# Digital Short Notes

Follow me @uncademy/bvreddy
for the full length course
93980 21419

Use the Code: **BVREDDY**, to get maximum discount ,
complete notes ,DDPs and Short Notes

## Positive logic system

High voltage corresponds to logic " 1"
Maximum positive value is taken as logic '1

+5V ----> logic "1"

0V ----> logic "0 "


## Negative logic system

High voltage corresponds to logic " 0"
Maximum positive value is taken as logic '0 '

+5V ----> logic "0"

0V ----> logic "1 "


A positive logic system is converted into negative logic system by using the concept of duality

Finding the dual of a given Boolean expression

1. $* \leftrightarrow +$
2. $0 \leftrightarrow 1$
3. Keep the variables as it is

**BVREDDY**

**OR -Operation**

| |
|---|
| A+ 0 = A |
| 1+A= 1 |
| A+A=A |
| A+$\bar{A}$= 1 |

**AND-Operation**

| |
|---|
| A∗ $\mathbf{1} = A$ |
| A∗ $\mathbf{0} = 0$ |
| A∗ $A = A$ |
| A∗ $\bar{A} = 0$ |

**BVREDDY**

**Commutative Law**

A+B= B+A

$A * B = B * A$

**Associative Law**

A+B+C= (A+B)+C= (B+C)+A= (C+A)+B

$A * B * C = (A * B) * C = (B * C) * A = (C * A) * B$

**Trick to find Complement**

1. Find the dual
2. Complement the each literals

**Distribution Law (Mingle wala)**

A(B+C)=AB+AC

A+BC= (A+B)(A+C)

D- Morgan's Law

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

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

Transposition theorem ( T- 1)

**(A+B)(A+C)= A+BC**

**Transposition theorem ( T- 2)**

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

**Canonical form : Each minterm ( maxterms ) contains all the Boolean variables**

**F(A, B,C )= ABC+ $\bar{A}BC + AB\bar{C}$ ---->SOP**

**F(A, B, C) = (A+B+C) (A+$\bar{B} + C$)($\bar{A} + B + \bar{C}$) --> POS**

**Minimal Form** : The minimized form of Boolean expression

**F(A, B,C )= $BC + AB$**

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

Literal : A Boolean variable either in normal form (or ) complimented form is known as literal

Minterm :Each term in canonical SOP representation is known as minterm

Maxterm: Each term in canonical POS representation is known as maxterm

**BVREDDY**

Consensus theorem ( Rajinikanth wala)

$$AB + \bar{A}C + BC = AB + \bar{A}C$$

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

1. Maximum possible minterms = $2^n$
2. Maximum possible maxterms = $2^n$

BVREDDY

3. Number of minterm's + number of maxterm's = $2^n$
4. The sum of all the minterms = **ONE**
5. The product of all maxterms = **ZERO**
6. Minterm's and maxterm's of same index are **compliment** to each other
7. By using 2- Boolean variables total number of possible Boolean functions = 16
8. By using n- Boolean variables total number of possible Boolean functions = $2^{2^n}$
9. By using 2- Boolean variables total number of possible Boolean functions having at most 3- minterms = $4_{C_0} + 4_{C_1} + 4_{C_2} + 4_{C_3} = 15$
10. By using 2- Boolean variables total number of possible Boolean functions having at most 3- maxterms = 15
11. By using 2- Boolean variables total number of possible Boolean functions having 3- minterms = $4_{C_3} = 4$
12. By using n- Boolean variables total number of possible Boolean functions having 2- minterms = $2^n{}_{C_2}$
13. By using 5- Boolean variables total number of possible Boolean functions having at most 3- minterms = $32_{C_0} + 32_{C_1} + 32_{C_2} + 32_{C_3}$

BVREDDY

Neutral Function :

The number of minterms = number of maxterms

Mutually exclusive terms

The mutually exclusive term of $m_i$ is $m_{2^n-i-1}$

BVREDDY

Self Dual Expression

If one time dual of the Boolean expression result the same expression , then it is called as self dual expression

Eg : f = AB+BC+AC

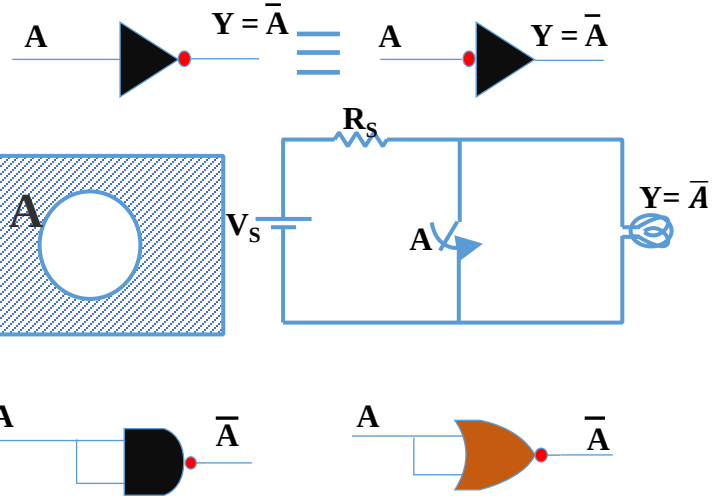
**Conditions for the given expression is self dual**

1. The number of minterms = number of maxterms (Neutral Function)
   number of minterms+ number of maxterms = $2^n$
   number of minterms = number of maxterms = $2^{n-1}$
2. If $m_i$ belongs to f , then $m_{2^n-i-1}$ should belongs to $\bar{f}$
3. The number of self dual functions = $2^{2^{n-1}}$

# NOT GATE

$$Y = \bar{A}$$

The output is the complement of the input



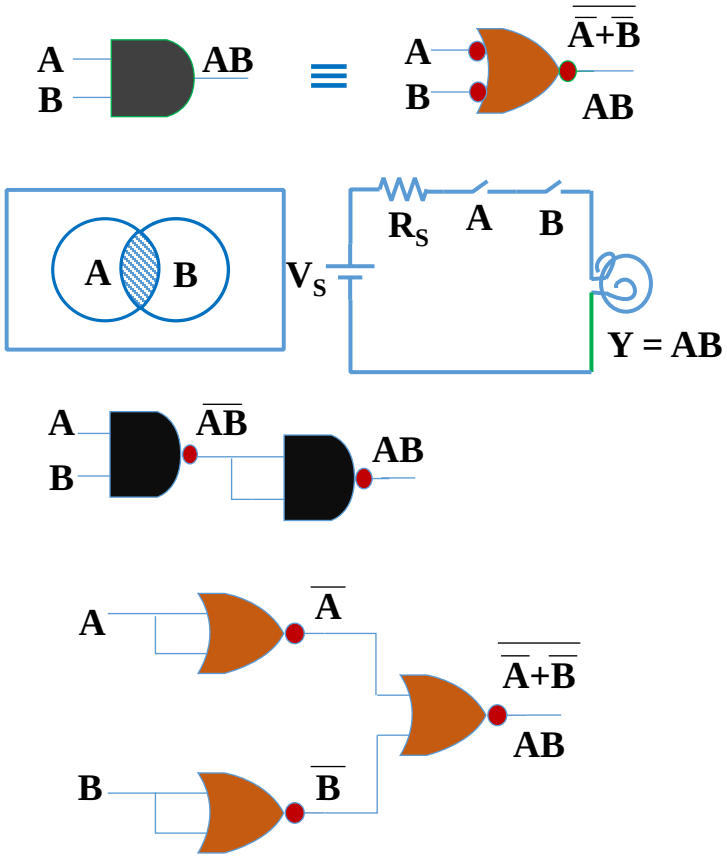



# AND GATE

$$Y = AB$$



- Output is '0' if any one input '0'
- Y = AB = $\Sigma(3) = \Pi(0,1,2)$
- Enable input ⟹ 1
- Disable input ⟹ 0
- Commutative law ⟹ Obeys
- Associative law ⟹ Obeys

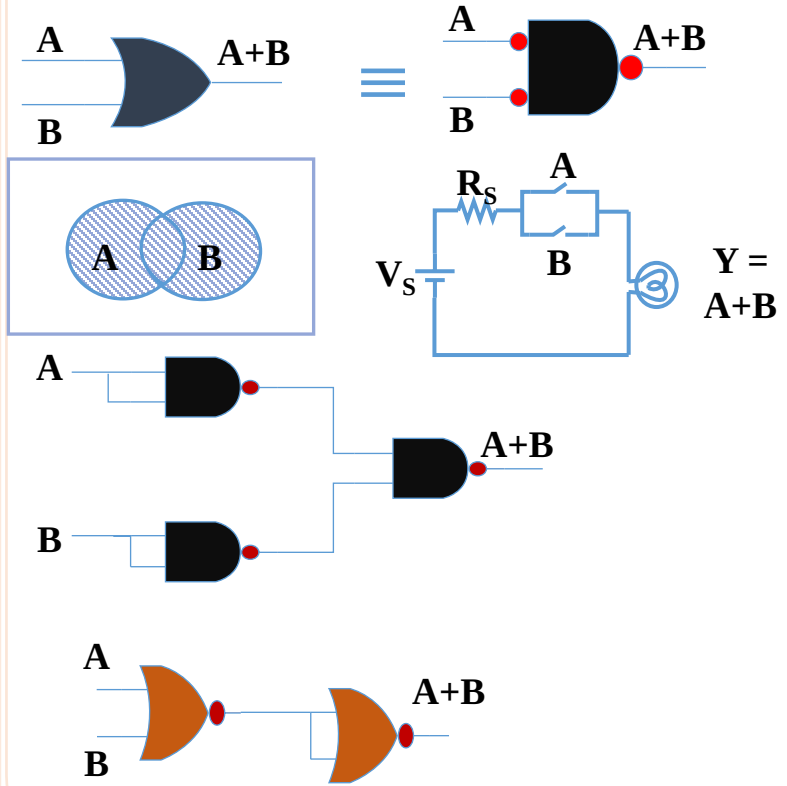



# OR GATE

$$Y = A+B$$



- Output is '1' if anyone of the inputs are '1'
- Y = A+B = $\Sigma(1,2,3) = \Pi(0)$
- Enable input ⟹ 0
- Disable input ⟹ 1
- Commutative law ⟹ Obeys
- Associative law ⟹ Obeys

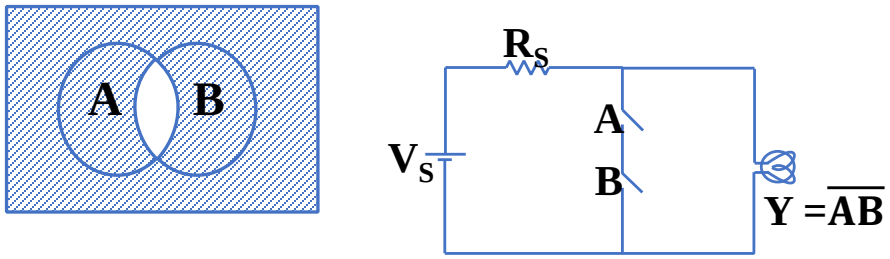


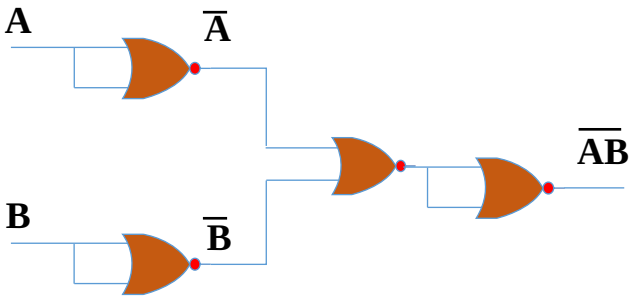
## NAND GATE

$$Y = \overline{AB}$$

- Output is '1' if any one input is '0'
- $Y = \overline{AB} = \sum(0,1,2) = \prod(3)$
- Enable input --1
- Disable input– 0
- Commutative law ---> Obeys
- Associative law ----> not Obeys

BVREDDY

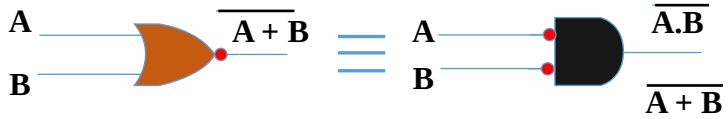



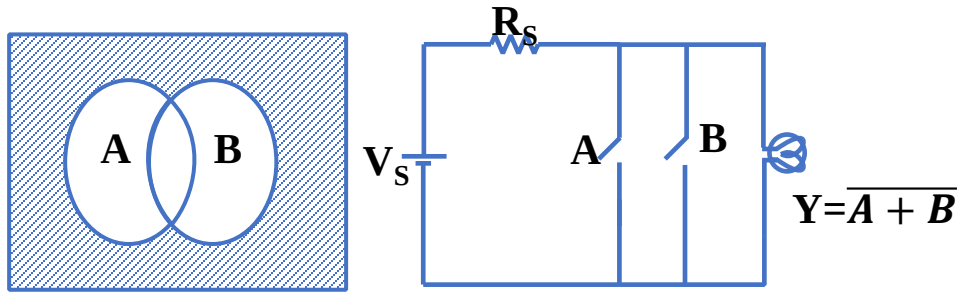

BVREDDY

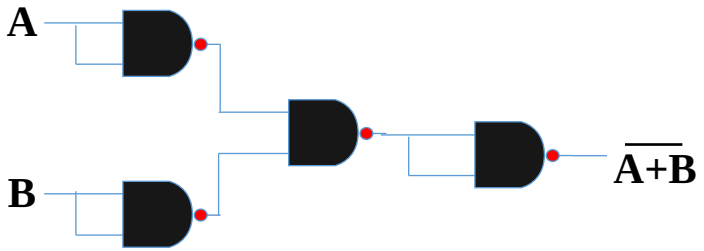
## NOR- GATE

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

- Output is '0' if any one of the input is '1'
- $Y = \overline{A + B} = \sum(0) = \prod(1,2,3)$
- Enable input --0
- Disable input– 1
- Commutative law ---> Obeys
- Associative law ----> not Obeys



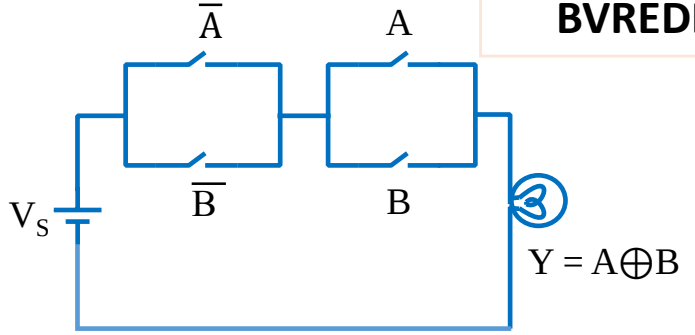

BVREDDY

BVREDDY

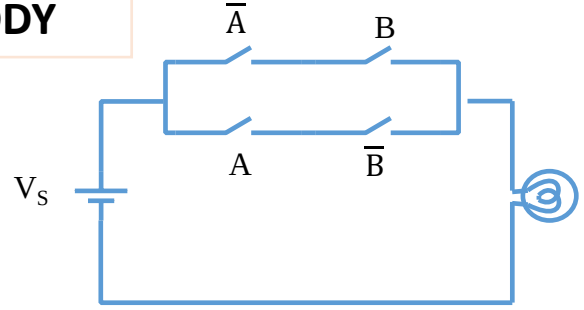
# EX-OR GATE

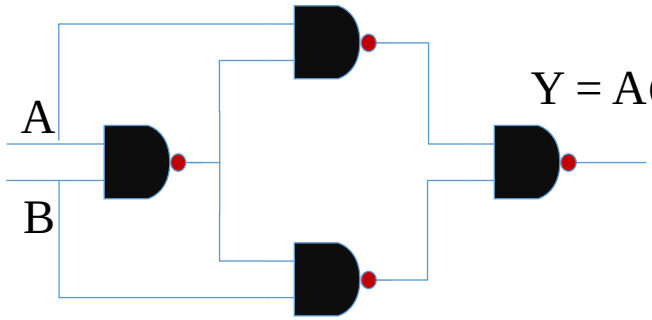
- Output is '1' for odd number of '1's in the input
- $\mathbf{Y = A \oplus B = \sum(1,2) = \Pi(0,3)}$
- $\mathbf{Y = A \oplus B \oplus C = \sum(1,2,4,7)}$
- $\mathbf{Y = A \oplus B \oplus C \oplus D = \sum(1,2,4,7,8,11,13,14)}$
- Commutative law $\Longrightarrow$ Obeys
- Associative law $\Longrightarrow$ Obeys
- $A \oplus 0 = A$
- $A \oplus 1 = \overline{A}$
- $A \oplus A = 0$
- $A \oplus \overline{A} = 1$
- $A \oplus A \oplus A \oplus \ldots\ldots\ldots\ldots n \text{ times} = \begin{cases} A, & n \text{ is odd} \\ 0, & n \text{ is even} \end{cases}$
- $A \oplus \bar{A}B = A + B$
- $AB \oplus BC = B(A \oplus C)$

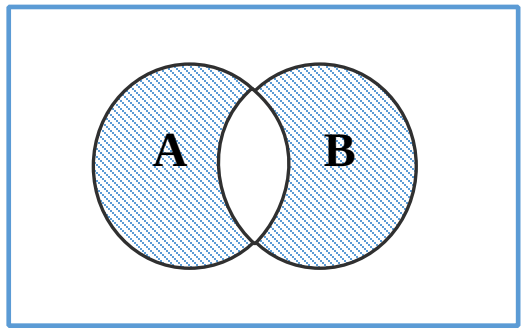
BVREDDY

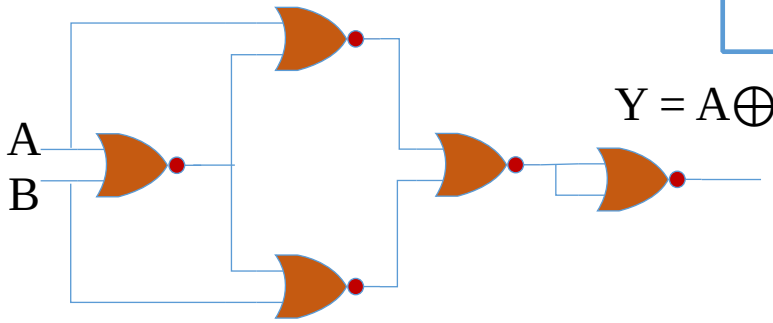



BVREDDY



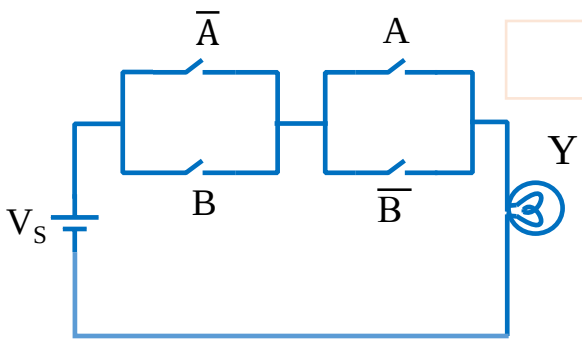

BVREDDY

**Use the Code: BVREDDY, to get maximum discount , complete notes ,DDPs and Short Notes**

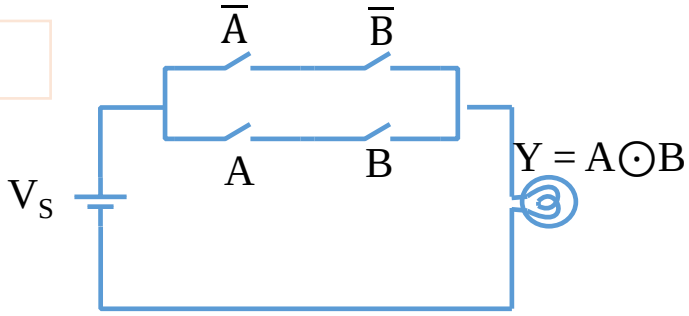
# EX-NOR GATE

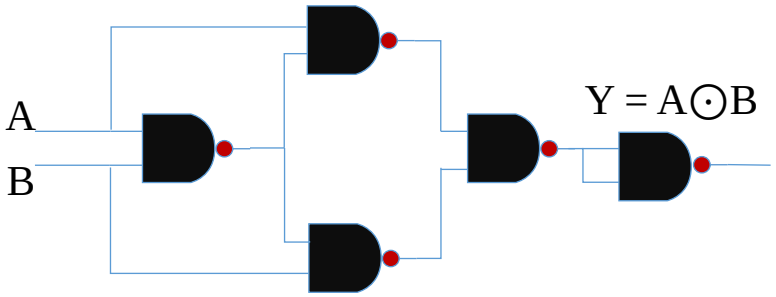
- Output is '1' for even number of '1's in the input
- $Y = A \odot B = \sum(0,3) = \Pi(1,2)$
- Commutative law $\Longrightarrow$ Obeys
- Associative law $\Longrightarrow$ not Obeys
- $A \odot 0 = \overline{A}$
- $A \odot 1 = A$
- $A \odot A = 1$
- $A \odot \overline{A} = 0$
- $A \odot A \odot A \odot \ldots\ldots\ldots\ldots n \text{ times} = \begin{cases} \overline{A}, & n \text{ is odd} \\ 1, & n \text{ is even} \end{cases}$
- $\overline{A \odot B} = A \oplus B$
- $A \oplus \overline{B} = A \odot B$
- $\overline{A} \oplus B = A \odot B$
- $\overline{A} \oplus \overline{B} = A \oplus B$
- $A \odot B \odot C = \sum(0,3,5,6)$
- $A \oplus B \oplus C = \sum(1,2,4,7)$
- $(A \odot B) \odot C = \sum(1,2,4,7)$
- $(A \odot C) \odot B = \sum(1,2,4,7)$
- $A \oplus B \oplus C = (A \odot B) \odot C = (A \odot C) \odot B$
- $A \odot B = \overline{A} \oplus B = A \oplus \overline{B} = \overline{A} \odot \overline{B}$
- $A \oplus B = A \odot \overline{B} = \overline{A} \odot B = \overline{A} \oplus \overline{B}$
- $\overline{A \oplus B \oplus C} = A \odot B \odot C = [A \oplus B] \odot C = A \odot [B \oplus C]$

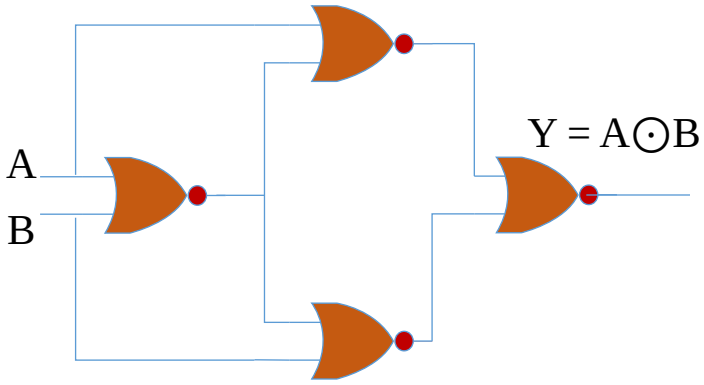




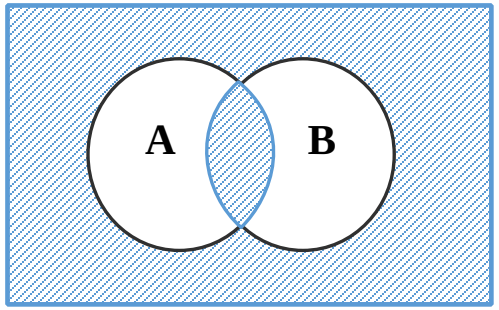







| EX-OR GATE | EX-NOR GATE |
|---|---|
| Output is '1' for odd number of '1's in the input | Output is '1' for even number of '1's in the input |
| Odd number of 1's detector | Even number of 1's detector |
| Inequality detector | Equality detector |
| Anti-coincident gate | Coincident gate |

| | No. of NAND GATES | No . of NOR GATES |
|---|---|---|
| NOT | 1 | 1 |
| AND | 2 | 3 |
| OR | 3 | 2 |
| EX-OR | 4 | 5 |
| EX-NOR | 5 | 4 |
| NAND | 1 | 4 |
| NOR | 4 | 1 |

- For a n- variable Boolean expression , the maximum number of literals = n
- For a n- variable K- Map if group is done by considering $2^m$ number of cells , then the resulting term from that group contains ( n- m ) number of literals .
- 8 *cells* – $2^3$ cells → Octet --> 3 variables eliminated
- 4 *cells* – $2^2$ cells → Quad ---> 2 variables eliminated
- 2 *cells* – $2^1$ cells → Pair ---> 1 variables eliminated

- Minimal expression may not be unique .
- The minimal expression = ( All EPI's ) + ( Optional PI's )
- If all PI's are EPI's , then the minimal expression is unique
- The sufficient condition for a K-map to have unique solution is number of PI's = number of EPI's

**Use the Code : BVREDDY**

# K- Map

Implicant : Each minterm in canonical SOP expression is known as Implicant .

Prime Implicant is a product term , obtained by combining maximum possible cells in the K- Map. While doing so make sure that a smaller group is not completely inside a bigger group .

**Essential Prime Implicant :** A prime Implicant is an EPI , if and only if it contains at least one minterm which is not covered by multiple groups

**All EPI's are PI's , but vice versa not true**

**EPI ≤ PI**

**Minterm mode**

| AB \ CD | $\bar{C}\bar{D}$ | $\bar{C}D$ | $CD$ | $C\bar{D}$ |
|---|---|---|---|---|
| $\bar{A}\bar{B}$ | $\bar{A}\bar{B}\bar{C}\bar{D}$ 0 | $\bar{A}\bar{B}\bar{C}D$ 1 | $\bar{A}\bar{B}CD$ 3 | $\bar{A}\bar{B}C\bar{D}$ 2 |
| $\bar{A}B$ | $\bar{A}B\bar{C}\bar{D}$ 4 | $\bar{A}B\bar{C}D$ 5 | $\bar{A}BCD$ 7 | $\bar{A}BC\bar{D}$ 6 |
| $AB$ | $AB\bar{C}\bar{D}$ 12 | $AB\bar{C}D$ 13 | $ABCD$ 15 | $ABC\bar{D}$ 14 |
| $A\bar{B}$ | $A\bar{B}\bar{C}\bar{D}$ 8 | $A\bar{B}\bar{C}D$ 9 | $A\bar{B}CD$ 11 | $A\bar{B}C\bar{D}$ 10 |

**Maxterm mode**

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

**Use the Code : BVREDDY ,to get the maximum discount**

## Number systems

- Base (b) is always a positive integer .
- In general b ≥ 0

| Base | Different digits |
|---|---|
| 2 ( Binary ) | 0 , 1 |
| 8( Octal ) | 0,1,2,3,4,5,6,7 |
| 10 ( Decimal ) | 0,1,2,3,4,5,6,7 ,8,9 |
| 16 (Hexadecimal) | 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F |

## r' s Complement

BVREDDY

r' s Complement of the number (N) $= r^n - N$

r --------> Radix

n -------> number of integer digits

N --------> given number

Use the Code : BVREDDY

## (r-1) ' s Complement

(r-1) ' s Complement of the number (N) $= r^n - r^{-m} - N$

r --------> Radix

n -------> number of integer digits

m -------> number of decimal digits

N --------> given number

BVREDDY

(r-1) ' s Complement of the number (N) $= r^n - r^{-m} - N$

r' s Complement of the number (N) = (r-1)'s complement + $r^{-m}$

if m= 0

r' s Complement of the number (N) = (r-1)'s complement + 1

## Unsigned Number Representation

BVREDDY

- Strictly applicable for positive numbers
- There is no sign bit concept

  + 5 ------> 101

  - 5 -----> not allowed
- Range = 0 to $2^n - 1$

## Signed Magnitude representation

- Valid for both positive and negative numbers .
- Sign bit concept is used .


Sign bit = 0 , for ⊕Ve number

= 1, for ⊖ *ve number*

Range = - $(2^{n-1} - 1)$ to + $(2^{n-1} - 1)$

Use the Code : BVREDDY

## 1's Compliment representation

In this ⊕Ve numbers are represented as normal binary number with MSB '0'

**BVREDDY**

**Representation of ⊖ *ve number***

1. Write the binary equivalent of magnitude
2. Take its 1's compliment

➢ Range = - $(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

**Overflow**

Over flow occurs in signed arithmetic operations if two same sign numbers are added and result exceeds with given number of bits . Overflow can be avoided by taking extra bits

**1.By using carry bits**

$C_{in}$ ⟶ *carry into MSB*

$C_{out}$ ⟶ *carry out from MSB*

if $C_{in} \oplus C_{out} = 0$ , no overflow occurs

$C_{in} \oplus C_{out} = 1$ , over flow occurs

2. By using Sign Bits

X ----------> Sign bit of 1st number

Y ----------> Sign bit of 2nd number

Z----------> Sign bit of Resultant

**BVREDDY**

**Over flow = $XY\bar{Z} + \bar{X}\bar{Y}Z$**

## 2's Compliment representation

In this ⊕Ve numbers are represented as normal binary number with MSB '0'

**Representation of ⊖ *ve number***

1. Write the binary equivalent of magnitude
2. Take its 2's compliment

➢ Range = - $(2^{n-1})$ to $+(2^{n-1} - 1)$

## BCD (Binary Coded Decimal)Code

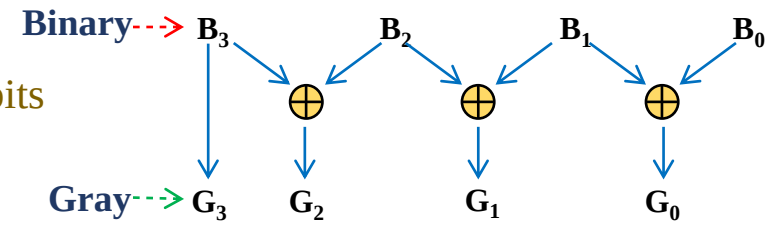
In this code each decimal number is represented by a separate group of 4- bits

➢ It uses only 0 to 9

➢ 0 to 9 are valid BCD Code

➢ 10, 11, 12 , 13 , 14 ,15 are invalid BCD Code

➢ Coding method is very simple but it requires more number of bits .

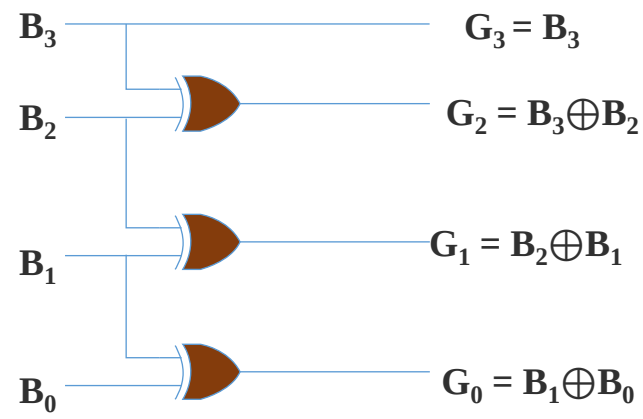



## EX-3 Code

The EX-3 code can be derived from the natural BCD code by adding 3 to each coded number

Valid EX -3 : 3 ,4,5,6,7,8,9,10,11,12

Invalid EX-3 : 0,1,2,13,14,15




## Gray Code

➢ Non weighted code

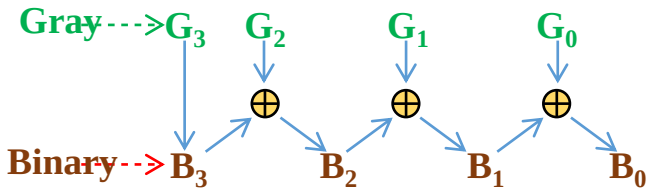
➢ Unit distance code

➢ Cyclic code

➢ Reflective code

➢ Minimum error code

**BVREDDY**

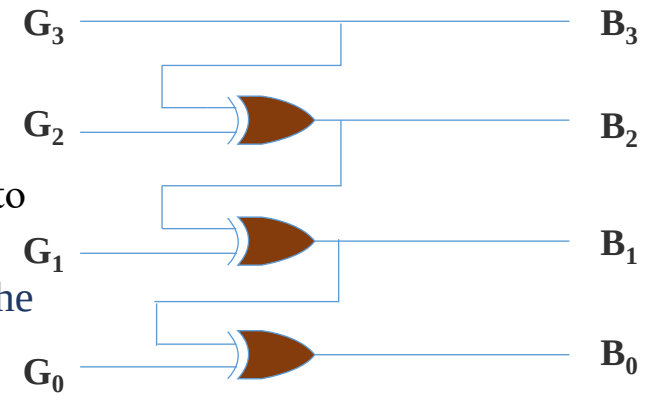



BVREDDY

## SELF COMPLEMENTING CODE

A code is said to be self complementing, if the 1' complement of a number N is equal to the 9's complement of the number.

➢ For a code to be self complementing, the sum of all its weights must be 9 .




| 2421 | 5211 | 4311 | 3321 | EX-3 |
|---|---|---|---|---|

BVREDDY

## HA

1. Logical expression for Sum = $A \oplus B$
2. Logical expression for Carry = $AB$
3. Minimum number of NAND Gates = 5
4. Minimum number of NOR Gates = 5

## FA

1. Logical expression for Sum = $A \oplus B \oplus C$
2. Logical expression for Carry = $AB+(A \oplus B)C$
3. Minimum number of NAND Gates = 9
4. Minimum number of NOR Gates = 9

## HS

1. Logical expression for Difference = $A \oplus B$
2. Logical expression for Barrow= $\bar{A}B$
3. Minimum number of NAND Gates = 5
4. Minimum number of NOR Gates =5

## FS

1. Logical expression for Difference= $A \oplus B \oplus C$
2. Logical expression for Barrow = $\bar{A}B + (\overline{A \oplus B})C$
3. Minimum number of NAND Gates = 9
4. Minimum number of NOR Gates = 9

## Half Adder

| A | B | Sum | Carry |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

## Half Subtractor

| A | B | Difference | Barrow |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

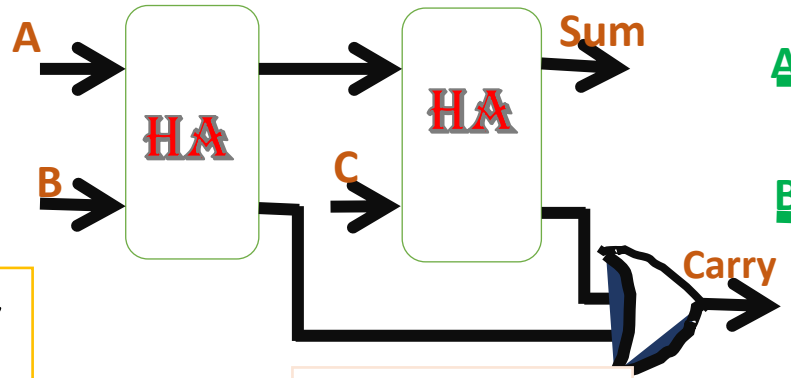
## Full Adder

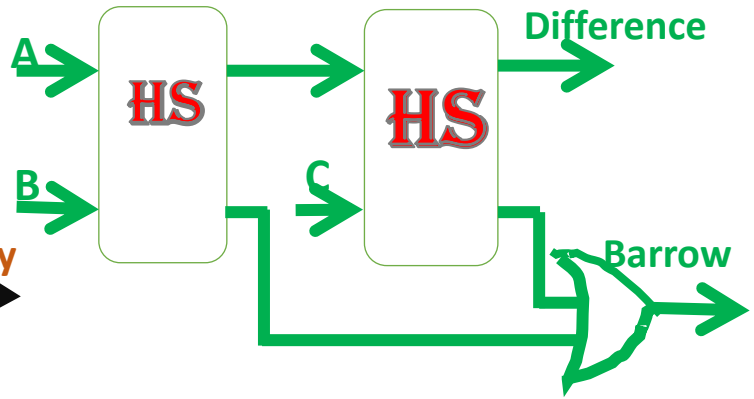
| A | B | C | Sum | Carry |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

## Full Subtractor

| A | B | C | Difference | Barrow |
|---|---|---|---|---|
| 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 |

BVREDDY




BVREDDY

**Use the Code : BVREDDY**

**FS : A- B- C**

Difference = $A \oplus B \oplus C$

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

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

**FS : B- C- A**

Difference = $A \oplus B \oplus C$

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

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

**FS : C- A- B**

Difference = $A \oplus B \oplus C$

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

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

## Binary Multiplier

Number of AND gates required = $m \times n$

Number of Adders required = m+n-2

m----------> number of bits in A

n ---------->number of bits in B

In general for n- bit Parallel Adder

**Worst case Delay = (n-1)$\left((t_{pd})_{carry} + Max(sum, carry)\right)$**

## Look Ahead Carry Adder

- In this adder ,the carry dependency of Ripple Carry Adder (RCA) is eliminated
- This is the fastest adder among all
- This adder have the maximum complexity

**Hardware Requirements**

L1 : n- XOR + n- AND

L2 : $\frac{n(n+1)}{2}$ – AND

L3 : n- OR

carry

L4 : n- XOR

sum

Total number of gates for carry = $3n + \frac{n(n+1)}{2}$

Total number of gates for sum = $4n + \frac{n(n+1)}{2}$

Worst delay for Carry = Max( xor ,and)+ $(t_{pd})_{and}$ + $(t_{pd})_{or}$

Worst case delay for Sum = Max( xor ,and)+ $(t_{pd})_{and}$ + $(t_{pd})_{or}$ + $(t_{pd})_{xor}$

BVREDDY

## For n- bit Magnitude Comparator

Total number of input combinations = $2^{2n}$

Lesser than combinations = $\frac{2^{2n} - 2^n}{2}$

Greater than combinations = $\frac{2^{2n} - 2^n}{2}$

Equal combinations = $2^n$

**For 3- bit magnitude comparator**

$Y_1(A<B) = \bar{a}_2 b_2 + (a_2 \odot b_2)\bar{a}_1 b_1 + (a_2 \odot b_2)(a_1 \odot b_1)\bar{a}_0 b_0$

$Y_2(A=B) = (a_2 \odot b_2)(a_1 \odot b_1)(a_0 \odot b_0)$

$Y_3(A>B) = a_2 \bar{b}_2 + (a_2 \odot b_2)a_1 \bar{b}_1 + (a_2 \odot b_2)(a_1 \odot b_1)a_0 \bar{b}_0$

BVREDDY

For 4-bit Magnitude Comparator

$Y_1(A<B) = \overline{a_3}\, b_3 + (a_3 \odot b_3)(\overline{a_2}\, b_2) + (a_3 \odot b_3)(a_2 \odot b_2)(\overline{a_1}\, b_1) + (a_3 \odot b_3)(a_2 \odot b_2)(a_1 \odot b_1)(\overline{a_0}\, b_0)$

$Y_2(A=B) = (a_3 \odot b_3)\,(a_2 \odot b_2)\,(a_1 \odot b_1)\,(a_0 \odot b_0)$

$Y_3(A>B) = a_3 \overline{b_3} + (a_3 \odot b_3)(a_2\, \overline{b_2}) + (a_3 \odot b_3)(a_2 \odot b_2)(a_1 \overline{b_1}) + (a_3 \odot b_3)(a_2 \odot b_2)\,(a_1 \odot b_1)(a_0\, \overline{b_0})$

BVREDDY

## Multiplexer (MUX)

- Data selector
- Many to one
- Universal logic gate
- Parallel to serial converter

$2^n \times 1$

$2^n$ ----------> number of data inputs

n ----------> number of select inputs

1 ----------> number of outputs

BVREDDY

## Demultiplexer

- One input to many output
- Data distributor
- One to many circuit

$1 \times 2^n$

n---------> number of select lines

$2^n$ --------> number of output lines

1 ----------> number of inputs

## Decoder

Decoder is a multi input ,multi output logic circuit which coverts coded input into coded output , where the input and output codes are different

$n \times 2^n$

n ----------> number of inputs

$2^n$ ----------> number of outputs

Decoder is a special case of Demultiplexer , in which the select lines or Demultiplexer are treated as input's to the decoder and input of Demultiplexer is treated as Enable input of the Decoder

**Inputs ⟺ Enable**

**Select lines ⟺ Inputs**

| Logic Gate | Number of MUX required |
|---|---|
| BUFFER | 1 |
| NOT | 1 |
| AND | 1 |
| OR | 1 |
| NAND | 2 |
| NOR | 2 |
| EX-OR | 2 |
| EX-NOR | 2 |
| HA | 3 |
| HS | 2 |

## Encoder

Encoder is a combinational circuit , which is used to convert

1. Octal to binary ( 8 × 3 encoder )
2. Decimal to Binary ( 10× 4 encoder )
3. Hexadecimal to Binary ( 16 × 4 encoder )

$2^n X\ n$

n ----------> number of outputs

$2^n$ ----------> number of inputs

- For an Encoder at a time only one among the all inputs is high , reaming all inputs should be zero
- If multiple inputs are simultaneously high, then the output is not valid, to avoid this restriction we will go for priority encoder.

1. By using one 4 × 1 Mux → All 2 variable functions; Some but not All 3 variable functions

BVREDDY

2. By using one 4 × 1 Mux + NOT Gate → All 2 variable functions; All 3 variable functions

3. By using one 8 × 1 Mux → All 3 variable functions; Some but not All 4 variable functions

4. By using one 8 × 1 Mux + NOT Gate → All 3 variable functions; All 4 variable functions

5. n- variable function → **One $2^n \times 1$ MUX**; **One $2^{n-1} \times 1$ MUX + one NOT Gate**

BVREDDY

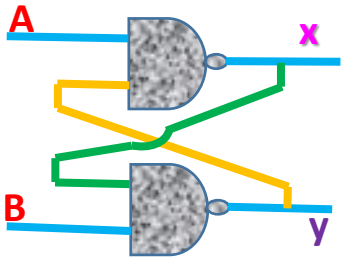
## Sequential Circuits

The logic circuit whose outputs at any instant of time depends on the present inputs as well as on the past outputs are called sequential circuits, in sequential circuits ,the output signals are fed back to the input side .

BVREDDY

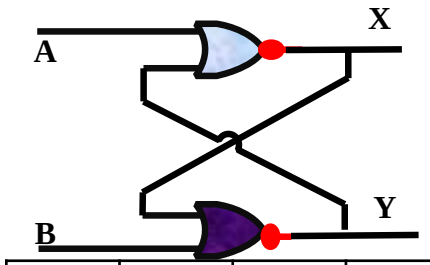
- Out put of combinational circuit depends on input combinations .
- Output of sequential circuits depends on input sequence.
- For unequal delay of gates also the operation is valid

**NAND LATCH**




| A | B | X | Y |
|---|---|---|---|
| 0 | 0 | **1** | **1** |
| 0 | 1 | **0** | **1** |
| 1 | 0 | **1** | **0** |
| 1 | 1 | **Memory** | |

**NOR LATCH**




| A | B | X | Y |
|---|---|---|---|
| 0 | 0 | **1** | **1** |
| 0 | 1 | **1** | **0** |
| 1 | 0 | **0** | **1** |
| 1 | 1 | **Memory** | |

BVREDDY

For **SR NAND** latch , if the input sequence is **00** ---------> **11** , then the following cases arises

- If the delay of both gates are same then we don't have any stable output , the output is oscillatory , this condition is known as critical race
- However if the delay of both gates are not equal then there exist a stable output , but it depends on the individual delay of the gates

For **SR NOR** latch , if the input sequence is **11** ---------> **00** , then the following cases arises

- If the delay of both gates are same then we don't have any stable output , the output is oscillatory , this condition is known as critical race .
- However if the delay of both gates are not equal then there exist a stable output , but it depends on the individual delay of the gates .

### FLIP FLOP

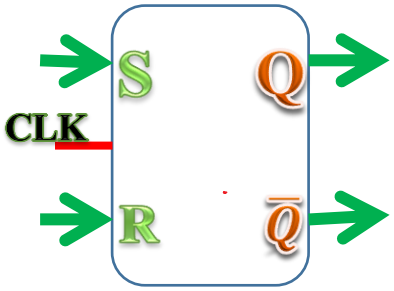
In a latch the output changes immediately in response to external input , so to have an additional control , we are introducing a signal called " ***CLOCK*** " , whose purpose is same as Enable pin of Decoder.

**Latch +Clock = Flip Flop**

Latches are universally not unique and hence their truth tables are not unique .

Flip Flops are universally unique , and their truth tables are unique .

BVREDDY

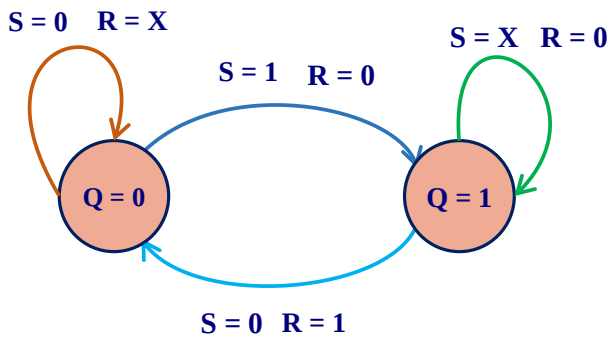


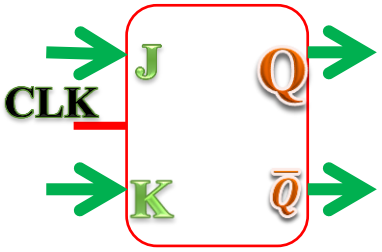

BVREDDY

| CLK | S | R | Q+ | State |
|---|---|---|---|---|
| 0 | × | × | Q | Memory |
| 1 | 0 | 0 | Q | Memory |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 1 | 0 | 1 | Set |
| 1 | 1 | 1 | × | Invalid |

| CLK | S | R | Q | Q+ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | × |
| 1 | 1 | 1 | 1 | × |

| Q | Q+ | S | R |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | X | 0 |

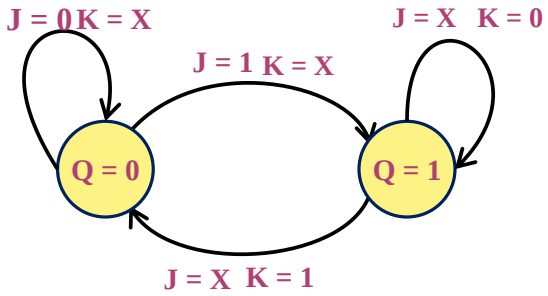



| CLK | J | K | Q+ | State |
|---|---|---|---|---|
| 0 | × | × | Q | Memory |
| 1 | 0 | 0 | Q | Memory |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 1 | 0 | 1 | Set |
| 1 | 1 | 1 | $\bar{Q}$ | Toggle |

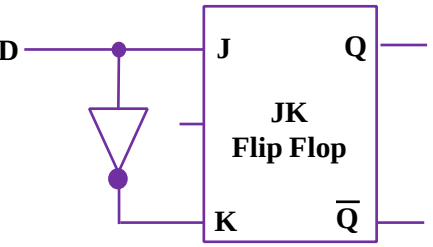
| CLK | J | K | Q | Q+ |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

| Q | Q+ | J | K |
|---|---|---|---|
| 0 | 0 | 0 | X |
| 0 | 1 | 1 | X |
| 1 | 0 | X | 1 |
| 1 | 1 | X | 0 |




BVREDDY

D Flip Flop



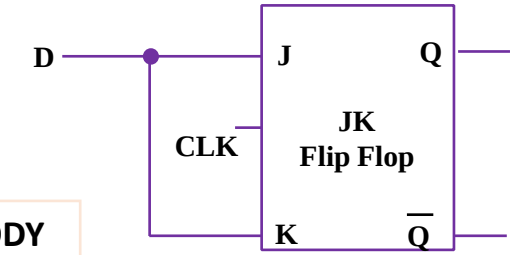

$Q^+ = D$

| CLK | D | Q+ |
|---|---|---|
| 0 | X | Hold |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| CLK | D | Q | Q+ |
|---|---|---|---|
| 0 | X | Q | Q |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

BVREDDY

T Flip Flop



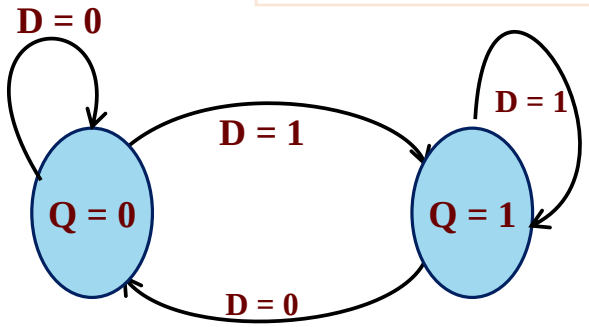

$Q^+ = T \oplus Q$

| CLK | T | Q+ |
|---|---|---|
| 0 | X | Hold |
| 1 | 0 | Q |
| 1 | 1 | Toggle |

| CLK | T | Q | Q+ |
|---|---|---|---|
| 0 | X | Q | Q |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

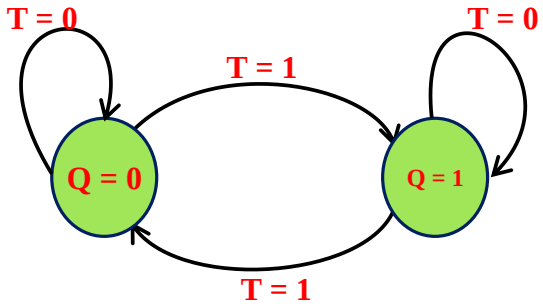
| Q | Q+ | D |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

BVREDDY




**Use the Code : BVREDDY**

| Q | Q+ | T |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |




BVREDDY

BVREDDY

## Race Around Condition

The output of the FF changes to 0 →1 →0 …. Continuously at the starting of the next clock the output is uncertain , which is called as Race Around Condition (RAC )

RAC occurs in any FF if the following conditions satisfies

1. If the FFs are operated in level triggering
2. if (tpd ) < (Tclk )on ,
3. If the FFs are operated in Toggle mode

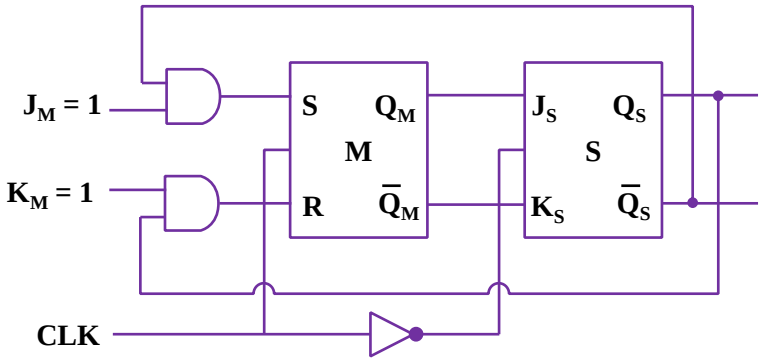
If the above 3 conditions satisfies simultaneously then there is a continuous race in the output of the FF between 0 and 1 to reach the next state , who will be the winner of the race in not certain , that depends on tpd and ( Tclk ) on .

### Remedy

1. ( Tclk )on < (tpd ) < T
2. By using Edge triggered FF
3. By using Master Slave FF

BVREDDY

## Master – Slave Flip Flop




1. In case of Master Slave configuration , Master is applied with input clock and Slave is applied with inverted clock , so out of two FFs at a time only one of the FF respond and other will not respond . As a result, Many times toggling in a single clock cycle has been converted to one time toggle , hence *RAC is avoided* .
2. In Master Slave configuration , command signal is generated by master FF and the response of the command signal is given by slave FF
3. Master slave FF can store 1 – bit of data

BVREDDY

**JK to SR**
J = S
K = R

**JK to D**
J = D
K = $\bar{D}$

**JK to T**
J = T
K = T

**SR to JK**
S = $J\bar{Q}$
R = $KQ$

**SR to D**
S = D
R = $\bar{D}$

**SR to T**
S = $T\bar{Q}$
R = $TQ$

**D to SR**
D = $S+\bar{R}Q$

**D to JK**
D = $J\bar{Q}+\bar{K}Q$

**D to T**
D = $T \oplus Q$

**T to SR**
T = $S\bar{Q}+RQ$

**T to JK**
T = $J\bar{Q}+KQ$

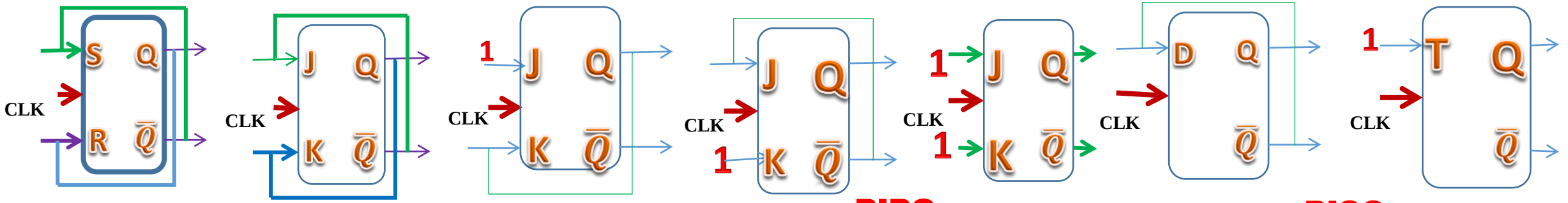
**T to D**
T = $D \oplus Q$

CONVERSATION of FLIP FLOP

BVREDDY

## Toggle Modes

BVREDDY



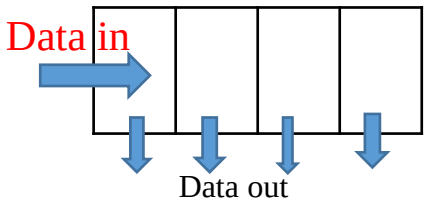

## SISO


- SISO Configuration has only
  - 1- input
  - 1- output
- For SISO configuration

for storing = (n) CP
for retrieving = (n-1) CP
Total number clock pulses = 2n-1

BVREDDY

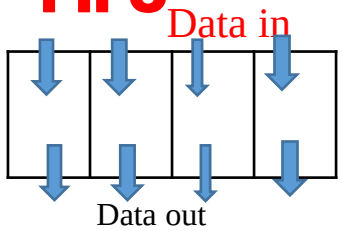
## SIPO




- SIPO Configuration has only
  - 1- input
  - 4- output
- For SIPO configuration

for storing = (n ) CP
for retrieving = 0 CP
Total number clock pulses = n

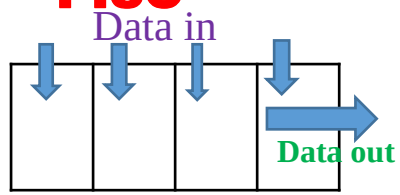
## PIPO




- PIPO Configuration has only
  - 4- input
  - 4- output
- For PIPO configuration

for storing = 1 CP
for retrieving = 0 CP
Total number clock pulses = 1

## PISO




- PISO Configuration has only
  - 4- input
  - 1- output
- For PISO configuration

for storing = 1 CP
for retrieving = (n-1)CP
Total number clock pulses = n

## Counters

**State of a Counter** : Any possible output of a counter is known as its state , for a n – bit counter the maximum possible states are $2^n$
The states which are counted by the counter are called as *valid states* , and the states which are not counted (skipped) by the counter are called as invalid states .

Modulus of a Counter : The minimum number of clocks needed to get the counting pattern repeats is called as Modulus of a counter

BVREDDY

Design equation of a counter

$$2^n \geq N$$

$$n \geq log_2 N$$

n----> number of Flip Flops
N-----> MOD no. of a counter

BVREDDY

## ASYNCHRONOUS COUNTER

BVREDDY

- Different FFs are applied with different clocks
- For only one FF external clock is applied ,which is LSB and output of one FF will acts as clock to next FFs
- FFs are operated in toggle mode
- Fixed counting sequence
  1. up counter
  2. down counter

**ये वक्त भी गुजर जाएगा**
**This time will also pass**

- ⊖ve Edge trigger and Q as a clock --------> Up counter
- ⊖ve Edge trigger and $\bar{Q}$ as a clock --------> Down counter
- ⊕ve Edge trigger and Q as a clock --------> Down counter
- ⊕ve Edge trigger and $\bar{Q}$ as a clock --------> Up counter
- The disadvantages of the ripple counter is that transition states are present due to delay of the FF ( Decoding errors) .
- If only one FF changes its state ,then no transition states will be present , if more than one FF changes its states than transition states present.

BVREDDY

- To avoid decoding errors strobe signal is used .
- Strobe signal is kept low for 3tpd , for 3- bit counter , so that transition states are not reflected, and after 3tpd strobe signal is made high .
- If delay each FF is $t_{pd}$ , *then*

$$T_{CLK} \geq n\, t_{pd}$$

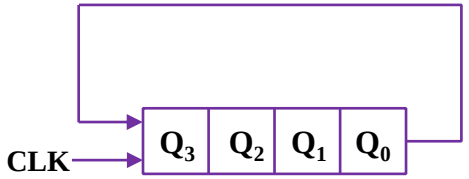
$$f_{CLK} \leq \frac{1}{t_{pd}}$$

**Use the Code : BVREDDY**

## RING COUNTER

- Ring counter is a synchronous counter , it is a shift register in which last FF output is connected to the first FF input .
- In ring counter only one FF output is logic '1 ' and it will rotate with clock .
- Ring counter performs right shift operation .

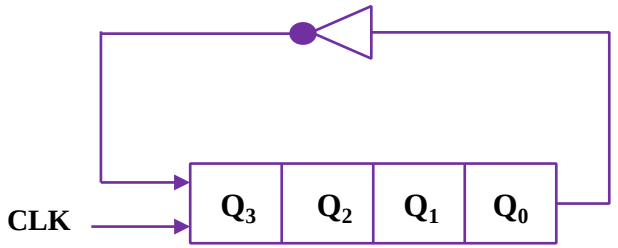
BVREDDY




- Decoding logic of ring counter is simple and does not require any external logic circuit
- If all the outputs of FFs initially zero , then the Ring counter does not start .
- If more than one FF outputs' are high initially, then the ring counter enters into unused state and never come out of unused state , this is called as **Lock out problem** .

BVREDDY

## JOHNSON RING COUNTER




Johnson Ring counter
Twisted Ring counter
Switch tail counter
Walking Counter
Creeping counter
Mobies counter

| Ring counter | Johnson ring counter |
|---|---|
| 1. Mod No = n BVREDDY | 1. Mod No = 2n |
| 2. Number of used states= n<br>Number of unused states = $2^n - n$ | 2. Number of used states= 2n<br>Number of unused states =$2^{2n} - n$ |
| 3.Time period of each FF = n($T_{CLK}$) | 3.Time period of each FF = 2n($T_{CLK}$) |
| 4. Frequency of each FF = $\frac{f_{clk}}{n}$ | 4. Frequency of each FF = $\frac{f_{clk}}{2n}$ |
| 5. Suffer from lock out problem | 5. Suffer from lock out problem |
| 6.Decoding logic is simple | 6.Decoding logic requires AND and NOR gates |

## FINITE STATE MACHINE

Synchronous Sequential circuits are also called as Finite State Machine ( FSM )

There are two types of FSMs

1. Mealy State Machine

- The output of Mealy State Machine is a function of present state as well as present input
- to detect n – bit sequence by using Mealy modal n number of states are required

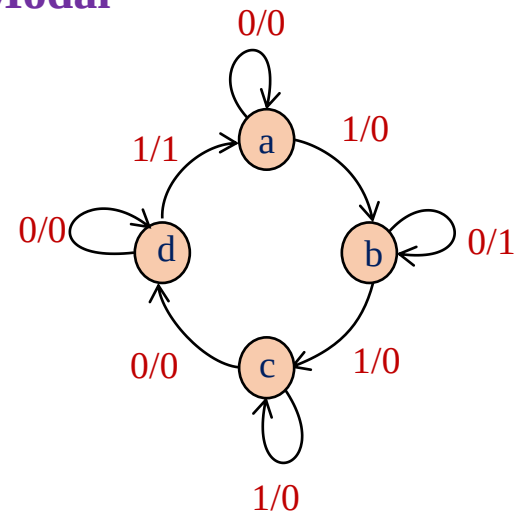
BVREDDY

2. Moore State Machine

- The output of Moore State Machine is a function of present state only
- To detect **n – bit sequence** by using Mealy modal **(n+1)** number of states are required

## Mealy Modal

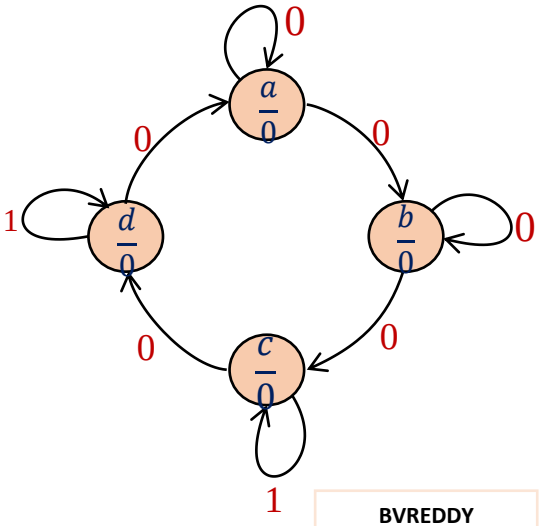
| Present state | NS , O/P | |
|---|---|---|
| | X =0 | X= 1 |
| a | a , 0 | b , 0 |
| b | b, 1 | c,0 |
| c | d, 0 | c , 0 |
| d | d, 0 | a , 1 |




BVREDDY

## Moore Modal

| Present state | Next State | | Output |
|---|---|---|---|
| | X =0 | X=1 | |
| a | a | b | 0 |
| b | b | c | 0 |
| c | d | c | 0 |
| d | a | d | 1 |




BVREDDY