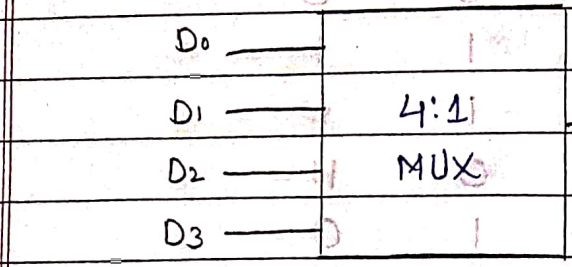
Unit 3

Date
Page

* Multiplexers
* • Not processing
  • Resource sharing only.



Data lines o/p line

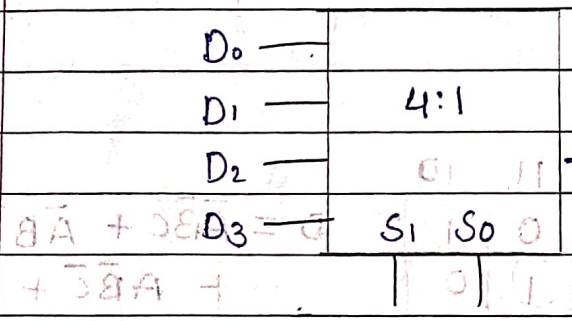
		2 bit
n=2	m=1	1 00 D_0
n=4	m=2	2 01 D_1
n=8	m=3	3 10 D_2
		4 11 D_3

n:1

$n = 2^m$

m = no. of things

n = no. of base.



(key) (door)

A	B	Y = A·B
0	0	0
0	1	0
1	0	0
1	1	1

AND gate

Y=0

Y=B

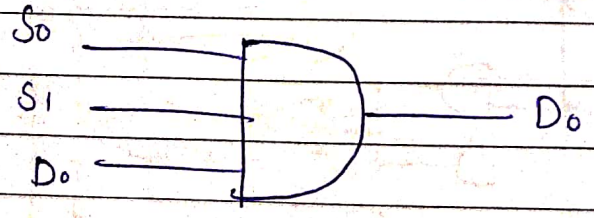
Enable I/P

A

B

Data

* AND gate is enabling gate



Scanned with CamScanner

• Implement given boolean expression using MUX

$f(A,B,C) = A\bar{B} + Bc + \bar{A}\bar{c}$

$= A\bar{B}c + A\bar{B}\bar{c} + ABc + \bar{A}Bc + \bar{A}B\bar{c} + \bar{A}\bar{B}\bar{c}$

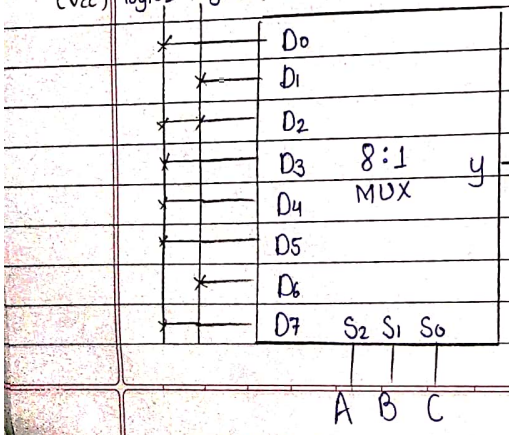
A	B	C	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

$= \Sigma m(0,2,3,4,5,7)$

No. of variables = 3 → No of selectlines

No. of conditions = $2^3 = 8$

∴ 8:1 MUX is required.



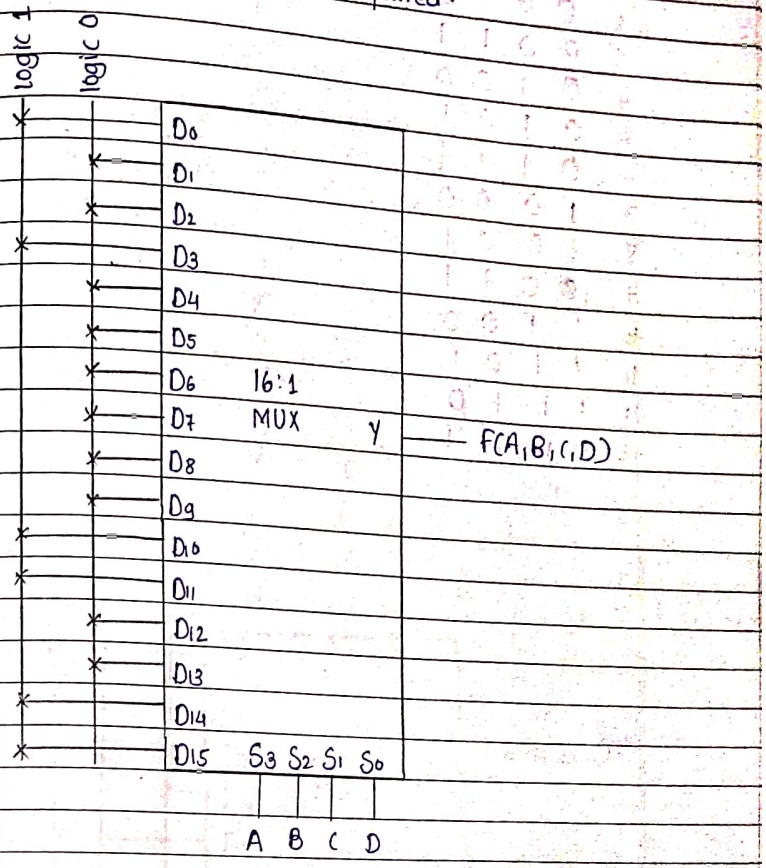
• Implement given boolean expression using MUX

$f(A,B,C,D) = \Sigma m(0,3,10,11,14,15)$

No. of variables = 4 = No. of selectlines.

No. of conditions = $2^4 = 16$

∴ 16:1 MUX is required.

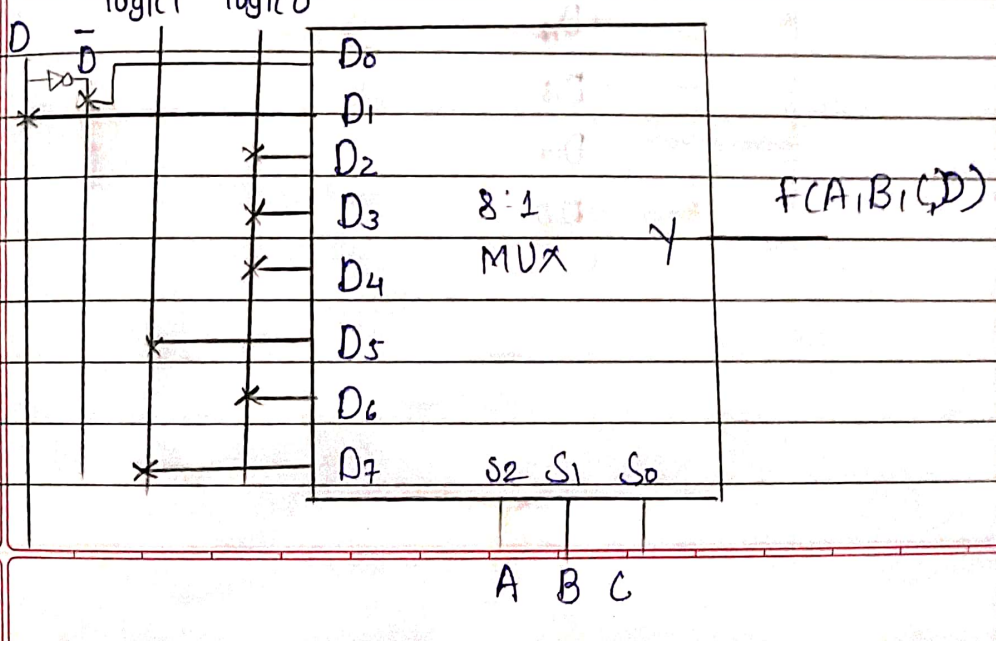


Scanned with CamScanner

Implement given boolean expression using 8:1 MUX

$Y = f(A, B, C, D) = (0, 3, 10, 11, 14, 15)$

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



Scanned with CamScanner

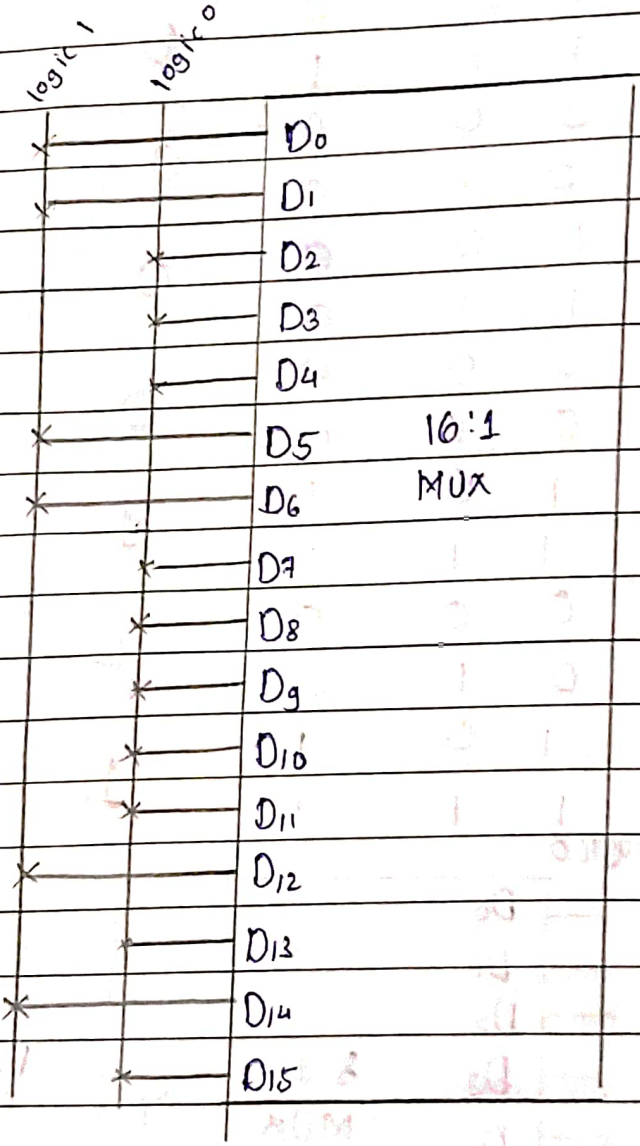
Q. 4 Variable f^n ① 16:1 MUX

$y = f(A, B, C, D) = \sum m(0, 1, 5, 6, 12, 14)$

No. of variables = 4

No. of conditions $= 2^4 = 16$.

∴ 16:1 MUX is required.

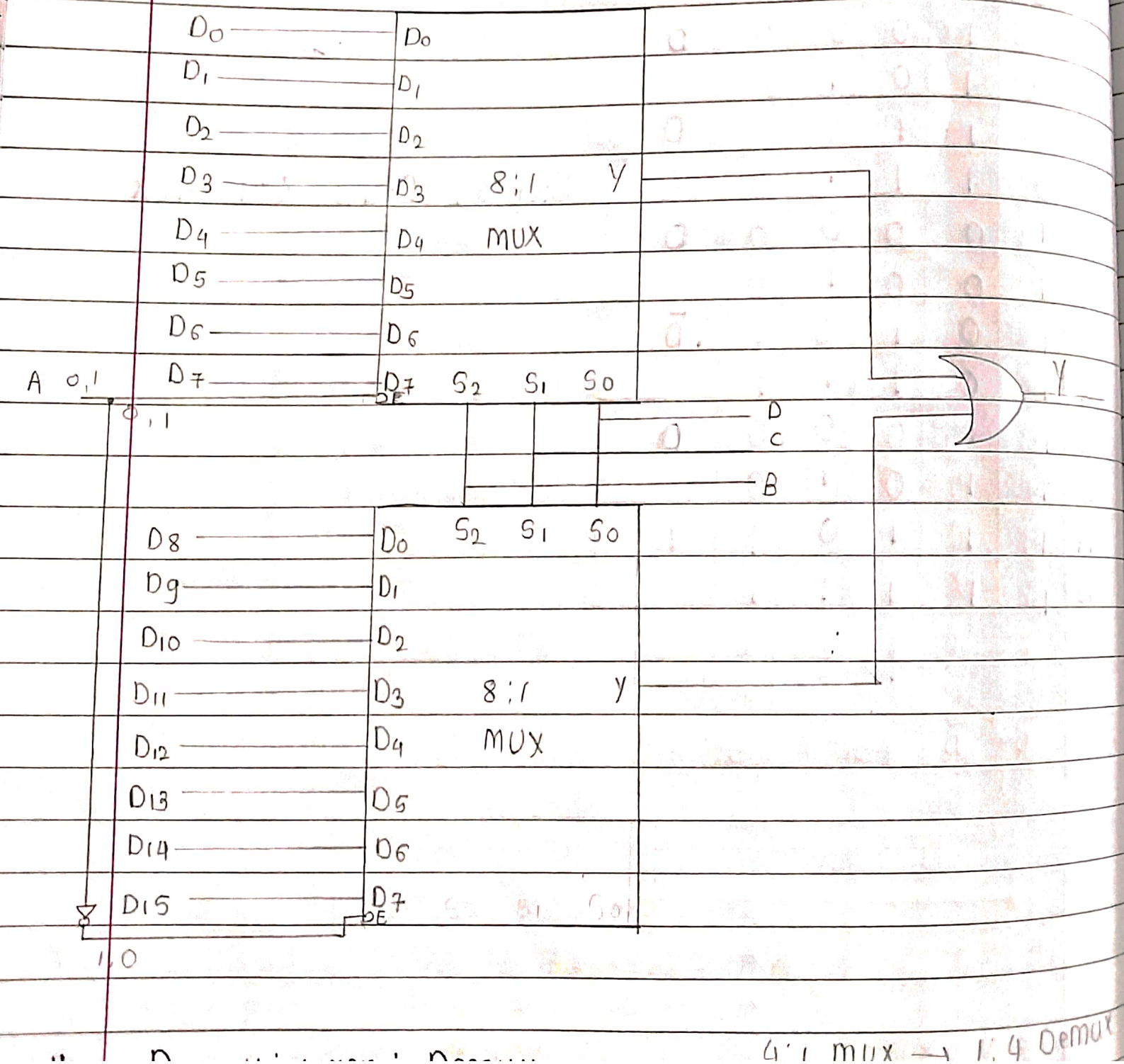


Scanned with CamScanner

Date: 16/10/23 YOUVA

Que 4 variable function ① 16 : 1 MUX .
4 variable function ① 8 : 1 MUX
4 variable function ② 8 : 1 MUX . → 16 : 1 → cascading

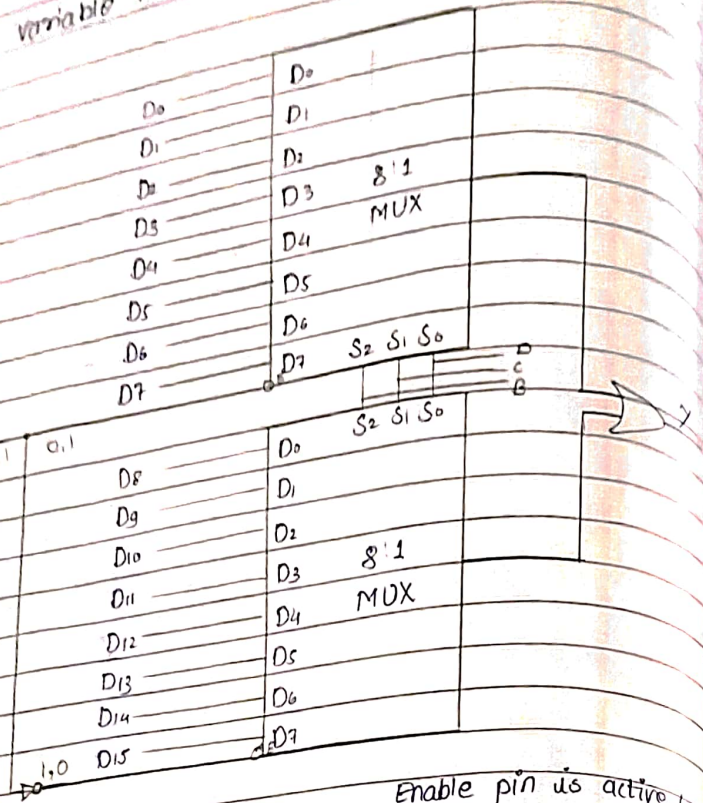
Soln



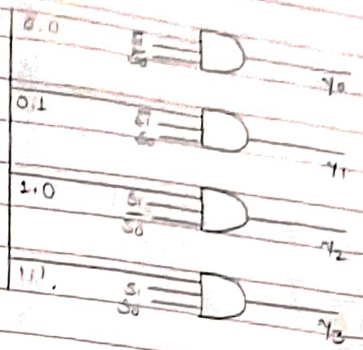
4:1 mux → 1:4 Demux

Scanned with CamScanner

4 variable 1 to 16 8:1 MUX



Demultiplexer Circuit

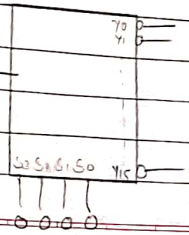


Q. Implement following boolean expressions using De-multiplexer.

1] $y_1 = f_1(A,B,C,D) = \Sigma m(8,9,10,11)$
$y_2 = f_2(A,B,C,D) = \Sigma m(3,5,12,1)$
$y_3 = f_3(A,B,C,D) = \Sigma m(0,1,2,3)$

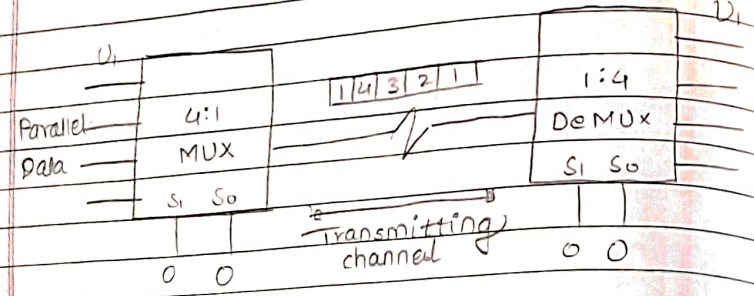
NOTE: • De-Multiplexers are useful to implement multi-objective functions in which the boolean variables are common but the functions are different.

• The outputs of De-MUX's are active low, i.e.

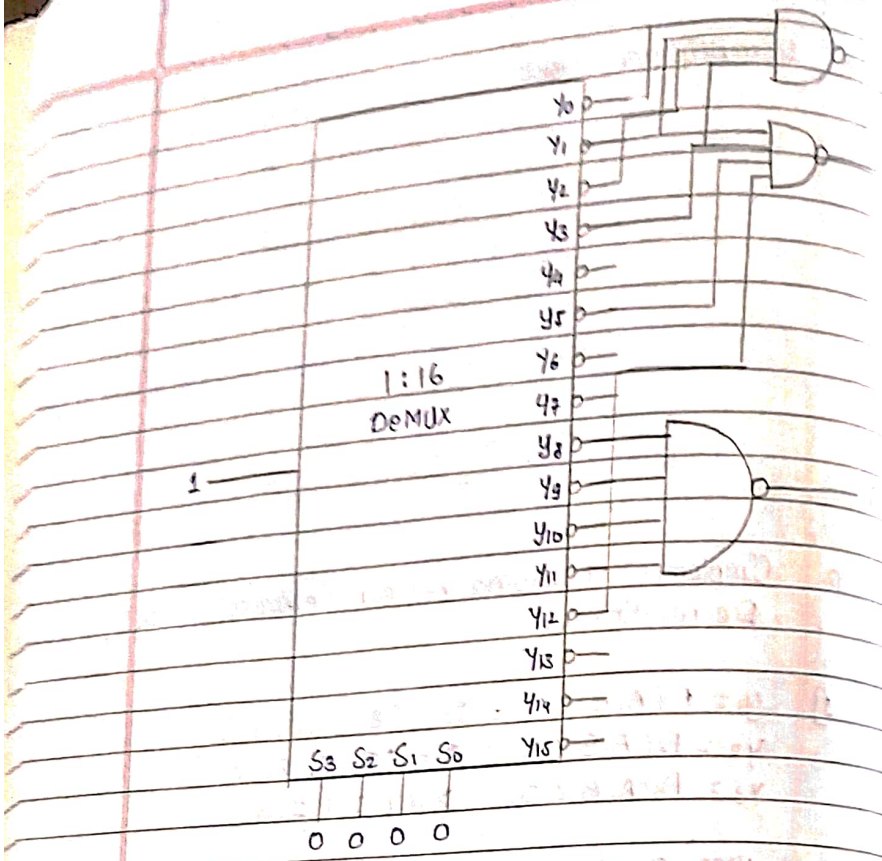


NAND

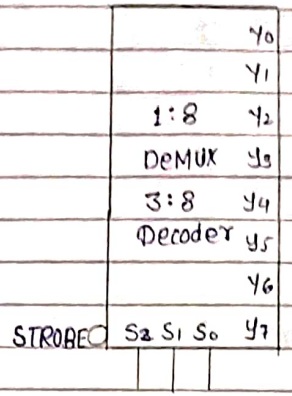
A	B	$Y = \overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0



Scanned with CamScanner

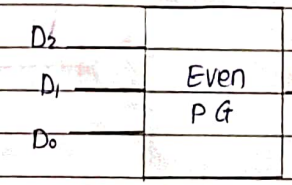


* DeMUX as Decoder



* Parity - can be even or odd.
* Generator
* Checker.

Q. Design 3-bit parity (even) generator



D2	D1	D0	Pe

Scanned with CamScanner

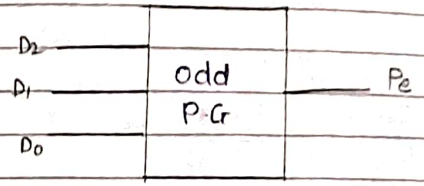
Truth table

D_2	D_1	D_0	P_e
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

D_2 \ D_1D_0	00	01	11	10
0		1		1
1	1		1	

$P_e = D_2 \oplus D_1 \oplus D_0$

Q. Design 3-bit odd parity generator



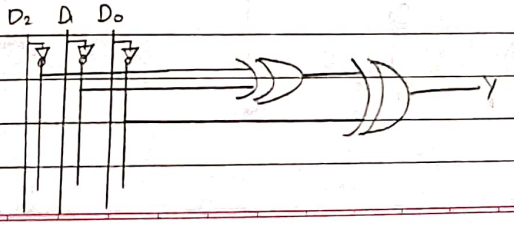
Truth table:

D_2	D_1	D_0	P_e
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

D_2 \ D_1D_0	00	01	11	10
0	1 (0)	(1)	1 (3)	(2)
1	(4)	1 (5)	(7)	1 (6)

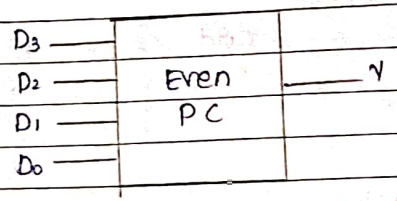
$P_e = \bar{D_2}\bar{D_1}\bar{D_0} + \bar{D_2}D_1D_0 + \bar{D_2}\bar{D_1}D_0 + D_2D_1\bar{D_0}$

$= \bar{D_2} \oplus \bar{D_1} \oplus \bar{D_0}$



Scanned with CamScanner

Q. Design 4-bit parity checker.



D3	D2	D1	D0

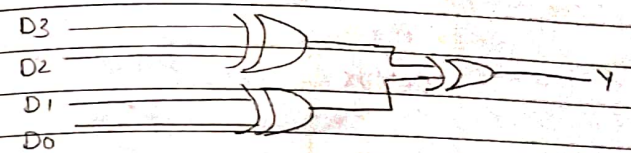
Truth table:

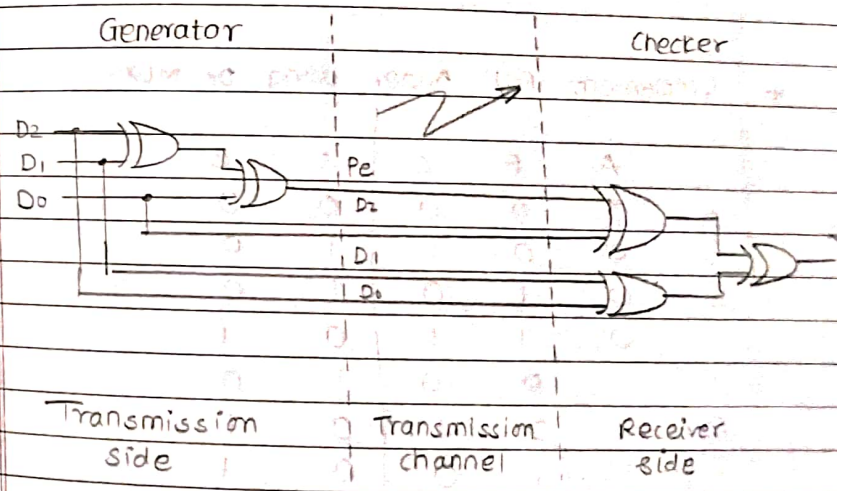
D_3	D_2	D_1	D_0	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

K-map:

D_3D_2 \ D_1D_0	00	01	11	10
00	0	1 1	3	1 2
01	1 4	5	1 7	6
11	12	1 13	15	1 14
10	1 8	9	1 11	10

$Y = D_3 \oplus D_2 \oplus D_1 \oplus D_0$

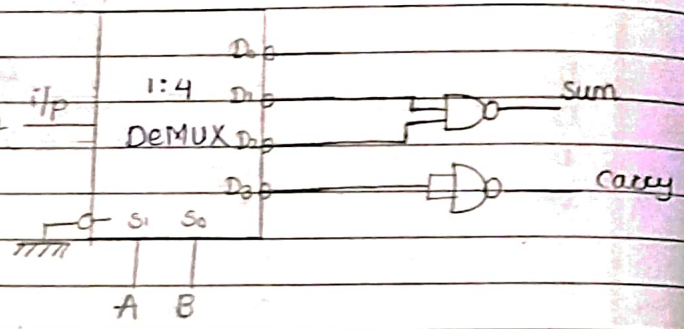


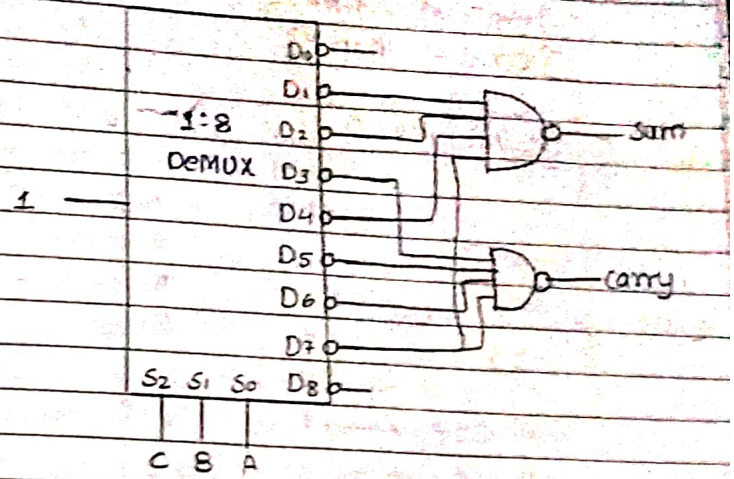


Scanned with CamScanner

* Implement Half Adder using De-MUX

	A	B	S	C
0	0	0	0	0
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1





* Implement Full Adder using De-MUX.

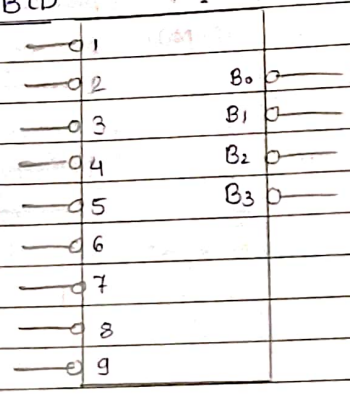
	A	B	C	S	C
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Scanned with CamScanner

* Priority Encoders

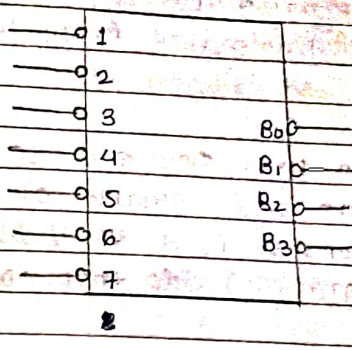
1] Decimal — BCD
2] Octal — BCD

1] Decimal – BCD



Decimal i/p									BCD o/p			
1	2	3	4	5	6	7	8	9	B_3	B_2	B_1	B_0
1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	0
X	0	1	1	1	1	1	1	1	1	1	0	1
X	X	0	1	1	1	1	1	1	1	1	0	0
X	X	X	0	1	1	1	1	1	1	0	1	1
X	X	X	X	0	1	1	1	1	1	0	1	0
X	X	X	X	X	0	1	1	1	1	0	0	1
X	X	X	X	X	X	0	1	1	1	0	0	0
X	X	X	X	X	X	X	0	1	0	1	1	1
X	X	X	X	X	X	X	X	0	0	1	1	0

2] Octal – BCD Priority encoder



Octal i/p							BCD o/p			
1	2	3	4	5	6	7	B_3	B_2	B_1	B_0
1	1	1	1	1	1	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	0
X	0	1	1	1	1	1	1	1	0	1
X	X	0	1	1	1	1	1	1	0	0
X	X	X	0	1	1	1	1	0	1	1
X	X	X	X	0	1	1	1	0	1	0
X	X	X	X	X	0	1	1	0	0	1
X	X	X	X	X	X	0	1	0	0	0

Scanned with CamScanner

* PLD ⇒ Programmable logic Devices.
  - ROM ⇒ Read only Memory
  - PLA ⇒ Programmable logic Array
  - PAL ⇒ Programmable Array logic.

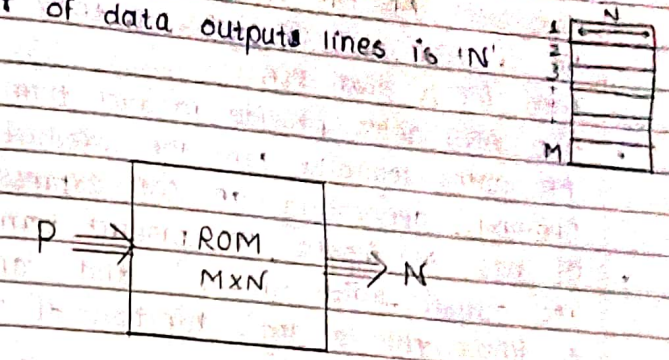
- Programmable logic Device is IC that is user can configure & is capable of implementing logic functions. It is a VLSI (very large scale Integration) chip that contains a regular structure.
  It is programmed by the user to perform the functions required.
- Several advantages are offered.

Advantages:
1] Short design cycle.
2] low development cost.
3] Reduction in board space required.
4] Reduction in power requirement.
5] Design security
6] Compact circuitary
7] High switching speed.
8] Higher densities.

* ROM as PLD

- A Read Only Memory (ROM) is basically a combination circuit and can be used to implement a logic function.

- A ROM of size 'M x N' has 'M' no of locations and 'N' no. of bits can be stored at each location.
- The number of address input is 'P' where $2^P = M$.
  Number of data outputs lines is 'N'.



A 'P' variable, 'N' output logic functions can be implemented using a ROM of size '$2^P \times N$' Since all the possible 2^P minterms are effectively generated.

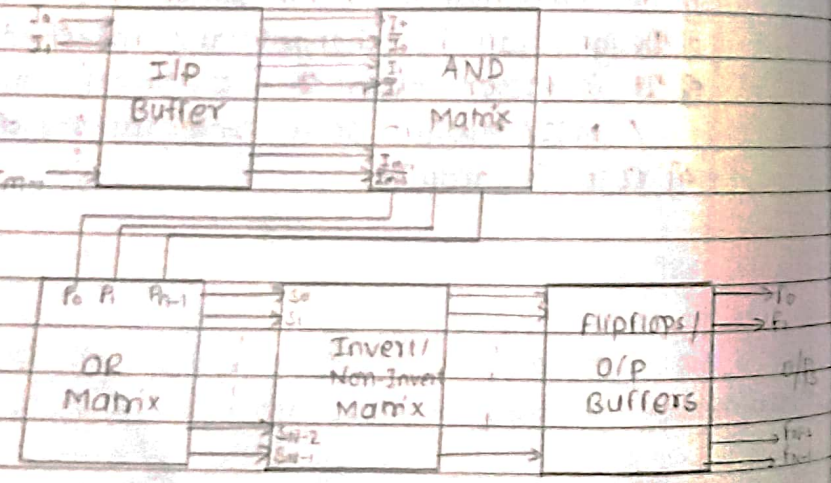
Advantages of ROM
1] No simplification / minimization required.
2] Designs can be changed and modified rapidly.
3] It is faster than SSI (Small scale Integration) / MSI (Medium scale Integration) circuits.
4] Cost is reduced.

Scanned with CamScanner

PLA

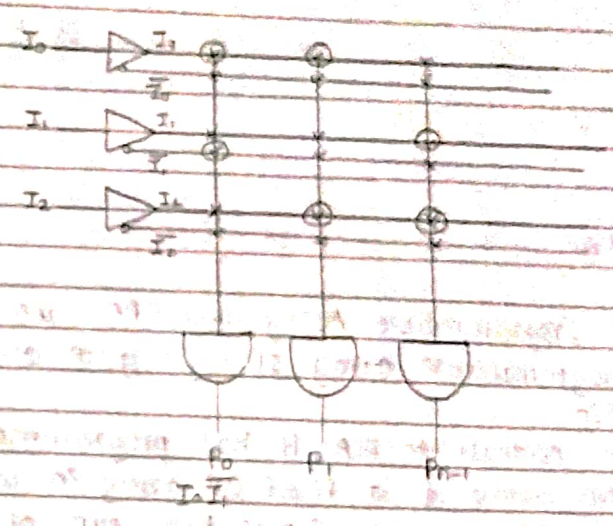
- A PLA (Programmable logic Array) consists of two level AND/OR circuits on a single chip.
- No of AND and OR gates & their inputs are fixed for a given PLA
- The AND gates provide product terms. OR gates logically sum the product terms thereby, generating an SOP expression.
- It has 'M' inputs, 'n' product terms and 'N' outputs with $n < 2^M$ and can be used to implement a logic function of 'M' variables with 'N' outputs
- Since all of the possible '2^M' minterms are marked not available.
Therefore, logic minimization is required to accomodate a given logic function

Block Diagram



- Input buffer produces inverted as well as non-inverted output. for generating the required product term the unwanted links are open through the method of programming.

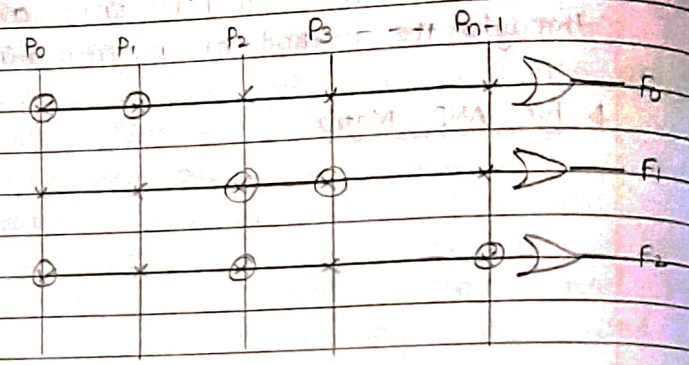
For AND Matrix



$F_0 = I_0\overline{I_1} + I_0 I_2 + I_1 I_2$

Scanned with CamScanner

▷ For OR Matrix:

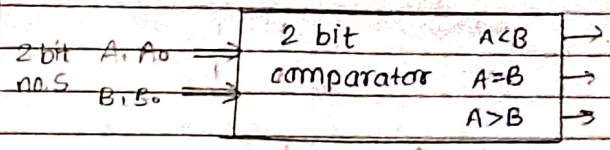


* PAL

- A Programmable Array logic (PAL) is a programmable array of logic gate on a single chip
- In contrast to PLA, it has programmable AND array & a fixed OR array in which each OR gate gets inputs from sum of the AND gates i.e. all the AND gate outputs are not connected to any OR gate.
- Input and output circuits of PAL are similar to those of PLA

The no. of fusible links in a PAL is a product of ~~two~~ '2M' & 'n' where 'M' is the no. of input variables & 'n' is the no. of product terms.

* Design 2-bit comparator



Truth table:

2 bit No.s				output		
A		B				
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	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	0	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	1	0
1	0	1	1	1	0	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	0	1	0

Scanned with CamScanner

- Design (Solve All)
- Realize / Implement (Any one circuit Diag)

A<B

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$= \bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0$

A=B

A_1A_0 \ B_1B_0	00	01	11	10
00	1			
01		1		
11			1	
10				1

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

$= (\overline{A_0 \oplus B_0}) \cdot (\overline{A_1 \oplus B_1})$

A>B:

A_1A_0 \ B_1B_0	00	01	11	10
00				
01	1			
11	1	1		1
10	1	1		1

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

Scanned with CamScanner