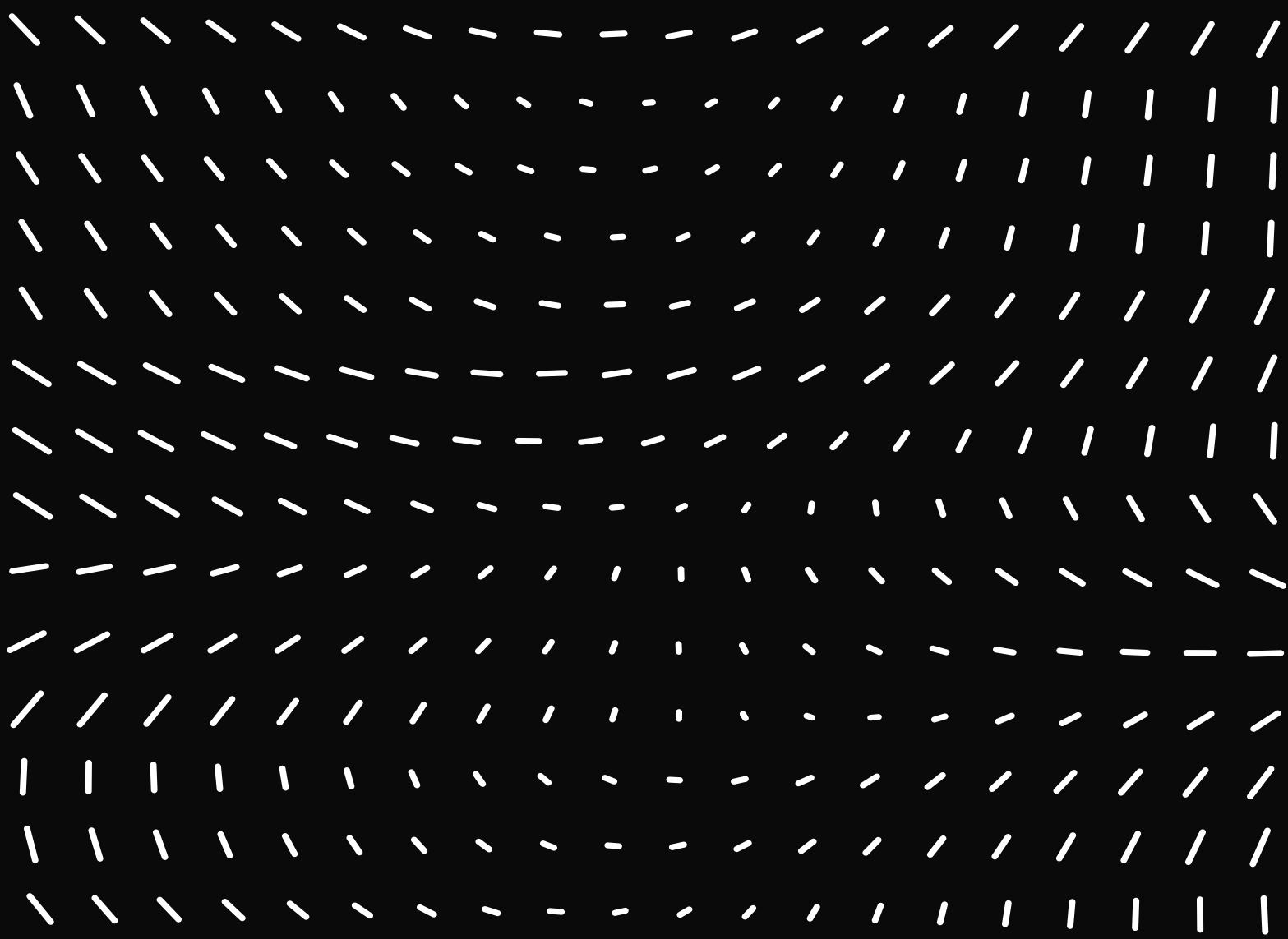
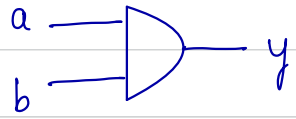
DIGITAL SYSTEM DESIGN

- MANVITH PRABHU



EC200

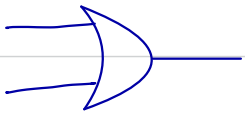
Basic Logic gates

1> **AND :** 

$y = a \cdot b$

Truth table :

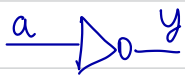
Input		Output
a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

2> **OR:** 

$y = a + b$

Truth table :

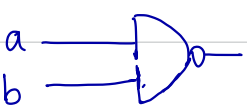
Input		Output
a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

3> **NOT:** 

$y = \bar{a}$

a	y
0	1
1	0

Derived gates

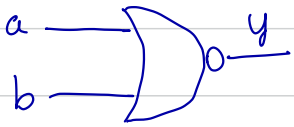
1> **NAND:** 

$y = \overline{a \cdot b}$

Truth table:

Input		Output
a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

2> NOR:

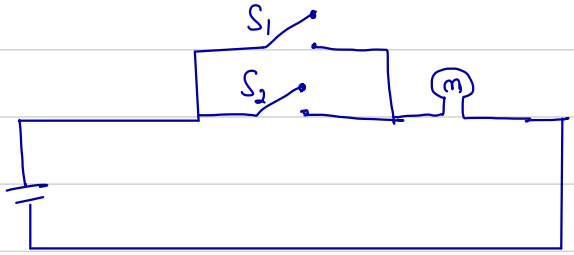


$y = \overline{a+b}$

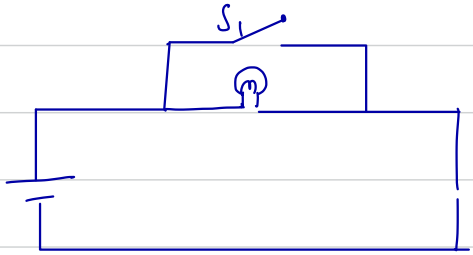
Truth table

Input		Output
a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

- Nand and Nor are universal gates.
- If number of gate counts are similar, we use NAND or NOR.

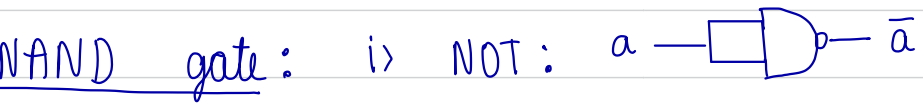


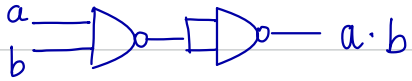
→ OR gate

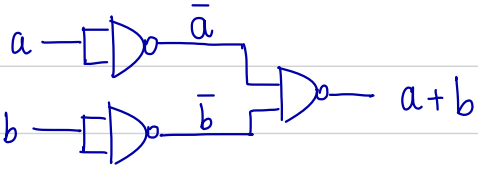


→ NOT gate

Designing basic gates using universal gates.

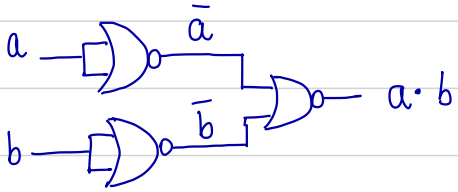
1) Using NAND gate: i) NOT: 

ii) AND: 

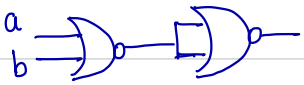
iii) OR: 

i.e $\overline{\bar{a}\cdot\bar{b}} = a+b$

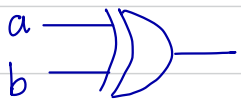
2) Using NOR gate: i) NOT : 

ii) AND : 

i.e $\overline{\bar{a}+\bar{b}} = a\cdot b$

iii) OR: 

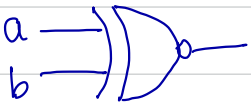
Exclusive gates

1) XOR gate: 

$a\oplus b = y$, $a\oplus b = \bar{a}b + a\bar{b}$

Truth table :

Input		Output
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

2) XNOR gate: 

$y = a\odot b$, $a\odot b = ab + \bar{a}\bar{b}$

Truth table:

Input		Output
a	b	y
0	0	1
0	1	0
1	0	0
1	1	1

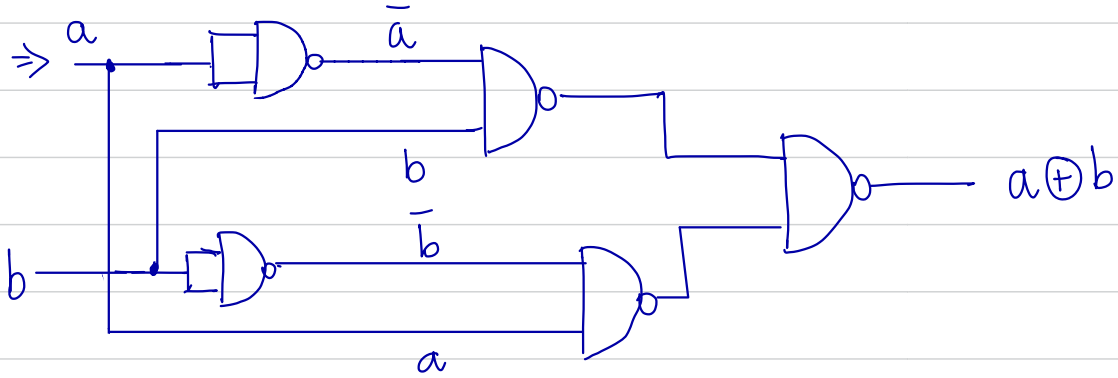
Implementation of XOR using NAND:

NOTE: De Morgan's law: 1> $\overline{a \cdot b} = \bar{a} + \bar{b}$

2> $\overline{a + b} = \bar{a} \cdot \bar{b}$

We have $y = a \oplus b = \bar{a}b + a\bar{b} = \overline{\overline{\bar{a}b + a\bar{b}}}$

$= \overline{(\bar{a}b) \cdot (a\bar{b})}$



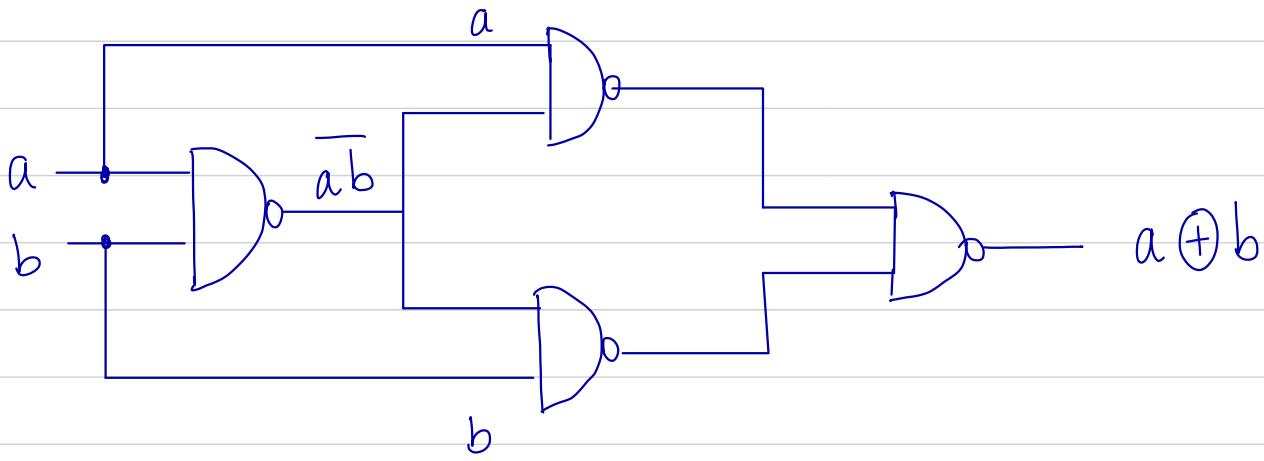
Also we have $y = a \oplus b = \overline{a \odot b} = \overline{ab + \bar{a}\bar{b}}$

$= \overline{ab} \cdot \overline{\bar{a}\bar{b}}$ $= \overline{ab} \cdot (a + b)$

$= \overline{ab} \cdot a + \overline{ab} \cdot b$

$= \overline{(\overline{ab \cdot a})} \cdot \overline{(\overline{ab \cdot b})}$ //

⇒



Conversion of number systems

1) $(17)_{10} = (?)_2$

2 | 17
2 | 8 - 1
2 | 4 - 0
2 | 2 - 0
1 - 0

$\Rightarrow (10001)_2$

2) $(0.17)_{10} = (?)_2$

0.17 X 2 = 0.34 → 0
0.34 X 2 = 0.68 → 0
0.68 X 2 = 1.36 → 1
0.36 X 2 = 0.72 → 0
0.72 X 2 = 1.44 → 1

$\Rightarrow (0.00101\ldots)_2$ //

Representation of negative numbers

Sign magnitude method : MSB = 1 ⇒ -ve number
MSB = 0 ⇒ +ve number

$(-7)_{10} = (1\ 111)_2$

$(6) = (0\ 110)_2$

Compliment system: Radix complement → 2's complement
Diminished radix complement → 1's complement.

Boolean laws:

1) $X+0=X$ $\quad X\cdot 1=X$

$X+1=1$ $\quad X\cdot 0=0$

2) Idempotent laws: $X+X=X$ $\quad X\cdot X=X$

3) Involution law: $(\overline{\overline{X}}) = X$

4) Laws of complimentarity: $X+\overline{X}=1$ $\quad X\cdot\overline{X}=0$

5) Commutative law: $X+Y=Y+X$ $\quad XY=YX$

6) Distributive laws: $X(Y+Z)=XY+XZ$ $\quad X+YZ=(X+Y)(X+Z)$

7) Simplification theorems: $XY+X\overline{Y}=X$ $\quad (X+Y)(X+\overline{Y})=X$

$X+XY=X$ $\quad X(X+Y)=X$

$(X+\overline{Y})Y=XY$ $\quad X\overline{Y}+Y=X+Y$

8) De Morgan's laws: $\overline{(X+Y+Z+\dots)} = \overline{X}\,\overline{Y}\,\overline{Z}\dots$ $\quad \overline{(XYZ\dots)} = \overline{X}+\overline{Y}+\overline{Z}+\dots$

9) Duality: $(X+Y+Z+\dots)^D = XYZ\dots$ $\quad (XYZ\dots)^D = X+Y+Z+\dots$

10) Consensus theorem:

i) $XY + YZ + X'Z = XY + X'Z$

ii) $(X+Y)(Y+Z)(X'+Z) = (X+Y)(X'+Z)$

i) Proof: $XY + X'Z + (X+X')YZ$

$= XY + X'Z + XYZ + X'YZ$

$= XY(1+Z) + X'Z(1+Y)$

$= XY + X'Z$ //

ii) Theorem for multiplying out and factoring out

$(X+Y)(X'+Z) = XZ + X'Y$

$XY + X'Z = (X+Z)(X'+Y)$

Laws and Theorems of Boolean Algebra

Operations with 0 and 1:

1. $X + 0 = X$
2. $X + 1 = 1$

1D. $X \cdot 1 = X$

2D. $X \cdot 0 = 0$

Idempotent laws:

3. $X + X = X$

3D. $X \cdot X = X$

Involution law:

4. $(X')' = X$

Laws of complementarity:

5. $X + X' = 1$

5D. $X \cdot X' = 0$

Commutative laws:

6. $X + Y = Y + X$

6D. $XY = YX$

Associative laws:

7. $(X + Y) + Z = X + (Y + Z) = X + Y + Z$

7D. $(XY)Z = X(YZ) = XYZ$

Distributive laws:

8. $X(Y + Z) = XY + XZ$

8D. $X + YZ = (X + Y)(X + Z)$

Simplification theorems:

9. $XY + XY' = X$

9D. $(X + Y)(X + Y') = X$

10. $X + XY = X$

10D. $X(X + Y) = X$

11. $(X + Y')Y = XY$

11D. $XY' + Y = X + Y$

DeMorgan's laws:

12. $(X + Y + Z + \ldots)' = X'Y'Z' \ldots$

12D. $(XYZ \ldots)' = X' + Y' + Z' + \ldots$

Duality:

13. $(X + Y + Z + \ldots)^D = XYZ \ldots$

13D. $(XYZ \ldots)^D = X + Y + Z + \ldots$

Theorem for multiplying out and factoring:

14. $(X + Y)(X' + Z) = XZ + X'Y$

14D. $XY + X'Z = (X + Z)(X' + Y)$

Consensus theorem:

15. $XY + YZ + X'Z = XY + X'Z$

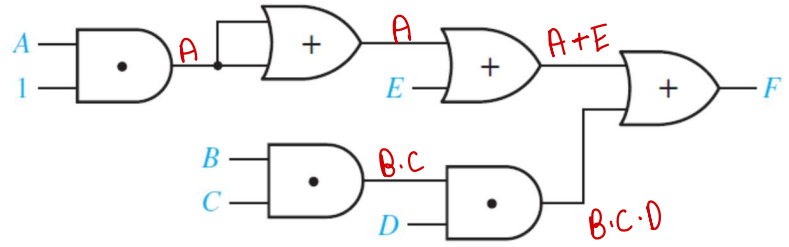
15D. $(X + Y)(Y + Z)(X' + Z) = (X + Y)\,(X' + Z)$

EC200 - DIGITAL SYSTEM DESIGN

Tutorial 1

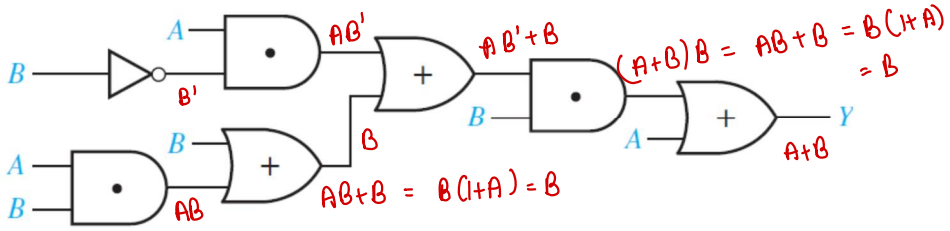
1. For each of the following circuits, find the output and design a simpler circuit having the same output. (*Hint:* Find the circuit output by first finding the output of each gate, going from left to right, and simplifying as you go.)

(a)

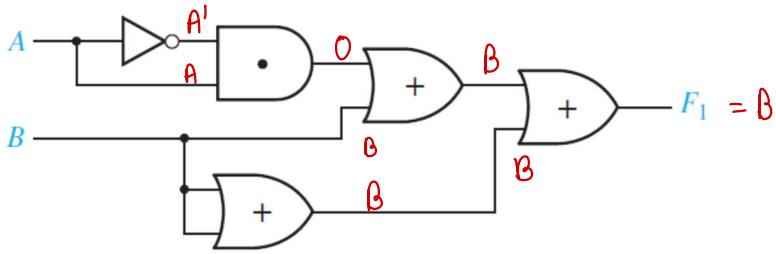


$F = A+E+(B\cdot C\cdot D)$

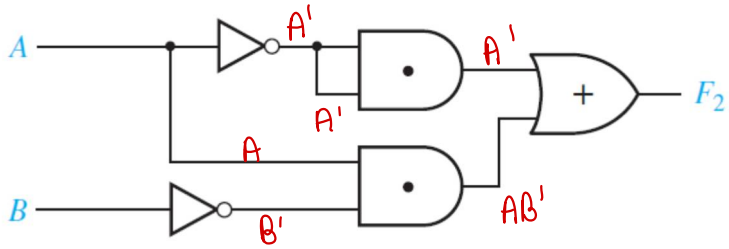
(b)



(c)

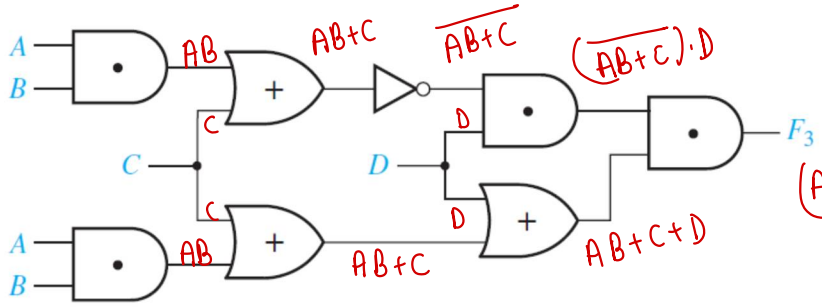


(d)



$A'+AB'$
$= (A'+A)\cdot(A'+B')$
$= A'+B'$ //

(e)



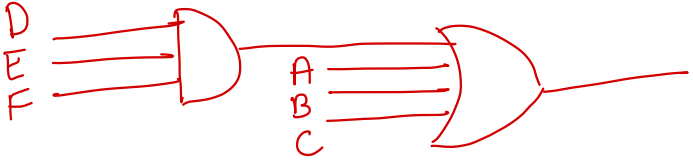
$(AB+C+D)\cdot(\overline{AB+C})\cdot D$
let $AB+C = X$
$(X+D)\cdot\bar{X}\cdot D$
$\bar{X}D\cdot D = \bar{X}D$
$(\overline{AB+C})\cdot D$
$\bar{A}\bar{B}\cdot\bar{C}\cdot\bar{D}$

2. Factor each of the following expressions to obtain a product-of-sums:
   (a) $AB + C'D'$
   (b) $WX + WY'X + ZYX$
   (c) $A'BC + EF + DEF'$ $= (A'+E)(B+E)(C+E)(A'+F+D)(B+F+D)(C+F+D)$
   (d) $XYZ + W'Z + XQ'Z$ $= Z(W'+X)(W'+Q'+Y)$
   (e) $ACD' + C'D' + A'C$ $= (A'+D')(C+D')$
   (f) $A + BC + DE$

2a) $(AB+C')\cdot(AB+D') = (C'+A)(C'+B)\cdot(D'+A)\cdot(D'+B)$

b) $WX(1+Y') + ZYX = WX+ZYX = X(W+ZY) = X\cdot(W+Y)\cdot(W+Z)$

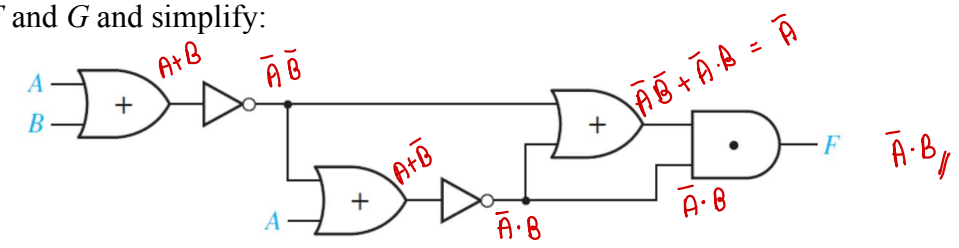
f) $(A+B+D)(A+B+E)(A+C+D)(A+C+E)$

3a> 

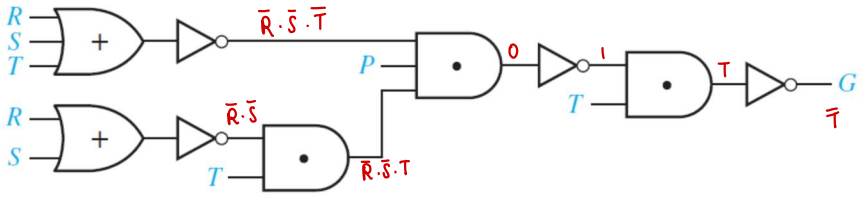
3. Draw a circuit that uses only one AND gate and one OR gate to realize each of the following functions:
   (a) $(A + B + C + D)(A + B + C + E)(A + B + C + F)$ = $(A+B+C)\cdot DEF$
   (b) $WXYZ + VXYZ + UXYZ$ = $XYZ\cdot(W+V+U)$

4. Simplify the following expressions to a minimum sum of products
   (a) $[(AB)' + C'D]'$ = $(AB' + c'D')' = ((AB)')'\cdot(C'D')' = AB\cdot(C+D') = ABC + ABD'$
   (b) $[A + B(C' + D)]'$
   (c) $((A + B')C)'(A + B)(C + A)'$ = $\bar{A}B\bar{C}$

5. Find F and G and simplify:



(a)

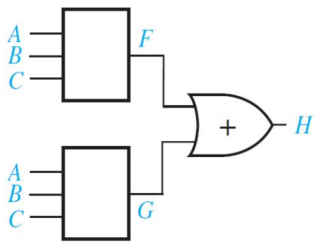


(b)

6. For each of the following Boolean (or switching) algebra expressions, indicate which, if any, of the following terms describe the expression: product term, sum-of products, sum term, and product-of-sums. (More than one may apply.)
   (a) $X'Y$ = Product , SOP
   (b) $XY' + YZ$ = SOP
   (c) $(X' + Y)(WX + Z)$ None
   (d) $X + Z$ = Sum , POS , SOP
   (e) $(X' + Y)(W + Z)(X + Y' + Z')$ = POS

7. In the following circuit, $F = (A' + B)C$. Give a truth table for G so that H is as specified in its truth table. If G can be either 0 or 1 for some input combination, leave its value unspecified (x).

H = F+G

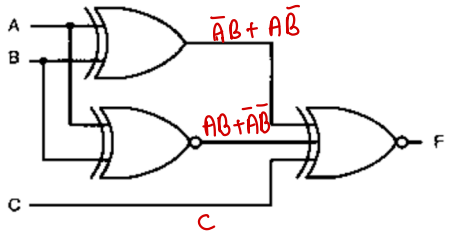


A	B	C	H	F	G
0	0	0	0	0	0
0	0	1	1	1	X
0	1	0	1	0	1
0	1	1	1	1	X
1	0	0	0	0	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	X

8. Implement the Boolean function $Y = AB + CD$ using minimum number of 2-input NAND gates. → 3 NAND gates.

4b> $\bar{A}\cdot\overline{B(C'+D)} = \bar{A}\cdot(\bar{B} + C\cdot\bar{D}) = \bar{A}\bar{B} + \bar{A}C\bar{D}$

9. Find out all possible input combinations of A, B and C that give output $F = 1$ for the logic circuit shown below.

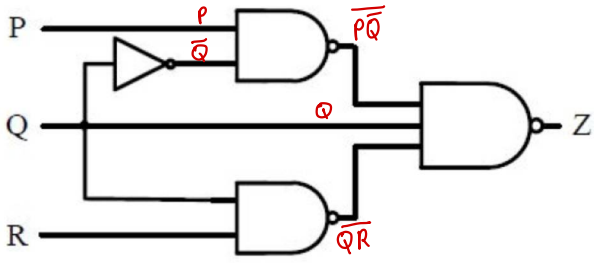


A	B	C	A⊕B	A⊙B	F
0	0	0	0	1	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	1	0	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	1	0	1	1

10. Eliminate the redundant terms and minimize the following expressions
(a) $XY + X'Z + YZ$ $= XY + X'Z$
(b) $A'B' + AC + BC' + B'C + AB$ $= A'B' + AC + AB + BC' = A'B' + AC + BC'$
(c) $(X + Y)(X' + Z)(Y + Z)$ $= (X+Y)(X'+Z)$
(d) $(a + b + c')(a + b + d')(b + c + d')$ $= (a+b+c')(a+b+d')$
(e) $A'C'D + A'BD + BCD + ABC + ACD'$
$= A'C'D + BCD + ABC + ACD' = (A'C'D + BCD + ACD')$

11. Digital input signals A, B, C with A as the MSB and C as the LSB are used to realize the Boolean function $F = m_0 + m_2 + m_3 + m_5 + m_7$, where m_i denotes the i^{th} minterm. In addition, F has a don't care for m_1. Find a simplified expression for F.

12. Find the minimal expression of the output Z of the logic circuit shown below



$\overline{(P\bar{Q}) \cdot Q \cdot \bar{Q}R}$
$= \overline{P\bar{Q}} + \bar{Q} + QR$
$= P\bar{Q} + \bar{Q} + QR$
$= \bar{Q}(1+P) + QR = \bar{Q} + QR$
$= (\bar{Q} + Q) \cdot (\bar{Q} + R) = (\bar{Q} + R)$

13. The output of a majority circuit is 1 if a majority (more than half) of its inputs are equal to 1, and the output is 0 otherwise. Construct a truth table for a three-input majority circuit and derive a simplified sum-of-products expression for its output.

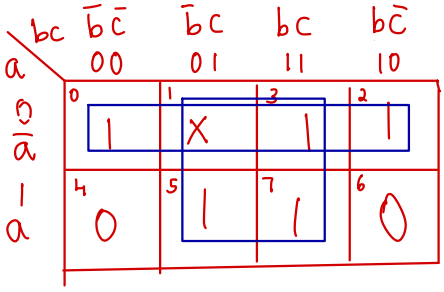
14. Obtain the truth table of the following functions, and express each function in sum-of-minterms and product-of-maxterms form:
(a) $(b + c'd)(a' + cd')$ $= a'b + bcd' + a'c'd$
(b) $(ad + b'c + bd')(b + d)$
(c) $(b + d)(b + d')(a + c)$
(d) $ad + bcd + ab'c' + b'c'd'$

15. Determine whether the following Boolean equation is true or false.
(a) $y'z' + yz' + x'z = x' + xz'$ True
(b) $x'y' + xz' + yz = y'z' + xy + x'z'$ False, when $x=0, y=1, z=1$

16. How many switching functions of two variables (x and y) are there? Give each function in truth table form and in reduced algebraic form. 2^4

17. Simplify the following Boolean functions to sum-of-products form using Karnaugh maps
(a) $F(x, y, z) = \sum(0, 2, 6, 7)$
(b) $F(x, y, z) = \sum(0, 2, 3, 4, 6)$

(a)



$y = \bar{a} + c$

(c) $F(x,y,z) = \sum(3,5,6,7)$
(d) $F(x,y,z) = \sum(0,1,3,4,5)$
(e) $F(x,y,z) = \sum(1,3,5,7)$
(f) $F(x,y,z) = xy + x'y'z' + x'yz'$
(g) $F(A,B,C,D) = A'B'C'D' + AC'D' + B'CD' + A'BCD + BC'D$
(h) $F(A,B,C,D) = C'D + A'B'C + ABC' + AB'C$
(i) $F(w,x,y,z) = x'z + w'xy' + w(x'y + xy')$

18. Find all the prime implicants for the following Boolean functions, and determine which are essential
(a) $F(w,x,y,z) = \sum(0,2,4,5,6,7,8,10,13,15)$
(b) $F(A,B,C,D) = \sum(0,2,3,5,7,8,10,11,14,15)$
(c) $F(w,x,y,z) = \sum(1,3,6,7,8,9,12,13,14,15)$
(d) $F(A,B,C,D) = \sum(0,2,3,5,7,8,10,11,13,15)$

19. Simplify the following expressions to (i) sum-of-products (ii) product-of-sums
(a) $F = x'z' + y'z' + yz' + xy$
(b) $F = ACD' + C'D + AB' + ABCD$
(c) $F = (A + C' + D')(A' + B' + D')(A' + B + D')(A' + B + C')$
(d) $F = ABC + AB'D + BCD$

20. Simplify the following Boolean functions F, together with don't-care conditions d, and then express the simplified function in sum-of-minterms form
(a) $F(x,y,z) = \sum(0,1,3,5,7);\ d(x,y,z) = \sum(2,4,6)$
(b) $F(A,B,C,D) = \sum(0,4,8,10,14);\ d(x,y,z) = \sum(2,6,12)$
(c) $F(A,B,C,D) = \sum(5,6,7,11,14,15);\ d(x,y,z) = \sum(3,9,13)$
(d) $F(A,B,C,D) = \sum(2,4,7,10,12,14);\ d(x,y,z) = \sum(0,3,6,8,13)$

21. Simplify the following functions using K-map, and implement them with NAND gate circuits:
(a) $F(A,B,C,D) = A'B'C + AC' + ACD + ACD' + A'B'C'$
(b) $F(A,B,C,D) = A'B'C'D + CD' + AC'D$
(c) $F(A,B,C,D) = (A' + C' + D')(A' + C')(C' + D')$
(d) $F(A,B,C,D) = A' + AB + B'C + ACD$

22. Implement the following Boolean function F, together with the don't-care conditions d, using no more than two NOR gates:
$F(A,B,C,D) = \sum(2,4,10,12,14);\ d(A,B,C,D) = \sum(0,1,5,8)$

SOP - NAND gates
POS - NOR gates

23. Draw a logic diagram using only two-input ~~NAND~~ NOR gates to implement the following function:

$$F(A,B,C,D) = (A \oplus B)'(C \oplus D)$$

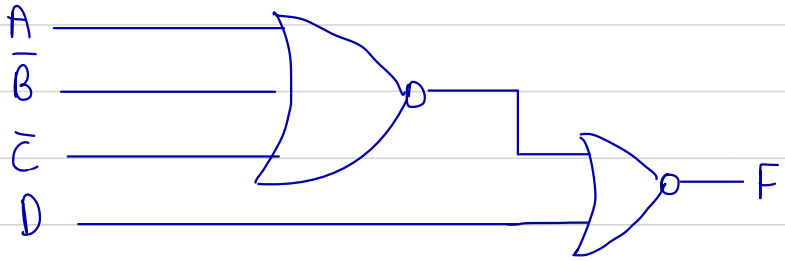
24. Obtain a minimal sum-of-products expression for the output F of the logic circuit shown below

15a) LHS = $z'(y+y') + x'z$
= $z' + x'z$
= $(z'+z)(z'+x')$
= $x'+z'$ //
RHS = $(x'+x)(x'+z')$
= $x'+z'$
∴ LHS = RHS //

22)

AB \ CD	C+D (00)	C+D̄ (01)	C̄+D̄ (11)	C̄+D (10)
00 A+B	0: X	1: X	3: 0	2: 1
01 A+B̄	4: 1	5: X	7: 0	6: 0
11 Ā+B̄	12: 1	13: 0	15: 0	14: 1
10 Ā+B	8: X	9: 0	11: 0	10: 1

$\therefore F = \bar{D} \cdot (A + \bar{B} + \bar{C})$



23) $F = (A \oplus B)(C \oplus D)$

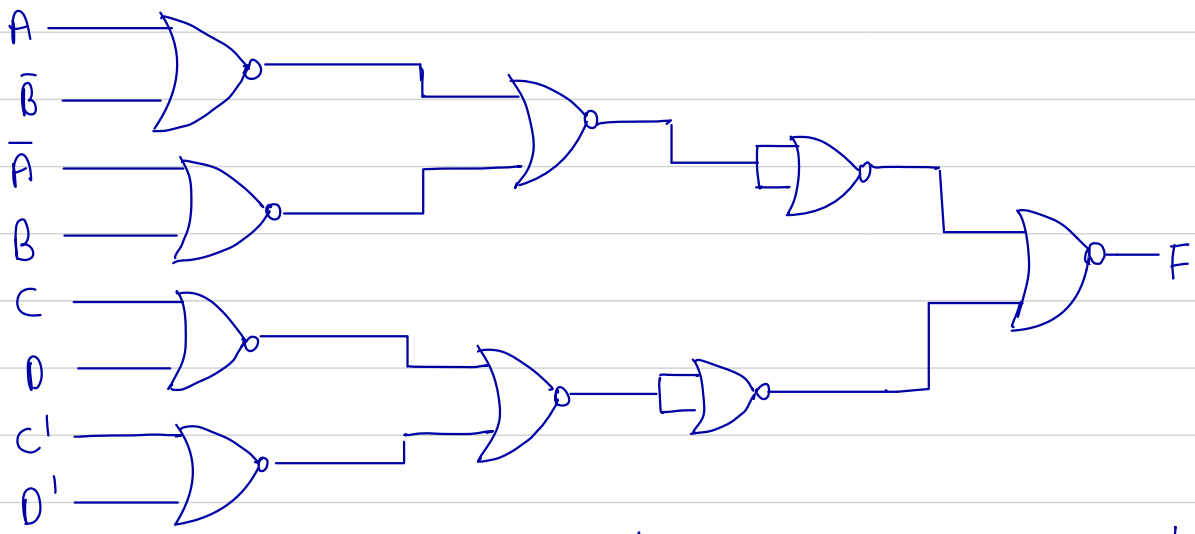
$= (A \odot B)(C \oplus D)$

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

$= AB\bar{C}D + ABC\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D}$

AB \ CD	C+D (00)	C+D̄ (01)	C̄+D̄ (11)	C̄+D (10)
00 A+B	0: 0	1: 1	3: 0	2: 1
01 A+B̄	4: 0	5: 0	7: 0	6: 0
11 Ā+B̄	12: 0	13: 1	15: 0	14: 1
10 Ā+B	8: 0	9: 0	11: 0	10: 0

$F = (A + \bar{B}) \cdot (\bar{A} + B) \cdot (C + D) \cdot (\bar{C} + \bar{D})$

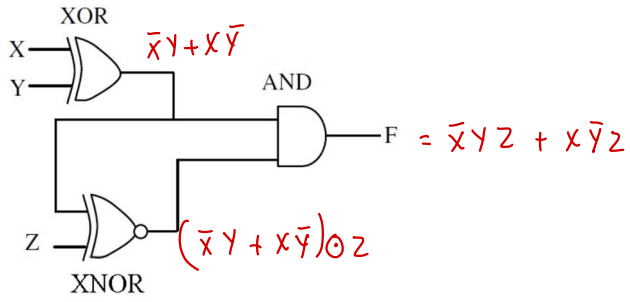


30>

Flow rate	I/P A	B	C	D	E	O/P Y	Z
< 10 gal/min	0	0	0	0	0	1	0
10 - 20 gal/min	1	0	0	0	0	1	0
20 - 30 gal/min	1	1	0	0	0	1	1
30 - 40 gal/min	1	1	1	0	0	0	1
40 - 50 gal/min	1	1	1	1	0	0	1
≥ 50 gal/min	1	1	1	1	1	0	0

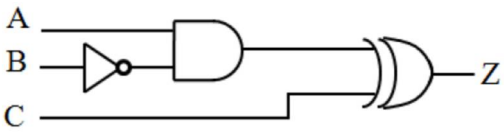
$Y = \bar{A}B\bar{C}\bar{D}\bar{E} + A\bar{B}\bar{C}\bar{D}\bar{E} + AB\bar{C}\bar{D}\bar{E}$

$Z = AB\bar{C}\bar{D}\bar{E} + ABC\bar{D}\bar{E} + ABCD\bar{E}$



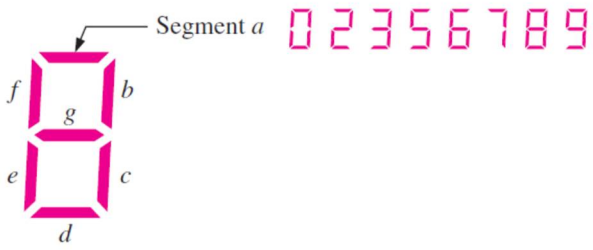
25. Convert the Boolean expression $F(A,B,C) = X'YZ' + ZY'Z' + XYZ' + XYZ$ to canonical product-of-sums form.

26. All the logic gates shown in the figure have a propagation delay of 20 ns. Let $A = C = 0$ and $B = 1$ until time $t = 0$. At $t = 0$, all the inputs flip (i.e., $A = C = 1$ and $B = 0$) and remain in that state. Find the duration (in ns) for which the output $Z = 1$, for $t > 0$.



27. A digital system is required to amplify a binary-encoded audio signal. The user should be able to control the gain of the amplifier from a minimum to a maximum in 100 increments. Find the minimum number of bits required to encode the control signals.

28. In a 7-segment display, each of the seven segments is activated for various digits. For example, segment *a* is activated for the digits 0, 2, 3, 5, 6, 7, 8, and 9, as illustrated in Figure below. Since each digit can be represented by a BCD code, derive an SOP expression for segment *a* using the variables *ABCD* and then minimize the expression using a Karnaugh map.

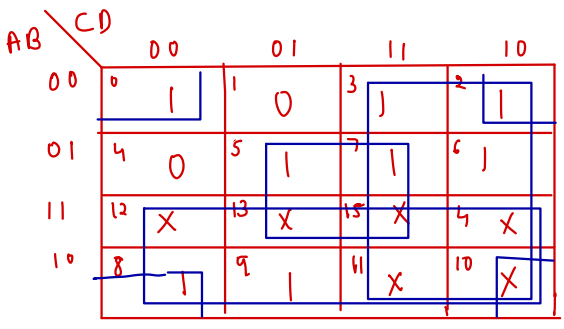


29. Two products are sold from a vending machine, which has two push buttons P_1 and P_2. When a button is pressed, the price of the corresponding product is displayed in a 7 - segment display.

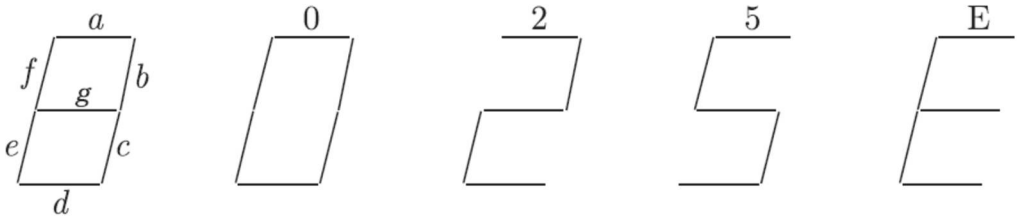
If no buttons are pressed, '0' is displayed signifying 'Rs 0'.
If only P_1 is pressed, '2' is displayed, signifying 'Rs. 2'.
If only P_2 is pressed '5' is displayed, signifying 'Rs. 5'.
If both P_1 and P_2 are pressed, '*E*' is displayed, signifying 'Error'.

The names of the segments in the 7 - segment display, and the glow of the display for '0', '2', '5' and 'E' are shown below.

28>



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



Push buttons pressed/not pressed in equivalent to logic 1/0 respectively. A segment glowing/not glowing in the display is equivalent to logic 1/0 respectively.

(a) Write the Boolean functions corresponding to each segment (*a* to *g*) in terms of the inputs P_1 and P_2 .
(b) What are the minimum number of NOT gates and 2-input OR gates required to design the logic of the driver for this 7-segment display?

30. A flow rate sensing device used on a liquid transport pipeline functions as follows. The device provides a 5-bit output where all five bits are zero if the flow rate is less than 10 gallons per minute. The first bit is 1 if the flow rate is at least 10 gallons per minute; the first and second bits are 1 if the flow rate is at least 20 gallons per minute; the first, second, and third bits are 1 if the flow rate is at least 30 gallons per minute; and so on. The five bits, represented by the logical variables *A*, *B*, *C*, *D*, and *E*, are used as inputs to a device that provides two outputs *Y* and *Z*.
(a) Write an equation for the output *Y* if we want *Y* to be 1 iff the flow rate is less than 30 gallons per minute.
(b) Write an equation for the output *Z* if we want *Z* to be 1 iff the flow rate is at least 20 gallons per minute but less than 50 gallons per minute.

31. Each of three coins has two sides, heads and tails. Represent the heads or tails status of each coin by a logical variable (*A* for the first coin, *B* for the second coin, and *C* for the third) where the logical variable is 1 for heads and 0 for tails. Write a logic function $F(A, B, C)$ which is 1 iff exactly one of the coins is heads after a toss of the coins. Express *F*
(a) as a minterm expansion.
(b) as a maxterm expansion.

32. A priority encoder circuit has four inputs, x_3, x_2, x_1 and x_0. The circuit has three outputs: z, y_1, y_0. If one of the inputs is 1, z is 1 and $y_1$and y_1 represent a 2-bit, binary number whose value equals the index of the highest numbered input that is 1. For example, if x_2 is 1 and x_3 is 0, then the outputs are $z = 1$, and $y_1 = 1$ and $y_0 = 0$. If all inputs are 0, $z = 0$ and y_1 and y_0 are don't-cares.
(a) List in decimal form the minterms and don't-care minterms of each output.
(b) List in decimal form the maxterms and don't-care maxterms of each output.

33. A bank vault has three locks with a different key for each lock. Each key is owned by a different person. To open the door, at least two people must insert their keys into the assigned locks. The signal lines *A*, *B*, and *C* are 1 if there is a key inserted into lock 1, 2, or 3, respectively. Write an equation for the variable *Z* which is 1 iff the door should open.

$Z = AB + BC + CA$

32

	I/P				O/P	
x_3	x_2	x_1	x_0	z	y_1	y_0
0	0	0	0	0	X	X
0	0	0	1	1	0	0
0	0	1	0	1	0	1
0	0	1	1	1	0	1
0	1	0	0	1	1	0
0	1	0	1	1	1	0
0	1	1	0	1	1	0
0	1	1	1	1	1	0
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

$$z = \Sigma m(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$$

$$y_1 = \Sigma m(4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$$
$$d = 0$$

$$y_0 = \Sigma m(2, 3, 8, 9, 10, 11, 12, 13, 14, 15) \qquad d = 0$$

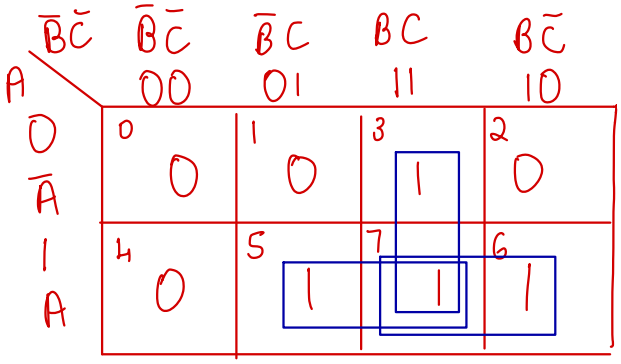
34. Design a logic circuit to produce a HIGH output only if the input, represented by a 4-bit binary number, is greater than twelve or less than three. First develop the truth table and then draw the logic diagram.

35. Develop the logic circuit necessary to meet the following requirements:
A battery-powered lamp in a room is to be operated from two switches, one at the back door and one at the front door. The lamp is to be on if the front switch is on and the back switch is off, or if the front switch is off and the back switch is on. The lamp is to be off if both switches are off or if both switches are on. Let a HIGH output represent the on condition and a LOW output represent the off condition.

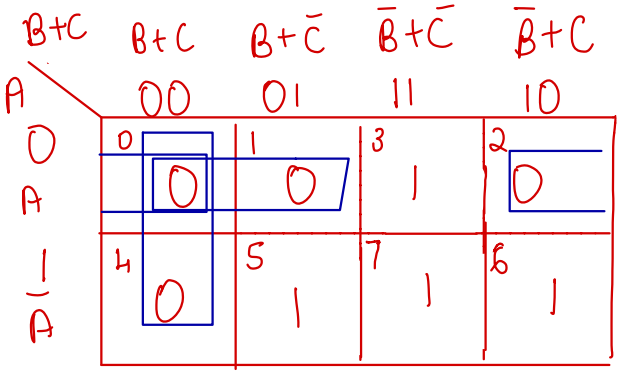
XOR

(3>

A	B	C	Y
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



$AB + BC + AC$ //



$(A+C)\cdot(B+C)(A+B)$

1's complement : change 0 to 1 and 1 to 0.

$0110 \longleftrightarrow 1001$

2's complement : Add 1 to LSB of 1's complement.

$$\begin{array}{r} 0110 \longleftrightarrow 1001 \\ +\quad\quad 1 \\ \hline 1010 \end{array}$$ //

$(+5)_{10} \longrightarrow 0101 \xrightarrow{\text{1's compliment}} 1010 \xrightarrow{2\text{'s}} 1011$ // (-5)

Range of numbers:

unsigned : 0 to $2^n - 1$

signed : -2^{n-1} to $2^{n-1} - 1$ (2's compliment)

$-2^{n-1} + 1$ to $2^{n-1} - 1$ (1's compliment)

Because in 1's compliment (0000 & 1111 represent 0)

Addition & Subtraction of binary numbers

$5 + 2 \Rightarrow$

$$\begin{array}{r} 0101 \\ 0011 \\ \hline 0111 \end{array}$$ //

$5 - 2 \Rightarrow 5 + (-2) \Rightarrow$

$-2 \Rightarrow$

$$\begin{array}{r} 0010 \\ 1101 \\ \hline 1110 \end{array} \quad \begin{array}{r} \overset{1}{} \\ 0101 \\ 1110 \\ \hline 10011 \end{array} = 3$$ //

Number after MSB is ignored.

$-5+2 \Rightarrow$ 1011 + 0010 = 1101

1101 − 1 = 1100 → 0011 // $\Rightarrow (-3)$ //

$-5-2 \Rightarrow$ 1011 + 1110 = 11001 $\Rightarrow$ 1001 // (-7) //

$+4+2 \Rightarrow$ 0100 + 0010 = 0110 $\Rightarrow$ 0110 //

1's compliment addition

$4+(-2) \Rightarrow$ 0100 + 1101 = 10001

0001 + 1 = 0010 //

- In 1's compliment method carry is added.
- In 2's compliment method carry is ignored.

Literals: All input variables and their compliments are called literals.

Sum terms: Sum of literals

Ex: x+z, z+y etc.

Product term: Product of literals.
Ex: $x\bar{y}$, xyz etc.

Sum of product: Ex: $x\bar{y} + yz + \bar{z}x$

Product of sum: Ex: $(x+\bar{y})(\bar{x}+y+z)$

Normal terms: In these kind of terms every literal occurs only once.

Minterms and maxterms

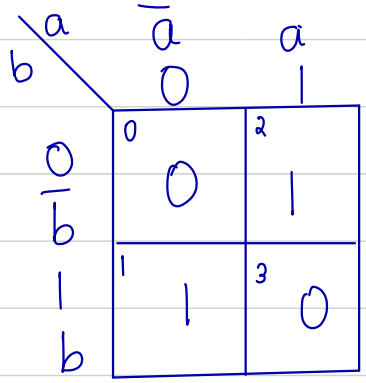
x	y	z	minterms:		Maxterms	
0	0	0	m_0	$\bar{x}\bar{y}\bar{z}$	M_0	$x+y+z$
0	0	1	m_1	$\bar{x}\bar{y}z$	M_1	$x+y+\bar{z}$
0	1	0	m_2	$\bar{x}y\bar{z}$	M_2	$x+\bar{y}+z$
0	1	1	m_3	$\bar{x}yz$	M_3	$x+\bar{y}+\bar{z}$
1	0	0	m_4	$x\bar{y}\bar{z}$	M_4	$\bar{x}+y+z$
1	0	1	m_5	$x\bar{y}z$	M_5	$\bar{x}+y+\bar{z}$
1	1	0	m_6	$xy\bar{z}$	M_6	$\bar{x}+\bar{y}+z$
1	1	1	m_7	xyz	M_7	$\bar{x}+\bar{y}+\bar{z}$

Canonical form: Output written in terms of minterms (SOP) or Maxterms (POS) is called canonical form

K-Map (Karnaugh Map): It is used to get simplified expression

1)

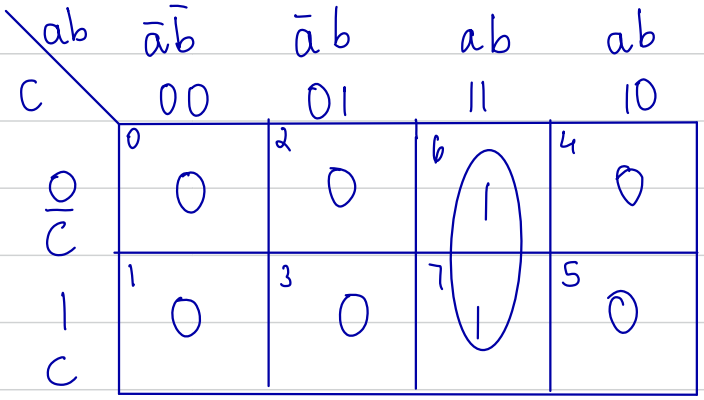
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0



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

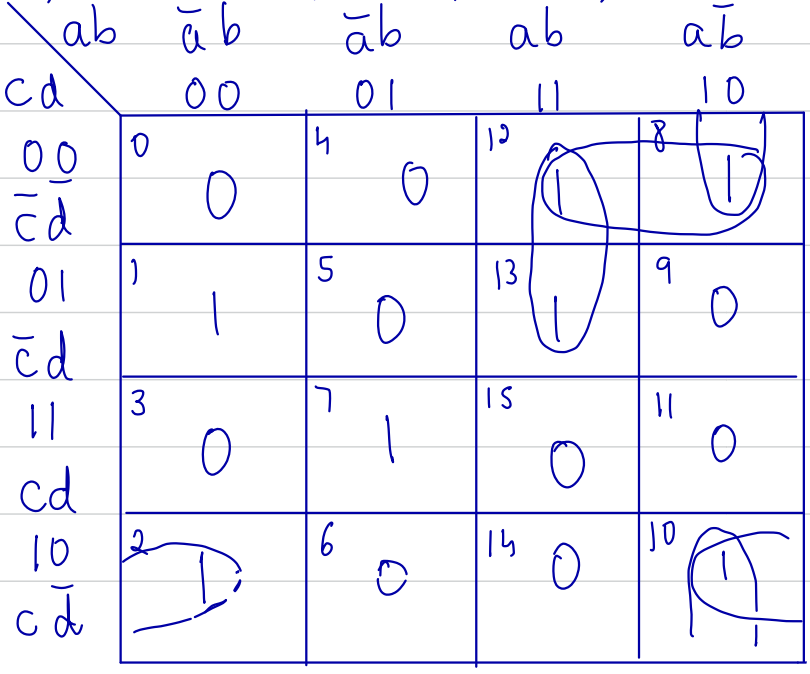
2)

a	b	c	y
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	1
1	1	1	1



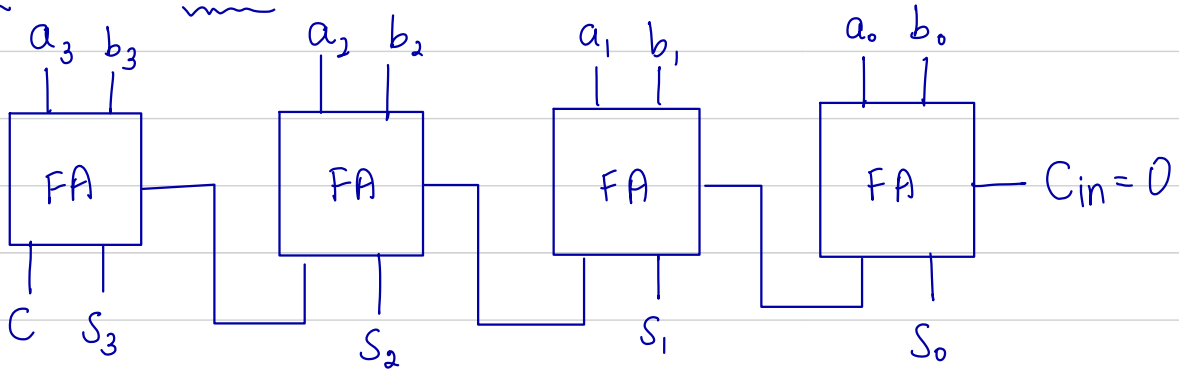
$y = ab$

3) $\Sigma(m_1, m_2, m_7, m_8, m_{10}, m_{12}, m_{13})$



$= ab\bar{c} + a\bar{c}\bar{d} + \bar{b}c\bar{d} + \bar{a}bcd + \bar{a}\bar{b}\bar{c}d$

Parallel adders



Ripple carry adders

a	b	C_{in}	C_{out}	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	P
1	0	0	0	
1	0	1	1	
1	1	0	1	G
1	1	1	1	

$G_i = ab$

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

$\therefore C_i = G_i + P_i C_{i-1}$

$C_0 = G_0 + P_0 C_{-1}$

$C_1 = G_1 + P_1 C_0 = G_1 + P_1 (G_0 + P_0 C_{-1})$

$\therefore C_1 = G_1 + P_1 G_0 + P_1 P_0 C_{-1}$

(by) $C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{-1}$

Binary codes

→ Used for representing discrete elements of information in binary.

Ex: Decimal numbers

→ Binary coded decimals (BCD)

- A set of n elements can be coded with $(\log_2 n)$ distinct codes

i.e n bit binary code can distinctly represent one element in a set of 2^n elements

BCD: $0 \rightarrow 9$ i.e 10 elements which needs minimum 4 bits.

	BCD	Binary
$(12)_{10}$	0001 0010	1100
$(734)_{10}$	0111 0011 0100	1011011110

BCD addition: We have to correct result if the result is out of range.

