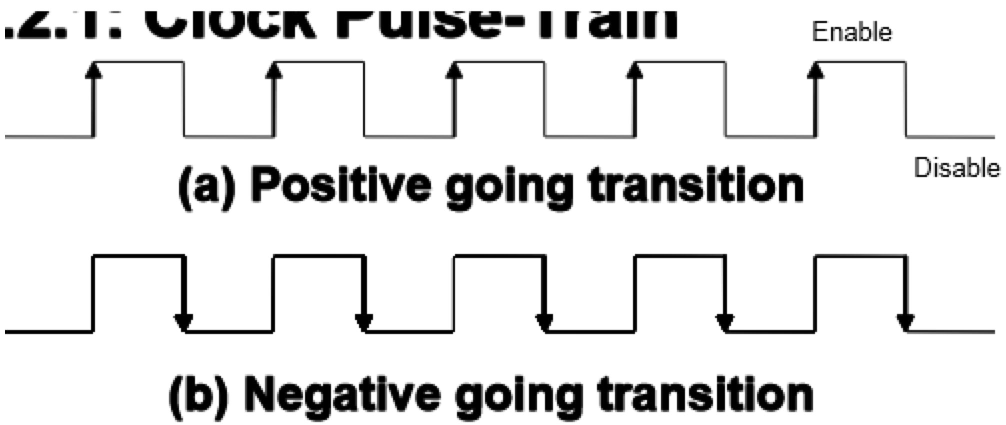


S	R	Q	$\bar{Q}$	STATUS
0	0	Q	$\bar{Q}$	HOLD (NoChange)
0	1	0	1	RESET
1	0	1	0	SET
1	1	0	0	INVALID

- From the description of the NOR gate latch operation, it shows that the SET and RESET inputs are Active HIGH.
- The SET input will set Q = 1 when SET is 1 (HIGH). RESET input will reset Q when RESET is 1 (HIGH).

The CLOCK

- When the clock changes from a LOW state to a HIGH state, this is called the positive-going transition (PGT) or positive edge triggered.
- When the clock changes from a HIGH state to a LOW state, it is called negative going transition (NGT) or negative edge triggered.

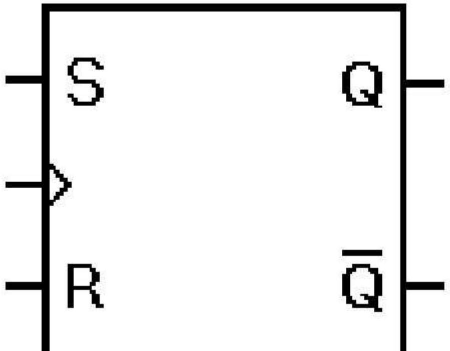

.2.1: Clock Pulse-Train

(a) Positive going transition

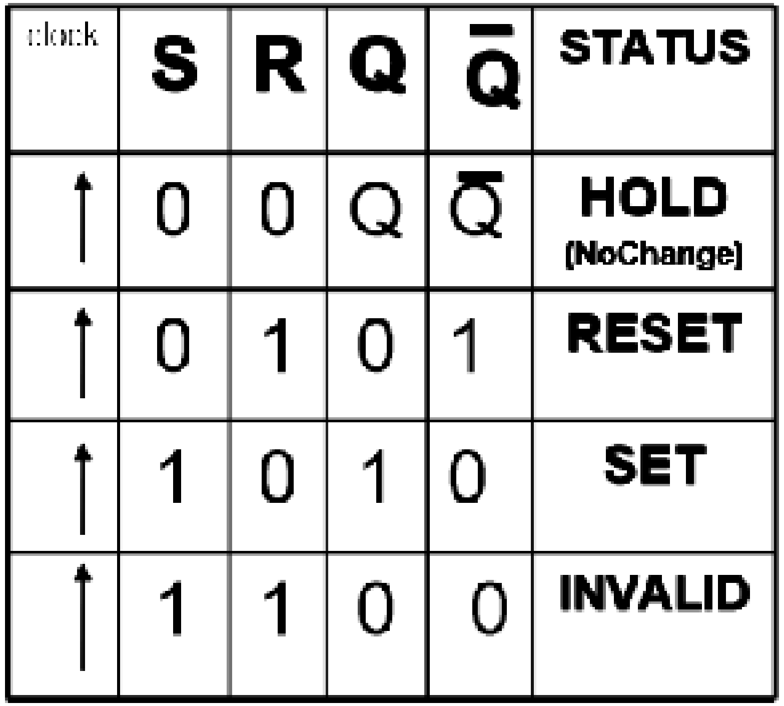
(b) Negative going transition

Clocked SR Flip Flop

- Additional clock input is added to change the SR flip- flop from an element used in asynchronous sequential circuits to one, which can be used in synchronous circuits.
- The clocked SR flip flop logic symbol that is triggered by the PGT is shown in Figure.
- Its means that the flip flop can change the output states only when clock signal makes a transition from LOW to HIGH.



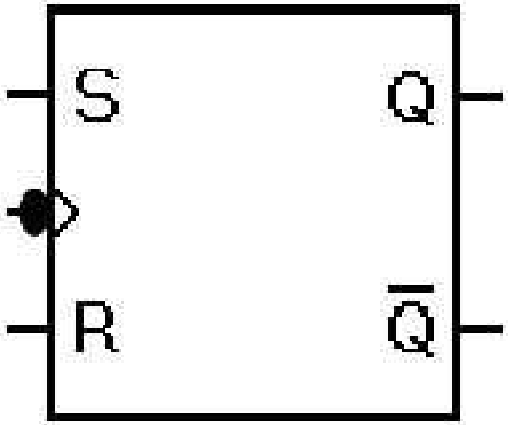
Clocked RS Flip Flop



clock	S	R	Q	$\bar{Q}$	STATUS
↑	0	0	Q	$\bar{Q}$	HOLD (NoChange)
↑	0	1	0	1	RESET
↑	1	0	1	0	SET
↑	1	1	0	0	INVALID

- The Truth Table in figure shows how the flip flop output will respond to the PGT at the clocked input for the various combinations of SR inputs and output.
- The up arrow symbol indicates PGT.

Clocked SR Flip Flop

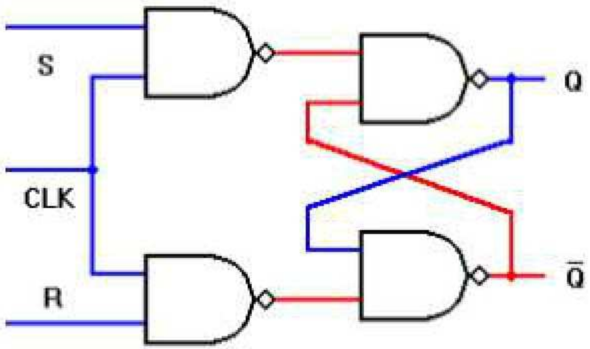


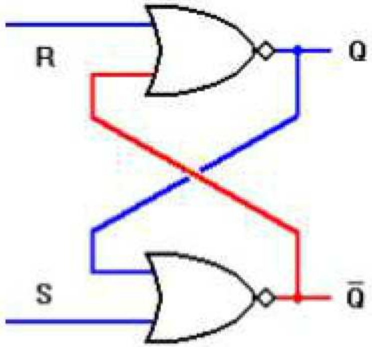
- The clocked SR Flip Flop logic symbol that is triggered by the NGT is shown in Figure.
- It means that the Flip flop can change the output states only when clocked signal makes a transition from HIGH to LOW

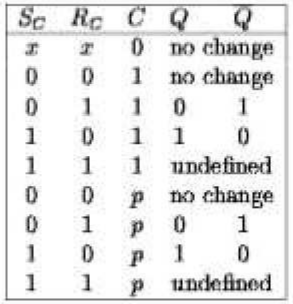
Clocked SR Flip Flop

- CLOCKED SR FLIP FLOP LOGIC CIRCUIT

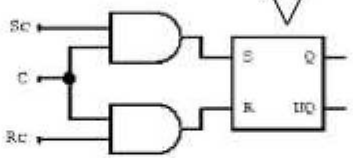
If used NOR Gate NOR Gate NOR Gate NOR Gate, must used AND Gate in front.

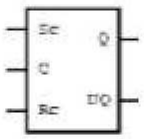






S_C	R_C	C	Q	Q
x	x	0	no change	
0	0	1	no change	
0	1	1	0	1
1	0	1	1	0
1	1	1	undefined	
0	0	p	no change	
0	1	p	0	1
1	0	p	1	0
1	1	p	undefined	

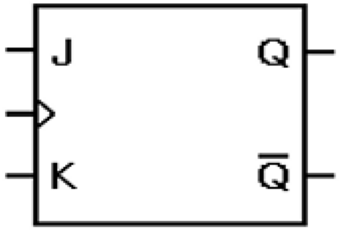




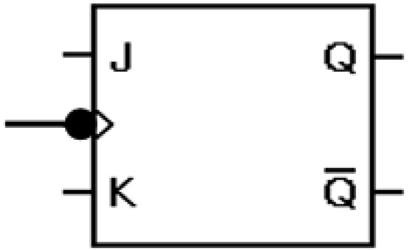
JK Flip Flop - Symbol

- Another types of Flip flop is JK flip flop.
- It differs from the RS flip flops when J=K=1 condition is not indeterminate but it is defined to give a very useful changeover (toggle) action.
- Toggle means that Q and ¯ will switch to their opposite states.
- The JK Flip flop has clock input Cp and two control inputs J and K.
- Operation of Jk Flip Flop is completely described by truth table in Figure.

- Figure 4.3.1 : **PGT JK Flip flop symbol**

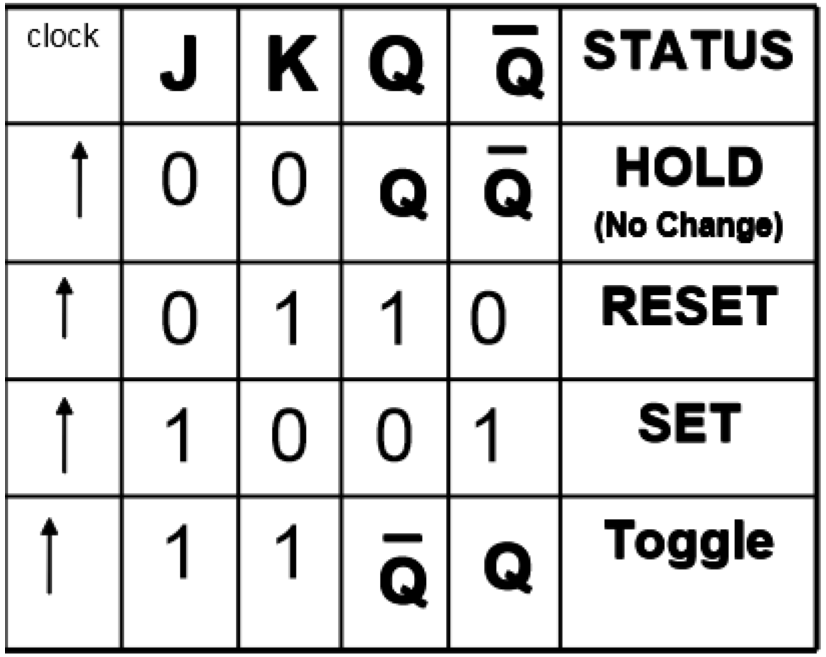


- Figure 4.3.2 : **NGT JK Flip flop symbol**



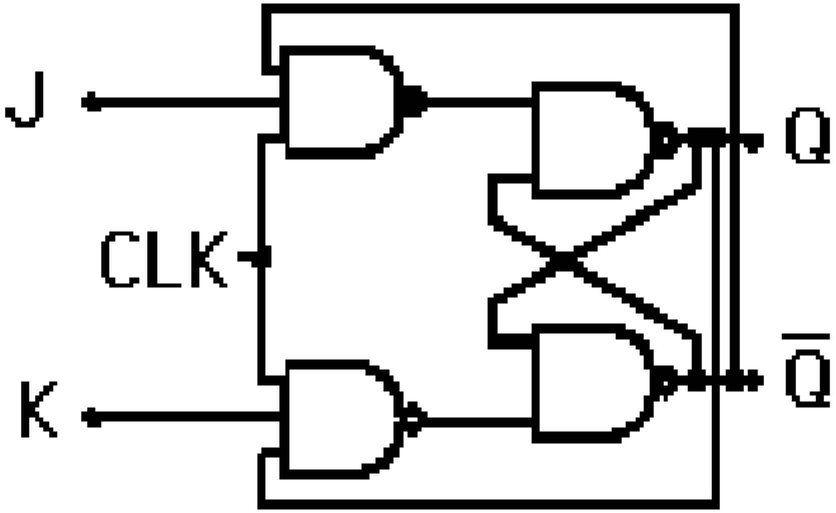
JK Flip Flop – Truth Table And Logic Circuit

Figure 4.3.3: **Truth Table for JK Flip Flop**



clock	J	K	Q	$\overline{Q}$	STATUS
↑	0	0	Q	$\overline{Q}$	**HOLD** **(No Change)**
↑	0	1	1	0	**RESET**
↑	1	0	0	1	**SET**
↑	1	1	$\overline{Q}$	Q	**Toggle**

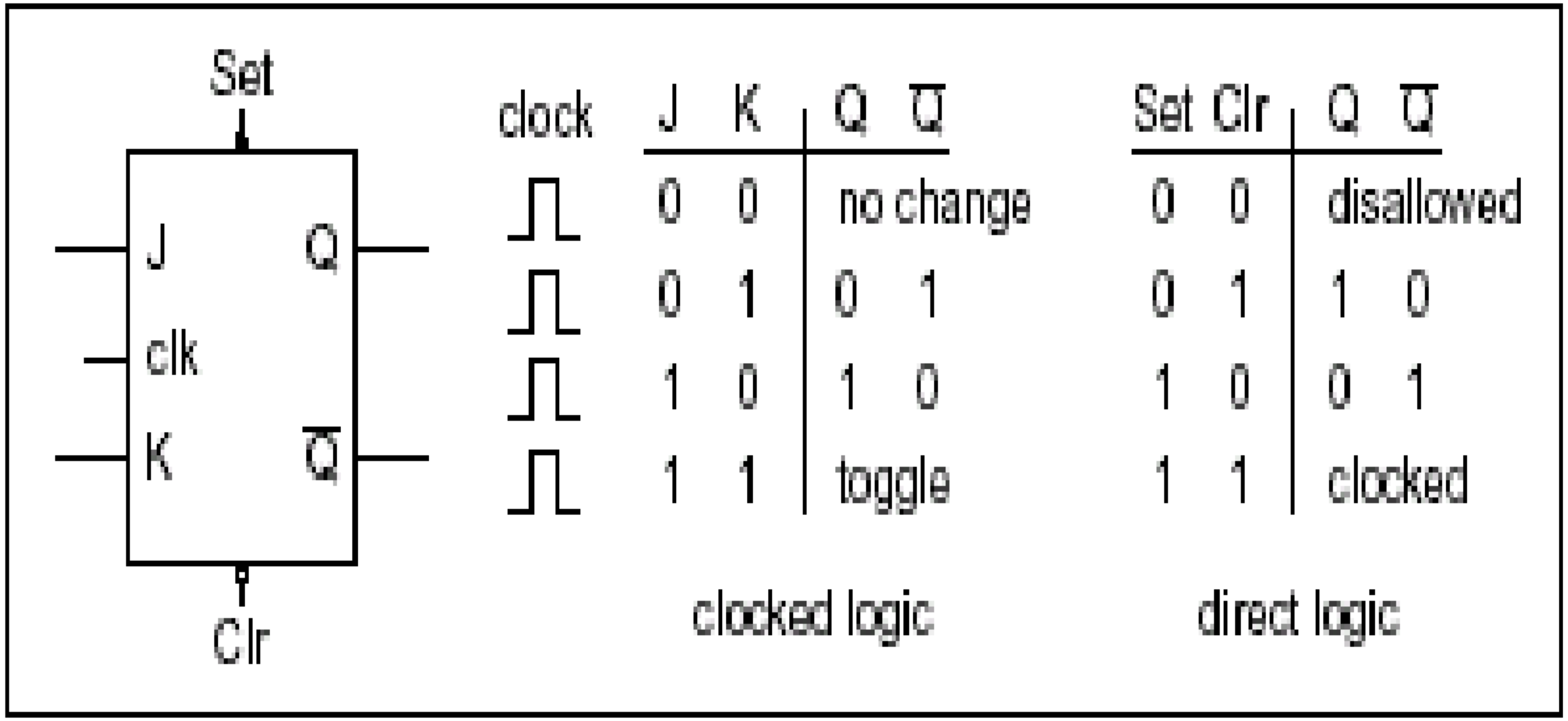
- Figure 4.3.4: **JK FLIP FLOP LOGIC CIRCUIT**



JK Flip Flop with Asynchronous Input

- The J and K inputs are called synchronous inputs since they only influence the state of the flip flop when the clocked pulse is present.
- This flip flop can also have other inputs called Pre-set (or SET) and clear that can be used for setting the flip flop to 1 or resetting it to 0 by applying the appropriate signal to the Pre-set and Clear inputs.
- These inputs can change the state of the flip flop regardless of synchronous inputs or the clock

JK Flip Flop with Preset and Clear



clock	J	K	Q	$\overline{Q}$
⎍	0	0	no change	
⎍	0	1	0	1
⎍	1	0	1	0
⎍	1	1	toggle	

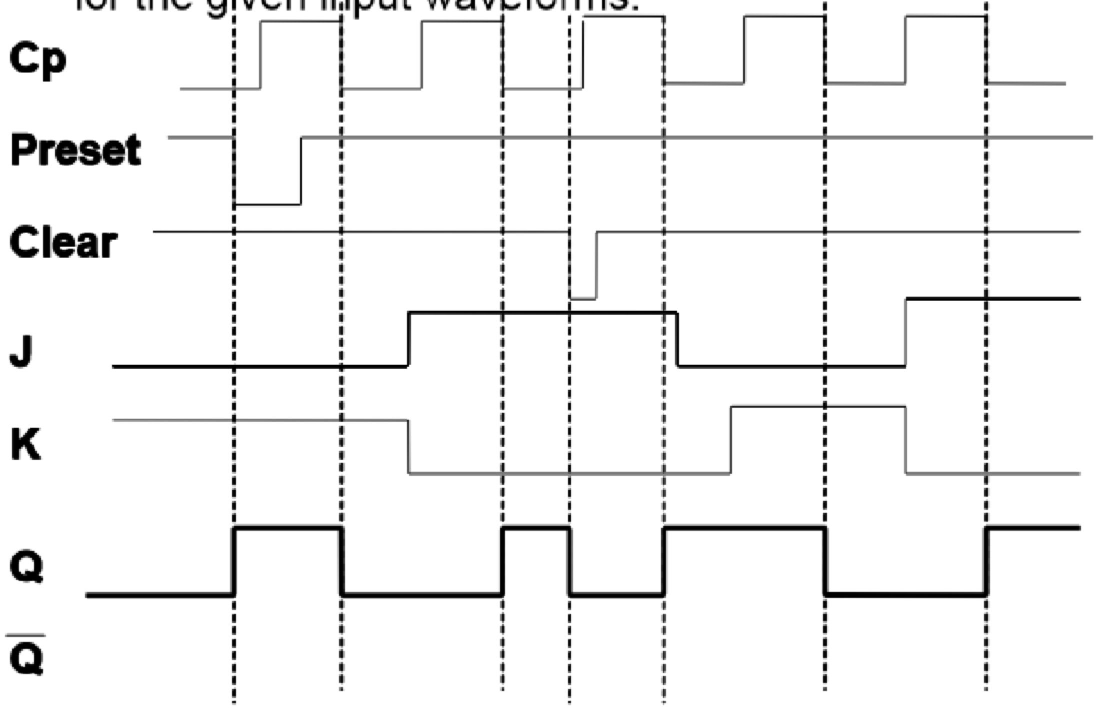
clocked logic

Set	Clr	Q	$\overline{Q}$
0	0	disallowed	
0	1	1	0
1	0	0	1
1	1	clocked	

direct logic

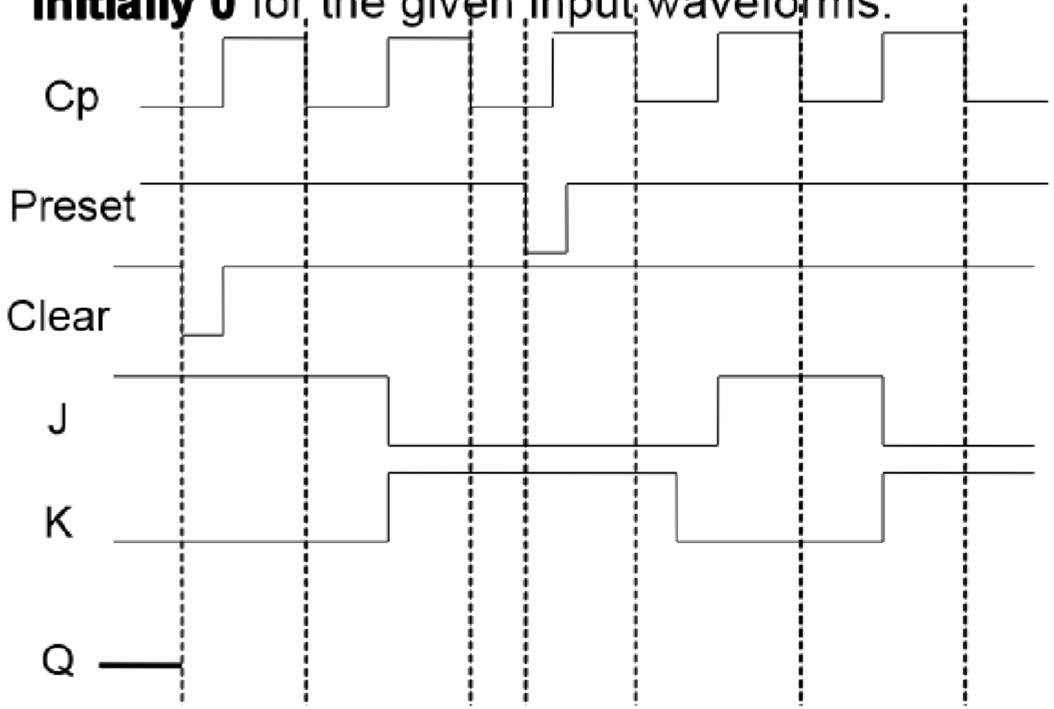
JK Flip Flop with Asynchronous Input

- **Example 4.4.2** : The output of clocked JK flip flop which output **initially 0** for the given input waveforms.



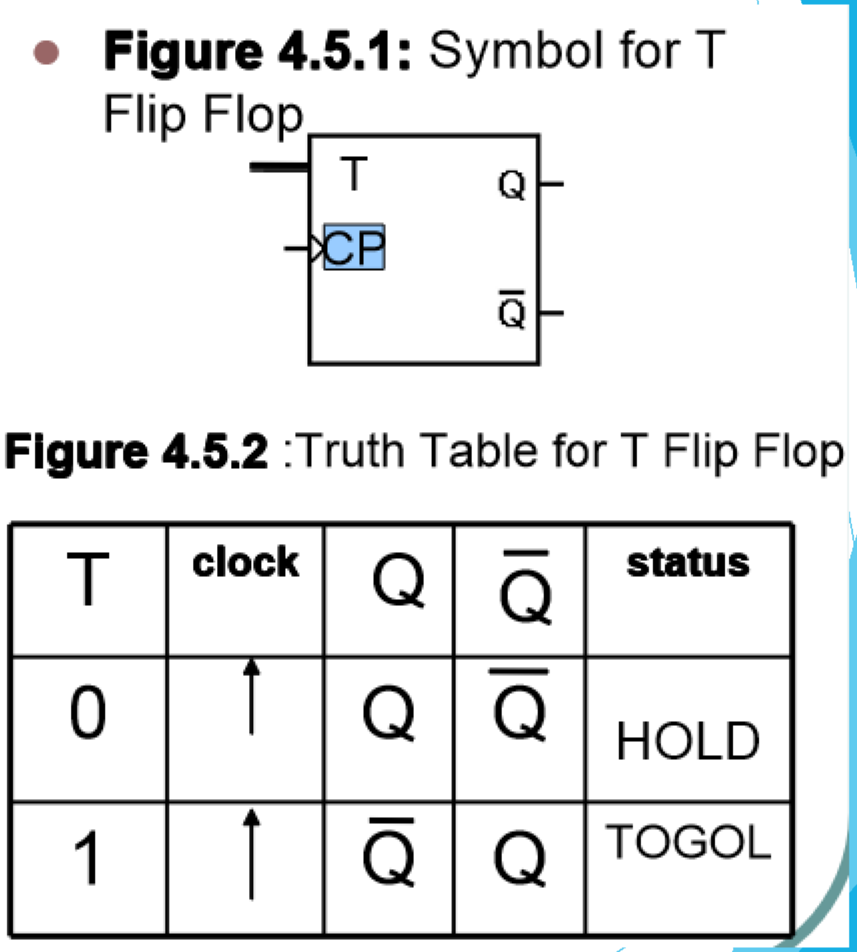
JK Flip Flop with Asynchronous Input

- **Exercise 4.4.3** : The output of clocked JK flip flop which output **initially 0** for the given input waveforms.



T Flip Flop - Symbol

- The T flip flop has only the Toggle and Hold Operation.
- If Toggle mode operation. The output will toggle from 1 to 0 or vice versa.



- **Figure 4.5.1:** Symbol for T Flip Flop

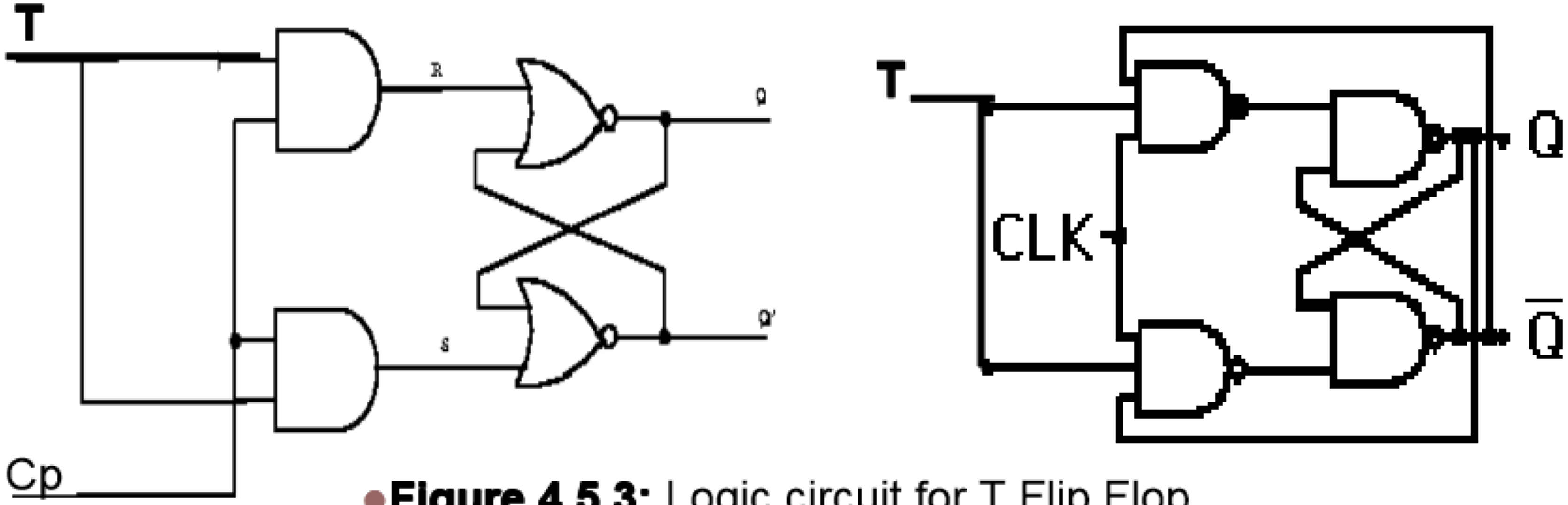
Figure 4.5.2 :Truth Table for T Flip Flop

T	clock	Q	$\overline{Q}$	status
0	↑	Q	$\overline{Q}$	HOLD
1	↑	$\overline{Q}$	Q	TOGOL

T Flip Flop – Logic Circuit

- Logic circuit T Flip flop using NOR gate

T Flip flop using NAND gate

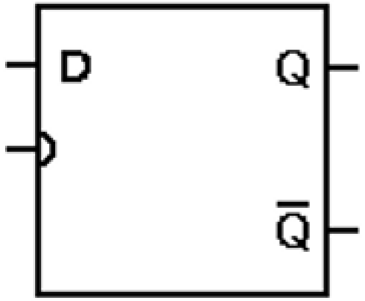


●**Figure 4.5.3:** Logic circuit for T Flip Flop

D Flip Flop

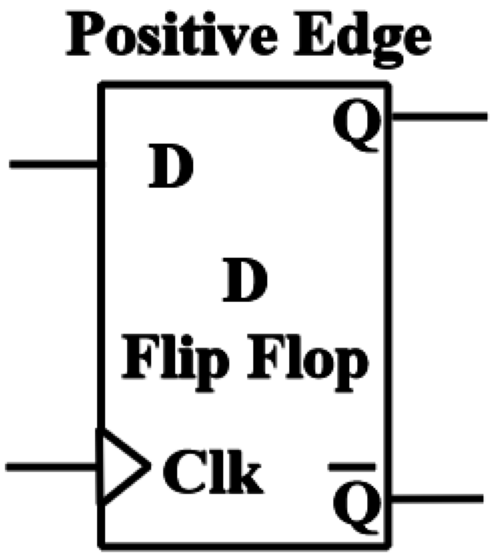
- Also Known as Data Flip flop
- Can be constructed from RS Flip Flop or JK Flip flop by addition of an inverter.
- Inverter is connected so that the R input is always the inverse of S (or J input is always complementary of K).
- The D flip flop will act as a storage element for a single binary digit (Bit).

- **Figure 4.6.1 :**
- **D Flip flop symbol**



D Flip Flop - Symbol

- PGT
- NGT



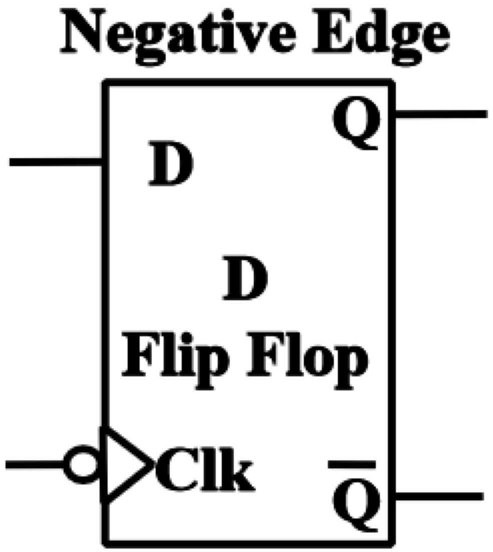
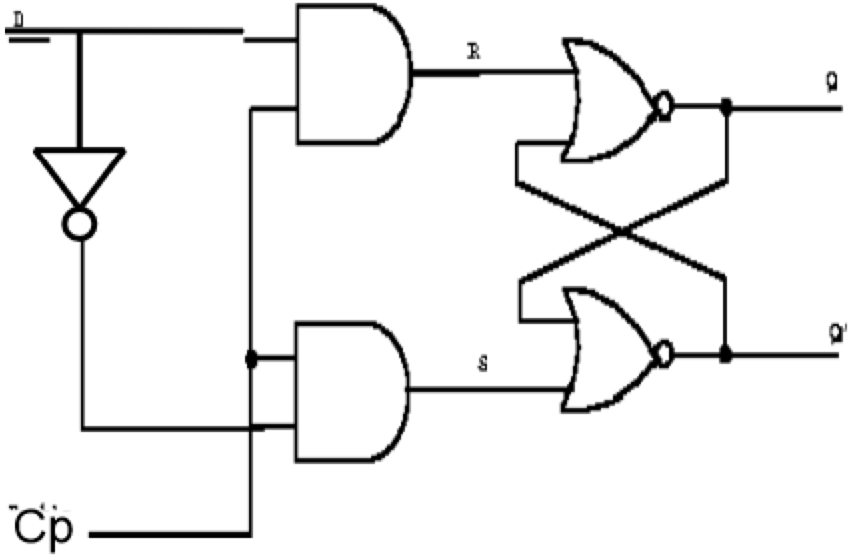


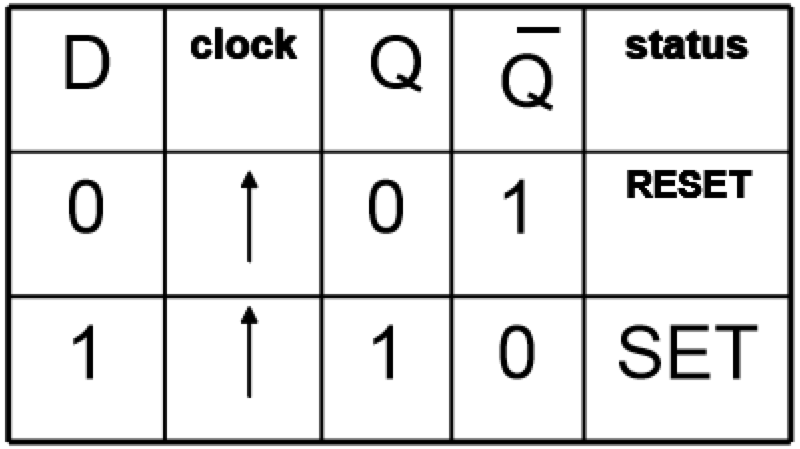
Figure 4.6.2 : D Flip flop symbol using JK Flip Flop / SR Flip Flop

D Flip Flop- Logic circuit-Truth Table

- **Figure 4.6.3:** Logic circuit for D Flip Flop



- **Figure 4.6.4:** Truth Table for D Flip Flop

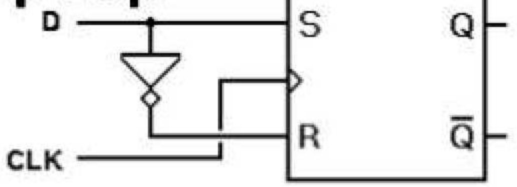


D	clock	Q	$\overline{Q}$	status
0	↑	0	1	RESET
1	↑	1	0	SET

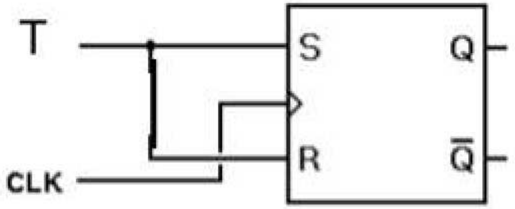
T Flip Flops and D Flip Flops can be Built using JK Flip Flop

- The JK flip flop is considered as a universal flip flop.
- A combination of JK flip flop and an inverter can construct a D Flip Flop as shown in Figure.
- It also can construct T Flip Flop by combine both J and K inputs with HIGH level input as shown in Figure.

- Figure 4.7.1 : **D Flip flop symbol using JK Flip Flop / SR Flip Flop**

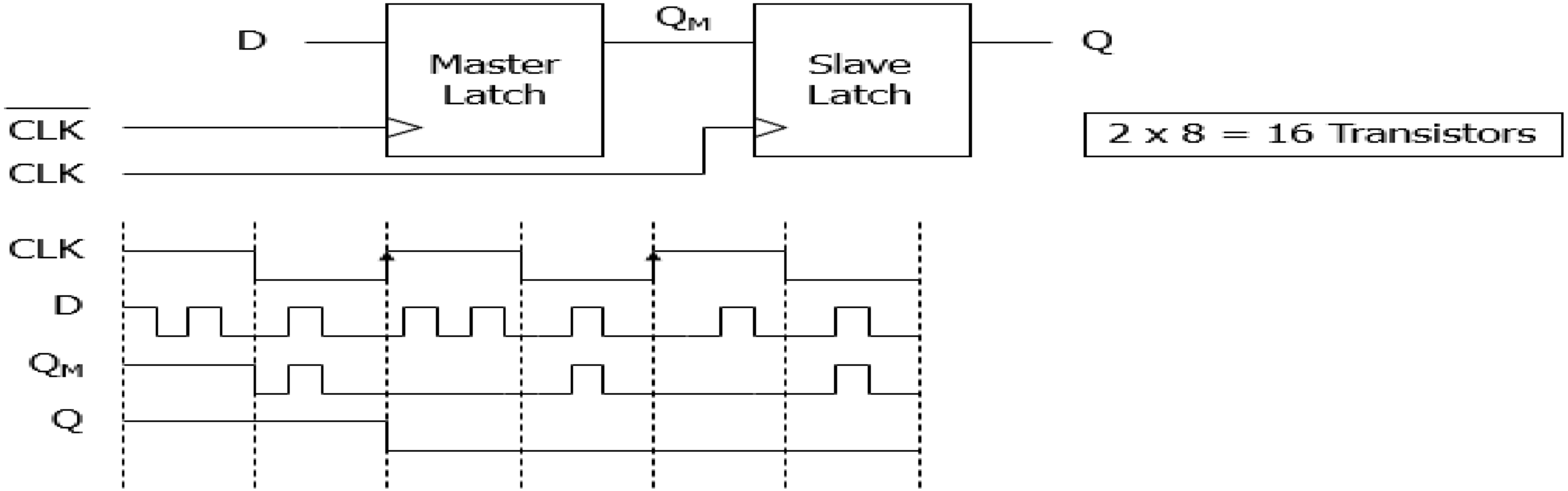


- Figure 4.7.2 : **T Flip flop symbol using JK Flip Flop / SR Flip Flop**

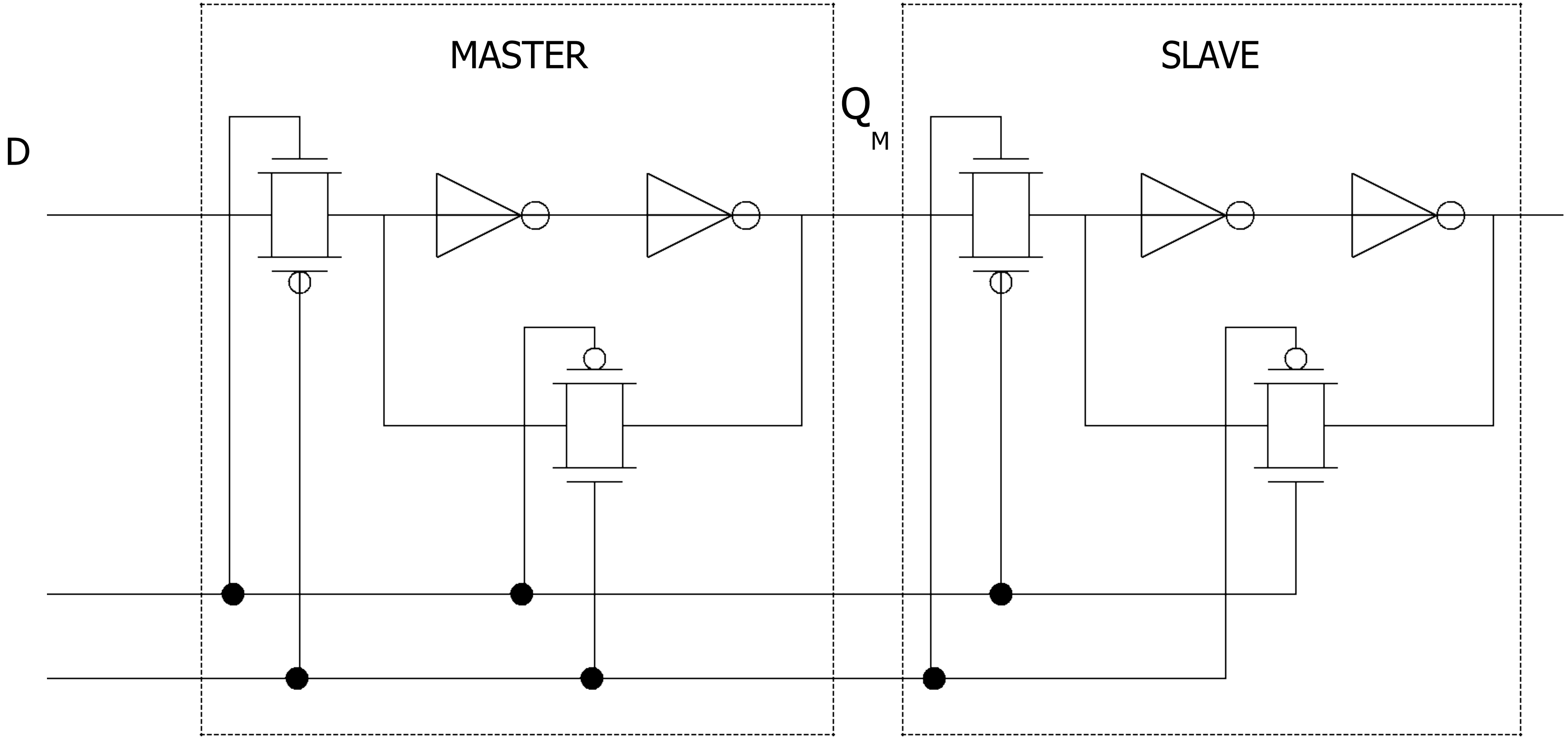


Master-Slave Edge-Triggered Flip-Flop

- Can connect two level-sensitive latches in Master-Slave configuration to form edge-triggered flip-flop.
- Master latch “catches” value of “D” at “Q_M” when CLK is low.
- Slave latch causes “Q” to change only at rising edge of CLK.

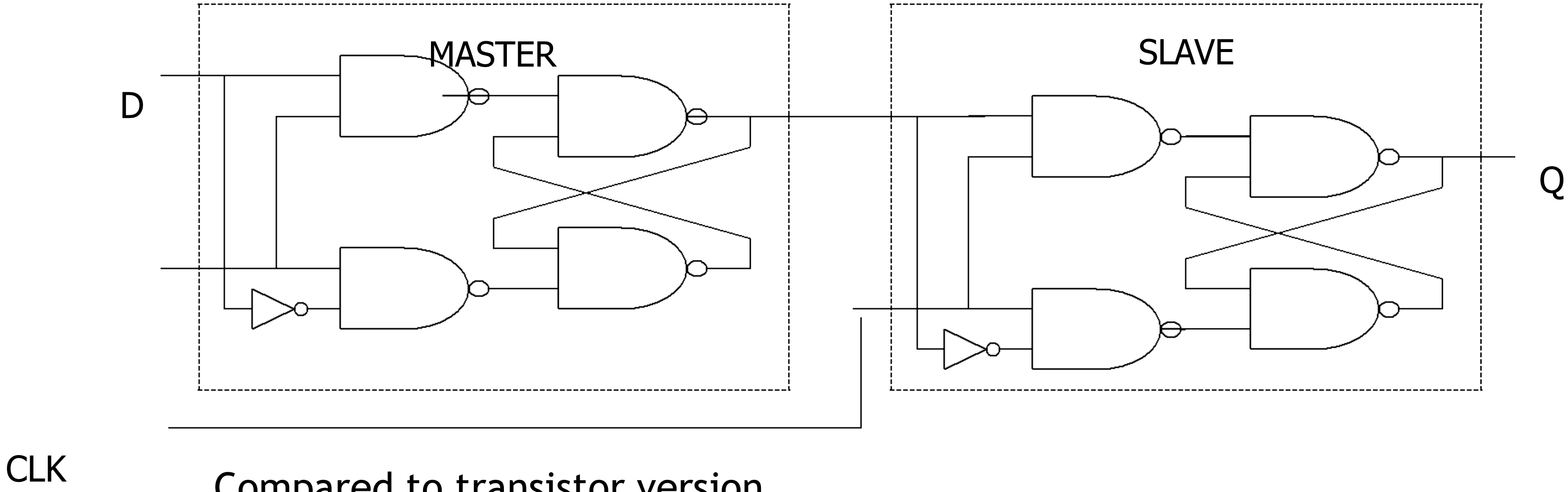


Master-Slave Edge-Triggered Flip-Flop



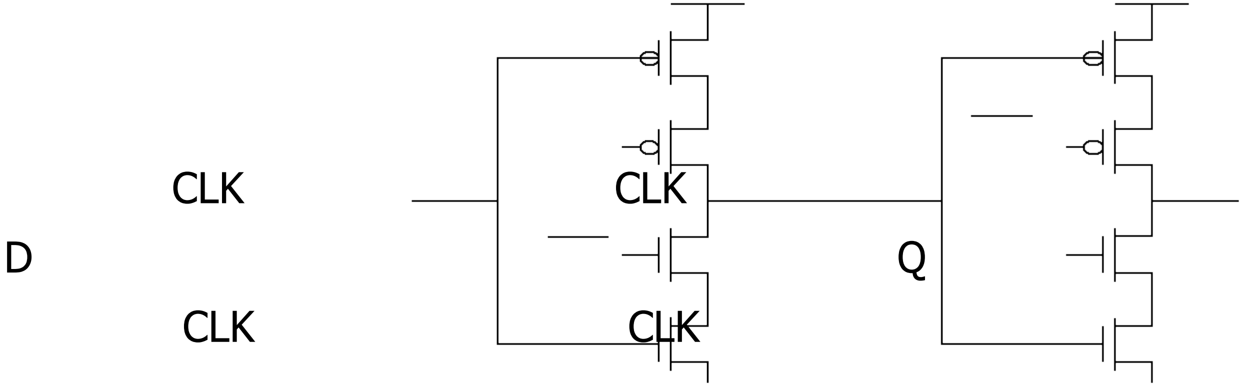
Master-Slave Edge-Triggered Flip-Flop

Master-Slave configuration



Compared to transistor version

36 Transistors



8 Transistors

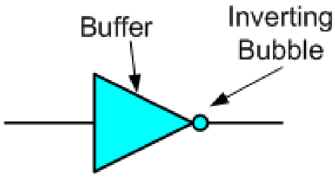
Universal Gates

Lesson Objectives:

In addition to AND, OR, and NOT gates, other logic gates like NAND and NOR are also used in the design of digital circuits.

The NOT circuit inverts the logic sense of a binary signal.

The small circle (bubble) at the output of the graphic symbol of a NOT gate is formally called a negation indicator and designates the logical complement.



The objectives of this lesson are to learn about:

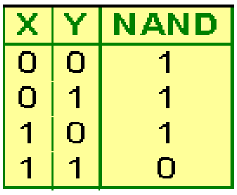
1. Universal gates - NAND and NOR.
2. How to implement NOT, AND, and OR gate using NAND gates only.
3. How to implement NOT, AND, and OR gate using NOR gates only.
4. Equivalent gates.
5. Two-level digital circuit implementations using universal gates only.
6. Two-level digital circuit implementations using other gates.

NAND Gate:

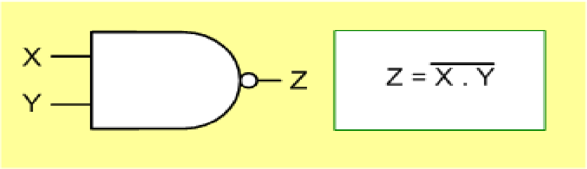
The **NAND** gate represents the complement of the AND operation. Its name is an abbreviation of NOT **AND**.

The graphic symbol for the NAND gate consists of an AND symbol with a bubble on the output, denoting that a complement operation is performed on the output of the AND gate.

The truth table and the graphic symbol of NAND gate is shown in the figure.



X	Y	NAND
0	0	1
0	1	1
1	0	1
1	1	0



$Z = \overline{X . Y}$

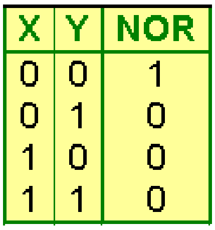
The truth table clearly shows that the NAND operation is the complement of the AND.

NOR Gate:

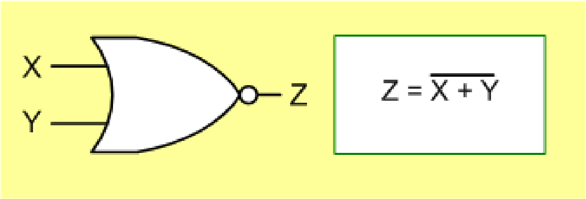
The **NOR** gate represents the complement of the OR operation. Its name is an abbreviation of NOT **OR**.

The graphic symbol for the NOR gate consists of an OR symbol with a bubble on the output, denoting that a complement operation is performed on the output of the OR gate.

The truth table and the graphic symbol of NOR gate is shown in the figure.



X	Y	NOR
0	0	1
0	1	0
1	0	0
1	1	0



The truth table clearly shows that the NOR operation is the complement of the OR.

Universal Gates:

A universal gate is a gate which can implement any Boolean function without need to use any other gate type.

The NAND and NOR gates are universal gates.

In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

In fact, an AND gate is typically implemented as a NAND gate followed by an inverter not the other way around!!

Likewise, an OR gate is typically implemented as a NOR gate followed by an inverter not the other way around!!

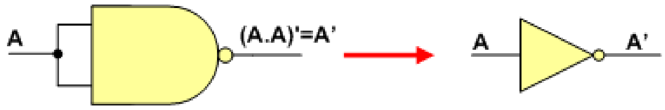
NAND Gate is a Universal Gate:

To prove that any Boolean function can be implemented using only NAND gates, we will show that the AND, OR, and NOT operations can be performed using only these gates.

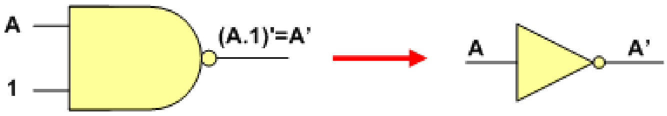
Implementing an Inverter Using only NAND Gate

The figure shows two ways in which a NAND gate can be used as **an inverter (NOT gate)**.

1. All NAND input pins connect to the input signal **A** gives an output **A'**.

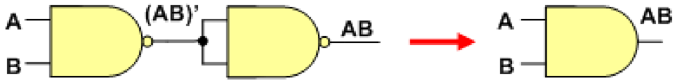


2. One NAND input pin is connected to the input signal **A** while all other input pins are connected to logic **1**. The output will be **A'**.



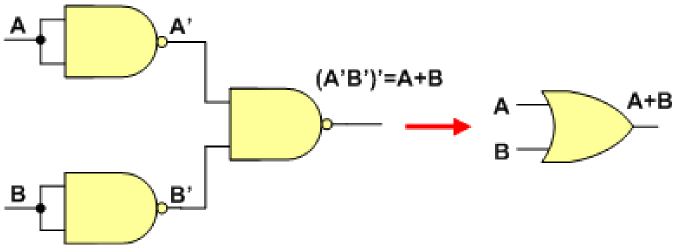
Implementing AND Using only NAND Gates

An AND gate can be replaced by NAND gates as shown in the figure (The AND is replaced by a NAND gate with its output complemented by a NAND gate inverter).



Implementing OR Using only NAND Gates

An OR gate can be replaced by NAND gates as shown in the figure (The OR gate is replaced by a NAND gate with all its inputs complemented by NAND gate inverters).



Thus, the NAND gate is a universal gate since it can implement the AND, OR and NOT functions.

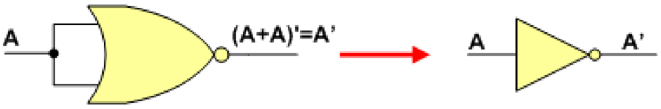
NAND Gate is a Universal Gate:

To prove that any Boolean function can be implemented using only NOR gates, we will show that the AND, OR, and NOT operations can be performed using only these gates.

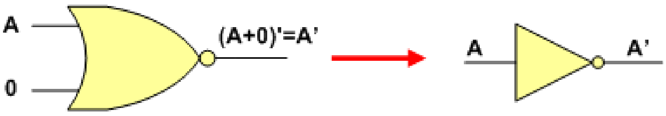
Implementing an Inverter Using only NOR Gate

The figure shows two ways in which a NOR gate can be used as **an inverter (NOT gate)**.

1. All NOR input pins connect to the input signal **A** gives an output **A'**.

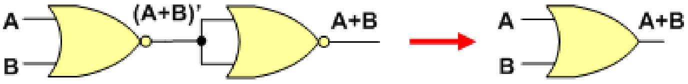


2. One NOR input pin is connected to the input signal **A** while all other input pins are connected to logic **0**. The output will be **A'**.



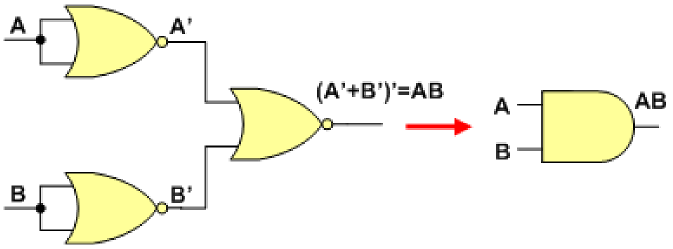
Implementing OR Using only NOR Gates

An OR gate can be replaced by NOR gates as shown in the figure (The OR is replaced by a NOR gate with its output complemented by a NOR gate inverter)



Implementing AND Using only NOR Gates

An AND gate can be replaced by NOR gates as shown in the figure (The AND gate is replaced by a NOR gate with all its inputs complemented by NOR gate inverters)

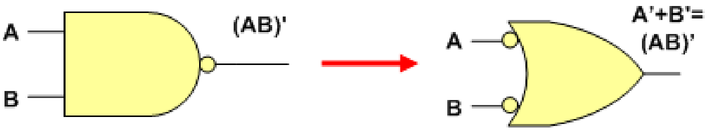


Thus, the NOR gate is a universal gate since it can implement the AND, OR and NOT functions.

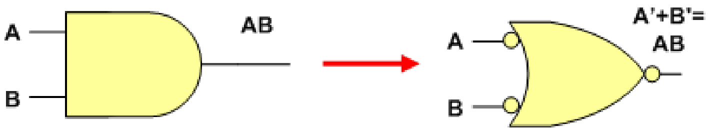
Equivalent Gates:

The shown figure summarizes important cases of gate equivalence. Note that bubbles indicate a complement operation (inverter).

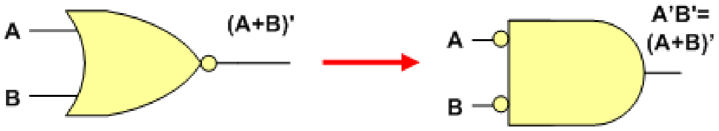
A NAND gate is equivalent to an inverted-input OR gate.



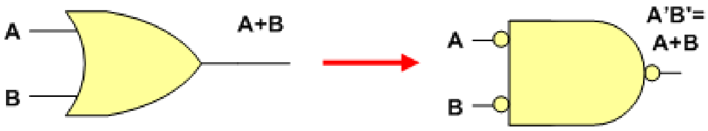
An AND gate is equivalent to an inverted-input NOR gate.



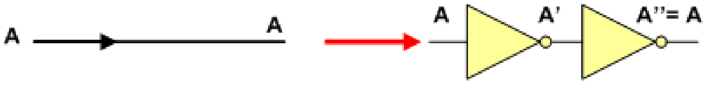
A NOR gate is equivalent to an inverted-input AND gate.



An OR gate is equivalent to an inverted-input NAND gate.



Two NOT gates in series are same as a buffer because they cancel each other as A’’ = A.



Two-Level Implementations:

We have seen before that Boolean functions in either SOP or POS forms can be implemented using 2-Level implementations.

For SOP forms AND gates will be in the first level and a single OR gate will be in the second level.

For POS forms OR gates will be in the first level and a single AND gate will be in the second level.

Note that using inverters to complement input variables is not counted as a level.

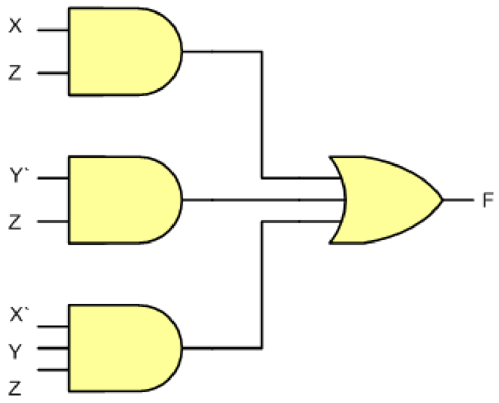
We will show that SOP forms can be implemented using only NAND gates, while POS forms can be implemented using only NOR gates.

This is best explained through examples.

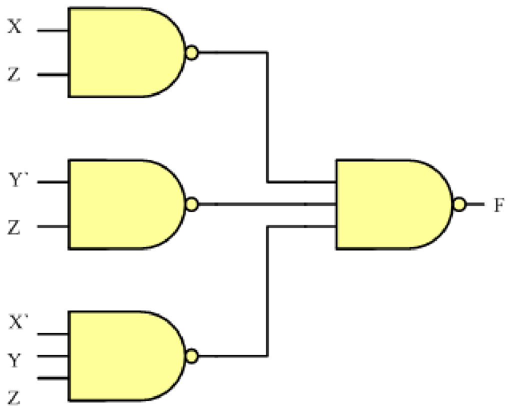
Example 1: Implement the following SOP function

$$F = XZ + Y'Z + X'YZ$$

Being an SOP expression, it is implemented in 2-levels as shown in the figure.



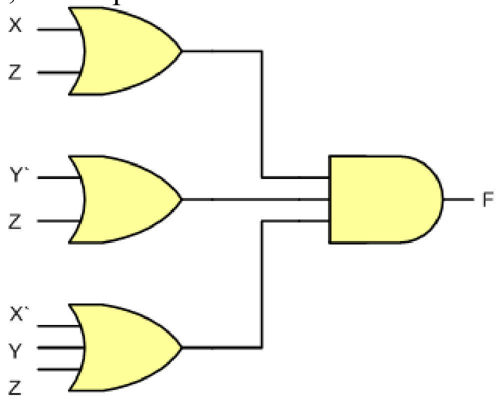
Introducing two successive inverters at the inputs of the OR gate results in the shown equivalent implementation. Since two successive inverters on the same line will not have an overall effect on the logic as it is shown before.
(see animation in authorware version)
By associating one of the inverters with the output of the first level AND gate and the other with the input of the OR gate, it is clear that this implementation is reducible to 2-level implementation where both levels are NAND gates as shown in Figure.



Example 2: Implement the following POS function

F = (X+Z) (Y’+Z) (X’+Y+Z)

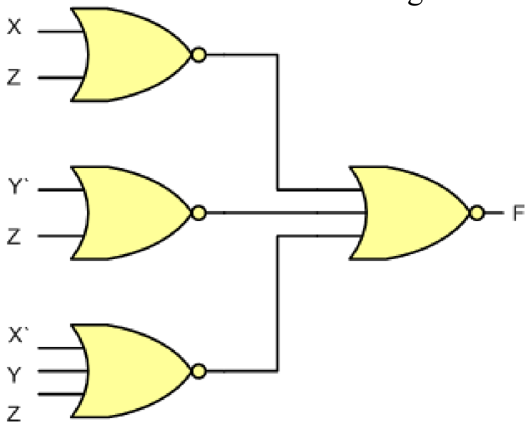
Being a POS expression, it is implemented in 2-levels as shown in the figure.



Introducing two successive inverters at the inputs of the AND gate results in the shown equivalent implementation. Since two successive inverters on the same line will not have an overall effect on the logic as it is shown before.

(see animation in authorware version)

By associating one of the inverters with the output of the first level OR gates and the other with the input of the AND gate, it is clear that this implementation is reducible to 2-level implementation where both levels are NOR gates as shown in Figure.



There are some other types of 2-level combinational circuits which are

- NAND-AND
- AND-NOR,
- NOR-OR,
- OR-NAND

These are explained by examples.

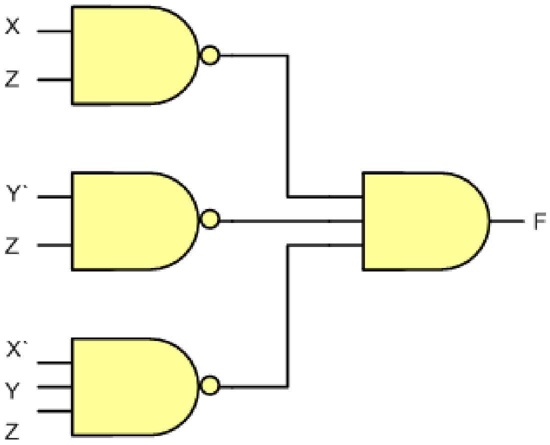
AND-NOR functions:

Example 3: Implement the following function

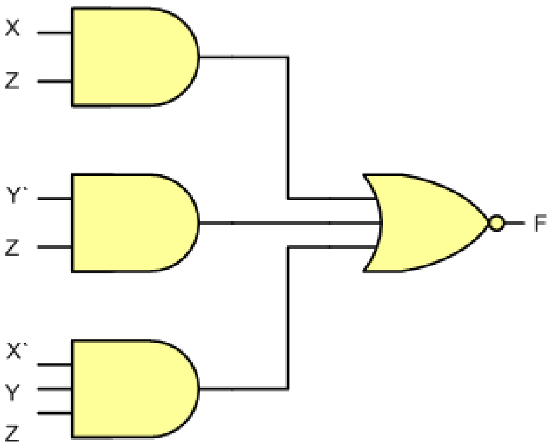
$F = \overline{XZ + \overline{Y}Z + \overline{X}YZ}$ **or**

$\boldsymbol{\overline{F} = XZ + \overline{Y}Z + \overline{X}YZ}$

Since **F'** is in SOP form, it can be implemented by using NAND-NAND circuit. By complementing the output we can get **F**, **or** by using ***NAND-AND*** circuit as shown in the figure.



It can also be implemented using ***AND-NOR*** circuit as it is equivalent to NAND-AND circuit as shown in the figure. (see animation in authorware version)



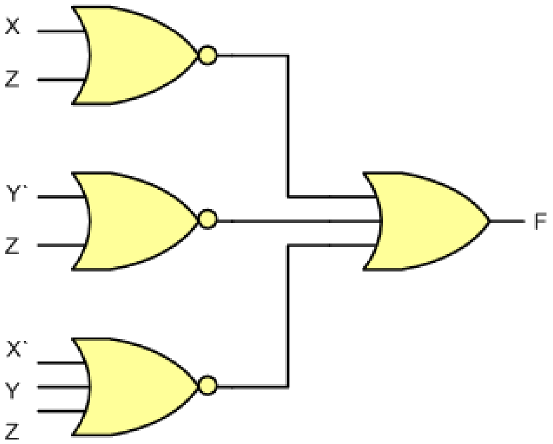
OR-NAND functions:

Example 4: Implement the following function

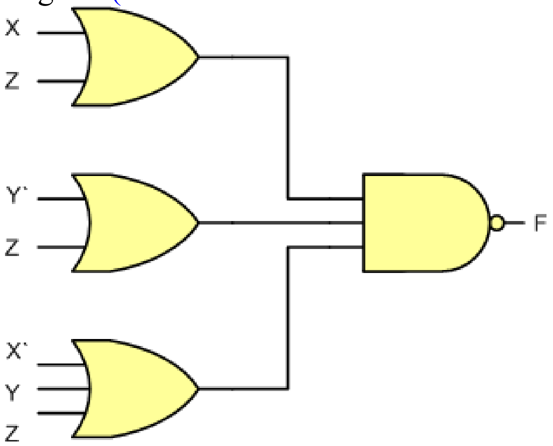
$$F = \overline{(X+Z).(\overline{Y}+Z).(\overline{X}+Y+Z)}$$ **or**

$$\boldsymbol{\overline{F} = (X+Z)(\overline{Y}+Z)(\overline{X}+Y+Z)}$$

Since **F'** is in POS form, it can be implemented by using NOR-NOR circuit. By complementing the output we can get **F**, **or** by using ***NOR-OR*** circuit as shown in the figure.



It can also be implemented using ***OR-NAND*** circuit as it is equivalent to NOR-OR circuit as shown in the figure. (see animation in authorware version)



XOR - XNOR Gates

Lesson Objectives:

In addition to AND, OR, NOT, NAND and NOR gates, exclusive-OR (XOR) and exclusive-NOR (XNOR) gates are also used in the design of digital circuits.
These have special functions and applications. These gates are particularly useful in arithmetic operations as well as error-detection and correction circuits.

XOR and XNOR gates are usually found as 2-input gates. No multiple-input XOR/XNOR gates are available since they are complex to fabricate with hardware.

The objectives of this lesson are to learn about:

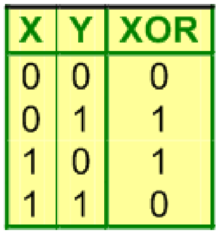
1. XOR gates and XNOR gates
2. Their properties of operation and basic identities
3. Odd function and Even function
4. Parity generation and checking.

XOR Gate:

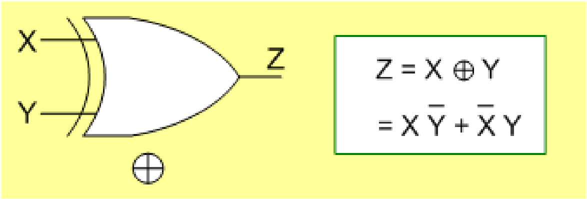
The exclusive-OR (XOR), operator uses the symbol ⊕, and it performs the following logic operation:

$\mathbf{X \oplus Y = X\,Y' + X'\,Y}$

The graphic symbol and truth table of XOR gate is shown in the figure.



X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0



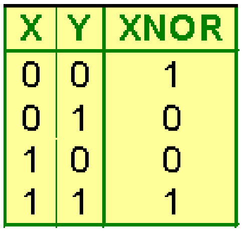
The result is 1 only when either X is equal to 1 or Y is equal to 1, but not when both X and Y are equal to 1.

XNOR Gate:

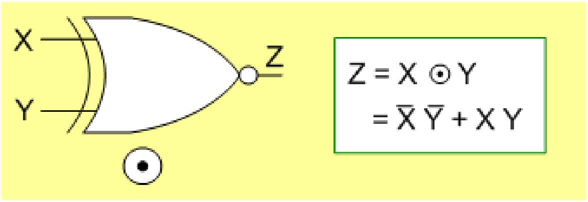
The exclusive-NOR (XNOR), operator uses the symbol ⊙, and it performs the following logic operation

$\mathbf{X \odot Y = X\,Y + X'\,Y' = (X \oplus Y)'}$

The graphic symbol and truth table of XNOR (Equivalence) gate is shown in the figure.



X	Y	XNOR
0	0	1
0	1	0
1	0	0
1	1	1



The result is 1 when either both X and Y are 0’s or when both are 1’s. That is why this gate is often referred to as the **Equivalence** gate.

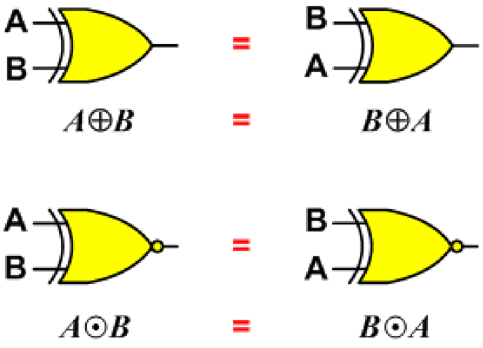
The truth tables clearly show that the exclusive-NOR operation is the complement of the exclusive-OR.

This can also be shown by algebraic manipulation as follows:
(X ⊕ Y)’ = (X Y’ + X’ Y)’
= (X Y’)’ (X’ Y)’ = (X’ + Y) (X + Y’)
= (XY + X’Y’)
= X ⊙ Y

Properties of XOR/XNOR Operations:

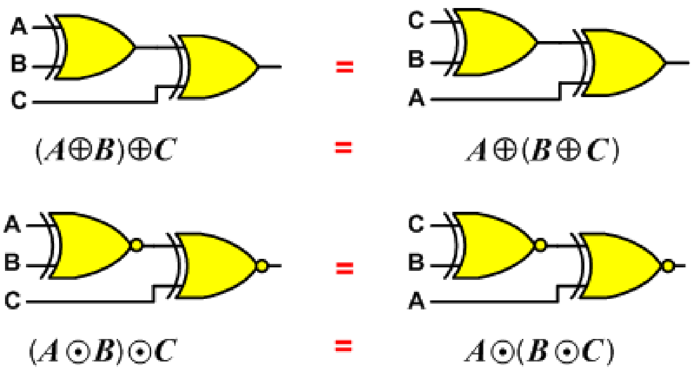
1- Commutativity

- A ⊕ B = B ⊕ A, and
- A ⊙ B = B ⊙ A



2- Associativity

- A ⊕ (B ⊕ C) = (A ⊕ B) ⊕ C, and
- A ⊙ (B ⊙ C) = (A ⊙ B) ⊙ C

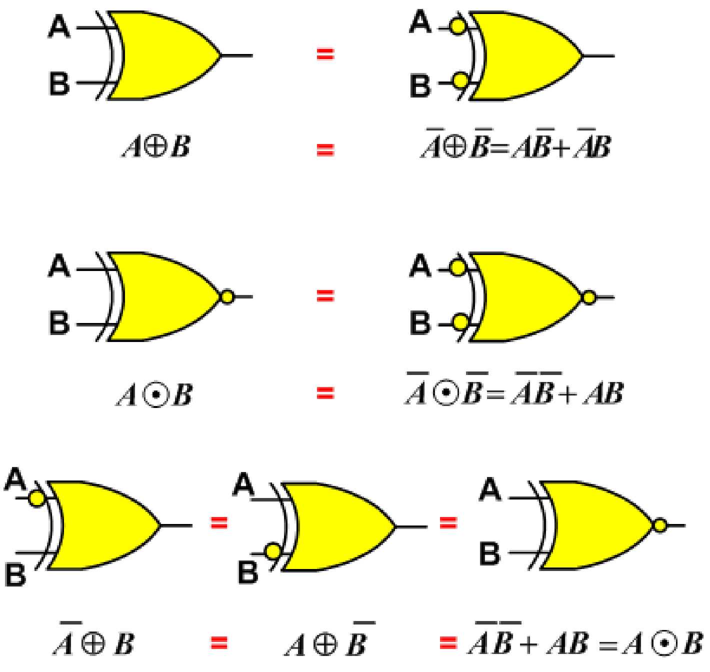


Basic Identities of XOR Operation:

Any of the following identities can be proven using either truth tables or algebraically by replacing the ⊕ operation by its equivalent Boolean expression:

- X ⊕ 0 = X
- X ⊕ 1 = X’
- X ⊕ X = 0
- X ⊕ X’ = 1
- X ⊕ Y’ = X’ ⊕ Y = (X ⊕ Y)’ = X ⊙ Y

The figure provides a graphical presentation of important XOR/XNOR rules and gate equivalence.

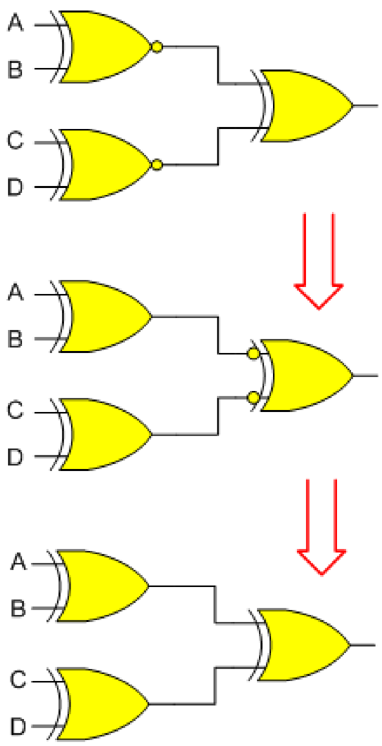

A
B
=
A
B
$A\oplus B$
=
$\bar{A}\oplus\bar{B}=A\bar{B}+\bar{A}B$
A
B
=
A
B
$A\odot B$
=
$\bar{A}\odot\bar{B}=\bar{A}\bar{B}+AB$
A
B
=
A
B
=
A
B
$\bar{A}\oplus B$
=
$A\oplus\bar{B}$
$=\bar{A}\bar{B}+AB=A\odot B$

Example:

Show that **(A ⊙ B) ⊕ (C ⊙ D) = A ⊕ B ⊕ C ⊕ D**

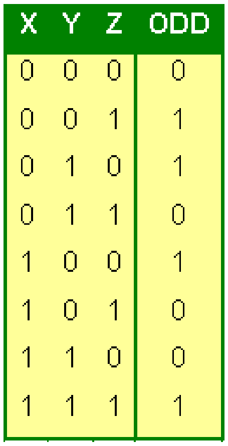
Proving the above identity is easier done using graphical equivalence between gates as specified by the previous figure.

The following figure shows a step-by-step approach starting by the logic circuit corresponding to the left-hand-side of the identity and performing equivalent gate transformations till a circuit is reached that corresponds to the right-hand-side of the identity.

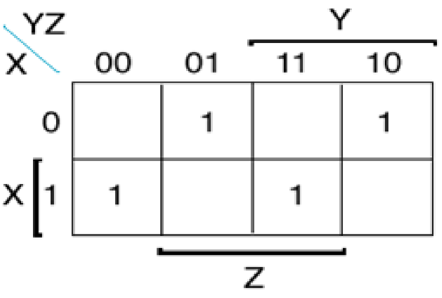


ODD Function:

As shown in the K-map, **X ⊕ Y ⊕ Z = 1, IFF** *(if and only if)* the number of 1's in the input combination is ***odd***.



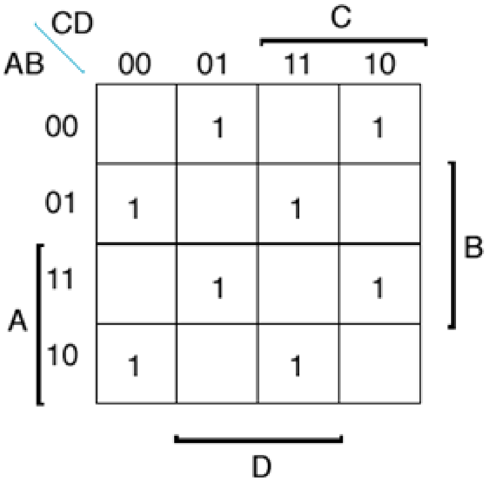
X	Y	Z	ODD
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



X \ YZ	00	01	11	10
0		1		1
1	1		1	

(a) X ⊕ Y ⊕ Z

Likewise, **A ⊕ B ⊕ C ⊕ D = 1**, **IFF** the number of 1’s in the input combination is ***odd***.

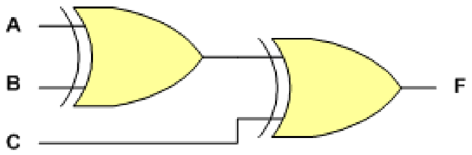


AB \ CD	00	01	11	10
00		1		1
01	1		1	
11		1		1
10	1		1	

(b) A ⊕ B ⊕ C ⊕ D

In general, an exclusive-OR function of n-variables is an *odd function* which has a value of 1 **IFF** the number of 1’s in the input combination is ***odd***, otherwise it has a value of 0.

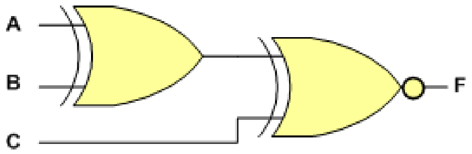
Since XOR gates are only designed with 2 inputs, the 3-input XOR function is implemented by means of two 2-input XOR gates, as shown in figure.



EVEN Function:

The complement of an odd function is an ***even function***. The even function is equal to 1 when the number of 1’s in the input combination is even.

The complement of an odd function (an ***even function***) is obtained by replacing the output gate with an exclusive-NOR gate, as shown in figure.



Parity Generation and Checking:

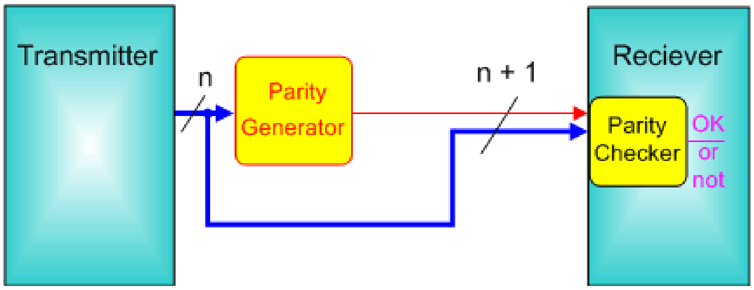
Exclusive-OR functions are very useful in systems using parity bits for error-detection.

A parity bit is used for the purpose of detecting errors during transmission of binary information.

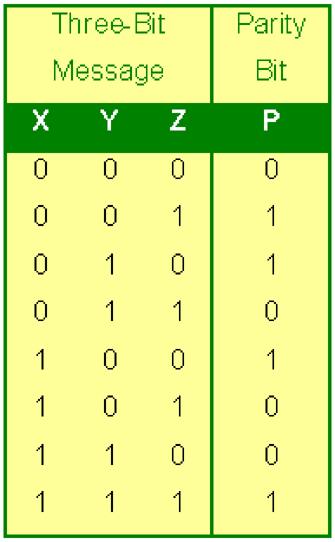
A parity bit is an extra bit included with a binary message to make the **total number of 1’s** in this message (including the parity bit) either **odd** or **even**.

The message, including the parity bit, is transmitted and then checked at the receiving end for errors. An error is detected if the checked parity does not correspond with the one transmitted.

The circuit that generates the parity bit at the transmitter side is called a *parity generator.* The circuit that checks the parity at the receiver side is called a *parity checker.*



As an example, consider a 3-bit message to be transmitted together with an even parity bit. The table shows the ***truth table for the even parity generator***.

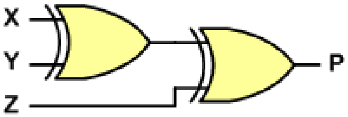


Three-Bit Message			Parity Bit
X	Y	Z	P
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

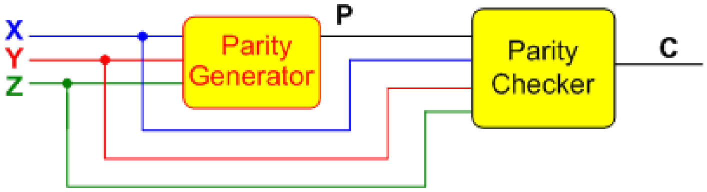
The three bits, **X, Y,** and **Z,** constitute the message and are the inputs to the *even parity generator* circuit whose output is the parity bit ***P***.

For even parity, whenever the message bits (**X, Y& Z**) have an odd number of 1's, the parity bit *P* must be 1. Otherwise, *P* must be 0.
Therefore, *P* can be expressed as a three-variable exclusive-OR function: $\mathbf{P = X \oplus Y \oplus Z}$

The logic diagram for the even parity generator circuit is shown in the figure.



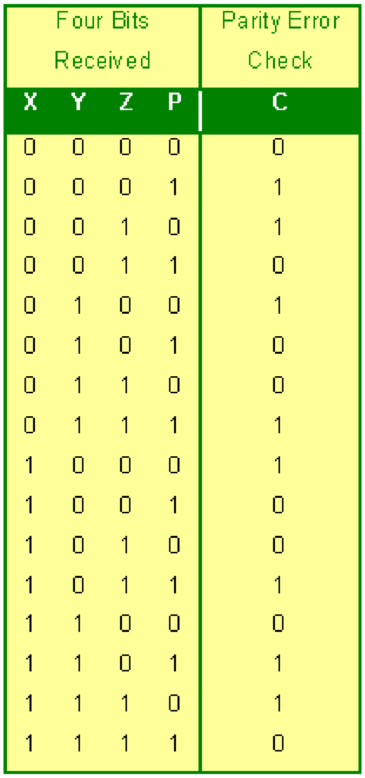
The 4 bits (**X, Y, Z & *P***) are *transmitted* to their destination, where they are applied to a *parity-checker circuit* to check for possible errors in the transmission.



Since the information was transmitted with even parity, the received four bits must have an even number of 1's.

The parity checker generates an error signal (***C* = *1***), whenever the received four bits have an odd number of 1's.

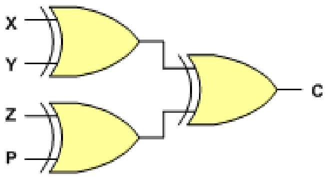
The table below shows the ***truth table for the even-parity checker.***



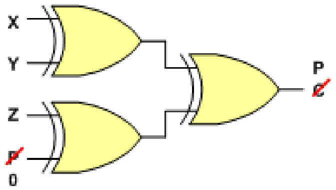
Four Bits Received				Parity Error Check
X	Y	Z	P	C
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

Obviously, the parity checker error output signal *C* is given by the following expression:
$\mathbf{C = X \oplus Y \oplus Z \oplus P}$

The logic diagram of the even-parity checker is shown in the figure.



It is worth noting that the parity generator can also be implemented with the circuit of this figure if the input *P* is connected to logic-0 and the output is marked with P. This is because Z ⊕ 0 = Z, causing the value of *Z* to pass through the gate unchanged.



The advantage of this is that the same circuit can be used for both parity generation and checking.

K-maps

Minimization of Boolean expressions

- *The minimization will result in reduction of the number of gates (resulting from less number of terms) and the number of inputs per gate (resulting from less number of variables per term)*
- *The minimization will reduce cost, efficiency and power consumption.*
- $\mathbf{y(x+x`)=y.1=y}$
- $\mathbf{y+xx`=y+0=y}$
- $\mathbf{(x`y+xy`)=x\oplus y}$
- $\mathbf{(x`y`+xy)=(x\oplus y)`}$

Minimum SOP and POS

- The *minimum sum of products (MSOP)* of a function, *f*, is a SOP representation of *f* that contains the fewest number of product terms and fewest number of literals of any SOP representation of *f*.

Minimum SOP and POS

- *f= (xyz +x`yz+ xy`z+ …..)*

Is called sum of products.

The + is sum operator which is an OR gate.

The product such as xy is an AND gate for the two inputs x and y.

Example

- ***Minimize the following Boolean function using sum of products (SOP):***

- $f(a,b,c,d) = \sum m(3,7,11,12,13,14,15)$

	abcd	
3	0011	**a`b`cd**
7	0111	**a`bcd**
11	1011	**ab`cd**
12	1100	**abc`d`**
13	1101	**abc`d**
14	1110	**abcd`**
15	1111	**abcd**

Example

$f(a,b,c,d) = \sum m(3,7,11,12,13,14,15)$

$=a`b`cd + a`bcd + ab`cd + abc`d`+ abc`d + abcd` + abcd$

$=cd(a`b` + a`b + ab`) + ab(c`d` + c`d + cd` + cd)$

$=cd(a`[b` + b] + ab`) + ab(c`[d` + d] + c[d` + d])$

$=cd(a`[1] + ab`) + ab(c`[1] + c[1])$

$=ab+ab`cd + a`cd$

$=ab+cd(ab` + a`)$

$=ab+ cd(a + a`)(a`+b`)$

$= ab + a`cd + b`cd$

$= ab +cd(a` + b`)$

Minimum product of sums (MPOS)

- The *minimum product of sums (MPOS)* of a function, *f*, is a POS representation of *f* that contains the fewest number of sum terms and the fewest number of literals of any POS representation of *f.*
- The zeros are considered exactly the same as ones in the case of sum of product (SOP)

Example

$f(a,b,c,d) = \prod M(0,1,2,4,5,6,8,9,10)$

$=\sum m(3,7,11,12,13,14,15)$

=[(a+b+c+d)(a+b+c+d`)(a+b`+c`+d`)

(a`+b+c`+d`)(a`+b`+c+ d)(a`+b`+c+ d`) (a`+b`+c`+d)(a`+b`+c`+d`)]

Karnaugh Maps (K-maps)

- Karnaugh maps -- A tool for representing Boolean functions of up to six variables.
- K-maps are tables of rows and columns with entries represent 1`s or 0`s of SOP and POS representations.

Karnaugh Maps (K-maps)

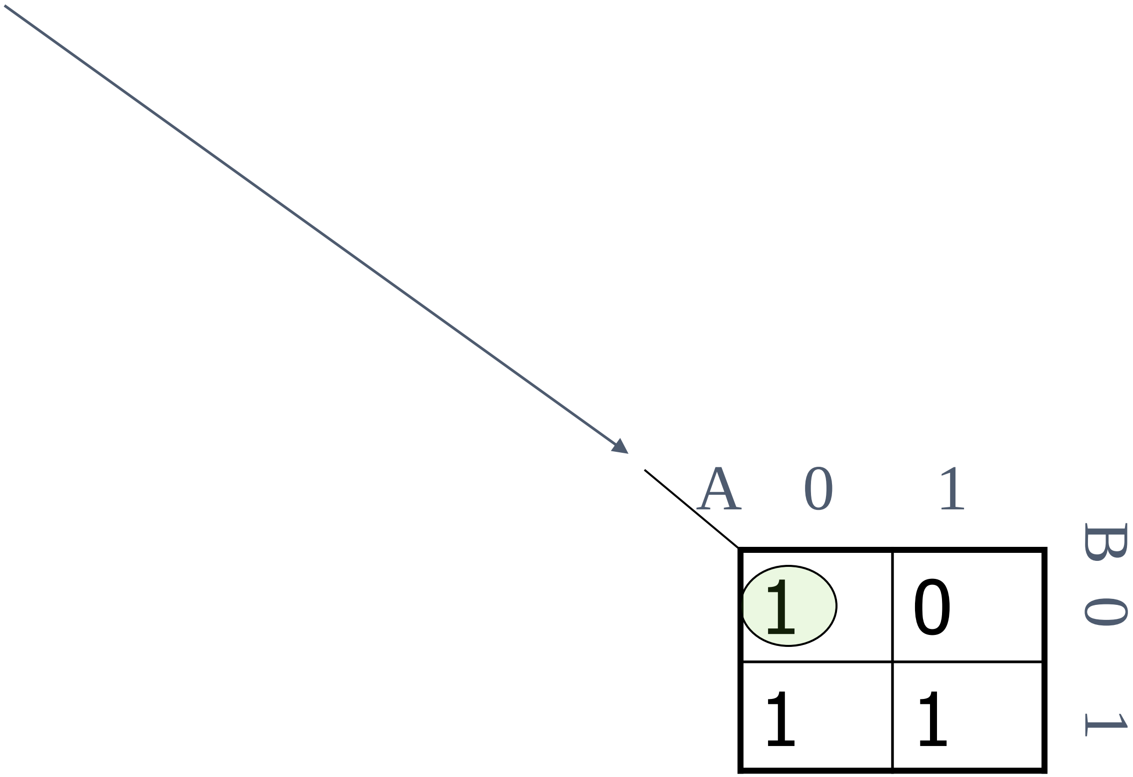
- An *n*-variable K-map has 2^n cells with each cell corresponding to an *n*-variable truth table value.
- K-map cells are labeled with the corresponding truth-table row.
- K-map cells are arranged such that adjacent cells correspond to truth rows that differ in only one bit position (*logical adjacency*).

Karnaugh Maps (K-maps)

- If m_i is a minterm of *f*, then place a 1 in cell *i* of the K-map.
- If M_i is a maxterm of *f*, then place a 0 in cell *i.*
- If d_i is a don't care of *f*, then place a *d or x* in cell *i.*

Examples

- *Two variable K-map f(A,B)=∑m*(0,1,3)=A`B`+A`B+AB



A	0	1	B
	1	0	0
	1	1	1

Three variable map

- *f(A,B,C)* = ∑*m*(0,3,5)= A`B`C`+A`BC+AB`C

A`B` 0 0	A`B 0 1	A B 1 1	A B` 1 0
1	C` 0	A`B`C`	
	1 C 1 A`BC		1 AB`C

Maxterm example

		(A+B)	(A+B`)	(A`+B`)	(A`+B)
		A`B`	A`B	AB	AB`
C	C`		0	0	0
C`	C	0		0	

$f(A,B,C) = \prod M(1,2,4,6,7)$

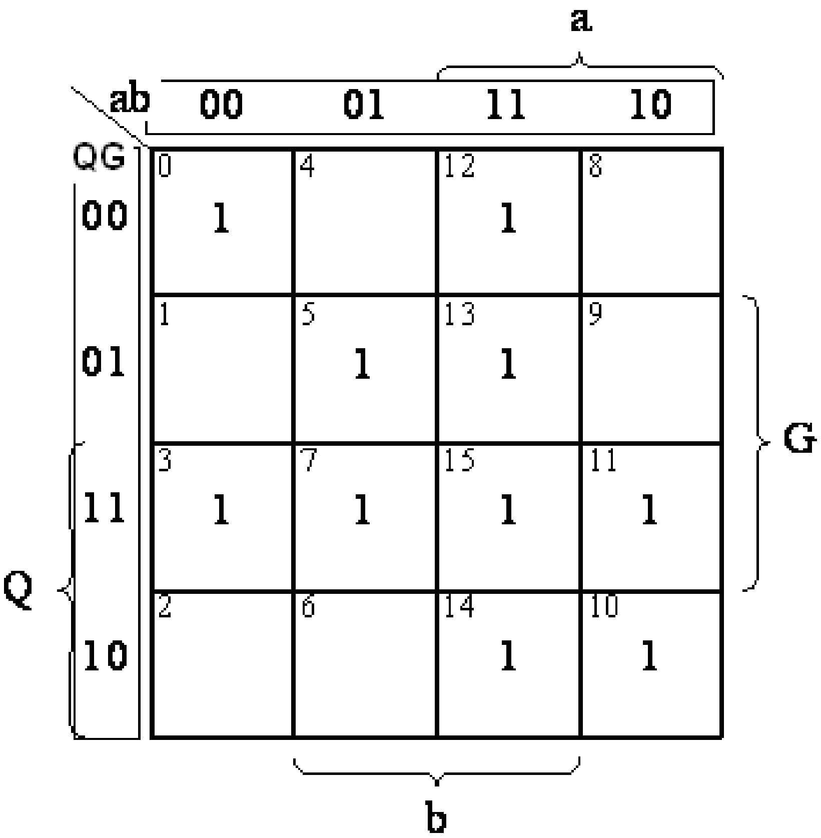
=(A+B+C`)(A+B`+C)(A`+B+C) )(A`+B`+C) (A`+B`+C`)

Note that the complements are (0,3,5) which are the minterms of the previous example

Four variable example

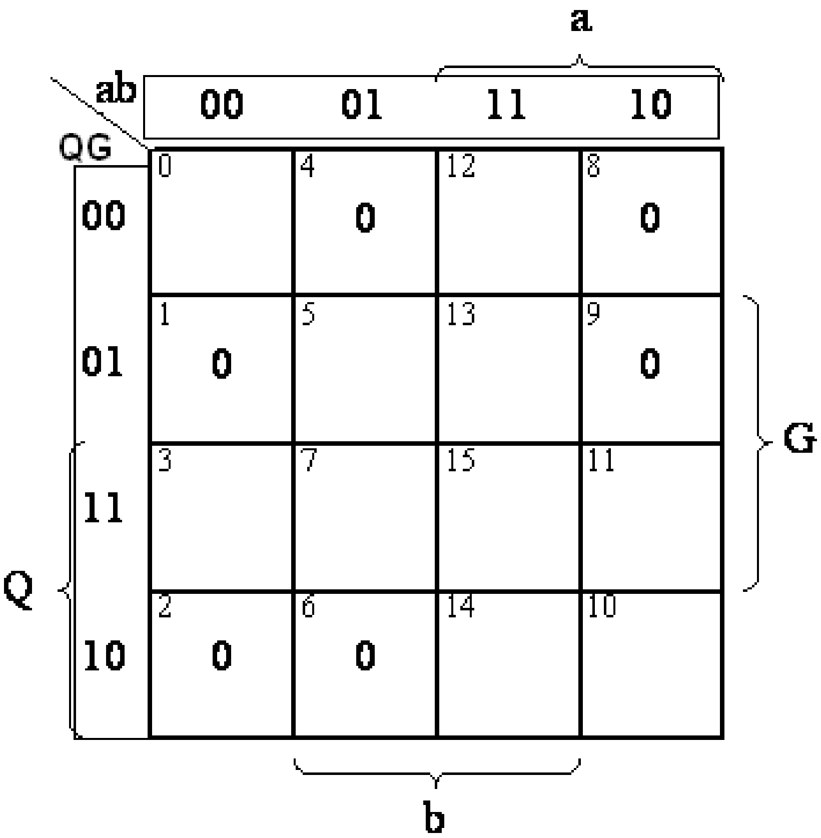
(a) Minterm form. (b) Maxterm form.

$$f(a,b,Q,G) = \sum m(0,3,5,7,10,11,12,13,14,15) = \prod M(1,2,4,6,8,9)$$



QG \ ab	00	01	11	10
00	0 1	4	12 1	8
01	1	5 1	13 1	9
11	3 1	7 1	15 1	11 1
10	2	6	14 1	10 1

(a)



QG \ ab	00	01	11	10
00	0	4 0	12	8 0
01	1 0	5	13	9 0
11	3	7	15	11
10	2 0	6 0	14	10

(b)

Simplification of Boolean Functions Using K-maps

- K-map cells that are physically adjacent are also logically adjacent. Also, cells on an edge of a K-map are logically adjacent to cells on the opposite edge of the map.

- If two logically adjacent cells both contain logical 1s, the two cells can be combined to eliminate the variable that has value 1 in one cell's label and value 0 in the other.

Simplification of Boolean Functions Using K-maps

- This is equivalent to the algebraic operation, aP + a'P =P where P is a product term not containing a or a'.

- A group of cells can be combined only if all cells in the group have the same value for some set of variables.

Simplification Guidelines for K-maps

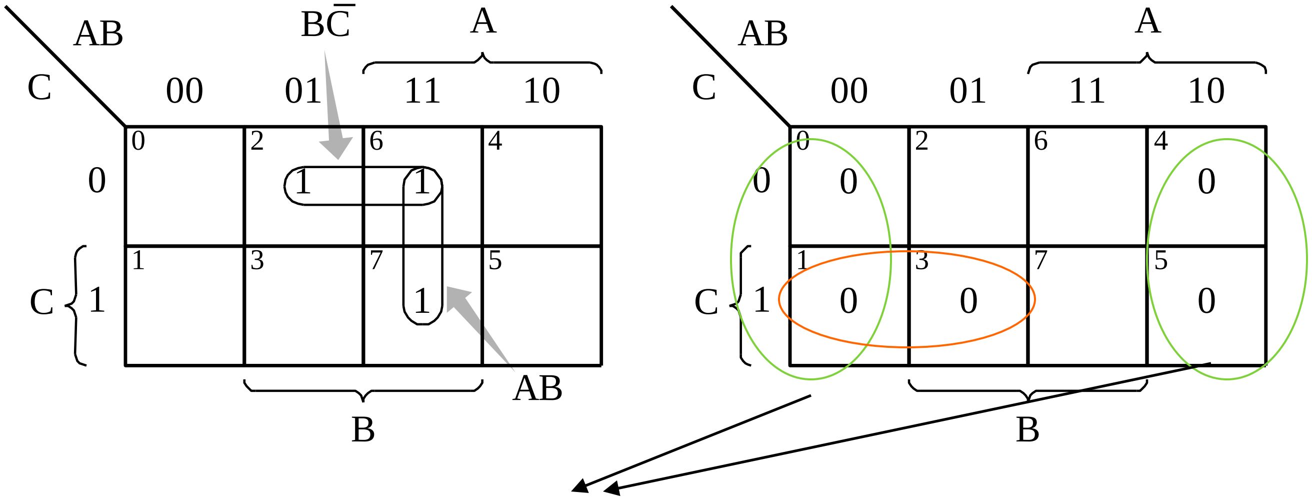
- Always combine as many cells in a group as possible. This will result in the fewest number of literals in the term that represents the group.
- Make as few groupings as possible to cover all minterms. This will result in the fewest product terms.
- Always begin with the largest group, which means if you can find eight members group is better than two four groups and one four group is better than pair of two-group.

Example

Simplify f= A`BC`+ A B C`+ A B C **using;**

(a) Sum of minterms. (b) Maxterms.

- Each cell of an *n*-variable K-map has *n* logically adjacent cells.



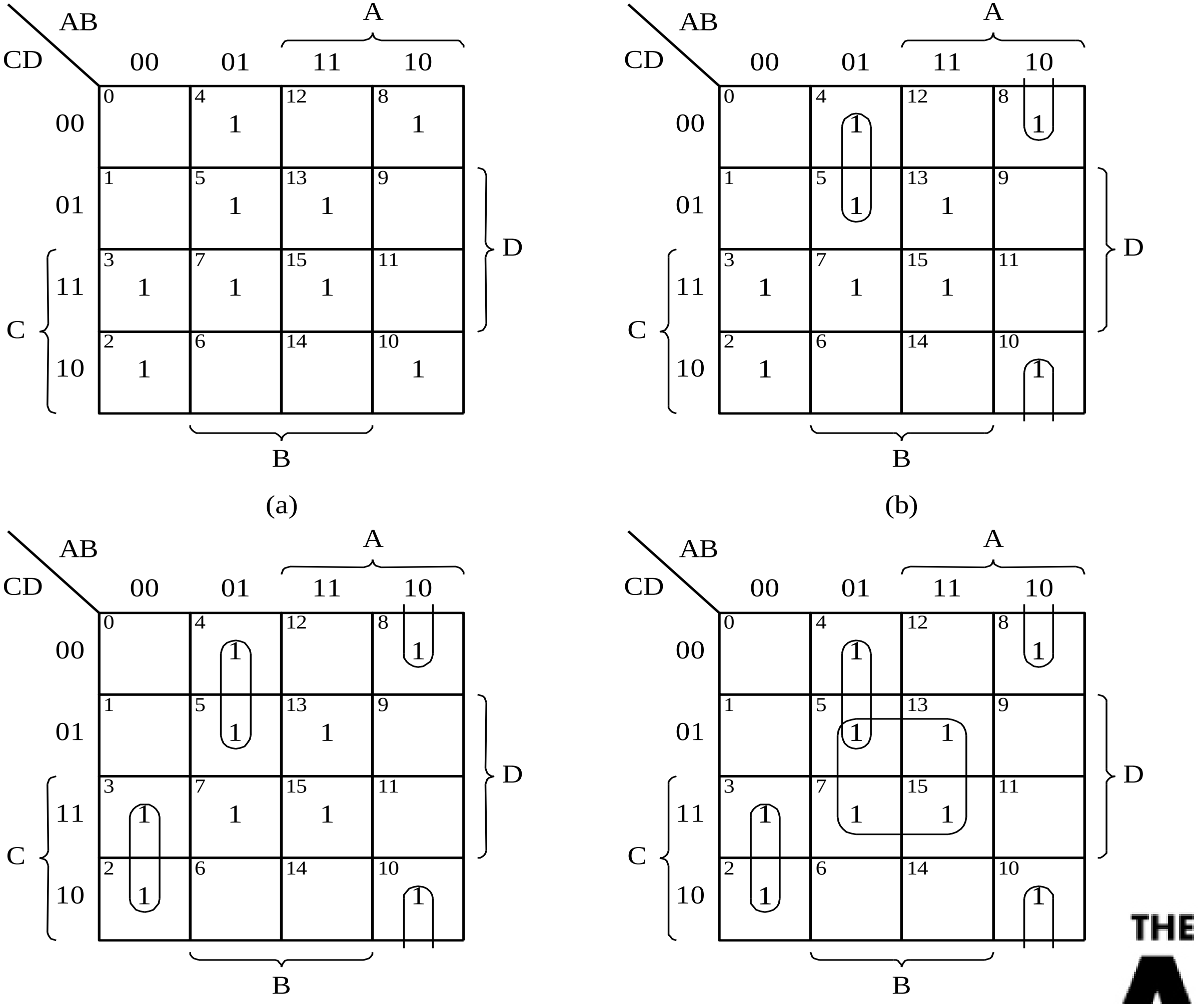
F`= B`+ A`C  F = B(A+C`)

a- f(A,B,C) = AB + BC′

b- f(A,B,C) = B(A + C′)

Example Simplify

$$f(A,B,C,D) = \sum m(2,3,4,5,7,8,10,13,15)$$



CD \ AB	00	01	11	10
00	0	4 1	12	8 1
01	1	5 1	13 1	9
11	3 1	7 1	15 1	11
10	2 1	6	14	10 1

(a)

CD \ AB	00	01	11	10
00	0	4 1	12	8 1
01	1	5 1	13 1	9
11	3 1	7 1	15 1	11
10	2 1	6	14	10 1

(b)

CD \ AB	00	01	11	10
00	0	4 1	12	8 1
01	1	5 1	13 1	9
11	3 1	7 1	15 1	11
10	2 1	6	14	10 1

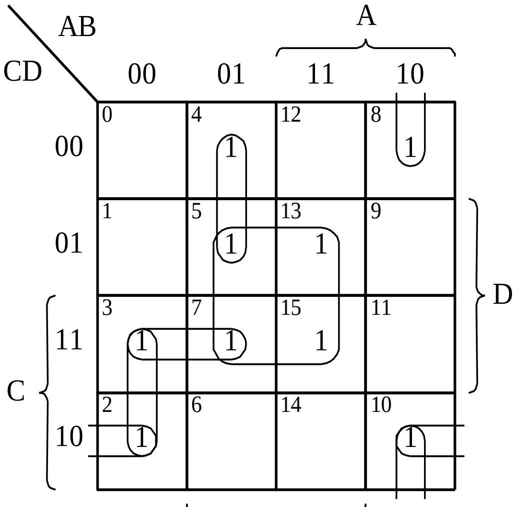
(c)

CD \ AB	00	01	11	10
00	0	4 1	12	8 1
01	1	5 1	13 1	9
11	3 1	7 1	15 1	11
10	2 1	6	14	10 1

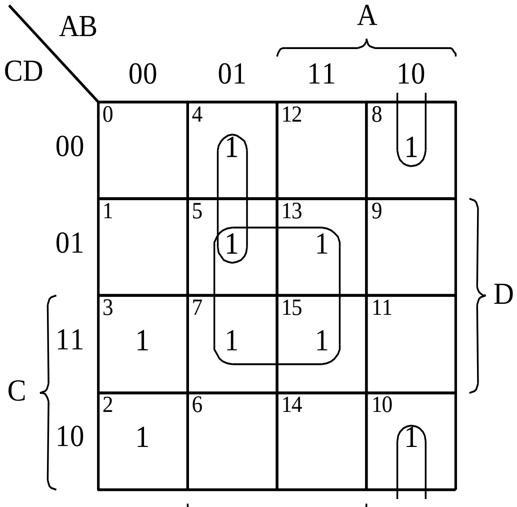
(d)

Example Multiple selections

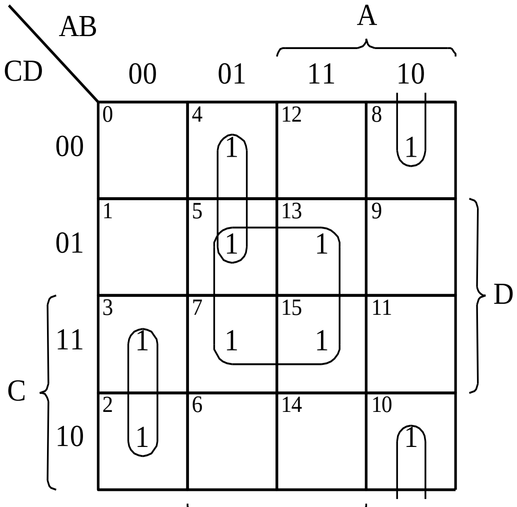
$$f(A,B,C,D) = \sum m(2,3,4,5,7,8,10,13,15)$$



(a)



(b)

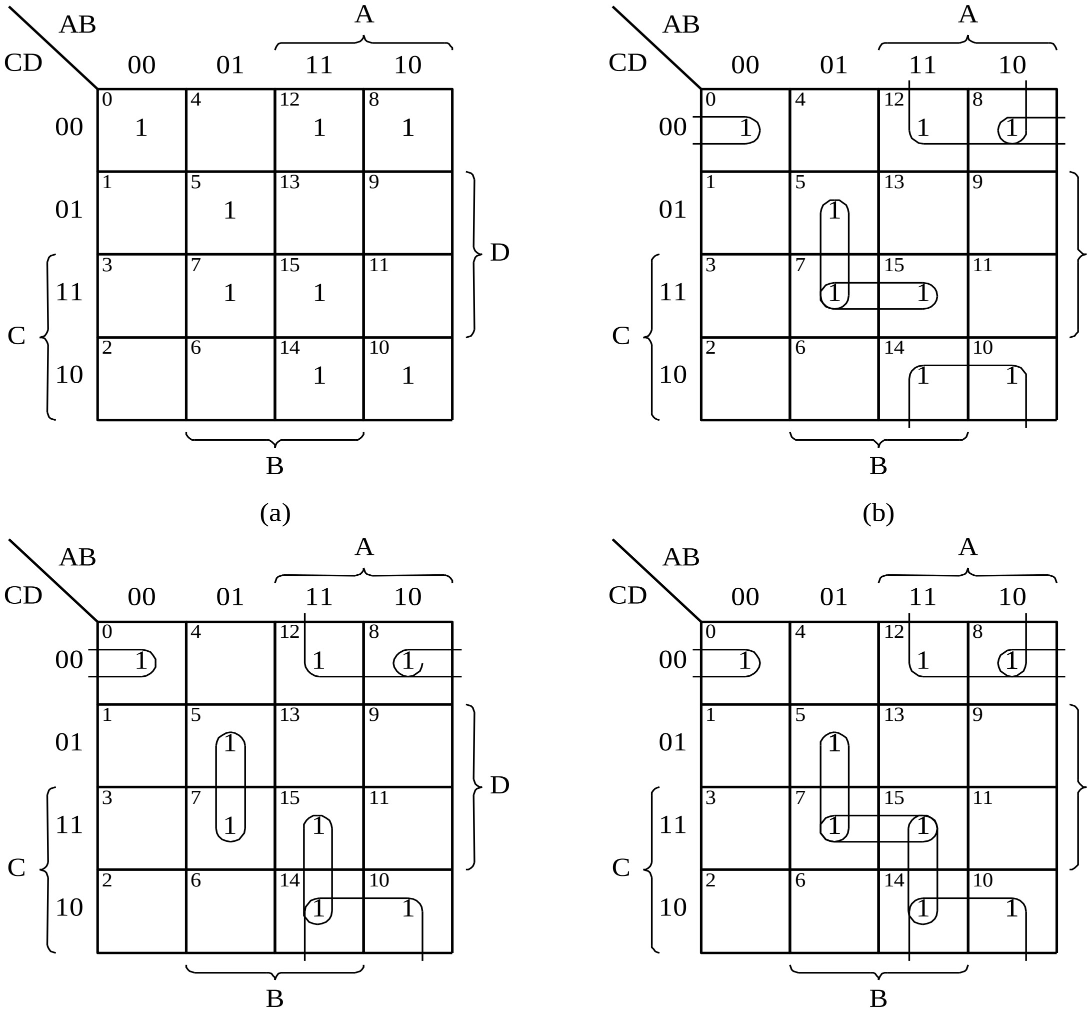


(c)

c produces less terms than a

Example Redundant selections

$$f(A,B,C,D) = \sum m(0,5,7,8,10,12,14,15)$$



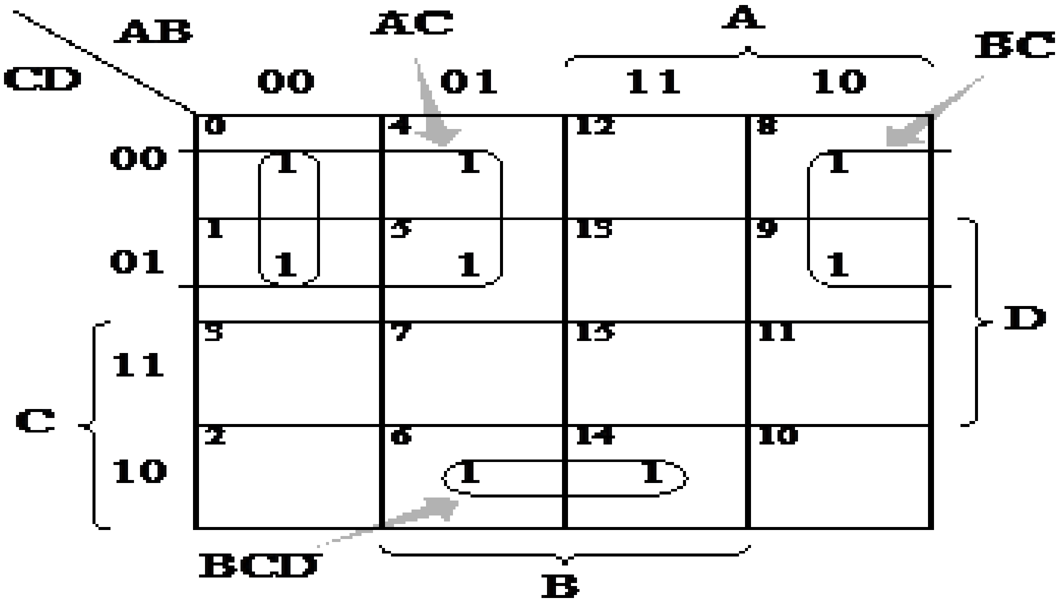
(a)

(b)

(c)

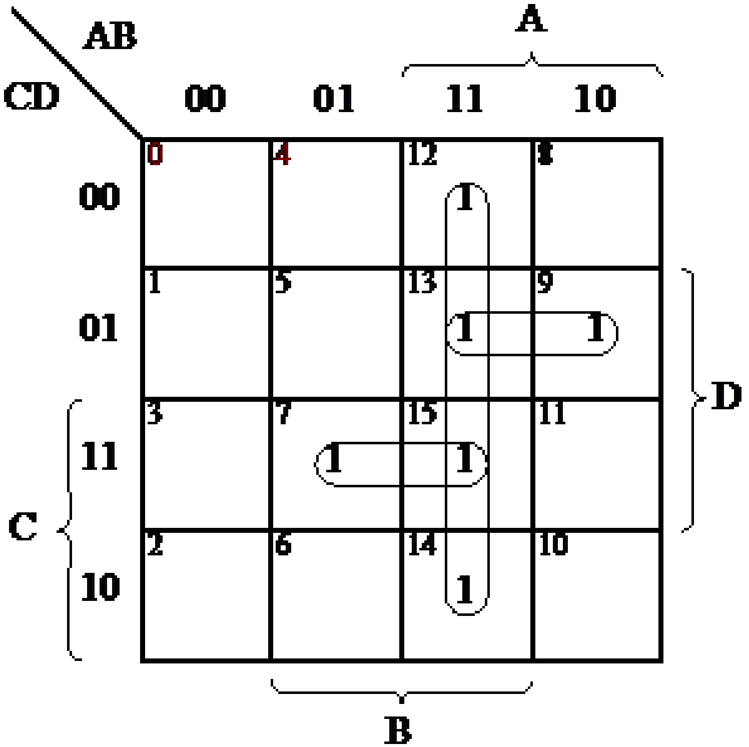
(d)

Example



CD \ AB	00	01	11	10
00	0 1	4 1	12	8 1
01	1 1	5 1	13	9 1
11	3	7	15	11
10	2	6 1	14 1	10

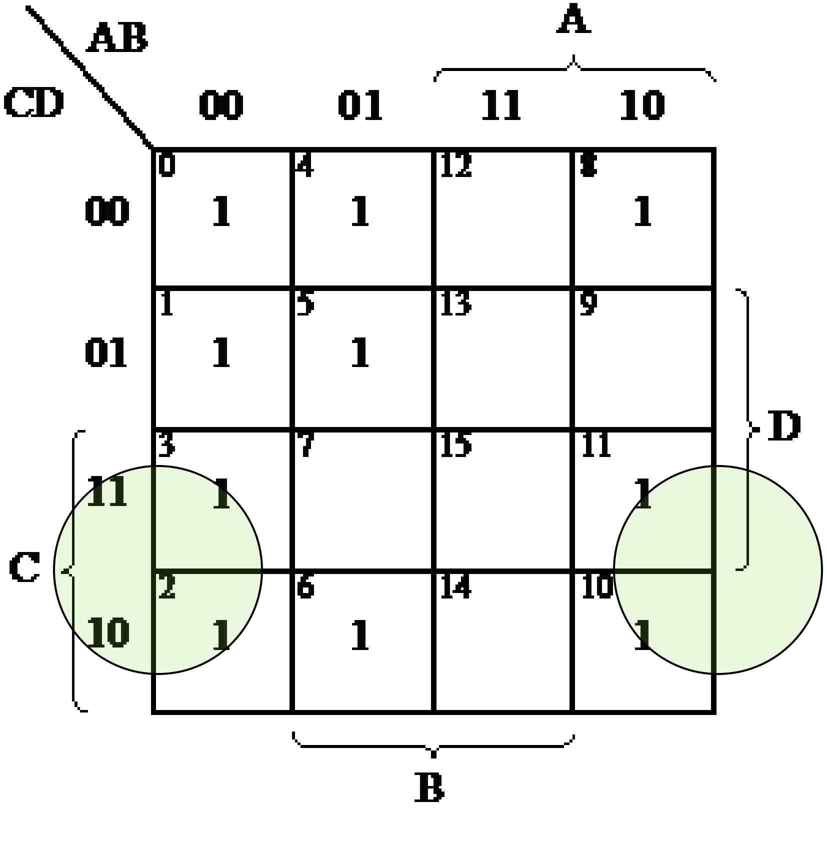
Example



CD \ AB	00	01	11	10
00	0	4	12 1	8
01	1	5	13 1	9 1
11	3	7 1	15 1	11
10	2	6	14 1	10

(a)

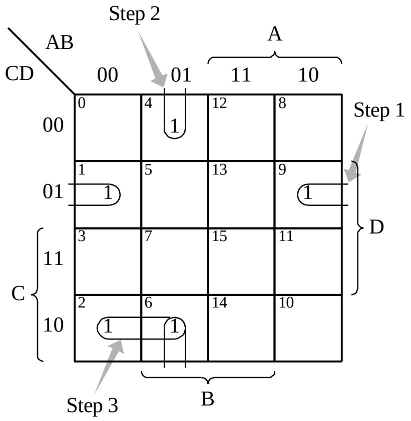
Example



CD \ AB	00	01	11	10
00	0 1	4 1	12	8 1
01	1 1	5 1	13	9
11	3 1	7	15	11 1
10	2 1	6 1	14	10 1

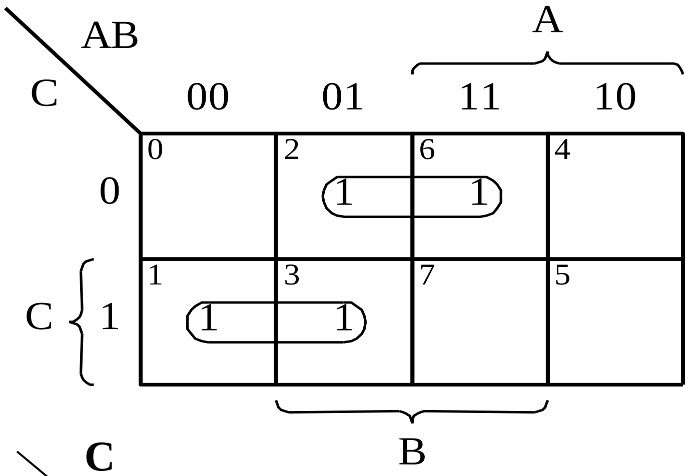
(b)

$f(A,B,C,D) = \sum m(1,2,4,6,9)$

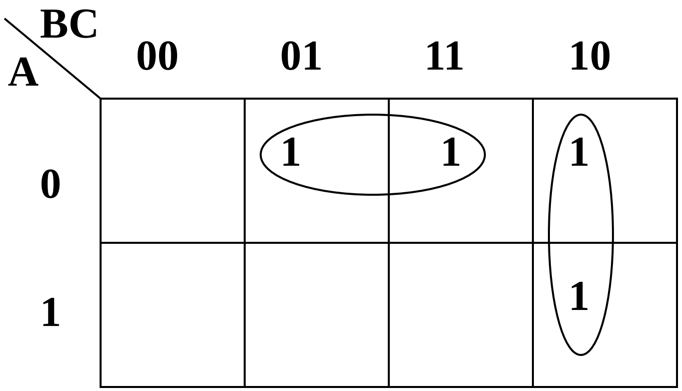


Different styles of drawing maps

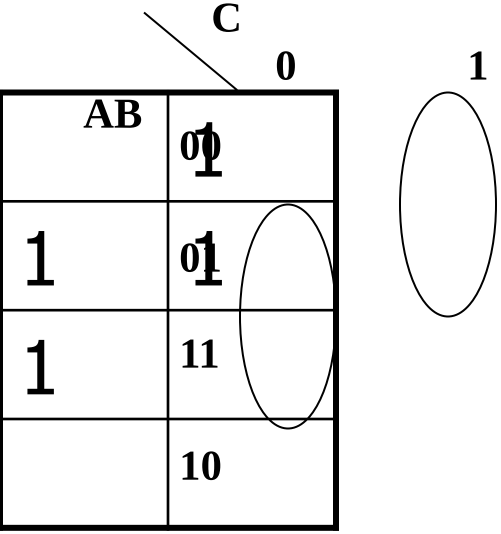
$f(A,B,C)=\sum m(1,2,3,6)=A'C+BC'$



C \ AB	00	01	11	10
0	0	2 1	6 1	4
1	1 1	3 1	7	5

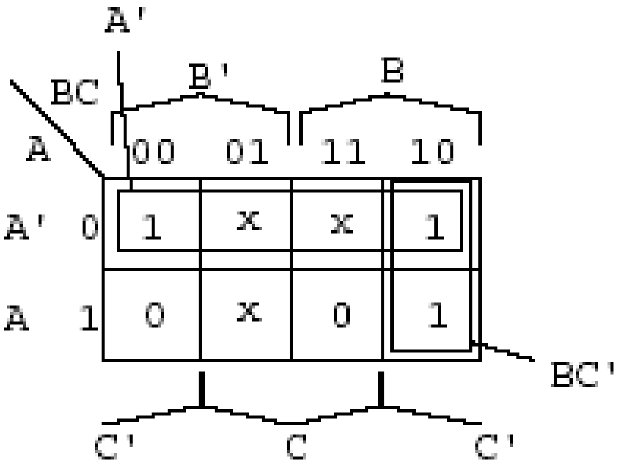


A \ BC	00	01	11	10
0		1	1	1
1				1



AB	00 1
1	01 1
1	11
	10

Don’t-care condition



BC / A	00	01	11	10
A' 0	1	x	x	1
A 1	0	x	0	1

- Minterms that may produce either 0 or 1 for the function.
- They are marked with an ´ in the K-map.
- This happens, for example, when we don’t input certain minterms to the Boolean function.
- These don’t-care conditions can be used to provide further simplification of the algebraic expression.

(Example) F = A\`B\`C\`+A\`BC\` + ABC\`

d=A\`B\`C +A\`BC + AB\`C

F = A\` + BC\`

Basic Concepts

The maximum size of the memory that can be used in any computer is determined by the addressing scheme.

16-bit addresses = 2^{16} = 64K memory locations

Most modern computers are byte addressable.

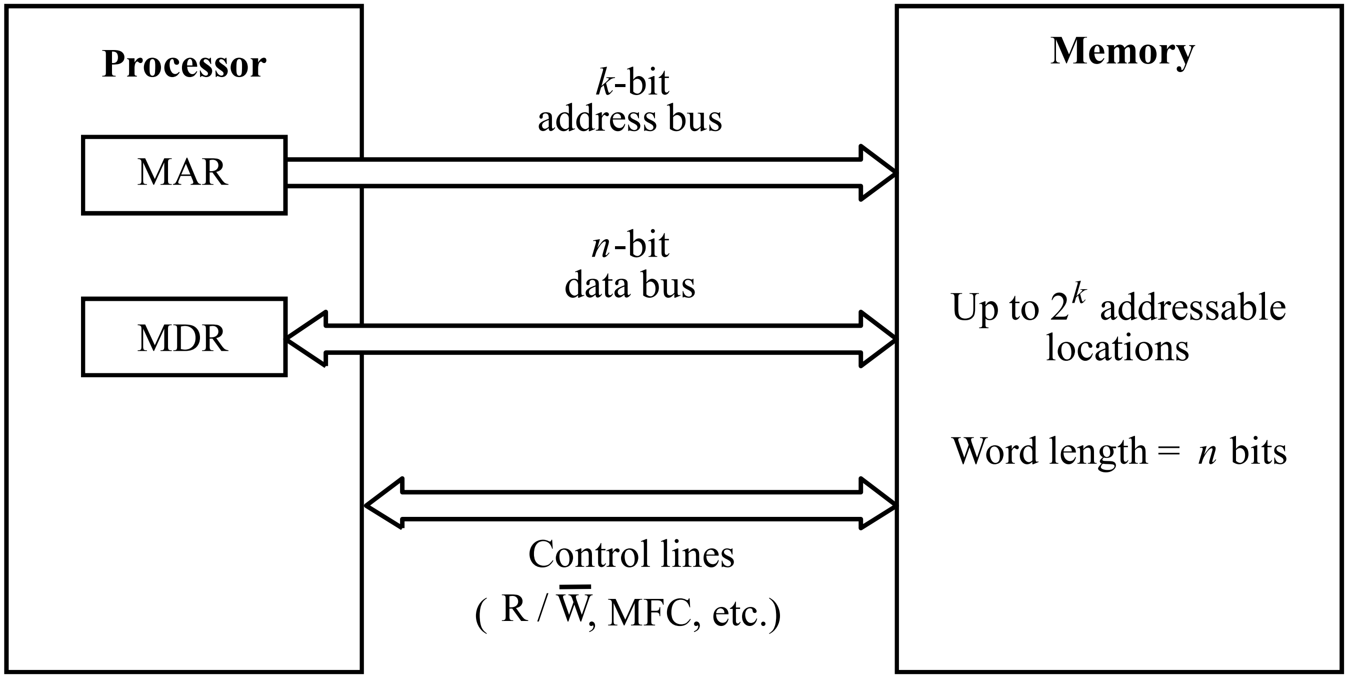
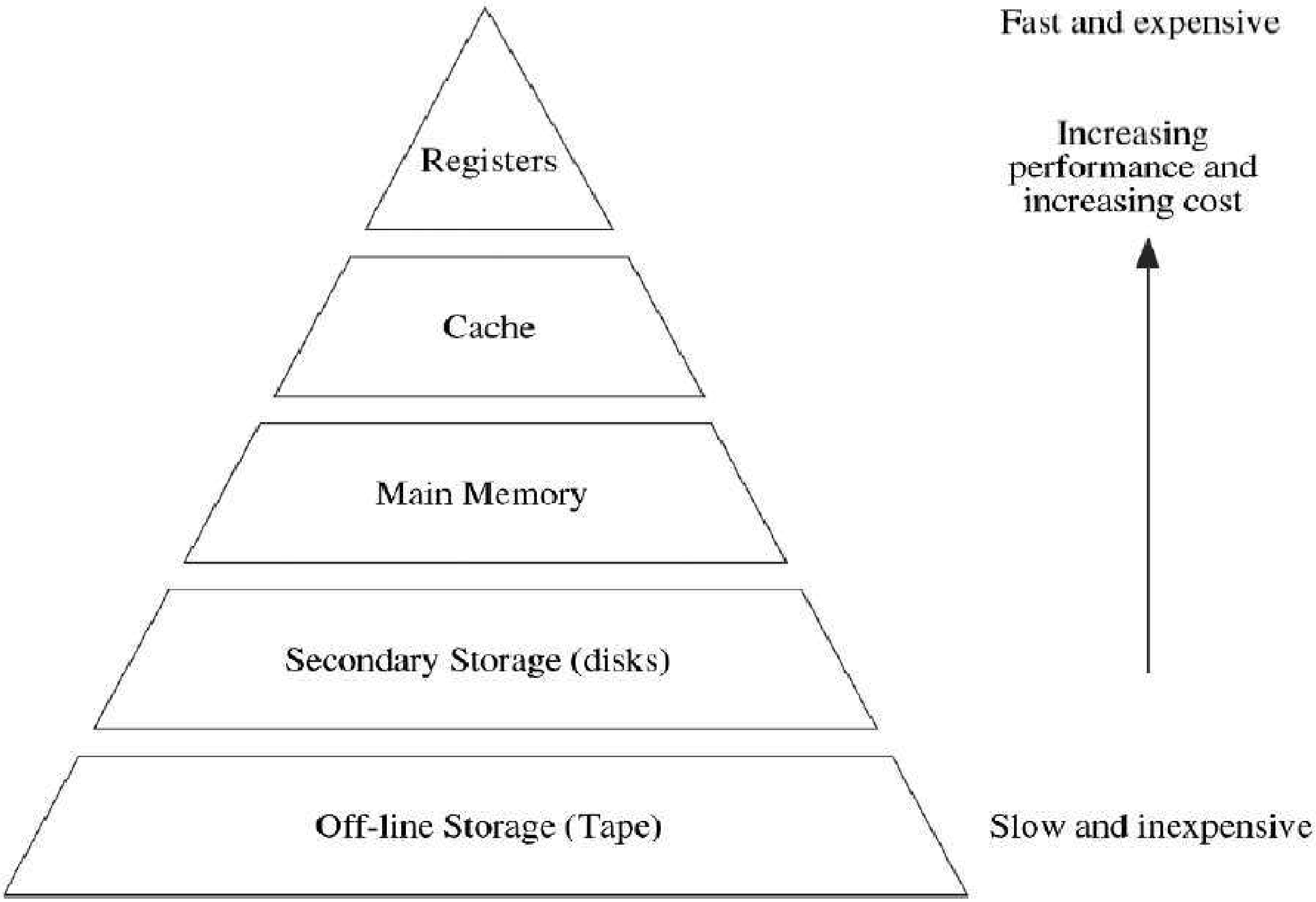


Figure 5.1. Connection of the memory to the processor.

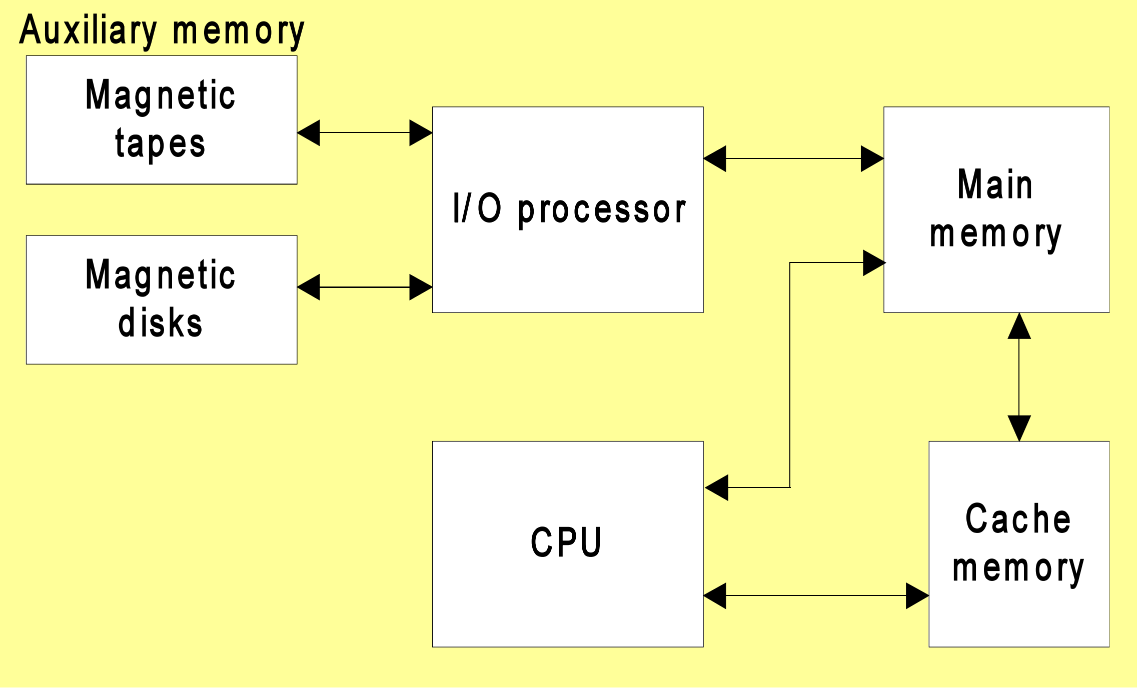
The Memory Hierarchy



Main Memory : memory unit that communicates directly with the CPU (RAM)

Auxiliary Memory : device that provide backup storage (Disk Drives)

Cache Memory : special very-high-speed memory to increase the processing speed (Cache RAM)



Basic Concepts

Memory access time :the average time taken to read a unit of information

Memory cycle time: - the average time lapse between two successive read operations

Semiconductor RAM Memories

Internal organization of memory chips

- Each memory cell can hold one bit of information.
- Memory cells are organized in the form of an array.
- One row is one memory word.
- All cells of a row are connected to a common line, known as the "word line".
- Word line is connected to the address decoder.
- Sense/write circuits are connected to the data input/output lines of the memory chip.

Internal Organization of Memory Chips

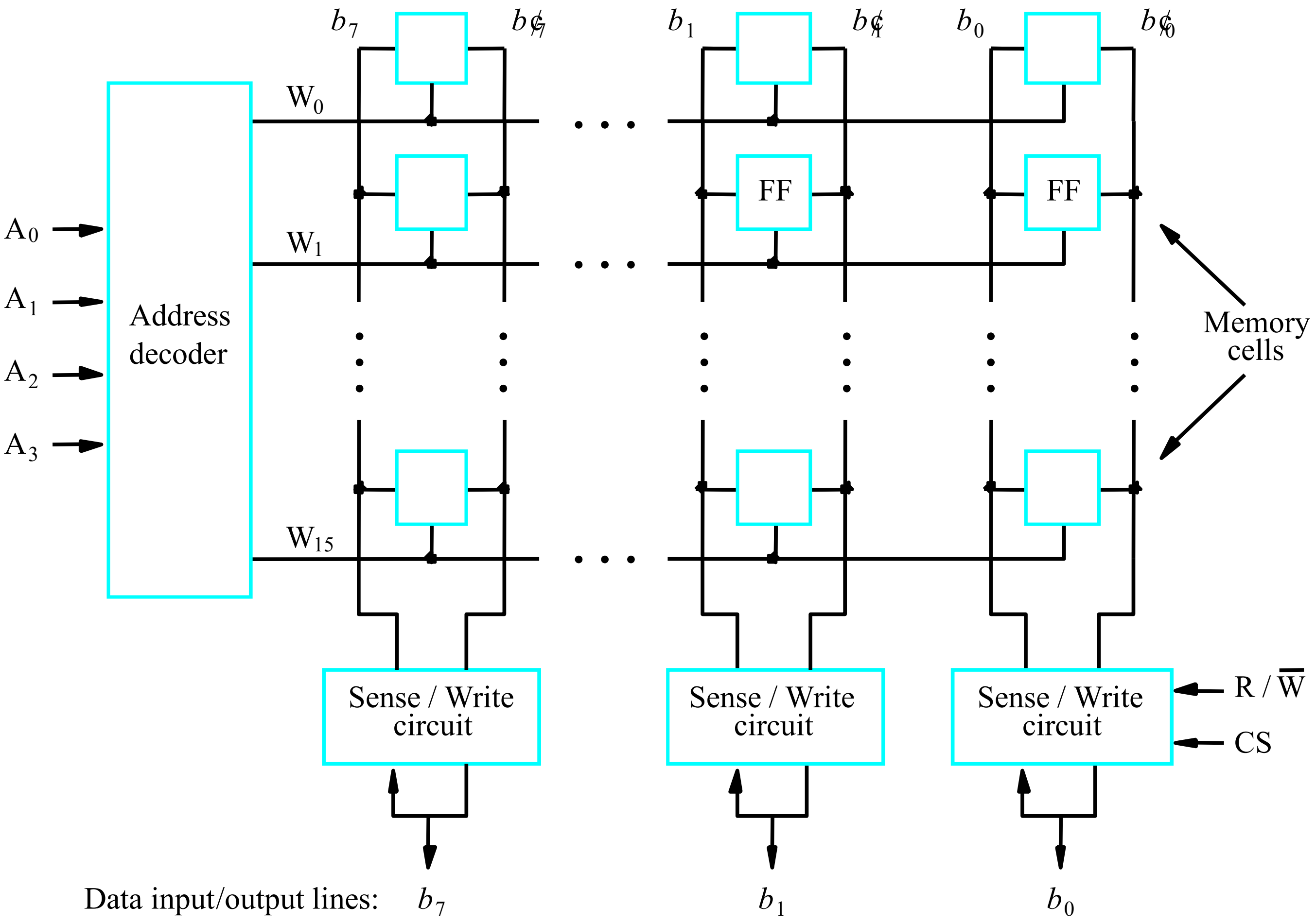


Figure 5.2. Organization of bit cells in a memory chip.

A Memory Chip

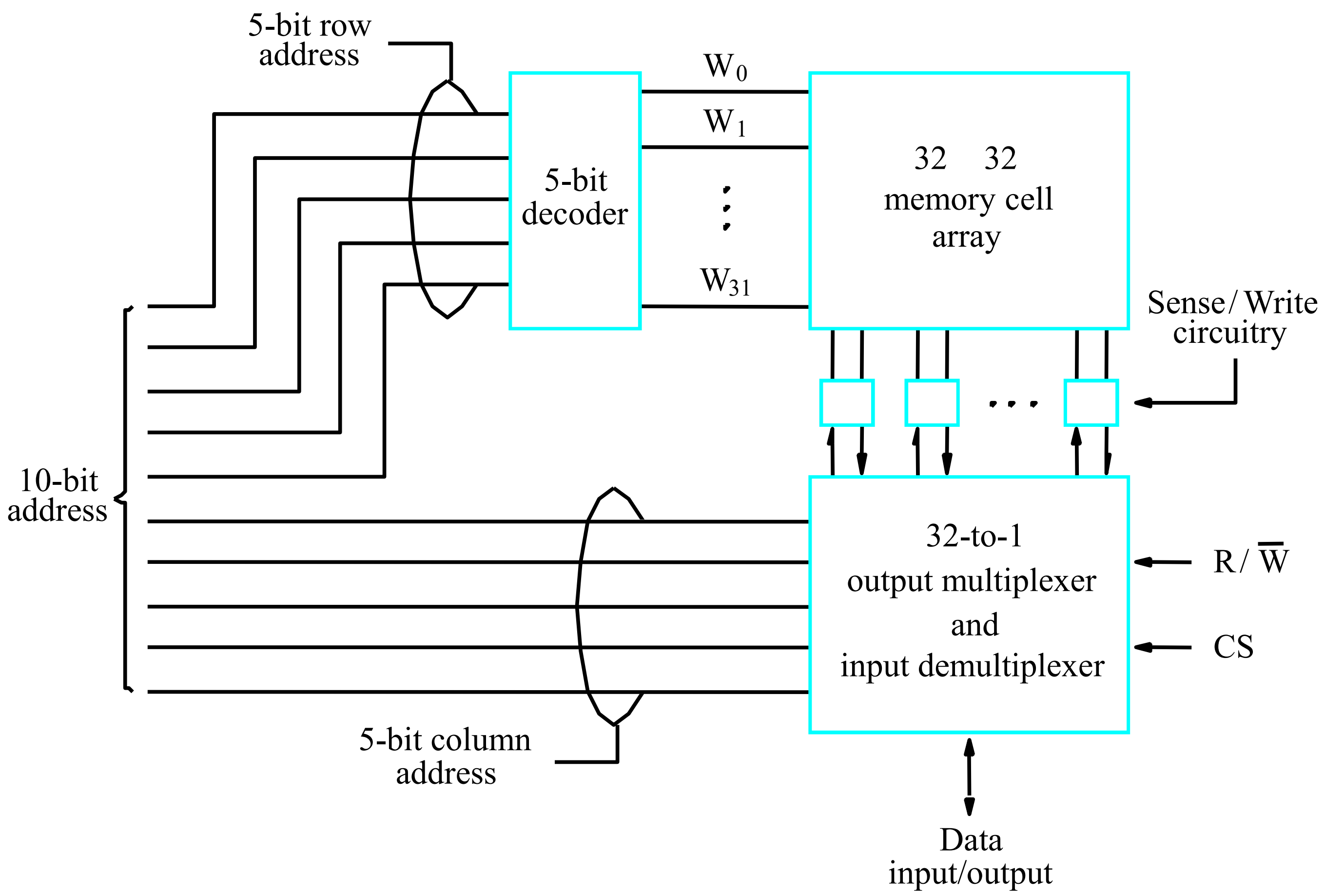


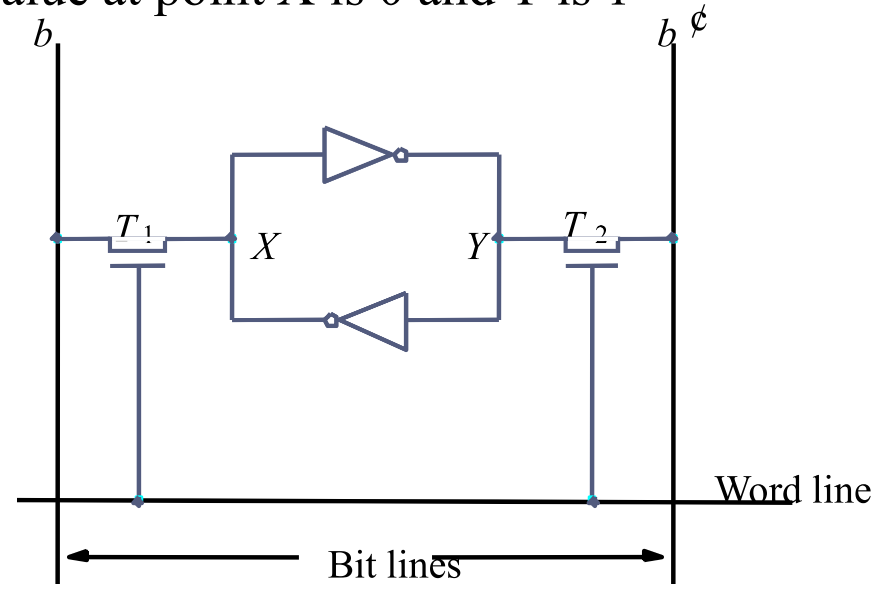
Figure 5.3. Organization of a 1K(1024) □ 1 memory chip

SRAM Cell

- Two transistor inverters are cross connected to implement a basic flip-flop.
- The cell is connected to one word line and two bits lines by transistors T1 and T2
- When word line is at ground level, the transistors are turned off and the latch retains its state.
- Two states

State 1:if the logic value at point X is 1 and Y is 0

State 0: if the logic value at point X is 0 and Y is 1



- Read operation

I.the word line is activated to close switches T1 and T2.

II.Sense/Write circuits at the bottom, monitor the state of b and b’ and set the o/p accordingly.

- Write operation

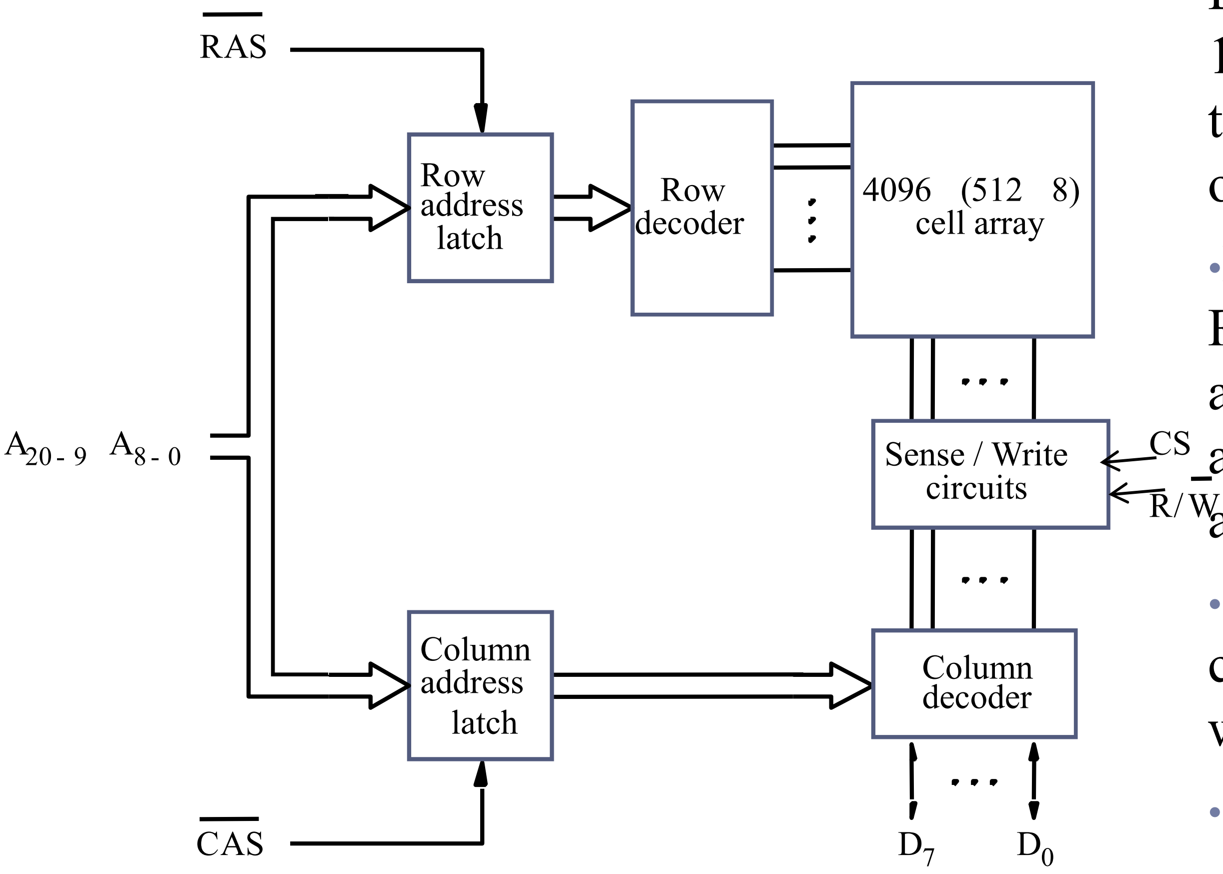
I.State of cell is set by placing the appropriate value on b and b’

II.Activating word line

Asynchronous DRAMs

- Static RAMs (SRAMs):
  - Consist of circuits that are capable of retaining their state as long as the power is applied.
  - Volatile memories, because their contents are lost when power is interrupted.
  - Access times of static RAMs are in the range of few nanoseconds.
  - the cost is usually high.
- Dynamic RAMs (DRAMs):
  - Do not retain their state indefinitely.
  - Contents must be periodically refreshed.
  - Contents may be refreshed while accessing them for reading.

Asynchronous DRAMs



16 megabit DRAM chip

Each row can store 512 bytes. 12 bits to select a row, and 9 bits to select a group in a row. Total of 21 bits.

- First apply the row address, RAS signal latches the row address. Then apply the column address, CAS signal latches the address.
- Timing of the memory unit is controlled by a specialized unit which generates RAS and CAS.
- This is asynchronous DRAM

Synchronous DRAMs

The operations of SDRAM are controlled by a clock signal.

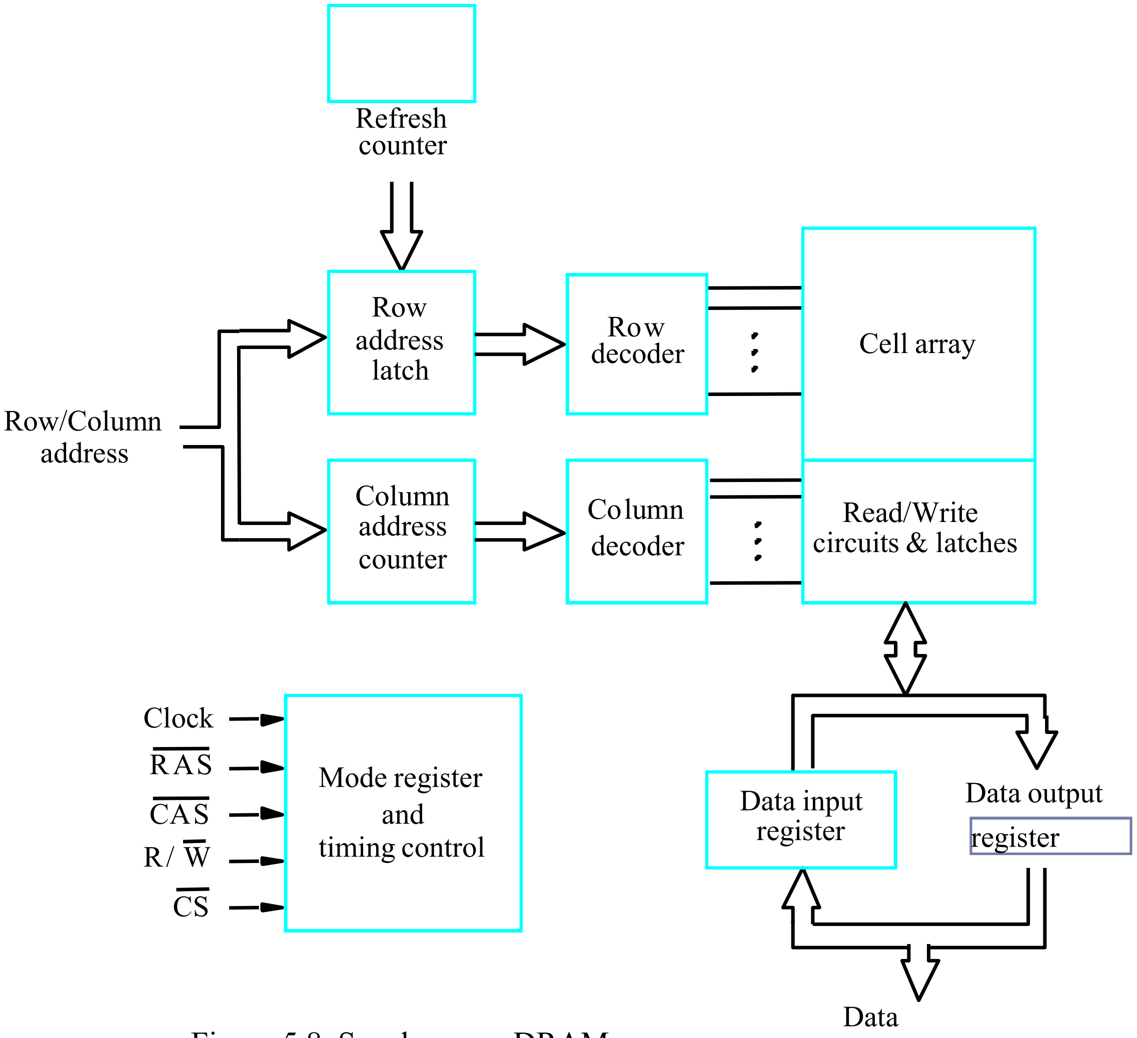


Figure 5.8. Synchronous DRAM.

- •Operation is directly synchronized with processor clock signal.
- •The outputs of the sense circuits are connected to a latch.
- •During a Read operation, the contents of the cells in a row are loaded onto the latches.
- •During a refresh operation, the contents of the cells are refreshed without changing the contents of the latches.
- •Data held in the latches correspond to the selected columns are transferred to the output.

Synchronous DRAMs

Refresh circuits are included (every 64ms).

Clock frequency > 100 MHz

Intel PC100 and PC133

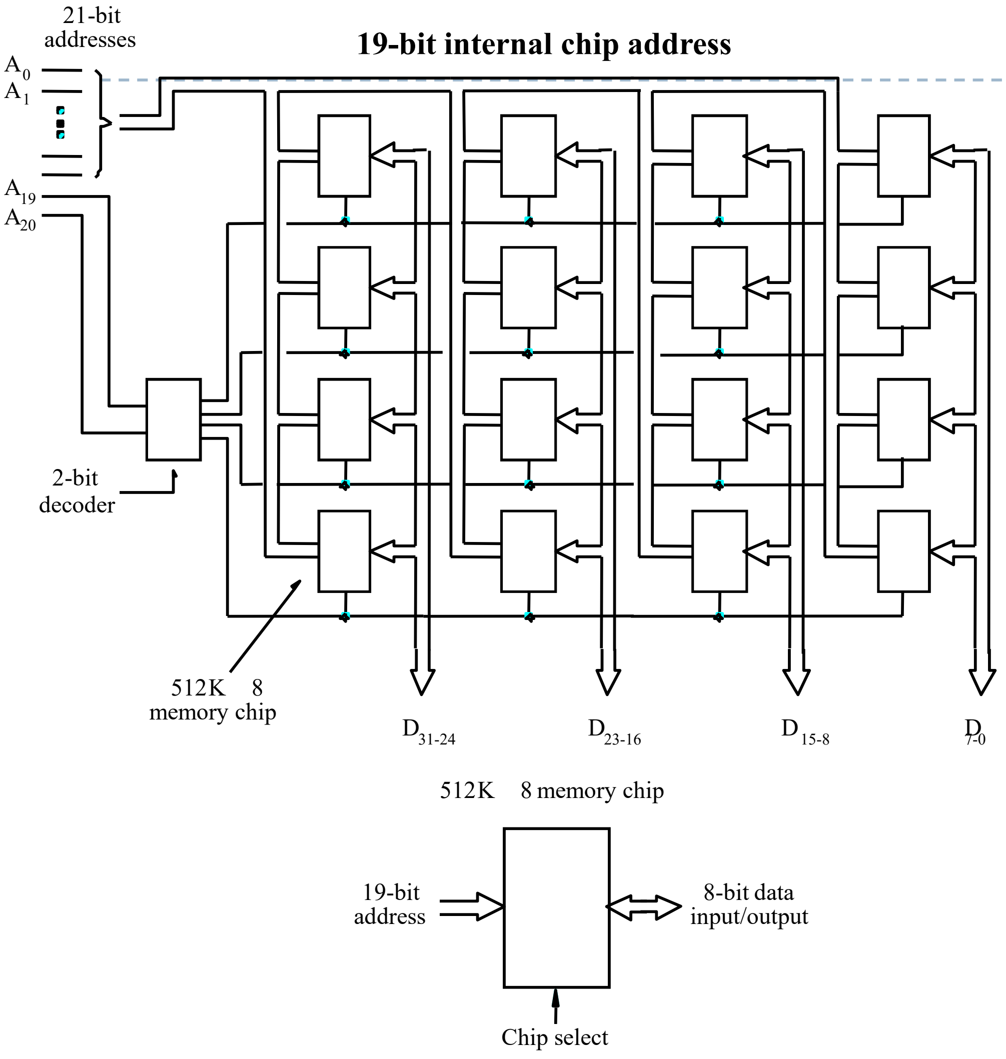
Latency and Bandwidth

- Memory latency – the amount of time it takes to transfer a word of data to or from the memory.
- Memory bandwidth – the number of bits or bytes that can be transferred in one second. It is used to measure how much time is needed to transfer an entire block of data.
- Bandwidth is the product of the rate at which data are transferred (and accessed) and the width of the data bus.

DDR-SDRAM

- Double-Data-Rate SDRAM
- Standard SDRAM performs all actions on the rising edge of the clock signal.
- DDR SDRAM accesses the cell array in the same way, but transfers the data on both edges of the clock.
- The cell array is organized in two banks. Each can be accessed separately.

Static memories



Organization of a 2M * 32 memory module using 512 K * 8 static chips

- Implement a memory unit of 2M words of 32 bits each.
- Use 512x8 static memory chips.
- Each column consists of 4 chips.
- Each chip implements one byte position.
- A chip is selected by setting its chip select control line to 1.
- Selected chip places its data on he data output line, outputs of other chips are in high impedance state.
- 21 bits to address a 32-bit word.
- High order 19 bits are needed to select the row and lower order 2 bits for select column by activating the four Chip Select signals.
- 19 bits are used to access specific byte locations inside the selected chip.

Memory System Considerations

- The choice of a RAM chip for a given application depends on several factors:

  Cost, speed, power, size…

- SRAMs are faster, more expensive, smaller.
- DRAMs are slower, cheaper, larger.

Dynamic memories

- Large dynamic memory systems can be implemented using DRAM chips in a similar way to static memory systems.
- Placing large memory systems directly on the motherboard will occupy a large amount of space.
  - □Also, this arrangement is inflexible since the memory system cannot be expanded easily.
- Packaging considerations have led to the development of larger memory units known as SIMMs (Single In-line Memory Modules) and DIMMs (Dual In-line Memory Modules).
- Memory modules are an assembly of memory chips on a small board that plugs vertically onto a single socket on the motherboard.
  - □Occupy less space on the motherboard.
  - □Allows for easy expansion by replacement.

Memory Controller

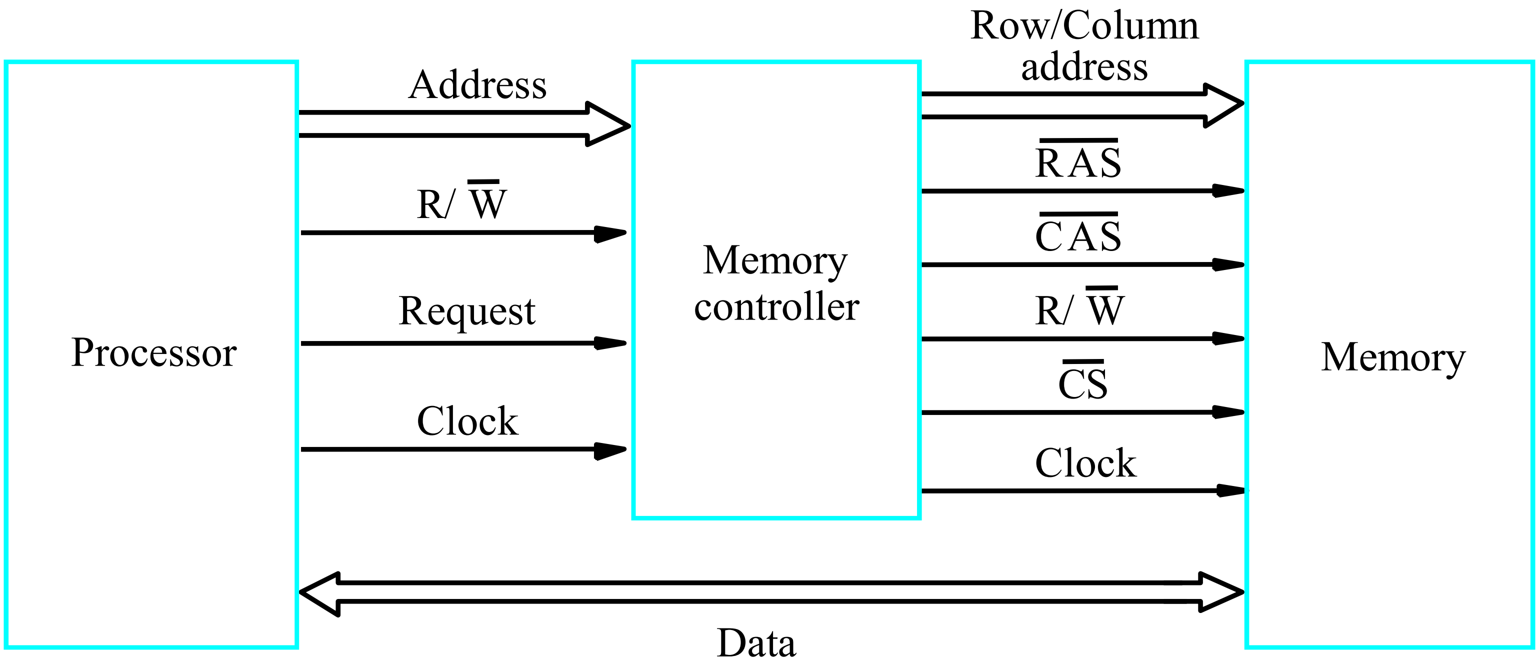


Figure 5.11. Use of a memory controller.

Read-Only Memories

ROM

- ROM is used for storing programs that are **PERMENTLY** resident in the computer and for tables of constants that do not change in value once the production of the computer is completed
- The ROM portion of main memory is needed for storing an initial program called *bootstrap loader,*

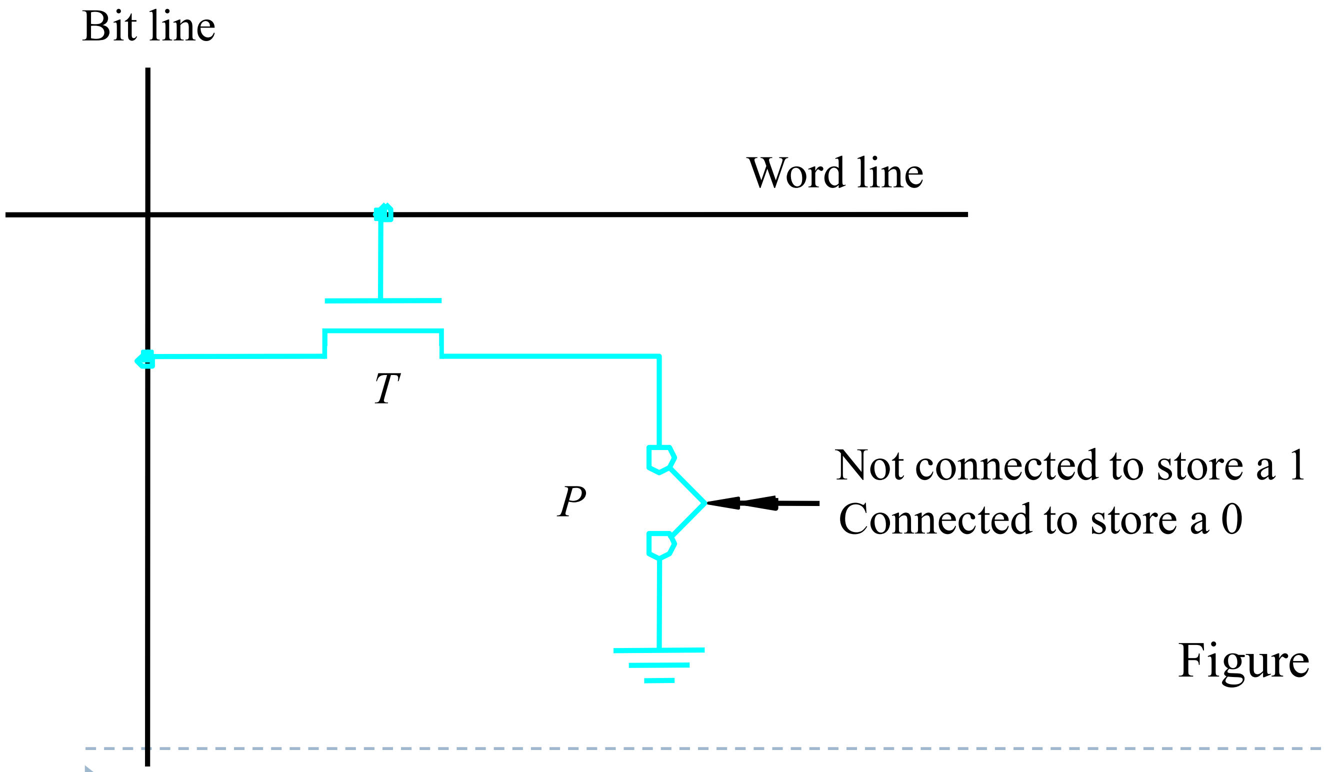


Figure 5.12. A ROM cell.

Read-Only-Memory

- ROM
- PROM: programmable ROM
- EPROM: erasable, reprogrammable ROM
- EEPROM: can be programmed and erased electrically

Flash Memory

- Difference: only possible to write an entire block of cells ,but read the contents of a single cell
- Low power consumption
- Use in portable equipment
  - Flash cards
  - Flash drives

Cache Memories

Cache

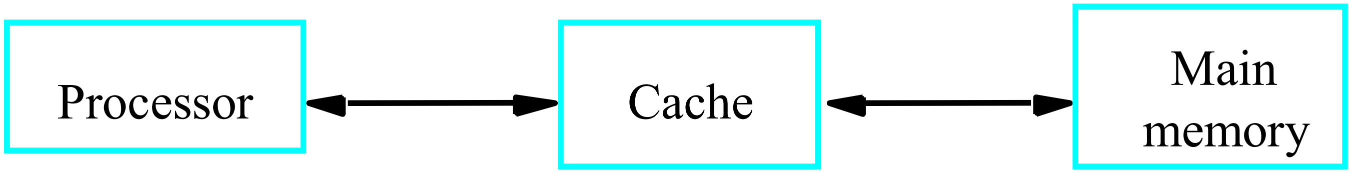


Figure 5.14. Use of a cache memory.

- Processor issues a Read request, a block of words is transferred from the main memory to the cache, one word at a time.Subsequent references to the data in this block of words are found in the cache.
- At any given time, only some blocks in the main memory are held in the cache. Which blocks in the main memory are in the cache is determined by a “mapping function.
- When the cache is full, and a block of words needs to be transferred
  from the main memory, some block of words in the cache must be replaced. This is determined by a “replacement algorithm”.

Cache memory

- If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, Thus reducing the total execution time of the program
- Such a fast small memory is referred to as cache memory
- The cache is the fastest component in the memory hierarchy and approaches the speed of CPU component.
- When CPU needs to access memory, the cache is examined

- Locality of reference

  - temporal

  - spatial
- Cache block – cache line
  - A set of contiguous address locations of some size

Principle of Locality

Principle of locality (or locality of reference):

- Program accesses a relatively small portion of the address space at any instant of time.
- Temporal locality and spatial locality.

Temporal locality (locality in time):

- Keep most recently accessed data items closer to the processor.

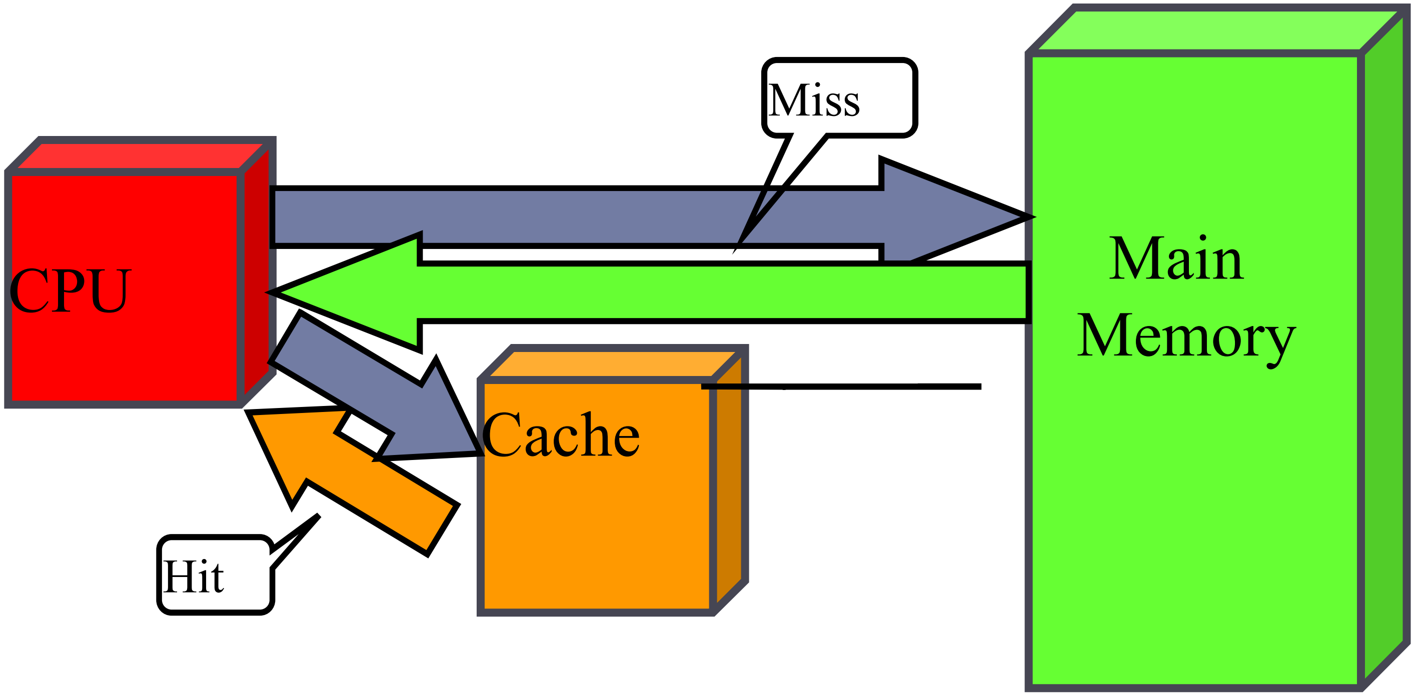
Spatial locality (locality in space):

- Move blocks consisting of contiguous words to ‘upper’ levels.

Cache Hit: data appears in some block

- Hit rate: the fraction of memory access found in the upper level.

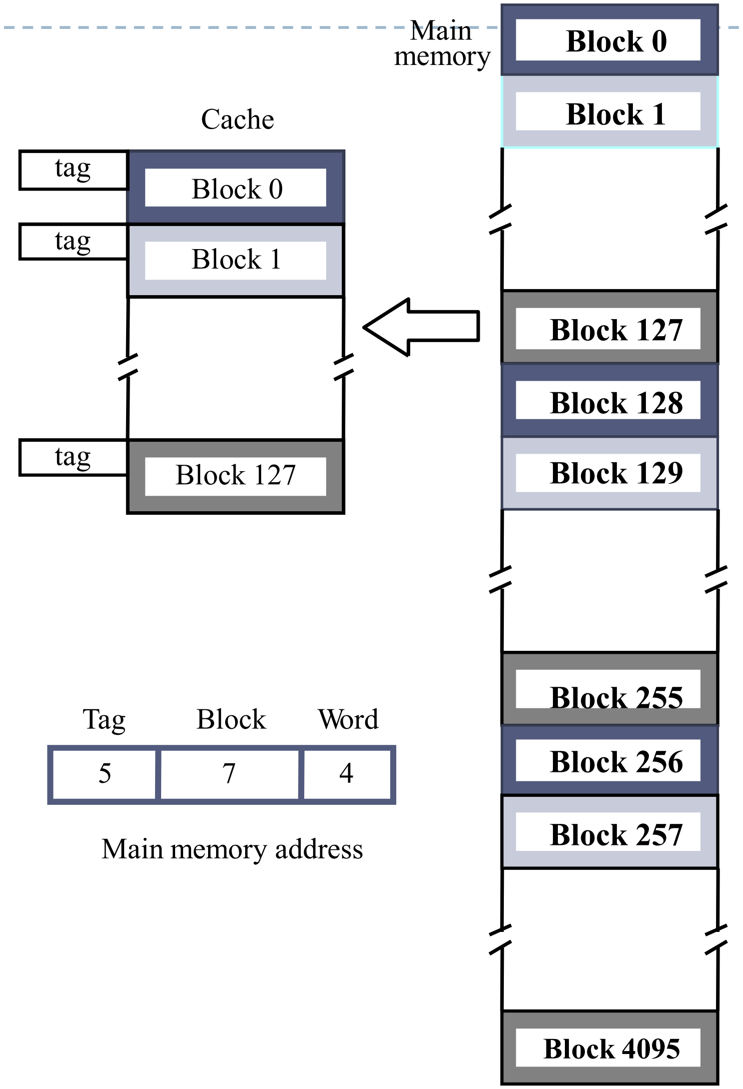
Cache Miss: data is not found, and needs to be retrieved from a block.


Miss
CPU
Main Memory
Cache
Hit

- The basic characteristic of cache memory is its fast access time, Therefore, very little or no time must be wasted when searching the words in the cache
- The transformation of data from main memory to cache memory is referred to as a **mapping** process, there are three types of mapping:
    - Associative mapping
    - Direct mapping
    - Set-associative mapping

- In general case, there are 2^k words in cache memory and 2^n words in main memory .
- The n bit memory address is divided into two fields: k-bits for the index and n-k bits for the tag field.

Direct mapping



- Block j of the main memory maps to j modulo 128 of the cache. 0 maps to 0, 129 maps to 1.
- More than one memory block is mapped onto the same position in the cache.
- May lead to contention for cache blocks even if the cache is not full.
- Resolve the contention by allowing new block to replace the old block, leading to a trivial replacement algorithm.
- Memory address is divided into three fields:
  - Low order 4 bits determine one of the 16 words in a block.
  - When a new block is brought into the cache, the next 7 bits determine which cache block this new block is placed in.
  - High order 5 bits determine which of the possible 32 blocks is currently present in the cache. These are tag bits.
- Simple to implement but not very flexible.

Tag	Block	Word	
5	7	4	Main memory address

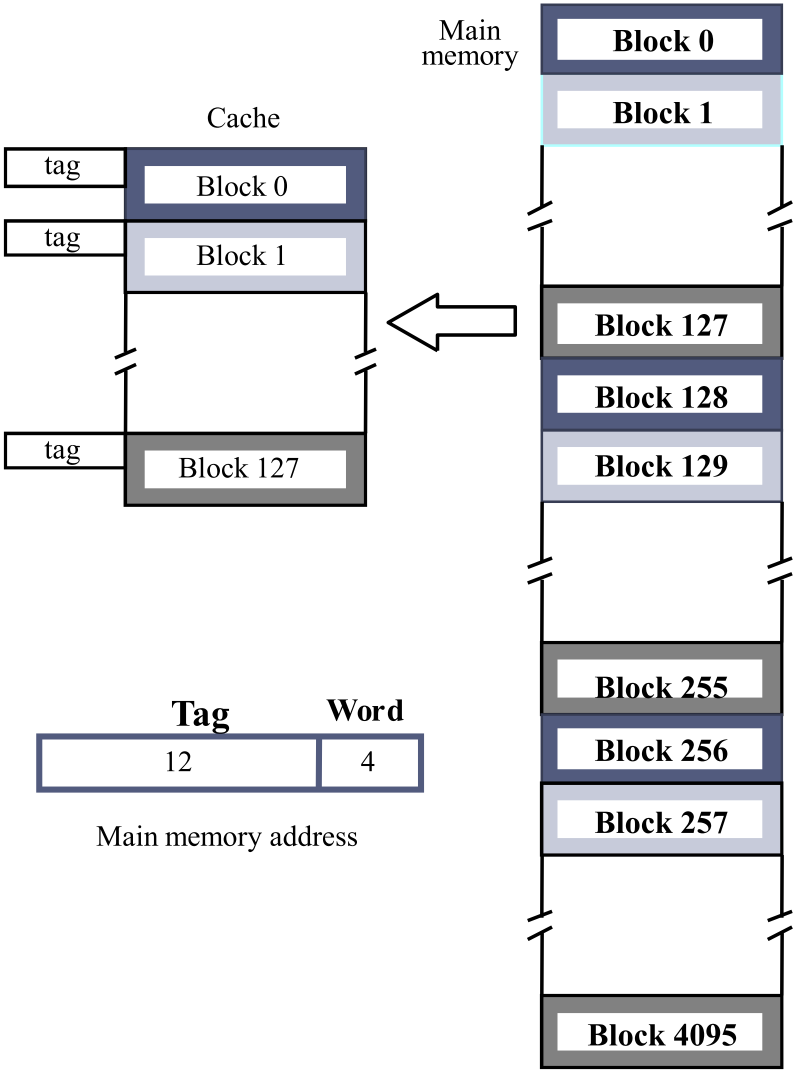
11101,1111111,1100

Tag: 11101

Block: 1111111=127, in the 127th block of the cache

Word:1100=12, the 12th word of the 127th block in the cache

Associative mapping



•Main memory block can be placed into any cache position.

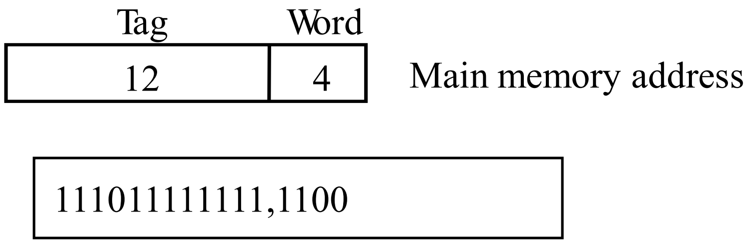
•Memory address is divided into two fields:

- Low order 4 bits identify the word within a block.

-*high order* 12 tag bits Identify which of the 4096 blocks that are resident in the cache $4096=2^{12}$.

•Flexible, and uses cache space efficiently.

•Replacement algorithms can be used to replace an existing block in the cache when the cache is full.



Tag Word

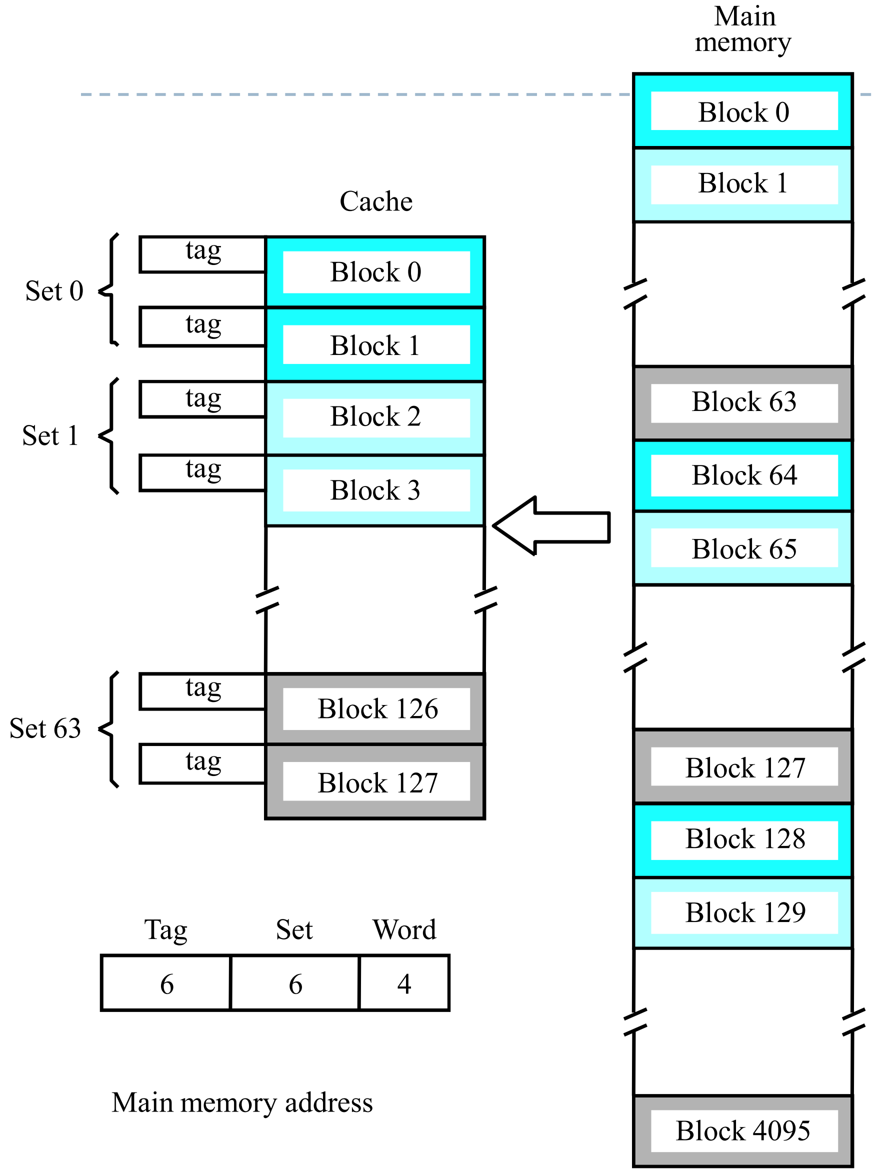
12 4 Main memory address

111011111111,1100

Tag: 111011111111

Word:1100=12, the 12th word of a block in the cache

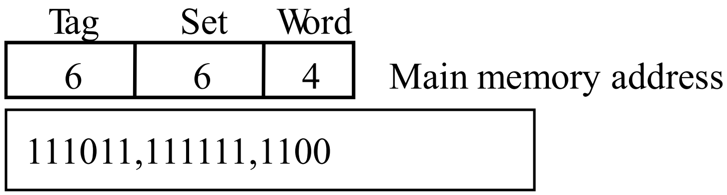
Set-Associative Mapping



Tag	Set	Word
6	6	4

Main memory address

- Blocks of cache are grouped into sets.
- Mapping function allows a block of the main memory to reside in any block of a specific set.
- Divide the cache into 64 sets, with two blocks per set.
- Memory address is divided into three fields:
  - 6 bit Set field determines the set number.
  - High order 6 bit fields are compared to the tag fields of the two blocks in a set.
- Number of blocks per set is a design parameter.



Tag	Set	Word
6	6	4

Main memory address

111011,111111,1100

Tag: 111011

Set: 111111=63, in the 63th set of the cache

Word:1100=12, the 12th word of the 63th set in the cache

Replacement Algorithms

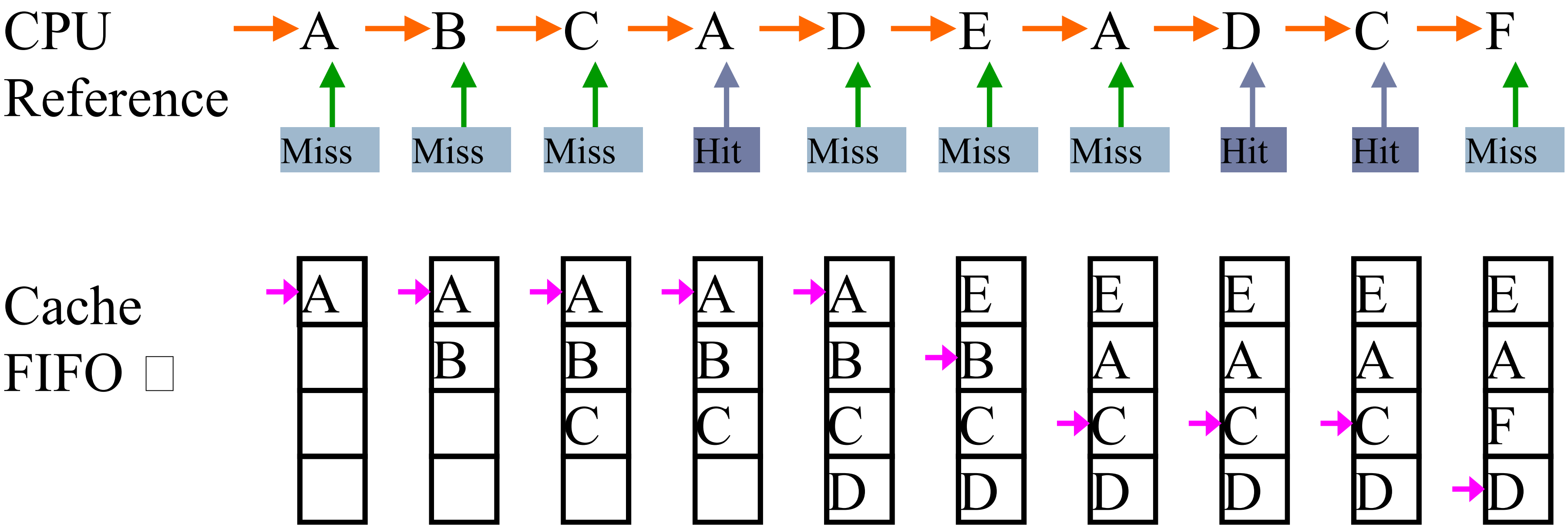
- Difficult to determine which blocks to kick out
- The cache controller tracks references to all blocks as computation proceeds.
- Increase / clear track counters when a hit/miss occurs

- For Associative & Set-Associative Cache

  Which location should be emptied when the cache is full and a miss occurs?

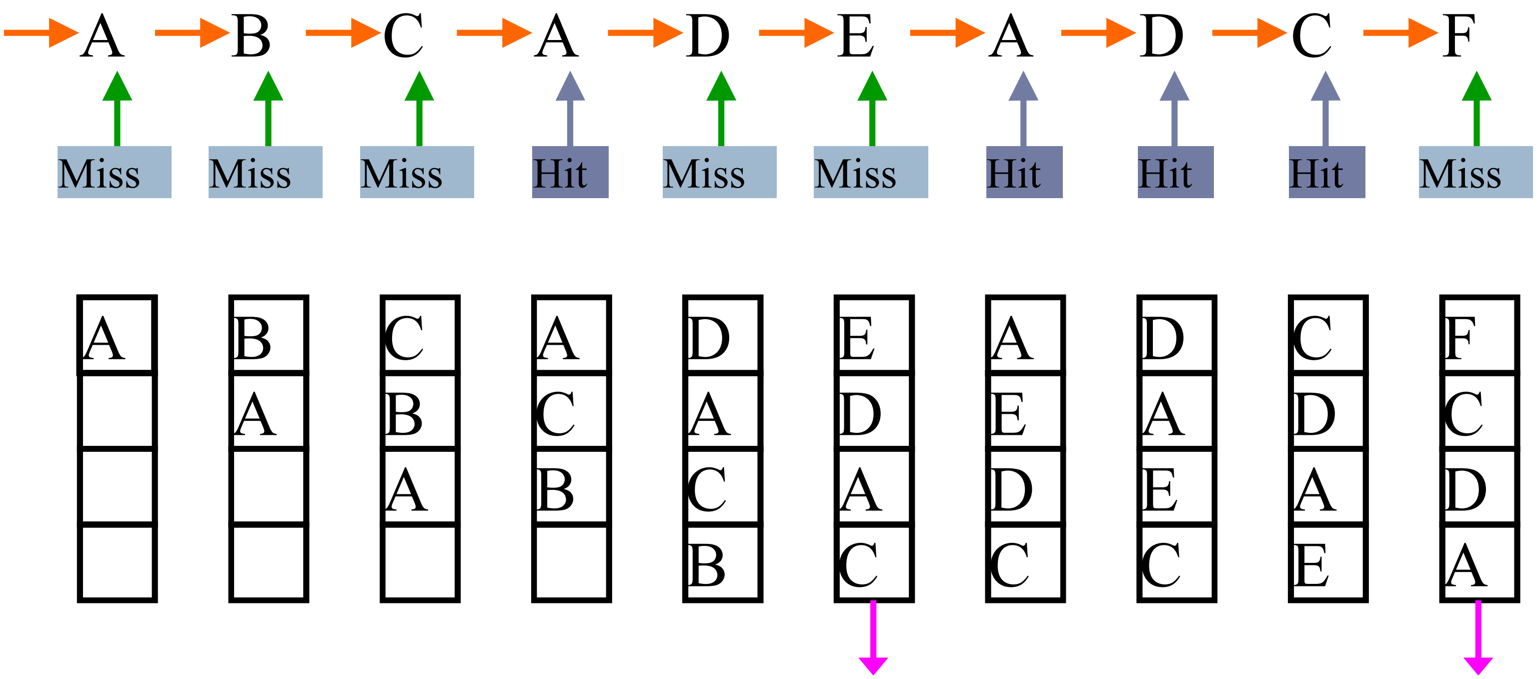
  - First In First Out (FIFO)
  - Least Recently Used (LRU)

Replacement Algorithms



CPU Reference	→A	→B	→C	→A	→D	→E	→A	→D	→C	→F
	Miss	Miss	Miss	Hit	Miss	Miss	Miss	Hit	Hit	Miss
Cache FIFO □	→A	→A	→A	→A	→A	E	E	E	E	E
		B	B	B	B	→B	A	A	A	A
			C	C	C	C	→C	→C	→C	F
					D	D	D	D	D	→D

Replacement Algorithms



CPU Reference	A	B	C	A	D	E	A	D	C	F
	Miss	Miss	Miss	Hit	Miss	Miss	Hit	Hit	Hit	Miss
Cache LRU □	A	B	C	A	D	E	A	D	C	F
		A	B	C	A	D	E	A	D	C
			A	B	C	A	D	E	A	D
					B	C	C	C	E	A

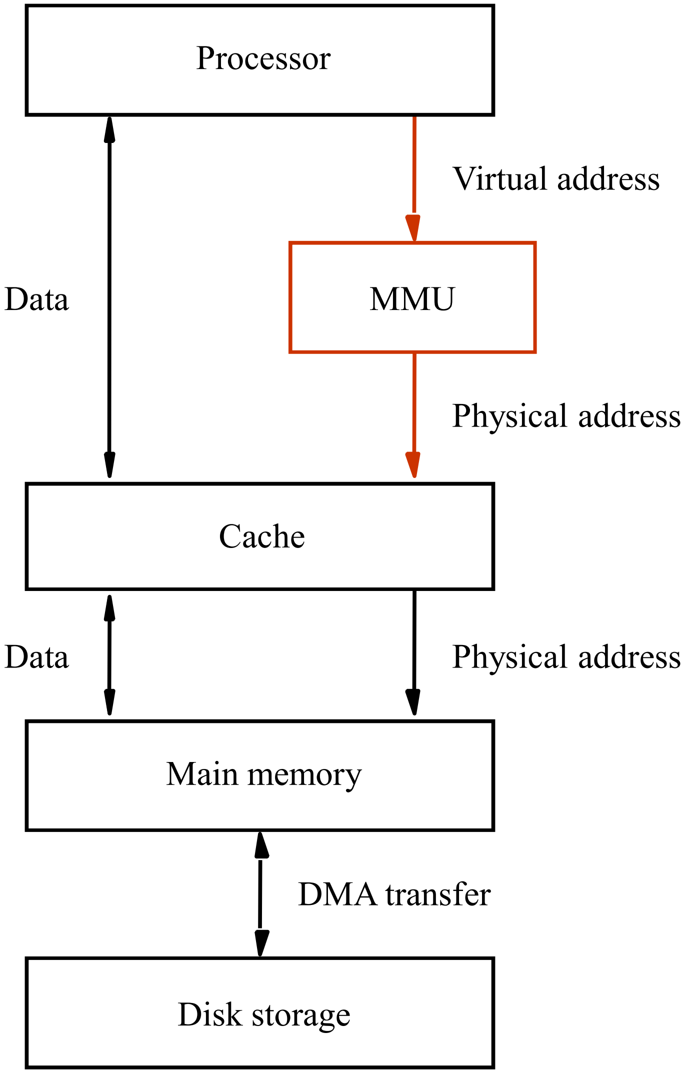
Virtual Memories

- Main memory smaller than address space

  Example: 32-bit address allows an address space of 4G bytes, but main memory may only be a few hundred megabytes.

- Parts of program not in main memory are stored on secondary storage devices, such as disks.
- Techniques that automatically move program and data blocks into the physical main memory when they are required for execution are called virtual-memory techniques.
- Operating system moves programs and data automatically between the physical main memory and secondary storage (virtual memory).

Virtual memory organization

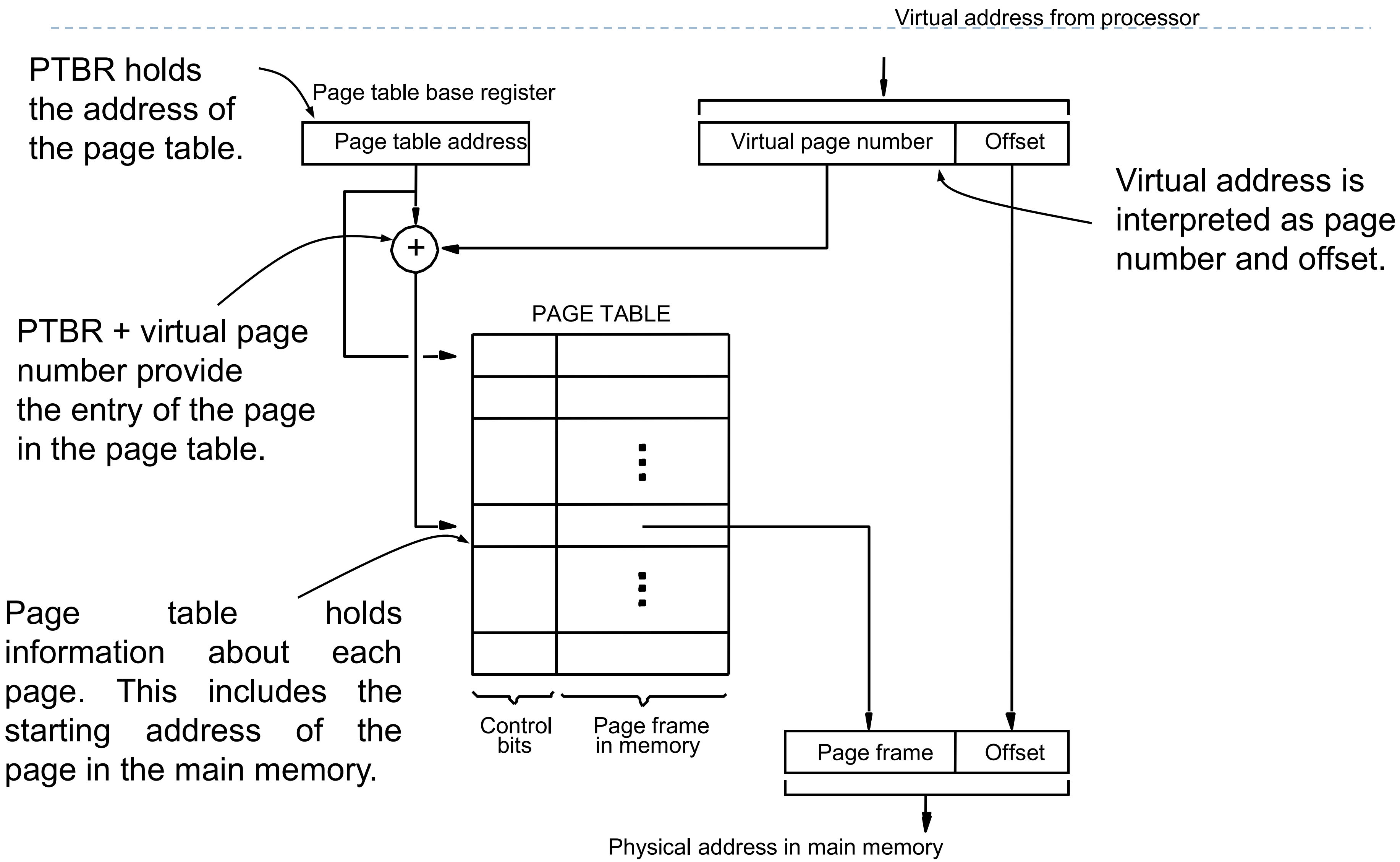


- Memory management unit (MMU) translates virtual addresses into physical addresses.
- If the desired data or instructions are in the main memory they are fetched as described previously.
- If the desired data or instructions are not in the main memory, they must be transferred from secondary storage to the main memory.
- MMU causes the operating system to bring the data from the secondary storage into the main memory.
- Virtual addresses will be translated into physical addresses.
- The virtual memory mechanism bridges the size and speed gaps between the main memory and secondary storage – similar to cache

Address translation

- Assume that program and data are composed of fixed-length units called pages. A page consists of a block of words that occupy contiguous locations in the main memory.
- Page is a basic unit of information that is transferred between secondary storage and main memory.
- Size of a page commonly ranges from 2K to 16K bytes.
- Each virtual or logical address generated by a processor is interpreted as a virtual page number (high-order bits) plus an offset (low-order bits) that specifies the location of a particular byte within that page.

- Information about the main memory location of each page is kept in the page table.
- Area of the main memory that can hold a page is called as page frame.
- Starting address of the page table is kept in a page table base register.
- Page table entry for a page includes:
  - Address of the page frame where the page resides in the main memory.
  - Some control bits.
- Virtual page number generated by the processor is added to the contents of the page table base register.
  - This provides the address of the corresponding entry in the page table.
  - The contents of this location in the page table give the starting address of the page if the page is currently in the main memory.



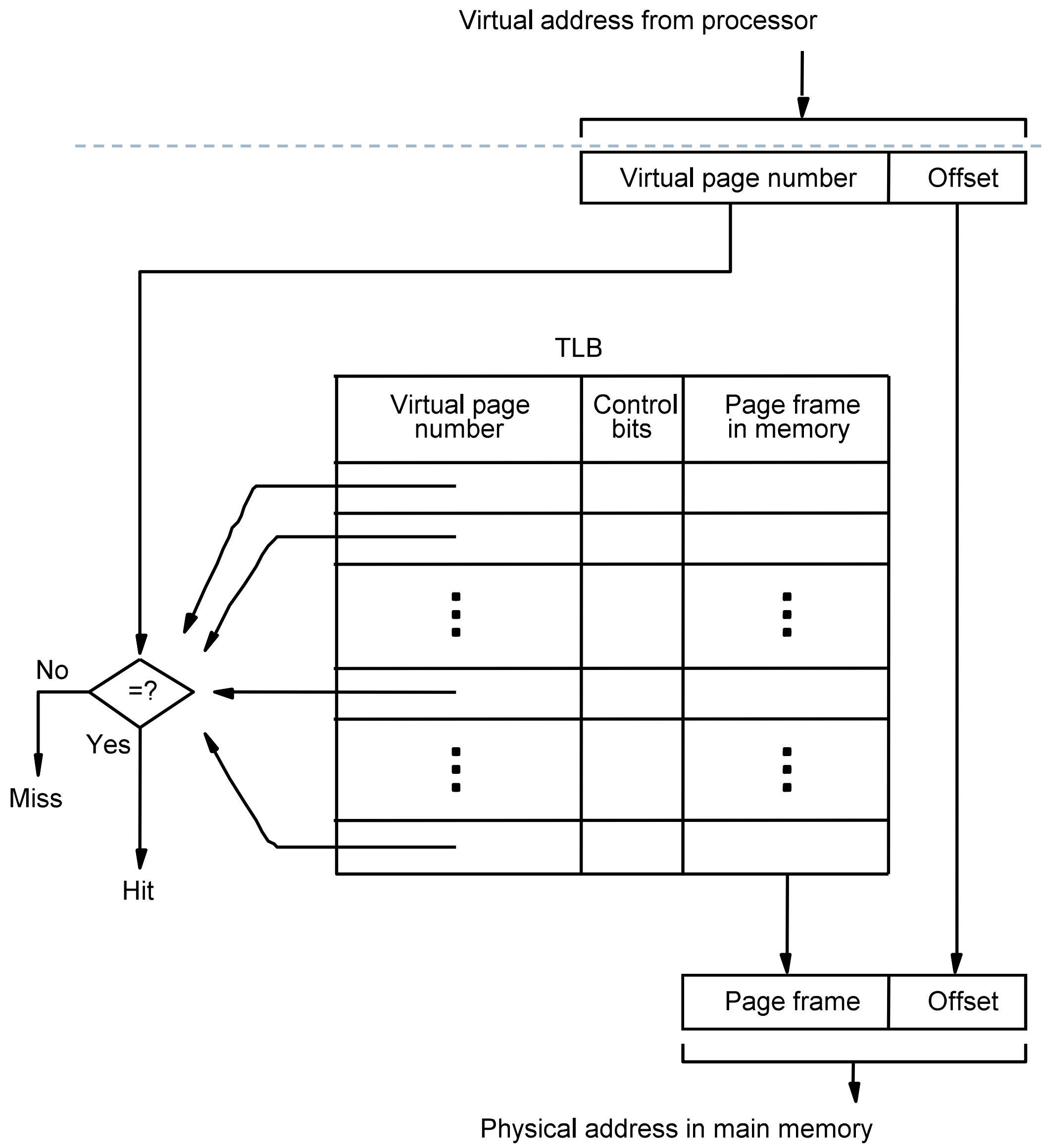
PTBR holds the address of the page table.

Virtual address is interpreted as page number and offset.

PTBR + virtual page number provide the entry of the page in the page table.

Page table holds information about each page. This includes the starting address of the page in the main memory.

- Page table entry for a page also includes some control bits which describe the status of the page while it is in the main memory.
- The page table information is used by the MMU for every access, so it is supposed to be with the MMU. which consists of the page table entries that correspond to the most recently accessed pages,
- A small cache called as Translation Lookaside Buffer (TLB) is included in the MMU.
  - TLB holds page table entries of the most recently accessed pages.
- In addition to the above for each page, TLB must hold the virtual page number for each page.



- High-order bits of the virtual address generated by the processor select the virtual page.
- These bits are compared to the virtual page numbers in the TLB.
- If there is a match, a hit occurs and the corresponding address of the page frame is read.
- If there is no match, a miss occurs and the page table within the main memory must be consulted.

- if a program generates an access to a page that is not in the main memory, a page fault is occur.
- Upon detecting a page fault by the MMU, following actions occur:
  - MMU asks the operating system to intervene by raising an exception.
  - Processing of the active task which caused the page fault is interrupted.
  - Control is transferred to the operating system.
  - Operating system copies the requested page from secondary storage to the main memory.
  - Once the page is copied, control is returned to the task which was interrupted.

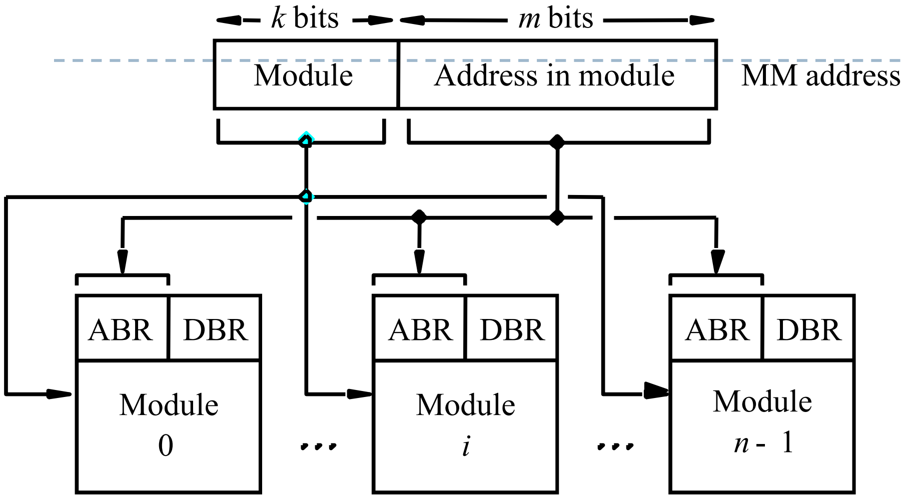
Performance considerations

- A key design objective of a computer system is to achieve the best possible performance at the lowest possible cost.
  - Price/performance ratio is a common measure of success.
- Performance of a processor depends on:
  - How fast machine instructions can be brought into the processor for execution.
  - How fast the instructions can be executed.

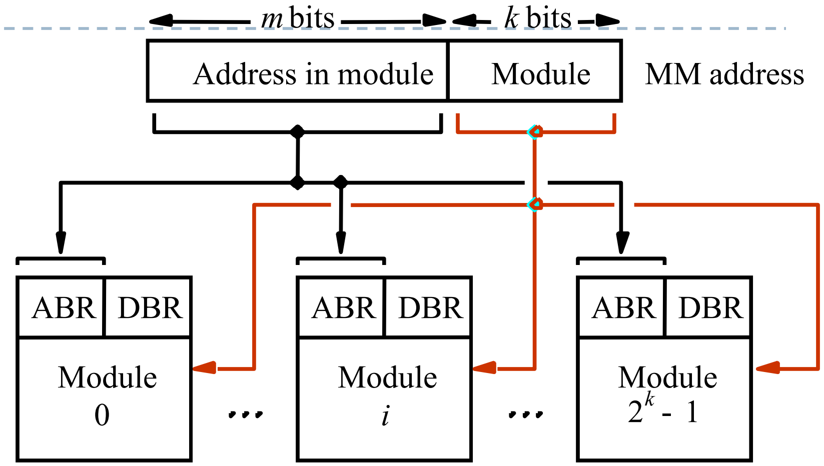
Interleaving

- Divides the memory system into a number of memory modules. Each module has its own address buffer register (ABR) and data buffer register (DBR).
- Arranges addressing so that successive words in the address space are placed in different modules.
- When requests for memory access involve consecutive addresses, the access will be to different modules.
- Since parallel access to these modules is possible, the average rate of fetching words from the Main Memory can be increased.

Methods of address layouts



Consecutive words in a module



Consecutive words in consecutive modules.

- Consecutive words are placed in a module.
- High-order k bits of a memory address determine the module.
- Low-order m bits of a memory address determine the word within a module.
- When a block of words is transferred from main memory to cache, only one module is busy at a time.

•Consecutive words are located in consecutive modules.

•Consecutive addresses can be located in consecutive modules.

•While transferring a block of data, several memory modules can be kept busy at the same time.

Hit Rate and Miss Penalty

- The success rate in accessing information at various levels of the memory hierarchy – hit rate / miss rate.
- A miss causes extra time needed to bring the desired information into the cache.
- Hit rate can be improved by increasing block size, while considering the cache size .
- T_{ave}=hC+(1-h)M
  - $T_{ave:}$ average access time experienced by the processor
  - h: hit rate
  - M: miss penalty, the time to access information in the main memory
  - C: the time to access information in the cache

Caches on the Processor Chip

- On chip vs. off chip
- Two separate caches for instructions and data or Single cache for both
- the advantage of separating caches – parallelism, better performance
- Level 1 and Level 2 caches, are used in high performance processors
- L1 cache – faster and smaller. Access more than one word simultaneously and let the processor use them one at a time. Is on the processor chip
- L2 cache – slower and larger. May be implemented externally using SRAM chips.
- Average access time: $t_{ave} = h_1C_1 + (1-h_1)h_2C_2 + (1-h_1)(1-h_2)M$
  where h is the hit rate, C is the time to access information in cache, M is the time to access information in main memory.

Other Performance Enhancements

Write buffer

- Write-through:
  - Each write operation involves writing to the main memory.
  - If the processor has to wait for the write operation to be complete, it slows down the processor.
  - Processor does not depend on the results of the write operation.
  - Write buffer can be included for temporary storage of write requests.
  - Processor places each write request into the buffer and continues execution.
- Write-back:
  - Block is written back to the main memory when it is replaced.

Prefetching

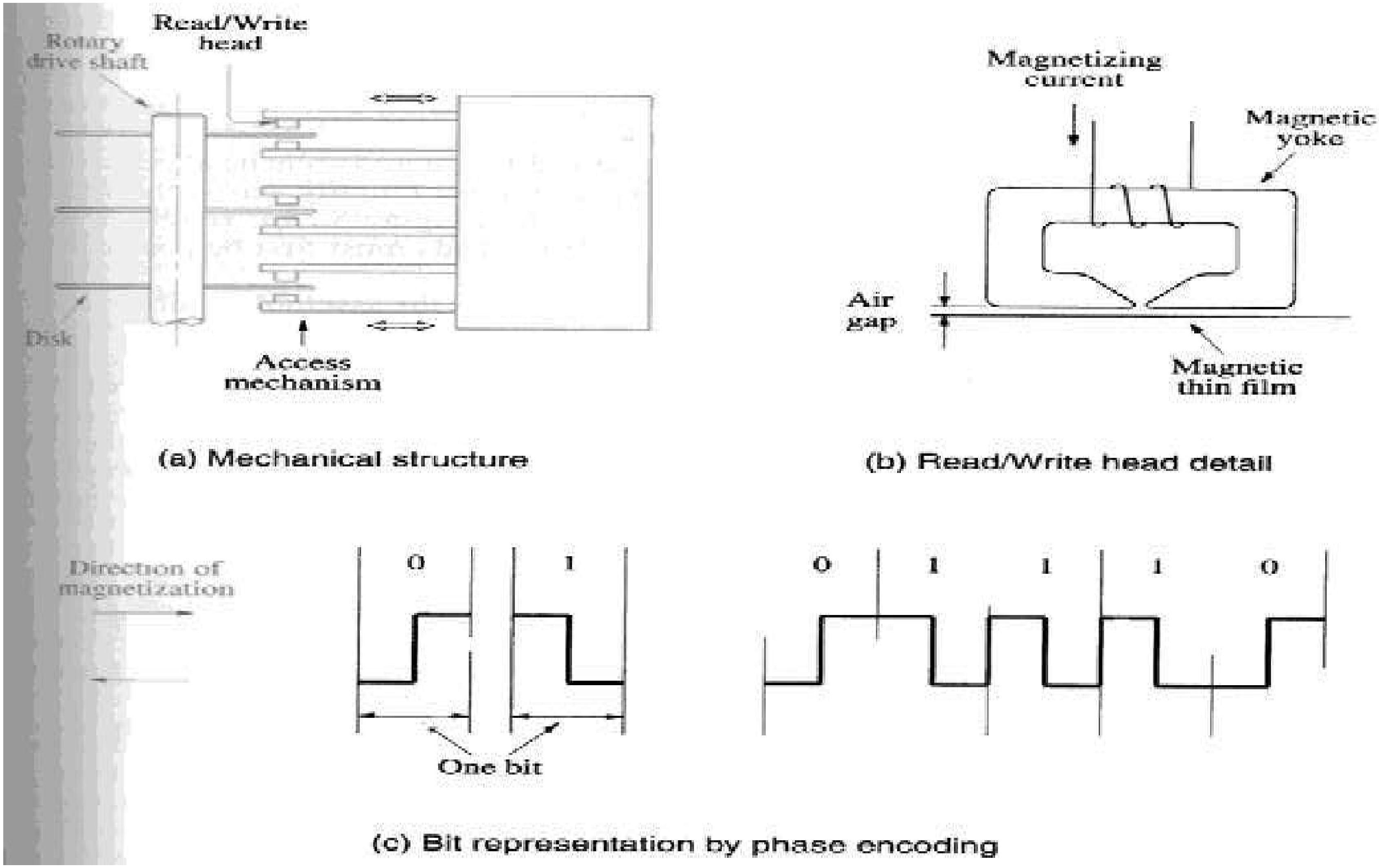
- New data are brought into the processor when they are first needed.
- Processor has to wait before the data transfer is complete.
- Prefetch the data into the cache before they are actually needed, or before a Read miss occurs.
- Prefetching can be accomplished through software by including a special instruction in the machine language of the processor.
  - Inclusion of prefetch instructions increases the length of the programs.
- Prefetching can also be accomplished using hardware:
  - Circuitry that attempts to discover patterns in memory references and then prefetches according to this pattern.

Lockup-Free Cache

- Prefetching scheme does not work if it stops other accesses to the cache until the prefetch is completed.
- A cache of this type is said to be “locked” while it services a miss.
- Cache structure which supports multiple outstanding misses is called a lockup free cache.
- Since only one miss can be serviced at a time, a lockup free cache must include circuits that keep track of all the outstanding misses.
- Special registers may hold the necessary information about these misses.

Secondary Storage

Magnetic Hard Disks



(a) Mechanical structure

(b) Read/Write head detail

(c) Bit representation by phase encoding

Organization of Data on a Disk

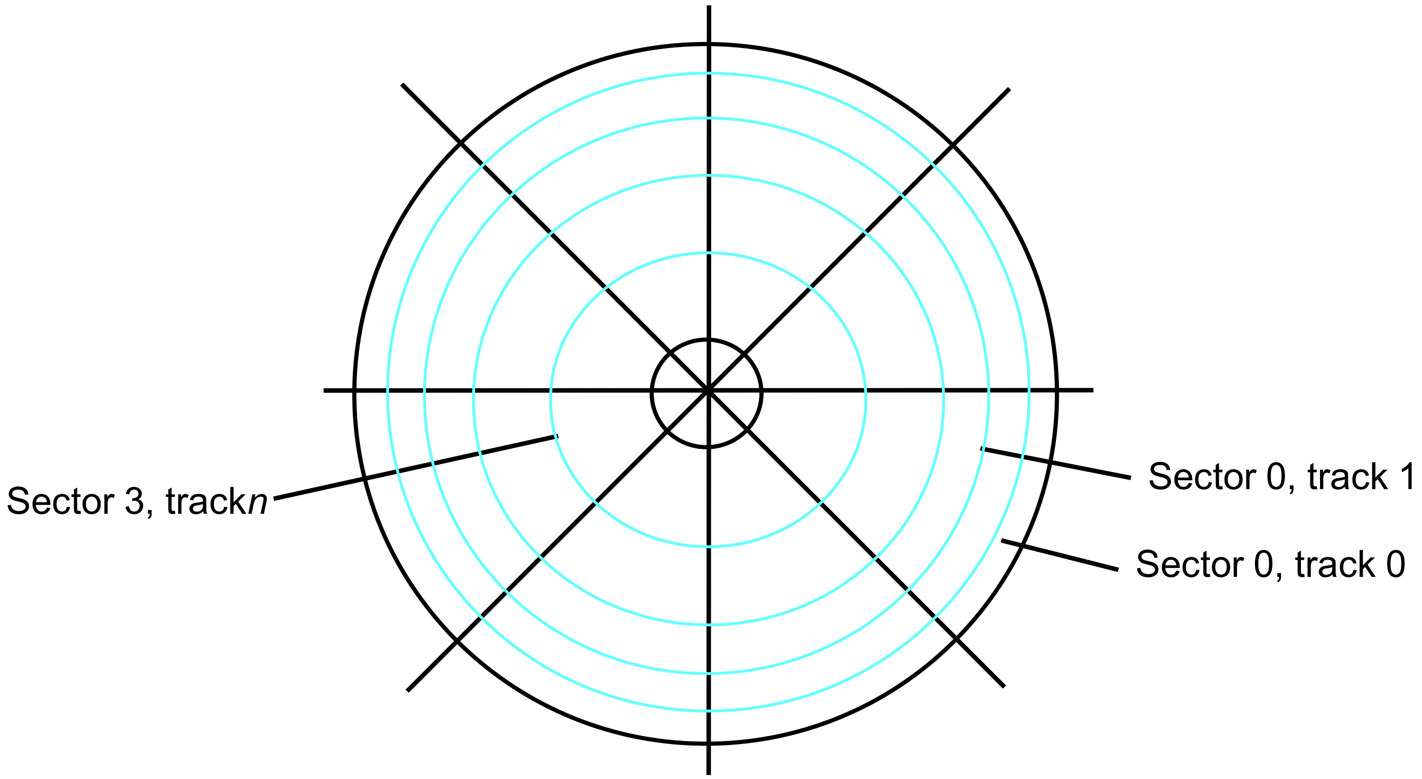


Figure 5.30. Organization of one surface of a disk.

Access Data on a Disk

- Sector header
- Following the data, there is an error-correction code (ECC).
- Formatting process
- Difference between inner tracks and outer tracks
- Access time – seek time / rotational delay (latency time)
- Data buffer/cache

Disk Controller

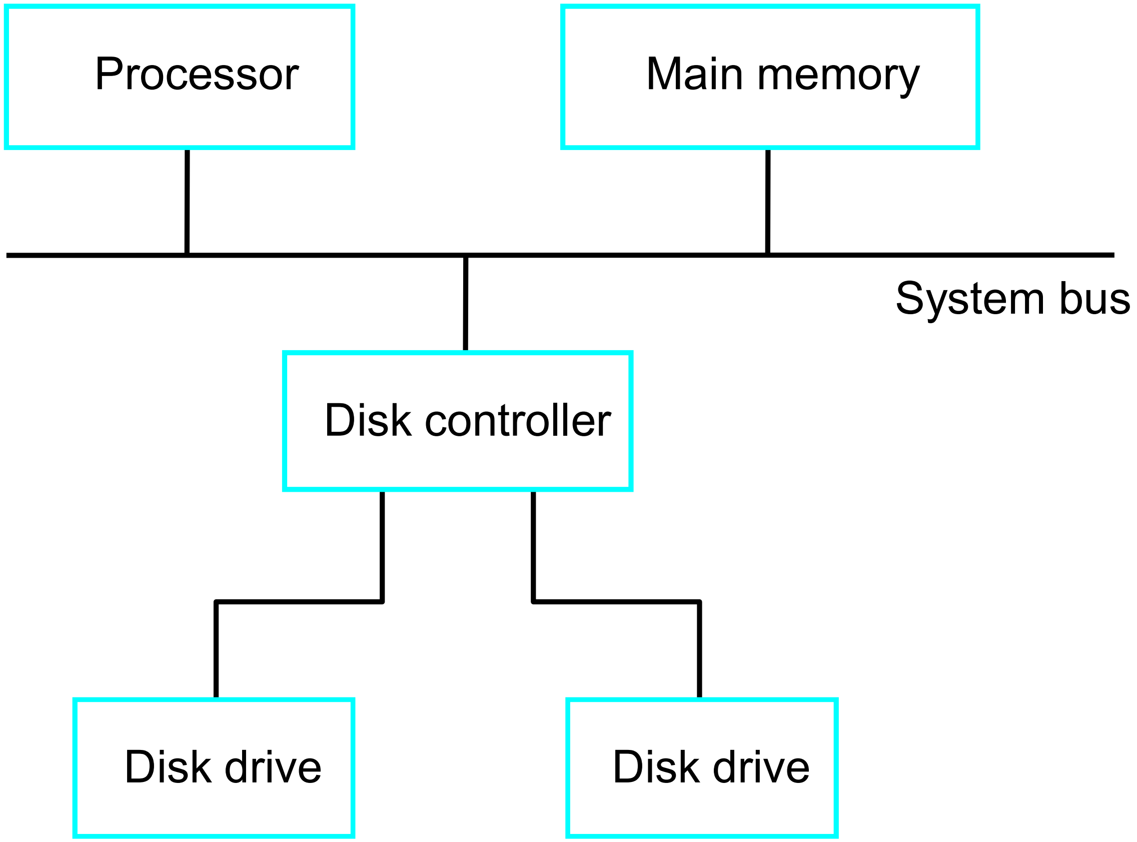


Figure 5.31. Disks connected to the system bus.

Disk Controller

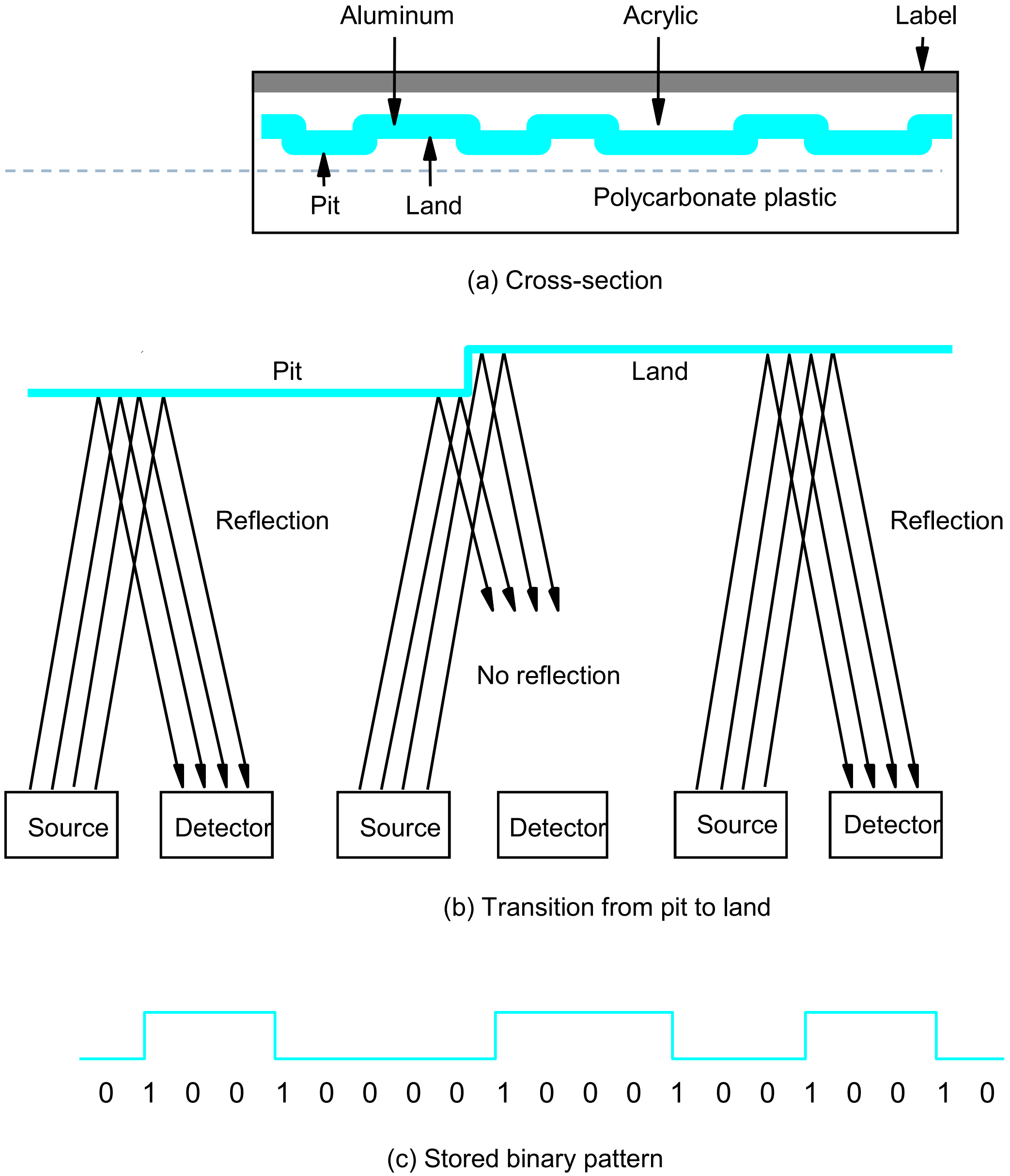
- Seek
- Read
- Write
- Error checking

RAID Disk Arrays

- Redundant Array of Inexpensive Disks
- Using multiple disks makes it cheaper for huge storage, and also possible to improve the reliability of the overall system.
- RAID0 – data striping
- RAID1 – identical copies of data on two disks
- RAID2, 3, 4 – increased reliability
- RAID5 – parity-based error-recovery

Optical Disks

- CD-ROM
- CD-Recordable (CD-R)
- CD-ReWritable (CD-RW)
- DVD
- DVD-RAM



(a) Cross-section

(b) Transition from pit to land

(c) Stored binary pattern

Figure 5.32. Optical disk.

Magnetic Tape Systems

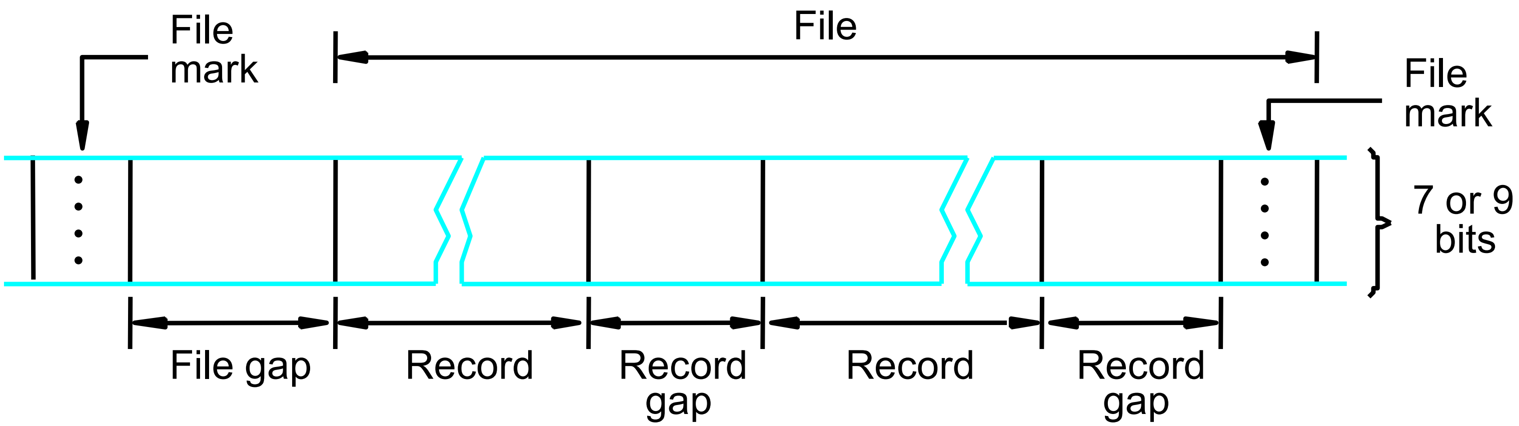


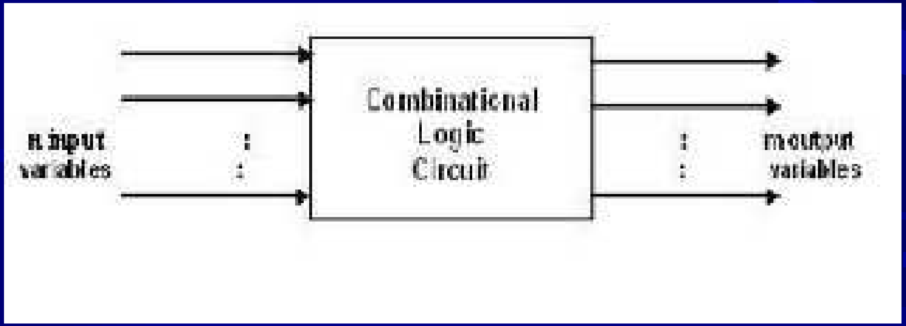
Figure 5.33. Organization of data on magnetic tape.

overview

- ***What is combinational logic circuit ?***
- ***Examples of combinational logic circuits***
- ***Binary-adder***
- ***Binary-subtractor***
- ***Binary-multiplier***
- ***Decoders***
- ***Multiplexers***

Combinational logic circuit

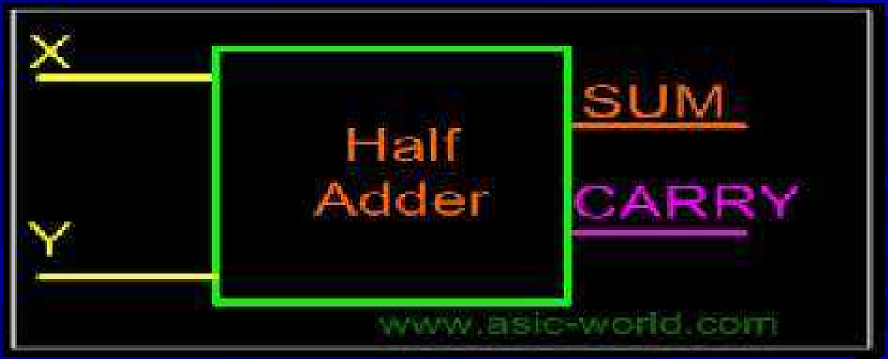
- Combinational circuits consists of logic gates whose outputs depends on the present inputs .They have no memory element .
- It consists of input variables , logic gates & output variables .



Binary-Adder

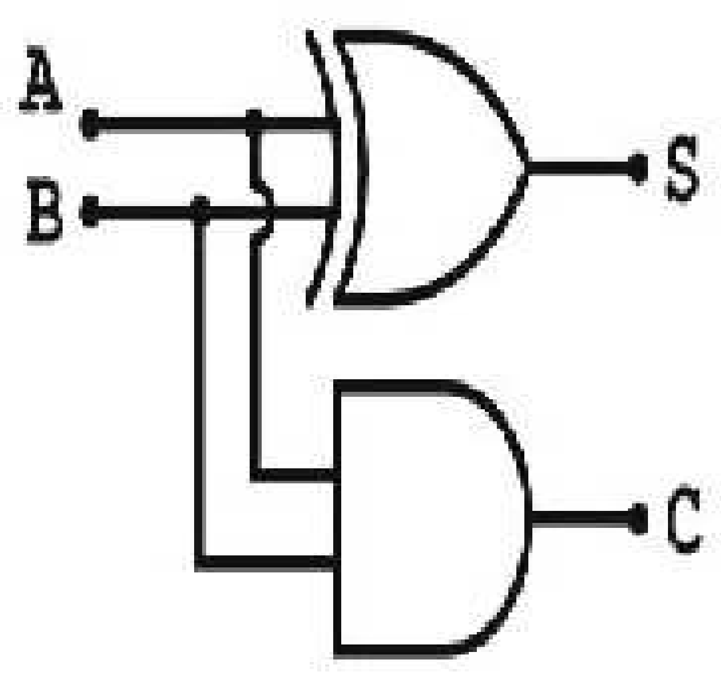
- ***Half-Adder :***
- ***_A combinational circuit that performs the addition of two bits at a time is called " Full-Adder ‘’ .***
- ***The input variables designate the augend and addend bits ; the output variables produce the sum & carry .______***

_



Circuit & truthtable of Half-Adder

- **Circuit diagram**

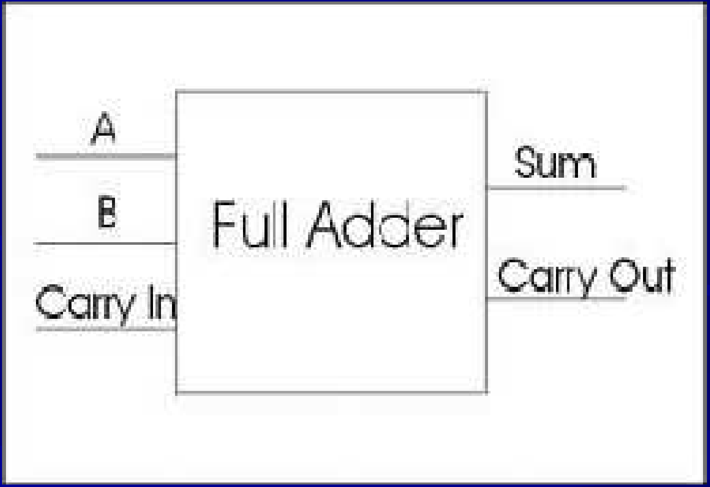


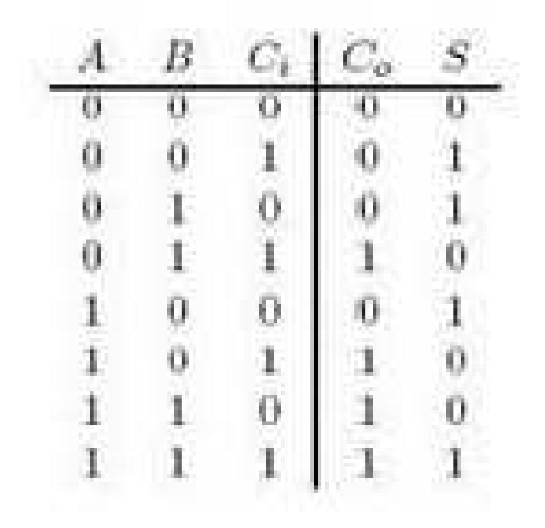
- **Truthtable**

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Full-Adder :

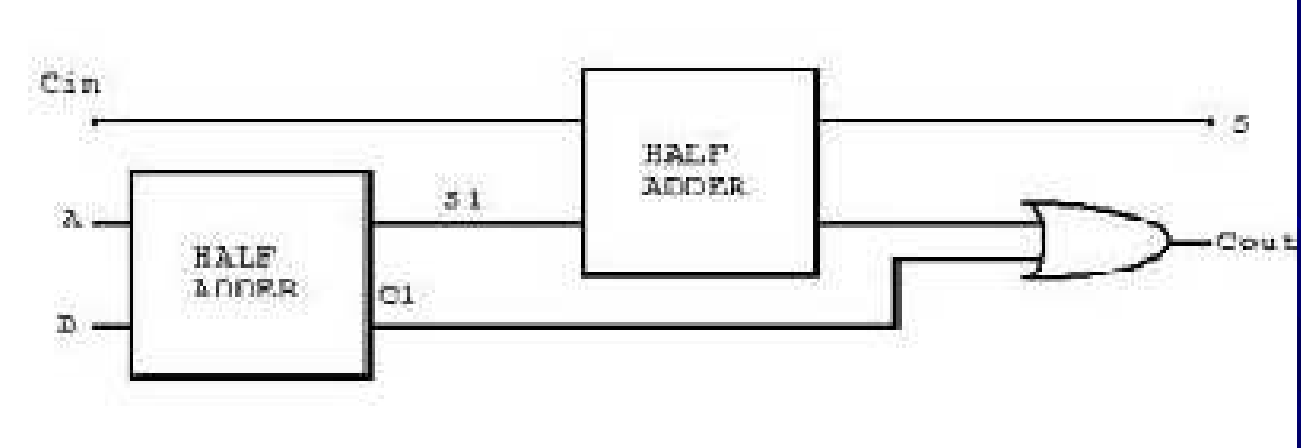
- ***A combinational circuit that performs the addition of three bits at a time is called " Full-Adder "***
- ***Block diagram & the truth table of Full-Adder .***

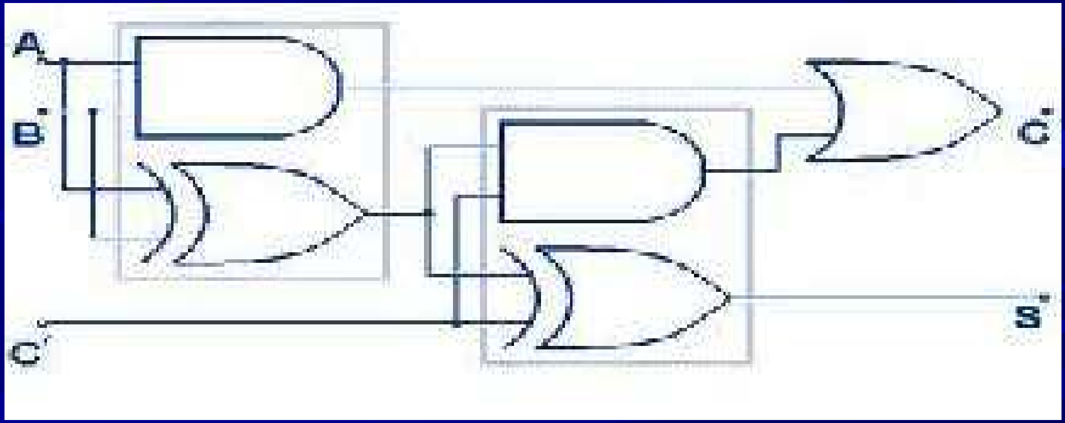




A	B	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Circuit diagrams of full-adder

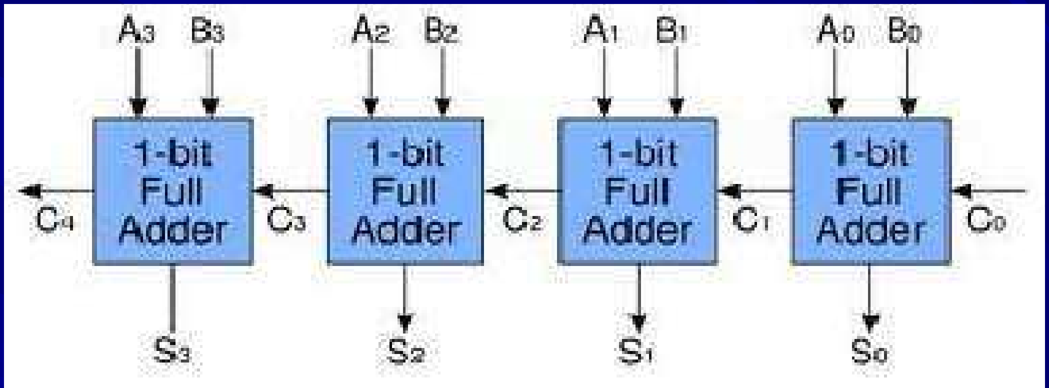




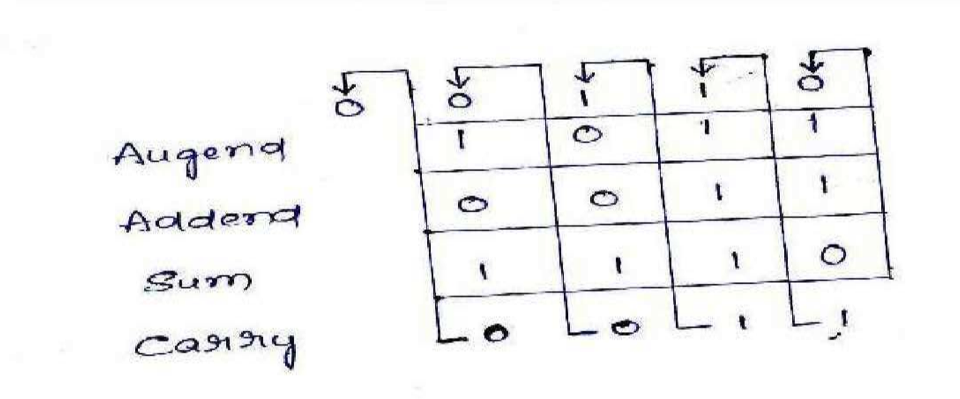
Binary-Adder

- *A Binary-Adder is a digital circuit that produces the arthimatic sum of two binary numbers at a time .*
- *It can also construct by cascading no of full-adders we get a N-bit binary-adder circuit .*
- *An N-bit adder requires n full-adders with each output carry connected to the input carry to the next full-adder .*

- For adding 4-bit binary numbers we need four full-adders as shown below :



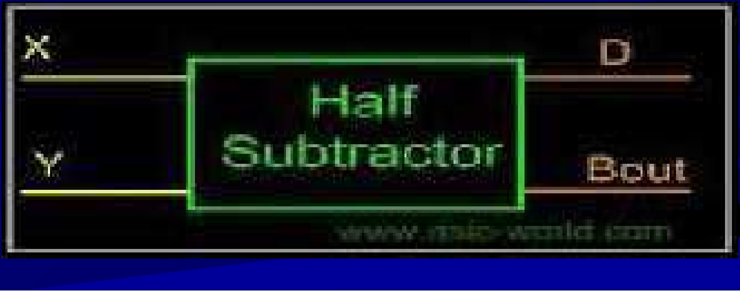
consider two binary numbers A=1011 & B =0011



	0	0	1	1	0
Augend		1	0	1	1
Addend		0	0	1	1
Sum		1	1	1	0
Carry		0	0	1	1

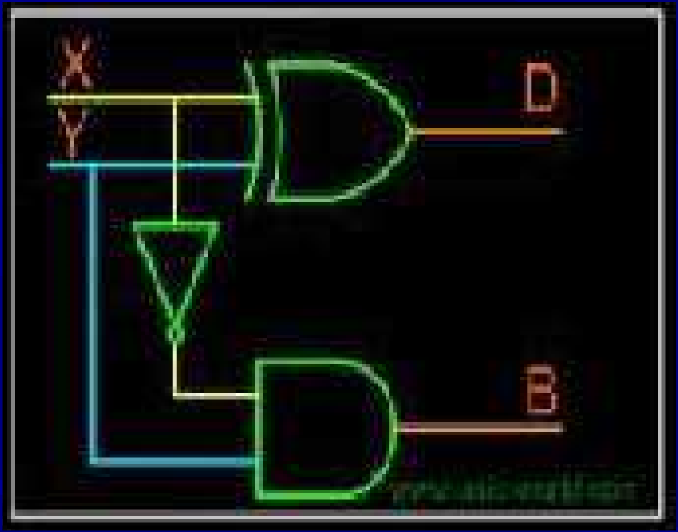
Binary-subtractor

- **Half-subtractor :**
- ***_A combinational circuit which performs the subtraction of two bits at a time is called " Half-subtractor " .***
- ***It has two inputs & two outputs . The two inputs x & y form the minuend & subtrahend & D is the difference output & B is the borrow output .***

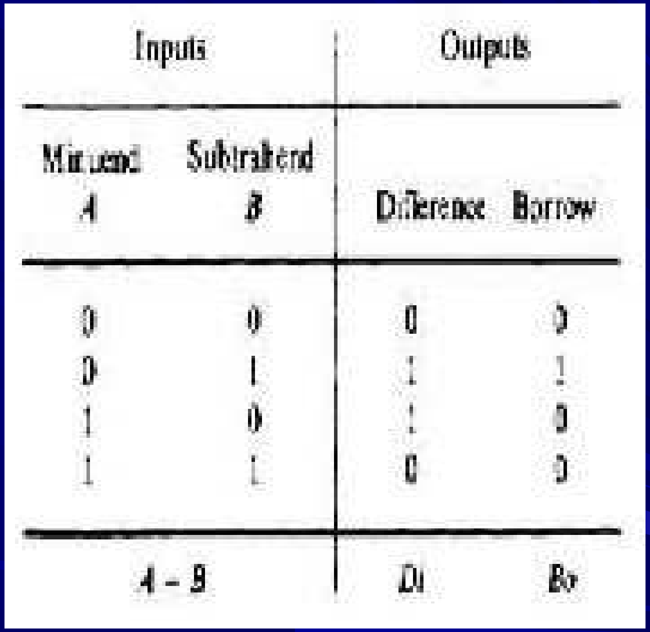


Circuit & truthtable of half-subtractor

- ***circuit diagram***



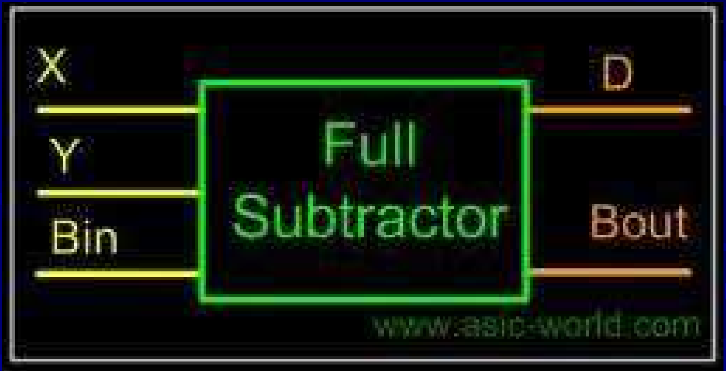
- ***Truthtable***



Inputs		Outputs	
Minuend A	Subtrahend B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0
A - B		Di	Bo

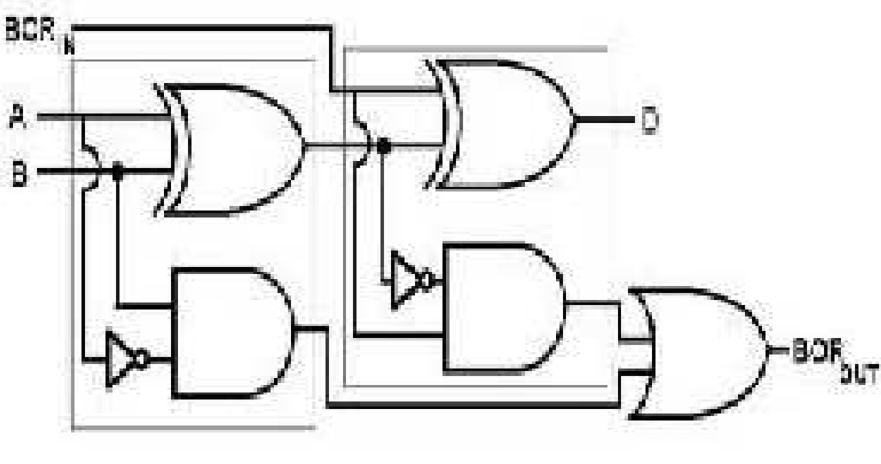
Full subtractor

- ***A combinational circuit which performs the subtraction of three bits at a time is called " Full subtractor " .***
- ***A full subtractor has three inputs x y Bin & two outputs D & Bout .***



Circuit & truthtable of full subtractor

- **circuit diagram**



- **Truthtable**

X Y b	D	B
000	0	0
001	1	1
010	1	1
011	0	1
100	1	0
101	0	0
110	0	0
111	1	1

Binary-subtractor

- ***The subtraction can be done by taking the 2’s compliment of B and adding it to A .***
- ***2’s compliment can be obtained by taking the 1’s compliment and adding 1 to least significant bit .***
- ***1’s compliment can be obtained by changing 1’s into 0’s & 0’s into 1’s .***

Four-bit adder-subtractor

154 Digital Design

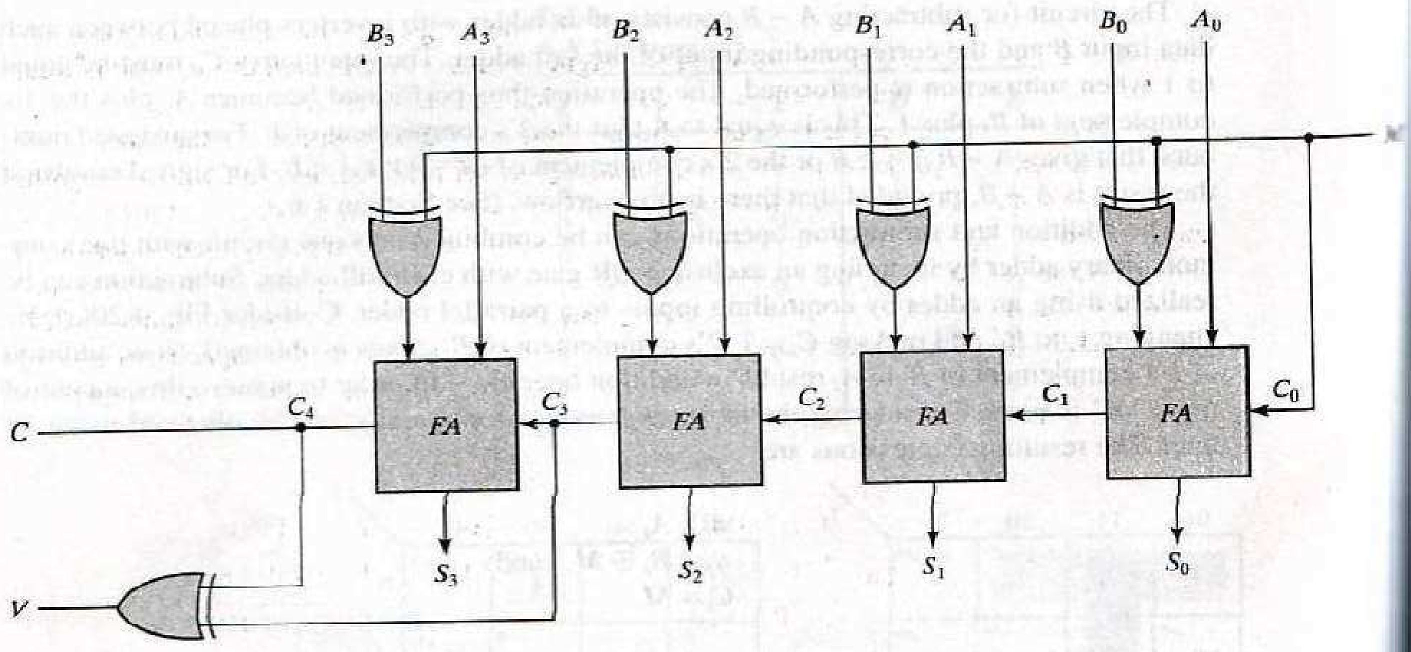


FIGURE 4.20
(b) Four-bit adder–subtractor

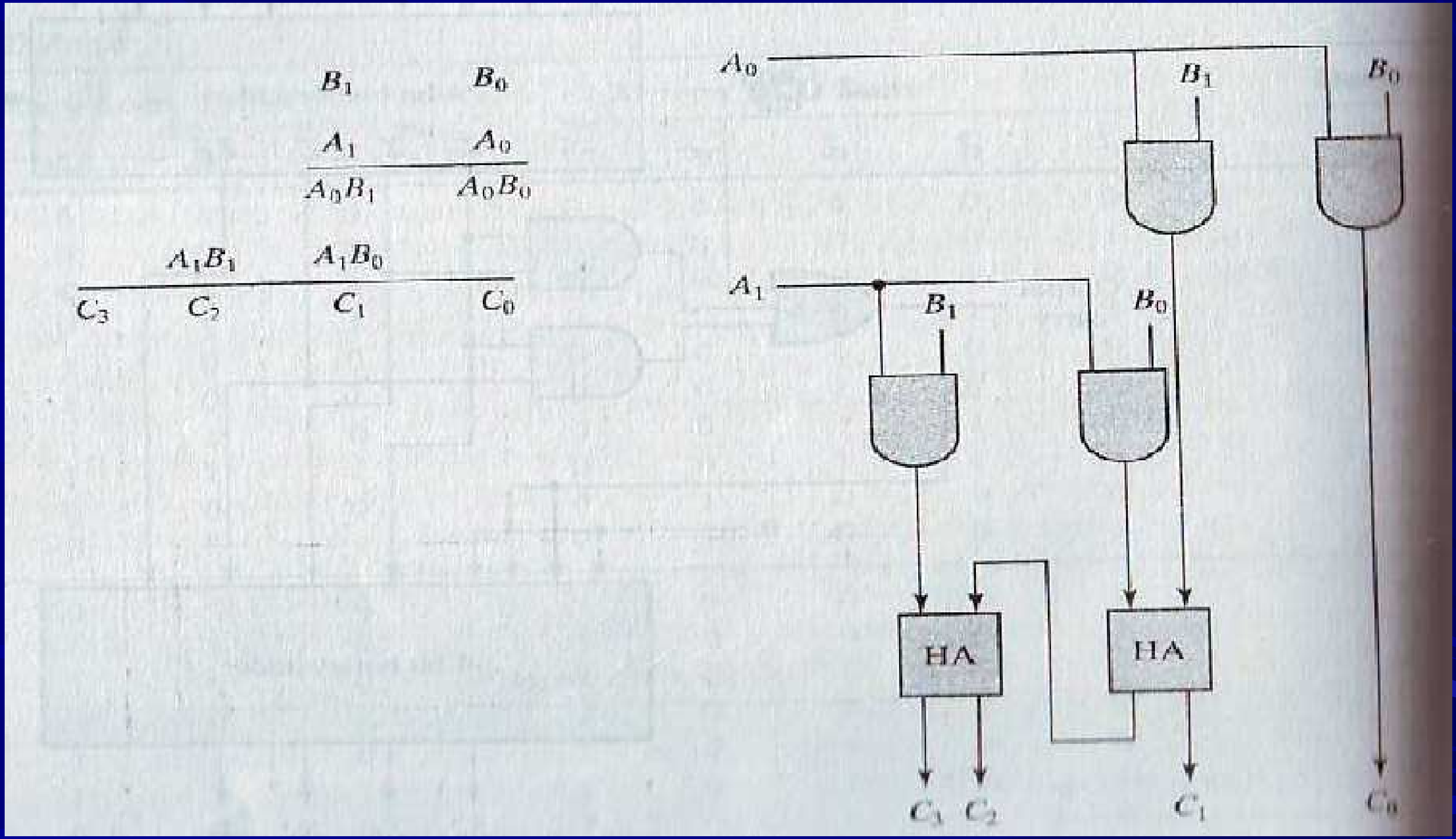
Continues…..

- In this circuit the mode input m controls the operation . When m=0 , the circuit is an adder circuit . When m=1, the circuit becomes a subtractor .
- The c & v are two outputs . C is for carry & v is for overflow . If v=0 after an addition or subtraction ,then no overflow occurred and the n-bit result is correct . If v=1 , then the result of the operation contains n+1 bits .

Binary-multiplier

- ***The multiplication of binary numbers is performed in the same way as multiplication of decimal numbers .***
- ***If we take the multiplication of 2-bit numbers . The multiplicand bits are B1 & B0 , the multiplier bits are A1 & A0 and the product is C3,C2,C1,C0 .***

2-bit by 2-bit multiplier circuit

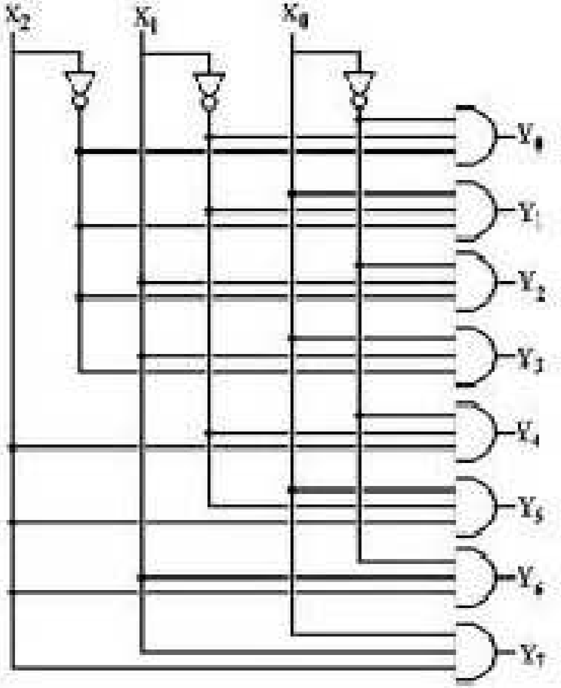


Decoders

- *A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n unique output lines .*
- *Generally a decoder is called as n to m line decoder .*
- *A decoder is used to convert binary form to any other desired form .*

Three - to - eight line decoder

- This three - to - eight line decoder is used to binary to octal conversions .



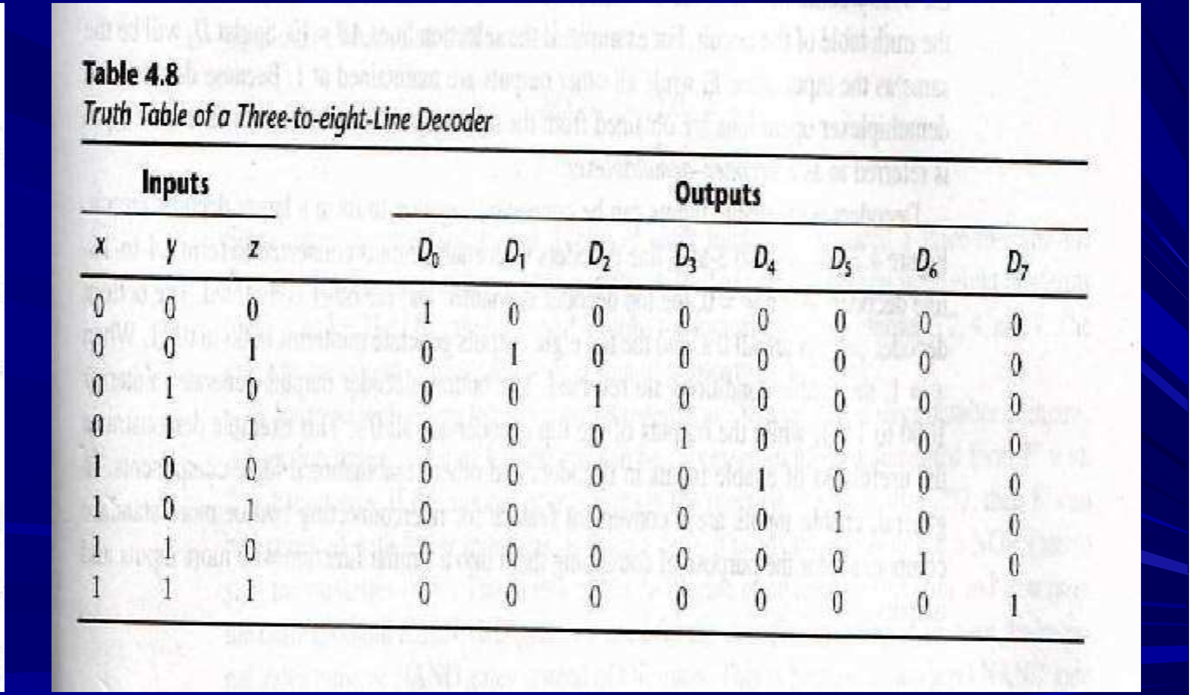


Table 4.8
Truth Table of a Three-to-eight-Line Decoder

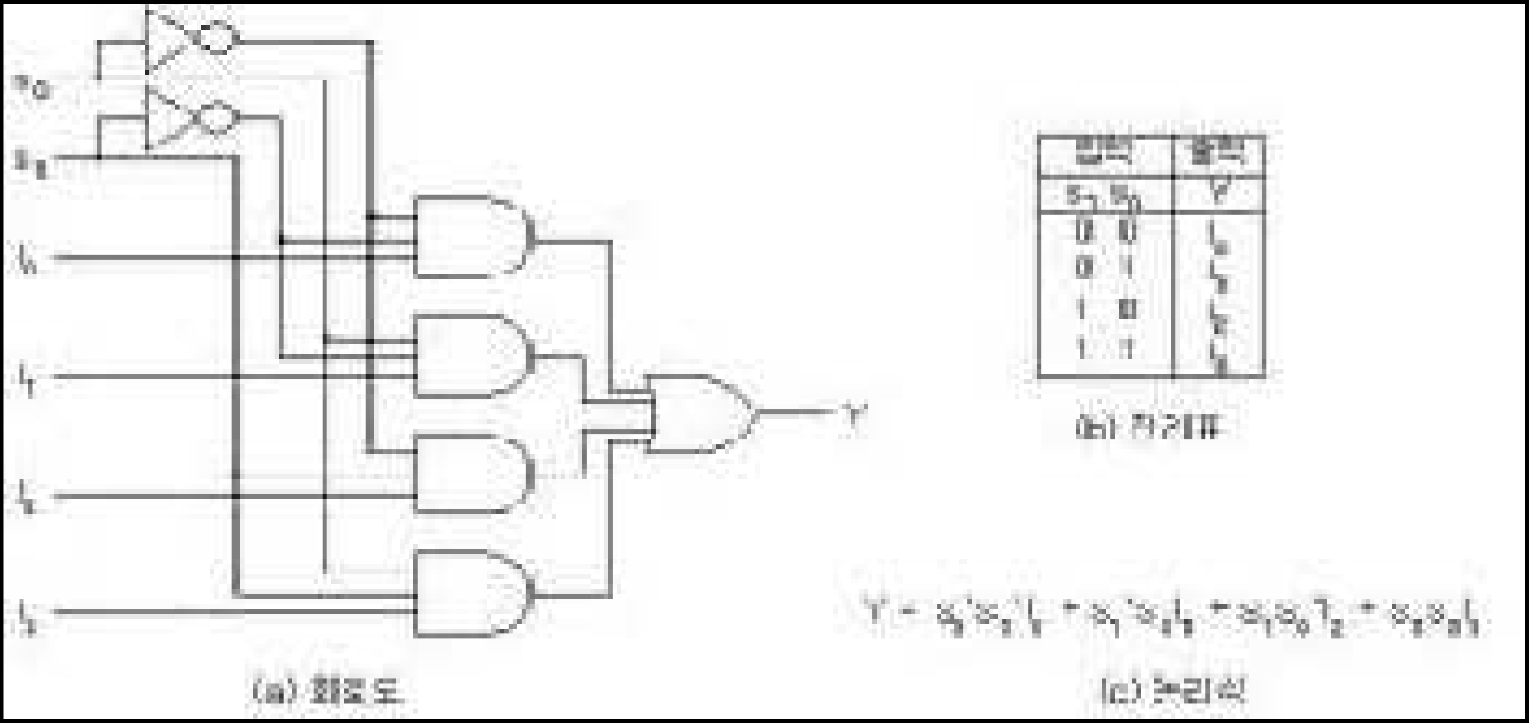
Inputs			Outputs							
x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Multiplexer

- ***multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line .***
- ***Normally there are 2n input lines and n selection lines whose bit combinations determines which input is selected .***
- ***A multiplexer is also called ‘ Data selector ’***

Four - to - one line multiplexer

circuit & truth table :



(a) 회로도

입력		출력
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

(b) 진리표

$$Y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

(c) 논리식

THANK YOU