CHAPTER – 6

MSI AND LSI COMPONENTS IN COMBINATIONAL LOGIC DESIGN

SSI (Small Scale Integration)

- No. of gates = upto 10 per IC
- **Example:** Logic gates (AND, OR, NAND, NOR), 74XX series (7400)

MSI (Medium Scale Integration)

- No. of gates = 10 to 100 per IC
- **Example:** Flip-flops, adder/counters, MUX/DEMUX, encoder, decoder, 74XXX and 45XXX series, etc.

LSI (Large Scale Integration)

- No. of gates = 100 – 10,000 per IC
- **Example:** Programmable logic array (PLA), read only memory (ROM), small memory chip, etc.

VLSI (Very Large Scale Integration)

- No. of gates = 10,000 – 1,00,000 per IC
- **Example:** Complex programmable logic device

ULSI (Ultra Large Scale Integration)

- No. of gates = 1,00,000 – 10,00,000 per IC
- **Example:** 8- and 16-bit microprocessor

GSI (Giga Scale Integration)

- No. of gates > 10,00,000 per IC
- **Example:** Pentium V processor

6.1 Binary Adder and Subtractor

Binary Parallel Adder (BPA)

A binary parallel adder is a digital function that produces the arithmetic sum of two binary numbers in parallel. It consists of full adders

connected in cascade, with the output carry from one full adder connected to the input carry of the next full adder. An 'n' bit parallel adder requires 'n' full adders.

Construction of 4-Bit Binary Parallel Adder

4-bit binary parallel adder requires 4 full adders.

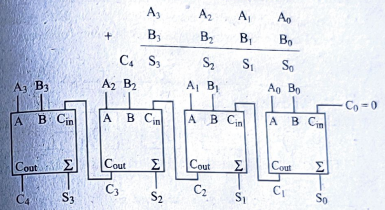


Fig.: 4-bit binary parallel adder.

Above binary parallel adder uses a ripple carry method in which the carry output of each full adder is connected to the carry input of the next higher order stage.

Controlled Inverter

When invert is LOW, it transmits the 8-bit input to the output.

When invert is HIGH, it transmits the 1's complement of the input.

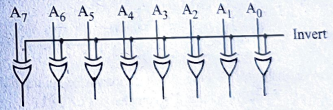


Fig.: Circuit diagram of a controlled inverter

Invert	A	F
0	0	0 (F = A)
0	1	1(F = A)
1	0	1(F = $\bar{A}$)
1	1	0(F = $\bar{A}$)

Importance:

During subtraction, we first need to take 2's complement of the subtrahend then we add the complemented subtrahend to obtain the answer. With a controlled inverter, we can produce a 1's complement.

Binary Adder-Subtractor

A binary adder-subtractor is a combinational circuit that performs the arithmetic operations of addition and subtraction with binary numbers.

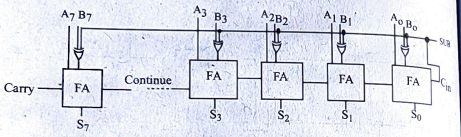


Fig.: 8-bit adder/subtractor

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	B7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
+	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

When SUB = LOW, the circuit performs the addition. The binary numbers B_7 to B_0 passes through the controlled inverter with no change. The full adders then produce the correct output sum as

0←SUB

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
	B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
+	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

When SUB = HIGH, the circuit performs the subtraction. Therefore, the controlled inverter produces 1's complement of B_0 to B_7. Furthermore, SUB is the carry into the first full adder (i.e. 1's complement +1 = 2's complement). The circuit process the data like this.

1 ←SUB

	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
+	$\overline{B_7}$	$\overline{B_6}$	$\overline{B_5}$	$\overline{B_4}$	$\overline{B_3}$	$\overline{B_2}$	$\overline{B_1}$	$\overline{B_0}$
	S_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

6.3 Magnitude Comparator

Magnitude comparator is a combinational circuit, designed to compare the two n-bit words applied as its input. The comparator has three outputs namely greater (A > B), lesser (A < B) and equal (A = B). Depending upon the result of comparison, one of these outputs will go high.

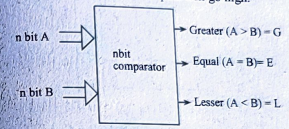
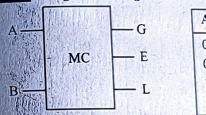


Fig.: Block diagram of an n-bit comparator.

EXAMPLE: Design 1-bit magnitude comparator.



A	B	G	E	L
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Fig.: Block diagram of a 1-bit comparator.

$G = A\overline{B}$

$E = A \odot B$

$L = \overline{A}B$

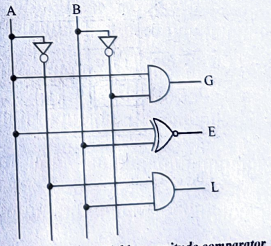
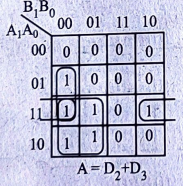


Fig.: Logic diagram of 1-bit magnitude comparator

EXAMPLE 3 Design 2-bit magnitude comparator.

Inputs				Outputs		
A_1	A_0	B_1	B_0	G	E	L
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

For G (A >B),

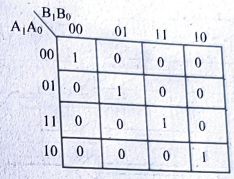


$A = D_2 + D_3$

$G = A_1\overline{B}_1 + A_0\overline{B}_1\overline{B}_0 + A_1A_0\overline{B}_0$

$= A_1\overline{B}_1 + A_0\overline{B}_0(\overline{B}_1 + A_1)$

For E,



$E = \overline{A}_1\overline{A}_0\overline{B}_1\overline{B}_0 + \overline{A}_1A_0\overline{B}_1B_0 + A_1A_0B_1B_0 + A_1\overline{A}_0B_1\overline{B}_0$

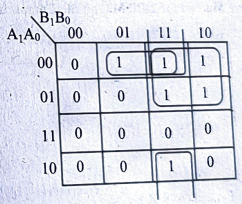
$= \overline{A}_1\overline{B}_1(\overline{A}_0\overline{B}_0 + A_0B_0) + A_1B_1(A_0B_0 + \overline{A}_0\overline{B}_0)$

$= \overline{A}_1\overline{B}_1(A_0 \odot B_0) + A_1B_1(A_0 \odot B_0)$

$= (\overline{A}_1\overline{B}_1 + A_1B_1)(A_0 \odot B_0)$

$= (A_1 \odot B_1)(A_0 \odot B_0)$

For L,



$L = \overline{A}_1B_1 + \overline{A}_1\overline{A}_0B_0 + \overline{A}_0B_1B_0$

$= \overline{A}_1B_1 + \overline{A}_0B_0(\overline{A}_1 + B_1)$

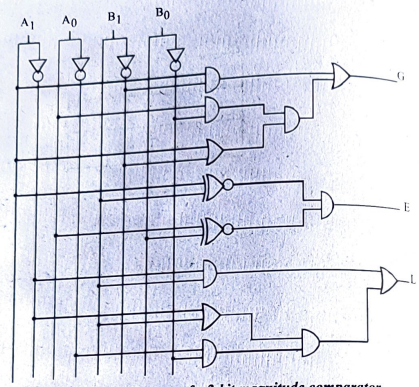


Fig.: Circuit diagram of a 2-bit magnitude comparator

Alternative Method for 2-Bit Magnitude Comparator

$G = A\bar{B}$

$E = A \odot B$

$L = \bar{A}\,B$ or $(G + E)'$

Let, $A = A_1\,A_0$

$B = B_1\,B_0$

For equal $(A = B)$,

$(A_1 = B_1)$ and $(A_0 = B_0)$

$E = (A_1 \odot B_1) \cdot (A_0 \odot B_0) = E_1 \cdot E_0$

For greater $(A > B)$,

Case I: $A_1 = 1, B_1 = 0 \Rightarrow A_1\,\bar{B}_1$

Case II: $(A_1 = B_1)\,(A_0 = 1$ and $B_0 = 0) \Rightarrow (A_1 \odot B_1) \cdot (A_0\,\bar{B}_0)$

$\therefore\ G = A_1\,\bar{B}_1 + (A_1 \odot B_1)A_0\,\bar{B}_0 = G_1 + E_1G_0$

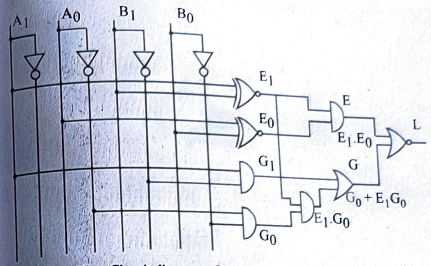


Fig.: Circuit diagram of a 2-bit magnitude comparator

6.4 Decoder and Encoder

6.4.1 Decoder

A decoder is a combinational circuit that convert binary information from n input lines to a maximum of 2^n unique output lines. If the n-bit coded information has unused combinations, the decoder may have fewer than 2^n outputs.

