UNIT -5

Memories and Asynchronous Sequential Logic

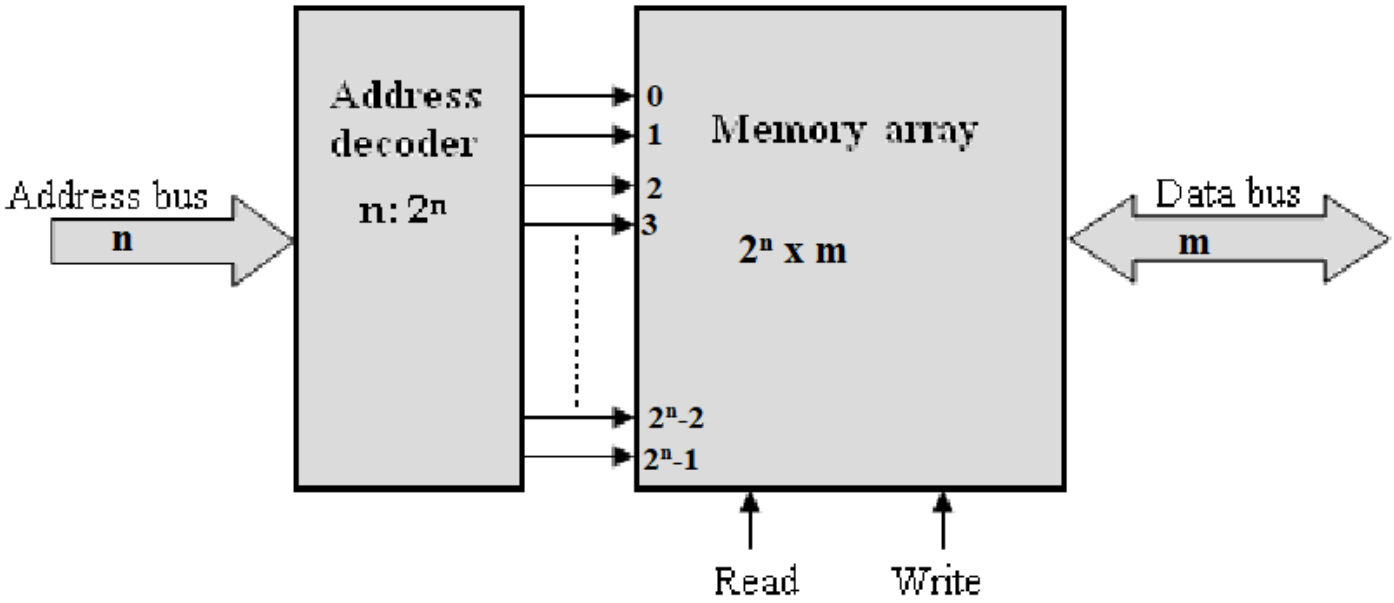
TOPIC – 1

Introduction to Memorie Devices

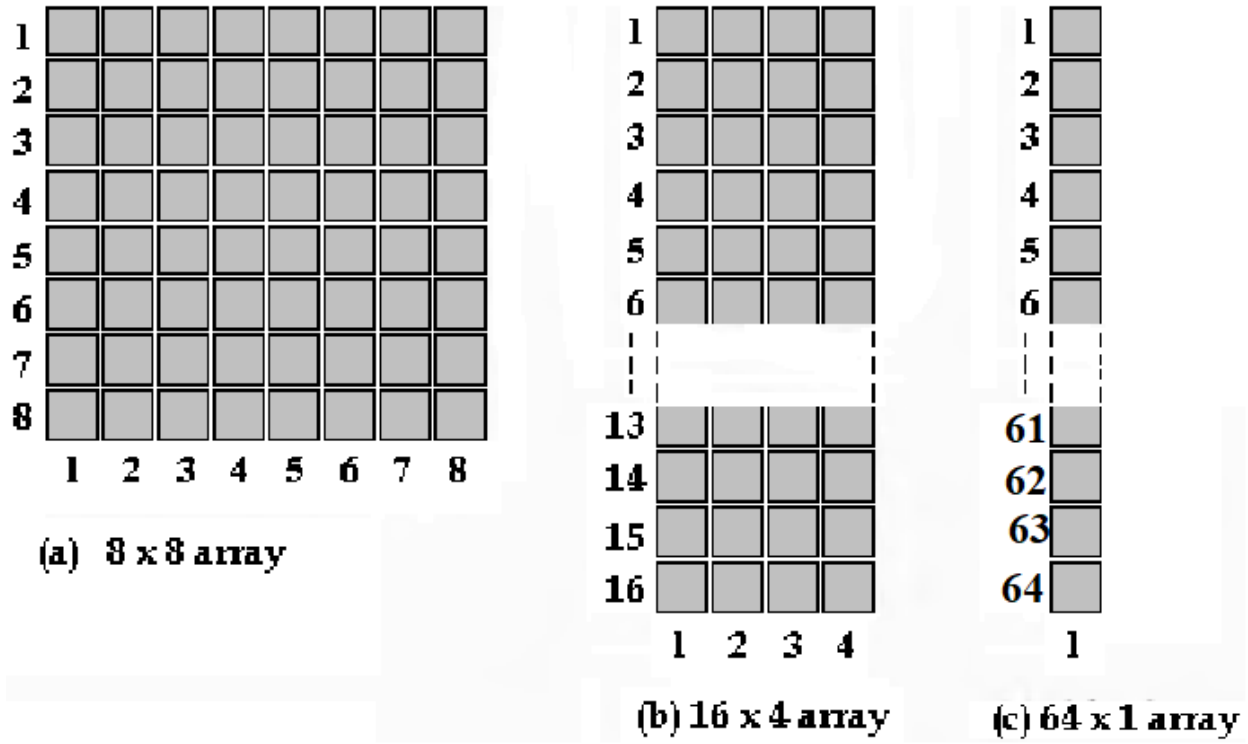
A memory unit is a collection of cells capable of storing a large quantity of binary information.

A memory unit is a device to which binary information is transferred for storage and from which information is retrieved when needed for processing.

In data processing takes place, information from memory is transferred to selected registers in the processing unit.



Intermediate and final results obtained in the processing unit are transferred back to be stored in memory. Binary information received from an input device is stored in memory, and information transferred to an output device is taken from memory.



Types Of Memories:

There are two types of memories that are used in digital systems:

- *Random-access memory* (RAM)
- *Read-only memory* (ROM).

RAM :

Ram stores new information for later use. The process of storing new information into memory is referred to as a memory *write* operation.

The process of transferring the stored information out of memory is referred to as a memory *read* operation. RAM can perform both write and read operations.

ROM :

It can perform only the read operation. This means that suitable binary information is already stored inside memory and can be retrieved or read at any time.However, that information cannot be altered by writing.

ROM is a *programmable logic device* (PLD).

The binary information that is stored within such a device is specified in some fashion and then embedded within the hardware in a process is referred to as *programming* the device.

The word "programming" here refers to a hardware procedure which specifies the bits that are inserted into the hardware configuration of the device.

ROM is one example of a PLD. Other such units are the programmable logic array (PLA), programmable array logic (PAL), and the field-programmable gate array (FPGA).

PLD'S:

A PLD is an integrated circuit with internal logic gates connected through electronic paths that behave similarly to fuses.

In the original state of the device, all the fuses are intact. Programming the device involves blowing those fuses along the paths that must be removed in order to obtain the particular configuration of the desired logic function.

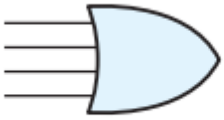
A typical PLD may have hundreds to millions of gates interconnected through hundreds to thousands of internal paths.

In order to show the internal logic diagram of such a device in a concise form, it is necessary to employ a special gate symbology applicable to array logic.

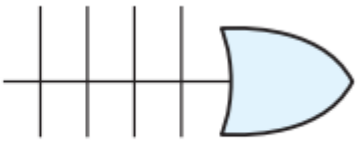
The logic symbols for a multiple-input OR gate into the gate, we draw a single line entering the gate.

The input lines are drawn perpendicular to this single line and are connected to the gate through internal fuses.

In a similar way, we can draw the array logic for an AND gate.



(a) Conventional symbol



(b) Array logic symbol

UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 2
RANDOM ACCESS MEMORY

A memory unit is a collection of storage cells, together with associated circuits needed to transfer information into and out of a device.

The time it takes to transfer information to or from any desired random location is always the same—hence the name *random-access memory,* abbreviated RAM.

A memory unit stores binary information in groups of bits called *words* . A word in memory is an entity of bits that move in and out of storage as a unit.

A memory word is a group of 1's and 0's and may represent a number, an instruction, one or more alphanumeric characters, or any other binary-coded information.

A group of 8 bits is called a *byte* . Most computer memories use words that are multiples of 8 bits in length. Thus, a 16-bit word contains two bytes, and a 32-bit word is made up of four bytes.The capacity of a memory unit is usually stated as the total number of bytes that the unit can store.

- A memory unit stores binary information in groups of bits called words.
  1 byte = 8 bits
  1 Word = 2 bytes

The communication between a memory and its environment is achieved through datainput and output lines, address selection lines, and control lines that specify the direction of transfer.

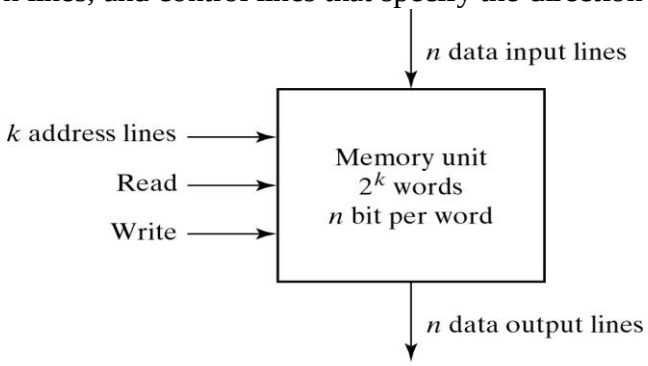


Fig. 7-2 Block Diagram of a Memory Unit

Content of a memory:

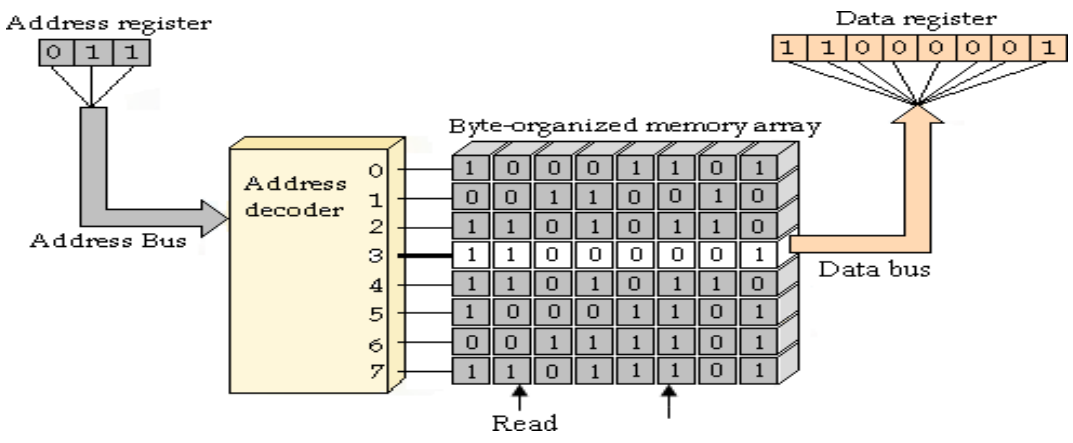
- Each word in memory is assigned an identification number, called an address, startingfrom 0 up to 2^k-1, where k is the number of address lines.
- The number of words in a memory with one of the letters K=2^{10}, M=2^{20}, or G=2^{30}.64K = 2^{16} 2M = 2^{21} 4G = 2^{32}

Read operation:

Transferring a stored word out of memory:

1.Apply the binary address of the desired word to the address lines.

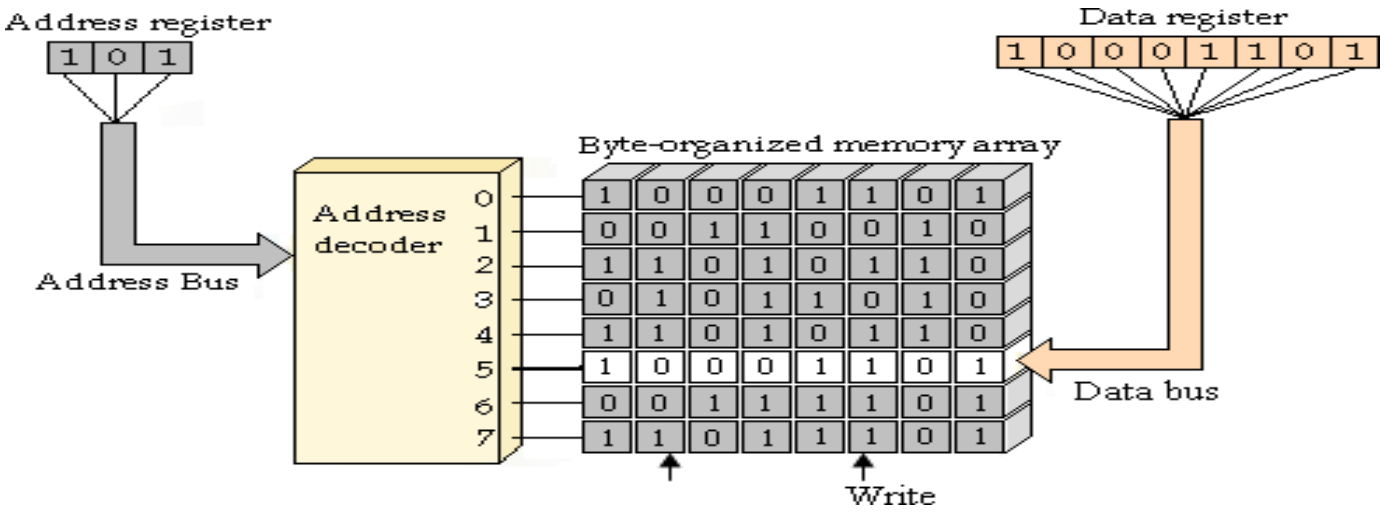
2.Activate the read input.



Write operations :

- Transferring a new word to be stored into memory:

1. Apply the binary address of the desired word to the address lines.
2. Apply the data bits that must be stored in memory to the data input lines.
3. Activate the write input.



Commercial memory sometimes provide the two control inputs for reading and writing in a somewhat different configuration in table

Control Inputs to Memory Chip

Memory Enable	Read/Write	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

Types of RAM:

(i) Dynamic Ram
(ii) Static RAM

Static RAM (SRAM):

All static RAMs are characterized by flip-flop memory cells.
As long as dc power is applied to a static memory cell, it can retain a 1 or 0 state indefinitely.
If power is removed, the stored data bit is lost.
The cell is selected by an active level on the Select line and a data bit (1 or 0) is written into the cell by placing it on the Data in line.
A data bit is read by taking it off the Data out line.

Dynamic RAM (DRAM):

Capacitors as storage elements and cannot retain data very long without the capacitors being recharged by a process called refreshing.
Both SRAMs and DRAMs will lose stored data when dc power is removed and, therefore, are classified as volatile memories.
Data can be read much faster from SRAMs than from DRAMs.
However, DRAMs can store much more data than SRAMs for a given physical size and cost because the DRAM cell is much simpler, and more cells can be crammed into a given chip area than in the SRAM.