$8 + 4 = 12$

```
  1000          1100
+ 0100        + 0110
------        ------
  1100        10010010
```

Correction factor (add 6) If $S > 9$ or Cout = 1

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, −2, −1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
	1010	0101	0000	0001
Unused	1011	0110	0001	0010
bit	1100	0111	0010	0011
combi-	1101	1000	1101	1100
nations	1110	1001	1110	1101
	1111	1010	1111	1110

Decimal	Gray Code	Binary
0	0000	0000
1	0001	0001
2	0011	0010
3	0010	0011
4	0110	0100
5	0111	0101
6	0101	0110
7	0100	0111
8	1100	1000
9	1101	1001
10	1111	1010
11	1110	1011
12	1010	1100
13	1011	1101
14	1001	1110
15	1000	1111

Binary to Gray

B_2	B_1	B_0	G_2	G_1	G_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

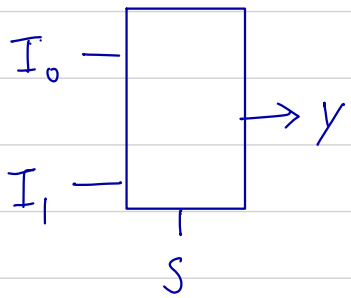
$G_2 = B_2$ $\quad G_1 = B_2 \oplus B_1$ $\quad G_0 = B_1 \oplus B_0$

Gray to Binary: $B_2 = G_2$ $\quad B_1 = G_1 \oplus G_2$

$B_2 = G_1 \oplus G_2 \oplus G_3$

Multiplexer: A digital switch

2-1 Multiplexer (2:1 MUX)

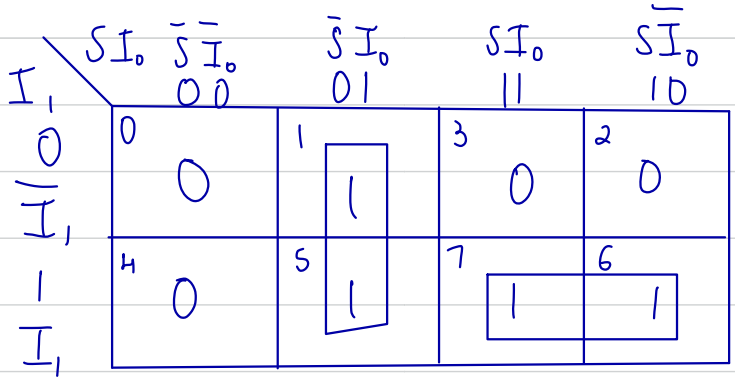


Function table:

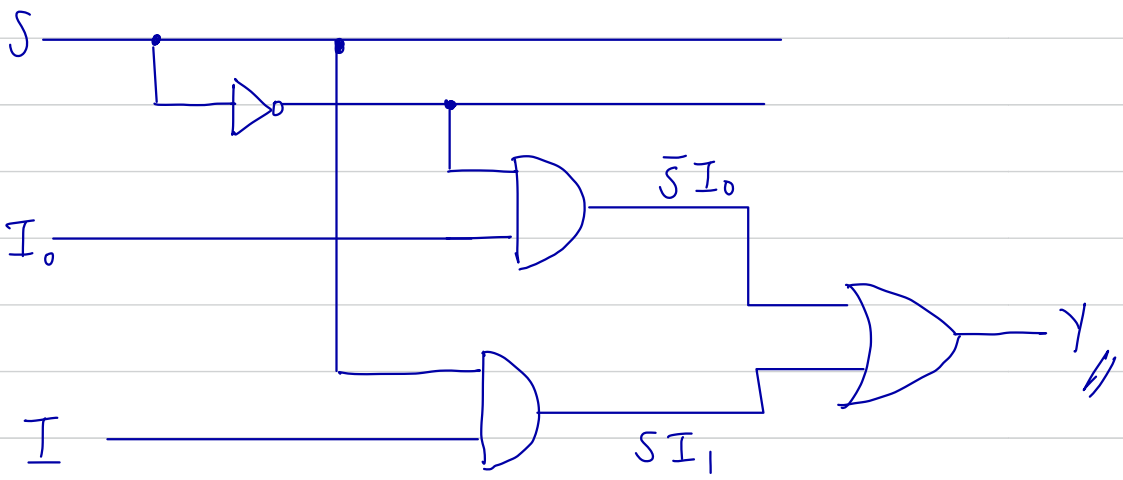
S	Y
0	I_0
1	I_1

Truth table

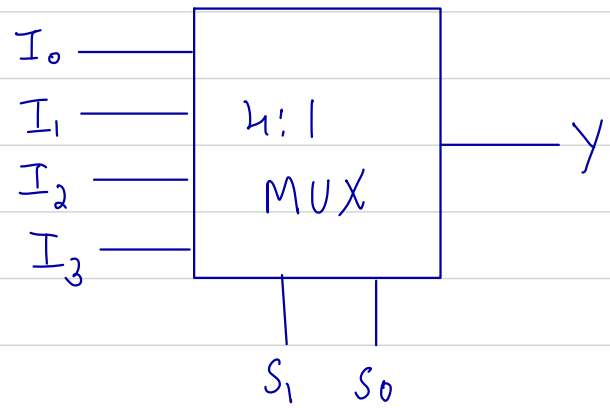
S	I_0	I_1	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



$\therefore Y = \bar{S}I_0 + SI_1$



4:1 MUX

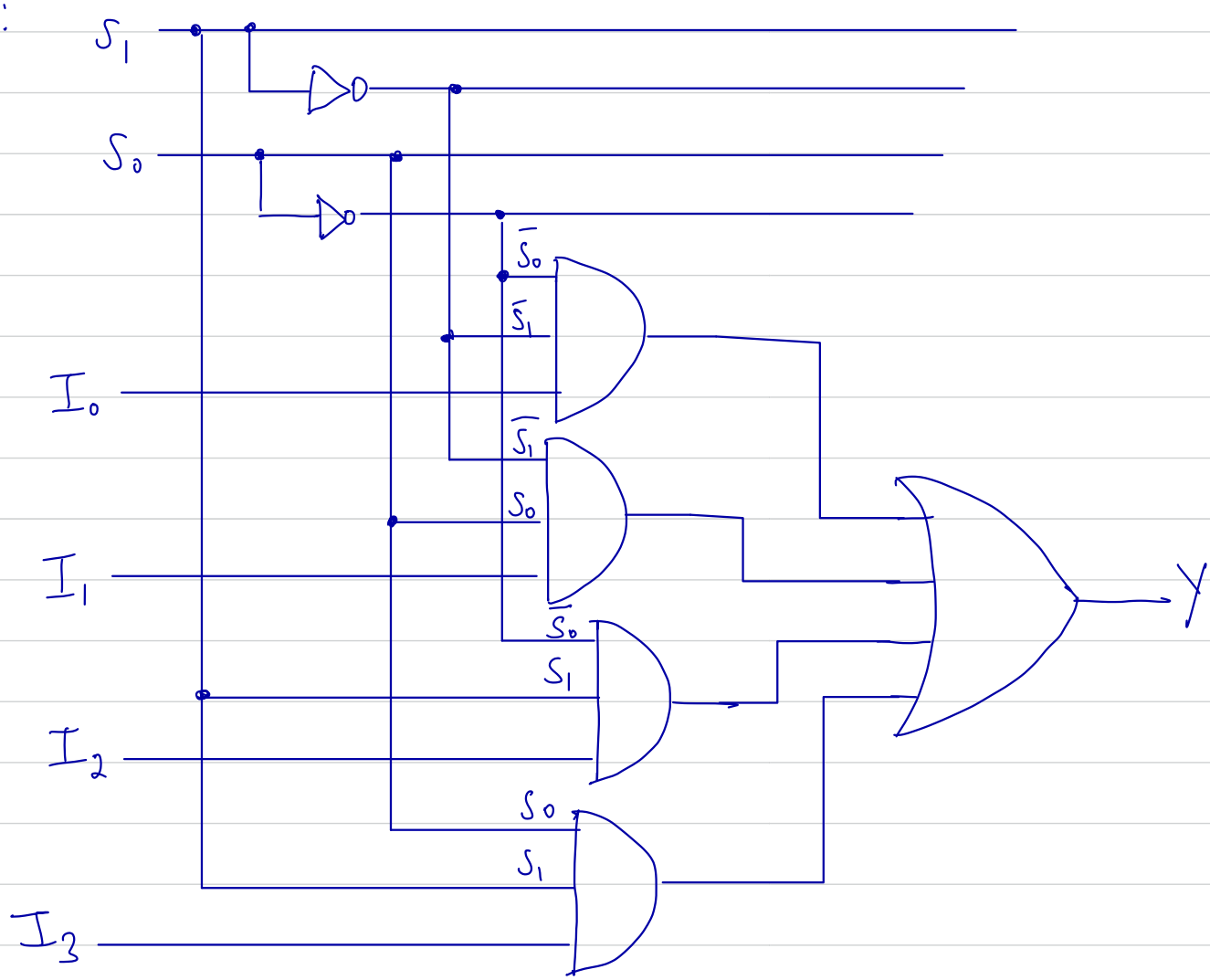


NOTE: 2^n:1 MUX has n switch

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

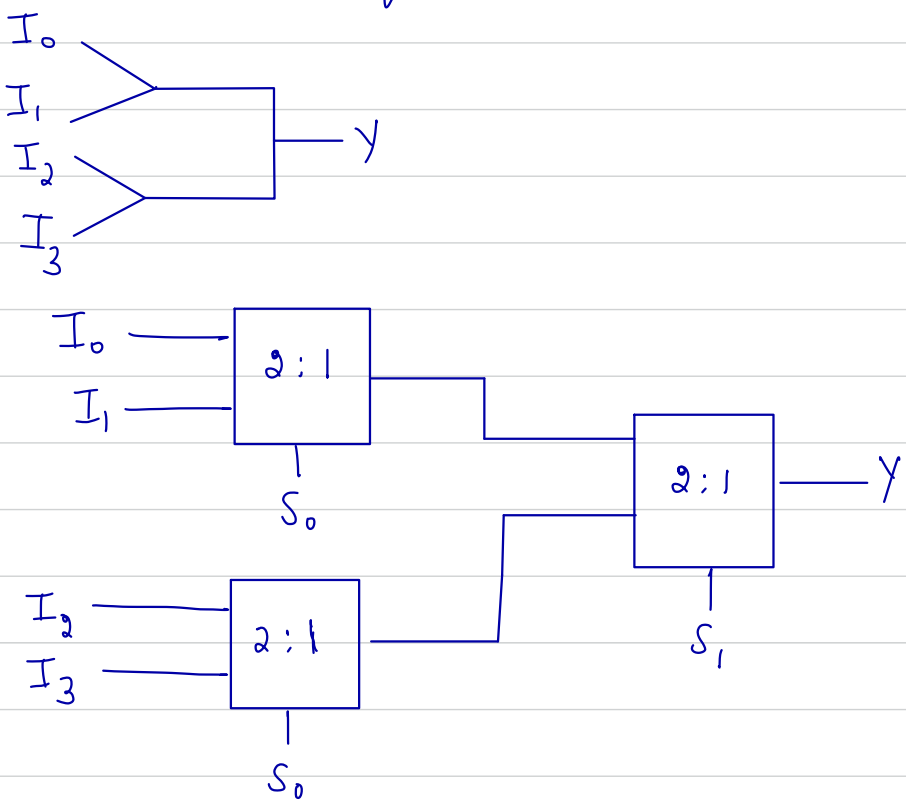
$$Y = \bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_1 + S_1 \bar{S}_0 I_2 + S_1 S_2 I_2 //$$

Circuit:

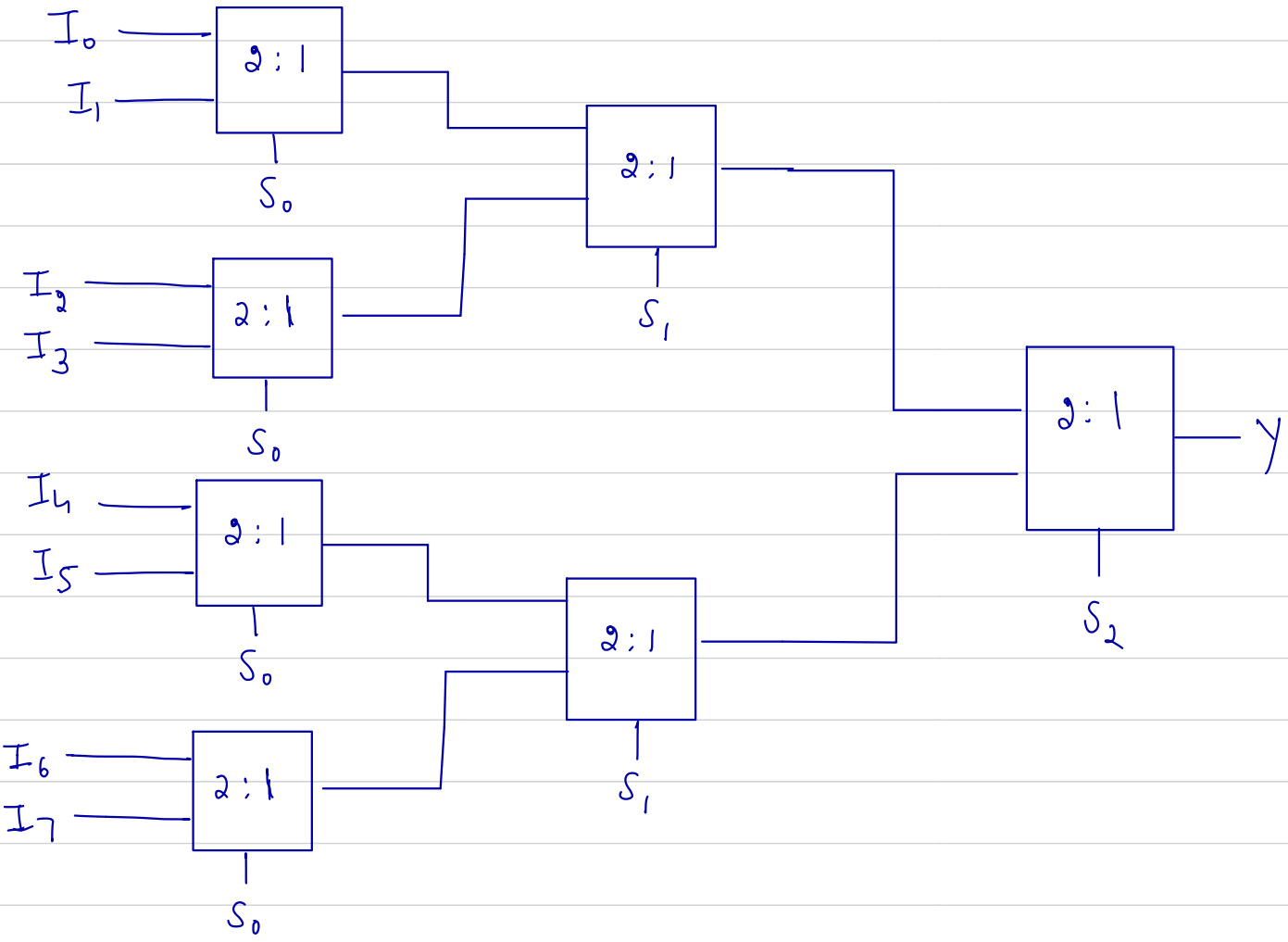


Multiplexer tree

4:1 MUX using 2:1 MUX



8:1 MUX using 2:1 MUX

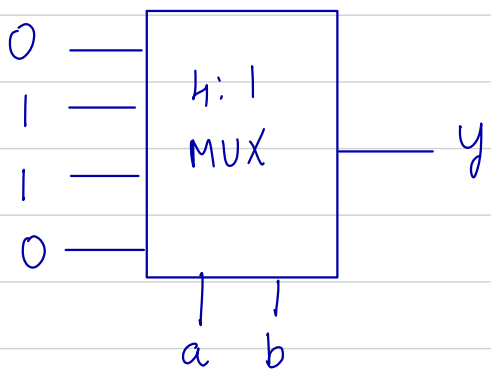


Logic design with MUX

1> Implement XOR function using MUX

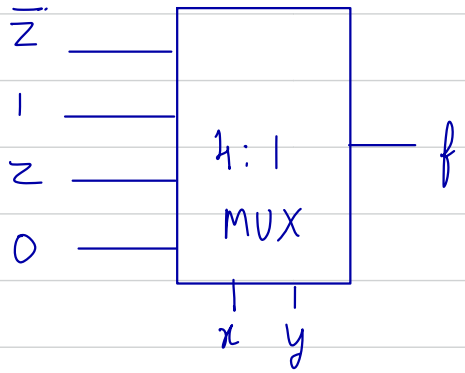
Note that n variable function has $2^n : 1$ MUX

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0



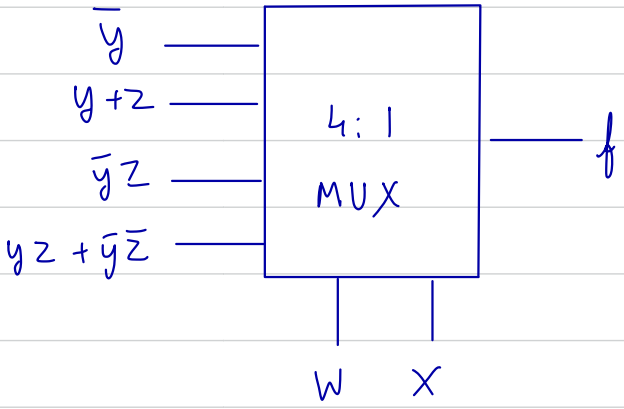
2) $f(x, y, z) = \Sigma m(0, 2, 3, 5)$ using 4:1 MUX

	Ip		op
x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



3) $f(w, x, y, z) = \Sigma m(0, 1, 5, 6, 7, 9, 12, 15)$ using 4:1 MUX

	ip			op
w	x	y	z	f
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

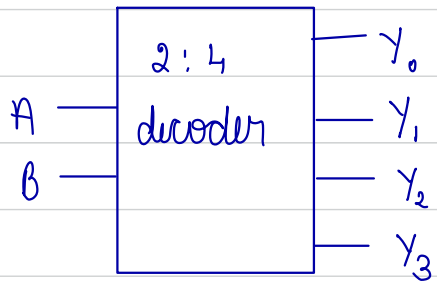


Decoders

1> Encoder: Encode information : Many lines to few lines.
2> Decoder: Decode information : Few to many lines.

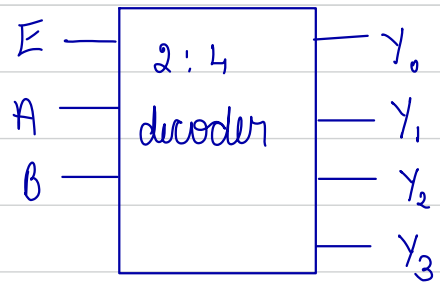
Only one output line will be high for a given input combination.

2 to 4 (2:4) decoder



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

with enable



E	A	B	Y_0	Y_1	Y_2	Y_3
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	0	1
0	X	X	0	0	0	0

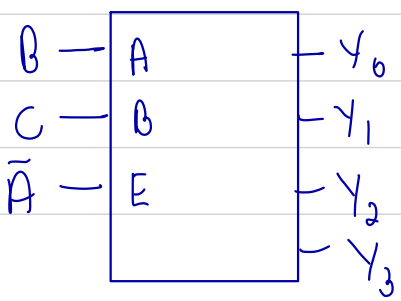
Decoder with enable as demultiplexer:

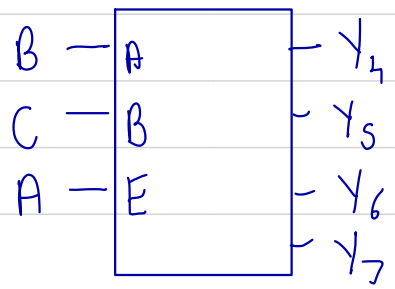
A → S_1 B → S_0 E → data

A S_1	B S_0	E $data_{in}$	Y_0	Y_1	Y_2	Y_3
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	1	0	0
1	0	0	0	0	0	0
1	0	1	0	0	1	0
1	1	0	0	0	0	0
1	1	1	0	0	0	1

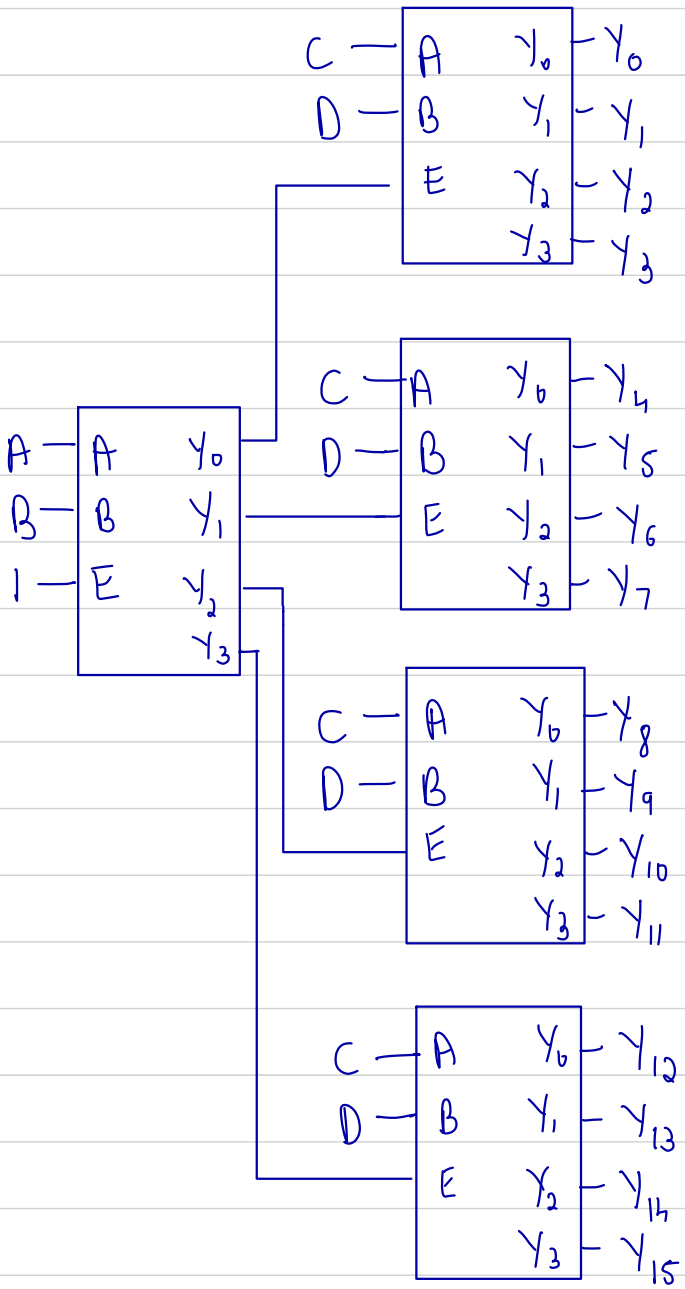
3-8 decoder

Input			Output							
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





4 to 16 decoder using 2-4 decoder:



Construction of 2-4 decoder using basic gates

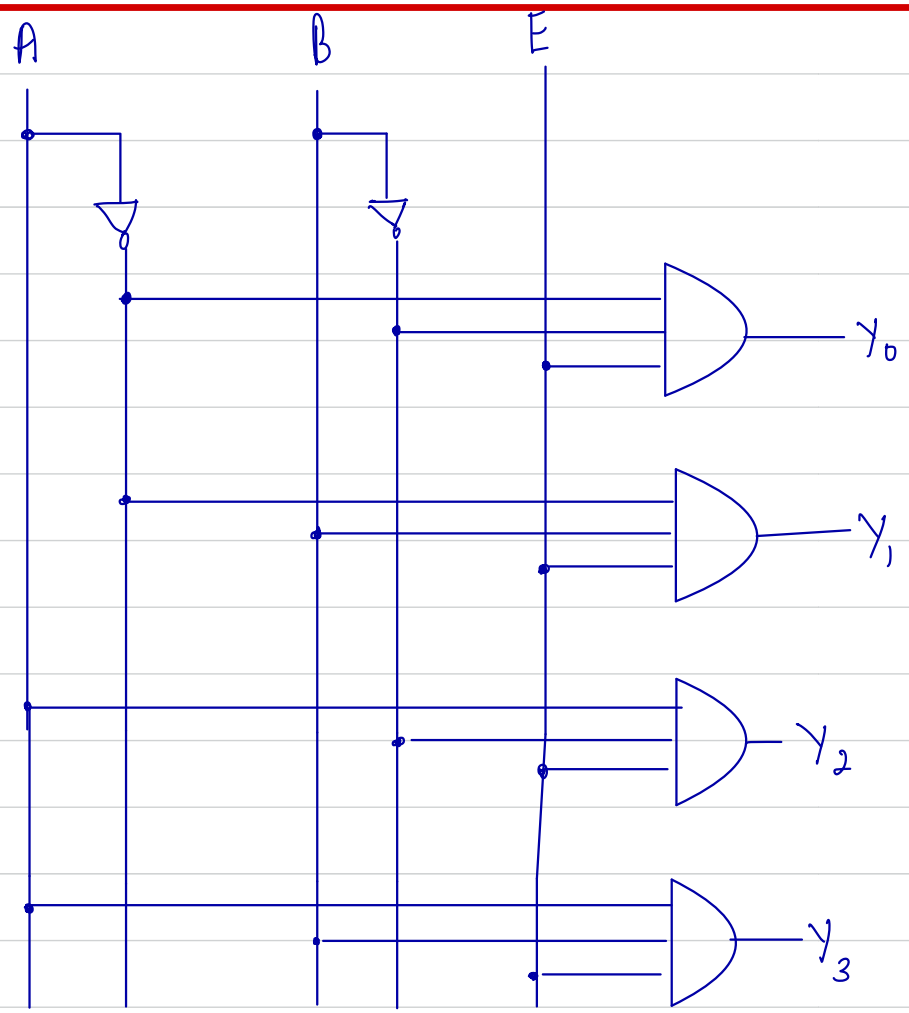
E	A	B	Y_0	Y_1	Y_2	Y_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	X	X	0	0	0	0

$\Rightarrow Y_0 = E\bar{A}\bar{B}$

$Y_1 = E\bar{A}B$

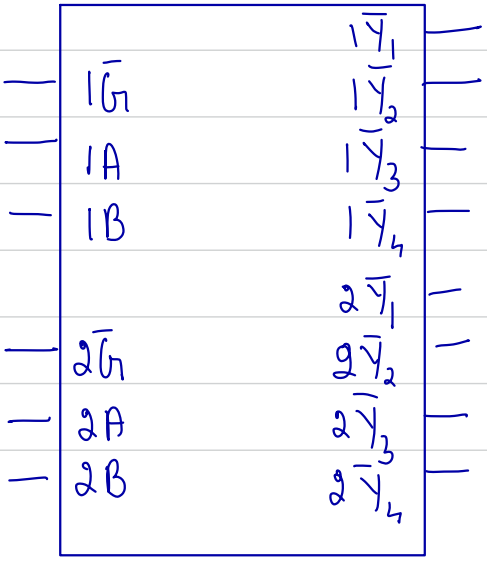
$Y_2 = EA\bar{B}$

$Y_3 = EAB$



IC 74139 → active low 2-4 decoder

active low signal: These signals are active when their value is low i.e zero.

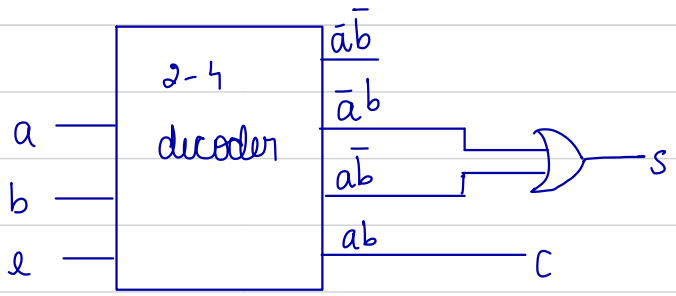


i/p			o/p			
$\bar{G}$	A	B	$\bar{Y}_0$	$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

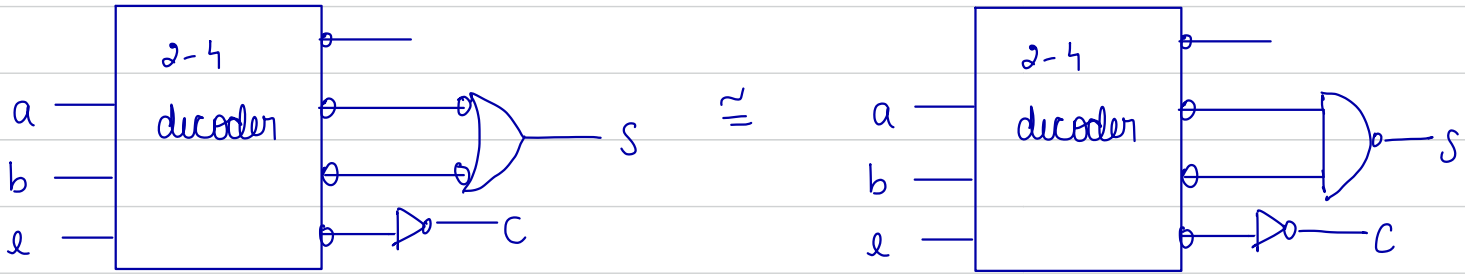
Half adders using 2-4 decoder

i/p		o/p	
a	b	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

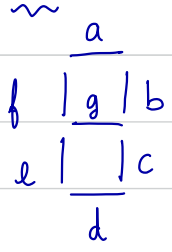
active high

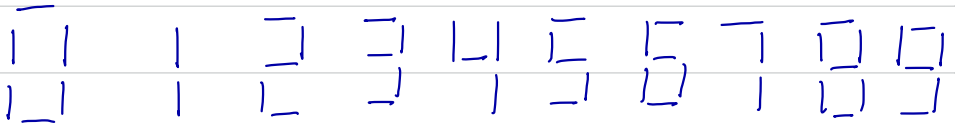


active low;

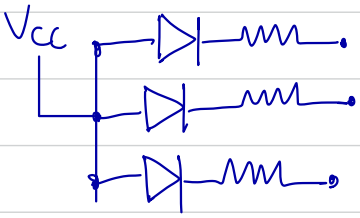


Seven segment display



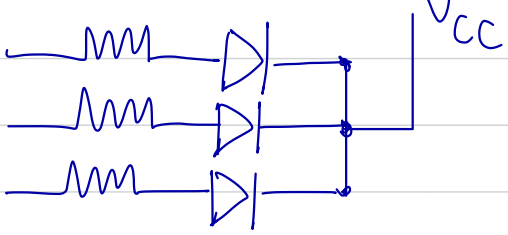
→ 

Common anode:

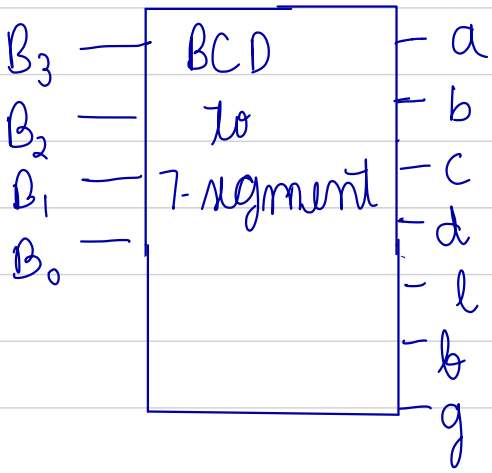


To glow any segment open terminal should be connected to 0.

Common cathode



BCD to seven segment decoder



For common anode display:

Decimal	B3	B2	B1	B0	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	1	1
2	0	0	1	0	0	0	1	0	0	1	0
3	0	0	1	1	0	0	0	0	1	1	0
4	0	1	0	0	1	0	0	1	1	0	0
5	0	1	0	1	0	1	0	0	1	0	0
6	0	1	1	0	0	1	0	0	0	0	0
7	0	1	1	1	0	0	0	1	1	1	1

Decimal	B_3	B_2	B_1	B_0	a	b	c	d	e	f	g
8	1	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	1	0	0
10	1	0	1	0	x	x	x	x	x	x	x
11	1	0	1	1	x	x	x	x	x	x	x
12	1	1	0	0	x	x	x	x	x	x	x
13	1	1	0	1	x	x	x	x	x	x	x
14	1	1	1	0	x	x	x	x	x	x	x
15	1	1	1	1	x	x	x	x	x	x	x

$a =$

$b = B_2 \bar{B}_1 B_0 + B_3 B_1 \bar{B}_0$

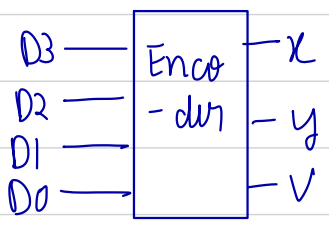
Encoder: Encodes data into binary code

Priority encoder: Priority given to inputs to generate o/p code.

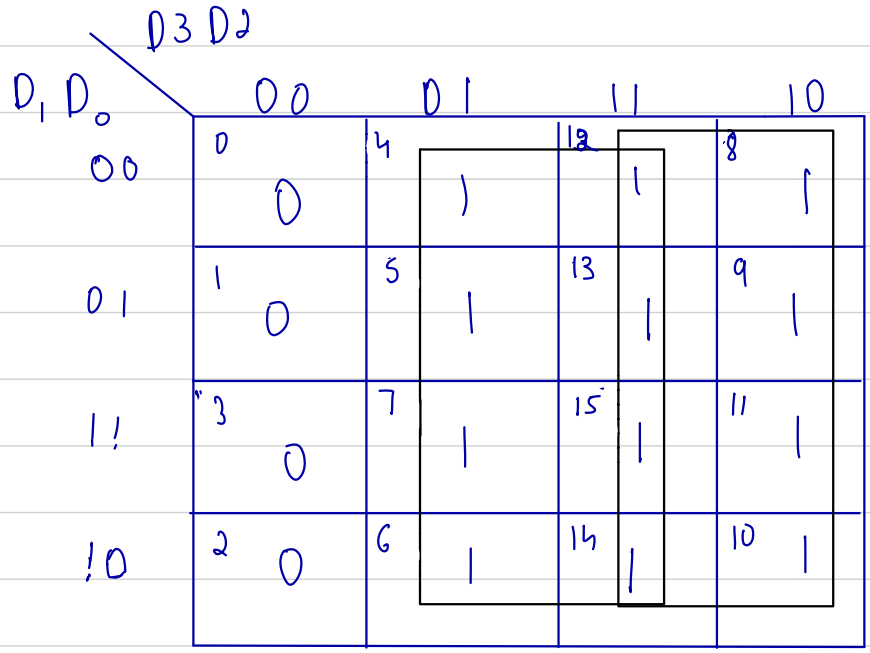
4 to 2 priority encoder

$D_3 \rightarrow$ highest priority, $D_0 \rightarrow$ least priority

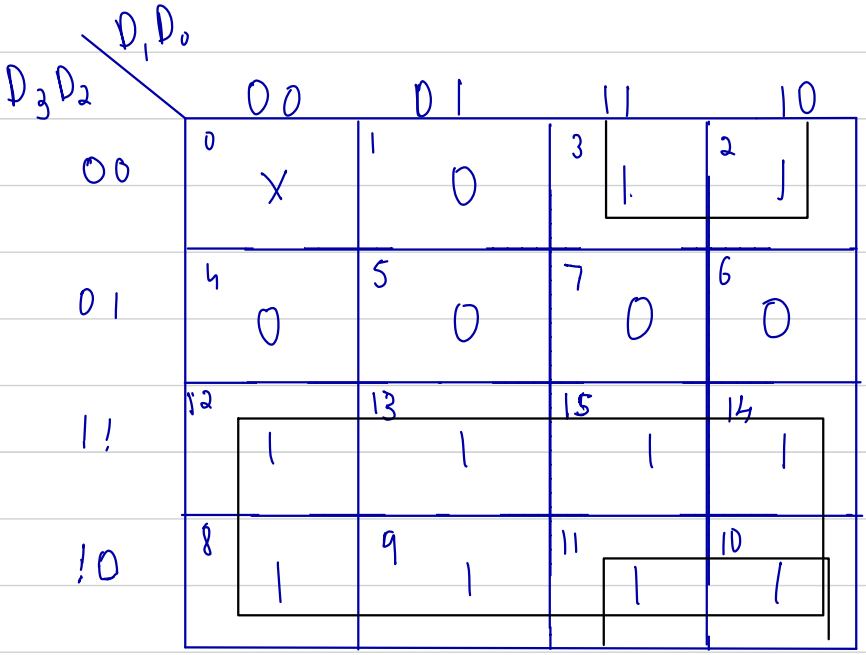
D_3	D_2	D_1	D_0	x	y	V
0	0	0	0	x	x	0
0	0	0	1	0	0	1
0	0	1	x	0	1	1
0	1	x	x	1	0	1
1	x	x	x	1	1	1



For x:



$x = D_3 + D_2$



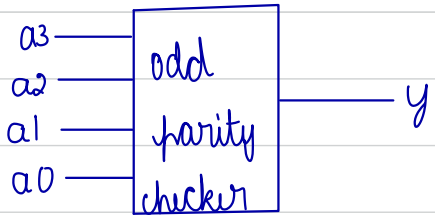
$y = D_3 + \overline{D}_2 D_1$

$V = D_0 + D_1 + D_2 + D_3$

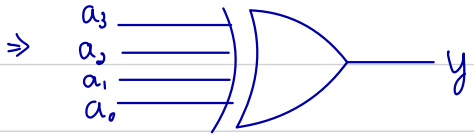
Parity generator / checker

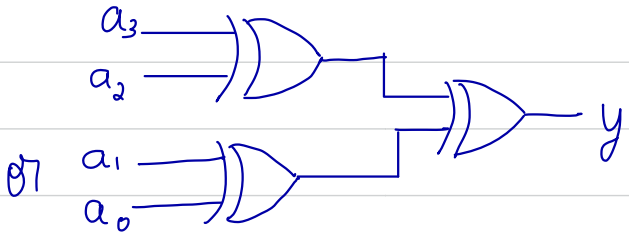
(One of the error detection method.

Based on number of ones in the sequence.
Odd number - odd parity , even number - even parity

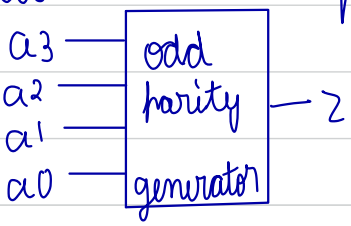


$y = a_3 \oplus a_2 \oplus a_1 \oplus a_0$





Parity generator: Parity generate will append a bit at the end based on parity.
Odd parity generator will append 1 for even parity and 0 for odd parity. i.e it maintains odd parity

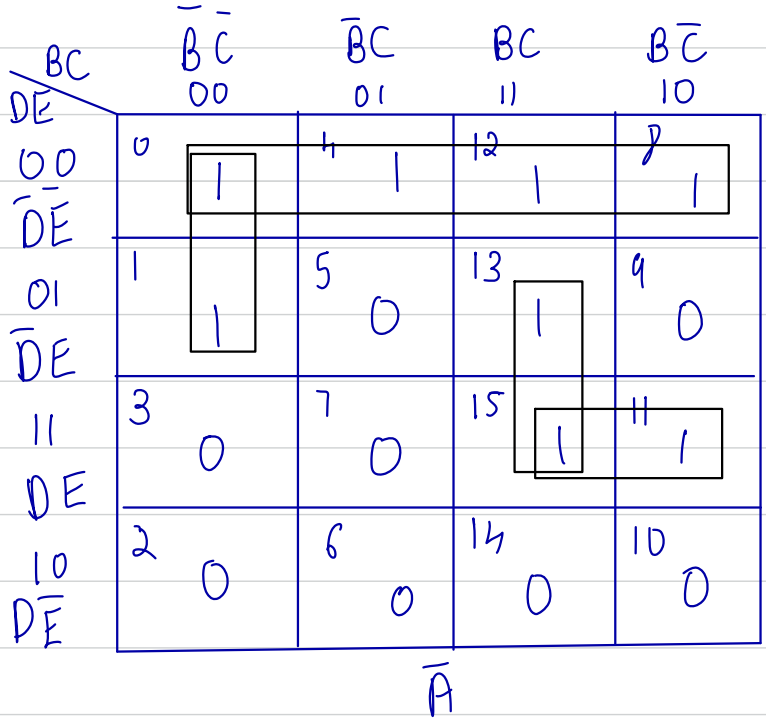


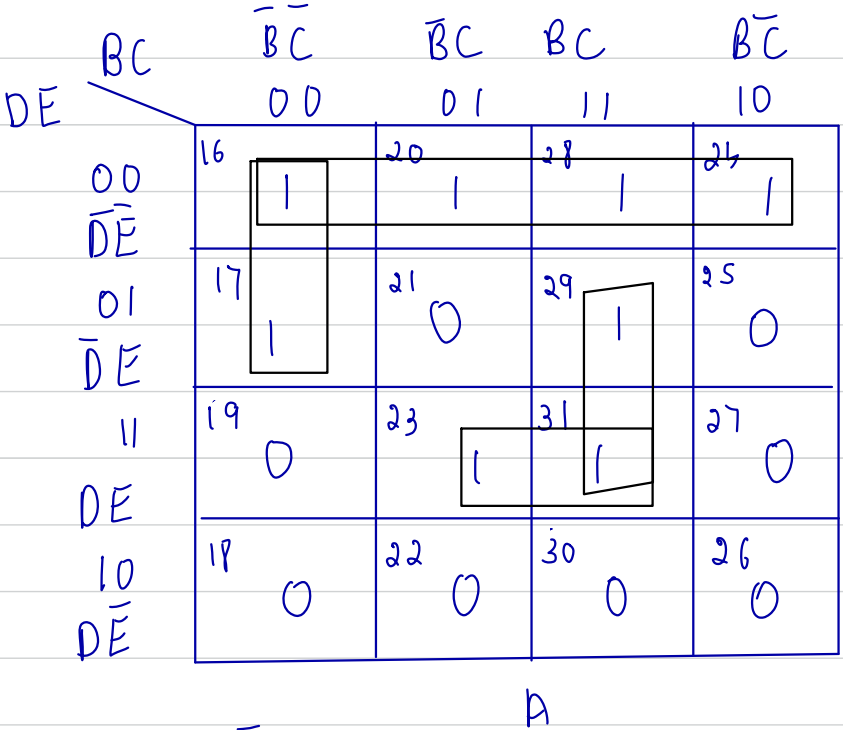
$z = \bar{y} = a_3 \odot a_2 \odot a_1 \odot a_0$

5 variable K-map

f(A, B, C, D, E)

Ex: $F(A, B, C, D, E) = \sum m(0, 1, 4, 8, 11, 12, 13, 15, 16, 17, 20, 23, 24, 28, 29, 31)$

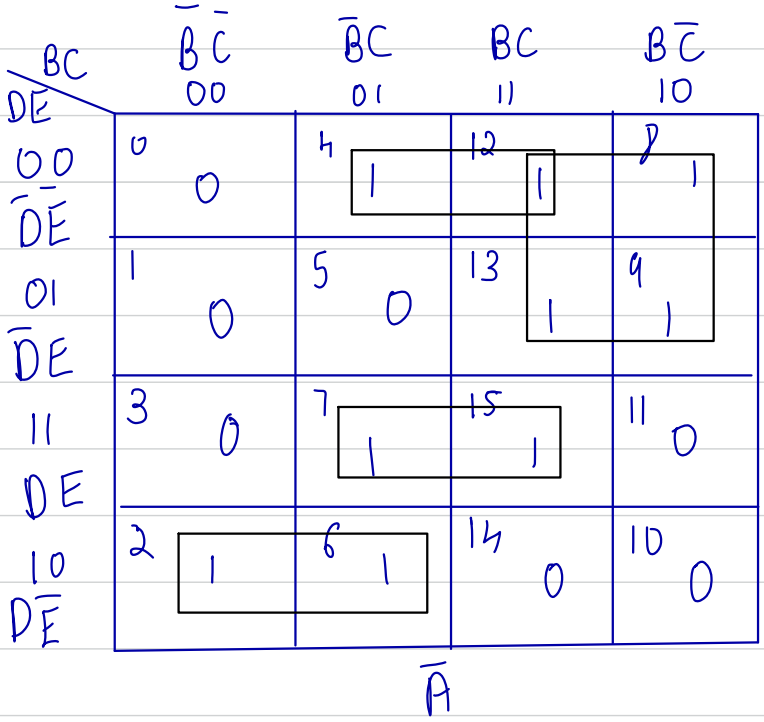


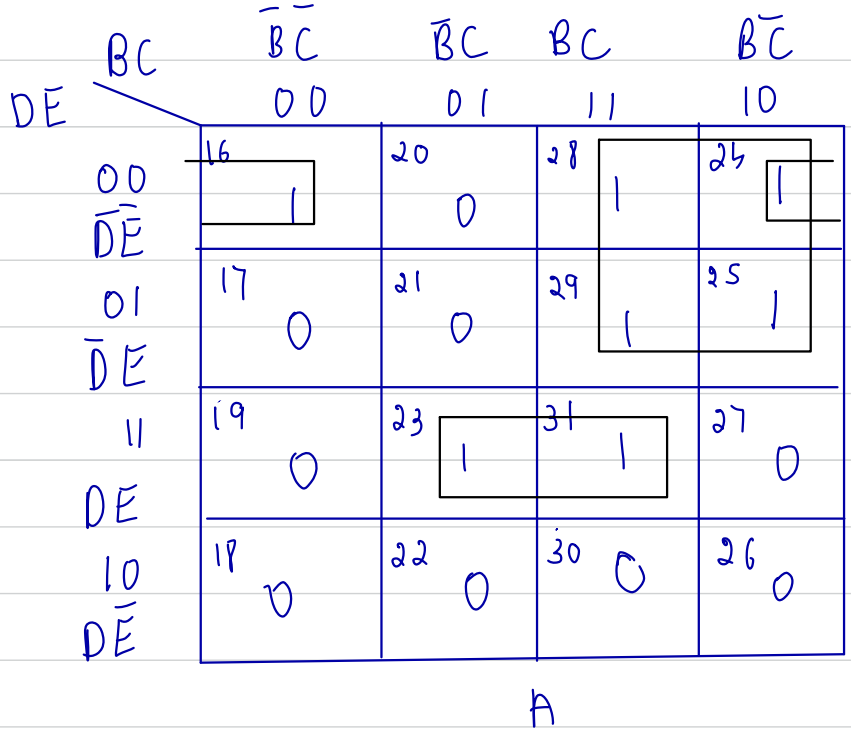


$$\therefore F = \bar{A}\bar{D}\bar{E} + A\bar{D}\bar{E} + \bar{A}BCE + ABCE + \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}BD\bar{E} + ACD\bar{E}$$

$$= \bar{D}\bar{E} + BCE + \bar{B}\bar{C}\bar{D} + \bar{A}BD\bar{E} + ACD\bar{E}$$

Eg2 $F(A, B, C, D, E) = \sum m(2, 4, 6, 7, 8, 9, 12, 13, 15, 16, 23, 24, 25, 28, 29, 31)$.





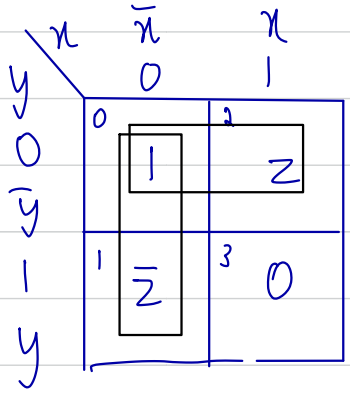
$$F = B\bar{D} + \bar{A}C\bar{D}\bar{E} + CDE + \bar{A}BD\bar{E} + A\bar{C}\bar{D}E$$

Variable entered map

- Variable & its compliment are entered in the map.
- It reduces the map order by 1
- Useful when some variable occur less frequently

Ex: $f(x, y, z) = \Sigma m(0, 1, 2, 5)$

x	y	z	f	
0	0	0	1	$\bar{y}\bar{z}$
0	0	1	1	
0	1	0	1	
0	1	1	0	$y\bar{z}$
1	0	0	0	
1	0	1	1	xz
1	1	0	0	
1	1	1	0	



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

→ Variables are grouped with itself or 1

Ex2: $F(w, x, y, z) = \Sigma m(4, 5, 11, 12, 13, 15)$

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

wx \ yz	$\bar{y}\bar{z}$ (00)	$\bar{y}z$ (01)	yz (11)	$y\bar{z}$ (10)
$\bar{w}\bar{x}$ (0)	0	0	0	1
$\bar{w}x$ (1)	0	z	1	z
w (1)	0	1	1	z

$$f = wyz + x\bar{y}z$$

Introduction to Verilog

Layout → Transistor level → Gate level → Architectural level → System level.

Design flow: customer requirements

↓

Specifications

↓

Architectural level synthesis

↓ verification

RTL design (Register transfer logic)

↓ Front end

Logic synthesis

↓ verification

Physical Layout ← Back-end

↓ verification

Test planning

↓

Fabrication

↓

Test → chips to customer

Synthesis: Process of transforming one representation in the design abstraction hierarchy to another representation.

Hardware descriptive language (HDL)

- Textual description of a circuit
- Used for design entry in computers
- High level programming language used to model hardware for simulation verification and synthesis
- Have special hardware related constructs.
- Standard description for exchange of designs.
- Concurrent statement execution. (Parallel execution)

Ex: Verilog, VHDL (But verilog is used mostly)

Verilog code (2:1 MUX)

- Module - basic unit. It describes ports & functionality.

```
module mux2to1 (Y, S, I0, I1);
    input I0;
    input I1;
    input S;
    output Y;

wire sbar, tmp1, tmp2;
not (sbar, S);
and (tmp1, sbar, I0);
and (tmp2, S, I1);
or (Y, tmp1, tmp2);
endmodule;
```

Syntax

```
module  module_name (port_list);
```

declarations:

reg, wire, parameter,
input, output, inout,
function, task, ...

Statements:

initial statement
always statement
Module instantiation
Gate instantiation
continuous assignment
UDP instantiation
....

```
endmodule;
```

Test bench

Syntax for mux 2:1 -

```
module  Tb_mux ()
 reg  s, a, b;
 wire  y;
```

```
mux2to1 (s, a, b, y);
initial
 begin
 S=0;   a=0;  b=0;
 #1
 S=0;   a=0;  b=1;
 #1
S=0;   a=1;  b=0;
 #1
 :
 :

S=1;  a=1;  b=1;
#1
$ finish.
end
endmodule;
```

2nd method:

```
 module  tb mux2to1  ( );
 reg  s, a, b;
reg [2:0] i;
wire y;
mux2to1 (s, a, b, y);
initial  begin
for (i=0; i<8; i=i+1)  begin
#1 {s, a, b} = i;
end
$ finish;     end     endmodule;
```

4:1 MUX using 2:1 MUX

```
module mux4to1 (
  input [1:0] Sel
  input [3:0] I,
  output Y
);
wire st10, st11;
mux2to1 M1(Sel[0], I[0], I[1], st10),
        M2(Sel[0], I[2], I[3], st11),
        M3(Sel[1], st10, st11, Y);
endmodule
```

- This is structural model.

Data flow representation

- Represented as a set of continuous assignments.
- Use keyword 'assign'.
- Mostly for describing the Boolean equation of combinational logic.
- Use operators that act on binary operands to produce a binary result.
- Bitwise & logical operator - ~(1010) is 0101; !(1010) is 0.

Test bench for 4:1 MUX

```
module mux4to1tb ( );
 reg [1:0] s;
 reg [3:0] inp;
 integer i, j;
 wire y;
 mux4to1 (s, inp, y)
 initial begin
 for (i=0; i<4; i=i+1) begin
    s=i;
    for (j=0; j<16; j=j+1) begin
    #1 inp=j
    end
end
 $finish;
 end
endmodule
```

Verilog Operators

{ }	concatenation
+ - * / **	arithmetic
%	modulus
> >= < <=	relational
!	logical NOT
&&	logical AND
||	logical OR
==	logical equality
!=	logical inequality
===	case equality
!==	case inequality
? :	conditional

~	bit-wise NOT
&	bit-wise AND
|	bit-wise OR
^	bit-wise XOR
^~ ~^	bit-wise XNOR
&	reduction AND
|	reduction OR
~&	reduction NAND
~|	reduction NOR
^	reduction XOR
~^ ^~	reduction XNOR
<<	shift left
>>	shift right

2:1 MUX in dataflow model:

```
module mux2to1 (
    input S,
    input I0,
    input I1,
    output Y );
assign Y = (!S && I0) || (S && I1);
 endmodule;
```

Structural Representation

- Use gate level primitives, user defined primitives, module instances (hierarchy)
- Representation similar to interconnected hardware
- Ports - data type - net - connection
- Named port assignments and ordered port assignments
- Port name of sub module (port name of outer module)

Net & Variable types

- Predefined by language - user defined type not allowed.
- Net - node in a circuit.
- Commonly used net - wire and tri.

- Variable - abstract data storage element
- Commonly used variable - integer & reg

Net:

- Wire - connect output of logic element to input of other logic element.
- Scalar wire decleration:
  - wire x;
  - wire cin, sout;
- Vector wire decleration: wire [3:0] S;
  wire [1:2] arr;

Wire: Bit selection operation - f = S[0] - index can be variable
Part selection operation - arr = S[2:0]

Tri: - Represents nodes connected in tristate form.
Eg: tri z; tri [7:0] dt;
- Used to enhance readability of codes involving tri state gates.

Variables: Commonly used for behavioral modelling
Eg: reg [2:0] cnt; integer k;
→ Memory - 2D array :- eg: Reg [7:0] Reg [3:0]
R - 4 8-bi variable - R[1] - 8 bit variable - R[3][7]

Behavioral Representation

→ Structural representation using gate level primitives - difficult of large designs
→ Use verilog procedural statement.
→ Used inside always block
→ Procedural statement ex: If - else : case : loop
→ Datatype - Variable (reg) - assigned a value inside always block and retains value till a new value is assigned.

Syntax of always statement

always @(sensitivity_list)
[begin]
[procedural assignment statements]
[if - else statements]
[case statements]
[while, repeat & for loops]
[task and function calls]
[end]

2:1 MUX in behavioral style:

```
1> module mux21(S, a, b, Y)
input S, a, b;
output Y;
reg Y;
```

```
always @(s, a, b) begin // <------ Sensitivity list
  if (s== 1'b 0)
     Y <= a;
  else      Y <= b;
end
endmodule
```

2)
```
module    mux21 (s, a, b, Y)
 input    s, a, b;
 output   Y;
 reg Y;
 always @ (s, a, b) begin
 case (s)
0: Y <= a;
1: Y <= b;
endcase
end
endmodule
```

Loop Statements

1) for:-

```
for ( initial_index ; terminal_index ; increment)
 begin
       statement;
end
```

2> repeat:

```
repeat ( constant value)
begin
        statement;
end
```

3> while:

```
while ( condition)
begin
        statement;
end
```

Test bench verification

```
`timescale 1ns / 1ps    // Here 1ns is scale & 1ps is precision.
module muxtb( );
reg S;
reg I0;
reg I1;
wire Y;
integer i;
mux21 m1(S, I0, I1, Y)
initial begin
for (i=0; i<8; i=i+1)
begin
        {S, I0, I1} = i;
      # 5
```

```
if ((S=1'b0) && (y!=I0)) fail();
else if ((S=1'b1) && (y!=I1)) fail();
 else pass();
end
$ display ("Design verified succesfully");
$ finish();
end
        task fail();
  begin
        $ display ("when, S, a, b = %b, %b, %b, output
= %b is wrong", S, I0, I1, Y);
 $ finish();
 end
endtask

task pass();
 $ display ("when S, a, b = %b, %b, %b, output = %b",
S, I0, I1, Y);
 endtask
endmodule
```

Blocking versus Non-blocking Procedural assignments

- Blocking (=) or Non blocking (<=)
- LHS - register type variable (reg, integer, time, real)
  - Bit select (a[3]) or part select (b[3:1]) of variable
  - concatenation of any of the above ({S, a[3], b[3:1]})
- RHS - any expression consisting of net or reg that evaluates to

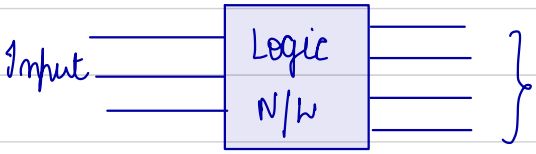
Blocking procedural assignments

- Sequential execution
- LHS variable updates before next statement is executed
- Sequential execution limited to that procedural block.
- Recommended style for combinational circuits.

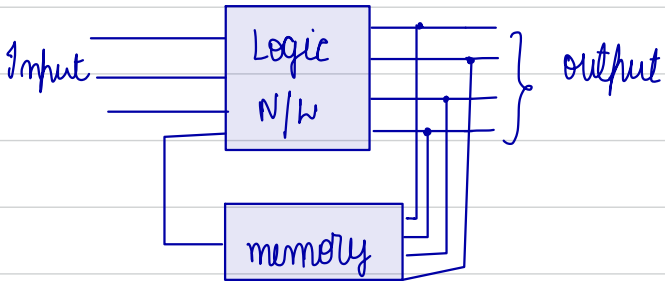
Non blocking assignments

- Schedule events on LHS variable without blocking any other assignment within the block.
- Variables updated simultaneously at the end of that simulation time.
- Concurrent procedural assignment
- Recommended style for sequential circuits.

Sequential network



combinational circuit



Sequential output

Types of sequential network.

Asynchronous network: output is shown immediately whenever there is a change in input.

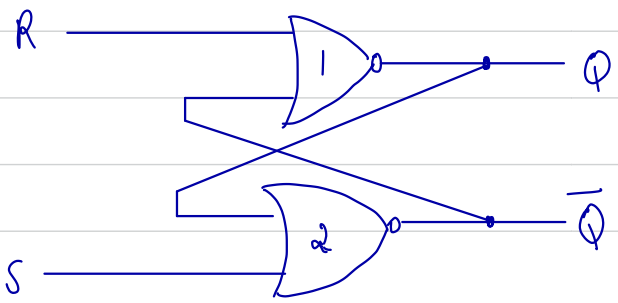
Synchronous network: output is shown after some time whenever there is a change in input. Clock signal is used to create a delay.

Latch: - Fundamental building block

1) SR latch: (S-R = Set - Reset)

nor

a	b	Y
0	0	1
0	1	0
1	0	0
1	1	0

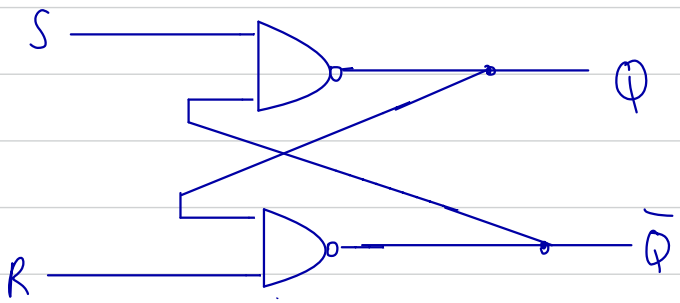


R	S	Q	$\bar{Q}$
0	0	Q_{n-1}	$\bar{Q}_{n-1}$
0	1	1	0
1	0	0	1
1	1	0	0

→ Function table

→ But invalid combination as Q & $\bar{Q}$ both are 0

If R, S changes from 11 to 00 it becomes metastable.



