Unit-2

Combinational logic and Sequential logic circuits: Introduction to combinational logic circuits, Half/Full adders subtractors, Parallel adders/subtractors, Binary comparators Decoders, encoders, multiplexers. Basic bistable elements, SR flipflop, D flipflop, JK flipflop T Flipflop, Master slave JK flipflop, characteristic equations, conversion of flipflops.

* Half Adder

Half adders adds two one-bit numbers and produce two outputs sum and carry.

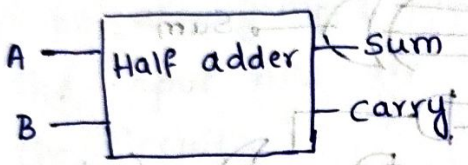


fig: Block diagram

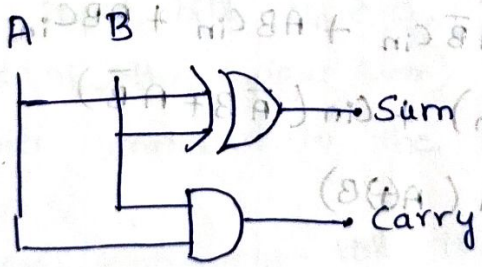
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

fig: truthtable.

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

$Carry = AB$

∴ Implementation.

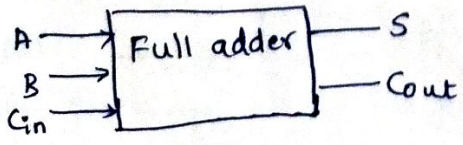


* Full adder

Full adder is a combinational logic circuit that performs addition of three single bit numbers.

A, B, Cin ⇒ Inputs

Cin ⇒ Carry of previous stage.



S → Sum is LSB

$Cout \rightarrow Carry_{Final}$ is MSB

* Full adder is used in computer ALU to perform arithmetic operations.

A	B	Cin	S	Cout
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

fig: truth table.

Sum

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

~~$S = \bar{A}\bar{B}C_{in} + A\bar{B}\bar{C}_{in}$~~

$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in}$

$= C_{in}(AB + \bar{A}\bar{B}) + \bar{C}_{in}(\bar{A}B + A\bar{B})$

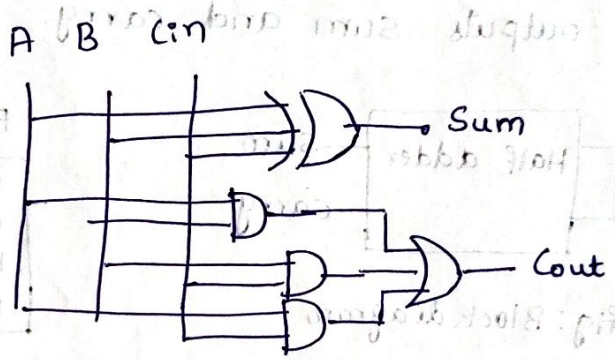
$= C_{in}(\overline{A \oplus B}) + \bar{C}_{in}(A \oplus B)$

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

$C_{out} = BC_{in} + AB + AC_{in}$.

Cout

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



★ Full adder using Half adders :-

We have,

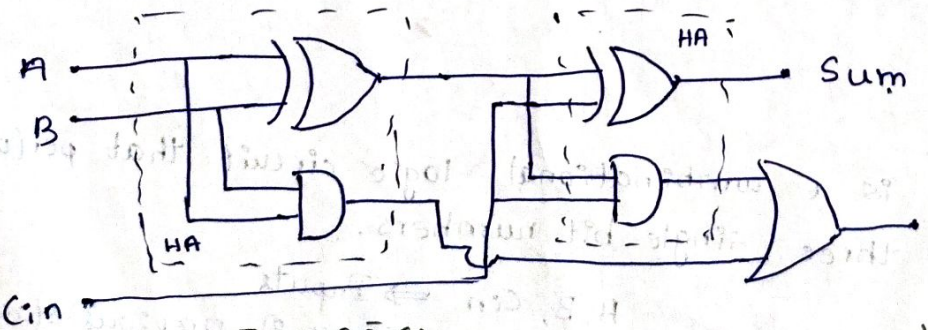
$S = A \oplus B \oplus C_{in}$.

Consider the truth table,

$C_{out} = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in}$

$C_{out} = AB(C_{in} + \bar{C}_{in}) + C_{in}(\bar{A}B + A\bar{B})$

$= AB + C_{in}(A \oplus B)$



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

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

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

$AB + BC_{in} + A\bar{B}C_{in}$

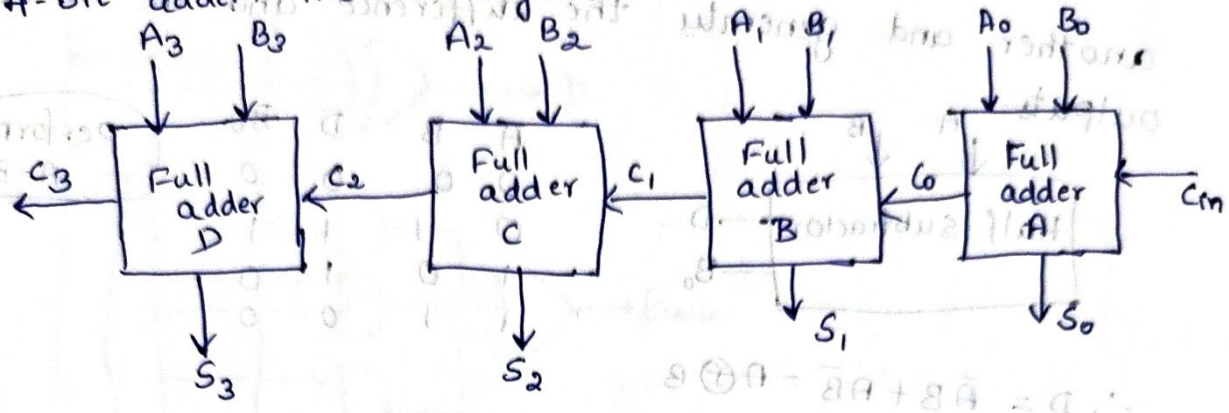
$BC_{in} + A(B + \bar{B}C_{in})$

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

$C_{out} = BC_{in} + AB + AC_{in}$

4-bit parallel adder / ripple carry adder :-

* Multiple bit adders can be configured by cascading full adders.
* Four full adders can be cascaded to configure a 4-bit adder. The arrangement is as shown.

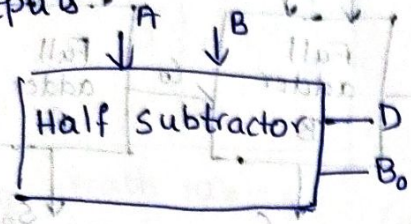


* The arrangement is called parallel adder since all the input bits are applied at the same time to the circuit.
* Firstly, the full adder A adds two bits A_0 and B_0 along with the carry C_{in} to generate the sum S_0 (the first bit of output sum) and the carry C_0 which is connected to the next adder in chain.
* Next, the full adder B uses this carry bit C_0 to add with the input bits A_1 & B_1 to generate sum S_1 (the second bit of the output sum) and the carry C_1 which is further connected to the next adder in chain and so on.
* The process continuess till the last full adder D uses carry bit C_2 to add with inputs A_3 & B_3 to generate the last bit of the output S_3 along with last carry out C_3.
* Hence in parallel adder output do not appears at the same time. The correct carry input to the n^{th} stage will only appear after the addition is completed in all previous $(n-1)$ stage.
* The carry output of each stage will appear only after the propagation delay associated with that stage.

* The carry thus ripples through each stage and hence this adder is referred to as a ripple carry adder.

* Half Subtractor :-

A Half subtractor subtracts one binary bit from another and generates the difference and the borrow outputs.

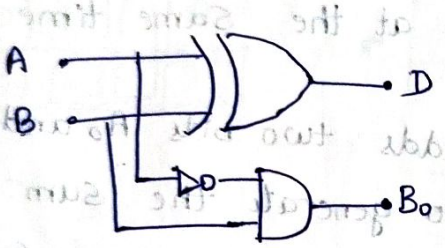


A	B	D	B_o
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

performs A-B

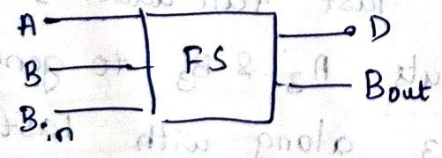
$\therefore D = \bar{A}B + A\bar{B} = A \oplus B$

$B_o = \bar{A}B$



* Full Subtractor :-

A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, taking into account borrow of the previous adjacent lower minuend bit. The circuit has 3 inputs and 2 outputs.



A	B	B_{in}	D	B_{out}
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

D

A \ B_{in}	00	01	11	10
0	0	(1)	0	(1)
1	(1)	0	(1)	0

$D = \bar{A}\bar{B}B_{in} + \bar{A}B\bar{B}_{in} + A\bar{B}\bar{B}_{in} + ABB_{in}$

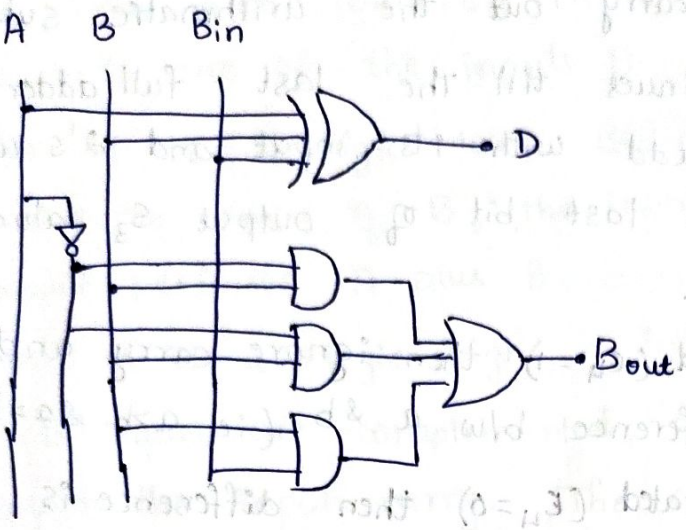
$= B_{in}(\bar{A}\bar{B} + AB) + \bar{B}_{in}(\bar{A}B + A\bar{B})$

$= A \oplus B \oplus B_{in}$

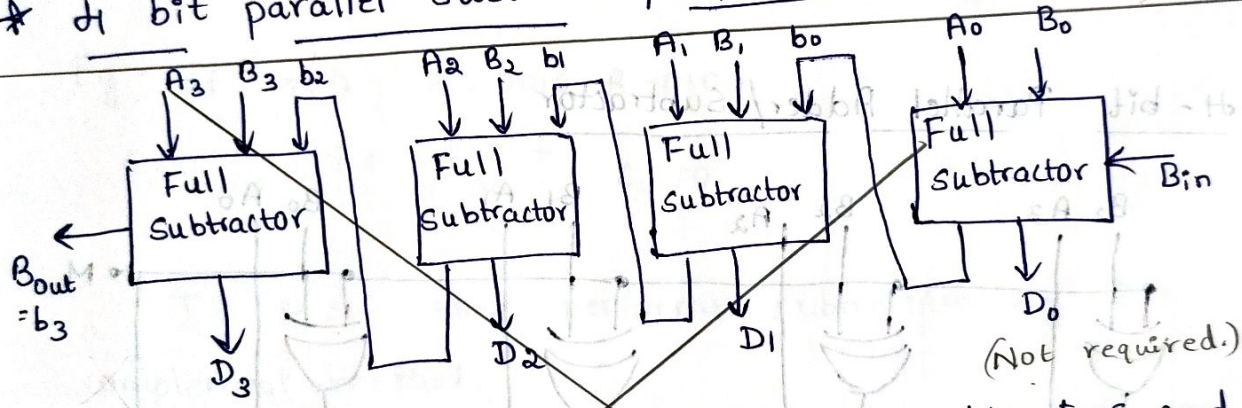
Bout

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

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

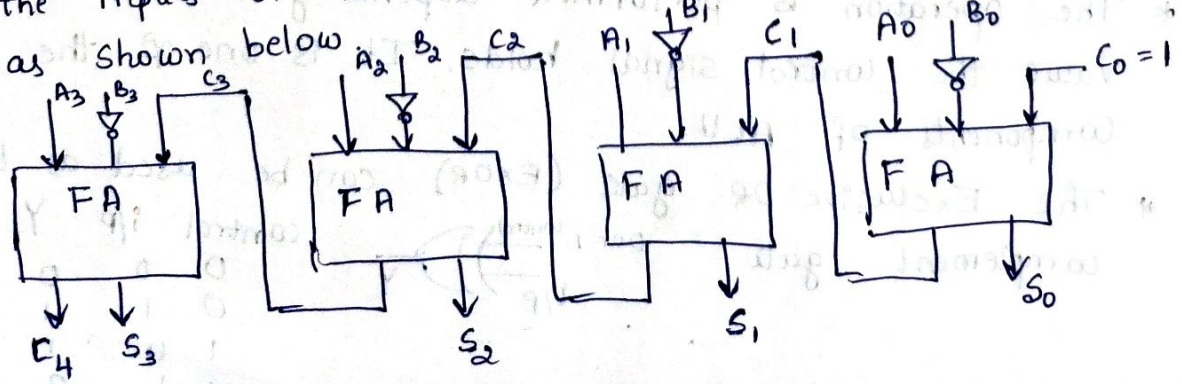


* n bit parallel subtractor / ripple subtractor:



* These cascaded forms are also ripple subtractors and the difference outputs do not appear simultaneously with the application of inputs. The output at any stage will be valid only after the borrow of the previous stage has been generated.

* We know that generally subtraction is performed using 2's complement method. This concept is used to alter the inputs of the n-bit full adder to perform subtraction as shown below



* The inverter (NOT gate) is used to obtain 1's complement of the subtrahend (B) and the carry at bit position 0, C_0 is set to 1 in order to add 1 to the 1's complement to obtain 2's complement. This is further added to A to carry out the arithmatic subtraction.

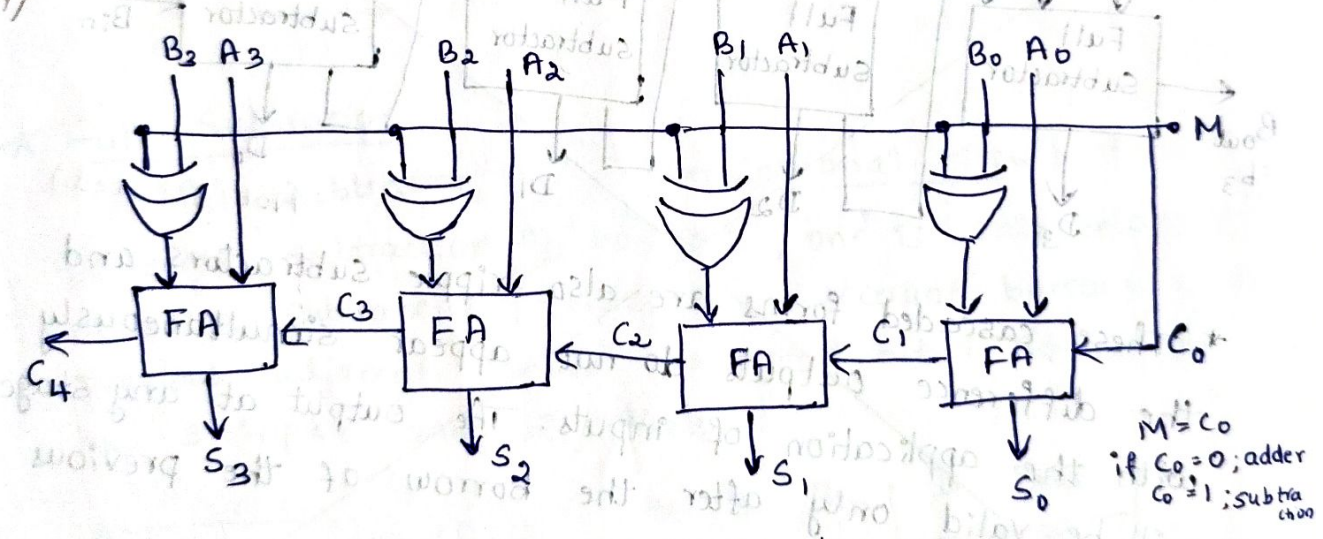
* The process continues till the last full adder uses the carry bit to add with its input and 2's complement of B to generate last bit of output S_3 along with carry bit C_4.

* If carry generated ($C_4 = 1$) then ignore carry and result will be difference b/w a & b (i.e a>b & a=b case).

* If no carry generated ($C_4 = 0$) then difference is 2's complement of the result with negative sign.

* 4-bit Parallel Adder/Subtractor

Imp



* In digital circuits, a binary adder/subtractor is capable of performing both addition and subtraction of binary numbers in one circuit itself.

* The operation is performed depending on the binary value the control signal holds. It is one of the components of ALU.

* The Exclusive OR gate (EXOR) can be used as true complement gate.

control	i/P	Y
0	0	0
0	1	1
1	0	1
1	1	0

When the mode input (M) is at a low logic, ie '0', the circuit acts as an adder and when the mode input is at a high logic ie '1', the circuit acts as a subtractor.

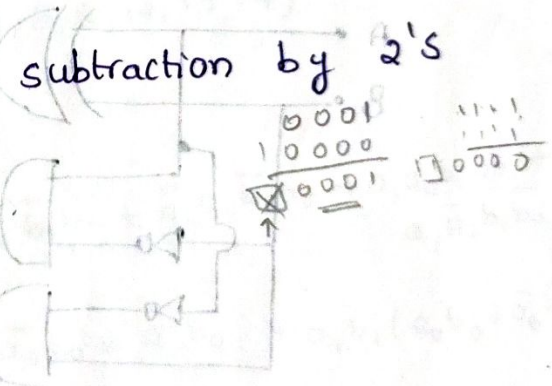
* The XOR gate connected in series recieves input M and one of the inputs B.
* When M is at logic low, $B \oplus 0 = B$. The full adders receive the value of B, the input carry is 0, and circuit performs A plus B.
* When M is at logic high, $B \oplus 1 = \overline{B}$ & $C_0 = 1$. The B inputs are complemented, and a '1' is added through the input carry. The circuit performs the operation A plus the 2's complement of B.

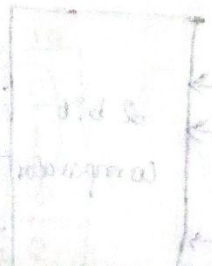
Eg:- If M=0; A=0010, B=0100

C=0 $\therefore$ S = 0010 + 0100 = 0110 $C_4 = 0$

If M=1, then performs subtraction by 2's complement method.

$\therefore$ 1011 + 111 = 1100 $\quad$ 0010 + 1100 = 1110 $\quad C_4 = 0$.





* Binary Comparators :-

Comparators are designed to compare the magnitude of two binary numbers and indicate whether one is greater than, ~~the~~ less than or equal to other.

One-bit comparator :-

One-bit comparators compare two one-bit numbers A and B and produce three outputs $A=B$, $A>B$ and $A<B$.

Inputs A	Inputs B	outputs $A=B$	outputs $A>B$	outputs $A<B$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

truth table

∴ O/p equations are

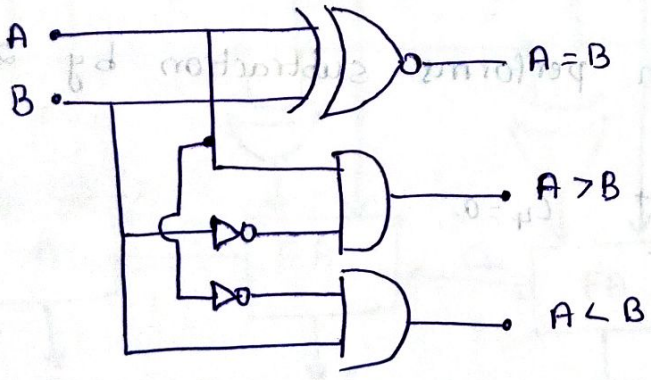
$A=B : \overline{A}\,\overline{B} + AB$

$= \overline{A \oplus B}$

$A>B = A\,\overline{B}$

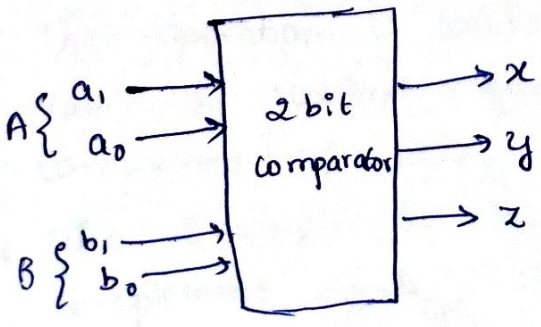
$A<B = \overline{A}\,B$

∴ The implementation / logic circuit is.



2-bit comparator :-

A two-bit comparator will accept 2 two-bit numbers and generates 3 outputs.



Truth table for 2-bit comparator

Cell No	A: a_1	a_0	B: b_1	b_0	A=B: x	A>B: y	A<B: z
0	0	0	0	0	1	0	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1
4	0	1	0	0	0	1	0
5	0	1	0	1	1	0	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	0	1
8	1	0	0	0	0	1	0
9	1	0	0	1	0	1	0
10	1	0	1	0	1	0	0
11	1	0	1	1	0	0	1
12	1	1	0	0	0	1	0
13	1	1	0	1	0	1	0
14	1	1	1	0	0	1	0
15	1	1	1	1	1	0	0

$\therefore$ $A=B = f(a_1, a_0, b_1, b_0) = \sum(0, 5, 10, 15)$

$A>B = f(a_1, a_0, b_1, b_0) = \sum(4, 8, 9, 12, 13, 14)$

$A<B = \sum(1, 2, 3, 6, 7, 11)$

Using K Map:

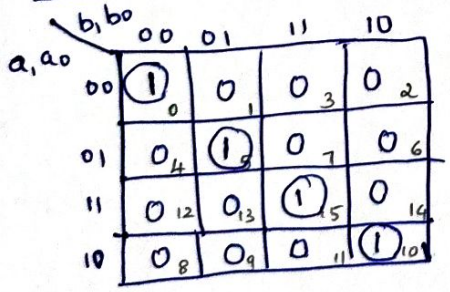
$f = \bar{a}_1 \bar{a}_0 \bar{b}_1 \bar{b}_0 + \bar{a}_1 a_0 \bar{b}_1 b_0 + a_1 a_0 b_1 b_0 + a_1 \bar{a}_0 b_1 \bar{b}_0$

$\therefore f = \bar{a}_1 \bar{b}_1 (\bar{a}_0 \bar{b}_0 + a_0 b_0) + a_1 b_1 (a_0 b_0 + \bar{a}_0 \bar{b}_0)$

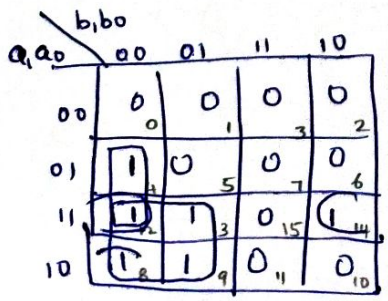
$= (\bar{a}_0 \bar{b}_0 + a_0 b_0)(a_1 b_1 + \bar{a}_1 \bar{b}_1)$

$= \overline{(a_0 \oplus b_0)}\ \overline{(a_1 \oplus b_1)}$

A=B

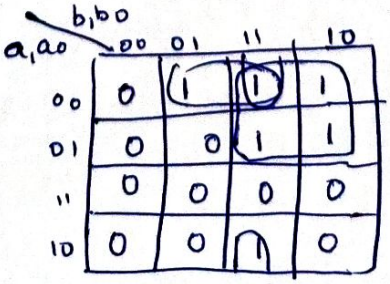


A>B



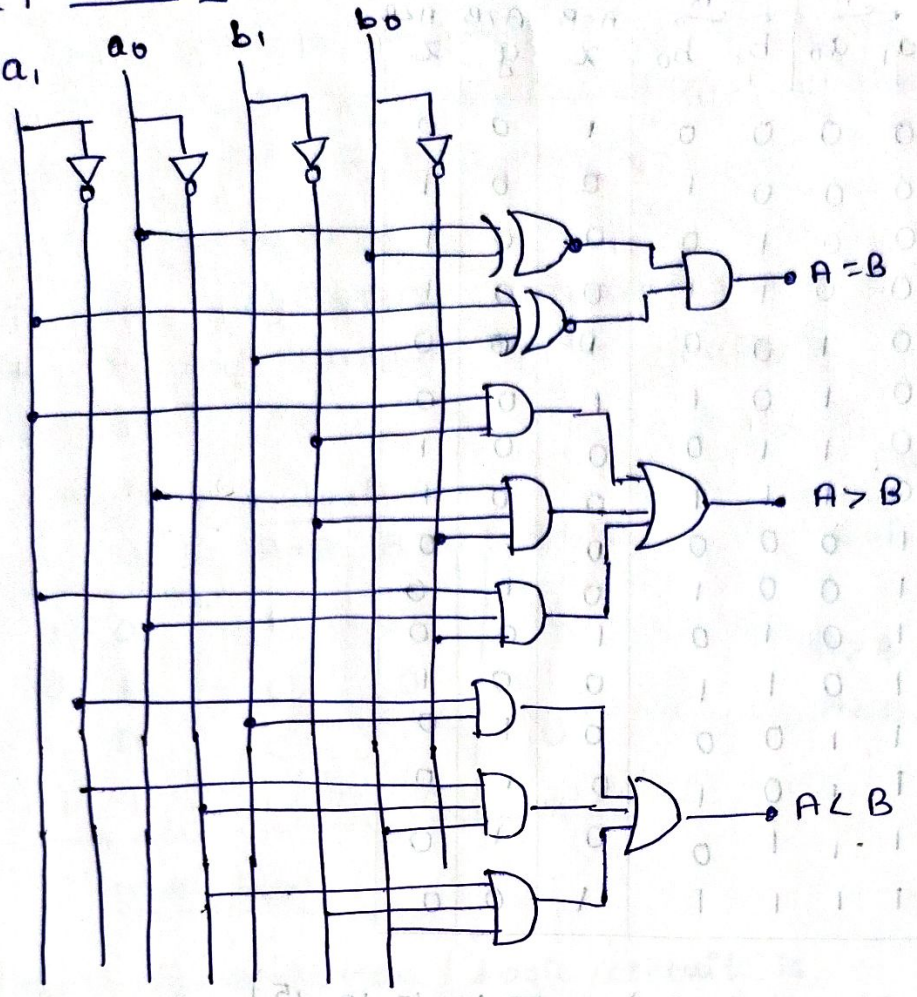
$f = a_1 \bar{b}_1 + a_0 \bar{b}_1 \bar{b}_0 + a_1 a_0 \bar{b}_0$

A<B



$f = \bar{a}_1 b_1 + \bar{a}_1 \bar{a}_0 b_0 + \bar{a}_0 b_1 b_0$

Implementation

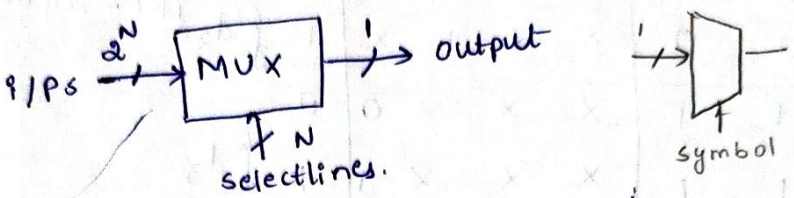


★ Multiplexers (MUX)

* A MUX is a digital switch that has multiple inputs and single output. The select lines determine which input is connected to the output.

Or

A multiplexer places data at one of its 2^n input lines selected by an n-bit address, on its output line.



Note:-

MUX is also called as data selector

Multiplexer → Multiplex means many into one

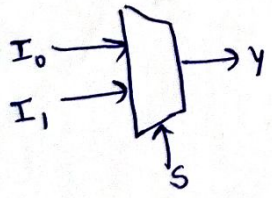
Enable input is also called as strobe input.

MUX types:

1. 2 to 1 MUX (1 select line)
2. 4 to 1 (2 " ")
3. 8 to 1 " (3 " ")
4. 16 to 1 " (4 " ")

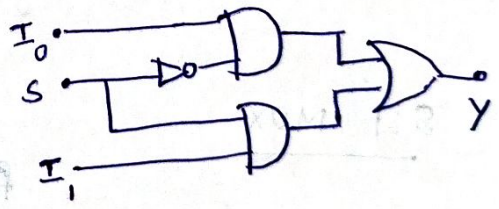
Imp

★ 2 to 1 MUX

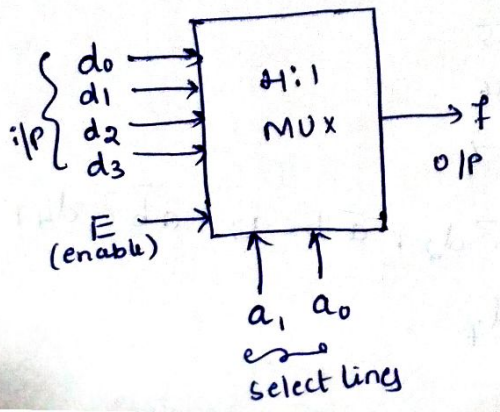


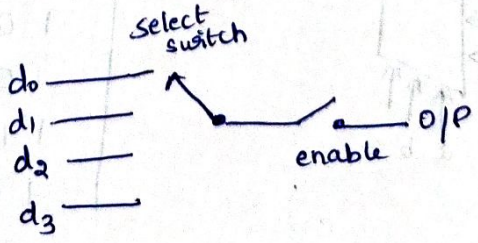
S	Y
0	I_0
1	I_1

$\therefore Y = I_0\bar{S} + I_1 S$



★ 4 to 1 MUX





Functional table

E	a_1	a_0	d_0	d_1	d_2	d_3	f	
0	x	x	x	x	x	x	0	
1	0	0	0	x	x	x	0	d_0
1	0	0	1	x	x	x	1	d_0
1	0	1	x	0	x	x	0	d_1
1	0	1	x	1	x	x	1	d_1
1	1	0	x	x	0	x	0	d_2
1	1	0	x	x	1	x	1	d_2
1	1	1	x	x	x	0	0	d_3
1	1	1	x	x	x	1	1	d_3

When E = 0, irrespective of i/p f = 0

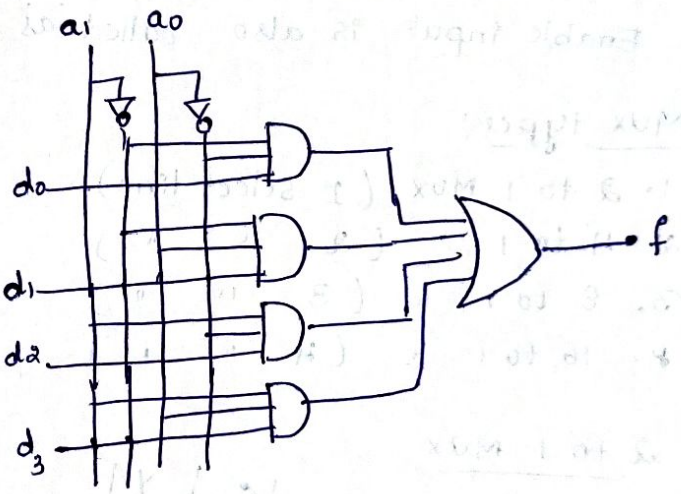
If E = 1; data at the selected i/p appears at the o/p.

Not required

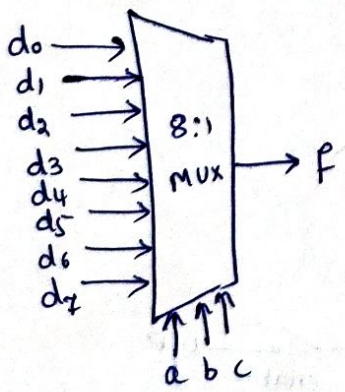
Condensed table

E	a_1	a_0	f
0	x	x	0
1	0	0	d_0
1	0	1	d_1
1	1	0	d_2
1	1	1	d_3

$\therefore f = \bar{a}_1 \bar{a}_0 d_0 + \bar{a}_1 a_0 d_1 + a_1 \bar{a}_0 d_2 + a_1 a_0 d_3$



8:1 MUX



a	b	c	f
0	0	0	d_0
0	0	1	d_1
0	1	0	d_2
0	1	1	d_3
1	0	0	d_4
1	0	1	d_5
1	1	0	d_6
1	1	1	d_7

Implement (HW) using gates

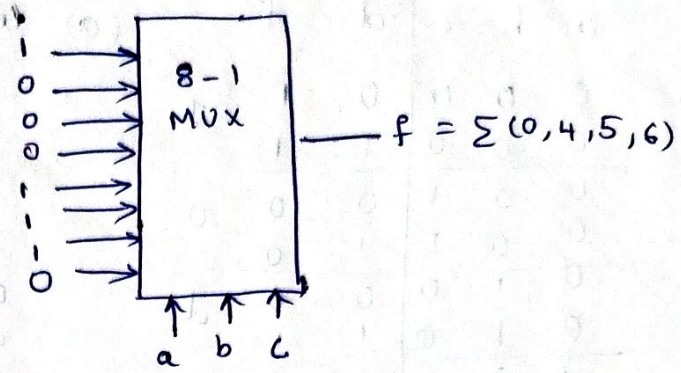
$\therefore f = \bar{a}\bar{b}\bar{c} d_0 + \bar{a} b c d_1 + \bar{a} b \bar{c} d_2 + \bar{a} b c d_3 + a \bar{b} \bar{c} d_4 + a \bar{b} c d_5 + a b \bar{c} d_6 + a b c d_7$

* Logic design using multiplexer :-

1. Implement $f(a,b,c) = \Sigma(0,4,5,6)$ using 8 to 1 MUX

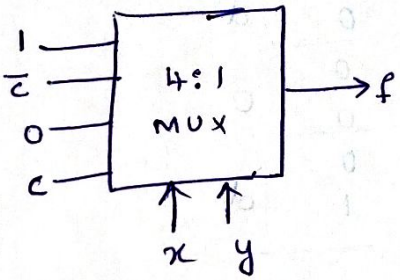
	a	b	c	f	
0	0	0	0	1	d_0
1	0	0	1	0	d_1
2	0	1	0	0	d_2
3	0	1	1	0	d_3
4	1	0	0	1	d_4
5	1	0	1	1	d_5
6	1	1	0	1	d_6
7	1	1	1	0	d_7

$\therefore$ $d_0 = d_5 = d_4 = d_6 = 1$
$d_1 = d_2 = d_3 = d_7 = 0$



2) $f(a,b,c) = \Sigma(0,1,2,7)$ using 4:1 MUX.

a	b	c	f	
0	0	0	1	1
0	0	1	1	
0	1	0	1	$\bar{c}$
0	1	1	0	
1	0	0	0	0
1	0	1	0	
1	1	0	0	c
1	1	1	1	

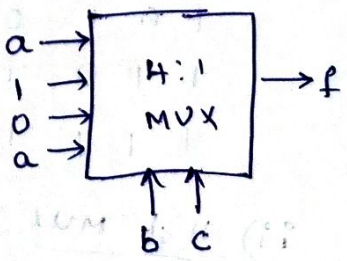


3) $f(a,b,c) = \Sigma(1,4,5,7)$ using 4:1 MUX and using b and c as address lines / select lines.

a	b	c	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Rearrange

a	b	c	f	
0	0	0	0	a
1	0	0	1	
0	0	1	1	1
1	0	1	1	
0	1	0	0	0
1	1	0	0	
0	1	1	0	a
1	1	1	1	



4) Implement the following function,

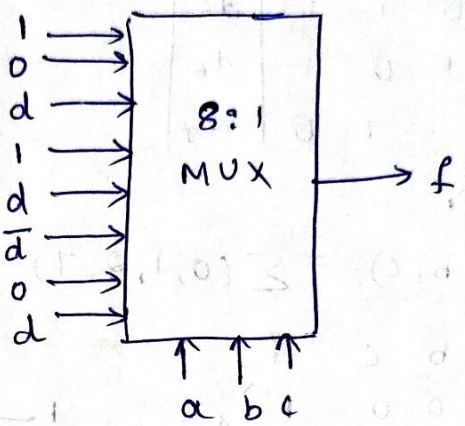
$f(a,b,c,d) = \sum m(0,1,5,6,7,9,10,15)$

a) Using 8:1 MUX ; treat a, b, c as select lines

~~(b) Using 4:1 MUX ; treat a & b as select lines~~

a	b	c	d	f	
0	0	0	0	1	1
0	0	0	1	1	
0	0	1	0	0	0
0	0	1	1	0	
0	1	0	0	0	d
0	1	0	1	1	
0	1	1	0	1	1
0	1	1	1	1	
1	0	0	0	0	d
1	0	0	1	1	
1	0	1	0	1	$\bar{d}$
1	0	1	1	0	
1	1	0	0	0	0
1	1	0	1	0	
1	1	1	0	0	d
1	1	1	1	1	

(a) Compare d with f ; a,b,c select lines



5) Implement the function $f(x,y,z) = \pi M(0,2,4,5)$ using

i) 8:1 MUX

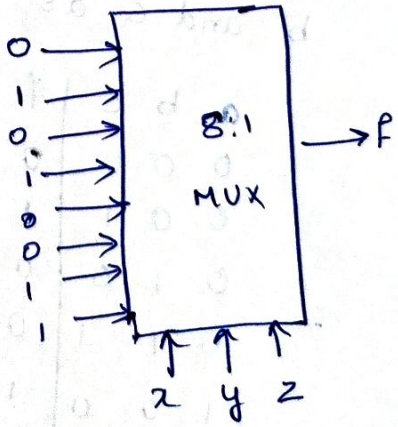
ii) 4:1 MUX by considering x and z as select lines.

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

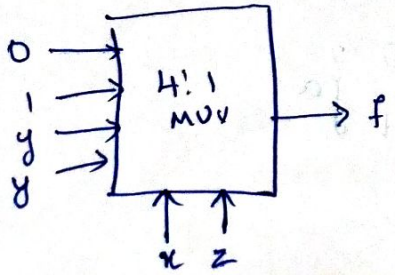
If x z select lines.

x	z	y	f	
0	0	0	0	0
0	0	1	0	
0	1	0	1	1
0	1	1	1	
1	0	0	0	y
1	0	1	1	
1	1	0	0	y
1	1	1	1	

i) 8:1 MUX



ii) 4:1 MUX



6) Implement $u = a + b\bar{c}$ using 4:1 MUX

converting into SOP,

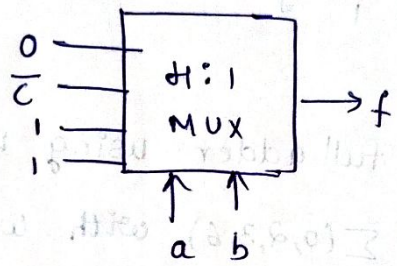
$u = a + b\bar{c}$

$= a(b+\bar{b})(c+\bar{c}) + b\bar{c}(a+\bar{a})$

$= abc + ab\bar{c} + a\bar{b}c + a\bar{b}\bar{c} + ab\bar{c} + \bar{a}b\bar{c}$

$= \underset{(1,1,0)}{ab\bar{c}} + \underset{(1,0,1)}{a\bar{b}c} + \underset{(1,0,0)}{a\bar{b}\bar{c}} + \underset{(0,1,0)}{\bar{a}b\bar{c}} + \underset{(1,1,1)}{abc}$

a	b	c	u	
0	0	0	0	0
0	0	1	0	
0	1	0	1	$\bar{c}$
0	1	1	0	
1	0	0	1	1
1	0	1	1	
1	1	0	1	
1	1	1	1	1

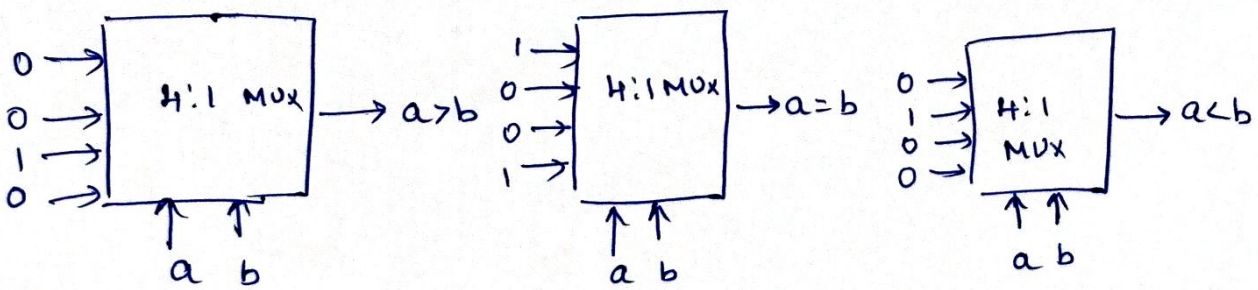


HW Implement (1) $u = a + b\bar{c} + b$ using 8:1 & 4:1 MUX

(2) $f(a,b,c) = \Sigma(0,1,3,4)$ using 4:1 & 8:1 MUX

7) Implement 1 bit comparator using 4:1 MUX.

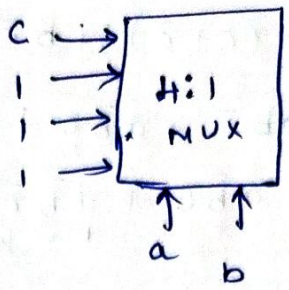
a	b	a>b	a=b	a<b
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0



8) Implement following expression using 4:1 MUX.

$f(a,b,c) = a+b+c$

a	b	c	f	
0	0	0	0	} c
0	0	1	1	
0	1	0	1	} 1
0	1	1	1	
1	0	0	1	} 1
1	0	1	1	
1	1	0	1	} 1
1	1	1	1	



HW: 1) Implement full adder using 4x1 Multiplexer.

2) $f(w,x,y) = \Sigma(0,2,3,6)$ with w, x as select lines and w, x, y as select lines.

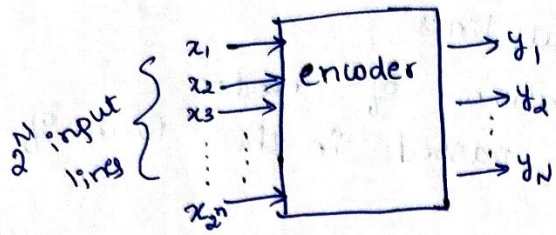
w, x, y — 8:1 MUX.

w, x — 4:1 MUX

★ Encoder:

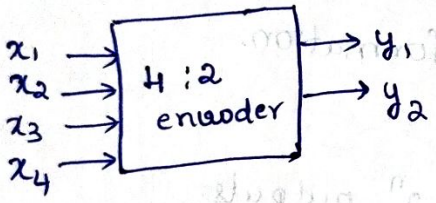
An encoder is combinational circuit that has a maximum of 2^n input lines and n output lines, hence it encodes the information from 2^n inputs to an n-bit code

* Used to convert one binary code to another.



★ 4 x 2 (4 to 2) Encoder :-

* 4 input lines and 2 output lines.

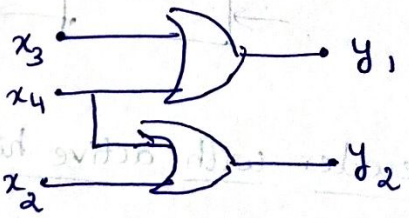


x_1	x_2	x_3	x_4	y_1	y_2
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

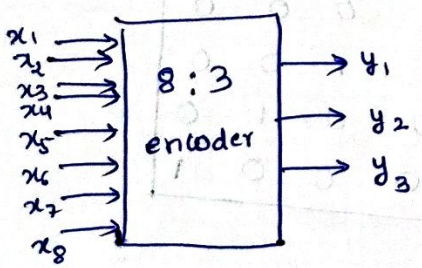
∴ logical expressions,

$y_1 = x_3 + x_4$

$y_2 = x_2 + x_4$



★ 8 to 3 encoders :-



Truth table

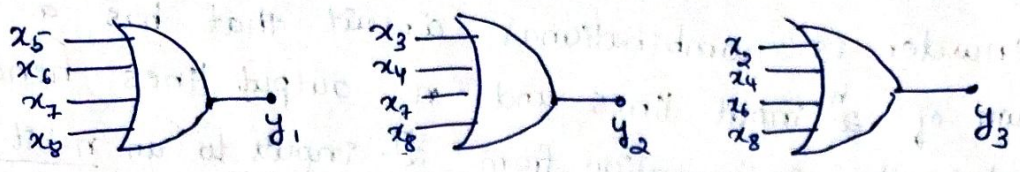
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y_1	y_2	y_3
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

∴ logical expression,

$y_1 = x_5 + x_6 + x_7 + x_8$

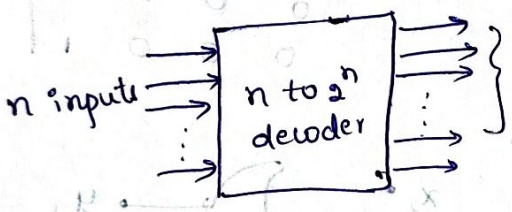
$y_2 = x_3 + x_4 + x_7 + x_8$

$y_3 = x_2 + x_4 + x_6 + x_8$



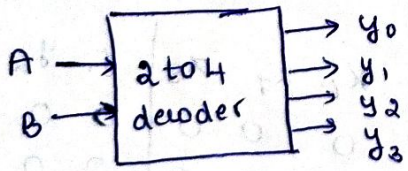
* Decoders :-

* Decoders are the combinational circuit which has n input lines and 2^n output lines.
* It performs reverse operation of encoder.
* The binary information is passed in the form of 'n' input lines
* The output lines define 2^n bit code for the binary information.
* At a time, only one input line is activated for simplicity. The produced 2^n bit output code is equivalent to the binary information.



* 2 to 4 decoder with active high outputs :-

* Here there are 2 input lines and $2^2 = 4$ output lines.



Truth table

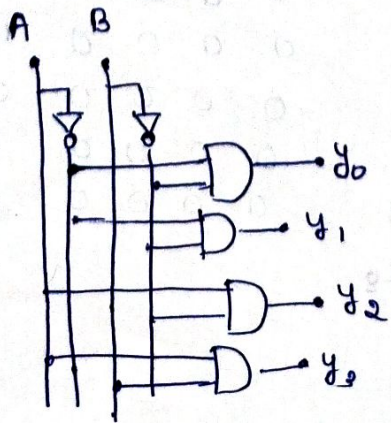
A	B	y_0	y_1	y_2	y_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

$\therefore y_0 = \overline{A}\,\overline{B}$

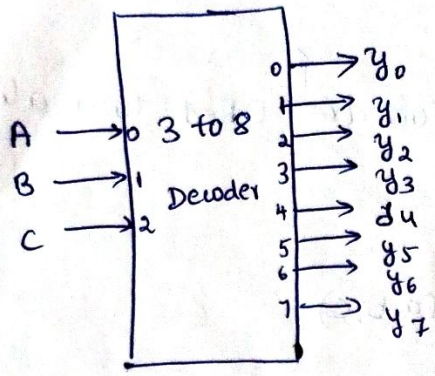
$y_1 = \overline{A}B$

$y_2 = A\overline{B}$

$y_3 = AB$



★ 3 to 8 decoder with active high outputs :-

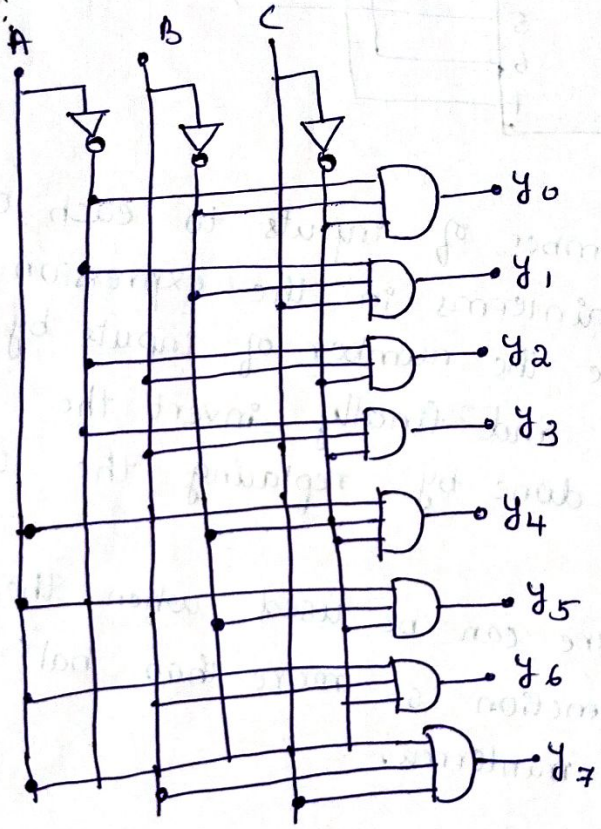


Truth table :-

A	B	C	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_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

Implementation :

$y_0 = \bar{A}\bar{B}\bar{C}$
$y_1 = \bar{A}\bar{B}C$
$y_2 = \bar{A}B\bar{C}$
$y_3 = \bar{A}BC$
$y_4 = A\bar{B}\bar{C}$
$y_5 = A\bar{B}C$
$y_6 = AB\bar{C}$
$y_7 = ABC$

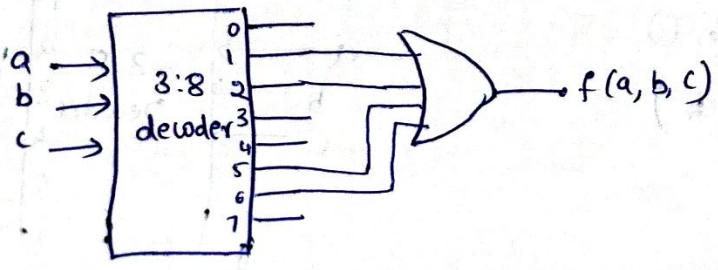


★ Logic design using decoder :-

1) Implement $f(a,b,c) = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}c + ab\bar{c}$ using decoder.

$f(a,b,c) = \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}c + abc = \Sigma(1,2,5,6)$

It can be implemented using 3:8 decoder

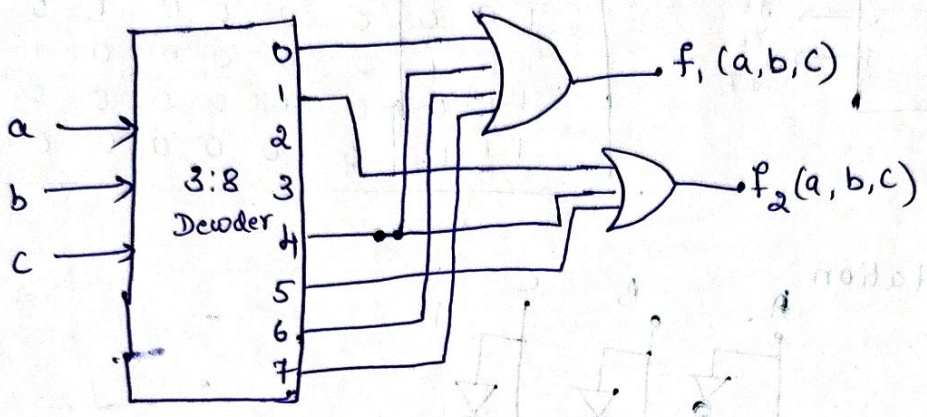


2) Implement the following functions using a 3 to 8 line decoder.

$f_1(a,b,c) = \Sigma(0,4,6,7)$

$f_2(a,b,c) = \Sigma m(1,4,5)$

The minterms require for each function are ORed to realize the function as shown below.



Note :- The number of inputs to each OR gate will be number of minterms in the expression. We can reduce the number of inputs by implementing $\bar{f}$ instead of f and finally invert the OR o/p by f. This can be done by replacing the OR gate with a NOR gate.

This procedure can be used when the number of minterms in a function is more than half the total number of possible minterms.

3) Implement the following functions using a decoder minimizing the number of inputs to be summed.

$f_1(a,b,c) = \Sigma m(0,2,3,5,6,7)$

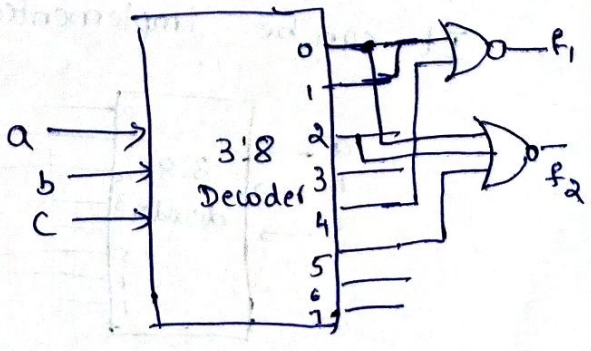
$f_2(a,b,c) = \Sigma m(1,3,4,6,7)$

soln: $f_1(a,b,c) = \Sigma m(0,2,3,5,6,7) = \Pi(1,4)$ Implementation is

$\bar{f} = \Sigma(1,4)$

$f_2 = \Sigma(1,3,4,6,7) = \Pi(0,2,5)$

$\bar{f_2} = \Sigma(0,2,5)$



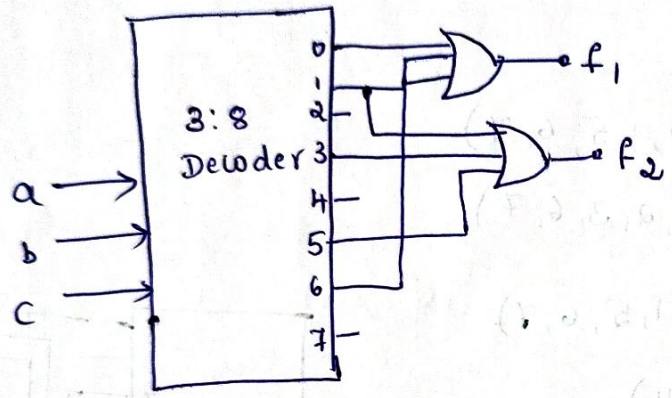
4) Implement the following functions expressed in maxterm canonical form in two possible ways using a 3:8 decoder.

$$f_1(a,b,c) = \Pi M(2,3,4,5,7)$$
$$f_2(a,b,c) = \Pi M(0,2,4,6,7)$$

1) We know,

$$f_1(a,b,c) = \Pi M(2,3,4,5,7) = \Sigma(0,1,6)$$
$$f_2(a,b,c) = \Pi M(0,2,4,6,7) = \Sigma(1,3,5)$$

∴ Implementation is,

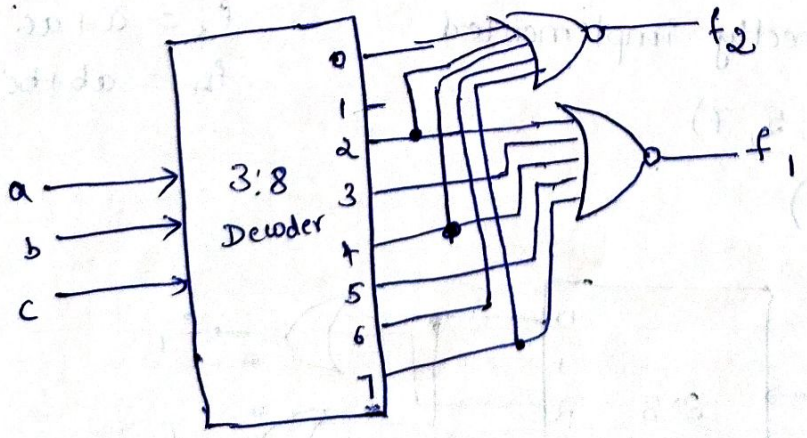


OR

2) Can also implement by $\bar{f_1}$ & $\bar{f_2}$ replacing OR by NOR gate

$$f_1 = \Sigma(0,1,6) \Rightarrow \bar{f_1} = \Sigma(2,3,4,5,7)$$
$$f_2 = \Sigma(1,3,5) \Rightarrow \bar{f_2} = \Sigma(0,2,4,6,7)$$

∴



(H/W)

1) Implement 1 bit comparator and half adder (sum & carray) using 2:4 line decoder

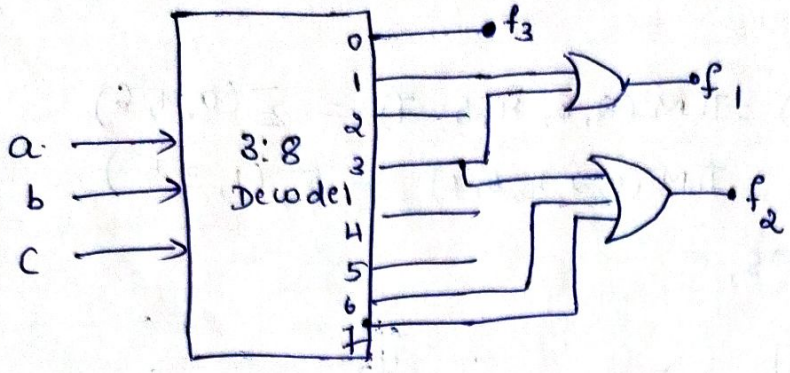
2) Implement full adder (sum & carry) using 3:8 decoder.

5) Implement following function with minimal gate i/ps

$f_1(a,b,c) = \sum m(1,3)$

$f_2(a,b,c) = \sum m(3,6,7)$

$f_3(a,b,c) = \sum m(0)$



6) $f_1(a,b,c) = \sum m(0,1,5,6,7)$

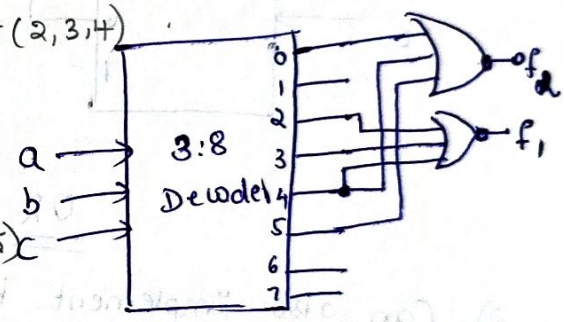
HW $f_2(a,b,c) = \sum m(1,2,3,6,7)$

$f_1(a,b,c) = \sum m(0,1,5,6,7) = \pi(2,3,4)$

$\therefore \overline{f_1} = \sum m(2,3,4)$

$f_2(a,b,c) = \sum m(1,2,3,6,7) = \pi(0,4,5)$

$\overline{f_2} = \sum m(0,4,5)$



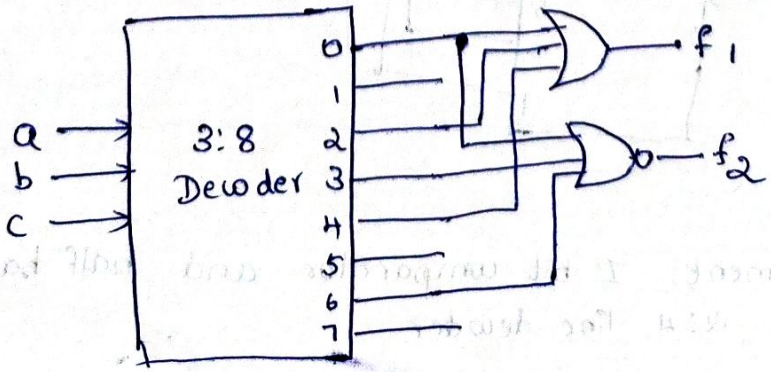
7) $f_1(a,b,c) = \sum m(0,2,4)$

$f_2(a,b,c) = \sum m(1,2,4,5,7)$

f_1 can be directly implemented.

$f_2 = \sum(1,2,4,5,7) = \pi(0,3,6)$

$\overline{f_2} = \sum(0,3,6)$



(8) $f_1(x,y,z) = \overline{x}y + z$

$f_2(x,y,z) = xy + yz$

$f_3 = \overline{a} + ac$

$f_4 = ab + bc + ac$

HW Implement the following using decoder:

(3) 1) $f_1(x, y, z) = \overline{xy} + z$

$f_1 = \bar{x} + \bar{y} + z$

$f_1 = \bar{x}(y+\bar{y})(z+\bar{z}) + \bar{y}(x+\bar{x})(z+\bar{z}) + z(x+\bar{x})(y+\bar{y})$

$\therefore f_1 = \bar{x}y z + \bar{x}y\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z} + x\bar{y}z + x\bar{y}\bar{z} + \bar{x}\bar{y}z + \bar{x}\bar{y}\bar{z} + xyz + x\bar{y}z + \bar{x}yz + \bar{x}\bar{y}\bar{z}$

$f_1 = \underset{(000)}{\bar{x}\bar{y}\bar{z}} + \underset{(011)}{\bar{x}yz} + \underset{(010)}{\bar{x}y\bar{z}} + \underset{(001)}{\bar{x}\bar{y}z} + \underset{(101)}{x\bar{y}z} + \underset{(100)}{x\bar{y}\bar{z}} + \underset{(111)}{xyz}$

$f_1 = \sum(0, 1, 2, 3, 4, 5, 7) = \pi(6)$

$\bar{f}_1 = \sum(6)$

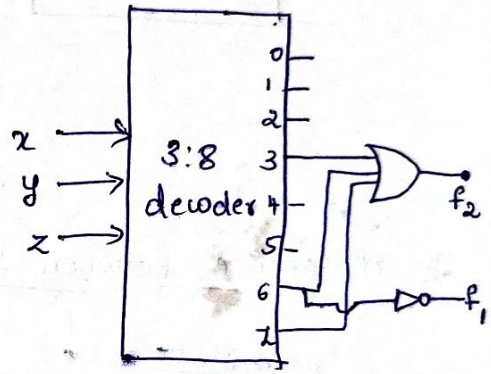
2) $f_2 = xy + yz$

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

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

$f_2 = xy\bar{z} + \bar{x}yz + xyz$

$f_2 = \sum m(3, 6, 7)$



9) $f_1 = \bar{a} + ac$

$= \bar{a}(b+\bar{b})(c+\bar{c}) + ac(b+\bar{b})$

$= \bar{a}bc + \bar{a}b\bar{c} + \bar{a}\bar{b}c + \bar{a}\bar{b}\bar{c} + abc + a\bar{b}c$

$= \bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + \bar{a}bc + a\bar{b}c + abc$

$f_1 = \sum(0, 1, 2, 3, 5, 7)$

$\bar{f}_1 = \sum(4, 6)$

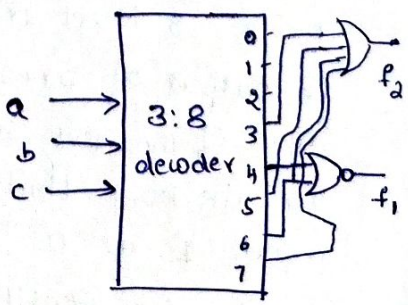
$f_2 = ab + bc + ac$

$= ab(c+\bar{c}) + bc(a+\bar{a}) + ac(b+\bar{b})$

$= abc + ab\bar{c} + abc + \bar{a}bc + abc + a\bar{b}c$

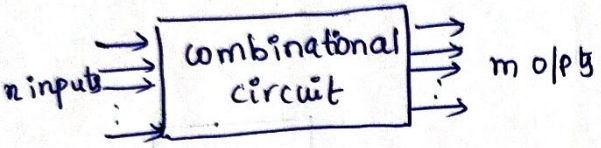
$= abc + ab\bar{c} + \bar{a}bc + a\bar{b}c$

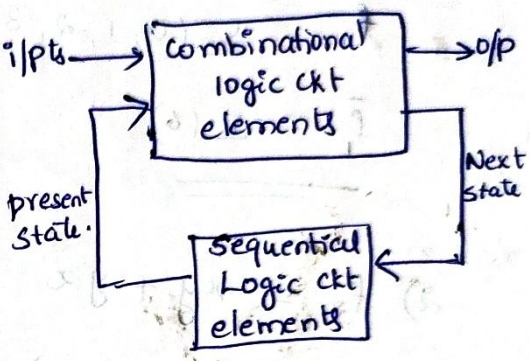
$f_2 = \sum(3, 5, 6, 7)$



★ Imp Difference between Combinational and Sequential logic circuits :-

Combinational	Sequential
1. Output depends only on present input	1. Output depends on present input and past output
2. Memory element is absent	2. Memory element is present.
3. No clock signal is applied	3. Clock signal is required
4. No feedback loop	4. Feedback is present



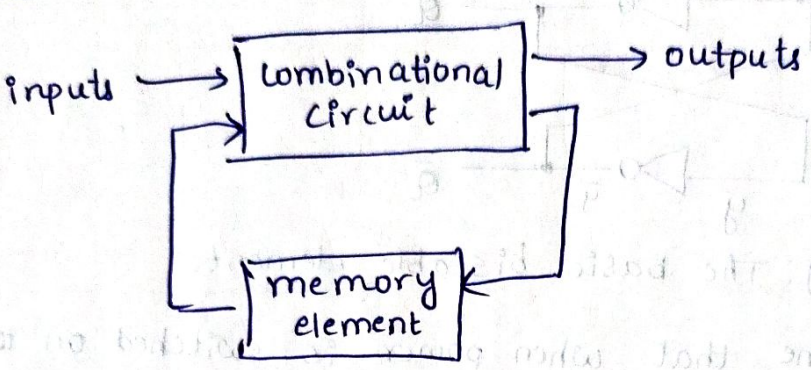


★ Difference between Latches and Flip-flops :-

Latches	Flipflops
1. Latches are building blocks of sequential circuits and built from logic gates.	1. Flipflops are also building blocks of sequential circuits, built from the latches.
2. Latch continuously checks its inputs and changes its output correspondingly	2. Flipflop continuously checks its inputs and changes its output only at times determined by clocking signal.
3. It works based on enable input	3. Works on the basis of clockpulses.
4. It is level triggered. ie output of present state and input of the next state depends on the level that is binary input '1' or '0'	4. It is an edge triggered, it means output and next state input changes when there is change in clock pulse, whether it may a +ve or -ve pulse.
5. Latch is sensitive to the duration of the pulse.	5. Flipflop is sensitive to a signal change.

Sequential circuits

* The outputs at any instant are dependent not only upon the inputs present at that instant but also upon past history of inputs.
* Sequential circuits are said to have memory.
* All sequential circuits require the existance of feedback.



Types

1. Synchronous sequential circuits:-
   * A synchronous sequential circuit is one in which its behaviour is determined by the values of signals at only discrete instants of time.
   * These circuits are controlled by master clock.
   * Circuits change their state as well as their output at specific clock instances.

2. Asynchronous sequencial circuits
   * Operation are not controlled by any clock pulses.
   * The output responds immediately to a change in input.

Note:

Memory element in sequential circuit is flipflop. A flipflop is a sequential circuit which can store a '1' or '0' indefinitely. Thus it has 2 stable state. It (~~continuously~~) continues indefinitely in one (~~sta~~) of these two stable states until a change is promoted by the required change in its input.

* The Basic Bistable Element :-

A circuit which can indefinitely store a '1' or '0' is called the basic bistable element.

* It consists of cross coupled inverters with two outputs named Q and $\bar{Q}$.

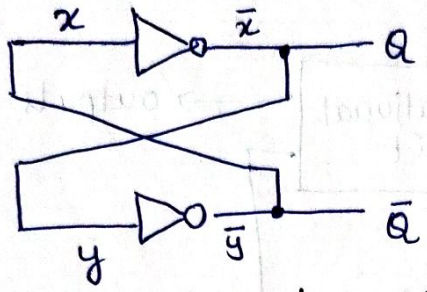


fig: The basic bistable element.

Let us assume that when power is switched on to the circuit,

input x is set to 1, therefore $\bar{x}=0$, $Q=0$.

This implies $y=0$ & $\bar{y}=\bar{Q}=1$. So, $Q=0$, $\bar{Q}=1$ and circuit continues in this state until power to the circuit is switched off.

Similarly when power is ON, if $x=0$, $\bar{x}=1$ & $Q=1$. this implies $y=1$, $\bar{y}=0$ & $\bar{Q}=0$.

Now $Q=1$ & $\bar{Q}=0$ is other stable state of the circuit in which it indefinitely remains until power to the circuit is switched off.

* It is used to store binary symbols.
* Stored symbol is refferred to as content or state of the element.
* When the device is storing '1' is said to be 'set' or in '1-state'
* When storing '0' is said to be 'reset' or in '0-state'.

* Flip-flop :-

A flipflop is a bistable circuit with input lines.

* The flipflop remains in one of its two stable states until power is switched off or untill an input signal triggers a change in state.
* Inputs to flip-flop can be of two types.
  1. Synchronous or gated inputs :- signal change produces an change in output only when some control signal occurs.
  2. Asynchronous or direct inputs :- signal change produces an immediate change in output.

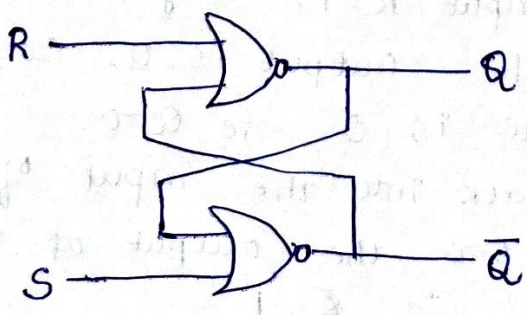
* Latches :-

A latch is a class of flip-flops whose output responds immediately to appropriate changes in the input inspite of the presence of an enable or clock input.

* Latch has feedback path to retain the information hence it can be a memory device.
* Latch can store one bit of information as long as the device is powered on.
* These are level triggered devices.

1) SR Latch :- (Set-Reset Latch)

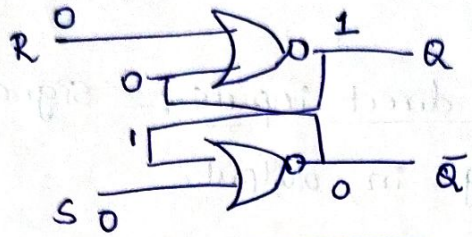
It is a circuit with two cross-coupled NOR gates, two inputs S for set and R for Reset and 2 outputs Q and $\overline{Q}$.



Operation:-

* When S=0, R=0 :- Hold Mode

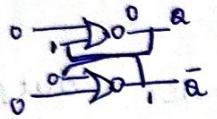
This is the normal resting state of the circuit and it has no effect on the output states. Q and $\bar{Q}$ will remain in whatever state they were in prior to the occurrence of this input condition. So here next state is same as present state. The mode of operation is HOLD (no change) mode.



Let Q=1 & $\bar{Q}$=0 be the present state if S=0, R=0 then Q=1 & $\bar{Q}$=0 (same as earlier case)

∴ No change.

or



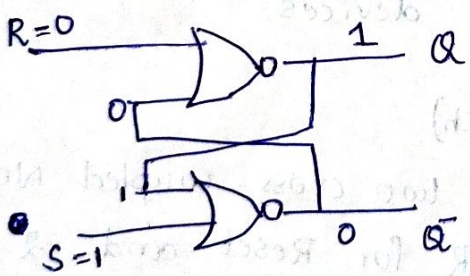
Present state = Next state
No change

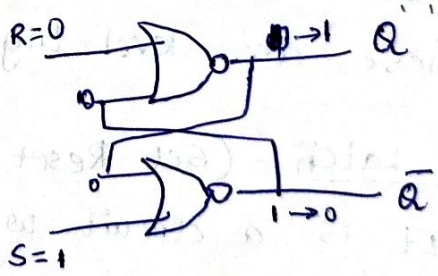
* When S=1, R=0 : SET Mode

Consider the input S=1. Anytime the input of a NOR gate is 1 the output is 0. So, output of the second NOR gate is 0, ie $\bar{Q}$=0.

$\bar{Q}$=0 is fed back into the input of the first NOR gate. So, with R=0, the output of first NOR gate is 1. ie. Q=1.

OR		NOR
0	0	1
0	1	0
1	0	0
1	1	0

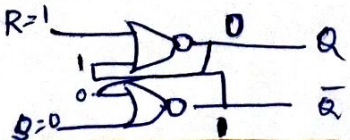




* When S=0, R=1 : RESET Mode

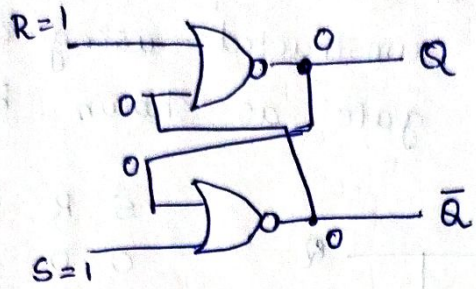
Consider the input R=1. Any time the input of NOR gate is 1 the output is 0. So, the output of first NOR gate is 0, ie Q=0.

Q=0 is fed back into the input of second NOR gate. So, with S=0, the output of the second NOR gate is 1' ie $\bar{Q}$=1



* When S=1, R=1 :. Invalid Mode

If S=1, R=1, This condition tries to Set and reset the NOR gate latch at the same time, it produces $Q = \bar{Q} = 0$. This is an unexpected condition and is not used. The two outputs should be the inverse of each other.



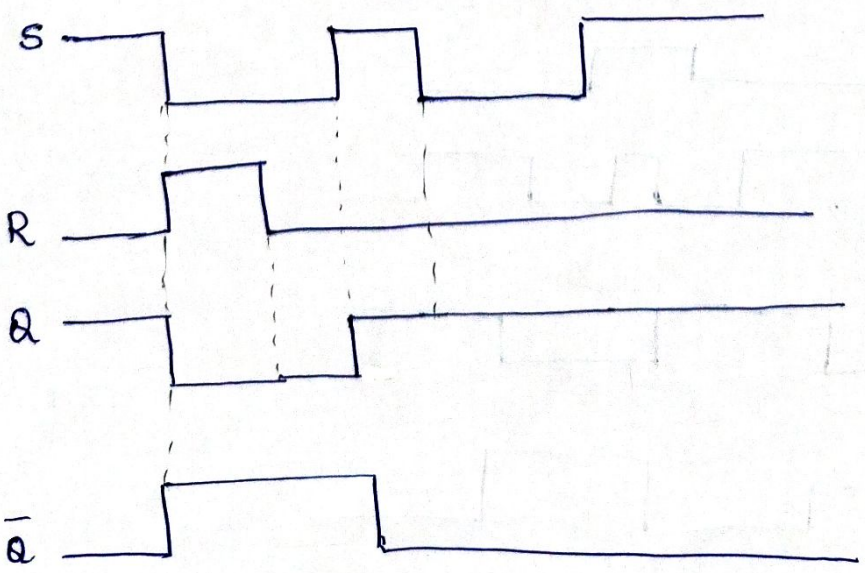
:. Invalid

If the inputs are returned to 1 simultaneously, the output states are unpredictable. This condition should not be used, and when circuits are constructed the design should avoid S = R = 1 condition.

:. The truth table is,

Input		output	
S	R	Q^+	$\bar{Q}^+$
0	0	previous state (Q $\bar{Q}$)	
0	1	0	1
1	0	1	0
1	1	Illegal / forbidden state.	

Timing diagram of SR latch



* Gated SR Latch

In many digital systems, it is desirable that the circuit responds only at or during some prescribed time decided by another input called the enable, gate or clock input.

A gated SR latch has synchronous inputs S and R along with a enable or clock input C.

A gated SR latch is constructed using a $\bar{S}\bar{R}$ latch together with NAND gate as shown below.

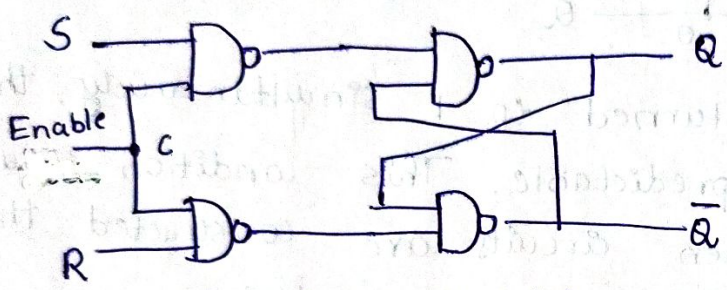


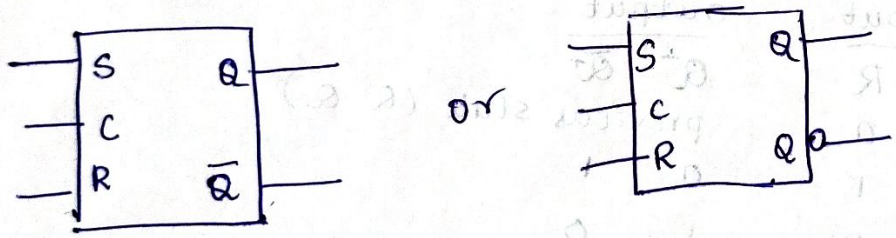
fig: Gated SR latch

S	R	C	Q^+	$\bar{Q}^+$
0	0	1	Q	$\bar{Q}$
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1*
x	x	0	Q	$\bar{Q}$

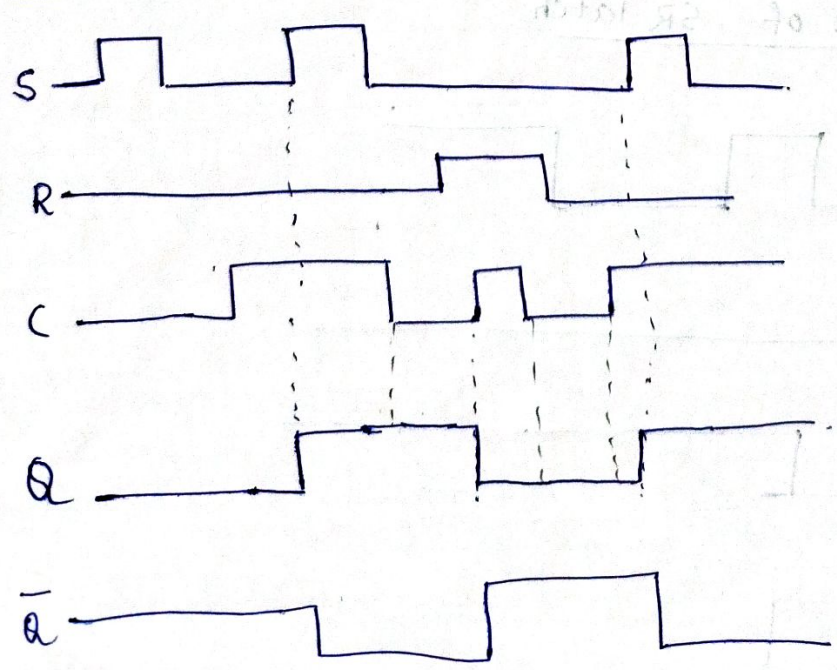
* → forbidden.

Note :- Explain the truth table.

Symbol

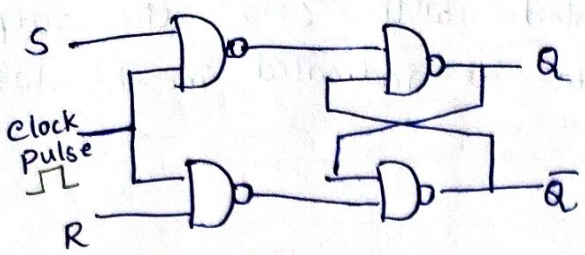


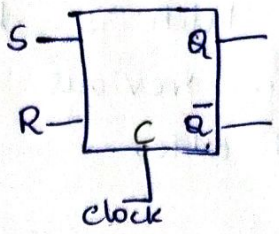
Timing diagram :-



SR flip-flop

If enable to a gated SR latch is clock pulse then the circuit is called SR flipflop.





Truth table

S	R	C	Q^+	$\bar{Q}^+$
0	0	⎍	Q	$\bar{Q}$
0	1	⎍	0	1
1	0	⎍	1	0
1	1	⎍	?	?

Explain truth table

✱ Gated D Latch :-

The very structure of gated D latch avoids the S=R=1 condition. Hence this is designed to eliminate the forbidden input problem.
It has single input D and two outputs Q and $\bar{Q}$.

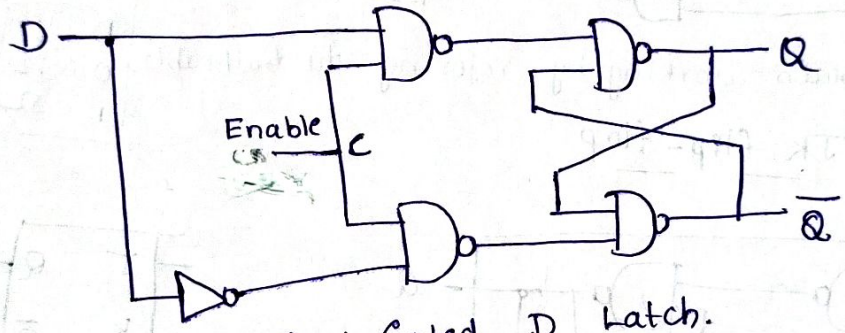
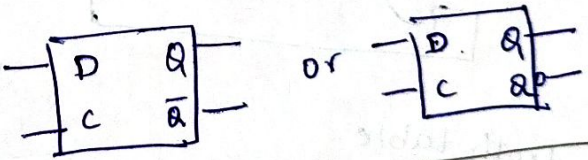


fig: Gated D Latch.

Truth table

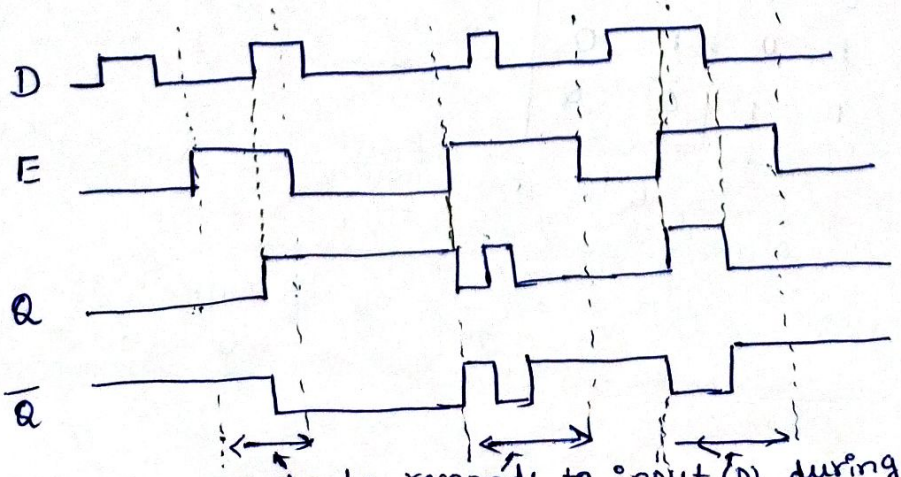
D	C	Q^+	$\bar{Q}^+$
0	1	0	1
1	1	1	0
X	0	Q	$\bar{Q}$

Symbol



Note:- Explain truthtable by taking D=0; Let Q=0 $\bar{Q}$=1 & so on.
Like SR latch.

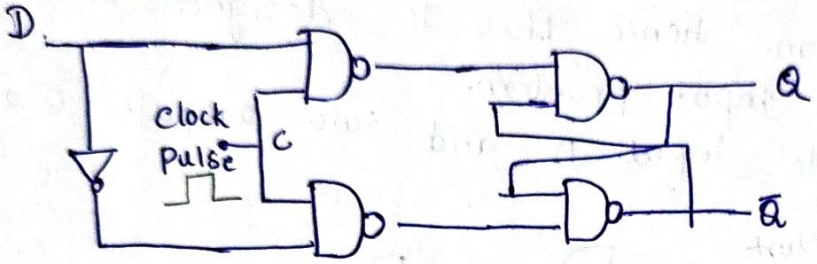
Timing diagram



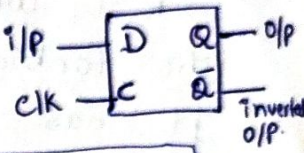
When the latch is enabled with C=1, the signal at the D input appears at the Q output. The output is said to follow the input when the latch is enabled. When the latch is disabled with C=0, the output retains its previous state as indicated in the last row of truth table.

* D flipflop :-

If enable to a gated D-latch is clock pulse, then the circuit is called D-flip-flop.



symbol

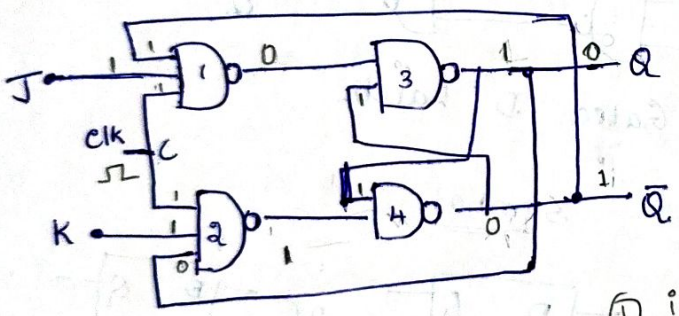


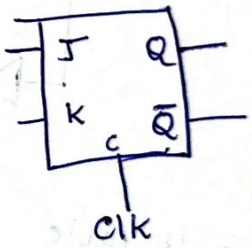
Truth table

D	C	Q^+	$\bar{Q}^+$
0	⎍	0	1
1	⎍	1	0

Note:- Explain working by refering to truthtable.

* Clocked JK flip-flop





Truth table

clk	J	K	Q^+	$\bar{Q}^+$
0	X	X	Q	$\bar{Q}$
⎍	0	0	Q	$\bar{Q}$
⎍	0	1	0	1
⎍	1	0	1	0
⎍	1	1	$\bar{Q}$	Q

① if J=0=K, Let Q=0, $\bar{Q}$=1
when clk is applied, output of NAND 1 is '1', NAND 2 is '1' output from NAND 3 is 0 & NAND 4 is 1. ∴ Q^+=0, $\bar{Q}^+$=1 same as present state (NO change)

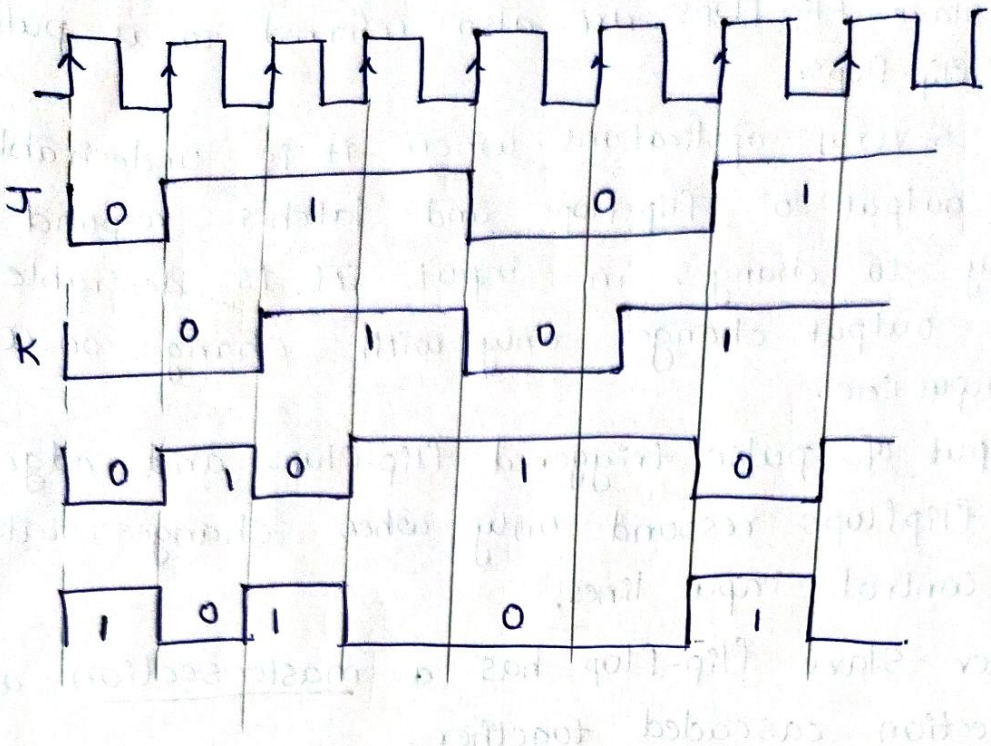
② When J=0, K=1, Q=0 $\bar{Q}$=1
NAND1 o/p-1, NAND2→1, NAND3:-
NAND 4 →1 ∴ Q^+=0, $\bar{Q}^+$=1

③ When J=1, K=0, Q=0 $\bar{Q}$=1
o/p
NAND 3 →1, NAND 4 →0 ∴ NAND 1→0, NAND 2 →1
Q^+=1 $\bar{Q}^+$=0

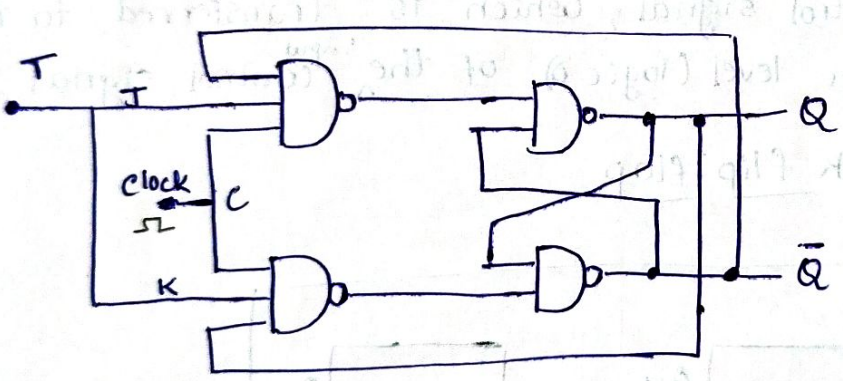
④ When J=1, K=1, Q=0, $\bar{Q}$=1 ; o/p of NAND 1→0
NAND 2 → 1, NAND 3 → 1, NAND 4 →0 ∴ Q^+=1 $\bar{Q}^+$=0

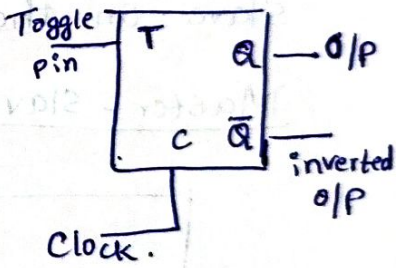
$Q^+ = \bar{Q}$ & $\bar{Q}^+ = Q$.

Timing diagram:-



* Clocked T-flipflop



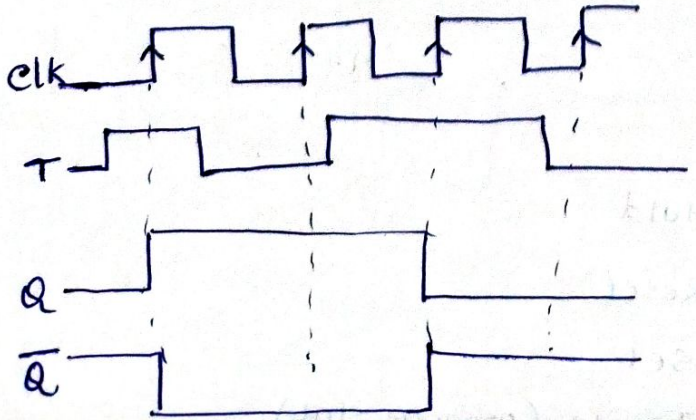


Truth table

clk	T	Q^+	$\bar{Q}^+$
0	X	Q	$\bar{Q}$
⎍	0	Q	$\bar{Q}$
⎍	1	$\bar{Q}$	Q

Note: Explain operation similar to clocked JK FF

Timing diagram



★ Master - Slave Flip-Flops:-

* Master - Slave Flip-flops are also referred to as pulse-triggered flip-flops.
* There are several applications where it is undesirable that the output of flip-flops and latches respond immediately to changes in input. It is desirable that the output changes only with changes on a control input line.
* The output of pulse triggered flip-flops and edge triggered flipflops respond only when changes take place on their control input lines.
* The Master - Slave flip-flop has a master section and a slave section cascaded together.
* The master registers the data on one level (say logic 1) of the input control signal, which is transferred to the slave on the other level (logic 0) of the input control signal.

Master - Slave JK flip-flop

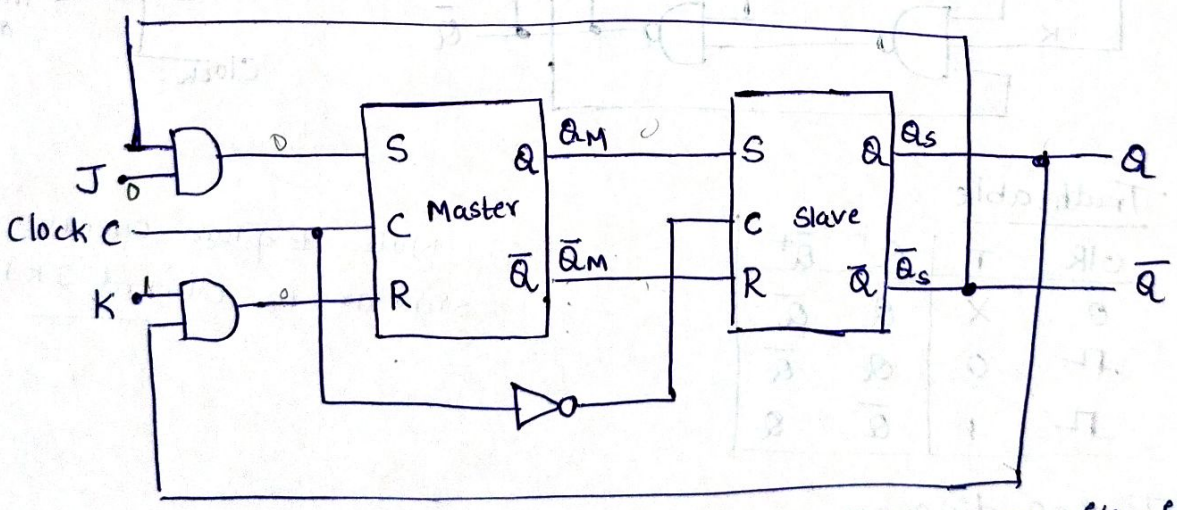


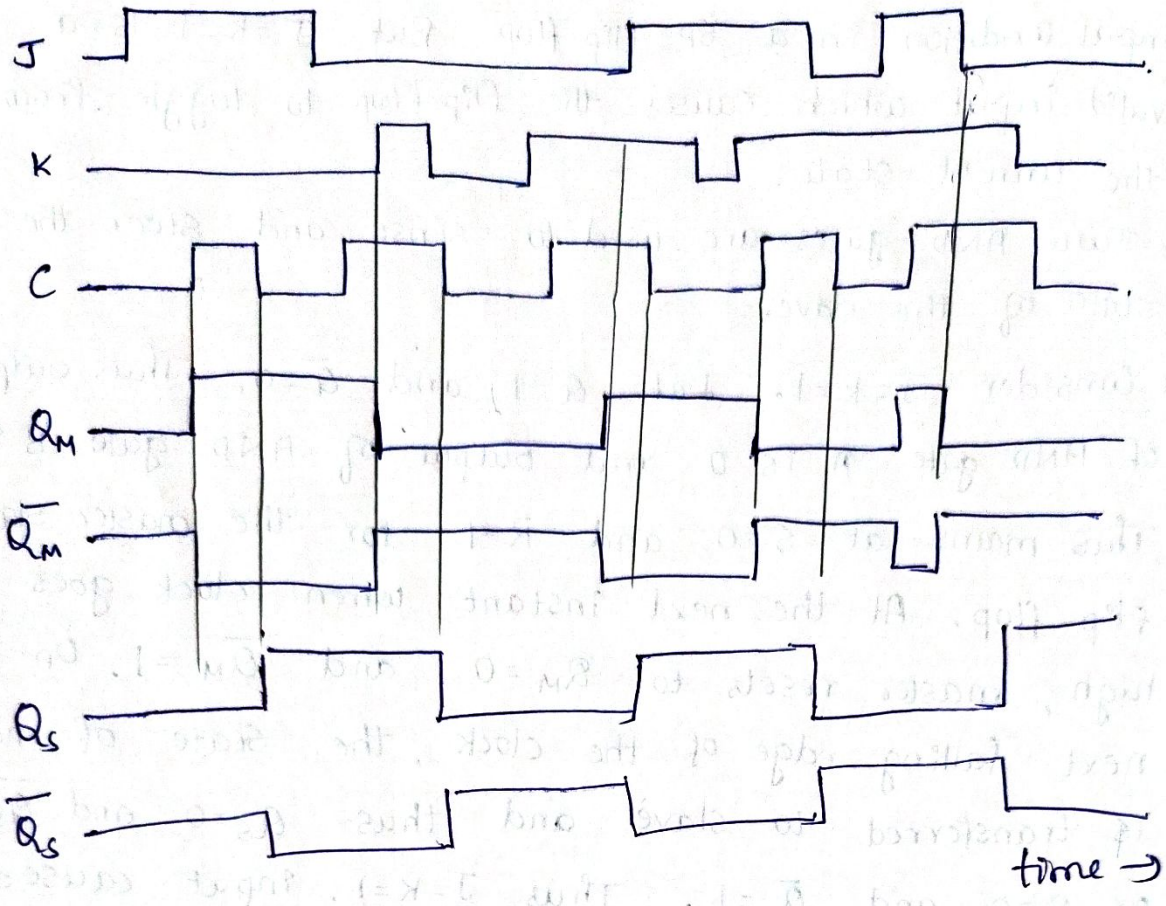
fig: Schematic of master-slave JK flip-flop.

Truthtable/ function table

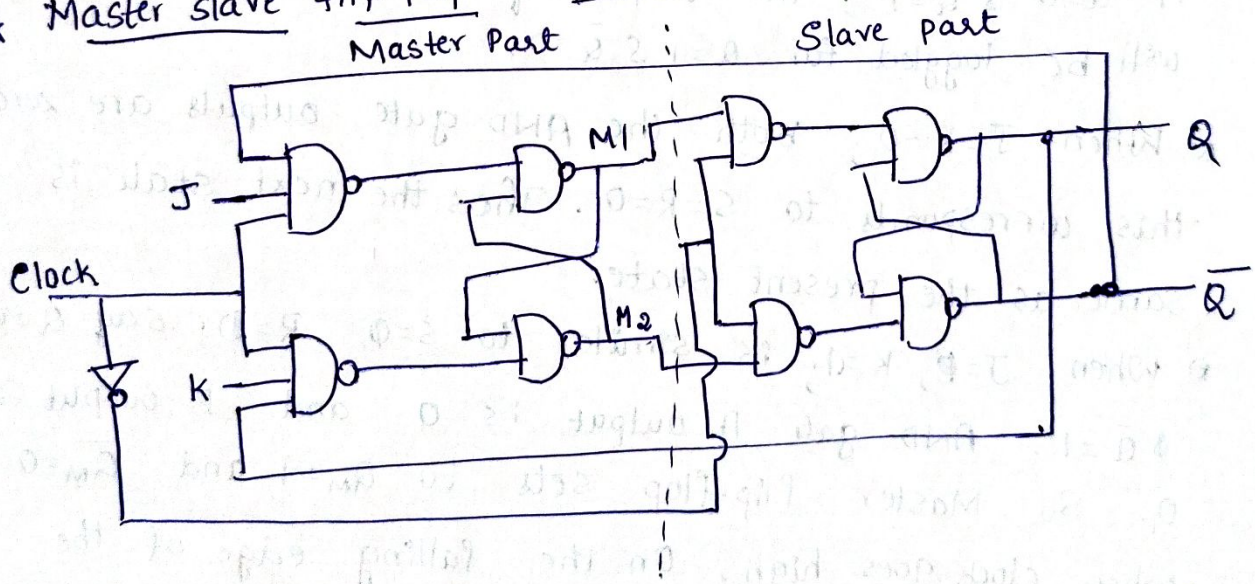
J	K	Clk	Q^+	$\bar{Q}^+$	
0	0	⊓	Q	$\bar{Q}$	Hold
0	1	⊓	0	1	Reset
1	0	⊓	1	0	set
1	1	⊓	$\bar{Q}$	Q	Toggle (opposite state)
X	X	0	Q	$\bar{Q}$	Hold (previous state)

* In this J corresponds to S and K corresponds to R inputs of SR flip-flip. S=R=1 is an undesirable input condition in a SR flip-flop, But J=K=1 is a valid input which causes the flip-flop to toggle from the current state.
* Two AND gates are used to sense and steer the state of the slave.
* Consider J=K=1. Let Q=1 and $\bar{Q}$=0. Thus output of AND gate A is 0 and output of AND gate B is 1. This means at S=0 and R=1 for the master slave flip-flop. At the next instant when clock goes high, master resets to Q_M=0 and $\bar{Q}_M$=1. On the next falling edge of the clock, the state of master is transferred to slave and thus Q_S=0 and $\bar{Q}_S$=1 or Q=0 and $\bar{Q}$=1. Thus J=K=1. input caused the output to toggle. Similarly at J=K=1 initially if Q=0 & $\bar{Q}$=1; the output of master-slave flipflop will be toggled for Q=1 & $\bar{Q}$=0.
* When J=K=0; both the AND gate outputs are zero. this corresponds to S=R=0. Thus the next state is same as the present state.
* When J=1, K=0; is similar to S=1, R=0; say Q=0 & $\bar{Q}$=1; AND gate A output is 1 and B output is 0. So Master flip-flop sets to Q_M=1 and $\bar{Q}_M$=0 when clock goes high. On the falling edge of the Clock, the slave also sets $\bar{Q}_S$=1 and $\bar{Q}_S$ = 0;
* When J=0; K=1; Let Q=0; $\bar{Q}$=1; output of AND gate A is 0, Output of AND gate B is 0. ∴ S=0; R=0 Q_M=0, $\bar{Q}_M$=1 (no change) ∴ Q_S=0 & $\bar{Q}_S$ = 1 after the application of clock.

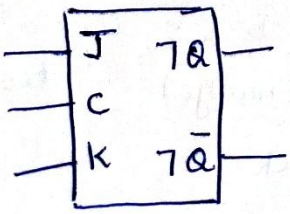
Timing diagram for Master-slave JK flipflop



* Master slave flip-flop using NAND gates :-



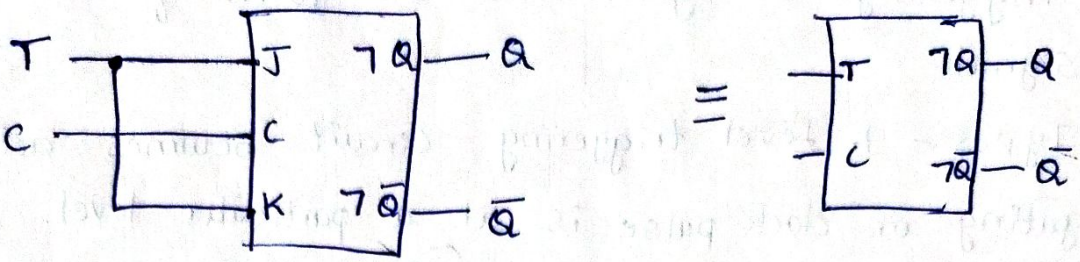
Logic symbol



Note:- Explain the operation → as in case of master slave JK flipflop
↑ refer

* Master slave T flipflop :-

* This can be configured from a master-slave JK flip-flop as shown in fig below. Not included

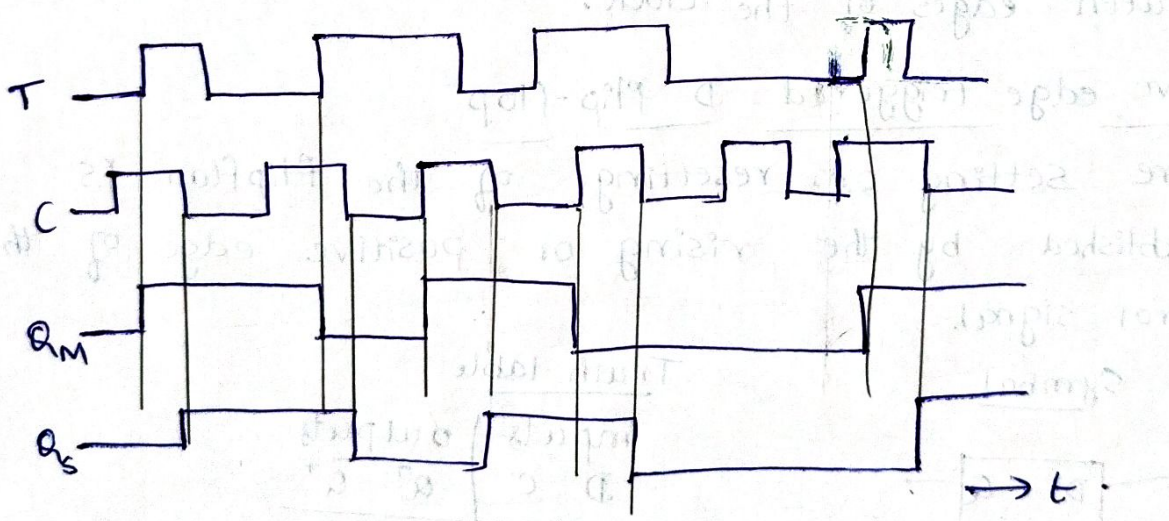


Truth table

T	C	Q^+	$\bar{Q}^+$
0	⊓	Q	$\bar{Q}$
1	⊓	$\bar{Q}$	Q
	0	Q	$\bar{Q}$

* When T = 0 (J = K = 0), the next state of the T flip-flop equals to its present state.
* When T = 1 (J = K = 1); next state is complement of the present state.

Timing diagram



* Concept of Edge Triggered Flip-Flops :-

* Triggering means making the circuit active or allowing it to receive inputs. Triggering makes circuit synchronous.
* Triggering is given in the form of clock or gate signal.
* Types - 1. Level triggering : Circuit becomes active when gating or clock pulse is at a particular level.

2. Edge triggering : Circuit becomes active at the negative edge or positive edge of the clock signal.

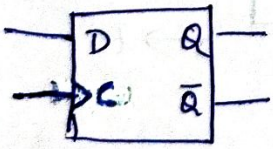
* Edge triggered flip-flops:

* Uses just one of the edges of the control to affect the reading of information on input lines. ie either at positive edge or negative edge of the clock.
* Such flip-flops ignore changes in the inputs between edges of the clock.

Positive edge triggered D flip-flop

Here setting or resetting of the flipflop is established by the rising or positive edge of the control signal.

Symbol



Truth table

inputs		outputs	
D	C	Q^+	$\bar{Q}^+$
0	↑	0	1
1	↑	1	0
X	0	Q	$\bar{Q}$
X	1	Q	$\bar{Q}$

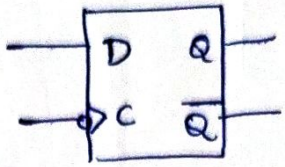
The ↑ in the C column of truth table indicates that D appears at the output at the positive edge of the clock.

 indicates C is positive edge sensitive input.

* Negative edge triggered D flip flop.

* Falling edge is used to sample the D input.
* Configured by adding an inverter to the control clock input of the positive edge triggered D flip-flop.

D	C	Q^+	$\bar{Q}^+$
0	↓	0	1
1	↓	1	0
x	0	Q	$\bar{Q}$
x	1	Q	$\bar{Q}$



↓ → indicates that D input during the 1 to 0 transition of the clock gets latched at the output. The changes in D between the negative edges of the clock are ignored.

* Characteristic Equations

* Characteristic equations are the algebraic description of the next state table of a flip-flop.
* It is obtained by constructing the K-map for Q^+ in terms of present state and information input variable.

1) For SR flip-flop

Function table:-

S	R	Q^+
0	0	Q
0	1	0
1	0	1
1	1	-

Next state table

Row No.	S	R	Q	Q^+
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	-
7	1	1	1	-

K-map for next state table

S \ RQ	00	01	11	10
0	0	1	0	0
1	1	1	-	-

$$\therefore \boxed{Q^+ = S + \bar{R}Q}$$

2) For JK flipflop :-

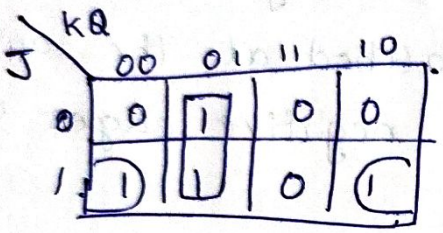
Function table

J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	$\bar{Q}$

Next state table

Row. No.	J	K	Q	Q^+
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

K-map for next state table



∴ characteristic equation :-

$$Q^+ = J\bar{Q} + \bar{K}Q$$

3) For D flip-flop

Function table

D	Q^+
0	0
1	1

Next state table

R. No.	D	Q	Q^+
0	0	0	0
1	0	1	0
2	1	0	1
3	1	1	1

K-map

D \ Q	0	1
0	0	0
1	1	1

∴ characteristic equation :- $Q^+ = D$

OR $Q^+ = D\bar{Q} + DQ$

$= D(Q + \bar{Q})$

$Q^+ = D$

4) For T-flipflop

Function table

T	Q^+
0	Q
1	$\bar{Q}$

Next state table

R.No	T	Q	Q^+
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	0

$$Q^+ = \bar{T}Q + T\bar{Q}$$

$$Q^+ = T \oplus Q$$

* Conversion of flip-flops :-

1. JK flip-flop to D-flip-flop :-

We have,

Next state table for D-flip-flop -

D	Q	Q^+
0	0	0
0	1	0
1	0	1
1	1	1

Next state table for JKFF :-

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

$\Rightarrow$

Application table / excitation table

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

∴ The characteristic table of D FF and excitation table of JKFF is

D	Q	Q^+	J	K
0	0	0	0	X
0	1	0	X	1
1	0	1	1	X
1	1	1	X	0

Using K map to find boolean expression for J & K in terms of D :-

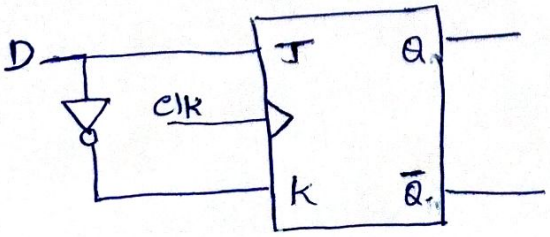
D \ Q	0	1
0	0	X
1	1	X

$J = D$

D \ Q	0	1
0	X	1
1	X	0

$K = \bar{D}$

∴ Conversion of JKFF to DFF :-



JK flip-flop to T flip-flop :-

Next state table for T FF

T	Q	Q^+
0	0	0
0	1	1
1	0	1
1	1	0

JKFF Next state table

J	K	Q	Q^+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

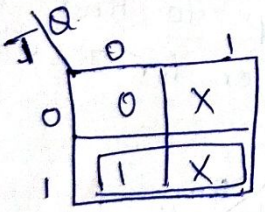
Application table

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

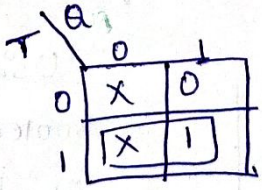
Step 1 : Construct characteristic table of T flip-flop and excitation table of JK FF.

T	Q	Q^+	J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

Step 2 :- Using the K-map find Boolean expression of J & K interms of T.



$J = T$



$K = T$

Step 3: Construct the circuit diagram of the conversion of JK flip-flop into T-flip-flop.