Digital Logic design

Topics.

- Digital system & Binary number
- Boolean Algebra and Logic gates
- Gate level minimization (K-map)
- Combinational circuits
- Sequential circuits
  - Flip Flops
  - Counters
  - Shift Registers
- ~~Deg~~ Design of RTL

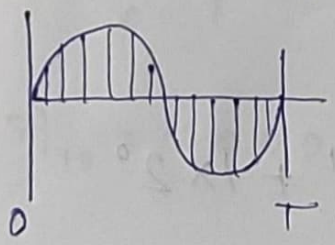
Signal - A signal is a physical quantity which depends on one or more than one quantity.

$y = x(t)$ signal

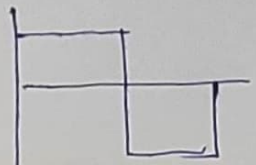
y → dependent

t → independent

- Electrical signals



undefined or non-discrete amplitude



Amplitude = 2

- If the no. of amplitude levels are finite then that signal is called digital signal.

Number Systems

① Decimal Number system (Base / Radix = 10)

0, 1, 2, 3, 4, 5, 6, 7, 8, 9 — digits

$(432)_{10} = 4 \times 100 + 3 \times 10 + 2 \times 1$

$10^2 \quad 10^1 \quad 10^0 \quad = 4 \times 10^2 + 3 \times 10^1 + 2 \times 10^0$

Base	Number System
2	Binary
8	Octal
10	Decimal
16	Hexadecimal

② Binary number system (Base = 2)

0 & 1

Binary digits

Bits

$(1101)_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

3 2 1 0

2 2 2 2

$= 8 + 4 + 1$

$= (13)_{10}$

$(13)_{10} = (1101)_2$

2	13	
2	6	1
2	3	0
2	1	1
	0	1

$(1101.001)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

$= 16 + 8 + 4 + 1 + \frac{1}{8}$

$= 29 + 0.125$

$= (29.125)_{10}$

$(17.25)_{10} = (10001.01)_2$

2	17	
2	8	1
2	4	0
2	2	0
2	1	0
	0	1

$0.25 \times 2 = 0.50 \quad 0$

$0.50 \times 2 = 1.00 \quad 1$

$(.25)_{10} = (01)_2$

$(17)_{10} = (10001)_2$

Standard form

$a_n \ldots a_3 a_2 a_1 a_0 . a_{-1} a_{-2} a_{-3} \ldots a_{-m}$

$= a_n r^n \ldots + a_2 r^2 + a_1 r^1 + a_0 r^0 + a_{-1} r^{-1} + a_{-2} r^{-2} + \ldots$
$\ldots a_{-m} r^{-m}$

Q- $(1010.11)_2$

$= 1\times 2^4 + 1\times 2^3 + 0\times 2^2 + 1\times 2^1 + 0\times 2^0 +$
$1\times 2^{-1} + 1\times 2^{-2}$

$= 16 + 8 + 2 + \frac{1}{2} + \frac{1}{4}$

$= 26 + 0.50 + 0.25$

$= (26.75)_{10}$

Q- $(127.4)_8$

$8 \mid 127$
$\quad 1$

$= 1\times 8^2 + 2\times 8^1 + 7\times 8^0 + 4\times 8^{-1}$

$= 64 + 16 + 7 + \frac{1}{2}$

$= (87.50)_{10}$

Q- $(4021.2)_5$

$= 4\times 5^3 + 0\times 5^2 + 2\times 5^1 + 1\times 5^0 + 2\times 5^{-1}$

$= 500 + 10 + 1 + \frac{2}{5}$

$= 511 + 0.4$

$= (511.4)_{10}$

Q - $(A21F)_{16}$

$= 10\times 2^3 + 2\times 2^2 + 1\times 2^1 + 15\times 2^0$

$= 80 + 8 + 2 + 15$

$= (105)_{10}$

$= 10\times 16^3 + 2\times 16^2 + 1\times 16^1 + 15\times 16^0$

$= 40960 + 512 + 16 + 15$

$= (41503)_{10}$

Q- $(BA9)_{13}$

$= 11\times 13^2 + 10\times 13^1 + 9\times 13^0$

$= 1859 + 130 + 9$

$= (1998)_{10}$

* $r = 10 \longrightarrow r$?

$r = 10 \longrightarrow r = 2$

$(41)_{10}$

$$\begin{array}{r|l|l} 2 & 41 & \\ 2 & 20 & 1 \\ 2 & 10 & 0 \\ 2 & 5 & 0 \\ 2 & 2 & 1 \\ 2 & 1 & 0 \\ & 0 & 1 \end{array}$$

$(41)_{10} = (101001)_2$

$$(101001)_2 = 1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$
$$= 32 + 8 + 1$$
$$= (41)_{10}$$

$(41.25)_{10} = (101001.01)_2$

$0.25 \times 2 = 0.50 \quad 0$
$0.50 \times 2 = 1.00 \quad 1$

$(87.5)_{10} = (127.4)_8$

$$\begin{array}{r|l|l} 8 & 87 & \\ 8 & 10 & 7 \\ 8 & 1 & 2 \\ & 0 & 1 \end{array}$$

$0.5 \times 8 = 4.00$

$(511.4)_{10} = (4021.2)_5$

$$\begin{array}{r|l|l} 5 & 511 & \\ 5 & 102 & 1 \\ 5 & 20 & 2 \\ 5 & 4 & 0 \\ & 0 & 4 \end{array}$$

$0.4 \times 5 = 2$

$(41503)_{10} = (A21F)_{16}$

$$\begin{array}{r|l|l} 16 & 41503 & \\ 16 & 2593 & 15 \\ 16 & 162 & 1 \\ 16 & 10 & 2 \\ & 0 & 10 \end{array}$$

3 - bit

0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

$(471)_8 \rightarrow (\underline{100}\ \underline{111}\ \underline{001})_2$

$(\underline{11}\ \underline{000}\ \underline{111})_2 \rightarrow (307)_8$

~~4 - bit~~

~~0~~
~~0~~
~~0~~
~~0~~
~~0~~
~~0~~
~~0~~
~~1~~

4 - bit

0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	A
1	0	1	1	B
1	1	0	0	C
1	1	0	1	D
1	1	1	0	E
1	1	1	1	F

$(2FA1)_{16} = (0010111110100001)_2$

Q1- Convert the following nos. to decimal.

(a) $(4310)_5$, (b) $(198)_{12}$ (c) $(435)_8$ (d) $(345)_6$

Q2 Convert the hexadecimal no. $(64CD)_{16}$ to binary and then the binary to octal.

Q3- Convert the following nos. to decimal

(a) $(26.24)_8$ (b) $(10110.0101)_2$ (c) $(DAD.AB)_{16}$

(d) $(16.5)_{16}$

Q4- Determine the base of the number system for each case for the following operation to be correct.

(a) $\frac{14}{2} = 5$, (b) $\frac{54}{4} = 13$, (c) $24+17=40$

(d) $8+19=27$

Sol^n (a) $\left(\frac{14}{2}\right)_r = \frac{(14)_r}{(2)_r}$

$= \frac{(r+4)_{10}}{(2)_{10}}$

$\Rightarrow \frac{(r+4)_{10}}{(2)_{10}} = (5)_{10}$

$\Rightarrow \frac{r+4}{2} = 5$

$\Rightarrow r = 6$

(b) $\left(\frac{54}{4}\right)_r = \frac{(54)_r}{(4)_r}$

$= \frac{(5\times r^1 + 4\times r^0)_{10}}{4} = 1\times r^1 + 3\times r^0$

$\Rightarrow \frac{5r+4}{4} = r+3$

$\Rightarrow 5r+4 = 4r+12$

$\Rightarrow r = 8$

(c) $24+17=40$

$\Rightarrow (2\times r^1 + 4\times r^0)_{10} + (1\times r^1 + 7\times r^0) = (4\times r^1 + 0\times r^0)$

$\Rightarrow (2r+4) + (r+7) = 4r$

$\Rightarrow 2r+4+r+7 = 4r$

$\Rightarrow 3r+11 = 4r$

$\Rightarrow r = 11$

(d) $8+19 = 27$

$\Rightarrow 8 + (1\times r+9) = 2\times r^1 + 7\times r^0$

$\Rightarrow 8 + r + 9 = 2r + 7$

$\Rightarrow 17 + r = 2r + 7$

$\Rightarrow r = 10$

$(26.24)_8$

$\begin{array}{r|l} 8 & 26 \\ 8 & 2 \quad 2 \\ & 0 \quad 2 \end{array}$ 24×8

$\Rightarrow 2\times 8^1 + 6\times 8^0 + 2\times 8^{-1} + 4\times 8^{-2}$

$\Rightarrow 16 + 6 + \frac{2}{8} + \frac{4}{64}$

$\Rightarrow 22 + \frac{1}{4} + \frac{1}{16}$

$\Rightarrow 22 + 0.25 + 0.062$

$\Rightarrow (22.312)_{10}$

(b) $(10110.0101)_2$

$= 1\times 2^4 + 0\times 2^3 + 1\times 2^2 + 1\times 2^1 + 0\times 2^0 + 0\times 2^{-1}$
$+ 1\times 2^{-2} + 0\times 2^{-3} + 1\times 2^{-4}$

$= 16 + 4 + 2 + \frac{1}{4} + \frac{1}{16}$

$= (22.312)_{10}$

(c) $(DAD.AB)_{16} = 13\times 16^2 + 10\times 16^1 + 13\times 16^0 + 10\times 16^{-1}$
$+ 11\times 16^{-2}$

$=$

(d) $(16.5)_{16} = 1\times 16^1 + 6\times 16^0 + 5\times 16^{-1}$
$= 16 + 6 + \frac{5}{16}$

$= 22 + \frac{5}{16}$

$=$

Binary Arithmetic

- Binary Addition

$2^n - 1$

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

$(21)_{10}$ 10101 (A)

$(14)_{10}$ + 01110 (B)

35 100011

$(100011)_2 = 32 + 2 + 1 = 35$

10111 $(23)_{10}$

01111 $(15)_{10}$

\+ 100110 $= 32 + 4 + 2$

$= 38$

Binary Subtraction

A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$$\begin{array}{rl} 10111 & (23)_{10} \\ -\ 01111 & (15)_{10} \\ \hline 01000 & = (8)_{10} \end{array}$$

Binary multiplication

$$\begin{array}{rl} 1011 & (11)_{10} \\ \times\ 101 & (5)_{10} \\ \hline 1011 & \\ 0000\times & \\ 1011\times\times & \\ \hline 110011 & = 32+16+4+2+1 \\ & = (55)_{10} \end{array}$$

$$\begin{array}{r} (236)_7 \\ +\ (455)_7 \\ \hline (1024)_7 \end{array} \qquad \begin{array}{r} (ABCD)_{16} \\ (FDAO)_{16} \\ \hline (1A96\,0D)_{16} \end{array}$$

$$\begin{array}{r} (AAA)_{12} \\ (999)_{12} \\ \hline (1887)_{12} \end{array}$$

A = 10
B = 11

Q1- Add and multiply the following nos. without converting them to decimal

(1) $(1011)_2$
$(101)_2$

(2) $(2B)_{16}$
$(34)_{16}$

(3) $(767)_8$
$(37)_8$

Complement of numbers

- Simplify the subtraction
- radix complement (r's complement)
- diminished radix complement ($(r-1)$'s complement)

$r = 10$, 10's complement
9's complement

$r = 2$, 2's complement
1's complement

$r = 8$, 8's complement
7's complement

$r = 16$, 16's complement
15's complement

* diminished radix comp.
$(r-1)$'s complement

$(N)_r$ having 'n' digits

$(r-1)$'s comp of the above number is
$(r^n - 1) - N$

$r = 10$

9's complement of $(22098)_{10}$

$n = 5$, $N = 22098$, $r = 10$

$$\begin{array}{r} 99999 \\ -\ 22098 \\ \hline 77901 \end{array}$$

$\therefore$ 9's complement of $(22098)_{10}$ is 77901.

* A 9's complement of a decimal number is obtained by subtracting each digit from 9.

$r = 2$

1's complement of $(1101)_2$

$(2^4 - 1) - N$

$$\begin{array}{r} 1111 \\ -\ 1011 \\ \hline 0100 \end{array}$$

* In binary 1's complement can be obtained by subtracting each bit from 1
* complement or NOT gate of each bit gives 1's complement.

- radix complement
  (r's) complement

$(N)_r$ having 'n' digits
(r's) complement of the above number is
$(r^n) - N$ [when $N \neq 0$]

- 10's complement of $(22098)_{10}$

$$\begin{array}{r} 99999 \\ 22098 \\ \hline 77901 \end{array} \qquad \begin{array}{r} 77901 \\ + \quad 1 \\ \hline 77902 \end{array}$$

$\therefore$ 10's complement of $(22098)_{10}$ is $(77902)_{10}$

- 2's complement of $(11011)_2$

$(11011)_2$

$$\begin{array}{r} 00100 \\ + \quad 1 \\ \hline 00101 \end{array}$$

$\therefore$ 2's complement of $(11011)_2$ is $(00101)_2$

- 16's complement of $(ABC)_{16}$

$$\begin{array}{r} FFF \\ -ABC \\ \hline 543 \end{array} \leftarrow \text{15's complement}$$

$$\begin{array}{r} 543 \\ + \quad 1 \\ \hline 544 \end{array} \leftarrow \text{16's complement}$$

Subtraction using (r-1)'s complement

$A = (23567)_{10}$
$B = (0942)_{10}$

$A - B = A + (-B)$

① $$\begin{array}{r} 23567 \\ -01942 \\ \hline \end{array}$$

9's complement

② $$\begin{array}{r} 99999 \\ 01942 \\ \hline 98057 \end{array}$$

③ Addition

④ overflow bit is added

$B - A$

① subtraction of two ~~using~~ n-digits unsigned nos using

Step 1 : Make sure that both nos. should have same no. of digits or bits

Step 2 : Calculate the (r−1)'s complement of subtrahend B and add the result with minuend A

A − Minuend
− B − subtrahend

Step 3 : ① If the result contains a carry then add the carry to the least significant digit and the result is a +ve result

② If the result doesn't contain a carry then take (r−1)'s complement of the result and put a (−ve) sign to the left

Subtraction using radix complement

```
 23      99      86
-13     -13     +23      +10
 ---     ---     ---      ---
 86      86     109

 13      99      76      99
-23     -23     +13     -89      -10
 ---     ---     ---     ---      ---
 76      76      89      -10

 01101     01101     11111
-10111    +01000    -01010
 ------    ------    ------
           10101      00000
```

```
 23
-13
```

using 10's complement and 2's complement

```
 99      86      +09   Ans
-13     +23      +1
 ---     ---     ---
 86     109      +10

 10111     10010     01001     2's
-01101    +10111     +1
 ------    ------    ------
           101001    01010

 13      99      77      99
-23     -23     +13     -89
 ---     ---     ---     ---
 76      76      90      -10
```

```
 4637     9999     7420     9's complement
-2579     2579     4637
 ----     ----     -----
          7420     12057
                    +  1
                    -----
                     2058

 4637     9999     7421     10's complement
-2579     2579    +4637     2058
 ----     ----    -----
          7420     02058
           +1
          ----
          7421

 101000    101010     1's complement
-010101   +101000
 ------    -------
          1010010
            +    1
          -------
           010011
```

```
 101000      101010
-010101     +101000
 ------      ------
            1010010
            +      1
            -------
           (1)010011
            -------
             010011
```

Binary Code

* BCD / (8421)

Binary coded decimal

BCD = Binary
(D ≤ 9)

