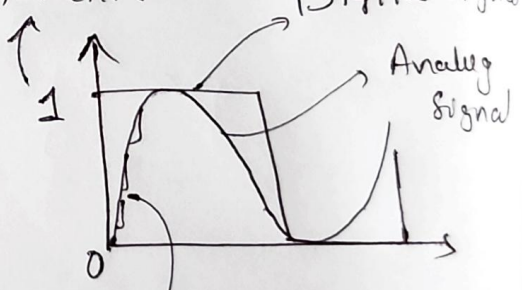
Digital Electronics :→

- Discrete Signals
- Noise → High frequency
- Number System
  - decimal → Set of numbers (0 - 9) → digits
    - ↳ 10 ~~stuff~~ Symbols. Symbols
  - → Octal Number System (0 - 7) → Octal digits
  - → Hexadecimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) → Hex
    - 16 Symbols
    - → Hexadecimal digits

Binary number system :-

two (2) → (0, 1)
↓
Bits (Binary digits)
Bits

(can be any thing 10V, 5V etc.)



- Base → Count of Symbols in a set (Number System) → Many steps

∴ Base of Decimal — 10
Octal — 8
Hexadecimal — 16
Binary — 2

b^1 b^0 ← weight

$\infty \ldots\ldots 1\ 0$ ← Position

99.

eg:- 99

$$\text{Value} = \sum_{-\infty}^{+\infty} \text{Digit} \times \text{weight}$$

ANY Number to decimal

$$\text{Value} = \sum \text{Digit} * (\text{Base})^{\text{Position}}$$

(Always Decimal) $= 9 \times 10^1 + 9 \times 10^0$

Octal number → to decimal

① $(537)_8$ ② 346

2 1 0 ← Position

Value of $(537)_8$

$$\text{Value} = \sum \text{Octal digit} * (\text{Base})^{\text{position}}.$$

$$= 5 \times 8^2 + 3 \times 8^1 + 7 \times 8^0$$

$$= 320 + 24 + 7$$

$$= (351)_{10}$$

← converted.

Hexadecimal → Decimal

2 1 0 ← Position

1 A 2

① $(1A2)_{16}$

$$\text{Value} = \sum \text{Hex digit} * (\text{Base})^{\text{Position}}$$

$$= 1 \times 16^2 + A \times 16^1 + 2 \times 16^0$$

$$= 256 + 160 + 2$$

$$= (418)_{10}$$

A → 10

Binary to decimal

1) $(1110)_B$

3 2 1 0 ← Position

1 1 1 0

Value = $\sum$ bit $\times$ (base)$^{\text{Position}}$

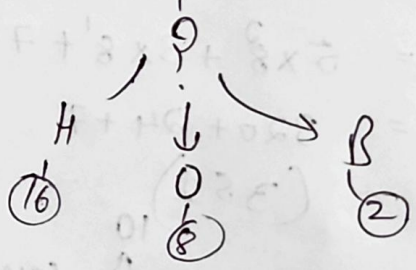
Value = $1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

Value = $8 + 4 + 2 + 0$

Value = $(14)_{10}$

Conversion of Decimal to Other number System

10



Procedure →

Required base Rb | Decimal number

Q1 ← Quotient

r_1 → Remainder

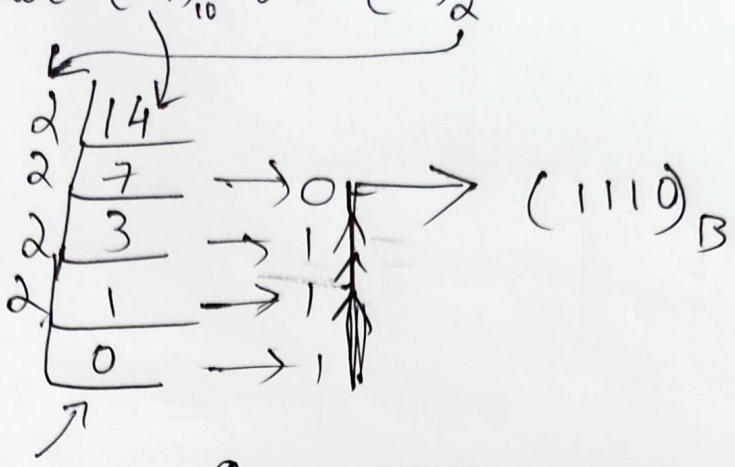
r_2

r_n

Q_n

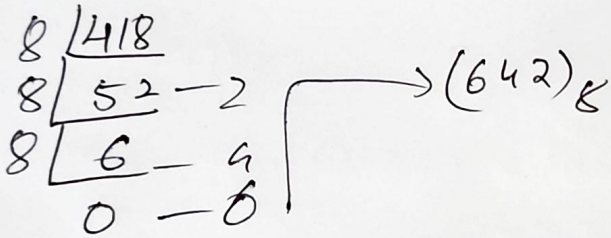
$= 0$

ex :- Convert $(14)_{10}$ to $(\ )_2$

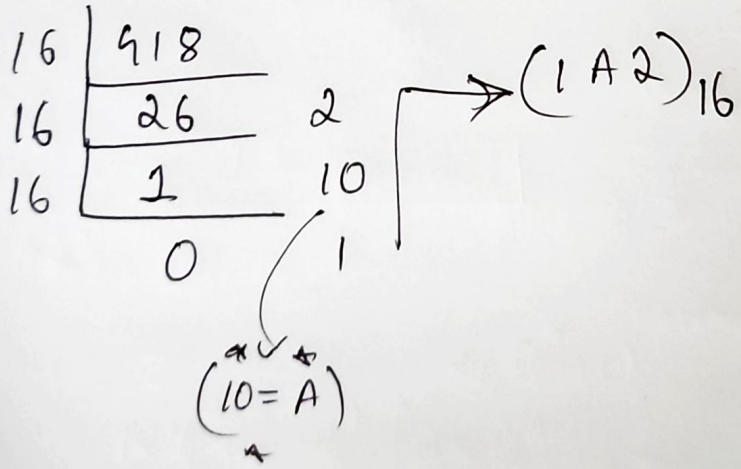


★ Stop when Quotient is zero ★

Convert $(418)_{10}$ to $(\ )_8$



Convert (418) to $(\ )_{16}$



Number System:-

- Decimal number system
- Binary number system
- Hexadecimal number system
- Octal number system

Decimal number system:- → Base = 10

- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Octal number system:- → Base = 8

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

Hexadecimal number system → Base = 16

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

Binary number system:- → Base = 2

- 0, 1 → "Bits"

Base is number of symbols (digits) in the number system

Conversions b/w number system:-

Conversion b/w any system to decimal

$$\text{Value in decimal} = \sum \text{Digit} \times (\text{Base})^{\text{Position}}$$

This is called Weight

For instance:-

1 0 ← position

9 9

This is decimal itself

$\therefore$ Value $= 9 \times 10^1 + 9 \times 10^0 = 99$

Q $(537)_8 \rightarrow (?)_{10}$

$5 \times 8^2 + 3 \times 8^1 + 7 \times 8^0 = (351)_{10}$

Q $(1A2)_{16} = (?)_{10}$

Equivalent

$1 \times 16^2 + A \times 16^1 + 2 \times 16^0 = (418)_{10}$

Q $(1110)_2 = (?)_{10}$

$1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = (14)_{10}$

Conversion from decimal to other systems:-

Required Base R.B

Decimal number	
Quotient 1 (Q1)	Remainder r_1
Q2	r_2
⋮	

Until quotient is zero

Qn	.... r_n
0	

Converted number = $(q_n r_{n-1} r_{n-2} \ldots r_2 r_1)$

Q $(14)_{10} \longrightarrow (?)_2$

2 | 14		
2 | 7	- 0	↑
2 | 3	- 1	↑
2 | 1	- 1	↑
0	- 1	↑

$(1110)_2$ → Required number in Binary

Q $(418)_{10} \rightarrow (?)_8$

8	418	
8	52	- 2
8	6	- 4
	0	- 6

$(642)_8$ → Required number in octal

Q $(418)_{10}$ to $(?)_{16}$

16	418	
16	26	- 2
16	1	- A
	0	- 1

$(1A2)_{16}$ → Required Number in Hexadecimal

Boolean Algebra :-

★ Rules :-

→ Compliment :- $\bar{X} / x' / X^{\nu}$

$0 = \bar{1}$

$1 = \bar{0}$

★ $\overline{(\bar{A})} = A$

★ $A + 0 = A$
$A + 1 = 1$
$A + A = A$
$A \cdot A = A$
$A + \bar{A} = 1$
$A \cdot 0 = 0$
$A \cdot \bar{A} = 0$

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

↑ AND operator (Dot operator) ↑ Logical Product

OR operator (+ operator) ↑ Logical Sum

★ $\bar{A} + \bar{A} B = A + B$

SoP form :-
↳ Sum of Product → Minterm (m_x) in Canonical form

Q Simplify :- $Y = ABC + \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + B\bar{C} + A\bar{C} + A$

Soln :-

$Y = ABC + \bar{B}\bar{C}(\bar{A}+A) + B\bar{C} + A(\bar{C}+1)$

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

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

$= ABC + \bar{C} + A = A(1+BC) + \bar{C} = A + \bar{C}$

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

POS expression :-
↳ Product of Sum

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

↳ This can also be Similarly simplified as above using boolean algebra rules

★ Standard form of SoP & PoS is called canonical form

If there are 3 variables A, B, C

→ normal SOP, Eg: $AB + B\bar{C}\bar{A}$
↳ Because C is missing in first term

→ Standard SOP, Eg: $\bar{A}BC + AB\bar{C}$

Canonical form !!

!!! Even for POS Each term of the Standard form should contain all the I/P variables be it in normal form or complimented form

Logic Gates :-
↳ Has many inputs and single output

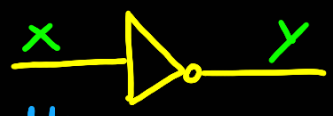
Basic Gates :- i) NOT
ii) AND
iii) OR

Universal Gates :- i) NAND
ii) NOR

Other Gates :- i) XOR
ii) XNOR

NOT Gate:-

Symbol:-

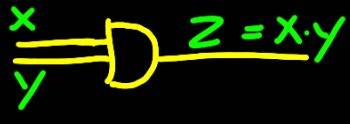


Truth table

X	Y
0	1
1	0

AND Gate:-

Symbol:-



Truth table:-

X	Y	Z = X·Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Gate:-

Symbol:-

Truth table:-

X	Y	Z = X+Y
0	0	0
0	1	1
1	0	1
1	1	1

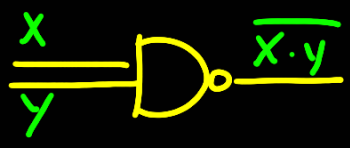
Universal Gates:-

- NAND gate
- NOR gate

Universal Gates can be used to realise any other gate.

NAND gate:-

Symbol:-

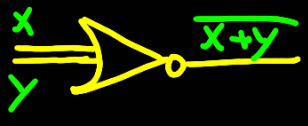


Truth table:-

X	Y	$Z = \overline{X \cdot Y}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate:-

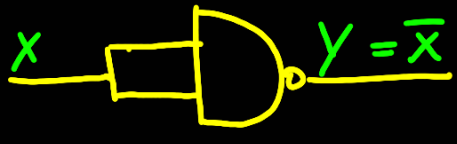
Symbol:-



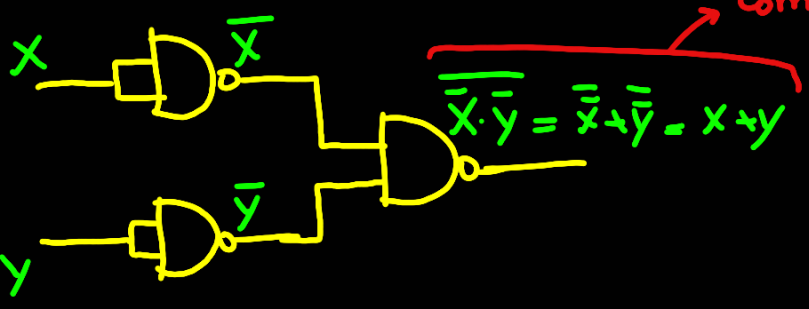
Truth table:-

X	Y	$Z = \overline{X+Y}$
0	0	1
0	1	0
1	0	0
1	1	0

NOT using NAND:-



OR using NAND:-

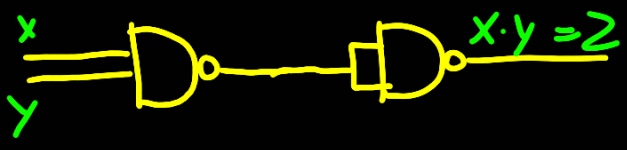


Tourth table:-

X	y	$\bar{x}$	$\bar{y}$	$\bar{x}\cdot y$	$\overline{\bar{x}\cdot\bar{y}} = z$
0	0	1	1	1	0
0	1	1	0	0	1
1	0	0	1	0	1
1	1	0	0	0	1

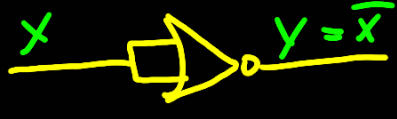
This is OR Gate

AND using NAND:-

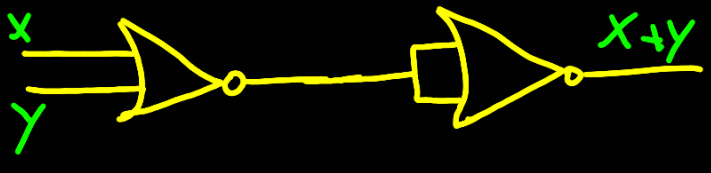


Prove using tourth table

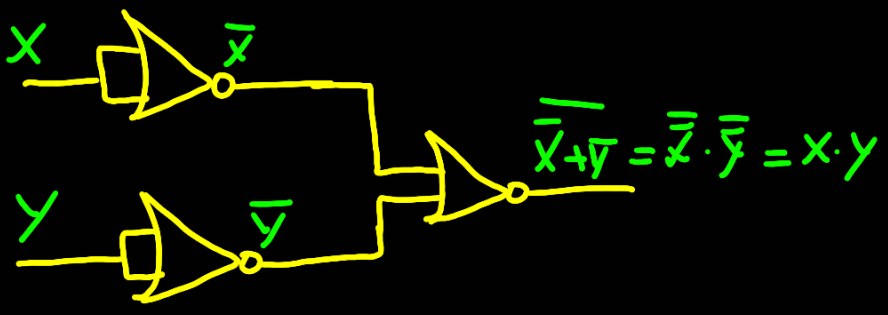
NOT using NOR:-



OR using NOR:-



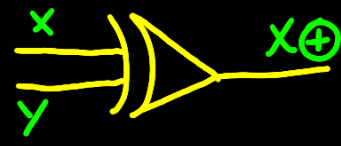
AND using NOR:-



Prove using Tourth Table

XOR Gate:- Gate whose O/P is 0 when I/P are equal else it is 1

Exclusively OR



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

Truth table:-

X	Y	$Z = x \oplus y$	Minterm
0	0	0	$\bar{x}\cdot\bar{y}$
0	1	1	$\bar{x}\cdot y$
1	0	1	$x\cdot\bar{y}$
1	1	0	$x\cdot y$

$Z = \Sigma(m_1, m_2)$

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

Expression for XOR

XNOR is compliment of XOR

De Morgan's Theorem:- Just distribute the compliment on top even to the logic operators, that's it

Theorem 1:- Compliment of Sum is product of individual compliments

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

Theorem 2:- Compliment of Product is sum of individual Compliments

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

Proof:-

A	B	$\bar{A}$	$\bar{B}$	$A\cdot B$	$A+B$	$\bar{A}\cdot\bar{B}$	$\bar{A}+\bar{B}$	$\overline{A\cdot B}$	$\overline{A+B}$
0	0	1	1	0	0	1	1	1	1
0	1	1	0	0	1	0	1	1	0
1	0	0	1	0	1	0	1	1	0
1	1	0	0	1	1	0	0	0	0

From the above T·T ⇒ $\overline{A+B} = \bar{A}\cdot\bar{B}$ → 1st Theorem

$\overline{A\cdot B} = \bar{A}+\bar{B}$ → 2nd Theorem

Q Simplify and realize logic circuit:-

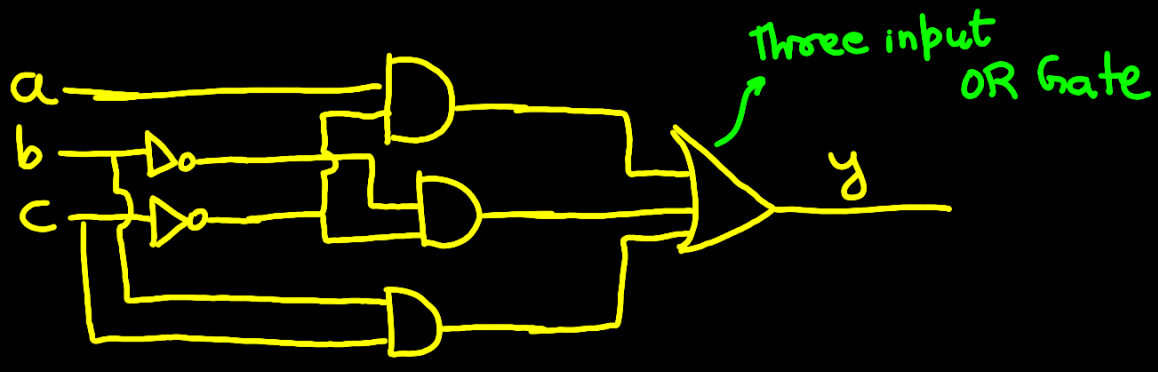
$$y = abc + \bar{a}\bar{b}\bar{c} + \bar{b}\bar{c} + a\bar{c} + bc$$

$$\Rightarrow y = abc + \bar{c}(\bar{b}(a+1)) + a\bar{c} + bc$$

$$= abc + \bar{b}\bar{c} + a\bar{c} + bc$$
$$= bc(a+1) + \bar{b}\bar{c} + a\bar{c}$$
$$= bc + \bar{b}\bar{c} + a\bar{c}$$

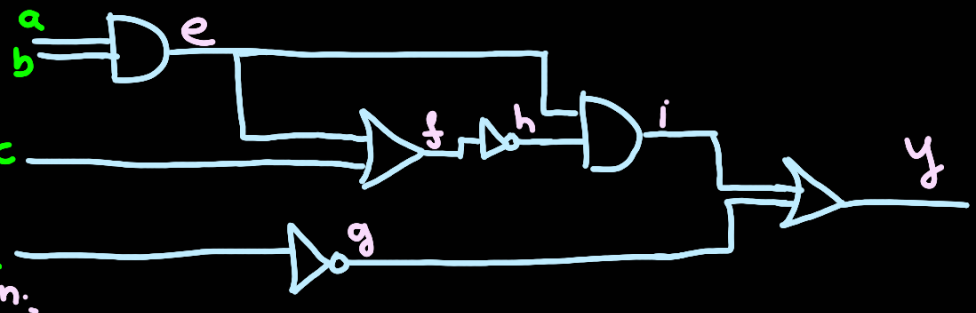
$$\therefore y = \bar{b}\bar{c} + bc + a\bar{c}$$

Logic ckt:-



Ill ly we can make ckt for unsimblified ckt too

Q Find the boolean expression for the following:-



Soln:-

$$y = i + g$$

$$g = \bar{d}$$
$$i = e \cdot h$$
$$e = a \cdot b$$

$$h = \bar{f}$$
$$f = e + c = a \cdot b + c$$

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

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

$i = e \cdot h = (a \cdot b)(\bar{a}\bar{c} + \bar{b}\bar{c})$

$= a\bar{a}\bar{c} + a\bar{b}\bar{c} + b\bar{a}\bar{c} + b\bar{b}\bar{c}$ (0, 0)

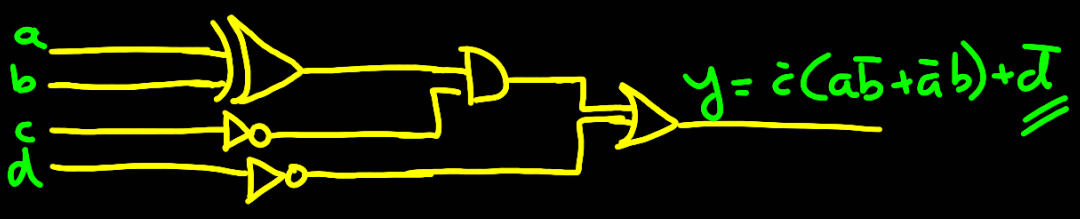
$i = a\bar{b}\bar{c} + \bar{a}b\bar{c}$

$y = i + g = a\bar{b}\bar{c} + \bar{a}b\bar{c} + \bar{d}$

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

XOR

Simplified Ckt:-



Combinational circuits:

↳ O/P depends only on inputs

i.e No I/P ⇒ No O/P

Half adders & Full adders

Adds 2 bits and gives 2 O/P Sum & Carry

Adds 3 bits and gives 2 O/P Sum & Carry } 3rd bit is carry in from previous step

Half adder:-

Let a, b be two I/P bits

Truth table:-

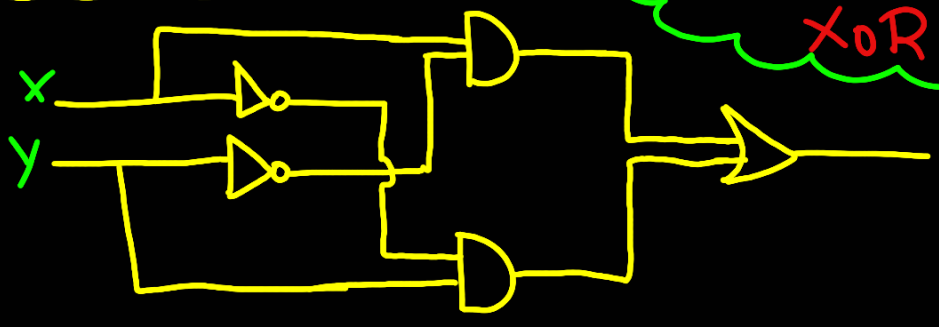
X	Y	Sum	Carry	
0	0	0	0	m_0
0	1	1	0	m_1
1	0	1	0	m_2
1	1	0	1	m_3

Consider Sum O/P :-

$$S = \sum(m_1, m_2) = \bar{x}y + x\bar{y}$$

This is Basically XOR Gate

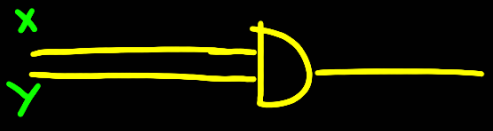
Sum ckt :-



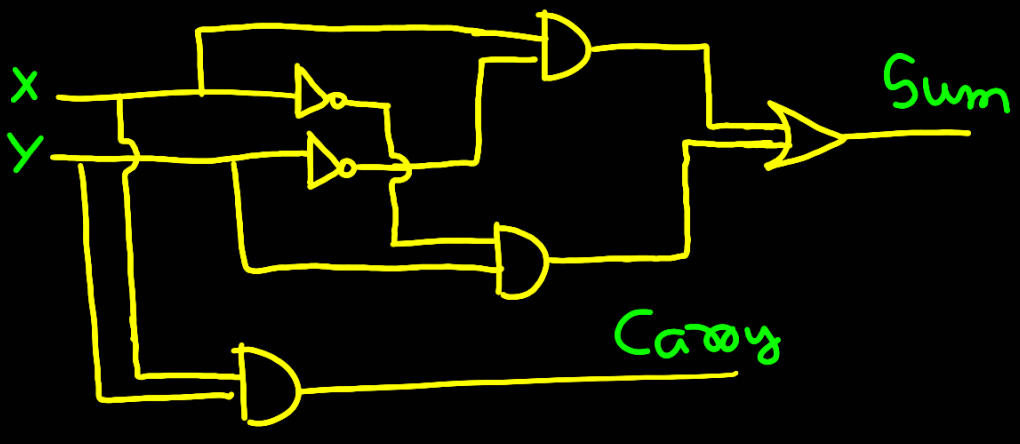
Consider Carry O/P :-

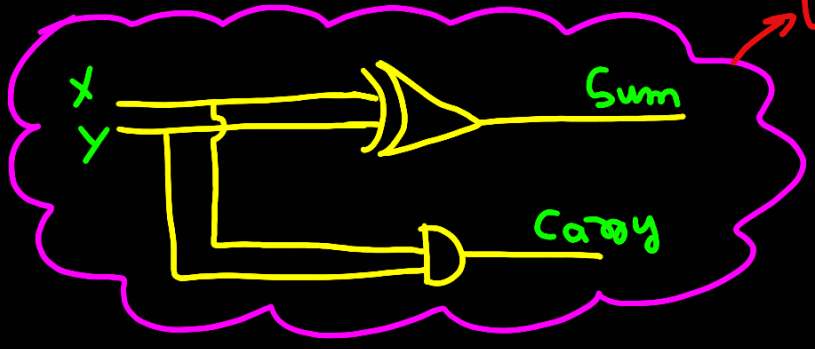
$$C = \sum m_3 = xy$$

Carry Ckt :-



☆ Complete Half adder circuit :-





→ Ultra Simple Ckt

#Full adder:-

↳ Adds 3 bits

Two inputs are X & Y

Third one is Carry in (C_i)

Truth table:-

	X	Y	C_i	S	C_o
m_0	0	0	0	0	0
m_1	0	0	1	1	0
m_2	0	1	0	1	0
m_3	0	1	1	0	1
m_4	1	0	0	1	0
m_5	1	0	1	0	1
m_6	1	1	0	0	1
m_7	1	1	1	1	1

$Sum = \sum(m_1, m_2, m_4, m_7)$

$Carry = \sum(m_3, m_5, m_6, m_7)$

Consider Sum:-

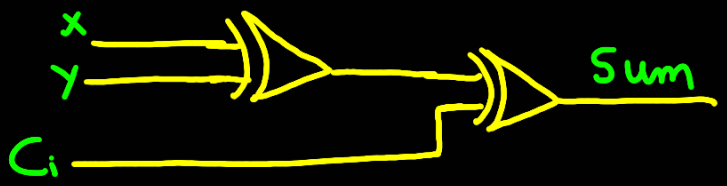
$$S = \bar{X}\bar{Y}C_i + \bar{X}Y\bar{C}_i + X\bar{Y}\bar{C}_i + XYC_i$$

$$= C_i(\bar{X}\bar{Y} + XY) + \bar{C}_i(\underbrace{\bar{X}Y + X\bar{Y}}_{XOR})$$

$$S = C_i \oplus (X \oplus Y)$$

Simplest Possible Ckt

Sum Ckt:-



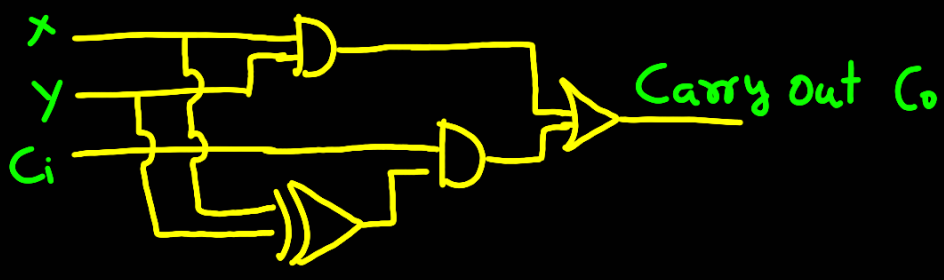
Consider Carry/Carry out (C_o):-

$$C_o = \bar{X}\,Y\,C_i + X\bar{Y}\,C_i + XY\,\bar{C}_i + XY\,C_i$$

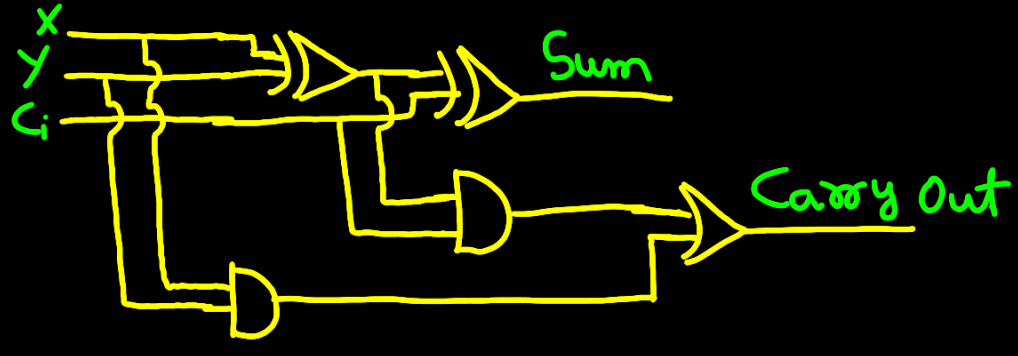
$$= C_i(\bar{X}\,Y + X\bar{Y}) + XY(\bar{C}_i + C_i)$$

$$C_o = C_i(X \oplus Y) + XY$$

Carry out Ckt:-



☆ Complete Full adder Ckt



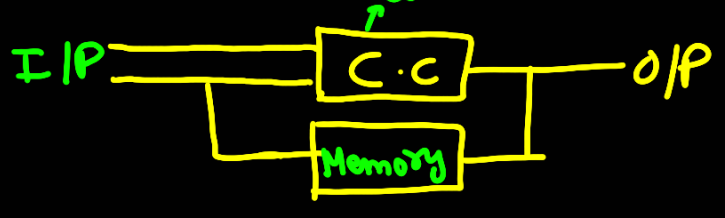
Sequential Circuits:-

↳ Depends on not only the I/P but also on previous O/P

↓

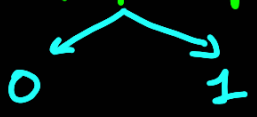
Memory ← PAST

Block Diagram:-



Memory → (to hold data)
- ↳ Also a logical circuit (Combinational)
- ↳ Use logic gates
  - ↳ Universal Gates
    - ↳ NAND / NOR

1 bit of data can be held in a "Flip Flop"

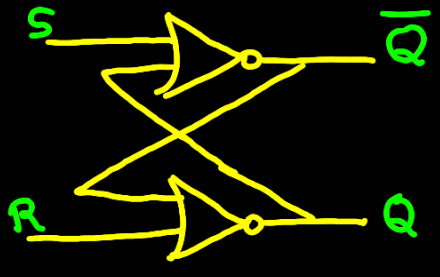


Flip Flops:-

- SR f/f → Set-Reset f/f
- D f/f → Data f/f
- JK f/f → I/P are J & K
- T f/f → Toggle f/f

SR Flip-Flop:-

NOR Based

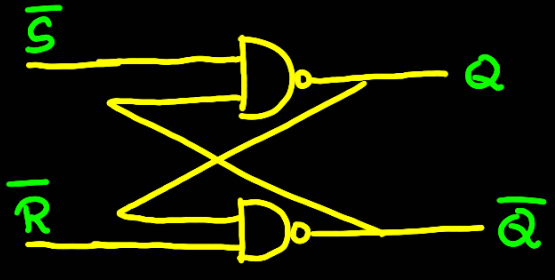


Uses 2 NOR Gates

I/P are S & R

O/P are $\bar{Q}$ & Q

NAND Based



Uses 2 NAND Gates

I/P are $\bar{S}$ & $\bar{R}$

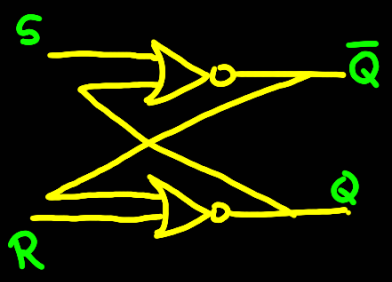
O/P are Q & $\bar{Q}$

The outputs are cross coupled and fed back to I/P

★ Characteristics Table / Transition Table:-

NOR Based:-

"Active high" ← Responds to high values

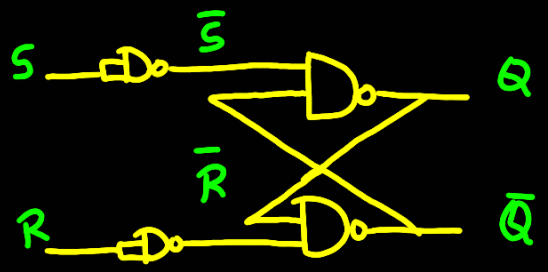


S	R	Q	$\bar{Q}$	
0	0	No change		
0	1	0	1	O/P is SET
1	0	1	0	O/P is RESET
1	1	0	0	Invalid I/P

Q & $\bar{Q}$ are complimentary to each other, thus cannot have same value

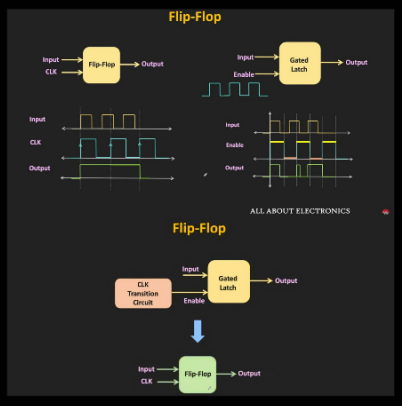
NAND Based:-

"Active low" ← Responds to Low values



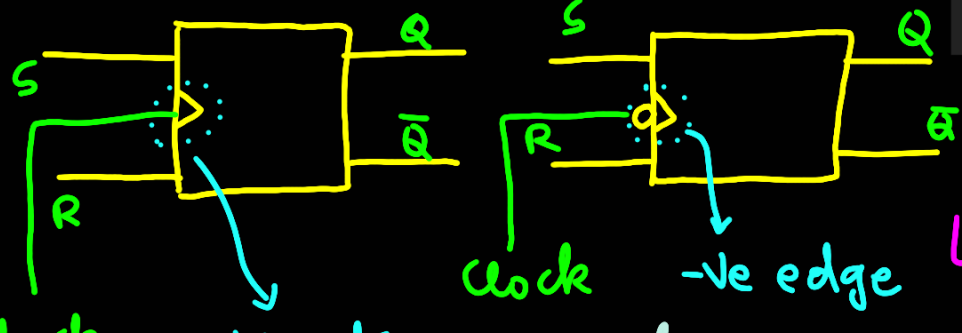
$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	
0	0	1	1	→ Invalid I/P
0	1	1	0	→ o/p is set
1	0	0	1	→ " " Reset
1	1	No change		

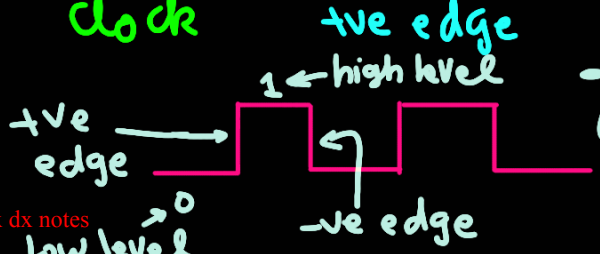
} o/p is Q



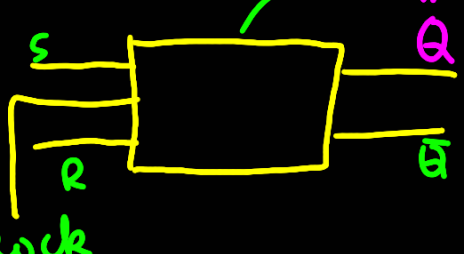
Gated Latch (Clocked Latch)

↳ is flip flop

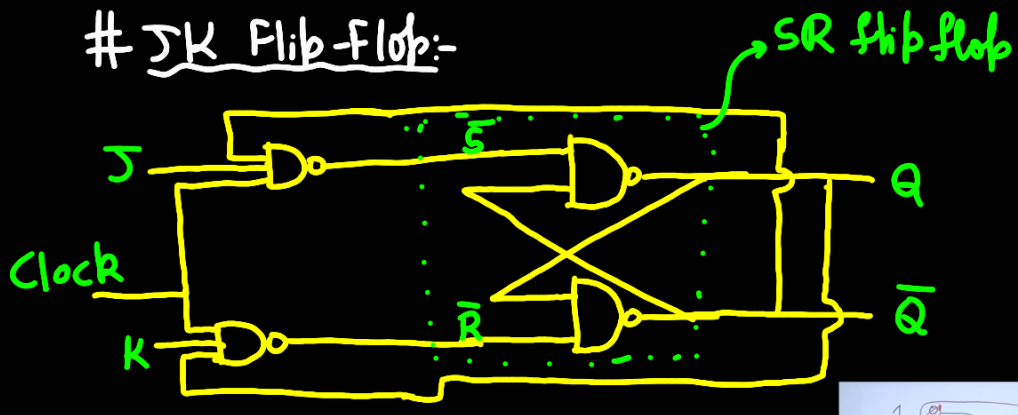




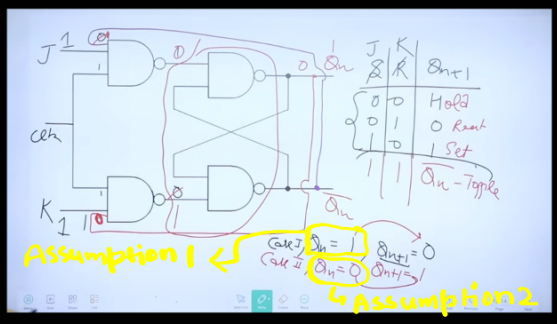
Level Triggered Flip Flop



JK Flip-Flop:-



Since in SR flip flop the 1 1 I/P is invalid to avoid that and make use of that state also we need J K flip flop where if both the inputs are high we get some value in the O/P which is a toggle, where it exchanges the pre existing values



Characteristics table

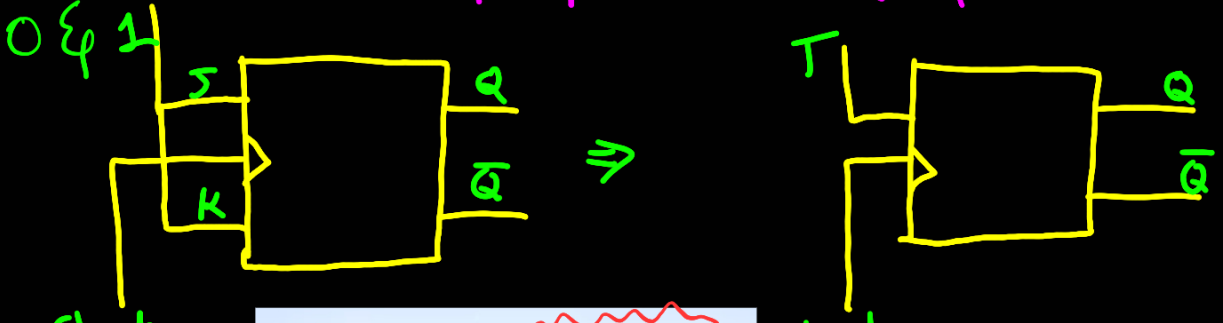
J	K	Next State
0	0	No change
0	1	(J) set
1	0	(K) set
1	1	Toggle

} The value of J/K is set

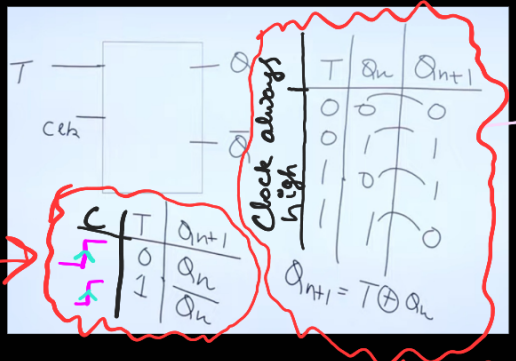
↳ Toggles, the pre-existing o/p if I/P is 11 (exchanges)

Toggle Flip-Flop:-

↳ Short J & K & make them 1 & 0



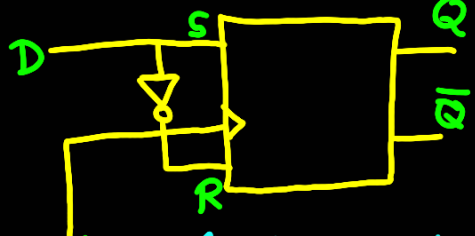
Gen. truth table for T flip flop



If toggle is low, previous o/p is retained, if toggle is high, previous o/p is toggled

↳ Charachterstic table

D Flip-Fl :-



D	Clock	Next output (Q_{n+1})
1	↑	1
0	↑	0

o/p follows the data

o/p is the D

Qn+1 (final O/P) depends only on the I/P which is D and *DOES NOT* depend on the first O/P which is Q

Shift Registers:-

↓

Records/Stores data

↓

Number

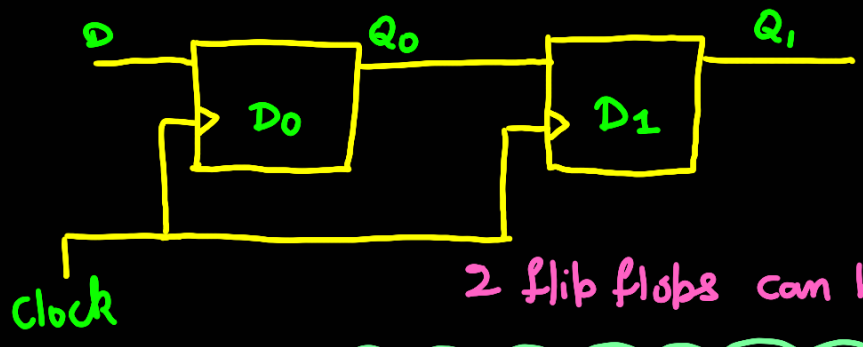
↓

Binary number

1 0 } bits

A flip flop can hold/store 1 bit of data either '0' or '1'

To store data >1, multiple flip flops are required.



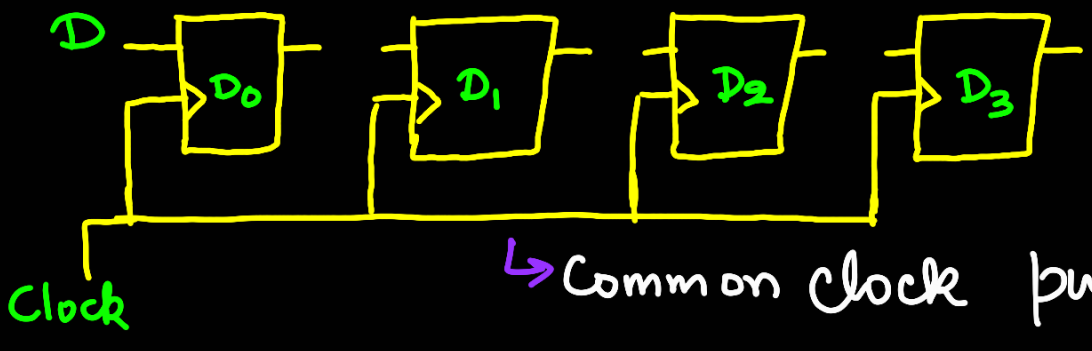
2 flip flops can hold 4 Data values

∴ N f/fs can hold max of = $2^N - 1$ data values

Suitable is D ffs

* Group of flip flops act as a register, which can hold a value.

4-bit Shift Registers:-



We need to perform two operations :-

1) Loading the register
2) Reading the register

★ Loading the register:-

→ Load the data parallelly / serially

Serial Loading	Parallel Loading
★ One data bit is loaded at every +ve pulse of the clock, ∴ rising edge clk	★ All are loaded together
★ Data is loaded at first flip flop	★ All at once
★ Slow loading	★ Fast loading

② Reading the register:-

Reading of data maybe in parallel or series

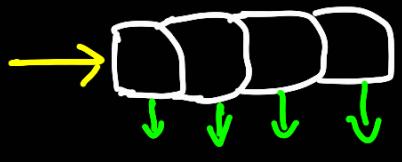
Shift Register Operations:-

yellow → I/P

green o/p

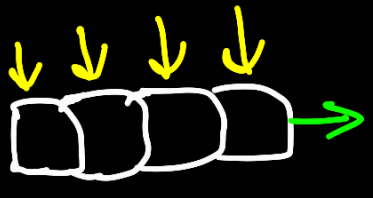
① Serial In Serial out (SISO)

② SIPO

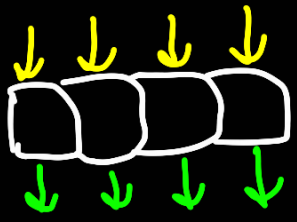


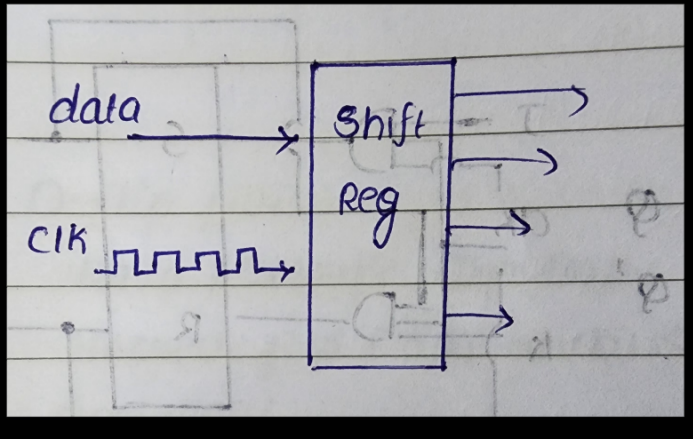
4 Bits → 4 flip flops

③ PISO



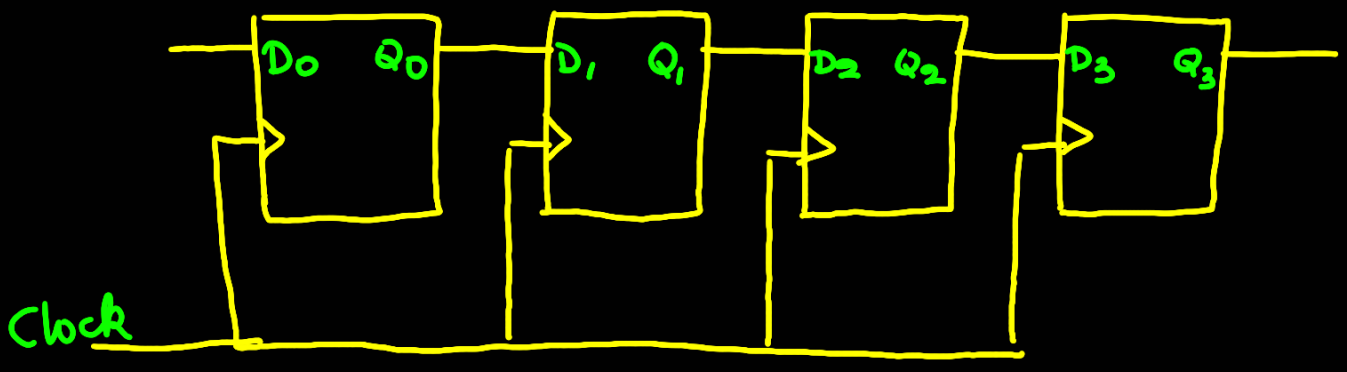
④ PIPO





#SISO:-

* Note : Rising Edge Clock pulse

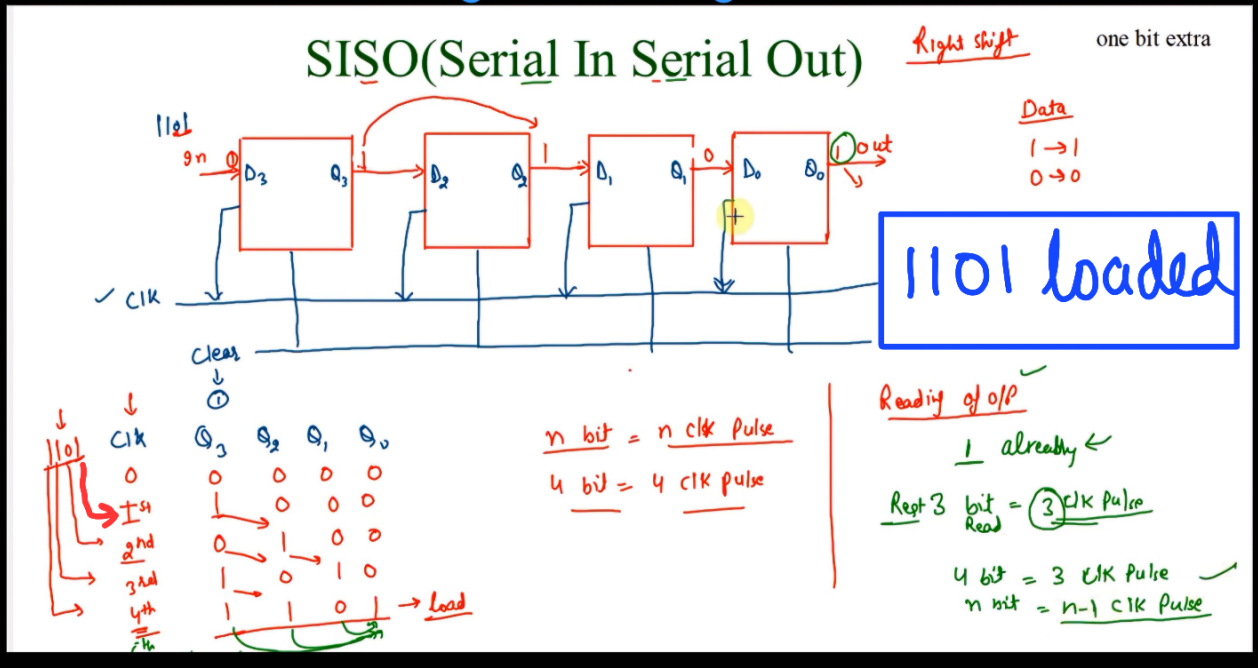


* The Shift Register is 4 bits in size (Can be N bits)
* Clock common to all f/fs
* Data is fed to the 1st f/f in series and data read serially out from last f/f
* The o/p of the previous f/f is I/P for the next f/f
* Thus data shifts from D_0 to D_3
* To load the data 'N' clock pulses are required where N is no. of f/f. In this case 4
* To read the data n-1 Clock pulse are req. in this case 3
* Total time required to read & load is $2n-1$

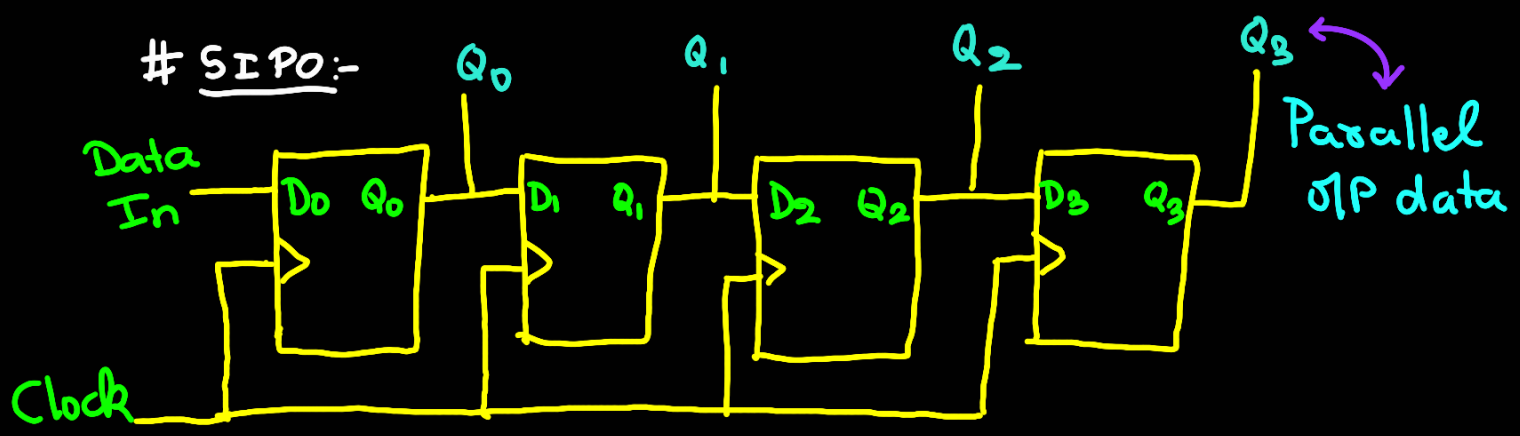
$$n + n - 1$$

load ← n + $n-1$ → read

Loading & Reading of SISO



SIPO:-

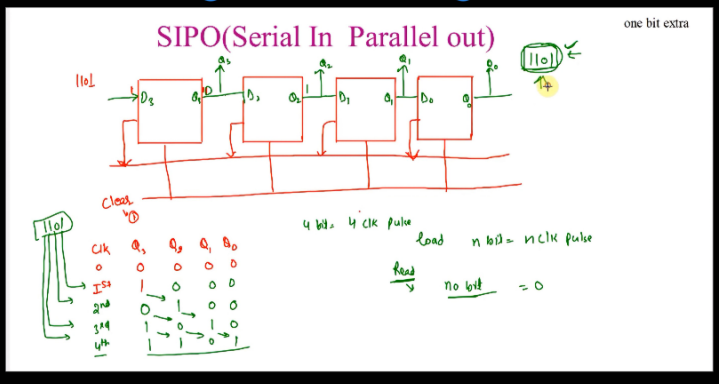


* Loading is Serial
* Reading is parallel
* To load it takes 'N' clock pulses, here it is 4
* At the end of the N^{th} clock pulse, data is ready to be read at once, without any clock pulse separately for reading
* Total time required to read & load is n

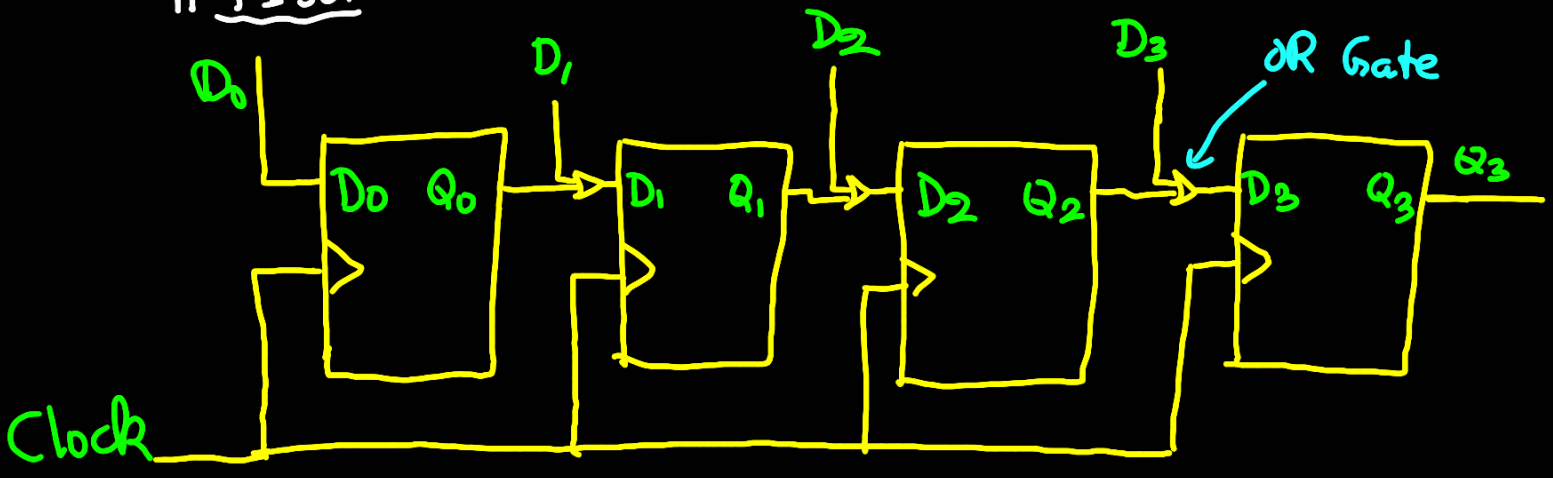
$n + 0$

load ↵ ↳ read

Loading & Reading of SIPO



PISO:-

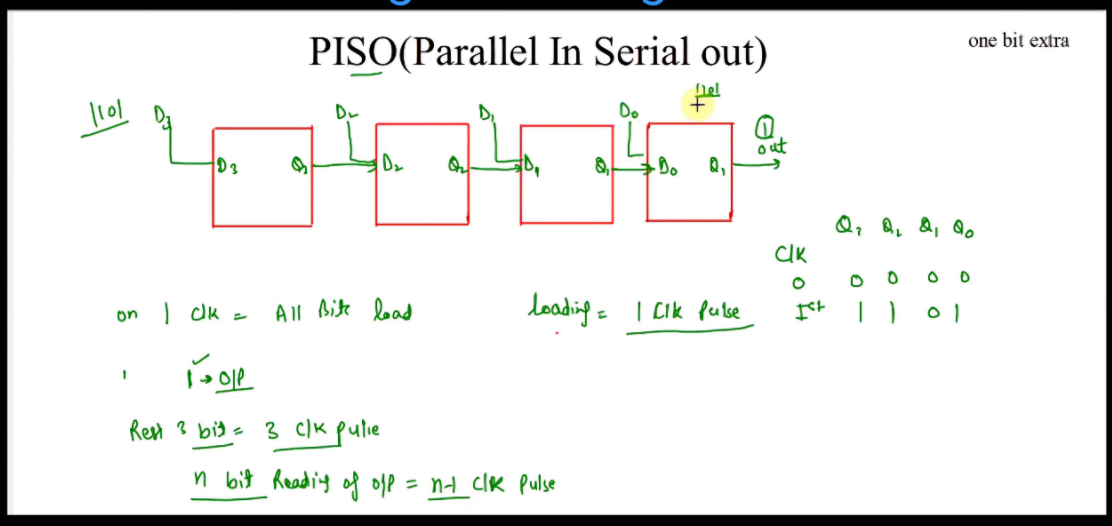


* Data is loaded to all the f/f in parallel at once
* Data is read serially from D_3
* 1 Clock loads all the bits parallely
* For reading we need $n-1$ clock pulse
* Total no. of Clock for reading & writing = $2n$

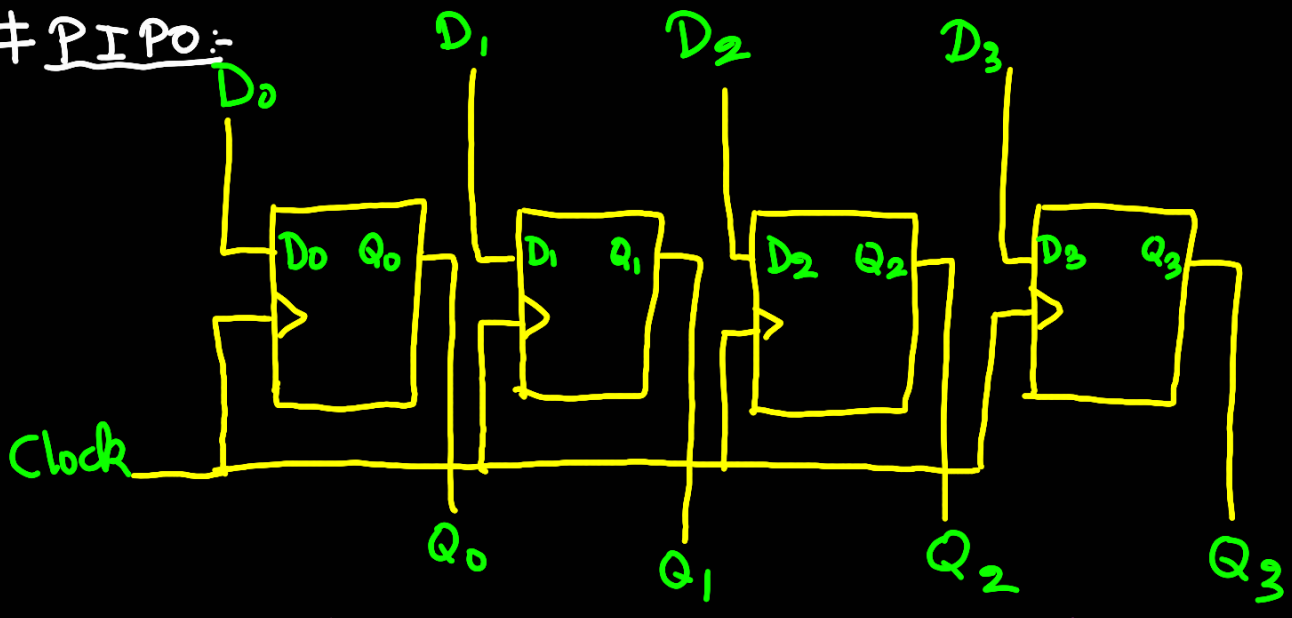
$1 + 2n-1$

load ↵ ↳ Read

Loading & Reading of PISO



PIPO:-

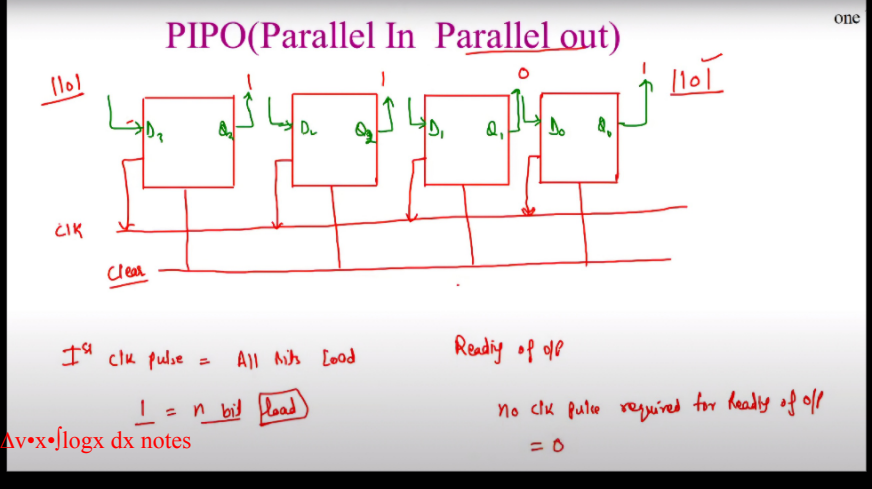


* Data loaded in 1101 at once, so only 1 clk pulse needed
* Data can be read immediately after loading, so for reading 0 clock pulse is needed.
* Total of clk pulse needed is 1

1 + 0

1 → load, 0 → read

Loading & Reading of PIPO



NOTE:
SISO is the slowest register
PIPO is the fastest register

Mode	Loading	Reading	Total
SISO	n	n-1	2n-1
SIPO	n	0	n
PISO	1	n-1	n
PIPO	1	0	1

Counters:-

- It is a Sequencial ckt consisting of f/fs
- Counters indicate a no of events, and events are applied through clock pulses
- Counter value is updated on every clock pulse

* Counters can indicate max value

* Max values depend on no. of f/f (n) & is given by $2^N - 1$

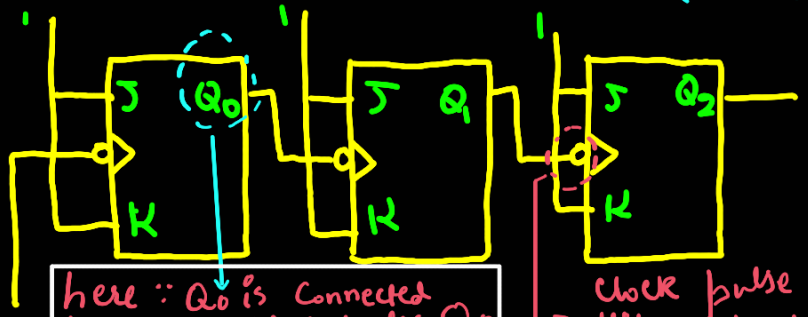
* If it has 3 bits then it is 3-bit counter and it can count from $0 \to 2^{n-1}$, i.e $0 \to 2^{3-1}$

$\Rightarrow 0 \to 7$ (8 counts)

2^n counts

* Counter can be
  - Asynchronous
  - Synchronous

3-bit Asynchronous counters :- (Ripple Counter)



here :: Q_0 is connected to bubbled clock pulse, Q_0 operates in falling edge of clock pulse

Clock pulse with bubble, meaning flip flop works only in falling edge i.e when sa. pulse comes down ↓ only

* Only when purple arrow encountered (falling edge) the toggling takes place. And initially all are assumed to be of zero

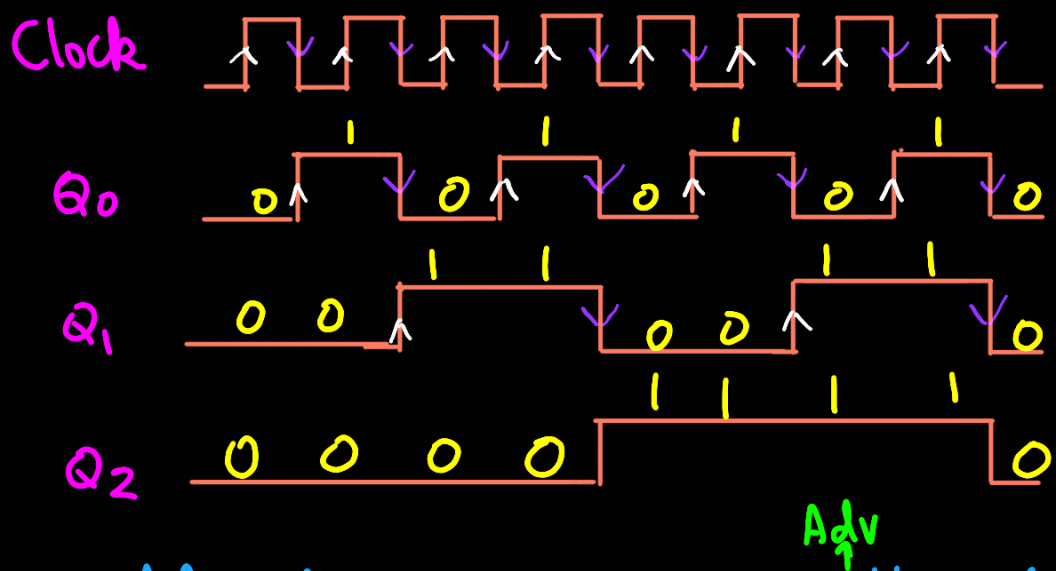
* Similarly Q_1 depends on bubbled clock pulse of Q_0 & So on....

★ In case of asynchronous counter, clock is applied to first f/f

★ Subsequent f/f have clock input fed from o/p of previous f/f, as a result it is slow, because they depend on previous f/f's clock pulse/o/p

Counter output :-

Timing waveform



Respond to -ve edge of each subsequent clock

and the clock for next f/f depends on the o/p of previous f/f

★ Asynchronous counters are simple and slow

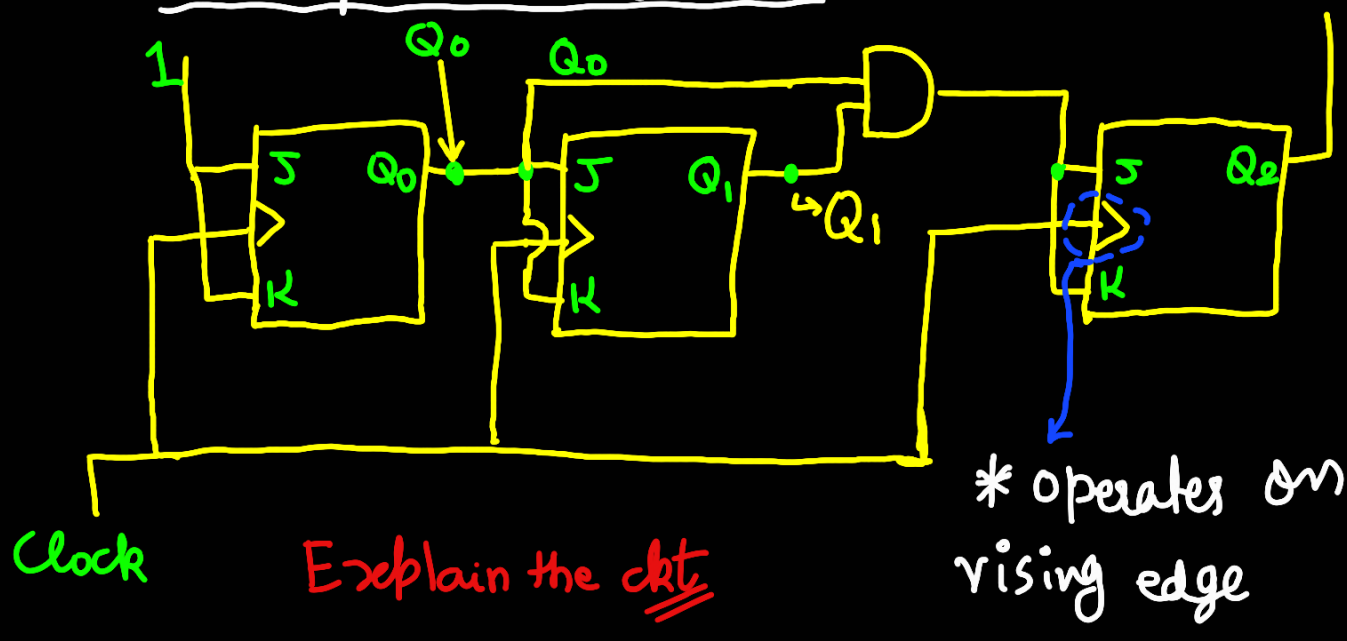
★ Propagation delay ↑es with more no. of f/fs in the counter

★ N f/f counters will have $N \times t_p$ times of delay

where N → No. of f/f

t_p → Propagation delay of f/f

3-bit Synchronous Counters:-



Explain the ckt