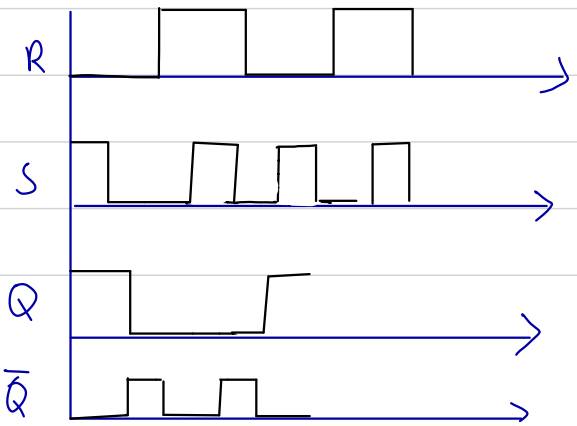
← Another type

R	S	Q	$\bar{Q}$
0	0	0	0
0	1	0	1
1	0	1	0
1	1	Q_{n-1}	$\bar{Q}_{n-1}$

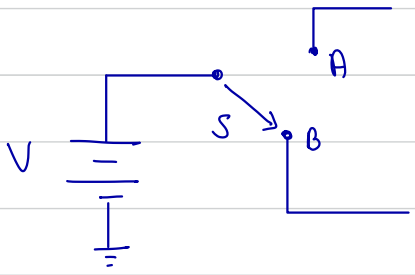
→ Not allowed

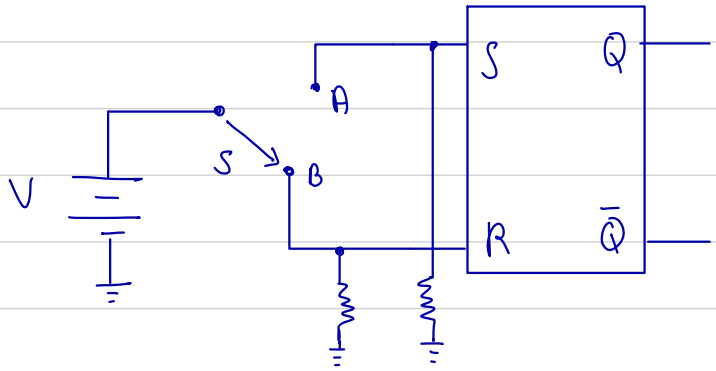
Symbol:

Timing diagram:



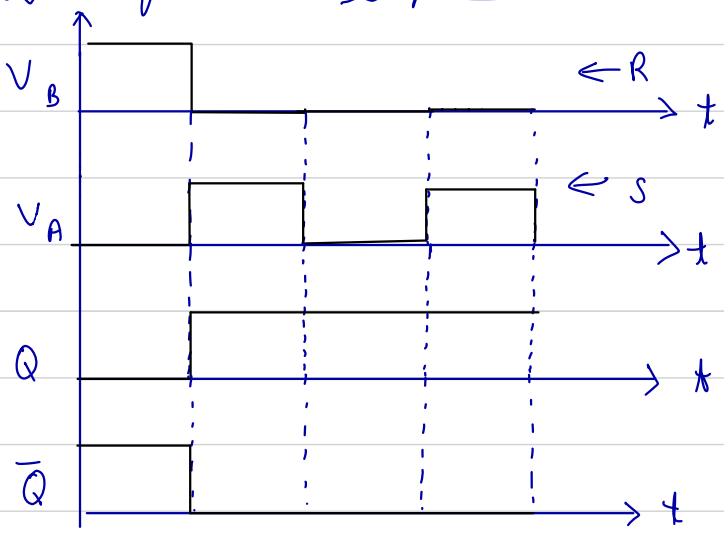
Switch debounce



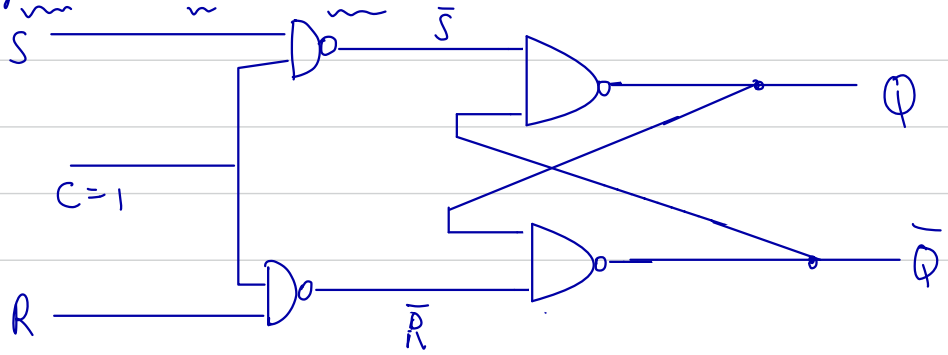


→ Switch debounce

Timing diagram



Gated SR latch

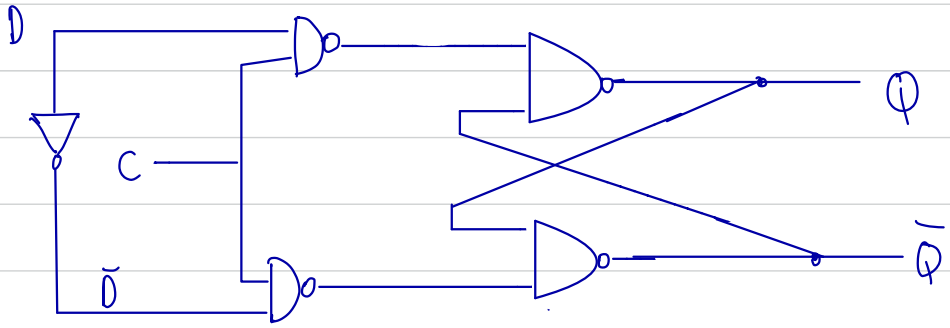


Function table

i/p			o/p	
C	S	R	Q	$\bar{Q}$
0	x	x	Q_{n-1}	$\bar{Q}_{n-1}$
1	0	0	Q_{n-1}	$\bar{Q}_{n-1}$
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

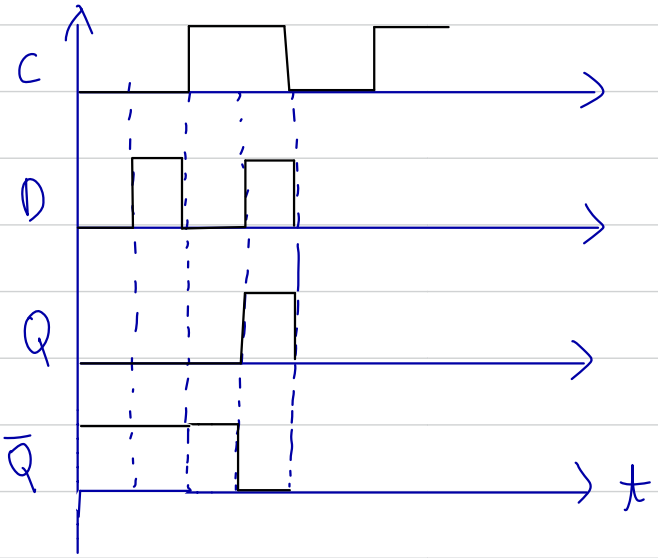
→ not valid

Data latch (D - latch)

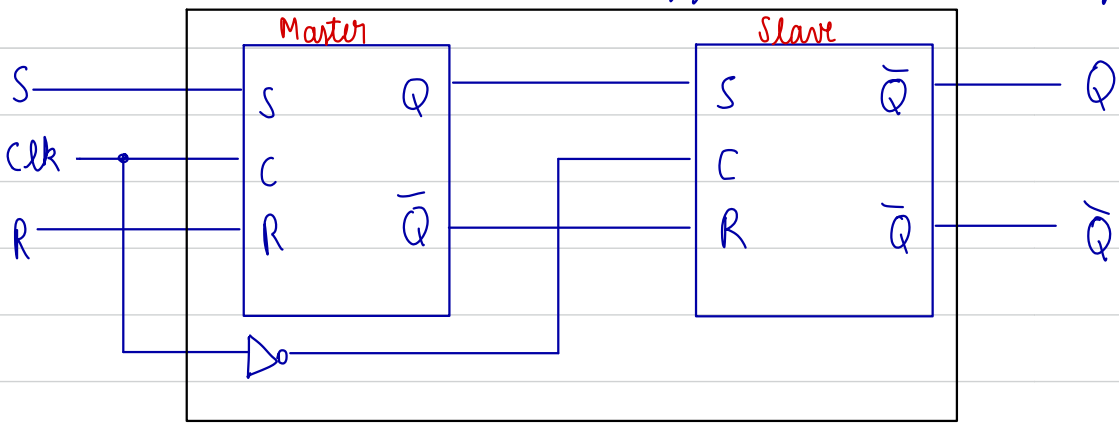


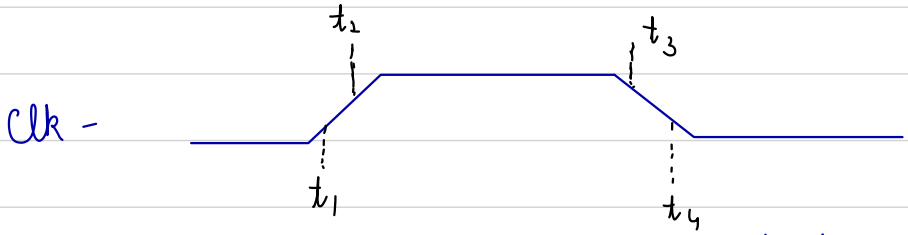
Function table

i/p		o/p	
C	D	Q	$\bar{Q}$
0	0	Q_{n-1}	$\bar{Q}_{n-1}$
0	1	Q_{n-1}	$\bar{Q}_{n-1}$
1	0	0	1
1	1	1	0



Flipflops → Output changes with change in control line: Masterslave (pulse triggered) or edge triggered.





$t < t_1$ & $t > t_4$ → slave enabled

$t_2 < t < t_3$ → master enabled

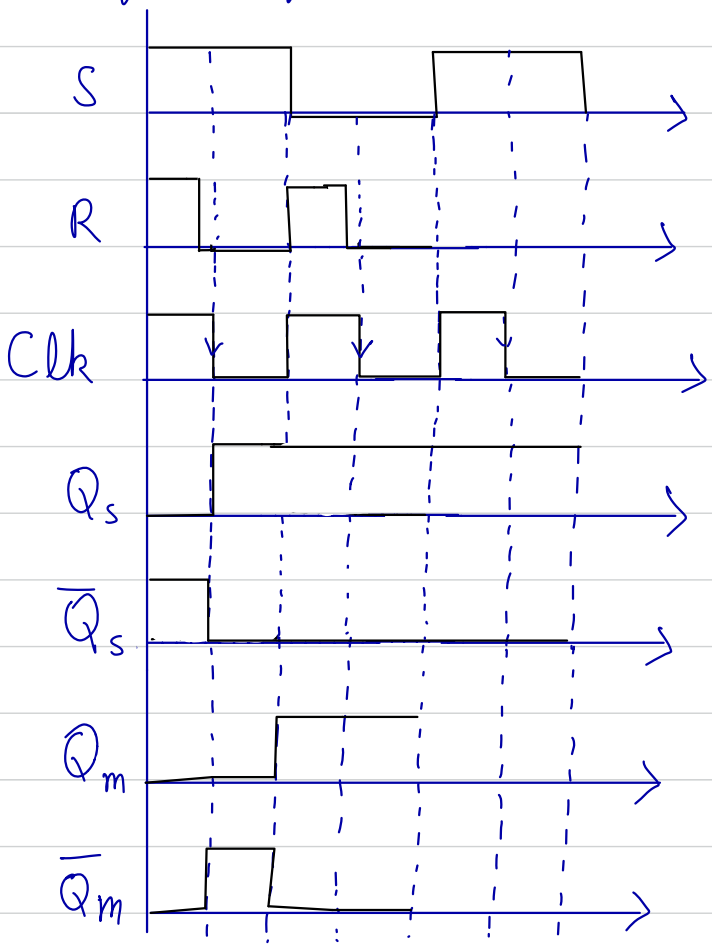
$t_1 < t < t_2$ & $t_3 < t < t_4$ → both are disabled.

Function table

i/p			o/p	
S	R	C	Q	$\bar{Q}$
1	0	⊓	1	0
0	1	⊓	0	1
0	0	⊓	Q_{n-1}	$\bar{Q}_{n-1}$
1	1	⊓	0	0 → not allowed

Output is shown only after complete pulse (⊓)

Timing diagram:



o/p depends on S, R as well as ↑, ↓ & duration of  ⇒ Pulse triggered.

5-Variable K Map

Simplify the Boolean function using K Map

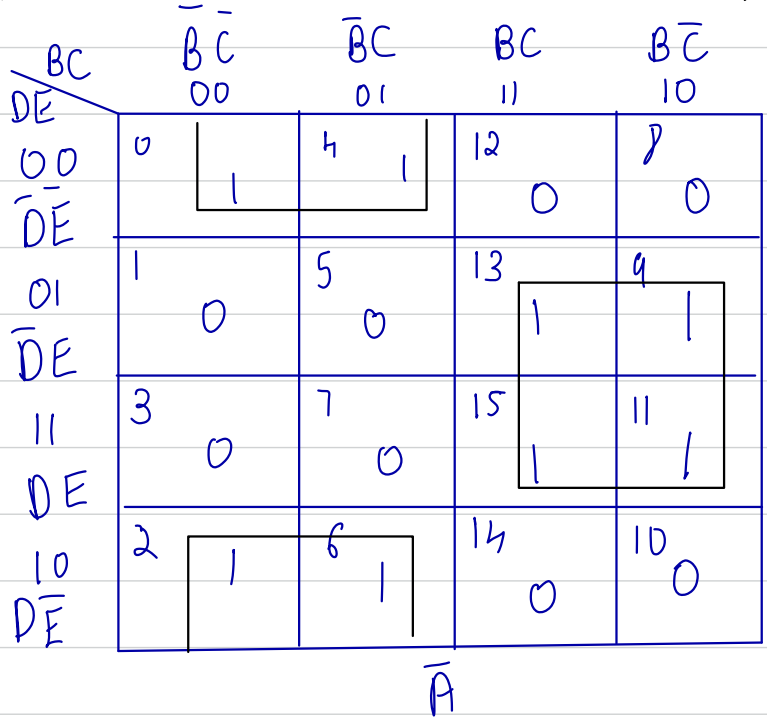
1. F (A, B, C, D, E) = ∑m (0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)
2. F (x1, x2, x3, x4, x5) = ∑m (1, 2, 3, 6, 8, 9, 14, 17, 24, 25, 26, 27, 30, 31) + ∑d (4, 5)
3. F (P, Q, R, S, T) = ∏M(0, 5, 6, 8, 9, 10, 11, 16, 20, 22, 24, 25, 26, 27)

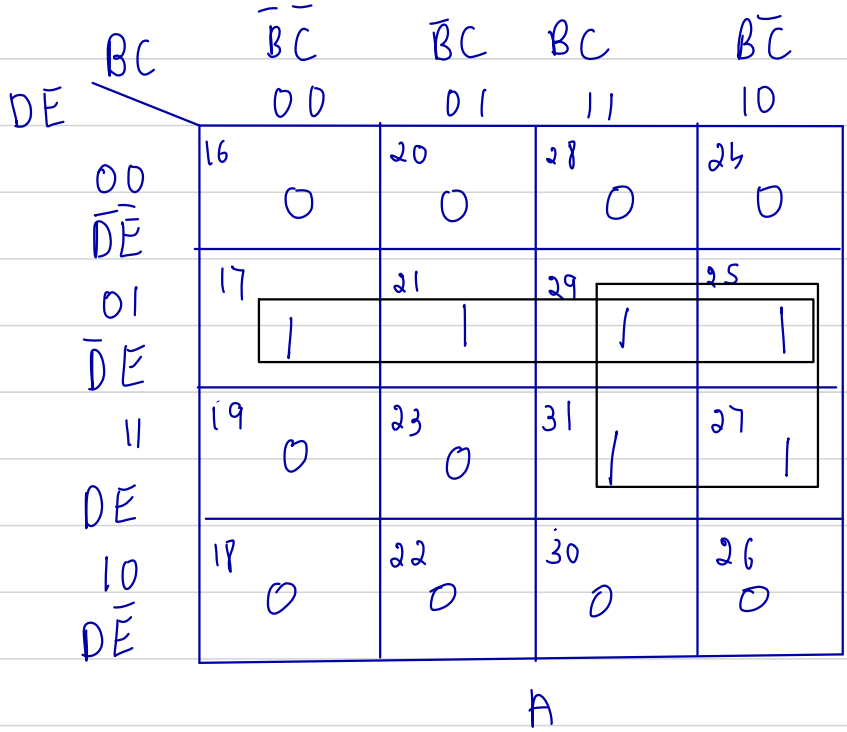
VEM Method

Simplify the Boolean function using VEM Method

1. F(A, B, C) = ∑m (0, 1, 4, 5, 7)
2. F (P, Q, R, S) = ∑m (1, 5, 6, 12, 13, 14) + ∑d (2, 4)
3. F (P, Q, R, S, T) = ∑m (0, 2, 4, 6, 7, 9, 12, 13, 15, 18, 21, 22, 23, 24, 25, 26, 27)

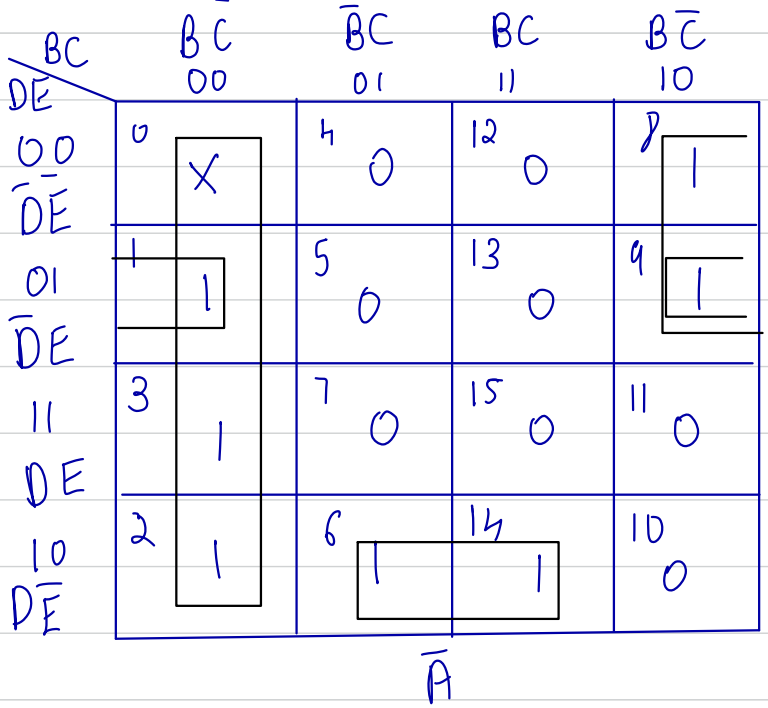
Q1) $F(A,B,C,D,E) = \Sigma m(0,2,4,6,9,11,13,15,17,21,25,27,29,31)$

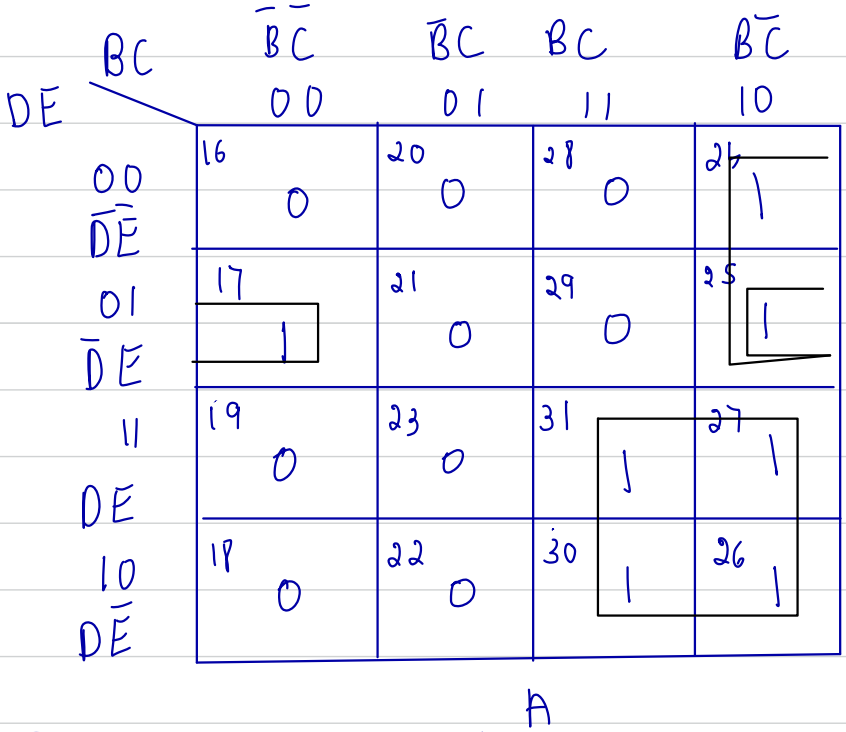




$F = BE + A\bar{D}E + \bar{A}\bar{B}\bar{E}$

2> $F(A,B,C,D,E) = \Sigma m(1, 2, 3, 6, 8, 9, 14, 17, 24, 25, 26, 27, 30, 31) + \Sigma d(0)$

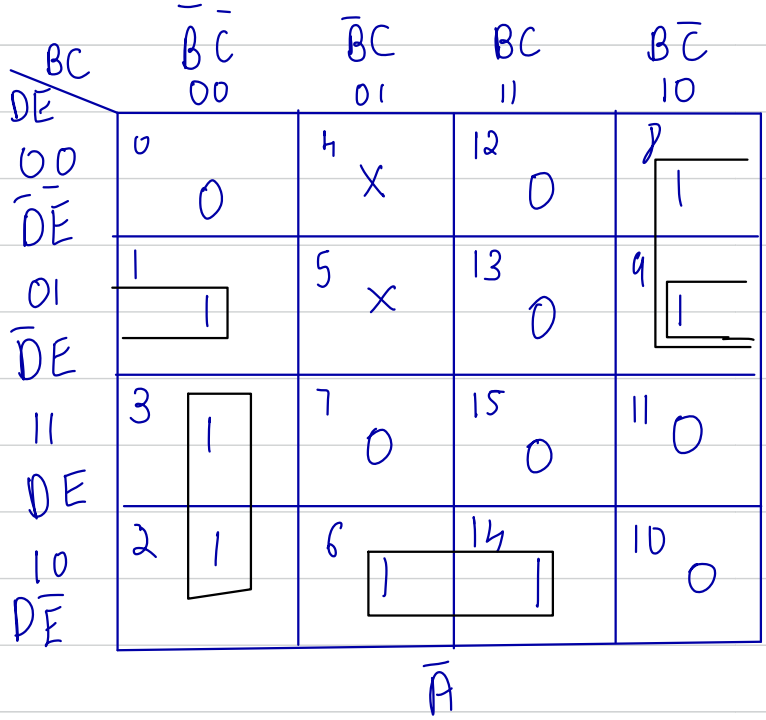


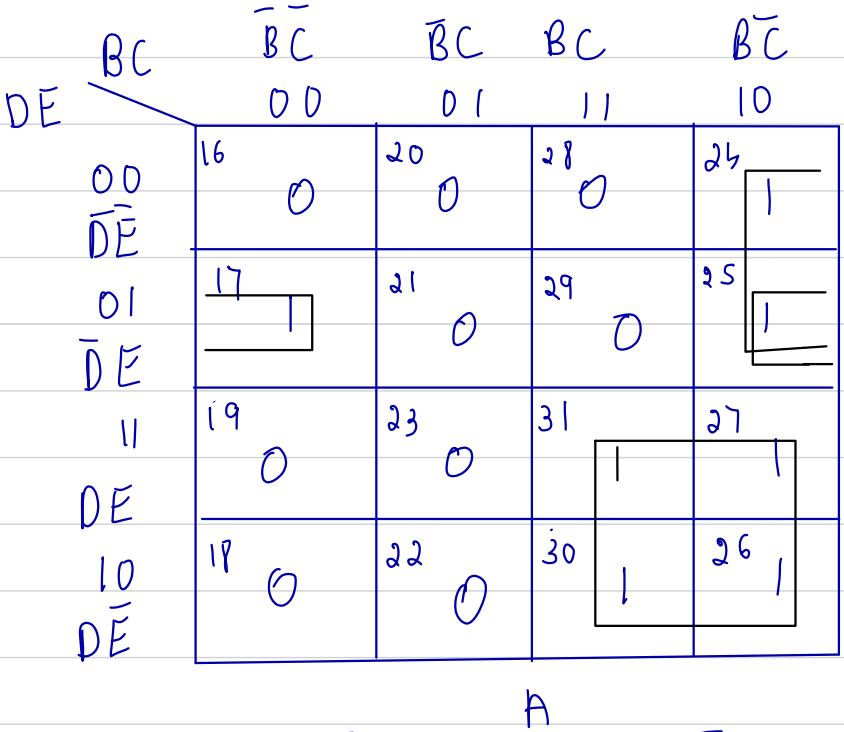


$F = \bar{A}\bar{B}\bar{C} + B\bar{C}\bar{D} + \bar{A}CD\bar{E} + ABD + \bar{C}\bar{D}\bar{E}$

3> $F(A,B,C,D,E) = \Sigma m(1,2,3,6,8,9,14,17,24,25,26,27,30,31) + \Sigma d(4,5)$

→





$F = \bar{A}\bar{B}\bar{C}D + \bar{C}\bar{D}E + \bar{A}CD\bar{E} + ABD + B\bar{C}\bar{D}$

4> $F(P,Q,R,S,T) = \pi M(0,5,6,8,9,10,11,16,20,22,24,25,26,27)$

→

S+T \ Q+R	Q+R (00)	Q+R̄ (01)	Q̄+R̄ (11)	Q̄+R (10)
00 S+T	0 (0)	4 1	12 1	8 0
01 S+T̄	1 1	5 (0)	13 1	9 0
11 S̄+T	3 1	7 1	15 1	11 0
10 S̄+T	2 1	6 0	14 1	10 0

P

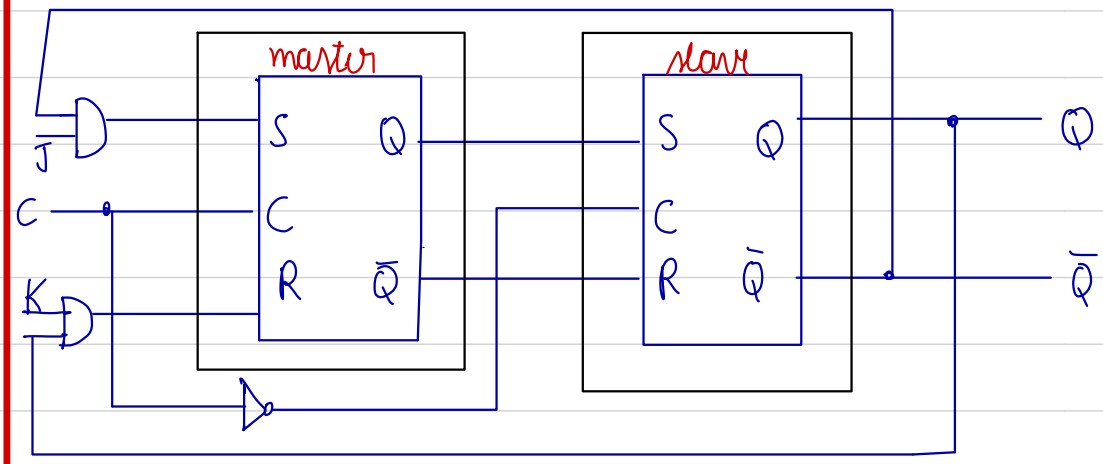
S+T \ Q+R	Q+R (00)	Q+R̄ (01)	Q̄+R̄ (11)	Q̄+R (10)
00 S+T	16 (0)	20 0	28 1	24 0
01 S+T̄	17 1	21 1	29 1	25 0
11 S̄+T	19 1	23 1	31 1	27 0
10 S̄+T	18 1	22 0	30 1	26 0

P̄

$$F = (\bar{Q}+R)\cdot(Q+R+S+T)(Q+\bar{R}+\bar{S}+T)(\bar{P}+Q+S+T)(P+Q+\bar{R}+S+\bar{T})$$

JK flipflop

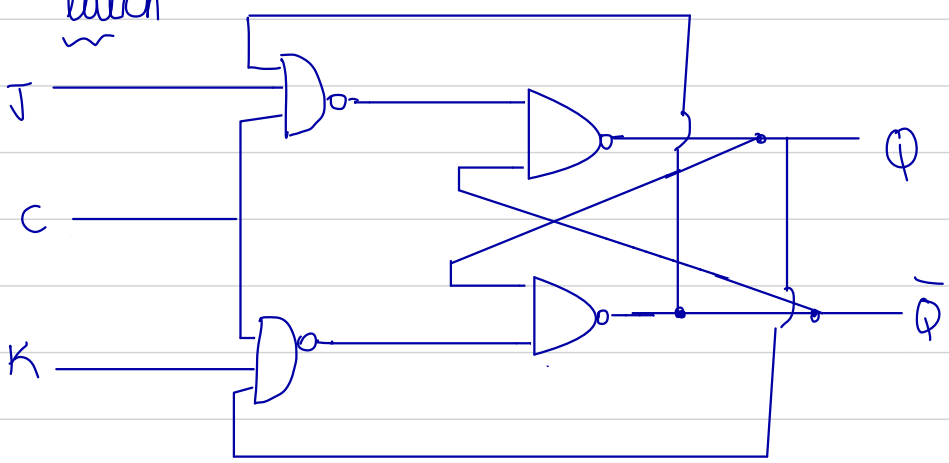
At '11' in JK flipflop will toggle o/p.



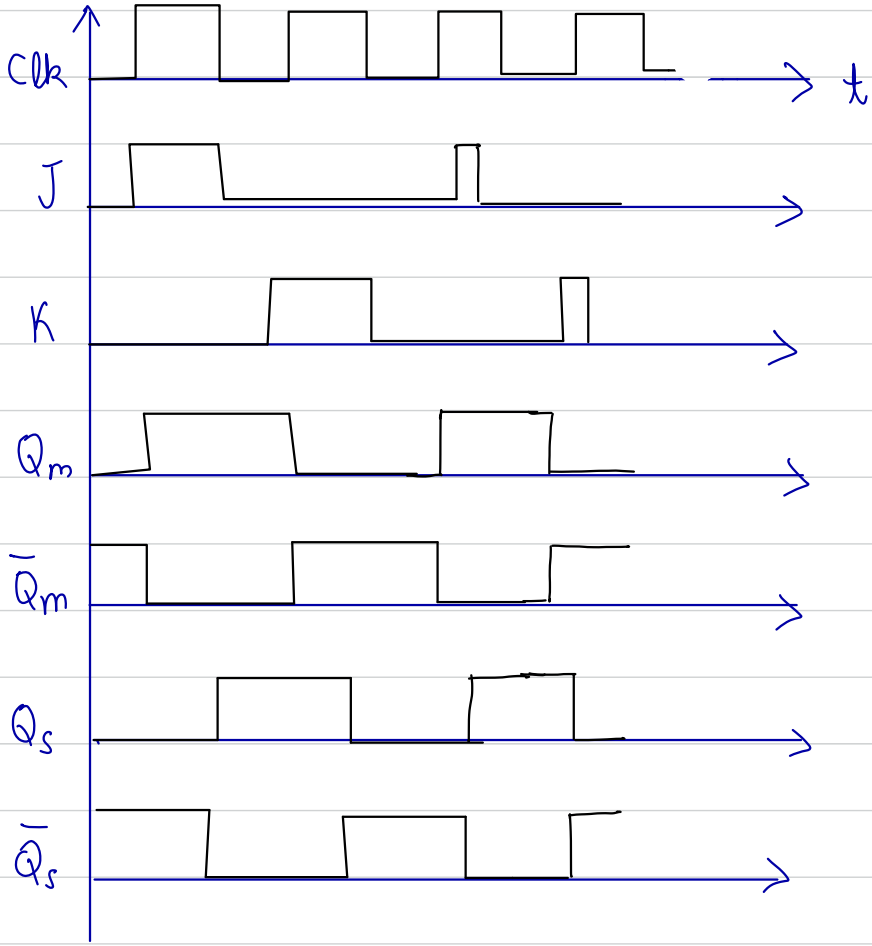
Function table

J	K	C	Q	$\bar{Q}$
X	X	0	Q_{n-1}	$\bar{Q}_{n-1}$
0	0	⊓	Q_{n-1}	$\bar{Q}_{n-1}$
0	1	⊓	0	1
1	0	⊓	1	0
1	1	⊓	Toggle	

JK latch



Timing diagram for JK flipflop



1's catch:

$Q = 0$

$J = 0 \rightarrow 1 \rightarrow 0$

$K = 0$

$clk = 1$

$Q = 0 \rightarrow 1$

0's catch

$Q = 1$

$clk = 1$

$J = 0$

$K \rightarrow 0 \rightarrow 1$

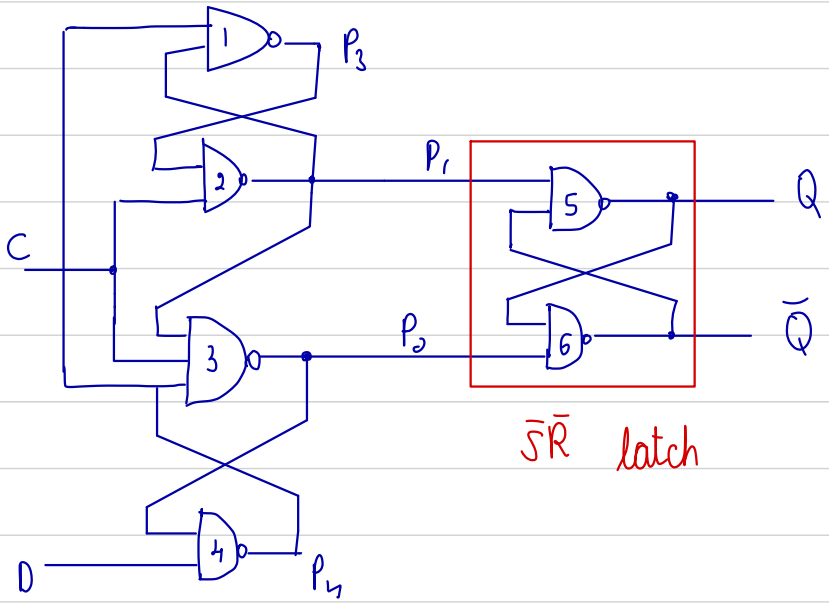
Flipflops: (JK, SR, D, T (toggle))

T	C	Q_n	$\bar{Q}_n$
0	⊓	Q_{n-1}	$\bar{Q}_{n-1}$
1	⊓	$\bar{Q}_{n-1}$	Q_{n-1}

← T flipflop

MS flipflop : • 0's & 1's catching will have T/2 delay in O/P.
• So edge triggered flipflops are used.
• only one clock edge to control.

+ve edge triggered D - flipflops



when C = 0

$P_1 = 1$ $P_2 = 1$ $\Rightarrow$ $Q_n = Q_{n-1}$ $\bar{Q}_n = \bar{Q}_{n-1}$

when C = 1

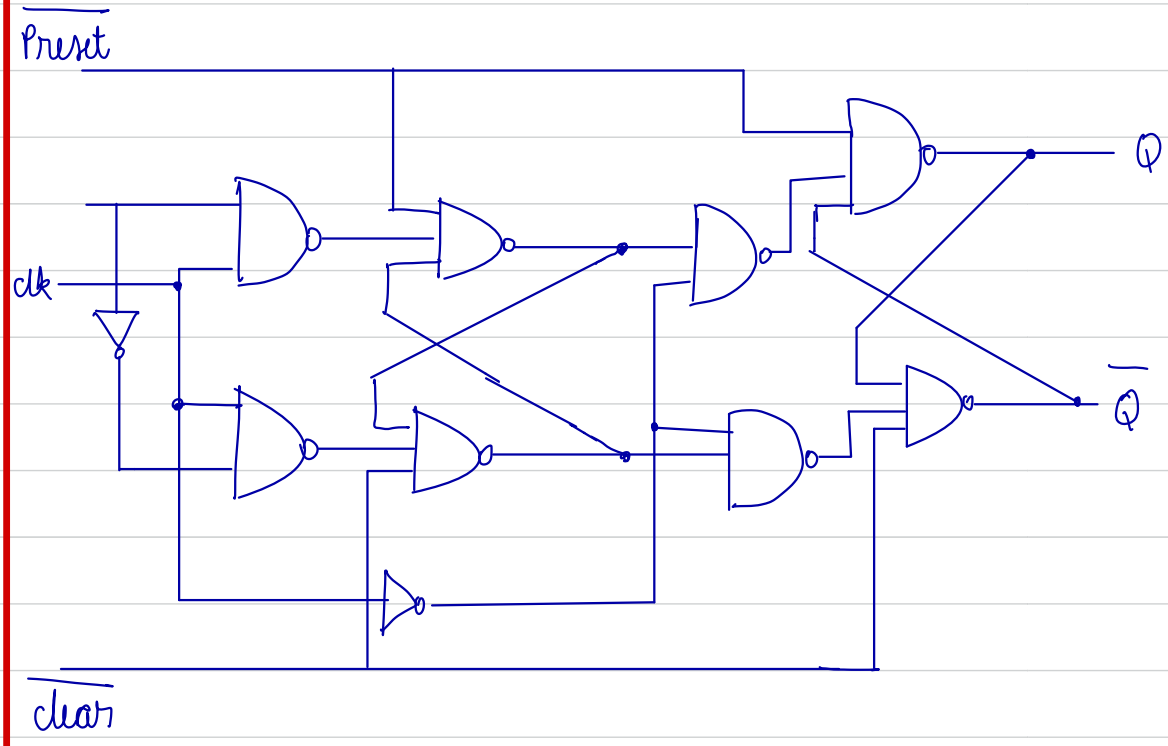
- If $P_3 = 0$: $P_1 = 1$ $P_2 = 0$, $P_4 = 1$
- If $P_3 = 1$: $P_1 = 0$ $P_2 = 1$

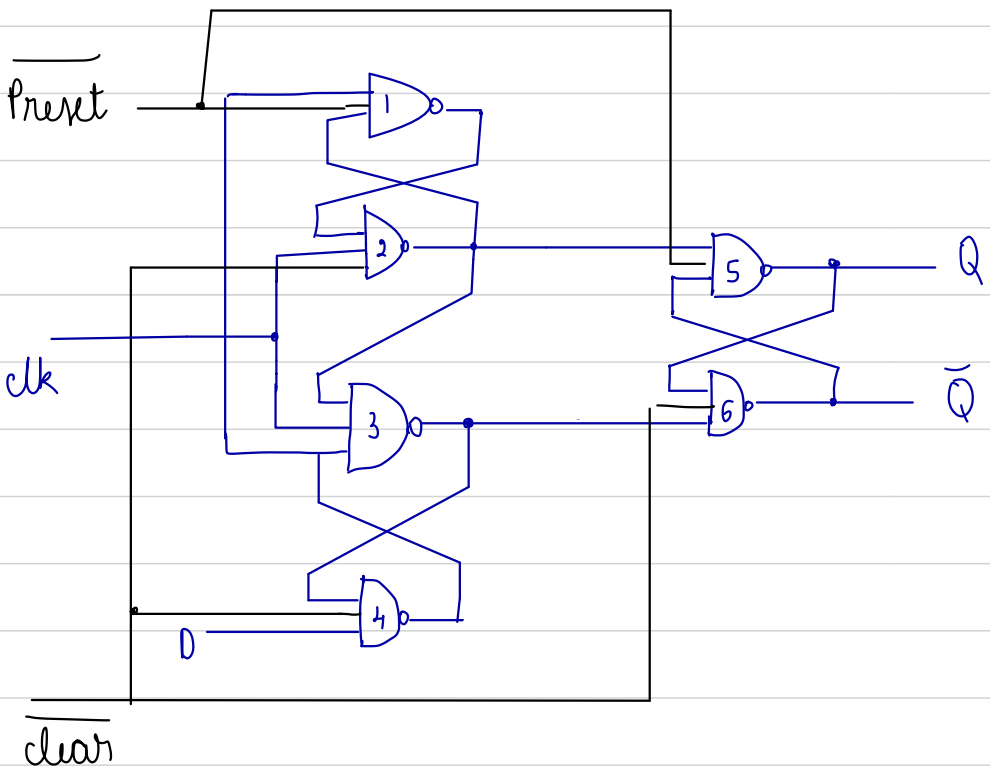
Function table :

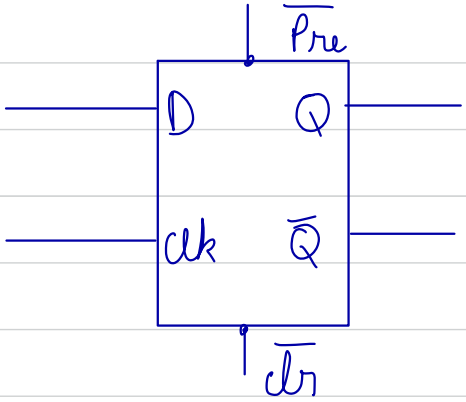
		i/p		o/p	
	gate	C = 0 D = 0		C = 1 D = 0	
O/P :	1	0		0	
	2	1		1	
	3	1		0	
	4	1		1	
	5	Q_{n-1}		0	
	6	$\bar{Q}_{n-1}$		1	

gate	C=0 D=1	C=1 D=1
1	1	0
2	1	1
3	1	1
4	0	0
5	Q_{n-1}	1
6	$\overline{Q}_{n-1}$	0

Asynchronous preset and clear




Preset
clk
D
clear
1
2
3
4
5
6
Q
$\bar{Q}$



Function table

i/p				o/p	
$\overline{Pre}$	$\overline{clr}$	D	clk	Q	$\overline{Q}$
0	1	x	x	1	0
1	0	x	x	0	1
1	1	0	↑	0	1
1	1	1	↑	1	0

Registers:- collection of flipflops capable of storing and modifying multiple data bits

shift register: A register capable of shifting its content with clock

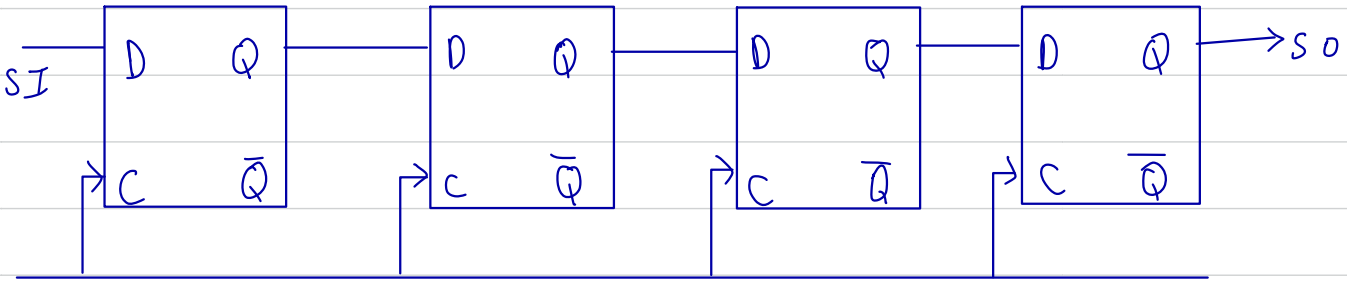
4 bit register: 

types: Serial in - serial out - SISO
Serial in - Parallel out - SIPO

3) Parallel in serial out - PISO

4) Parallel in parallel out - PIPO

SISO unidirection shift register (4 bit)

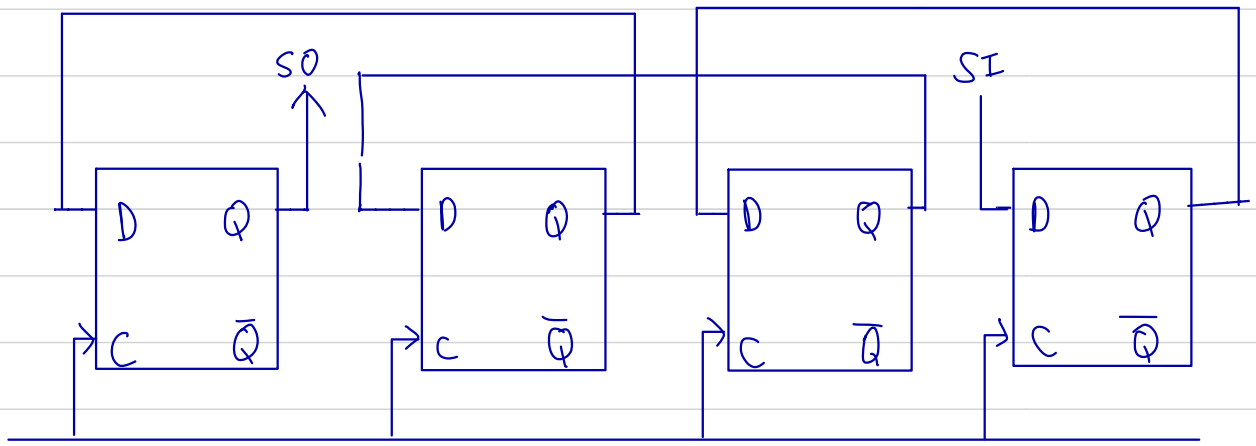


Serial input - SI SO - Serial output

Right shift ↑

SI = 0110

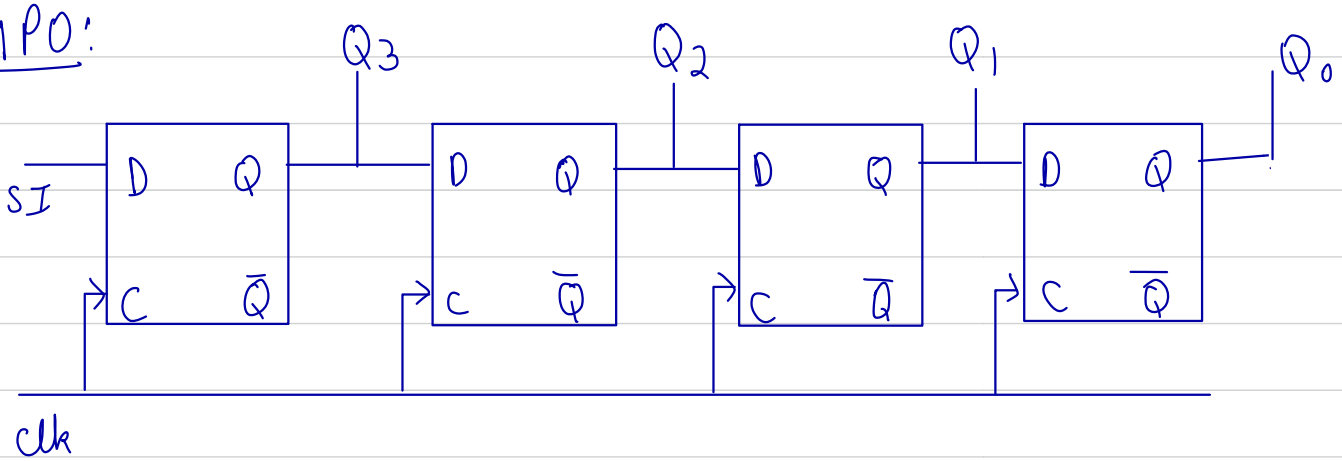
SO = 0110000 (7 clock pulses)



Left shift

↑ 0
↑ 0
↑ 0
↑ 0
↑ 1
↑ 1
↑ 0

2> SIPO:

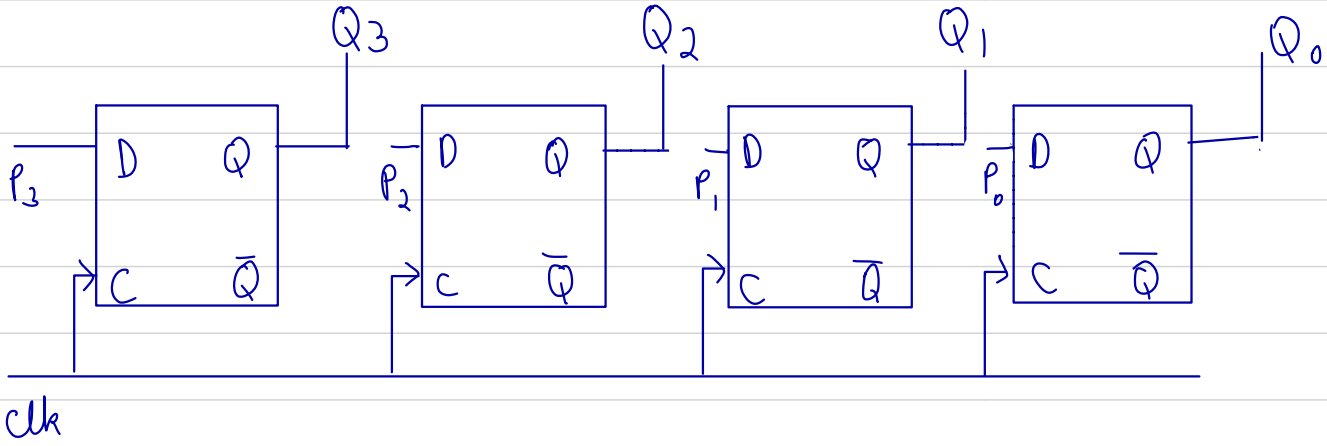


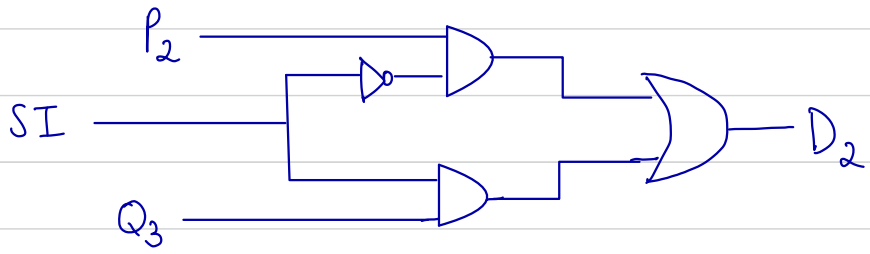
Data i/p & o/p (i/p - 0110)

clk

	Q_3	Q_2	Q_1	Q_0
	0	0	0	0
↑	0	0	0	0
↑	1	0	0	0
↑	1	1	0	0
↑	0	1	1	0

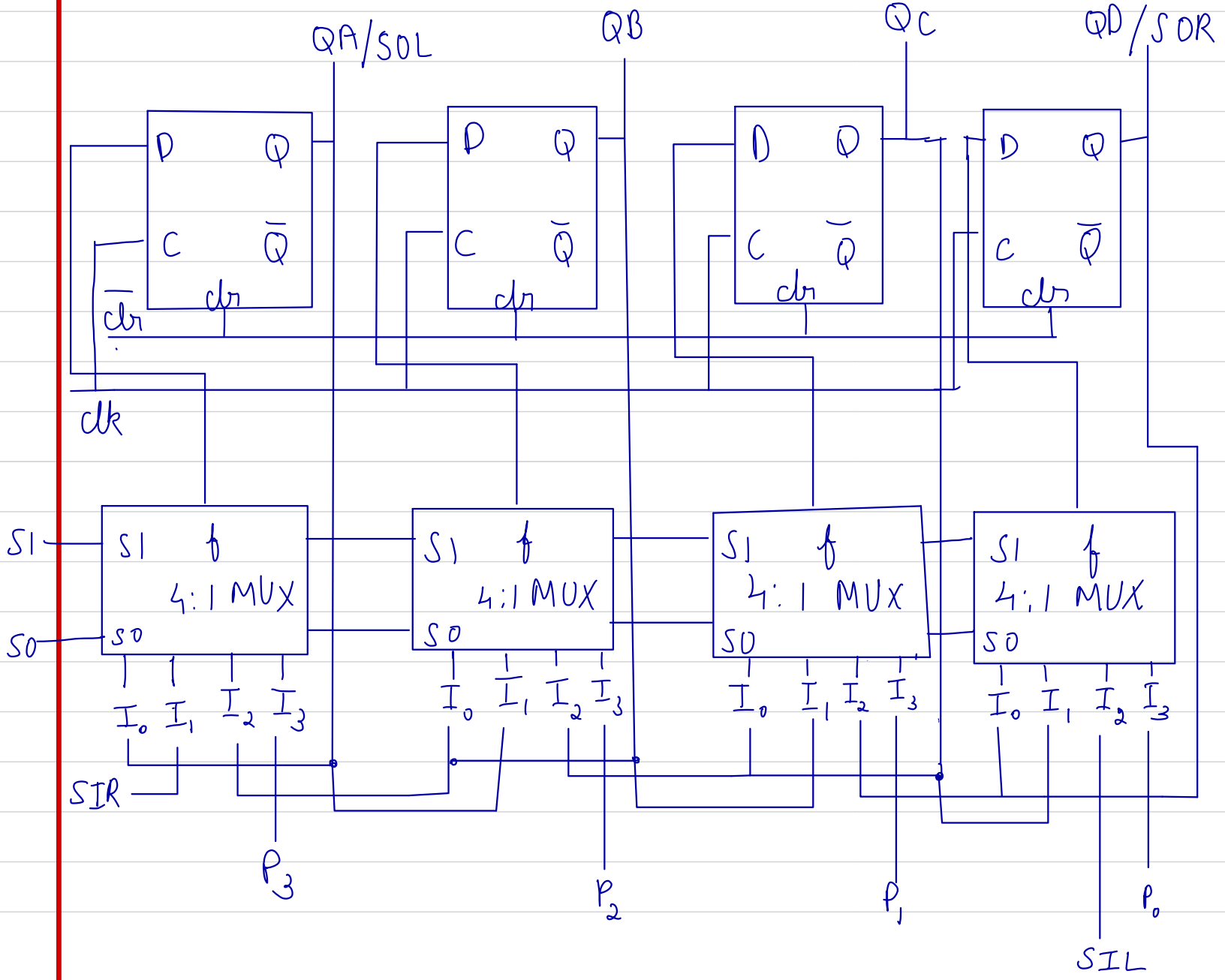
3> PIPO:




P_2
SI
Q_3
D_2

Universal bidirectional shift registers

select lines		Reg operation
S_0	S_1	
0	0	hold
0	1	shift right
1	0	shift left
1	1	parallel load

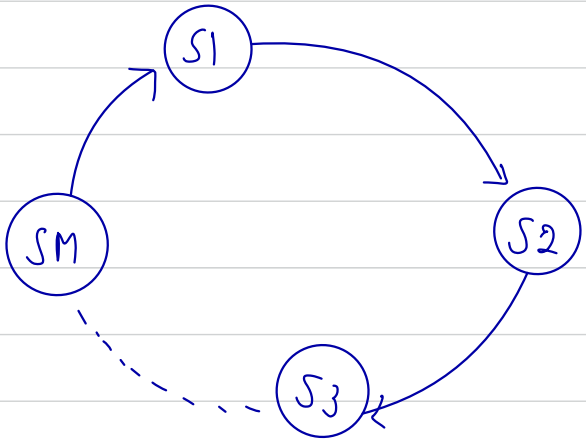


Counters: (Pattern generators)

Simple counters: Increment / decrement count with the application of clock

Module 'M' counter ⇒ has 'M' distinct states / counts

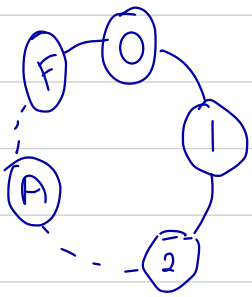
State diagram:



Binary Ripple counter

count sequence in Hexadecimal:

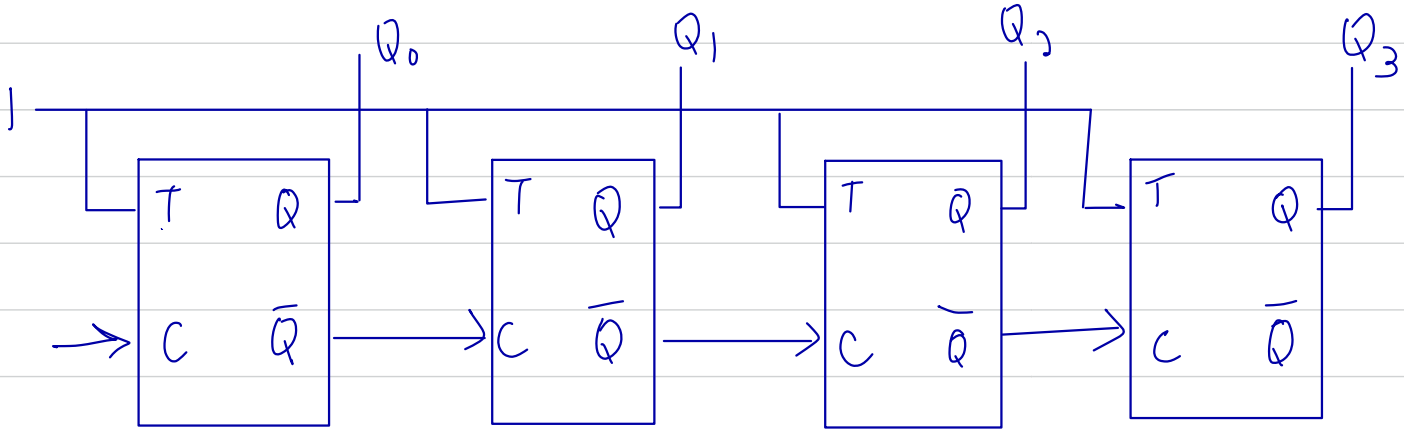
0 → 1 → 2 → 3 → ..... 8 → 9 → A → B → ... → F → 0



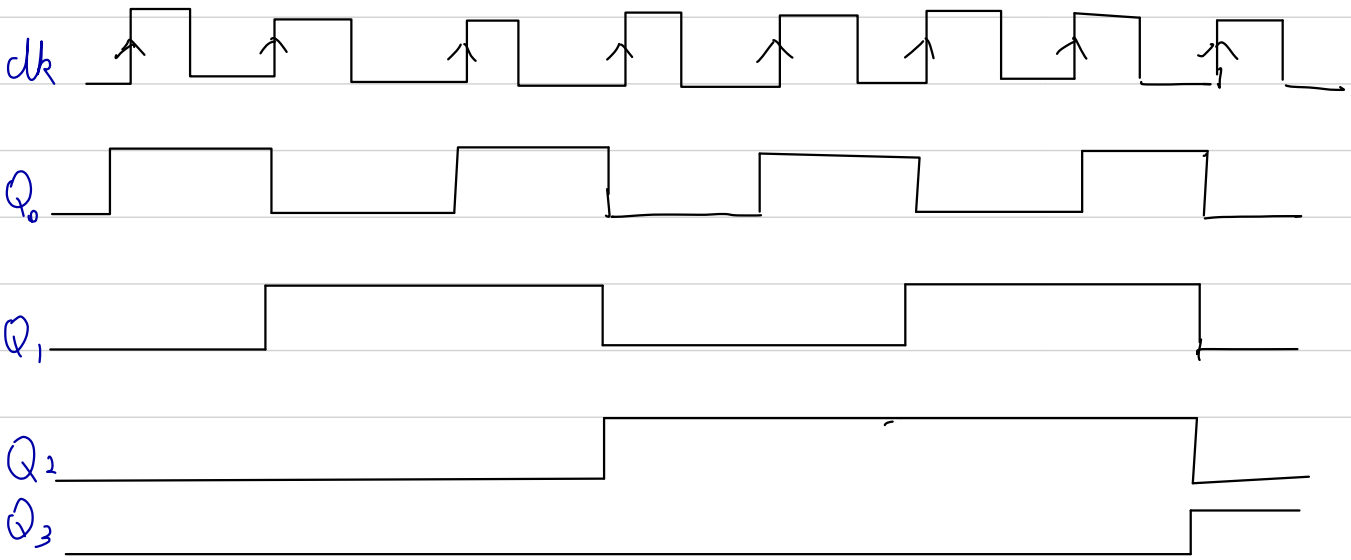
T flipflops are used

count sequence in binary:

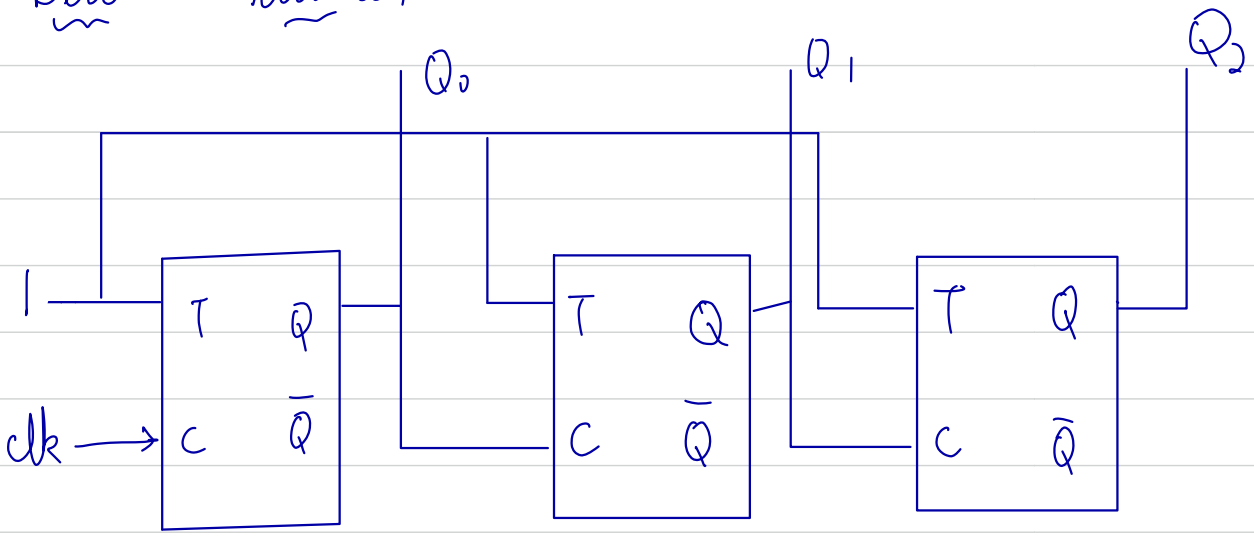
clk	Q_3	Q_2	Q_1	Q_0
↑	0	0	0	0
↑	0	0	0	1
↑	0	0	1	0
↑	0	0	1	1
↑	0	1	0	0
↑	0	1	0	1
↑	0	1	1	0
↑	0	1	1	1



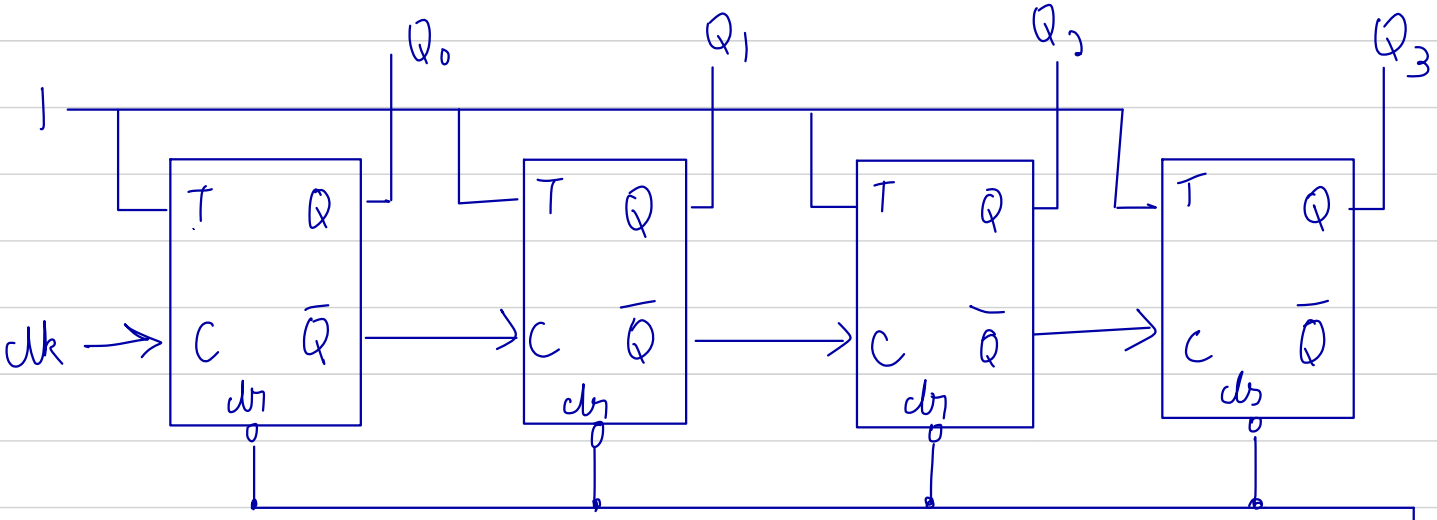
Timing diagram:



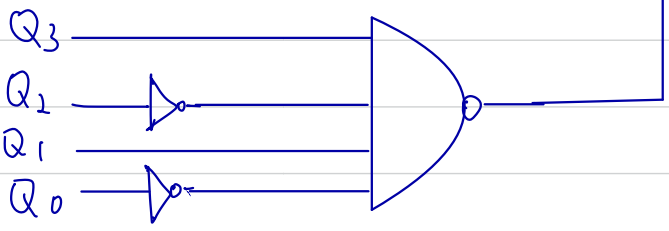
Down counter



Decade counter ; (use of preset & clear inputs to control start & end count)



clear when
$Q_3 Q_2 Q_1 Q_0 = 1010$



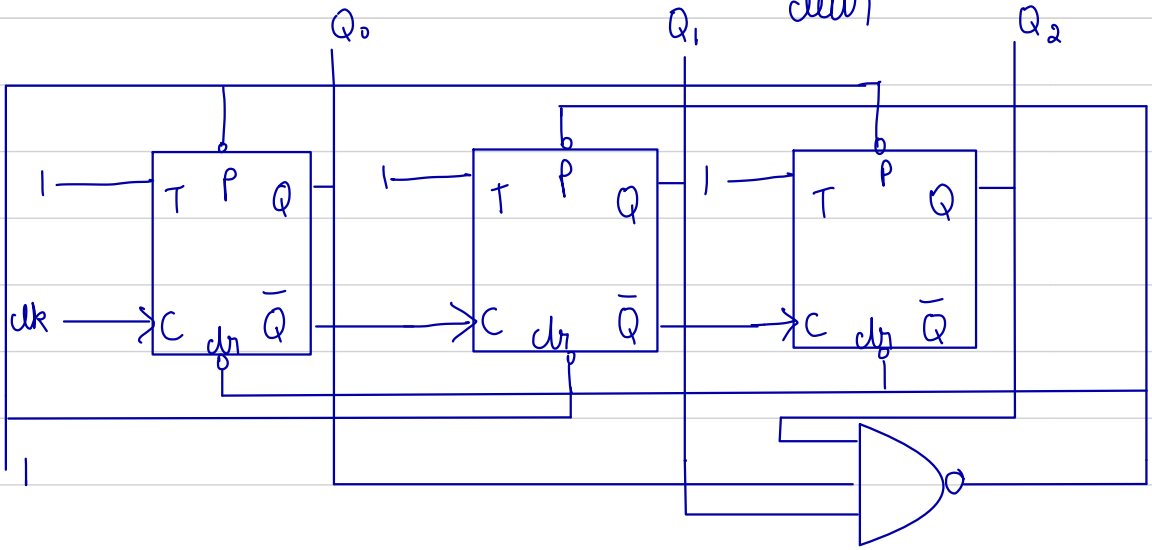
Ripple counter to count from 2 to 6

when count = 7 → preset to 2

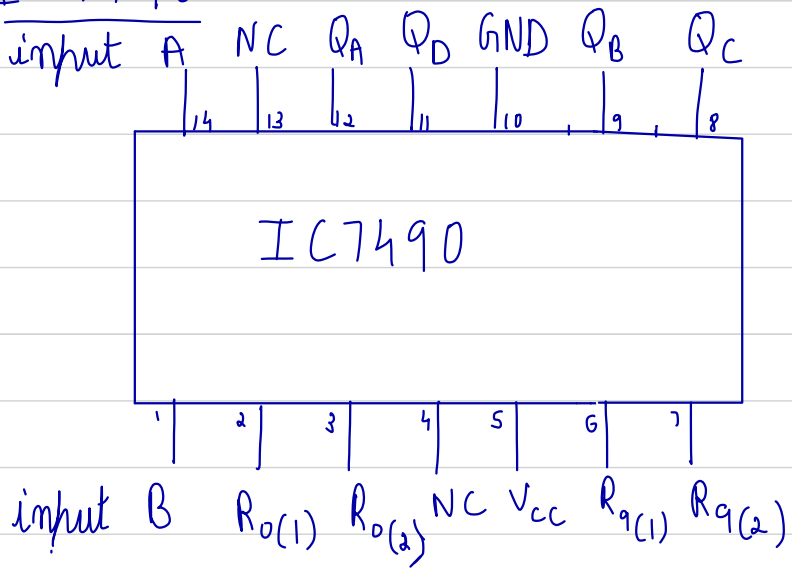
$Q_2 Q_1 Q_0 = 111 \rightarrow$ 0 1 0

Preset

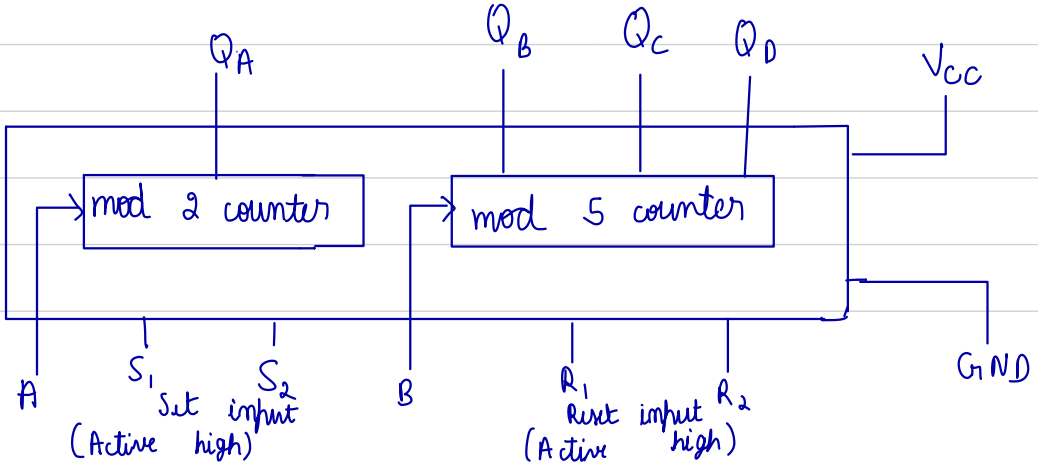
clear



IC7490:



Block diagram



Reset / count function table :

Reset inputs				outputs			
$R_0(1)$	$R_0(2)$	$R_9(1)$	$R_9(2)$	Q_D	Q_C	Q_B	Q_A
1	1	0	x	0	0	0	0
1	1	x	0	0	0	0	0
x	x	1	1	1	0	0	1
x	0	x	0	count			
0	x	0	x	count			
0	x	x	0	count			
x	0	0	x	count			

Next state table & characteristic equation

D flipflop

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

$Q^+ = D$

T flipflop

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

$Q^+ = T \oplus Q$

JK flipflop

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

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

SR flipflop

S	R	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	X
1	1	1	X

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

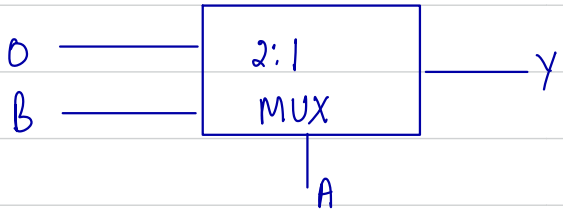
Tutorial 3

1) How many 2:1 MUX are required to implement
   i) AND ii) OR iii) XOR

Ans

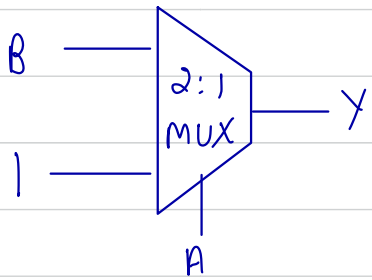
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

i) AND



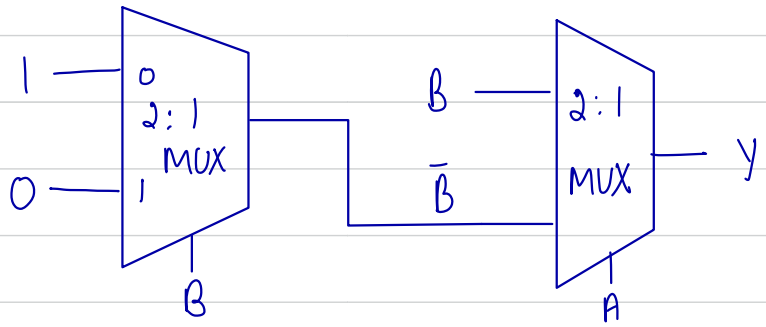
ii) OR:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

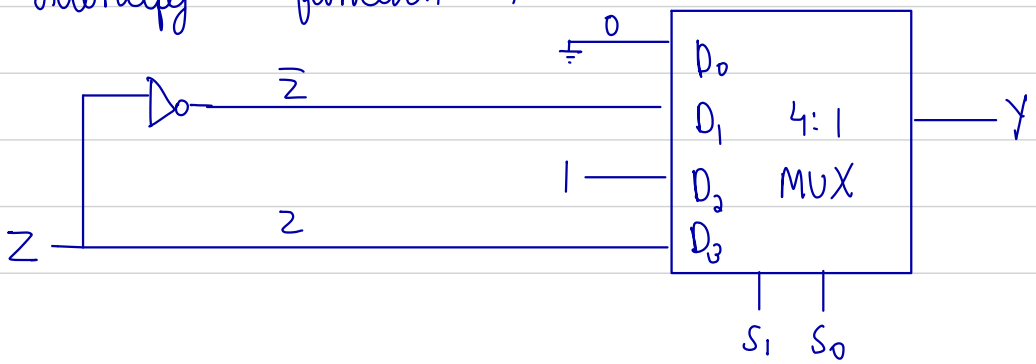


iii)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0



2) Identify function Y:



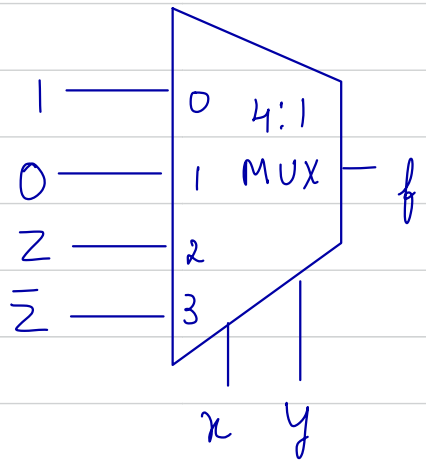
$$Y = \bar{S}_1 S_0 \bar{Z} + S_1 \bar{S}_0 + S_1 S_0 Z$$
$$= S_0 (\bar{S}_1 \bar{Z} + S_1 Z_1) + S_1 \bar{S}_0$$
$$= S_0 (S_1 \odot Z_1) + S_1 \bar{S}_0$$

or $Y = \Sigma m(2, 4, 5, 7)$

3> Implement $f(x, y, z) = \Pi M(2, 3, 4, 7)$ using 4:1 MUX

Ans

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

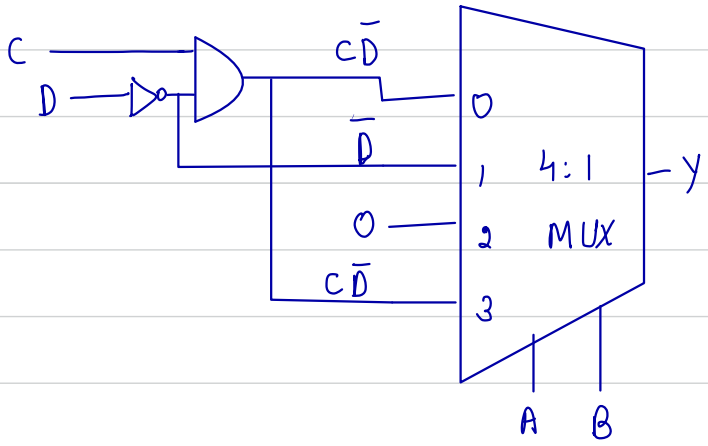


4> Implement $Y(A, B, C, D) = \Sigma(2, 4, 6, 14)$ using 4:1 MUX and gates

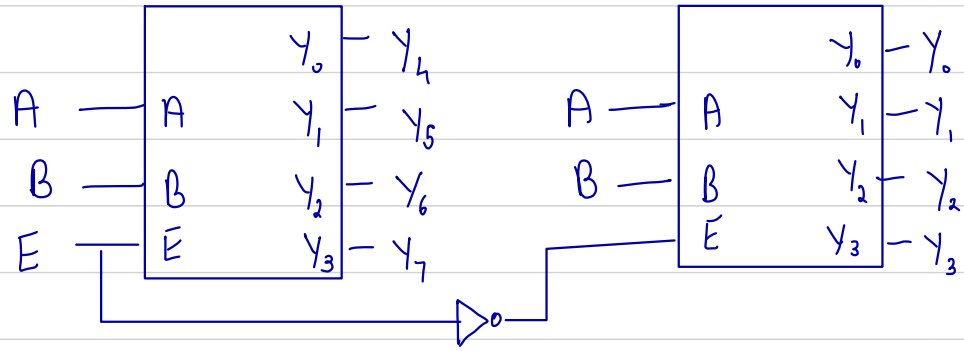
Ans

A	B	C	D	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0

A	B	C	D	Y
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

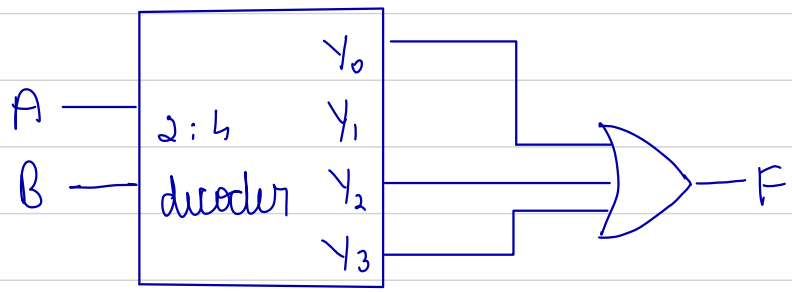


5> Implement 3:8 decoder using 2:4 decoder

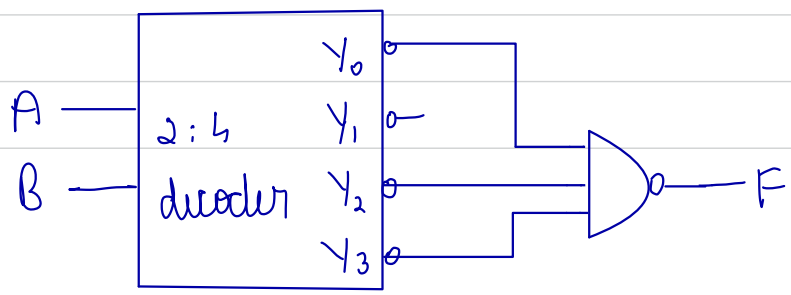


6> Implement $F(A,B) = \Sigma m(0,2,3)$ using a decoder with i> active high output
ii> active low output

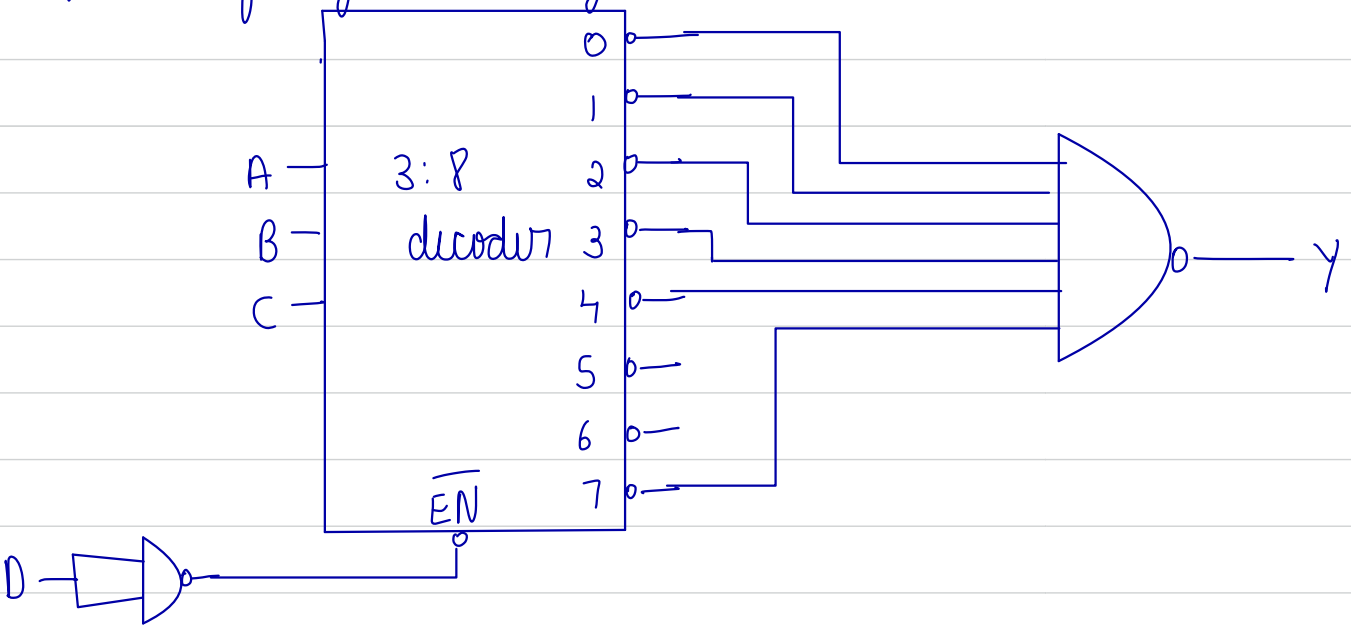
Ans i>



ii>



7) Output of given logic circuit is :



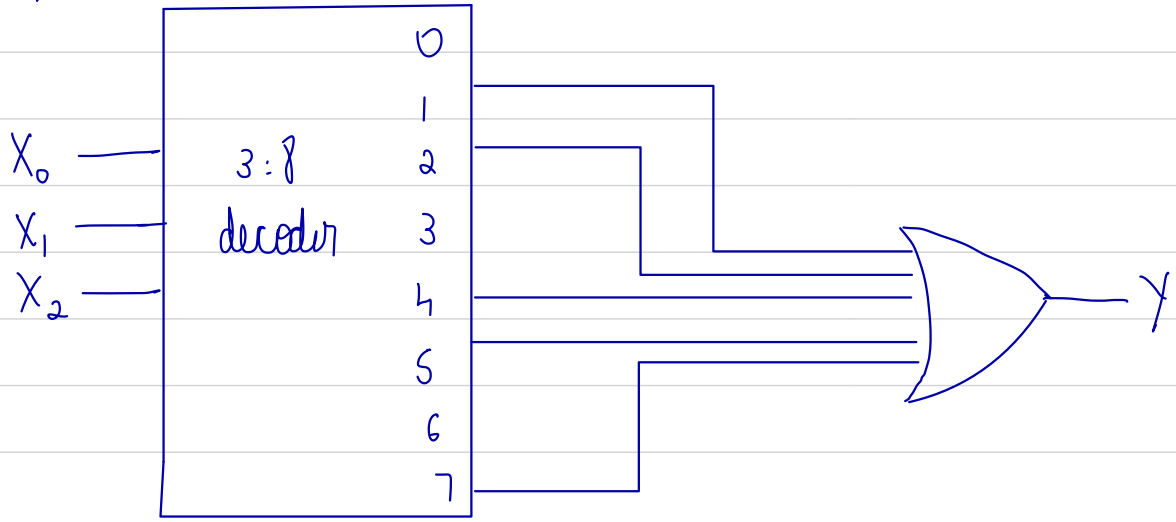
Ans $Y = \left(\sum m(0, 1, 2, 4, 7)\right) \cdot D$

	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	0 : 1	1 : 1	3 : 0	2 : 1
A	4 : 1	5 : 0	7 : (1)	6 : 0

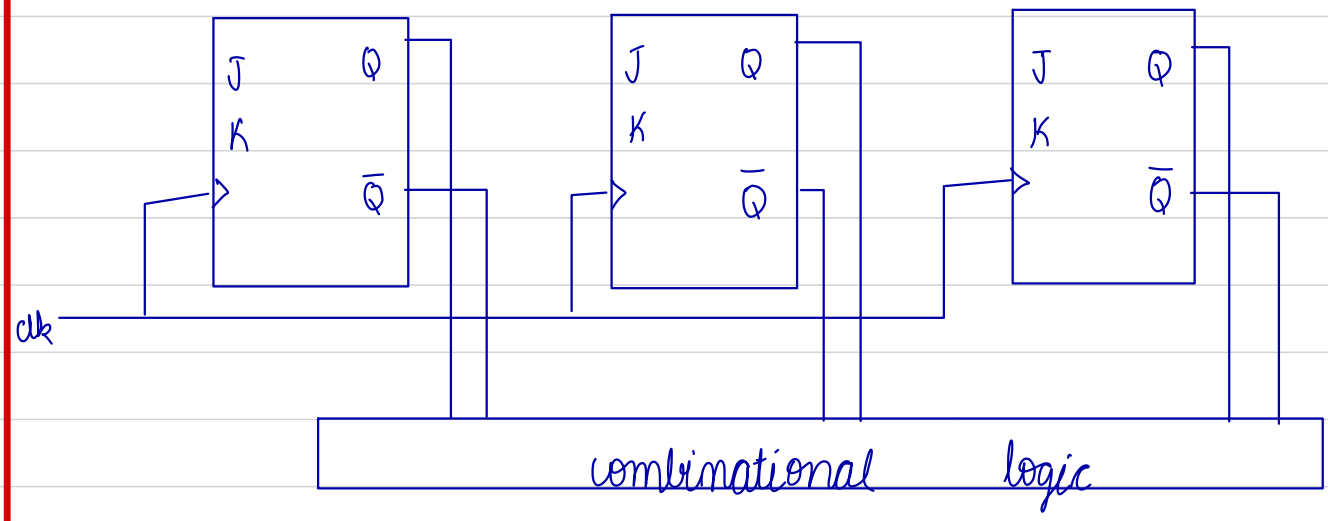
$Y = (\bar{A}\bar{B} + \bar{A}\bar{C} + \bar{B}\bar{C} + ABC) \cdot D$

8) Implement $Y(X_0, X_1, X_2) = \sum(1, 2, 4, 5, 7)$ using active high o/p 3:8 decoder

Ans



Synchronous counter design



Application table

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

D

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

T

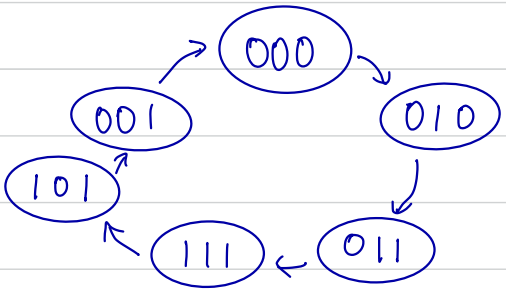
JK:

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

SR:

Q	Q^+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

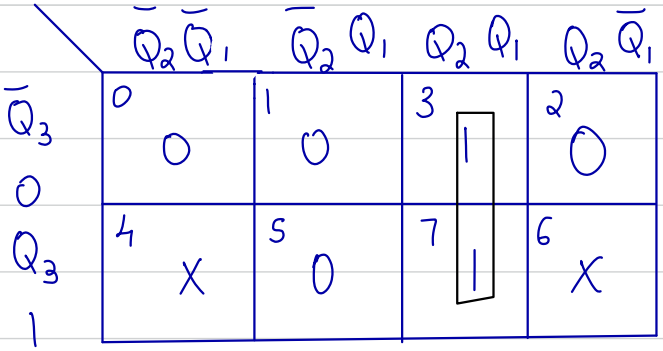
Design mod 6 Gray code counter (synchronous)



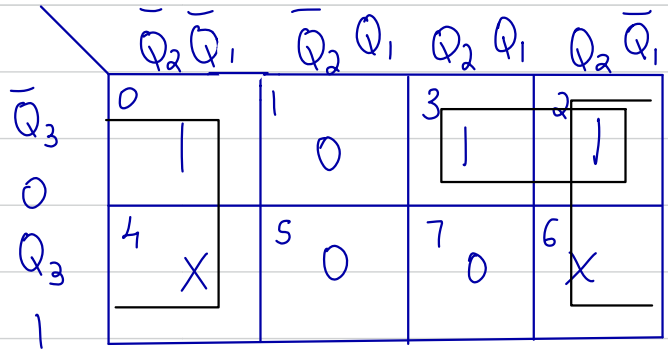
0 → 2 → 3 → 7 → 5 → 1 → 0

Excitation table for DFF

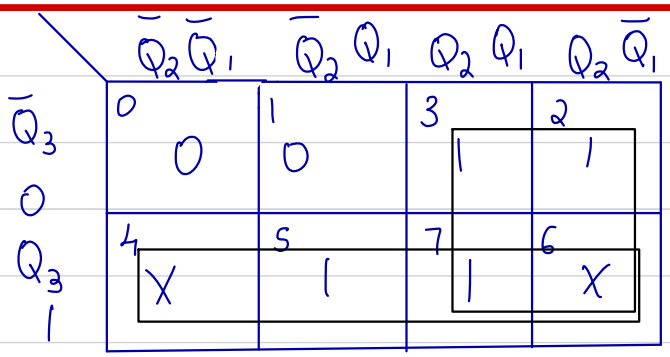
Present state			Next state			flipflop inputs		
Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+	D_3	D_2	D_1
0	0	0	0	1	0	0	1	0
0	1	0	0	1	1	0	1	1
0	1	1	1	1	1	1	1	1
1	1	1	1	0	1	1	0	1
1	0	1	0	0	1	0	0	1
0	0	1	0	0	0	0	0	0



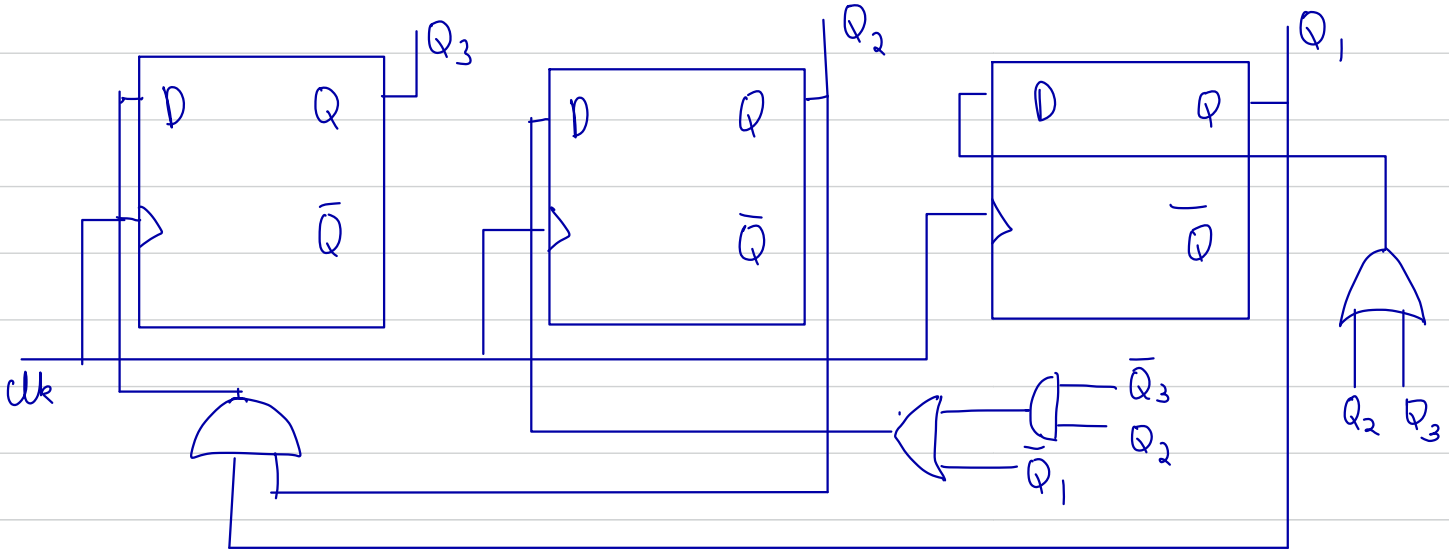
$D_3 = Q_2 Q_1$



$D_2 = \bar{Q}_1 + \bar{Q}_3 Q_2$

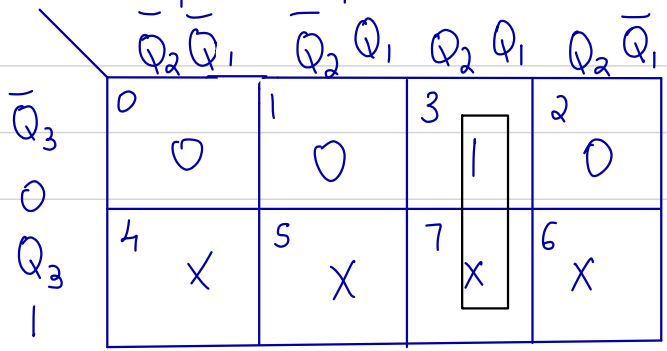


$$D_1 = Q_2 + Q_3$$



Using JK flipflop

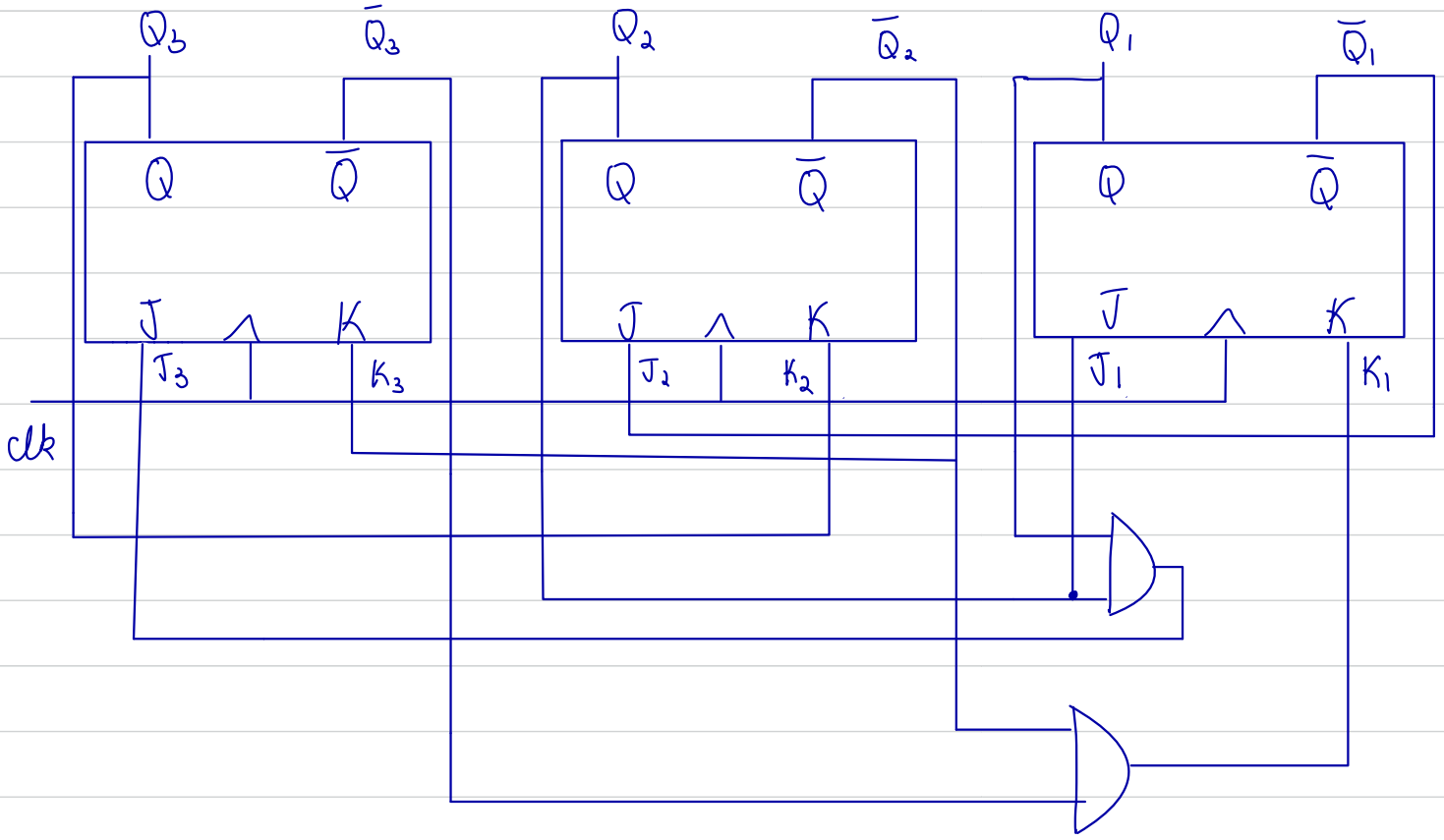
Present state			Next state			flipflop inputs					
Q_3	Q_2	Q_1	Q_3^+	Q_2^+	Q_1^+	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	1	0	0	X	1	X	0	X
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	1	1	1	X	X	0	X	0
1	1	1	1	0	1	X	0	X	1	X	0
1	0	1	0	0	1	X	1	0	X	X	0
0	0	1	0	0	0	0	X	0	X	X	1



$$J_3 = Q_1 Q_2$$

$K_3 = \bar{Q}_2$ $\quad J_2 = \bar{Q}_1$ $\quad K_2 = Q_3$ $\quad J_1 = Q_2$

$K_1 = \bar{Q}_3 \bar{Q}_2$

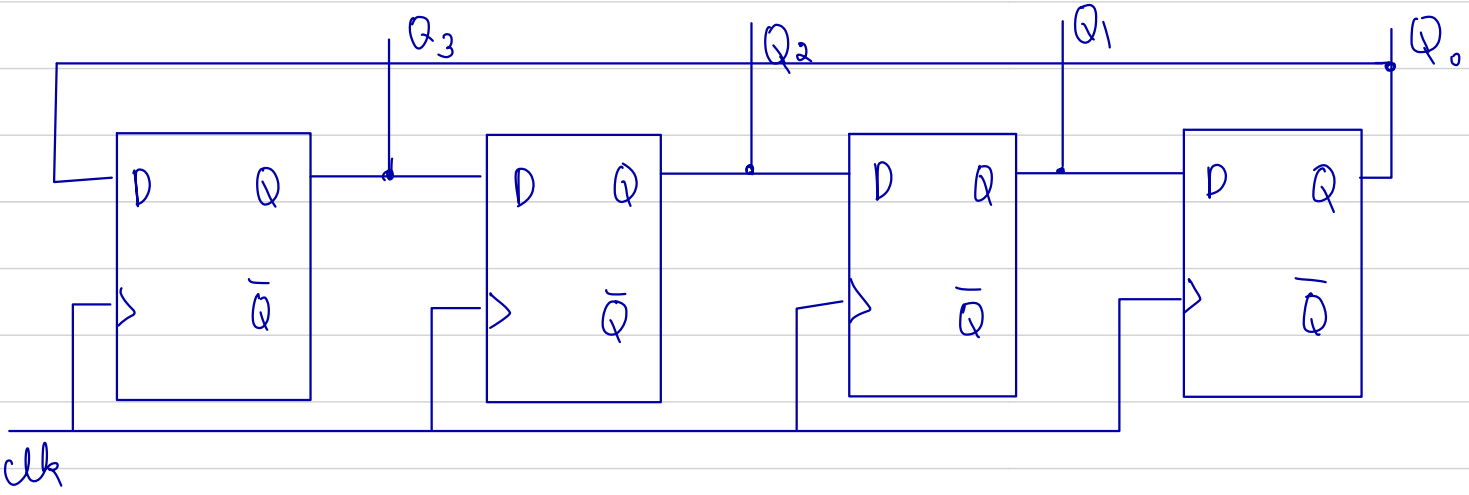


Counter design using shift register

non binary counters: — Ring counters
                     — Switch tail counter

Ring counter → N - bit counter
only one bit is set at a given count.

Hence n bit shift register ⇒ N distinct count.

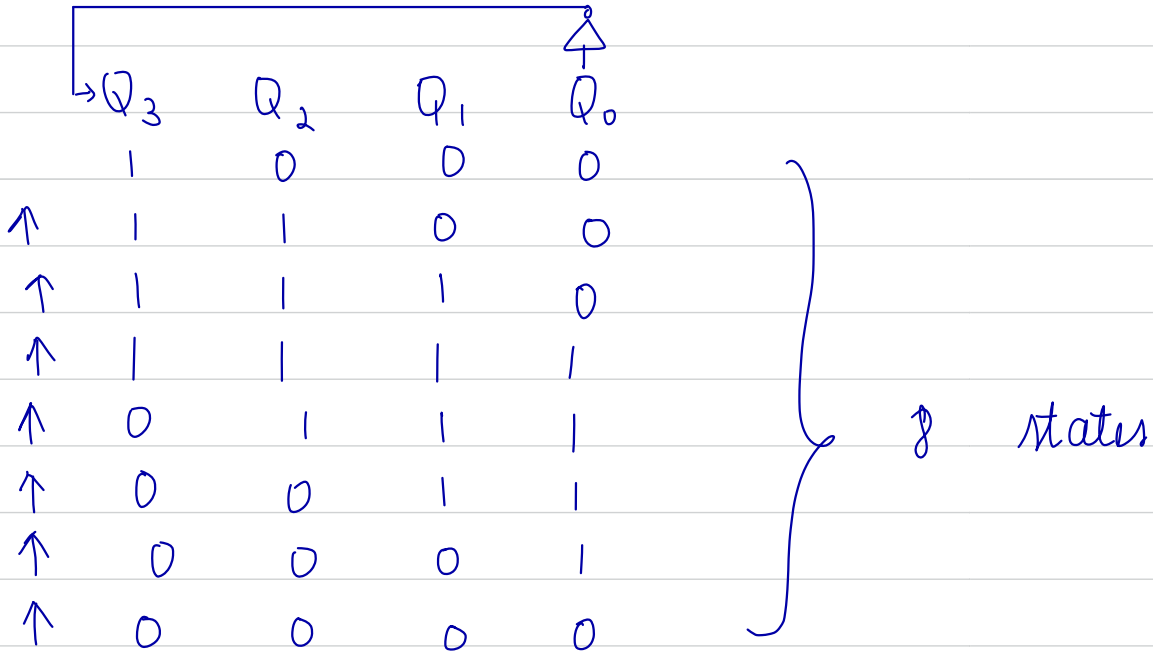


It needs initialisation before counting

	Q_3	Q_2	Q_1	Q_0	
	0	0	0	1	
↑	1	0	0	0	4 counts
↑	0	1	0	0	
↑	0	0	1	0	
↑	0	0	0	1	

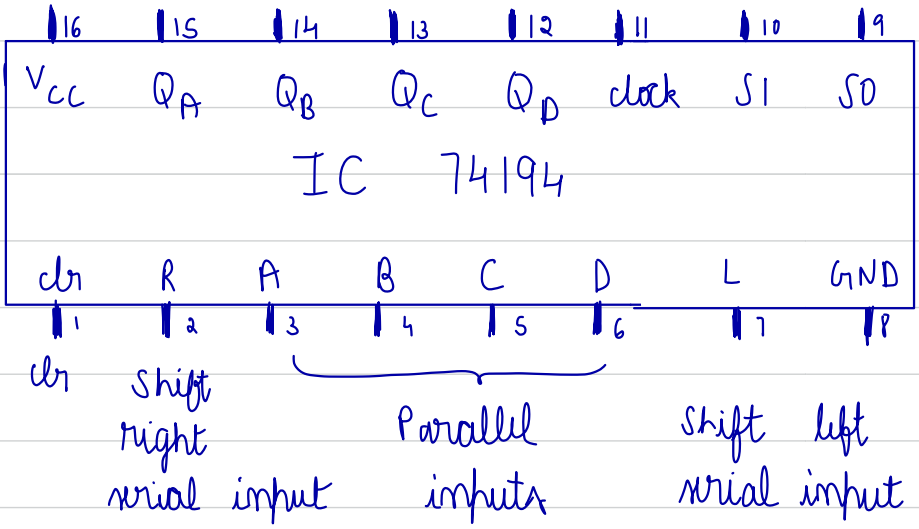
Johnson counter:
If we give input to 1st flipflop from $\bar{Q}$
we get $2n$ states for n flipflops

i. e



Universal shift register

IC74194:



Function table

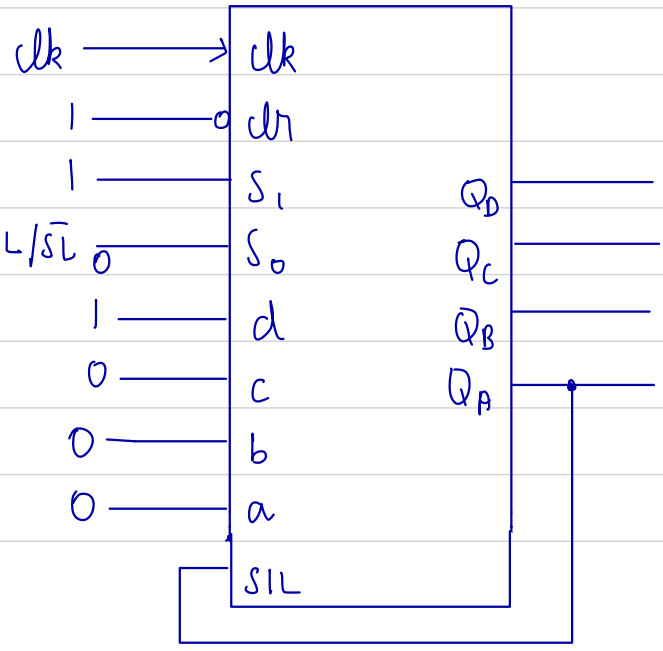
Inputs											Outputs			
clear	Mode		clock	Serial		Parallel					Q_A	Q_B	Q_C	Q_D
	S_1	S_0		L	R	A	B	C	D					
L	X	X	X	X	X	X	X	X	X		L	L	L	L
H	X	X	↓	X	X	X	X	X	X		Q_{A_0}	Q_{B_0}	Q_{C_0}	Q_{D_0}
H	H	H	↑	X	X	a	b	c	d		a	b	c	d
H	L	H	↑	X	H	X	X	X	X		H	Q_{An}	Q_{Bn}	Q_{Cn}
H	L	H	↑	X	L	X	X	X	X		L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	L	↑	H	X	X	X	X	X		Q_{Bn}	Q_{Cn}	Q_{Dn}	H
H	H	L	↑	L	X	X	X	X	X		Q_{Bn}	Q_{Cn}	Q_{Dn}	L
H	L	L	X	X	X	X	X	X	X		Q_{A_0}	Q_{B_0}	Q_{C_0}	Q_{D_0}

$S_1S_0 = 00$ → hold

$S_1S_0 = 01$ → Shift right

$S_1S_0 = 10$ → Shift left

$S_1S_0 = 11$ → Parallel load



clk	Q_A	Q_B	Q_C	Q_D
↑	0	0	0	1
↑	0	0	1	0
↑	0	1	0	0
↑	1	0	0	0
↑	0	0	0	1

Arbitrary counter using shift registers

Based on present state choose SIL, SIR, S1, S0 which produce required next states

Ex: count sequence: $0 \to 8 \to 12 \to 6 \to 13 \to 11 \to 7 \to 3 \to 1 \to 0$

	Present state				Next state				Shift reg inputs			
	Q_A	Q_B	Q_C	Q_D	Q_A^+	Q_B^+	Q_C^+	Q_D^+	SIL	SIR	S1	S0
0	0	0	0	0	1	0	0	0	0	1	0	1
8	1	0	0	0	1	1	0	0	0	1	0	1
12	1	1	0	0	0	1	1	0	0	0	0	1
6	0	1	1	0	1	1	0	1	1	0	1	0
13	1	1	0	1	1	0	1	1	1	0	1	0
11	1	0	1	1	0	1	1	1	1	0	1	0
7	0	1	1	1	0	0	1	1	0	0	0	1
3	0	0	1	1	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	0	1

$$S_1 = \bar{Q}_D Q_C Q_B \bar{Q}_A + Q_D Q_A (Q_C \oplus Q_B)$$

$$S_0 = \bar{Q}_C \bar{Q}_D \bar{Q}_A + \bar{Q}_D Q_B Q_A + Q_D \bar{Q}_B \bar{Q}_A$$

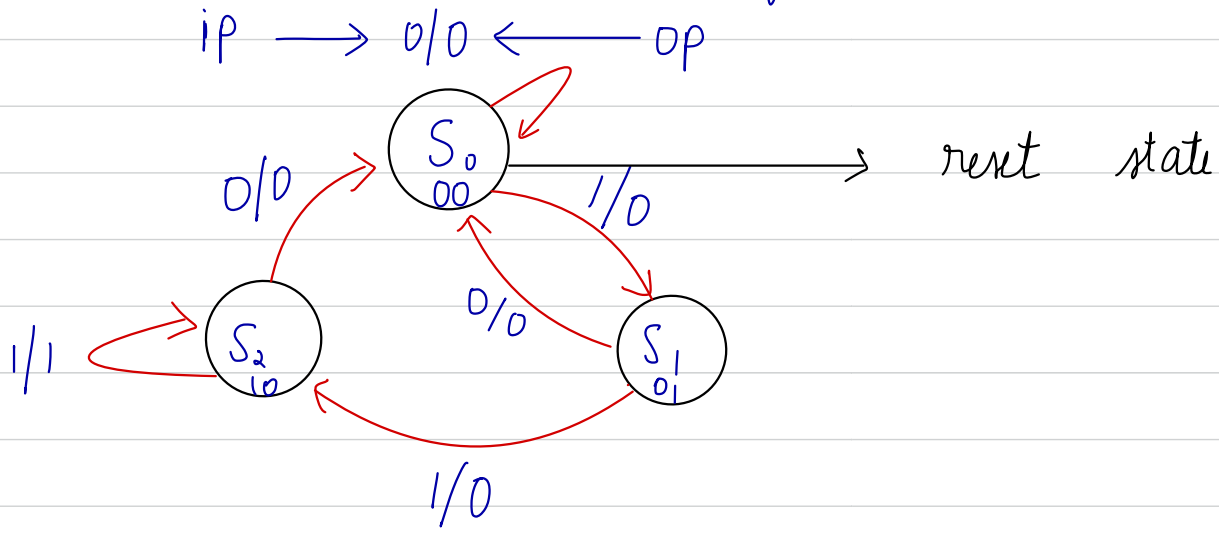
$$SIL = S1$$

$$SIR = \bar{Q}_C \bar{Q}_B \bar{Q}_D$$

Sequence detector

Detect the occurance of a pattern in the input sequence:

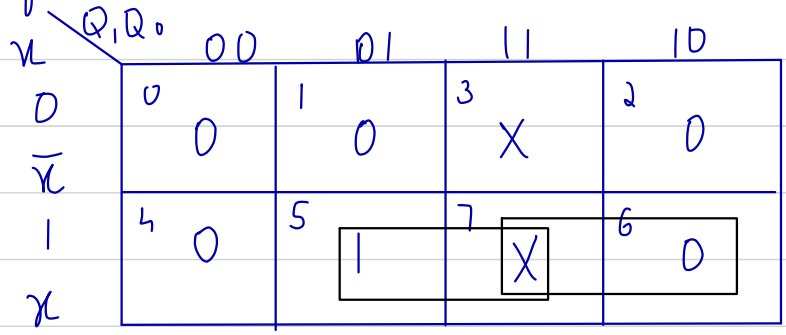
Detect 111 in input of 0's & 1's, example:
0110110100011111101 ..... using DFF



Function table

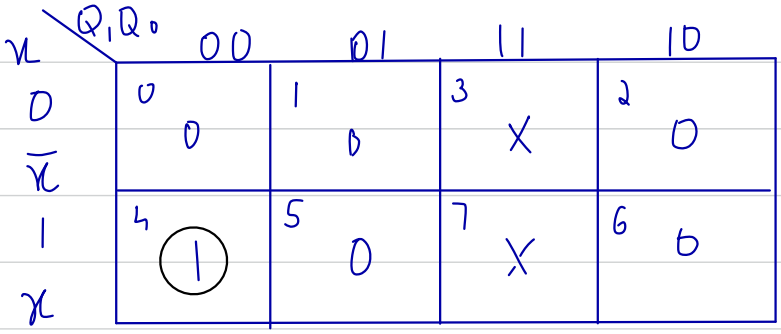
Q_1Q_0	x	$Q_1^+Q_0^+$			
Present state	i/p	Next state	output (z)	D_1	D_0
S_0 (00)	0	S_0 (00)	0	0	0
S_0 (00)	1	S_1 (01)	0	0	1
S_1 (01)	0	S_0 (00)	0	0	0
S_1 (01)	1	S_2 (10)	0	1	0
S_2 (10)	0	S_0 (00)	0	0	0
S_2 (10)	1	S_2 (10)	1	1	0

K Map for D1



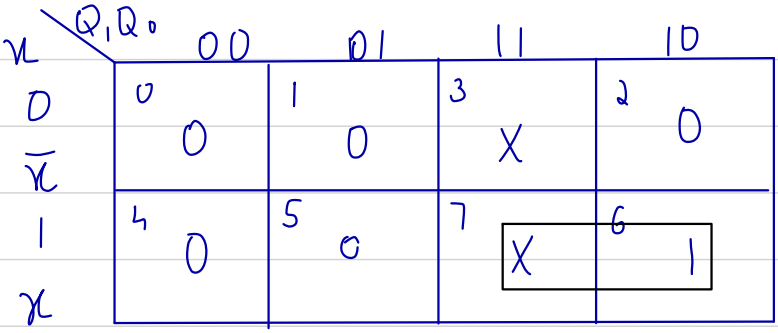
$D_1 = xQ_0 + xQ_1 = x(Q_1 + Q_0)$

K Map for D_0

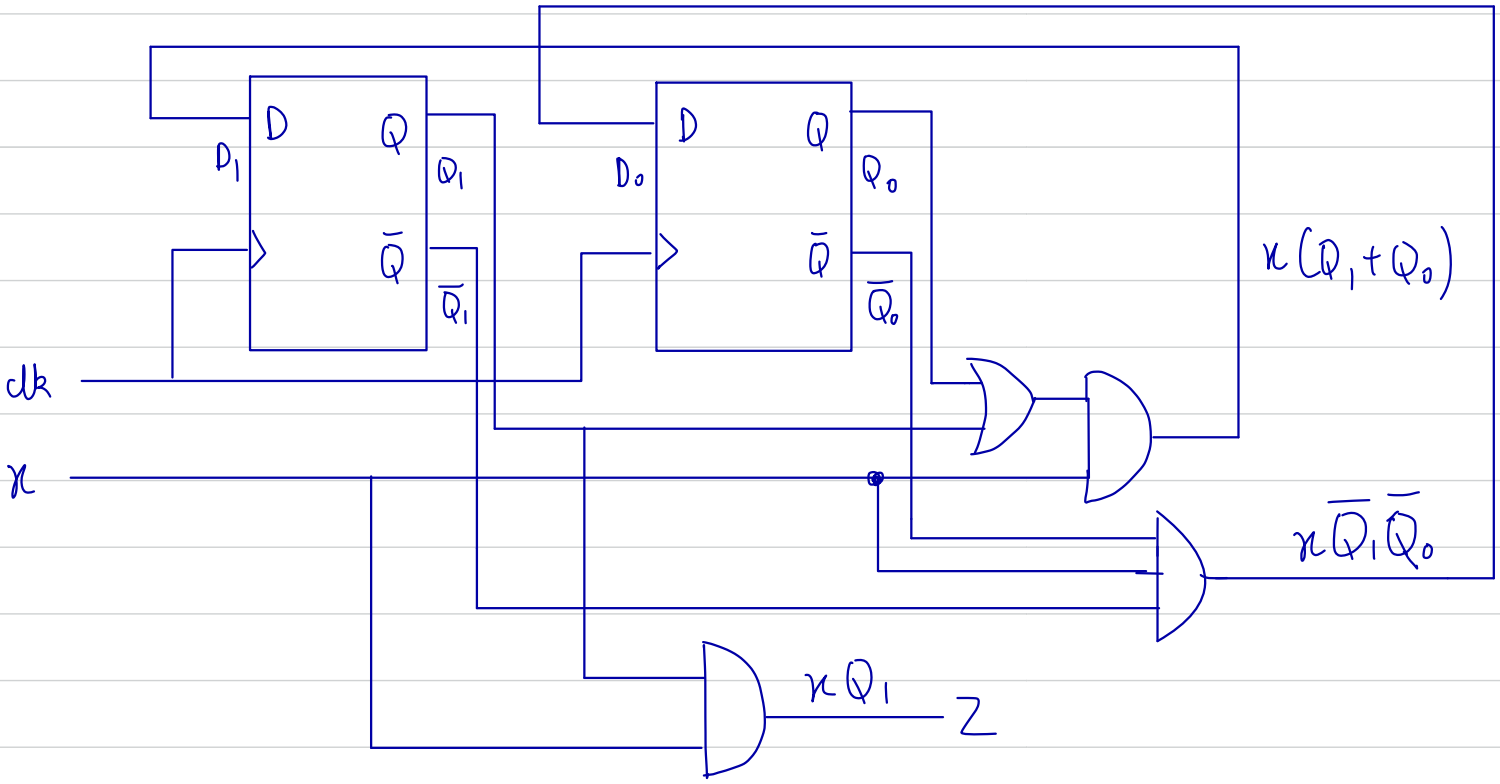


$D_0 = x\bar{Q}_1\bar{Q}_0$

K Map for Z



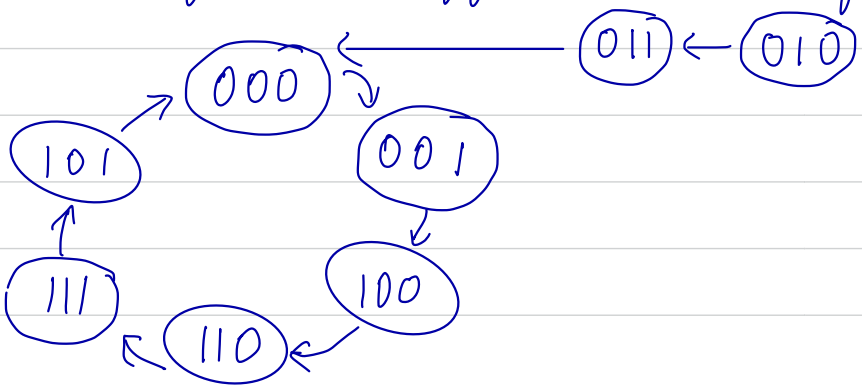
$Z = xQ_1$ //



Self correcting counters

All unused states eventually leading to normal counting sequence after one or more count.

- Design a synchronous mod 6 counter with counting sequence $0 \to 1 \to 4 \to 6 \to 7 \to 5 \to 0$ using positive edge triggered T flipflop.