Decimal	Binary	BCD
0	0000	
1	0001	
2	0010	
3	0011	
4	0100	← Same upto 9
5	0101	
6	0110	
7	0111	
8	1000	
9	1001	

x decimal – 4x BCD

(457) BCD

→ (0100 0101 0111) BCD

(589)

→ (0101 1000 1001)

(448)

→ (0100 0100 1000)

```
 448        0100 0100 1000
+589       +0101 1000 1001
 ----      -------------
 1037       1010 1100 0001
           +0110 +0110 +0110
           -----------------
           10000 0011  0111
           -----------------
           0001  0000  0011  0111
           ----------------------
            1     0     3     7
```

```
 184        0001 1000 0100
+576       +0101 0111 0110
 ---       -------------
 760        0111 10000 1010
             +0110  0110
             -----  ----
              0110  00000
             -----------
             0111  0110  0000
             ----------------
              7     6     0
```

* Different codes (Excess-3, 2421, 84-2-1)

Decimal	BCD (8421)	2421 (MSB ← LSB)	Excess-3 (+3)	84-2-1
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	0101	1000	1011
6	0110	0110	1001	1010
7	0111	0111	1010	1001
8	1000	~~1000~~	1011	1000
9	1001	1111	1100	1111

9's of decimal = 1's of Binary

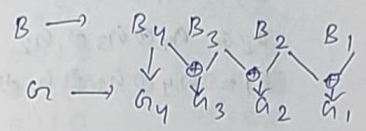
2421 → self complementary code

Excess-3 →

* 2421 & Excess-3 are self complementary codes
* Such codes have a property that the 9's complement of the decimal number can be obtained directly by 1's complementing the binary number.
* BCD, 2421, 84-2-1 are weighted codes except Excess-3

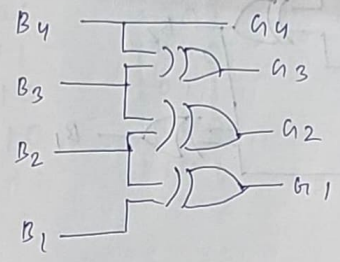
Gray code

* used in minimization technique



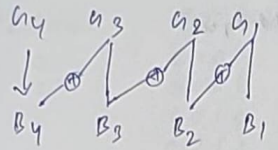
9 – 1001

G – 1101



Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101

Gray code to binary

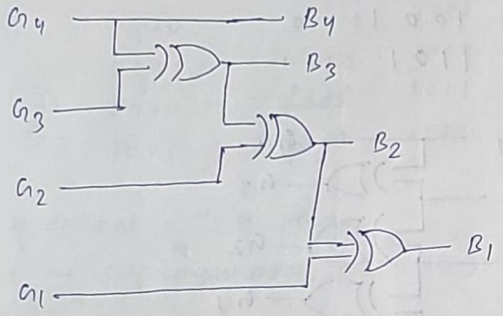


$B_4 = G_4$

$B_3 = G_4 \oplus G_3$

$B_2 = G_4 \oplus G_3 \oplus G_2$

$B_1 = G_4 \oplus G_3 \oplus G_2 \oplus G_1$



3-bit

Decimal	Binary	Gray code
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

4-bit (Gray to Binary)

Gray	Binary	
0000	0000	0
0001	0001	1
0011	0010	2
0010	0011	3
0110	0100	4
0111	0101	5
0101	0110	6
0100	0111	7
1100	1000	8
1101	1001	9
1111	1010	
1110	1011	
1010	1100	
1011	1101	

Parity Bits

* Error detection (1 bit) during transmission.
* Parity is added to original binary.
* Parity generator

PARITY

Satisfied = 0
Not satisfied = 1

EVEN

* 1's even = P = 0
* 1's odd = P = 1

ODD

* 1's odd = P = 0
* 1's even = P = 1

Even Parity Table

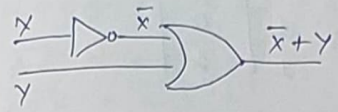
B_3	B_2	B_1	P (Even)	P (odd)
0	0	0	0	1
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	

B_3	B_2	B_1	(odd P)
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

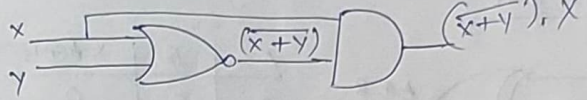
Logic Gates & Boolean algebra

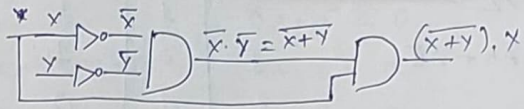
1) $\bar{x} + y$ 2) $(\overline{x+y}) \cdot x$

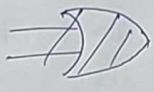
1)



2)

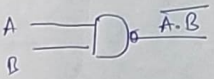






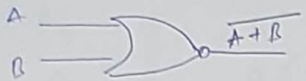
* NAND / NOR → universal logic gates

NAND gate



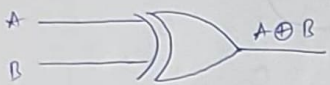
A	B	$AX = \overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate



A	B	$X = \overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

XOR gate



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

A	B	$X = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Properties of boolean algebra

1. Closure $\{+, \cdot\}$
2. Commutative $\{+, \cdot\}$
3. Associativity $\{+, \cdot\}$
4. ~~Inverse~~ Identity $\left\{\begin{matrix} + \\ 0 \end{matrix}, \begin{matrix} \cdot \\ 1 \end{matrix}\right\}$

$\boxed{e * x = x * e = x}$

$0 + x = x + 0 = x$

$1 \cdot x = x \cdot 1 = x$

5. Inverse $\left\{\begin{matrix} + \\ 1 \end{matrix}, \begin{matrix} \cdot \\ 0 \end{matrix}\right\}$

$x + \bar{x} = 1$

$x \cdot \bar{x} = 0$

6. Distributive

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

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

7. De morgan's law

$\overline{A \cdot B} = \bar{A} + \bar{B}$

$\overline{A + B} = \bar{A} \cdot \bar{B}$

Postulates

1. $x + 0 = x$ $\qquad$ $x \cdot 1 = x$
2. $x + \bar{x} = 1$ $\qquad$ $x \cdot \bar{x} = 0$
3. $x + x = x$ $\qquad$ $x \cdot x = x$
4. $x + 1 = 1$ $\qquad$ $x \cdot 1 = x$
5. $\overline{(\bar{x})} = x$
6. $x + y = y + x$ $\qquad$ $x \cdot y = y \cdot x$
7. $x + xy = x$ $\qquad$ $x(x + y) = x$

Q. Prove 1. $x + x = x$

L.H.S : $(x + x) \cdot 1$

$\Rightarrow (x + x) \cdot (x + \bar{x})$

$\Rightarrow x + x\bar{x}$

$\Rightarrow x$ = R.H.S

Q- Prove $x+1=1$

L.H.S - $(x+1)\cdot 1$

$\Rightarrow (x+1)\cdot(x+\bar{x})$

$\Rightarrow \cancel{xx} + x\bar{x} + \cancel{1\cdot x} + 1\cdot\bar{x}$

$\Rightarrow x + 0 + x + \bar{x}$

$\Rightarrow x + 1\cdot\bar{x}$

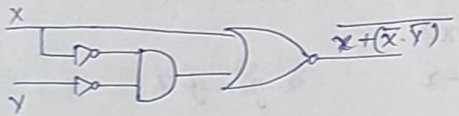
$\Rightarrow (x+\bar{x})$

$\Rightarrow 1$

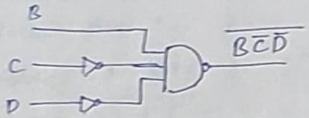
Q: $Z = \overline{x+(\bar{x}\cdot\bar{y})}$

$A = \overline{B\bar{C}D}$

Z =



A =



- $F = \overline{(\overline{AB}+\overline{CD})}$

$= \overline{(\overline{AB})(\overline{CD})}$ - NAND expression

- $F = \overline{\overline{(A+B)(C+D)}}$

$= \overline{(\overline{A+B})+(\overline{C+D})}$ - NOR expression

Canonical & Standard Form

$F_1 = x'yz' + x'y'z$

Find complement using De-morgan's Law & direct method

De-morgan Law

$(\bar{F_1}) = \overline{(\bar{x}y\bar{z} + \bar{x}\bar{y}z)}$

$= \overline{\bar{x}y\bar{z}}\cdot\overline{\bar{x}\bar{y}z}$

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

Duality method

$F_1 = (x'+y+z')\cdot(x'+y'+z)$

$= (x+y'+z)(x+y+z')$

$$F_2 = [x(\bar{y}\bar{z} + yz)]$$

$$\bar{F}_2 = \overline{[x(\bar{y}\bar{z} + yz)]}$$

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

$$= \bar{x} + (\overline{\bar{y}\bar{z}}) \cdot \overline{yz}$$

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

Duality method

$$F_2 = [x + (y'+z') \cdot (y+z)]$$

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

Minterm & Maxterm

Minterm - Each minterm is a standard which is obtained from an AND term of n variables, with each variable is being primed when the corresponding bit is 0 and unprimed when bit is 1

x	y	z	(min) Term	Minterm	(max) Term	Maxterm
0	0	0	$\bar{x}\bar{y}\bar{z}$	m_0	$x+y+z$	M_0
0	0	1	$\bar{x}\bar{y}z$	m_1	$x+y+\bar{z}$	M_1
0	1	0	$\bar{x}y\bar{z}$	m_2	$x+\bar{y}+z$	M_2
0	1	1	$\bar{x}yz$	m_3	$x+\bar{y}+\bar{z}$	M_3
1	0	0	$x\bar{y}\bar{z}$	m_4	$\bar{x}+y+z$	M_4
1	0	1	$x\bar{y}z$	m_5	$\bar{x}+y+\bar{z}$	M_5
1	1	0	$x\bar{y}\bar{z}$	m_6	$\bar{x}+\bar{y}+z$	M_6
1	1	1	xyz	m_7	$\bar{x}+\bar{y}+\bar{z}$	M_7

$$F = xy + y'z$$

$$= xy(z+z') + y'z(x+x')$$

$$= xyz + xyz' + xy'z + x'y'z$$

$$= m_7 + m_6 + m_5 + m_1$$

$$= \Sigma m(1, 5, 6, 7)$$

$$F = A + B'C$$

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

$$= ABC + ABC' + \underline{AB'C} + AB'C' + \underline{AB'C} + A'B'C$$

$$= m_7 + m_6 + m_5 + m_4 + m_1$$

$$= \Sigma m(1, 4, 5, 6, 7)$$

$$= \Pi M(0, 2, 3) \rightarrow \text{Max term}$$

$$= M_0 \cdot M_2 \cdot M_3$$

Canonical form

→ Sum of minterms or Product of maxterm

→ Maxterm = $\overline{\text{Minterm}}$

Maxterm - Each maxterm is a additive quantity which is obtained from or term of n variable with each variable is primed when it is 1 and unprimed when it is 0

$F = xy + y'z$

$= xyz + xyz' + xy'z + x'y'z$

(product) $= (x'+y'+z') \cdot (x'+y'+z) \cdot (x'+y+z') \cdot (x+y+z)$

$= M_7 \cdot M_6 \cdot M_5 \cdot M_1$

$= \Pi M(1, 5, 6, 7)$

$F = xy + x'z$

$= xy(z+z') + x'z(y+y')$

$= xyz + xyz' + x'yz + x'y'z$

$= m_7 + m_6 + m_3 + m_1$

$= \Sigma m(7, 6, 3, 1)$

$F = xy + x'z$

$= xy + x'z$

$= (xy + x')(xy + z)$

$= (x'+y)(x+x')(x+z)(y+z)$

$= (x'+y)(x+z)(y+z)$

$= (x'+y+zz')(x+yy'+z)(x'+y+z)$

$= (x'+y+z)(x'+y+z')(x+y+z)$
$(x+y'+z)(x+y+z)(x'+y+z)$

$= (x'+y+z)(x'+y+z')(x+y+z)$
$(x+y'+z)$

$= M_4 \cdot M_5 \cdot M_0 \cdot M_2$

$= \Pi M(0, 2, 4, 5)$

$F = x' + y(zx + z')$

K-Map

① It is the pictorial form of a truth table

② It is a diagram made up of squares where each square represents one minterm of the function which needs to be minimized.

③ It is also a visual diagram of all possible ways in which a function can be represented

④ The simplified expression produced by the K-Map is of either POS or of SOP form.

⑤ This K-Map always provides minimum no. of gates with minimum no. of inputs.

⑥ The distance between one square to another square, the change is always by 1 bit and it is designed by gray code.

2-~~bit~~ variable K-map

$x \backslash y$	0	1
0	$x'y'$	$x'y$
1	xy'	xy

$F = \Sigma m(3)$

$x \backslash y$	y	y'	y
x' 0			
x 1			1

$F = xy$

$F = \Sigma m(1, 2, 3)$

$x \backslash y$	y	y'	y
x'			1
x		1	1

$F = y + x$

3-variable K-Map

$x \backslash yz$	yz	$y'z'$ 00	$y'z$ 01	yz 11	yz' 10
x' 0					
x 1					

$F = \Sigma m(1, 3)$

$x \backslash yz$	00	01	11	10
x' 0		1	1	
x 1				

$F = x'z$

$F = \Sigma m(1, 3, 6, 7)$

$x \backslash yz$	yz	$y'z'$ 00	$y'z$ 01	yz 11	yz' 10
x' 0			1	3	
x 1				6	7

$x'z + xy$

$F(x,y,z) = \Sigma(2,3,4,5)$

$x \backslash yz$	$y\bar{z}$ 00	$\bar{y}\bar{z}$ 01	yz 11	$y\bar{z}$ 10
$\bar{x}$ 0			1	1
x 1	1	1		

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

$F(x,y,z) = \Sigma(4,6,7)$

$x \backslash yz$	$y\bar{z}$ 00	$\bar{y}\bar{z}$ 01	yz 11	$y\bar{z}$ 10
$\bar{x}$ 0			1	
x 1	1		1	1

~~$x\bar{y} + x$~~

$yz + x\bar{z}$

$F(x,y,z) = \Sigma(0,2,4,5,6)$

$x \backslash yz$	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$	1			1
x	1	1		1

$z + x\bar{y}$

$F(x,y,z) = \Pi(0,1,2)$
$= \Sigma(3,4,5,6,7)$

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$			1	
x	1	1	1	1

~~$x\bar{y} + yz + xy$~~

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$			1	
x	1	1	1	1

$x + yz$

$F(x,y,z) = \Pi(4,5,6,7)$
$= \Sigma(0,1,2,3)$

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{x}$	1	1	1	1
x				

~~$\bar{x}\bar{y} + \bar{x}y$~~ $\bar{x}$

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

$\bar{x}$

$F = A'C + A'B + AB'C + BC$

a) Express this f^n as sum of minterms

b) Find the minimal SOP expression

$F = A'C(B+B') + A'B(C+C') + AB'C + (A+A')BC$

$= A'CB + A'CB' + \underline{A'BC} + A'BC' + AB'C + ABC + \underline{A'BC}$

$= ABC + AB'C + A'BC' + A'BC + A'CB' + A'CB$

4 - variable K-map

ab \ cd	$\bar{c}\bar{d}$ 00	$\bar{c}d$ 01	cd 11	$c\bar{d}$ 10
$\bar{a}\bar{b}$ 00	0	1	3	2
$\bar{a}b$ 01	4	5	7	6
ab 11	12	13	15	14
$a\bar{b}$ 10	8	9	11	10

