Mod-3

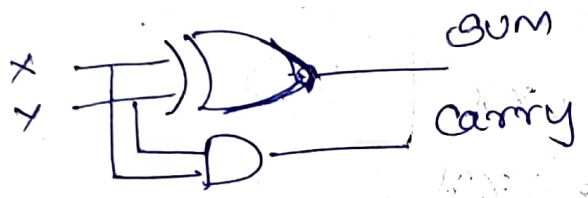
Half Adder

2 I/P 2O/P Combinational circuit

x	y	Sum(LSB)	Carry (MSB)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = x'y + xy' = x \oplus y$

$C = xy$



Full Adder

3 I/P 2O/P Combinational circuit

A	B	C_{in}	Sum(LSB)	C-OUT (MSB)
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 =

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

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

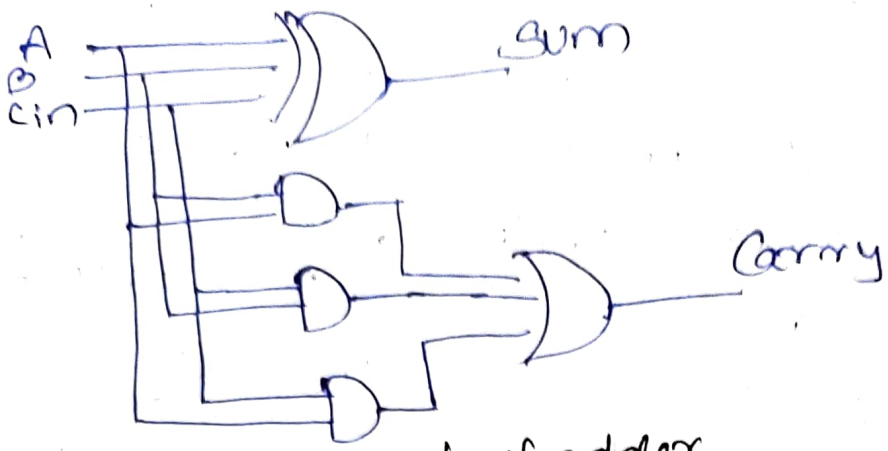
$A \oplus B = (\bar{A}B + A\bar{B}) \oplus C = (\bar{A}B + A\bar{B})\bar{C} + \overline{(\bar{A}B + A\bar{B})}\,C$

$\rightarrow \bar{A}B\bar{C} + A\bar{B}\bar{C} + \overline{(\bar{A}B)}\,\overline{(A\bar{B})}\,C$

$\bar{A}B\bar{C} + A\bar{B}\bar{C} + (A+\bar{B})(\bar{A}+B)\,C$

$\rightarrow \bar{A}B\bar{C} + A\bar{B}\bar{C} + A\bar{A}C + AB\bar{C} + \bar{A}\bar{B}C + \cancel{\bar{B}BC}$

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



⊛ Full adder using half adder

$S = x \oplus y \oplus z$

$C = xy + yz + zx$

$= (x \oplus y)z + zx$

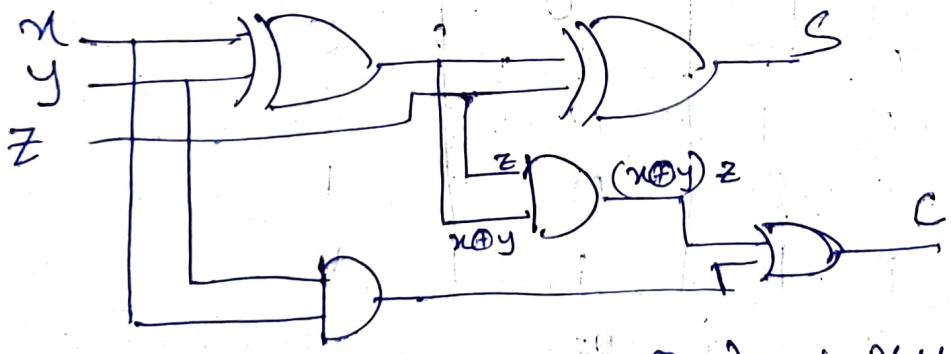
$\Rightarrow (\bar{x}y + x\bar{y})z + zx$

$\Rightarrow \bar{x}yz + x\bar{y}z + zx \Rightarrow x\bar{y}z + z(x + \bar{x}y)$

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

$\Rightarrow y(x + z\bar{y}) + xz \Rightarrow y(x+z) + xz$

$\Rightarrow xy + yz + zx$ ⊛



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

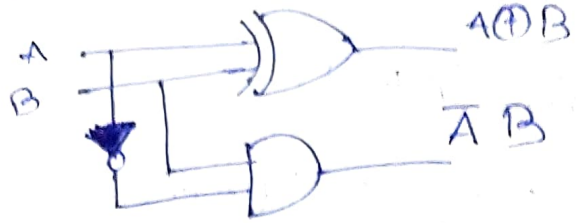
⊛ Half subtractor

2 I/P 2 O/P

A	B	Diff (LSB)	Borrow (MSB)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = A \oplus B$

$B = \bar{A}B$



⊛ Full Subtractor

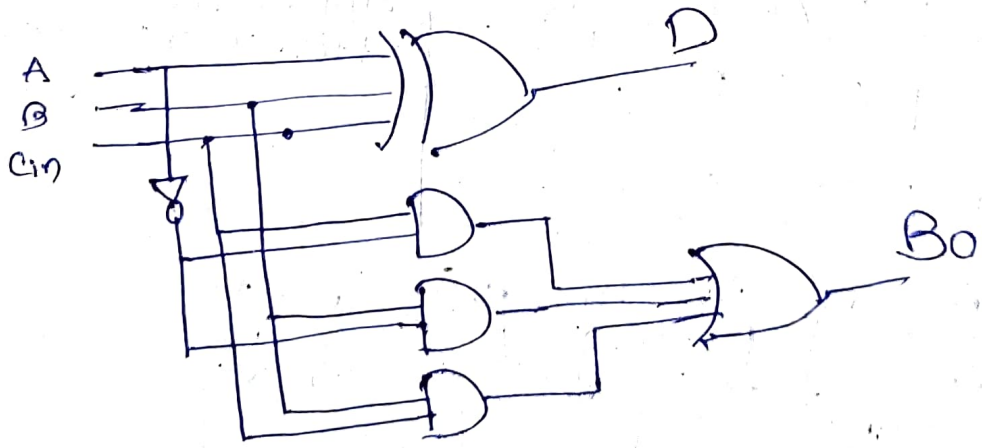
A	B	C_{in}	D	B_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$D = A \oplus B \oplus C_{in}$

$B_{out} = \bar{A}C_{in} + \bar{A}B + BC_{in}$

$\Rightarrow \bar{A}B + (\bar{A} + B)C_{in}$

B_{out}	A \ C_{in}	00	01	11	10
	0	0	1	1	1
	1	0	0	1	0



Borrow

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

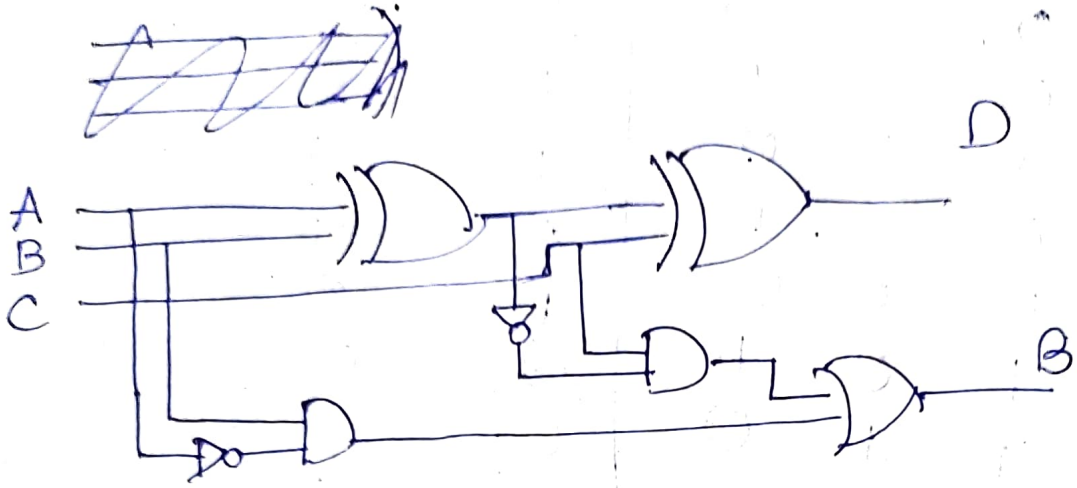
$\because \bar{A}B + \bar{A}C_{in} + BC_{in} = \bar{A}BC_{in} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC + \bar{A}BC$

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

$\bar{A}B(C+\bar{C}) + C(AB+\bar{A}\bar{B})$

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

Full Subtractor using half Subtractor



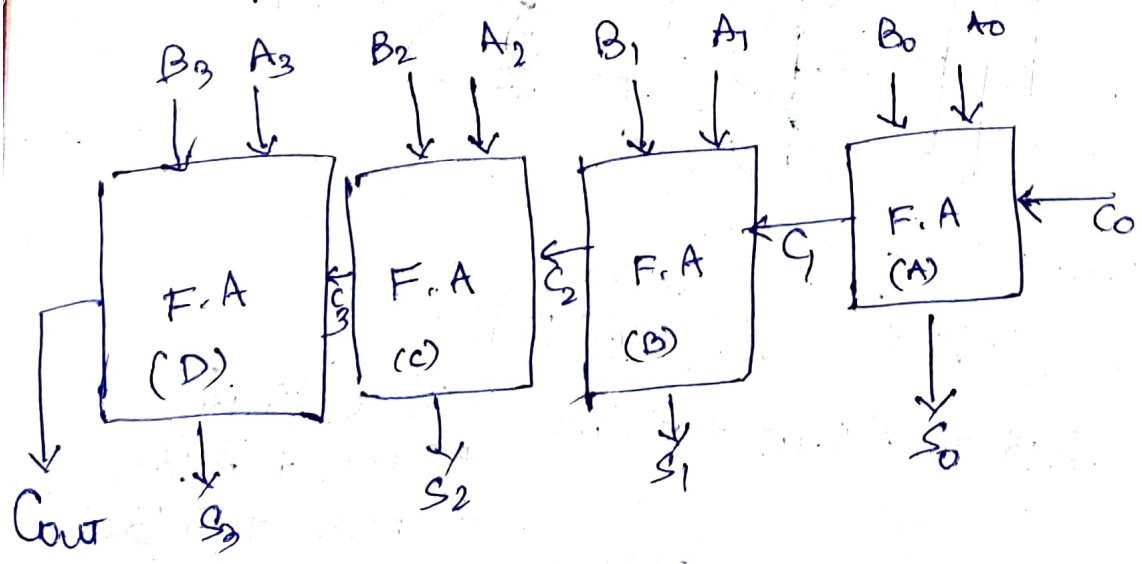
⊛ Ripple Carry Adder

It is a binary adder used to add two n-bit binary numbers.

→ Full adders can be connected in series.

The carry bit ripples from one adder to the next.

$$\begin{array}{r} A \quad 1\ 1\ 0\ 1 \quad (13) \\ + \\ B \quad 1\ 0\ 1\ 1 \quad (11) \\ \hline 1\ 1\ 0\ 0\ 0 \end{array}$$



Delay of Ripple Adder

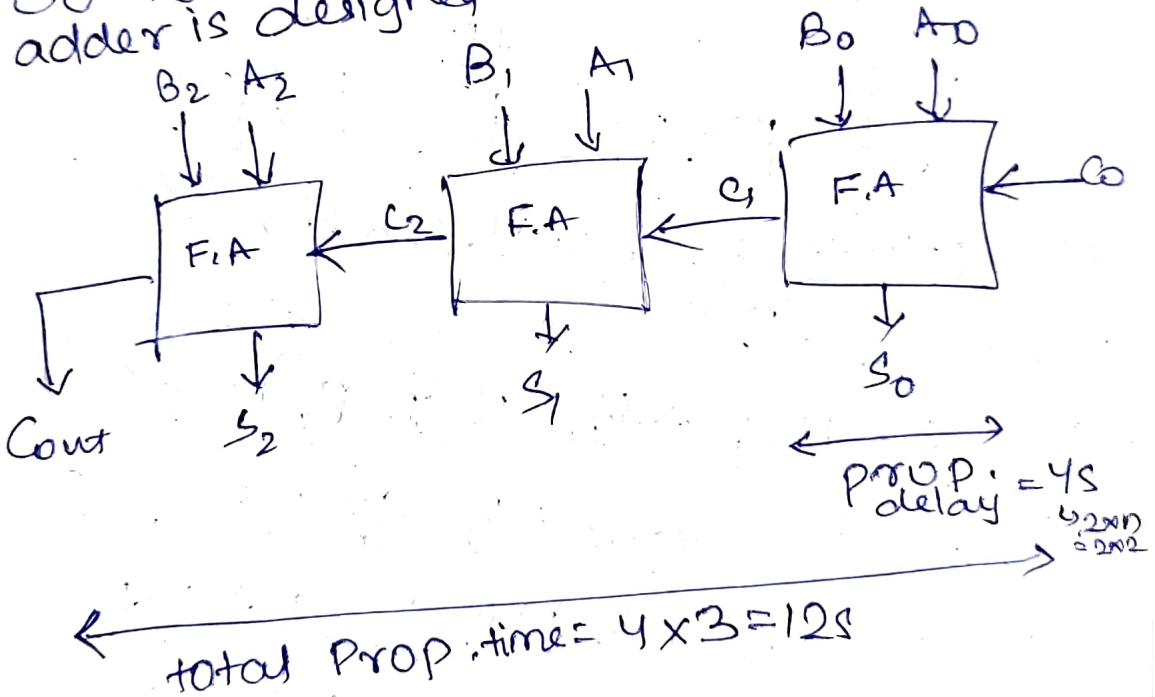
→ In a combinational ckt, the signal must propagate through the gates.

→ So the total propagation time equal to the propagation delay of a typical gate times multiplied with the gate levels in the circuit.

→ The propagation delay is the time takes the carry to propagate through the full adder.

→ For n-bit parallel adder the total gate delay will be 2n..

→ So to avoid this carry-look-ahead adder is designed



Carry look-Ahead Adder

→ Reduces the wait time to calculate the result of the larger bit of the adder

→ If the addition will always carry regardless of whether there is input carry, then A & B are called carry generator.

↓ Either A or B = 1
must Cin = 1

Both A & B = 1

The addition two bit input A & B is said to propagate if the addition will propagate whenever there is input carry i.e. $C_{in} = 1$.

The output

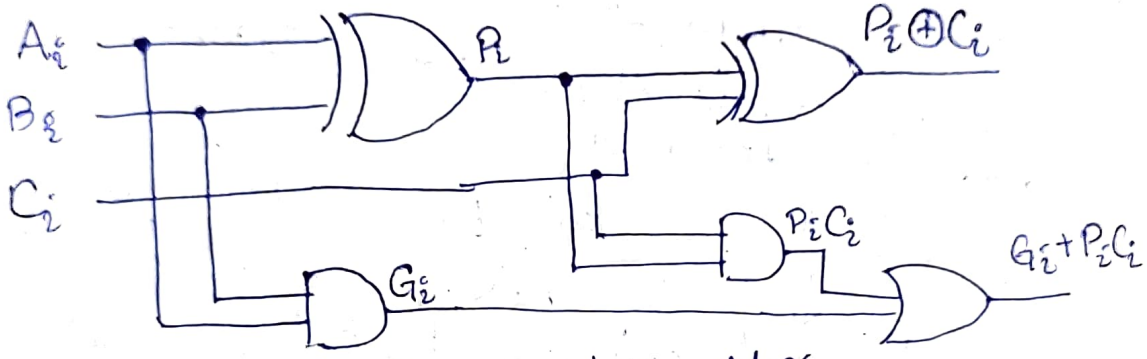
$S_i = Sum = P_i \oplus C_i$

$C_{i+1} = G_i + P_i C_i$

} Here

$P_i = A_i \oplus B_i$

$G_i = A_i B_i$



Lets apply these for a 4bit adder

$C_1 = G_0 + P_0 C_0$

$C_2 = G_1 + P_1 C_1 \Rightarrow G_1 + P_1 (P_0 C_0) = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 \Rightarrow G_2 + P_2 (G_1 + P_1 P_0 C_0)$

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

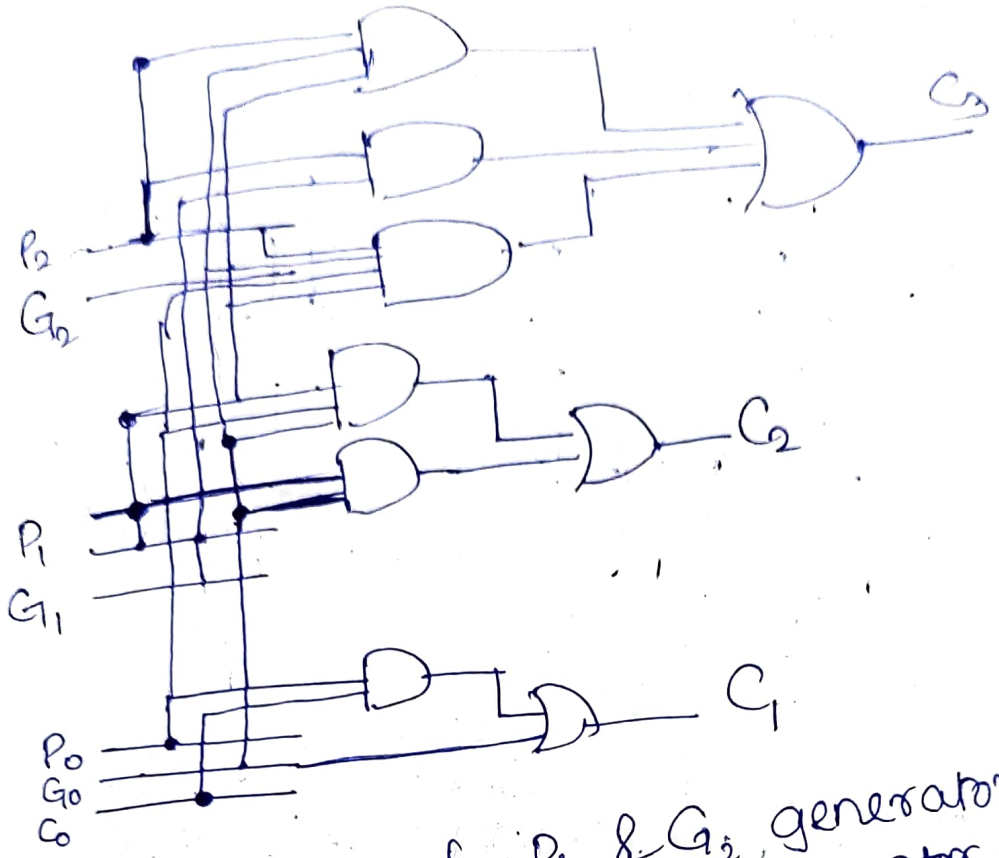
$C_4 = G_3 + P_3 C_3 \Rightarrow G_3 + P_3 G_2 + P_3 P_2 G_1 +$

$P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$

→ These shows that C_2, C_3, C_4 donot depend on previous carry-in

→ As soon as C_0 is computed, C_4 can reach ready state.

→ this is a two level circuit



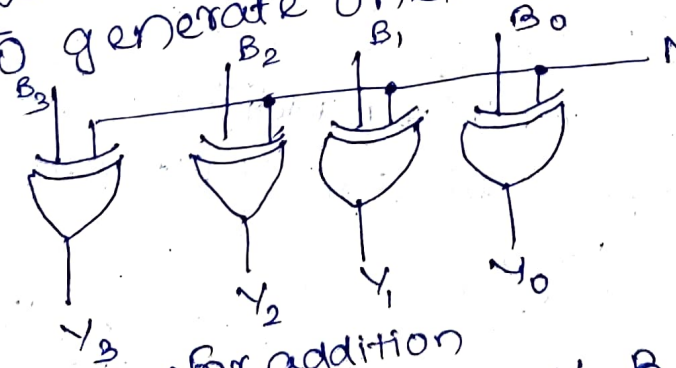
One gate delay for P_i & G_i generator, two gate delay for carry generator & one gate delay for sum generator

✳ Adder-Subtractor

For subtraction → 2's complement

⊙ One's complement Circuit

To generate one's comp. XOR is needed



$0 \oplus 1$
$1 \cdot 1 + 0 \cdot 0 \cdot 1$
$1 \cdot 0 + 0 \cdot 1 \cdots 0$

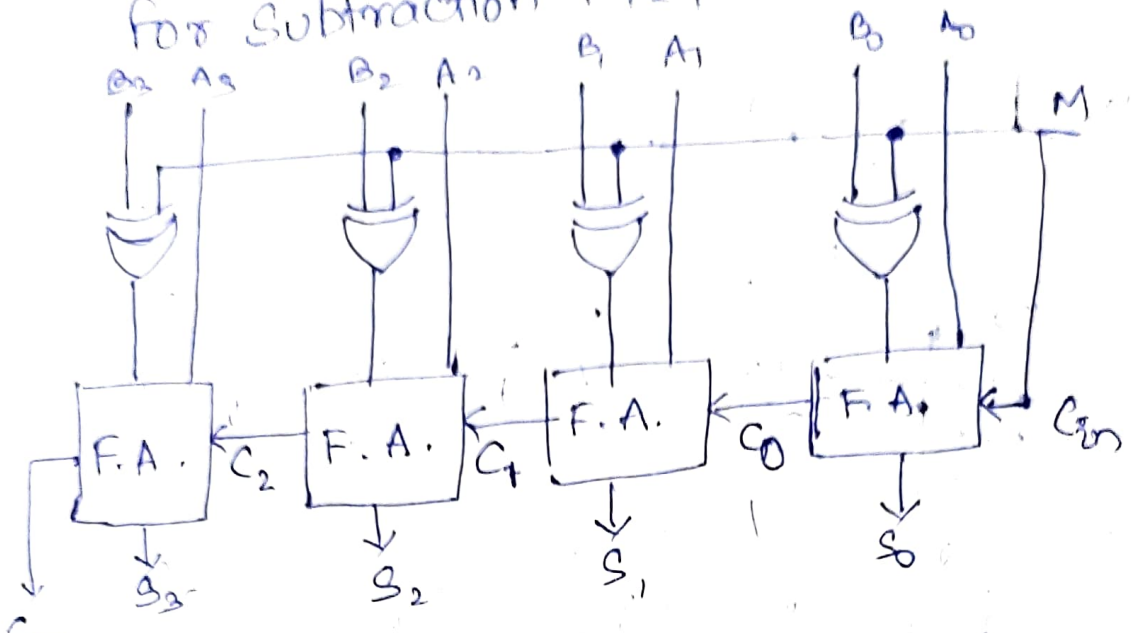
$0 \oplus 0$
$0 \cdot 1 + 1 \cdot 0$
0
$1 \cdot 1 + 0 \cdot 0$
1

If (M=0) → For addition $Y_0 = B_0$ $Y_1 = B_1$ $Y_2 = B_2$ $Y_3 = B_3$

(M=1) → For subtraction $Y_0 = \overline{B_0}$ $Y_1 = \overline{B_1}$ $Y_2 = \overline{B_2}$ $Y_3 = \overline{B_3}$

So for addition M = 0

for subtraction M = 1



$A_3\ A_2\ A_1\ A_0$

$1\ 0\ 1\ 1$ ← 11

$+\ 1\ 0\ 0\ 1$ ← 9

$1\ 0\ 1\ 0\ 0$ ← 20

$A_3\ A_2\ A_1\ A_0$

$1\ 0\ 1\ 1$ ← 11

$-\ 1\ 0\ 0\ 1$ ← 9

$0\ 1\ 1\ 0$

$+\ \ \ 1$

$0\ 1\ 1\ 1$

1's → 2's

$1\ 0\ 1\ 1$

$+\ 0\ 1\ 1\ 1$

$1\ 0\ 0\ 1\ 0$

$+\ 0\ 0\ 1\ 0$ (Ans)

If M = 0 $\quad B \oplus 0 = B$

M = 1 $\quad B \oplus 1 = \overline{B}$

✱ BCD Adder

Add two numbers as binary addition

If result > 9 then add 6 in 4 bit combination

Binary					BCD.				
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1
0	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					

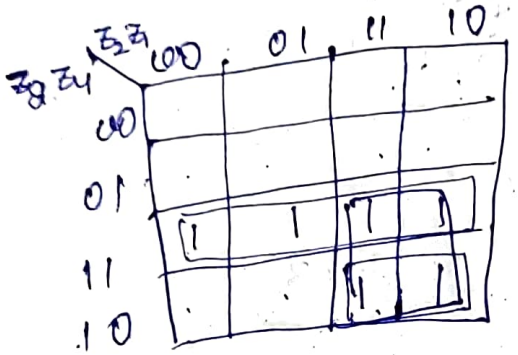
(0 - 9)

BCD sum = Binary sum

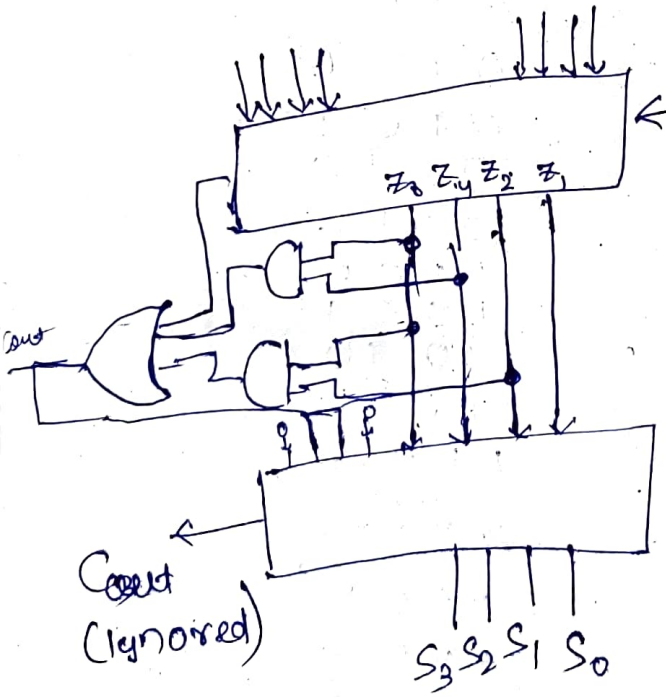
Same

10 - 15

K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	0	1	1	1	0	0	1	0	12
0	1	1	0	0	1	0	0	1	1	13
0	1	1	0	1	1	0	0	1	1	14
0	1	1	1	0	1	0	1	0	0	15
0	1	1	1	1	1	0	1	0	1	



$f = Z_8 Z_4 + Z_8 Z_2$



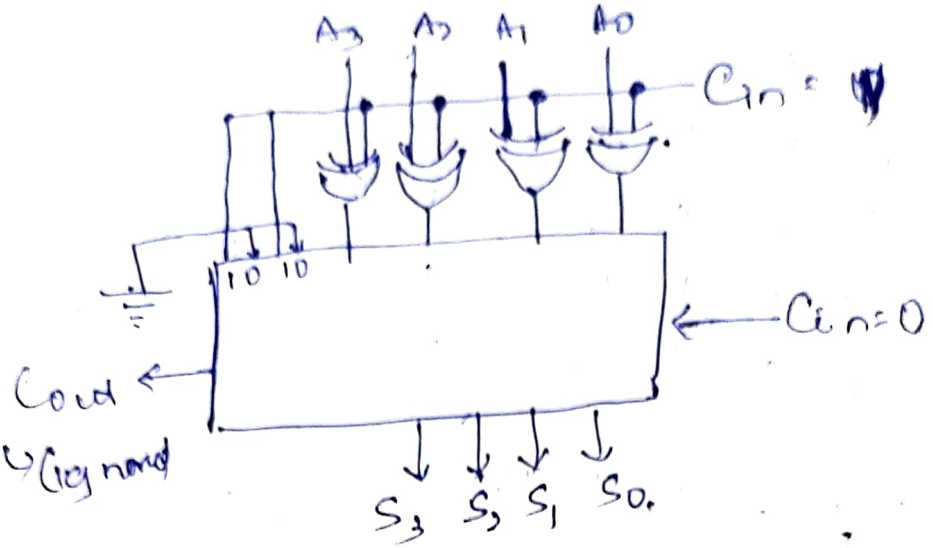
0110 when
(i) K = 1
(ii) $Z_8(Z_4 + Z_2)$
(iii) $Z_8 Z_2'$

⊛ BCD subtractor

Subtraction using 9's complement:
(i) Find 9's complement of -ve number.
(ii) Add the numbers using BCD adder
(iii) If carry is generated, add carry to the result, otherwise find the 9's compliment of the result.

S-1
Invert each bit using XOR gator

S-2 Add 10 ie $(1010)_2$ to it



BCD to Excess-3

Decimal Number	BCD A B C D	Excess 3 W X Y Z
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0
10		
11		
12		
13		
14		
15		

W

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$W = A + BD + BC$

X

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

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

4

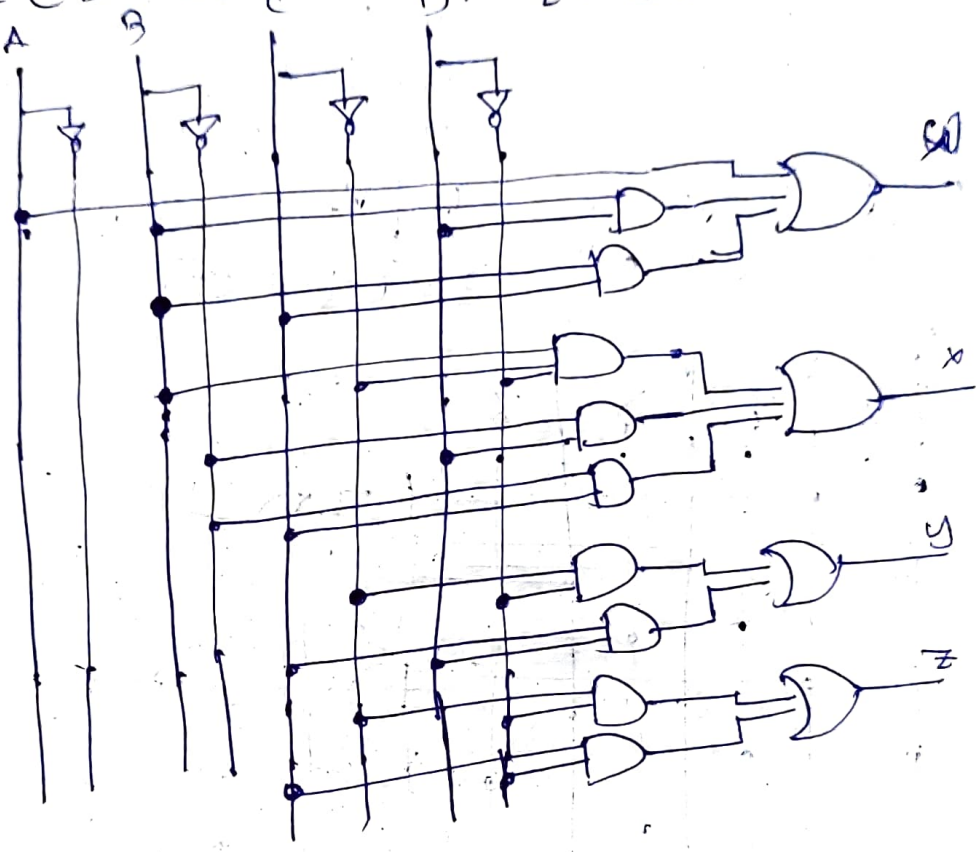
AB \ CD	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

$y = \bar{C}\bar{D} + CD$

2

AB \ CD	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$z = \bar{C}\bar{D} + C\bar{D}$



Excess-3 to BCD

Decimal	Excess-3	BCD
0	0 0 1 1	X X X X
1	0 1 0 0	X X X X
2	0 1 0 1	X X X X
3	0 1 1 0	0 0 0 0
4	0 1 1 1	0 0 0 1
5	1 0 0 0	0 0 1 0
6	1 0 0 1	0 0 1 1
7	1 0 1 0	0 1 0 0
8	1 0 1 1	0 1 0 1
9	1 1 0 0	0 1 1 0
10	1 1 0 1	0 1 1 1
11	1 1 1 0	1 0 0 0
12	1 1 1 1	1 0 0 1
13		X X X X
14		X X X X
15		X X

AB \ CD	00	01	11	10
00	x	x	0	x
01	0	0	0	0
11	1	x	x	x
10	0	0	1	0

$W = AB + ACD$

AB \ CD	00	01	11	10
00	x	x	0	x
01	0	0	1	0
11	0	x	x	x
10	1	1	0	1

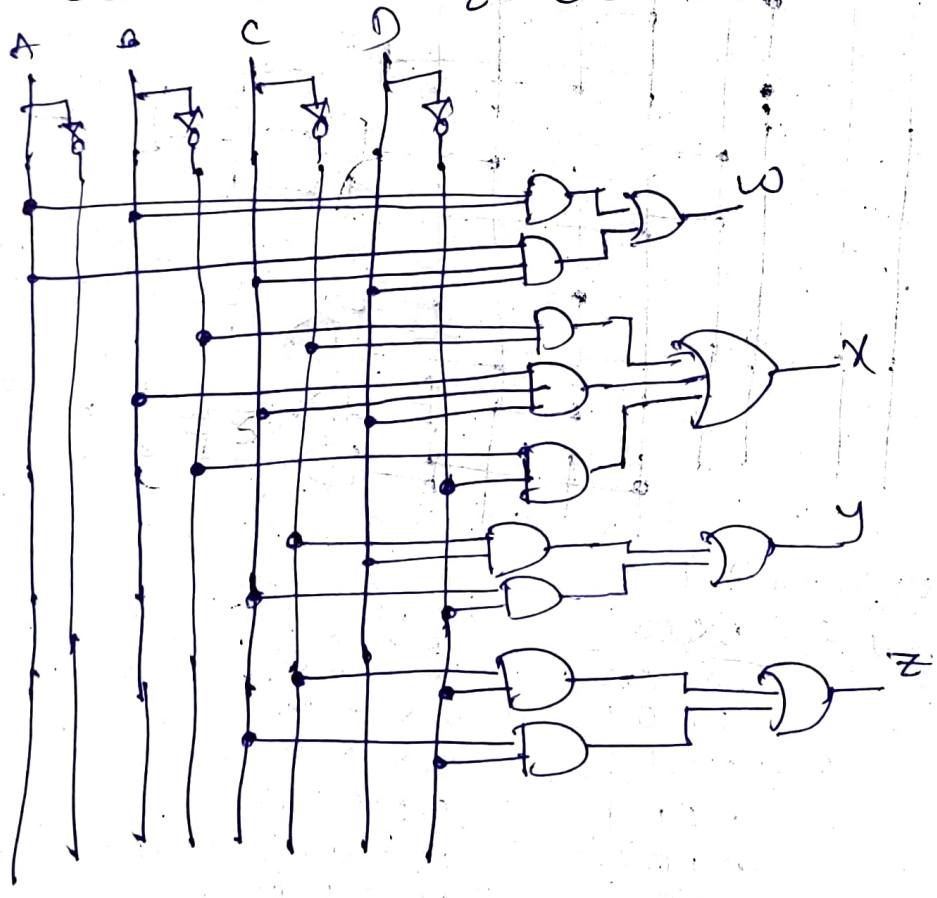
$X = \bar{B}\bar{C} + BCD + \bar{B}\bar{D}$

AB \ CD	00	01	11	10
00	x	x	0	x
01	0	1	0	1
11	0	x	x	x
10	0	1	0	1

$Y = \bar{C}D + C\bar{D}$

AB \ CD	00	01	11	10
00	x	x	0	x
01	1	0	0	1
11	1	x	x	x
10	1	0	0	1

$Z = \bar{C}\bar{D} + C\bar{D}$



Binary to Gray code

Binary	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

add ignore carry

0010
0010
0011

0111

0101
0111
0110

0110
0100
0100
0111

0000 0000
0001 0001
0010 0011

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$W = A$

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

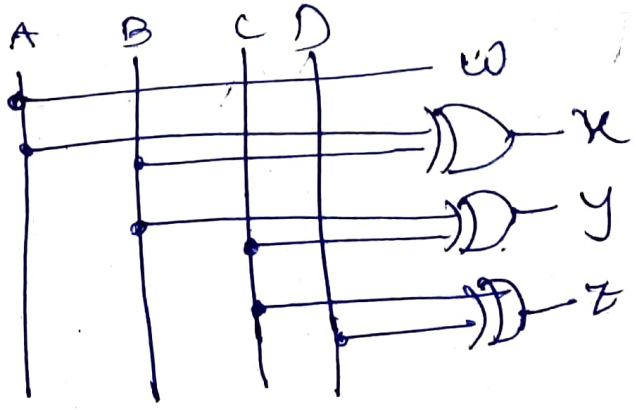
$x = A\bar{B} + \bar{A}B = A \oplus B$

AB \ CD	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$y = \bar{C}B + C\bar{B} = C \oplus B$

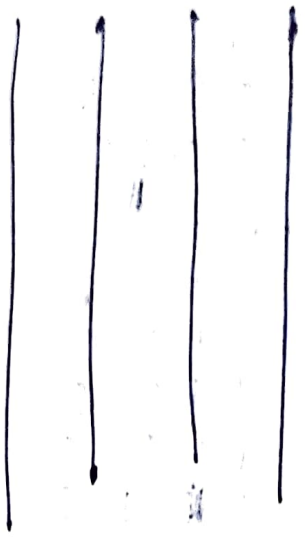
AB \ CD	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$z = \bar{C}D + C\bar{D} = C \oplus D$



Gray to Binary

W	X	Y	Z
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1



AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$W = A$

AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	1	1
10	0	0	0	0

$x = B$

AB \ CD	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	0	0	1	1
10	0	0	1	1

Exor

Gray to Binary

1 0 1 0 ... Add diagonally

1 1 0 0 ... ignore carry

*Comparator

A magnitude digital comparator circuit that compares two digital or binary numbers

Condⁿ A>B A=B A<B

1-Bit magnitude comparator

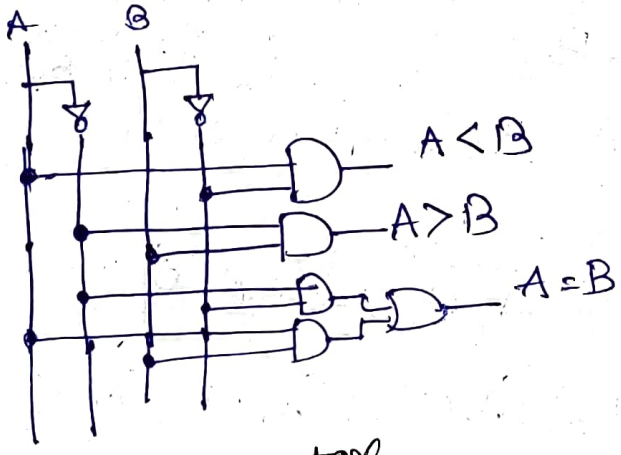
→ Used to compare two bits

A	B	A>B	A=B	A<B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

$A>B \Rightarrow \overline{A}B$

$A<B \Rightarrow A\overline{B}$

$A=B \Rightarrow \overline{A}\,\overline{B} + AB$



Note

$\overline{(A>B)+(A<B)} = (A=B)$

2 Bit comparator

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	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

A>B

A_1A_0 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$A>B \Rightarrow A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$

A=B

A_1A_0 \ B_1B_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

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

$\Rightarrow (A_0 \odot B_0)(A_1 \odot B_1)$ (XNOR)

A<B

A_1A_0 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$A<B \Rightarrow \bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0$

Logic diagram

(*) 4 Bit magnitude Comparator

↳ 8 inputs $A_3\ A_2\ A_1\ A_0$, $B_3\ B_2\ B_1\ B_0$.

Case1 When A = B

(1) If $A_3=B_3$ & $A_2=B_2$ & $A_1=B_1$ & $A_0=B_0$

In general

$A=B \Rightarrow A_iB_i + \bar{A}_i\bar{B}_i = A_i \odot B_i$

So $A=B \Rightarrow (A_3 \odot B_3)(A_2 \odot B_2)(A_1 \odot B_1)(A_0 \odot B_0)$

$A=B \Rightarrow x_3 . x_2 . x_1 . x_0$

Case2 A > B

(a) $A_3 = 1$ $B_3 = 0$

(b) $A_3 = B_3$, $A_2 = 1$ $B_2 = 0$

(c) $A_3 = B_2$, $A_2 = B_2$, $A_1 = 1$ $B_1 = 0$

(d) $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$, $A_0 = 1$ $B_0 = 0$

$A>B \Rightarrow A_3\bar{B}_3 + x_3A_2\bar{B}_2 + x_3x_2A_1\bar{B}_0 + x_3x_2x_1A_0\bar{B}_0$

Case 3 $A < B$

(a) $A_3 = 0$ $B_3 = 1$

(b) $A_3 = B_3$. $A_2 = 0$ $B_2 = 1$

(c) $A_3 = B_3$, $A_2 = B_2$. $A_1 = 0$ $B_1 = 1$

(d) $A_3 = B_3$. $A_2 = B_2$, $A_1 = B_1$, $A_0 = 0$ $B_0 = 1$

$A < B \Rightarrow \overline{A_3} B_3 + x3 \overline{A_2} B_2 + x_3 x_2 \overline{A_1} B_1 + x_3 x_2 x_1 \overline{A_0} B_0$

Logic diagram

A comparator performing the comparison operation to more than four bits by cascading two or more 4-bit comparators is called cascading comparator.

⊙ Applications

→ Used in CPU & MCU

→ Used as process controllers & for servo motor control

→ Used in password verification & biometric applications

⊛ Error detection & correction codes

→ During transmission of binary data, noise may also be added. So there may be error in recieved data i.e. $1 \rightarrow 0$ or $0 \rightarrow 1$.

→ We can't avoid the interference of noise.

Error detection code

↓

Used to detect the errors present in the recieved data

ex: Parity code

* Error correction code

↓

Used to correct the errors present in the data

ex: Hamming code

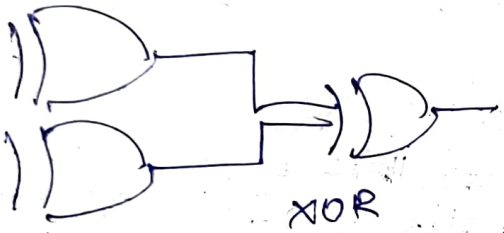
⊛ Parity Code

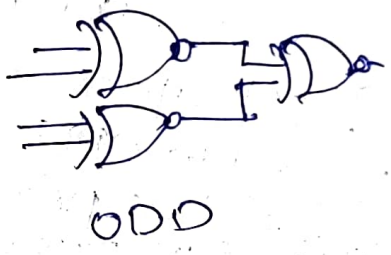
→ A parity bit is an extra bit included in binary message.

⊙ even parity system: Add 1 if there is odd number of 1's in string

⊙ Odd Parity System: Add 1 if there is even no. of 1's string

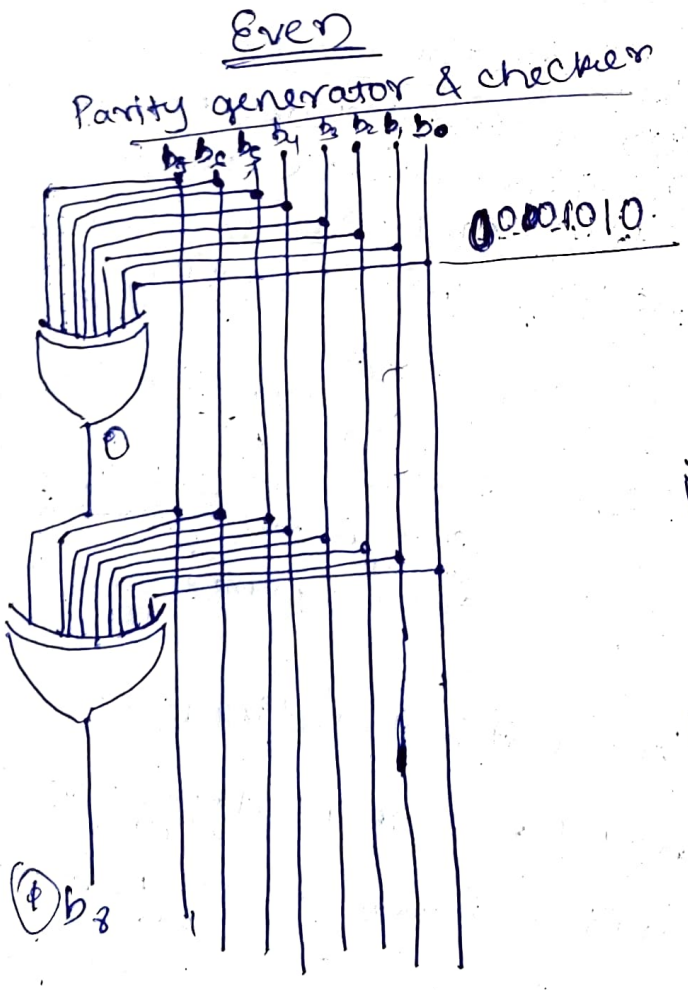
D_3	D_2	D_1	D_0	Even parity	Odd Parity
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	0	1



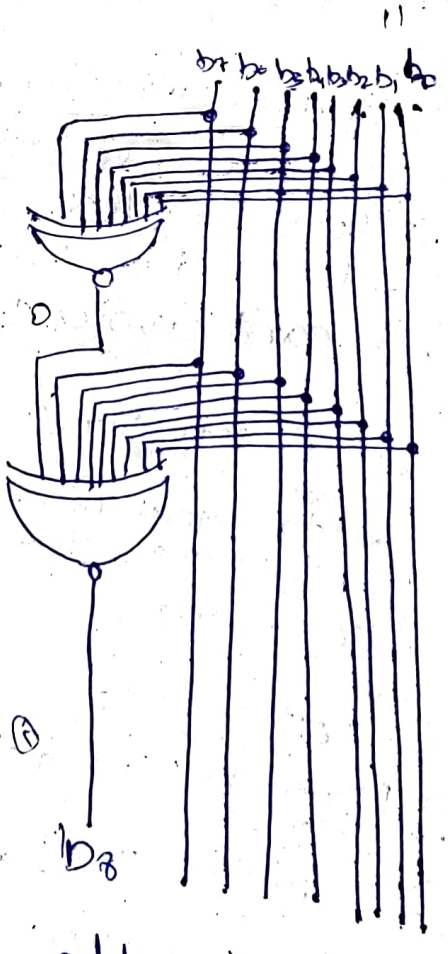


Even

Parity generator & checker



Even parity generator & checker



Odd parity generator & checker

0 → all ok

1 → change in bit

Limitation

→ Cannot correct the incorrect bit
→ Doesn't tell which bit is incorrect
→ Only error in a single bit would be identified

⊛ Hamming Code

→ Useful for both detection & correction of error in recieved data.
→ The code uses multiple parity bit and we have to place these parity bits in the posⁿ of powers of 2.
→ K = No. of parity bit
Min value of k is $2^k \geq n + k + 1$
(n → no. of bits in binary code)

For ex:
In a 4-bit binary code
$2^k \geq 4 + k + 1 \Rightarrow 2^k \geq 5 + k$

So $k = 3$
4 data bit & 3 parity bits.

#Imp
Format: d7 d6 d5 p4 d3 p2 p1
4 2 1

d → data bit p → parity bit
d3 p2 p1

$p1 = d7 \oplus d5 \oplus d3$
$p2 = d7 \oplus d6 \oplus d3$
$p3 = d7 \oplus d6 \oplus d5$

Example
Even parity 7 bit hamming code for 1011

d7 d6 d5 p4 d3 p2 p1
1 0 1 ? 1 ? ?

$p1 = d7 \oplus d5 \oplus d3 = 1$
$p2 = d7 \oplus d6 \oplus d3 = 0$
$p4 = d7 \oplus d6 \oplus d5 = 0$

1010101
Final hamming code

Hamming code received is 1010001

d7	d6	d5	p4	d3	p2	p1
1	0	1	0	0	0	1

Steps

To find the posⁿ of error

Even

$x = d7 \oplus d5 \oplus d3 \oplus p1$

$y = d7 \oplus d6 \oplus d3 \oplus p2$

$z = d7 \oplus d6 \oplus d5 \oplus p4$

So $x = 1$

$y = 1$

$z = 0$

$zyx = 011 \Rightarrow (3)$ i.e. 3rd bit is erroneous.

Corrected code is

1010101

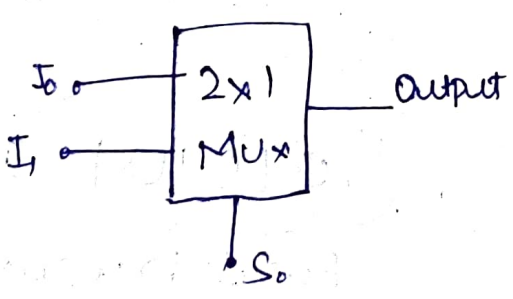
Multiplexer

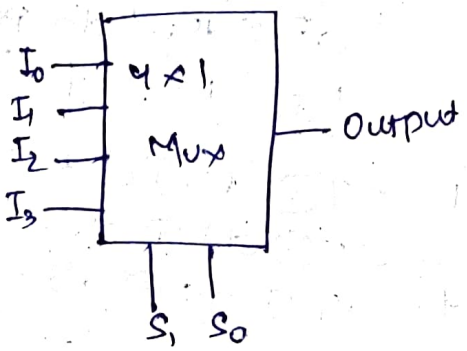
→ A mux is a device that has many inputs and single output.

→ The particular input chosen for output is determined by the value of control line.

No. of control lines = $\log_2 n$ n = no. of input

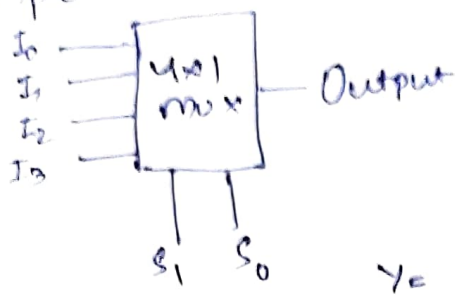
→ It is also called as data selector





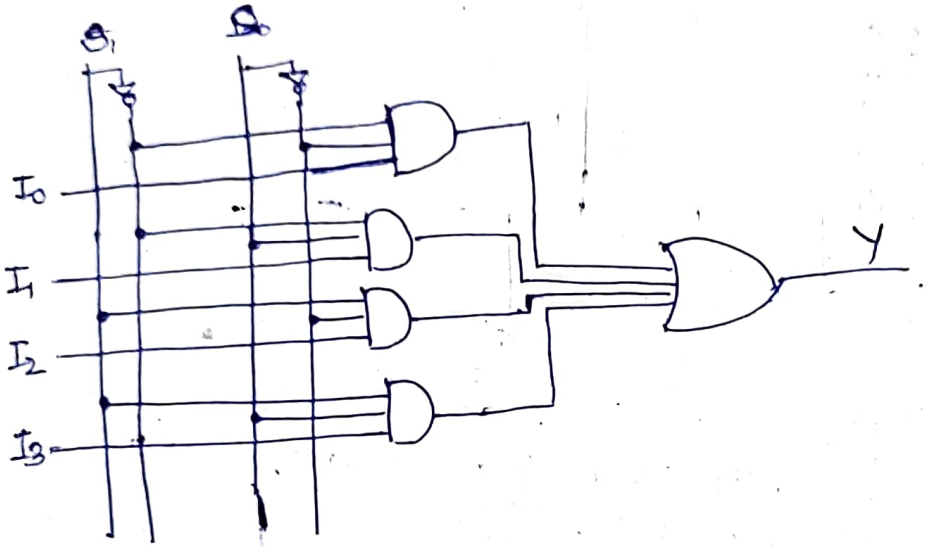
⊛ 4×1 Mux

4 inputs 1 output 2 control bits



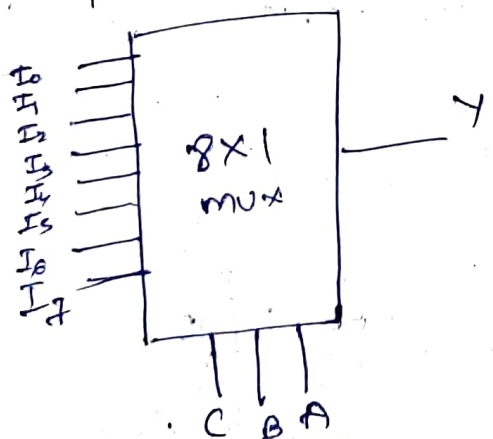
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_0 I_3$

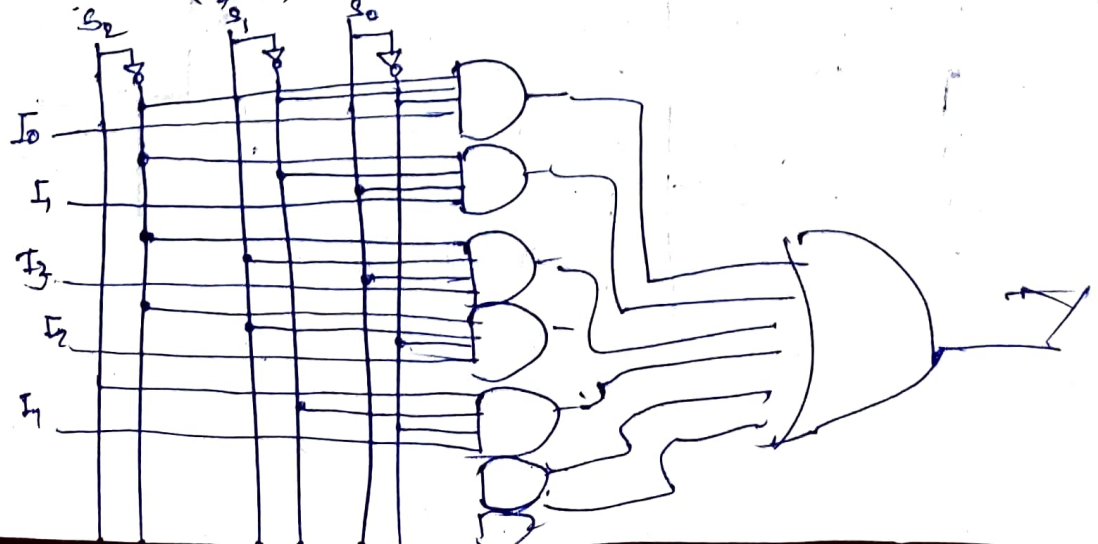


⊛ 8×1 MUX

8 input 1 output 3 control lines



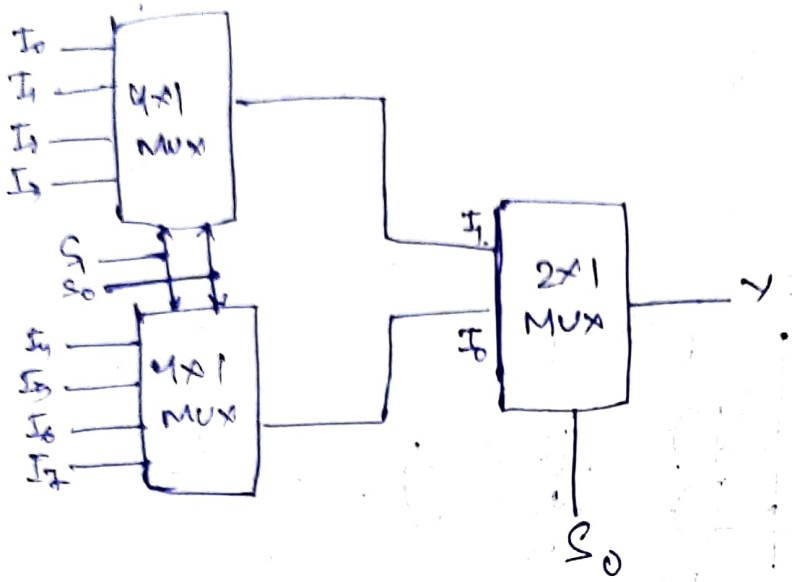
S_2	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	0	0	I_4
1	0	1	I_5
1	1	0	I_6
1	1	1	I_7



8×1 MUX using 4×1 MUX

$n_1 = 8$ $\quad \frac{8}{4} = 2$

$n_2 = 4$ $\quad \frac{2}{4} = 0.5$.



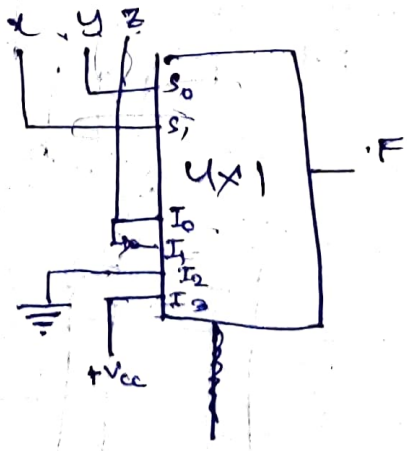
Application

1. Used to inc. efficiency of comm. system
2. Reduce the no. of wires
3. Used to integrate multiple audio signals.

✱ Boolean Func? implementation using MUX

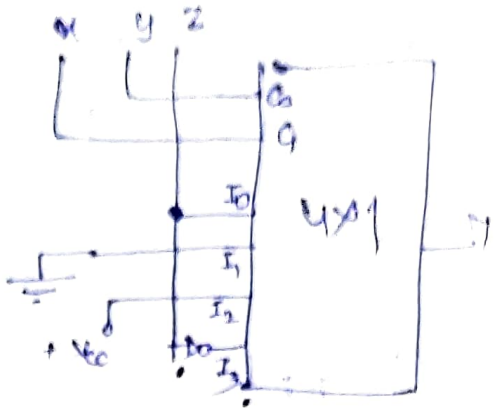
$\Sigma(1,2,6,7) = F(x,y,z)$

x	y	z	F	
0	0	0	0	F = z
0	0	1	1	
0	1	0	1	$F = \bar{z}$
0	1	1	0	
1	0	0	0	F = 0
1	0	1	0	
1	1	0	1	F = 1
1	1	1	1	



$G_1(x,y,z) = m(1,4,5,6)$

x	y	z	G_1	
0	0	0	0	$G = z$
0	0	1		
0	1	0	0	$G = 0$
0	1	1	0	
1	0	0	1	$G = 1$
1	0	1	1	
1	1	0	1	$G = z'$
1	1	1	0	

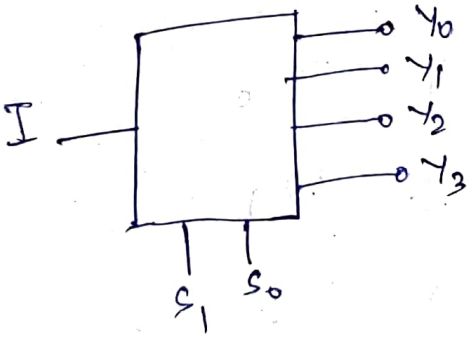


✱ Demultiplexer

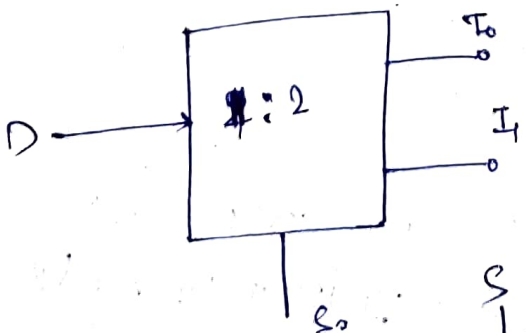
→ Circuit which can distribute multiple outputs from a single input.

→ No. of select lines = $\log_2 N$ m = no. of output line

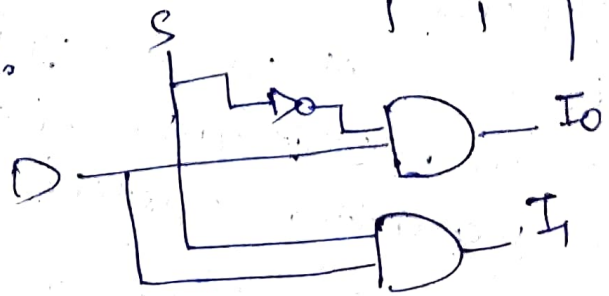
→ Also called as Data distributor



1:2 Demultiplexer

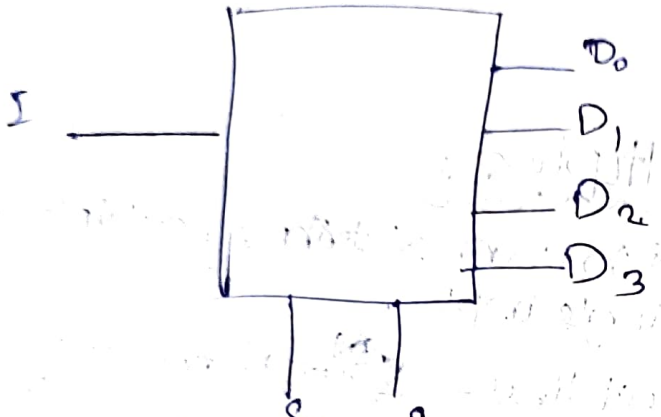


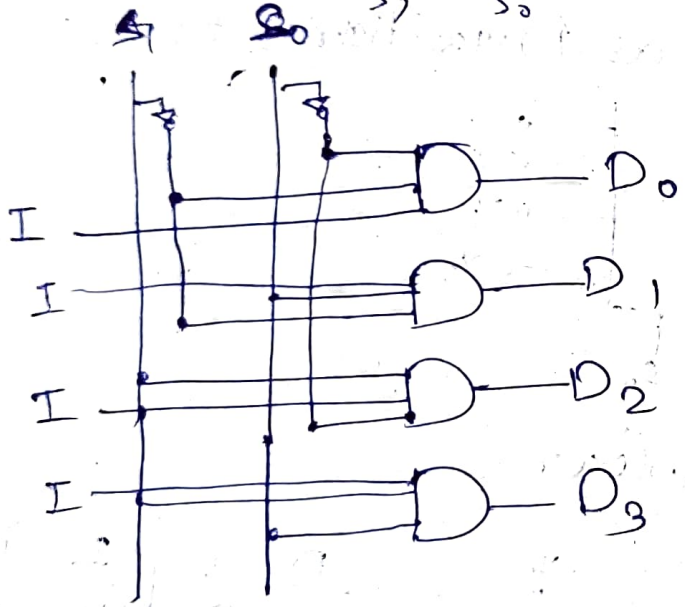
D	S_0	I_0	I_1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1



1 : 4 Demultiplexer

S_1	S_0	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	0





Application

1. Widely used in microprocessor, DE
2. Storage unit of the output of ALU
3. Security monitoring system
4. Data acquisition system
5. Diff. functional unit enablation -