Excitation Table

Present state			Next state			Flipflop inputs		
Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+	T_2	T_1	T_0
0	0	0	0	0	1	0	0	1
0	0	1	1	0	0	1	0	1
1	0	0	1	1	0	0	1	0
1	1	0	1	1	1	0	0	1
1	1	1	1	0	1	0	1	0
1	0	1	0	0	0	1	0	1

T_2 :

Q_2 \ Q_1Q_0	00	01	11	10
0	0	1	X	X
1	0	1	0	0

$$T_2 = \bar{Q}_1 Q_0$$

T_1 :

Q_2 \ Q_1Q_0	00	01	11	10
0	0	0	X	X
1	1	0	1	0

$$T_1 = Q_1 Q_0 + Q_2 \bar{Q}_1 \bar{Q}_0$$

T_0 :

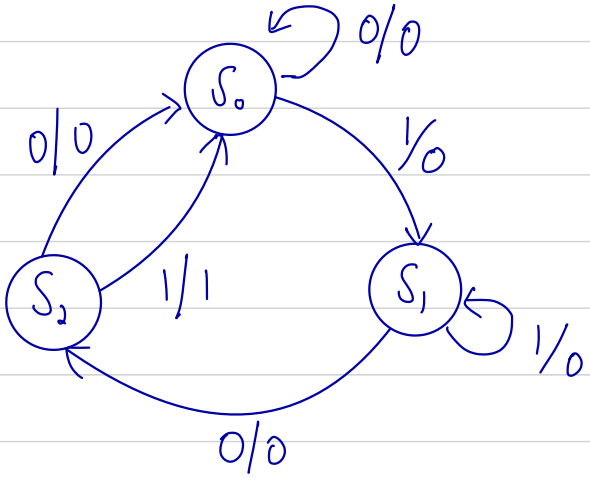
Q_2 \ Q_1Q_0	00	01	11	10
0	1	1	X	X
1	0	1	0	1

$$T_0 = \bar{Q}_2 + Q_1 \bar{Q}_0 + \bar{Q}_1 Q_0$$

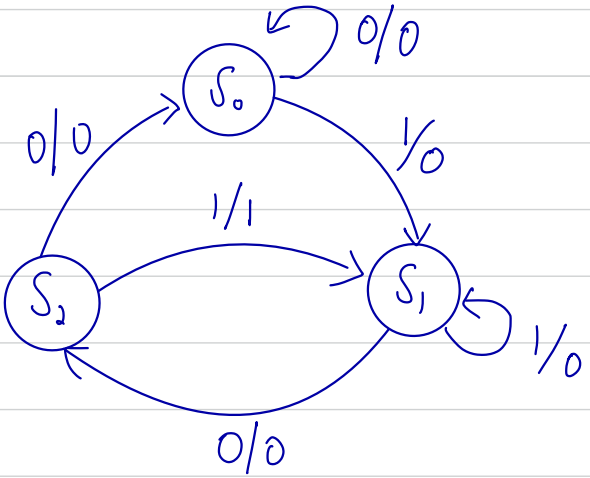
Unused states 010 & 011

Q_2	Q_1	Q_0	T_2	T_1	T_0	Q_2^+	Q_1^+	Q_0^+
0	1	0	0	0	1	0	1	1
0	1	1	0	1	1	0	0	0

Q> Detect 101 without overlap

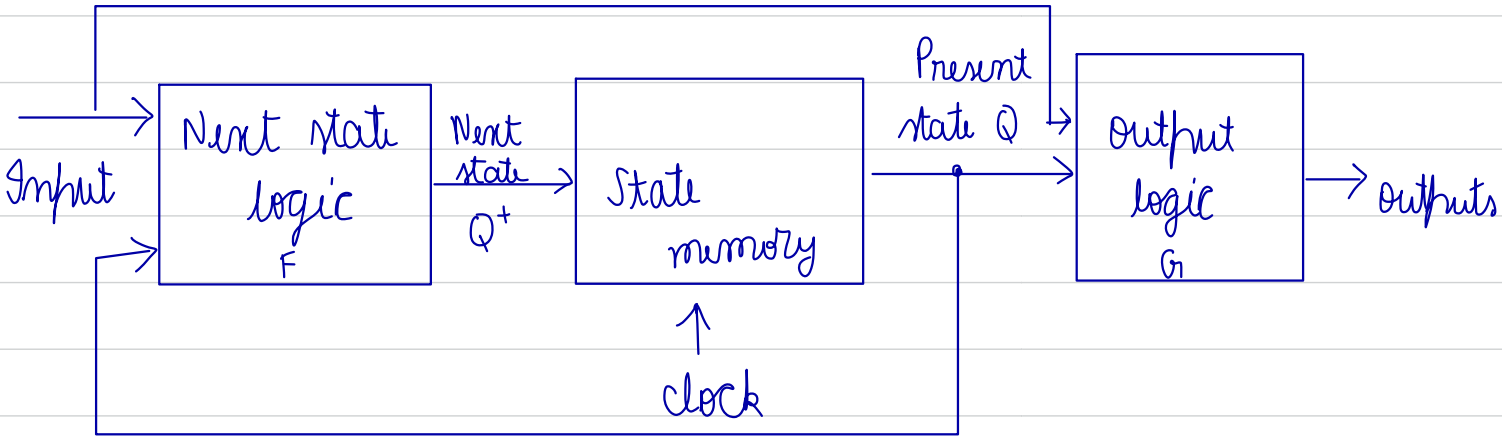


Q> Detect 101 with overlap



Synchronous Sequential Networks

General structure: Mealy machine



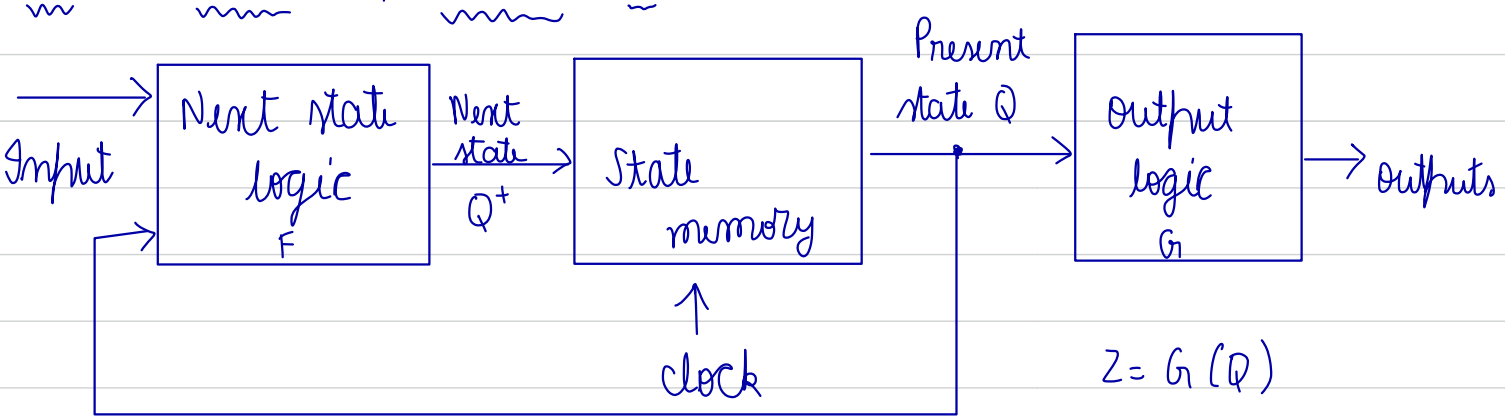
Elements of state machine

State memory - memory element - set of 'N' flipflops - clocked synchronously

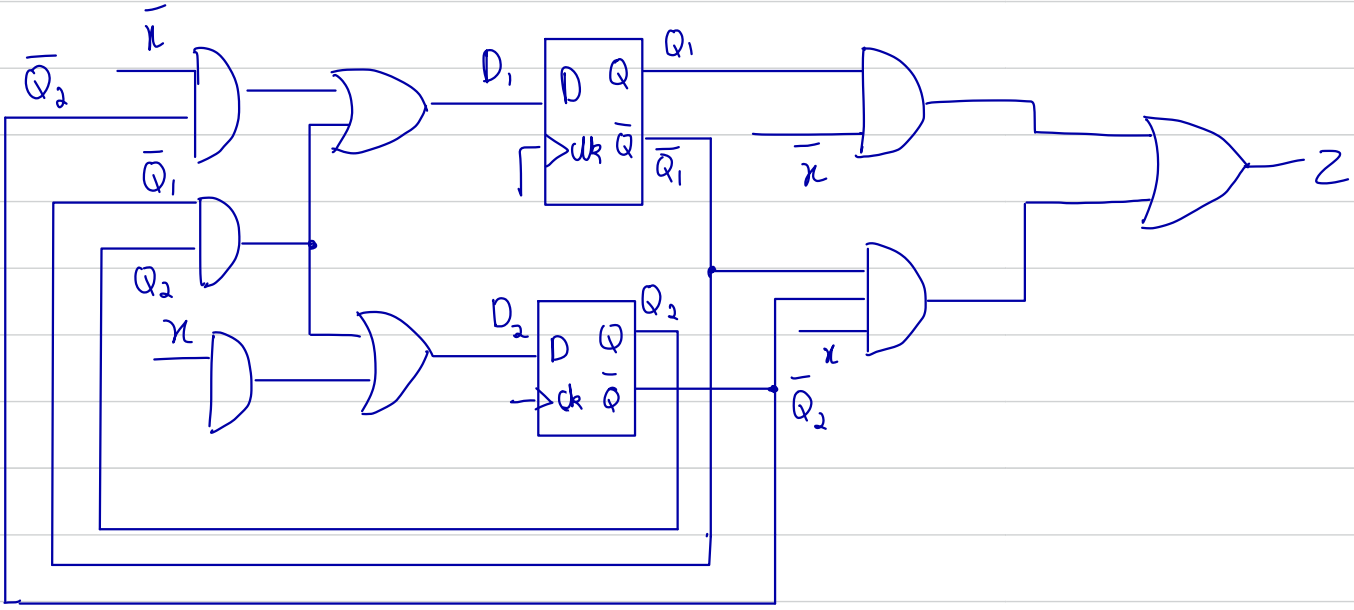
Next state logic: (current state decoder) - next state as a function of present state & input - combinational network.
$Q^+ = F(X, Q)$

Output logic (output decoder): combinational network
$Z = G(X, Q)$ - Mealy machine / class A machine

Moore machine / Machine B



Analysis of clocked sequential networks



Excitation and output expression (Assign state & excitation variables & then algebraic expressions for these variables)

$D_1 = \bar{x}\bar{Q}_2 + \bar{Q}_1 Q_2$

$D_2 = \bar{Q}_1 Q_2 + x\bar{Q}_1$

$Z = \bar{x} Q_1 + \bar{Q}_1 \bar{Q}_2 x$

Transition equations : Next state as a function of present state & input

- Use characteristic equations & excitation equations.

DFF → char equation → $Q^+ = D$

$\therefore Q_1^+ = D_1 = \bar{x}\bar{Q}_2 + \bar{Q}_1 Q_2$

$Q_2^+ = D_2 = \bar{Q}_1 Q_2 + x Q_1$

Transition tables : Tabular form representation:

→ 3 sections → Present state, next state, output

Present state		Next state				output	
		Q_1^+		Q_2^+		Input (x)	
Q_1	Q_2	i/p (x) 0	1	0	1	0	1
0	0	1	0	0	1	0	1
0	1	1	1	1	1	0	0
1	0	1	0	0	0	1	0
1	1	0	0	0	0	1	0

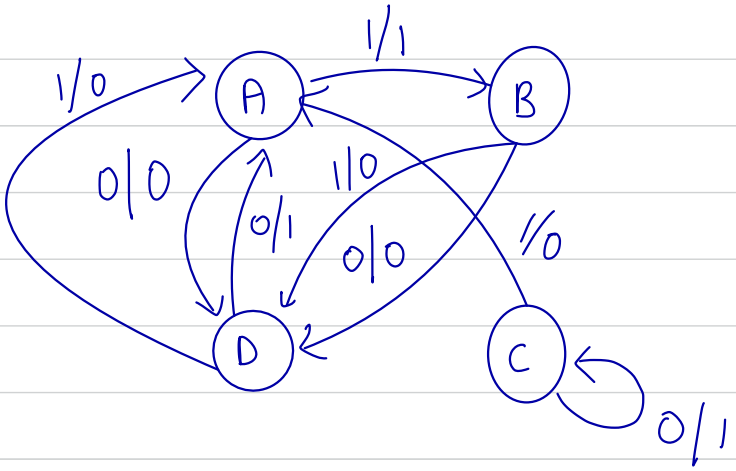
Alternate way : excitation table (PS, NS, input to FF)

State table : Alphanumeric symbols assigned to states instead of binary codes

00 → A 01 → B 10 → C 11 → D

Present state Q_1 Q_2	Next state Q_1^+ Q_2^+		output Input (x)	
i/p (x)	0	1	0	1
A	C	B	0	1
B	D	D	0	0
C	C	A	1	0
D	A	A	1	0

State diagram: Graphical representation of state table



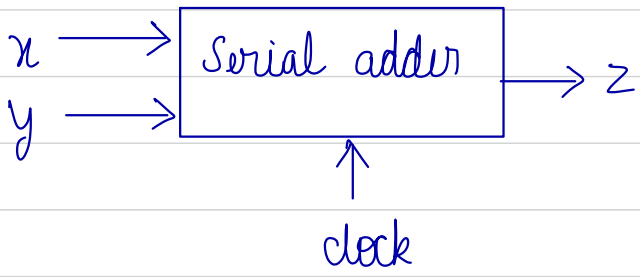
Network terminal behavior

Input sequence: 0 0 1 1 0 1 1 1 0 1
State sequence: A C C A B D A B D A
output sequence: 0 1 0 1 0 0 1 0 1 1

- Synthesis of synchronous sequential circuits

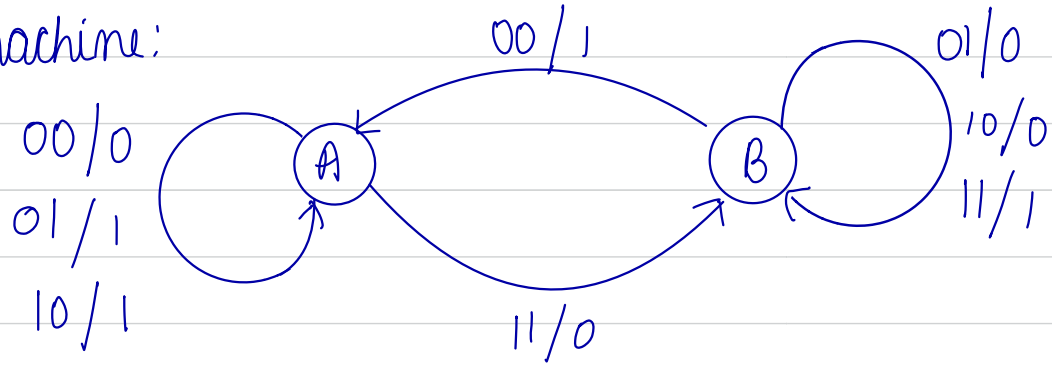
State diagram → State table → Transition table → Excitation table → Excitation & output equation → Logic diagram

Serial adder



Mealy machine:

A → C = 0
B → C = 1



A

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

B

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

Present State	Next state input (xy)				Output S input (xy)			
	00	01	10	11	00	01	10	11
A	A	A	A	B	0	1	1	0
B	A	B	B	B	1	0	0	1

Present state Q	Input (xy)	Next state	flipflop input T
A (0)	00	A(0)	0
A (0)	01	A(0)	0
A (0)	10	A(0)	0
A (0)	11	B(1)	1
B (1)	00	A(0)	1
B (1)	01	B(1)	0
B (1)	10	B(1)	0
B (1)	11	B(1)	0

Kmap for T :

PS \ xy	00	01	11	10
0 ($\bar{Q}$ A)	0	0	1	0
1 (Q B)	1	0	0	0

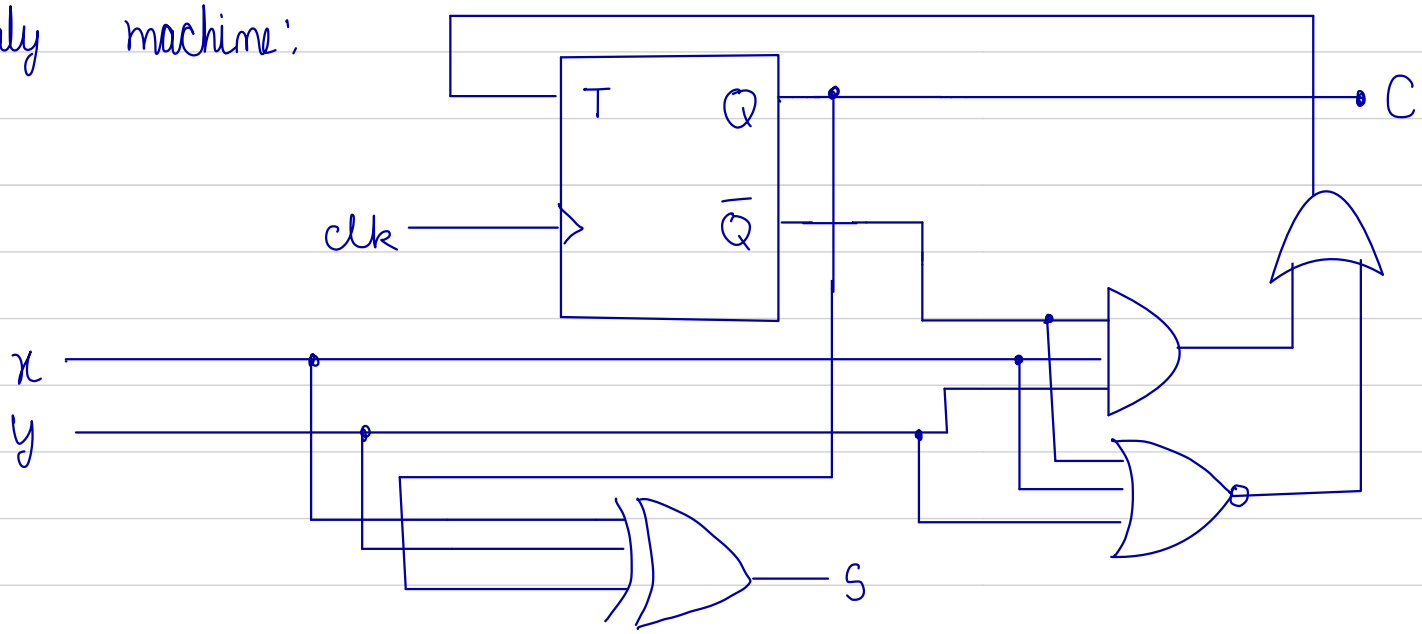
$T = \bar{Q}xy + Q\bar{x}\bar{y} = \bar{Q}xy + \overline{\bar{Q}+x+y}$

K map for S :

PS \ xy	00	01	11	10
0 (A)	0	1	0	1
1 (B)	1	0	1	0

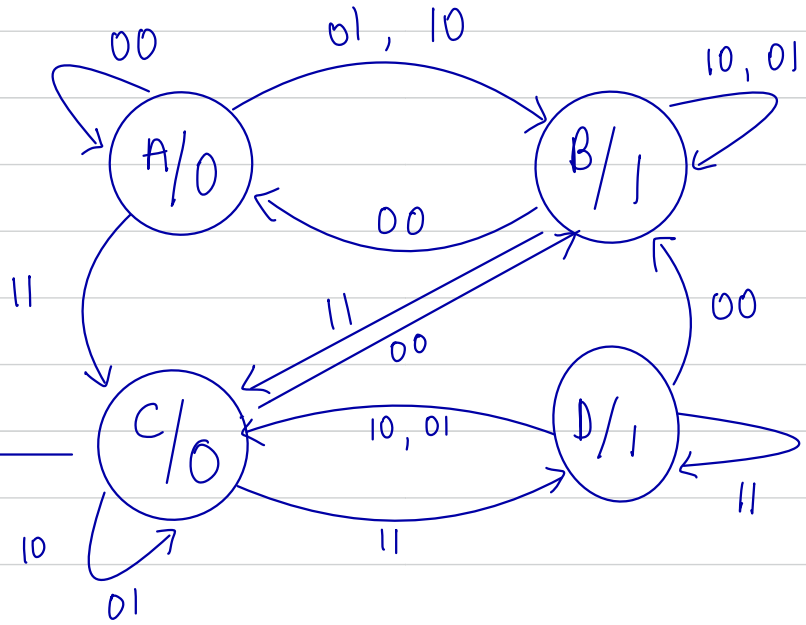
$Z = Q \oplus x \oplus y$

Mealy machine:



Moore machine:

A → S = 0 C = 0
B → S = 1 C = 0
C → S = 0 C = 1
D → S = 1 C = 1



State / output

Present State Q_2Q_1	Next state 00	01	10	11	output (z)	flipflop ip (T_2T_1) 00	01	10	11
A (00)	A	B	B	C	0	00	01	01	10
B (01)	A	B	B	C	1	01	00	00	11
C (10)	B	C	C	D	0	11	00	00	01
D (11)	B	C	C	D	1	10	01	01	00

K map for T_2:

PS \ xy	00	01	11	10
00 A	0	0	1	0
01 B	0	0	1	0
11 D	1	0	0	0
10 C	1	0	0	0

$T_2 = \bar{Q}_2 xy + Q_2 \bar{x}\bar{y} = \bar{Q}_2 xy + \overline{\bar{Q}_2 + x + y}$

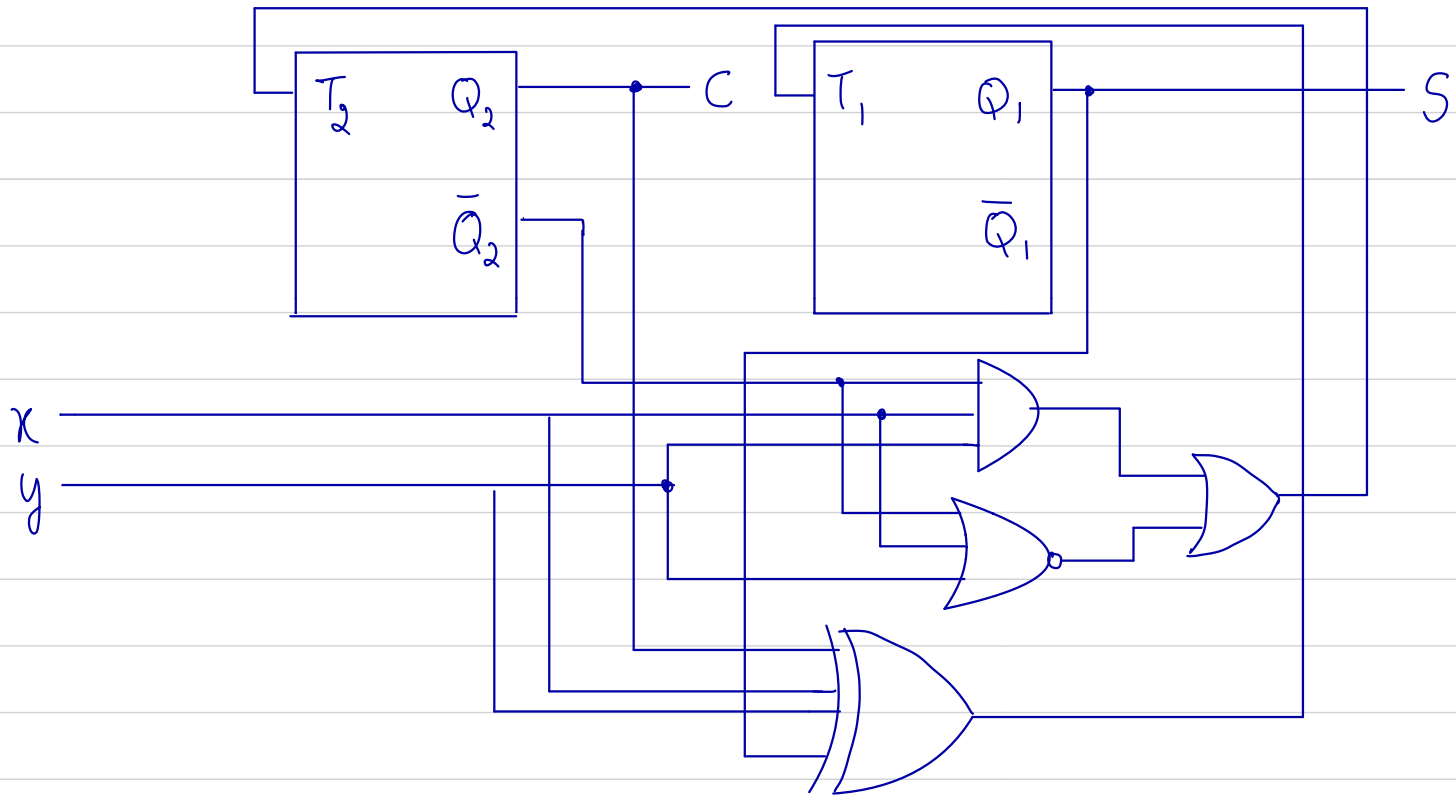
K map for T_1

PS \ xy	00	01	11	10
00 A	0	1	0	1
01 B	1	0	1	0
11 D	0	1	0	1
10 C	1	0	1	0

$T_1 = Q_1 \oplus Q_2 \oplus x \oplus y$

$S = Q_1$

$C = Q_2$



Tutorial:

Q1> Convert from decimal to BCD.

i> 35 - 0011 0101

ii> 98 - 1001 1000

iii> 170 - 0001 0111 0000

iv> 2469 - 0010 0100 0110 1001

Q2> Convert BCD to decimal

i> 1000 0110 - 86

ii> 0011 0101 0001 - 351

iii> 1001 0100 0111 1000 - 9478

Q3> Add the following BCD numbers -

i> 1001 + 0100 = 1101 > 9

1101 + 0110 = 0001 0011 = 13

ii) 1001 + 1001 = 10010 > 9

10010 + 0110 = 00011000 = 18

iii) 0001 0110 + 0001 0101 = 0010 1011

0010 1011 + 0000 0110 = 0011 0001 = 31

iv) 0110 0111 + 0101 0011 = 1011 1010

1011 1010 + 0000 0110 = 1100 0000

1100 0000 + 0110 0000 = 0001 0010 0000 = 120

Q4) Perform BCD addition of 8765 3943

Ans 1000 0111 0110 0101 + 0011 1001 0100 0011 = 1011 10000 1010 1000

1011 10000 1010 1000 + 0001 0001 0110 0000 = 1100 0001 0000 1000

1100 0001 0000 1000 + 0110 0110 0000 0000 = 0001 0010 0111 0000 1000 → 12708

Q5) Find 984 - 599 using 10's complement BCD subtraction

Ans 9's complement

999
-599
400

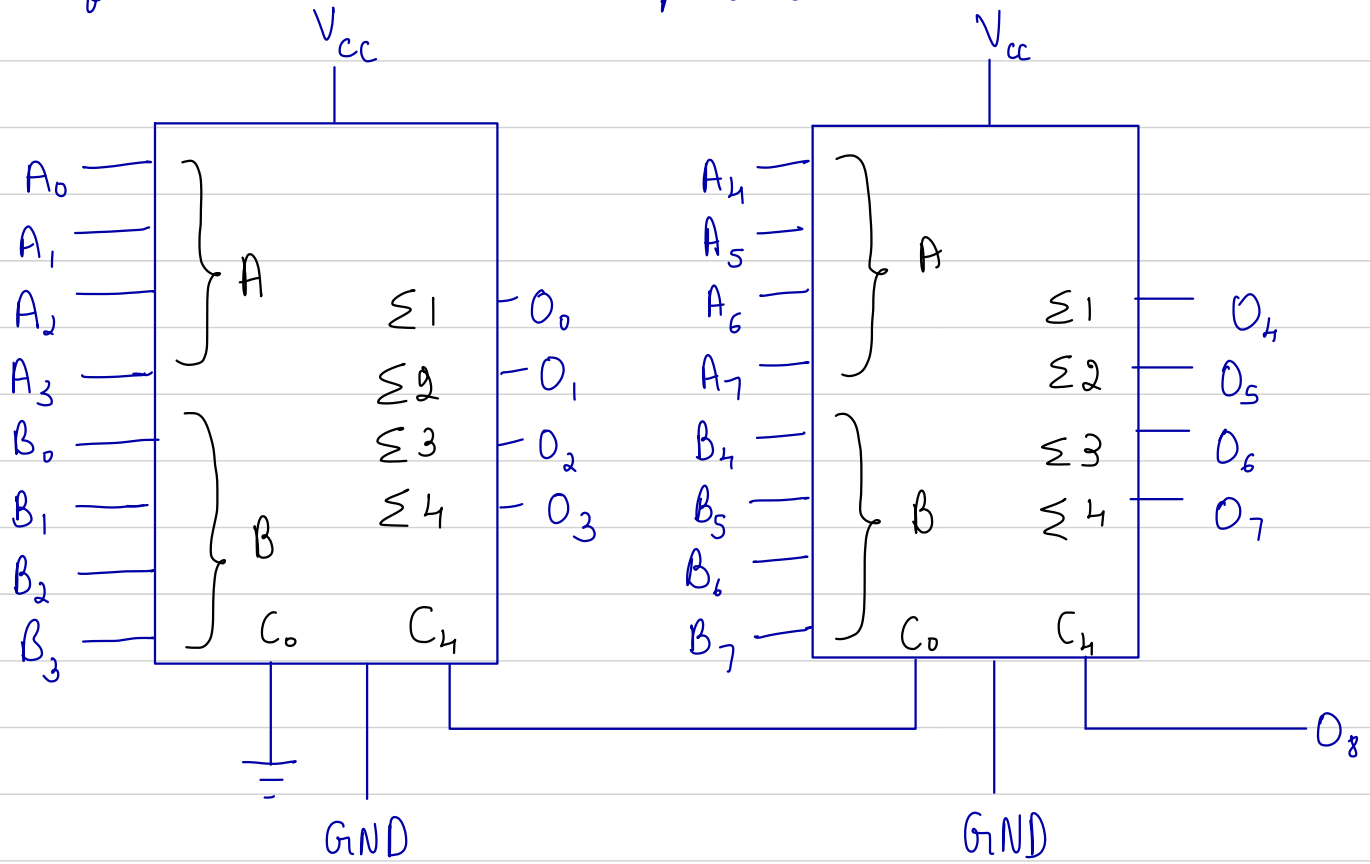
10's complement = 9's complement + 1 = 401

984 + 401 $\Rightarrow$

$$\begin{array}{r} 1001\ \ 1000\ \ 0100 \\ +\ 0100\ \ 0000\ \ 0001 \\ \hline 1101\ \ 1000\ \ 0101 \\ +\ 0110\ \ 0000\ \ 0000 \\ \hline \times 1)\ 0011\ \ 1000\ \ 0101 \end{array} = 385_{/}$$

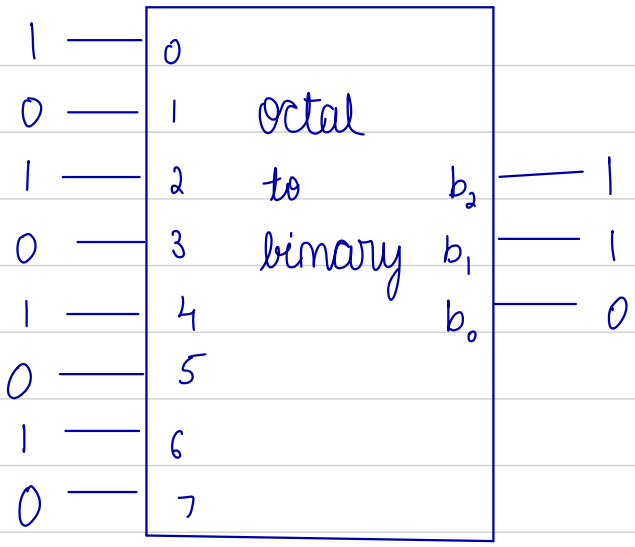
Q6) Show how 2 74HC283 adders can be connected to form an 8-bit parallel adder.

Ans

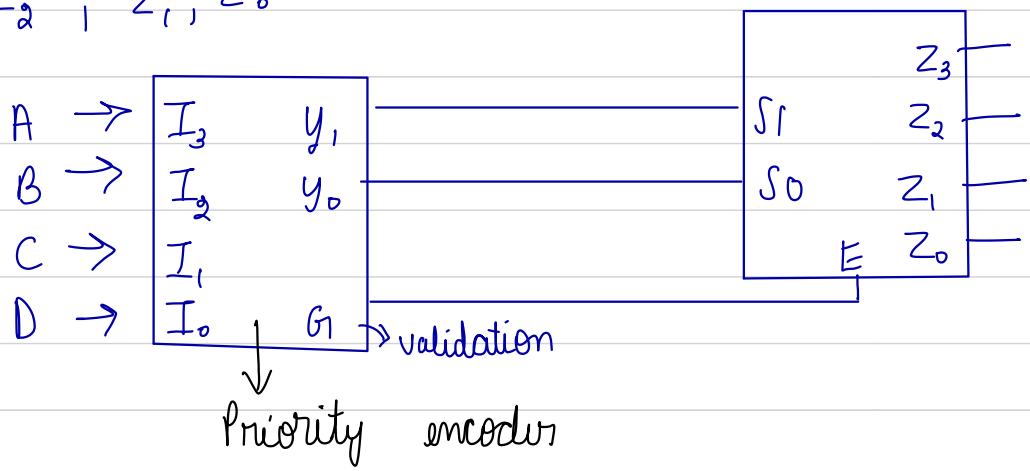


Q7) If an octal to binary priority encoder has 0, 2, 4, 6 input at active high level, active high binary output is ?

Ans



Q 8) What functions of A, B, C & D are realised by Z_3, Z_2, Z_1, Z_0



Ans

I_3	I_2	I_1	I_0	G1	Y_1	Y_0
0	0	0	0	0	X	X
0	0	0	1	1	0	0
0	0	1	X	1	0	1
0	1	X	X	1	1	0
1	X	X	X	1	1	1

$Y_1 = I_3 + I_2 = A + B$

$Y_0 = I_3 + \bar{I}_2 I_1 = A + \bar{B}C = A + \bar{A}B + \bar{A}\bar{B}C = A + \bar{A}(B+C)$

$Z_3 = Y_1 Y_0 G_1 = (A+B)(A+\bar{B}C)(A+B+C+D)$

$Z_2 = Y_1 \bar{Y}_0 G_1 = (A+B)(\overline{A+\bar{B}C})(A+B+C+D)$

$Z_1 = \bar{Y}_1 Y_0 G_1 = (\overline{A+B})(A+\bar{B}C)(A+B+C+D)$

$Z_0 = \bar{Y}_1 \bar{Y}_0 G_1 = (\overline{A+B})(\overline{A+\bar{B}C})(A+B+C+D)$

$\because G_1 = A+B+C+D$

Q10> A priority encoder has 4 inputs X3, X2, X1, X0. The circuit has 3 output Z, Y1, Y0. If one of the input is 1, Z is 1, & Y1, Y0 represent 0, 2 bit binary number whose value equals the index the highest numbered i/p i.e 1. If all the i/p are 0, Z=0 & Y1 & Y0 are don't cares

a> List in decimal form the min terms & don't care min terms of each o/p.

b> List in decimal terms, the max terms & don't care max terms.

Ans

X_3	X_2	X_1	X_0	Z	Y_1	Y_0
0	0	0	0	0	x	x
0	0	0	1	1	0	0
0	0	1	X	1	0	1
0	1	X	X	1	1	0
1	X	X	X	1	1	1

$Y_1 = \Sigma(4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15) + \Sigma d(0)$

$Y_0 = \Sigma(2, 3, 8, 9, 10, 11, 12, 13, 14, 15) + \Sigma d(0)$

$Z = \Sigma(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$

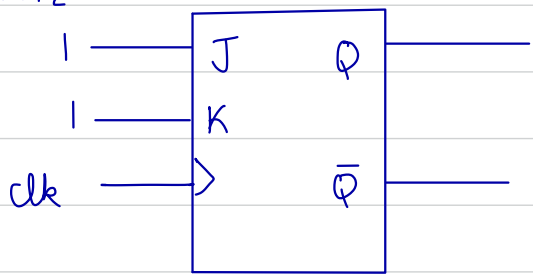
$Y_1 = \pi(1,2,3) + \pi_d(0)$

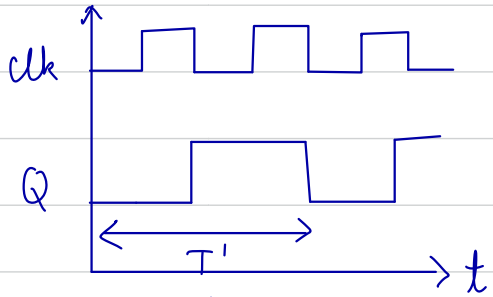
$Y_0 = \pi(1,4,5,6,7) + \pi_d(0)$

$Z = \pi(0)$

Q11> Determine the o/p frequency if clock frequency is 5 kHz

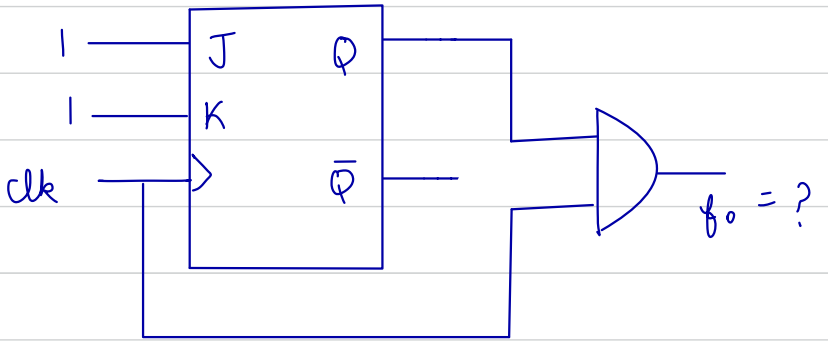
a>

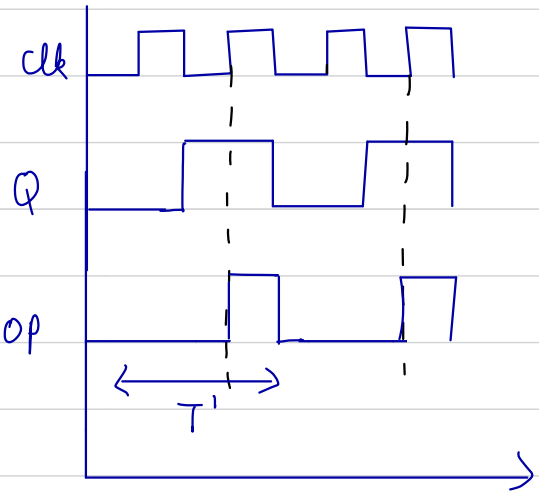




Ans $T' = 2T$ $\qquad f' = \frac{f}{2} = \frac{5}{2} = 2.5\ kHz$

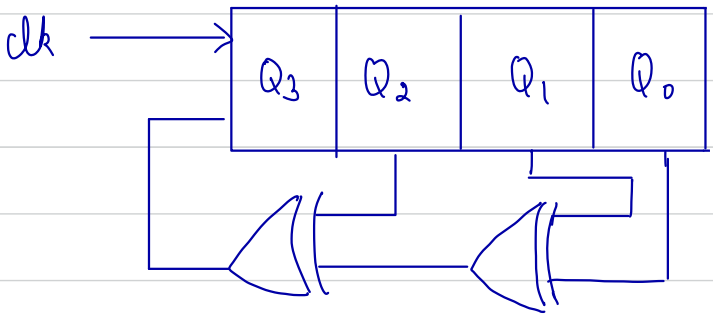
b>





$f_0 = \frac{f}{2} = 2.5\ kHz$

Q12> In the following shift register, the number of clock pulses to bring shift registers back to initial value of 1001 is ?



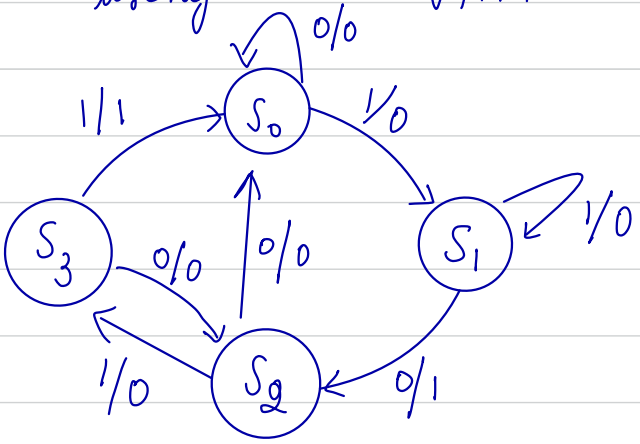
Ans

	Q_3	Q_2	Q_1	Q_0
start →	1	0	0	1
	1	1	0	0
	1	1	1	0
	0	1	1	1
	1	0	1	1
	0	1	0	1
	0	0	1	0
	1	0	0	1

} 7 clocks //

Q) Create a mealy machine sequence detector for '1011' without overlap. using JKFF

Ans)



Excitation table

Present state	Next state i/p (x)		o/p i/p (x)		flipflop input							
					0				1			
	0	1	0	1	J_1	K_1	J_0	K_0	J_1	K_1	J_0	K_0
S_0 (00)	S_0	S_1	0	0	0	x	0	x	0	x	x	1
S_1 (01)	S_2	S_1	0	0	1	x	x	1	0	x	x	0
S_2 (10)	S_0	S_3	0	0	x	1	0	x	x	0	1	x
S_3 (11)	S_2	S_0	0	1	x	0	x	1	x	1	x	1

J_1 :

x \ Q_1Q_0	00	01	11	10
0	0	1	3 X	2 X
x				
1	4 0	5 0	7 X	6 X
$\bar{x}$				

$J_1 = Q_0\bar{x}$

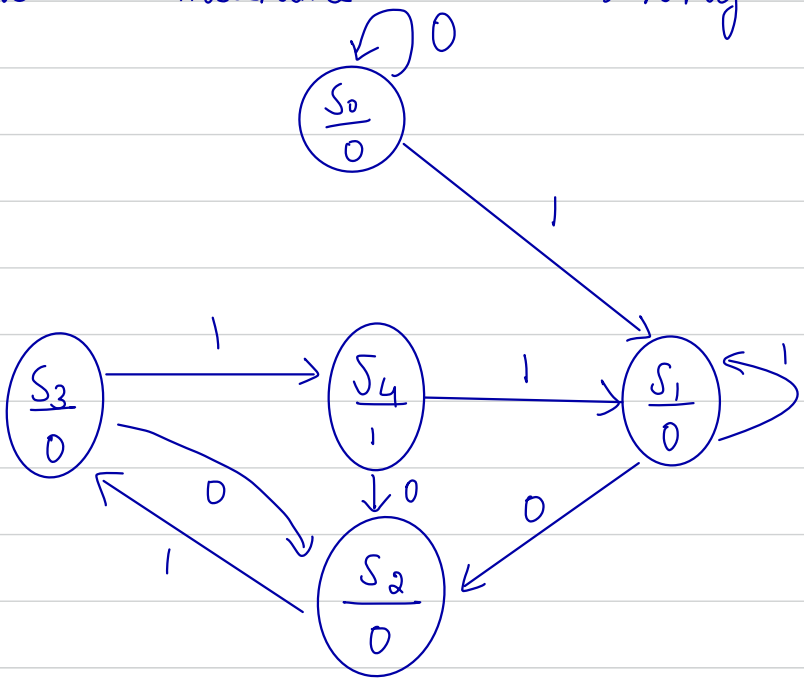
K_1

x \ Q_1Q_0	00	01	11	10
0	0 X	1 X	3 0	2 1
1	4 X	5 X	7 1	6 0

$K_1 = Q_0 \oplus x$

$J_0 = x$ $K_0 = \bar{x} + Q_1$ $Z = Q_1 Q_0 x$

Moore machine using DFF



$S_0 = 000$ $S_1 = 001$ $S_2 = 010$ $S_3 = 011$ $S_4 = 100$

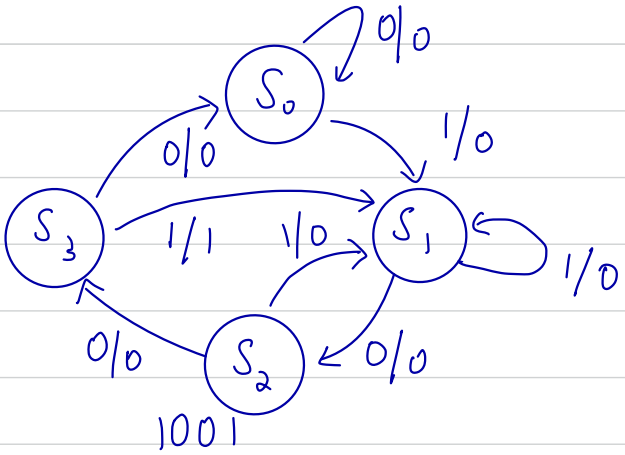
$Z = Q_2$ $D_2 = x Q_1 Q_0$

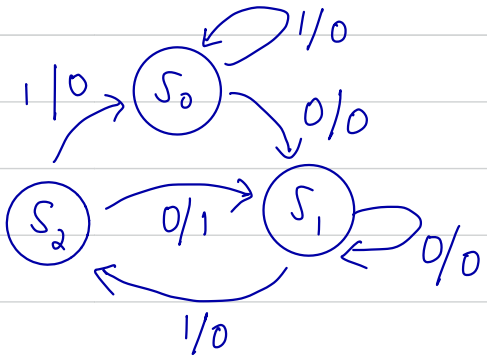
$D_1 = Q_2 Q_0 + \bar{x} Q_0 + x Q_1 Q_0$

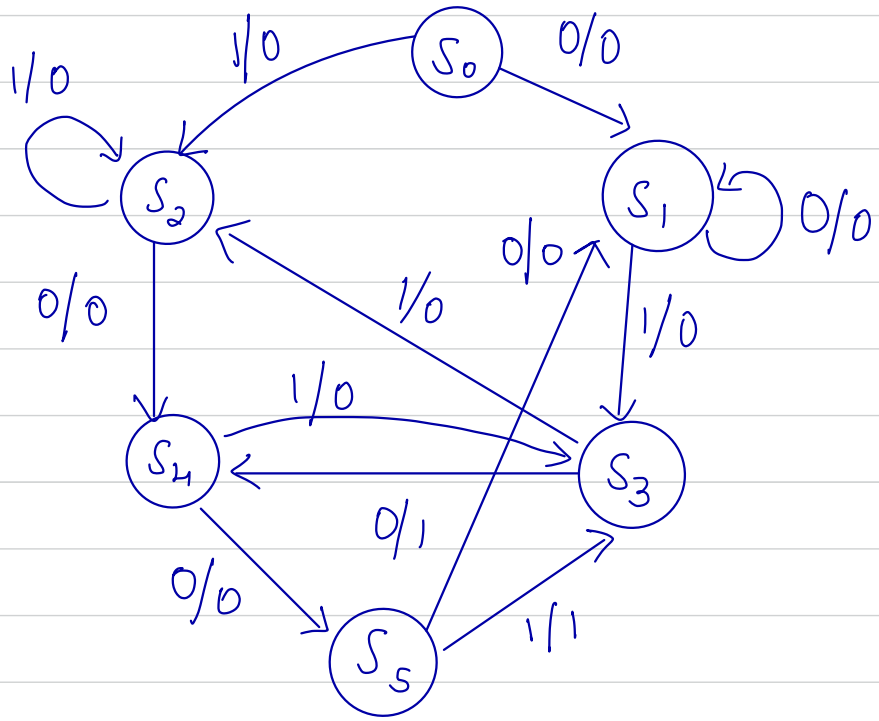
$D_0 = x \overline{Q_0} + \overline{Q_1} x$

Q) Detect 1001 or 010 with overlap as mealy machine.

Ans







S_0 → rust
S_1 → 0
S_2 → 1
S_3 → 01
S_4 → 10
S_5 → 100

State assignment

1) one-hot assignment: At a time only one bit is high. Advantage: No decoder required.
Eg: $S_0 = 001$ $S_1 = 010$ $S_2 = 100$
Disadvantages: More flipflops are required.

2> Binary assignment: States are assigned with binary numbers

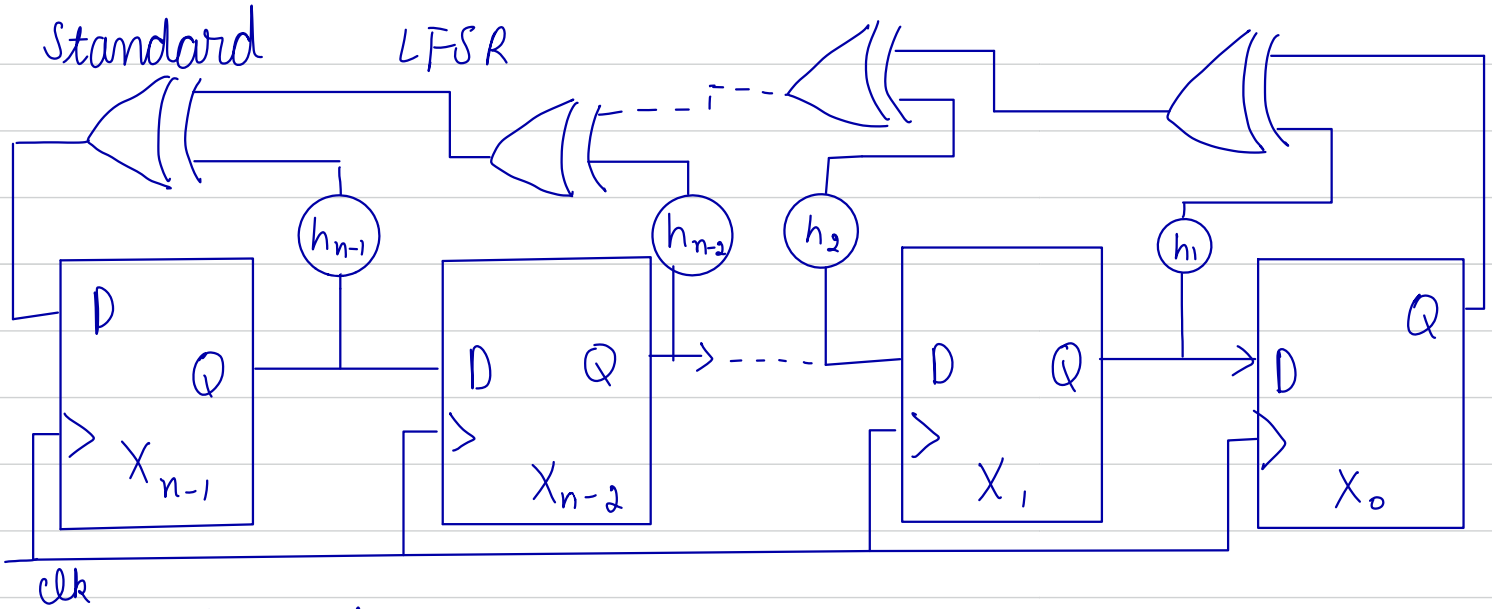
3> Random assignment: Any random numbers can be assigned to states, but first state should be all zeros or all ones.

Sequence generator

Linear feedback shift register (LFSR)

2 types: 1> Standard LFSR
2> Modular LFSR

1> Standard LFSR



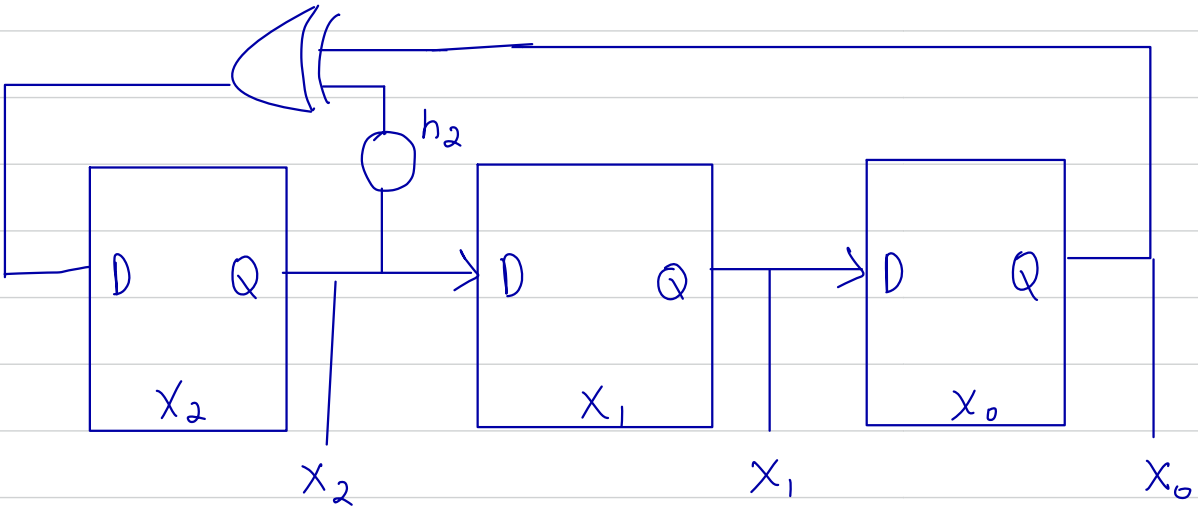
Maximal length LFSR - LFSR that can generate nearly exhaustive states ($2^n - 1$ distinct states)

characteristic equation:

$$f(x) = 1 + h_1 x + h_2 x^2 + h_3 x^3 + \dots + h_{n-1} x^{n-1} + x^n$$

where $h_i = 0$ or 1

Q> Write characteristic equation:



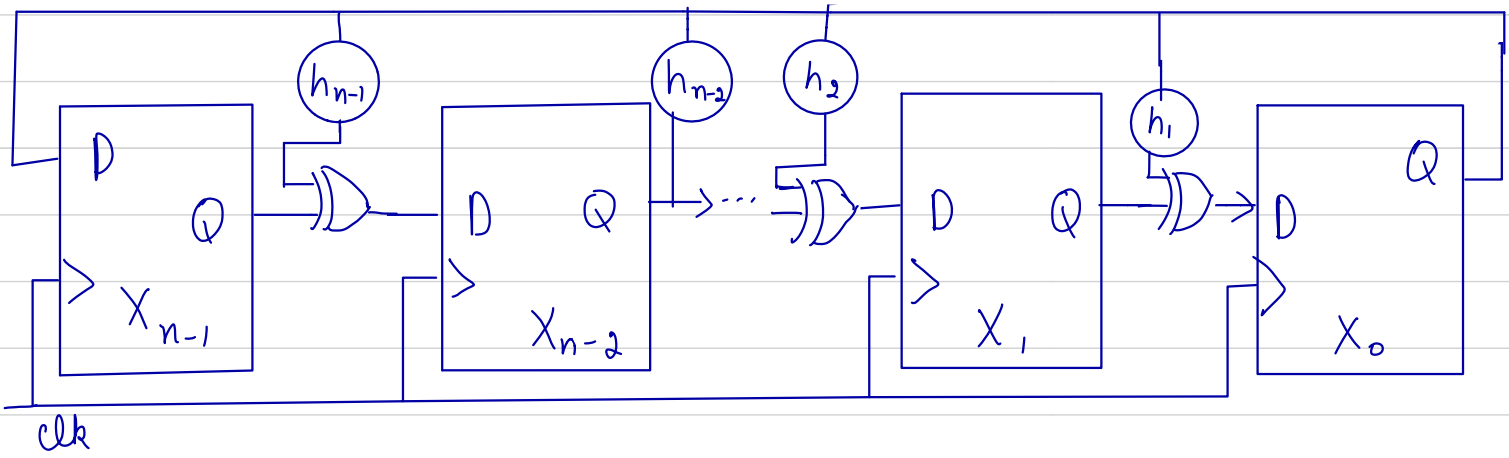
Ans

$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \end{bmatrix}$$

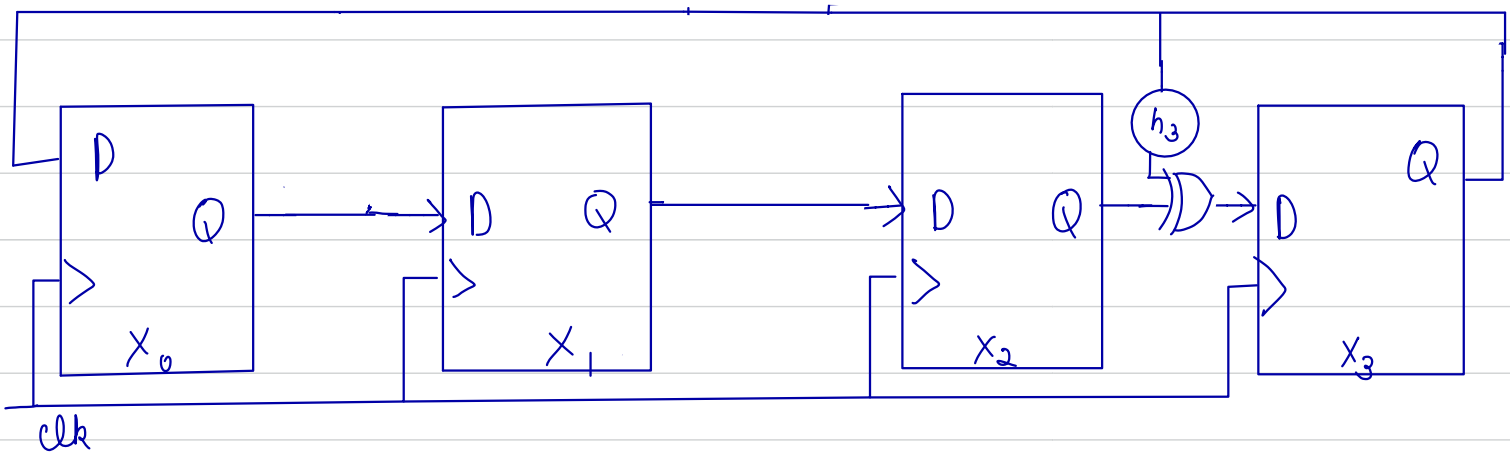
$X_0(t+1) = X_1(t)$
$X_1(t+1) = X_2(t)$
$X_2(t+1) = X_0(t) + X_2(t)$ $\quad h_1 = 0 \quad h_2 = 1$
$f(x) = 1 + x^2 + x^3$,

Modular LFSR

Faster than standard LFSR - one XOR gate.