$2^0 = 1 \Rightarrow 4$

$2^1 = 2 \Rightarrow 3$

$2^2 = 4 \Rightarrow 2$

$2^3 = 8 \Rightarrow 1$

$Y(A,B,C,D) = \Sigma(2, 3, 12, 13, 14, 15)$

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	0	1	1 3	1 2
$\bar{A}B$ 01	4	5	7	6
AB 11	1 12	1 13	1 15	1 14
$A\bar{B}$ 10	8	9	11	10

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

$Y(w,x,y,z) = \Sigma(0, 1, 2, 3, 5, 6, 11, 13)$

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	1 0	1 1	1 3	1 2
$\bar{A}B$ 01	4	1 5	7	6
AB 11	12	1 13	15	14
$A\bar{B}$ 10	8	9	1 11	10

$\bar{A}\bar{B} + \bar{A}C\bar{D} + B\bar{C}D + \bar{B}CD$

$Y(A,B,C,D) = \Sigma(0, 2, 5, 7, 8, 10, 13, 15)$

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1			1
$\bar{A}B$		1	1	
AB		1	1	
$A\bar{B}$	1			1

$BD + \bar{B}\bar{D}$

Don't Care condition

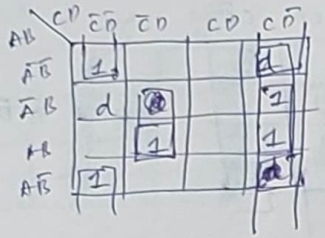
$F = \Sigma(1, 3, 7, 11, 15) + d\,\Sigma(0, 2, 5) = F(w, x, y, z)$

	$\bar{y}\bar{z}$	$\bar{y}z$	yz	$y\bar{z}$
$\bar{w}\bar{x}$	d	1	1	d
$\bar{w}x$		d	1	
wx			1	
$w\bar{x}$			1	

$\bar{w}z + yz$

$\bar{w}\bar{x} + yz$

$= (A, B, C, D) = \Sigma(0, 6, 8, 13, 14) + d(2, 4, 10)$



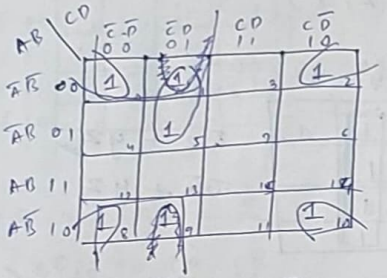
$\cancel{\bar{B}\bar{C}\bar{D} + B\bar{C}D + B\bar{C}\bar{D}}$

$C\bar{D} + \bar{B}\bar{C}\bar{D} + AB\bar{C}D$

POS (maxterm) using K-map

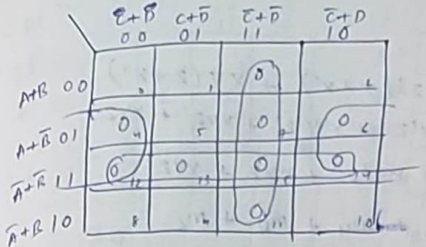
$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$

$= \Pi(3, 4, 6, 7, 11, 12, 13, 14, 15)$



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

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



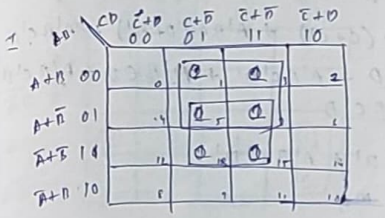
$(\bar{B}+D).(\bar{C}+\bar{D}).(\bar{A}+\bar{B})$

1. $F(A, B, C, D) = \Pi(1, 3, 5, 7, 13, 15)$
2. $F(A, B, C, D) = \Pi(1, 3, 6, 9, 11, 12, 14)$

Minimized expression should be POS & SOP

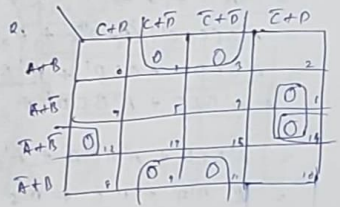
1. $x'z' + y'z' + xy$
2. $(A'+B)(A'+C')(A'+D)$
3. $\cancel{(A)}$ $A.(B+C.D')$

1.



$(A+\bar{D}).(\bar{B}+\bar{D})$

2.



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

1. $x'z' + y'z' + xy$

$= x'z'(y+y') + y'z'(x+x') + xy(z+z')$

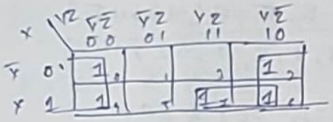
$= x'yz' + x'y'z' + xy'z' + x'y'z' + xyz + xyz'$

$= x'yz' + x'y'z' + xy'z' + xyz + xyz'$

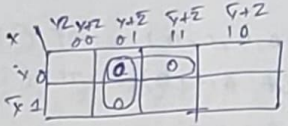
$= m_2 + m_0 + m_4 + m_7 + m_6$

$= \Sigma m(0, 2, 4, 6, 7)$

$= \Pi M(1, 3, 5)$



$\bar{z} + xy$



$(x+\bar{z}).(y+\bar{z})$

3. $A'B(A'+C')(A'+D)$

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

$= A'B + A'BD + A'BC + A'BC'D$

$= A'B(C+C')(D+D') + A'BD(C+C') + A'BC(D+D') + A'BC'D$

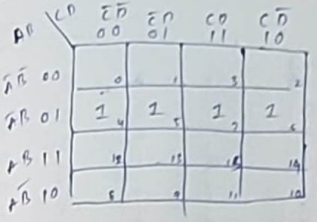
$= A'BCD + A'BCD' + A'BC'D + A'BC'D' + A'BCD + A'BC'D$
$+ A'BCD + A'BCD' + A'BC'D$

$= A'BCD + A'BC'D + A'BCD' + A'BC'D'$

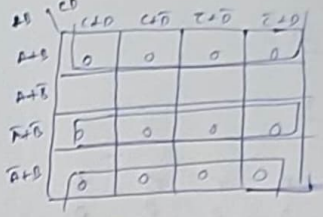
$= m_7 + m_5 + m_6 + m_4$

$= \Sigma(4, 5, 6, 7)$

$= \Pi(0, 1, 2, 3, 8, 9, 10, 11, 12, 13, 14, 15)$



$\bar{A}B$



$B.(\bar{A}+\bar{B})$

5. $A(B+CD')$

$= AB + ACD'$

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

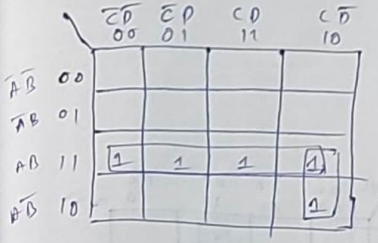
$= ABCD + ABCD' + ABC'D + ABC'D' + ABCD' + AB'CD'$

$= ABCD + ABCD' + ABC'D + ABC'D' + AB'CD'$

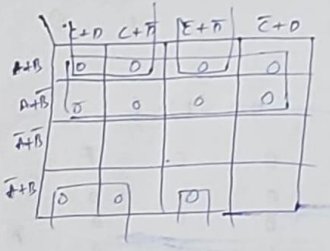
$= m_{15} + m_{14} + m_{13} + m_{12} + m_{10}$

$= \Sigma(10, 12, 13, 14, 15)$

$= \Pi M(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 11)$



$AB + AC\bar{D}$



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

Lab III

Obj - 1

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00				
$\bar{A}B$ 01		1	1	1
AB 11				
$A\bar{B}$ 10	1	1	1	1

$A\bar{B} + \bar{A}BD + \bar{A}BC$

Obj. 2

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0		1	1	
A 1		1	1	1

$BC + AC + AB$

Obj - 3

A	B	C	D	Even parity
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

$x \oplus 0 = x$

$x \oplus 1 = \bar{x}$

$x \oplus x = 0$

$x \oplus \bar{x} = 1$

$(x \oplus y)' = (x \oplus y') = x' \oplus y$

$A \oplus B = B \oplus A$

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

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

$\Rightarrow A \oplus (B'C + BC')$

$\Rightarrow A'(B'C + BC') + A(B'C + BC')'$

$\Rightarrow A'(B'C + BC') + A(BC + B'C')$

$\Rightarrow A'B'C + A'BC' + ABC + AB'C'$

$= \Sigma m(1, 2, 4, 7)$

$A \oplus B \oplus C \oplus D$

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

$= (A'B + AB') \oplus (C'D + CD')$

$= (A'B + AB')(C'D + CD')' + (A'B + AB')'(C'D + CD)$

$= (A'B + AB')(CD + C'D') + (AB + A'B')(C'D + CD')$

$= A'BCD + A'BC'D' + AB'CD + AB'C'D' + ABC'D + ABCD' + A'B'C'D + A'B'CD'$

$= \cancel{\Sigma m(}$

$= m_7 + m_4 + m_{11} + m_8 + m_{13} + m_{14} + m_1 + m_2$

$= \Sigma m(1, 2, 4, 7, 8, 11, 13, 14)$

Combinational circuit

Design Procedure

1. From the specification of the circuit determine the required no. of input and output and assign symbol to each.
2. Derive the truth table that define the relationship between input and output
3. Obtain the boolean function for each output in term of input.
4. Draw the logic diagram and verify the correctness using truth table.

Code conversion

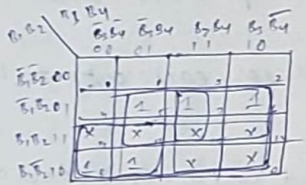
BCD → Excess-3

Excess-3 → BCD

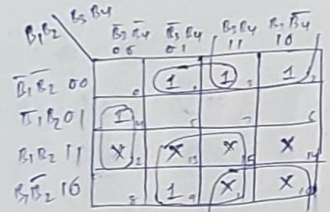
Gray code → Binary

Binary → Gray code

BCD	Excess-3
$B_1\ B_2\ B_3\ B_4$	$E_1\ E_2\ E_3\ E_4$
0 0 0 0	0 0 1 1
0 0 0 1	0 1 0 0
0 0 1 0	0 1 0 1
0 0 1 1	0 1 1 0
0 1 0 0	0 1 1 1
0 1 0 1	1 0 0 0
0 1 1 0	1 0 0 1
0 1 1 1	1 0 1 0
1 0 0 0	1 0 1 1
1 0 0 1	1 1 0 0

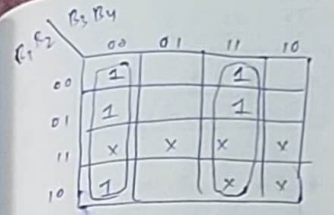


$E_1 = B_2\bar{B_3} + B_4 B_2 + B_1 \bar{B_3}$

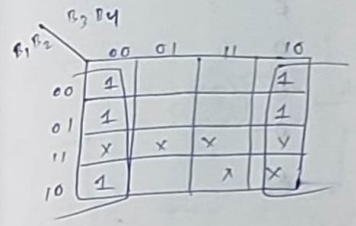


$E_2 = B_1 B_4 + \cancel{B_2 \bar{B_3} \bar{B_4}} + \bar{B_2} B_3 + \bar{B_1}\,\bar{B_2} B_4$

$= B_1 B_4 + \bar{B_2} B_3 + \bar{B_1}\,\bar{B_2} B_4$



$E_3 = \bar{B_3}\,\bar{B_4} + B_3 B_4$



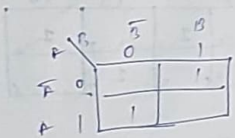
$E_4 = \bar{B_4}$

Adder

1. Half Adder
2. Full Adder

Half Adder

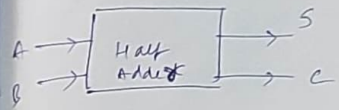
A	B	S (sum)	C (carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



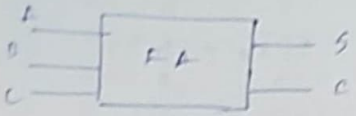
$S = A'B + AB'$

$= A \oplus B$

$C = AB$



Full Adder



A	B	C	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

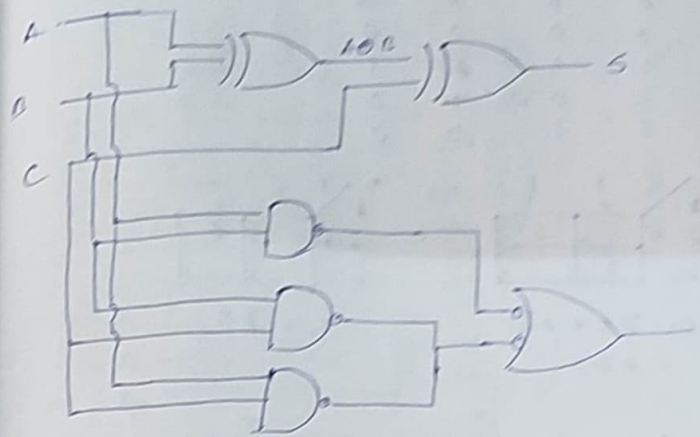
A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0		1		1
A 1	1		1	

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

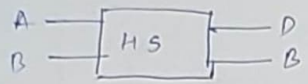
$= A \oplus B \oplus C$

A \ BC	$\bar{B}\bar{C}$ 00	$\bar{B}C$ 01	BC 11	$B\bar{C}$ 10
$\bar{A}$ 0		1	1	
A 1		1	1	1

$C = BC + AC + AB$

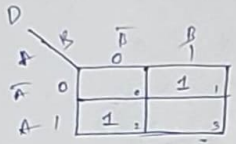


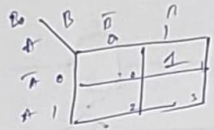
Half subtractor



$A - B = D$

A	B	D	B_0
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

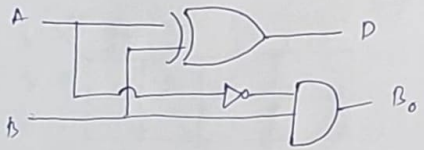




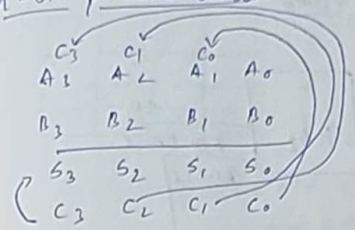
$D = \bar{A}B + A\bar{B}$

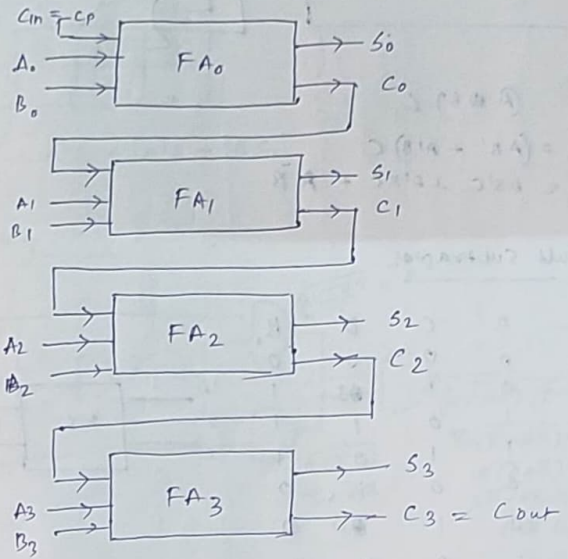
$= A \oplus B$

$B_0 = \bar{A}B$

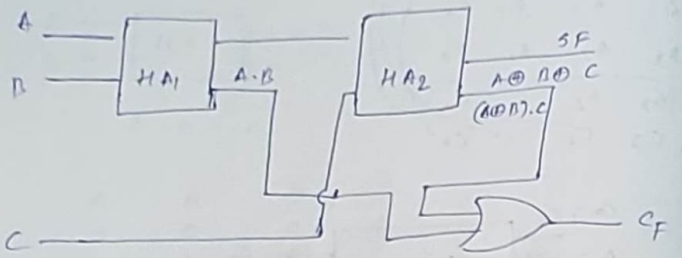


n-bit parallel adder





Design FA using HA circuits



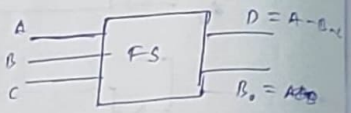
$(A \oplus B).C$

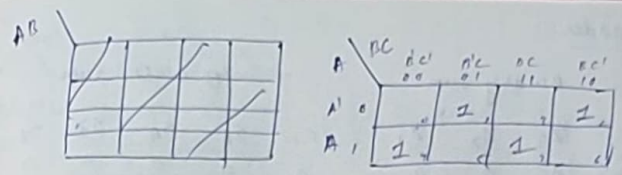
$= (AB' + A'B)C$

$= AB'C + A'BC + A \cdot B$

Full Subtractor

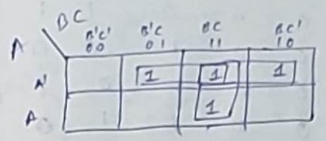
A	B	C	D	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1





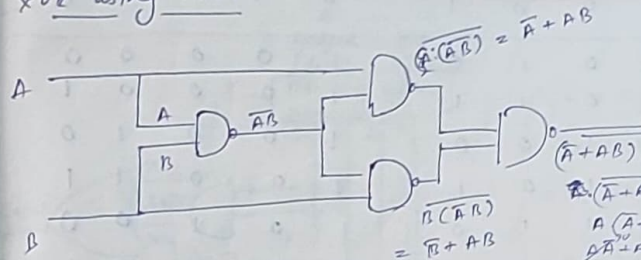
$D = A'B'C + AB'C' + ABC + A'BC'$

$= A \oplus B \oplus C$



$B_o = A'C + A'B + BC$

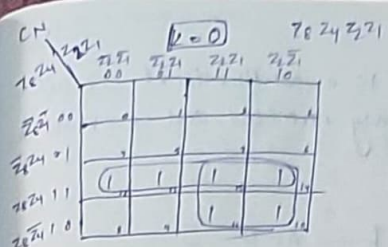
XOR using NAND

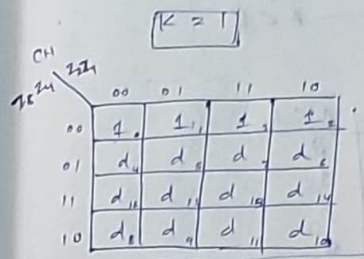


BCD adder

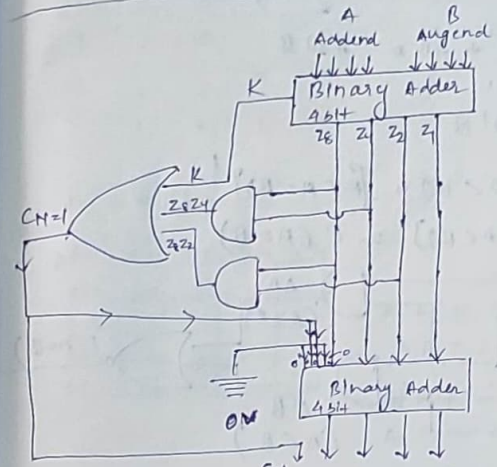
Binary sum

D	K	Z_8	Z_4	Z_2	Z_1	CN	S_8	S_4	S_2	S_1
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	0	0	0	0
19	1	0	0	1	1	1	1	0	0	1





$C_N = K + Z_8Z_4 + Z_8Z_2$



(BCD Adder)

$C_n = 0$ (sum ≤ 9)

$S_8\ S_4\ S_2\ S_1$

$= Z_8\ Z_4\ Z_2\ Z_1$

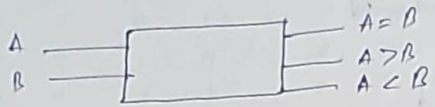
$C_n = 1$ (sum > 9)

$S_8\ S_4\ S_2\ S_1 + 0110$

$= Z_8\ Z_4\ Z_2\ Z_1$

② Magnitude Comparator

A magnitude comparator is a combinational circuit that compares two binary numbers and provides a result (=, >, <)



1 Bit - comparator

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

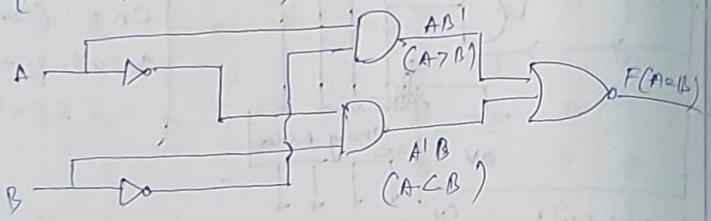
$F(A=B) = A'B' + AB = A \odot B$

$F(A>B) = AB'$

$F(A<B) = A'B$

$F(A>B) + F(A<B) = [F(A=B)]'$

or $[F(A>B) + F(A<B)]' = F(A=B)$



2 bit - comparator

A		B				
A_1	A_0	B_1	B_0	A = B	A > B	A < B
0	0	0	0	1	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	0	1	0
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	0	1	1	0	0	1
1	1	0	0	0	1	0
1	1	0	1	0	1	0
1	1	1	0	0	1	0
1	1	1	1	1	0	0

$F(A=B) =$

A_1A_0 \ B_1B_0	$B_1'B_0'$	B_1B_0'	B_1B_0	$B_1'B_0$
$A_1'A_0'$	1 0	1 1	2 3	2 2
A_1A_0'	4 5	1 6	7 7	6 6
A_1A_0	8 9	12 13	1 15	14 14
$A_1'A_0$	8 8	9 9	11 11	1 10

$= A_1'A_0'B_1'B_0' + A_1A_0'B_1B_0' + A_1A_0B_1B_0 + A'A_0B_1'B_0$

$= A_1'B_1'(A_0'B_0' + A_0B_0) + A_1B_1(A_0'B_0' + A_0B_0)$

$= (A_0'B_0' + A_0B_0)(A_1'B_1' + A_1B_1)$

$= (A_0 \odot B_0)(A_1 \odot B_1)$

F (A > B)

A_1A_0 \ B_1B_0	$B_1'B_0'$	$B_1'B_0$	B_1B_0	B_1B_0'
$A_1'A_0'$	0	1	3	2
$A_1'A_0$	1 4	5	7	6
A_1A_0	1 12	1 13	15	1 14
A_1A_0'	1 8	1 9	11	10

$A_1B_1' + A_0B_1'B_0' + A_1A_0B_0'$

F (A < B)

A_1A_0 \ B_1B_0	$B_1'B_0'$	$B_1'B_0$	B_1B_0	B_1B_0'
$A_1'A_0'$	0	1 1	1 3	1 2
$A_1'A_0$	4	5	1 7	1 6
A_1A_0	12	13	15	14
A_1A_0'	8	9	1 11	10

$A_1'B_1 + B_1B_0A_0' + A_1'A_0'B_0$

Q - What is the requirement of software simulation?

What is HDL?

HDL stands for Hardware Descriptive language

- It is a specialized computer language
- used to describe the structure and behaviour of electronic circuits.

Software to be used - Vivado Design suite

HDL operators

Some verilog HDL operator

Symbol	Operation
+	binary addition
−	binary subtraction
&	bitwise AND
|	bitwise OR
^	bitwise XOR
~	bitwise NOT
==	equality
>	greater than
<	less than
{}	concatenation
?:	conditional
&&	logical AND
||	logical OR
!	logical NOT

Conditional logic modified with boolean equation

$E = A + BC + B'D$

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

// verilog model circuit with Boolean expression

```
module_circuit_Boolean_CA (E,F,A,B,C,D);
  Output E,F;
  input A,B,C,D;
  assign E = A || (B&&C) || ((!B)&&D);
  assign F = ((!B)&&C) || ((B&&(C) &&(!D)));
  end module;
```

$O = S_1\bar{S}_0D + \bar{S}_1S_0D + ACE$

```
module Mitn (S1, S0, A,C,D,E,O);
  output O;
  input S1, S0, A,C,D,E;
  assign O = (S1&&(!S0)&&D) || ((!S1)&&S0&&D) ||
             ((A&&C)&&E)

end module
```

steps to execute any program using Vivado

1. ~~Boot~~ Create project, Next write Project Name, RTL (Register Tranfer level)
2. Add source: target and simulator language: verilog
3. Add constraint, nil, next
4. Default path: select board: set board
5. Add source: add or create design sources
6. Create File: Filename Full_adder, Finish
7. Define module: I/O, port definitions, ok yes, updating.
8. ~~Yes~~ in sources: design sources half adder module is added
9. Double click: prog ram adder will come
10. Double click in done: save for errors
11. verilog code is done: save: see for errors

4-bit magnitude comparator

$A = A_3\ A_2\ A_1\ A_0$

$B = B_3\ B_2\ B_1\ B_0$

$A = B \Rightarrow A_3 = B_3$

$\& \ A_2 = B_2$

$\& \ A_1 = B_1$

$\& \ A_0 = B_0$

$(A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)'(A_0 \oplus B_0)'$

$A > B \Rightarrow A = A_1 A_0$

$B = B_1 B_0$

1) $A_1 > B_1 \longrightarrow A_1 B_1'$

2) $A_1 = B_1$ $A_0 = 1$ $\longrightarrow A_1 A_0 B_0' + A_0 B_1' B_0'$

$B_0 = 0$

$A < B \Rightarrow A = A_1 A_0$

$B = B_1 B_0$

1) $A_1 < B_1 \longrightarrow B_1 A_1'$

2) $A_1 = B_1 \longrightarrow A_0' B_0 B_1 + A_0' B_0 A_1'$

$A_0 = 0$

$B_0 = 1$

① $A > B \Rightarrow A_3 > B_3$, $A_2 > B_2$ $A_1 > B_1$ $A_0 > B_0$

$A \rightarrow A_3\ A_2\ A_1\ A_0$

$B \rightarrow B_3\ B_2\ B_1\ B_0$

This comparator is used to compare binary nos 4 bits each and provides a condition in the output

Case 1) If $A_3 = 1$ & $B_3 = 0$

2) If $A_3 = B_3$ & $A_2 = 1$, $B_2 = 0$

3) If $A_2 = B_3$, $A_2 = B_2$ & $A_1 = 1$, $B_1 = 0$

4) If $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$ & $A_0 = 1$, $B_0 = 0$

② $A < B$

Case 1) If $A_3 = 0$ & $B_3 = 1$

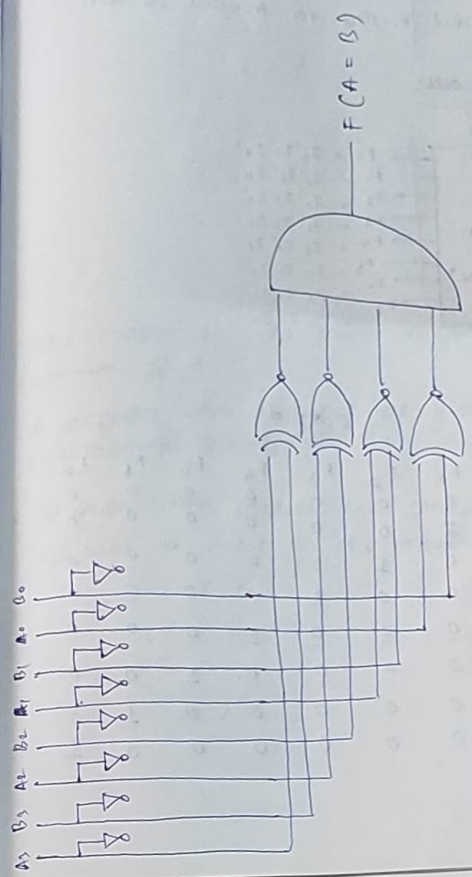
2) If $B_3 = A_3$ & $A_2 = 0$, $B_2 = 1$

3) If $B_3 = A_3$, $B_2 = A_2$ & $A_1 = 0$, $B_1 = 1$

4) If $B_3 = A_3$, $B_2 = A_2$, $B_1 = A_1$ & $A_0 = 0$, $B_0 = 1$

$A > B$: $F = A_3 B_3' + (A_3 \oplus B_3)' A_2 B_2' + (A_3 \oplus B_3)'(A_2 \oplus B_2)' A_1 B_1' + (A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)' A_0 B_0'$

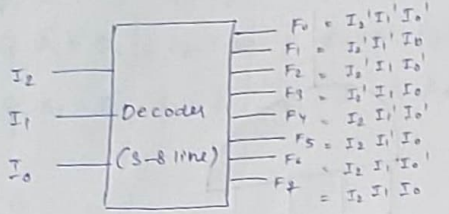
$A < B$: $F = A_3' B_3 + (A_3 \oplus B_3)' A_2' B_2 + (A_3 \oplus B_3)'(A_2 \oplus B_2)' A_1' B_1 + (A_3 \oplus B_3)'(A_2 \oplus B_2)'(A_1 \oplus B_1)' A_0' B_0$



Decoder

- Decoder is a combinational circuit that converts the binary information from n input lines to max 2^n - output lines.
- Every decoder is called n-line to m-line decoder

3-line to 8-line decoder

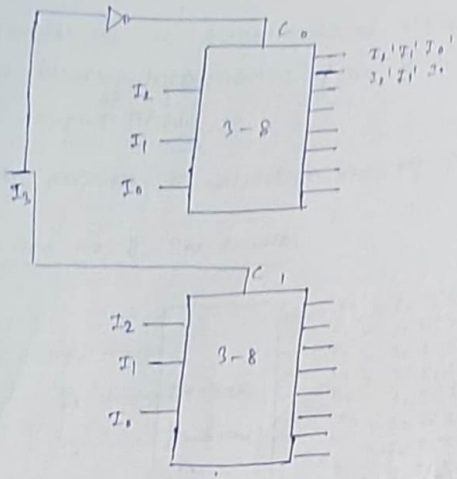


Truth table :

Inputs			Outputs							
I_2	I_1	I_0	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_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

- The output of the basic decoder represents the min term of SOP
- A decoder with external OR gate can design any combinational circuit
- Decoder with control line can function as De-Multiplexer and used in most of the ~~real life~~ digital applications.

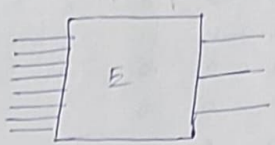
4 - 16 decoder



Encoder

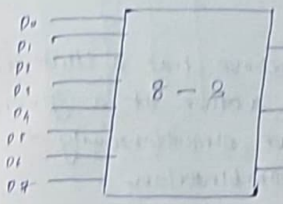
- It accepts 2^n lines as input and n lines of output.
- m - i/p to n - o/ps
- m - n encoder, $m \leq 2^n$

ex:



8 - 3 encoder

On encoder the output lines generate the binary codes corresponding to the input.



Truth table

Inputs								outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$A = \Sigma(4, 5, 6, 7) = D_4 + D_5 + D_6 + D_7$

$B = \Sigma(2, 3, 6, 7) = D_2 + D_3 + D_6 + D_7$

$C = \Sigma(1, 3, 5, 7) = D_1 + D_3 + D_5 + D_7$

Q - Design a 4 - 2 encoder (Assignment)

Limitations

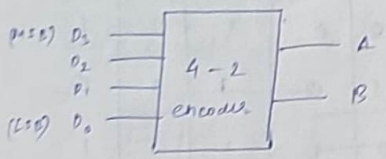
1) The encoder defined above has a limitation that only one input can be active at a given time. If two inputs are active simultaneously the output provides an undefined combination

2) To resolve this ambiguity encoder must establish an input priority to ensure only one input is encoded.

Priority Encoder

The priority encoder is an encoder circuit which includes the priority function.

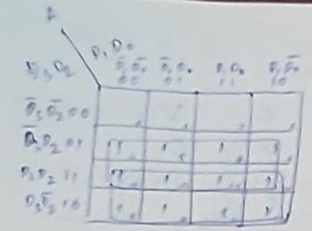
4 - 2 priority encoder



Truth table

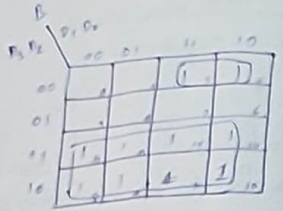
Inputs				outputs		
D_3	D_2	D_1	D_0	A	B	V
0	0	0	0	X	X	0
0	0	0	1	0	0	1
0	0	1	0	0	1	1
0	0	1	1	0	1	1
0	1	0	0	1	0	1
0	1	0	1	1	0	1
0	1	1	0	1	0	1
0	1	1	1	1	0	1
1	0	0	0	1	1	1
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	1	1	1
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	1	1	1

A



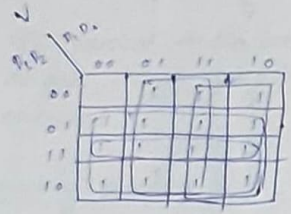
$A = D_2 + D_3$

B



$B = D_3 + D_2' D_1$

V



$V = D_3 + D_2 + D_1 + D_0$

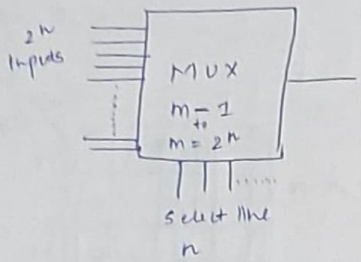
Gate level modelling

```
module half_Add (
    input a;
    input b;
    output s;
    output carry;
);
xor g1(sum, a, b)
and g2(carry, a, b)
end module
```

```
module full_Add (
    input a;
    input b;
    input c;
    output sum;
    output carry;
);
wire w1, w2, w3;
```

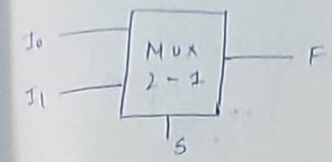
Multiplexer

It is a combinational circuit which selects binary information from one of many input lines and directs to a single output line.



- The selection of the particular ~~select~~ input line is controlled by a set of selection line.
- Generally a MUX is called as $n-1$ MUX where 2^n inputs and n select lines
- Any $(m-1)$ MUX / (2^n-1) MUX can be constructed from $n-2^n$ decoder
- Any MUX can be designed using AND gate with 2^n input lines and all the outputs of AND gate are applied to an OR

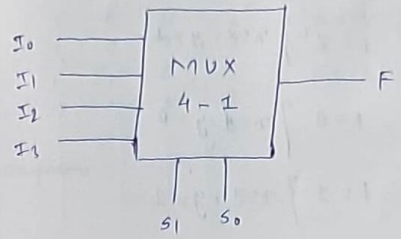
2-1 MUX



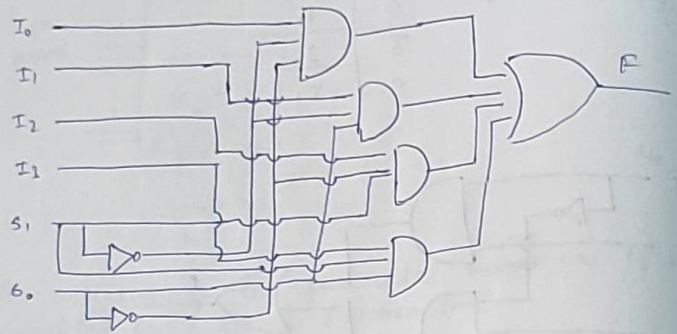
I_1 I_0 F

S	F
$\bar{S}$	I_0
S	I_1

4-1 MUX

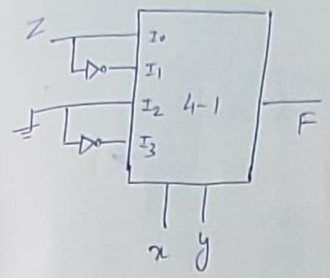


S_1	S_0	F
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



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

x	y	z	F	Condition
0	0	0	0	$F = z$ } $x=0, y=0$
0	0	1	1	
0	1	0	1	$F = z'$ } $x=0, y=1$
0	1	1	0	
1	0	0	0	$F = 0$ } $x=1, y=0$
1	0	1	0	
1	1	0	1	$F = 1$ } $x=1, y=1$
1	1	1	1	

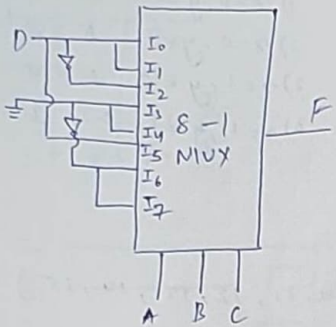


1) $x=0, y=0$ $F = z$
2) $x=0, y=1$ $F = z'$
3) $x=1, y=0$ $F = 0$
4) $x=1, y=1$ $F = 1$

Q) $F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$ using 8-1 MUX

A	B	C	D	F	Condition
0	0	0	0	0	$F = D$ } $A=0, B=0, C=0$
0	0	0	1	1	
0	0	1	0	0	$F = D$ } $A=0, B=0, C=1$
0	0	1	1	1	
0	1	0	0	1	$F = D'$ } $A=0, B=1, C=0$
0	1	0	1	0	
0	1	1	0	0	$F = 0$ } $A=0, B=1, C=1$
0	1	1	1	0	
1	0	0	0	0	$F = 0$ } $A=1, B=0, C=0$
1	0	0	1	0	
1	0	1	0	0	$F = D$ } $A=1, B=0, C=1$
1	0	1	1	1	
1	1	0	0	1	$F = 1$ } $A=1, B=1, C=0$
1	1	0	1	1	
1	1	1	0	1	$F = 1$ } $A=1, B=1, C=1$
1	1	1	1	1	

~~Q Implem~~



Q - Implement the given boolean f^n using 8X1 MUX
$F(A,B,C,D) = \Sigma(0,1,2,3,4,8,9,15)$

Q - Implement a full adder using 2 8X1 MUX

Q - Implement a full adder using 4 X 1 MUX

Q - Construct a 16X1 MUX using 2 8X1 and 1 2X1

VHDL

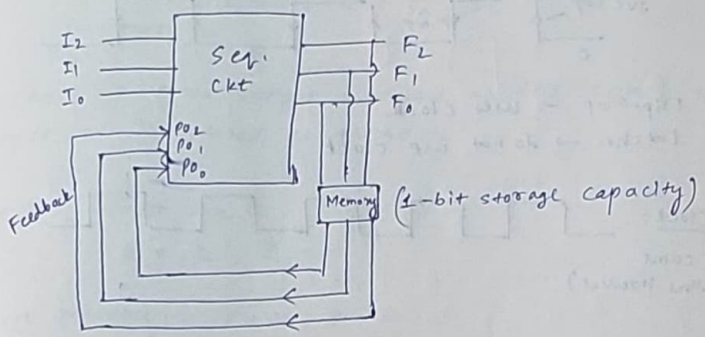
Verilog HDL is one of the most common hardware description language used by the IC designers.

variables not allowed → $, &, @, -, %

Keywords → assign, case, wire, reg, and, or, nand, module

Sequential Circuits

It is a circuit where the present output depends on present inputs as well as past output/outputs



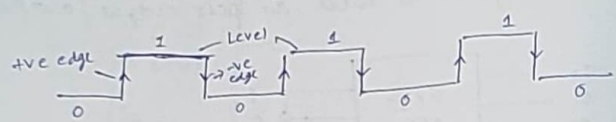
Feedback

∴ Combinational + Memory = Sequential Circuit
∴ "BUFFER" Ckt ⇒ where inputs is equal to output

Latch & Flip Flop

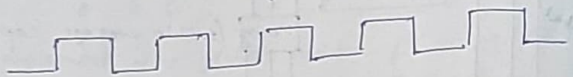
- A Latch is the basic storage element.
- A latch only stores a single-bit (0/1)
- A storage element in a digital ckt can maintain a binary stage indefinitely until power off or directed by input to change or switch states.
- A latch is a level sensitive device where Flip-Flop is an edge sensitive (+ve edge triggering or -ve edge "

* Storage elements that operates with signal level are called as latches whereas controlled by clock translation is called as Flip-Flop



Flip-Flop → uses clock
Latches → do not use clock

Clock:

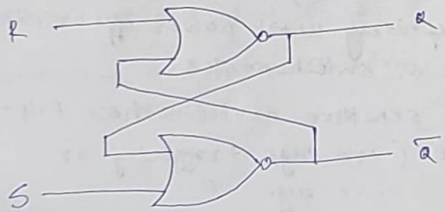


(same time Interval)

SR Latch

- It can be designed using NOR and NAND
- It needs two cross coupled NAND/NOR gate
- 'S' stands for Set
  'R' stands for Reset

SR Latch using NOR



S	R	Q	$\overline{Q}$	
0	0	Prev Q	Prev $\overline{Q}$	→ Memory
0	1	0	1	→ Reset
1	0	1	0	→ Set
1	1	Not used		

(Final table)

Case I : When R = 1, S = 0
Q = 0, $\overline{Q}$ = 1

Next Instance :- R = 0, S = 0
Q = 0, $\overline{Q}$ = 1
(Memory) ⇒

Because previous output remains as the same output it is stored as Memory

Case II : When R = 0, S = 1
Q = 1, $\overline{Q}$ = 0

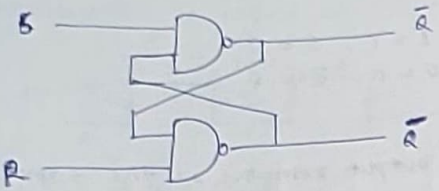
Next Instance :- R = 0, S = 0
Q = 1, $\overline{Q}$ = 0
(Memory) ⇒

Case III : When R = 1, S = 1
Q = 0, $\overline{Q}$ = 0

Q & $\overline{Q}$ should be complement of each other but it does not satisfies

Next Instance :- when R = 0, S = 0
Q = 1, $\overline{Q}$ = 0 ⇒ Not possible

SR Latch using NAND



NAND

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Case I: When $S = 1$, $R = 0$

$Q = 1$, $\bar{Q} = 0$

Next Instance: $S = 1$, $R = 1$

$Q = 1$, $\bar{Q} = 0$

No change (Memory)

Case II: when $S = 0$, $R = 1$

$Q = 0$, $\bar{Q} = 1$

Next Instance: $S = 1$, $R = 1$

$Q = 0$, $\bar{Q} = 1$

No change (memory)

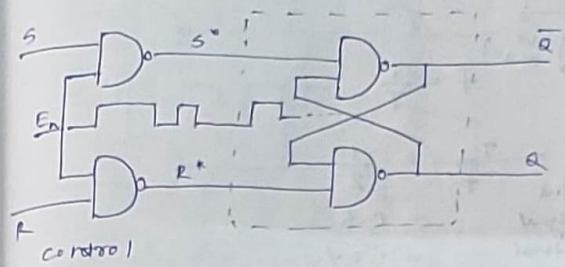
Case III: when $S = 0$, $R = 0$

$Q = 1$, $\bar{Q} = 1$ ⇒ Not possible

(~~Inter~~ Indeterminate / Not used / Forbidden)

S	R	Q	$\bar{Q}$	
0	0	Not used		
0	1	0	1	→ Reset
1	0	1	0	→ Set
1	1	Prev Q	Prev $\bar{Q}$	→ Memory (No change)

SR Latch with Control (Enable Input)



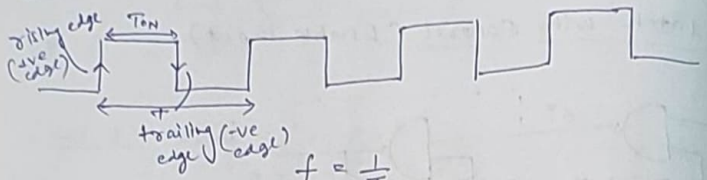
S^*	R^*	Q	$\bar{Q}$	State
0	0	1	1	Not used
0	1	0	1	Reset
1	0	1	0	Set
1	1	Prev Q	Prev $\bar{Q}$	Memory (No change)

En	S	R	Output
0	x	x	No change (Memory)
1	0	0	No change (Memory)
1	0	1	$Q = 1$, $\bar{Q} = 0$ (Set)
1	1	0	$Q = 0$, $\bar{Q} = 1$ (Reset)
1	1	1	Indeterminate

- Enable (control) is a level triggered signal

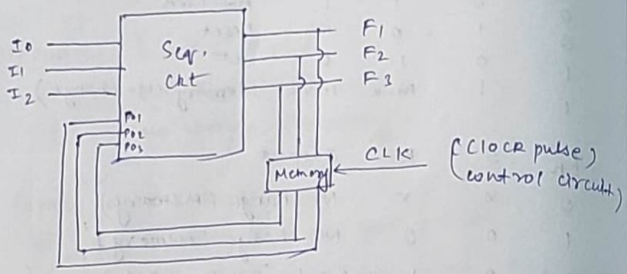
* In case of SR flipflops, the enable pin is replaced with a clock to make it a flip flop

Clock

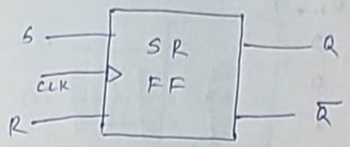


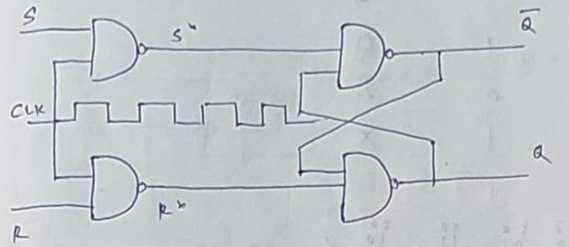
$$f = \frac{1}{T}$$

Duty cycle $= \frac{T_{ON}}{T} = \frac{1}{2}$
(ratio of the time for which the signal is high to the total time) $= 0.5$
$= 50\%$



SR Flip-Flops





$$S^* = \overline{(S \cdot CLK)} = \overline{S} + \overline{CLK}$$
$$R^* = \overline{R \cdot CLK} = \overline{R} + \overline{CLK}$$

Truth table for SR-FF

CLK	S	R	Q & $\bar{Q}$	Qn+1
0	x	x	No change (memory)	Qn
1	0	0	Memory	Qn
1	0	1	Q = 0, $\bar{Q}$ = 1	0
1	1	0	Q = 1, $\bar{Q}$ = 0	1
1	1	1	Not used	Invalid

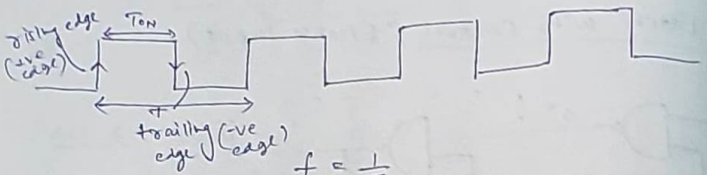
Qn → Present state

Q n+1 → Next state

- Enable (control) is a level triggered signal

* In case of SR flipflops, the enable pin is replaced with a clock to make it a flip flop

Clock

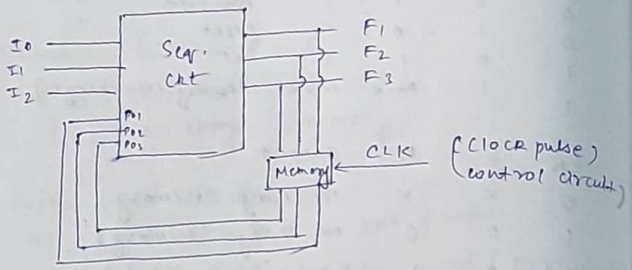


$$f = \frac{1}{T}$$

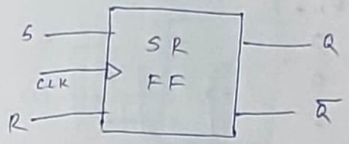
Duty cycle $= \frac{T_{ON}}{T} = \frac{1}{2}$

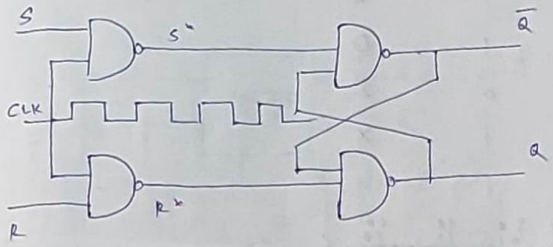
(ratio of the time for which the signal is high to the total time) $= 0.5$

$= 50\%$



SR Flip-Flops





$$S^* = \overline{(S \cdot CLK)} = \overline{S} + \overline{CLK}$$

$$R^* = \overline{R \cdot CLK} = \overline{R} + \overline{CLK}$$

Truth table for SR-FF

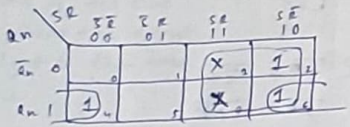
CLK	S	R	Q & $\bar{Q}$	Q_{n+1}
0	x	x	No change (Memory)	Q_n
1	0	0	Memory	Q_n
1	0	1	$Q = 0, \bar{Q} = 1$	0
1	1	0	$Q = 1, \bar{Q} = 0$	1
1	1	1	Not used	Invalid

$Q_n \rightarrow$ Present state

$Q_{n+1} \rightarrow$ Next state

Characteristic table

Q_n	S	R	Q_{n+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	X
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	X



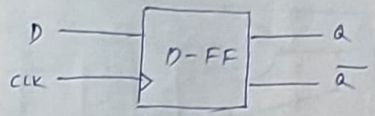
$$\boxed{Q_{n+1} = S + Q_n\bar{R}}$$

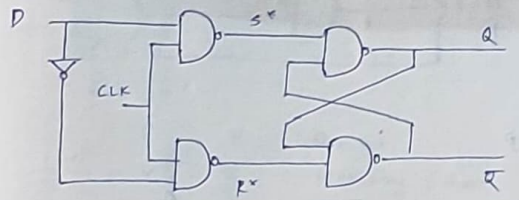
Excitation table

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

D - Flip Flops (Delay Flip-Flop / Data-Flip-Flop)

(D-FF from SR-FF)



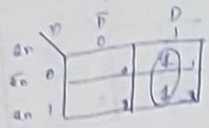


Truth table for D-FF

CLK	D	Q & $\bar{Q}$	Q_{n+1}	
0	X	No change (memory)	Q_n	
1	0	$Q=0, \bar{Q}=1$	0	(Reset)
1	1	$Q=1, \bar{Q}=0$	1	(Set)

Characteristic table

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

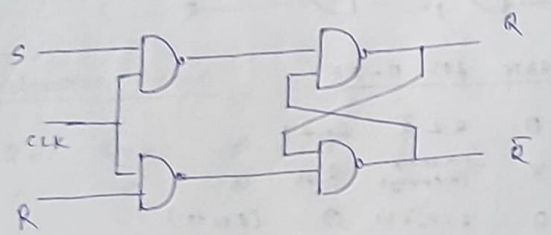


$Q_{n+1} = D$

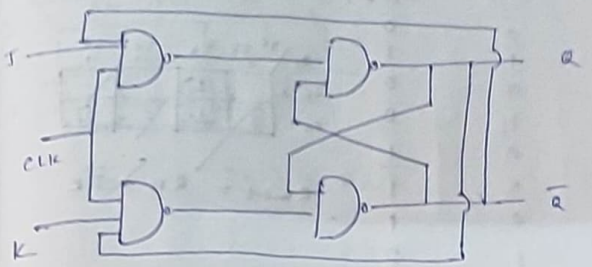
Excitation table

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

JK Flip Flop



CLK	S	R	Q_{n+1}
0	x	x	Q_n (Memory)
1	0	0	Q_n (Memory)
1	0	1	0
1	1	0	1
1	1	1	Not used



CLK	J	K	Q_{n+1}
0	x	x	Memory (Q_n)
1	0	0	Memory (Q_n)
1	0	1	0
1	1	0	1
1	1	1	TOGGLE ($\bar{Q}_n$)

$Q_{n+1} = \bar{Q}_n$ (Toggle)

$Q_{n+1} = Q_n$ (Memory)

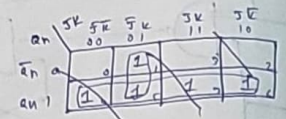
Assume : Q , $\bar{Q}$

$Q = 0$, $\bar{Q} = 1$

$(J = 1 , K = 1 , CLK = 1)$

Characteristic table

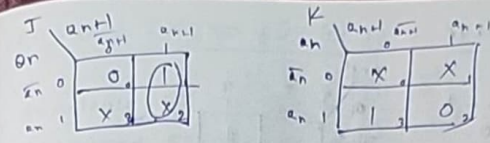
J	K	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



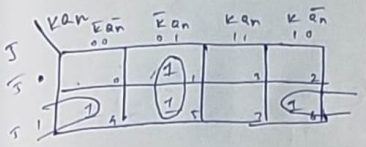
$Q_{n+1} = Q_n + \bar{J}K$

Excitation table

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

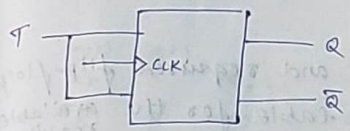


$J = Q_n + 1$ $\qquad$ $K = \overline{Q_n} + 1$



$Q_{n+1} = \overline{Q_n} J + Q_n \overline{K}$

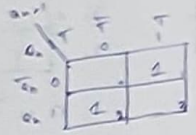
T- Flip-Flop



CLK	T	Qn+1
0	X	Memory (Qn)
1	0	Memory (Qn)
1	1	Toggle ($\overline{Q_n}$)

Characteristic table

Q_n	T	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0



$$Q_{n+1} = T\bar{Q}_n + \bar{T}Q_n$$

Excitation table

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Flip-Flop Conversion

Steps

1. Identify the available and required flip-flop.
2. Draw the characteristic table for the ~~available~~ required Flip-Flop
3. Draw the excitation table for the ~~available~~ ~~required~~ flip flop
4. Write the Boolean expression for the available flip-flop
5. Draw the circuit

Convert JK to D-FF

SR → T
SR → JK
T → D

1. Available FF = JK FF
   Required FF = D FF

2. Characteristic table (D-FF)

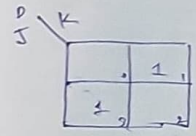
Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

3. Excitation table (JK-FF)

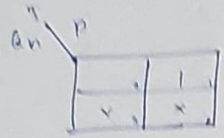
Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

4.

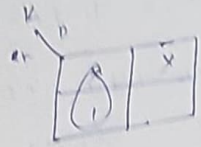
(I/P) Q_n	(I/P) D	Q_{n+1}	(O/P) J	(O/P) K
0	0	0	0	x
0	1	1	1	x
1	0	0	x	1
1	1	1	x	0



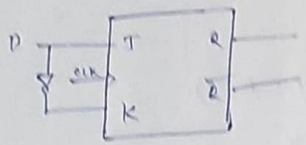
* The required flip-flop will be taken as input variable and the available flip-flop will be taken as output variable



$T = D$



$K = \bar{D}$



SR to T

① Available FF = SR FF
Required FF = T FF

② Characteristic table (T-FF)

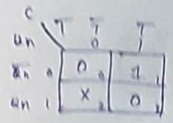
Qn	T	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

③ Excitation table (SR-FF)

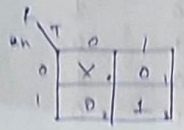
Qn	Qn+1	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

④

Qn (i/p)	T (i/p)	Qn+1	S (o/p)	R (o/p)
0	0	0	0	x
0	1	1	1	0
1	0	1	x	0
1	1	0	0	1

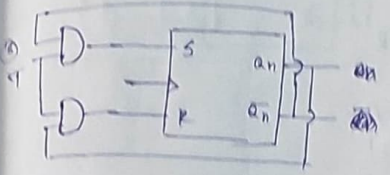


$S = T\bar{Q}_n$



$R = Q_n T$

⑤



SR to JK

① Available FF = SR-FF
Required FF = JK-FF

② Characteristic table (JK-FF)

Qn	J	K	Qn+1
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

③ Excitation table (SR-FF)

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

④

Qn	J	K	Qn+1	S	R
0	0	0	0	0	X
0	0	1	0	0	X
0	1	0	1	1	0
0	1	1	1	1	0
1	0	0	1	X	0
1	0	1	0	0	1
1	1	0	1	X	0
1	1	1	0	0	1

S

Qn \ JK	00	01	11	10
0	0	0	1	1
1	X	0	0	X

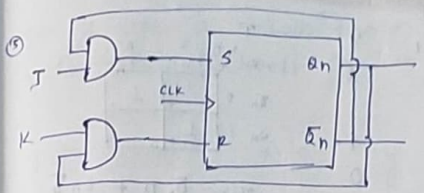
$S = \bar{Q}_n J$

R

Qn \ JK	00	01	11	10
0	X	X	0	0
1	0	1	1	0

$R = Q_n K$

⑤



T to D

① Available FF = T-FF
Required FF = D-FF

② Characteristic table (D-FF)

Q_n	D	Q_{n+1}
0	0	0
0	1	1
1	0	0
1	1	1

③ Excitation table (T-FF)

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

④

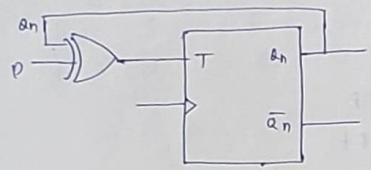
Q_n	D	Q_{n+1}	T
0	0	0	0
0	1	1	1
1	0	0	1
1	1	1	0

T	D	$\bar{D}$
Q_n		
$\bar{Q}_n$ 0	0	1
Q_n 1	1	0

$T = \bar{Q}_n D + Q_n \bar{D}$

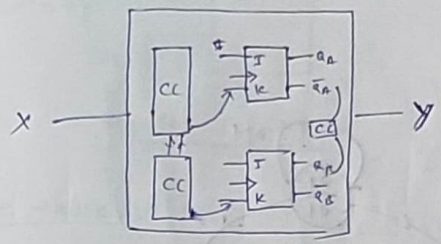
$= Q_n \oplus D$

⑤



Introduction to State table, state diagram and state equation

→ used in Design & Analysis of sequential circuit



State Table

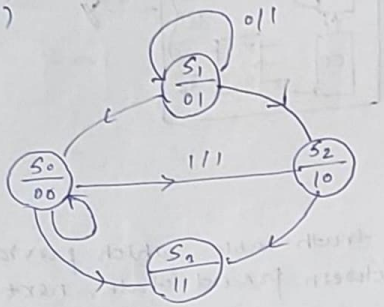
It is like a truth table which provides a relationship between present state, next state, real

P.S		x	N.S		y
$Q_A^{(t)}$	$Q_B^{(t)}$		$Q_A^{(t+1)}$	$Q_B^{(t+1)}$	
0	0	1	1	0	1

State Diagram

It is a graphical representation of state table
There are 2^n states for n Flip-flops.

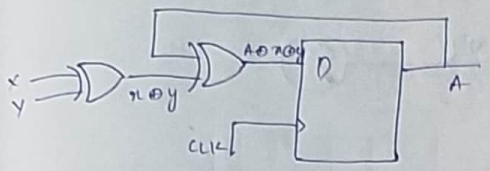
S_0 (00)
S_1 (01)
S_2 (10)
S_3 (11)



State equation

It should have L.H.S and R.H.S
The L.H.S will contain the next state
The R.H.S will contain the present state and inpu

q. Design and do the circuit analysis of a seq. ckt using D-FF



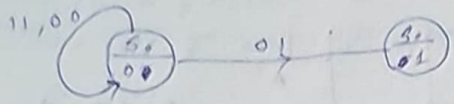
Seq ckt (D-FF)

$D = A \oplus x \oplus y$
$A(t+1) = D$
$= A(t) \oplus x \oplus y$

State table

Present state A(t)	I/P x	I/P y	Next state A(t+1)	O/P
0	0	0	0	
0	0	1	1	
0	1	0	1	
0	1	1	0	
1	0	0	1	
1	0	1	0	
1	1	0	0	
1	1	1	1	

State Diagram



State equation

$D = A \oplus x \oplus y$

$A(t+1) = D$

$= A(t) \oplus x \oplus y$

Mealy & Moore State Machines

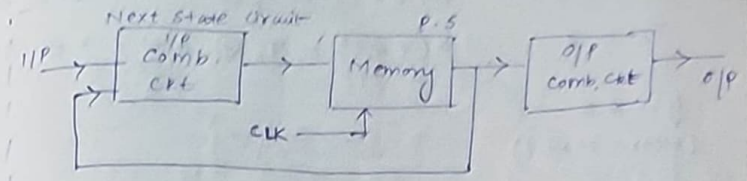
There are two models developed for representing sequential circuit :

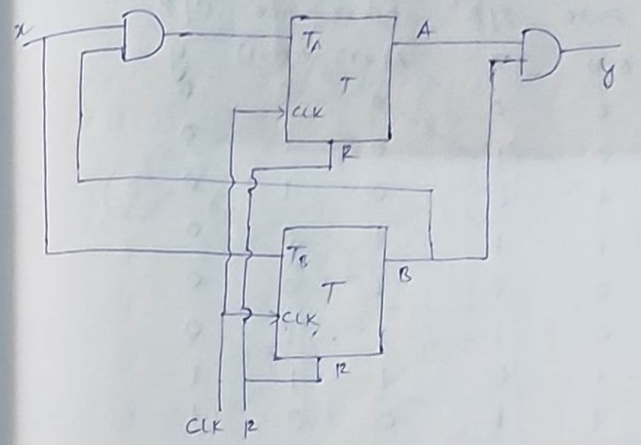
1) Moore circuit / Moore state Machine
2) Mealy circuit / Mealy state Machine

The identification of state machine is only achieved by how the output is generated

1) Moore State Machine - Here the output depends on or is a function of present state only. It never depends on the input

O/P α Present State & I/P





T-FF ⇒ $Q(t+1) = T \oplus Q$

For the ckt

$T_A = Bx$ $\quad A(t+1) = T_A \oplus A(t) = Bx \oplus A$

$T_B = x$ $\quad B(t+1) = T_B \oplus B(t) = x \oplus B$

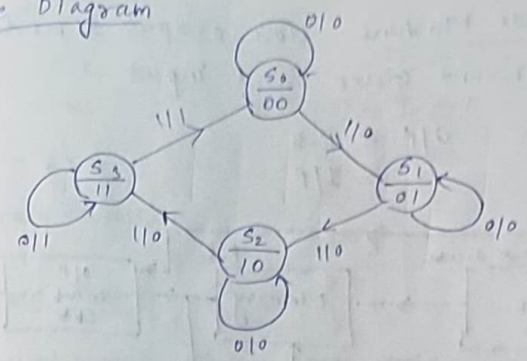
$y = A.B$

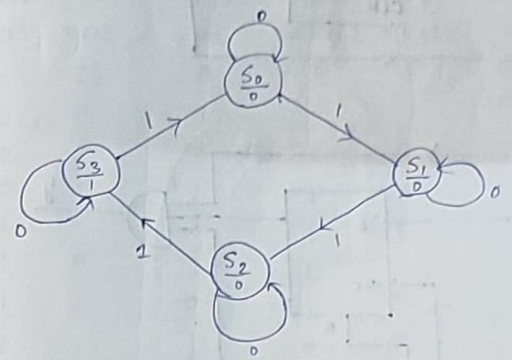
(Moore Model)

State Table

Present state		I/P	Next state		O/P
A(t)	B(t)	x	A(t+1)	B(t+1)	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	1

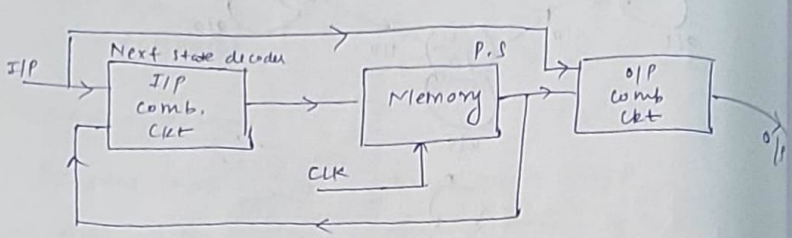
State Diagram

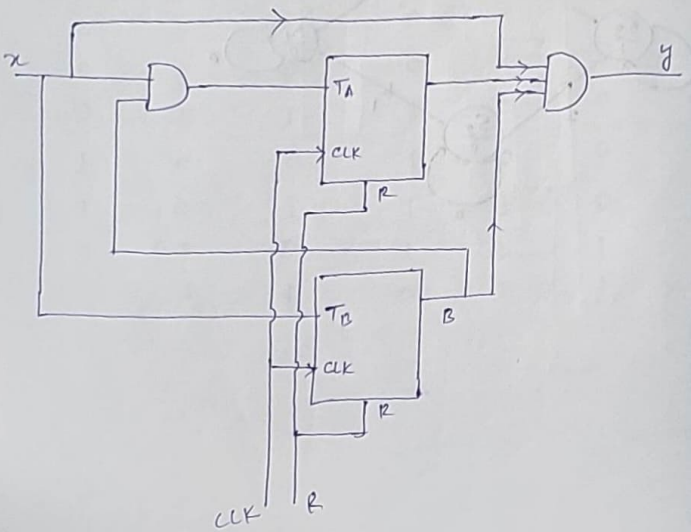




2) Mealy state Machine - Here output depends on both present state and input

O/P ∝ P.S
∝ I/P





For the ckt

$T_A = Bx$

$T_B = x$

$y = ABx$

(Mealy Model)

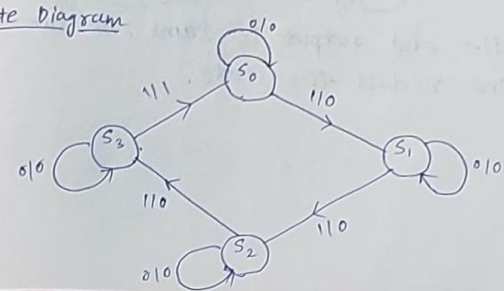
$A(t+1) = T_A \oplus A = Bx \oplus A$

$B(t+1) = T_B \oplus B = x \oplus B$

State Table

Present state		I/P	Next state		O/P
A(t)	B(t)	x	A(t+1)	B(t+1)	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	1	0
1	1	1	0	0	1

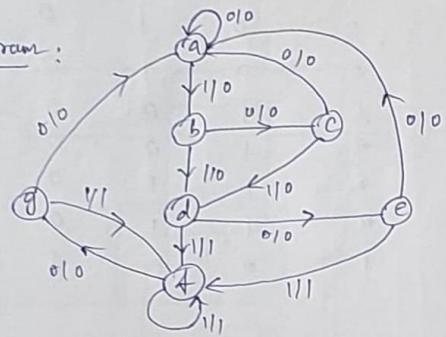
State Diagram



State reduction and assignment

P.S	N.S $x=0$	N.S $x=1$	output $x=0$	output $x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

State diagram:



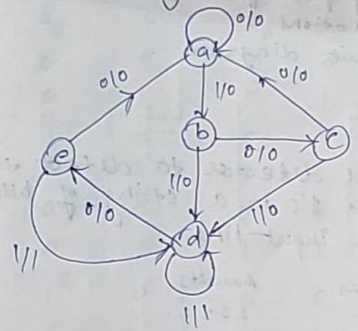
Note: If N.S and output is same, we can eliminate the P.S and reduce the state.

Reduced state

$e = g$ and $d = f$

P.S	N.S $x=0$	N.S $x=1$	output $x=0$	output $x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e=g	f=d	0	1
e	a	f=d	0	1

Reduced state diagram



Design procedure : a clocked sequential circuit

The procedure for designing synchronous seq ckt can be summarized by a list of recommended steps

- From the word description and specifications of the desired operation, derive a state diagram for the circuit
- Reduce the no. of ~~steps~~ states if necessary
- Assign binary values to the states
- Obtain the binary-coded state table
- Choose the type of flip-flop to be used
- Derive the simplified flip-flop input equations and output equations
- Draw the circuit diagram

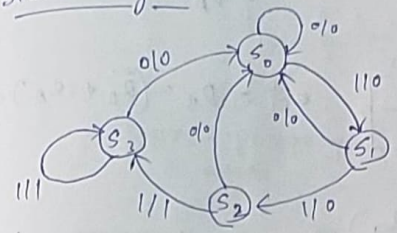
Sequence detector

Design a sequence detector to detect three or more consecutive 1's in a string of bits coming through an input line.

		Has	Awaits
S_0 =	reset	--	111...
S_1 =	1	1	11...
S_2 =	11	11	1....
S_3 =	111	111	....

ex:- x = 0 0 1 1 1 0 1 1 1 1 0 ...
y = 0 0 0 0 1 0 0 0 1 1 0 ...

State diagram



State table

P.S		Inputs	N.S.		Output
Q_A	Q_B	x	Q_A^+	Q_B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	1	1	1

State assignment :-

$S_0 = 00$ $S_2 = 10$
$S_1 = 01$ $S_3 = 11$

Q_A \ $Q_B x$	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$$Q_A^+ = Q_A x + Q_B x$$
$$= (Q_A + Q_B) x$$
$$= D_A$$

Q_B^+ \ $Q_A x$	00	01	11	10
0	0	1	0	0
1	0	1	1	0

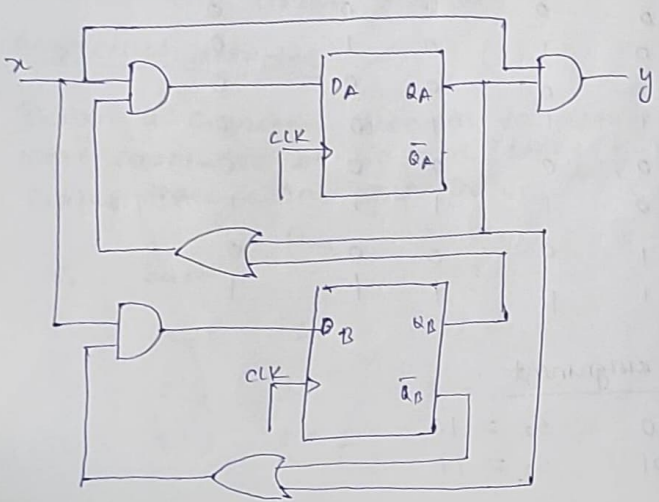
$$Q_B^+ = D_B = (\bar{Q}_B + Q_A) x$$

y \ $Q_A x$	00	01	11	10
0	0	0	0	0
1	0	1	1	0

$$y = Q_A x$$

$$N.S = D, \; D_A = Q_A^+, \; D_B = Q_B^+$$

State Diagram

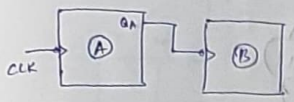


Counters

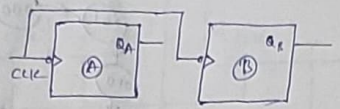
- A special type of sequential circuit used to count the pulse is known as a counter

Types of counters

- Ripple / Asynchronous Counters
- Synchronous counters



serial clock



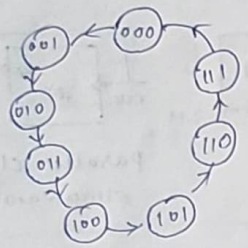
parallel clock and clocked simultaneously

Asynchronous / Ripple Counter	Synchronous Counter
1. Flip-Flops are connected in such a way that the o/p of first flip-flop drives the clock of next flip flop	1. There is no connection b/w o/p of first flip-flop and clock of next flip-flop.
2. Flip-Flops are not clocked simultaneously	2. Flip-Flops are clocked simultaneously
3. Circuit is simple for more number of states	3. Circuit becomes complicated as number of state increases
4. Speed is slow as clock is propagated through no. of stages	4. Speed is high as clock is given at a same time

Counters (Asyn. / Syn.)

- Up counters: 0-1-2-3 . . . .
- Down Counter: 7-6-5 0
- Up / Down Counter

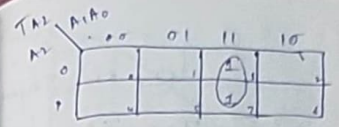
Design of 3 Bit binary up Counter : Synthesis using T-Flip Flop (Synchronous)



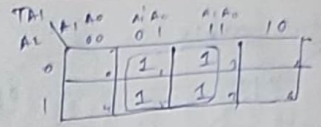
State diagram of three bit binary counter

State table

I/P (P.S)			O/P (N.S)			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A_2}	T_{A_1}	T_{A_0}
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

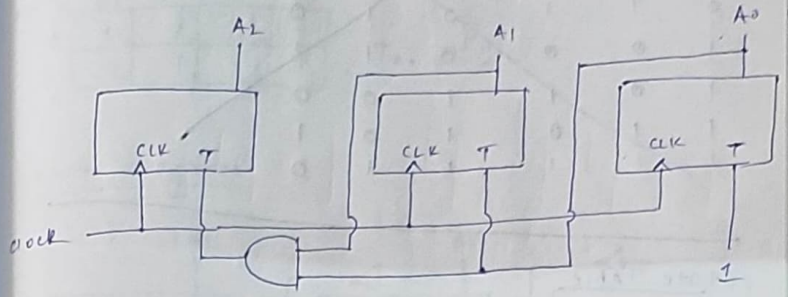


$T_{A_2} = A_1 A_0$

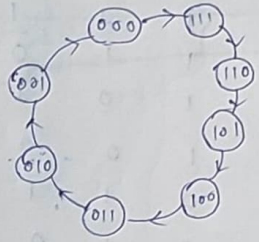


$T_{A_1} = A_0$

$T_{A_0} = 1$



Design of 3-bit binary down counter (Synchronous)



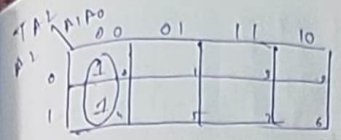
State diagram

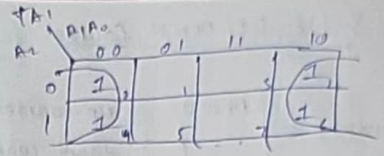
State table

I/P			O/P			Flip Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	T_{A_2}	T_{A_1}	T_{A_0}
0	0	0	1	1	1	1	1	1
0	0	1	0	0	0	0	0	1
0	1	0	0	0	1	0		
0	1	1	0	1	0			
1	0	0	0	1	1			
1	0	1	1	0	0			
1	1	0	1	0	1			
1	1	1	1	1	0			

State Table

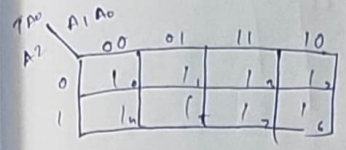
I/P (P.S)			O/P (N.S)			Flip-Flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	A_2	A_1	A_0
0	0	0	1	1	1	1	1	1
1	1	1	1	1	0	0	0	1
1	1	0	1	0	1	0	1	1
1	0	1	1	0	0	0	0	1
1	0	0	0	1	1	1	1	1
0	1	1	0	1	0	0	0	1
0	1	0	0	0	1	0	1	1
0	0	1	0	0	0	0	0	1





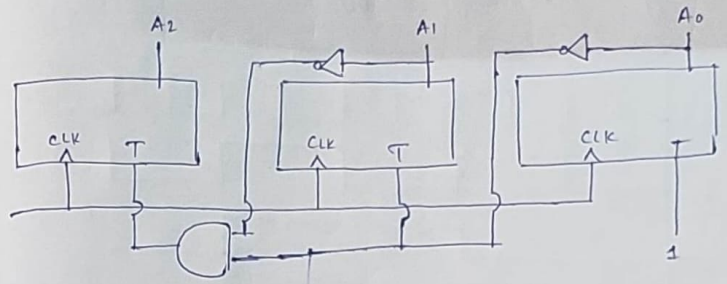
$T_{A_2} = A_1'A_0'$

$T_{A_1} = A_0'$



$T_{A_0} = 1$

State diagram



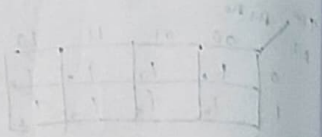
3-bit UP/DOWN Counter

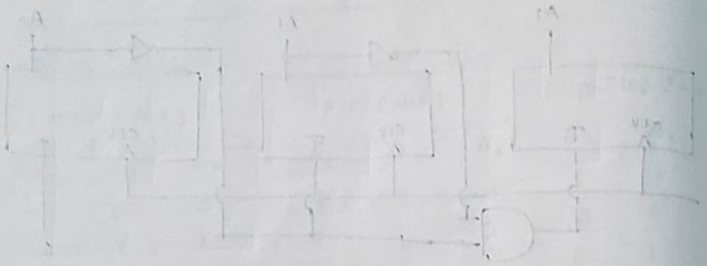
Control $\leftarrow \begin{cases} M = 0 & \text{up counter} \\ \cdot = 1 & \text{down counter} \end{cases}$

$T_C = \overline{M} Q_B Q_A + M \overline{Q_B} \overline{Q_A}$

$T_B = M \oplus Q_A$

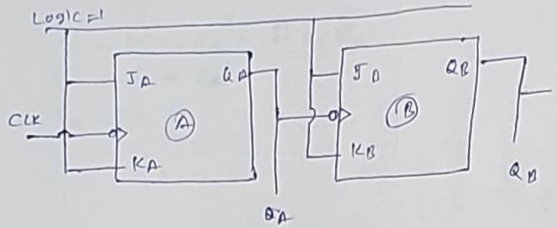
$T_A = 1$



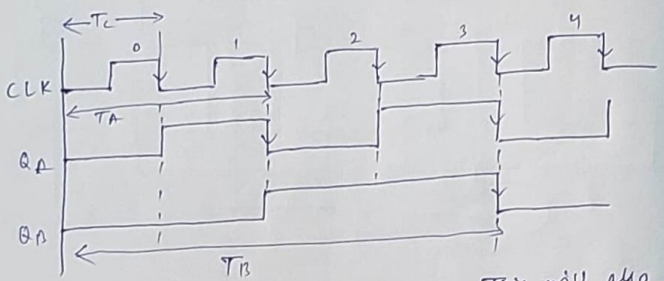


Asynchronous Counter as frequency divider

→ Asynchronous counter will work as a frequency divider



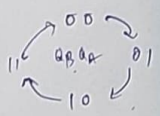
→ The asynchronous counter are normally designed using a -ve edge triggered clock pulse



$T_A = 2T_C$

$T_B = 2T_A = 4T_C$

This will also work as a 2-bit up counter



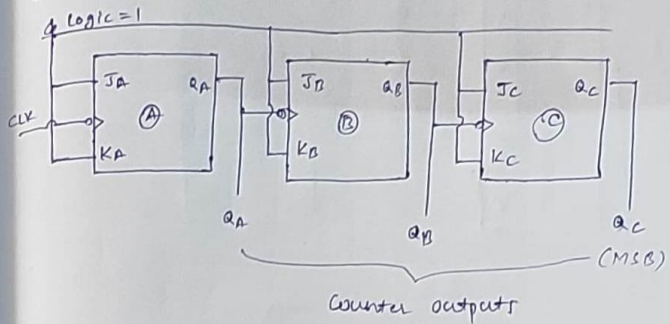
$f = \frac{1}{T}$

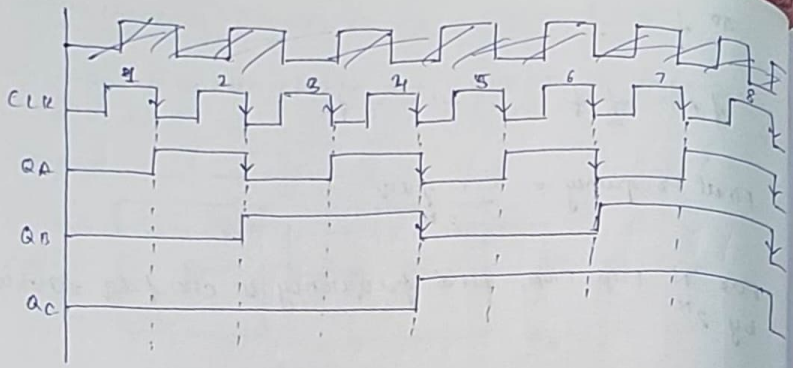
$f_A = \frac{1}{2} f_C$, $f_B = \frac{1}{4} f_C$

Final Frequency = $\frac{\text{clk freq}}{2^N}$

For 'N' Flip-Flops, final frequency is clk freq divided by 2^N

3-bit Asynchronous Up Counter




CLK
QA
QB
QC
1
2
3
4
5
6
7
8

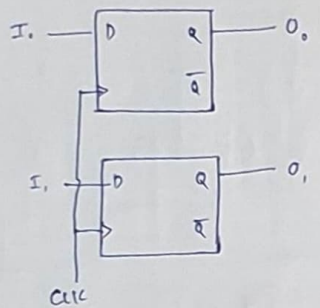
Shift Registers

→ Flip Flop is a 1 bit memory cell

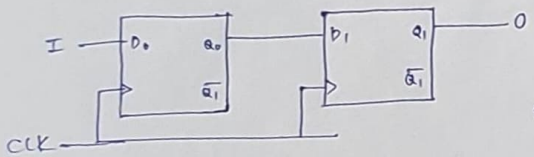
→ To increase the storage capacity, we have to use group of flip flop. This group of flip flop is known as Register.

→ The n bit register consists of 'n' number of flip flops and is capable of storing n bit word.

PIPO, SISO, SIPO and PISO



PIPO

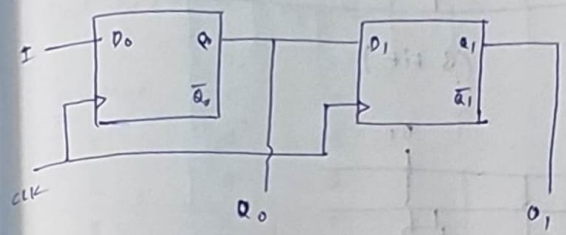


SISO

	Q_0	Q_1
Initial	x	x
1st clock	0	x
2nd clock	1	0
3rd clock	x	1
		0

2 pulse required to get the first output

3 pulse required to get the second output and so on



SIPO

2 clock pulse required to get 2 output

M = 0 load

M = 1 shift

shift / load

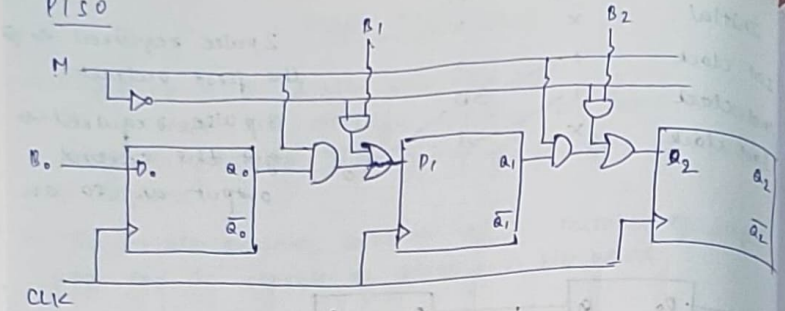
Data, B, Q

$Y = \bar{M}B + MQ$

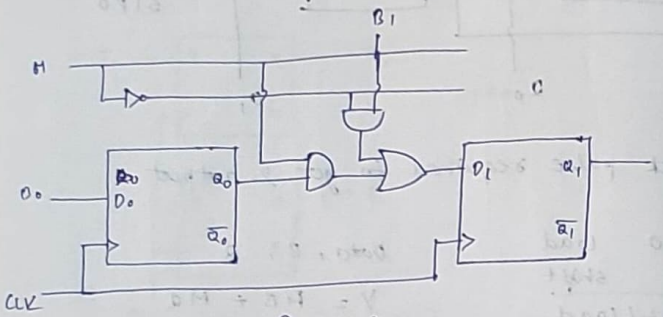
If $M = 0$, $Y = B$

If $M = 1$, $Y = Q$

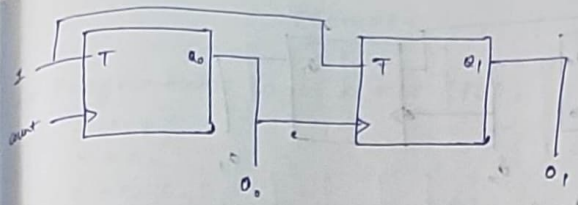
PISO



(3 bit)



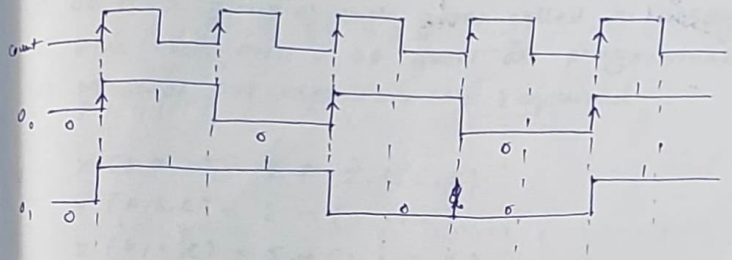
(2 bit)



Initially $Q_1 Q_0 = 00$

Find next 4 sequence

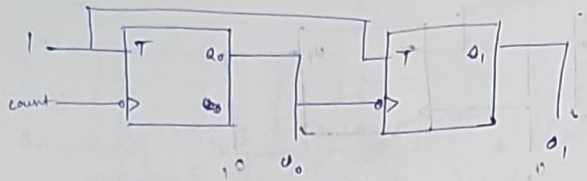
	Q_0	Q_1
Initial	0	0
1	1	1
2	0	1
3	1	0
4	0	0



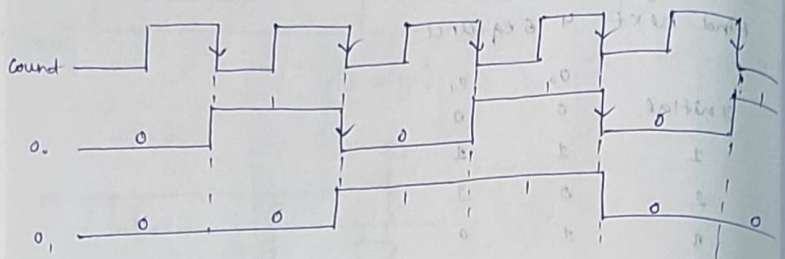
00, 11, 01, 10, 00, 11

↓

4th sequence

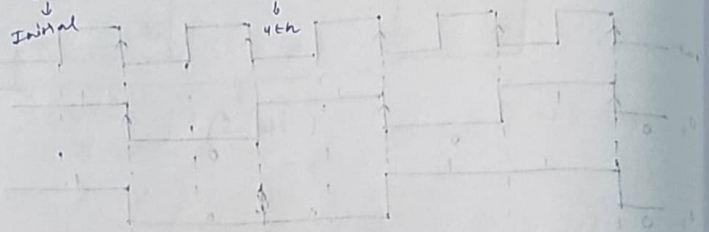


Initial $Q_1 Q_0 = 00$



00, 10, 01, 11, 00, 10

Initial 4th



Memory

→ Programmable Logic device (PLD)

- PLA
- PAL
- SPLD
- CPLD
- FPGA

1) PLA (Programmable logic Array)
2) PAL (Programmable Array logic)
3) SPLD (Simple programmable logic device)
4) CPLD (Complex programmable logic device)
5) FPGA (Field Programmable gate array)

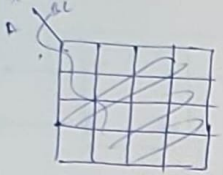
→ PLA :-

1) It is a group of logic gates called as array.
2) Here both AND & OR gates are programmable
3) Minimal SOP expressions are required

eg - $X(A,B,C) = \Sigma m(2,3,6,7)$

$Y(A,B,C) = \Sigma m(0,1,2)$

$Z(A,B,C) = \Sigma m(1,2,5,7)$



X

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

$X = B$

Y

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

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

Z

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

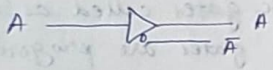
$Z = AC + \cancel{\bar{A}\bar{B}\bar{C}} + \cancel{\bar{B}\bar{C}}$

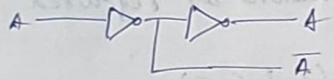
$= AC + \bar{A}B\bar{C} + \bar{B}C$

STEPS:

1) Find the minimal SOP expression
2) Identify the no. of input buffers (Input buffers must be equal to input variables)

No. of Buffers = 3

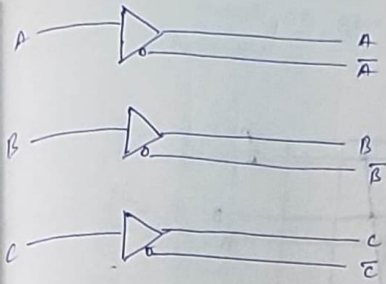




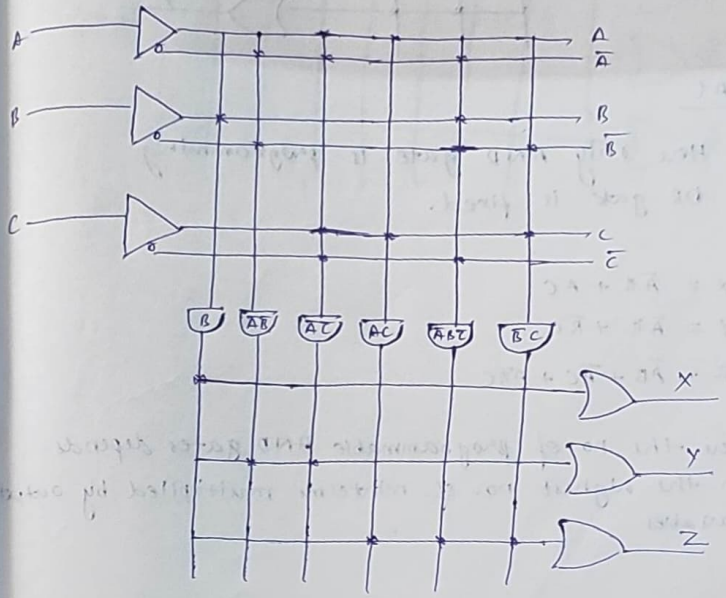
3) Implementing boolean function using PLA

~~4) T~~

5) The no. of programmable OR gates are equal to the no. of output expression

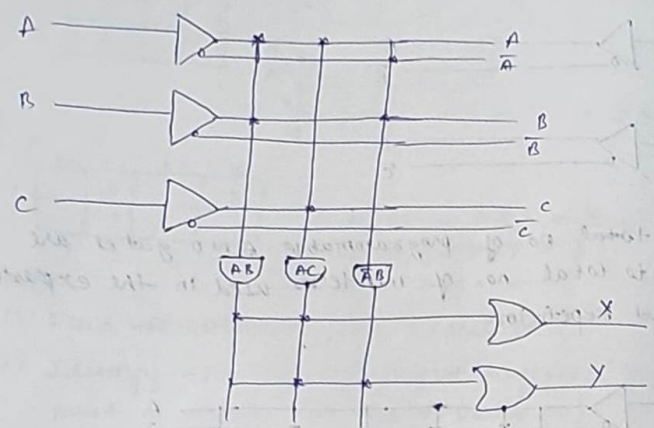


4) The ~~total~~ no. of programmable AND gates are equal to total no. of min-terms used in the expression without repetition



$X = AB + AC$

$Y = AC + A'B$



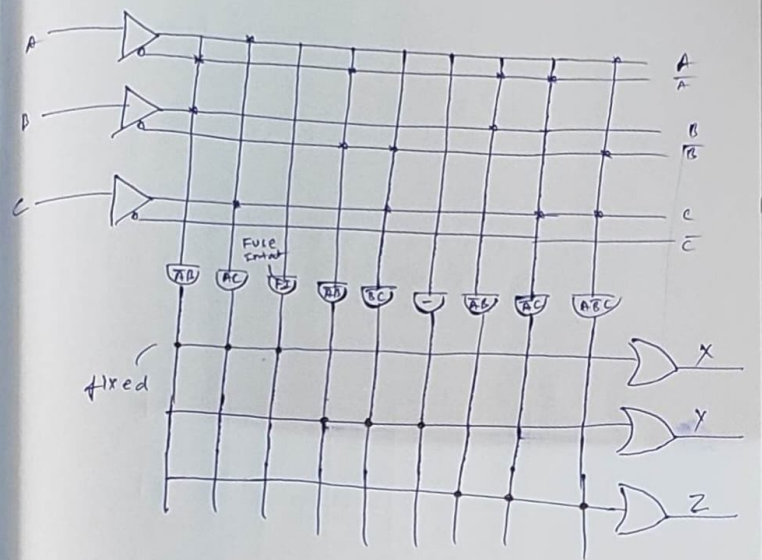
PAL

1) Here only AND gate is programming
2) OR gate is fixed.

$X = \bar{A}B + AC$

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

$Z = \bar{A}B + \bar{A}C + A\bar{B}C$

* Here the no. of programmable AND gates depends on the highest no. of minterms multiplied by output variables

* The no. of fixed OR gates are equal to the no. of outputs expressions