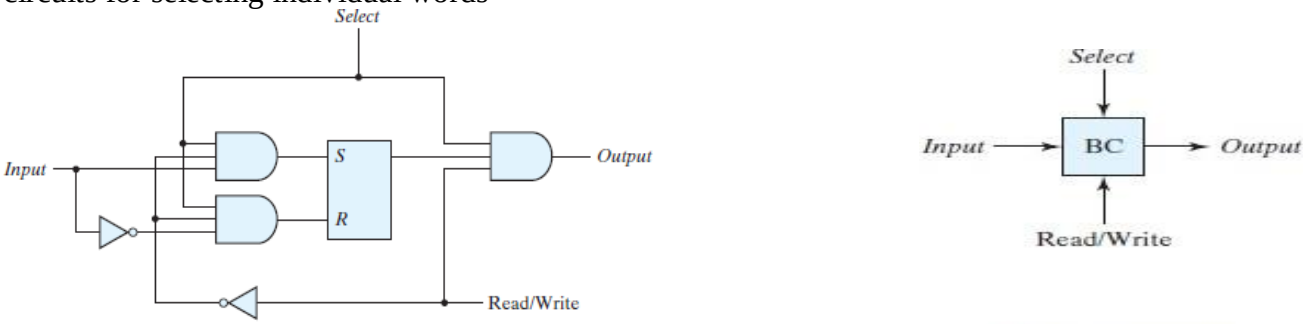
UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 3
MEMORY DECODING

Memory Decoding:

Decoding circuits are needed to select the memory word specified by the input address.

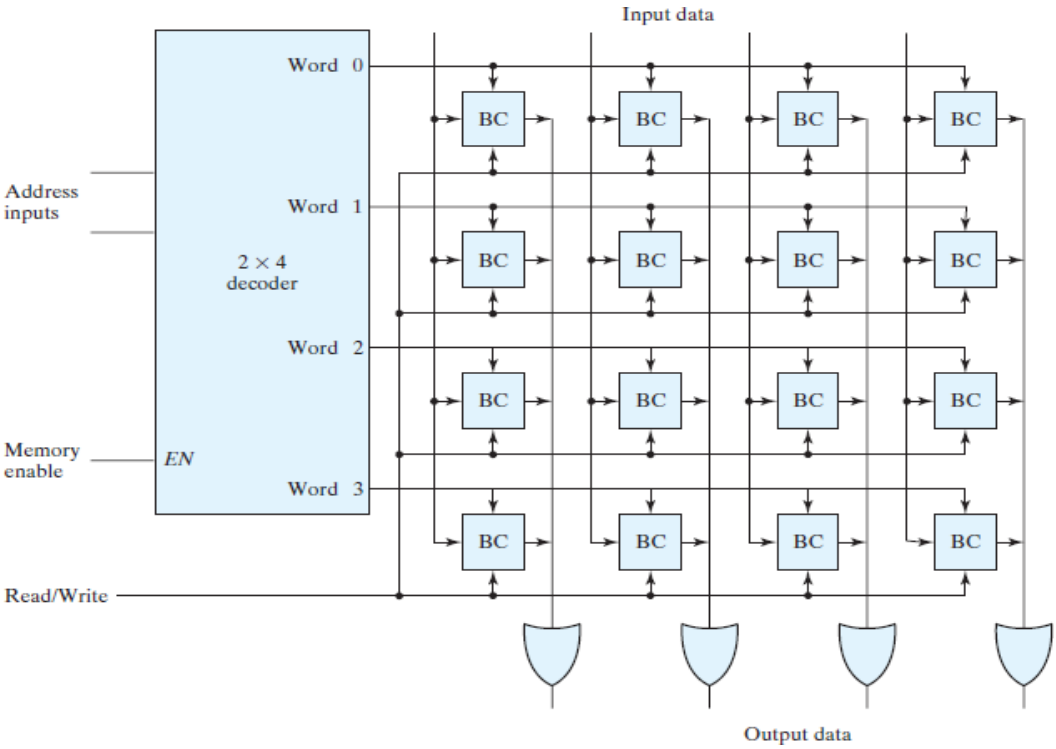
Internal Construction:

A RAM of m words and n bits per word consists of m * n binary storage cells and associated decoding circuits for selecting individual words



A 1 in the read/write input provides the read operation by forming a path from the latch to the output terminal.
A 0 in the read/write input provides the write operation by forming a path from the input terminal to the latch.

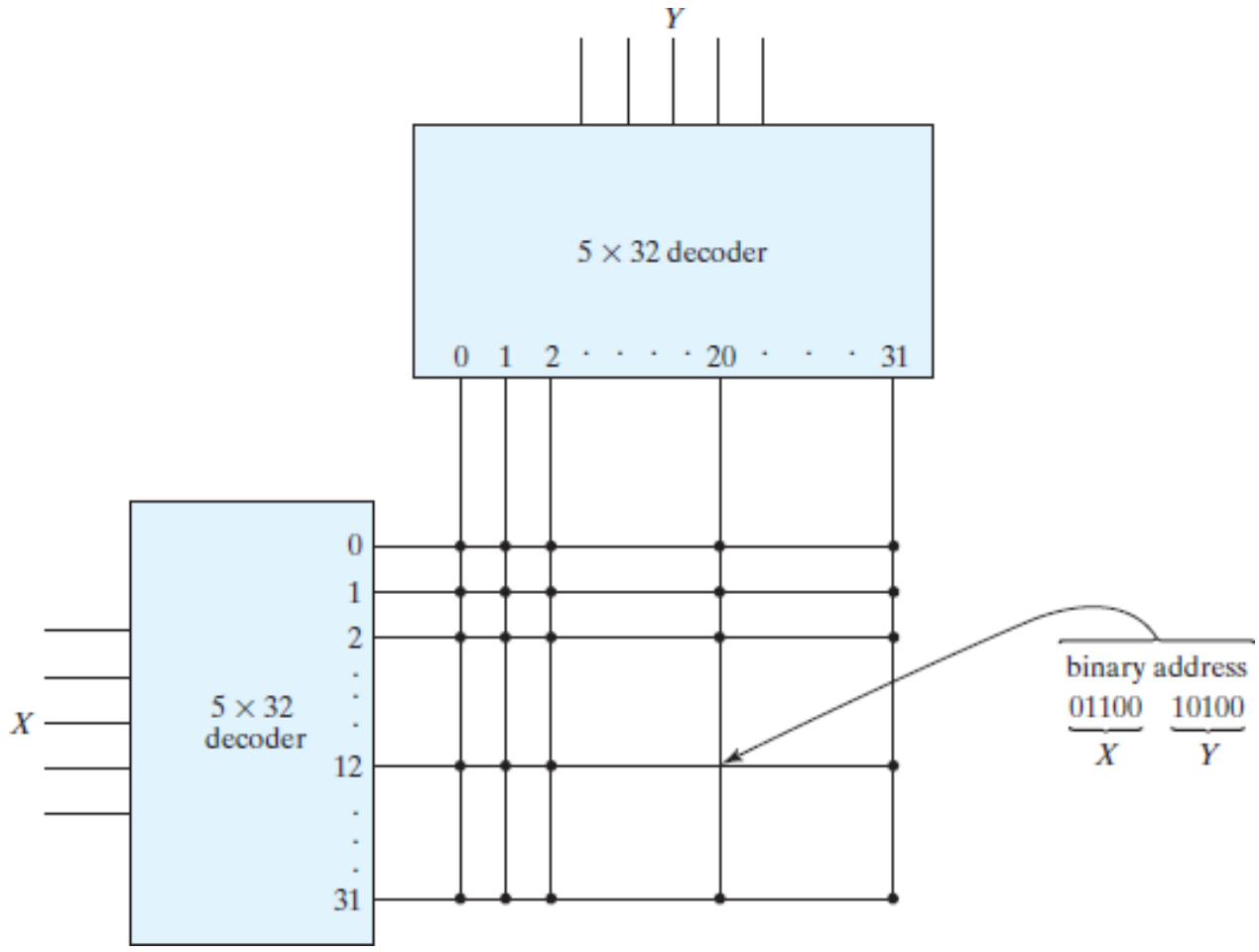
The logical construction of a small RAM is shown in Fig.
This RAM consists of four words of four bits each and has a total of 16 binary cells. The small blocks labeled BC represent the binary cell with its three inputs and one output
A memory with four words needs two address lines. The two address inputs go through a 2 * 4 decoder to select one of the four words. The decoder is enabled with the memory-enable input. When the memory enable is 0, all outputs of the decoder are 0 and none of the memory words are selected.
With the memory select at 1, one of the four words is selected,etc at 1, one of the four words is selected, dictated by the value in the two address lines. Once a word has been selected, the read/write input determines the operation. During the read operation, the four bits of the selected word go through OR gates to the output terminals.



A memory with 2k words of n bits per word requires k address lines that go into a k * 2k decoder.
Each one of the decoder outputs selects one word of n bits for reading or writing.

Coincident Decoding:

A decoder with k inputs and 2^k outputs requires 2^k AND gates with k inputs per gate.
The total number of gates and the number of inputs per gate can be reduced by employing two decoders in a two-dimensional selection scheme.
The basic idea in two-dimensional decoding is to arrange the memory cells in an array that is close as possible to square.
In this configuration, two k /2-input decoders are used instead of one k -input decoder.
One decoder performs the row selection and the other the column selection in a two-dimensional matrix configuration.

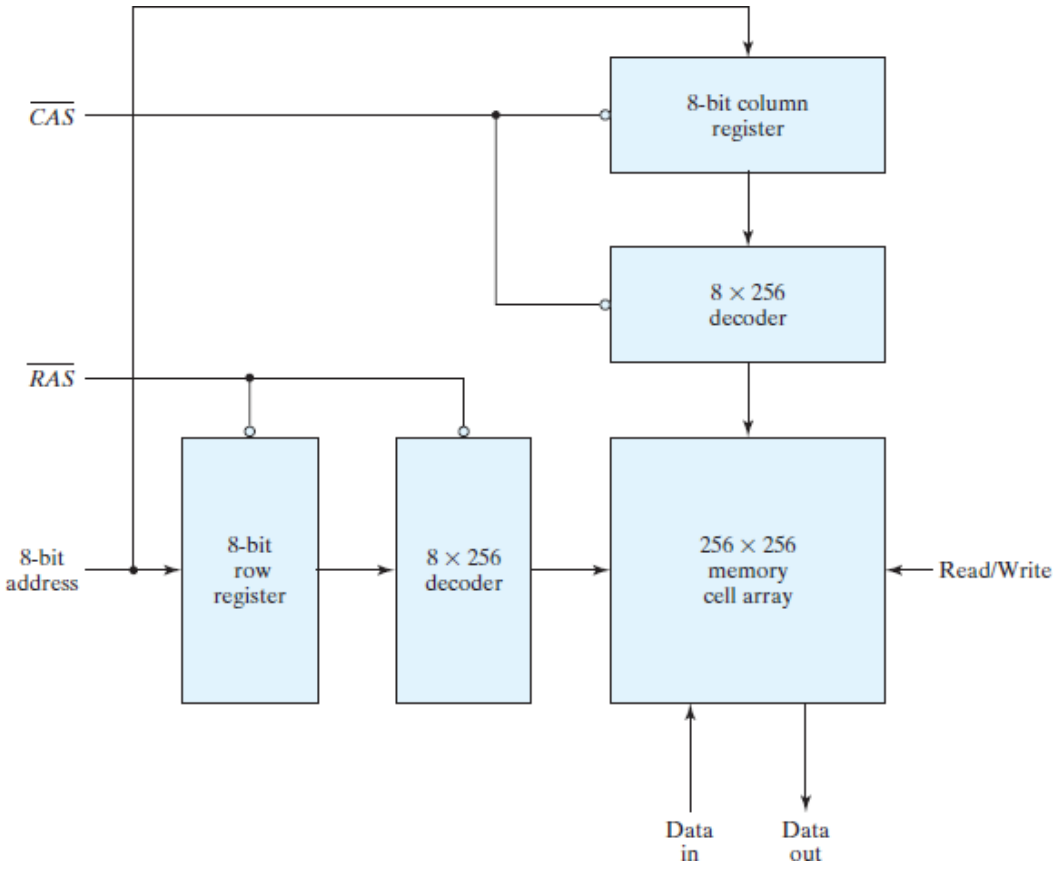


Instead of using a single 10 * 1,024 decoder, we use two 5 * 32 decoders.With the single decoder, we would need 1,024 AND gates with 10 inputs in each. In the two-decoder case, we need 64 AND gates with 5 inputs in each. The five most significant bits of the address go to input *X* and the five least significant bits go to input *Y*. Each word within the memory array is selected by the coincidence of one *X* line and one *Y* line. Thus, each word in memory is selected by the coincidence between 1 of 32 rows and 1 of 32 columns, for a total of 1,024 words. Note that each intersection represents a word that may have any number of bits.

Address Multiplexing:

To build memories with higher density, it is necessary to reduce the number of transistors in a cell. The DRAM cell contains a single MOS transistor and a capacitor. Because of their simple cell structure, DRAMs typically have four times the density of SRAMs. The cost per bit of DRAM storage is three to four times less than that of SRAM storage. A further cost savings is realized because of the lower power requirement of DRAM cells. These advantages make DRAM the preferred technology for large memories in personal digital computers. DRAM chips are available in capacities from 64K to 256M bits. To reduce the number of pins in the IC package, designers utilize address multiplexing whereby one set of address input pins accommodates

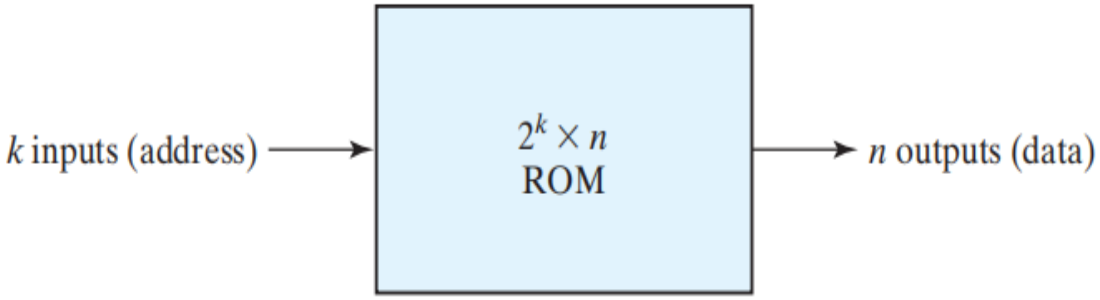
the address components. In a two-dimensional array, the address is applied in two parts at different times, with the row address first and the column address second. Since the same set of pins is used for both parts of the address, the size of the package is decreased significantly.We will use a 64K-word memory to illustrate the address-multiplexing idea. The memory consists of a two-dimensional array of cells arranged into 256 rows by 256 columns, for a total of $2^8 * 2^8 = 2^{16} = 64K$ words. There is a single data input line, a single data output line,and a read/write control, as well as an eight-bit address input and two address *strobes,*the latter included for enabling the row and column address into their respective registers. The row address strobe (RAS) enables the eight-bit row register, and the column address strobe (CAS) enables the eight-bit column register.The 16-bit address is applied to the DRAM in two steps using RAS and CAS. Initially, both strobes are in the 1 state. The 8-bit row address is applied to the address inputs and RAS is changed to 0. This loads the row address into the row address register. RAS also enables the row decoder so that it can decode the row address and select one row of the array. After a time equivalent to the settling time of the row selection, RAS goes back to the 1 level. The 8-bit column address is then applied to the address inputs, and CAS is driven to the 0 state. This transfers the column address into the column register and enables the column decoder. Now the two parts of the address are in their respective registers, the decoders have decoded them to select the one cell corresponding to the row and column address, and a read or write operation can be performed on that cell CAS must go back to the 1 level before initiating another memory operation.