Ex:



$$\begin{bmatrix} X_0(t+1) \\ X_1(t+1) \\ X_2(t+1) \\ X_3(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} X_0(t) \\ X_1(t) \\ X_2(t) \\ X_3(t) \end{bmatrix}$$

$f(x) = 1 + x^3 + x^4$

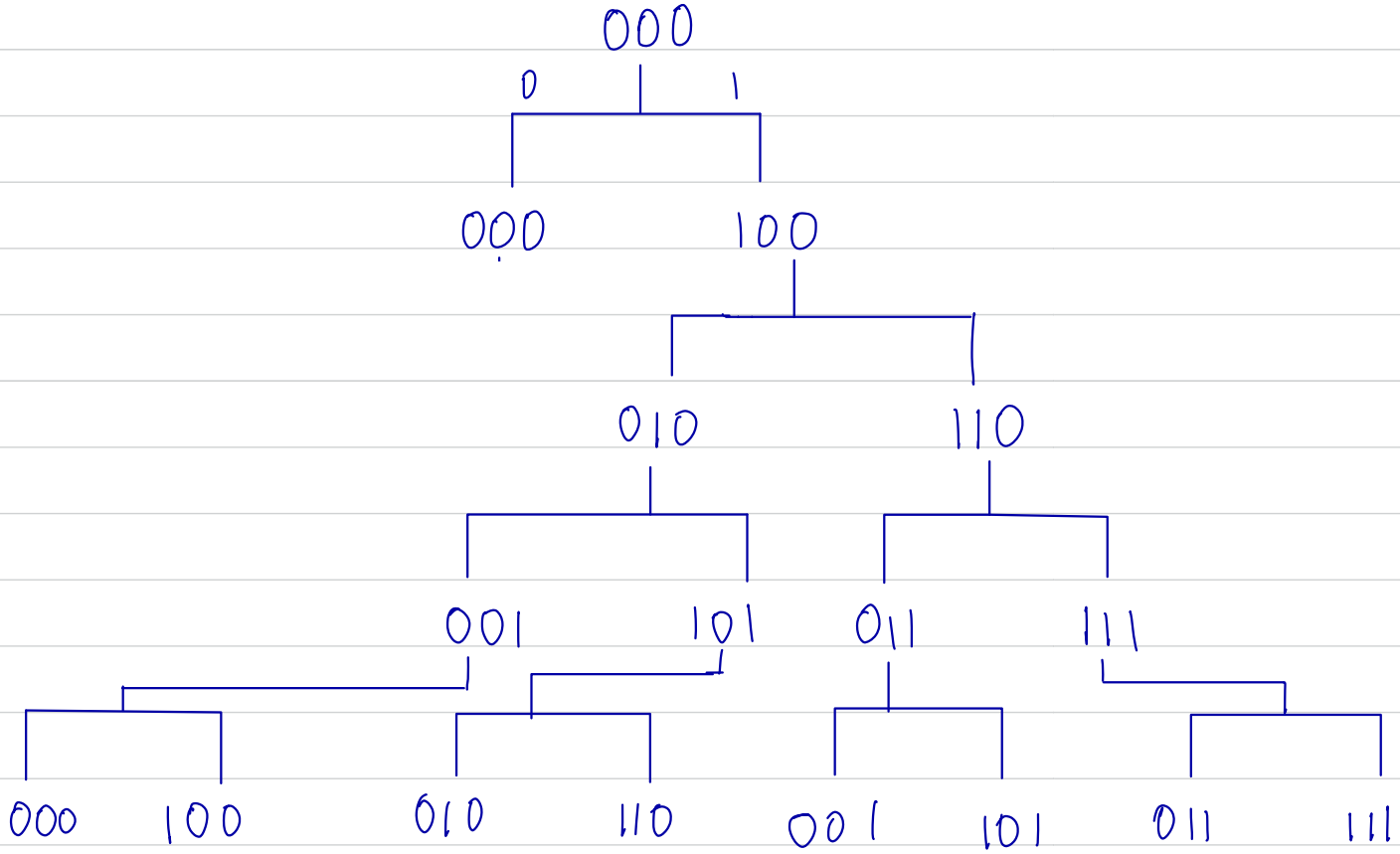
Binary sequence generator using shift register

Generate sequence 10110110

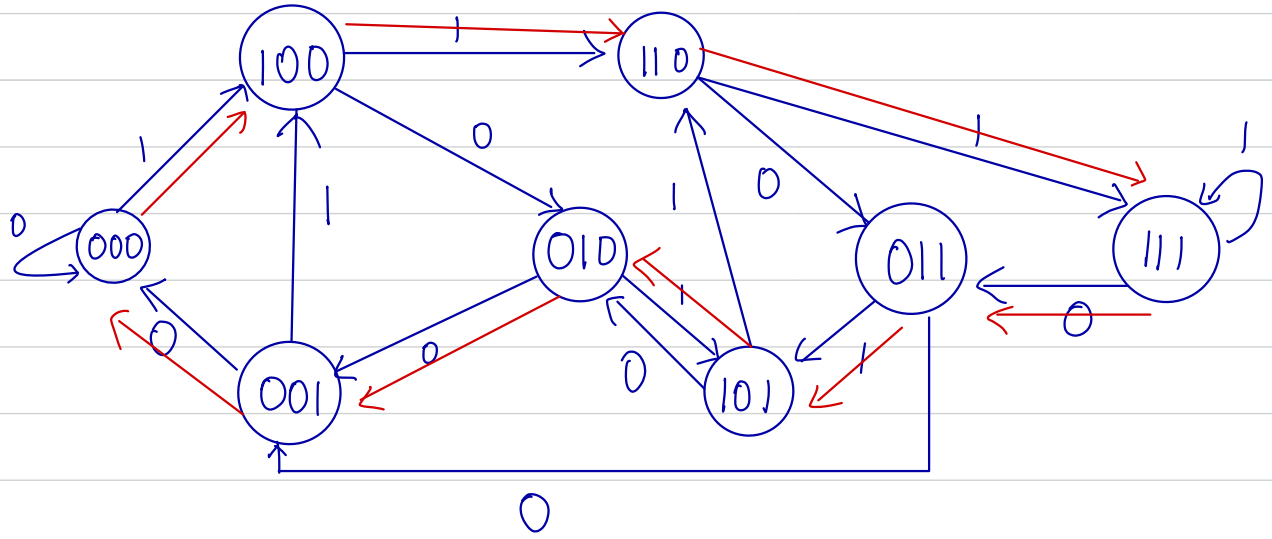
- need 3FF

→ Use universal shift register - trace the path which passes through all the states - design logic circuit for serial in and output

Binary tree diagram:



State diagram



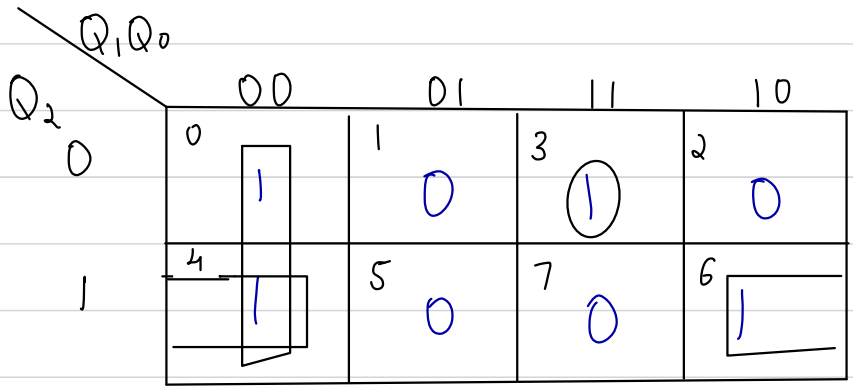
→ Path going through each node only once. (0 → 4 → 6 → 7 → 3 → 5 → 2 → 1 → 0)

PS $Q_2 Q_1 Q_0$	S_{in}	output Z
0 0 0	1	1
1 0 0	1	0
1 1 0	1	1
1 1 1	0	1
0 1 1	1	0
1 0 1	0	1
0 1 0	0	1
0 0 1	0	0

$S_0 \rightarrow S_4 \rightarrow S_6 \rightarrow S_7 \rightarrow S_3 \rightarrow S_5 \rightarrow S_2 \rightarrow S_1$

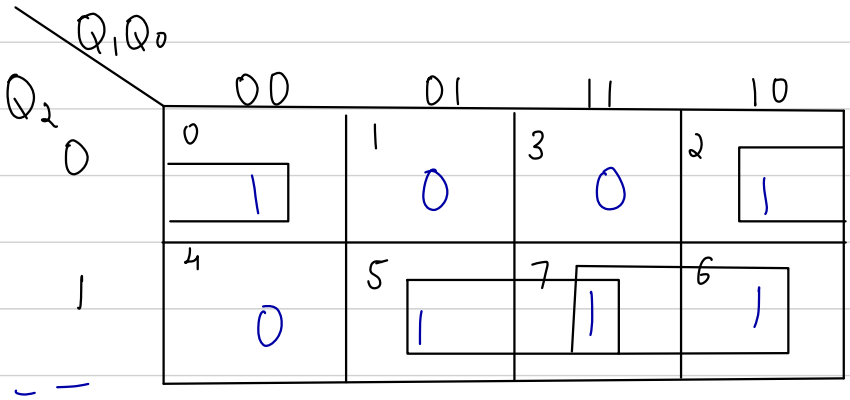
1 0 1 1 0 1 1 0

Sin:

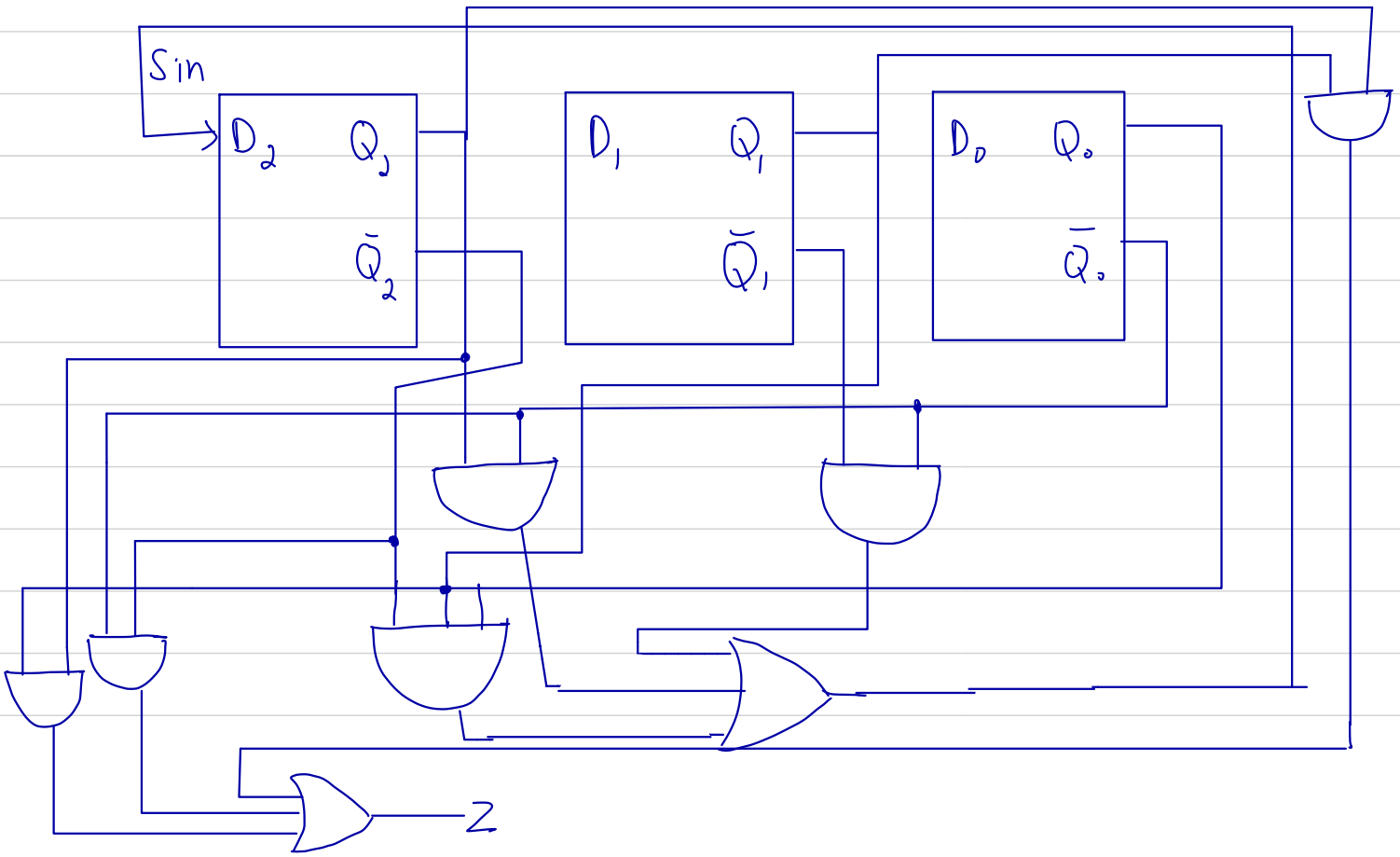


$Sin = \bar{Q}_1\bar{Q}_0 + Q_2\bar{Q}_0 + \bar{Q}_2 Q_1 Q_0$

Z:



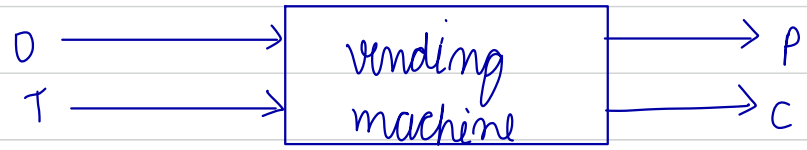
$Z = \bar{Q}_2\bar{Q}_0 + Q_2Q_0 + Q_2Q_1$



FSM state reduction:

State A & B equivalent
→ Next state for A & B - same/ equivalent
→ Same output

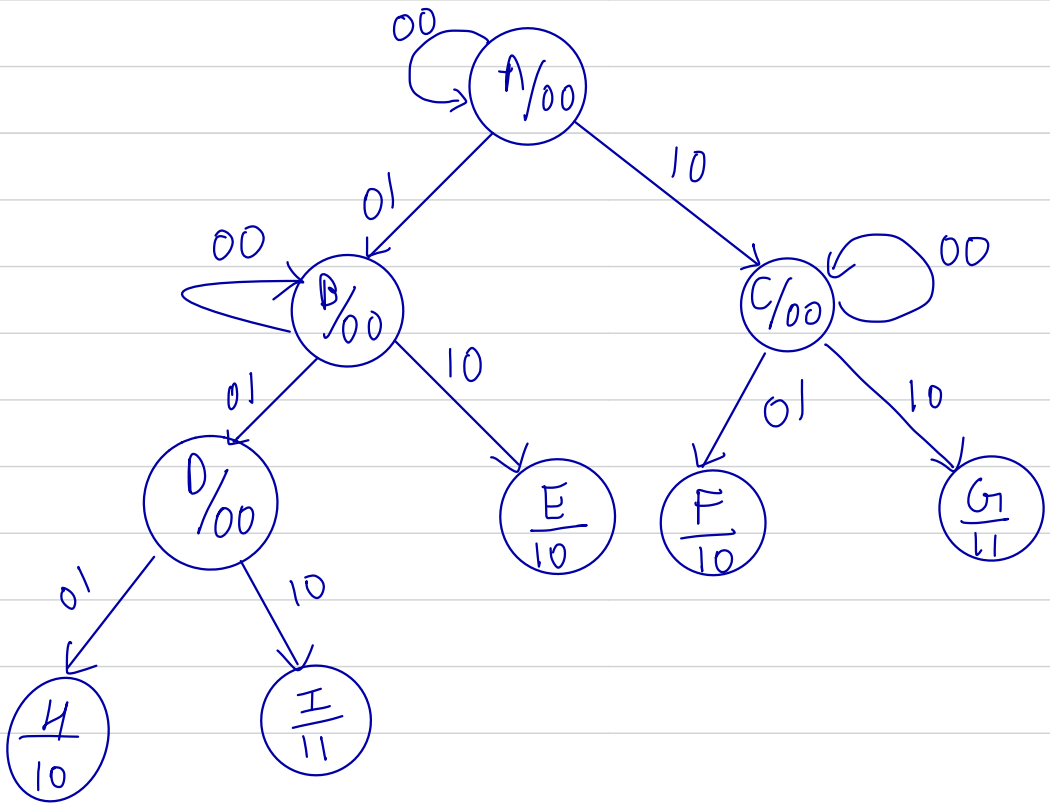
Q • Design a vending machine which releases a chocolate for ₹3. The machine accepts money only in terms of ₹2 & ₹1. Extra money will be paid back in ₹1 coins.



0 → ₹1 recieved
T → ₹2 recieved

P → Product dispensed
C → change dispensed.

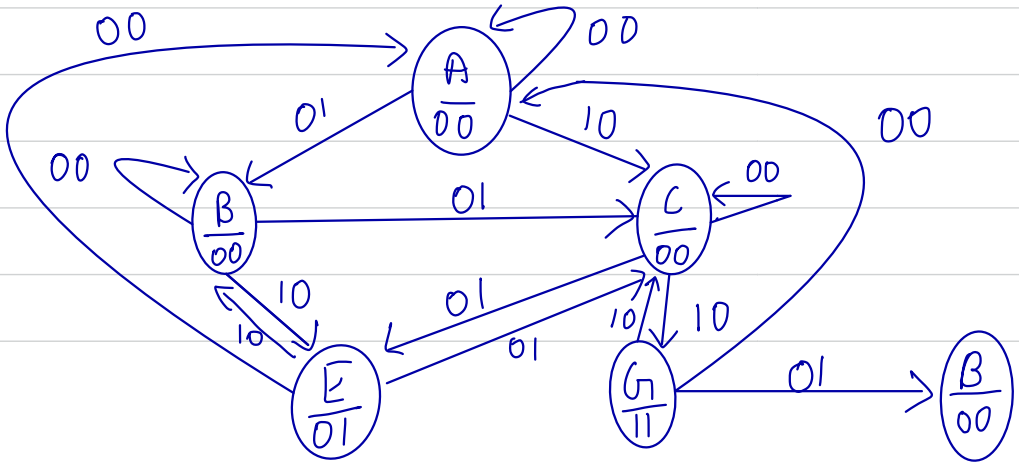
Moore machine:
input → TO
output → P C



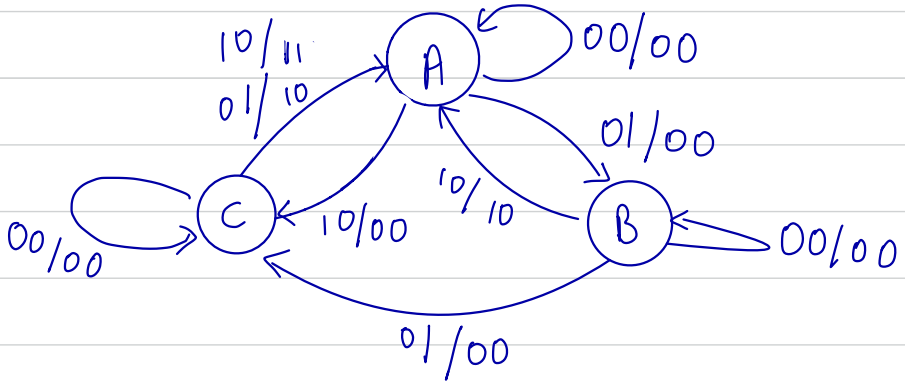
Present state	Next state I/p (T O) 00	01	10	output (PC)
A	A	B	C	00
B	B	~~B~~ C	E	00
C	C	~~F~~ E	G	00
~~D~~	~~D~~	~~H~~ E	~~I~~ G1	~~00~~
E	A	B	C	10
~~F~~	~~A~~	~~B~~	~~C~~	~~10~~
G1	A	B	C	11
~~H~~	~~A~~	~~B~~	~~C~~	~~10~~
~~I~~	~~A~~	~~B~~	~~C~~	~~11~~

Final table:

Present state	Next state 00	01	10	output (PC)
A	A	B	C	00
B	B	C	E	00
C	C	E	G1	00
E	A	B	C	10
G1	A	B	C	11



Mealy machine implementation



A = 0

B = 1

C = 2

Q_1 Q_2

MSB LSB

$J_1 = QQ_2 + \bar{Q}_2 T$

$J_2 = \bar{Q}_1 0$

$C = Q_1 T$

$K_1 = 0 + T$

$K_2 = K_1$

Q> Design a vending machine with input of denominations ₹1, ₹2, ₹5. It should dispense product & change.

Logic families

- Digital circuits classifications based on circuit technology.
- Popular logic families:
  - → Transistor - transistor logic (TTL)
  - → Emitter coupled logic (ECL)
  - → Metal oxide semiconductor (MOS)
  - → Complimentary metal oxide semiconductor (CMOS)

Digital circuit parameters

1) Fanout
2) Power dissipation
3) Propagation delay
4) Noise margin

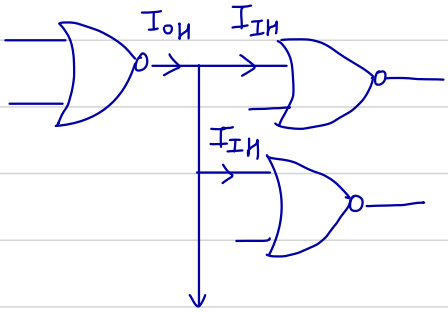
1) Fanout:

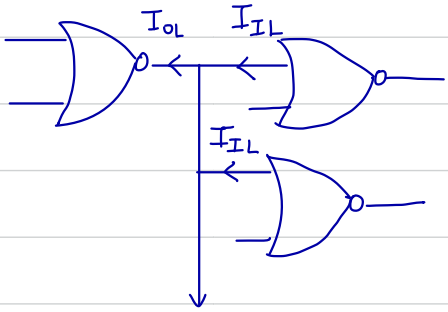
→ Number of standard loads that can be connected to the output of the gate without degrading its normal operation.

→ Calculated using the amount of current in the output of a gate and the amount of current needed in the input of each gate.

→ Output voltage High - gate provide current to load (gates)

- Output voltage low - gate provides current sink to load (gates)
- Fanout = $\min\left(\frac{I_{OH}}{I_{IH}}, \frac{I_{OL}}{I_{IL}}\right)$





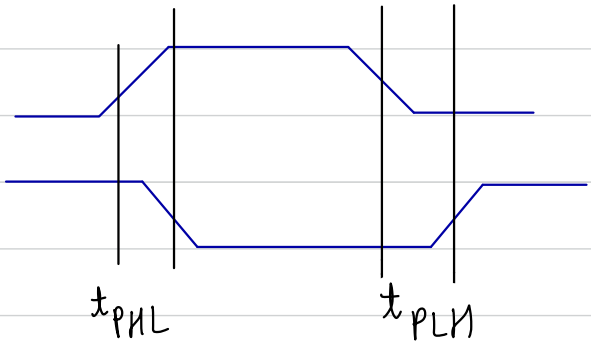
2> Power dissipation

- Power delivered to the gate from supply
- Expressed in mW
- $P_D = V_{CC} \times I_C$
- Since I_C depends on logic state of the gate, average current is considered for P_D calculation
- Average power = $V_{CC}\left(\frac{I_{CH} + I_{CL}}{2}\right)$
- P_D by an IC = P_D of a gate x number of gates in an IC.
- P_D for CMOS, $P_D = C_L \cdot V_{DD}^2 \times f$ (f - frequency) (C_L - load capacitor)

3> Propagation delay

Input :

output :

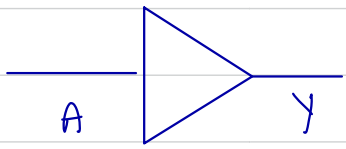


Average transition delay time for the signal to propagate from input to output when the binary signal changes value (alt: maximum time from input crossing 50% to output crossing 50% of final value)

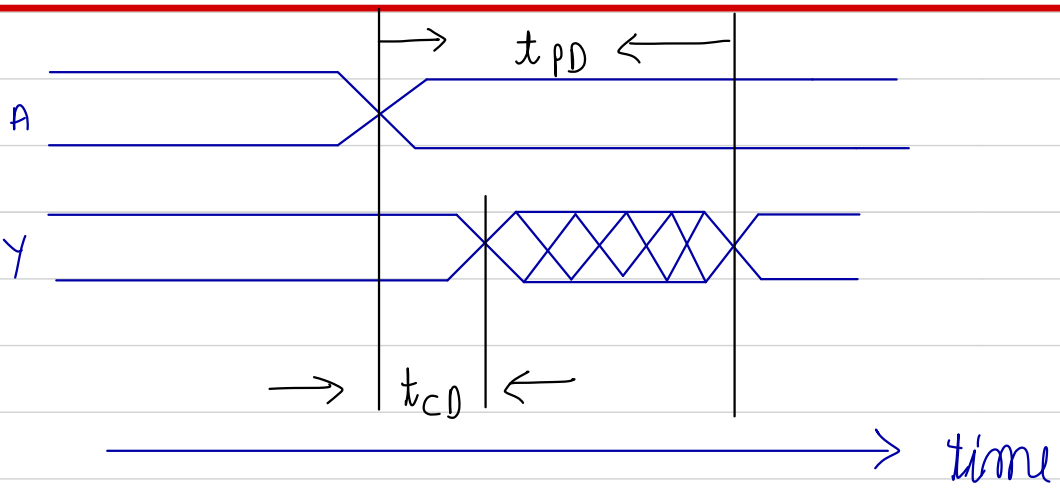
- Typically measured in ns or ps
- Average propagation delay = $\frac{t_{PHL} + t_{PLH}}{2}$ (max propagation delay may have to be considered in specific applications).

Rise time: Time taken to rise from 10% to 90% in a rise. Similarly fall time can be defined in vice versa manner.

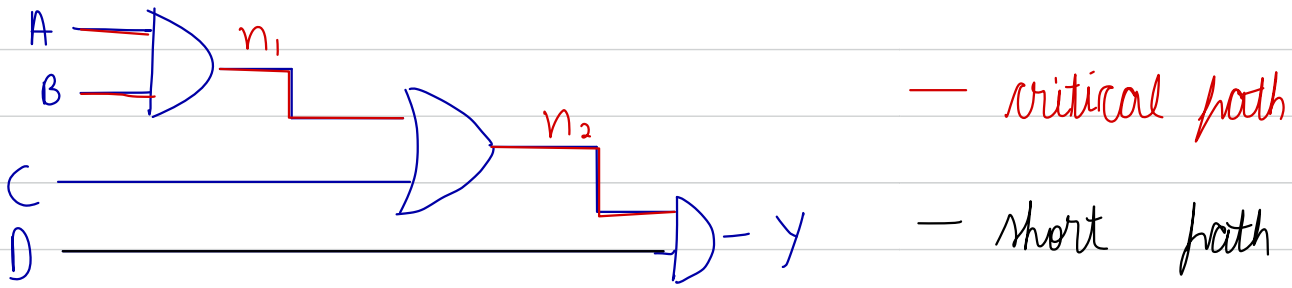
Contamination delay:



Minimum time from input crossing 50% to output crossing 50% (of previous value)

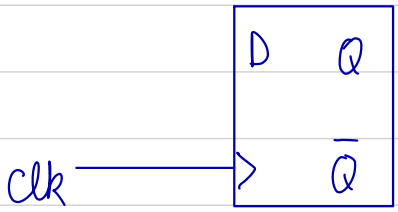


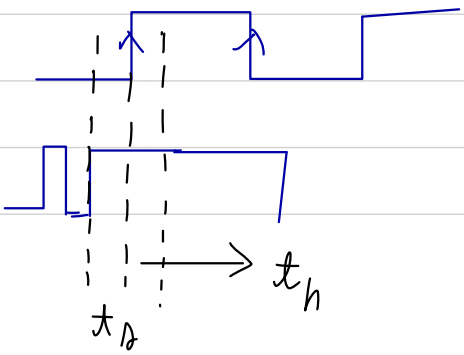
Critical path and short path:



- Critical path: path with max delay
- Short path: path with minimum delay
- For a circuit propagation delay = sum of propagation delay in critical path.
- Contamination delay = sum of contamination delay in short path.

Setup time:



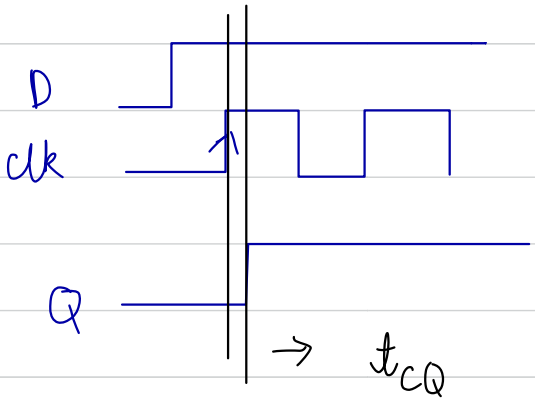


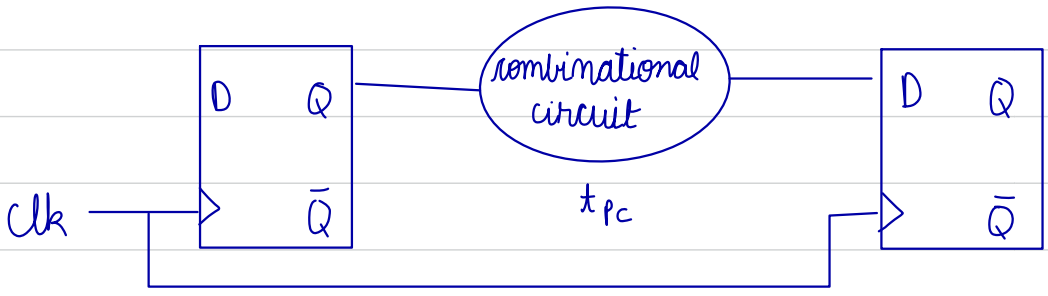
- The input to flipflops must be stable for some time before the clock edge. This is called "setup time", t_s.

- The input to flipflop must be stable for some time after the clock edge. This is called "hold time", t_h.

∴ The input should be stable for time = $t_s + t_h$

Propagation delay between clock & output is t_{CQ}





$T_{clk} \geq t_{CQ} + t_{pc} + t_{su}$

↓

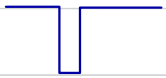
combinational propagation delay

$f_{clk} = \frac{1}{T_{clk}}$

Hazards: 1> Static 2> Dynamic

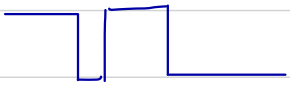
1> Static Hazard.

Static '0'



Static '1'

2> Dynamic Hazard:



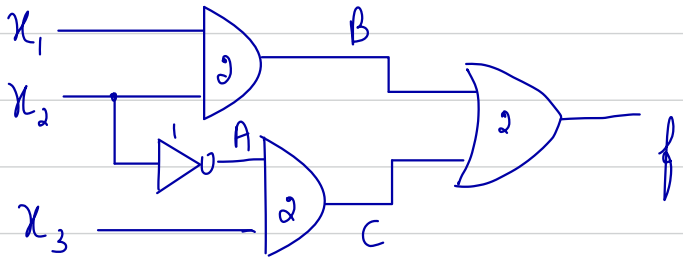
Dynamic '0'

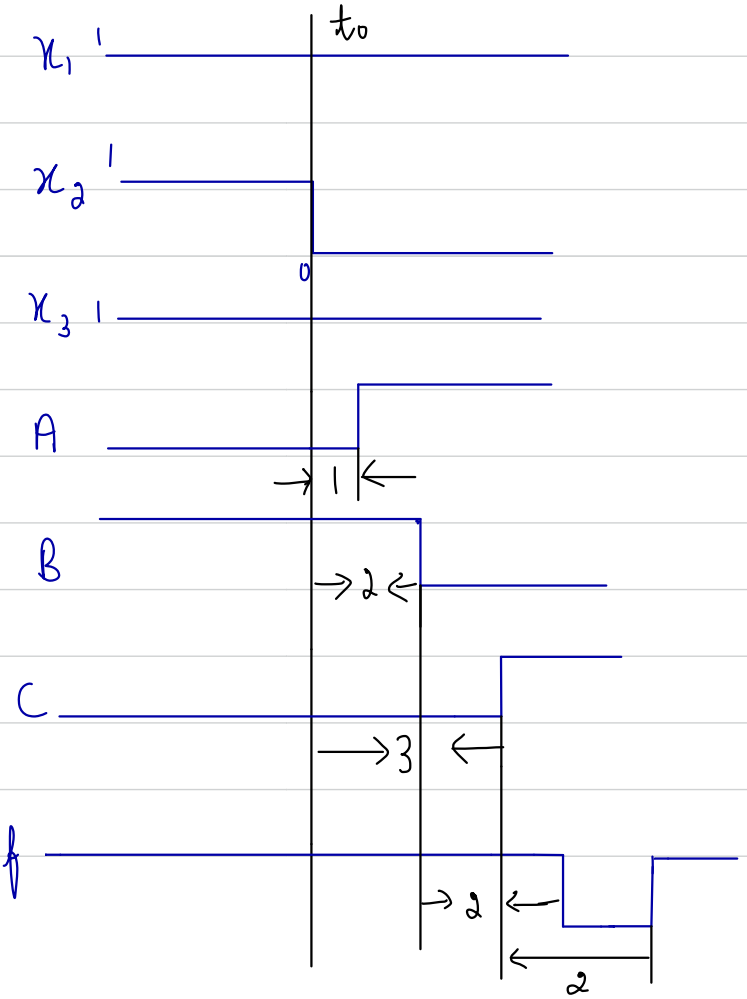
Dynamic 1

Static hazard example:

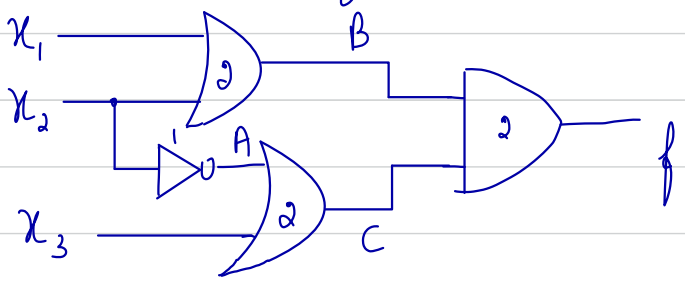
1> Static '1' hazard: 111 → 101

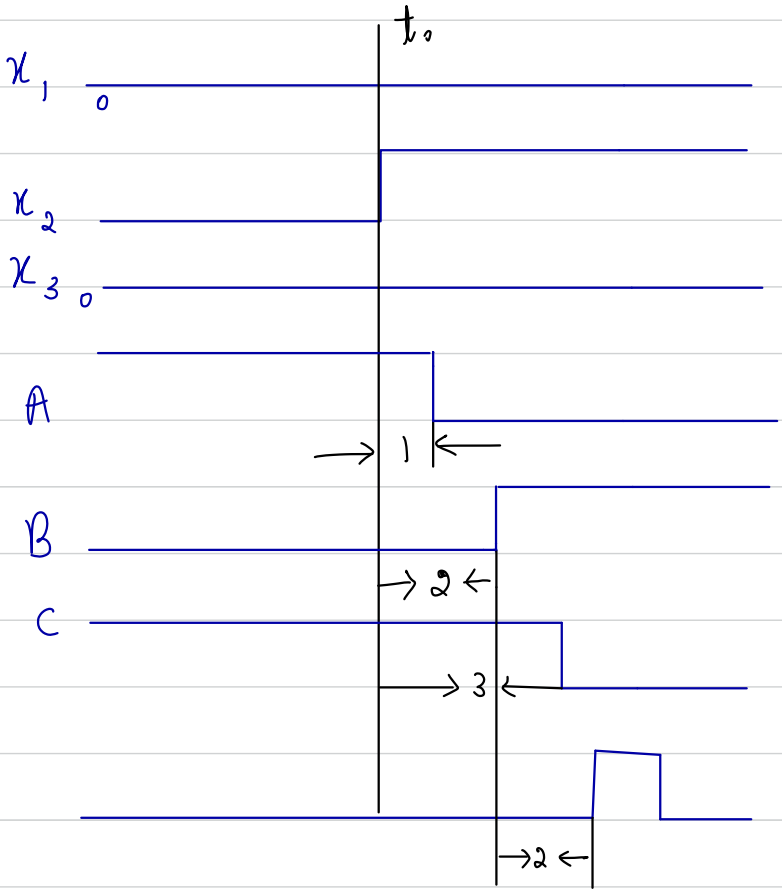
2 → Propagation delay



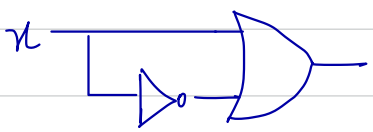


2> Static '0' hazard: 000 → 010

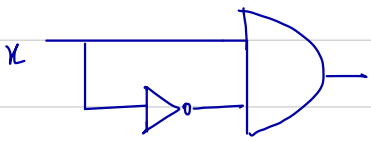




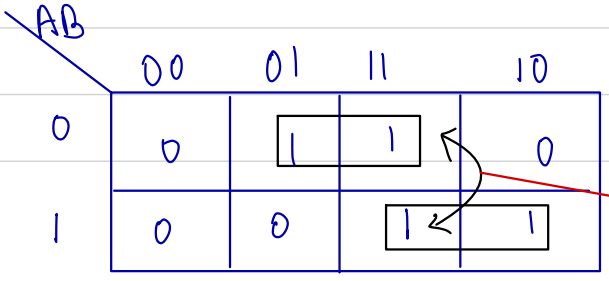
Static 1:



Static 0:



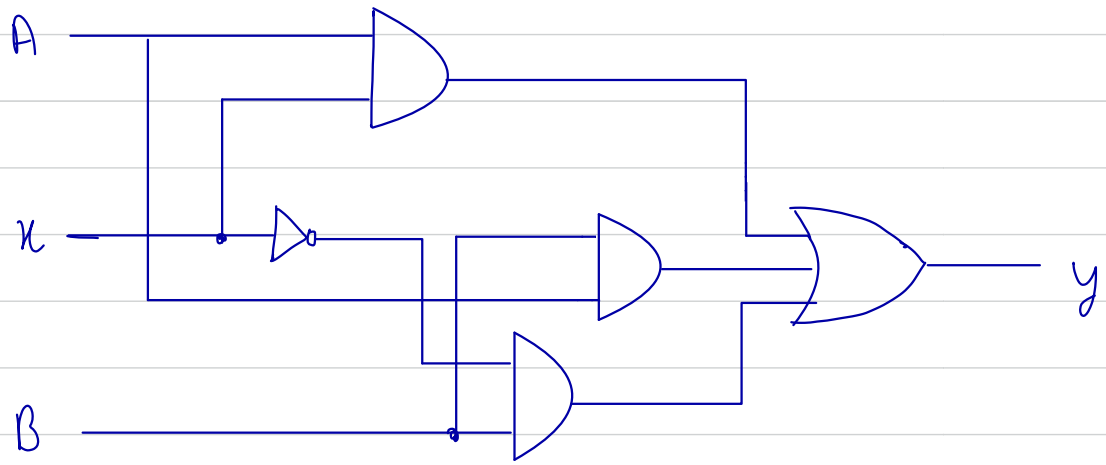
Identifying static hazard in K map.



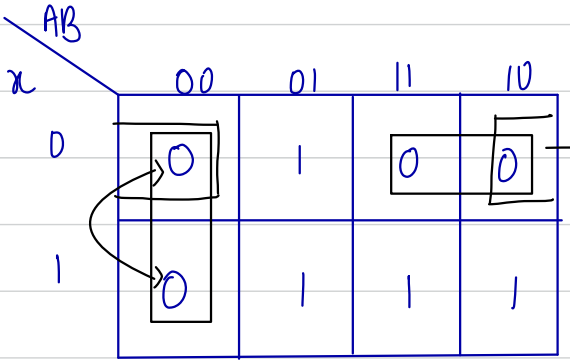
$y = \bar{x}B + xA$

This transition causes static '1' hazard

To prevent that we take $y = \bar{x}B + xA + AB$



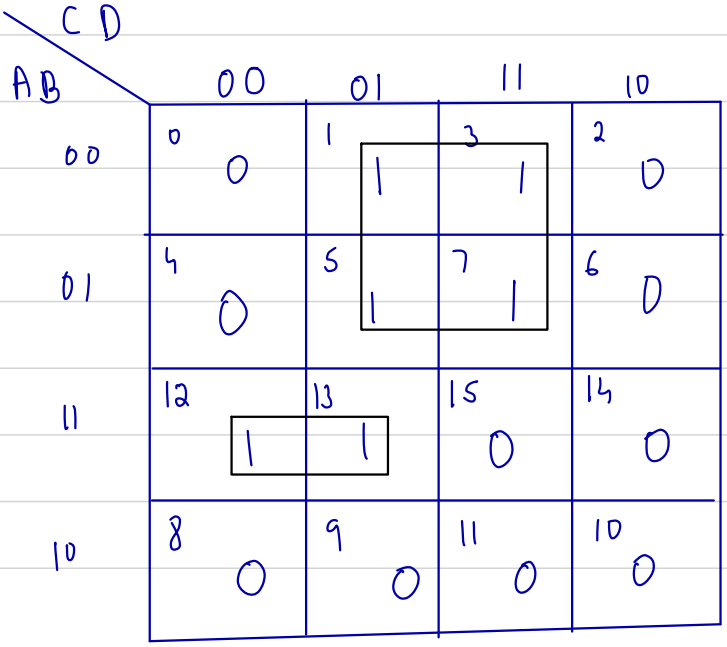
Static 0 hazard:



→ Add this redundant term

$y = (A+B)(x+\bar{A}) \cdot (x+B)$

Eg: $Y(A,B,C,D) = \Sigma m(1,3,5,7,12,13)$



$Y = \bar{A}D + AB\bar{C} + B\bar{C}D$

↓

Add this redundant term

SOP - Static '1' hazard
POS - Static '0' hazard.

Figure of merit

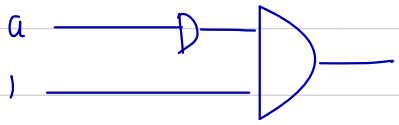
Higher the current, higher is the charging speed (lesser charging time).

Power delay produced, PDP = $P \times t_p$
$P \rightarrow$ Power dissipated $t_p \rightarrow$ propagation delay

Delay modules in verilog:

1) Inertial delay
2) Transport delay

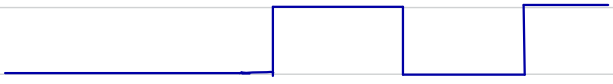
1) Inertial delay: If there are any quick changes in the input $\leq$ inertial delay, then it ignores the change.



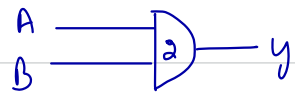
a:

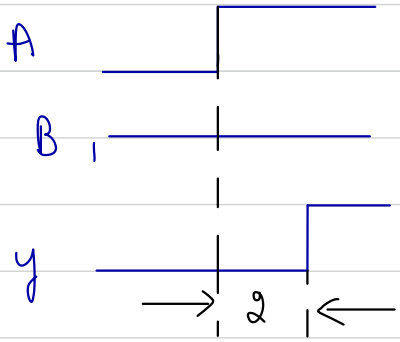
regular op:

inertial delay op:



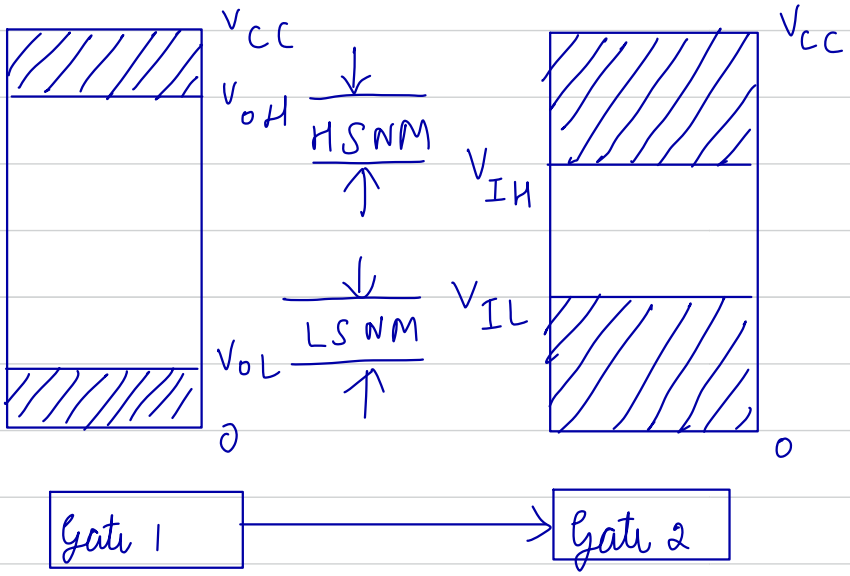
2> Transport delay:





Noise margin

Maximum noise voltage that can get added to input signal that does not cause an undesirable change in the output



HSNM - High state noise margin
LSNM - Low state noise margin

V_{IL} : Maximum input voltage recognised by gate as logic 0.

V_{IH} : Minimum input voltage recognised by gate as logic 1.

V_{OL} : Maximum voltage available at gate output corresponding to logic '0'

V_{OH} : Minimum voltage available at gate output corresponding to logic 1.

$NM_L = LSNM = V_{IL} - V_{OL}$

$NM_H = HSNM = V_{OH} - V_{IH}$

Logic family evolution

- Resistor Transistor logic - RTL
- Diode transistor logic - DTL
- Transistor transistor logic - TTL

→ 74 series - normal application (0-70°C)

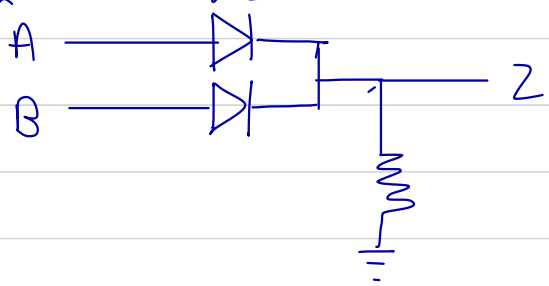
→ 54 series - military application (-55°C - 125°C)

→ LS - low power schottky, AS - advanced schottky, ALS - Advanced low power schottky

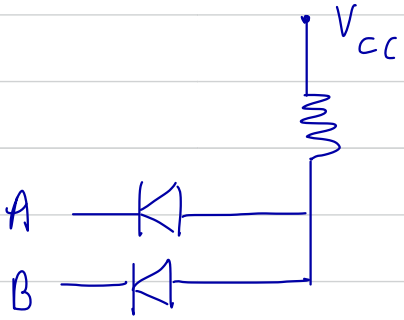
- MOS - metal oxide semiconductor

CMOS, HC - high speed, HCT → TTL compatible high speed, AC, ACT, BiCMOS

1) Diode logic



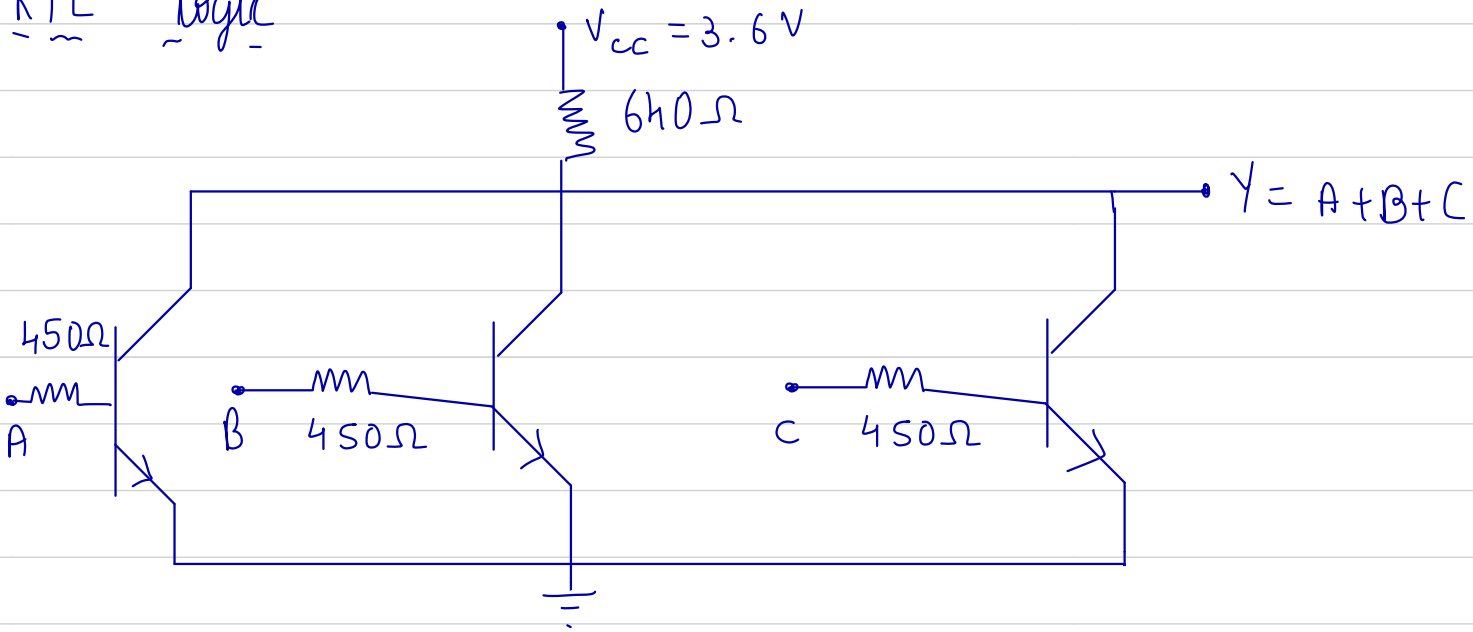
$Z = A + B$



$Z = A \cdot B$

→ Diode as switch
→ Simplest logic
→ Problem while cascading (voltage division)
→ Inversion not possible - needs active element.

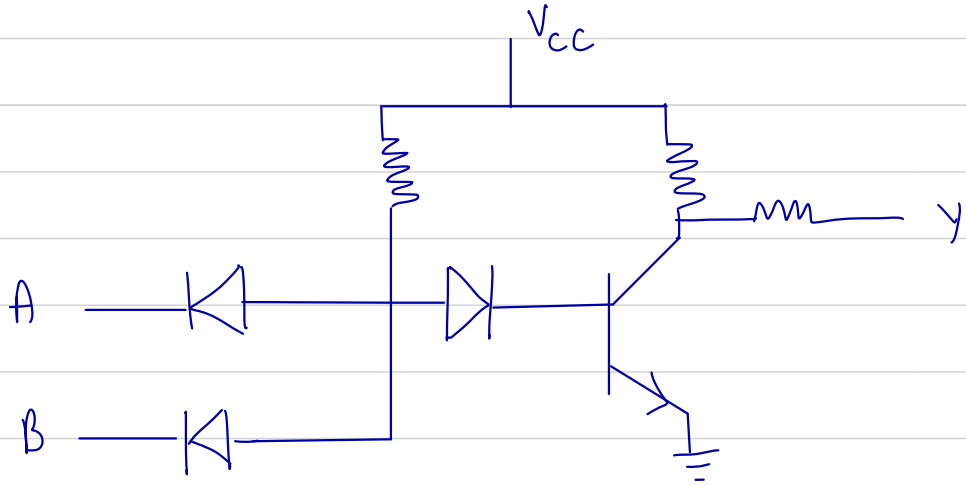
2) RTL logic



Voltage levels : 0.2 V (0) & 3.6 V (1)
Fanout < 5

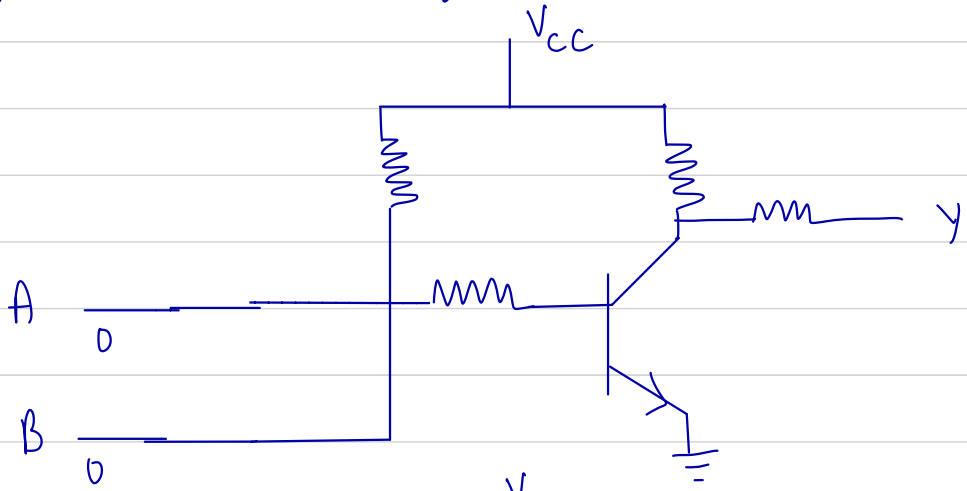
$P_D = 12\ mV$ $t_p = 25\ n$

3> DTL logic

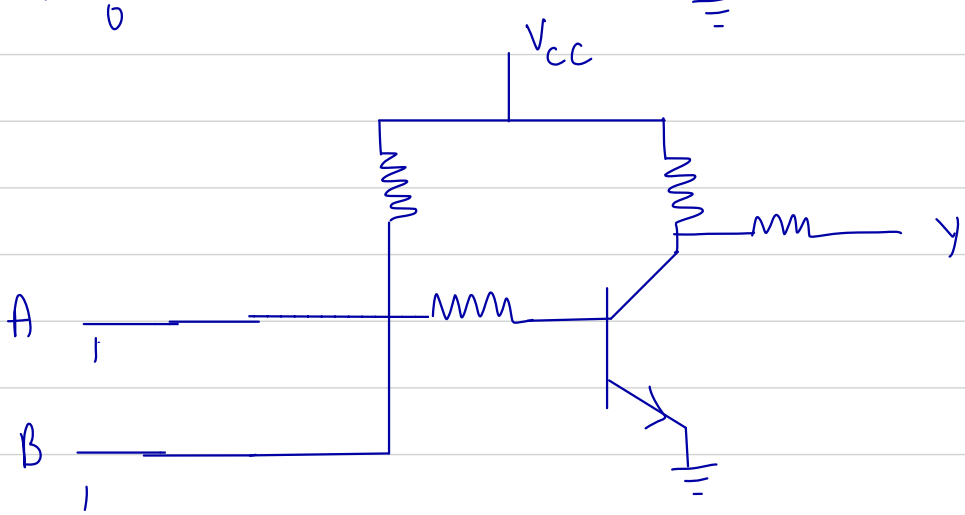


$Y = \overline{A \cdot B}$

- Fanout depends on base current of transistor
- I_{CL} - plays role on fanout (as no current flows through diode when RB ($V_o = 1$)

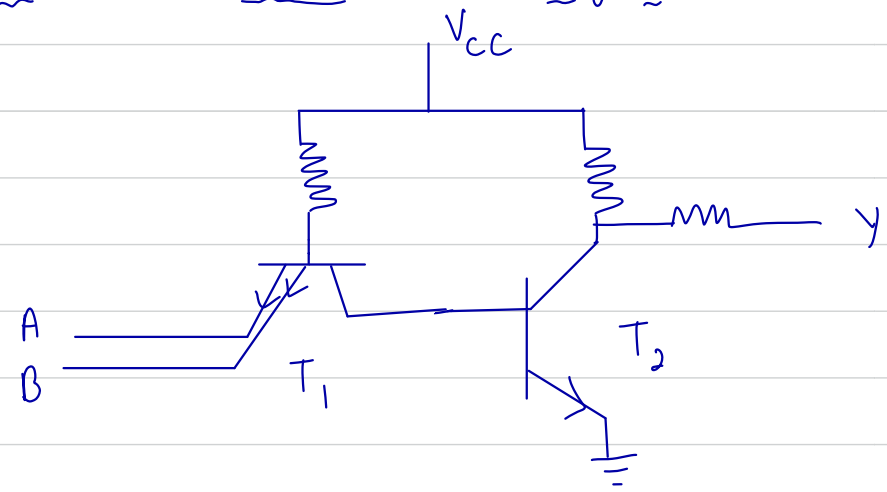


T - off
$Y = 1$



T - on
$Y = 0$

4> Transistor Transistor logic



$Y = \overline{A \cdot B}$

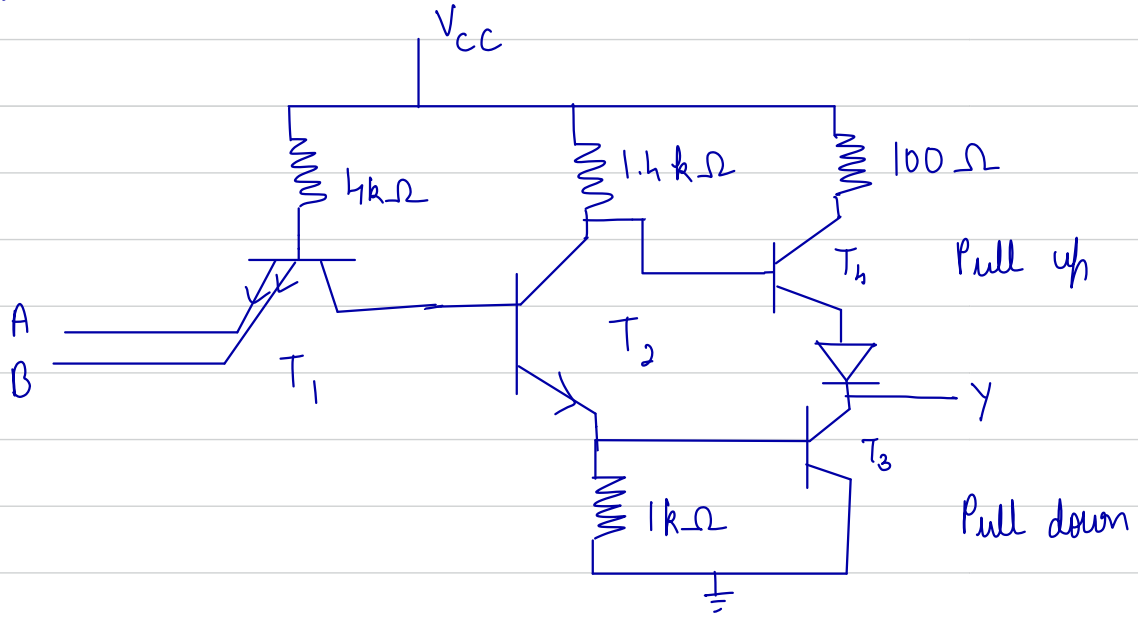
- Replace diode with multi-emitter transistor
- common collector junction & different emitter junction.

→ when A = 0, B = 0, base emitter junction of T_1 = forward bias, T_2 is off (current cannot flow out of base), Y = 1

→ When A = 1 B = 1, Base - emitter junction of T_1 is reverse biased (reverse active mode), T_2 = on, Y = 0

- I_C of T_1 support base charge removal of T_2.
- T_2 off - low R_C - to charge load capacitance faster.
- T_2 on - high R_C - have low $I_{C,sat}$ and reduce P_D
- Use R_C - Totem - pole output.

a) Totem-pole output



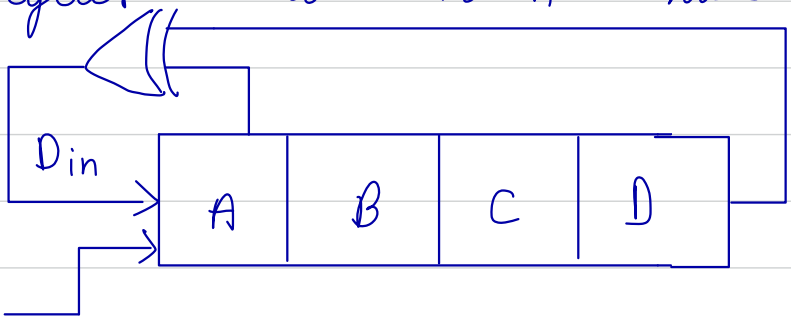
- Phase splitter - either t_3 or t_4 on any time
- A or B = 0 : T_2 off T_4 on
- A and B = 1 : T_2 on T_4 off

b) Schottky TTL (74S00)

- Schottky diode - avoid ringing - avoids input reaching large negative voltage.
- Schottky transistor - no saturation - no charge storage - high speed.
- Active pull down
- Darlington pair - high output current
- Low resistance

Tutorial

Q1> A 4 bit shift register is configured for right shift operation. $D \rightarrow A \rightarrow B \rightarrow C \rightarrow D$ as shown. If the present state ABCD = 1101. The no. of clock cycles to reach state 1111 is ?

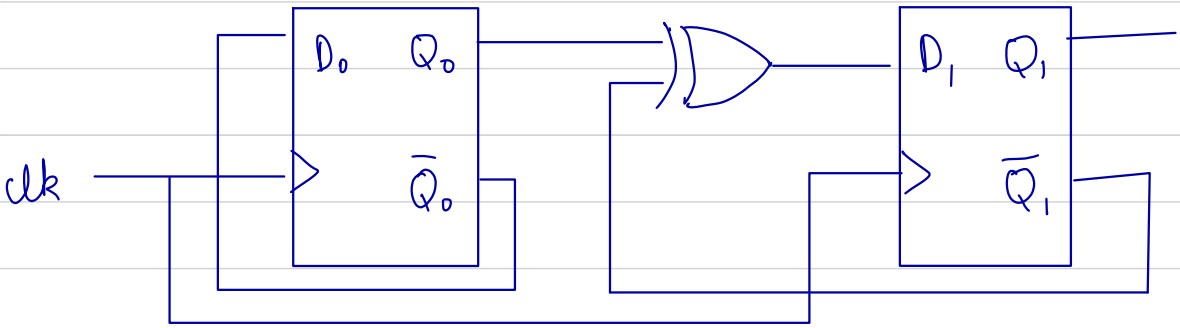


Ans

clk	A	B	C	D
–	1	1	0	1
↑	0	1	1	0
↑	0	0	1	1
↑	1	0	0	1
↑	0	1	0	0
↑	0	0	1	0
↑	0	0	0	1
↑	1	0	0	0
↑	1	1	0	0
↑	1	1	1	0
↑	1	1	1	1

10 clocks.

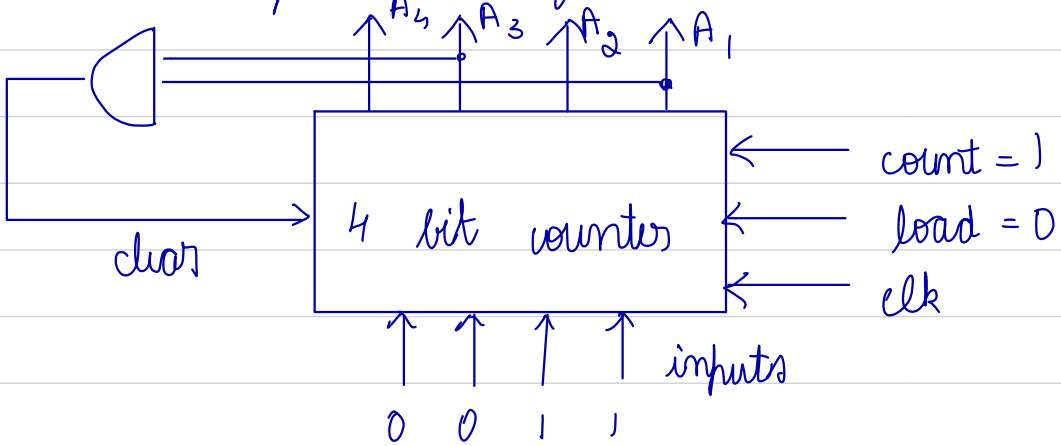
Q2> Find the state transition sequence for 2-bit string $Q_0 Q_1$ if the initial value of $Q_0 Q_1 = 00$



Ans $D_0^+ = \bar{Q}_0$ $\quad D_1^+ = Q_0 \oplus \bar{Q}_1$

clk	Q_0	Q_1	Q_0^+	Q_1^+
↑	0	0	1	1
↑	1	1	0	1
↑	0	1	1	0
↑	1	0	0	0

Q3> Control signal function of a 4 bit binary counter are given below, where x is a don't care. If the gate delays are negligible and if the counter starts at '0'. Find transition sequence cycle.



clr	clk	load	count	Function
1	X	X	X	clear to 0
0	X	0	0	No change
0	↑	1	X	load input
0	↑	0	1	count next

Ans 0 → 1 → 2 → 3 → 4 → 0

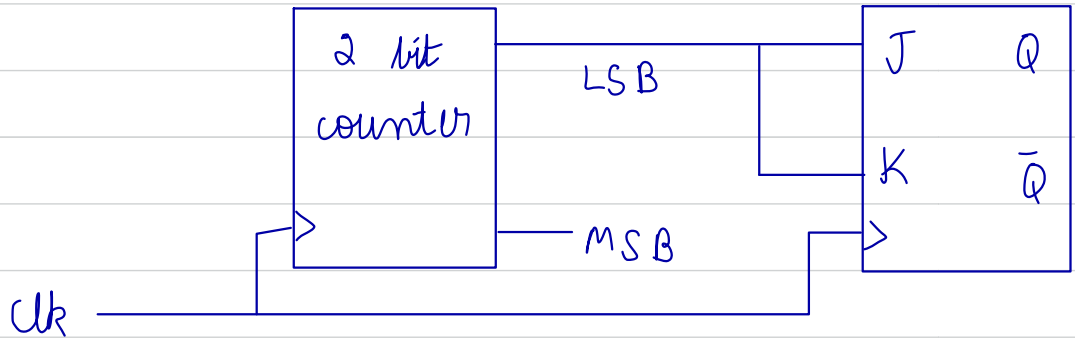
Q4> 4 bit johnson counter with an initial value of 0 is implemented. Find the sequence of counting.