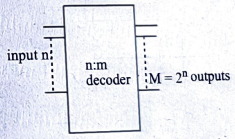


Fig.: Block diagram of a n:m decoder

Applications:

It has many applications. For example, in computer, it is used to select input/output ports. The binary port address is decoded and the appropriate peripherals is selected for communication. Each I/O port has a unique address.

1. 2:4 Decoder

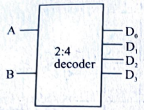


Fig.: Block diagram of a 2:4 decoder

Truth table

Input		Output			
A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

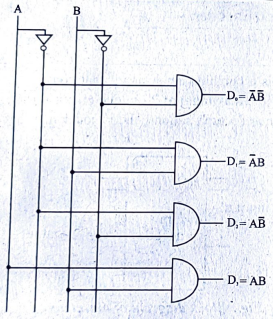


Fig.: 2: 4 decoder

2. 3×8 Decoder (Binary to Octal Converter)

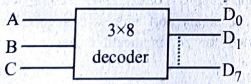


Fig.: Block diagram of a 3×8 decoder

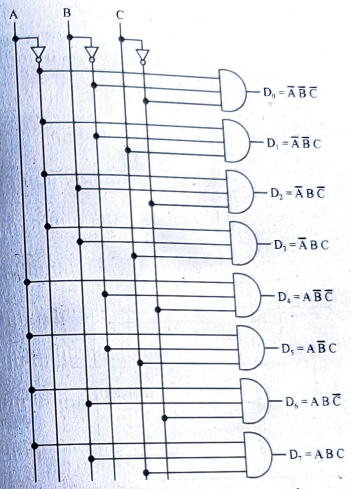


Fig.: Three-to-eight line decoder (binary to octal converter)

Truth table

Inputs			Outputs							
A	B	C	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

Construction of a 3 ×8 Decoder using 2×4 Decoder

For 3×8 decoder, no. of inputs = 3 (E, A, B), no. of outputs = 8

For 2×4 decoder, no. of inputs = 2 (A, B), no. of outputs = 4

Truth table

E	A	B	D	
0	0	0	D_0	2 × 4 decoder 'I' when E = 0
0	0	1	D_1	
0	1	0	D_2	
0	1	1	D_3	
1	0	0	D_4	2 × 4 decoder 'II' when E = 1
1	0	1	D_5	
1	1	0	D_6	
1	1	1	D_7	

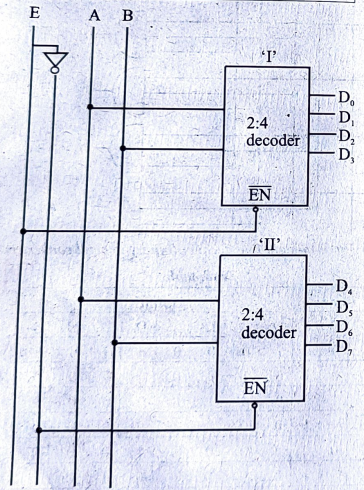


Fig.: 3×8 decoder using 2×4 decoder

3. **BCD to Decimal Decoder**

Inputs				Outputs									
A	B	C	D	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1

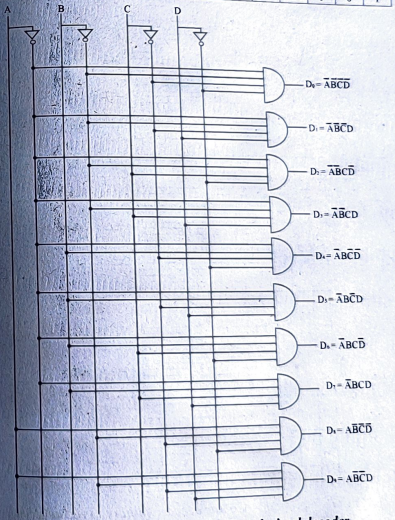


Fig.: Circuit diagram of a BCD to decimal decoder.

4. 4:16 Line Decoder (Binary to Hexadecimal Converter)

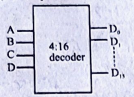


Fig.: Block diagram of a 4:6 decoder.

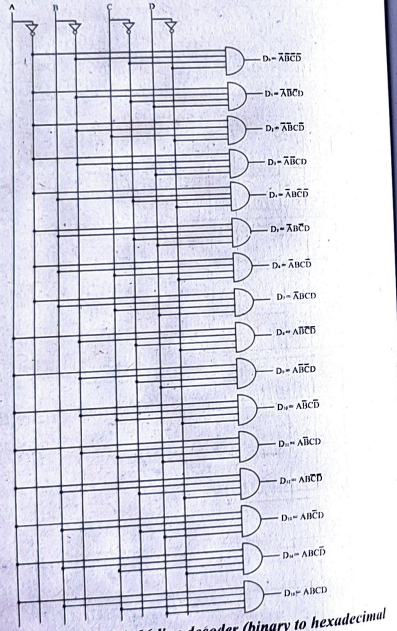


Fig.: Circuit diagram of 4×16 line decoder (binary to hexadecimal converter)

Decoder with Enable

In different application, an additional signal input called a strobe is added so that the gates are enabled and decoding take place. It is sometime desired to decode only during certain interval of time. In such cases, decoder with enable is used.

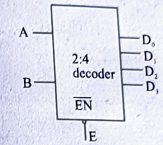


Fig.: A 2:4 decoder with an enable input

Inputs			Outputs			
E	A	B	D_0	D_1	D_3	D_3
1	×	×	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

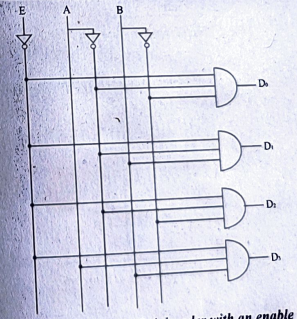


Fig.: Logic diagram of a 2:4 decoder with an enable input.

Decoder can work as a demultiplexer when the input (A, B) of decoder are made selection inputs (S_0, S_1) and enable signal as the data input.

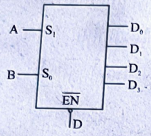


Fig.: Decoder working as a demultiplexer

6.4.2 Encoder

An encoder is a combinational circuit that performs the inverse operation of a decoder. It has 2^n inputs only one of which is active and n outputs. It converts an active input signal into a coded output signal.

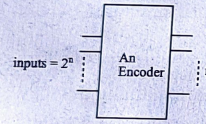


Fig.: Block diagram of an encoder

Limitations:

Only one input can be enabled at a time. If two inputs are enabled at the same time, then output is undefined.

1. 4:2 Encoder

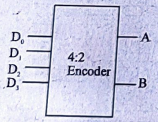


Fig.: Block diagram of a 4:2 encoder

$A = D_2 + D_3$

$B = D_1 + D_3$

Inupts				Outputs	
D_0	D_1	D_2	D_3	A	B
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

For A,

D_0D_1 \ D_2D_3	00	01	11	10
00	x	1	x	1
01	0	x	x	x
11	x	x	x	x
10	0	x	x	x

$A = D_2 + D_3$

For B,

D_0D_1 \ D_2D_3	00	01	11	10
00	x	1	x	0
01	1	x	x	x
11	x	x	x	x
10	0	x	x	x

$B = D_1 + D_3$

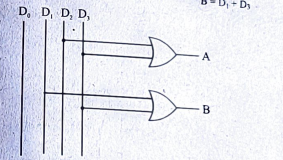


Fig.: 4×2 encoder

2. 8:3 Encoder (Octal to Binary Converter)

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_3 + D_5 + D_7$

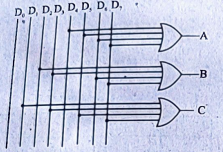


Fig.: Circuit diagram of an 8×3 encoder

Priority Encoder

This is a special type of encoder. In this encoder, priorities are given to the input lines. If two or more input lines are 1 at same time, then the input line with highest priority shall be considered.

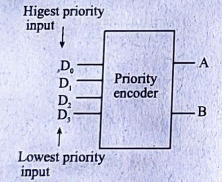


Fig.: Block diagram of a priority encoder.

4×2 Priority Encoder

Let D_0 has highest priority.

Truth table

Inputs				Outputs	
D_0	D_1	D_2	D_3	A	B
0	0	0	0	×	×
1	×	×	×	0	0
0	1	×	×	0	1
0	0	1	×	1	0
0	0	0	1	1	1

$A = \bar{D}_0 \bar{D}_1 D_2 + \bar{D}_0 \bar{D}_1 \bar{D}_2 D_3 = \bar{D}_0 \bar{D}_1 (D_2 + \bar{D}_2 D_3)$

$= \bar{D}_0 \bar{D}_1 (D_2 + \bar{D}_3)$

$B = \bar{D}_0 D_1 + \bar{D}_0 \bar{D}_1 \bar{D}_2 D_3$

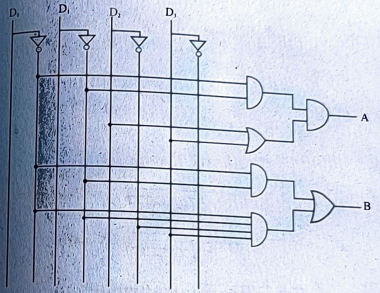


Fig.: Circuit diagram of a 4:2 priority enocoder

Construction of 8:3 Priority Encoder

Let D_7 has highest priority and D_0 has lowest priority

Truth table

Inputs								Outputs		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A	B	C
1	×	×	×	×	×	×	×	0	0	0
0	1	×	×	×	×	×	×	0	0	1
0	0	1	×	×	×	×	×	0	1	0
0	0	0	1	×	×	×	×	0	1	1
0	0	0	0	1	×	×	×	1	0	0
0	0	0	0	0	1	×	×	1	0	1
0	0	0	0	0	0	1	×	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$A = \overline{D}_7\,\overline{D}_6\overline{D}_5\,\overline{D}_4\,D_3 + \overline{D}_7\,\overline{D}_6\overline{D}_5\,\overline{D}_4\,\overline{D}_3\,D_2 + \overline{D}_7\,\overline{D}_6\overline{D}_5\,\overline{D}_4\,\overline{D}_3\,D_2D_1 +$

$\overline{D}_7\,\overline{D}_6\overline{D}_5\,\overline{D}_4\,\overline{D}_3\,\overline{D}_2\overline{D}_1D_0$

$B = \overline{D}_7\,\overline{D}_6D_5 + \overline{D}_7\overline{D}_6D_5D_4 + \overline{D}_7\overline{D}_6\overline{D}_5\overline{D}_4\overline{D}_3D_2D_1 +$

$\overline{D}_7\,\overline{D}_6D_5\,\overline{D}_4\,\overline{D}_3\,\overline{D}_2\overline{D}_1D_0$

$C = \overline{D}_7\,D_6 + \overline{D}_7\,\overline{D}_6\,\overline{D}_5\,D_4 + \overline{D}_7\overline{D}_6\overline{D}_5\overline{D}_4\,D_3D_2 + \overline{D}_7\,\overline{D}_6D_5\,\overline{D}_4\,D_3\,D_2D_1D_0$

It is very complex to draw logic circuit. So, we can write as 8:3 encoder.

$A = D_4 + D_5 + D_6 + D_7$

$B = D_2 + D_3 + D_6 + D_7$

$C = D_1 + D_2 + D_5 + D_7$

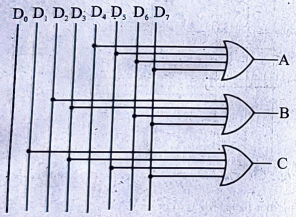


Fig.: Circuit diagram of an 8:3 encoder

6.5 Multiplexer and Demultiplexer

6.5.1 Multiplexer (Data Selector)

Multiplex means 'many to one'. A multiplexer is a combinational circuit which selects single information from multiple inputs one at a time with help of selection line. Multiplexing is the process of transmitting a large number of information over a single line. For 'n' inputs, there are 'm' selection line and a single output where $2^m = n$.

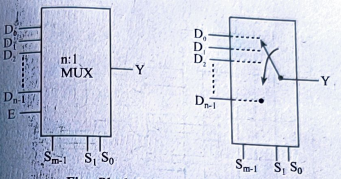


Fig.: Block diagram of a multiplexer.

Applications:

- As a building block in the CPU.
- Used in the telecommunication at the transmitter side.

Advantages of multiplexer:

- It reduces the number of wire. Hence, it reduces the circuit complexity and cost.
- We can implement many combinational circuits using MUX.

1. 4×1 MUX

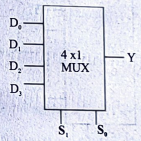


Fig.: Block diagram of a 4:1 multiplexer

Truth table

Inputs						Outputs
D_0	D_1	D_2	D_3	S_1	S_0	Y
1	0	0	0	0	0	1
0	1	0	0	0	1	1
0	0	1	0	1	0	1
0	0	0	1	1	1	1

$Y = \overline{S}_1\overline{S}_0D_0 + \overline{S}_1S_0\,D_1 + S_1\,\overline{S}_0\,D_2 + S_1\,S_0\,D_3$

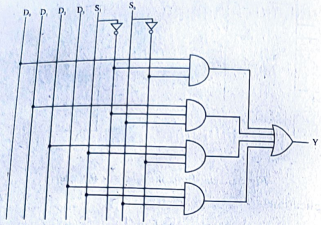


Fig.: Logic diagram of 4:1 MUX

6.5.2 Demultiplexer (Data Divider)

'Demultiplexer' means "one to many". A demultiplexer is a combinational circuit that receives information on a single input and transmits the same information over one of the possible output lines. Selection of output lines is controlled by selection line. For 'n' output, there is 'm' selection line and single input where $n = 2^m$.

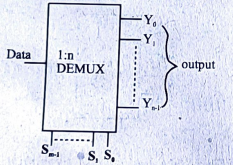


Fig.: Block diagram of a demultiplexer.

1. 1:4 DEMUX

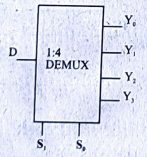


Fig.: Block diagram of a 1:4 demultiplexer.

Truth table

Inputs			Outputs			
D_i	S_1	S_0	Y_0	Y_1	Y_2	Y_3
D	0	0	D	0	0	0
D	0	1	0	D	0	0
D	1	0	0	0	D	0
D	1	1	0	0	0	D

$Y_0 = S_1 S_0 D_0 \quad Y_1 = S_1 S_0 D_1 \quad Y_2 = S_1 S_0 D_2 \quad Y_3 = S_1 S_0 D_3$

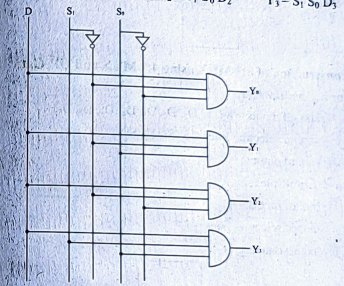


Fig.: Circuit diagram of a 1:4 demultiplexer

Construction of 4:1 Multiplexer using 2: 1 Multiplexer

For 4:1 multiplexer,

no. of inputs = 4 (D_0, D_1, D_2, D_3)

no. of selection lines = 2 (S_1, S_0)

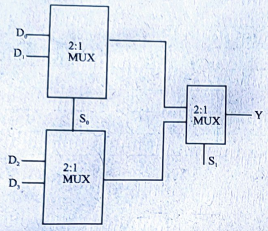
no. of outputs = 1

For 2:1 multiplexer,

no. of inputs = 2

no. of selection lines = 1

no. of outputs = 1



Construction of 8:1 MUX using 4:1 MUX and OR Gate

For 8:1 multiplexer,

no. of inputs = 8 (D_0, D_1, D_2, D_3, D_4, D_5, D_6, D_7)

no. of selection lines = 3 (S_2, S_1, S_0)

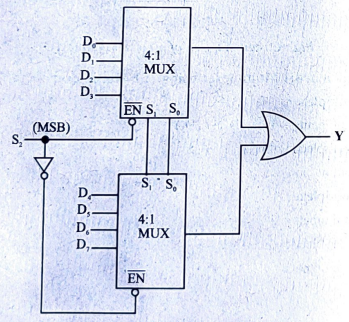
no. of outputs = 1

For 4:1 multiplexer,

no. of inputs = 4

no. of selection lines = 2

no. of outputs = 1



Construction of 32:1 MUX using 16 :1 MUX and 2:1 MUX

For 32:1 multiplexer,

no. of inputs = 32 (D_0, D_1, ……, D_{30}, D_{31})

no. of selection lines = 5 (S_4, S_3, S_2, S_1, S_0)

no. of outputs = 1

For 16:1 multiplexer,

no. of inputs = 16

no. of selection lines = 4

no. of outputs = 1

For 2:1 multiplexer,

no. of inputs = 2

no. of selection lines = 1

no. of outputs = 1

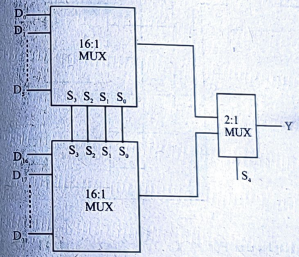
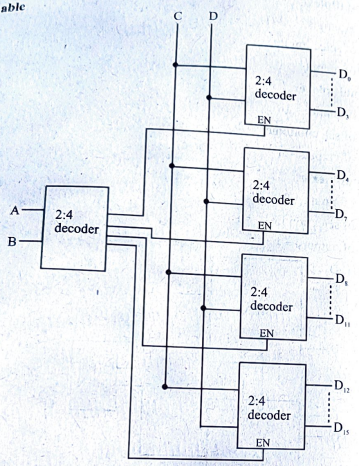


Fig.: 32:1 MUX using 16:1 MUX and 2:1 MUX

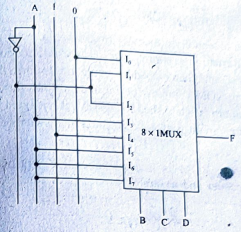
Construction of 4:16 Line Decoder with Five 2:4 Line Decoder with Enable



EXAMPLE: **Implement F(A, B, C, D) = Σm (1, 2, 4, 11, 12, 13, 14, 15) using 8×1 MUX.**

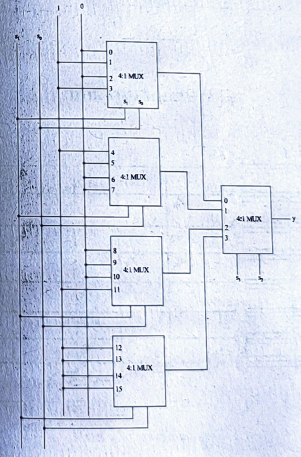
⇒

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	0	①	②	3	④	5	6	7
A	8	9	10	⑪	⑫	⑬	⑭	⑮
	0	$\bar{A}$	$\bar{A}$	A	1	A	A	A



EXAMPLE: **Implement F(A, B, C, D) = Σm (1, 3, 4, 11, 12, 13, 14, 15) using 4×1 MUX.**

⇒ No. of selection lines = 4 (S_3, S_2, S_1, S_0)



6.6 Read Only Memory (ROM)

A ROM is essentially a memory (or storage) device in which a fixed set of binary information is stored. The binary information must be first specified by the user and then embedded in the unit to form the required interconnection pattern. ROM contains special internal links that can be fused or broken. Once a pattern is established for a ROM, it remained fixed even if the power/supply to the circuit is switched off and then switched on again i.e., it is non-volatile.

Internally the ROM is a combinational circuit with AND gates connected as a decoder and a number of OR gates equal to the number of outputs in the unit.

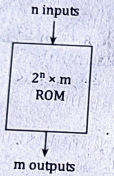


Fig.: Block diagram of a ROM

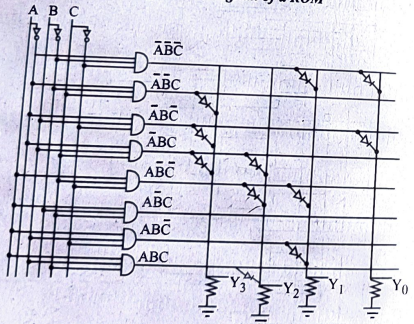


Fig.: Circuit diagram of a ROM

$Y_0 = \bar{A}\bar{B}\bar{C} + \bar{A}B\bar{C}$

$Y_1 = \bar{A}\bar{B}C + A\bar{B}\bar{C} + ABC$

$Y_2 = \bar{A}BC + A\bar{B}\bar{C} + ABC$

$Y_3 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC$

Example: 32 × 4 ROM

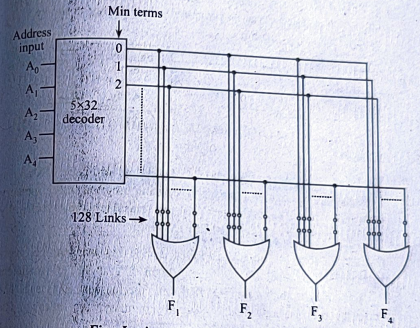


Fig.: Logic construction of a 32 × 4 ROM

6.6.1 Types of ROM

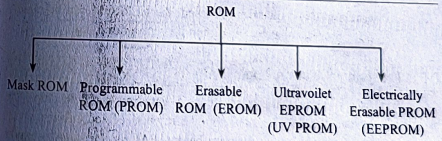


Fig.: Classification of ROM

1. Mask ROM

The mask ROM is usually referred to as a ROM. It is permanently programmed during the manufacturing process to provide widely used standard functions to provide user specified functions. Once the

memory is programmed during the manufacturing process, the user cannot alter the programs. E.g., CD (Compact Disk)

2. **PROM**

The PROM uses the fusing process to store bits – the memory link is burned open or left intact to present 0 or 1. The fusing process is irreversible; once PROM is programmed, it cannot be changed.

A PROM consists of a set of fixed (non-programmable) AND gates connected as a decoder and a programmable OR array as shown in the generalized block diagram.

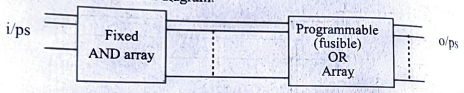


Fig.: Block diagram of PROM

3. **EPROM**

An EPROM is an erasable PROM. Unlike an ordinary PROM, an EPROM can be reprogrammed if an existing program in the memory array is erased. Erasable PROM is erasable and electrically reprogrammable. Basic types of EPROM are:

(i) **UV EPROM:** The UV EPROM device has transparent quartz window on the IC package. When UV light of sufficient frequency is shown for specified time, the memory is erased. The discharge period takes several minutes to hours. The programming is same as PROM.

(ii) **EEPROM:** An electrically erasable PROM can be erased and programmed with electrical pulse. Since it can be both electrically written into and electrically erased, the EEPROM can be rapidly programmed, and erased in circuit for reprogramming.

6.7 Programmable Array Logic (PAL)

PAL is a programmable array of logic gates on a single chip. The basic PAL consists of a programmable AND array and a fixed OR array with output logic as shown in block diagram below. PAL is the most common one–time programmable (OTP) logic device and is implemented with bipolar technology (TTL or ECL).

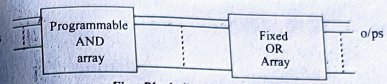


Fig.: Block diagram of PAL

6.8 Programmable Logic Arrays (PLA)

A PLA consists of a programmable AND array and a programmable OR array. The PLA is also called a Field Programmable Logic Array (FPLA) because the user in the field programs it, not the manufacturer.

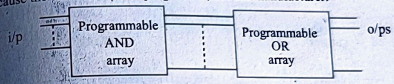


Fig.: Block diagram of PLA

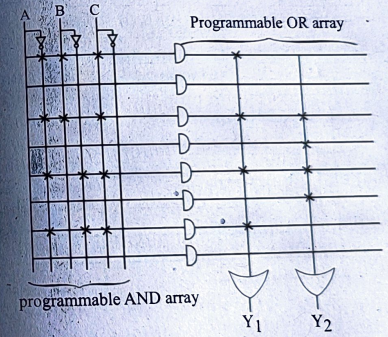


Fig.: Circuit diagram of PLA

EXAMPLE: Implement function using PLA:

$F_0 = \Sigma_m (0, 1, 4, 6), F_1 = \Sigma_m (2, 3, 4, 6, 7), F_2 = \Sigma_m (0, 1, 2, 6)$

$\Rightarrow \quad F_0 = \bar{A}\,\bar{B}\,\bar{C} + \bar{A}\,\bar{B}\,C + A\bar{B}\,\bar{C} + AB\bar{C}$

$= \bar{A}\,\bar{B} + A\bar{C}$

$F_1 = \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C} + AB\bar{C} + ABC$

$= \bar{A}B\bar{C} + BC + AB\bar{C} + AB\bar{C}$

$= B\bar{C} + BC + AB\bar{C}$

$= B + AB\bar{C}$

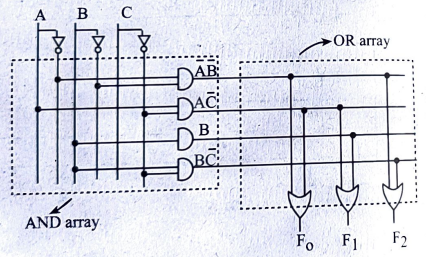
$= B + A\bar{C}$

$F_2 = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C}$

$= \bar{A}\bar{B} + B\bar{C}$

PLA table

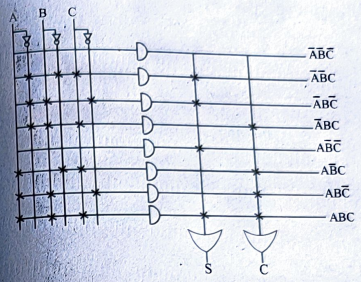
Product	Inputs			Outputs		
	A	B	C	F_0	F_1	F_2
$\bar{B}$	0	0	–	1	0	1
$A\bar{C}$	1	–	0	1	1	0
B	–	1	–	0	1	0
$B\bar{C}$	–	1	0	0	0	1



EXAMPLE **Implement the full adder using PLA.**

⇒ S = A'B'C + A'BC' + ABC' + ABC

C = AB + AC + BC



6.9 Binary Multiplication and Division

Multiplication is done with addition instructions and division with subtraction instructions. Therefore, an adder-subtractor is all that is needed for addition, subtraction, multiplication, and division.

For example, multiplication is equivalent to repeated addition.

$5 \times 4 = ?$

The first number is called *multiplicand* and the second number is called *multiplier*. Multiplying 5 by 4 is the same as adding 5 four times:

$5 \times 4 = 5 + 5 + 5 + 5$

6.9.1 Binary Multiplication

Four basic rules are:

a) $0 \times 0 = 0$

b) $0 \times 1 = 0$

c) $1 \times 0 = 0$

d) $1 \times 1 = 1$

E.g.,

111	110
× 100	× 101
000	110
000	000
111	110
11100	11110

6.9.2 Binary Division

The basic rules are:

a. $0 \div 1 = 0$ b. $1 \div 1 = 1$

```
10 ) 11010 ( 1101
     -10
      10
     -10
      010
      -10
       ×
```

```
11 ) 101001 ( 1101.01
     -11
      100
     -11
      101
      -11
      100
      -11
      100
      -11
        1
```

6.10 2-Bit Multiplier

		A_1	A_0
		B_1	B_0
		A_1B_0	A_0B_0
	A_1B_1	A_0B_1	×
C_2	S_2	S_1	S_o

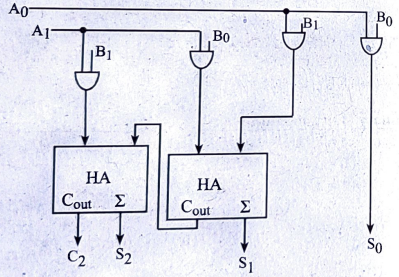


Fig.: Circuit diagram of a 2-bit multiplier

6.11 4-Bit Multiplier

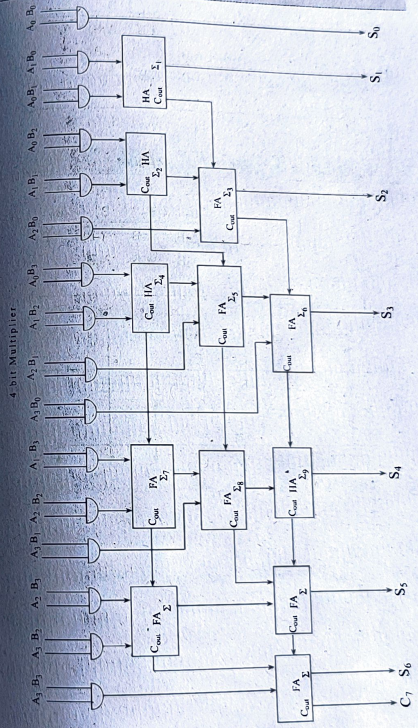


Fig.: Circuit diagram of a 4-bit multiplier

				A_3	A_2	A_1	A_0
			×	B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	
		A_3B_2	A_2B_2	A_1B_2	A_0B_2		
	A_3B_3	A_2B_3	A_1B_3	A_0B_3			
C_7	S_6	S_5	S_4	S_3	S_2	S_1	S_0

6.12 Multiplication using Binary Parallel Adder

				A_3	A_2	A_1	A_0
				B_3	B_2	B_1	B_0
				A_3B_0	A_2B_0	A_1B_0	A_0B_0
			A_3B_1	A_2B_1	A_1B_1	A_0B_1	
			C_2	C_1	C_0		
		C_3	S_3	S_2	S_1	S_0	
		A_3B_2	A_2B_2	A_1B_2	A_0B_2		
		C_6	C_5	C_4			
	C_7	S_7	S_6	S_5	S_4		
	A_3B_3	A_2B_3	A_1B_3	A_0B_3			
	C_{10}	C_9	C_3				
C_{11}	S_{11}	S_{10}	S_9	S_8			
M_7	M_6	M_5	M_4	M_3	M_2	M_1	M_0

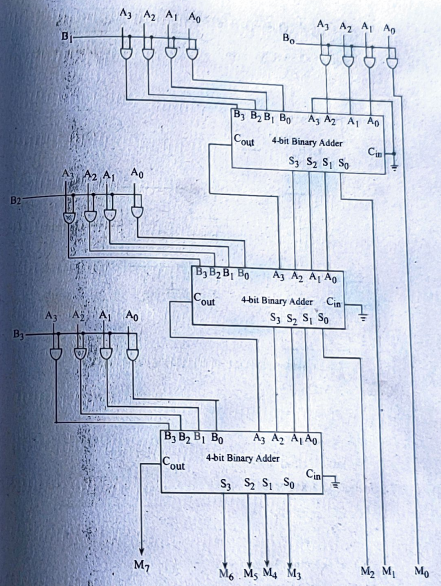


Fig.: 4-bit multiplier using binary parallel adder

SOLUTION TO IMPORTANT AND EXAM QUESTIONS

1. Construct 8×1 MUX, using 4×1 MUX and explain with truth table. [2021 Fall]

Solution:

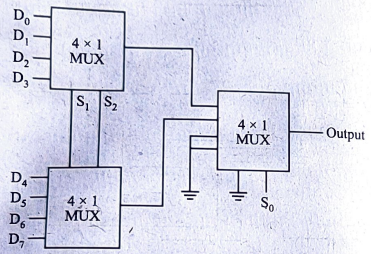


Fig.: Construction of 8×1 MUX using 4×1 MUX.

Given 4×1 MUX

No. of inputs = 4

No. of output = 1

No. of selection lines = 2

To design: 8×1 MUX

No. of inputs = 8

No. of outputs = 1

No. of selection lines = 3 (S_2, S_1, S_0)

Here, we use two 4×1 MUX to get 8 inputs and outputs of each 4×1 MUX are given to another 4×1 MUX. In this 4×1 MUX, two inputs and one selection line is grounded such that this MUX acts as 2×1 MUX.

The truth table of 8×1 MUX is as follows:

Select data inputs			Output
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

2. Implement the following three Boolean functions with a PLA:

$F_1 = \Sigma(0, 1, 3)$, $F_2 = \Sigma(1, 3, 4, 7)$, $F_3 = \Sigma(0, 2, 5, 6)$ [2021 Fall]

Solution:

Let, input variables are A, B, C.

For $F_1 = \Sigma(0, 1, 3)$

Using K-map,

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

$\therefore \quad F_1 = \bar{A}\bar{B} + \bar{A}C$

For $F_2 = \Sigma(1, 3, 4, 7)$

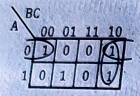
Using K-map,

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

$\therefore \quad F_2 = (\bar{A}C + B\bar{C} + A\bar{B}\bar{C})$

For $F_3 = \Sigma(0, 2, 5, 6)$

Using K-map,



$\therefore \quad F_3 = \bar{A}\,\bar{C} + B\bar{C} + A\bar{B}C$

PLA table

Product terms	Inputs			Outputs		
	A	B	C	F_1	F_2	F_3
$\bar{A}\,\bar{B}$	0	0	–	1	–	–
$\bar{A}C$	0	–	1	1	1	–
$B\bar{C}$	0	1	1	–	1	–
$A\bar{B}\,\bar{C}$	1	0	0	–	1	–
$\bar{A}\,\bar{C}$	0	–	0	–	–	1
$B\bar{C}$	–	1	0	–	–	1
$A\bar{B}C$	1	0	1	–	–	1

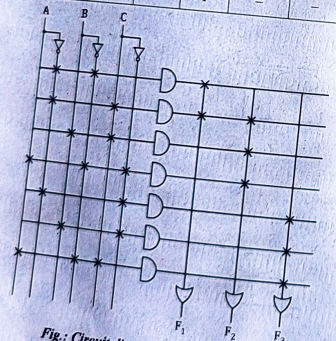


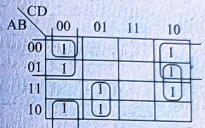
Fig.: Circuit diagram of required PLA.

3. **Simplify $F = \Sigma\,(0, 2, 4, 6, 8, 9, 13, 14)$ by using K–map. Implement the given circuit using decoder (use only the block diagram of decoder).**

Solution:

$F(A, B, C, D) = \Sigma(0, 2, 4, 6, 8, 9, 13, 14)$

Using K-map,



In SOP form,

$F(A, B, C, D) = A'D' + B'C'D' + AC'D + BCD'$

Now, designing the given functional system by using decoder.

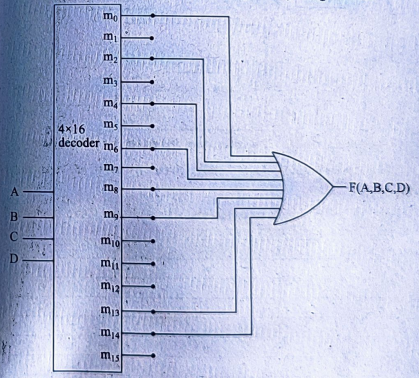


Fig.: 4×16 decoder circuit

4. **Construct a 5 × 32 decoder using 3 to 8 decoders and standard logic gates if necessary.**

Solution:

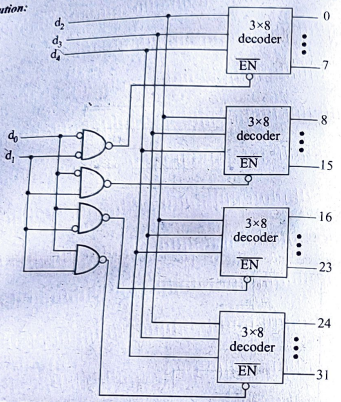


Fig.: Realization of 5×32 decoder using 3×8 decoders and logic gates

5. **Implement outputs of a full adder circuit by using 8:1 multiplexer.**

Solution:

For full adder circuit, let us suppose that A, B and C are inputs and S and C are sum and carry respectively.

A	B	C	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum $[S(x, y, z)] = \Sigma(1, 2, 4, 7)$

Carry $[C(x, y, z)] = \Sigma(3, 5, 6, 7)$

Now, implementing by using 8:1 multiplexer.

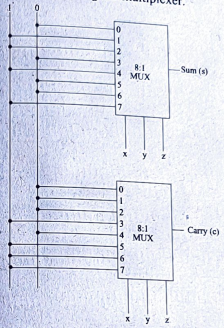
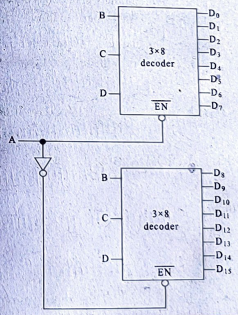


Fig.: Implementation of a full adder using 8:1 multiplexer

6. **Construct 4×16 decoder using 3×8 decoder.**

Solution:



A	B	C	D	Y
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	0	D_{12}
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

When A = 0, decoder 1 is enabled and decoder 2 is disabled. Hence the output from D_0 to D_7 is high as per the input combination of BCD.

When A = 1, decoder 2 is enabled and decoder 1 is disabled. Hence the output from D_8 to D_{15} is high as per the input combination of ABCD.

7. Design a 32:1 MUX using only 8:1 MUX. Use block diagrams.

Solution:

We have to design 32: 1 MUX using 8 : 1 MUX.

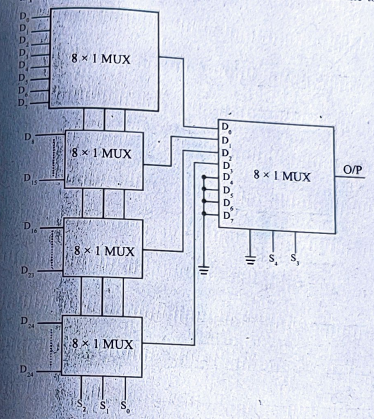
Given: 8 × 1 MUX

No. of selection lines is 3.

Design: 32 × 1 MUX

No. of selection lines is 5

Thus to get 32 no. of input lines, we need 4 number of 8 × 1 MUX. Let S_4 S_3 S_2 S_1 S_0 be the order of the input lines then S_0 S_1 acts as selector of MUX and $S_2S_3S_4$ acts as input to the selection line to the input.



8. Design a combinational circuit using PLD device as PLA (4×8×4) that is used to implement full subtract or function in which difference is represented as Di and borrow as Br. [Spring 2019]

Solution:

Truth table of full subtractor

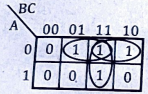
Inputs			Outputs	
A	B	C	Difference (Di)	Borrow (Br)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Using K-map,

For Di,

$\therefore$ $Di = \bar{A}\bar{B}C + A\bar{B}\bar{C} + \bar{A}B\bar{C} + ABC$

For Br,



$Br = \bar{A}C + \bar{A}B + BC$

PLA table

Product terms	Inputs			Outputs	
	A	B	C	Di	Br
$\bar{A}\bar{B}C$	0	0	1	1	–
$A\bar{B}\bar{C}$	1	0	0	1	–
$\bar{A}B\bar{C}$	0	1	0	1	–
ABC	1	1	1	1	–
$\bar{A}C$	0	–	1	–	1
$\bar{A}B$	0	1	–	–	1
BC	–	1	1	–	1

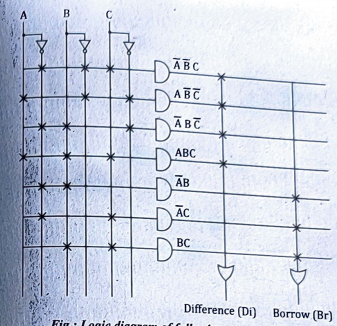


Fig.: Logic diagram of full subtractor using PLA.

9. Implement the following with appropriate MUX.

i. $F(A, B, C) = \Sigma(1, 4, 5, 6)$

ii. $F(A, B, C, D) = \Sigma(0, 1, 3, 8, 9, 15)$ [Spring 2019]

Solution:

i. $F(A, B, C) = \Sigma(1, 4, 5, 6)$

No. of variables = 3(A, B, C)

= no. of selection lines for MUX (S_2, S_1, S_0)

$\therefore$ The required MUX = $2^3 \times 1 = 8 \times 1$ MUX

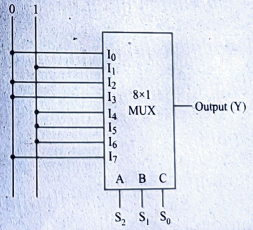


Fig.: Implementation of given function using 8×1 MUX.

ii. $F(A, B, C, D) = \Sigma(0, 1, 3, 8, 9, 15)$

No. of variables = 4(A, B, C, D)

= No. of selection lines (S_3, S_2, S_1, S_0).

∴ Required MUX = $2^4 \times 1 = 16 \times 1$ MUX

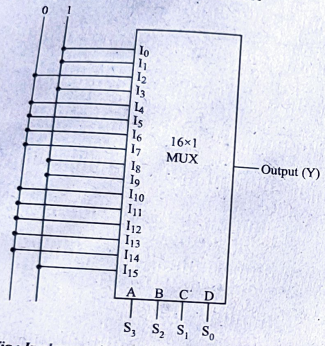


Fig.: Implementation of given function using 16×1 MUX

10. Design a combinational circuit using PLD device as PLA (4×8×4) that is used to implement full adder function in which sum is represented as Si and carry as Ci. [Fall 2019]

Solution:

The truth table of full adder is

Inputs			Outputs	
A	**B**	**C**	**Sum (Si)**	**Carry (Ci)**
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Using K-map

For Si,

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

$Si = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

For Ci,

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

$Ci = AB + AC + BC$

PLA Table

Product terms	Inputs			Outputs	
	A	**B**	**C**	**Sum (Si)**	**Carry (Ci)**
$\bar{A}\bar{B}C$	0	0	1	1	–
$\bar{A}B\bar{C}$	0	1	0	1	–
$A\bar{B}\bar{C}$	1	0	0	1	–
ABC	1	1	1	1	–
AB	1	1	–	–	1
AC	1	–	1	–	1
BC	–	1	1	–	1

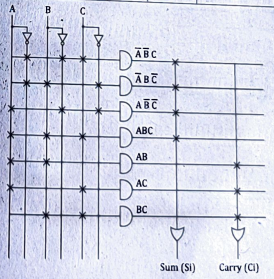


Fig.: Logic diagram of full adder using PLA

11. Implement the following Boolean function using 16:1 multiplexer:
$F(A, B, C, D, E) = \Sigma_m(2,4,5,7,10,14,15,16,17,25,26,30,31)$

[Spring 2018]

Solution:

$F(A,B,C,D,E) = \Sigma_m(2,4,5,7,10,14,15,16,17,25,26,30,31)$

No. of input variable = 5(A, B, C, D, E)

Selection lines = 4(B, C, D, E)

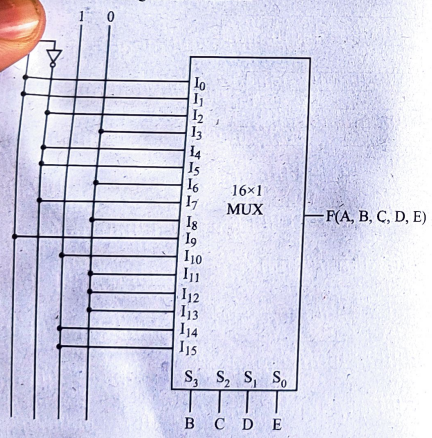
Input line = 1(A)

Size of MUX = 16 × 1

Implementation table for input 'A' to be connected,

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
A'	0	1	(2)	3	(4)	(5)	6	(7)	8	9	(10)	11	12	13	(14)	(15)
A	(16)	(17)	18	19	20	21	22	23	24	(25)	(26)	27	28	29	(30)	(31)
	A	A	A'	0	A'	A'	0	A'	0	A	1	0	0	0	1	1

Now, circuit diagram is as follows.



12. A combinational circuit is defined by the function,
$F_1(A, B, C) = \Sigma(3, 5, 6, 7)$
$F_2(A, B, C) = \Sigma(0, 2, 4, 7)$
Implement by using PLA.

[Fall 2018]

Solution:

$F_1(A, B, C) = \Sigma(3, 5, 6, 7)$

$F_2(A, B, C) = \Sigma(0, 2, 4, 7)$

Using K-map,

For F_1,

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

$\therefore \quad F_1 = BC + AC + AB$

For F_2,

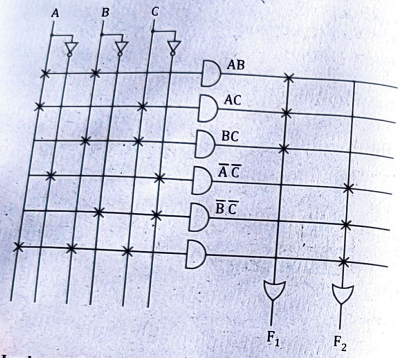
A \ BC	00	01	11	10
0	1	0	0	1
1	1	0	1	0

$\therefore \quad F_2 = \overline{B}\overline{C} + \overline{A}\overline{C} + ABC$

PLA table

Product Terms	Inputs A	Inputs B	Inputs C	Functions F_1	Functions F_2
BC	–	1	1	1	–
AC	1	–	1	1	–
AB	1	1	–	1	–
$\overline{B}\overline{C}$	–	0	0	–	1
$\overline{A}\overline{C}$	0	–	0	–	1
ABC	1	1	1	–	1

Now, the required PLA diagrams is



Implement the following:

i. **F(A, B, C) = Σ(1, 3, 5, 6) (using MUX)**

ii. **F_1 = Σ(0, 2, 5), F_2 = Σ(3, 4, 7), F_3 = Σ(6, 7) (using ROM)**

[Spring 2016]

Solution:

i. **F(A, B, C) = Σ(1, 3, 5, 6) (using MUX)**

Let, selection lines = 2(B, C)

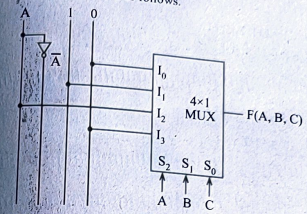
Input line = 1(A)

∴ Required MUX = 4 × 1 MUX

Implementation table for input 'A' to be connected:

	I_0	I_1	I_2	I_3
A'	0	①	2	③
A	4	⑤	⑥	7
	0	1	A	A'

Now, circuit diagram is as follows.



ii. **F_1 = Σ(0, 2, 5), F_2 = Σ(3, 4, 7), F_3 = Σ(6, 7) (using ROM)**

Let input variables = A, B, C

Since i/p = 3, we use 3 × 8 decoder

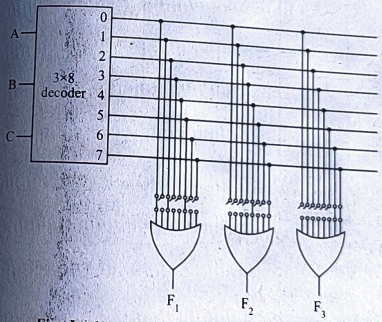


Fig.: Implementation of given function using ROM

14. **Implement the following using MUX.**

i. **F(A, B, C) = Σ(1, 3, 5, 6)**

ii. **F(A, B, C, D) = Σ(0, 1, 3, 4, 8, 9, 15)** [Fall 2016]

Solution:

i. **F(A, B, C) = Σ(1, 3, 5, 6)**

Selection lines = 2(B, C)

Input line = A

∴ Required MUX = $2^2 \times 1 = 4 \times 1$ MUX

Implementation table:

	I_0	I_1	I_2	I_3
$\bar{A}$	0	①	2	③
A	4	⑤	⑥	7
	0	1	A	$\bar{A}$

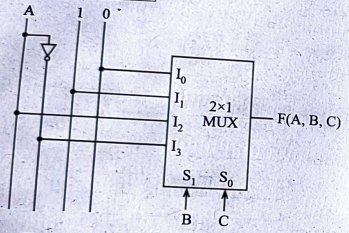


Fig.: Implementation of given function using MUX.

iii. **F(A, B, C, D) = Σ(0, 1, 3, 4, 8, 9, 15)**

Selection lines = 3(B, C, D)

Input line = 1(A)

∴ Required MUX = $2^3 \times 1 = 8 \times 1$ MUX

Implementation table:

	I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7
$\bar{A}$	⓪	①	2	③	④	5	6	7
A	⑧	⑨	10	11	12	13	14	⑮
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

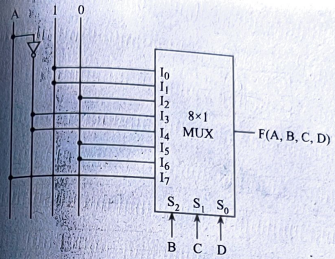


Fig.: Implementation of given function using MUX.

15. **Implement the following three Boolean function with a PLA.**

$F_1 = \Sigma(0, 1, 2, 4)$

$F_2 = \Sigma(0, 5, 6, 7)$

$F_3 = \Sigma(0, 3, 5, 7)$ [Fall 2015]

Solution:

$F_1 = \Sigma(0, 1, 2, 4)$, $F_2 = \Sigma(0, 5, 6, 7)$, $F_3 = \Sigma(0, 3, 5, 7)$

Using K-map,

For F_1,

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

∴ $F_1 = \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\bar{C}$

For F_2,

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

∴ $F_2 = \bar{A}\bar{B}\bar{C} + AC + AB$

For F_3,

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

$\therefore \quad F_3 = \overline{A}\overline{B}\overline{C} + AC + BC$

PLA table

Product Terms	Inputs A	B	C	Outputs F_1	F_2	F_3
$\overline{A}\overline{B}\overline{C}$	0	0	0	–	1	1
AC	1	–	1	–	1	1
BC	–	1	1	–	–	1
AB	1	1	–	–	1	–
$\overline{A}\overline{C}$	0	–	0	1	–	–
$\overline{A}\overline{B}$	0	0	–	1	–	–
$\overline{B}\overline{C}$	–	0	0	1	–	–

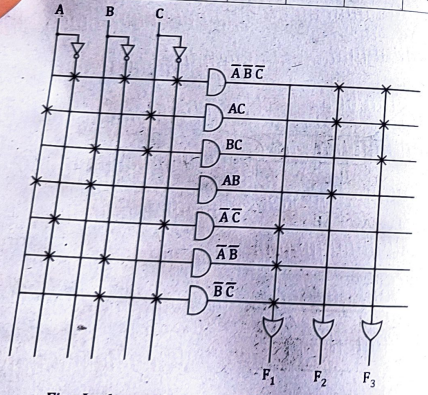


Fig.: Implementation of given functions using PLA

16. Derive a PLA program table for the combinational circuit that squares 3-bit number. Minimize the number of product terms. [Spring 2013]

Solution:

Truth table

A	B	C	Decimal Equivalent	Decimal Square no.	Output a	b	c	d	e	f
0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	1
0	1	0	2	4	0	0	0	1	0	0
0	1	1	3	9	0	0	1	0	0	1
1	0	0	4	16	0	1	0	0	0	0
1	0	1	5	25	0	1	1	0	0	1
1	1	0	6	36	1	0	0	1	0	0
1	1	1	7	49	1	1	0	0	0	1

Using K-map,

For a,

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

$\therefore \quad a = A\overline{B}$ and $a' = \overline{A} + B$

For b,

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

$\therefore \quad b = A\overline{B} + AC$ and $b' = \overline{A} + B\overline{C}$

For c,

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

$\therefore$ $c = \bar{A}B\bar{C} + A\bar{B}\bar{C}$ and $c' = \bar{A}\bar{B} + B\bar{C} + AB + B\bar{C}$

For d,

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

$\therefore$ $d = B\bar{C}$ and $d' = \bar{B} + C$

For e,

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

$\therefore$ $e = 0$

For f,

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

$\therefore$ $f = C$, $f' = \bar{C}$

We choose a, b, c, d, e, f

PLA table

Product terms	Inputs A	B	C	Outputs a	B	c	d	e	f
AB	1	1	–	1	–	–	–	–	–
$A\bar{B}$	1	0	–	–	1	–	–	–	–
AC	1	–	C	–	1	–	–	–	–
$\bar{A}BC$	0	1	1	–	–	1	–	–	–
$A\bar{B}\bar{C}$	1	0	1	–	–	1	–	–	–
$B\bar{C}$	–	1	0	–	–	–	1	–	–
C	–	–	1	–	–	–	–	–	1

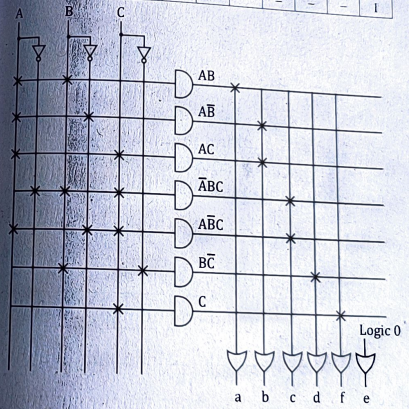


Fig.: Circuit diagram of PLA

17. **Implement the given four Boolean function using 8 × 4 PLA**

$A(x, y, z) = \Sigma(1, 2, 4, 6)$

$B(x, y, z) = \Sigma(0, 1, 6, 7)$

$C(x, y, z) = \Sigma(2, 6)$

$D(x, y, z) = \Sigma(1, 2, 3, 6, 7)$

[Spring 2013]

Solution:

Input variables = x, y, z

Functions = A, B, C, D,

Now, $A(x, y, z) = \Sigma(1, 2, 4, 6)$

$= \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$

$B(x, y, z) = \Sigma(0, 1, 6, 7)$

$= \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + xy\bar{z} + xyz$

$C(x, y, z) = \Sigma(2, 6)$

$= \bar{x}y\bar{z} + xy\bar{z}$

$D(x, y, z) = \Sigma(1, 2, 3, 5, 7)$

$= \bar{x}\bar{y}z = \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z + xyz$

Now, PLA table is as follows:

Product term	Inputs			Output functions			
	x	y	z	A	B	C	D
$\bar{x}\bar{y}\bar{z}$ (0)	0	0	0	–	1	–	–
$\bar{x}\bar{y}z$ (1)	0	0	1	1	1	–	1
$\bar{x}y\bar{z}$ (2)	0	1	0	1	–	1	1
$\bar{x}yz$ (3)	0	1	1	–	–	–	1
$x\bar{y}\bar{z}$ (4)	1	0	0	1	–	–	–
$x\bar{y}z$ (5)	1	0	1	–	–	–	1
$xy\bar{z}$ (6)	1	1	0	1	1	1	–
xyz (7)	1	1	1	–	1	–	1

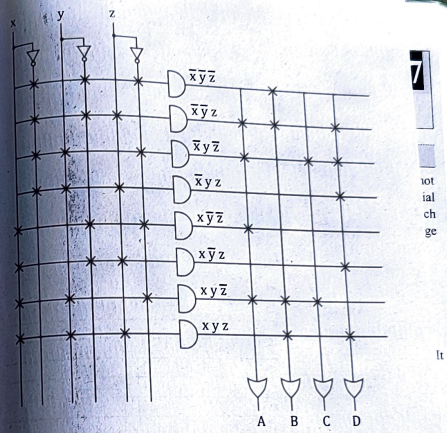


Fig.: Implementation of given function using PLA

18. **A combinational circuit is defined by the following three functions**

$F_1 = \bar{x}\bar{y} + xy\bar{z}$, $F_2 = \bar{x} + y$, $F_3 = xy + \bar{x}\bar{y}$

Design the circuit with a decoder & external gates. [Fall 2013]

Solution:

Firstly converting to standard form,

$F_1 = x'y' + xyz'$

$= x'y'(z + z') + xyz'$

$= x'y'z + x'y'z' + xyz'$

$= m_1 + m_0 + m_6$

$\therefore F_1 = \Sigma_m(0, 1, 6)$

$F_2 = \bar{x} + y = \bar{x} = (y + \bar{y})(z + \bar{z}) + y(x + \bar{x})(z + \bar{z})$

$= (\bar{x}y + \bar{x}\bar{y})(z + \bar{z}) + (xy + \bar{x})(z + \bar{z})$

$= (\bar{x}yz + \bar{x}y\bar{z} + x\bar{y}z + x\bar{y}\bar{z}) + (xyz + xy\bar{z} + \bar{x}yz + \bar{x}y\bar{z})$

$= \bar{x}yz + \bar{x}y\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + xyz + xy\bar{z}$

$= m_3 + m_2 + m_1 + m_0 + m_7 + m_6$

$\therefore \quad F_2 = \Sigma_m(0, 1, 2, 3, 6, 7)$

$F_3 \quad = xy + x\bar{y}$

$= xy(z + \bar{z}) + x\bar{y}(z + \bar{z})$

$= xyz + xy\bar{z} + x\bar{y}z + x\bar{y}\bar{z}$

$= m_7 + m_6 + m_1 + m_0$

$\therefore \quad F_3 = \Sigma_m(0, 1, 6, 7)$

Since, no. of input variables = 3

$\therefore$ Required decoder = 3 × 8 decoder

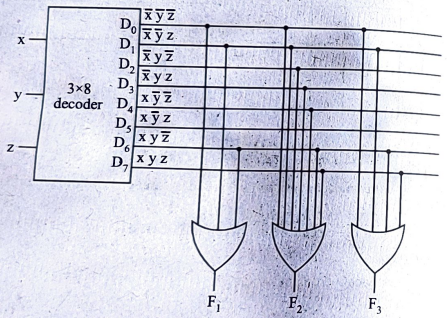


Fig.: Implementation of given functions using decoder and external gates

CHAPTER – 7

SEQUENTIAL LOGIC

7.1 Sequential Circuit

The logic circuits whose outputs at any instant of time depend not only on the present inputs but also on past outputs are called sequential circuits. A sequential circuit consists of a combinational circuit to which storage elements are connected to form a feedback path. The storage elements are devices capable of storing binary information.

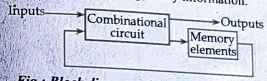


Fig.: Block diagram of a sequential circuit

Sequential circuit is slower in operation than combinational circuit. It may or may not contain clock input.

There are two types of sequential circuits:

i. **Synchronous sequential circuit**

It is a system whose behavior can be defined from the knowledge of its signals at discrete instant of time (i.e., by the clock signal parallelly driven by one clock signal).

ii. **Asynchronous sequential circuit**

It is a system whose behavior depends in the order in which its input signals change and can be affected at any instant of time (one's output is given to clock of another).

7.2 Event Driven Model and State Diagram

There are two distinct models by which asynchronous sequential logic circuit can be designed:

- Moore Model
- Mealy Model