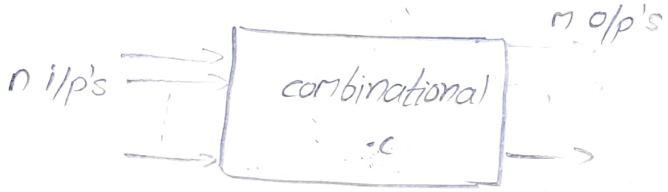
3. Combinational circuits

Digital circuits → combinational c
→ sequential c

Combinational c: interconnection of logic gates whose o/p depends only on inputs (current i/p)

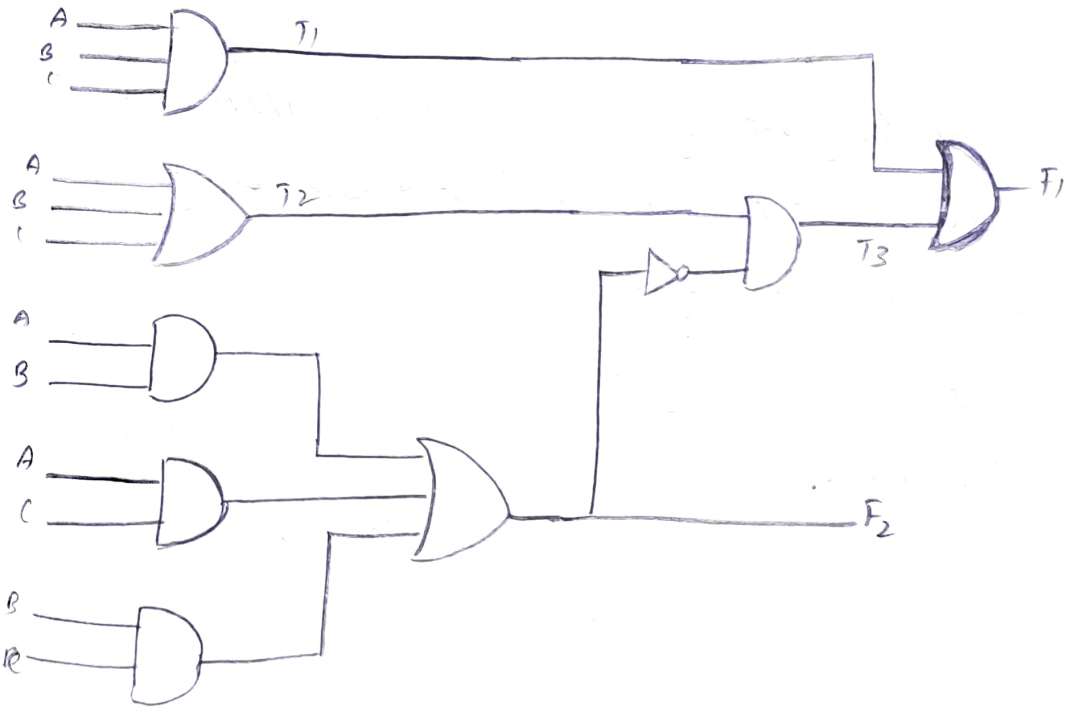


A combinational circuit consists of logic gates whose o/p's at any time are determined from only the present combination of inputs.

Combinational circuits consists of interconnection of logic gates. It reacts to the values of the signals at their inputs and produce the value of the o/p signal, transforming binary information from given input data to the required output data.

standard cc :- adder, sub, multiplexer, decoder, encoder

Analysis Procedure:-



$T_1 = ABC \qquad T_2 = A+B+C \qquad T_3 = T_2$

$F_2 = AB + AC + BC \qquad F_1 = T_1 T_3$

$F_1 = ABC + (T_2 F_2')$

$= ABC + ((A+B+C)(AB+AC+BC)')$

$= ABC + ((A+B+C)(A'+B')(A'+C')(B'+C'))$

$= ABC + (A+B+C)(A'+B'C')(B'+C')$

$= ABC + (A+B+C)(A'B'+A'C'+B'C')$

$F_1 = ABC + AB'C' + A'BC' + A'B'C$

$F_2 = AB+AC+BC$ → carry; → sum

A	B	C	F_2	T_1	T_2	F_2'	T_3	F_1
0	0	0	0	0	0	1	0	0
0	0	1	0	0	1	1	1	1
0	1	0	0	0	1	1	1	1
0	1	1	1	0	1	0	0	0
1	0	0	0	0	1	1	1	1
1	0	1	1	0	1	0	0	0
1	1	0	1	0	1	0	0	0
1	1	1	1	1	1	0	0	1

The analysis of the given logic circuit can be performed manually by finding the boolean functions or truth table or by using the computer simulation program.

Step-1:- Make sure that the given circuit is combinational ckt.

Step-2:- Obtain the o/p boolean functions in terms of i/p variables

A) label all the gate outputs and determine the boolean function for each gate o/p until the o/p's of circuit are obtained

B) by repeated substitution of previously defined functions obtain the o/p boolean functions in terms of i/p variables.

Design procedure

i/p specification → o/p logic circuit (c.c)

1. inputs and outputs (label)
2. T-T define relationship b/w i/p & o/p
3. k-map & write boolean function for each o/p
4. Draw logic circuit

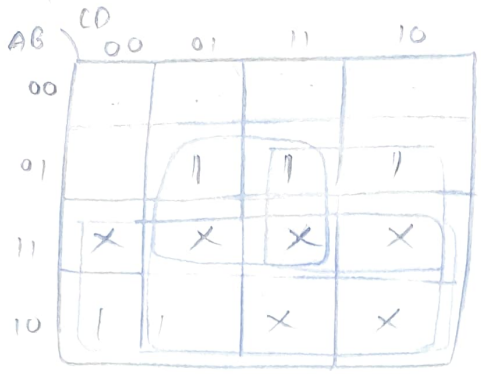
↓

L.C (c.c)

Design the logic circuit to convert the binary coded decimal no (BCD) to excess 3 code for the decimal digits.

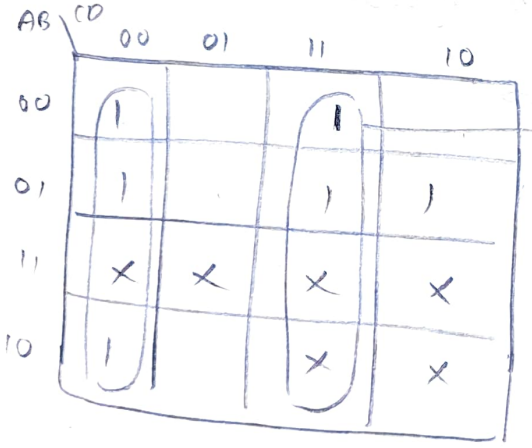
4 i/p variables — 4 o/p variables

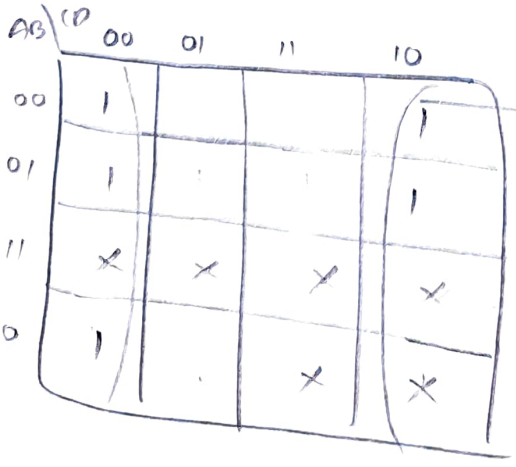
i/p BCD				o/p excess			
A	B	C	D	w	x	y	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0



$W = A + BC + BD$

$x = B'C + B'D + BC'D'$





$z = D'$

$w = A + BC + BD$

$= A + B(C+D)$

$y = CD + C'D'$

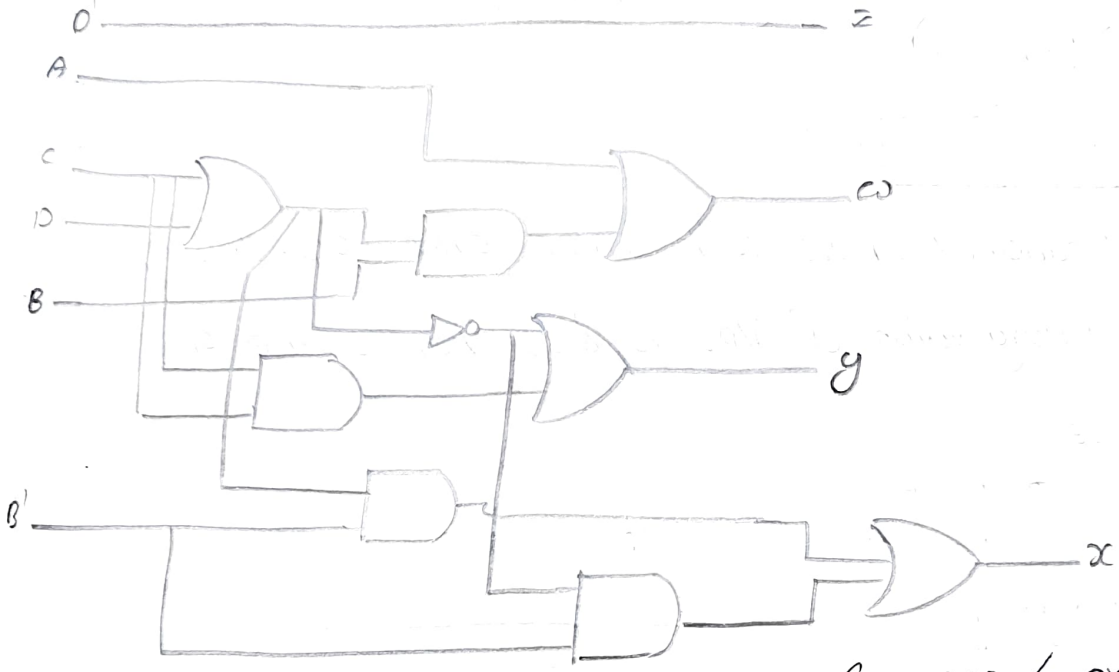
$= CD + (C+D)'$

$x = B'C + B'D + BC'D'$

$= B'(C+D) + B'(C+D)'$

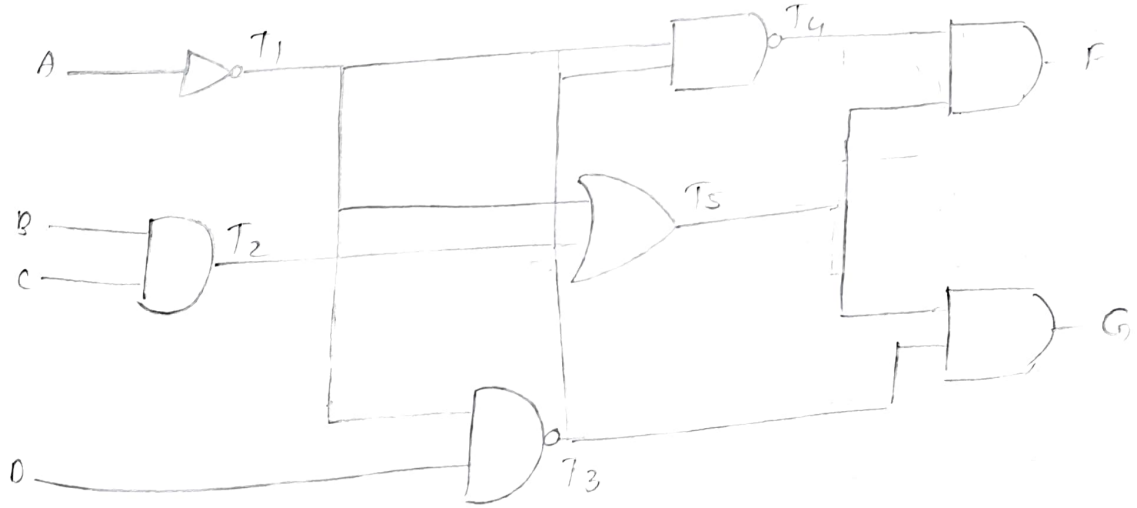
7-AND 3-OR	3-NOT

4-AND 4-OR 1-NOT	3-NOT



logic circuit for the code conversion from BCD to excess 3.

2) Obtain the simplified boolean expression for the output F & G in terms of input variable in the following circuit.



$T_1 = \bar{A}$ $\quad T_2 = BC$ $\quad T_3 = \overline{(\bar{A}+D)} = \bar{\bar{A}}+\bar{D} = A+\bar{D}$

$T_4 = (\overline{T_1 T_3}) \Rightarrow T_5 = T_1 + T_2$

$F = T_4 \cdot \bar{T_5}$ $\quad G = T_3 \bar{T_5}$

$F = T_4 \cdot \bar{T_5}$

$= (T_1 T_3)' (T_1 + \bar{T_2})$

$= (\bar{T_1} + \bar{\bar{T_3}})(\bar{T_1} + \bar{T_2})$

$= (A + \bar{A}D)(\bar{A} + BC)$

$\boxed{F = ABC + \bar{A}D + \bar{A}BCD}$

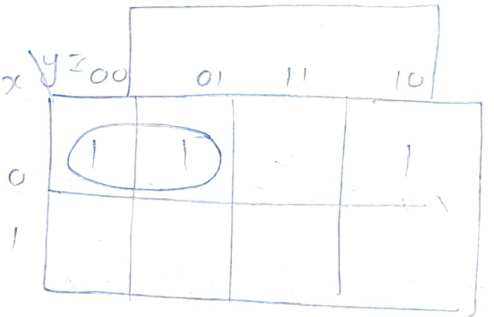
$G = T_3 \cdot \bar{T_5}$

$= (A+\bar{D})(\bar{A}+BC)$

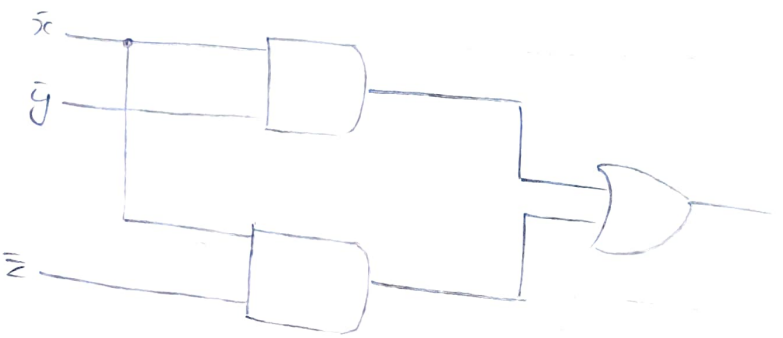
$\boxed{G = ABC + \bar{A}\bar{D} + BC\bar{D}}$

Q) Design a combinational circuit with 3 i/ps and 1 o/p. The o/p is 1 when the binary value of i/ps is less than 3. The o/p is 0 otherwise

x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

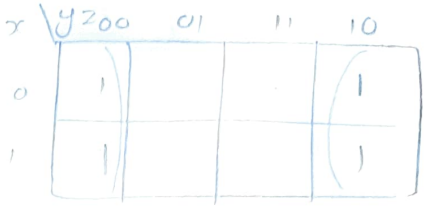


$F = \bar{x}\bar{y} + \bar{x}\bar{z}$

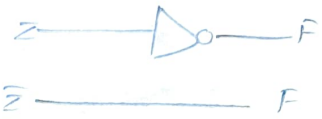


b) The output is 1 when the binary value of i/ps is an even no

x	y	z	F
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

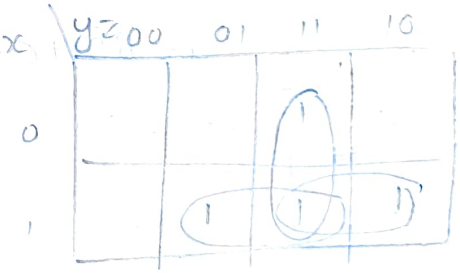


$F = \bar{z}$

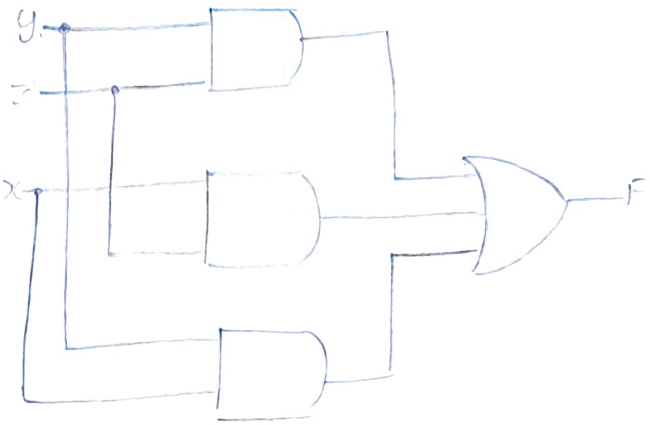


Q) A majority circuit is a combinational circuit whose o/p is 1 if the input variables has more 1's than 0's. The o/p is 0 otherwise. Design a 3 input majority circuit by finding circuits truth table, boolean function and logic diag.

x	y	z	F
0	0	0	0
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	1



$F = yz + xz + xy$

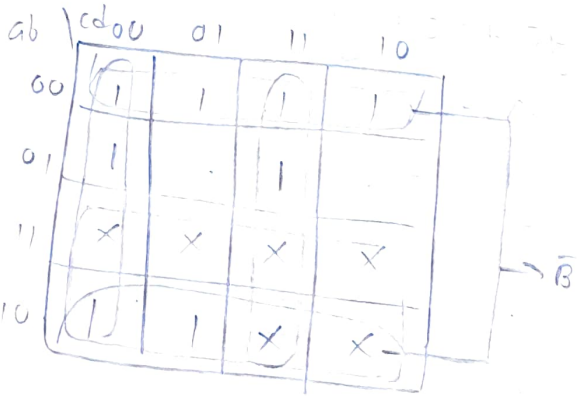


Sol:- 4 i/ps 7 o/ps

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	0	1	1

Case (i) :-

$a = \bar{B}\bar{D} + C + A + BD$

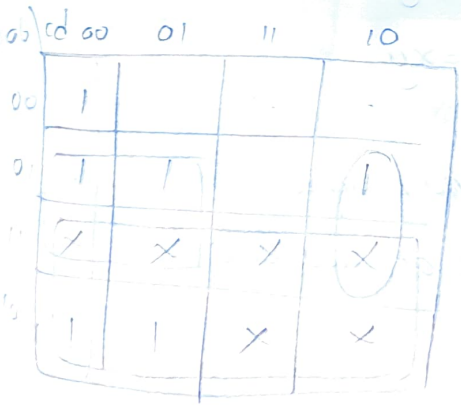


$b = \bar{C}\bar{D} + CD + \bar{B}$

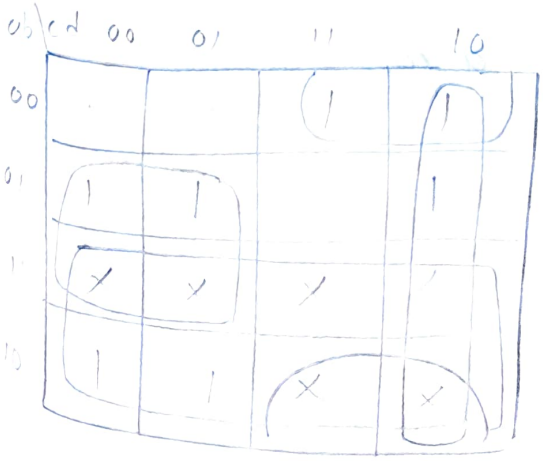
$$c = \bar{C} + D + B$$

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

$$\bar{e} = \bar{B}\bar{D} + C\bar{D}$$



$$f = A + BD + B\bar{C} + \bar{C}\bar{D}$$



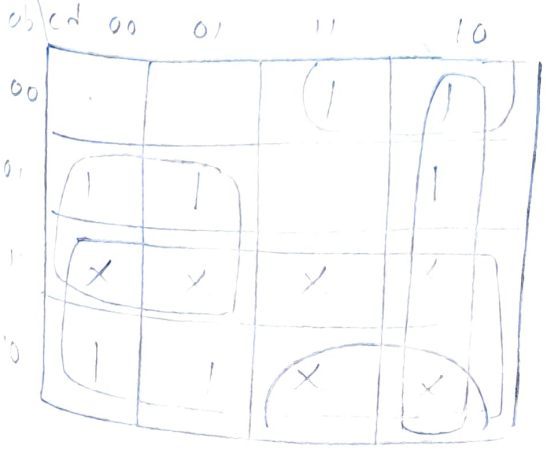
$$g = A + \bar{B}C + B\bar{C} + C\bar{D}$$

$$c = \bar{C} + D + B$$

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

$$e = \bar{B}\bar{D} + C\bar{D}$$

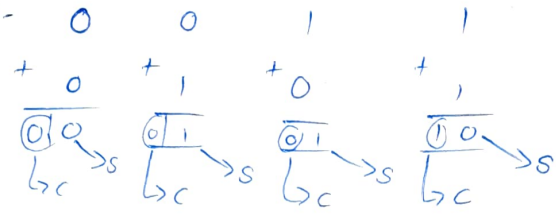
$$f = A + BD + B\bar{C} + \bar{C}\bar{D}$$

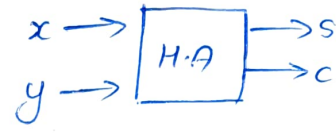


$$g = A + \bar{B}C + B\bar{C} + C\bar{D}$$

Binary Adder-Subtractor

Half adder:-





T-T:-

x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

sum

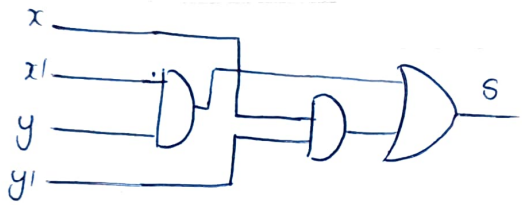
x \ y	0	1
0		1
1	1	

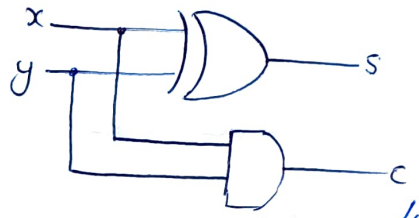
$S = xy' + x'y$

carry

x \ y	0	1
0		
1		1

$C = xy$





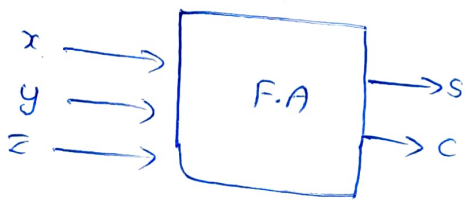
$S = x \oplus y$

$C = xy$

logic circuit for half adder

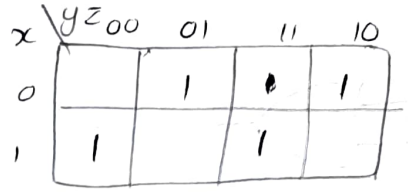
The combinational circuit that performs addition of two bits is called a half adder.

Full adder:- The combinational circuit that performs addition of three bits (2 significant bits, 1 previous carry) is a full adder

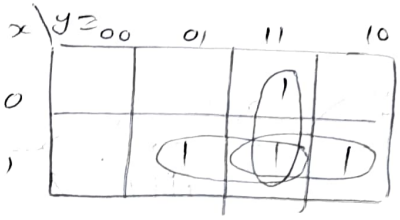


T.T:-

x	y	z	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

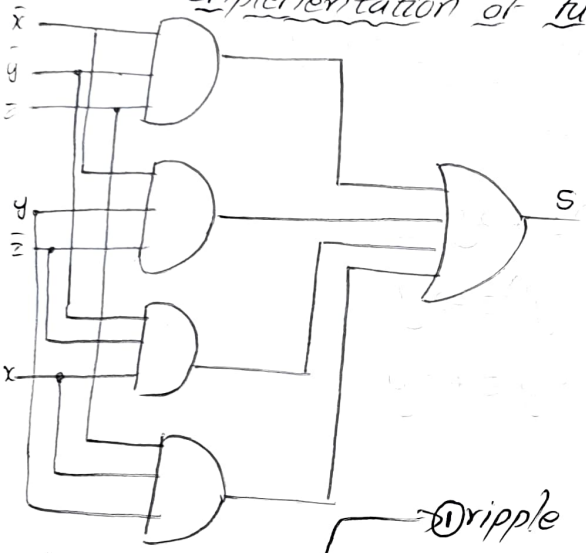


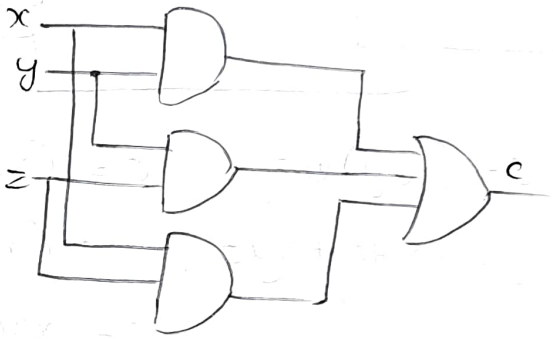
$S = \bar{x}\bar{y}z + \bar{x}y\bar{z} + x\bar{y}\bar{z} + xyz$



$C = xy + yz + zx$

Implementation of full adder in SOP form





→ ripple carry adder

Binary Adder:-

4-Bit Binary

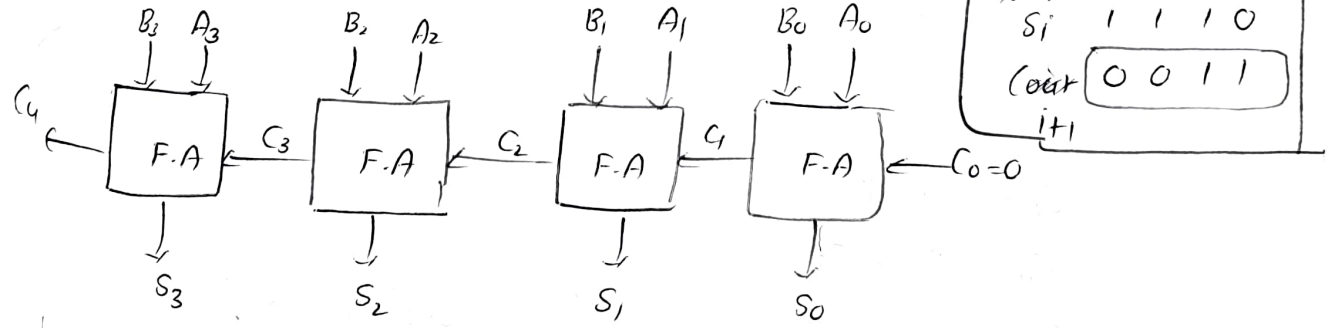
- 3 full adder and 1 half adder

(or)

4 full adder

Ex:-

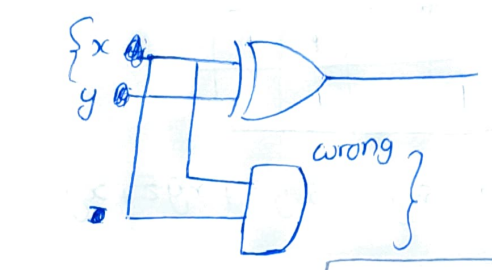
Cin	0	1	1	0
A_i	1	0	1	1
B_i +	0	0	1	1
Cout				
S_i	1	1	1	0
Cout $_{i+1}$	0	0	1	1

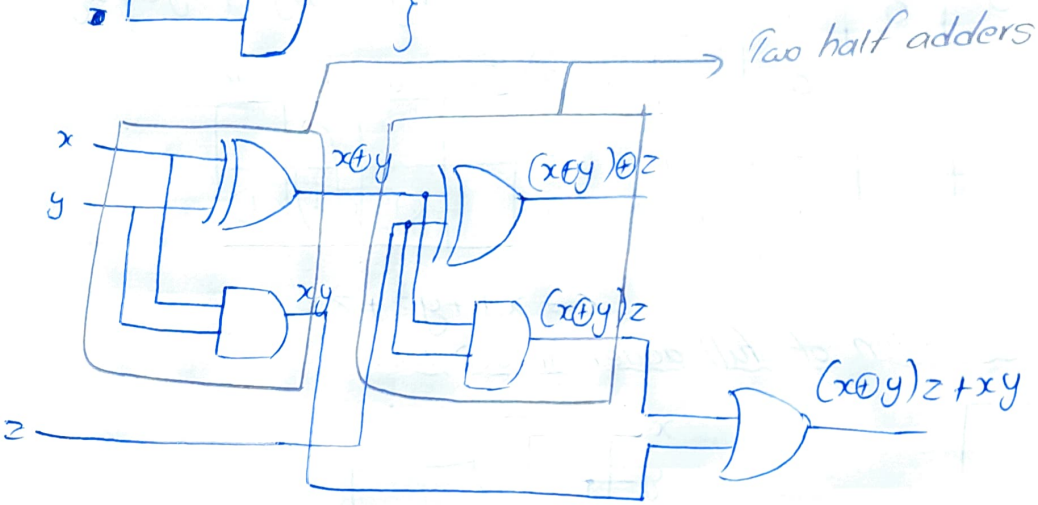


→ Cascading diagram of 4-bit adder

Implementation of Full adder using half adder & OR gate

Ha



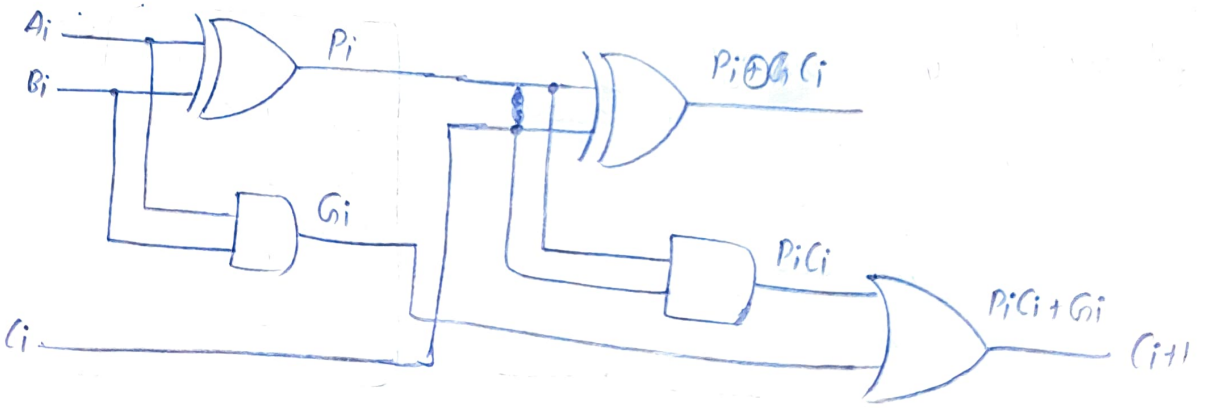


$S = (x \oplus y) \oplus z$ $\quad C = (x \oplus y)z + xy$

$S = xy'z' + x'y'z + x'yz' + xyz$ $\quad = (x'y + xy')z + xy$

$C = x'yz + xy'z + xy$

② carry lookahead adder



$S_i = P_i \oplus C_i$

$C_{i+1} = P_iC_i + G_i$

To get rid of the speed limit of a binary adder which is caused by carry propagation delay we can use the principle of carry lookahead logic

G_i is called carry generate (because it produces 1 when both A_i and B_i are 1 regardless of C_i)

$$G_i = A_i B_i$$

P_i is carry propagator (as it determines whether a carry in the stage i will propagate into stage i+1)

$$P_i = A_i \oplus B_i$$

The output sum and carry are expressed as

$$S_i = C_i \oplus P_i \qquad C_{i+1} = P_i C_i + G_i$$

C_0 - i/p carry

$$C_1 = P_0 C_0 + G_0$$

$$C_2 = P_1 C_1 + G_1$$

$$C_2 = P_1(P_0 C_0) + G_0) + G_1$$

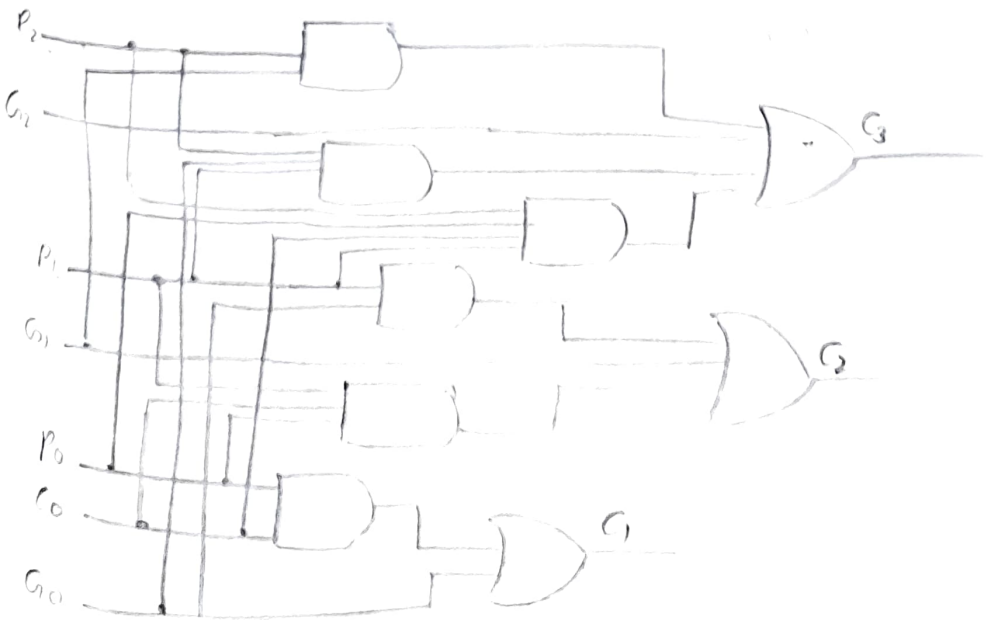
$$\boxed{C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0}$$

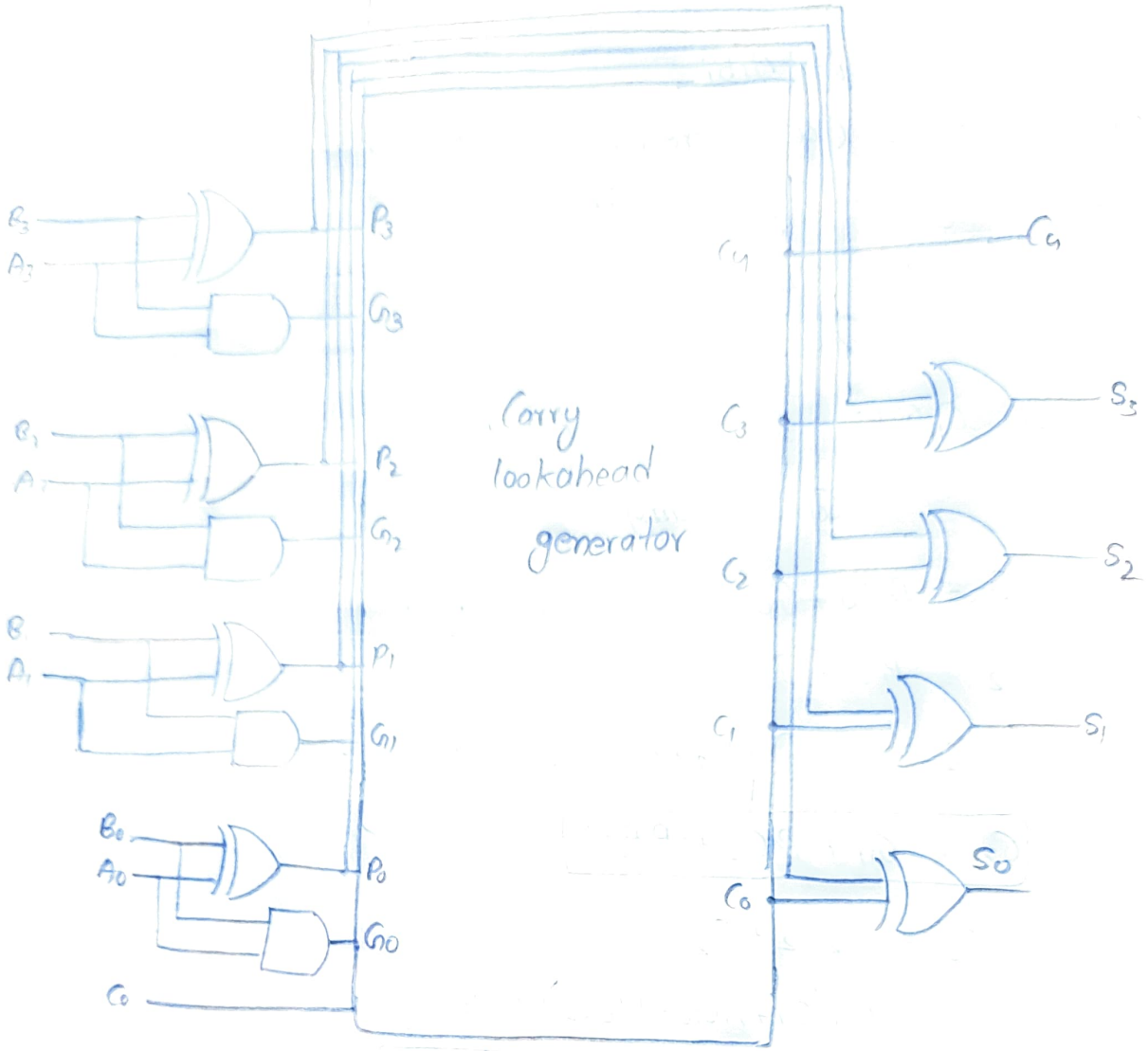
$$C_3 = P_2 C_2 + G_2$$

$$= P_2(G_1 + P_1 G_0 + P_1 P_0 C_0) + G_2$$

$$\boxed{C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0}$$

carry lookahead generator





Using the carry lookahead generator the propagation delay time is reduced because C_3 doesn't have to wait for C_2 & C_1 to propagate. C_3 is propagated at the same time as C_1 & C_2

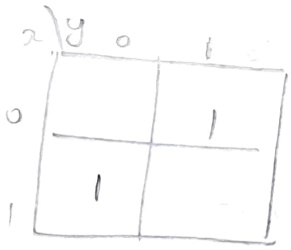
The gain in speed of binary adder is achieved at the expense of additional complexity.

Half subtractor:-

Truth table for Half subtractor:-

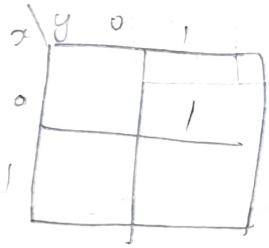
x	y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

B
1 0 (eg:-2)
-1
1

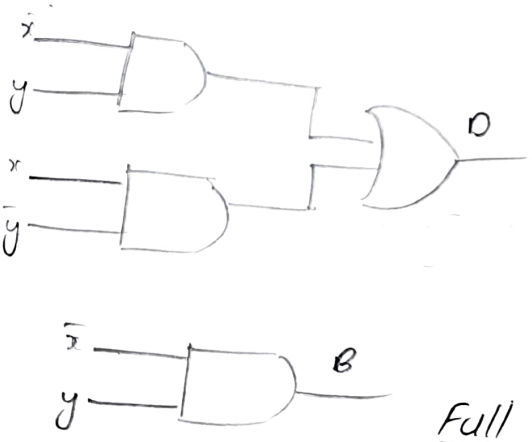


$D = \bar{x}y + x\bar{y}$

$D = x \oplus y$



$B = \bar{x}y$



Full subtractor

Limitation of half subtractor:- half subtractor don't consider borrowing from previous circuit which is needed in real time scenario

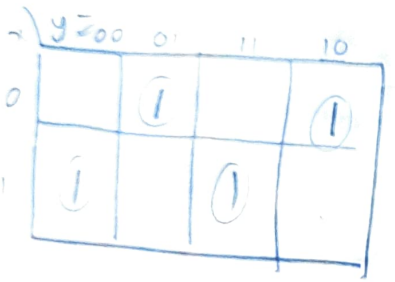
7-1 for F.S

$x - y - z$

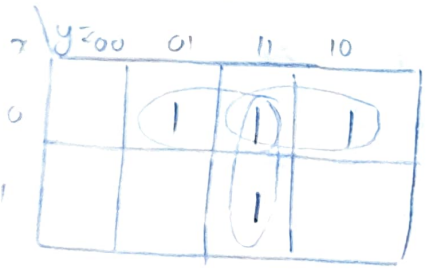
$\boxed{x-(y+z)}$

x	y	z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

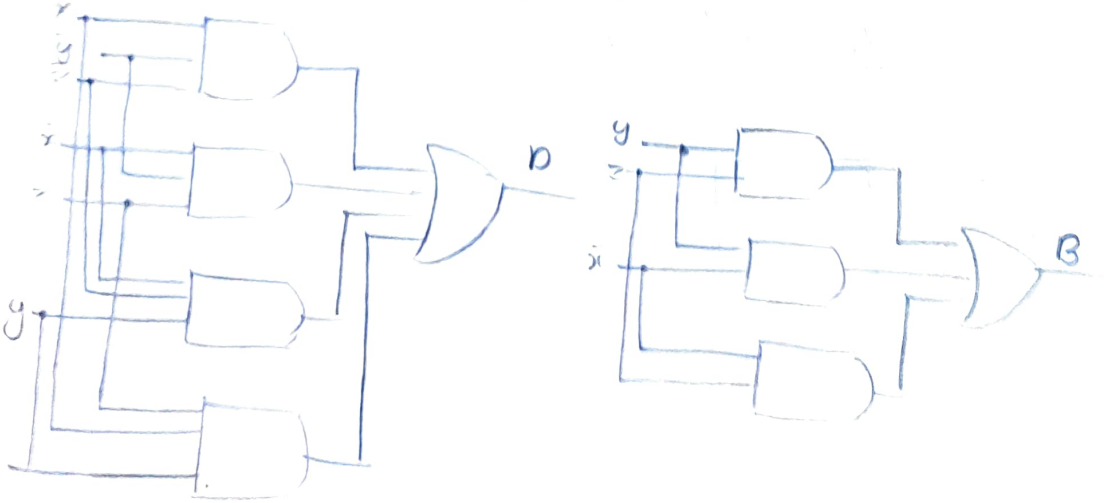
$1-(1+1)$ B $1 - 2 = 3-2 = 1$



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



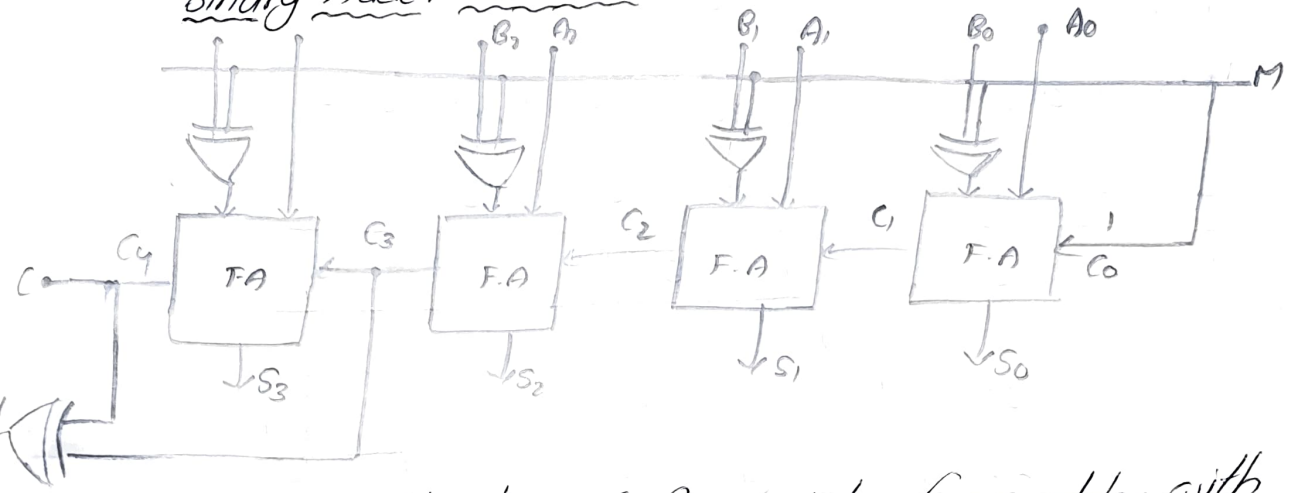
$B = yz + \bar{x}y + \bar{x}z$



Binary Subtractor:- 2 binary nos

$B \oplus 1 = B'$

Binary Adder-Subtractor

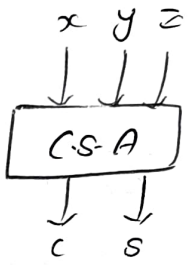


The circuit for subtracting A-B consists of an adder with inverters placed b/w each data input B and corresponding i/p of the full adder. The addition and subtraction operations can be combined into one circuit with one common binary adder by including an exclusive-OR gate with each full adder. The mode i/p M controls the operation. When $M=0$ $B \oplus 0 = B$ & $C_0 = 0$ $\Rightarrow$ The circuit is an adder. When $M=1$ $B \oplus 1 = B'$ & $C_0 = 1$ $\Rightarrow$ The circuit is a subtractor. Exclusive OR with o/p V is used for detecting an overflow.

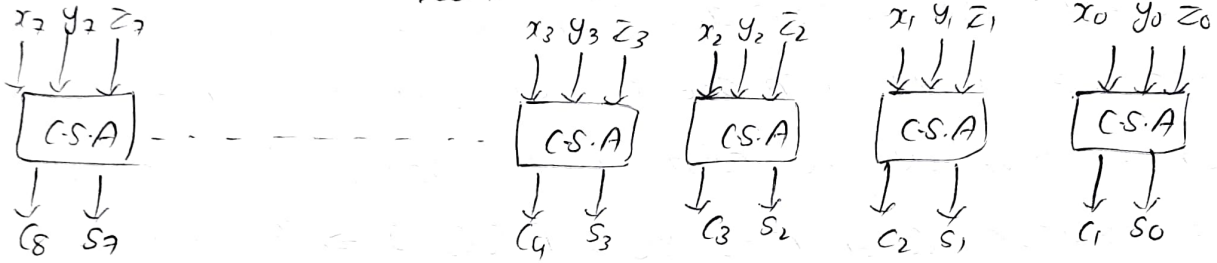
Carry save adder (Multiple operands)

x	1	1	0	1	1
y	0	1	1	0	0
z	0	1	0	0	1
C:	0	1	0	0	1
S:	1	1	1	1	0

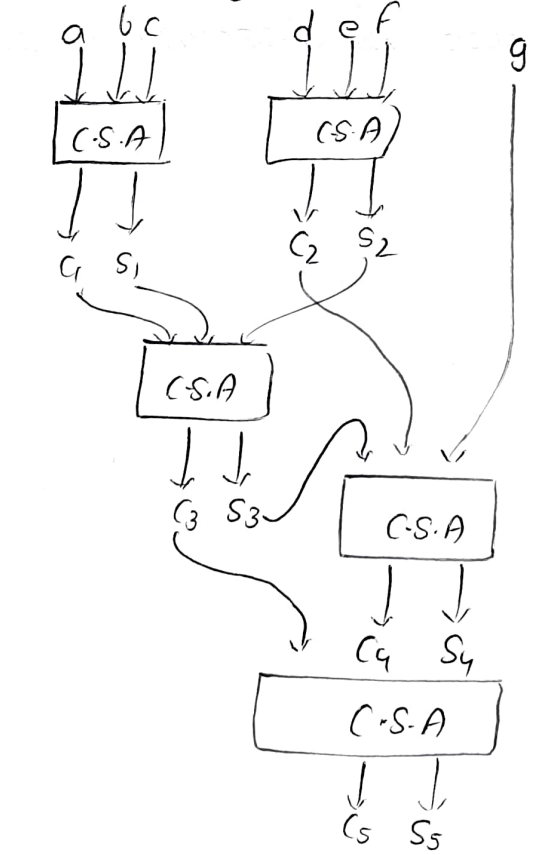
		1	1	1		
S:	1	1	1	1	0	
C:	0	1	0	0	1	(0+1)
	1	1	0	0	0	0



Add 3 - 8bit nos:-



a, b, c, d, e, f, g - 3 bit no



Q) Consider a 2 level logic implementation of 'carry lookahead generator'. Assume that all P_i and G_i are available for the carry generator circuit the 'and' and 'OR' gates can have any no of i/ps. The no of 'and' and 'OR' gates needed to implement the lookahead carry generator for a 4-bit adder with S_3, S_2, S_1, S_0 and C_4 as it's o/p's resp

Ans:- From carry lookahead generator
we find for C_1 1 'and' for C_2 2 'and', for C_3 3 'and', for C_4 4 'and'
So we require 10 'and' and 4 'OR' gates

Note:- For n-bit carry lookahead adder to evaluate all the carry bits it requires

(i) no of 'AND' gates = $\frac{n(n+1)}{2}$

(ii) no of 'OR' gates = n

Time after which the carry becomes available in ripple carry adder = (total no of full adder till it produces ur C_i) × carry propagation delay of full adder

Sum(S_x) time required =
(total no of full adders before full adder produces S_x) × (carry propagation delay of full adder)
+ sum propagation delay of full adder

Q) Propagation delay of exclusive OR and NAND gates ares 20ns, 15ns, & 10ns resp. The worst case delay of 16-bit adder will be ____

Sol. (Hints:- *** carry propagation delay = propagation delay of AND gate + propagation delay of OR gate
propagation delay of sum = prop delay of exclusive-OR) } for carry lookahead

carry propagation delay

$= 16(15+10) = (16 \times 25)$ ns

Q) A 4-bit carry lookahead adder which adds 4-bit nos is designed using AND, OR, NOT, NAND, NOR gates only. Assuming that all the i/ps are available in both complemented and uncomplemented form and delay of the generator — is 1 time unit. What is the propagation delay of the adder. Assume that the carry n/w has been implemented using 2-level AND-OR logic

Sol: 6 time unit

Q) A 16-bit ripple carry adder is realised using 16 identical full adders. The carry propagation delay of each full adder is 12 ns & the sum prop delay is 15 ns. What is the worst case delay of 16-bit adder

Sol: For ripple carry adder

carry prop delay = (tot no of full adder) × (carry prop delay of each full adder)

$= 16 \times 12$

sum prop delay = (no of full adders before full adder performs Sx) × (carry prop delay of each full adder) + sum prop delay of each

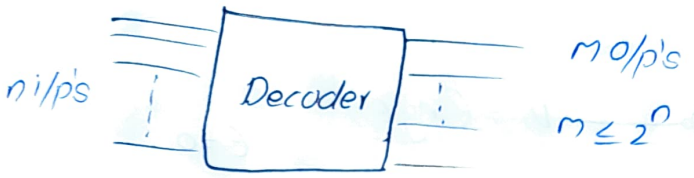
$\Rightarrow (16-1) \times 12 + 15$

$= 15 \times (12) + 15$

$= 195$ (Worst case delay)

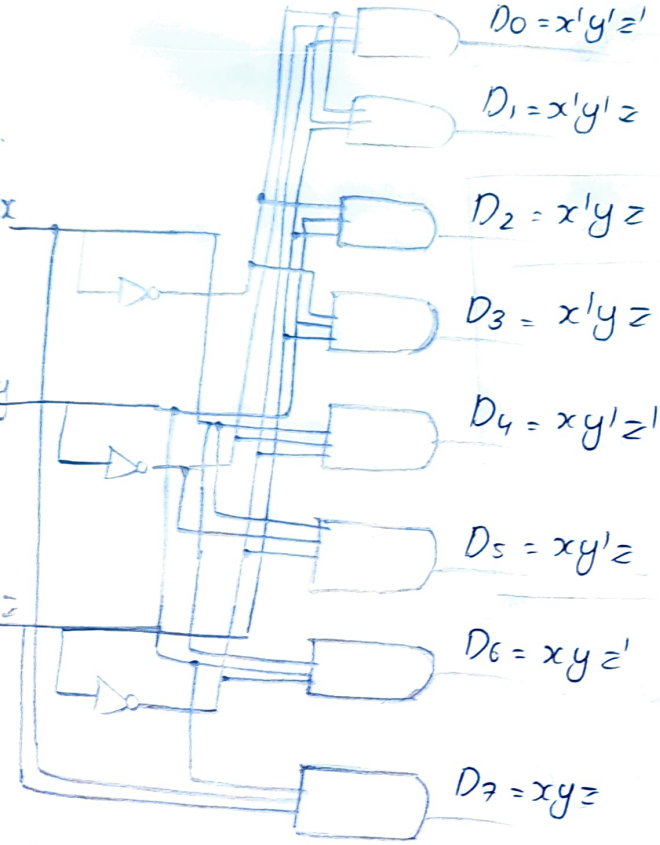
Decoder

is a combinational circuit



two to four decoder (2×4 decoder)

three to eight line decoder (3×8 decoder)



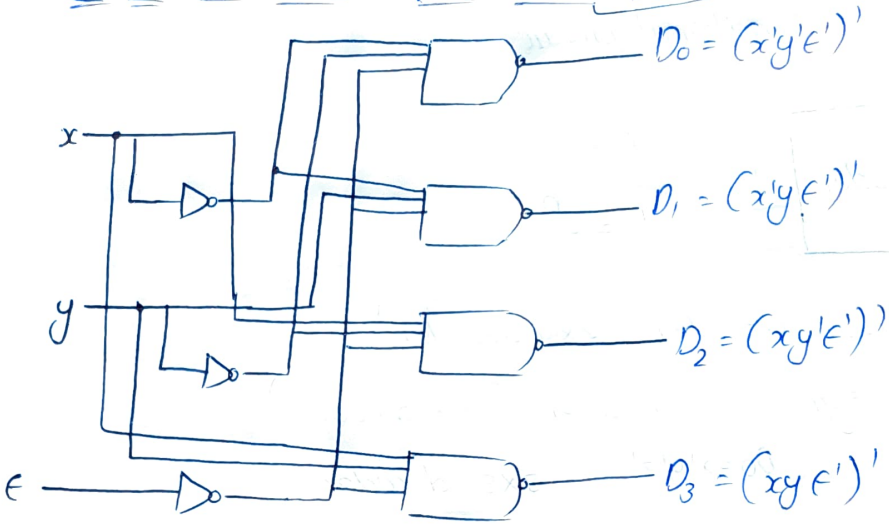
3×8 decoder

Truth table for three-eight line decoder

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

2 to 4 line decoder with enable i/p

Active low signal (if $\epsilon = 0$)



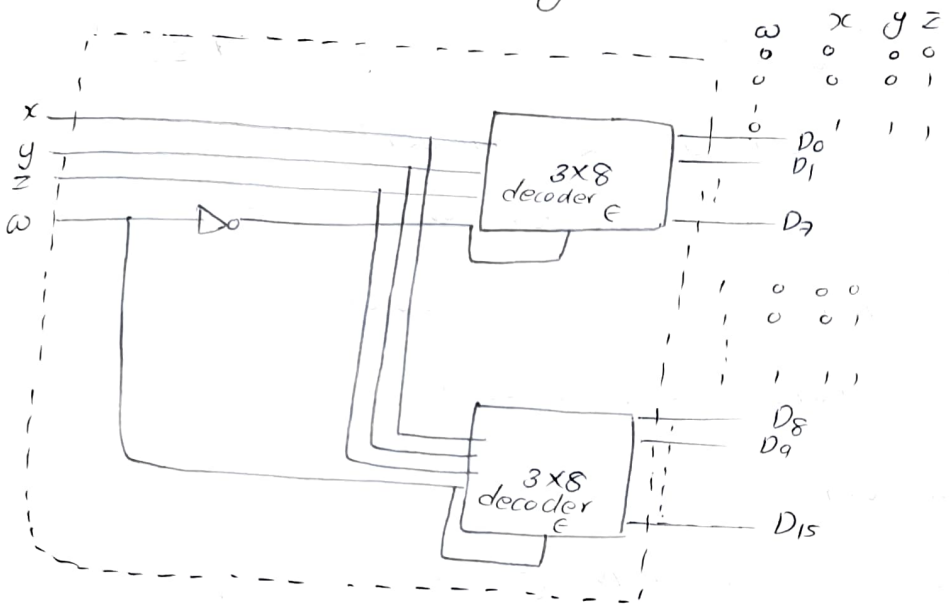
ϵ	x	y	D_0	D_1	D_2	D_3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

enable i/p — active low
└ active high

decoder — 3 to 8 line decoder
↓
(binary to octal decoder)

* A decoder works (or) operates as demultiplexer

4x16 decoder using 3x8 decoder



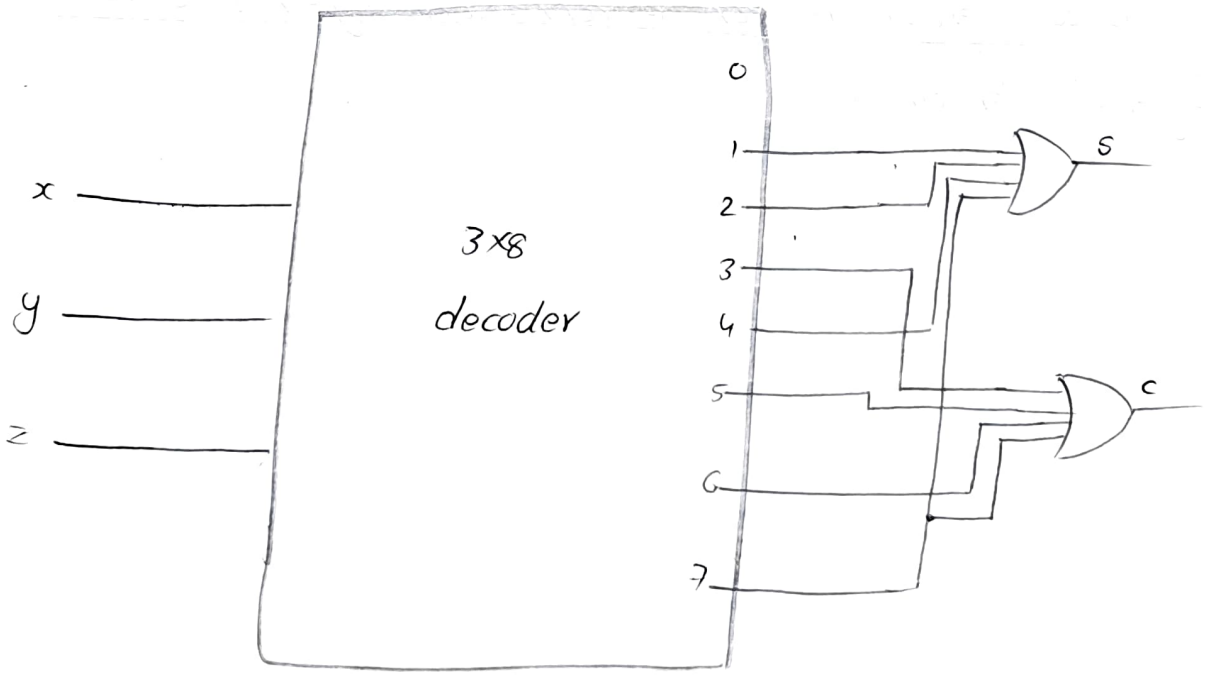
Combination of 4x16 line decoder using two 3x8 line decoder

sum of min terms (SOM)

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

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

x	y	z	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



Full adder using 3x8 line decoder

Note:- ① A decoder is a combinational circuit that converts binary information from n input lines to a max of 2^n unique output lines ($m \leq 2^n$)

② The purpose of decoder is to generate 2^n (or fewer) min terms of n i/p variables such that each combination of i/p will assert a unique o/p Ex:- BCD to 7 segment decoder

3 to 8 line decoder (binary to octal decoder)

BCD to decimal decoder

③ Some decoder are constructed with NAND gates

④ Decoders include one or more enable i/p's to control the circuit operation

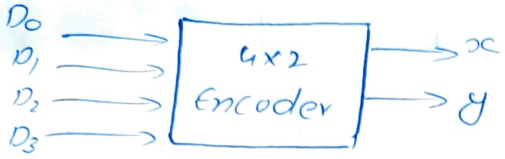
⑤ In general a decoder may operate with complemented or uncomplemented o/p's

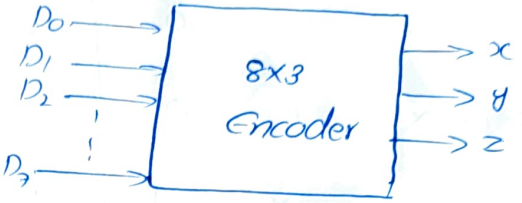
⑥ The enable i/p may be activated with 0 (or) 1 signal

⑦** A decoder with enable i/p can function as a demultiplexer (The circuit that receives information from a single line and directs it to one of the 2^n possible o/p lines)

⑧ Decoders with enable i/p can be connected together to form a larger decoder circuit

Encoder





↓

octal to binary no encoder

Truth table of octal to binary encoder

i/p's								o/p's		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
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

$$x = D_4 + D_5 + D_6 + D_7$$

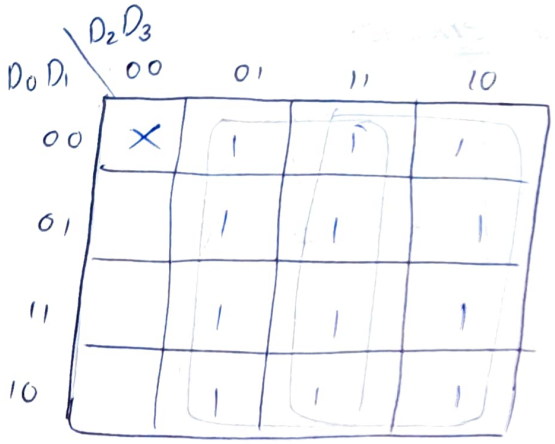
$$y = D_2 + D_3 + D_6 + D_7$$

$$z = D_1 + D_3 + D_5 + D_7$$

* When more than one i/p's are high, unexpected o/p will be resulted. So we go with the priority encoder. We consider the i/p with the higher subscript.

Priority encoder

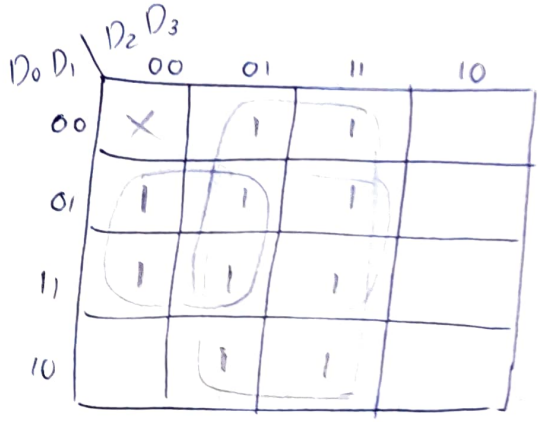
D_0	D_1	D_2	D_3	x	y	V → valid bit
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1



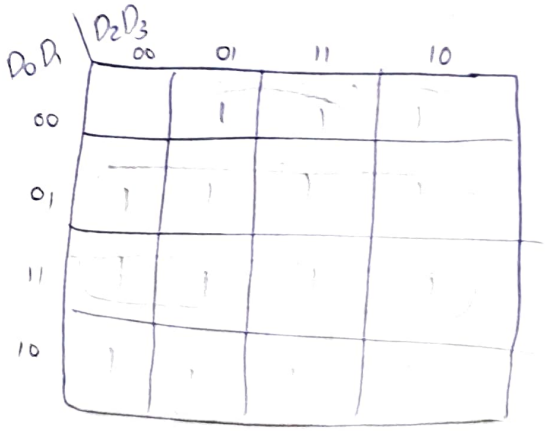
				x	y
0	1	0	0	0	1
1	1	0	0	0	1

				x	y
0	0	1	0	1	0
0	1	1	0	1	0
1	0	1	0	1	0
1	1	1	0	1	0

$$x = D_2 + D_3$$



$$y = D_3 + D_1 D_2' +$$

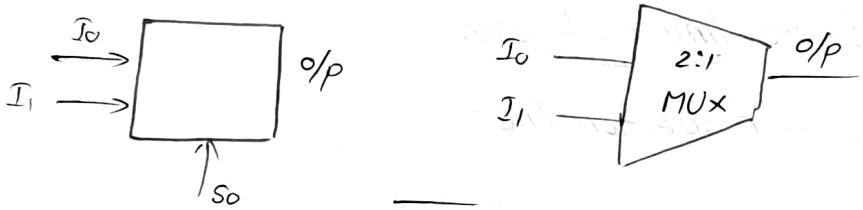


$$V = D_3 + D_2 + D_0 + D_1$$

logic diagram of priority encoder

Multiplexer (MUX)

↳ also called data selector



Encoder:- An encoder is a digital circuit that performs inverse operation of a decoder.

An encoder has 2^n (if fewer) i/p lines and n o/p lines.

The output lines as an aggregate generate the binary code corresponding to the i/p value.

Ex:- octal to binary encoder (8 x 3 encoder)

Priority encoder:- Priority encoder is an encoder combinational circuit that includes priority function. The operation of the priority encoder is such that if two or more i/p's are equal to 1 at the same time, the i/p having the highest priority

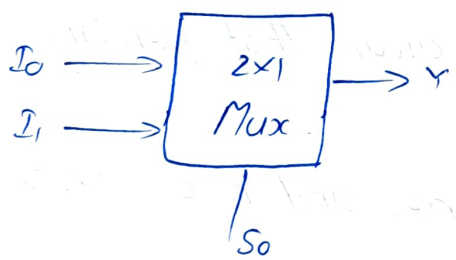
will take precedence. In addition to two o/p's x and y, the circuit has a third o/p designated by V (valid bit indicator) that is set to 1 when one or more i/p's are equal to 1. If all i/p's are 0, there is no valid i/p and (V = 0)

Multiplexer

A multiplexer is the combinational circuit that selects information from one of many i/p lines and directs it to a single o/p line. The selection of a particular i/p line is controlled by a set of selection lines. Normally there are 2^n i/p lines and n selection lines whose bit combinations determine which i/p is selected. A multiplexer is also called a data selector since it selects one of the many i/p's and steers the binary information to o/p line.

two to one line multiplexer

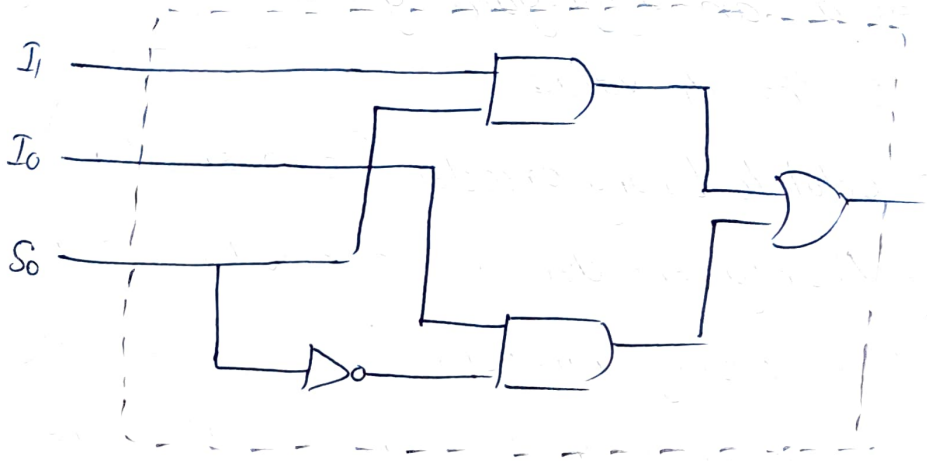
2^n i/p's — n selection lines



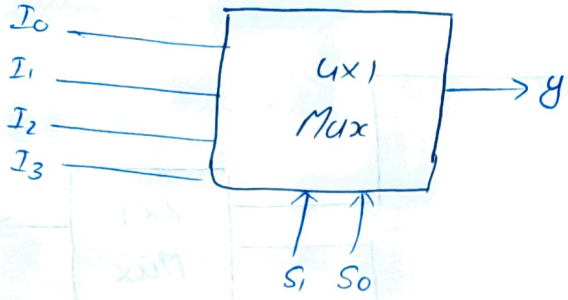
S_0	Y
0	I_0
1	I_1

↳ Function table

$Y = S_0'I_0 + S_0I_1$



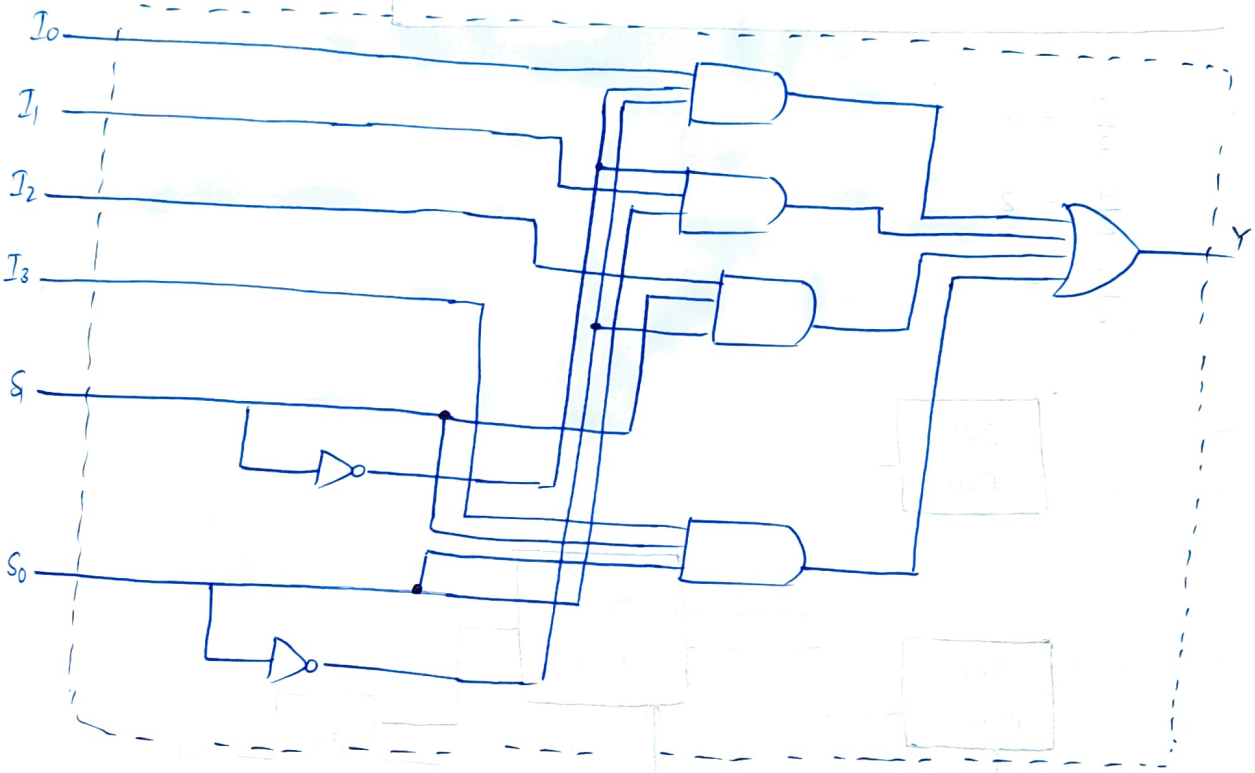
logic diagram 2×1 multiplexer



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

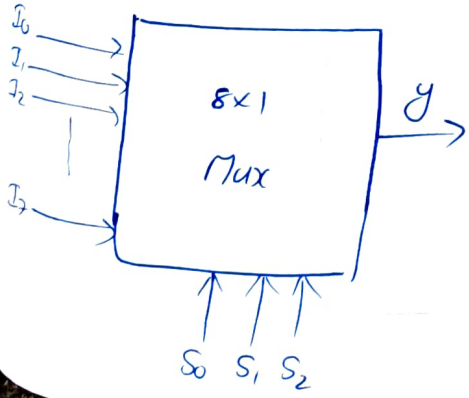
$$y = S_1'S_0'I_0 + S_1'S_0I_1 + S_1S_0'I_2 + S_1S_0I_3$$

Function table



logic diagram for 4x1 multiplexer

8x1 multiplexer



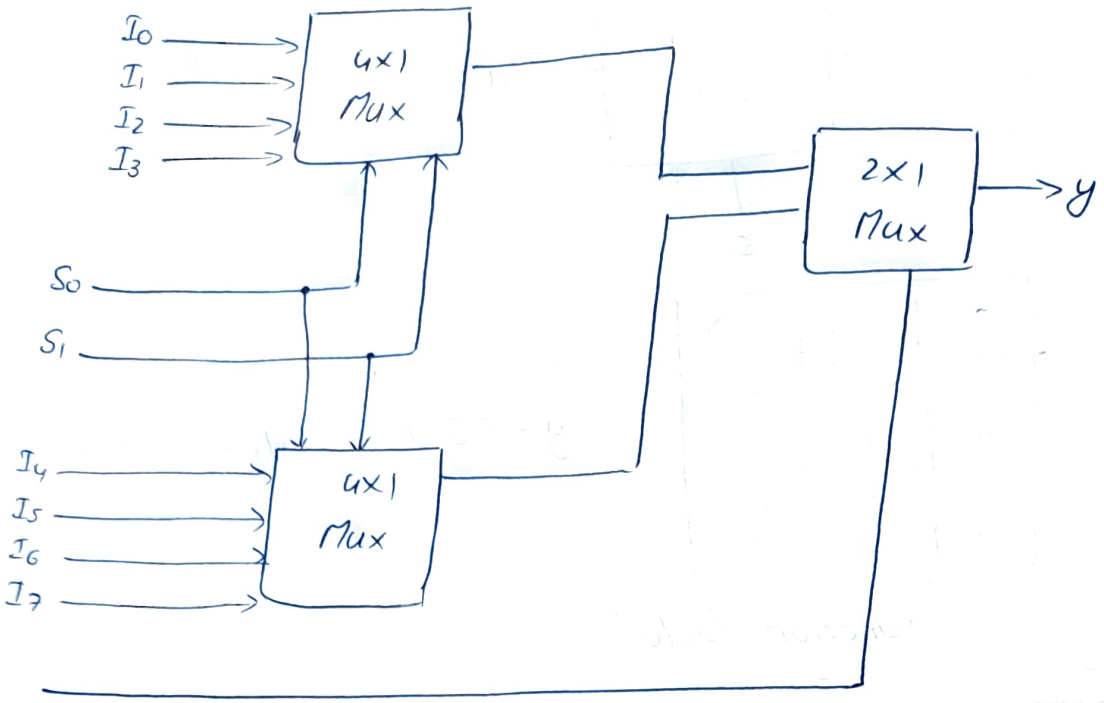
8x1 multiplexer using small multiplexer

8x1

$\frac{8}{4} = 2$ 2(two) 4x1 Mux

$\frac{2}{2} = 1$ one 2x1

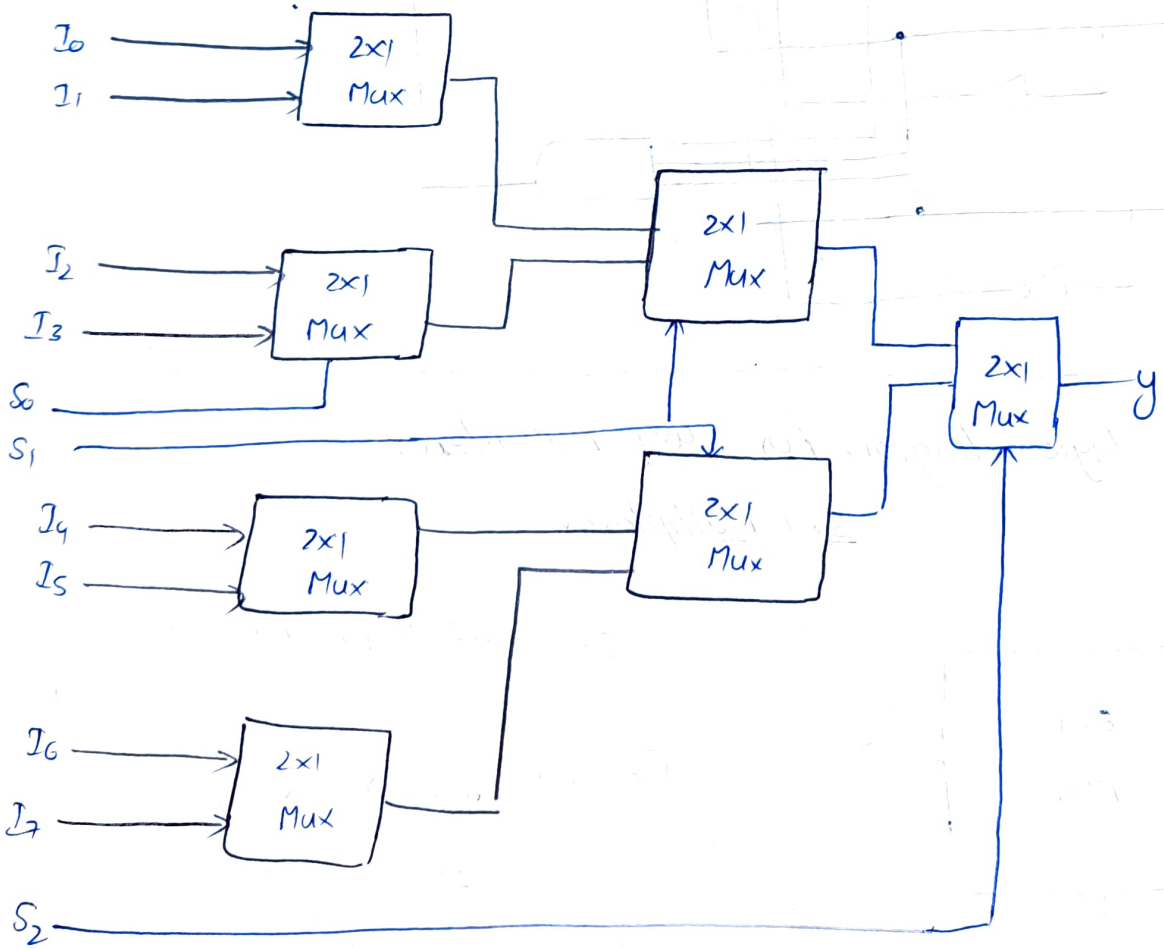
3 multiplexer



8x1 $\quad \frac{8}{2} = 4$

$\frac{4}{2} = 2$

$\frac{2}{2} = 1$



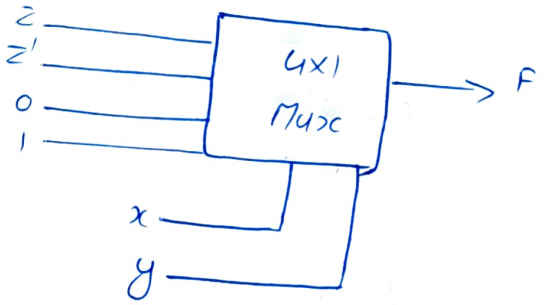
Implement following boolean function using multiplexer

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

n : 3 variables

n-1 = 2 selection lines

x	y	z	F	
0	0	0	0	F = z
0	0	1	1	
0	1	0	1	F = z'
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	

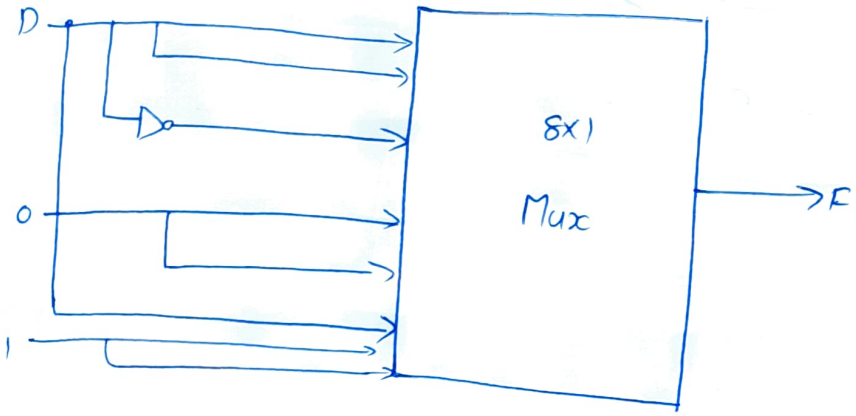


Implementation of F(x,y,z) using multiplexer

$$F(A,B,C,D) = \Sigma(1,3,4,11,12,13,14,15)$$

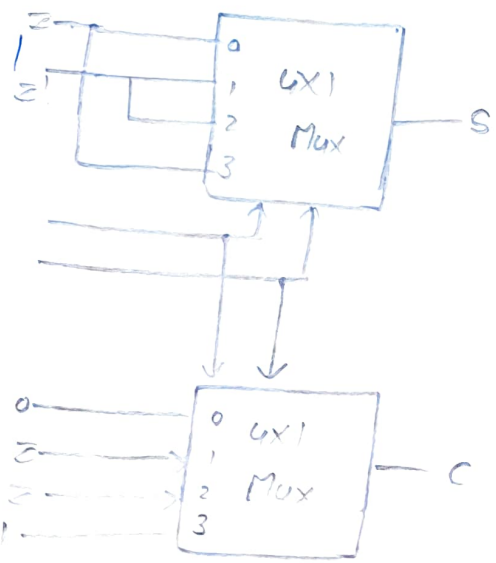
A	B	C	D		
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	F = D'
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = D'
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	

8x1 Mux to be used



Q) Implement the full adder using multiplexer

x	y	z	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	



No pending