Ans

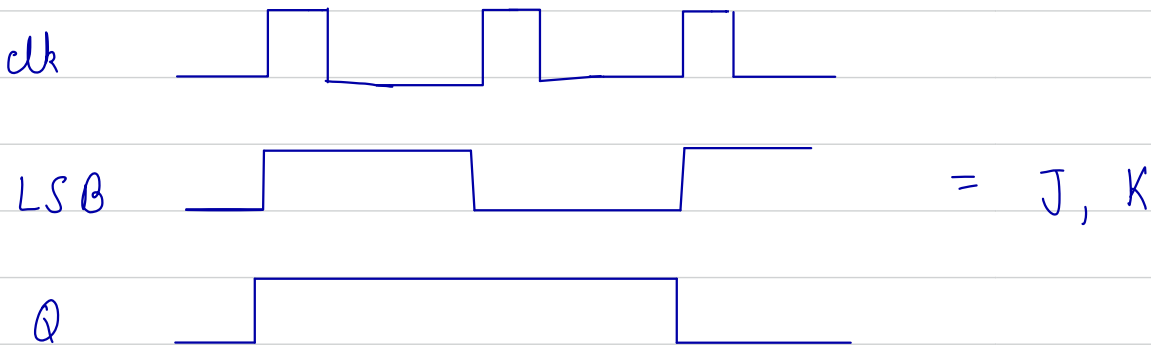
clk	count
–	0 0 0 0
↑	1 0 0 0
↑	1 1 0 0
↑	1 1 1 0
↑	1 1 1 1
↑	0 1 1 1
↑	0 0 1 1
↑	0 0 0 1
↑	0 0 0 0

} 8 counts

Q5> For the circuit shown, clock frequency is f_0 and duty cycle is 25%. For the signal at output Q of flipflop, find frequency and duty cycle.



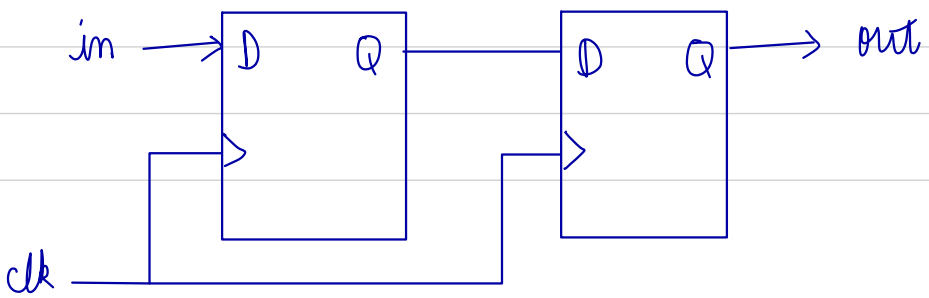
Ans



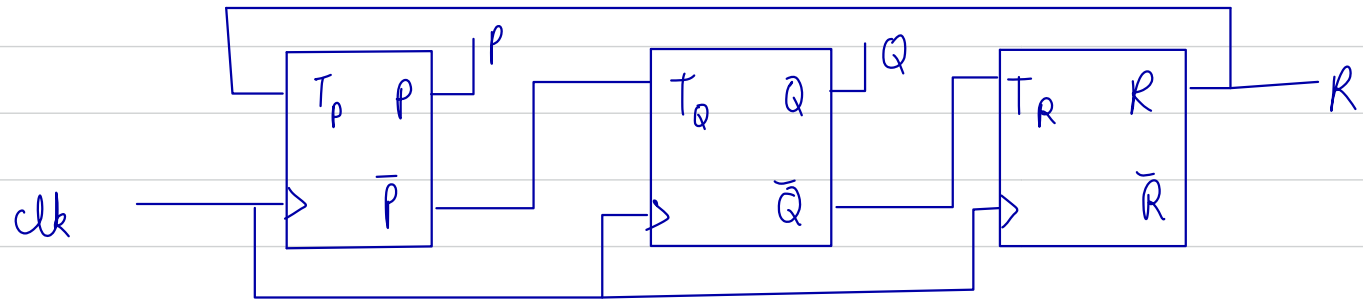
$\therefore f_Q = \frac{f_0}{4}$ // Duty cycle = 50% //

Duty cycle = $\frac{T_{on}}{T_{cycle}}$

Q6> Consider a sequential circuit shown in figure where both flipflops are positive edge triggered DFF, the number of states in state transition diagram of this circuit that have a transition back to the same state on some value of "in" is



Q7> Consider 3 bit counter implemented using T-flipflops assuming PQR at an initial state of 000, what are the next 3 states?



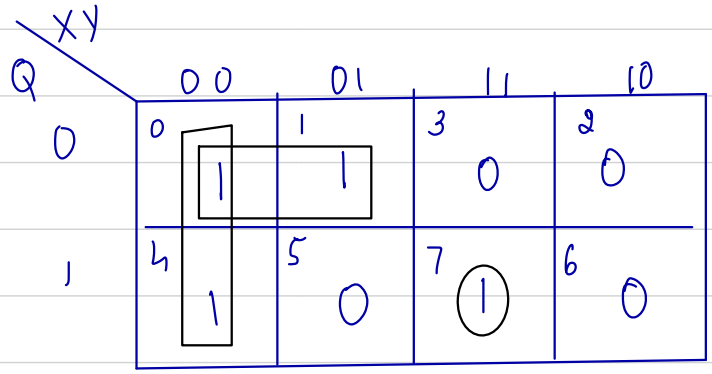
Q8> A new flipflop with inputs X & Y has the following property. Find the expression for next state in terms of X, Y & current state.

Ans

X	Y	current state	Next state
0	0	Q	1
0	1	Q	$\bar{Q}$
1	0	Q	0
1	1	Q	Q

Ans

Q	X	Y	Q^+
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



$Q^+ = \bar{X}\bar{Y} + \bar{Q}\bar{X} + QXY$

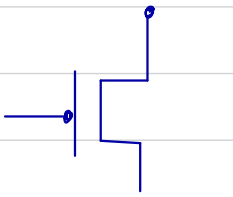
7A>

clk	P	Q	R
–	0	0	0
↑	0	1	1
↑	1	0	1
↑	0	0	0

6A>

5> CMOS logic

PMOS



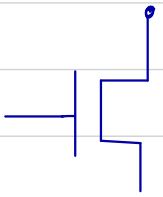
G = 0

closed

G = 1

open

NMOS



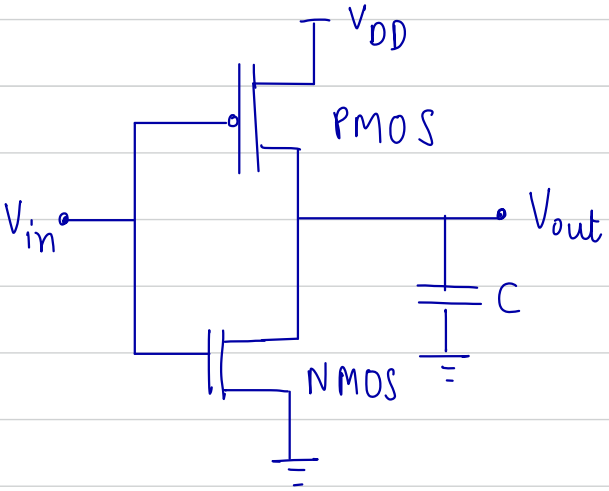
G = 0

open

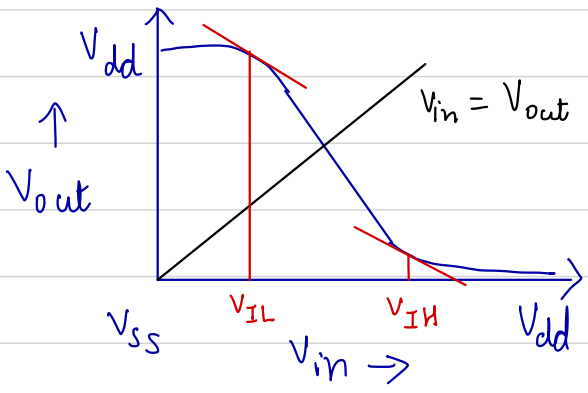
G = 1

closed

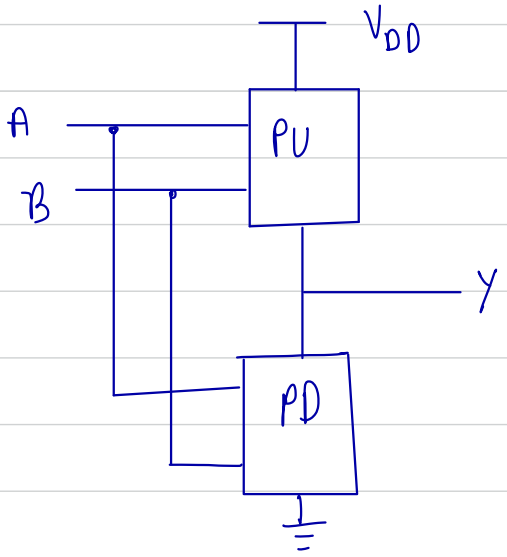
Inverter using CMOS



Transfer characteristics

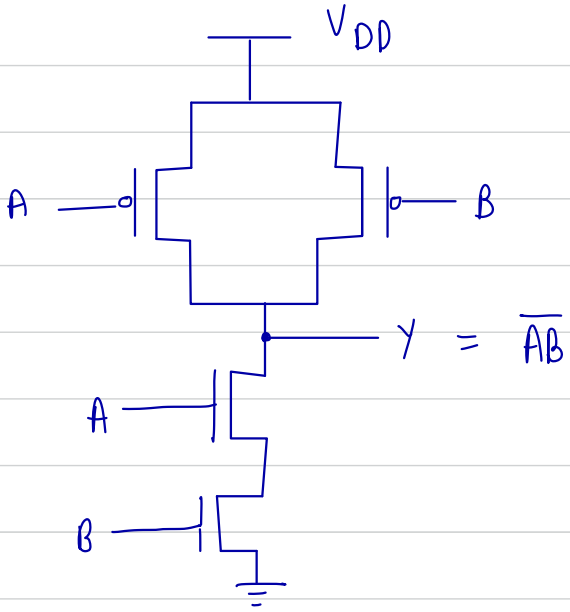


General structure of gates using CMOS

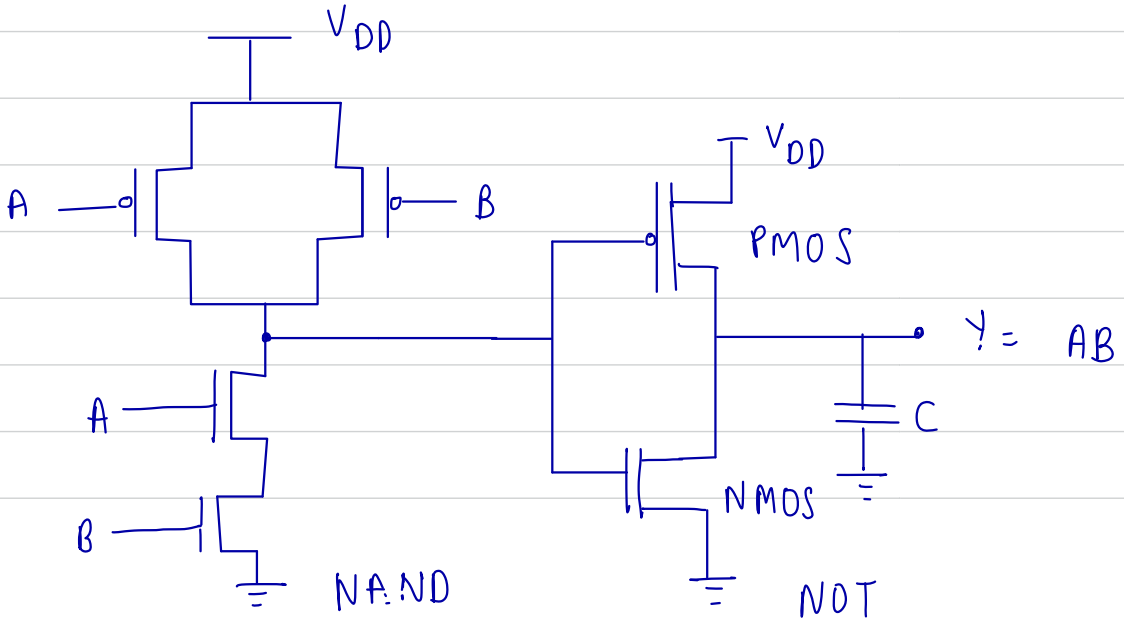


PU → Pull up logic
PD → Pull down logic

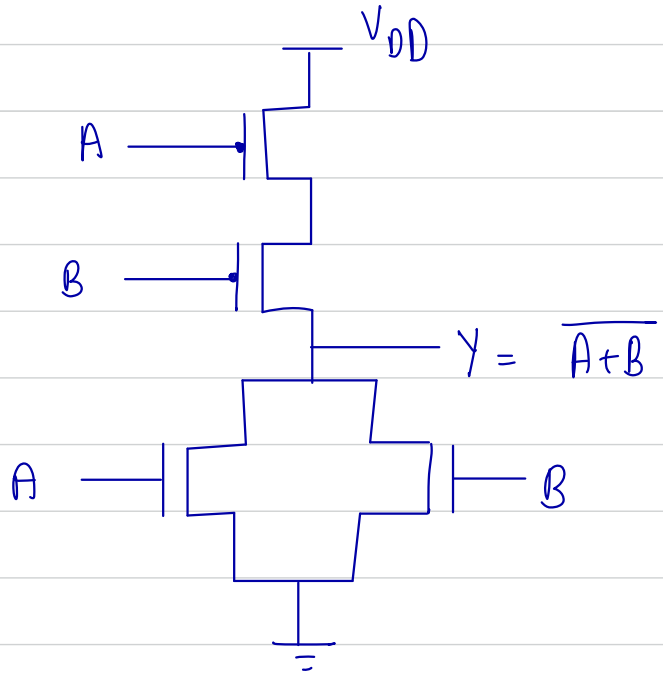
NAND:



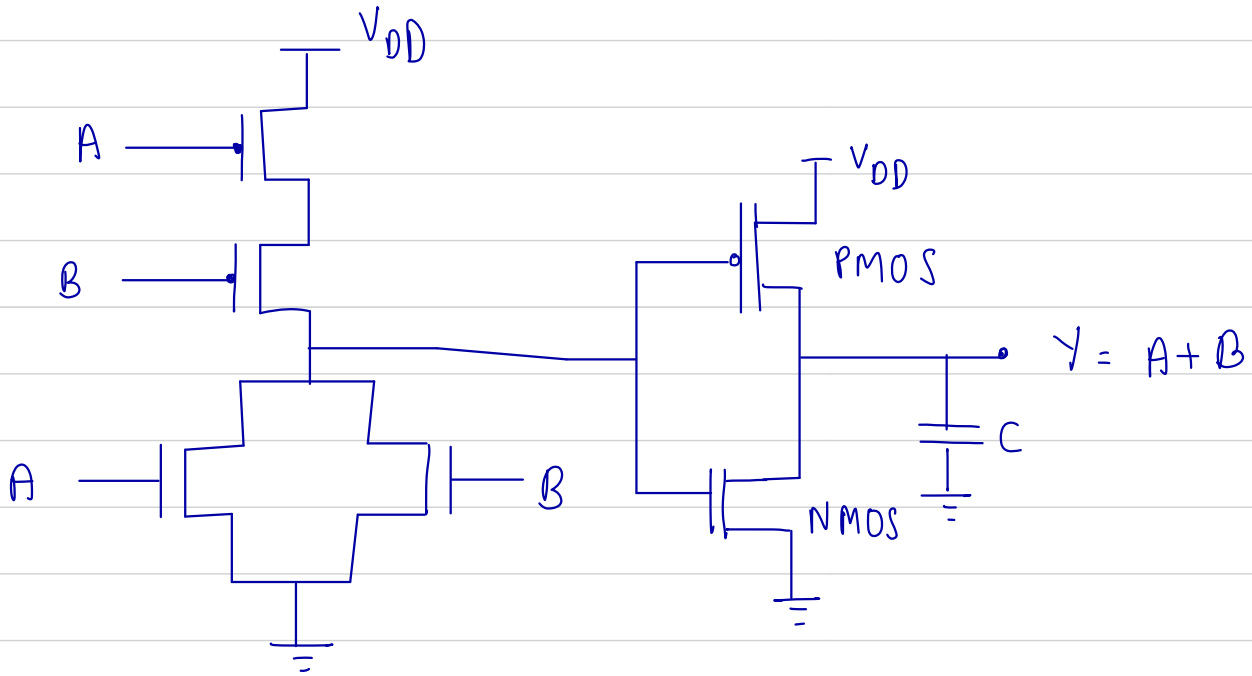
AND:



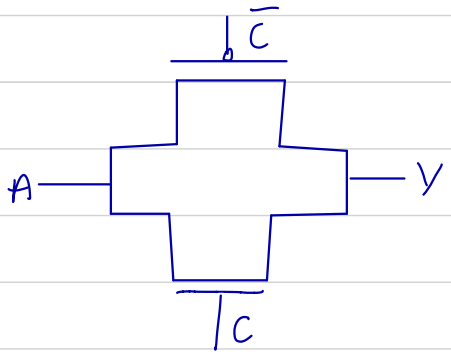
NOR gate:



OR gate:



Pass transistor logic (Transmission gate logic)

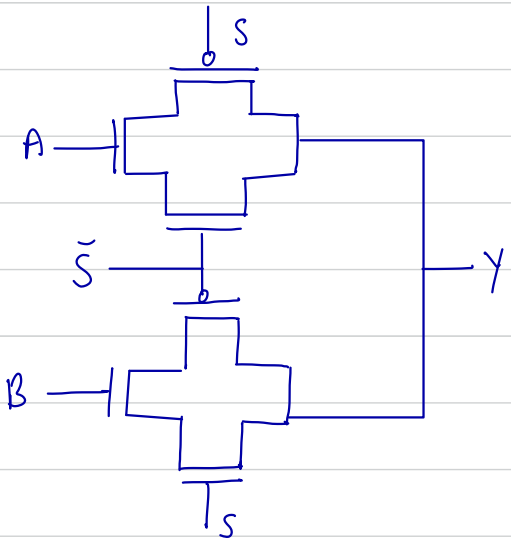


$Y = A$ if $C = 1$

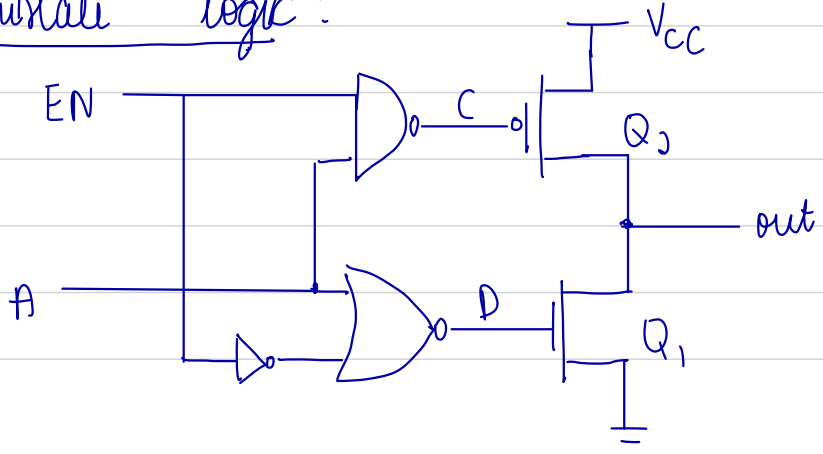
In transmission logic less number of gates are required.

MUX using transmission logic:

$\bar{S}A + SB$



Tristate logic:



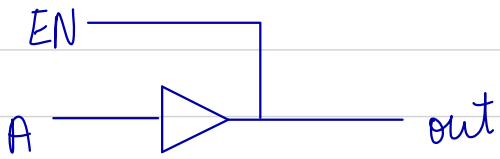
Z → high impedance state

EN = 0 C = 1 D = 0

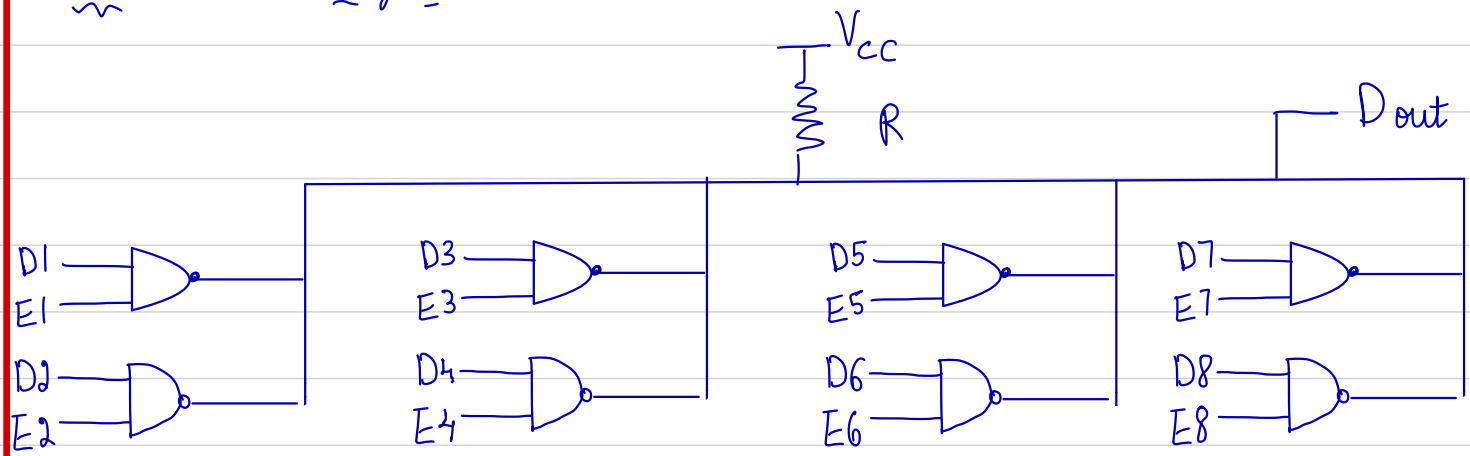
out = Z

EN = 1 C = $\bar{A}$ D = $\bar{A}$

out = A

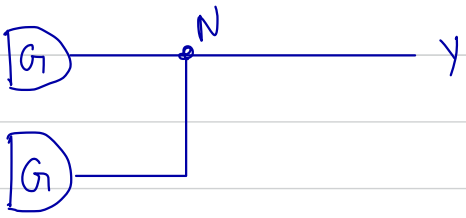


Wired logic



D - Data
E - Enable

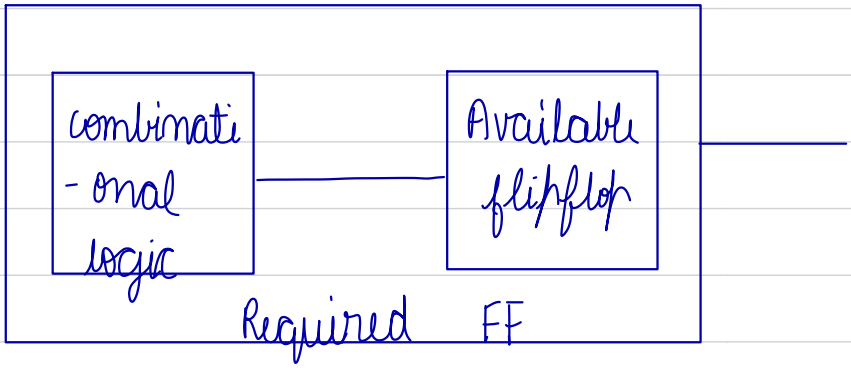
NOTE: 1>



Usually node N acts like AND gate, without use of another gate.

2> Only few logic families allow use of wired logic & high impedance state Z.

Flipflop conversion

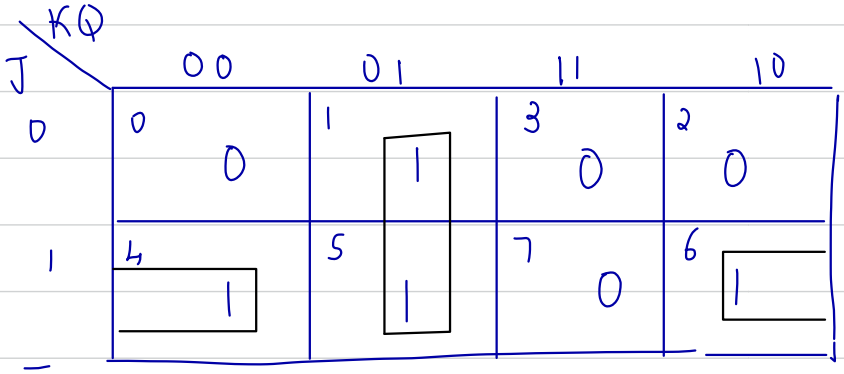


Q) Convert DFF to JK FF

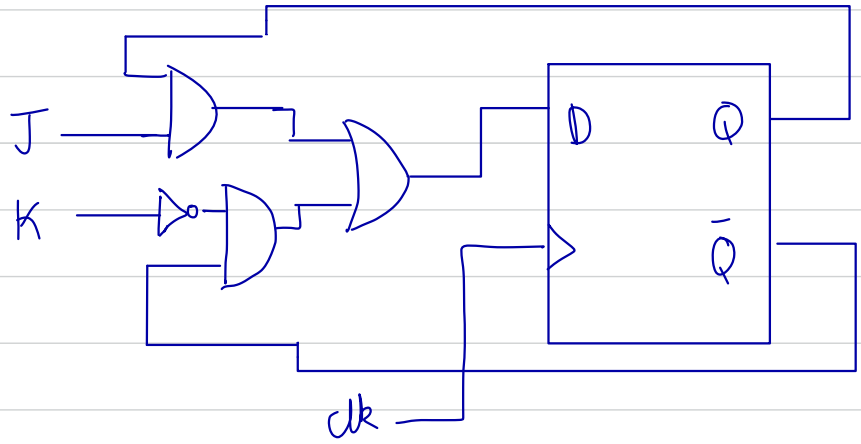
Ans

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

D:

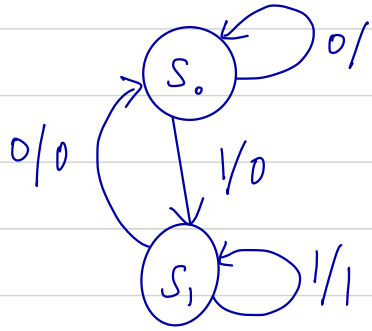


$D = J\bar{Q} + \bar{K}Q$



FSM implementation in Verilog:

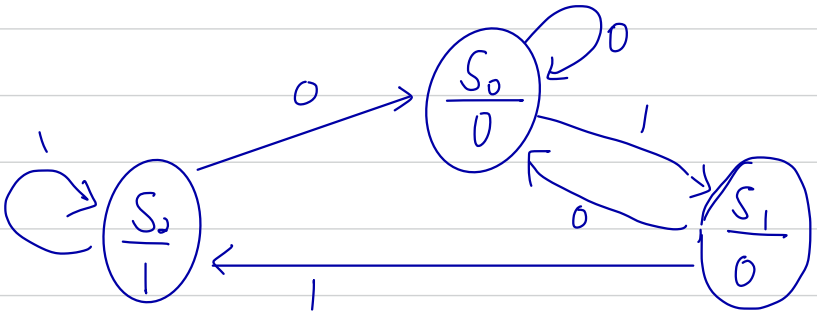
1>




```
module FSM (rst, clk, x, y);
 input rst, clk, x;
 output y;
 Parameter S0=0, S1=1;
 reg op;
 reg state, nxtstate;
 assign y=op;
 always @(posedge clk)
   if (rst)
    state <= S0;
   else state <= nxtstate;
```

```
always @ (state or x)
 begin
        case (state)
          S0: begin    op=x;
              if (x) nxtstate = S1;  end
          S1: begin
                  if (x)  begin
                          nxtstate = S1;
                            op = 1;   end
              else  begin
                          nxtstate = S0;
                           op = 0;   end
              end
endcase
end
endmodule
```

Moore machine implementation:




```
module FSM (rst, clk, x, y)
 input rst, clk, x;
 output y;
```

```
Parameter S0=0, S1=1, S2=2;
reg op;
reg [1:0] state; reg [1:0] nxtstate;
assign y=op;
always @(posedge clk or rst) begin
    if (rst) state <= S0;
    else state <= nxtstate;
always @(x or state) begin
  case (state)
S0: if (x) nxtstate <= S1;
    else nxtstate <= S0;
S1: if (x) nxtstate <= S2;
    else nxtstate <= S0;
S2: if (x) nxtstate <= S2;
    else nxtstate <= S0;
endcase
always @(state)
    case (state)
      S0: op=0;
      S1: op=0;
      S2: op=1;
 endcase
end
end
endmodule
```

Quine - Mccluskey method

Q) $f(A,B,C,D) = \Sigma m(0,1,2,3,5,7,8,10,12,13,15)$

Value	A	B	C	D	No of 1s	Grouping 1:		
0	0	0	0	0	0			
1	0	0	0	1		(0,1)	000_	
2	0	0	1	0	1	(0,2)	00_0	
8	1	0	0	0		(0,8)	_000	
3	0	0	1	1		(1,3)	00_1	(2,3) 001_
5	0	1	0	1	2	(1,5)	0_01	
10	1	0	1	0		(2,10)	_010	(8,10) 10_0
12	1	1	0	0		(8,12)	1_00 → $A\bar{C}\bar{D}$	
7	0	1	1	1		(3,7)	0_11	(5,7) 01_1
13	1	1	0	1	3	(5,13)	_101	(12,13) 110_
15	1	1	1	1	4	(7,15)	_111	(13,15) 11_1

→ $AB\bar{C}$

Grouping 2:

(0,1,2,3): 00__	$\bar{A}\bar{B}$
(0,2,1,3): 00__	
(0,2,8,10): _0_0	$\bar{B}\bar{D}$
(0,8,2,10): _0_0	
(1,3,5,7): 0__1	$\bar{A}D$
(1,5,3,7): 0__1	
(5,7,13,15): _1_1	BD
(5,13,7,15): _1_1	

Prime implicant table:

PI	0	1	2	3	5	7	8	10	12	13	15
$\bar{A}\bar{B}$	X	X	X	X							
$\bar{B}\bar{D}$	X		X				X	(X)			
$\bar{A}D$		X		X	X	X					
$B\bar{D}$					X	X				X	(X) → only 1 in column ⇒ Essential PI
$A\bar{C}\bar{D}$							X		X		
$AB\bar{C}$									X	X	
	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓

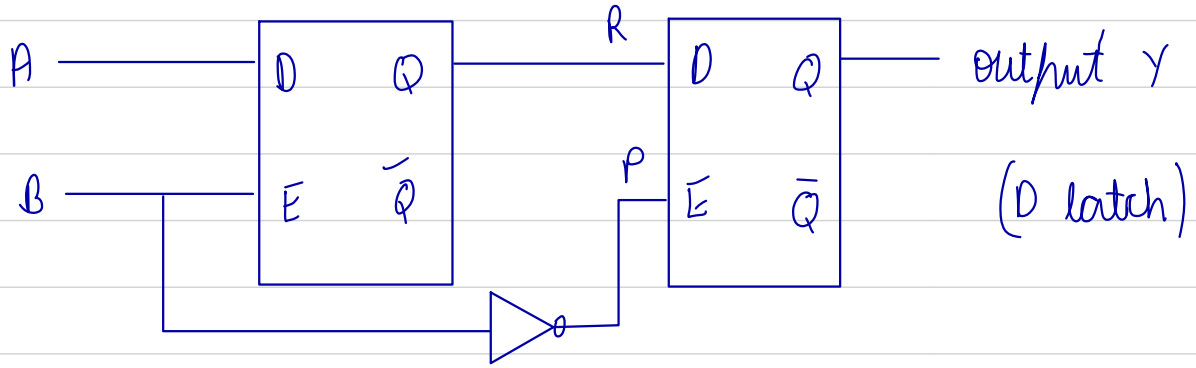
$$f = \bar{B}\bar{D} + BD + \bar{A}\bar{B} + A\bar{C}\bar{D}$$

or

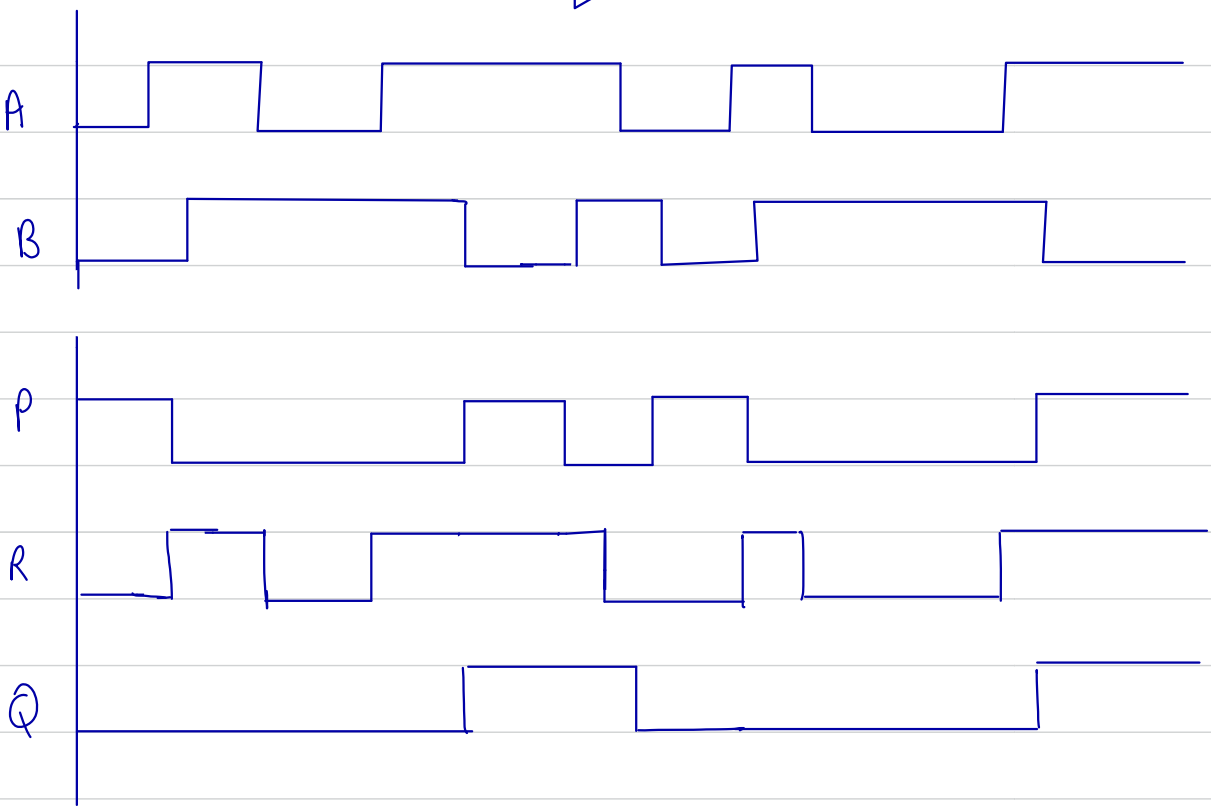
$$f = \bar{B}\bar{D} + BD + \bar{A}D + AB\bar{C}$$

Tutorial :

Q1> Determine the final O/p states over time for following circuit.

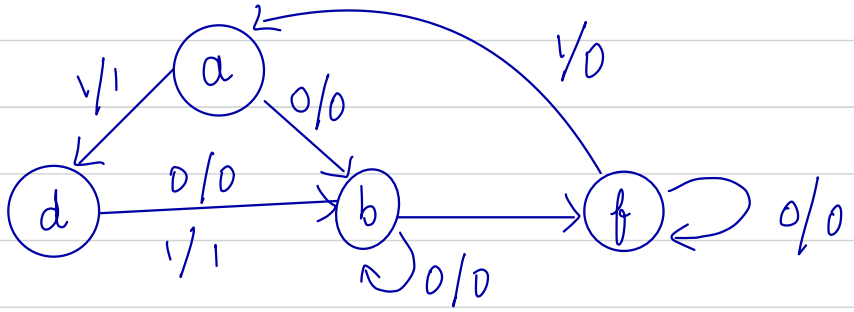


Ans



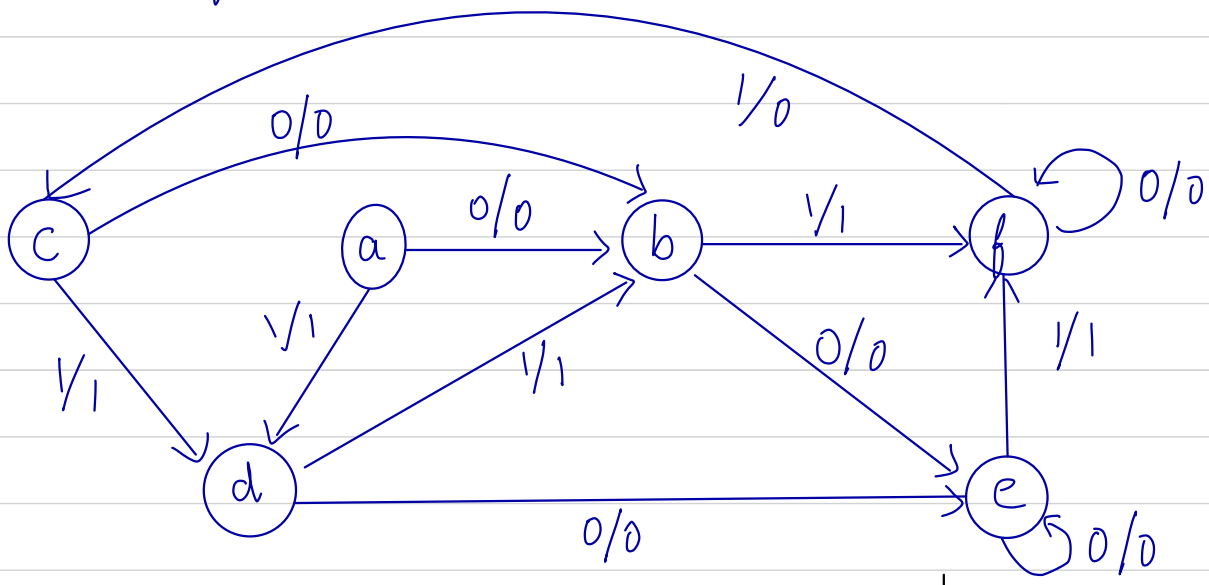
Q2> write state table for:

i>



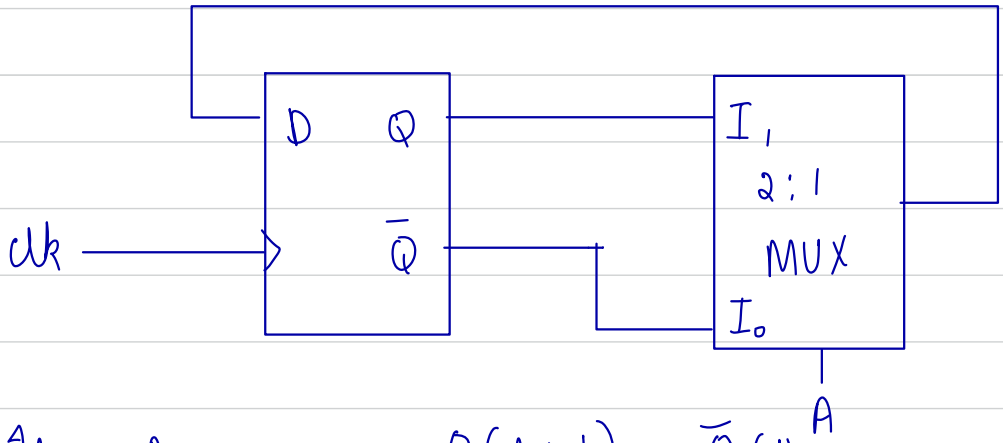
Present state	Next state i/p: 0	Next state 1	output ip: 0	output 1
a	b	d	0	1
b	b	f	0	1
d	b	b	0	1
f	f	a	0	0

ii>



Present state	Next state i/p: 0	Next state 1	output ip: 0	output 1
a	b	d	0	1
b	e	f	0	1
c	b	d	0	1
d	e	b	0	1
e	e	f	0	1
f	f	c	0	0

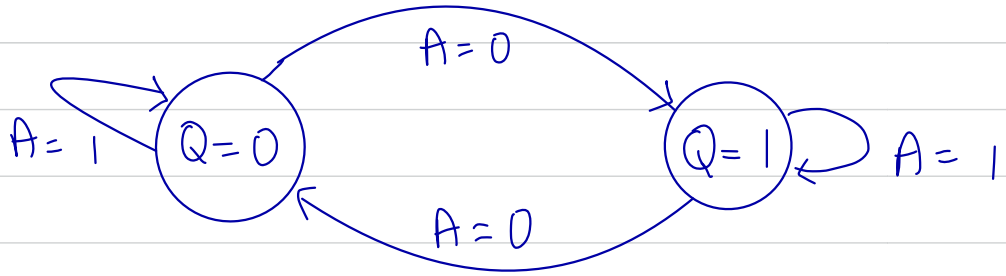
Q3> Draw the state diagram for:



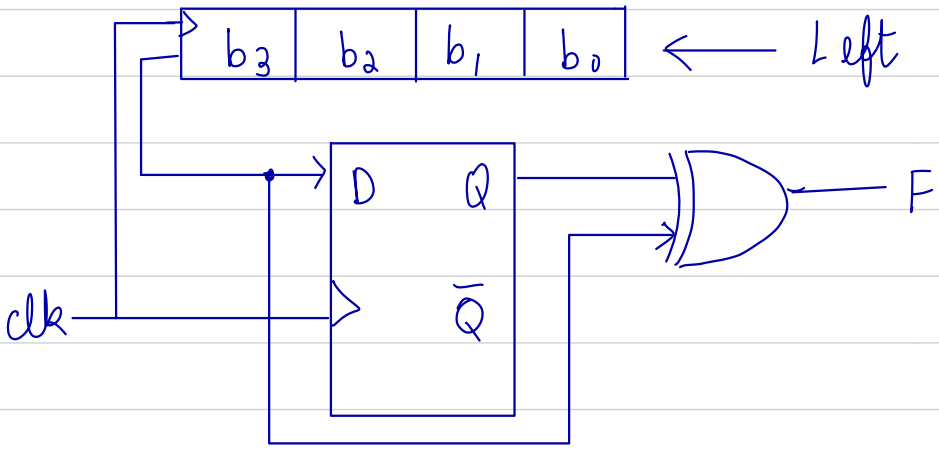
Ans If A = 0, $Q(t+1) = \bar{Q}(t)$

If A = 1, $Q(t+1) = Q(t)$

$\Rightarrow Q(t+1) = \bar{A}\,\bar{Q}(t) + AQ(t)$



Q4> Determine the function of following circuit;
First input to DFF is 0



Ans Initially, $D=0$, $Q=0$

$G_3 = 0 \oplus b_3 = b_3$

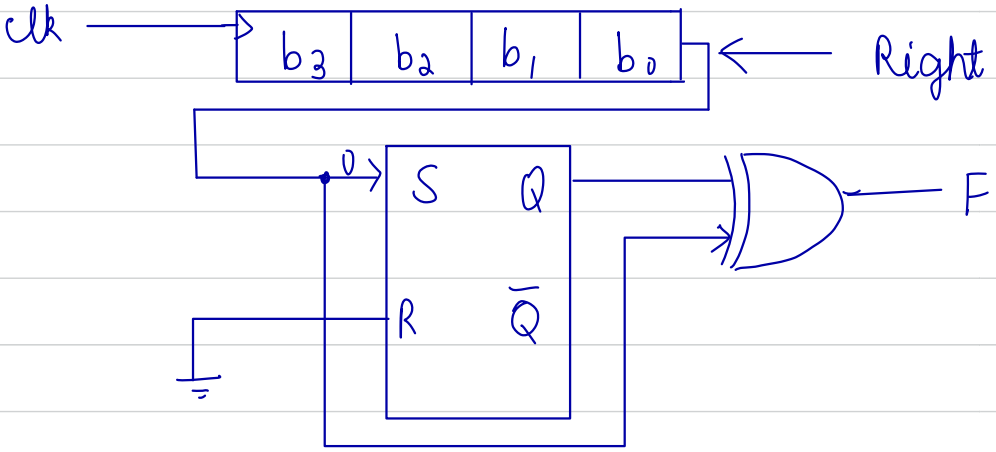
$G_2 = b_3 \oplus b_2$

$G_1 = b_2 \oplus b_1$

$G_0 = b_1 \oplus b_0$

⇒ It is a binary to gray code converter.

Q5> Determine the function of following circuit:



Ans $F_3 = b_0 \oplus 0 = b_0$

$F_2 = b_1 \oplus 0 = b_1$

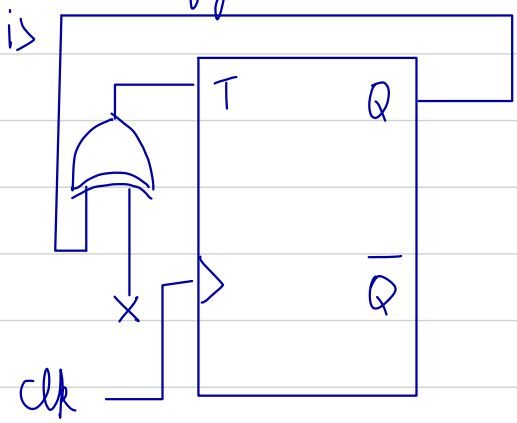
$F_1 = b_2 \oplus 0 = b_2$

$F_0 = b_3 \oplus 0 = b_3$

⇒ Reverses the order of bits.

Q6> Identify the flipflops in following:

i>



Ans: $T = X \oplus Q(t)$

In TFF, $Q(t+1) = T \oplus Q(t)$

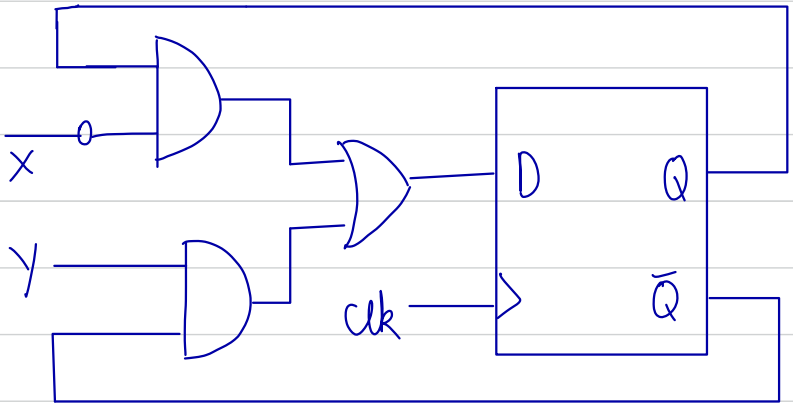
$\Rightarrow Q(t+1) = X \oplus Q(t) \oplus Q(t)$

$= X \oplus 0$

$= X$ //

⇒ D flipflop //

ii)



Ans: $Q(t+1) = \bar{X}Q + Y\bar{Q}$

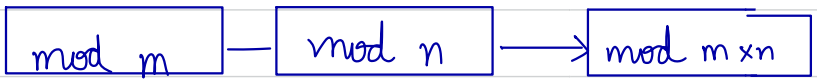
If $Y = J$ & $X = K$

$Q(t+1) = J\bar{Q} + \bar{K}Q$

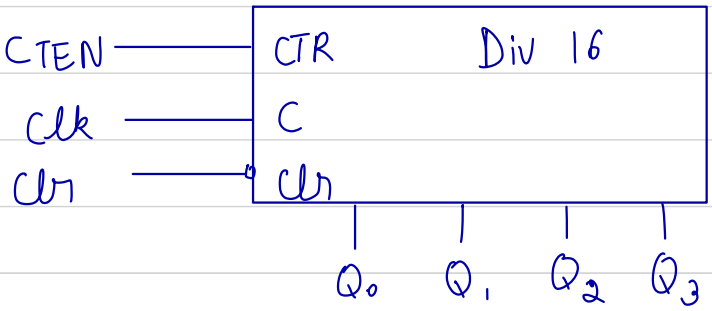
$\Rightarrow$ JK flipflop //

NOTE: If mod M & mod N counters are cascaded then we get mod 'M x N' counter.

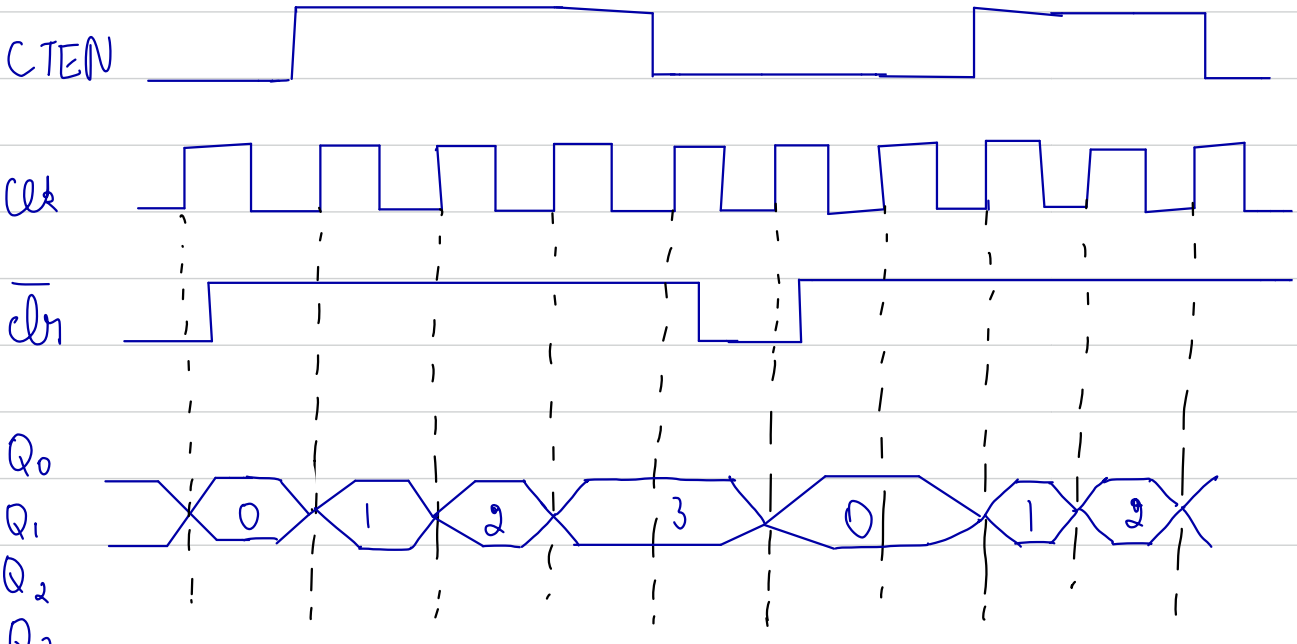
i.e



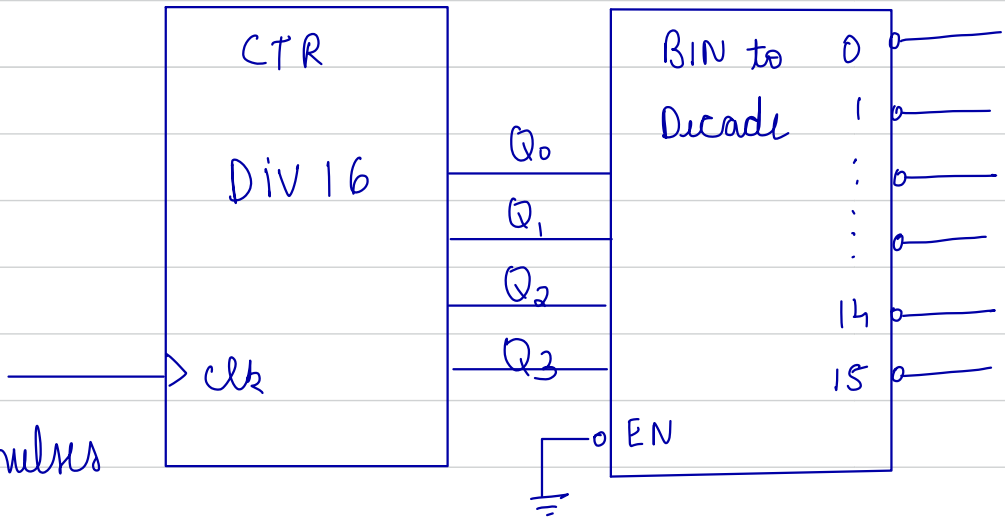
Q7) Find the output waveform



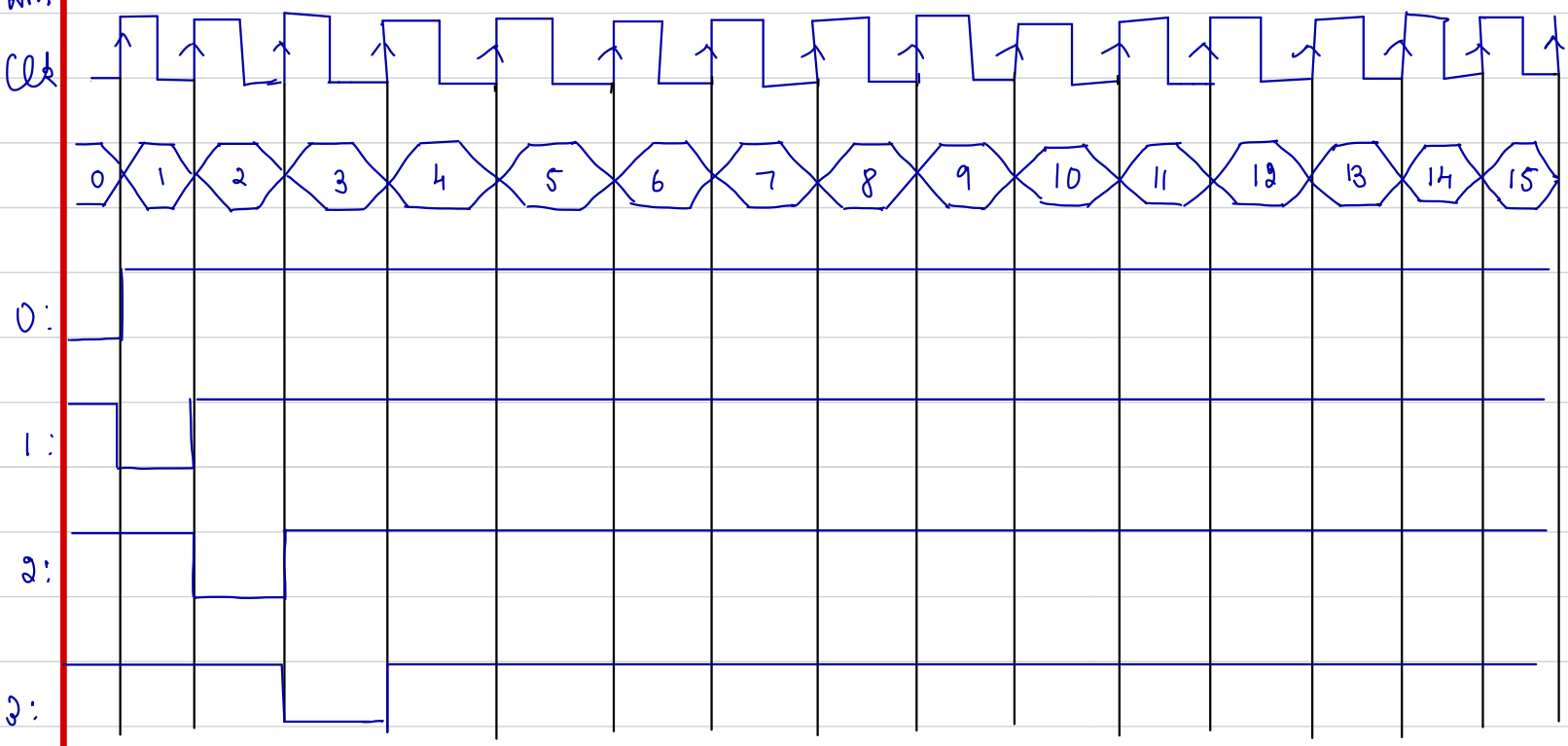
CT EN → count enable



Q8> Determine each of the decoder output waveform

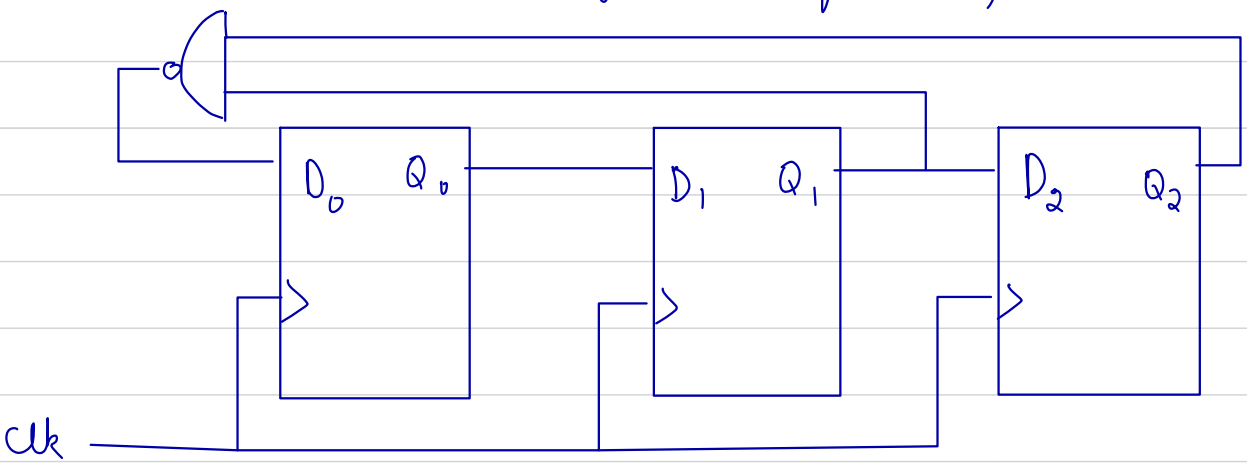


Ans



So on

Q9> Determine the counter sequence, initially at 000

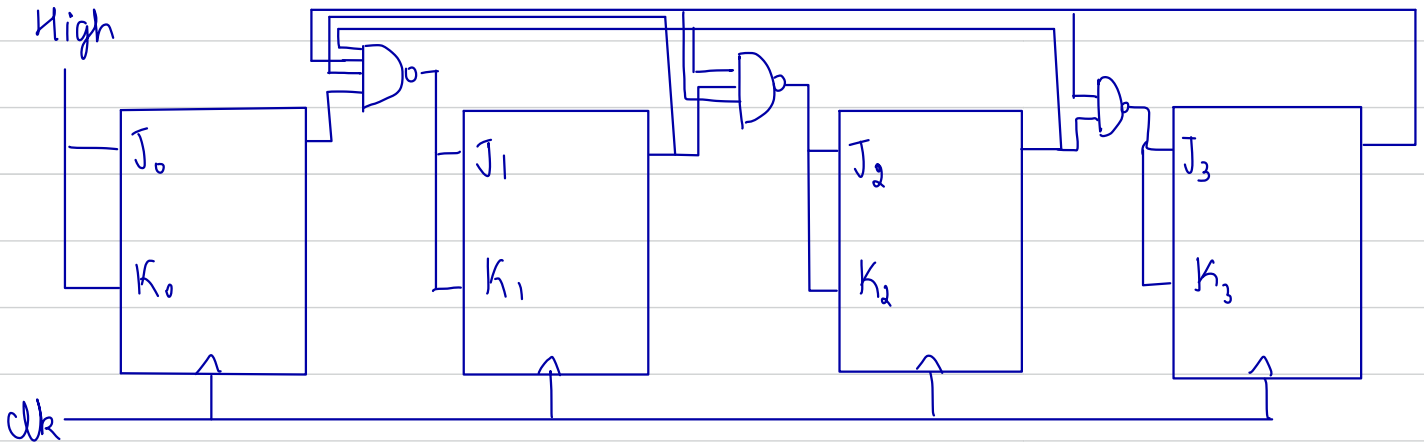


Ans

$$Q_0(t+1) = \overline{Q_2(t) \cdot Q_1(t)}$$
$$Q_1(t+1) = Q_0(t)$$
$$Q_2(t+1) = Q_1(t)$$

$000 \rightarrow 100 \rightarrow 110 \rightarrow 111 \rightarrow 011 \rightarrow 001 \rightarrow 100$

Q10> Determin the counter sequence, initial at 0000



Ans

$$J_0 = K_0 = 1$$
$$Q_0(t+1) = \overline{Q}_0(t)$$
$$J_1 = K_1 = \overline{Q_0 \cdot Q_1 \cdot Q_2 \cdot Q_3}$$
$$J_2 = K_2 = \overline{Q_1 \cdot Q_2 \cdot Q_3}$$
$$J_3 = K_3 = \overline{Q_2 \cdot Q_3}$$

0000 → 1111 → 1110 → 1101 → 1010 → 0101
0100 ← 1011 ← 0100 ←

Q> Use Quine - Mcklusky method to reduce:

$F(w,x,y,z) = \Sigma m(0, 1, 2, 3, 5, 7, 13, 15)$

0	0000 ✓	(0,1)	000_ ✓	(0,1,2,3)	00__ $\bar{w}\bar{x}$
1	0001 ✓	(0,2)	00_0 ✓	(0,2,1,3)	00__
2	0010 ✓	(1,5)	0_01 ✓	(1,5,3,7)	0__1
3	0011 ✓	(1,3)	00_1 ✓	(1,3,5,7)	0__1 $\bar{w}z$
5	0101 ✓	(2,3)	001_ ✓		
7	0111 ✓	(3,7)	0_11 ✓	(5,7,13,15)	_1_1
13	1101 ✓	(5,7)	01_1 ✓	(5,13,7,15)	_1_1 xz
15	1111 ✓	(5,13)	_101 ✓		
		(7,15)	_111 ✓		
		(13,15)	11_1 ✓		

Prime implicant table

PI	0	1	2	3	5	7	13	15
$\bar{w}\bar{x}$	(X)	X	(X)	X				
$\bar{w}z$		X		X	X	X		
xz					X	X	(X)	(X)
	✓	✓	✓	✓	✓	✓	✓	✓

$F = \bar{w}\bar{x} + xz$

Q> Use Quine - Mccluskey method to simplify:

$F(A,B,C,D) = \Sigma m(4,5,6,8,9,10,13) + d\Sigma(0,7,15)$

Ans

d0	0000 ✓	(d0, 4)	0-00	$\bar{A}\bar{C}\bar{D}$	(4, 5, 6, d7)	01-- $\bar{A}B$
4	0100 ✓	(d0, 8)	-000	$\bar{B}\bar{C}\bar{D}$	(4, 6, 5, d7)	01--
8	1000 ✓	(4, 5)	010- ✓		(5, d7, 13, d15)	-1-1 BD
5	0101 ✓	(4, 6)	01-0 ✓		(5, 13, d7, d15)	-1-1
6	0110 ✓	(8, 9)	100-	$A\bar{B}C$		
9	1001 ✓	(8, 10)	10-0	$A\bar{B}\bar{D}$		
10	1010 ✓	(5, d7)	01-1 ✓			
d7	0111 ✓	(5, 13)	-101 ✓			
13	1101 ✓	(6, d7)	011- ✓			
d15	1111 ✓	(9, 13)	1-01	$A\bar{C}D$		
		(d7, d15)	-111 ✓			
		(13, d15)	11-1 ✓			

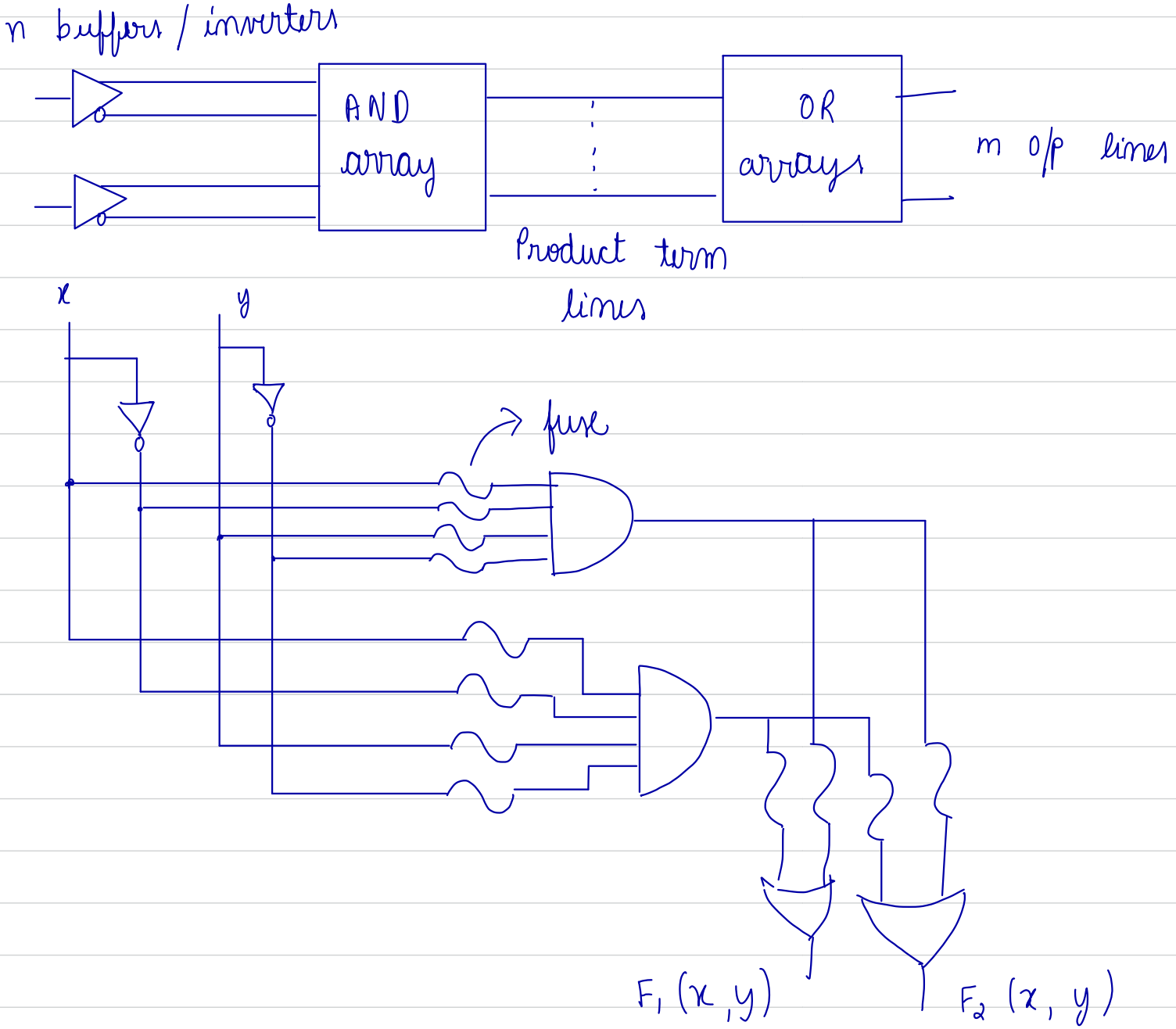
Prime implicant table

PI	4	5	6	8	9	10	13
$\bar{A}B$	✓	✓	(✓)				
BD		✓					✓
$\bar{A}\bar{C}\bar{D}$	✓						
$\bar{B}\bar{C}\bar{D}$				✓			
$A\bar{B}C$				✓	✓		
$A\bar{B}\bar{D}$				✓		(✓)	
$A\bar{C}D$					✓		✓
	✓	✓	✓	✓	✓	✓	✓

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

Programmable Logic devices (PLD)

→ It contains a large number of gates, flipflops and used to perform a particular function.



Blowing these fuses along the path that must be removed in order to obtain

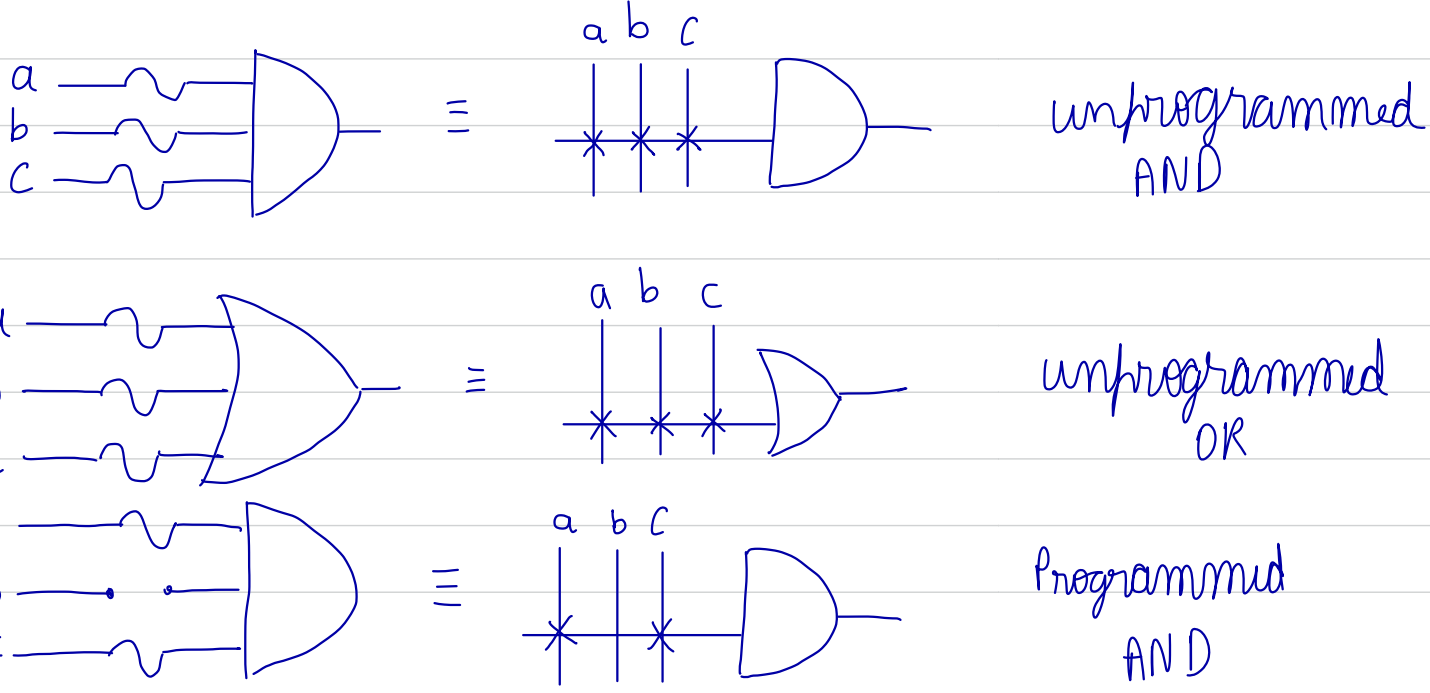
the particular configuration.

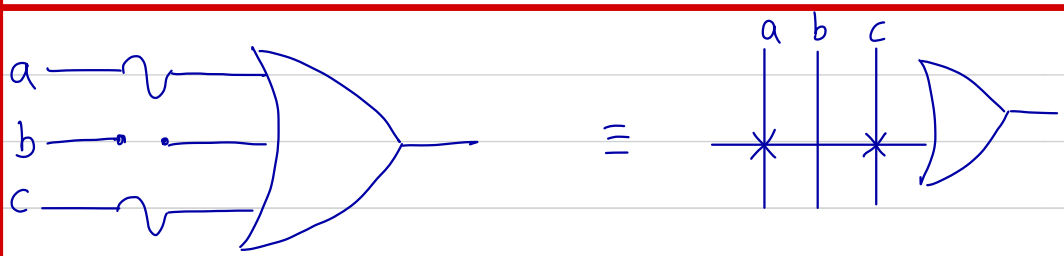
Advantages: Acquires less board space, faster, lower power requirements (i.e smaller power supplies), less costly assembly processes, higher reliability (fewer ICs & circuit connections means easier troubleshooting)

Device	AND array	OR array
PROM	–	Programmable
PAL	Programmable	–
PLA	Programmable	Programmable

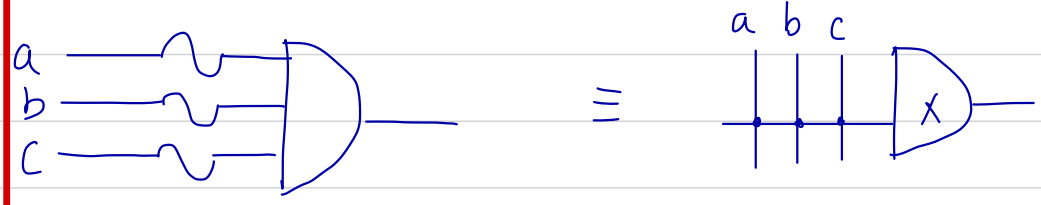
PAL - Programmable array logic
PLA - Programmable logic array

PAL16L8BNC → IC

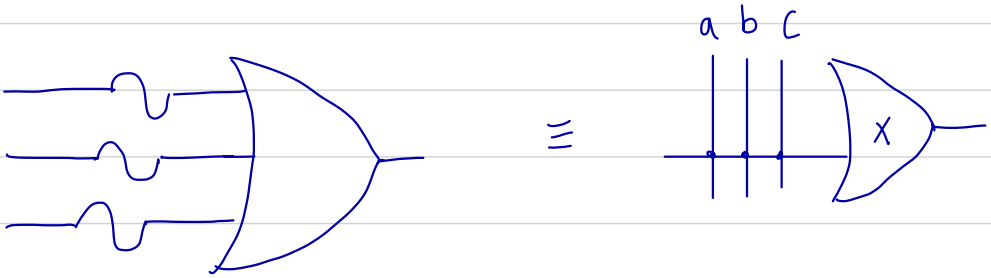




Programmed OR



Programmed AND



Programmed OR

ROM :

		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

PROM :

full adder

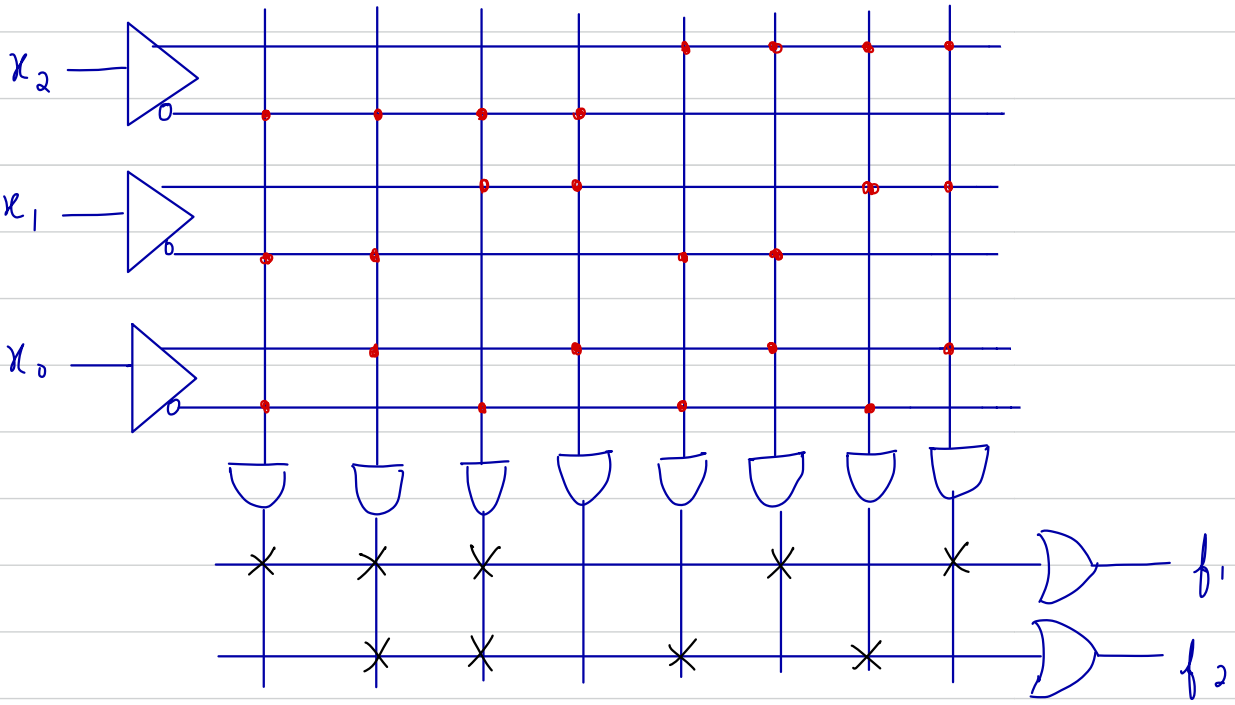
OR → Programmable

Address is fixed

3-8 decoder

Q> $f_1(x_2, x_1, x_0) = \Sigma m(0, 1, 2, 5, 7)$ using PROM

$f_2(x_2, x_1, x_0) = \Sigma m(1, 2, 4, 6)$



NOTE: • → Non programmable

X → Programmable

Always - synthesizable Initial - not

NOTE: Always → is a procedural construct
Forever → is a loop statement
for, while, repeat are also loop. (in verilog)

Size of PROM

No. of inputs - m bits
No. of outputs - n bits

∴ Size of PROM = $2^m \times n$

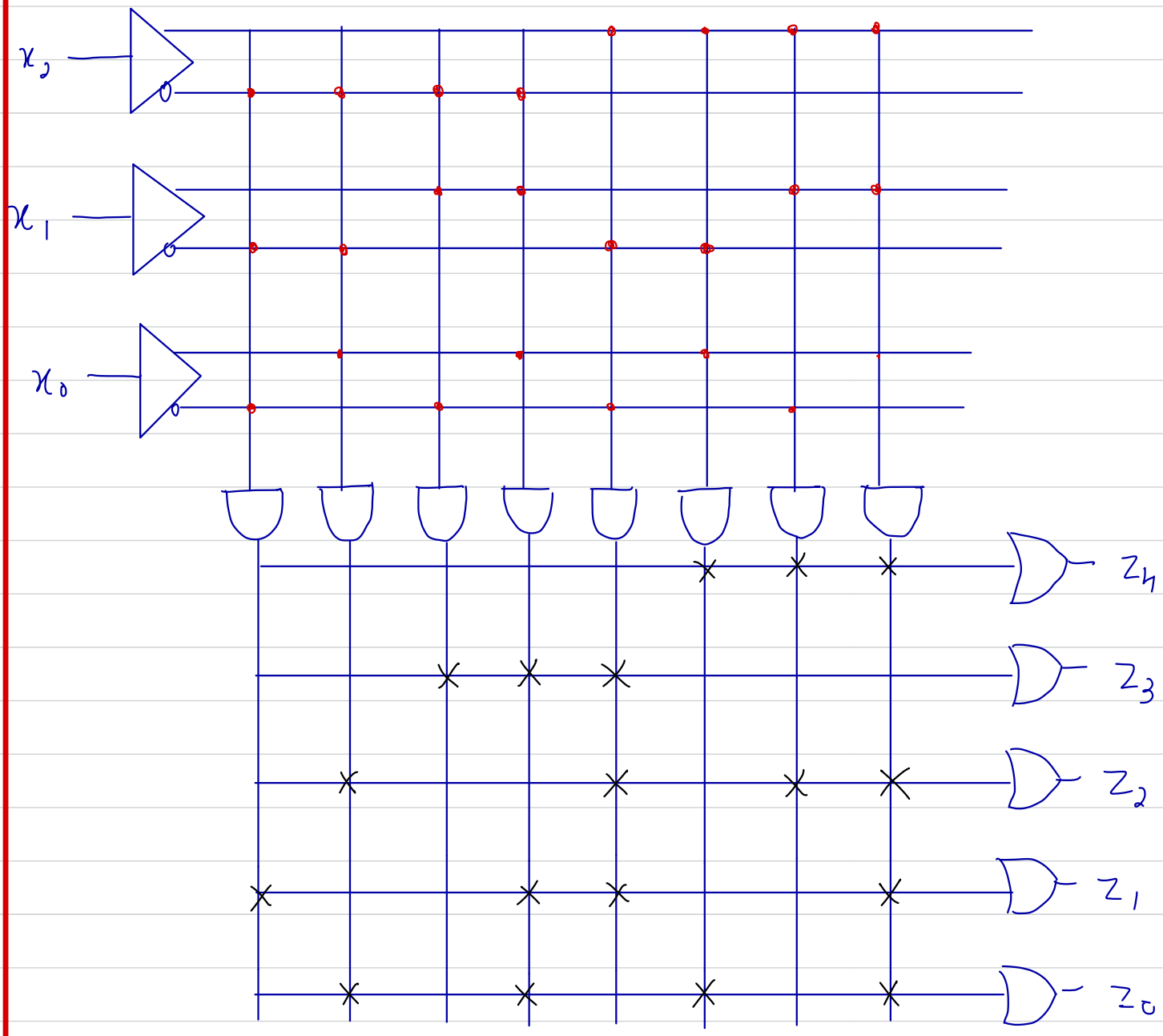
Eg1: $F(x) = 3x + 2$. Draw PROM:

ip - 3 bit
op - 5 bit

x	o/p
0	2
1	5
2	8
3	11
4	14
5	17
6	20
7	23

x_2	x_1	x_0	z_4	z_3	z_2	z_1	z_0
0	0	0	0	0	0	1	0
0	0	1	0	0	1	0	1
0	1	0	0	1	0	0	0
0	1	1	0	1	0	1	1
1	0	0	0	1	1	1	0
1	0	1	1	0	0	0	1
1	1	0	1	0	1	0	0
1	1	1	1	0	1	1	1

z_4 - 5, 6, 7
z_3 - 2, 3, 4
z_2 - 1, 4, 6, 7
z_1 - 0, 3, 4, 7
z_0 - 1, 3, 5, 7



Ej2: PAL: $f_1 = \Sigma m(1, 2, 4, 5, 7)$

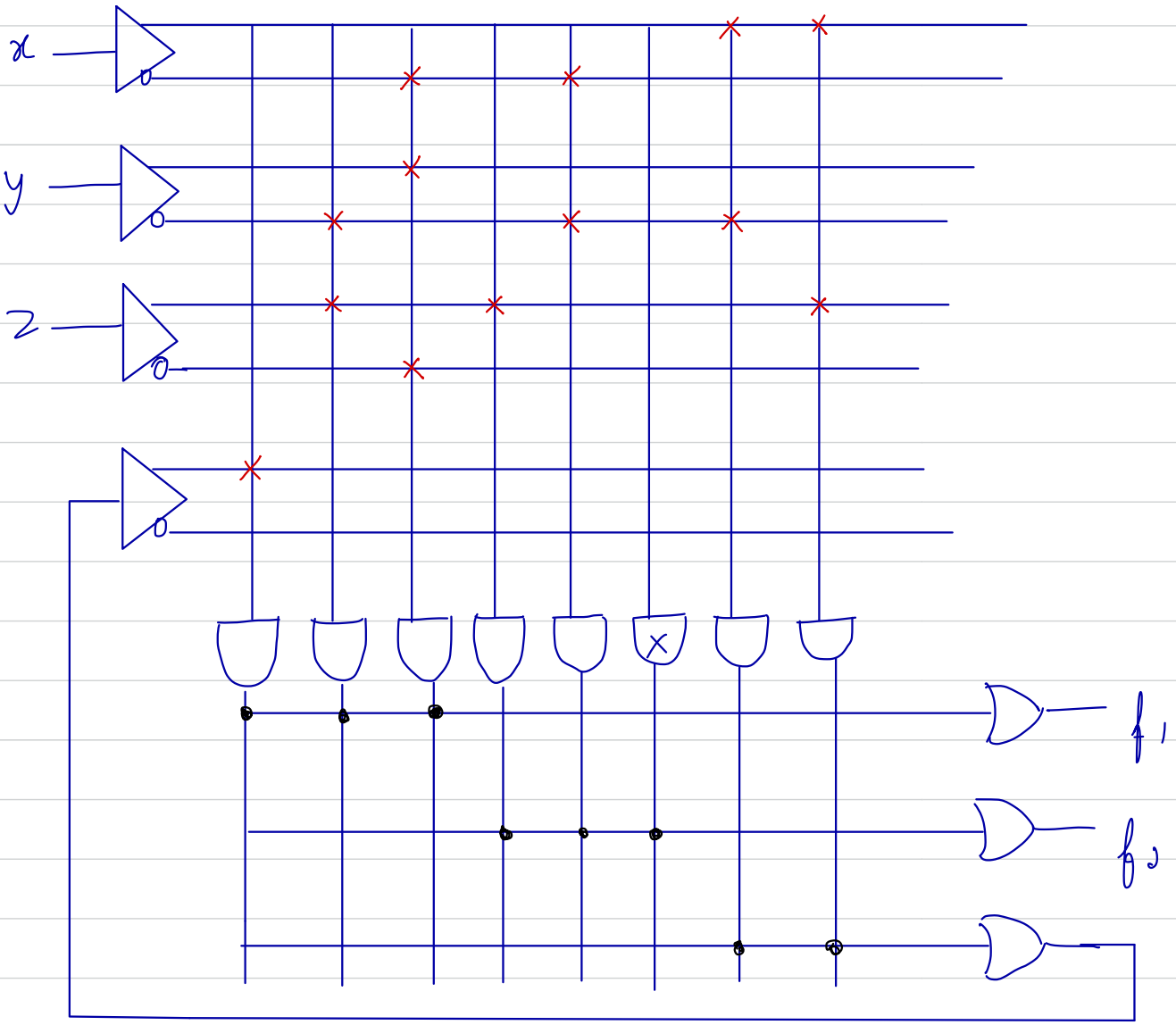
$f_2 = \Sigma m(0, 1, 3, 5, 7)$

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

$f_2 = z + \bar{x}\bar{y}$

$f_3 = x\bar{y} + xz$

$f_1 = f_3 + \bar{y}z + \bar{x}y\bar{z}$



PLA:

No of input = l
No of product term = m
No of output = n
Size of PLA = l x m x n

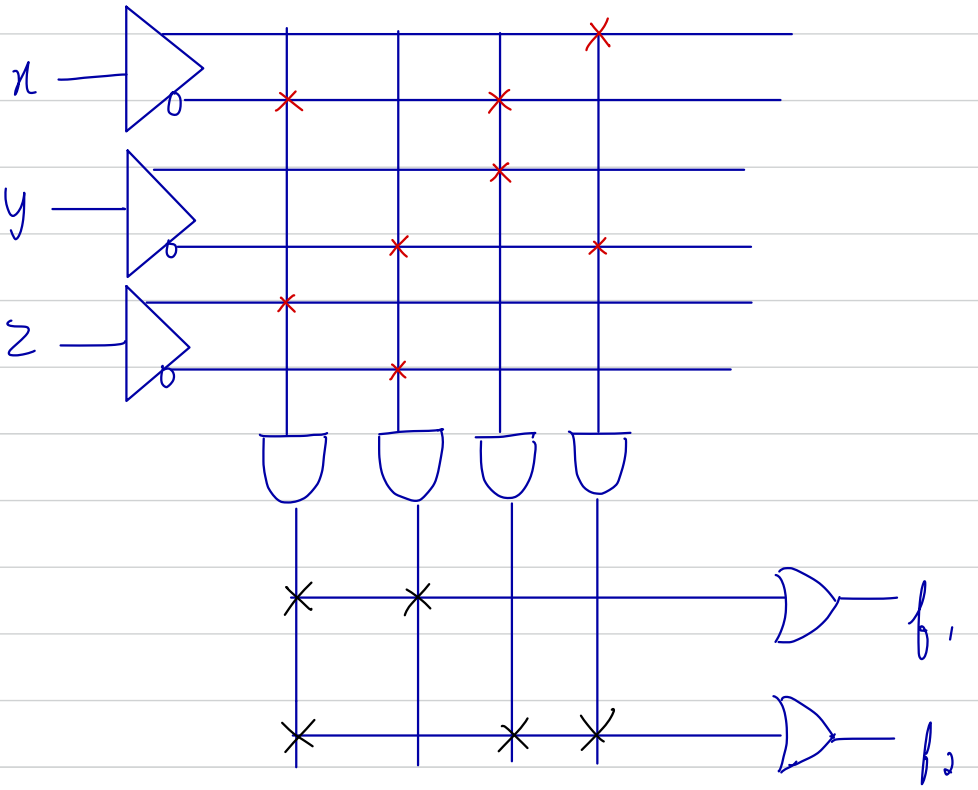
NOTE: X - Programmable
o - Non Programmable

1) $f_1(x, y, z) = \Sigma m(0, 1, 3, 4) = \bar{x}z + \bar{y}\bar{z}$

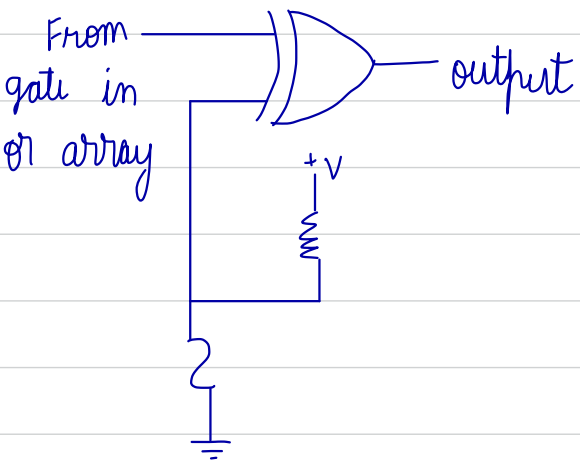
$f_2(x, y, z) = \Sigma m(1, 2, 3, 4, 5) = \bar{x}z + \bar{x}y + x\bar{y}$

Ans $l = 3 \qquad m = 4 \qquad n = 2$

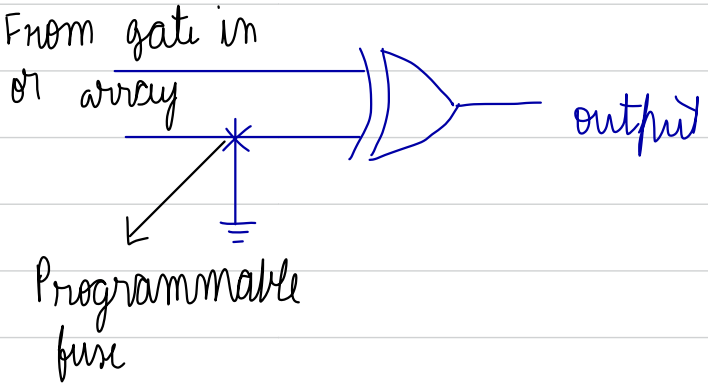
$\therefore$ Size $= 3 \times 4 \times 2 = 24$ //



EXOR gate with programmable fuse

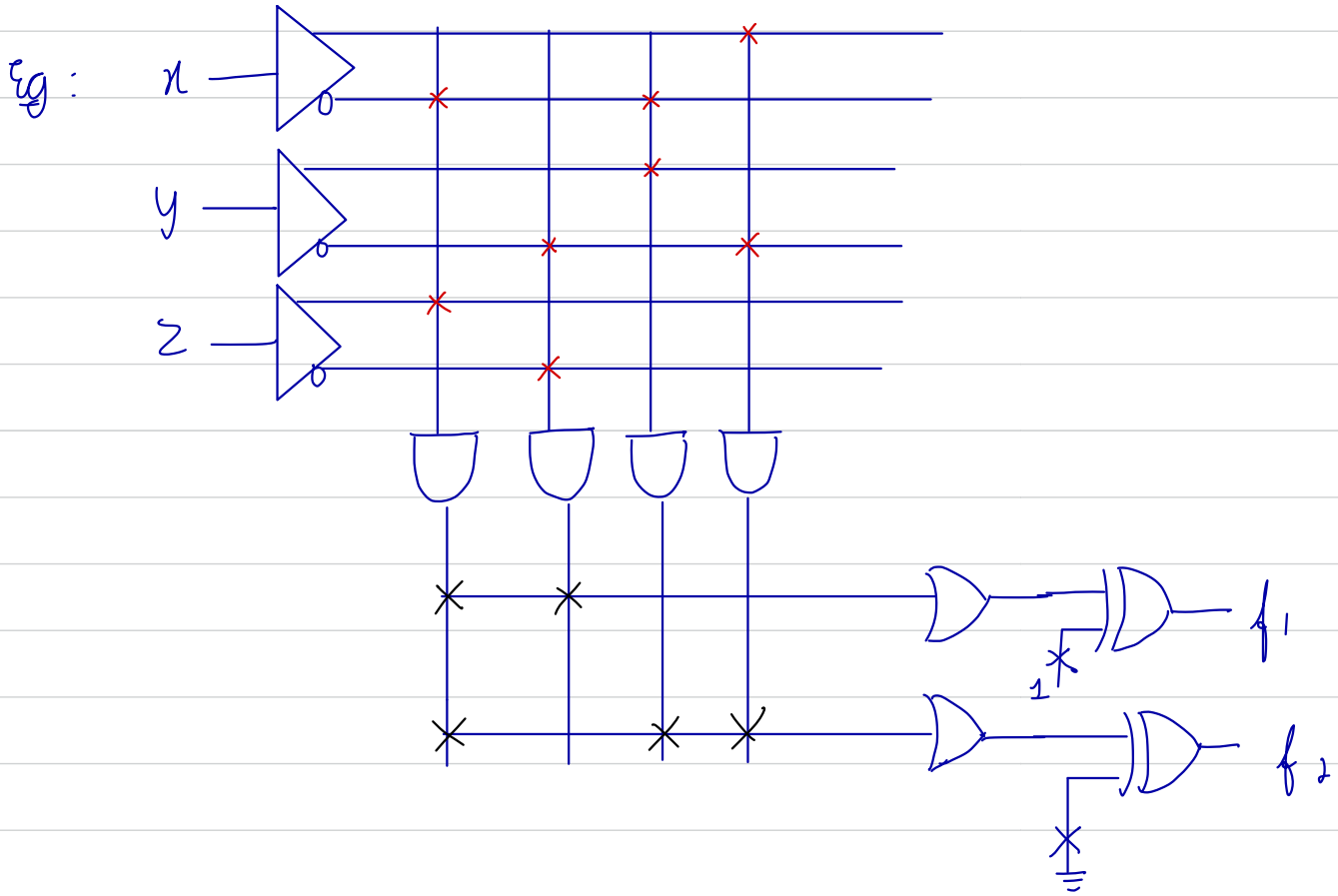


$\equiv$



PLA table

Product term	ip			op	
	x	y	z	f_1	f_2
$\bar{x}z$	0	–	1	1	1
$\bar{x}y$	0	1	–	0	1
$\bar{y}\bar{z}$	–	0	0	1	0
$x\bar{y}$	1	0	–	0	1
				C (active low)	T (active high)



Tutorial

Q1> Determine the excitation table for XY FF whose truth table is given as:

Ans

X	Y	Q(t+1)
0	0	1
0	1	Q(t)
1	0	$\bar{Q}(t)$
1	1	0

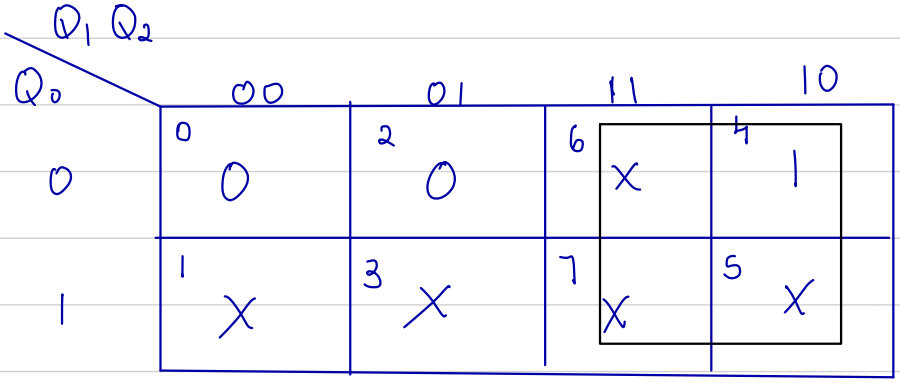
$\Rightarrow$

Q(t)	Q(t+1)	X	Y
0	0	x	1
0	1	x	0
1	0	1	x
1	1	0	x

Q2> Design a counter using JK FF that counts 0, 1, 2, 4, 5, 6, 0

Ans

PS			NS			J_2	K_2	J_1	K_1	J_0	K_0
Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+						
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	1	0	0	1	x	x	1	0	x
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	0	0	0	x	1	x	1	0	x



$J_2 = Q_1$

111 by

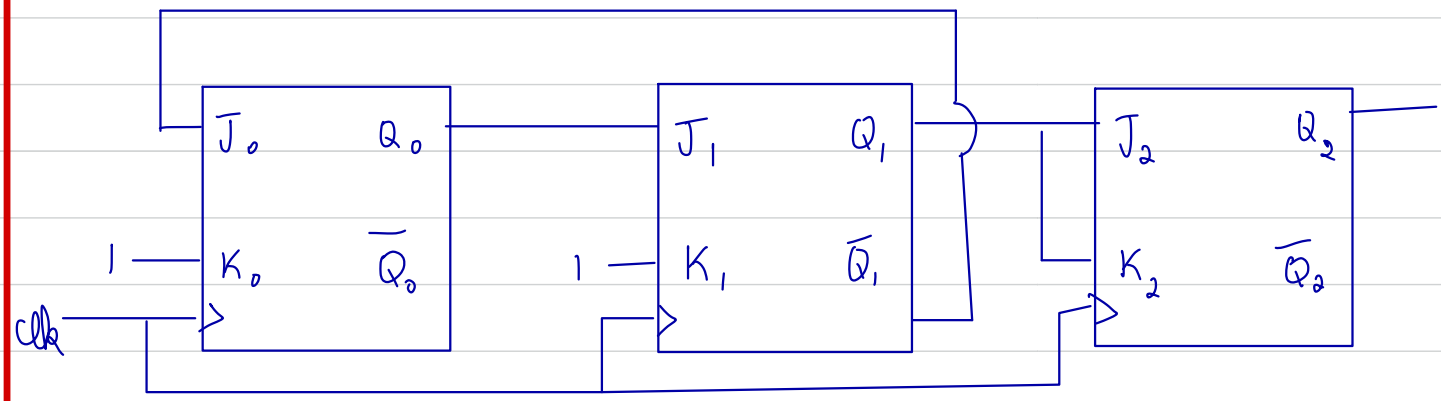
$J_1 = Q_0$

$J_0 = \bar{Q}_1$

$K_0 = 1$

$K_1 = 1$

$K_2 = Q_1$



Q3> A synchronous counter goes to 0, 3, 5, 6, 0 and flipflops inputs are $T_2 = Q_1$, $T_1 = 1$, & $T_0 = \bar{Q}_1$. Check if this self starting / self correcting counter.

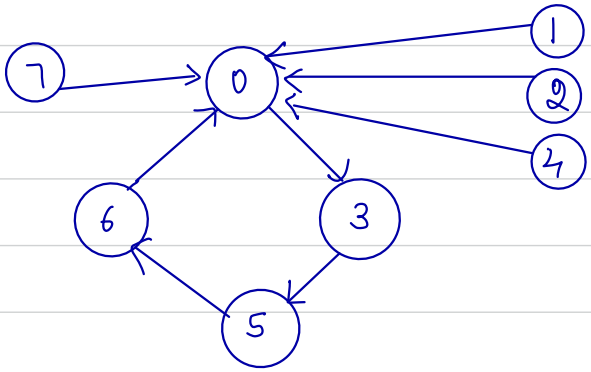
Ans Unused states 1, 2, 4, 7

Flip flop inputs			Present state			Next state		
T_2	T_1	T_0	Q_2	Q_1	Q_0	Q_2^+	Q_1^+	Q_0^+
0	1	1	0	0	1	0	1	0
1	1	0	0	1	0	1	0	0
0	1	1	1	0	0	1	1	1
1	1	0	1	1	1	0	0	1

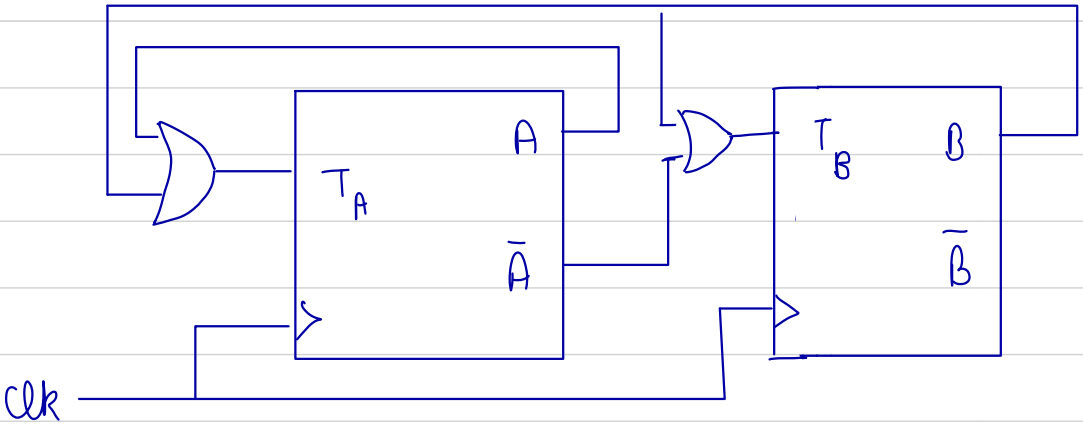
Not self correcting

Q4> Redesign the above counter to avoid lockdown condition.

Ans



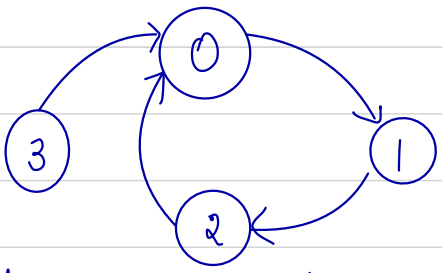
Q5> Determine the function of following circuit obtaining it's state diagram.



Ans $T_A = A + B$ $T_B = \bar{A} + B$

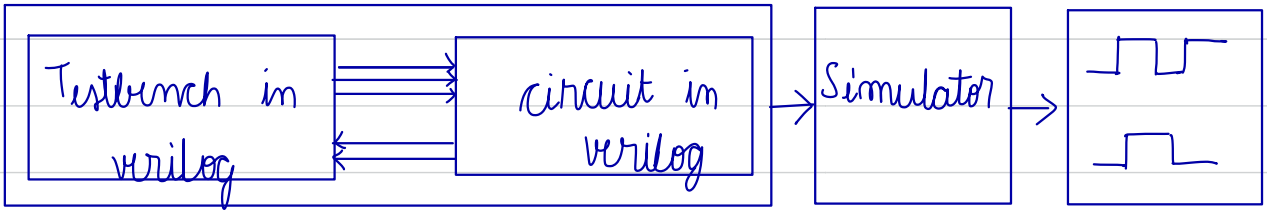
PS		FF i/p		NS	
A	B	T_A	T_B	A^+	B^+
0	0	0	1	0	1
0	1	1	1	1	0
1	0	1	0	0	0
1	1	1	1	0	0

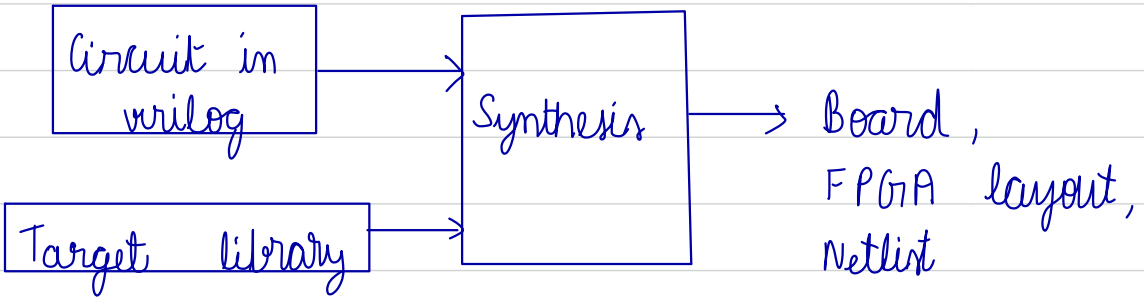
State diagram:



Mod - 3 self starting counter.

Verilog :





- Logic that is generated from synthesis tools depends highly on the code that is written in HDL.
- Good model - Efficient optimisation during synthesis.

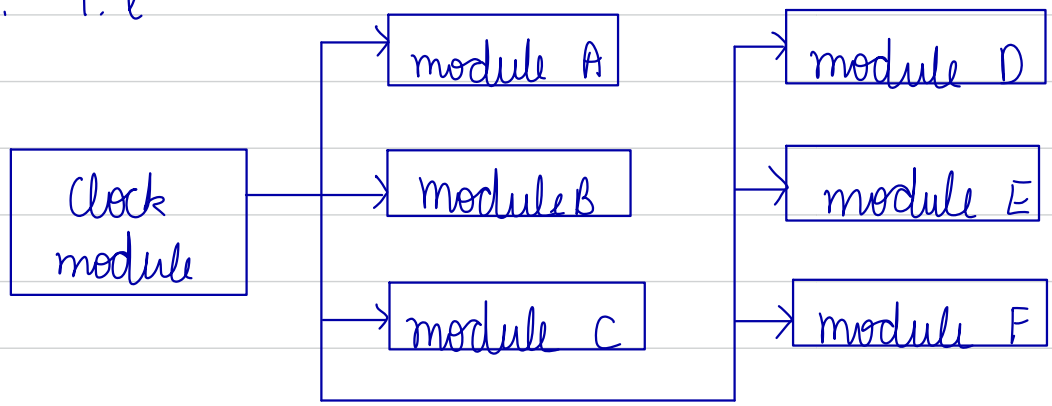
Coding Styles for synthesis :

1> Naming convention: It should be in such a way that integration becomes easier.

2> Design partitioning: Break a complex design into smaller modules. Partion size should be between 5000 to 15000 gates. Inter block signaling should be reduced as too many of them cause congestion in layout phase.

3> Clock. Global clock - non synthesisable - it should be added as analog block at the end.

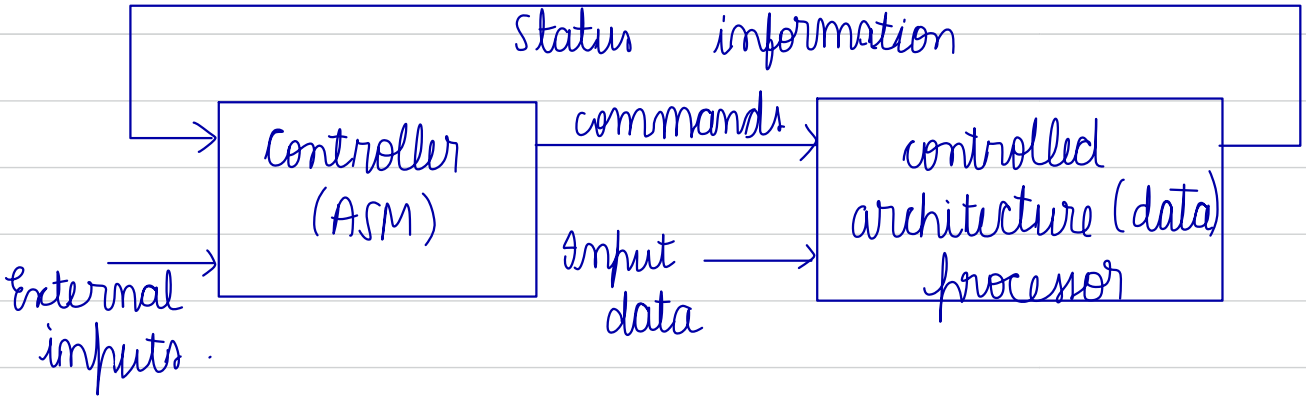
Internally generated clock should be used as little as possible. Use gated clock for control. Use component instantiation for controlled fanout. i.e

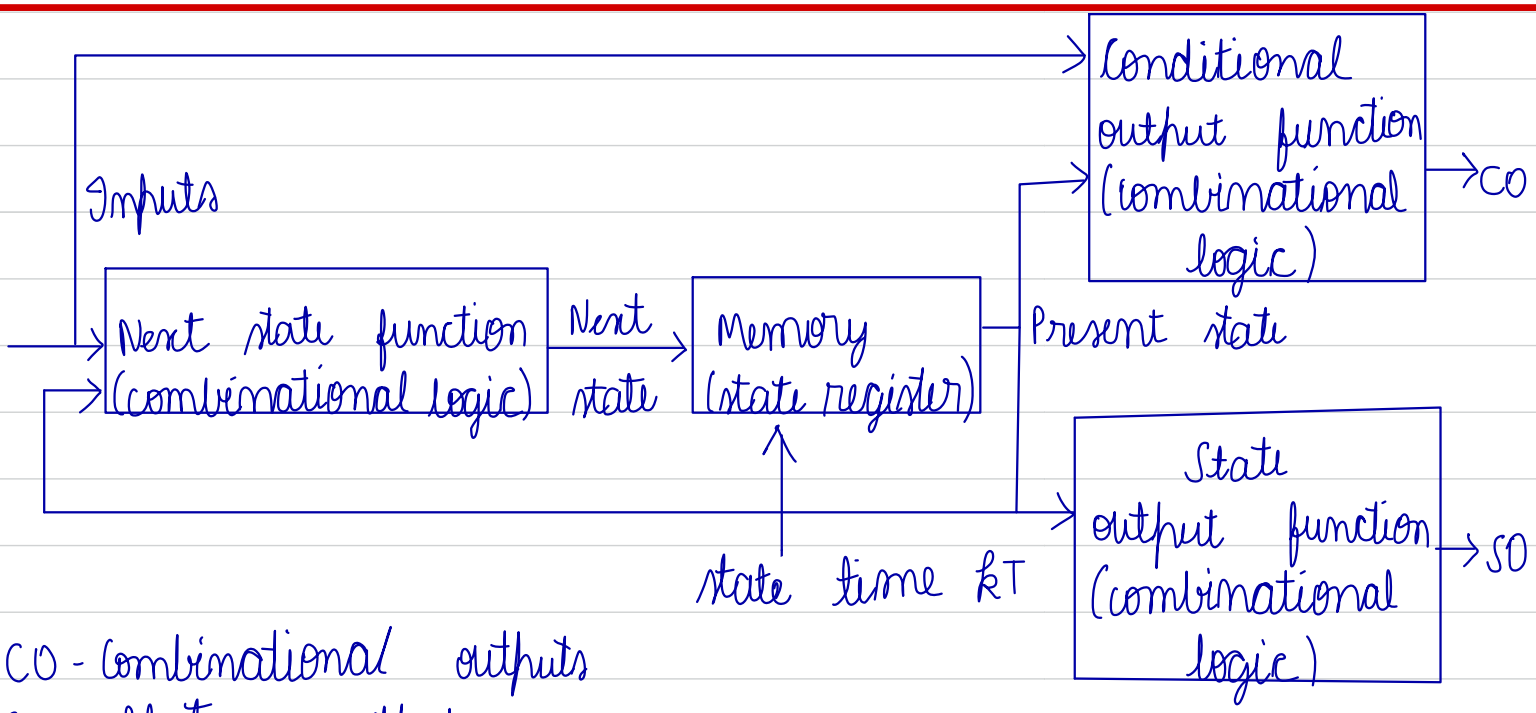


4) Reset - Can be synchronous or asynchronous.

5) Timing loop - It has potential to create timing glitch. It should be avoided for combinational circuits.

Algorithmic state machine (ASM)

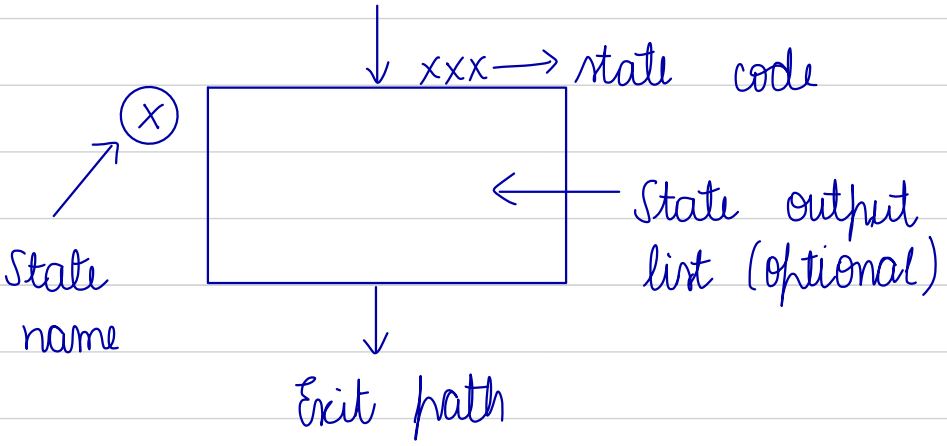




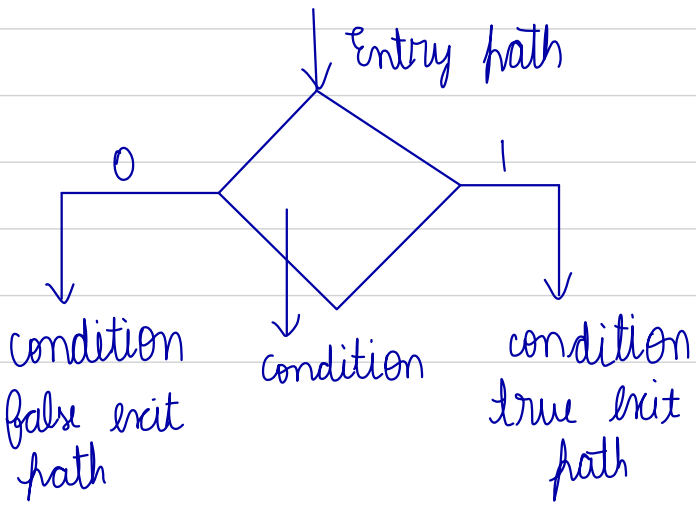
CO - Combinational outputs
SO - State outputs

Flow-chart representation

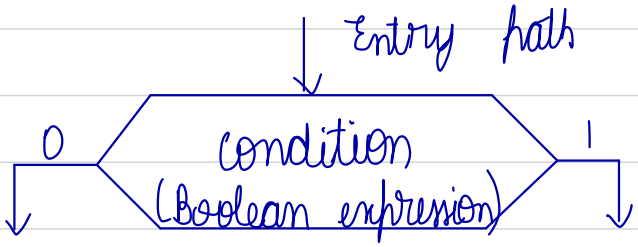
1> State box :



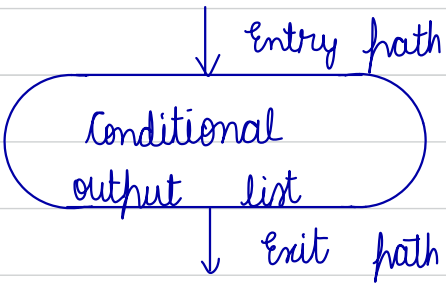
2> Decision box



or

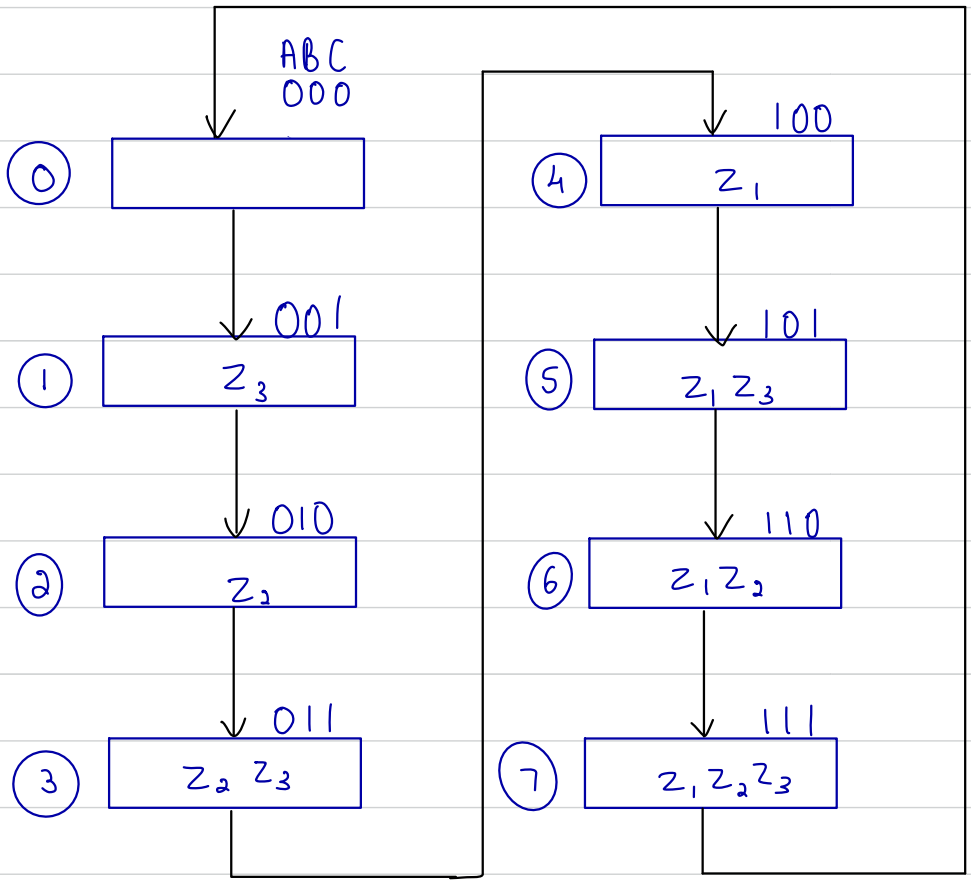


3> Conditional output box:

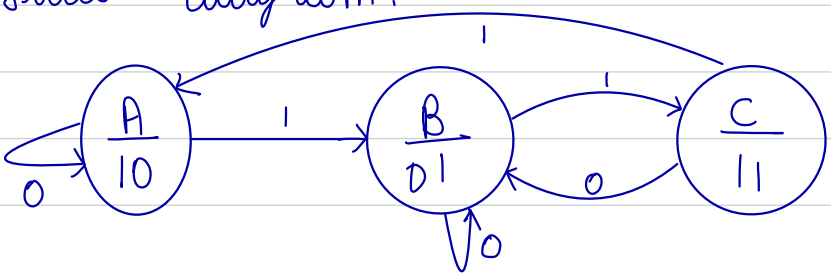


Examples:

1>



2> state diagram:



Input → x
Output → Z_1 Z_2

The given state diagram can be represented as: