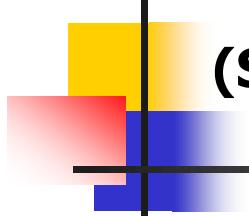


Digital Electronics

(Second Year B. Tech program in Computer Engineering)



Digital Electronics

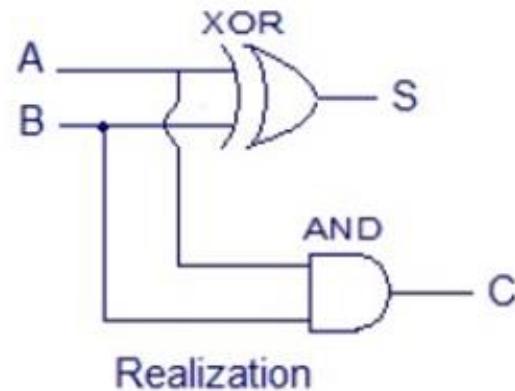
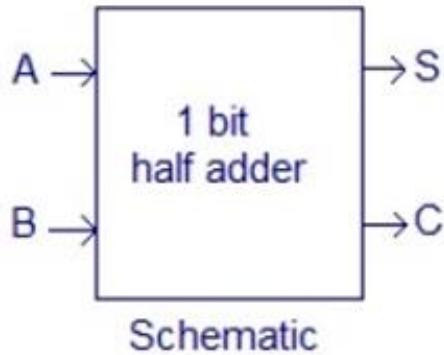
(Second Year B. Tech program in Computer Engineering)

Adder

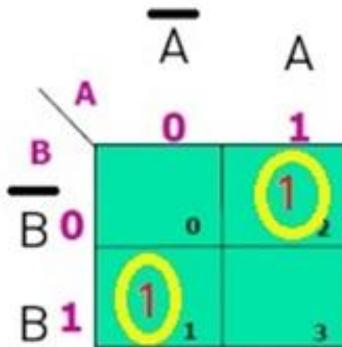
Half Adder :

Dec No	Inputs		Outputs	
	A	B	S	C
0	0	0	0	0
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1

Truth table

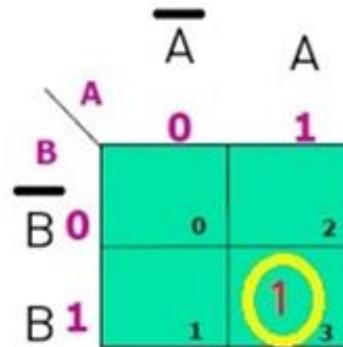


KMAP for Sum



$$\begin{aligned} \text{Sum} &= A'B + AB' \\ &= A \oplus B \end{aligned}$$

KMAP for Carry



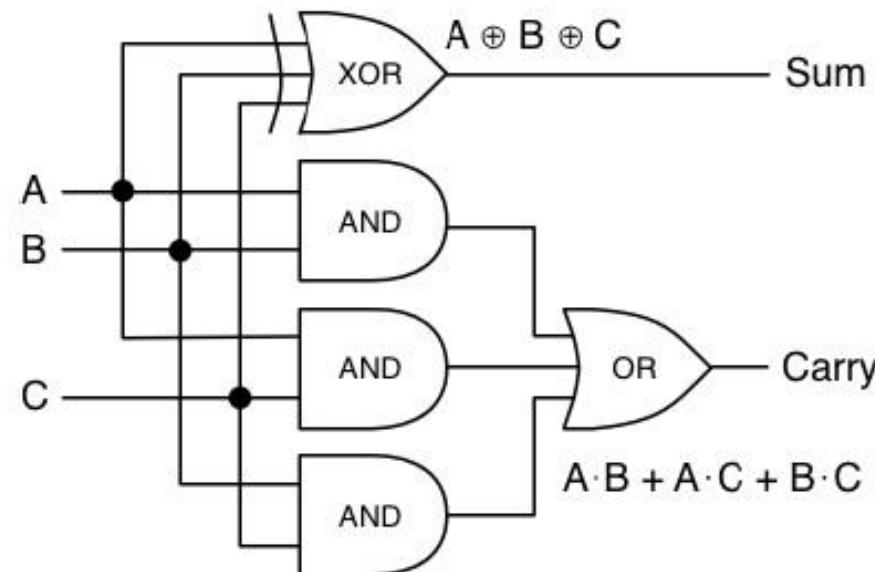
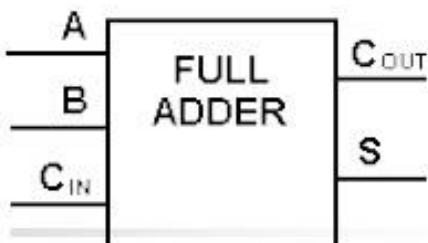
$$\text{Carry} = AB$$

Drawbacks:

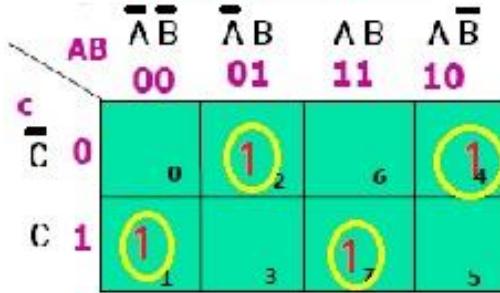
- 1. It is used for addition of single bit numbers only.
- 2. It doesn't consider a carry generated from previous additions.

Full Adder :

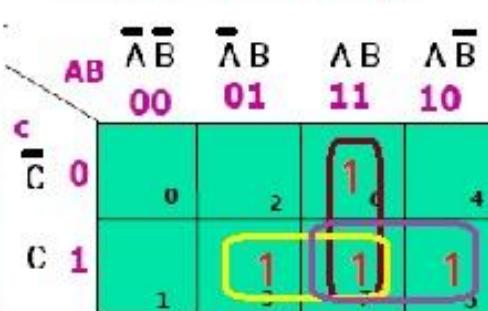
A	B	Carry In	Sum	Carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

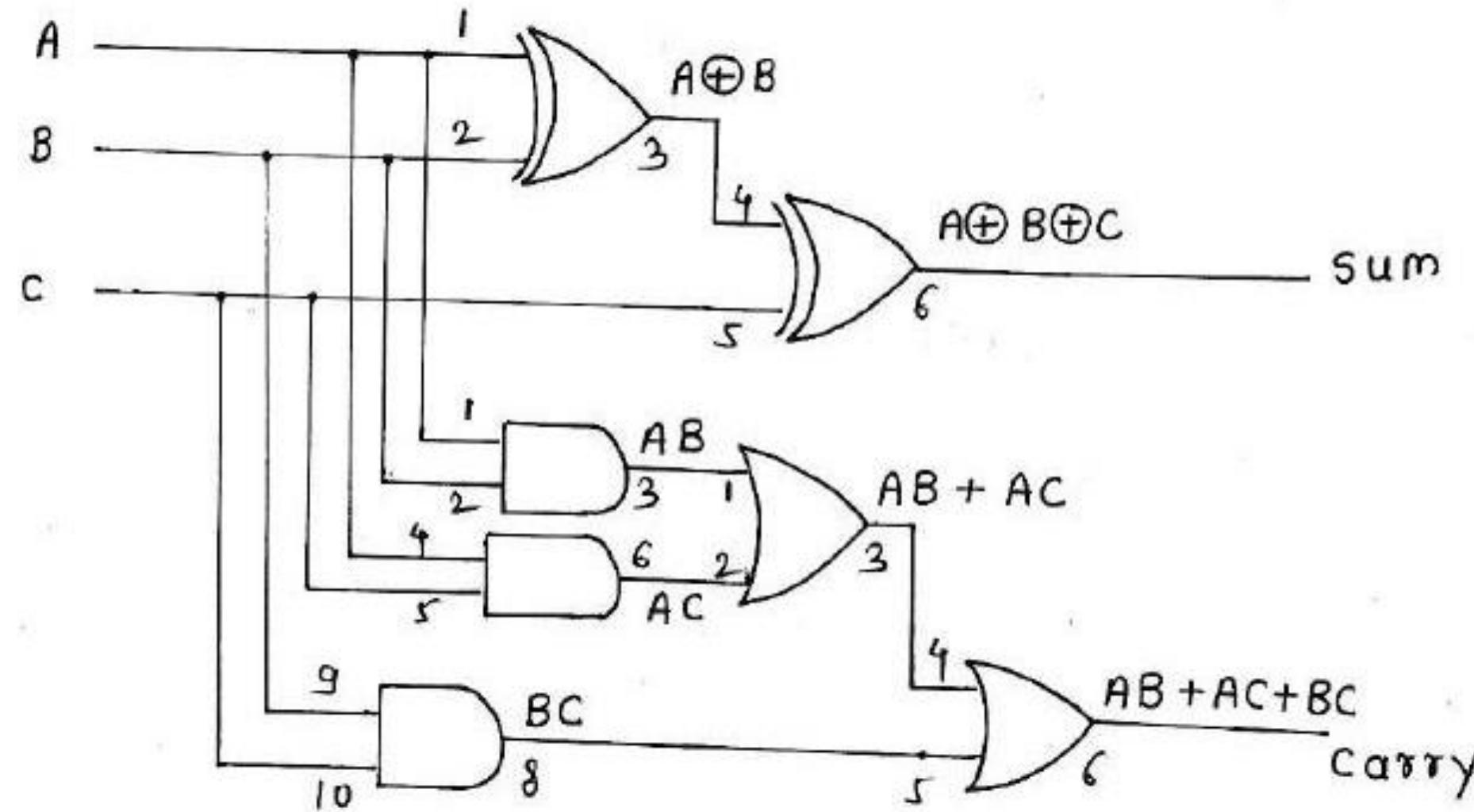


KMAP for Sum



KMAP for Carry

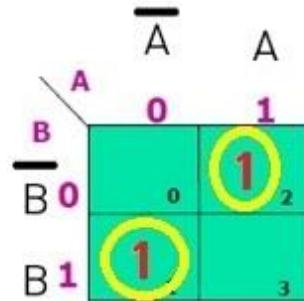




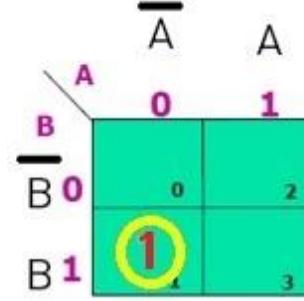
Half Subtractor :

A	B	D	B _r
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Kmap for Diff

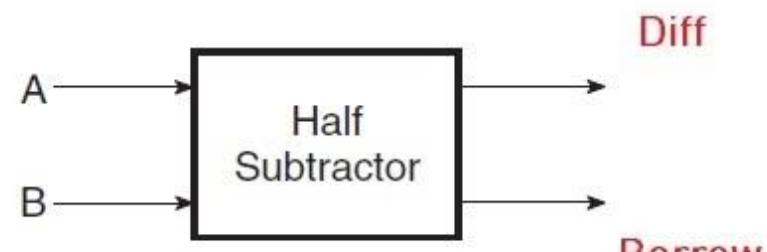
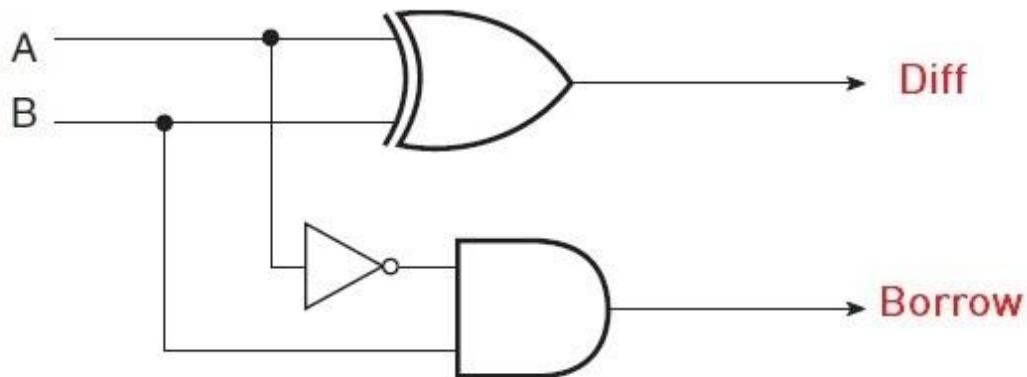


Kmap for Borrow



$$\begin{aligned} \text{Diff} &= A'B + AB' \\ &= A \oplus B \end{aligned}$$

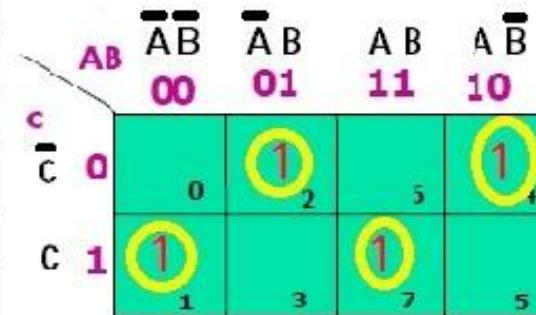
$$\text{Borrow} = A'B$$



Full Subtractor :

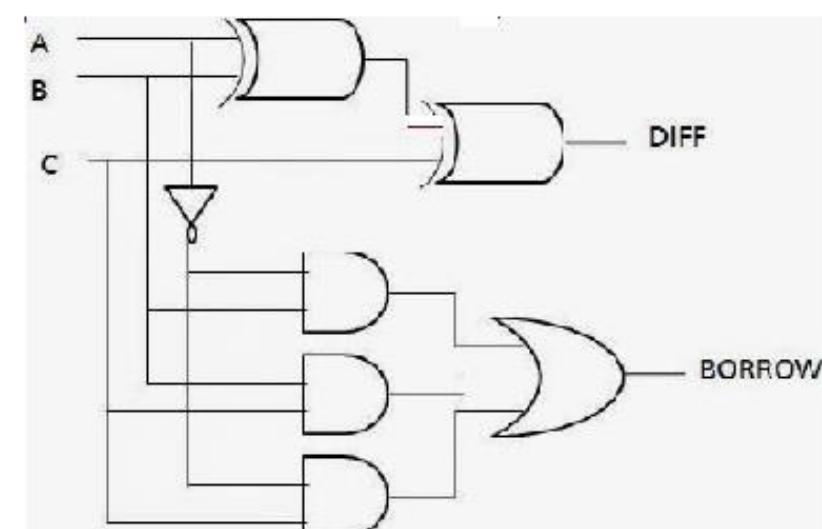
Dec No	Inputs			Outputs	
	A	B	C	Diff	Borrow
0	0	0	0	0	0
1	0	0	1	1	1
2	0	1	0	1	1
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	0
6	1	1	0	0	0
7	1	1	1	1	1

Kmap for Diff

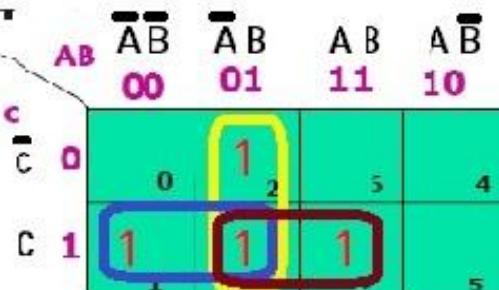


$$\text{Diff} = A'B'C + A'BC' + ABC + AB'C'$$

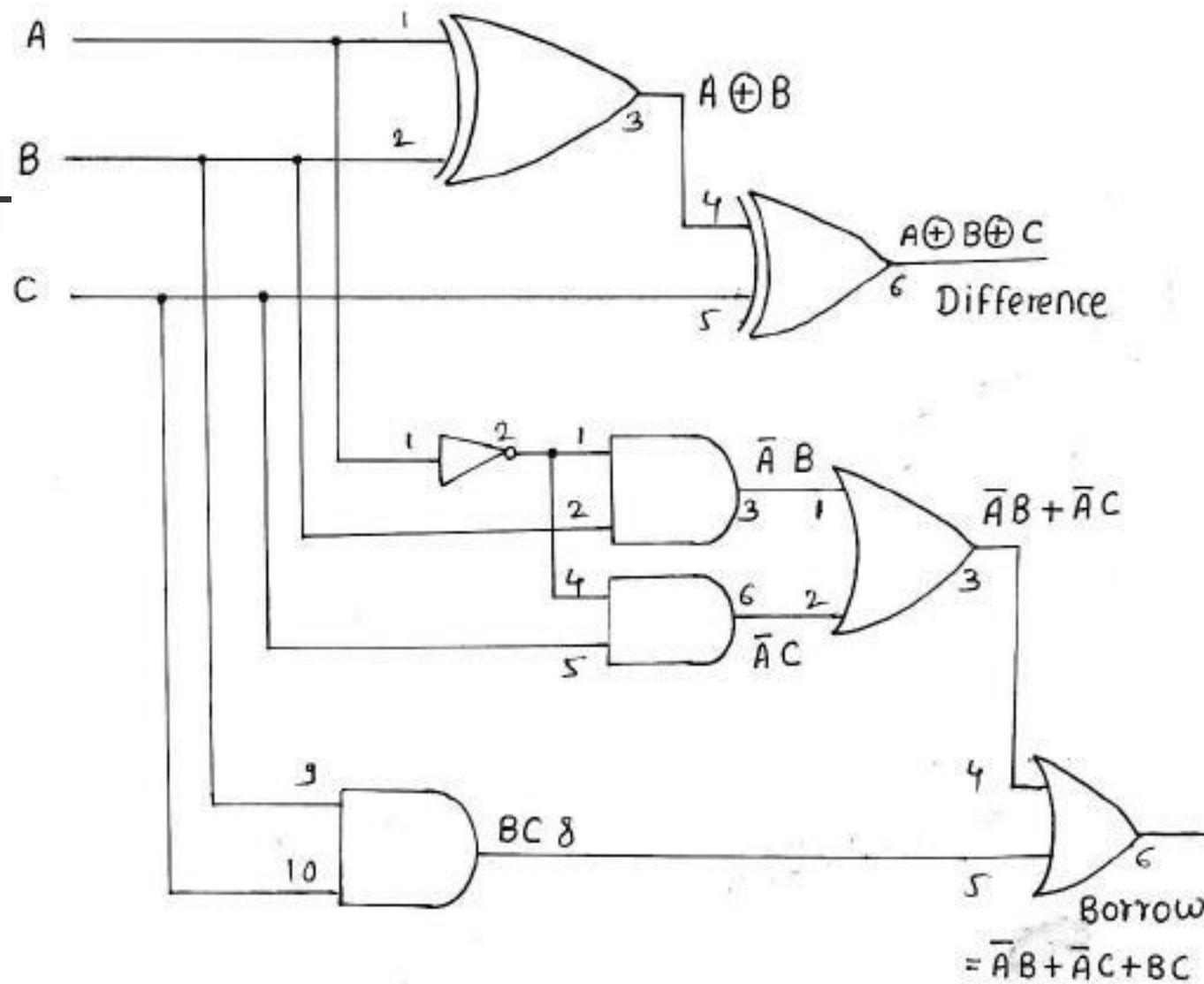
$$= A \oplus B \oplus C$$



Kmap for Borrow



$$\text{Borrow} = A'B + A'C + BC$$



Equations for Half Adder

$$\begin{aligned}\text{Sum} &= A'B + AB' \\ &= A \oplus B\end{aligned}$$

$$\text{Carry} = AB$$

Equations for Full Adder

$$\begin{aligned}\text{Sum} &= A'B'C + A'BC' + ABC + AB'C' \\ &= A \oplus B \oplus C\end{aligned}$$

$$\text{Carry} = AB + BC + AC$$

Equations for Half substractor

$$\begin{aligned}\text{Diff} &= A'B + AB' \\ &= A \ominus B\end{aligned}$$

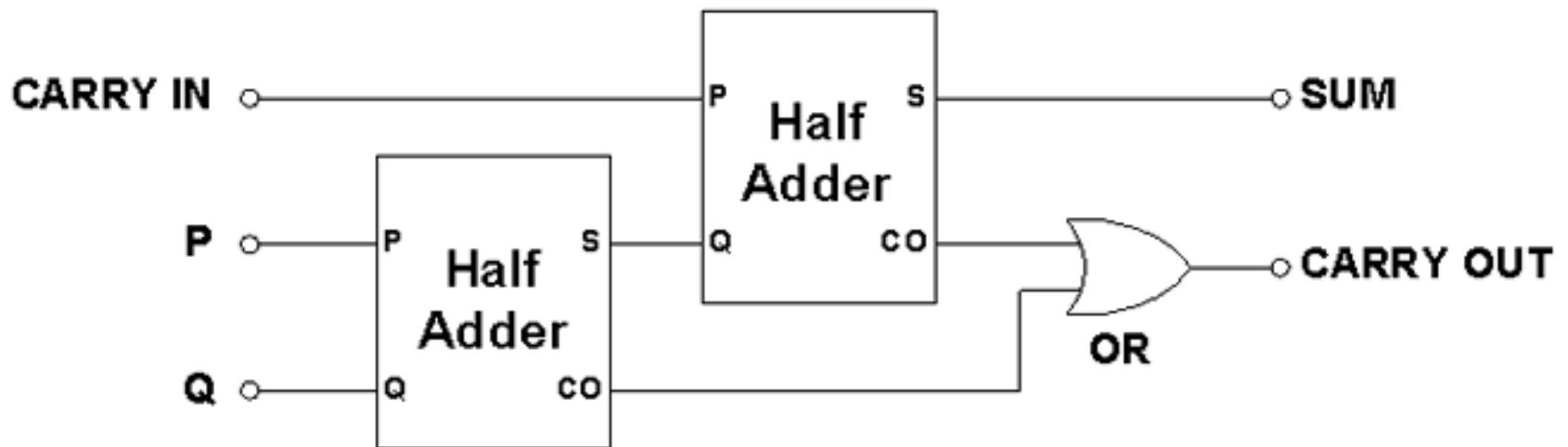
$$\text{Borrow} = A'B$$

Equations for Full Substrator

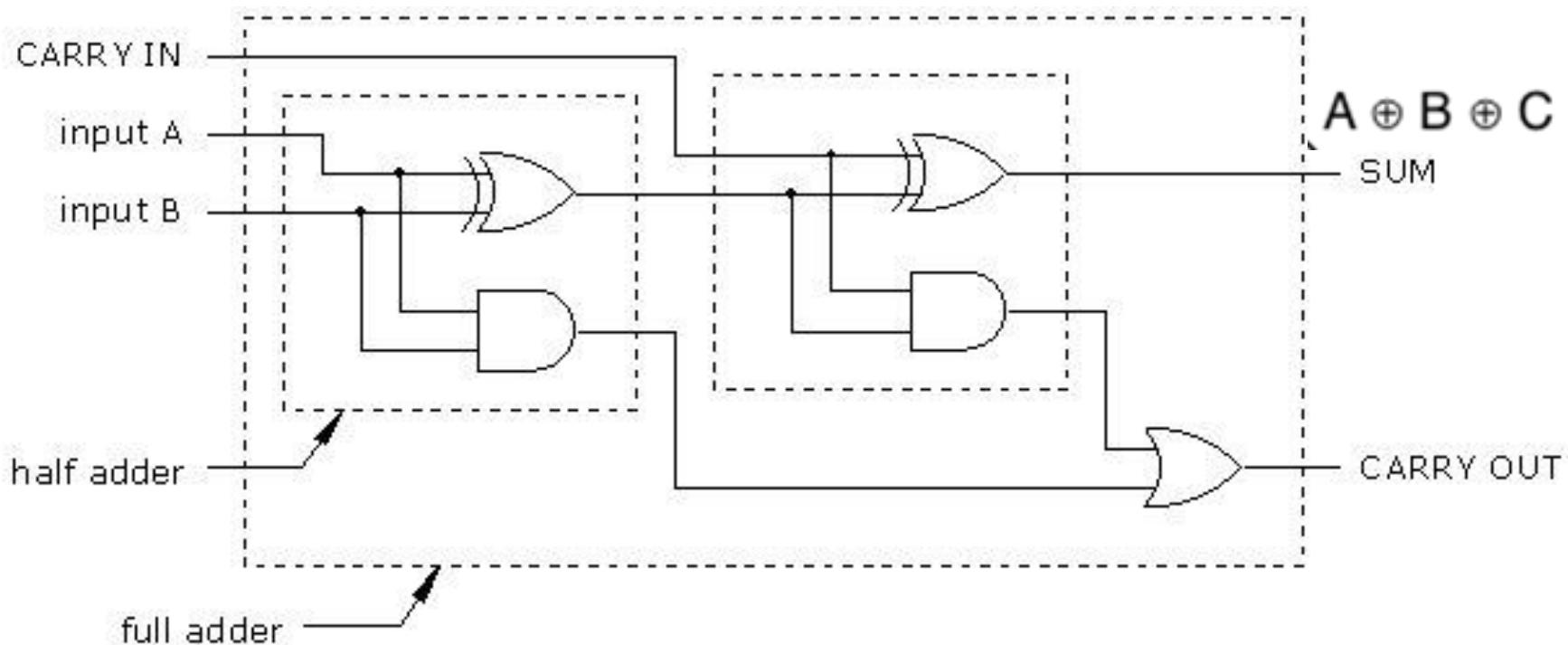
$$\begin{aligned}\text{Diff} &= A'B'C + A'BC' + ABC + AB'C' \\ &= A \oplus B \oplus C\end{aligned}$$

$$\text{Borrow} = A'B + A'C + BC$$

Carry Look Ahead Adder/ Fast adder/ Full Adder using two half adder & one OR gate :



Full Adder using two half adder & one or gate



$$\text{Sum} - A \oplus B \oplus C$$

$$\text{Carry} - A \oplus B \cdot C + A \cdot B$$

Full Subtractor using two half subtractor & one OR gate :

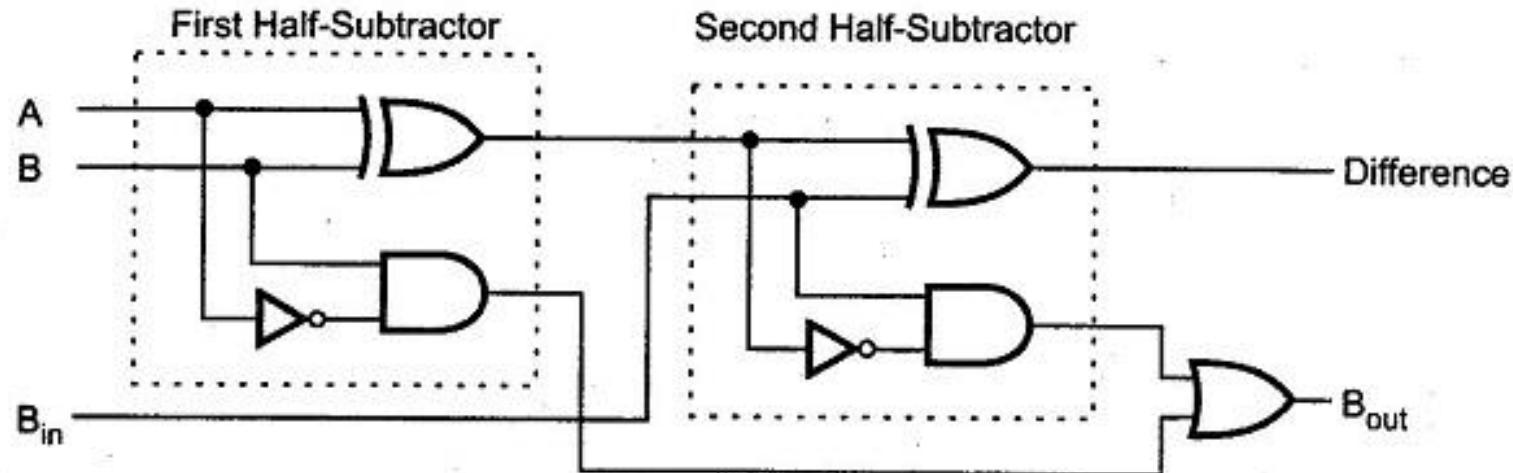
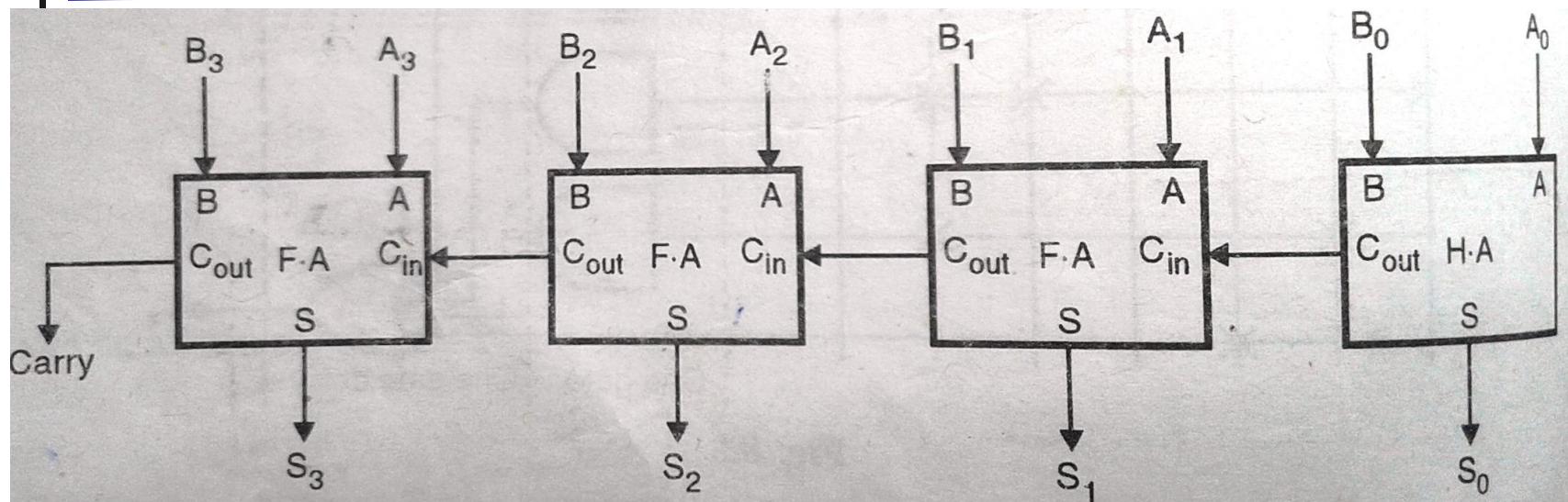
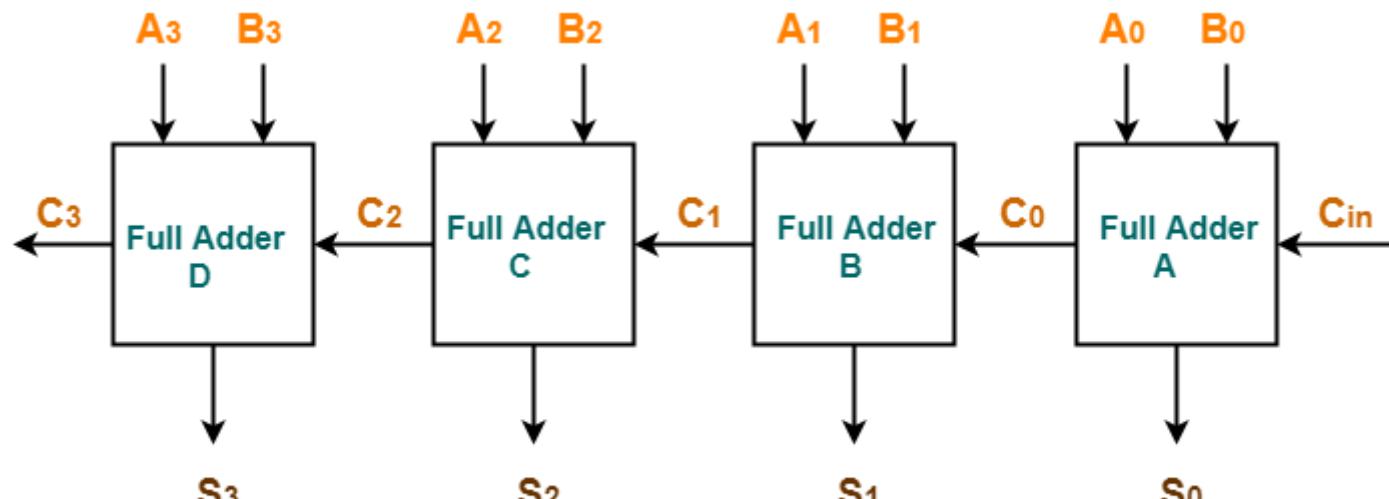


Fig. 3.24 Implementation of a full-subtractor with two half-subtractors and an OR gate

Full Adder using 3 full adder & 1 half adder

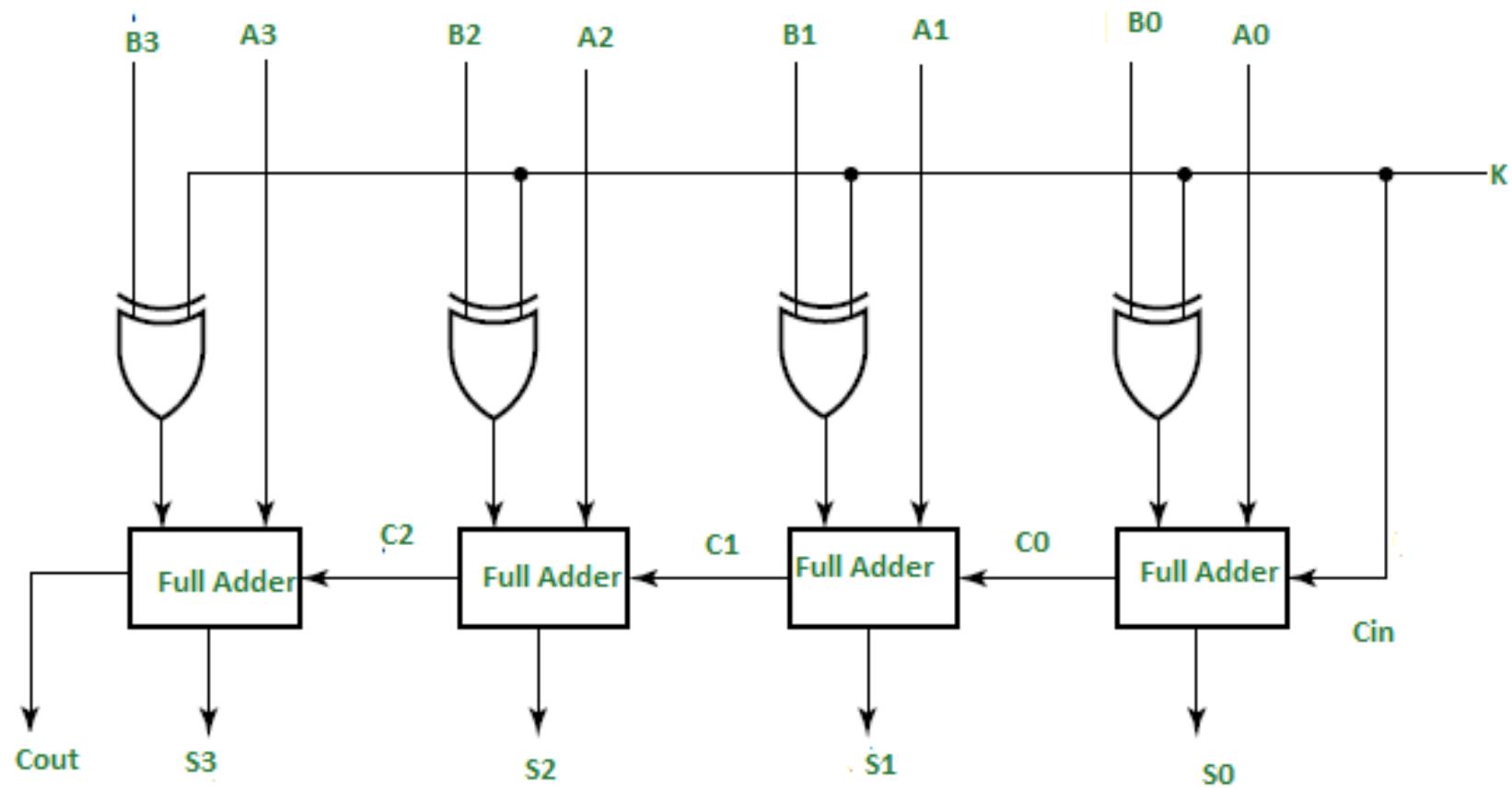


Full Adder using 4 full adder/ Ripple Carry Adder/ Parallel Adder



4-bit Ripple Carry Adder

4 bit Adder Subtractor



Carry =0, Adder and Carry =1, Subtractor

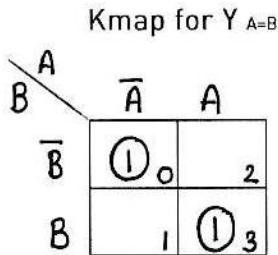
https://www.youtube.com/watch?v=9R_IPPAwht0

1 bit Comparator:

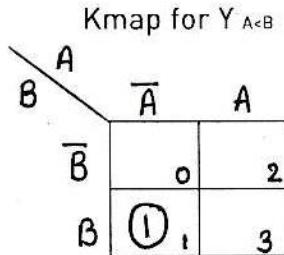
Truth Table:

Input		Output		
A	B	$Y_{A=B}$	$Y_{A>B}$	$Y_{A<B}$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

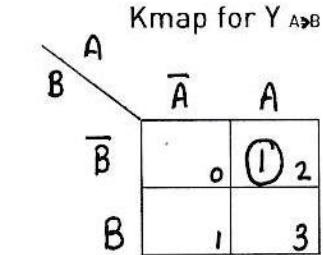
Kmap :



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

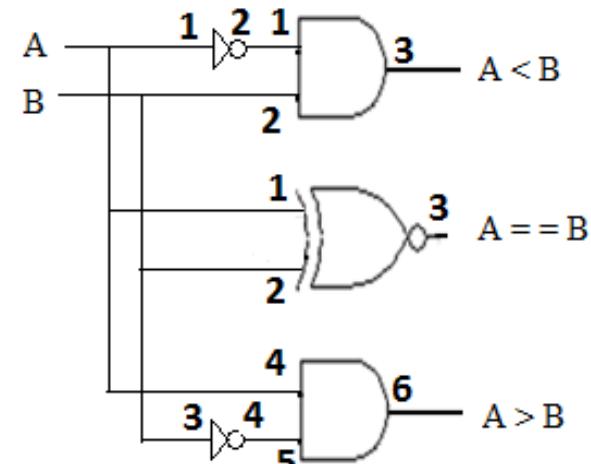


$$Y_{A < B} = \bar{A}B$$



$$Y_{A > B} = A\bar{B}$$

Circuit Diagram



1-bit magnitude comparator

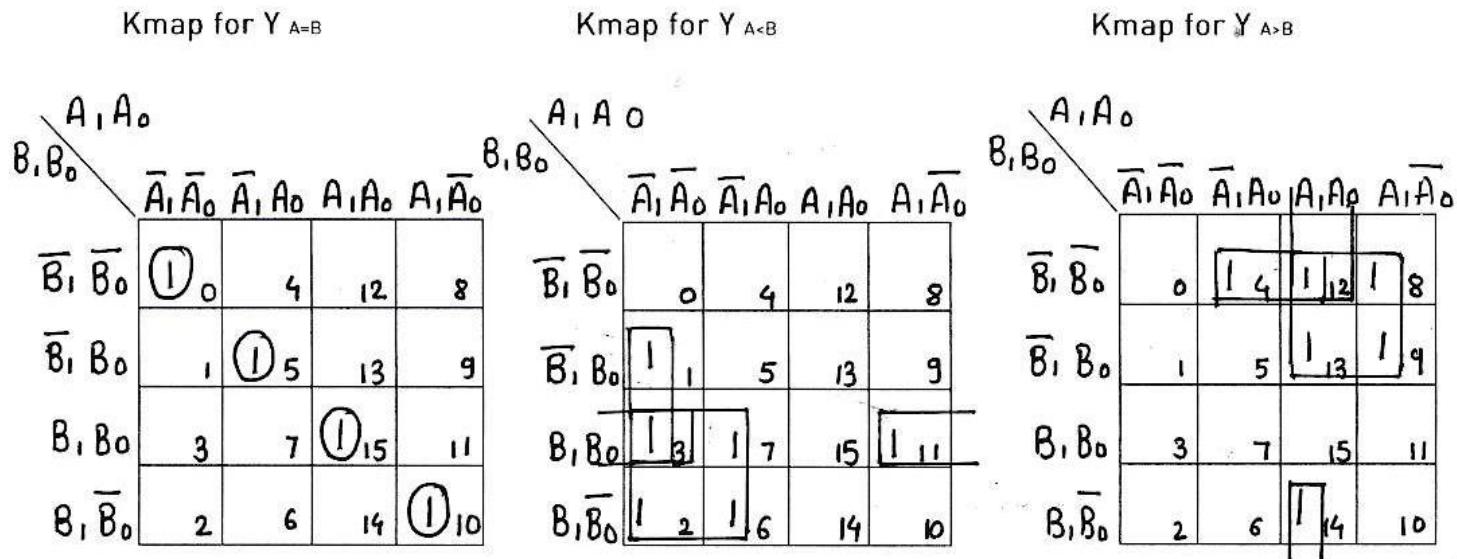
2 bit Comparator:

Truth Table:

Inputs				Outputs		
A ₁	A ₀	B ₁	B ₀	A>B	A=B	A<B
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

2 bit Comparator:

Kmap :



$$Y_{A < B} = \bar{A}_1 \cdot B_1 + \bar{A}_1 \bar{A}_0 B_0 + \bar{A}_0 B_0 B_1$$

$$Y_{A > B} = A_1 \bar{B}_1 + A_0 \bar{B}_0 \bar{B}_1 + A_1 \bar{B}_1 \bar{B}_0$$

$$Y_{A=B} = \bar{A}_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + \bar{A}_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \bar{A}_0 B_1 \bar{B}_0$$

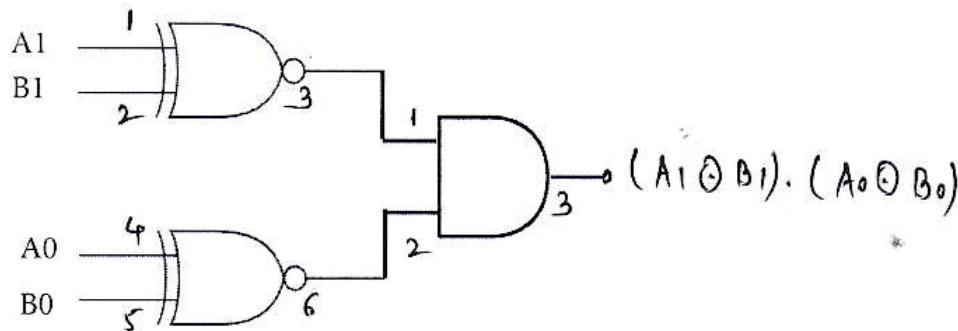
- $\bar{A}_1 \bar{B}_1 (\bar{A}_0 \bar{B}_0 + A_0 B_0) + A_1 B_1 (A_0 B_0 + \bar{A}_0 \bar{B}_0)$

- $\bar{A}_1 \bar{B}_1 (A_0 \odot B_0) + A_1 B_1 (A_0 \odot B_0)$

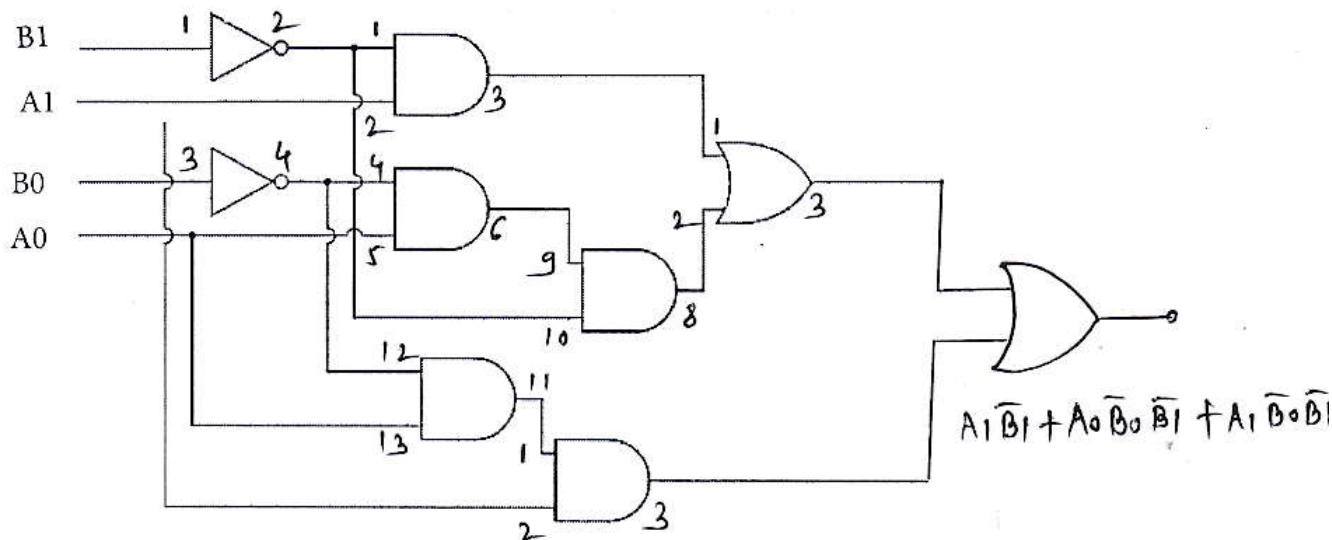
- $(A_1 \oplus B_1) \cdot (A_0 \odot B_0)$

2 bit Comparator:

Circuit Diagram :



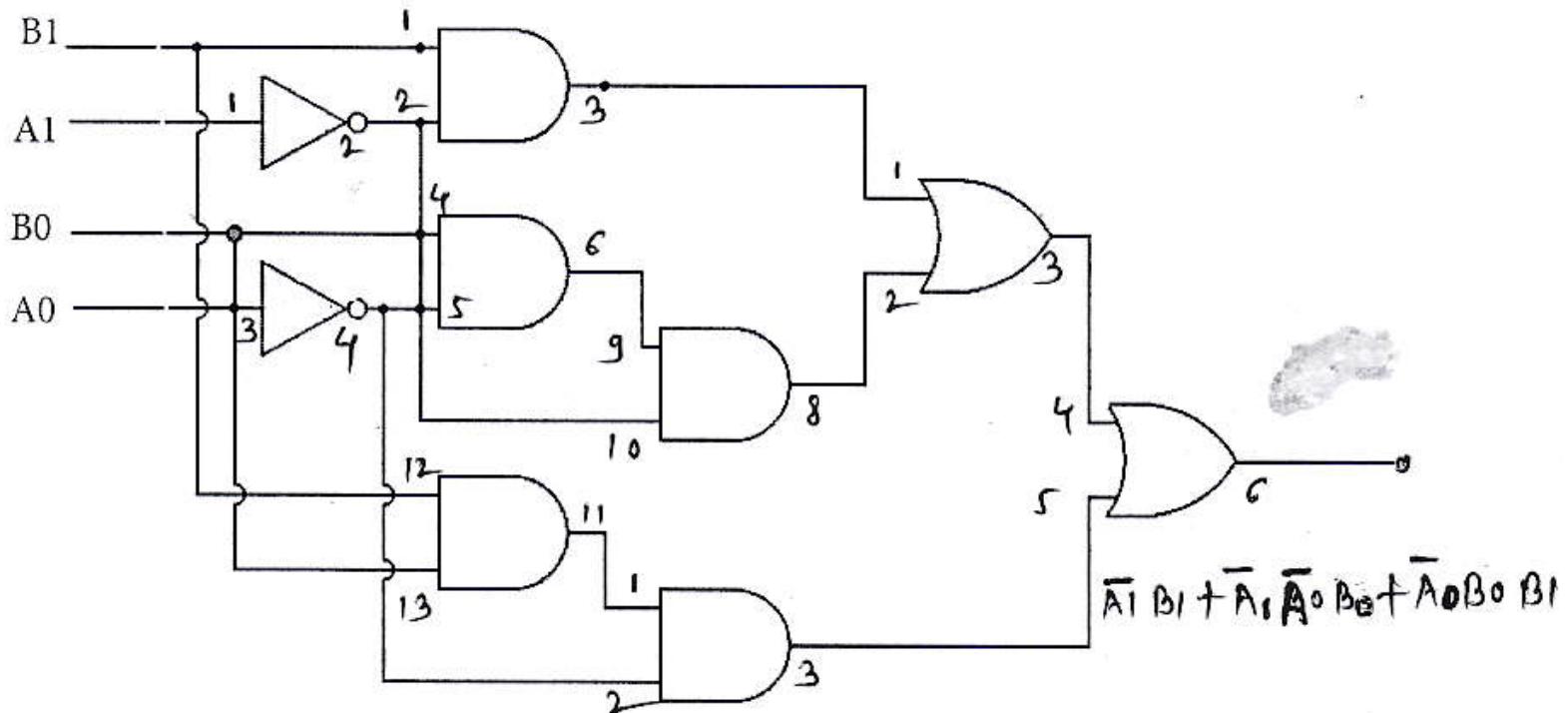
Circuit diagram for $Y_{A=B}$



Circuit diagram for $Y_{A>B}$

2 bit Comparator:

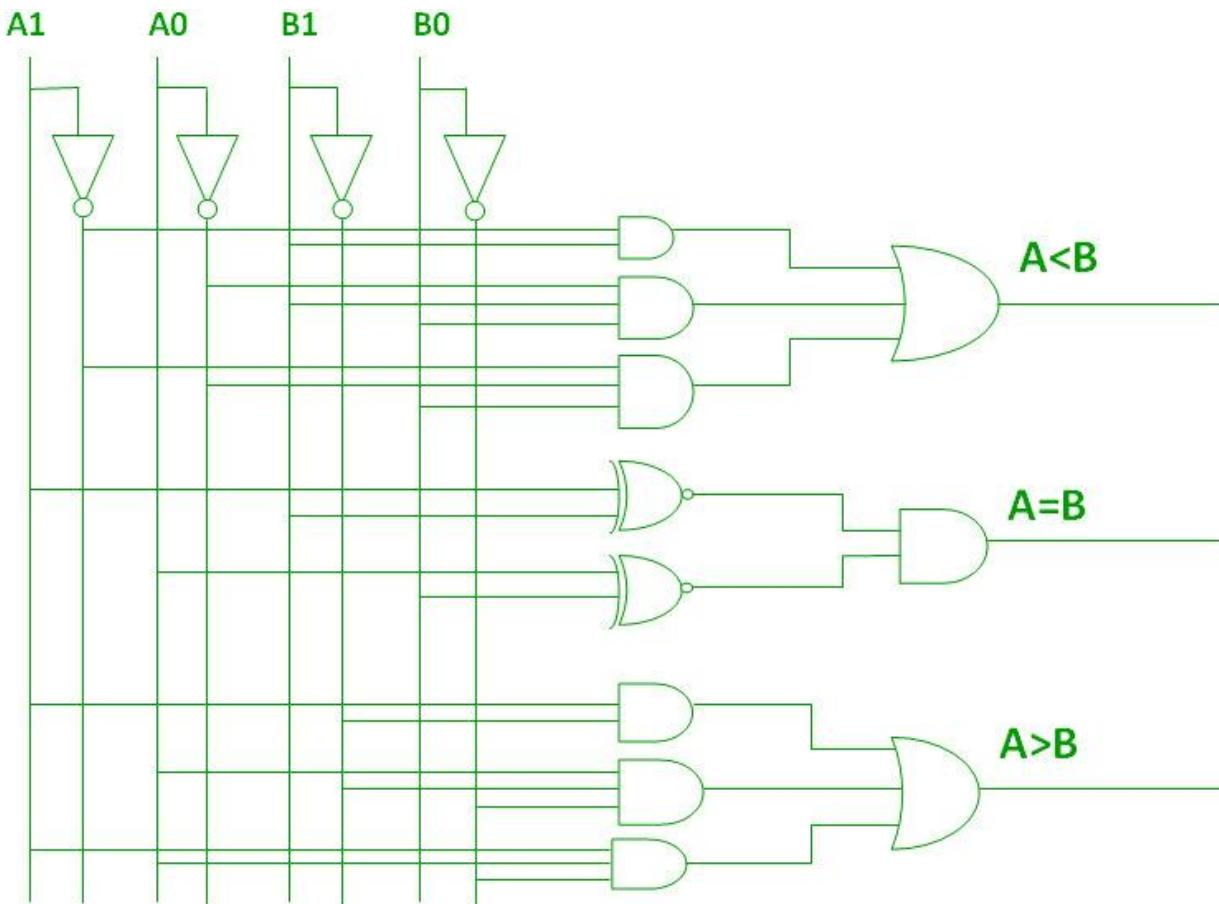
Circuit Diagram :



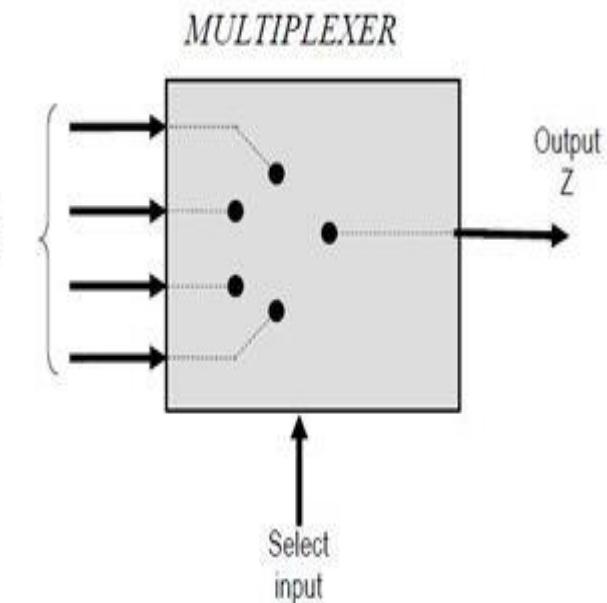
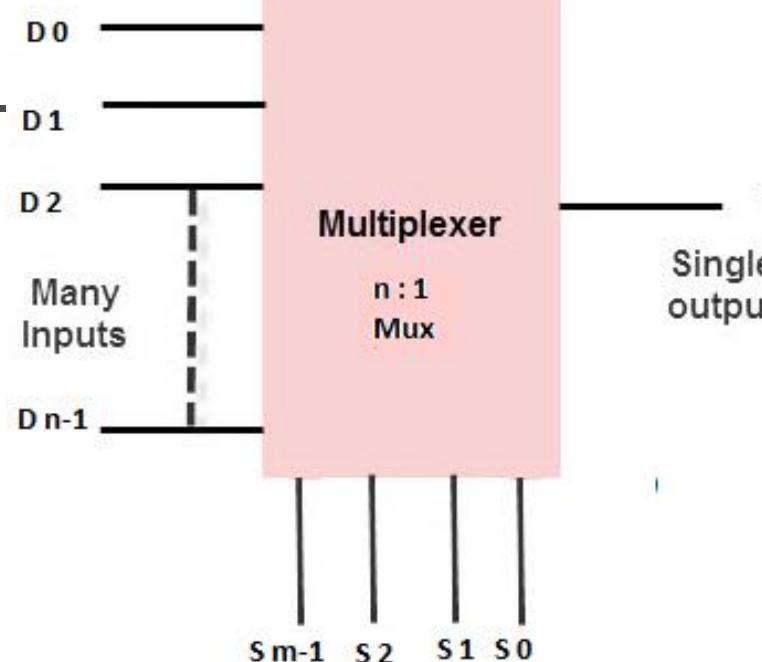
Circuit diagram for Y_{EQ}

2 bit Comparator:

Circuit Diagram :



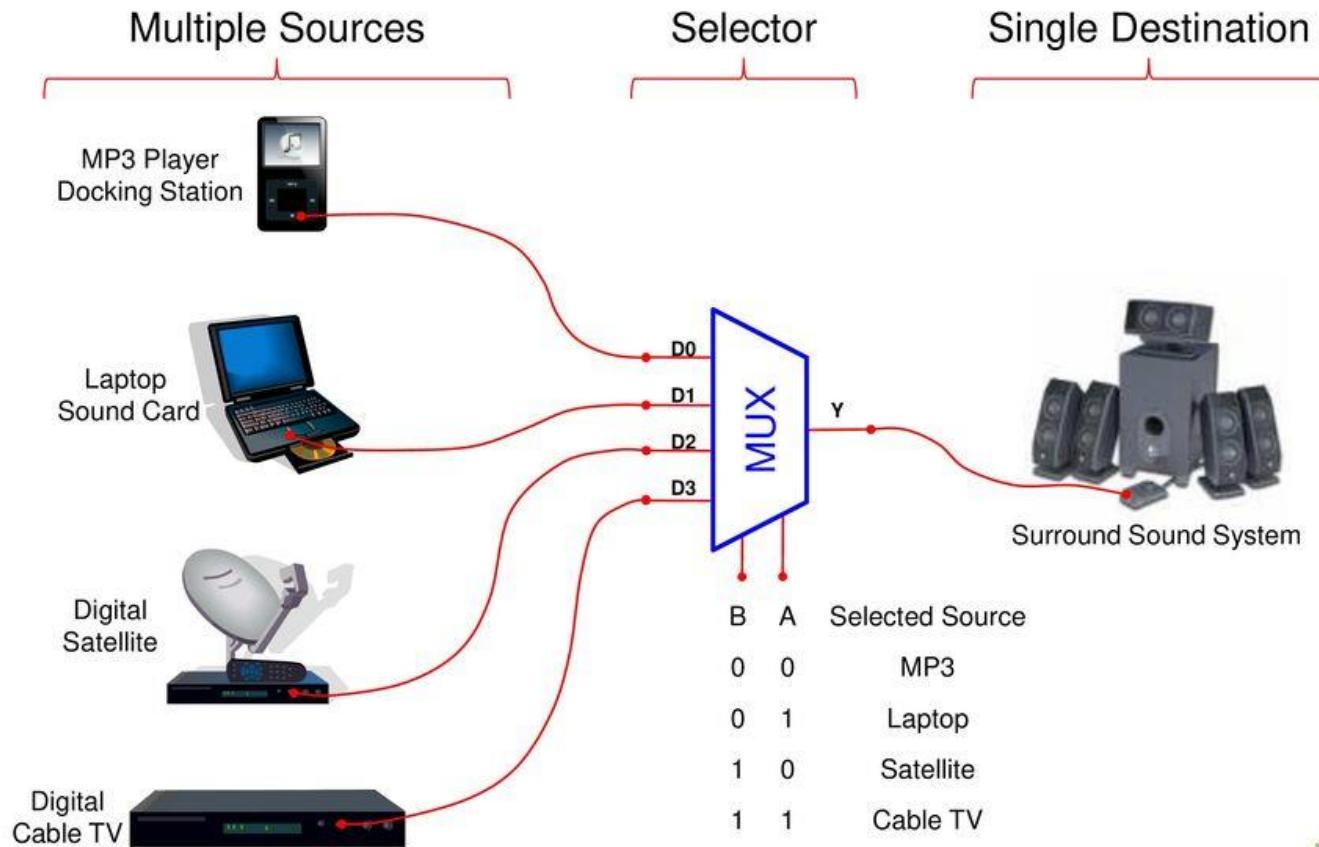
Multiplexer (Data Selector) :

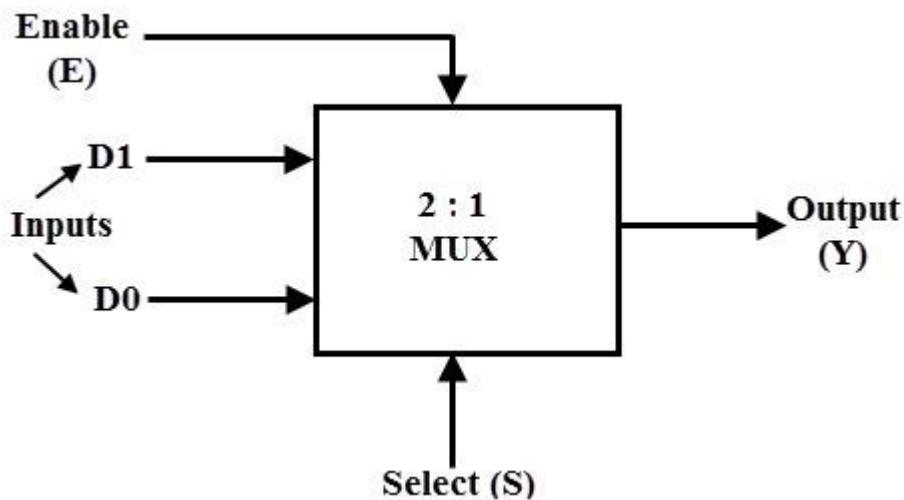
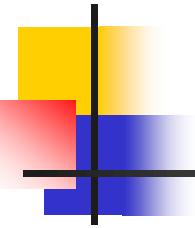


Logic circuit that selects binary information from one of its input line & direct it into a single output line.

Applications: Communication System, Computer Memory, Telephone Network , Transmission from the Computer System of a Satellite etc.

Typical Application of a MUX



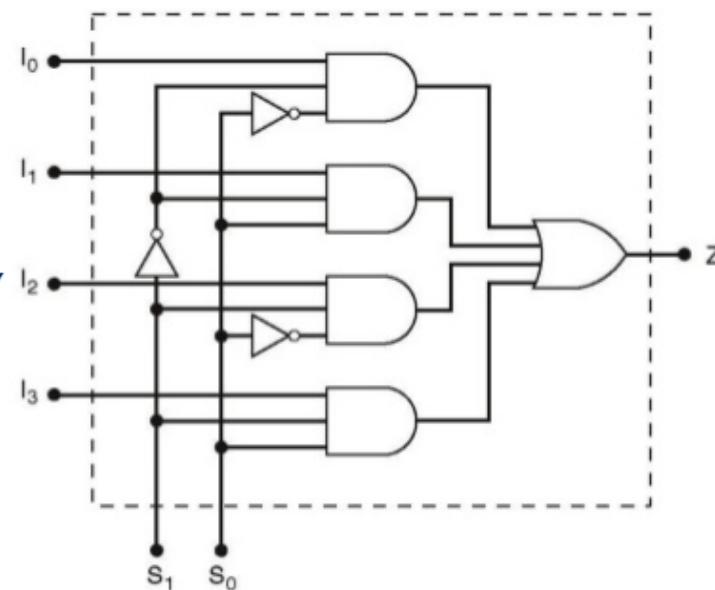
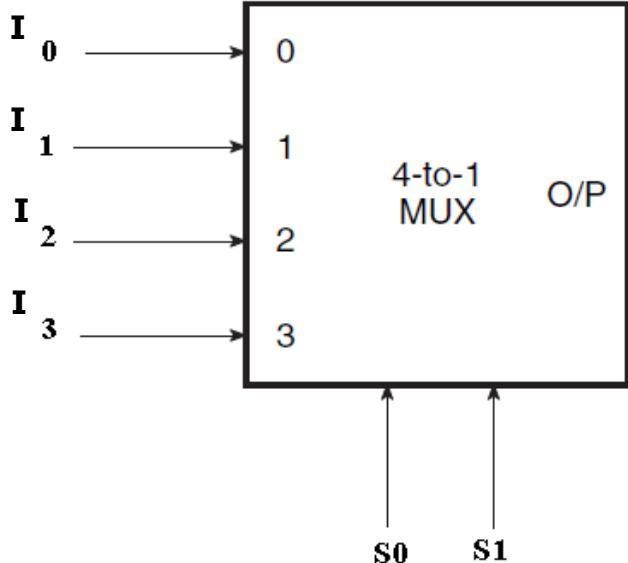


I/P S ₀	O/P Y
0	D ₀
1	D ₁

$$Y = \overline{S_0} D_0 + S_0 D_1$$

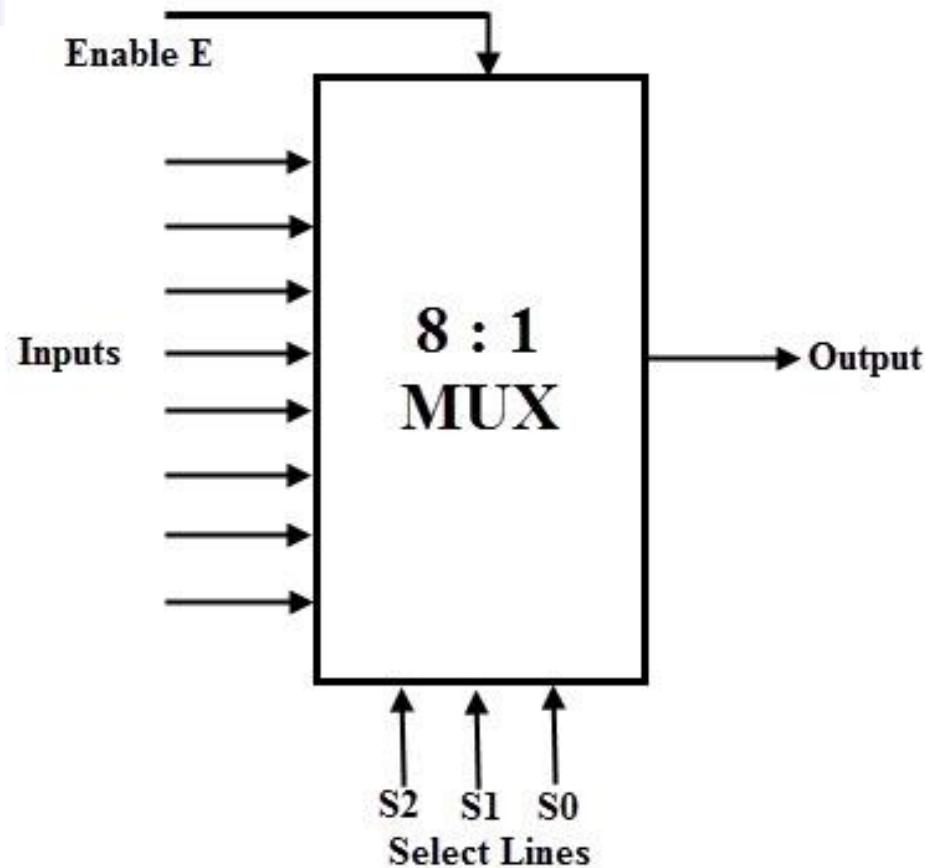
Multiplexer (Data Selector) :

4 : 1 Multiplexer

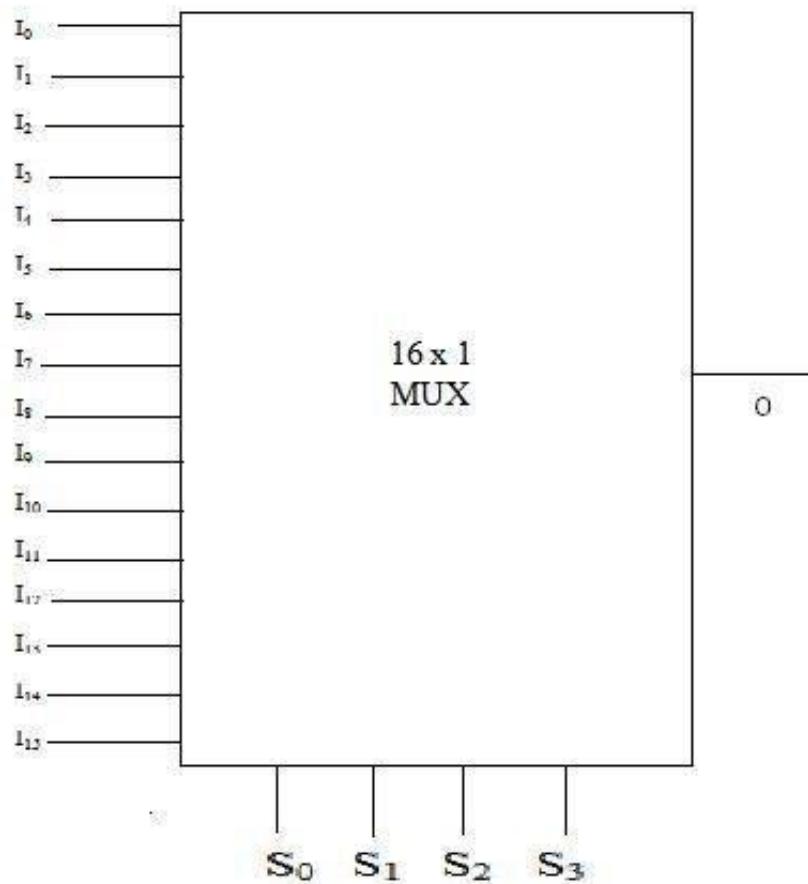


S_0	S_1	Z
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Multiplexer (Data Selector) :

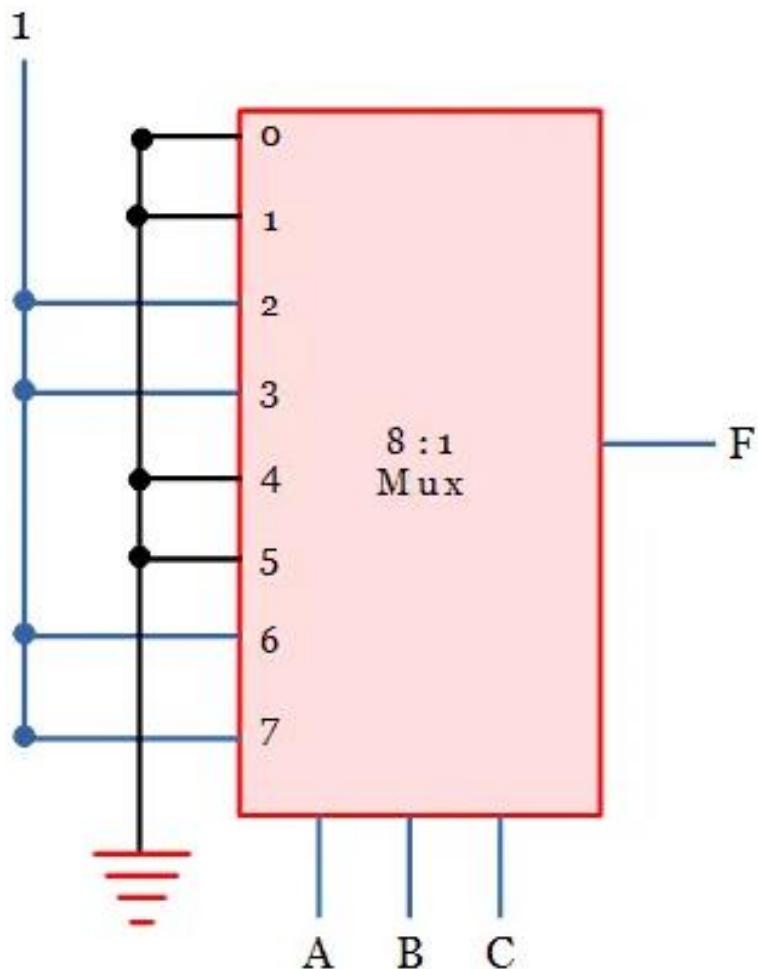


Multiplexer (Data Selector) :



Multiplexer Numerical :

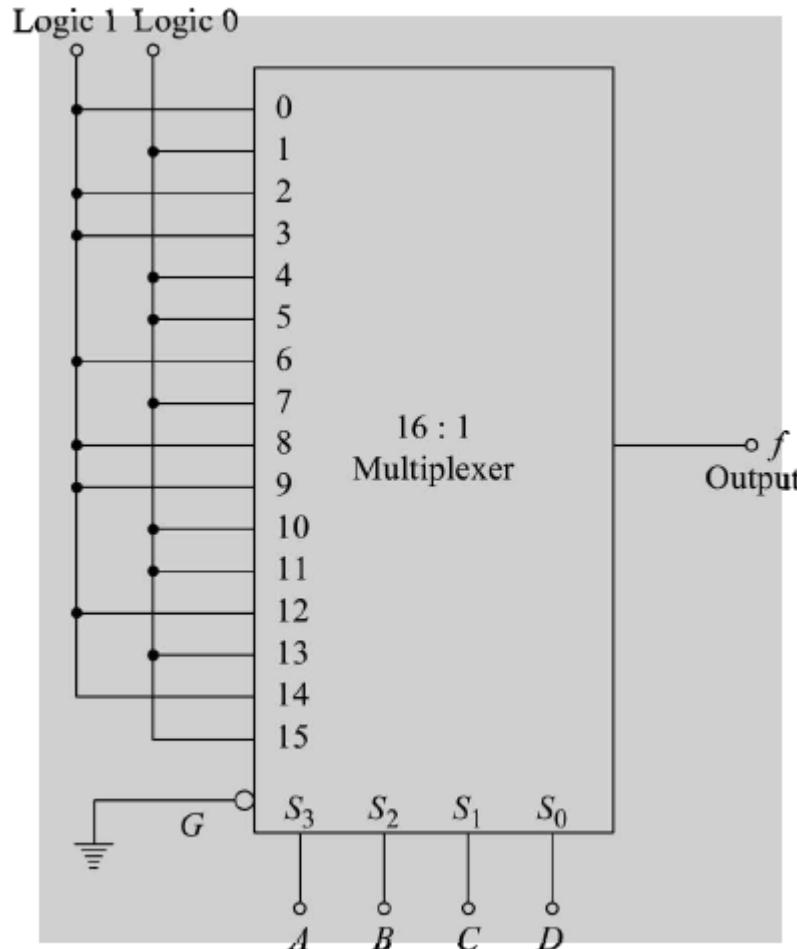
Implement given function using Multiplexer $F = m(2,3,6,7)$



Multiplexer Numerical :

Implement the expression using a multiplexer.

$$f(A, B, C, D) = \Sigma m(0, 2, 3, 6, 8, 9, 12, 14)$$



Implementation of Logic Expression

Multiplexer Numerical :

Realize logic function of truth table using Multiplexer

Inputs				Output
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0

Realize the logic function of truth table given in below table

Inputs				Output
A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Inputs			Output
A	B	C	Y
0	0	0	0
0	0	1	\bar{D}
0	1	0	\bar{D}
0	1	1	1
1	0	0	D
1	0	1	1
1	1	0	\bar{D}
1	1	1	D

The second method can also be used if the logic expression is specified.

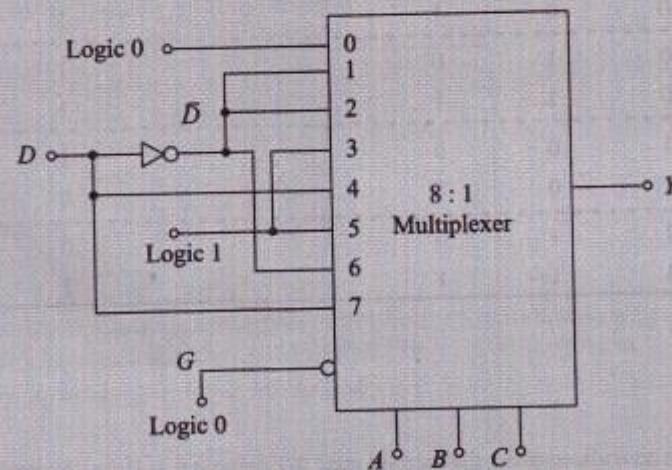


Fig. 6.4 Realisation of 4-variable Truth Table Using 8:1 Multiplexer

Multiplexer Numerical :

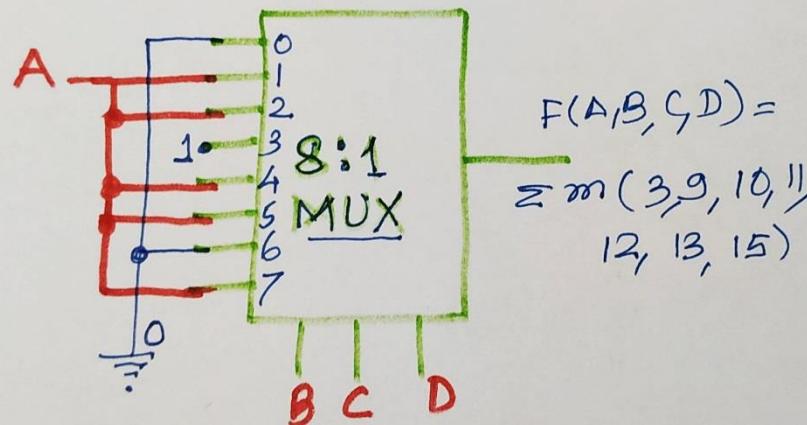
Implement given function using 8:1 Mux $F = m(3, 9, 10, 11, 12, 13, 15)$

Using 8:1 MUX

$$F(A, B, C, D) = \sum m(3, 9, 10, 11, 12, 13, 15)$$

\bar{A}	0	1	2	3	4	5	6	7
A	8	9	10	11	12	13	14	15

0 A A 1 A A 0 A



Multiplexer Numerical :

Implement given function using 8:1 Mux $F = m(0,3,4,7,8,9,13,14)$

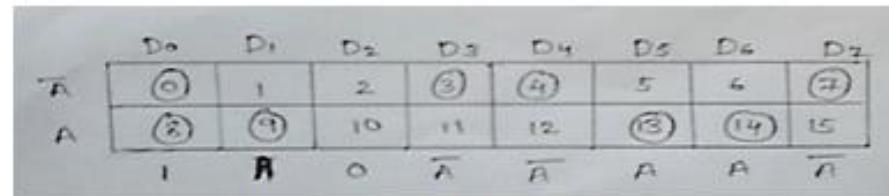


Fig11. K-map for given function

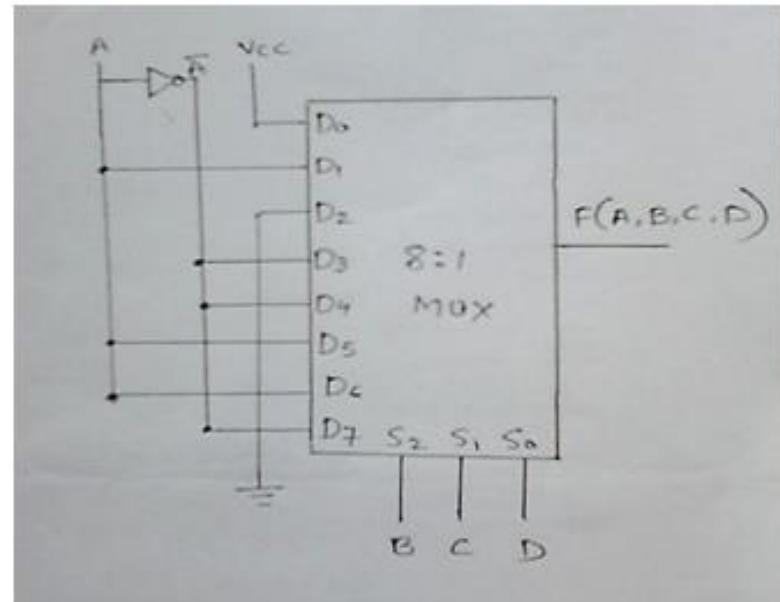
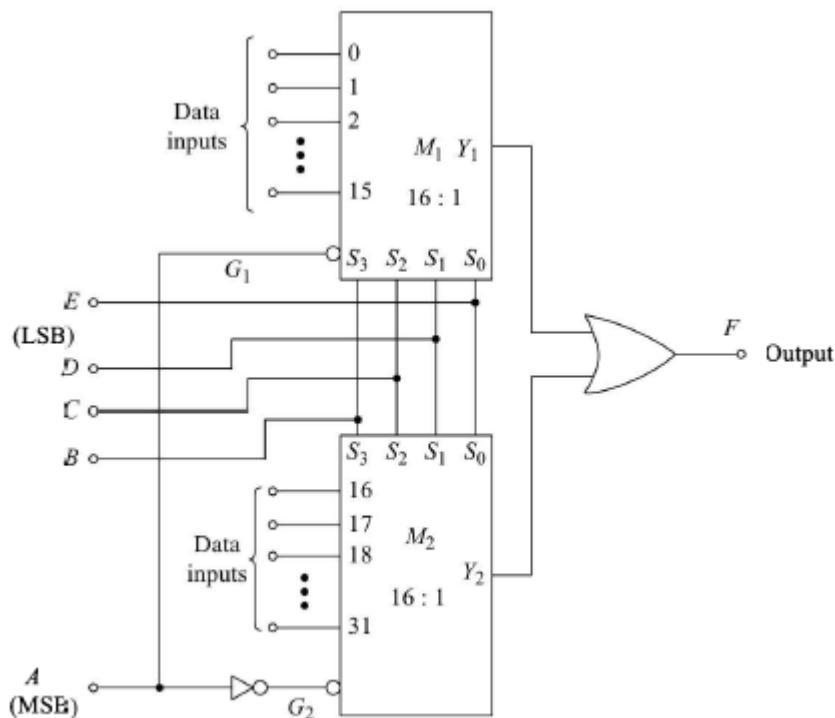


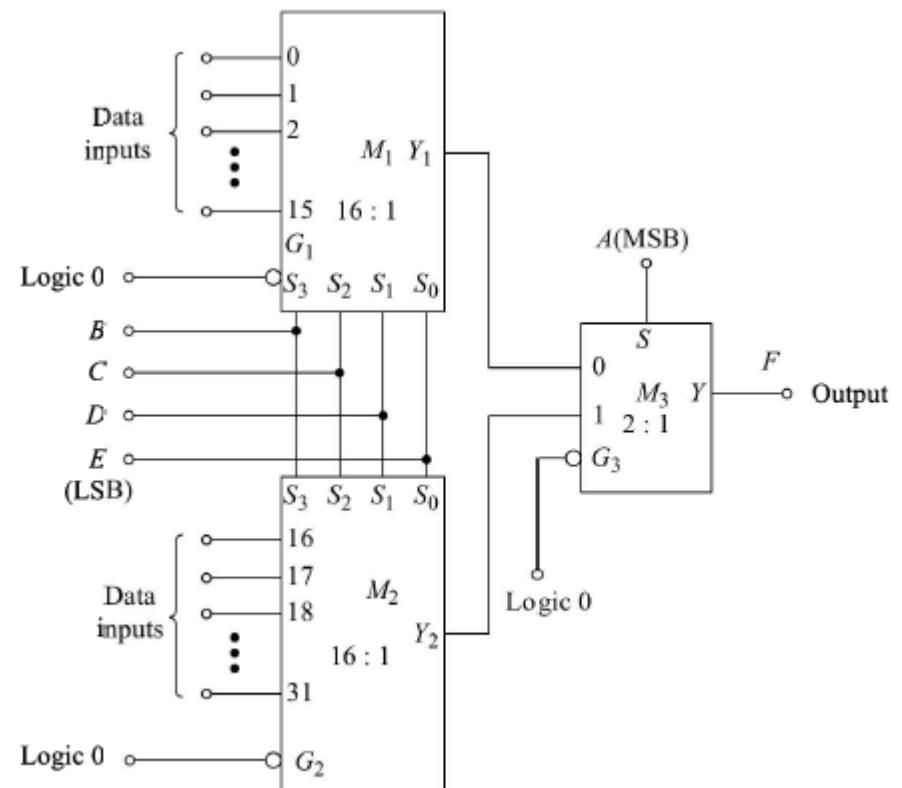
Fig12. Implementation using 8:1 MUX

Multiplexer Tree

Implementing higher order mux using lower order mux



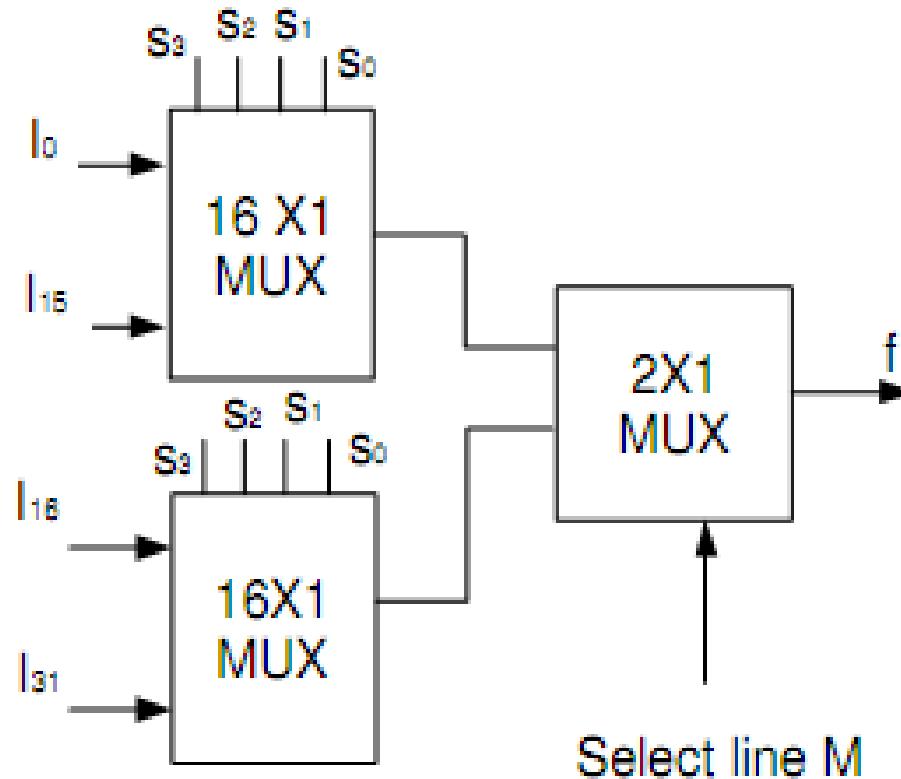
32:1 Multiplexer Using Two 16:1 Multiplexers and One OR Gate



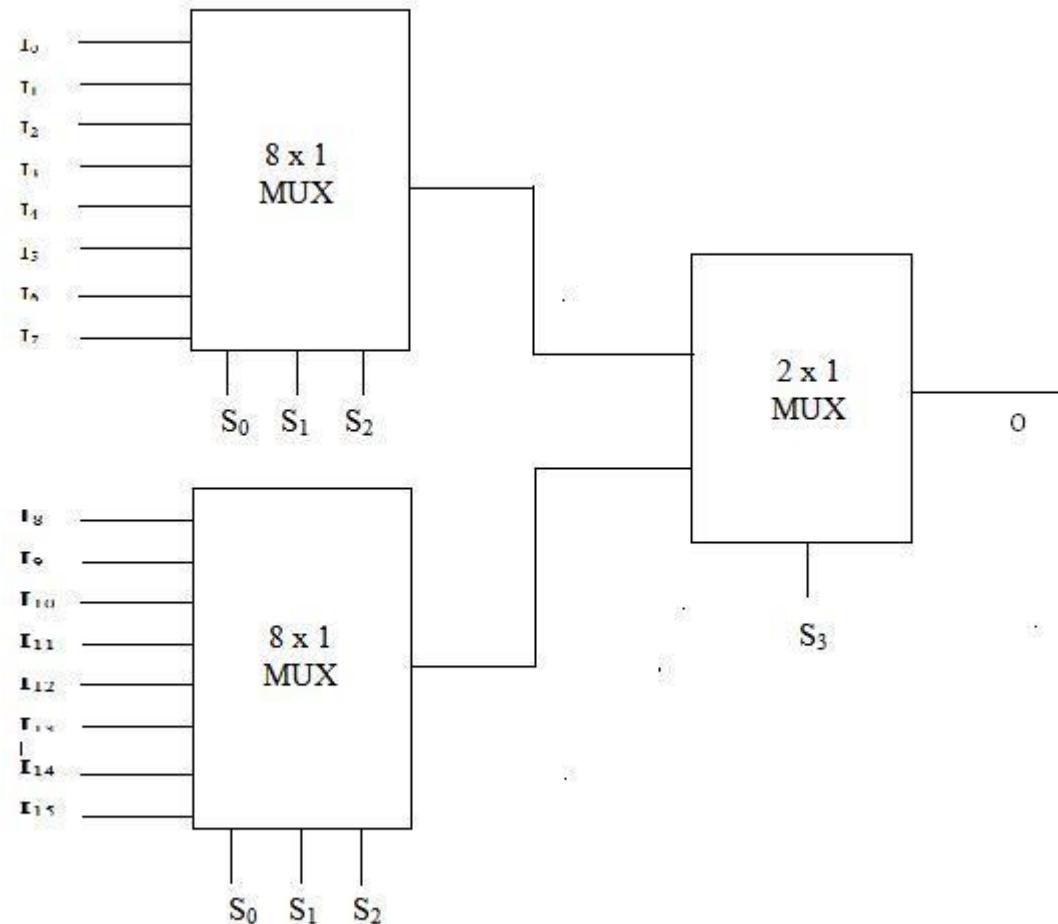
32:1 Multiplexer Using Two 16:1 Multiplexers and One 2:1 Multiplexer

Multiplexer Tree

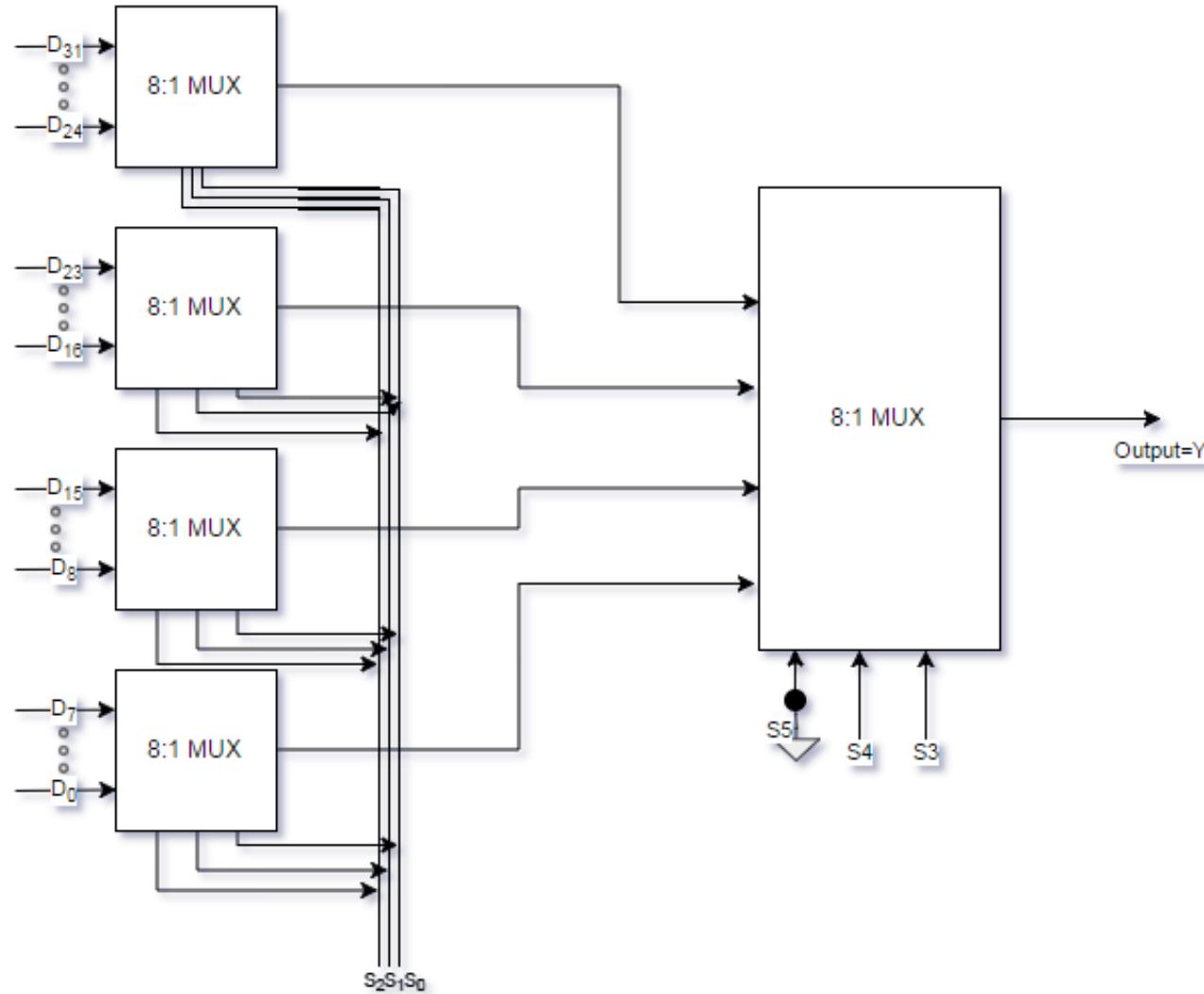
Implementing higher order mux using lower order mux