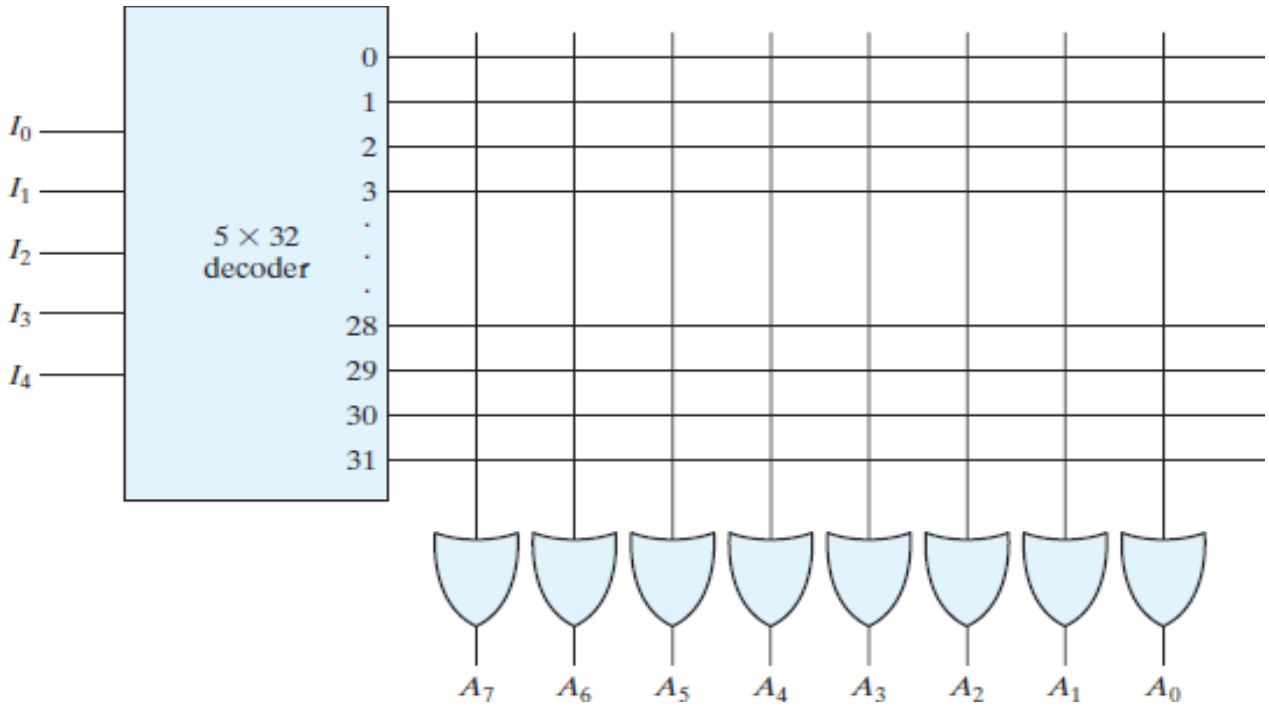
UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 4
READ ONLY MEMORY

Read Only Memory:

A ROM is essentially a memory device in which permanent binary information is stored. The binary information must be specified by the designer and is then embedded in the unit to form the required interconnection pattern. Once the pattern is established it stays within the unit even when power is turned off and on again.



It consists of k inputs and n outputs. The inputs provide the address for the memory and the outputs give the data bits of the stored word which is selected by the address. The number of words in a ROM is determined from the fact that k address input lines are needed to specify 2 k words.ROM does not have data inputs because it does not have a write operation.
Consider for example a 32 x 8 ROM. The unit consists of 32 words of 8 bits each. There are five input lines that form the binary numbers from 0 through 31 for the address. The Figure 2 shows the internal logic construction of the ROM. The five inputs are decoded into 32 distinct outputs by means of a 5 x 32 decoder. Each output of the decoder represents a memory address. The 32 outputs of the decoder are connected to each of the eight OR gates.

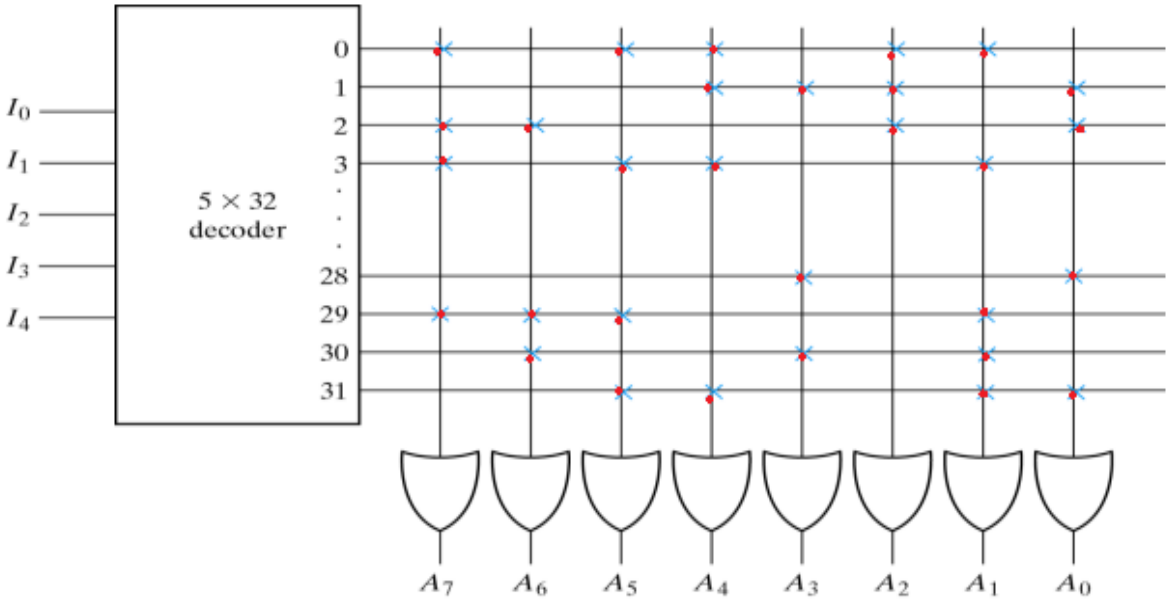


- The diagram shows the array logic convention used in complex circuits . Each OR gate must be considered as having 32 inputs.

- Each output of the decoder is connected to one of the inputs of each OR gate.
- Since each OR gate has 32 input connections and there are 8 OR gates, the ROM contains 32 x 8 = 256 internal connections.
- The 256 intersections are programmable
- The truth table shows the five inputs under which are listed all 32 addresses.
- Each address stores a word of 8 bits, which is listed in the outputs columns.

ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1



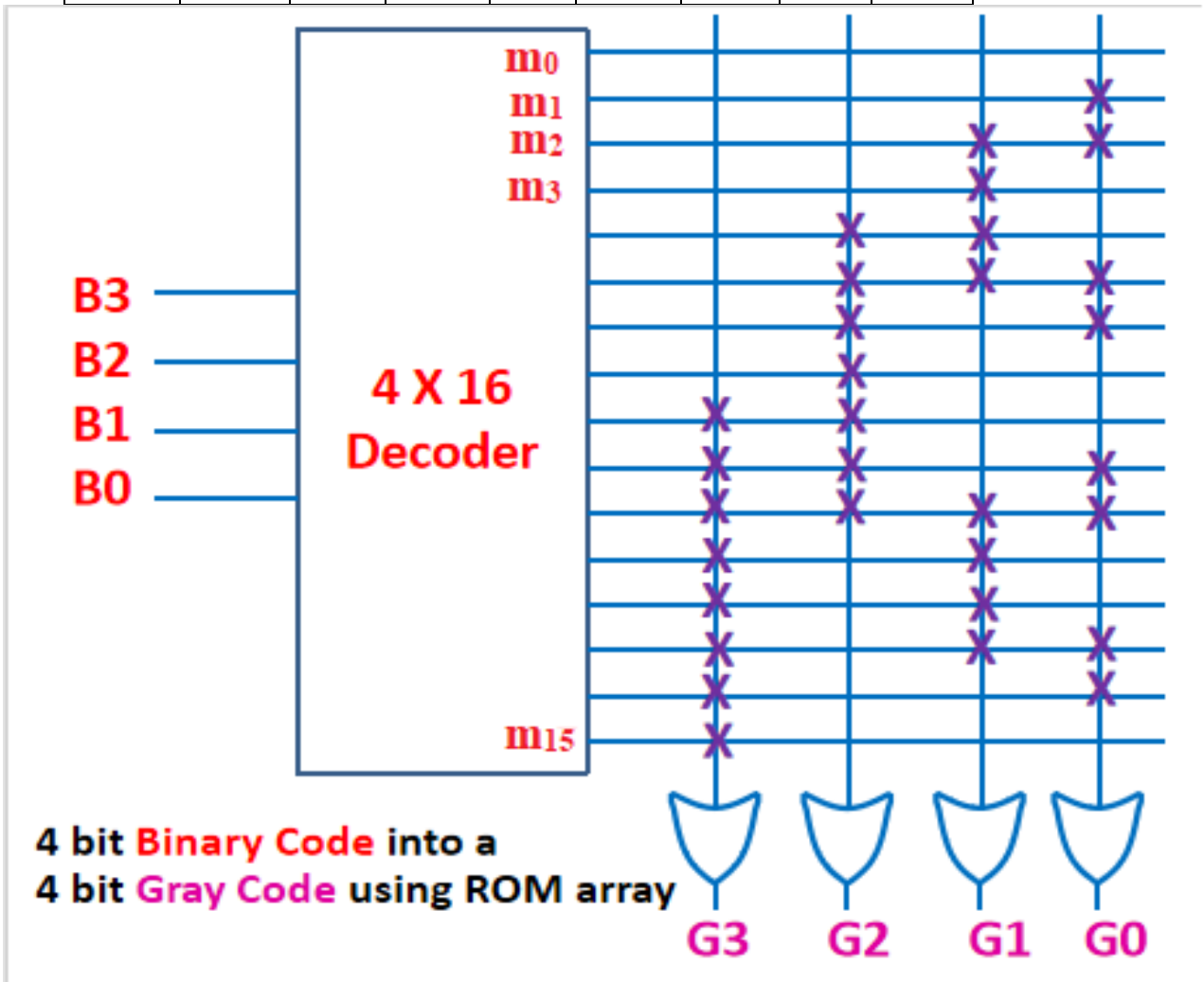
Combinational Circuit Implementation:

- A decoder generates the 2k minterms of the k input variables.
- By inserting OR gates to sum the minterms of Boolean functions, we were able to generate any desired combinational circuit.
- The ROM is essentially a device that includes both the decoder and the OR gates within a single device to form a minterm generator.
- By choosing connections for those minterms which are included in the function, the ROM outputs can be programmed to represent the Boolean functions of the output variables in a combinational circuit.
- The internal operation of a ROM can be interpreted in two ways.
- The first interpretation is that of a memory unit that contains a fixed pattern of stored words.
- The second interpretation is that of a unit which implements a combinational circuit.

EXAMPLE-1

Design a switching circuit that converts a 4bit binary code into a 4 bit Gray code using ROM array

Decimal Equivalent	Binary Input				Gray Output			
	B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

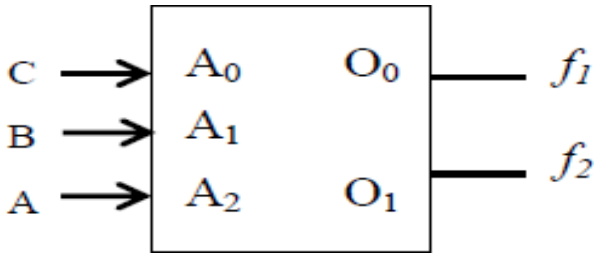


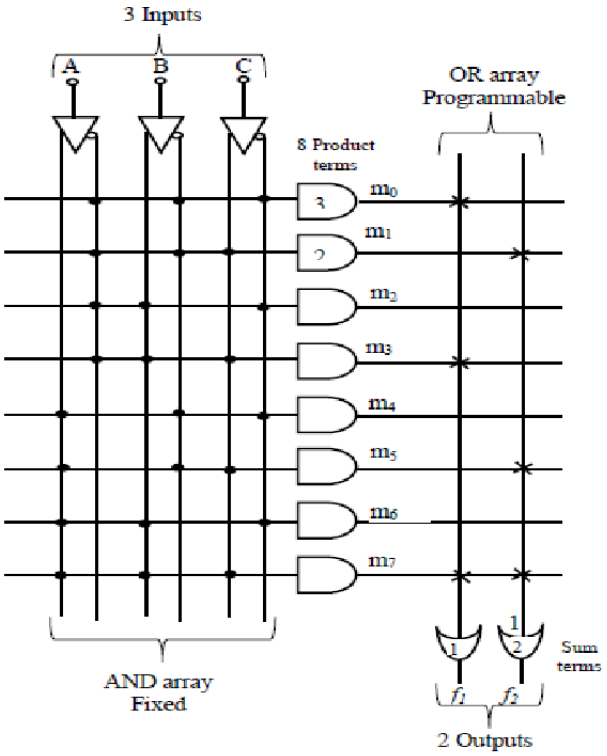
EXAMPLE-2

Implement the following function with ROM.

F1 (A,B,C)=∑m(0,3,7) and f2 (A,B,C)=∑m(1,5,7)

A_2	A_1	A_0	f_1	f_2
0	0	0	1	0
0	0	1	0	1
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1





Types of ROMs:

ROM : Read only memory: Its non volatile memory, ie, the information stored in it, is not lost even if the power supply goes off. It's used for the permanent storage of information. It also posses random access property. Information can not be written into a ROM by the users/programmers. In other words the contents of ROMs are decided by the manufactures.

The following types of ROMs an listed below :

(i) **PROM :** It's programmable ROM. Its contents are decided by the user. The user can store permanent programs, data etc in a PROM. The data is fed into it using a PROM programs.

(ii)EPROM : An EPROM is an erasable PROM. The stored data in EPROM's can be erased by exposing it to UV light for about 20 min. It's not easy to erase it because the EPROM IC has to be removed from the computer and exposed to UV light. The entire data is erased and not selected portions by the user. EPROM's are cheap and reliable.

(iii)EEPROM (Electrically Erasable PROM) : The chip can be erased & reprogrammed on the board easily byte by byte. It can be erased with in a few milliseconds. There is a limit on the number of times the EEPROM's can be reprogrammed, i.e.; usually around 10,000 times.

UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 5
PROGRAMMABLE LOGIG DEVICES

Programmable Logic Devices - PLD's:

- PLD is an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of product implementation.
- The PLD's can be reprogrammed in few seconds and hence gives more flexibility to experiment with designs.
- Reprogramming feature of PLDs also makes it possible to accept changes/modifications in the previously design circuits.

The advantages of using programmable logic devices are:

1. Reduced space requirements.
2. Reduced power requirements.
3. Design security.
4. Compact circuitry.
5. Short design cycle.
6. Low development cost.
7. Higher switching speed.
8. Low production cost for large-quantity production.

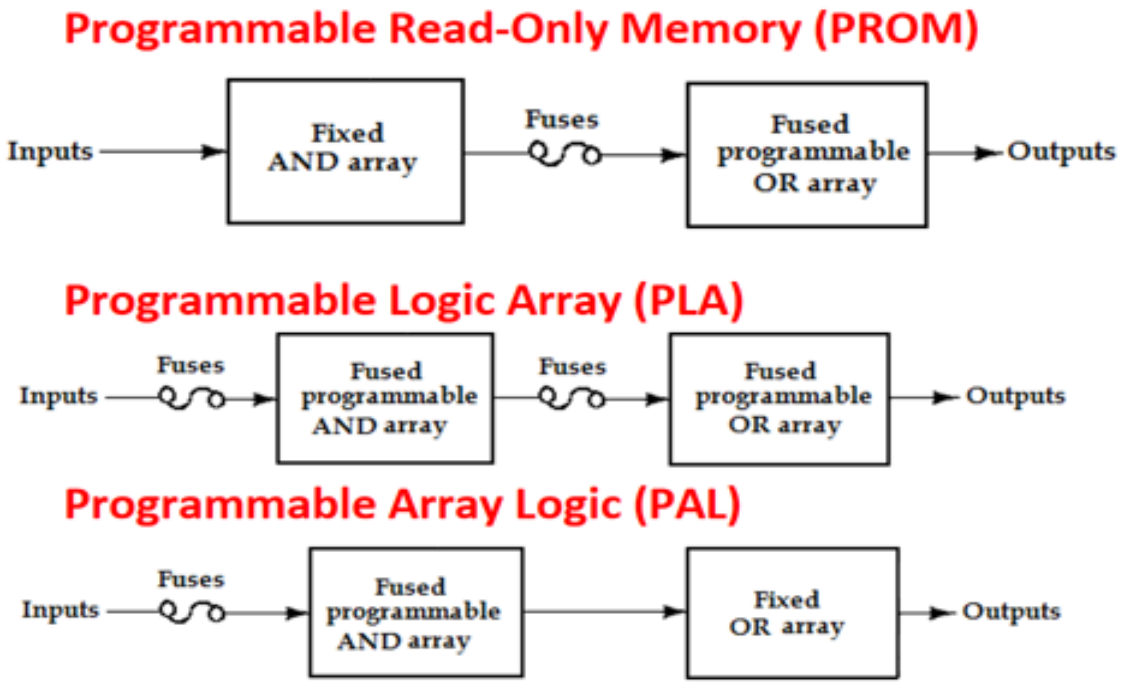
Classification Of PLD's:

- A combinational PLD is an integrated circuit with programmable gates divided into an AND array and an OR array to provide an AND-OR sum of product implementation.
- There are three major types of combinational PLDs and they differ in the placement of the programmable connections in the AND-OR array.

PROM : Programmable Read Only memorie(Fixed AND array constructed as a decoder and programmable OR array)

PAL : Programmable Array Logic(programmable AND array and fixed OR array)

PLA: Programmable Logic Array(Both the AND and OR arrays can be programmed)



Programmable Arrays

All PLDs consists of programmable arrays.

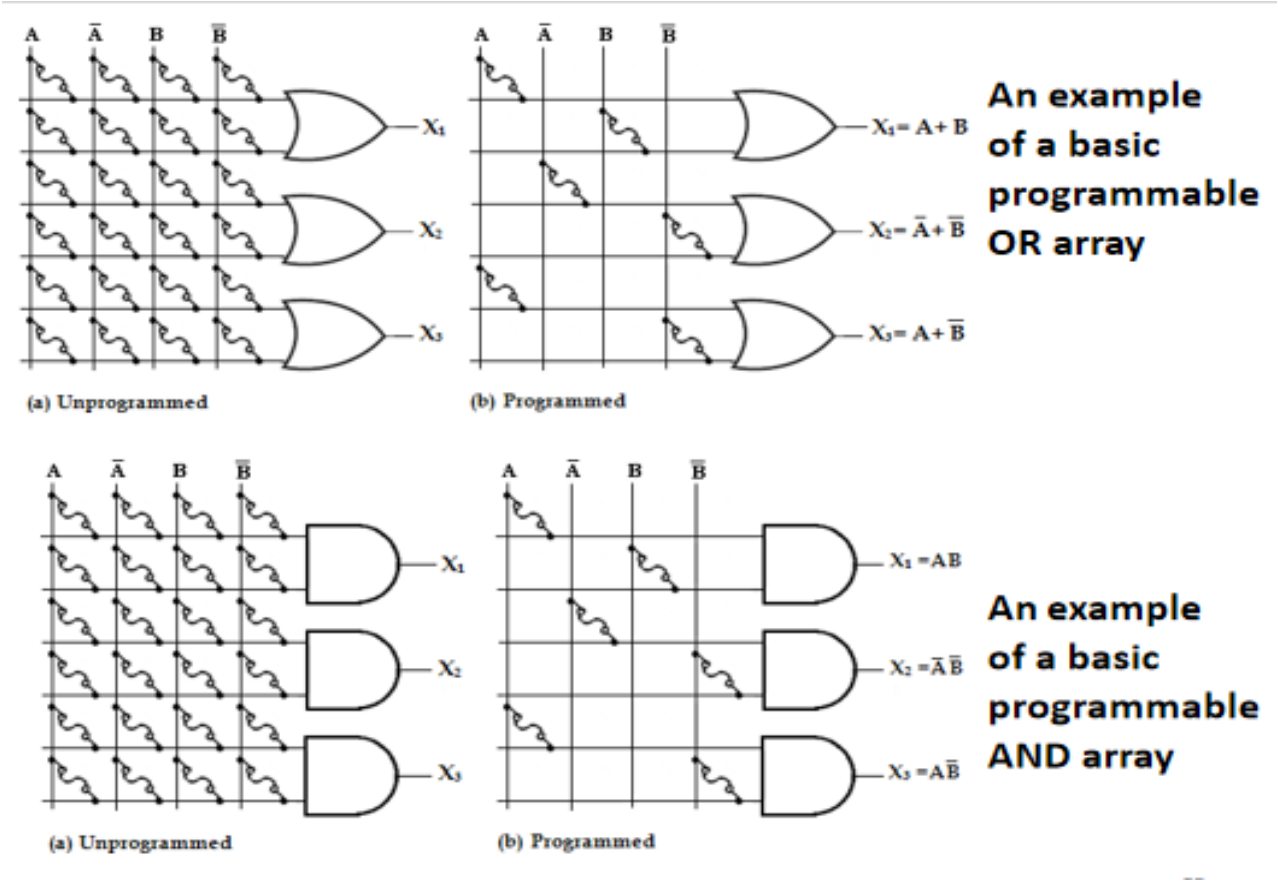
A programmable array is essentially a grid of conductors that form rows and columns with a fusible link at each cross point.Arrays can be either fixed or programmable.

The OR Array

It consists of an array of OR gates connected to a programmable matrix with fusible links at each cross point of a row and column.

The AND Array

This type of array consists of AND gates connected to a programmable matrix with fusible links at each cross points.



Programmable Logic Array:

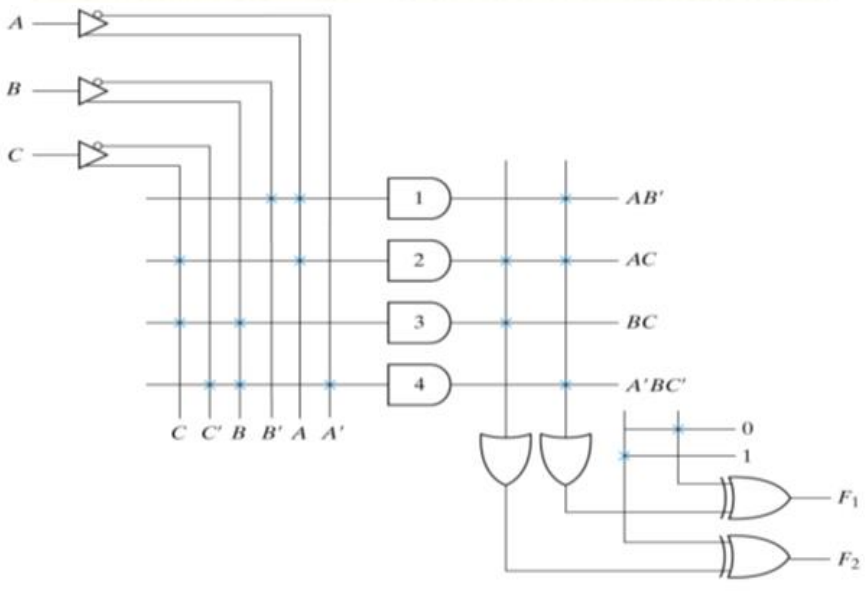
The PLA is similar to PROM in concept except that PLA does not provide full decoding of the variable and does not generate all the minterms . The decoder is replaced by an array of AND gates that can be programmed to generate any product term of the input variables. The product terms are then connected to OR gates to provide the sum of products for the required Boolean functions.

- The decoder in PROM is replaced by an array of AND gates that can be programmed to generate any product term of the input variables.
- The product terms are then connected to OR gates to provide the sum of products forthe required Boolean functions.
- The output is inverted when the XOR input is connected to 1 (since x⊕1 = x'). Theoutput doesn't change and connect to 0 (since x⊕0 = x).
    - F1 = AB'+AC+A'BC'
    - F2 = (AC+BC)'

- The diagram uses the array logic graphic symbols for complex circuits. Each input goes through a buffer and an inverter shown in the diagram with a composite graphic symbol which has both the true and complement outputs. Each input and its complement are connected to the inputs of each AND gate as indicated by the intersections between the vertical and horizontal lines. The outputs of the AND gates are connected to the inputs of each OR gate.
- The output of the OR gates goes to an XOR gate where the other inputcan be programmed to receive a signal equal to either logic 1 or 0.
- The output is inverted when the XOR input is connected to 1. The outputdoes not change when the XOR input is connected to 0 .
- The product terms generated in each AND gate are listed along the output of the gate in the diagram. The product term is determined from the inputs whose crosspoints are connected and marked with a X . The output of an OR gate gives the logic sum of the selected product terms. The output may be complemented or left in its true form depending on the connection for one of the XOR gate inputs.
- The programming table that specifies the PLA of Fig.4 is listed in Table 2. The PLA
- programming table consists of three sections .The first section lists the product terms numerically. The second section specifies the required paths between inputs and AND gates. The third section specifies the paths between AND and OR gates.

PLA Programming Table

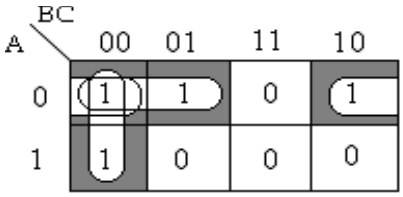
		Inputs			Outputs (T)	Outputs (C)
	Product Term	A	B	C	F_1	F_2
AB'	1	1	0	–	1	–
AC	2	1	–	1	1	1
BC	3	–	1	1	–	1
A'BC'	4	0	1	0	1	–



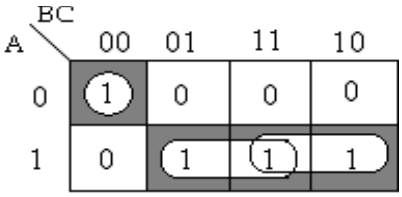
PLA with 3 Inputs, 4 Product Terms, and 2 Outputs

Implement the combinational circuit with a PLA having 3 inputs, 4 product terms and 2 outputs for the functions.F1 (A, B, C) = Σm (0, 1, 2, 4),F2 (A, B, C) = Σm (0, 5, 6, 7)

A	B	C	F1	F2
0	0	0	1	1
0	0	1	1	0
0	1	0	1	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	0	1

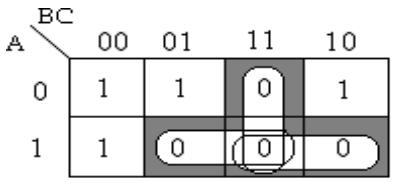


$F_1 = A'B' + A'C' + B'C'$

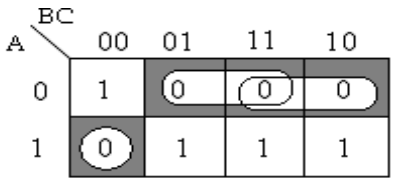


$F_2 = AC + AB + A'B'C'$

- With this simplification, total number of product term is 6. But we require only 4 product terms. Therefore find out F1‘ and F2‘.



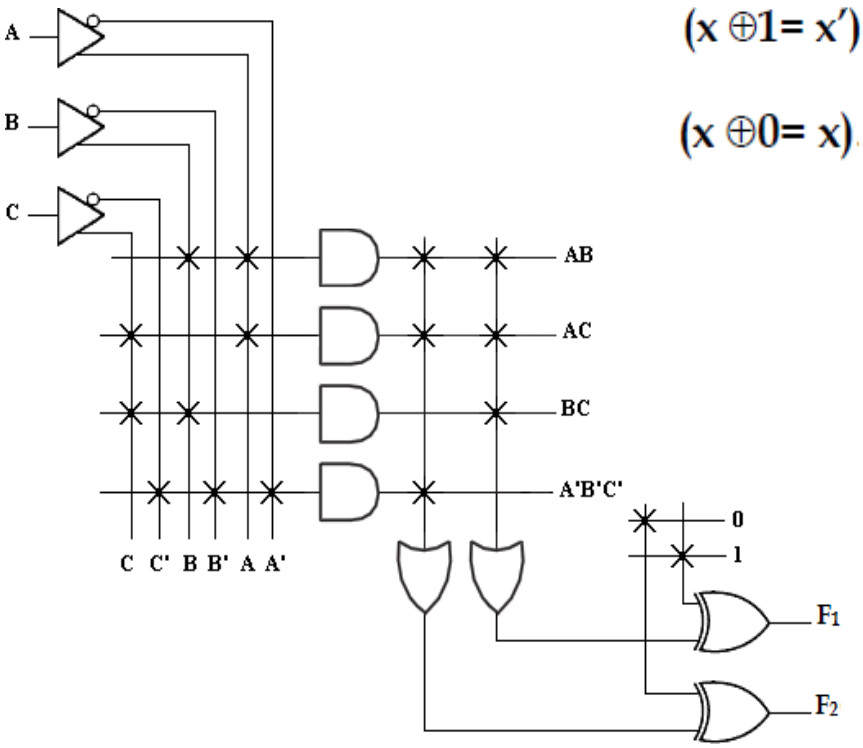
$F_1' = AC + BC + AB$



$F_2' = A'C + A'B + A'B'C'$

Now select,F1‘and F2,the product terms are AC, AB, BC and A‘B‘C‘.

	Product term	Inputs			Outputs	
		A	B	C	F_1(C)	F_2(T)
AB	1	1	1	-	1	1
AC	2	1	-	1	1	1
BC	3	-	1	1	1	-
A'B'C'	4	0	0	0	-	1



$(x \oplus 1 = x')$

$(x \oplus 0 = x)$

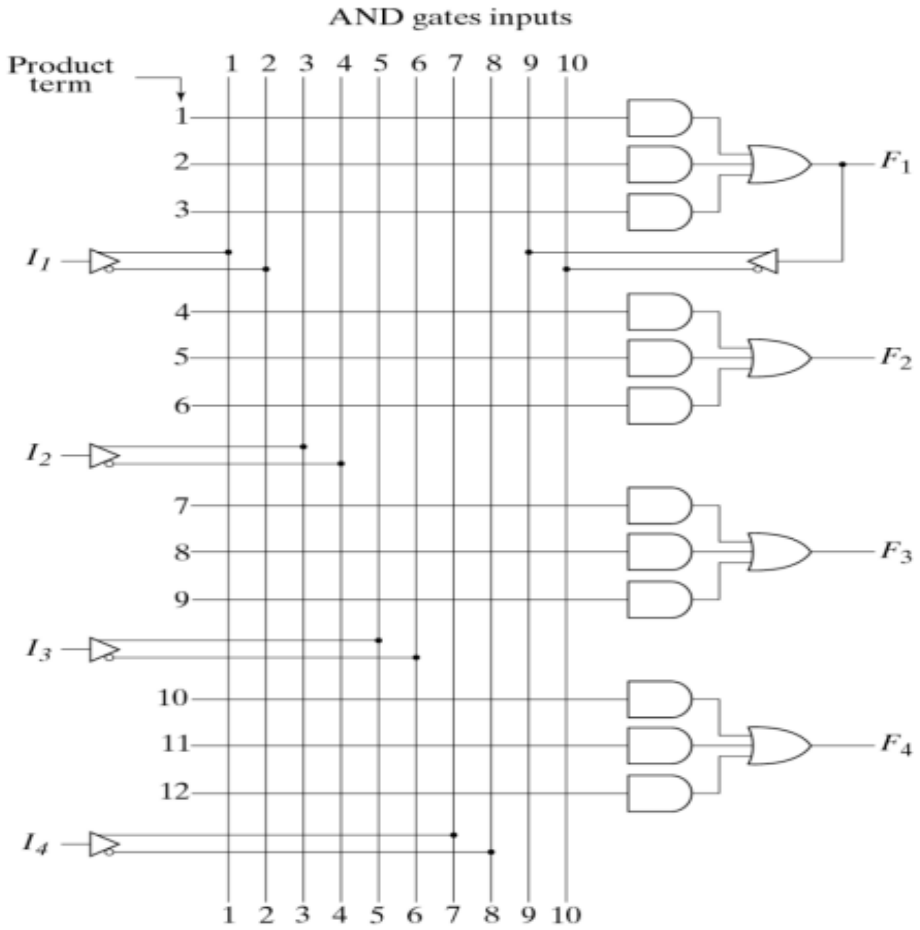
UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 6
PROGRAMMABLE ARRAY LOGIG

PROGRAMMABLE ARRAY LOGIC (PAL):

The programmable array logic (PAL) is a programmable logic device with a fixed OR array and a programmable AND array. Because only the AND gates are programmable the PAL is easier to program, but is not as flexible as the PLA.

It has four inputs and four outputs. Each input has a buffer inverter gate and each output is generated by a fixed OR gate. There are four sections in the unit, each being composed of a three wide AND-OR array.

Each AND gate has 10 programmable input connections. This is shown in the diagram by 10 vertical lines intersecting each horizontal line. The horizontal line symbolizes the multiple input configuration of the AND gate. One of the outputs is connected to a buffer inverter gate and then fed back into two inputs of the AND gates.



PAL with Four Inputs, Four Outputs, and Three-Wide AND-OR Structure

- When designing with a PAL, the Boolean functions must be simplified to fit into eachsection.
- Unlike the PLA, a product term cannot be shared among two or more OR gates. Therefore, each function can be simplified by itself without regard to common productterms.
- The output terminals are sometimes driven by three-state buffers or inverters.

PALExample 1:

w(A, B, C, D) = ∑(2, 12, 13) w = ABC’ + A’B’CD’

x(A, B, C, D) = ∑(7, 8, 9, 10, 11, 12, 13, 14, 15) x = A + BCD

y(A, B, C, D) = ∑(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15) w = A’B + CD + B’D’

z(A, B, C, D) = ∑(1, 2, 8, 12, 13) Z = ABC’ + A’B’CD’ + AC’D’ + A’B’C’D

= w + AC’D’ + A’B’C’D

Simplifying the four functions as following Boolean functions:

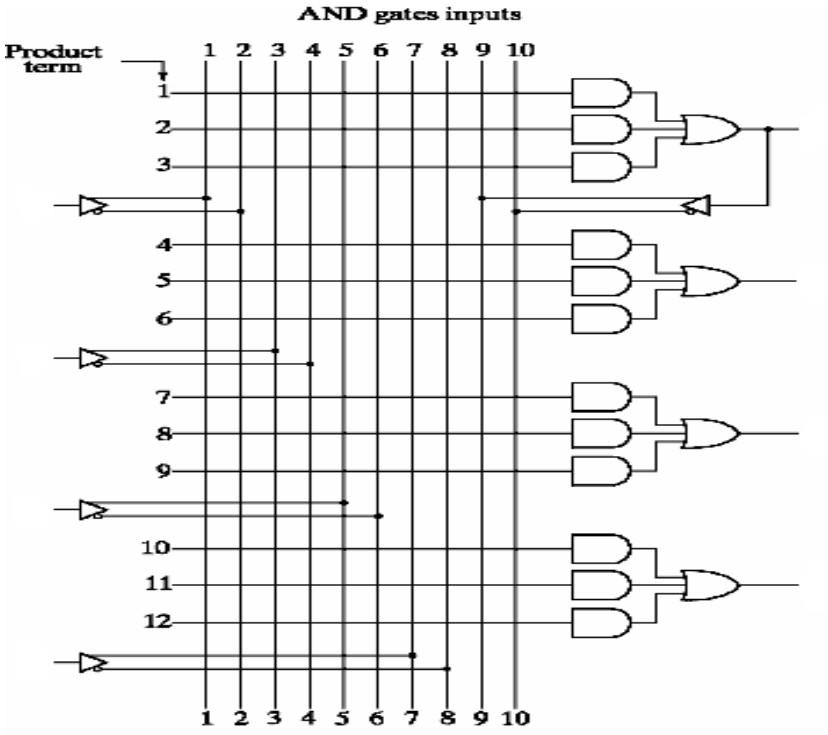
PAL Table:

- z has four product terms, and we can replace by w with two product terms, this will reduce the number of terms for z from four to three.
- The table is divided into four sections with three product terms. The first two sections need only two product terms to implement the Boolean function. The last section for output z needs four product terms. Using the output from w, we can reduce the functionto three terms.

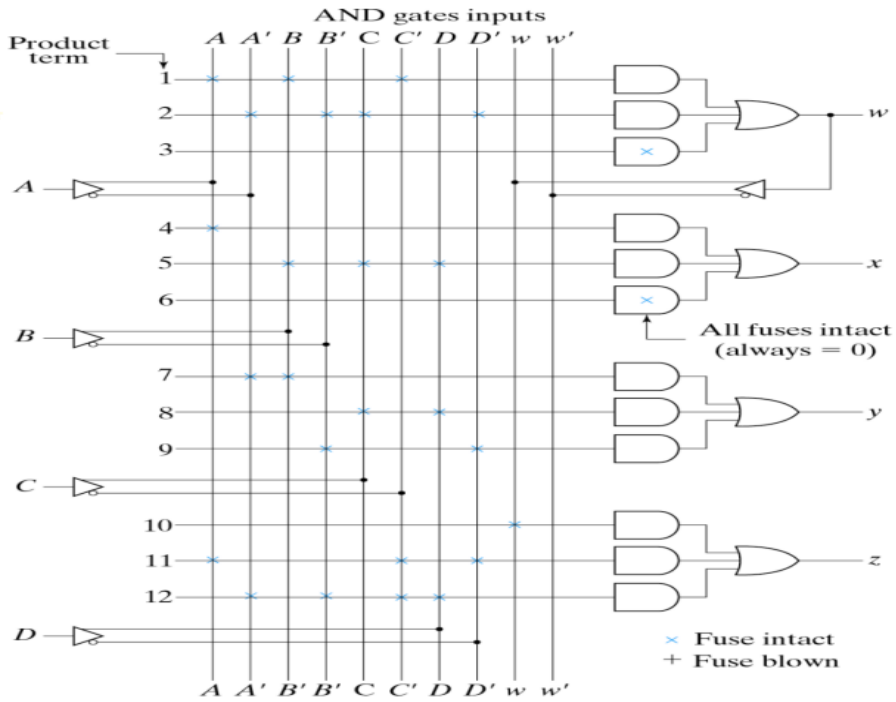
Table 7-6
PAL Programming Table

	AND Inputs					
Product Term	*A*	*B*	*C*	*D*	*W*	**Outputs**
1	1	1	0	–	–	$w = ABC'$
2	0	0	1	0	–	$+ A'B'CD'$
3	–	–	–	–	–	
4	1	–	–	–	–	$x = A$
5	–	1	1	1	–	$+ BCD$
6	–	–	–	–	–	
7	0	1	–	–	–	$y = A'B$
8	–	–	1	1	–	$+ CD$
9	–	0	–	0	–	$+ B'D'$
10	–	–	–	–	1	$z = w$
11	1	–	0	0	–	$+ AC'D'$
12	0	0	0	1	–	$+ A'B'C'D$

PAL implementation :



Fuse map for example :



Fuse Map for PAL as Specified in Table 7-6

For each 1 or 0 in the table, we mark the corresponding intersection in the diagram with the symbol for an intact fuse. For each dash we mark he diagram with blown fuses in both the true and complement inputs. If the AND gate is not used, we leave all its input fuses intact.

Since the corresponding input receives both the true and complement of each input variable we have AA'=O and the output of the AND gate is always O.

UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 7
SEQUENTIAL PROGRAMMABLE LOGIC DEVICES

Sequential Programmable Logic Devices:

Digital systems are designed with flip-flops and gates. Since the combinational PLD consists of only gates, it is necessary to include external flip-flops when they are used in the design.

Sequential programmable devices include both gates and flip-flops.

In this way, the device can be programmed to perform a variety of sequential-circuit functions.

There are three major types of sequential programming devices namely

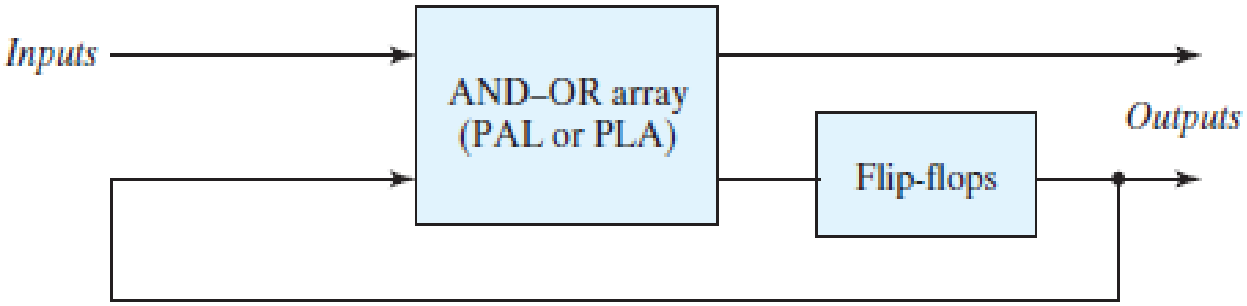
1. Sequential (or simple) programmable logic device (SPLD)
2. Complex programmable logic device (CPLD)
3. Field-programmable gate array (FPGA)

It is a circuit that contains a sum-of-products combinational logic function and an optional flip-flop.

The circuit outputs can be taken from the OR gates or from the outputs of the flip-flops.

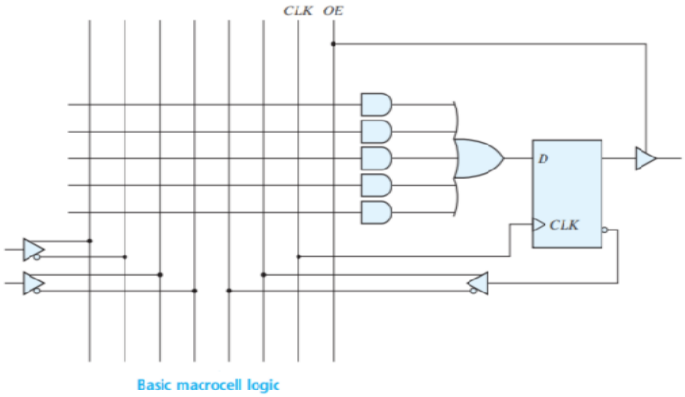
The configuration mostly used in an SPLD is the combinational PAL together with D flip-flops.

Each section of an SPLD is called a macrocell.



Macrocell:

- Macrocell is a circuit that contains a sum-of-products combinational logic function and an optional flip-flop.
- The AND–OR array is the same as in the combinational PAL.
- The output is driven by an edge-triggered D flip-flop connected to a common clock input and changes state on a clock edge.
- The output of the flip-flop is connected to a three-state buffer (or inverter) controlled by an output-enable signal marked in the diagram as OE .
- The output of the flip-flop is fed back into one of the inputs of the programmable AND gates to provide the present-state condition for the sequential circuit.

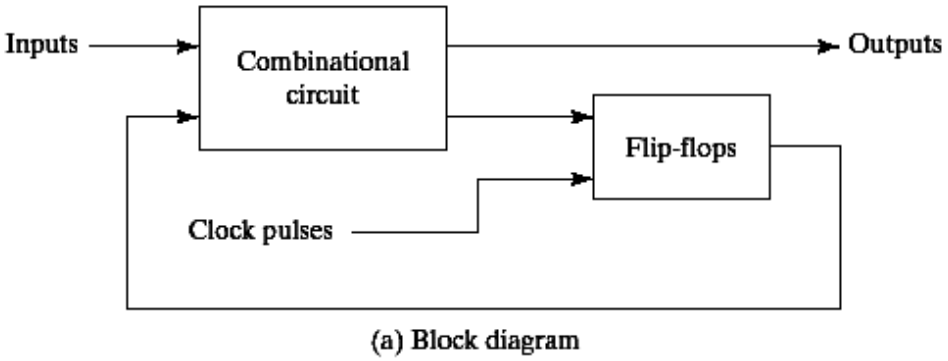


Basic macrocell logic

UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 8
ASYNCHRONOUS SEQUENTIAL CIRCUITS

Synchronous Sequential Circuits:

- The change of internal state occurs in response to the synchronized clock pulses.
- The memory elements are flip-flops.



(a) Block diagram

Advantages of Asynchronous sequential circuits:

1- Accelerate the speed of the machine (no need to wait for the next clock pulse).

2-Used when the input signals change independently of the clock pulses.

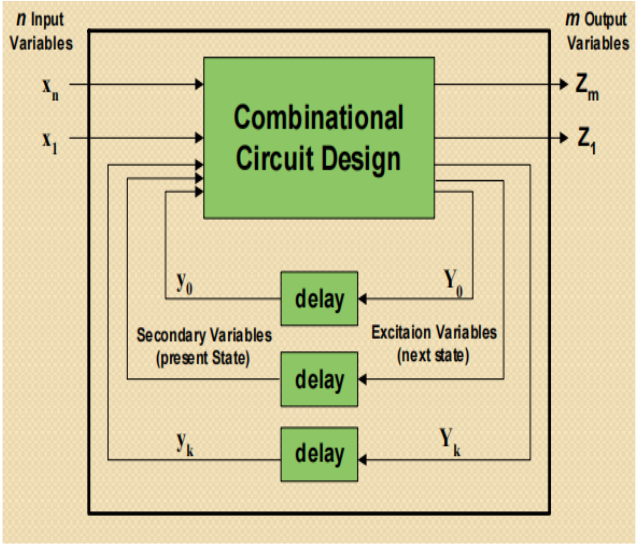
3- Simplify the circuit in the small independent circuits.

4- Used to communicate two circuits each have its own clock.

Asynchronous sequential circuits:

Internal states can change at any instant of time when there is a change in the input variables

- No clock signal is required
- Have better performance but hard to design due to timing problems
- The memory elements are either unclocked FF's or time-delay elements.
- The design of these circuits is more difficult than the design of synchronous circuits due to the timing problem.



- The delay elements provide short-term memory for the sequential circuits.
- Present state variables [y1..yk] are called secondary variables
- Next state variables [Y1..Yk] are called excitation variable.
- When an input variable changes, it takes a certain time to propagate through the combinational circuit to change y, and then Y takes a certain time to propagate through the delay element to become a new state.
- The circuit reaches a steady-state condition when yi = Yi for i=1,2,…, K.

Stable System:

- For a given value of input variables, the system is stable if the circuitreaches a steady state condition.

Fundamental-mode operation:

- This mode assumes that the one input signal changes at a time and only when the circuit is in stable condition.
- The time between two input changes must be longer than the time it takes the circuit to reach a stable state

UNIT -5

MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC

TOPIC – 9

ANALYSIS PROCEDURE

Analysis Procedure

The analysis consists of obtaining a table or a diagram that describes the sequence of internal states and outputs as a function of changes in the input variables.

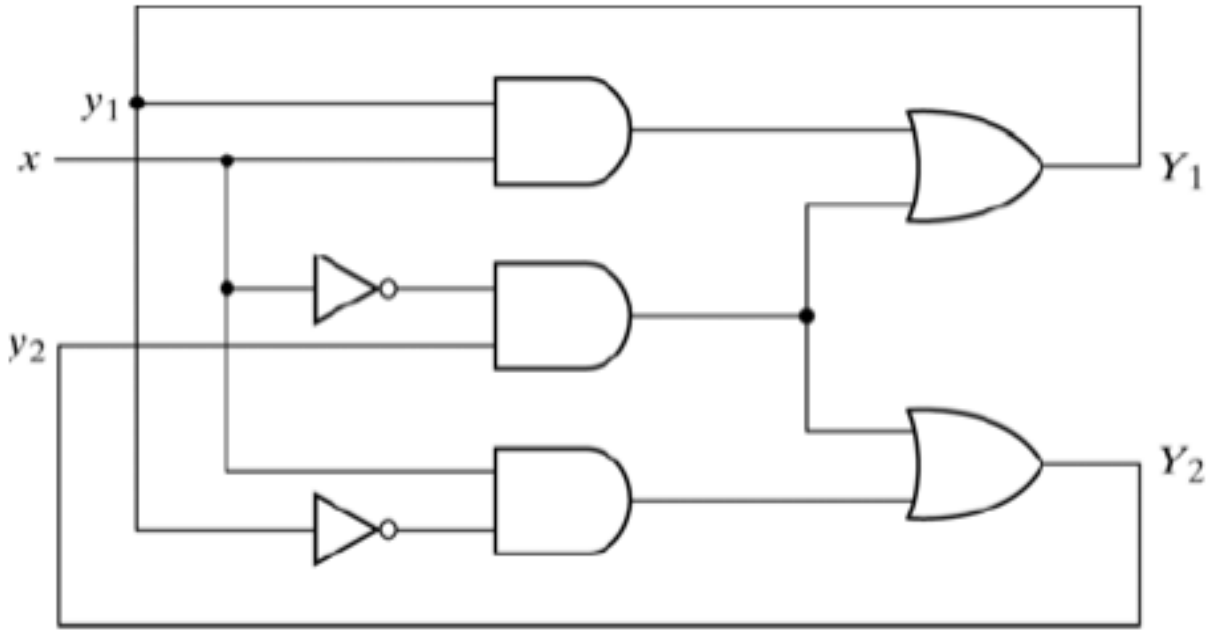
- Transition Table
- Flow Table
- Stability Consideration

Transition Table:

Transition table is useful to analyze an asynchronous circuit from the circuit diagram .

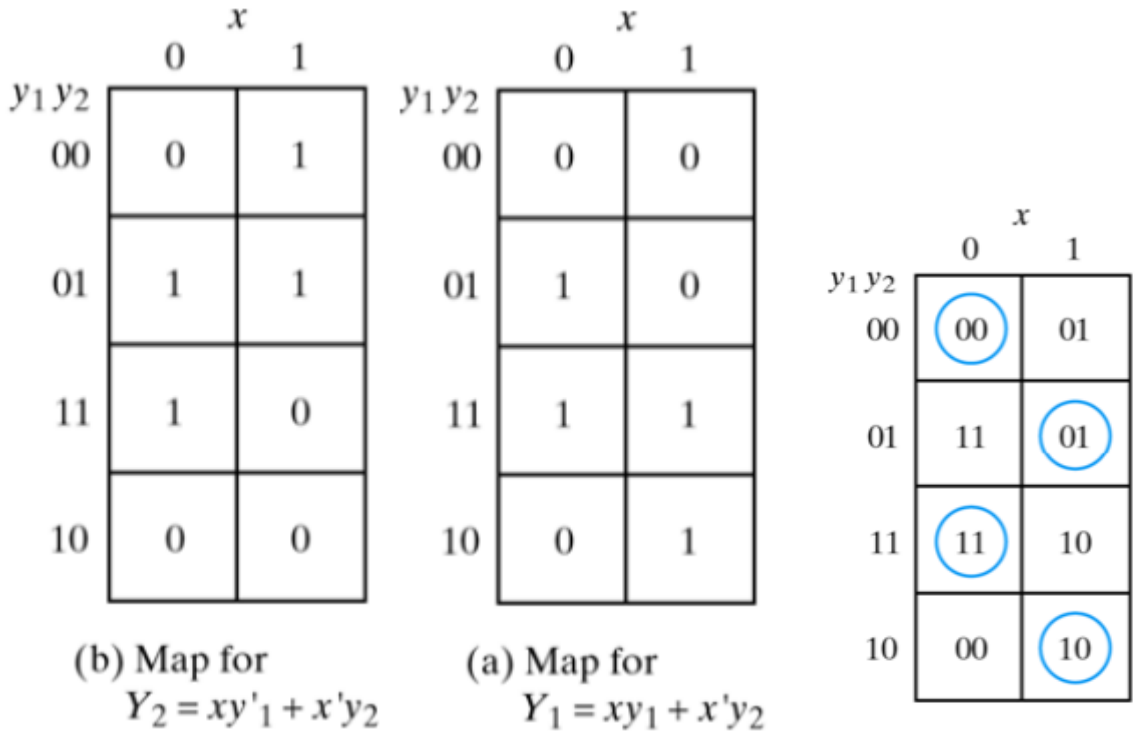
Procedure to obtain transition table:

1. Determine all feedback loops in the circuits
2. Mark the input (yi) and output (Yi) of each feedback loop
3. Derive the Boolean functions of all Y's
4. Plot each Y function in a map and combine all maps into one table
5. Circle those values of Y in each square that are equal to the value of y inthe same row

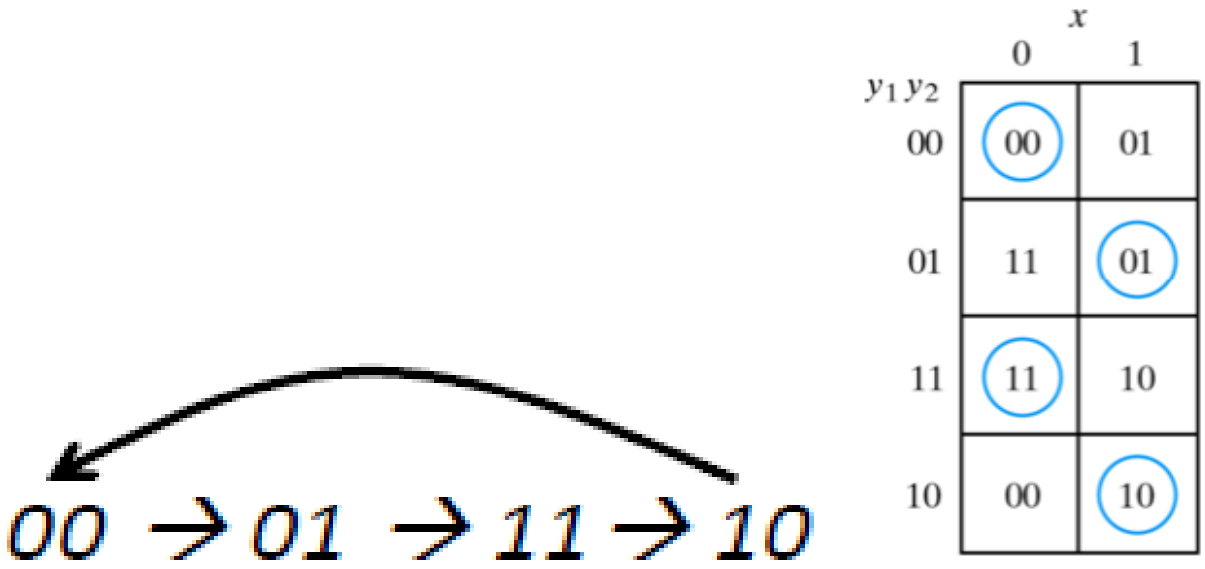


Y1 = xy1 + x'y2

Y2 = xy'1 + x'y2



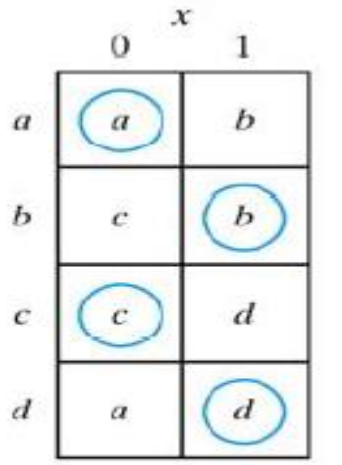
- If y=00 and x= 0 Y ==00(Stable)
- If x changes from 0 to 1 while y=00, the circuit changes Y to 01 which is temporary unstablecondition (Y != y)
- As soon as the signal propagates to make Y = 01, the feedback path causes a change in y to 01.(transition form the first row to the second row)
- If the input repeatedly alternates between 0 and 1, the circuit will repeat the sequence of states



- In an asynchronous sequential circuit, the internal state can change immediately after a change in the input.
- It is sometimes convenient to combine the internal state with input value together and call it the Total State of the circuit.
- (Total state = Internal state + Inputs)
- In the last example , the circuit has
- 4 stable total states: (y1y2x= 000, 011, 110, and 101)
- 4 unstable total states: (y1y2x= 001,010,111, and 100)

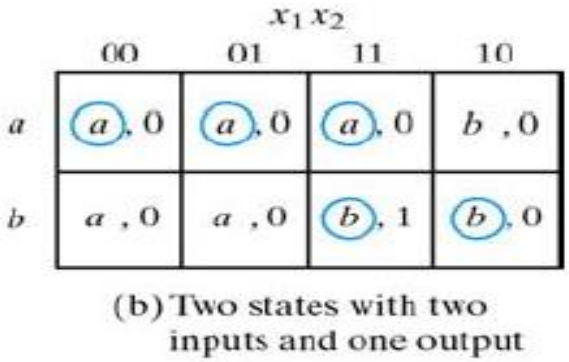
Flow Table:

- A flow table is similar to a transition table except that the internal state are symbolized with letters rather than binary numbers.
- It also includes the output values of the circuit for each stable state.



	x = 0	x = 1
a	(a)	b
b	c	(b)
c	(c)	d
d	a	(d)

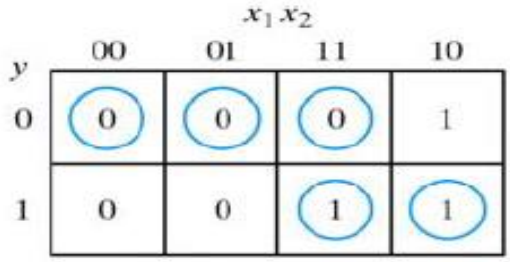
(a) Four states with one input



	x_1x_2 = 00	01	11	10
a	(a), 0	(a), 0	(a), 0	b, 0
b	a, 0	a, 0	(b), 1	(b), 0

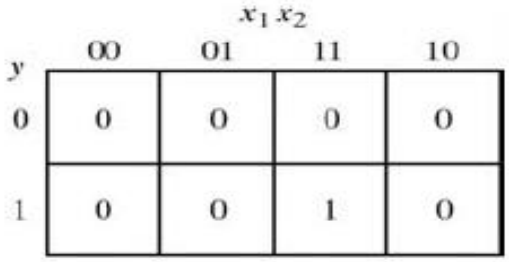
(b) Two states with two inputs and one output

- In order to obtain the circuit described by a flow table, it is necessary to convert the flow table into a transition table from which we can derive the logic diagram .
- This can be donethrough the assignment of a distinct binary value to each state.



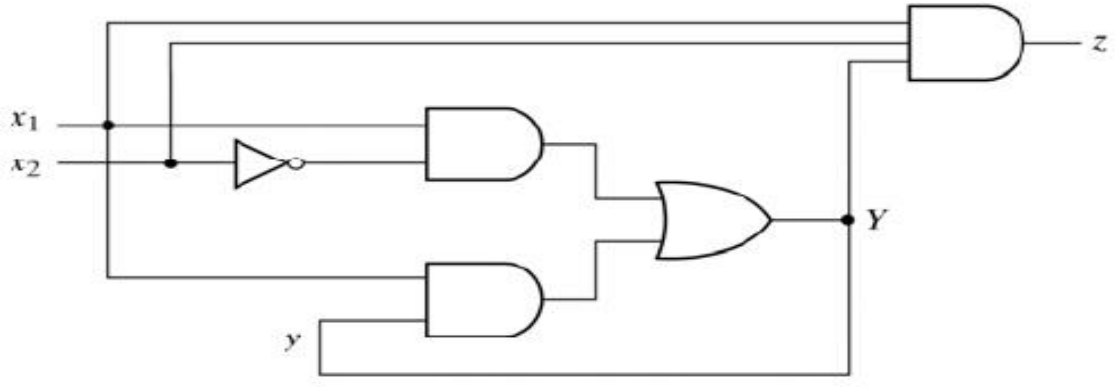
y	x_1x_2 = 00	01	11	10
0	(0)	(0)	(0)	1
1	0	0	(1)	(1)

(a) Transition table
$Y = x_1x'_2 + x_1y$



y	x_1x_2 = 00	01	11	10
0	0	0	0	0
1	0	0	1	0

(b) Map for output
$z = x_1x_2y$

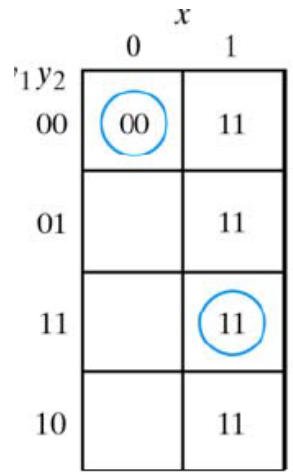


(c) Logic diagram

Race condition:

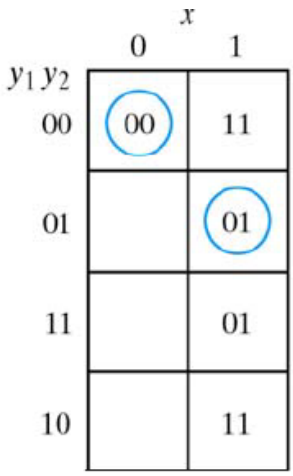
Two or more binary statevariables will change value when one input variable changes.

Cannot predict state sequence if unequal delay is encountered.



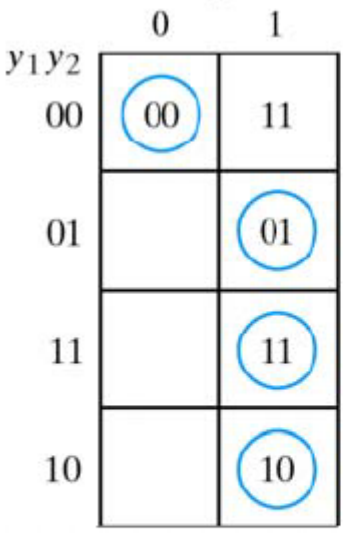
(a) Possible transitions:

$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10 \rightarrow 11$



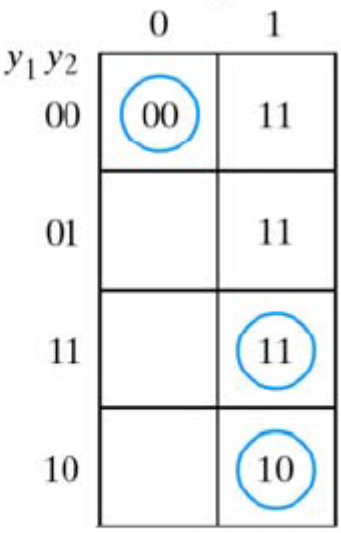
(b) Possible transitions:

$00 \rightarrow 11 \rightarrow 01$
$00 \rightarrow 01$
$00 \rightarrow 10 \rightarrow 11 \rightarrow 01$



(a) Possible transitions:

$00 \rightarrow 11$
$00 \rightarrow 01$
$00 \rightarrow 10$

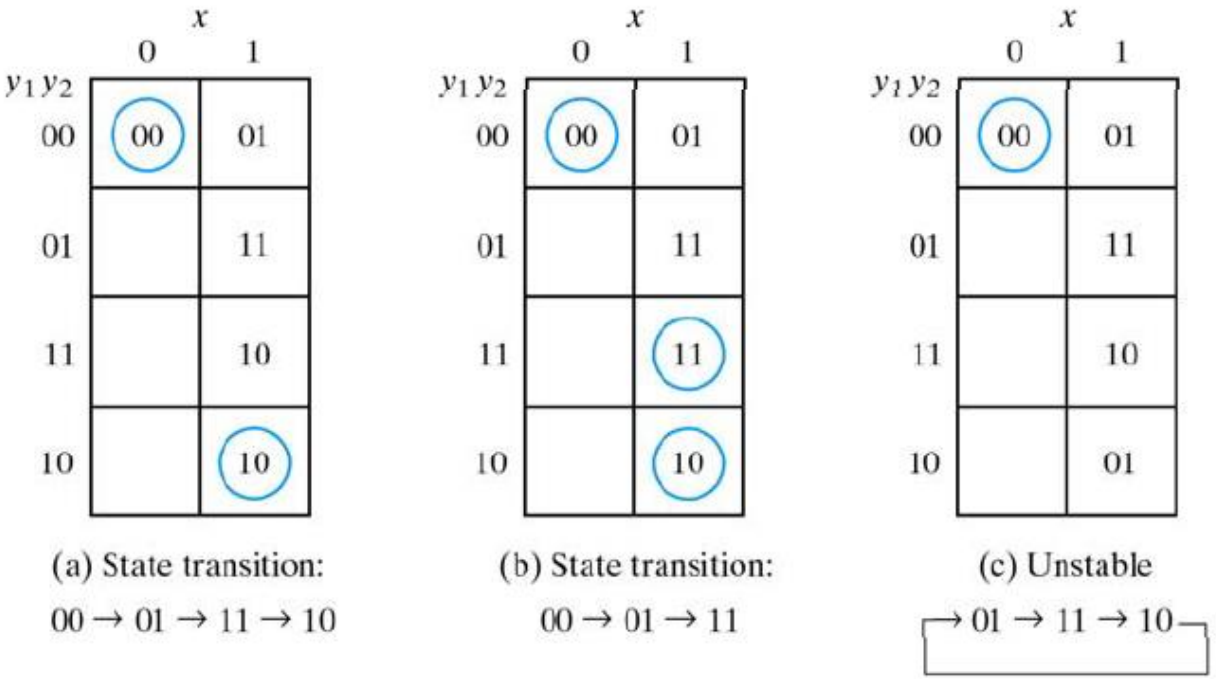


(b) Possible transitions:

$00 \rightarrow 11$
$00 \rightarrow 01 \rightarrow 11$
$00 \rightarrow 10$

Race Solution:

- It can be solved by making a proper binary assignment to the state variables.
- The state variables must be assigned binary numbers in such a way that only one state variable can change at any one time when a state transition occurs in the flow table.



Stability Check:

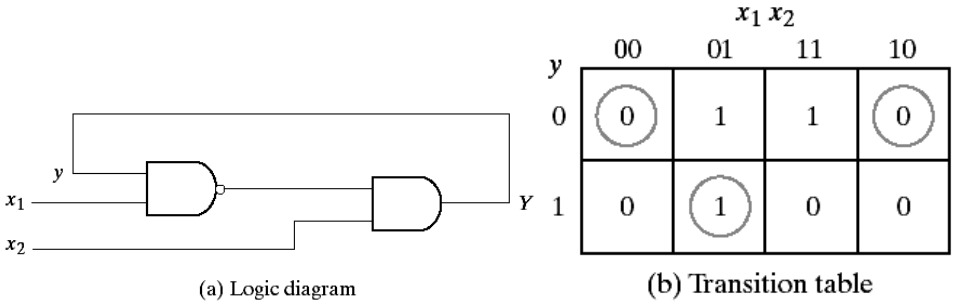
Asynchronous sequential circuits may oscillate between unstable states due to the feedback

Must check for stability to ensure proper operations can be easily checked from the transition table

Any column has no stable states unstable

Ex: when x_1x_2=11 in Fig. , Y and y are never the same

$Y = x'_1x_2 + x_2y$



UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 10
CIRCUITS WITH LATCHES

Circuits with Latches:

- The traditional configuration of asynchronous circuits is using one or more feedback loops
- No real delay elements.It is more convenient to employ the SR latch as a memory element in asynchronous circuits
- Produce an orderly pattern in the logic diagram with the memory elements clearly visible.
- SR latch is also an asynchronous circuit will be analyzed first using the method of ANALYSIS PROCEDURE asynchronous circuits.

Analysis Procedure:

Analysis Procedure for NOR latch based asynchronous circuit

(i) Label each latch o/p with Y_i and feed back path with y_i

(ii) Derive Boolean functions for S_i and R_i

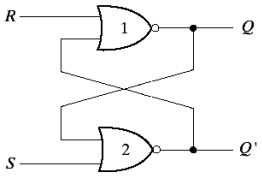
(iii) Check SR = 0 for each NOR latch

(iv) Evaluate Y = S + R'y for each latch

(v) Construct the transition table

(vi) Circle all stable states

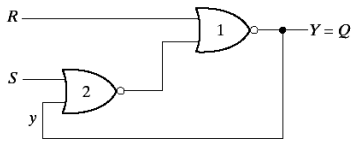
Circuits with Nor Latch:



(a) Crossed-coupled circuit

S	R	Q	Q'	
1	0	1	0	
0	0	1	0	(After SR = 10)
0	1	0	1	
0	0	0	1	(After SR = 01)
1	1	0	0	

(b) Truth table



(c) Circuit showing feedback

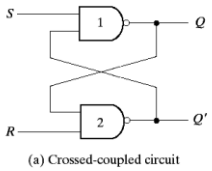
y \ SR	00	01	11	10
0	(0)	(0)	(0)	1
1	(1)	0	0	(1)

$Y = SR' + R'y$
$Y = S + R'y$ when $SR = 0$

(d) Transition table

S=1, R=1 (SR = 1) should not be used ⇒ SR = 0 is normal mode

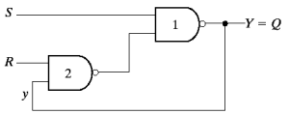
Circuits with Nand Latch:



(a) Crossed-coupled circuit

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(After SR = 10)
0	1	1	0	
1	1	1	0	(After SR = 01)
0	0	1	1	

(b) Truth table



(c) Circuit showing feedback

y \ SR	00	01	11	10
0	1	1	(0)	(0)
1	(1)	(1)	(1)	0

$Y = S' + Ry$ when $S'R' = 0$

(d) Transition table

S=0, R=0 (SR = 0) should not be used ⇒ SR =1 is normal mode .

UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 11
DESIGN PROCEDURE

Design procedure

1. Obtain a primitive table from specifications
2. Reduce flow table by merging rows in the primitive flow table
3. Assign binary state variables to each row of reduced table
4. Assign output values to dashes associated with unstable states to obtain the output map
5. Simplify Boolean functions for excitation and output variables
6. Draw the logic diagram.

Design Example:

Problem Statement:

Design a gated latch circuit (memory element) with two inputs, G(gate) and D(Data) and one output Q.

The Q output will follow the D input as long as G=1. when G goes to 0, the information that was present at the D input at the time of transition is retained at the Q output.

Primitive Flow Table:

• A primitive flow table is a flow table with only one stable total state (internal state + input) in each row.

• In order to form the primitive flow table , we first form a table with all possible total states.

- First, we fill in one square in each row belonging to the stable state in that row.
- Next we note that both inputs are not allowed to change at the same time, we enter dash marks in each row that differs in two or more variables from the input variables associated with the stable state.
- Next it is necessary to find values for two more squares in each row.
- The comments listed in the previous table may help in deriving the necessary information.
- All outputs associated with unstable states are marked with a dash.

	Input		Output	
State	D	G	Q	Comments
a	0	1	0	D=Q because G=1
b	1	1	1	D=Q because G=1
c	0	0	0	After states a or d
d	1	0	0	After state c
e	1	0	1	After states b or f
f	0	0	1	After state e

	DG = 00	01	11	10
a	c , –	(a) , 0	b , –	– , –
b	– , –	a , –	(b) , 1	e , –
c	(c) , 0	a , –	– , –	d , –
d	c , –	– , –	b , –	(d) , 0
e	f , –	– , –	b , –	(e) , 1
f	(f) , 1	a , –	– , –	e , –

Reduction of the Primitive Flow Table:

Two or more rows can be merged into one row if there are non_conflicting states and outputs in every columns.

After merged into one row:

- Don't care entries are overwritten
- Stable states and output values are included
- A common symbol is given to the merged row

	DG 00	01	11	10
a	c, –	(a), 0	b, –	–, –
c	(c), 0	a, –	–, –	d, –
d	c, –	–, –	b, –	(d), 0

	DG 00	01	11	10
b	–, –	a, –	(b), 1	e, –
e	f, –	–, –	b, –	(e), 1
f	(f), 1	a, –	–, –	e, –

(a) States that are candidates for merging

	DG 00	01	11	10
a, c, d	(c), 0	(a), 0	b, –	(d), 0
b, e, f	(f), 1	a, –	(b), 1	(e), 1

	DG 00	01	11	10
a	(a), 0	(a), 0	b, –	(a), 0
b	(b), 1	a, –	(b), 1	(b), 1

(b) Reduced table (two alternatives)

Transition Table and Logic Diagram:

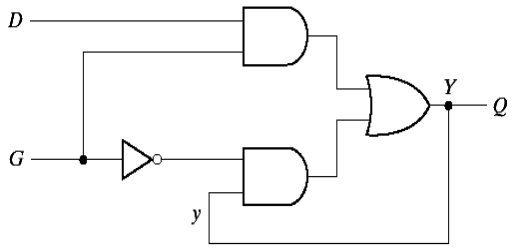
- In order to obtain the circuit described by the reduced flow table, it is necessary to assign a distinct binary value to each state.
- This converts the flow table to a transition table.
- A binary state assignment must be made to ensure that the circuit will be free of critical race. (This problem will be covered later) a=0, b=1 in this example.

	DG 00	01	11	10
y 0	0	0	1	0
1	1	0	1	1

(a) $Y = DG + G'y$

	DG 00	01	11	10
y 0	0	0	1	0
1	1	0	1	1

(b) $Q = Y$



UNIT -5

MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC

TOPIC – 12

Reduction of state and Flow Tables

Reduction of state and Flow Tables:

1. Implication Table
2. Merging of the Flow Table
3. Compatible Pairs
4. Maximal Compatibles
5. Closed Covering Condition

Implication Table:

- Equivalent States: Two states are equivalent if, for each possible input, they give exactly the same output and go to the same next states or to equivalent next states.
- Equivalent states can be combined into one state in the state table.
- The checking of each pair of states for possible equivalence in a table with a large number of states can be done systematically by means of an Implication Table.
- Implication Table: It is a chart that consists of squares, one for every possible pair of states.

Procedure to Evaluate Implication Table:

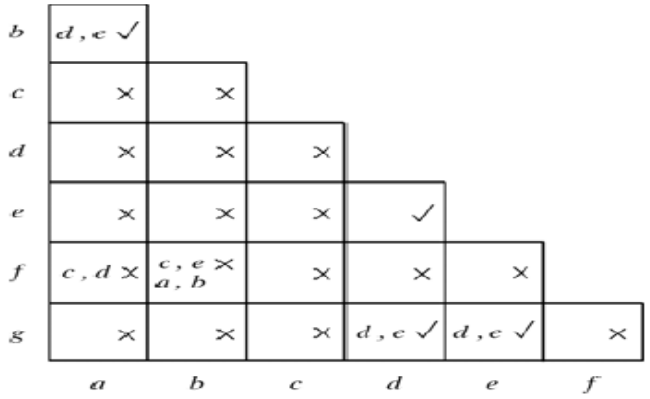
1. Place a cross in any square corresponding to a pair whose outputs are not equal

2. Enter a tick mark in the remaining squares the pairs of states that are implied by the pair of states representing the squares. (Start form the top square in the left column and going down and then proceeding with the next column to the right).

3. Make successive passes through the table to determine whether any additional squares should be marked with a 'x'.

State Table to Be Reduced

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	d	b	0	0
b	e	a	0	0
c	g	f	0	1
d	a	d	1	0
e	a	d	1	0
f	c	b	0	0
g	a	e	1	0



Merging of the Flow Table:

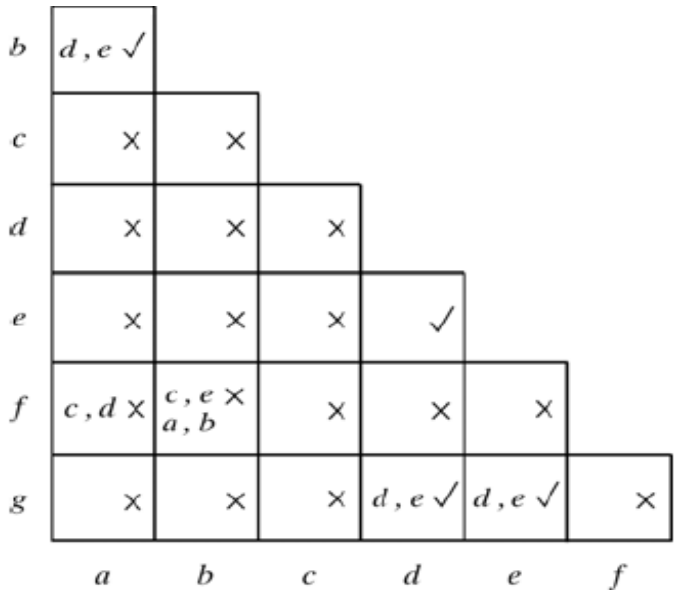
- The state table may be incompletely specified(Some next states and outputs are don't care).
- Primitive flow tables are always incompletely specified-Several synchronous circuits also have this property
- Incompletely specified states are not "equivalent" Instead, we are going to find "compatible" states
- Two states are compatible if they have the same output and compatible next states whenever specified.
- Three procedural steps:
  -Determine all compatible pairs
  - Find the maximal compatibles
  -Find a minimal closed collection of compatible

Compatible Pairs:

Finally we have [(a,b) ,(d,e) ,(d,g) , (e.g)] 4 states

State Table to Be Reduced

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	d	b	0	0
b	e	a	0	0
c	g	f	0	1
d	a	d	1	0
e	a	d	1	0
f	c	b	0	0
g	a	e	1	0



Maximal Coompatibles:

A group of compatibles that contains all the possible combinations of compatible states.

-Obtained from a merger diagram.

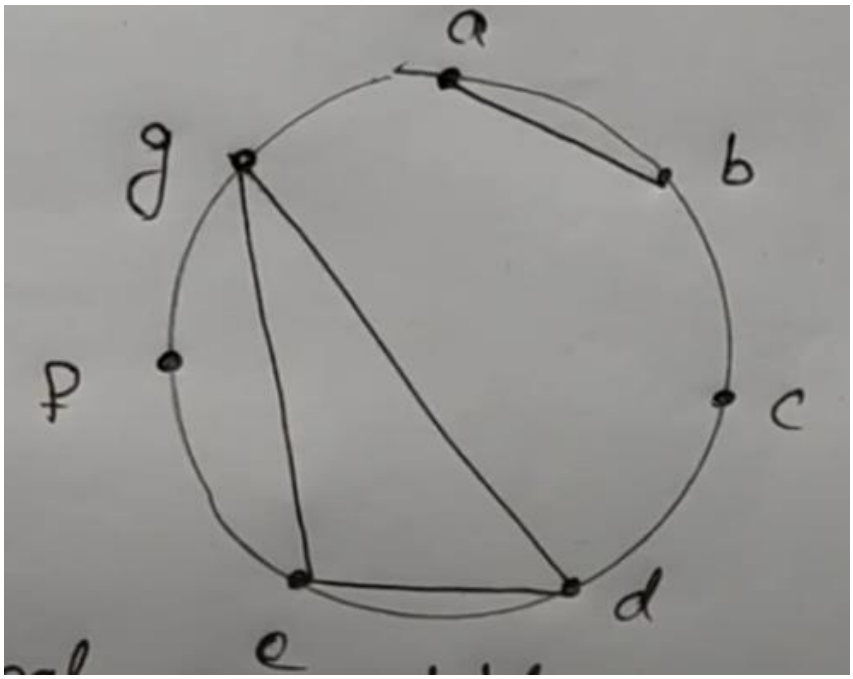
-A line in the diagram represents that two states are compatible.

n-state compatible n-sided fully connected polygon.

-All its diagonals connected.

Not all maximal compatibles are necessary.

-The maximal compatible states are [(a,b) ,(c),(d,e,g) , (f)]



Closed Covering Condition:

1. Cover all states.

2. Be closed: (The closure condition is satisfied if there are no implied states or if the implied states are included within the set)

-From the merger diagram, we determine the maximal compatibles:

(a,b) ,(c),(d,e,g) , (f)

-All the 4 states are included in this set.

- The implied states for (a,b) are (d,e,g). But (c) ,(f) are not include in the chosen set this set is not closed.

-A set of compatibles that will satisfy the closed covering condition is (a , b) (d,e,g).

Reduced State table:

State Table to Be Reduced

Present State	Next State		Output	
	$x = 0$	$x = 1$	$x = 0$	$x = 1$
a	d	b	0	0
b	e	a	0	0
c	g	f	0	1
d	a	d	1	0
e	a	d	1	0
f	c	b	0	0
g	a	e	1	0

Present state	Next state		Output	
	x=0	x=1	x=0	x=1
a	d	a	0	0
a	d	a	0	0
c	d	f	0	1
d	a	d	1	0
d	a	d	1	0
f	c	a	0	0
d	a	d	1	0

Present State	Next State		Output	
	x=0	x =1	x=0	x=1
a	d	a	0	0
c	d	f	0	1
d	a	d	1	0
f	c	a	0	0

UNIT -5

MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC

TOPIC – 13

HAZARDS

Hazards:

Hazards are unwanted switching transients that may appear at the output of a circuit because different paths exhibit different propagation delay.

Hazards occur in combinational and asynchronous circuits:

– In combination circuits, they may cause a temporarily false output value.

– In Asynchronous circuits, they may result in a transition to a wrong stable state.

Types Of Hazards:

1. Static 1-hazard □
2. Static 0-hazard
3. Dynamic hazard
4. Essential Hazards

Static hazard: A momentary output change when no output change should occur.

(a) Static 1-hazard

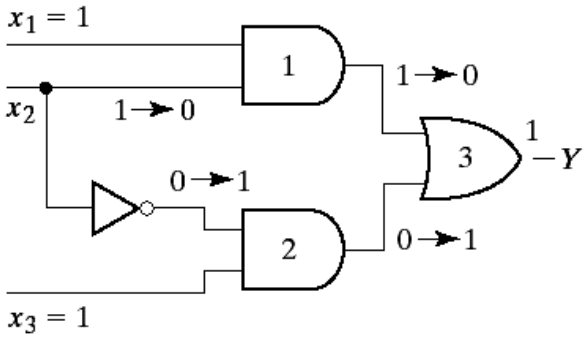
(b) Static 0-hazard

The dynamic hazard causes the output to change two, three or four times when it should change from 1 to 0 or from 0 to1.

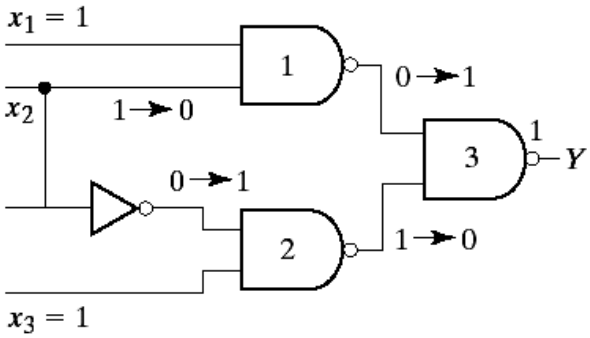
(c) Dynamic hazard

The occurrence of the hazard can be detected by inspecting the map of a particular circuit.

Two examples for static 1-hazard:



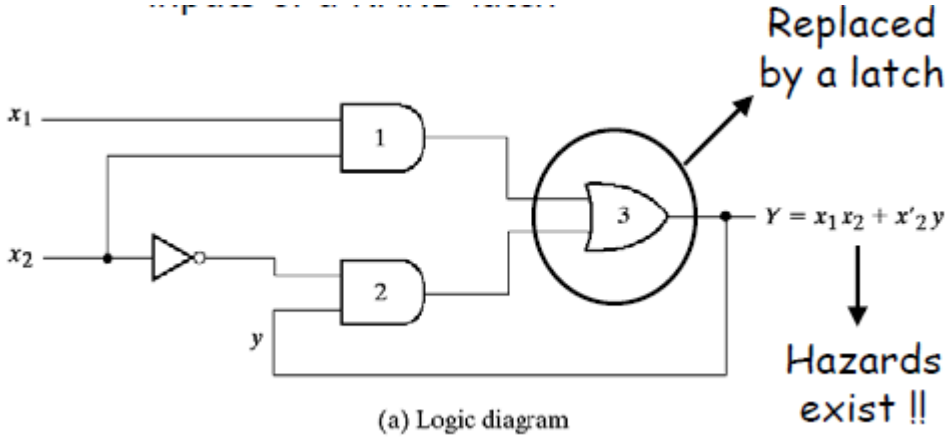
(a) AND-OR circuit



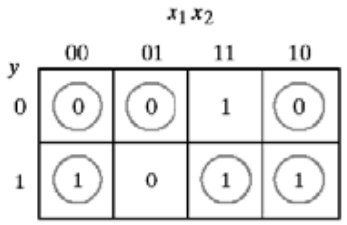
(b) NAND circuit

Remove Hazards with Latches

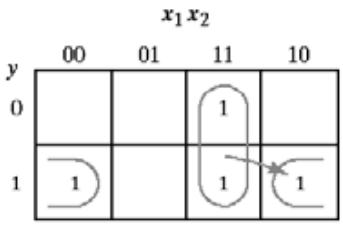
- Implement the asynchronous circuit with SR latches can also remove static hazards
- A momentary 0 has no effects to the S and R inputs of a NOR latch
- A momentary 1 has no effects to the S and R inputs of a NAND latch



(a) Logic diagram



(b) Transition table



(c) Map for Y

Esential Hazards:

Besides static and dynamic hazards, another type of hazard in asynchronous circuits is called: Essential Hazard Caused by unequal delays along two or more paths that originate from the same input cannot be corrected by adding redundant gate scan only be corrected by adjusting the amount of delay in the affected path.

- Each feedback path should be examined carefully !!

UNIT -5
MEMORIES AND ASYNCHRONOUS SEQUENTIAL LOGIC
TOPIC – 14
Design Example

Design Example-2:

- Recommended Design Procedure:

1. State the design specifications.
2. Derive a Primitive Flow Table.
3. Reduce the Flow Table by merging rows.
4. Make a race-free binary state assignment.
5. Obtain the transition table and output map.
6. Obtain the logic diagram using SR latches.

1) Design Specifications:

It is necessary to design a negative-edge-triggered T flip-flop. The circuit has two inputs T (toggle) and C (clock) and one output Q. The output state is complemented if T=1 and the clock changes from 1 to 0 (negative-edge-triggering). Otherwise, under all input condition, the output remains unchanged.

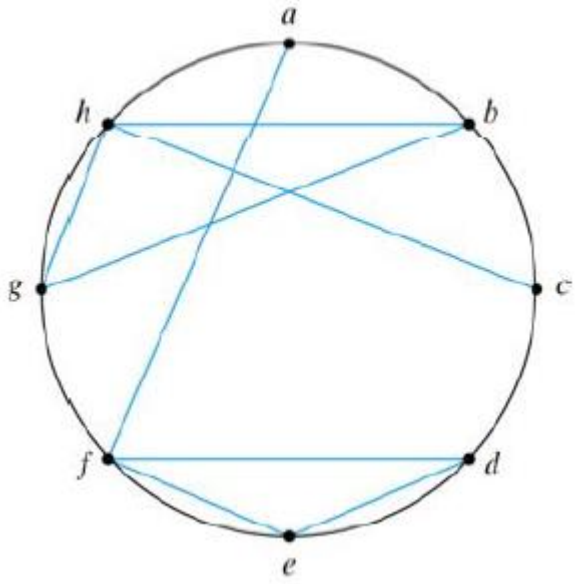
2) Primitive Flow Table:

	Inputs		Output	
State	T	C	Q	Comments
a	1	1	0	Initial output is 0
b	1	0	1	After state a
c	1	1	1	Initial output is 1
d	1	0	0	After state c
e	0	0	0	After state d or f
f	0	1	0	After state e or a
g	0	0	1	After state b or h
h	0	1	1	After State g or c

	TC = 00	TC = 01	TC = 11	TC = 10
a	– , –	f , –	(a), 0	b , –
b	g , –	– , –	c , –	(b), 1
c	– , –	h , –	(c), 1	d , –
d	e , –	– , –	a , –	(d), 0
e	(e), 0	f , –	– , –	d , –
f	e , –	(f), 0	a , –	– , –
g	(g), 1	h , –	– , –	b , –
h	g , –	(h), 1	c , –	– , –

Merging of the Flow Table:

	a	b	c	d	e	f	g
b	a , c ×						
c	×	b , d ×					
d	b , d ×	×	a , c ×				
e	b , d ×	e , g × b , d ×	f , h ×	✓			
f	✓	e , g × a , c ×	f , h × a , c ×	✓	✓		
g	f , h ×	✓	b , d ×	e , g × b , d ×	×	e , g × f , h ×	
h	f , h × a , c ×	✓	✓	d , e × c , f ×	e , g × f , h ×	×	✓



The maximal compatibles pairs are: (a , f) (b , g , h) (c , h) (d , e , f)

In this particular example, the minimal collection of compatibles is also the maximal compatibles set: (a , f) (b , g , h) (c , h) (d , e , f).

TC

	00	01	11	10
a, f	e , –	(f) , 0	(a) , 0	b , –
b, g, h	(g) , 1	(h) , 1	c , –	(b) , 1
c, h	g , 1	(h) , 1	(c) , 1	d , –
d, e, f	(e) , 0	(f) , 0	a , –	(d) , 0

(a)

TC

	00	01	11	10
a	d , –	(a) , 0	(a) , 0	(b) , –
b	(b) , 1	(b) , 1	c , –	(b) , 1
c	b , –	(c) , 1	(c) , 1	d , –
d	(d) , 0	(d) , 0	a , –	(d) , 0

(b)

4) State Assignment and Transition Table

No diagonal lines in the transition diagram:No need to add extra states.

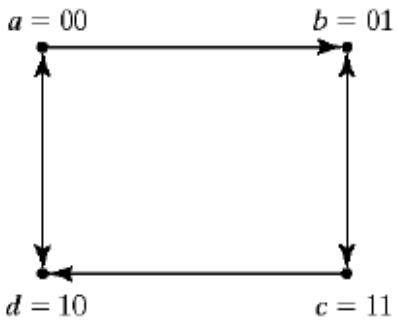


Fig. 9-43 Transition Diagram

$y_1 y_2$	TC = 00	TC = 01	TC = 11	TC = 10
$a = 00$	10	(00)	(00)	01
$b = 01$	(01)	(01)	11	(01)
$c = 11$	01	(11)	(11)	10
$d = 10$	(10)	(10)	00	(10)

(a) Transition table

$y_1 y_2$	TC = 00	TC = 01	TC = 11	TC = 10
00	0	0	0	X
01	1	1	1	1
11	1	1	1	X
10	0	0	0	0

(b) Output map $Q = y_2$

5) Logic Diagram:

y_1y_2	TC = 00	TC = 01	TC = 11	TC = 10
00	(1)	0	0	0
01	0	0	(1)	0
11	0	X	(X)	X
10	(X)	X	0	X

(a) $S_1 = y_2\,TC + y'_2\,T'C'$

y_1y_2	TC = 00	TC = 01	TC = 11	TC = 10
00	0	X	(X)	X
01	(X)	X	0	X
11	(1)	0	0	0
10	0	0	(1)	0

(b) $R_1 = y_2\,T'C' + y'_2\,TC$

y_1y_2	TC = 00	TC = 01	TC = 11	TC = 10
00	0	0	0	(1)
01	X	X	X	(X)
11	X	X	X	0
10	0	0	0	0

(c) $S_2 = y'_1\,TC'$

y_1y_2	TC = 00	TC = 01	TC = 11	TC = 10
00	X	X	X	0
01	0	0	0	0
11	0	0	0	(1)
10	X	X	X	(X)

(d) $R_2 = y_1\,TC'$