Multiplexer Tree : 16:1 Mux using 8:1 Mux



Multiplexer Tree



Multiplexer Tree

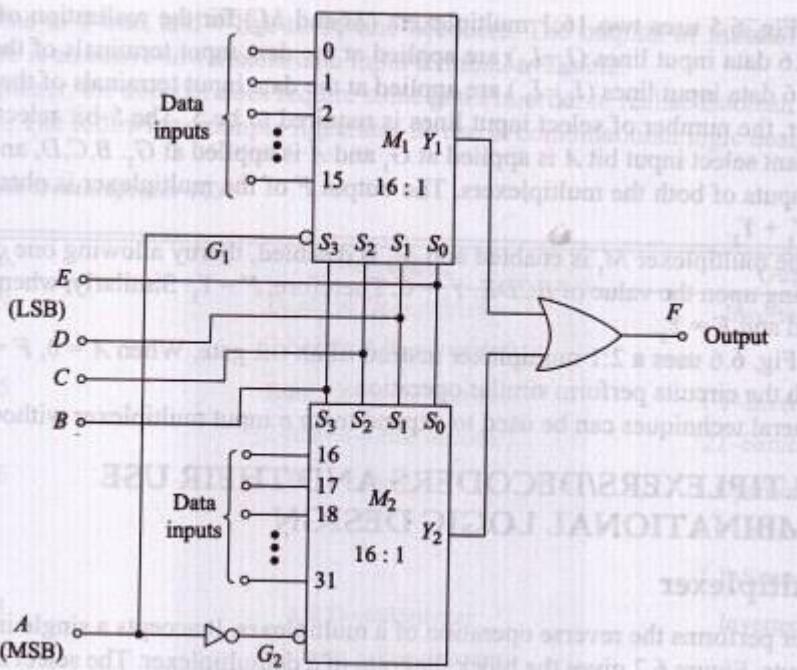


Fig. 6.5 32:1 Multiplexer Using Two 16:1 Multiplexers and One OR Gate

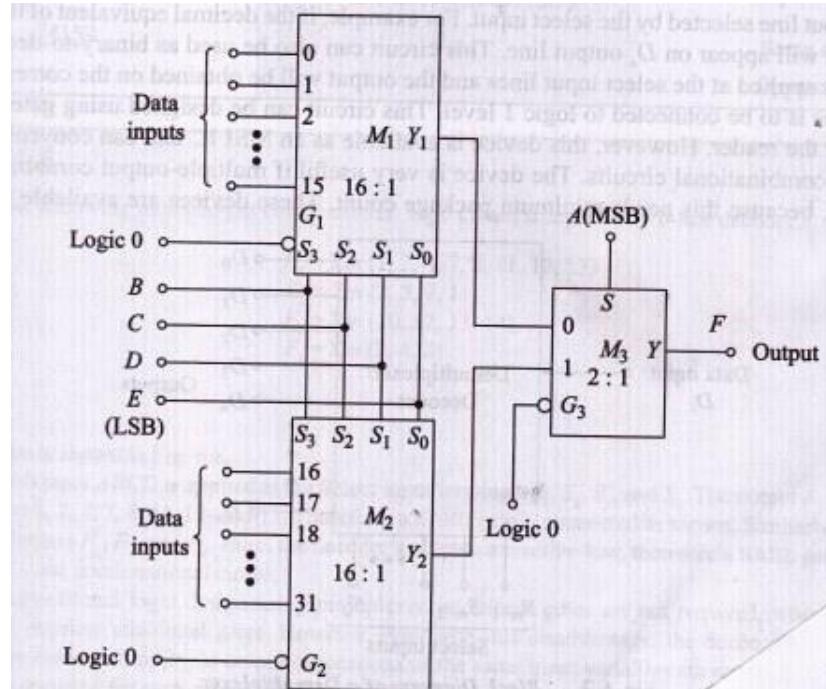
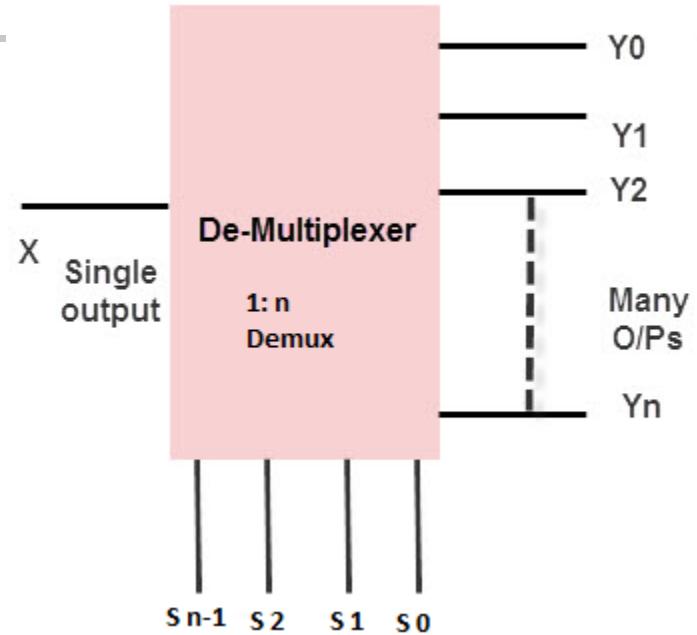
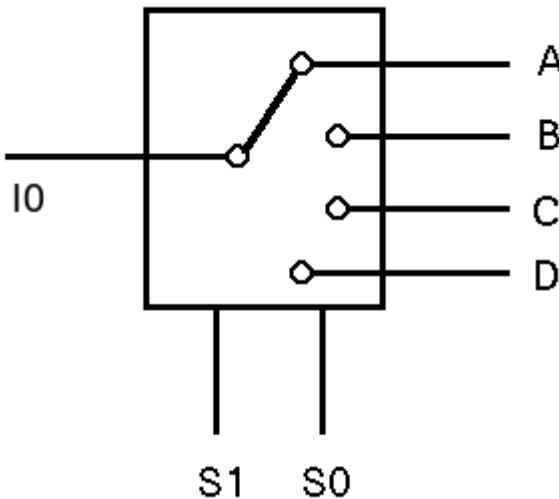


Fig. 6.6 32:1 Multiplexer Using Two 16:1 Multiplexers and One OR Gate

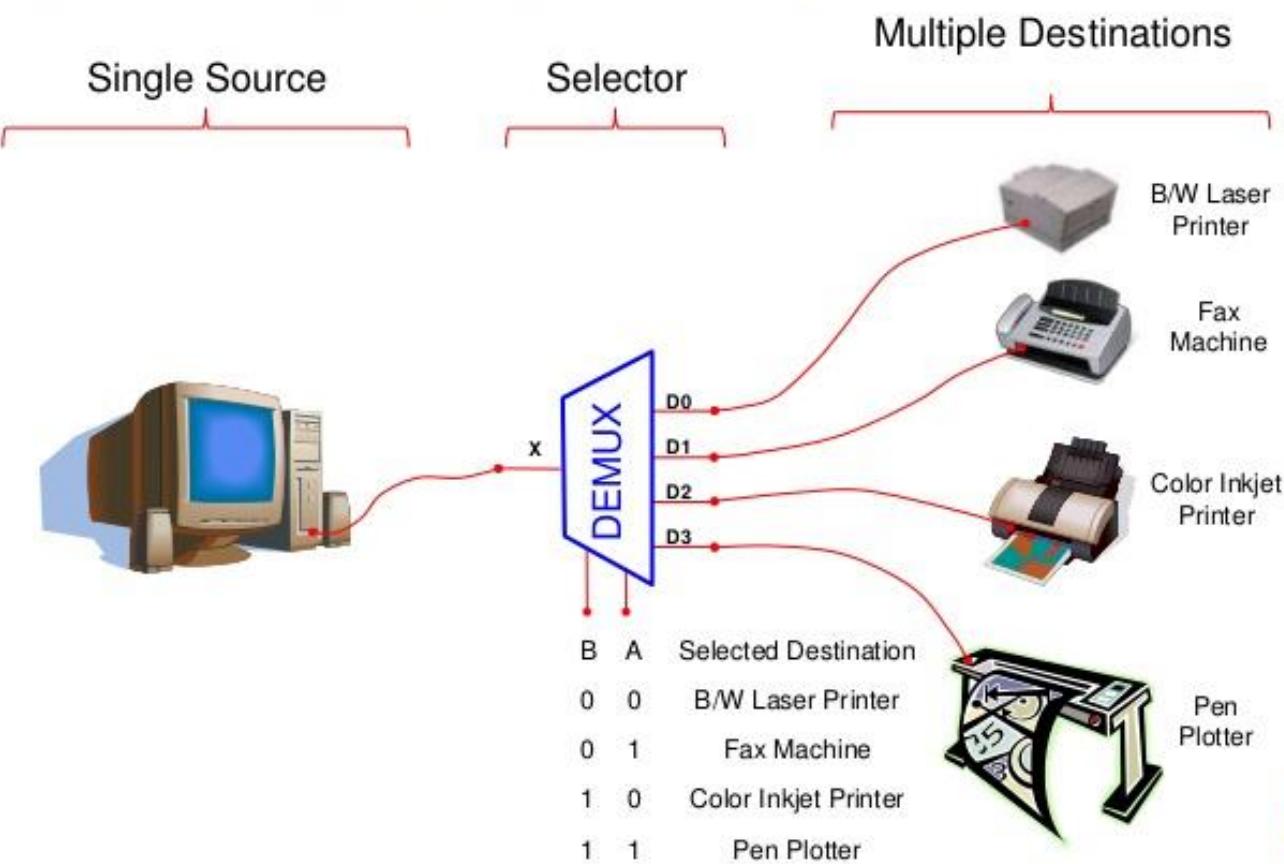
Demultiplexer (Data Distributor) :



Logic circuit that accepts single input & distributes it over several outputs.

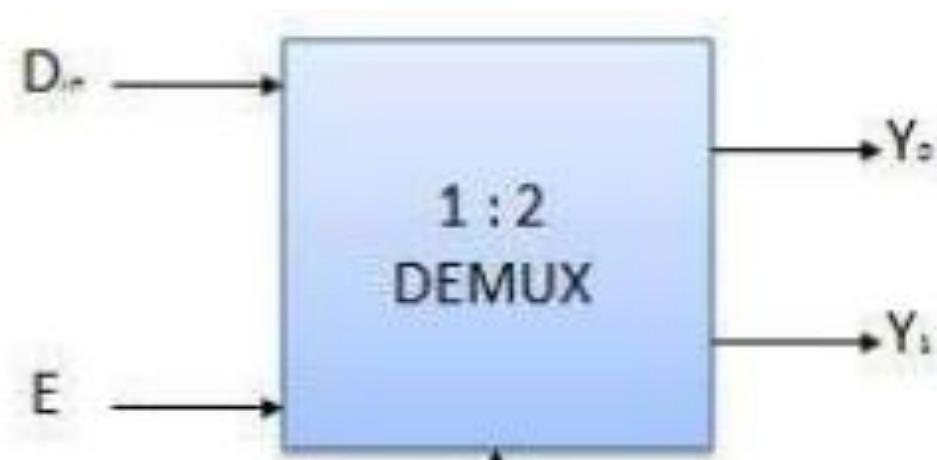
- Applications:**
- Communication System
 - ALU (Arithmetic Logic Unit)
 - Serial to parallel converter

TYPICAL APPLICATION OF A DEMUX

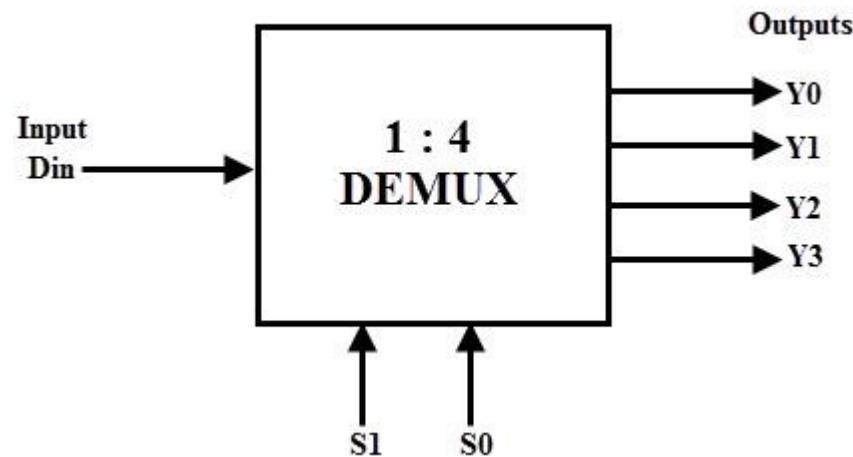


Demultiplexer (Data Distributor) :

- 1 : 2 demultiplexer
- 1 : 4 demultiplexer
- 1 : 16 demultiplexer
- 1 : 32 demultiplexer



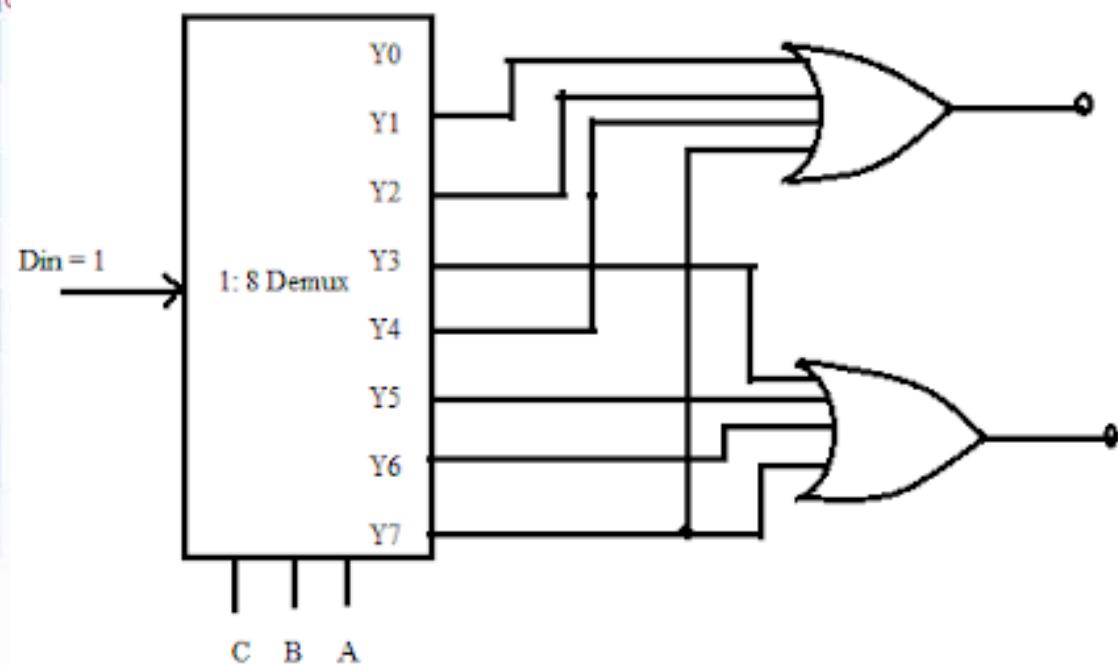
Demultiplixer (Data Distributor) :



Implement Full Adder using Demux

Truth table of 1 bit full adder:

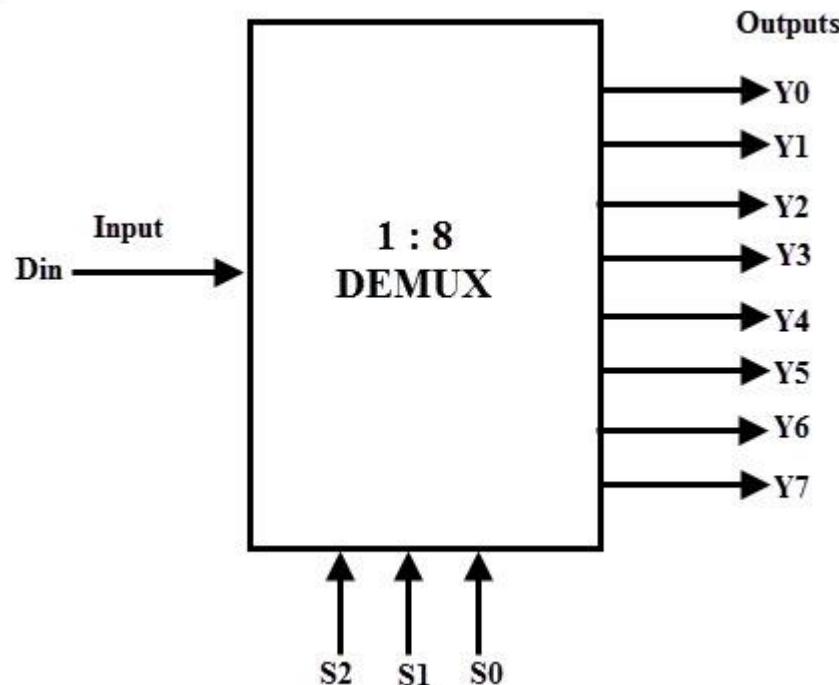
A	B	C _{In}	Sum	Carry _{Out}
0	0	0	0	0
1	0	1	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1



$$\text{Sum} = \sum m(1, 2, 4, 7)$$

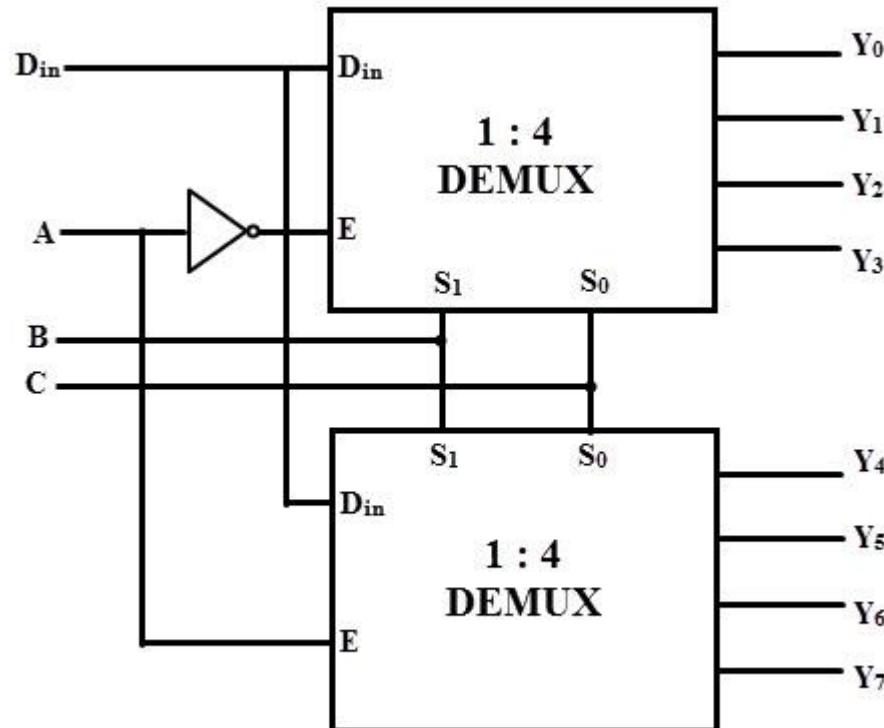
$$\text{Carry} = \sum m(3, 5, 6, 7)$$

Demultiplixer (Data Distributor) :



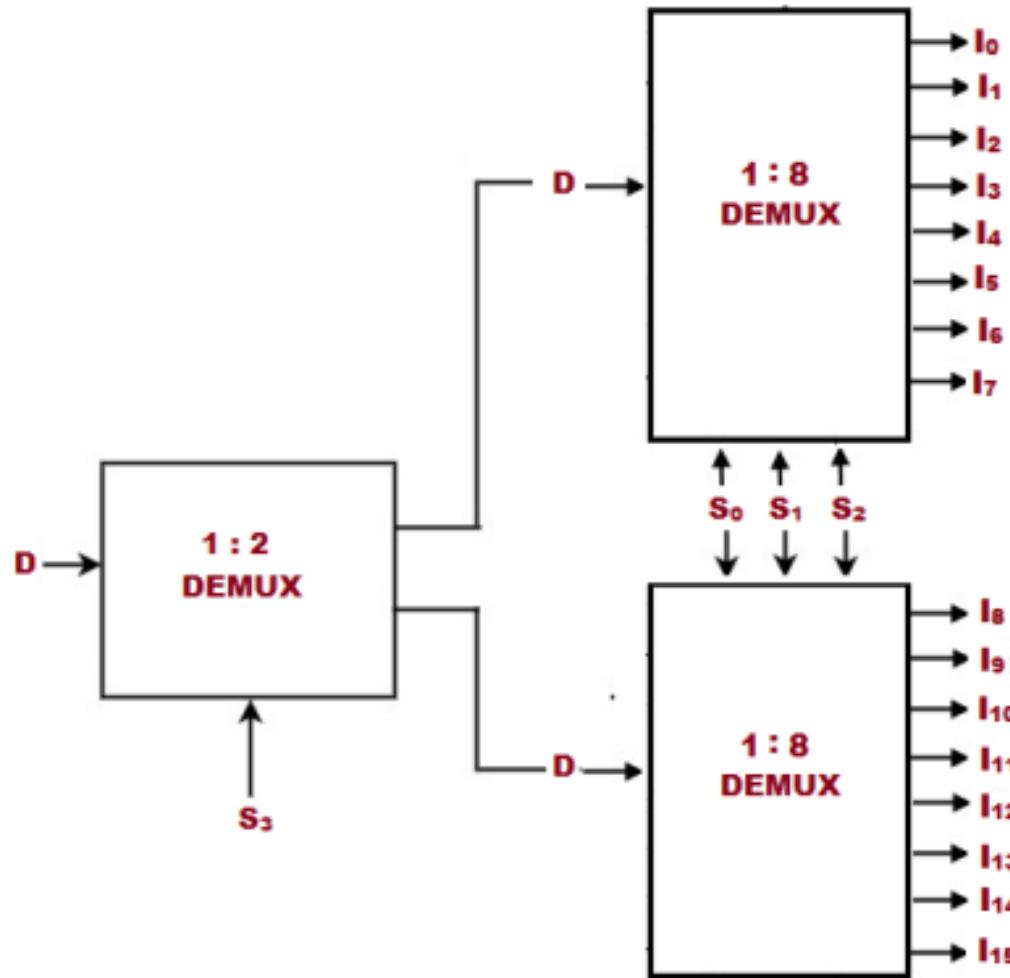
Demultiplexer Tree

1:8 Demux using 1:4 Demux

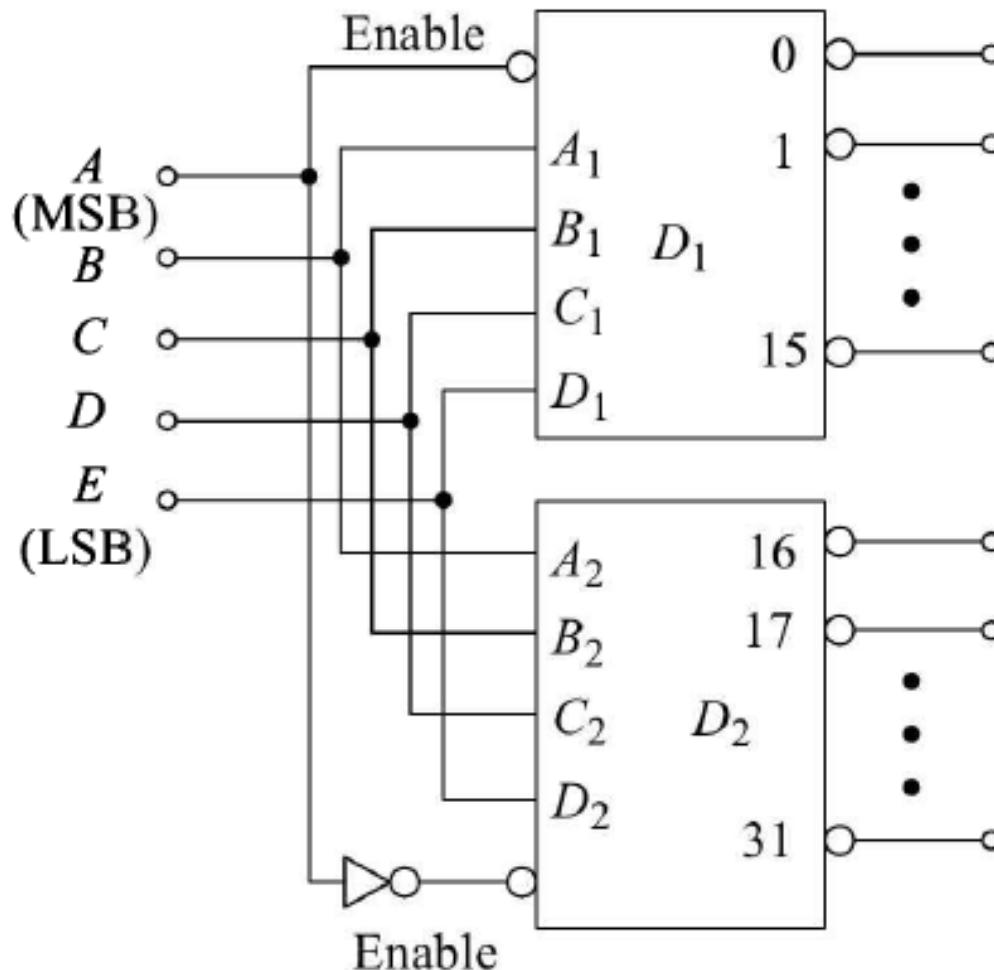


Demultiplexer Tree

1:16 Demux using 1:8 Demux & 1:2 Demux



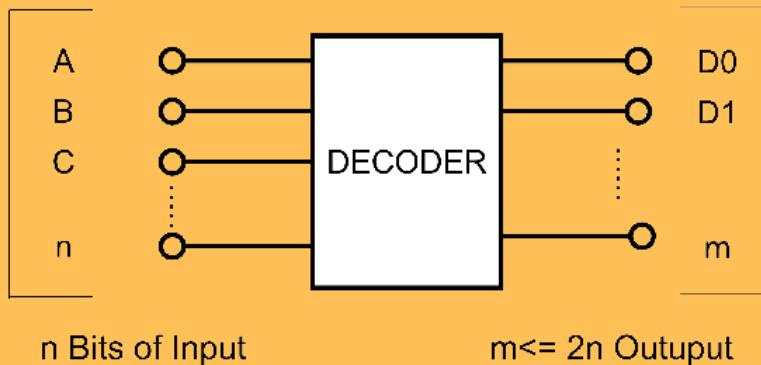
Demultiplexer Tree



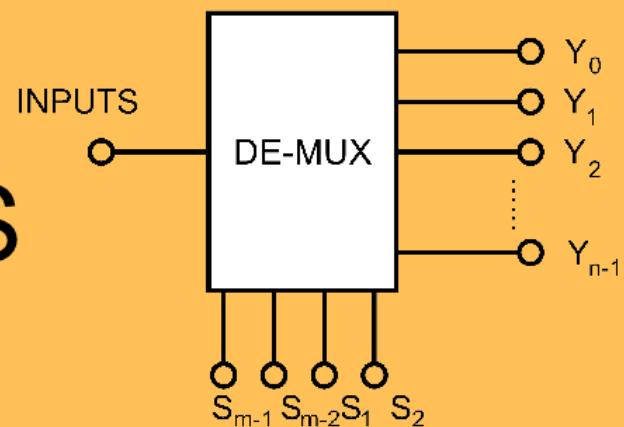
**5-Line-to-32-Line Decoder Using Two
4-line-to-16-Line Decoders**

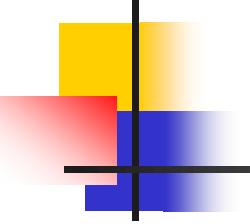
Decoder

Decoder & De-Multiplexer



VS

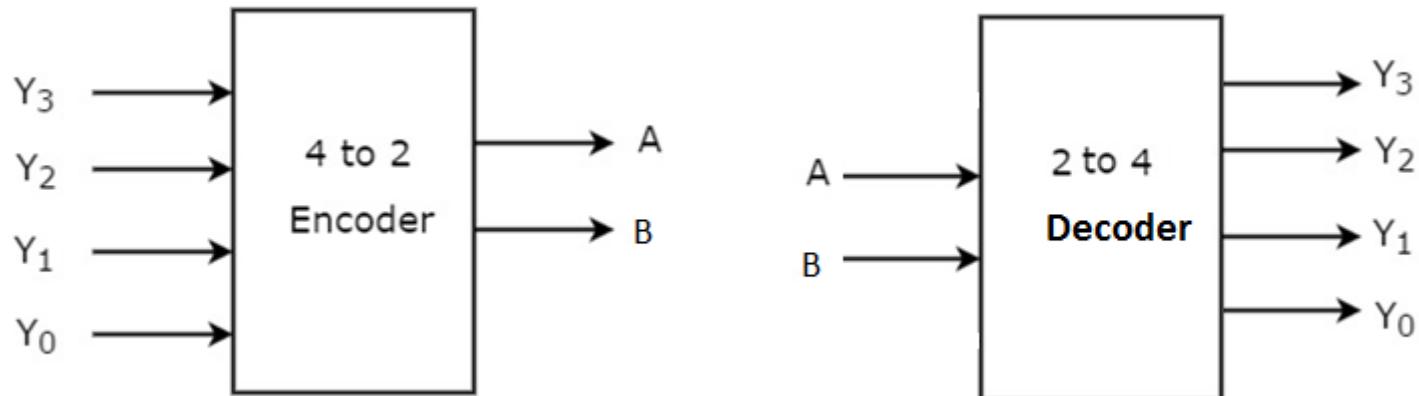




Decoder

	Demultiplexer	Decoder
Definition	1 data input 2^n outputs	It has n inputs 2^n outputs It has n control inputs
Characteristic	Connects the data input to the data output	Selects one of the 2^n outputs by decoding the binary value on the basis of n inputs
Reverse of	Multiplexer	Encoder

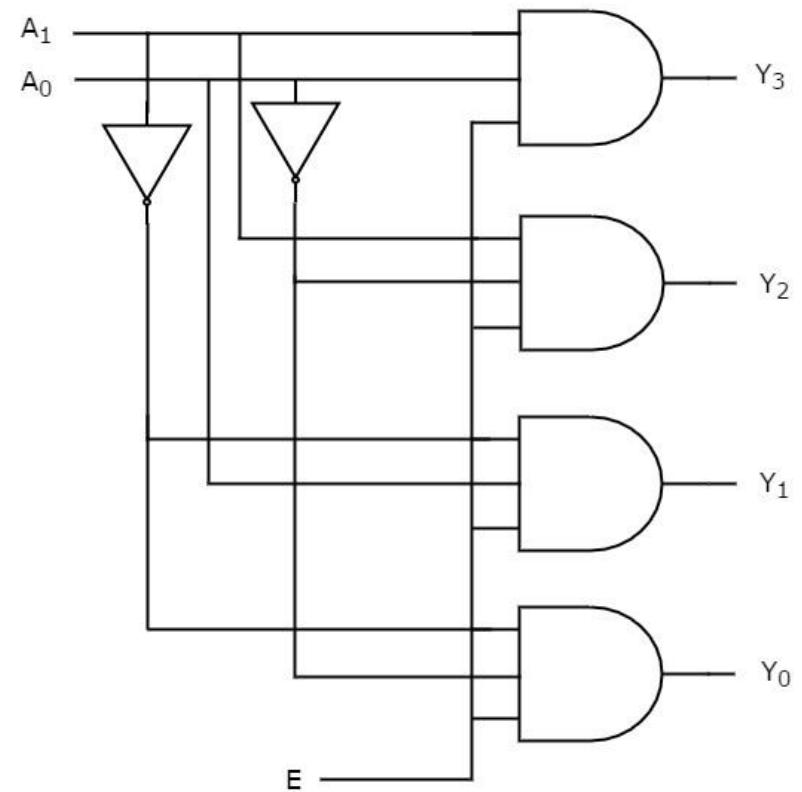
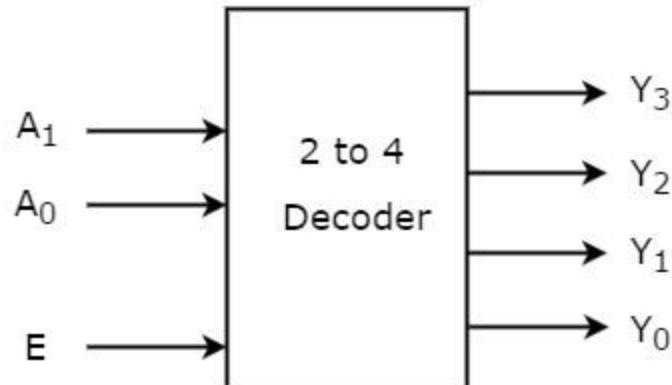
Encoder and Decoder



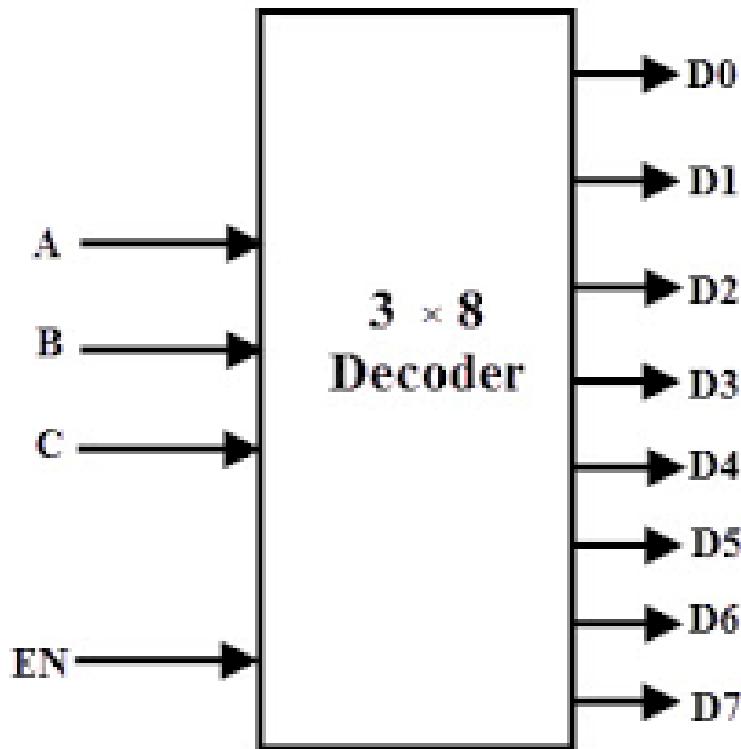
Encoder and Decoder

PARAMETER	ENCODER	DECODER
Input applied	Original message signal	Coded binary input
Output generated	Coded binary output	Original message
Input lines	2^n	n
Output lines	n	2^n
Operation	Simple	Complex
Basic logic element	OR gate	AND gate along with NOT gate
Applications	E-mail , video encoders etc.	Microprocessors, memory chips etc.

Decoder



Decoder



Decoder

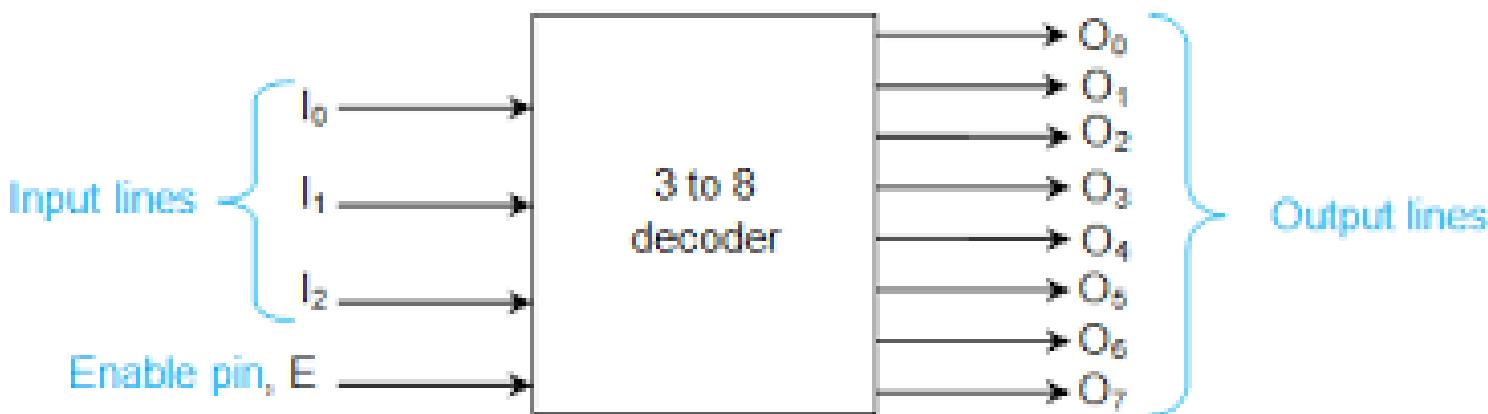
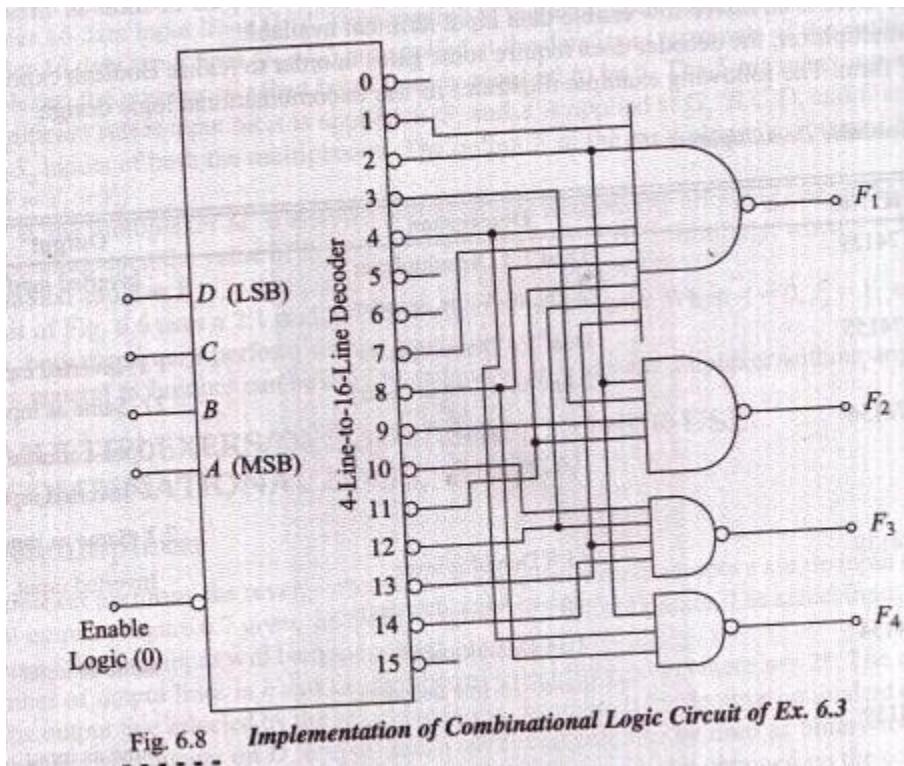


Figure 1 3 to 8 binary decoder

Implement following function using 4 to 16 line decoder

$$F_1 = \Sigma m (1, 2, 4, 7, 8, 11, 12, 13)$$
$$F_2 = \Sigma m (2, 3, 9, 11)$$
$$F_3 = \Sigma m (10, 12, 13, 14)$$
$$F_4 = \Sigma m (2, 4, 8)$$



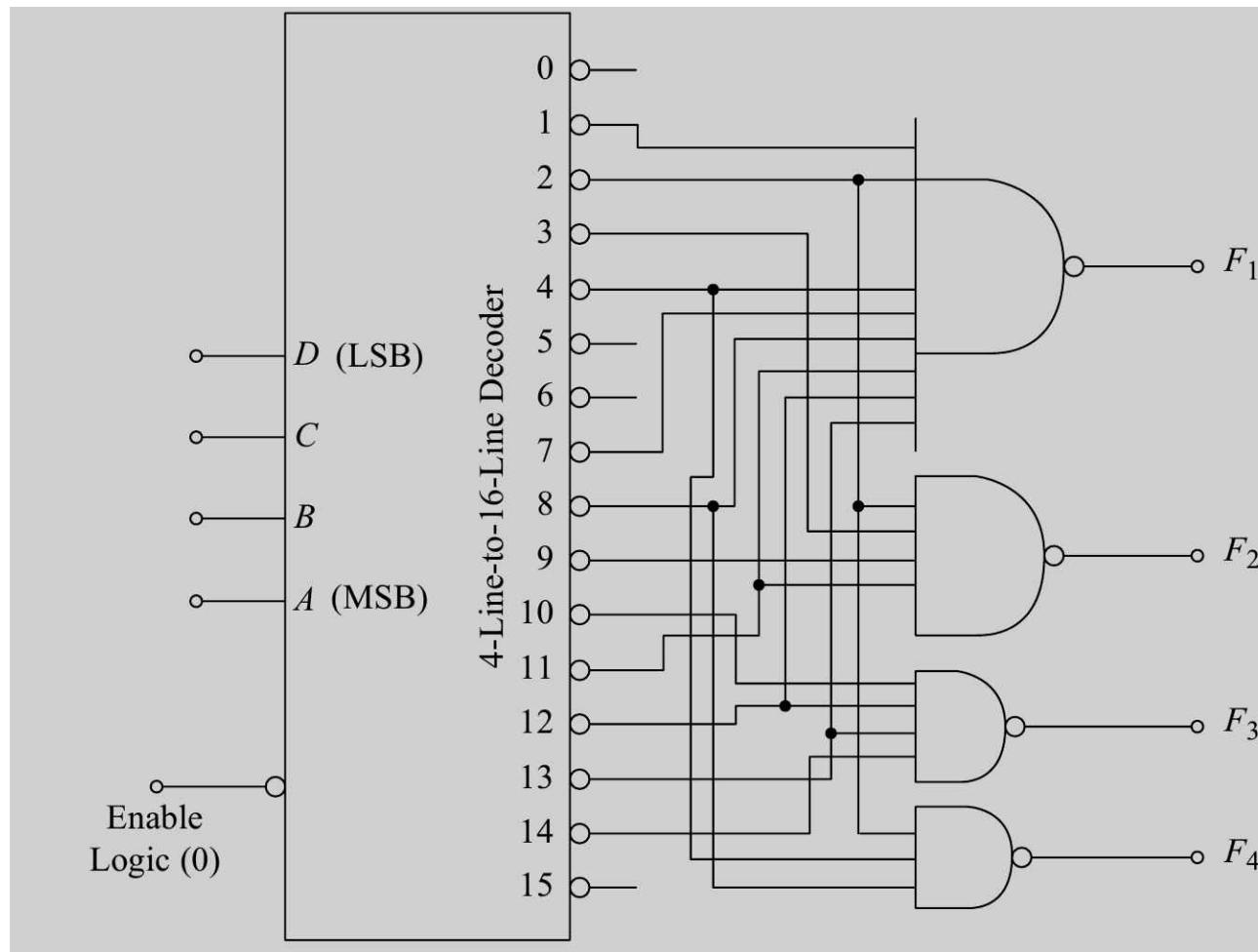
Implement the following multi-output combinational logic circuit using a 4-to-16-line decoder.

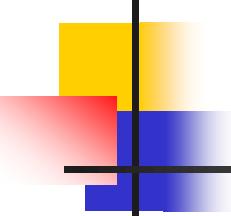
$$F_1 = \Sigma m (1, 2, 4, 7, 8, 11, 12, 13)$$

$$F_2 = \Sigma m (2, 3, 9, 11)$$

$$F_3 = \Sigma m (10, 12, 13, 14)$$

$$F_4 = \Sigma m (2, 4, 8)$$

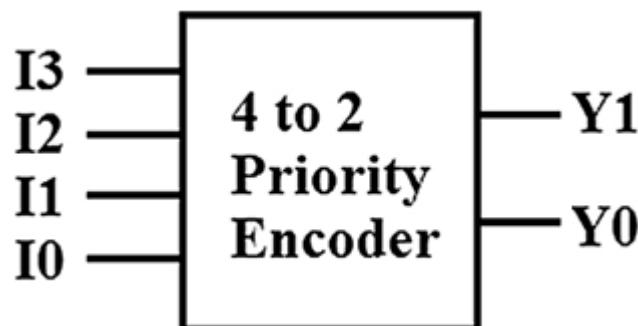




Priority Encoder

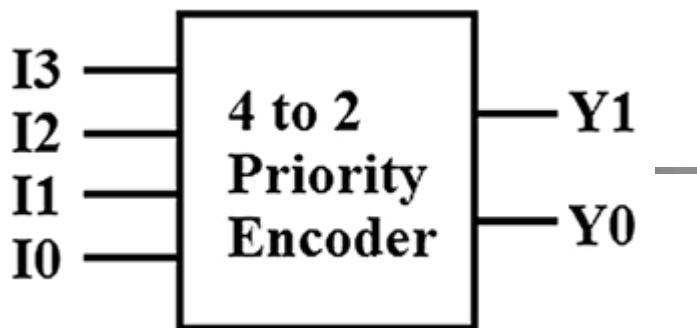
- It gives priority from highest to lowest
- We assume I3 has highest priority and I0 has lowest priority
- When I3 is 1 then, we don't need to look in to I2, I1 and I0 (No matter what is the input to I2, I1 and I0), Here I3 will get encoded with the output Y1=1 and Y0=1
- When I2 is 1 then, we don't need to look in to I1 and I0 (No matter what is the input to I1 and I0), Here I2 will get encoded with the output Y1=1 and Y0=1
- When I1 is 1 then, we don't need to look in to I0 (No matter what is the input to I0), Here I1 will get encoded with the output Y1=0 and Y0=1
- When I0 is 1 then, Here I0 will get encoded with the output Y1=0 and Y0=0

Priority Encoder



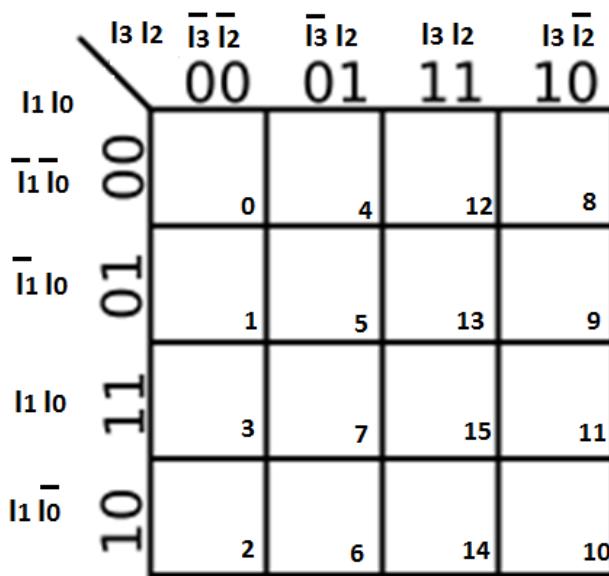
Input				Output	
I ₃	I ₂	I ₁	I ₀	Y ₁	Y ₀
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

Priority Encoder

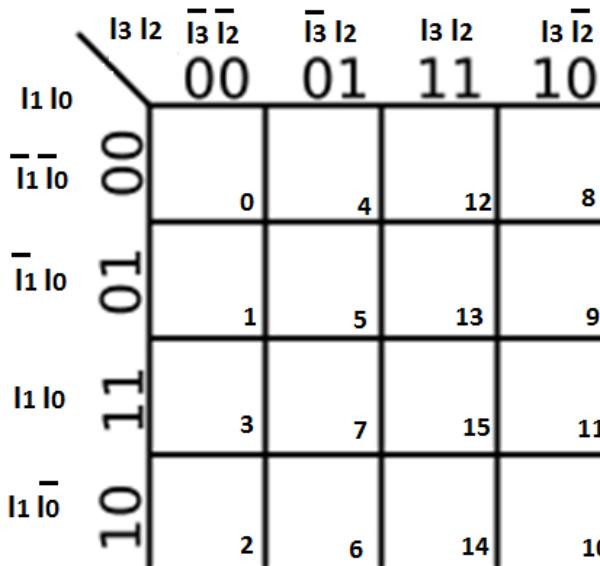


Input				Output	
I3	I2	I1	I0	Y1	Y0
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

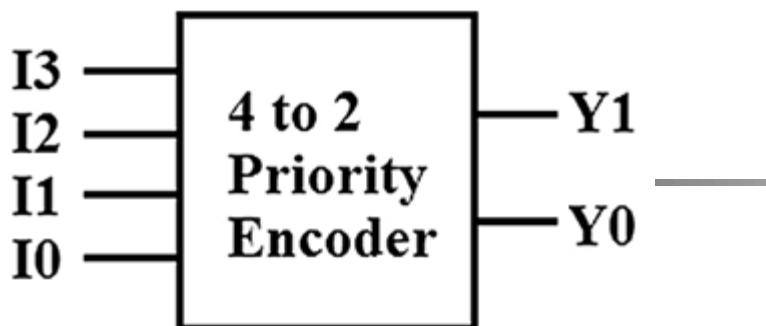
Kmap for Y1



Kmap for Y2

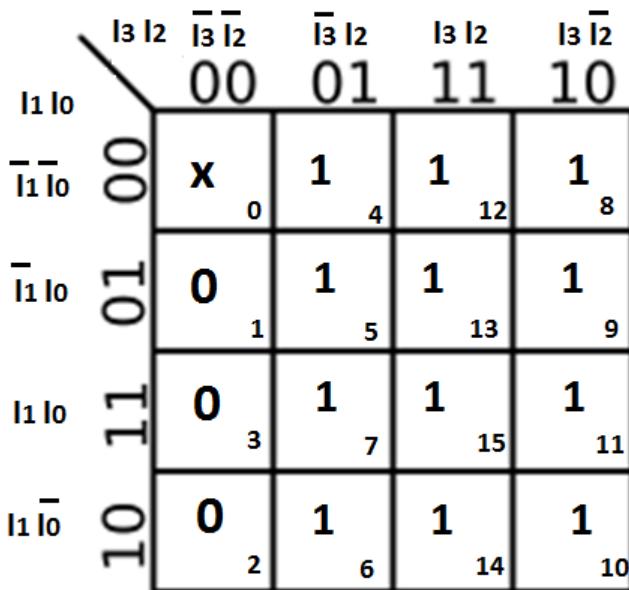


Priority Encoder

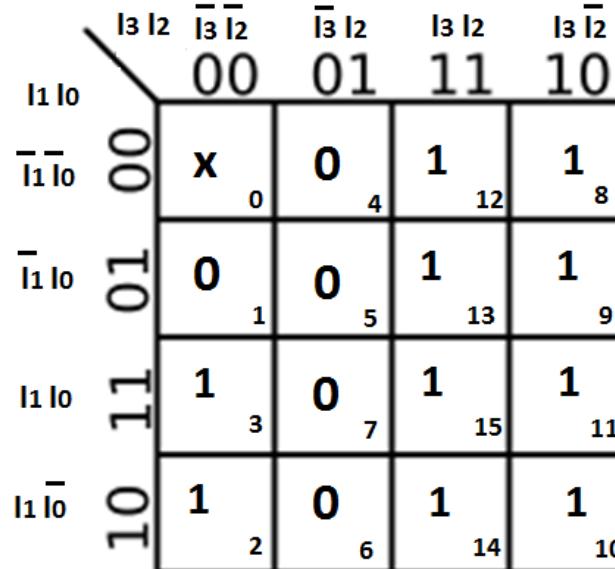


Input				Output	
I ₃	I ₂	I ₁	I ₀	Y ₁	Y ₀
0	0	0	0	X	X
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

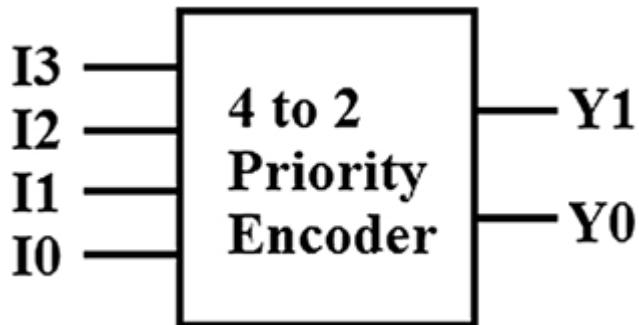
Kmap for Y₁



Kmap for Y₂

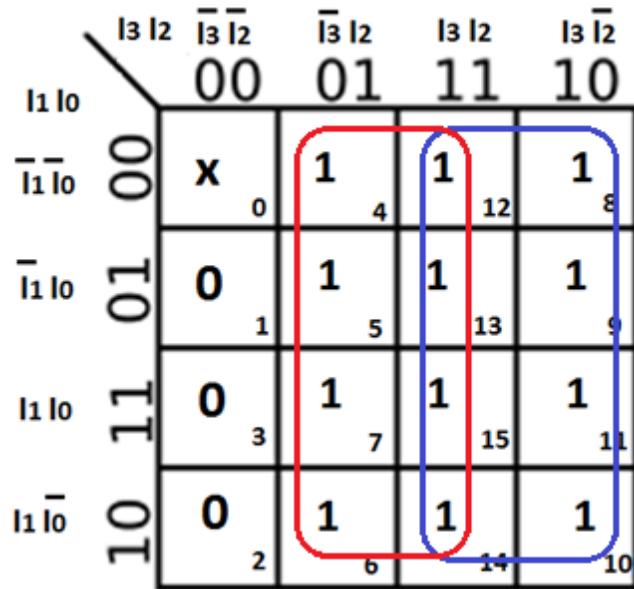


Priority Encoder



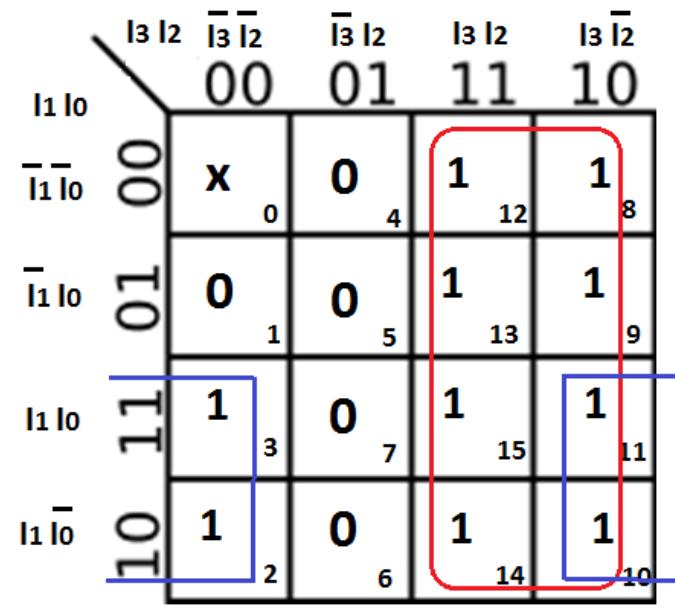
Input				Output	
I ₃	I ₂	I ₁	I ₀	Y ₁	Y ₀
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

Kmap for Y₁



$$Y_1 = I_3 + I_2$$

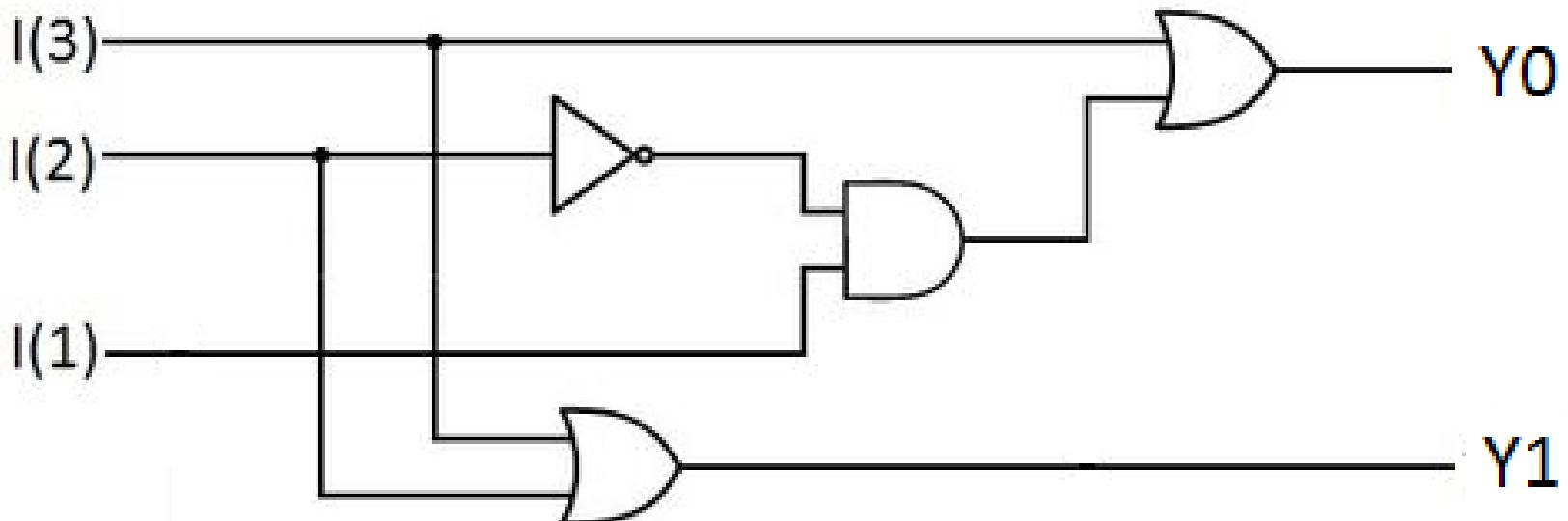
Kmap for Y₂



$$Y_2 = I_3 + I_1 \bar{I}_2$$

Priority Encoder

Circuit Diagram:

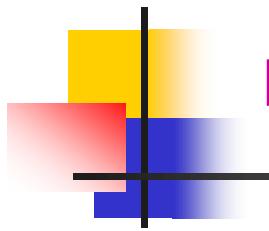


BCD Arithmetic

BCD Addition

BCD numbers can be added as long as certain rules are followed - these are.

if the addition produces a carry
or if it creates an invalid BCD number
then correct the digit by adding six where the error occurred.



BCD Addition: Example

$(7)_{10} + (5)_{10}$ BCD ADDITION

$$\begin{array}{r} \begin{array}{r} 111 \\ 0111 \\ + 0101 \\ \hline \end{array} & = 7 & \text{sum} > 9 \\ & & \\ \begin{array}{r} 1100 \\ + 0110 \\ \hline \end{array} & = 5 & \text{carry} = 0 \\ & & \\ \begin{array}{r} 10010 \\ \hline \end{array} & & \end{array}$$

BCD Addition: Example

Carry				1	
BCD for 9	1	0	0	1	
BCD for 5	0	1	0	1	+
<hr/>					
Carry	1	1	1		
Invalid BCD, sum > 9	1	1	1	0	
Add 0110 for correction	0	1	1	0	
<hr/>					
Valid output	1	0	1	0	0
<hr/>					

Sum of 9 and 5 is 14.

0	0	0	1	0	1	0	0
1				4			

BCD Addition: Example

$$\begin{array}{r} 8 \\ + 9 \\ \hline 17 \end{array} \quad \begin{array}{r} 1000 \\ + 1001 \\ \hline 0001 \\ + 0000 \\ \hline 0001 \end{array} \quad \begin{array}{l} \leftarrow \text{BCD for } 8 \\ \leftarrow \text{BCD for } 9 \\ \leftarrow \text{Incorrect BCD result} \\ \leftarrow \text{Add 6 for correction} \\ \leftarrow \text{BCD for } 17 \end{array}$$

BCD Adder

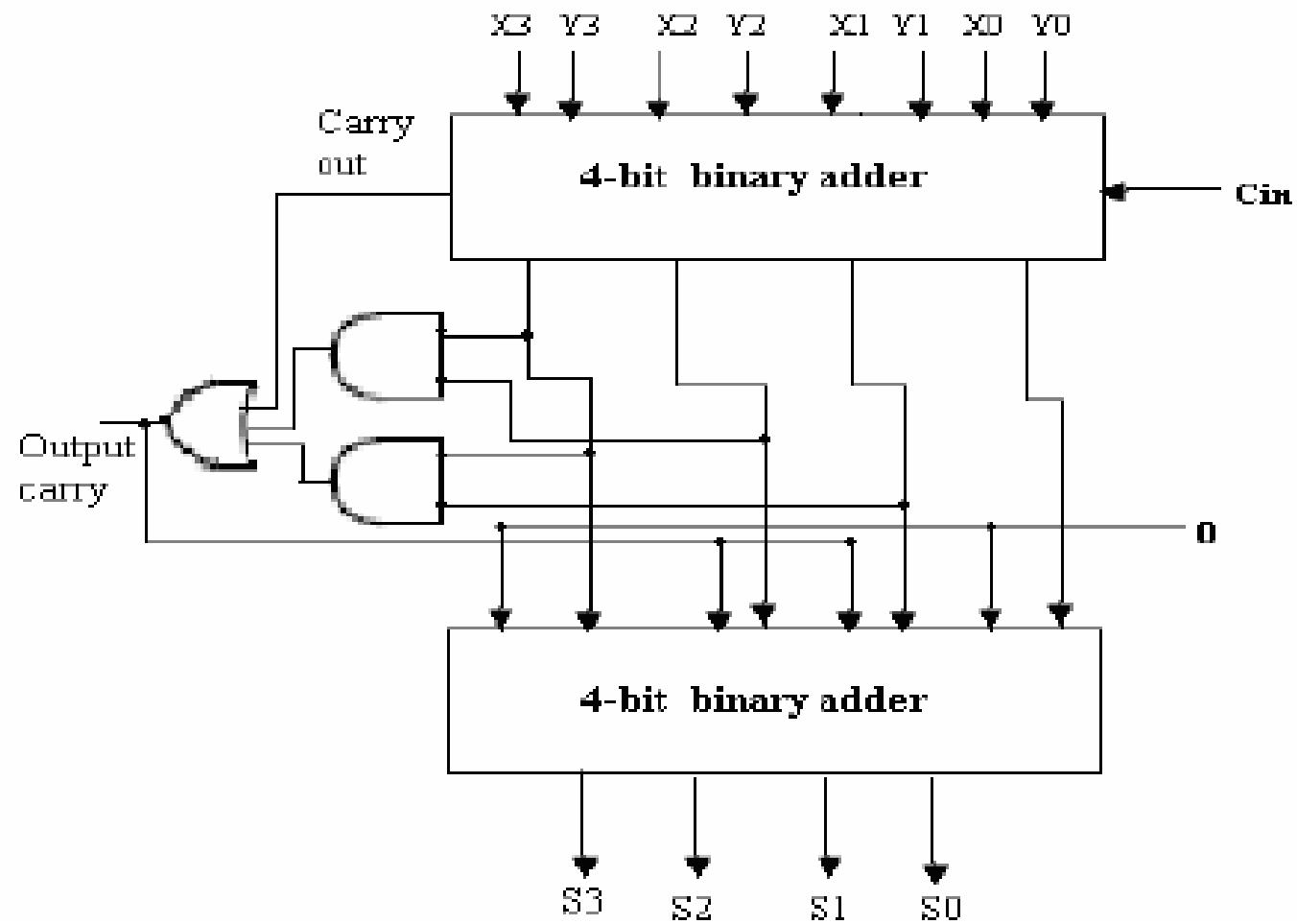
Inputs				Output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Inputs: S_3S_2 , S_1S_0

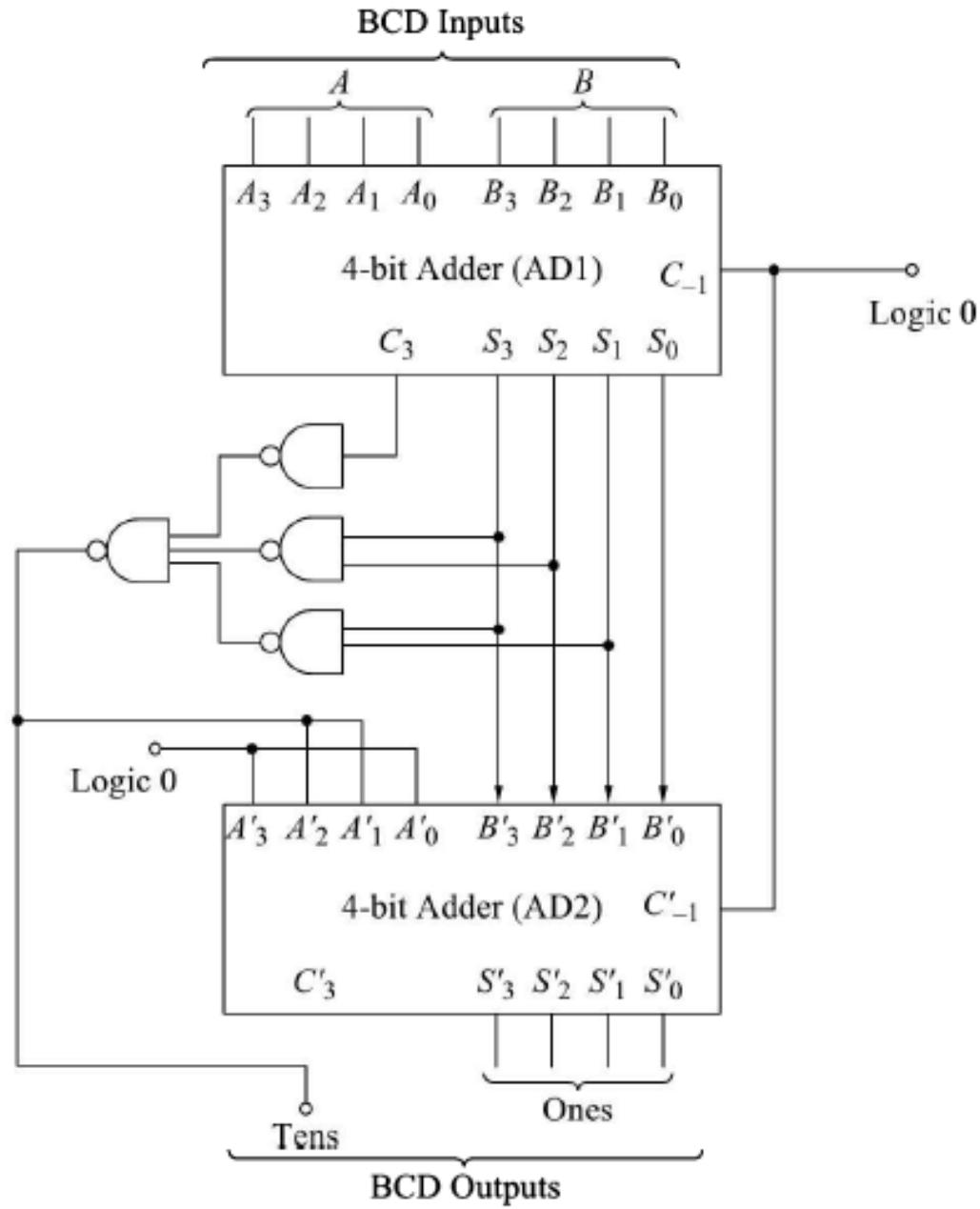
00	01	11	10
0	0	0	0
0	0	0	0
1	1	1	1
0	0	1	1

$$Y = S_3S_2 + S_3S_1$$

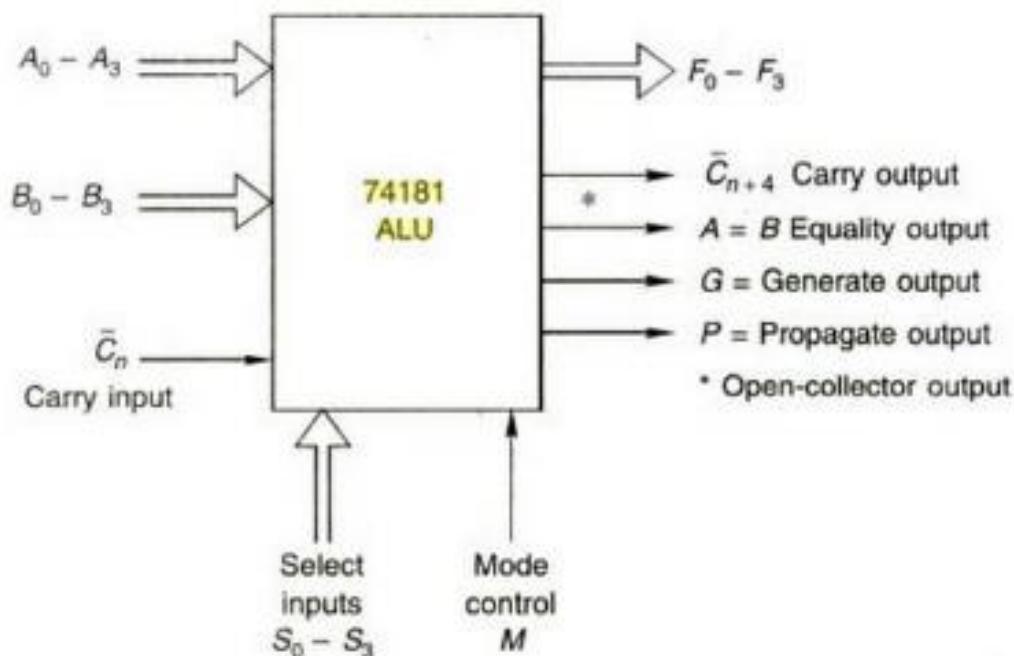
BCD Adder



BCD Adder



ALU Design



A and B – 4 bit binary data input

C_n – Carry input

F – 4 bit binary data output

C_{n+4} – Carry output

A=B – Logic 1 on this line indicates

$A=B$

G – Carry generate

P – Propagate output

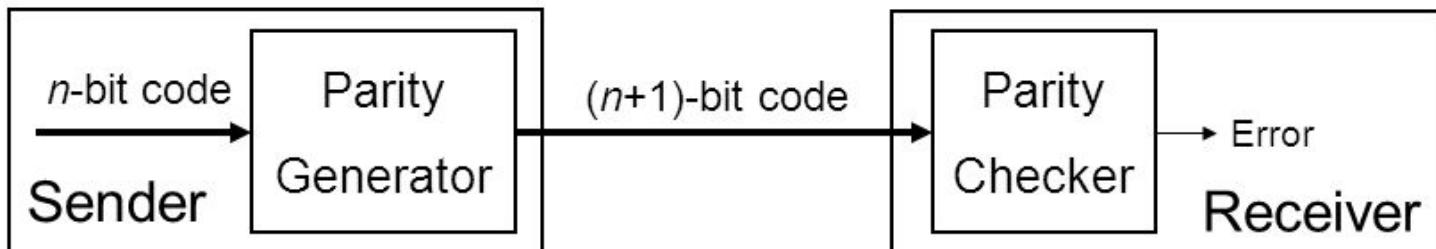
S – Selects one of the operation

M – indicates arithmetic or logical operation

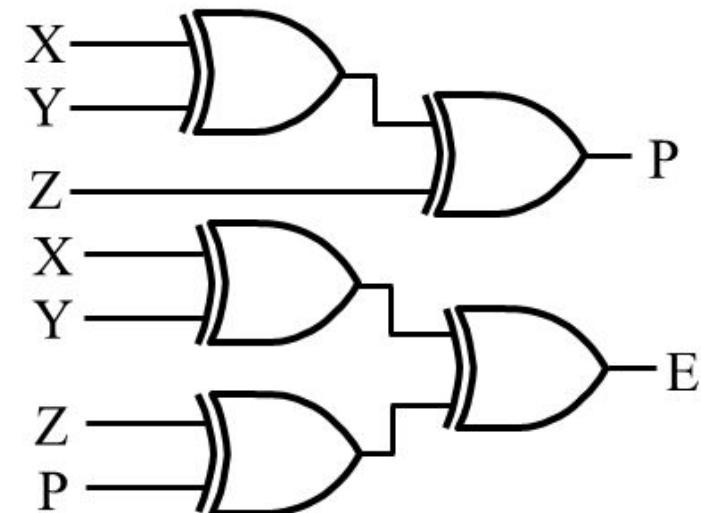
ALU Elementary Design: Function Table

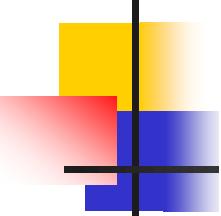
				Active high data			
Line	Selection			$M = 1$: Logic functions		$M = 0$; Arithmetic operations	
	S_3	S_2	S_1	S_0	$C_n = X$		
0	0	0	0	0	$F = \bar{A}$	$F = A$	$F = A \text{ PLUS } 1$
1	0	0	0	1	$F = \bar{A} + B$	$F = A + B$	$F = (A + B) \text{ PLUS } 1$
2	0	0	1	0	$F = \bar{A} \cdot B$	$F = A + \bar{B}$	$F = (A + \bar{B}) \text{ PLUS } 1$
3	0	0	1	1	$F = 0$	$F = \text{MINUS } 1$ (2's COMPL)	$F = \text{ZERO}$
4	0	1	0	0	$F = \bar{A} \bar{B}$	$F = A \text{ PLUS } A \bar{B}$	$F = A \text{ PLUS } A \bar{B} \text{ PLUS } 1$
5	0	1	0	1	$F = B$	$F = (A + B)$ PLUS $A \bar{B}$	$F = (A + B) \text{ PLUS } A \bar{B} \text{ PLUS } 1$
6	0	1	1	0	$F = A \oplus B$	$F = A \text{ MINUS } B$ MINUS 1	$F = A \text{ MINUS } B$
7	0	1	1	1	$F = A \bar{B}$	$F = A \bar{B} \text{ MINUS } 1$	$F = A \bar{B}$
8	1	0	0	0	$F = \bar{A} + B$	$F = A \text{ PLUS } AB$	$F = A \text{ PLUS } AB \text{ PLUS } 1$
9	1	0	0	1	$F = \bar{A} \oplus B$	$F = A \text{ PLUS } B$	$F = A \text{ PLUS } B \text{ PLUS } 1$
10	1	0	1	0	$F = B$	$F = (A + \bar{B})$ PLUS AB	$F = (A + \bar{B}) \text{ PLUS } AB \text{ PLUS } 1$
11	1	0	1	1	$F = AB$	$F = AB \text{ MINUS } 1$	$F = AB$
12	1	1	0	0	$F = 1$	$F = A \text{ PLUS } A$	$F = A \text{ PLUS } A \text{ PLUS } 1$
13	1	1	0	1	$F = A + \bar{B}$	$F = (A + B)$ PLUS A	$F = (A + B) \text{ PLUS } A \text{ PLUS } 1$
14	1	1	1	0	$F = A + B$	$F = (A + \bar{B})$ PLUS A	$F = (A + \bar{B}) \text{ PLUS } A \text{ PLUS } 1$
15	1	1	1	1	$F = A$	$F = A \text{ MINUS } 1$	$F = A$

Parity Generator & Checkers



- Design an even parity generator and checker for 3-bit codes
- Solution: Use 3-bit odd function to generate even parity bit
- Use 4-bit odd function to check for errors in even parity codes
- Operation: $(X,Y,Z) = (0,0,1)$ gives $(X,Y,Z,P) = (0,0,1,1)$ and $E = 0$
- If Y changes from 0 to 1 between generator and checker, then $E = 1$ indicates an error

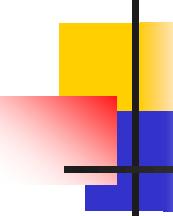




PARITY BIT

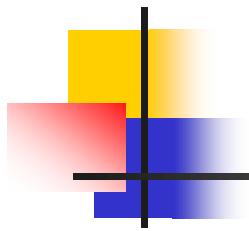
What Parity Bit is?

- The parity generating technique is one of the most widely used error detection techniques for the data transmission.
- **Parity bit** is added to the word containing data in order to make number of 1s either even or odd. Thus it is used to detect errors , during the transmission of binary data



PARITY GENERATOR AND CHECKER

- A **parity generator** is a combinational logic circuit that **generates the parity bit** in the transmitter. On the other hand, a circuit that **checks the parity** in the receiver is called **parity checker**.
- The sum of the data bits and parity bits can be even or odd . In **even parity**, the added parity bit will make the total **number of 1s an even amount** whereas in **odd parity** the added parity bit will make the total **number of 1s odd amount**.



EVEN PARITY GENERATOR



In **even parity** bit scheme, the parity bit is '**0**' if there are **even number of 1s** in the data stream and the parity bit is '**1**' if there are **odd number of 1s** in the data stream.

Even Parity Generator

3-bit message			Even parity bit generator (P)
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

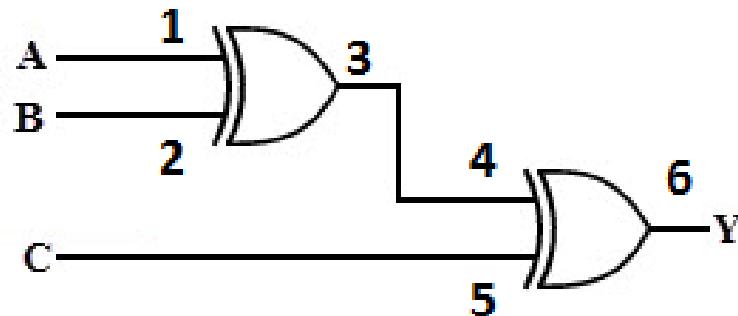
AB	AB	AB	AB	AB
$\bar{A}B$	$\bar{A}B$	$\bar{A}B$	$\bar{A}B$	$\bar{A}B$
00	01	11	10	10
0	2	5	4	4
1	1	3	7	5

$$P = \overline{A} \overline{B} C + \overline{A} B \overline{C} + A \overline{B} \overline{C} + A B C$$

$$= \overline{A} (\overline{B} C + B \overline{C}) + A (\overline{B} \overline{C} + B C)$$

$$= \overline{A} (B \oplus C) + A (\overline{B} \oplus \overline{C})$$

$$P = A \oplus B \oplus C$$



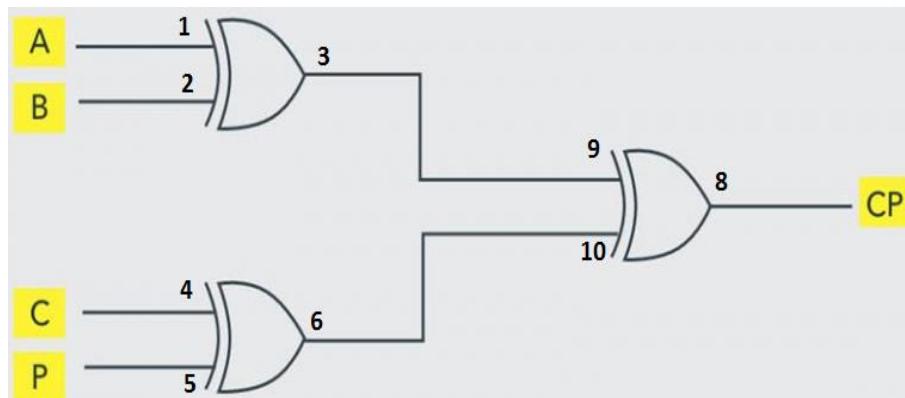
Even Parity Detector (Checker)

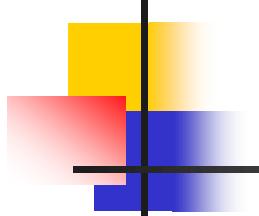
4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

AB

$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}P$	0	(1) ₄	12
$\bar{C}P$	1	5	(1) ₈
CP	3	(1) ₇	15
CP	2	6	(1) ₁₁

$$\begin{aligned}
 F &= \bar{A}\bar{B}(\bar{C}P + CP) + \bar{A}B(\bar{C}P + CP) + AB(\bar{C}P + CP) + A\bar{B}(\bar{C}P + CP) \\
 &= \bar{A}\bar{B}(C \oplus P) + \bar{A}B(\overline{C \oplus P}) + AB(\overline{C \oplus P}) + A\bar{B}(\overline{C \oplus P}) \\
 &= (C \oplus P)(\bar{A}\bar{B} + AB) + (\overline{C \oplus P})(\bar{A}B + A\bar{B}) \\
 &= (C \oplus P)(\overline{A \oplus B}) + (\overline{C \oplus P})(A \oplus B) \\
 &= C \oplus P \oplus A \oplus B \quad (\text{Assume } C = C \oplus P \text{ and } F = A \oplus B) \\
 &= E \oplus F \\
 F &= A \oplus B \oplus C \oplus P
 \end{aligned}$$





ODD PARITY GENERATOR

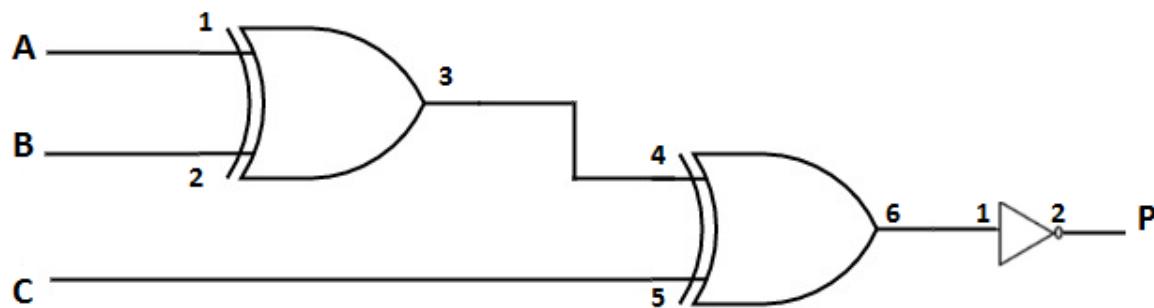
In **odd parity** bit scheme, the parity bit is '**1**' if there are **even number of 1s** in the data stream and the parity bit is '**0**' if there are **odd number of 1s** in the data stream.

Odd Parity Generator

3-bit message			Odd parity bit generator (P)
A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

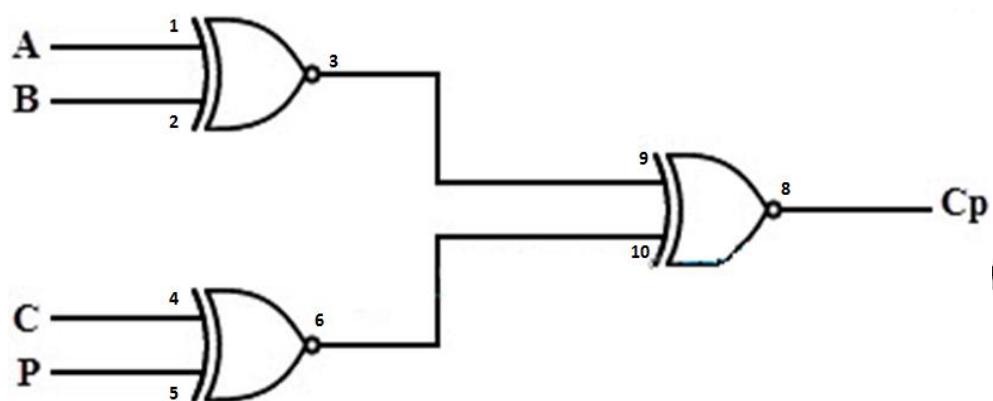
C	AB	00	01	11	10
0	(1) ₀	2	(1) ₆	4	
1	1	(1) ₃	7	(1) ₅	

$$\begin{aligned}
 P &= \overline{A} \ \overline{B} \ \overline{C} + \overline{A} \ B \ C + A \ B \ \overline{C} + \overline{A} \ B \ C \\
 &= \overline{A} (\overline{B} \ \overline{C} + B \ C) + B (A \ \overline{C} + \overline{A} \ C) \\
 &= \overline{A} (\overline{B} \oplus C) + A (B \oplus C) \\
 P &= \underline{\underline{A \oplus B \oplus C}}
 \end{aligned}$$



Odd Parity Detector (Checker)

4-bit received message				Parity error check C_p
A	B	C	P	
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1



AB

$\bar{A}\bar{B}$	$\bar{A}B$	AB	$A\bar{B}$
$\bar{C}P$	0	4	12
$\bar{C}P$	1	5	9
CP	3	7	15
$\bar{C}P$	2	6	10

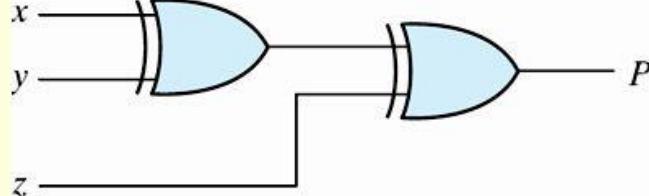
$$\begin{aligned}
 F &= \bar{A}\bar{B}(\bar{C}\oplus P) + \bar{A}B(\bar{C}\oplus \bar{P}) + AB(\bar{C}\oplus P) + A\bar{B}(\bar{C}\oplus \bar{P}) \\
 &= \bar{A}\bar{B}(C \oplus P) + \bar{A}B(C \oplus \bar{P}) + AB(C \oplus P) + A\bar{B}(C \oplus \bar{P}) \\
 &= (C \oplus P)(\bar{A}\bar{B} + AB) + (C \oplus \bar{P})(\bar{A}B + A\bar{B}) \\
 &= (C \oplus P)(\bar{A} \oplus B) + (C \oplus \bar{P})(A \oplus B) \\
 &= \bar{C} \bar{F} + C F \quad (\text{Assume } \bar{C} = C \oplus P \text{ and } F = A \oplus B) \\
 &= \bar{C} \oplus F \\
 &= C \oplus P \oplus A \oplus B
 \end{aligned}$$

$$F = \overline{A \oplus B \oplus C \oplus P}$$

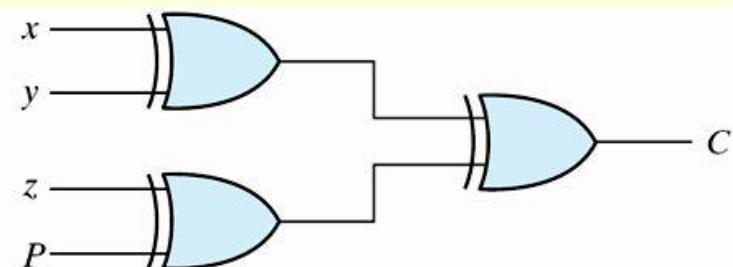
Parity Generation and Checking

■ Parity Generation and Checking

- A parity bit: $P = x \oplus y \oplus z$
- Parity check: $C = x \oplus y \oplus z \oplus P$
 - $C=1$: one bit error or an odd number of data bit error
 - $C=0$: correct or an even # of data bit error



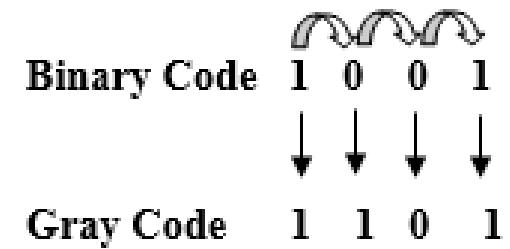
(a) 3-bit even parity generator



(b) 4-bit even parity checker

Figure 3.36 Logic Diagram of a Parity Generator and Checker

Binary to Gray Code Converter



Dec No.	B ₀	B ₁	B ₂	B ₃	G ₀	G ₁	G ₂	G ₃	Dec No.
0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	1
2	0	0	1	0	0	0	1	1	3
3	0	0	1	1	0	0	1	0	2
4	0	1	0	0	0	1	1	0	6
5	0	1	0	1	0	1	1	1	7
6	0	1	1	0	0	1	0	1	5
7	0	1	1	1	0	1	0	0	4
8	1	0	0	0	1	1	0	0	12
9	1	0	0	1	1	1	0	1	13
10	1	0	1	0	1	1	1	1	15
11	1	0	1	1	1	1	1	0	14
12	1	1	0	0	1	0	1	0	10
13	1	1	0	1	1	0	1	1	11
14	1	1	1	0	1	0	0	1	9
15	1	1	1	1	1	0	0	0	8

Binary to Gray Code Converter

Kmap for G_0 :

$B_0 B_1$	$\bar{B}_0 \bar{B}_1$	$\bar{B}_0 B_1$	$B_0 B_1$	$B_0 \bar{B}_1$
$\bar{B}_2 \bar{B}_3$	0	4	1 ₁₂	1 ₈
$\bar{B}_2 B_3$	1	5	1 ₁₃	1 ₉
$B_2 B_3$	3	7	1 ₁₅	1 ₁₁
$B_2 \bar{B}_3$	2	6	1 ₁₄	1 ₁₀

$$G_0 = B_0$$

Kmap for G_1 :

$B_0 B_1$	$\bar{B}_0 \bar{B}_1$	$\bar{B}_0 B_1$	$B_0 B_1$	$B_0 \bar{B}_1$
$\bar{B}_2 \bar{B}_3$	0	1 ₄	1 ₁₂	1 ₈
$\bar{B}_2 B_3$	1	1 ₅	1 ₁₃	1 ₉
$B_2 B_3$	3	1 ₇	1 ₁₅	1 ₁₁
$B_2 \bar{B}_3$	2	1 ₆	1 ₁₄	1 ₁₀

$$G_1 = \bar{B}_0 B_1 + B_0 \bar{B}_1$$

$$G_1 = B_0 \oplus B_1$$

Kmap for G_2 :

$B_0 B_1$	$\bar{B}_0 \bar{B}_1$	$\bar{B}_0 B_1$	$B_0 B_1$	$B_0 \bar{B}_1$
$\bar{B}_2 \bar{B}_3$	0	1 ₄	1 ₁₂	8
$\bar{B}_2 B_3$	1	1 ₅	1 ₁₃	9
$B_2 B_3$	1 ₃	7	1 ₁₅	1 ₁₁
$B_2 \bar{B}_3$	1 ₂	6	1 ₁₄	1 ₁₀

$$G_2 = B_1 \bar{B}_2 + \bar{B}_1 B_2$$

$$G_2 = B_1 \oplus B_2$$

Kmap for G_3 :

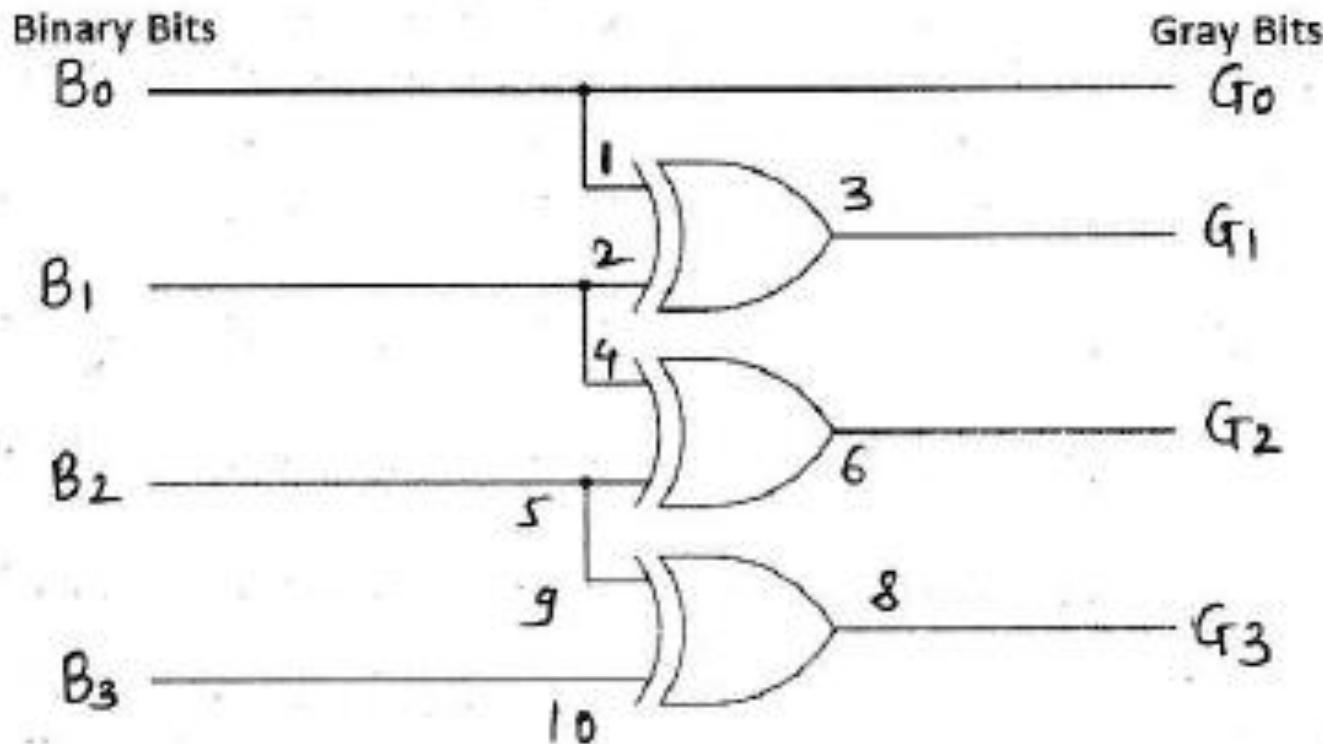
$B_0 B_1$	$\bar{B}_0 \bar{B}_1$	$\bar{B}_0 B_1$	$B_0 B_1$	$B_0 \bar{B}_1$
$\bar{B}_2 \bar{B}_3$	0	4	12	8
$\bar{B}_2 B_3$	1	1 ₅	1 ₁₃	1 ₉
$B_2 B_3$	3	7	15	11
$B_2 \bar{B}_3$	1 ₂	6	1 ₁₄	1 ₁₀

$$G_3 = \bar{B}_2 B_3 + B_2 \bar{B}_3$$

$$G_3 = B_2 \oplus B_3$$

Binary to Gray Code Converter

Circuit Diagram:



Gray Code 1 1 1 0
 ↓↑↓↑↓
 Binary Code 1 1 0 1

Gray to Binary Code Converter

Dec No.	G ₀	G ₁	G ₂	G ₃	B ₀	B ₁	B ₂	B ₃
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	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	0	0
7	0	1	1	1	0	1	0	1
8	1	0	0	0	1	1	1	1
9	1	0	0	1	1	1	1	0
10	1	0	1	0	1	1	0	0
11	1	0	1	1	1	1	0	1
12	1	1	0	0	1	0	0	0
13	1	1	0	1	1	0	0	1
14	1	1	1	0	1	0	1	1
15	1	1	1	1	1	0	1	0

Gray to Binary Code Converter

Kmap for B_0

		G ₀ G ₁			
		̄G ₀ ̄G ₁	̄G ₀ G ₁	G ₀ G ₁	G ₀ ̄G ₁
		0	4	1 ₁₂	1 ₈
		1	5	1 ₁₃	1 ₉
		3	7	1 ₁₅	1 ₁₁
		2	6	1 ₁₄	1 ₁₀

$$B_0 = G_0$$

Kmap for B_1

		G ₀ G ₁			
		̄G ₀ ̄G ₁	̄G ₀ G ₁	G ₀ G ₁	G ₀ ̄G ₁
		0	1 ₄	1 ₁₂	1 ₈
		1	1 ₅	1 ₁₃	1 ₉
		3	1 ₇	1 ₁₅	1 ₁₁
		2	1 ₆	1 ₁₄	1 ₁₀

$$B_1 = \overline{G_0}G_1 + G_0\overline{G_1}$$

$$B_1 = G_0 \oplus G_1$$

Gray to Binary Code Converter

Kmap for B_2

$G_2 G_3 \backslash G_0 G_1$	$\bar{G}_0 \bar{G}_1$	$\bar{G}_0 G_1$	$G_0 \bar{G}_1$	$G_0 G_1$
$\bar{G}_2 \bar{G}_3$	0	1 ₄		1 ₅
$\bar{G}_2 G_3$	1	1 ₅	1 ₃	1 ₉
$G_2 G_3$	1 ₃		1 ₅	1 ₁₁
$G_2 \bar{G}_3$	1 ₁	1 ₆	1 ₄	1 ₁₀

$$B_2 = \bar{G}_0 \bar{G}_1 G_2 + \bar{G}_0 G_1 \bar{G}_2 \\ G_0 G_1 G_2 + G_0 \bar{G}_1 \bar{G}_2$$

$$B_2 = G_0 \oplus G_1 \oplus G_2$$

Kmap for B_3

$G_2 G_3 \backslash G_0 G_1$	$\bar{G}_0 \bar{G}_1$	$\bar{G}_0 G_1$	$G_0 \bar{G}_1$	$G_0 G_1$
$\bar{G}_2 \bar{G}_3$	0	1 ₄		1 ₈
$\bar{G}_2 G_3$	1 ₁		1 ₃	1 ₉
$G_2 G_3$	3	1 ₇	1 ₅	1 ₁₁
$G_2 \bar{G}_3$	1 ₂	1 ₆	1 ₄	1 ₁₀

$$B_3 = \bar{G}_0 \bar{G}_1 \bar{G}_2 G_3 + \bar{G}_0 \bar{G}_1 G_2 \bar{G}_3 + \bar{G}_0 G_1 \bar{G}_2 \bar{G}_3 \\ + \bar{G}_0 G_1 G_2 G_3 + G_0 \bar{G}_1 \bar{G}_2 G_3 + G_0 G_1 \bar{G}_2 \bar{G}_3 \\ + G_0 \bar{G}_1 G_2 \bar{G}_3 + G_0 \bar{G}_1 G_2 G_3$$

$$B_3 = G_0 \oplus G_1 \oplus G_2 \oplus G_3$$

Gray to Binary Code Converter

Circuit Diagram:

