Number System

- Decimal System.
  use 10 symbol's
  (0,1,2,3 ..... 9)

ex: 22706
then each position has some weight

2	2	7	0	6
10^4	10^3	10^2	10^1	10^0

$\Rightarrow (22706)_{10}$

$\Rightarrow 2\times10^4 + 2\times10^3 + 7\times10^2 + 0\times10^1 + 6\times10^0$

$\Rightarrow 22706$

- Binary System
  use 2 symbol's (0,1)
  ex 011001 then each position has some weight.

0	1	1	0	0	1
2^5	2^4	2^3	2^2	2^1	2^0

$\Rightarrow (101100)_2 \Rightarrow (44)_{10}$

- Octal system use 8 symbol (0,1,2,3.....7)

ex: $(77)_8$ (Octal System) $\Rightarrow 7\times8^1 + 7\times8^0$
$\Rightarrow 56+7$
$\Rightarrow (63)$

ex: $(abc)_8 \Rightarrow (?)_{10}$
$8^2\ 8^1\ 8^0$
every position has a weight.

$\Rightarrow a\times8^2 + b\times8^1 + c\times8^0$
$\Rightarrow (64a+8b+c)_{10}$.

- Hexadecimal system.
  use 16 symbols.
  (0,1,2,3 ----- 9, A, B, C, D, E, F)
  10 11 12 13 14 15

ex: $(10F)_{16} = (?)_{10}$
$16^2\ 16^1\ 16^0$
Base 16 Hexidecimal System.

$\Rightarrow 1\times16^2 + 0\times16^1 + F\times16^0$
$\Rightarrow 16^2 + 15$
$\Rightarrow (271)_{10}$.

- Digital logic
  - → ~~No system~~
  - → ~~Self Compl^n code/ Binary to gray~~
  - → full adder & half adder circuit.
  - → Look ahead carry
- C programming
  - → hashing
  - → pointers.

Base γ System to Decimal System

Base γ System : $\gamma > 1 ; \gamma \in N$

Symbol's $= \gamma$ $\{0, 1, \ldots \gamma - 1\}$

Base $\gamma = 18$

Symbols: $0, 1, 2 \ldots 9, A, B, \ldots H$

Symbol 10 X

15 X

ex: $(abcd)_\gamma = (\ ?\ )_{10}$ ($a^3\ b^2\ c^1\ d^0$)

Base γ

$\Rightarrow a\times\gamma^3 + b\times\gamma^2 + c\times\gamma^1 + d\times\gamma^0$

ex: $(abc.de)_\gamma = (\ ?\ )_{10}$ ($a^2\ b^1\ c^0\ d^{-1}\ e^{-2}$)

Base γ

$\{0, 1, 2 \ldots \gamma - 1\}$

$\Rightarrow a\times\gamma^2 + b\times\gamma^1 + c\times\gamma^0 + d\times\gamma^{-1} + e\times\gamma^{-2}$

ex: $(11)_{16} = B$? X

$(11)_{10} = (B)_{16}$. ✓

$(11)_{16} \Rightarrow 1\times16^1 + 1\times16^0$

$\Rightarrow (17)_{10}$

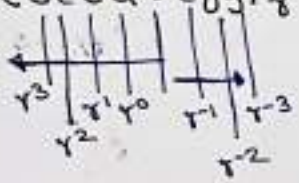
ex: $(101.01)_2 = (\ ?\ )_{10}$ ($2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}$)

$\Rightarrow 2^2\times1 + 1\times2^0 + 1\times2^{-2}$

$\Rightarrow 4 + 1 + \frac{1}{4} \Rightarrow \boxed{5.25}$

ex: convert Radix γ ⟶ Decimal.

$(dcba.efg)_\gamma \longrightarrow (d\times\gamma^3 + c\times\gamma^2 + b\times\gamma^1 + a\times\gamma^0 + e\times\gamma^{-1} + f\times\gamma^{-2} + g\times\gamma^{-3})_{10}$



Decimal to Base 'r' System conversion

We know $(abc)_r \Rightarrow (N = ar^2 + br + c)_{10}$

$N = (ar^2 + br + c)_{10} \Rightarrow (abc)_r$

Divide by "r"

Quotient: $ar + b = Q_1$

Remainder: $c = R_1$

$ar^2 + br + c = Q_1 r + R_1$

$ar^2 + br + c = (ar + b)$

$c = \boxed{N \bmod r}$

$Q_1 = (ar + b)$

So, $ar + b = Q_2 r + R_2$

$ar + b = ar + b$

Divide by r.

Quotient : $Q_2 : a$

Remainder : $R_2 : b$

$b = \boxed{\left(\frac{N}{r}\right) \bmod r}$

$Q_2 = a$

Divide by r

So, $a = Q_3 r + R_3$

$a = a$

Quotient : $Q_3 : 0$

Remainder : $R_3 : a$

ex: $(6079)_{10} = (\ ?\ )_8$

we are assuming to be $(xyzw)_8$

$\Rightarrow$ then $(8^3x + 8^2y + 8z + w) = 6079 = N$

w: $N \bmod 8 = 6079 \bmod 8 = 7$

$R_1 = 7$, $Q_1 = 759 = 8^2x + 8y + z$

z: $Q_1 \bmod 8 : 7$

$Q_2 = 94 = 8x + y$; $R_2 = 7$

y: $Q_2 \bmod 8 : 6$

$Q_3 : x$, $R_3 = 6$

x : $Q_3 \bmod 8 =$

Remainders

8	6079	
8	759	7 (LSB)
8	94	7
8	11	6
8	1	3
	0	1 (MSB)

ex: $(31)_{10} \longrightarrow (?)_{16}$.

16	31
16	1
	0

Remainders.
15 = F ↑
1

$(1F)_{16}$

ex: $(11)_{10} \longrightarrow (B)_{16}$

ex: $(11)_{16} \longrightarrow 16+1$
$(17)_{10}$

ex $(17)_{10} \Rightarrow (?)_{16}$

16	17	
16	1	1 ↑
	0	1

$(11)_{16}$

$(.ab)_{10} \longrightarrow (?)_r$
N

$\longrightarrow (.xyz)_r$
assuming.

$\frac{x}{r} + \frac{y}{r^2} + \frac{z}{r^3} = .ab = N$

Find x: (N*r) → integer part = x

$\left(\frac{x}{r} + \frac{y}{r^2} + \frac{z}{r^3}\right) r = x + \left(\frac{y}{r} + \frac{z}{r^2}\right) \rightarrow m$
fractional.

Find y: (m*r)
⋮
Soon

ex $(.152)_{10} \longrightarrow (?)_8$
$(.1156\ldots)_8$

$.152 \times 8$
1 ← 1 . 216

$.216 \times 8$
1 ← 1 . 728

$.728 \times 8$
5 ← 5 . 824

$.824 \times 8$
6 ← 6 . 592

ex: $(.25)_{10} \longrightarrow (\ ?\ )_2$

$(.01)_2$

$.25 \times 2 = 0.50$ → 0

$.50 \times 2 = 1.00$ → 1

ex: $(.3)_{10} \longrightarrow (\ ?\ )_2$

$.3 \times 2 = 0.6$ → 0

$.6 \times 2 = 1.2$ → 1

$.2 \times 2 = 0.4$ → 0

$.4 \times 2 = 0.8$ → 0

$.8 \times 2 = 1.6$ → 1

$.6 \times 2 = 1.2$

$(0.0100110\ldots.)_2$

$(0.0100\overline{1001})_2$

Keep repeating

ex: $(31.4)_{10} \longrightarrow (\ ?\ )_4$

(we need to solve Integer & fraction part separately)

Integer part.

$(31)_{10}$

4 | 31		
4	7	3
4	1	3
	0	1

$(133)_4$

fraction part

$.4 \times 4 = 1.6$ → 1

$.6 \times 4 = 2.4$ → 2

$.4 \times 4 = 1.6$ → 1

repeat.

$(.\overline{12})_4$

So, $(31.4)_{10} \longrightarrow (133.\overline{12})_4$

ex: $(23.31)_4 \longrightarrow (?)_7$

$(\ ?\ )_{10}$

So, $(2\ 3 . 3\ 1)$

$4^1\ \ 4^0\ \ 4^{-1}\ \ 4^{-2}$

$(2\times4 + 3 + \frac{3}{4} + \frac{1}{16})_{10}$

$\Rightarrow (11.8125)_{10}$

$(11.8125)_{10} \longrightarrow (\ ?\ )_7$

Integer part.

$(11)_{10} \longrightarrow (?)_7$

7 | 11		
7	1	4
	0	1

$(14)_7$

fraction part:

$(.\overline{54})_7$

.8125 × 7 → 5.6875 → .6875 × 7 → 4.8125 → .8125 × 7 → Repeat

So, $(23.31)_4 \longrightarrow (14.\overline{54})_7$

ex: why are we learning only 2, 8, 16 Binary system, why not 4, 5, 6, 7 ... ?

the reason behind this is that a no. can be too large if we try to covert it directly then it will take time

So, we convert them in power of 2.

So, that we can groop them.

Octal: 0 ..... 7 → Binary octal

Base $8 = 2^3$

000 — 0
001 — 1
⋮
111 — 7

ex. $(110\ 11\ 010\ 101\ 100\ 101)_2$

$(3\ 3\ 2\ 5\ 4\ 5)_8$

Same with the Hexadecim. we groop with 4.

ex. How it is Beneficial for computer.

$(1000)_{10} \longrightarrow (\ ?\ )_2$

$(1000)_{10} \longrightarrow (\ ?\ )_{16}$

16	1000	
16	62	8
16	3	14 → E
	0	3

$(3E8)_{16}$

If $(3E8)_{16} \longrightarrow (\ ?\ )_2$

$(0011\ 1110\ 1000)_2$

Binary addition

Decimal addition

```
  1  1
  2  3  0  4
+ 9  8  7  6
-------------
  2  1  8  0
```

Equal to Base → 10 - Base

10 - 10 = 0

8 + 3 = 11 - Base = 1

2 - Base ⇒ 2

- Binary Addition (Base = 2)

```
  1 1 1 1 1 1
  1 0 1 1 0 1
+ 1 1 0 1 1 1
-------------
1 1 0 0 1 0 0
```

1 + 1 = 2 - Base = 0

a	b	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Subtraction

ex: 78 - 12

64

18

another way.

82 - 64 ⇒

82 + 100 - 100 - 64

⇒ 82 + (100 - 64) - 100

⇒ 82 + (99 - 64) - 100 + 1

(no need to borrow)

⇒ 82 + 35 + 1 - 100

⇒ (1)18 - 100) ≡ (discarding end carry from 118)

(end carry)

```
  1
  1
------
  1 0
```

Carry, Sum

```
  1 1
+ 1 1
-------
  1 0
```

= (10) Binary

Complement operation will help us with subtraction by using the same idea.

Complement of a no. in Base r

r's, (r-1)'s complement

Radix / Base r (r ≥ 2)

- (r-1)'s Complement. (Diminished Radix Complement)
- r's Complement (Radix Complement)

For (N) in Base r ($r \geq 2$)

↓ no. of 'n' Digits

- ($r-1$)'s complement: $(r^n-1)-N$
- r's complement: (r^n-N)

ex: Base 10:

given: $N=82 \rightarrow n$: no. of digit $=2$

9's comp. of N: $(10^2-1)-N \Rightarrow 99-82$

$\Rightarrow 17$

10's comp. of N:

$(10^2-N) = 100-82$

$= 18.$

ex: Base 10:

$N = 082 \rightarrow n=3$

9's comp. of N: $(10^3-1)-N = 999-082 = 917$

10's comp. of N: $(917+1) = 918$

Base 10:

- 9's complement of N: 'n' Digit. $\underbrace{99\ldots9}_{\text{'n' time}} - N$
- 10's comp. of N: (9's comp. of N) +1

Base 2:

- 1's Complement of N: $\underbrace{11\ldots1}_{n \text{ time}} - N$ (Complement every bit)
- 2's complement of N: (1's comp. of N) +1

10's comp.: $N = \underline{edcba}\ \underline{000}$

$(9-e)(9-d)(9-c)(9-b)(10-a)\,000$

ex: 10's comp. of $\underline{8201129}\ \underline{600}$

$1798870\ 400$

Subtraction of unsigned no. (m,n) using r's compl.

ex: M - N
(unsigned no.)

Case 1: M ≥ N
Case 2: M < N

Case 1: M ≥ N

M
− N
——
+ (r's comp. of N)
——
(1) Result

Discard then end carry ← end carry

ex: M = 82, N = 64
using 10's comp.
82 + (100 − 64) − 100
(there is 10's comp.)

(82 + r's comp. of 64) − 100
Instead of this Discard the carry

ex: M = 28, N = 19
using 10's comp.
28 + (10's comp. of 19)
⇒ 28 + (81)
⇒ (109) ⇒ (09) //
Discard

Case 2: (M<N) → will be no end carry

M
− N
——
+ (r's comp. of N)
——
Result → Correct ans.:

−(r's Comp. of Result)

ex: 20
−78
——
20
+ (10's comp. of 78)
——

20
+22
——
42 → −(10's comp of 42)

So, **−58** //

ex: 34
−98
——
−54

34
+ (10's comp. of 98)

34
+12
——
36 → −(10's comp of 36) ⇒ (−64)

- Conclusion

Subtraction of unsigned no. (M, N) r's comp.

① (M−N)+r^n

Target M−N

① M + (r's comp. of N)

⇒ M + (r^n − N)

Case 1:
end carry exist ⟷ M ≥ N
Correct ans. Just Discard carry you have your correct ans

Case 2:
end carry Doesn't exist ⟷ M < N
Correct ans. −(r's comp. of result)

Practice Question

ex: 239 − 82

⇒ M − N

using (10's comp.)

$$\begin{array}{r} 239 \\ -\,82 \\ \hline \end{array} \Rightarrow \begin{array}{r} 239 \\ -082 \\ \hline \end{array} \Rightarrow \begin{array}{r} +239 \\ 918 \\ \hline (1)157 \end{array}$$

ans = 157

Discard as M ≥ N

Binary Subtraction

unsigned no. ⇒ M: 1111010
N: 1101011

M−N using 2's comp:

$$\begin{array}{r} 1111010 \\ -\,1101011 \\ \hline \end{array} \Rightarrow \begin{array}{r} 1111010 \\ +0010101 \\ \hline (1)00011111 \end{array}$$

Discard end carry (M ≥ N)

N-M using 2's comp.

1101011 N
- 1111010 M
⇒ 2's comp → 1101011 / 0000110 / 1110001

no end carry (N<M)

So, −(2's compln of results)

−(0001111)

Subtraction of unsigned no. using (r-1)'s compln

ex: M-N using 9's compln

84-24 = 82+(100−24)−100

= 82+(99−24)+1−100

= [82 + (9's compln of 24)] +1−100

in place of −100 just remove end carry

82 − 24 ⇒ 82 + 75 = 157

end carry → +1 → 58 correct ans.

M-N using (r-1)'s compln

M − N ⇒ M + (r-1)'s comp. of N = Result

case 1: if M≥N ⟷ End carry

Correct ans.:

Discard carry, Add +1 to the Result

(or)

Add the end carry to the result

case 2: if M<N ⟷ no end carry

Correct ans. ⇒ −((r-1)'s compln of Result).

using 9's compln

ex. 24 − 82 ⇒ 24 + 17 = 41

no end carry (M<N)

Correct ans. −(9's compln of 41) ⇒ −(58)

Proof:

24 − 82 ⇒ 24 + (100 − 82) − 100

⇒ 24 + (99 − 82) + 1 − 100

Result − 99 ⇒ −(99 − Result)

Correct ans.

⇒ −(9's comp. of Result).

Binary Subtraction using 1's comp.

ex. M = 1011, N = 100

⇒ 1011 − 0100 = 1011 + 1011 = 0110

M≥N end carry

Correct ans. 0110 → 0111

M = 100, N = 1101

0100 − 1011 ⇒ 0100 + 0010 = 0110

no end carry (M<N)

Correct ans. −(1's comp. of 0110)

⇒ −1001

⇒ −9

- r's, (r-1)'s complement for Radix Point

r = 10

ex: 39.502 $\xrightarrow{\text{9's compl}^n}$ 60.497 (subtract every digit by 9)

10's compl^n → 60.498

r = 2

10100.1011 $\xrightarrow{\text{1's compl}^n}$ 01011.0100

2's compl^n → 01011.0101

r's compl^n → $(r^n - N) = m \xrightarrow{\text{r's compl}^n} r^n - r^n + N = N$

N

(r-1)'s compl^n → $(r^n - 1) - N \xrightarrow{\text{(r-1)'s compl}^n} (r^n - 1) - r^n + 1 + N = N$

Representation of signed Binary no.

Decimal (+)99
(−)99

Computer → binary?

(-4) → | ¬ | 1 | 0 | 0 |

??

Signed Binary no.

- Sign-magnitude System
- Sign-complement System
  - 1's comp. System
  - 2's comp. System.

Sign magnitude System

+4 → 0100

−4 → 1100

'+' denote 0 &

'−' denote 1.

ex: 9 in 4bits?

(in SM system).

X ① 001 ⇒ −1

Signed no. Actual no.

100 → −0 } both are same but diff. represen (ambiguous)

000 → +0 }

In SM System

111 → 7 X

Signed no. → −3

1's Complement no. System

for +ve no.

$N \geq 0 \Rightarrow$ SM system = 1's comp. System

ex: (6) → 0 1 1 0 (SM System) | 0 1 1 0 (1st comp. System)

For −ve no.

N<0 then

in 1's comp. system, N is stored as (1's comp. of N).

operatn

ex: 5bits ← 1's comp. System

+9 → 0 1 0 0 1

−9 → 1 0 1 1 0

then +9 → 0 1 0 0 1 ↗ 1's compln

mistake's ex:

−8 in 4bit (1's compl. System).

0 1 1 1 X

+8: 1000 —(1st compln)→ 0111 X

ble +8 can't represent in 4bit we need 5bit

So, +8: 01000 ↘ 1's comp.

10111

So, −8 in 4bit is not sufficient (not possible).

N<0, in 1's Comp. Sys.

Find (+N) → 1's Comp. operation → N is in System.

ex: MSB 1001 ⇒ −(0110)

1001 → (−ve no.) → 1's Comp. → (−6)

Q. In Computer;

m | 1 | 0 | 0 | 1 |

Dependon yours Interpretation

then $(m)_{10}$?

1001

- if unsigned → (9)
- if signed.
  - if SM System → (−1)
  - if 1's Comp. System → (−6)

2's Complement no. System

- for (+ve) no. all Systemare same N≥0

SM System ≡ 1's comp. System ≡ 2's comp. System

- for N<0
  N will be Stored as 2's comp. of +N

ex: N: 8 → 8bit → 0 | 1 | 0 | 0 | 0 |

N: −5 → 1 | 1 | 0 | 1 | 1 |

(+5) = 00101 → 2's Comp. → 11011

ex: 1001

till's that it's a −ve no.

convert the whole no. in 2's comp.

⇒ −(0111)

⇒ −7.

In all System

+ve no. → 0 (MSB)

-ve no. → 1 (MSB)

ex. Signed no. 1 0 1 1

-ve

In S.M → -3

In System → -(0100) = -4

In 2's Comp. → -(0101) = -5

ex. Unsigned no. 1 0 1 1

In S.M = 11

1's comp = 11

2's comp = 11

these are for signed no.

Range of Signed Binary no. in various Representations

using 5-bit (in 2's system). ← for signed no.

(10) → +10 : 0 1 0 1 0

(-10) → +10 : 1 0 1 1 0

01010 —2's comp.→ 10110.

2's compln no. System is a weighted no. System

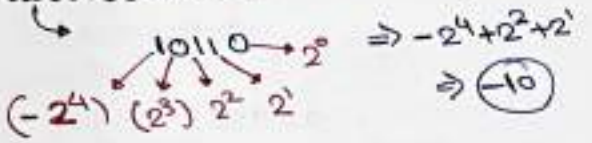
ex. binary no. Sys. → 2^3 2^2 2^1 2^0 : 1 1 0 1

Decimal no. Sys. → 902 : 10^2 10^1 10^0

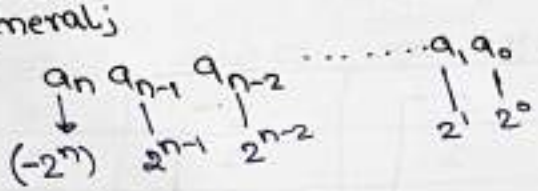
weighted no. System

with every position, some weight is Associated.

2's comp. system is a weight N.S.



In general;



ex: In 2's system

ex: 101101 —way1→ $-(010011) = -19$

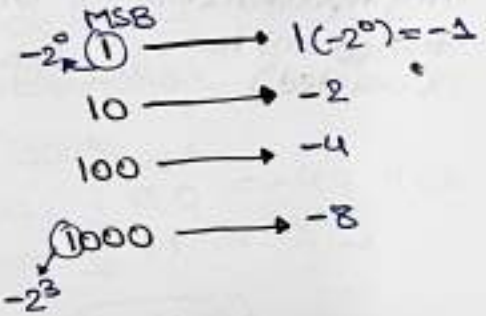
-ve

(-2^5)

way2 → $-2^5 + 13 \Rightarrow -32+13$

$\Rightarrow -19$

In 2's comp. system.



ex:
101 → -3
1101 → -3
11101 → -3
111101 → -3

doesn't matter ignore

ex: 0111111 → $2^6 - 1$

+ve

$\Rightarrow 63$

1111111 → (-1)

Range of each no. System for n-bit

① 3bit – 8 combination.

	unsigned	Signed mag.	1's comp.	2's comp.
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

they represent only 7 different no. (coasted)

Unique Representn

(1)101 → 1's comp. → −(010) = −2

-ve → 2's comp. → −(011) = −3

Non-weighted no. System
- Sign-magnitude
- 1's comp. system

(1)01011 → insm → (-11)

→ in 1's sys. → −(010100) = (-20)

- Range of sign magnitude Representation

using 3bit → **-3 to +3**

min: 1 1 1 ⇒ -3
max: 0 1 1 ⇒ +3

} 7 no. & one combinatn wasted.

Sign | magnitude

using nbit.

min: 1 1111...1 = $-(2^{n-1}-1)$

max: 0 111....1 ⇒ $2^{n-1}-1$
(n-1)'s

range

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

- Range of 1's complement Representation

using 3bit. **-3 to +3**

min: 1 0 0 = -3
max: 0 1 1 = +3

} 7 no. & one combinatn is wasted.

using nbit ↓ same range

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

- Range of 2's complement Representation (weighted NS)

For 3bit's. -2^2

min: 1 0 0 = -4
max: 0 1 1 = +3

} 8 no.

-4 to +3

using 'n' bit's

min: 1 00.... 0 = -2^{n-1}
(-2^{n-1})

max: 0 11.... 1 = $+(2^{n-1}-1)$

range

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

• Addition of 2 no. in the signed magnitude system.

$(+M) + (+N) : +(M+N)$

$(-M) + (-N) : -(M+N)$

$(+M) + (-N) : +(M-N)$ $N < M$

$(+M) + (-N) - (N-M)$ $N > M$

ex: $(+11) + (-13) = -(13-11)$

ex: $(-11) + (-17) = -(11+17)$

ex: $(+11) + (+17) = +(11+17)$

ex: $(-11) + (+17) = +(17-11)$

In sign mag;

M = 1011 −3

N = 0111 +7

① comparison of sign bit

② comparison of magnitude & it's sign will be used in result.

1011 + 0111

magnitude, bigger magnitude.

1111 } adding
−011 } both no.
100

Result: 0100 ⇒ 4

• Subtraction of signed no. in S.M system.

ex: $A - B \Rightarrow A + (-B)$

ex: $(+10) - (+5)$ ex $(-10) - (-5)$

$\Rightarrow (+10) + (-5)$ $\Rightarrow (-10) + (+5)$

In sign-magnitude system:

for addition of 2 signed no.

cost: ① comparison of sign bits

② If sign bit's Difference then comparison of magnitude

③ If sign are same then we do Addition → Adder Circuit.
else if Sign are Different then we do subtraction of magnitudes.
→ Subtractor circuit.

Cost { Comparison O(n)time
H/w { for Addition
for subtraction

Too much.

- Addition of 2 no in the 2's complement. System.

① All Signed no. in 2's Compln Representn

Signed int a=-6

1	0	1	0

unsigned int a=10

1	0	1	0

Signed int a=10 Invalid

using 4bit's (-8 → +7)

ex. (+5) + (-6) = -1
↓ ↓
00101 + 11010

```
  00101
  11010
 -------
  11111  → -1
```

② Signed Arthmatic in 2's compln

I/p → in 2's compln
O/p → Represent

eg: 5bit I/p
(+5) + (-4)
↓ ↓
00101 + 11100

```
  00101
  11100
 -------
 (1)00001
```

Discarded.

ex. (-5) + (-4) = -9
↓ ↓
11011 + 11100

```
  11011
  11100
 -------
 (1)10111  → (-9)
```

Discard. It's in 2's comp.

2's complement system for signed no.

$(\pm M) + (\pm N) = R$ → also in 2's comp. Rep.

($\pm M$) in 2's Compln, ($\pm N$) in 2's Compln

- Addition of 2 no. in the 1's Compln System

I/p, O/p — 1's comp. Representation.

1's compln Addition ⇒ we add end carry to result to get final result.

ex. (+5) + (+4) = 9

00101 + 00100

```
 00101
 00100
-------
 01001  → 9
```

ex. (+5) + (-4)

00101 + 11011

```
 00101
 11011
-------
(1)00000   (1 → Discard.)
    +1
-------
 00001
```

ex. (-5) + (+4)

11010 + 00100

```
 11010
 00100
-------
 11110  → -(1)
```

all are in 1's compln

ex. (-5) + (-4)

11010 + 11011

```
 11010
 11011
-------
(1)10101   (1 → Discard)
    +1
-------
 10110  → -9
```

is in 1's Compln

- 2's comp. system is better than 1's compl system for signed no.

① no. extra Addition of end carry

② no two Rep. of 0.

③ Range is more.

ex: 5bit : (+4) + (+9)

00100 + 01001

in 1's compln

00100
01001
01101 → 13

ex. (+4) + (-9) =

00100 → 10110

00100
10110
(1)01010 ⇒ (-5)

ex: (-4)+(+9)

11011 + 01001

11011
+ 01001
(1)00100 ← not the final result.
+1
00101 → (5)

ex: (-4) + (-9)

11011 + 10110

11011
10110
(1)10001
+1
10010 → (-13)

final result of addition

ex: (-11) + (-20) in 1's compln

+11: 00001011 & +20: 00010100

-11: 11110100 -20: 11101011

⇒ 11110100
+ 11101011
(1)11011111
+1
11100000 → (-31)

ex: (-11) + (-20) in 2's compln

+11: 00001011 · +20: 00010100

-11: 11110101 -20: 11101100

11110101
11101100
(1)11100001 → (-31)

discard

- 1's compln is more costly than 2's compln as we need to add the end carry at the end.
  ↓
  extra Addition

- converting to 1's compln & 2's compln take same time.
- So 2's compln is the by default system (in signed no

• Subtraction of two no. in the 2's complement System.

A → minuend, B → Subtrahend

ex: 5bit

+4: 00100

+9: 01001

(+4) − (+9)

00100 − 01001

00100 + (−01001)

2's compln of these

00100 + 10111

```
 00100
 10111
-----
 11011  → (-5)
```

ex: 5bit: −4: 11100 → A

−9: 10111 → B

A − B

(−4) − (−9)

A + 2's compln of −9

```
 11100
 01001
-----
(1)00101
```

Discard carry

← 11100 + 01001

A, B: signed no. in 2's compln Rep.

A+B

Discard end carry

Read the o/p in 2's compln

A − B

A + (2's comp. of B)

Discard end carry.

A, B; Signed no. in 1's compln Rep.

A+B

Add the end carry to the o/p & Read the o/p in 1's comp. Rep.

A − B

A + (1's compln of B)

Add end carry.

Overflow

nbit + nbits
Data Data
⇓
Sometime correct Result
needs n+1 bit

unsigned no. ⇒ 0 to 15

4bit Register

R_1: 4
R_2: 6
] 4+6 ⇒ no overflow

R_1: 10
R_2: 13
] overflow
10+13=23.
Can't put in 4bit)

overflow : Depend on
- Representation of Data
  - unsigned
  - Signed
    - Signed mag.
    - 1's compln
    - 2's compln
- "operation"
  - addition,
  - Subtraction,
  - multiplication.
  - .....

* gate syllabus for overflow operation: Addition.

- unsigned no. (4bit).

Range 0 to 15

R_1=10
R_2=2
] overflow (OF)

operation	OF
add	X
multiplication	✓
exponentiation	✓

Overflow
Definition:

a,b ⇒ n-bit's ; some operation #

a#b = final correct Ans.
Can't be Represented in n-bit.

ex: 4bit R_1=4, R_2=5. ; Addition

unsigned

Range: 0 to 15

R_1+R_2: 4+5: 9
⇒ no overflow

2's compln

Range: −8 to +7

R_1+R_2: 4+5: 9
⇒ overflow

Overflow

In case of Unsigned Addition

the final carray (carry out of MSB) ⟷ Overflow

1bit Range : 0 to 1 Carry

0	0	1	1
+0	+1	+0	+1
0	1	1	10 = 2
no OF	no OF	no OF	OF, need 2-bit

ex: 2bit → Range: 0 to 3

2+2: OF ✓

2+1: OF ✗

~~(-3)+(-4) non sense~~

~~(-2)+(+2) non sense~~

10	10
+10	01
00	11
OF	no OF

Result: unsigned no.:

$$\begin{array}{r} A: a_3\ a_2\ a_1\ a_0 \\ (+)\ B: b_3\ b_2\ b_1\ b_0 \\ \hline r_4\ \ r_3\ r_2\ r_1\ r_0 \end{array}$$

r_4: carry out of MSB

OF occurs iff $(r_4 = 1)$

In case of Signed no. Addition

OF can never occur if no. have opposite sign.

(1) if they have same sign then it might overflow.

(1) Signed magnitude Addition.

ex: 4bit Range: -7 to +7

ex: (+5)+(+3) OF

ex: (-6)+(-2) OF

ex: (+4)+(-3) OF can never occur

Overflow detection condition for SM system

	A+B	
A : S_A	M_A	} separately
+ B : S_B	M_B	
	Separately	

① if $S_A \neq S_B$ then no overflow

② if $S_A = S_B$ then

magnitude addition Decides overflow] overflow occurs if there is carry out of MSB of magnitude

(is always unsigned)

note

Overflow

Cout ⊕ Cin = 1 → for 2's compln addition.

ex: 4bit → Range -8 to +7

(-6) + (-3) → OF

1 1 1 0
1 1 0 1 1

Same Sign

OF
1 1 1 0
+ 0 1 1
0 0 1

SM	2's comp.
OF	no OF

-4, -4

In case of 2's compln Addition

ex: 4bit Range : -8 to +7

+4, -5 : never OF

+4, +5 : OF

-4, -4 : OF

-8, +1 : never OF

Condition for 1's & 2's compln

(+A)
(-B)
↓
no OF

(+A)
+ (+B)
- Result
OF

-A
+ -B
+Result
OF

+A
+B
+Result

-A
-B
- Result

→ no OF

2's compl^n Addition:

A: $a_3 a_2 a_1 a_0$
B: $b_3 b_2 b_1 b_0$
$r_3\ r_2\ r_1\ r_0$ — final sum we get

OF occurs (iff)

$\begin{bmatrix} a_3=0 \\ b_3=0 \\ r_3=1 \end{bmatrix}$ or $\begin{bmatrix} a_3=1 \\ b_3=1 \\ r_3=0 \end{bmatrix}$

$[\bar{a}_3 \bar{b}_3 r_3 + a_3 b_3 \bar{r}_3 = 1]$

ex: $(-5) + (-6) = (-11)$ OF

In 2's compl^n $[1011 + 1010]$

```
  1011
  1010
(1) 0101
```

(1) → Discard; 0101 → final result

$\left.\begin{matrix} a_n=1 \\ b_n=1 \\ r_n=0 \end{matrix}\right]$ OF

In case of 1's compl^n addition

condition: Similarly to 2's complement

ex:
$+5 \rightarrow 0101$
$-6 \rightarrow 1001$
$-1 \quad 1110$
no OF

ex: $-3 \rightarrow 1100$
$-4 \rightarrow 1011$
(1) 0111
+1
1000 — final result
no OF

$\left.\begin{matrix} a_n=1 \\ b_n=1 \\ r_n=1 \end{matrix}\right]$ no OF

when $a_n b_n \bar{r}_n = 1$?

Cout ← cin
$A = a_n\, a_{n-1} \cdots a_1$
$B = b_n\, b_{n-1} \cdots b_1$

→ iff $c_{in}=0$ and $C_{out}=1$

Cout=1 ← $c_{in}=1$ ✗ else the r_n will be 1.

```
 1 ....
 1 ....
 0
```

So, $c_{in}=0$ & $C_{out}=1$

C_{in}: in-carry into MSB

C_{out}: out carry out of MSB

When $\bar{a}_n \bar{b}_n r_n = 1$? iff $c_{in}=1$ and $c_{out}=0$

$c_{out}=0$ $c_{in}=1$

0 . . . . .
0 . . . . .
1

So, in 2's compl^n Addition only not for 1's compl^n.

OF occur (iff) $c_{in} \oplus c_{out} = 1$

Counter example why it will not work for 1's compl^r

ex. $c_{out}=1$ $c_{in}=0$

1 1 1 1
+ 1 0 0 0
0 1 1 1

In 2's compl^n: $a_n=1$, $b_n=1$, $r_n=0$] OF

$c_{out} \oplus c_{in} = 1 \oplus 0 \Rightarrow 1$] OF

$c_{out}=1$ $c_{in}=0$

1 1 1 1
1 0 0 0
0 1 1 1
+1
1 0 0 0
r_n

$a_n=1$, $b_n=1$, $r_n=1$] no OF

Conclusion

① unsigned Addition : OF (iff) Carry out of MSB

② Signed no. Addition: (All System)
if Sign Different: no OF
If Sign Same: may or may not OF

③ Sign magnitude Addition.
(i) Check Sign: Diff. ⇒ no OF
Same ⇒ magnitude Addition.

OF occur iff carry out of MSB of Magnitude.

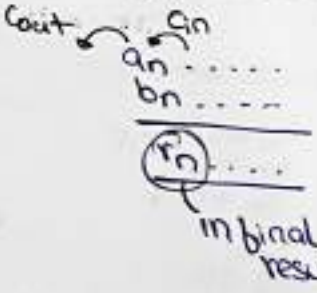
④ 1's compl^n ; 2's compl^n :

OF happens

(999)

$\bar{a}_n \bar{b}_n r_n + a_n b_n \bar{r}_n = 1$

⑤ only for 2s comp. Addition.

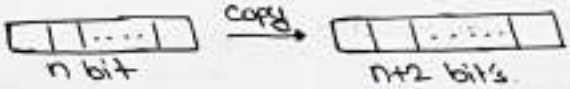


OF happens

(100)

$C_{in} \neq C_{out}$.

Sign Extension



Data Should not change

for unsigned no.

ex: 4bit → 5bit → 6bit

0101 → 00101 → 000101

5 5 5

ex: 4bit → 6bit

1101 → 001101

13 13

$a_n a_{n-1} \cdots a_1$ (n bit) → copy → $000\, a_n a_{n-1} \cdots a_1$ (n+3 bit's)

for signed no.

① Sign magnitude Rep.

4bit → 5bit → 6bit

0010 → 00010 → 000010

+2 +2 +2

1010 → 01010 → 10010 → 100010

-2 -10 -2 -2

ve 2 ve 2

a_n $a_{n-1} \cdots a_1$ → $a_n\, 000\, a_{n-1} \cdots a_1$

nbit magnitude Sign n+3bits magnitude

Sign.

② 1's Compl^n Rep. & 2's Compl^n Repl^n

4bit 5bit 6bit

0110 00110 000110

6

1011 11011 111011

-4 -4 -4

$a_n a_{n-1} \cdots a_1$ (n bit) $\xrightarrow{\text{Copy}}$ $a_n a_n a_n a_n a_{n-1} \cdots a_1$ (n+3 bit)

Binary code → are the pattern of 0's, 1's

- Discrete finite set of element

$\{a_1, a_2, \cdots, a_m\}$ (m element we have) → if we want to rep^n in binary code → min = n bit

Using n bit

we can rep^n 2^n element $\geq m \Rightarrow n \geq \log_2 m$

$$n = \lceil \log_2 m \rceil$$

ex: $\{a, b, c, d, e\}$ (m=5) → binary code → min. bit : 3 (n)

$2^n \geq m$ → $2^n \geq 5$ → $n \geq \log_2 5 \Rightarrow n \geq \lceil 2.\cdots \rceil$

$n = 3$ ✓

- you can have multiple Binary code for a particular element

ex: Decimal Digits

m=10

↓ Binary code

max. bit = 4.

Both are Binary code

	Binary no.	Gp code (2nd Binary no.)
0	0000	0000 0000
1	0001	0000 0010
2	0010	0000 0100
3	0011	0000 1000
⋮	⋮	⋮

Can prefer these code over the other b/c of simplicity

ASCII value.

65: A — 90: Z

97: a — 122: z

- why study so many different type of of Binary code when we already have Binary no.?

Reason ②

- Computer work with Binary no.
- I/p & o/p device work with Decimal no.

Reason ③ Application / Specific use.

Reason.

①

one Reason is

- So how do we represent → no., letters, colors, etc.

Binary Code

1. "BCD" Code. for Decimal digit's

Binary Code — Decimal

Decimal Symbol	BCD digit
0	0000
1	0001
2	0010
3	0011
⋮	⋮
9	1001

Similiar to Binary No. But it's different.

Data	Binary no.
23	10111

BCD no.

0010 0011

2 3

note

- a BCD no. is not a binary no.

ex. $(00101000)_{BCD} \Rightarrow (28)_{10}$

Un used combination (Invalid BCD codes)

1010
1011
1100
1101
1110
1111

ex. (0011 1100)BCD Invalid.

0011 → 3

1100 → Invalid BCD code/ combinat?.

BCD → a binary code → for Decimal digit's {0, 1, 2, ... 9}

BCD code also know as 8421 code

ex: $(0010)_{BCD} = (2)_{10}$

8 4 2 1 } weights

for any valid BCD 4digit code,
we can assign weight 8421 →

Some other code for decimal

ex: 2421 code. → is a SCC (Self compl^n code)

Decimal	2421 Coded.
0	0000
1	0001
2	1000 / 0010] Both are correct
3	1001 / 0011
4	1010, 0100
5	1011

By default. (preferred). [Smaller Binary value]

How to decide which to prefer? (Till 4)
↓ lower Decimal value as a Binary no.

after 4, we have Self compl^n code.

9's (4, 5) — 1's compl^n

is BCD code ⇒ SCC? (no)

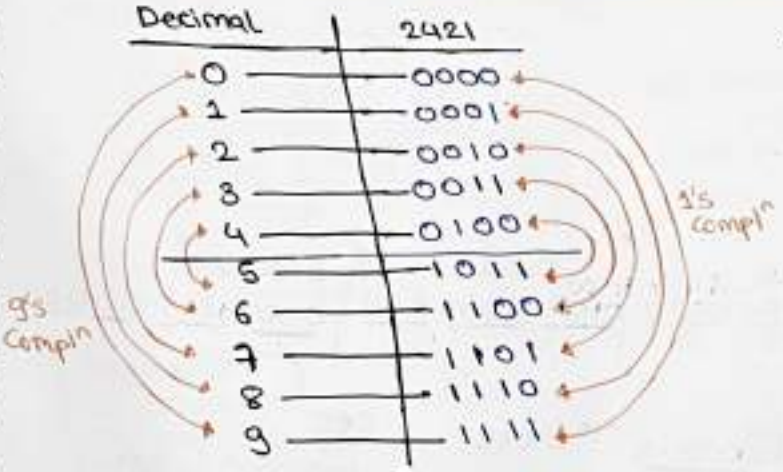
Decimal	BCD
2	0010
7	0111

9's (2, 7) — not 1's Comp.

Self complementary Code (SCC)

	code
2	0010
7	1101

9's (2, 7) — 1's.



Unused 2421 code.

0101
0110
0111
1000
1001
1010
} invalid

Excess-3 code

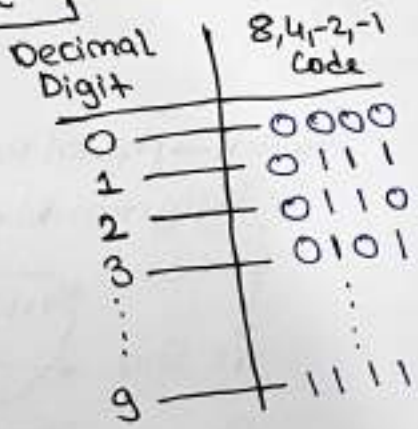
BCD code + +3

Unused Excess-3 code

0: 0000
1: 0001
2: 0010
13: 1101
14: 1110
15: 1111
} invalid.

• why Excess-3 when we already have BCD?
b/c of scc (self compln code)

8,4,-2,-1 code → is also SSC



For Decimal digit's:

Binary code / per decimal digit code
- BCD ≡ 8421 code
- 2421 code
- Excess-3 code
- 8,4,-2,-1 code

150 — BCD: 0001 0101 0000 (1 5 0) | Excess-3: 0100 1000 0011 | 2421 code: 0001 1011 0000

weighted code.
- BCD = 8,4,2,1
- 2421 code
- 8,4,-2,-1

Excess-3 is not weighted code → BCD+3

SCC
- Excess-3
- 2421
- 8,4,-2,-1

BCD is not SCC

note ..... given a weighted code abcd. is SCC then a+b+c+d=9 ⇒ SCC → a+b+c+d=9 ✗

gray code → Reflexive code.
many Applicatn ⇒ counter:

0 → 1 → 2 → 3 → 4 → 5 → 6 → 7

use switch to count

open = 0, close = 1

3 switch a b c

3 = 011 [a b c]
4 = 100 [a b c]

if we use Binary no. then all 3 switch/ bit need to be changed.

Switch → physical device → different delay

3 = 011 → 111 → 101
4 = 100

Temp. State. (Current State's) we don't want

Solution : only one bit change

n ⟷ n+1

only one bit / switch change → then use gray code

Decimal	Binary no.	Gray Code
0	00	00
1	01	01
2	10	11
3	11	10

2 bit diff.

Recursive construction of gray code

Decimal	Gray.
0	0
1	1
2	1
3	0

reverse these.

ex: Decimal n=14

binary encoding / binary no. = 1000 1

↳ gray code?

(G_n) gray code with 'n' bits

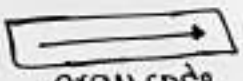
G_1 : 0, 1

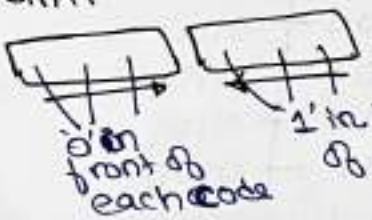
G_2 : 00, 01, 11, 10

$G_2 \longrightarrow G_3$

00, 01, 11, 10 → 000, 001, 011, 010, 110, 111, 101, 100

$G_n \longrightarrow G_{n+1}$





but there is time consuming method.

efficient Method

① Binary to gray code

ex: Decimal : m ⟶ gray code of m?

Binary of m: b_3 b_2 b_1 b_0

gray of m: b_3 $b_3 \oplus b_2$ $b_2 \oplus b_1$ $b_1 \oplus b_0$

ex: m=29 ⟶ gray code of 29?

binary no: 1 1 1 0 1

gray code: 1 0 0 1 1

Find gray code of 0 to 15?

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
⋮	⋮	⋮

② gray code ⟶ Binary code?

gray code ⟶ Binary

g_3 g_2 g_1 g_0

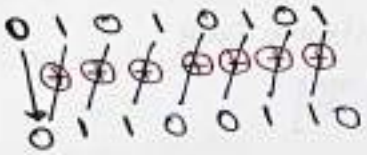
b_3 $b_3 \oplus g_2$ $b_2 \oplus g_1$ $b_1 \oplus g_0$

b_2 b_1

ex: gray code:

Binary

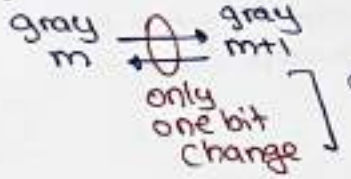
ex: Gray. :



binary

- properties of Gn (Gray code of length n)

1. Decimal



2. G_1: 0/0, 1/1

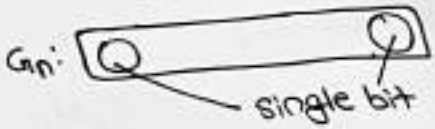
G_2: 00/0, 01/1, 11/3, 10/2

G_3: 000/0, 001/1, 011/3, 010/2, 110/6, 111/7, 101/5, 100/4

normal binary no.

① gray code are Reflexive code

② In Gn, gray code are cyclic code.



③ Gn is a permutation of the no. $0 \ldots\ldots 2^n - 1$

④ G_1 is embedded as first half of G_2

gray code finding another way.

Decimal no. = m.

↓

gray code = $m \oplus \lfloor m/2 \rfloor$ (⊕ : bitwise EXOR)

$m \oplus (m/2)$ — will give gray code of m.

ex. Decimal = 29

m : 29 : 11101

$\lfloor \frac{m}{2} \rfloor$: 14 : 01110

⊕ 10011

Introduction to fixed point & floating point Representation

Before floating point, we have seen.

- sign magnitude
- one's compln
- 2's compln

} integers no.

How do we encode the following.

- Real no. (ex: 3.14159)
- very large no. (ex: 6.02×10^{23})
- very small no. (ex: 6.626×10^{-34})
- Special no. (ex: ∞, NAN)

ex: 3.14159 what is the 1st thing come's in mind to represent this

[ ].[ ]

know as fixed point no.

Q. can we represent all integers accurately in float.

we have 32 bit for integers ≤ 32 bit for float.

assume 32bit for integers.

2^{32} integers $[0, \ldots\ldots, 2^{32}-1]$ slot

assume 32bit for float.

2^{32} float. no. (slot)

for every integer x ⟶ int x.

false $[x = (int)(float)(x)$

if we want to convert float ↓ integers.

so, we need to approx. the no. so that we can allocate it.

x = 32975.93 ↓ x = 32.975 in integers.

- for every integers we don't have float representation.

if both float & int has (say 32bits) same bit then float is not superior in any sense.

- Fixed point Representation.

ex: given
4bit for Integer
& 3bit fraction

3.5 ≡ 0011.100
↑ 1/2

ex. 3.20 → don't treat them as a whole no.
↑
- we can't even read/pronounce it as three point twenty ✗
- we will read this as three point two zero

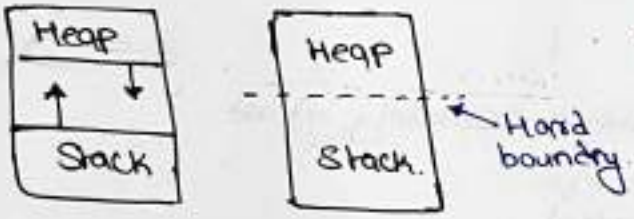
For ex: Integers part | fractional part

✗ option 1': 8bit | 8bit

✗ option 2': 12bit | 4bit

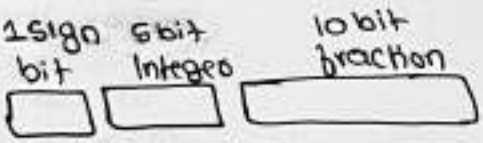
✗ option 3': 5bit | 11bit

b/c some integers need more bit in Integers part & Some need more bit in fractional part.

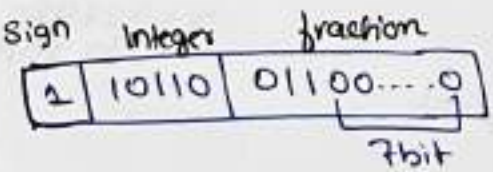


- just like in virtual memory we don't put hard boundary between heap & stack, we also don't want to put hard boundary b/w Integers & fraction.

ex: 16 bit fixed point system.



ex now represent -22.375



other way to Impl^n

-22.375
↓ ↓ unsigned

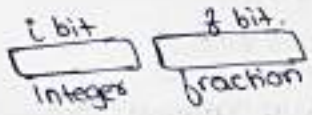
- Sign magnitude
- 2's comp^n
- 1's comp^n.

ex: CSE 2017

n-bit fixed point representation. Of unsigned real no. x use f bit for fraction.

let $i = n - f$

then range is.



ex:

$11111 \rightarrow 2^5 - 1 : 31$

$1111 \rightarrow 2^4 - 1 : 15$

$111\ldots1 = 2^k - 1$

$0.1 \rightarrow 2^{-1}$

$0.11 \rightarrow \frac{1}{2} + \frac{1}{2^2}$

ex 3.2 = 3 + 0.2

ex: 3.5 = 11.1 $\rightarrow 3 + 2^{-1}$

$3 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3}$

ex: 111.11 $\Rightarrow 2^2 + 2 + 2^0 + \frac{1}{2} + \frac{1}{2^2}$

$\Rightarrow 7 + 0.75$

$\Rightarrow 7.75$

method 1:

$0.\underbrace{111\ldots1}_{k \text{ time}} : \frac{1}{2^1} + \frac{1}{2^2} + \ldots + \frac{1}{2^k}$

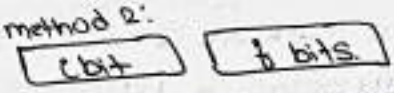
$\left[\frac{a(1-r^n)}{1-r}\right] \quad \frac{\frac{1}{2}\left(1 - \left(\frac{1}{2}\right)^k\right)}{1/2} = 1 - 2^{-k}$

$\underbrace{111\ldots1}_{k \text{ time}} \cdot \underbrace{111\ldots1}_{k \text{ time}}$

$2^k - 1 + 1 - 2^{-k}$

$\Rightarrow 2^k - 2^{-k}$

method 2:



$11.011 \Rightarrow 110.11 \times 2^{-1}$

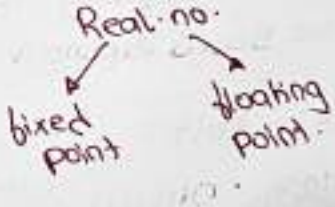
$\Rightarrow 1101.1 \times 2^{-2}$

$\Rightarrow 11011.0 \times 2^{-3}$

$\underbrace{111\ldots1}_{i+f} . 0 \times 2^{-f}$

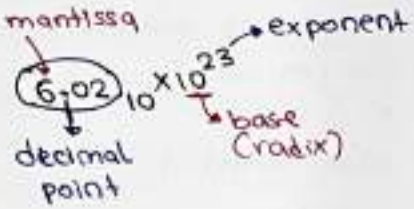
$\Rightarrow (2^{i+f} - 1) \times 2^{-f} \Rightarrow \boxed{2^i - 2^{-f}}$

- misconceptions
  - → fixed point representatn is only for integer no.
  - & floating point is for real no.



floating point Representation.

- Scientific notation (Decimal) ⇒



ex: $1879 = 0.01879 \times 10^5$

0.1879×10^4

1.879×10^3

1879×10^0

⋮

So many

- normalized form: exactly one digit (non-zero) to left of decimal point

ex: $101010_2 \Rightarrow 1.01010 \times 2^5$

coz it's floating no. ble. we are floating the dot to the extent where only one non zero is to the left of dot.

mantissa

ex: $1.01_2 \times 2^{-1}$ → exponent

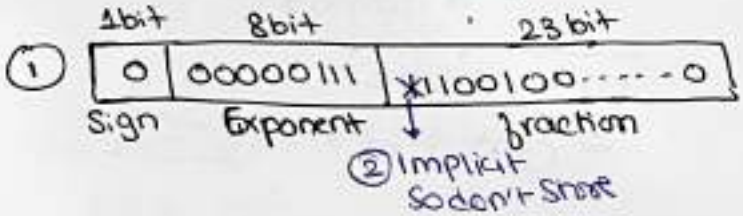
binary point

base

Floating-point Representation.

- 1st bit of the mantissa is always 1.

ex: $228_{10} = 11100100_2 = \underbrace{1}_{\text{is implied/implicit}} . \underbrace{11001}_{\text{Store the fraction only}} \times 2^7$



③ rather than storing the exponent we will Store Biased exponent.

Bias for 8bit:

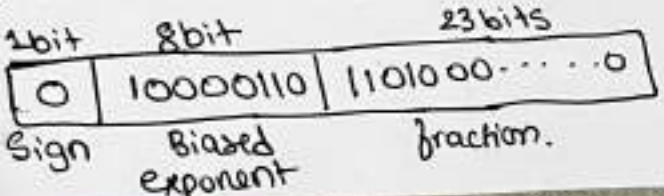
$127_{10}: 01111111_2$

- Biased exponent:

bias + exponent

$\Rightarrow 127 + 7 = 134 = 10000110_2$

So, IEEE 754 32bit of 228_{10}



• $228_{10} = 11100100_2 = 1.11001 \times 2^7$

1.11001×2^7

$\times * 2^8$

$7 + 127 = 134$

0 | 10000110 | 11001000...0

ex: -58.25

58.25 ⇒ 111010.01

⇒ $1.11010\,01 \times 2^5$

1bit	8bit	23bit
1	10000100	11010010000...0
sign	Exponent	fraction

127+5=133

ex: define own format. [represent in base 10]

Should mention whether 1 is implicit or not.

0	1	1	0	1	1	1	1	0

: 9 bit word.

Sign | sign of exponent | mantissa | exponent.

⇒ 1.1011×2^{-6}

⇒ $m = (1.1011)_2 = (1\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4})_{10} = (1.6875)_{10}$

mantissa

last 3bit, 110

$e = (110)_2 = (1\times2^2 + 1\times2^1 + 0\times2^0)_{10} = (6)_{10}$

no. in binary format.

$(1.1011)_2 \times 2^{(110)_2}$ ⇒ 1.6875×2^{-6}

⇒ 0.026367

ex: 6bit integer IEEE floating point

1	1	0	1	0	1

sign | exp. | frac.

⇒ $-1.01 \times 2^{(5)-3}$

-1.01×2^2

⇒ −101

⇒ (−5)

Bias = 3

not counting implicit bit

ex: 13 bit floating point no.

Bias: 15

01111

SEEEEE MMMMMMMM

Sign (1bit) | exponent (5bit) | mantaissa (7bit)

floating point representation of the fractn $-25/16$?

$\Rightarrow -25 \times 2^{-4}$

$\Rightarrow -1.1001 \times 2^{0}$

$\Rightarrow$ 1 | 01111 | 10010....0

Sign | 15+0 exponent | Mantissa.

ex: bit string 0x40800000,

0 | 10000000 | 10100.....0

Sign (1bit) | exponent (8bit) | Mantissa (23bit)

↓

we need to subtract as it's already exceeding 127

$\Rightarrow +1.101 \times 2^{128-127}$

$\Rightarrow 1.101 \times 2^{1} \Rightarrow 11.01$

$\Rightarrow 3.25_{10}$

- why the bias?

+127 (in single precision)

8 bit for exponent:

↳ largest +ve no. $\Rightarrow$ 01111111 $\Rightarrow$ 127

+128 (largest (in magnitude) (-ve) no. in 2's Complement)

+127 (largest (-ve) no. in 1's compln)

$(-1)^{S} \times 1.M \times 2^{e}$

• Bias value (• exponent bit)

In general Bias: $2^{e-1}-1$

Single precision 127

ϵ (-126 ..... 127)

Exp (1 ..... 254)

Double precision: 1023

ϵ (-1022 ..... 1023)

Exp (1 ..... 2046)

IEEE Standard notation

Exponent: 6 bit

Bias: 011111 ⇒ 31

5 on's

■ 3 case based on value of exponent

• Normalized
- when exp isn't all 0's or all 1's
- most common

• Denormalized
- when exp is all 0's
- Different interpretation of ϵ than normalized
- used for +0 & -0
- and other no. close to 0.

• Special
- when exp. is all 1's
- NAN, Infinities.

$(-1)^S \times 1.M \times 2^{E-127}$

only for normalized case. when ϵ is not all 1's or 0's.

• 2 Representation of zero.

0/1 | 00...0 (all zero) | 00.....0 (all zero)] reserved for zero

Sign | exponent | Mantissa

(Denormal)

• Representing small no.

0/1 | 00...0 (all zero) | not all 0's

Sign | exponent | Mantissa

⇓ denormalized form.

$[0.00\cdots0 \times 2^{-126}]$ format. (fix)

$0.f \times 2^{-126}$

• Representing infinity

0 | 11.....1 (all one's) | 00.....0 (all zero) = +∞

Sign | exponent | Mantissa

1 | 11.....1 (all one's) | 000.....0 (all zero) = -∞

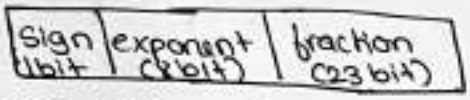
Sign | exponent | Mantissa

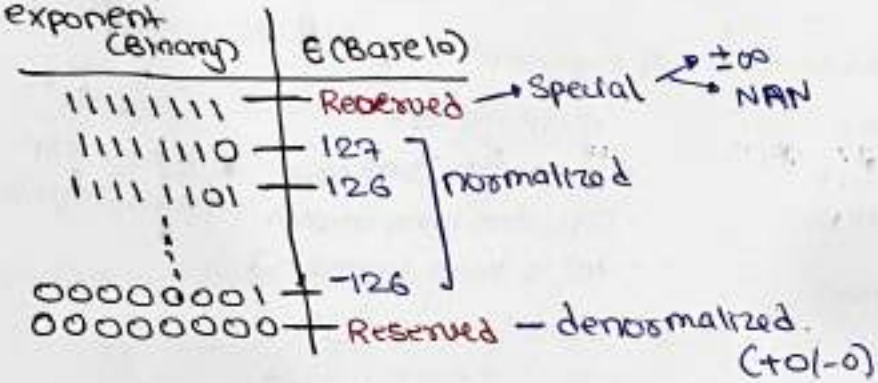
- Representing NAN (not a no.) ex: sqrt(x), $\frac{\infty}{\infty}$, $\infty + (-\infty)$.... etc.

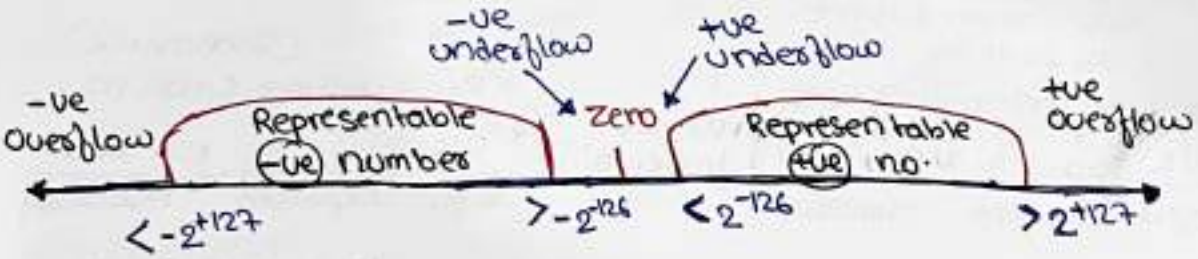
0/1 11.....1 any non-zero
Sign exponent fraction.

- Zero is not normalized.

Exponent







neither all 0's nor all 1's.

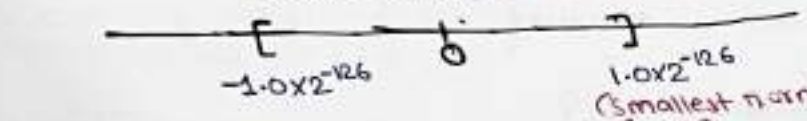
Smallest: 0000 0001 ⇒ 1 ⇒ Exp. -126

largest: 1111 1110 ⇒ 254 ⇒ Exp. 127

why infinity & NAN's are represented. not
bic rather than throwing error's or halting or an exception.

largest +ve denorm.

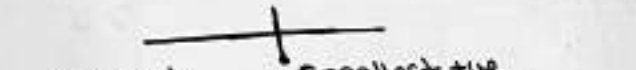
$0.111\ldots1 \times 2^{-126}$ — bic for denorms bic to match the boundary with norm's boundary



norm's | denorm's | norm's.

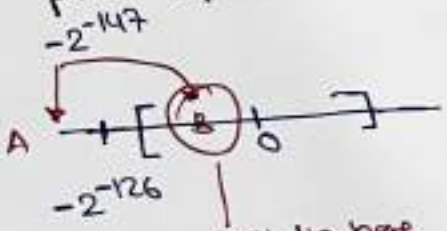
$$\frac{1}{2}+\frac{1}{2^2}+\frac{1}{2^3}+\ldots\ldots\infty = \frac{\frac{1}{2}}{1-1/2} = 1$$

- the exponent for denormalized float is (-126)



bic they want to keep them as close as possible to norm'.

ex. -2^{-147} in float point represent.



will lie here in denorm's form.

-2^{-147}

denorm.

$-0.\ldots\ldots\times2^{-126}$

ex. $0.1 \times 2^{-126} \Rightarrow 2^{-127}$

$0.01 \times 2^{-126} \Rightarrow 2^{-128}$.

$0.\underbrace{00\ldots\ldots01}_{20\ 0\text{'s}} \Rightarrow 2^{-147}$

$\Rightarrow 1\times2^{-21} \times 2^{-126}$

$\Rightarrow 2^{-147}$

$-2^{-149} \Rightarrow -0.00\underbrace{\ldots 01}_{20\ 0's} \times 2^{-126}$

1 | 00000000 | 20 0's then 1

Exponent | Mantissa

- why we need denormal form?

$1.\ldots\ldots \times 2^{e-127}$

e → 1 (Smallest where not all zero's)

→ 254 (Largest)

If no. is so small that Come's under range

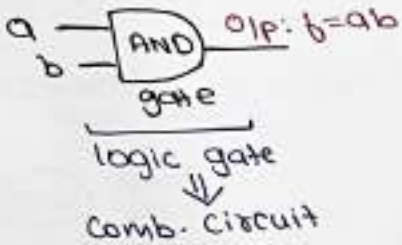
(By the way we might not even need denorms in our life b/c normalised already give too small range which is $> 2^{-126}$ or $< -2^{-126}$)

- formula used for denorms

denorm formula. = $-1^{sign} * 0.significand * 2^{-126}$

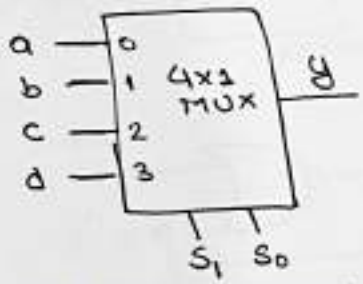
Combinational Circuits → Living in the present

Digital circuit whose o/p
Depend only on current i/p
(Current o/p doesn't depend on previous o/p / i/p



current

time	i/p	o/p
t=0	a=0 b=0	f=0
t=1	a=1 & b=0	f=0
t=2	a=1 & b=1	f=1



$y = f(S_1, S_0, a, b, c, d)$

S_1 S_0	y	time
0 1	b	t=0
1 1	d	t=1
1 0	c	t=10

doesn't depend

Sequential circuit

Counters O/P depends : f(current I/P, preo. O/P)

ex: X I/P → f ← O/P State

X	f	time
1	0	t=0
0	1	t=1
0	0	t=2

I/P same → O/P different

Current O/P $f = f(x, f_{prev})$

t=0 initial setup

X=1 current I/P

f=0 current O/P

feedback

t=ØX2

x=x0

f=ØX0 current O/P

1Ø preo. O/P

Current O/P $(y_t) = f(I_t, y_{(t-1)})$

current I/P; preo. O/P

O/P y at time = t

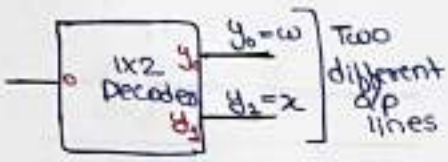
Sequential CKt → State = O/P

Current O/P (State) = f(current I/P ; past I/P ; past all O/P)

→ Depends on preo. history.

y_0 = O/P y at time t=0 ; y_1 = O/P y at time t=1

Combinational CKts



Seq. CKts

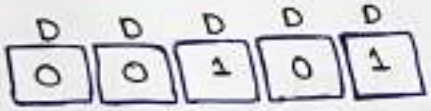
#o/p line = 1
= y

ex. In sequential circuit, how to remember previous o/p?
Some memory device / element

ex. what is the most Basic info. we need to store? single bit

Device D: can store one bit (0/1)

To store 00101 ⇒



Memory element/Storage

one bit storage device

⇓

we need to create

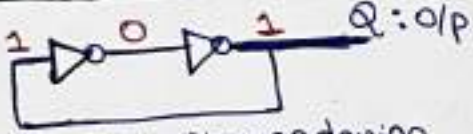
- what does "Remembering information" means?

Store (1) Retain until it's changed further

Device: D

x=1 → 1

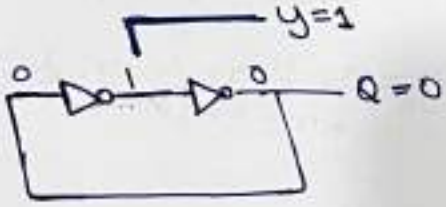
- Basic Memory element/ Device for one bit storage



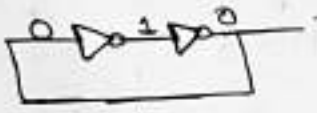
one bit Storage device

Storing 1, Retained

Can we say that this is a 2bit storage device? no



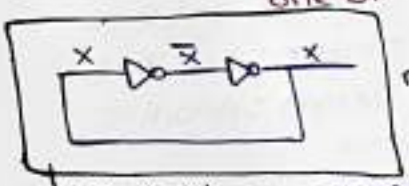
2 bit Storage device



fixed for indefinite time

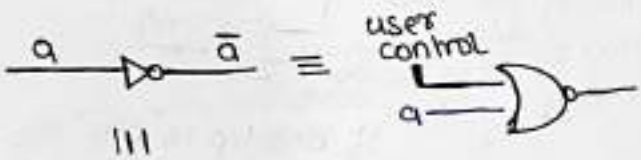
"Cross-coupled Inverters"

↳ we can Store & Retain one bit

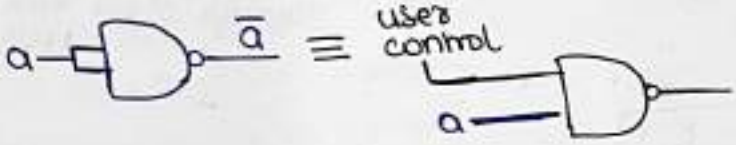


no user control

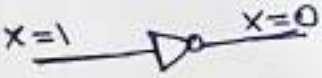
↳ Basic idea; to store one bit



|||

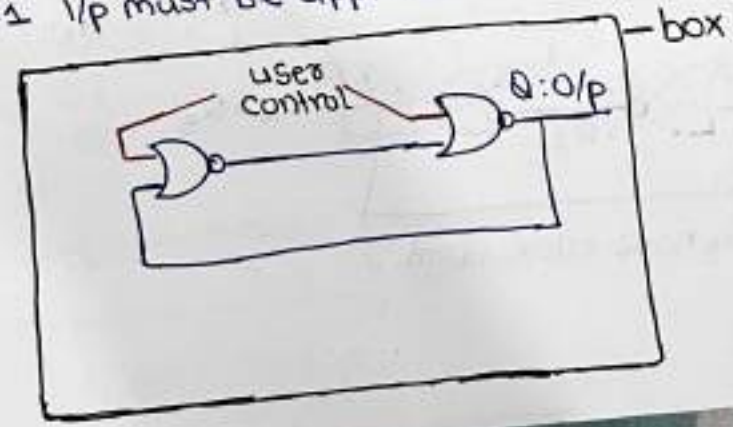


Store one bit:

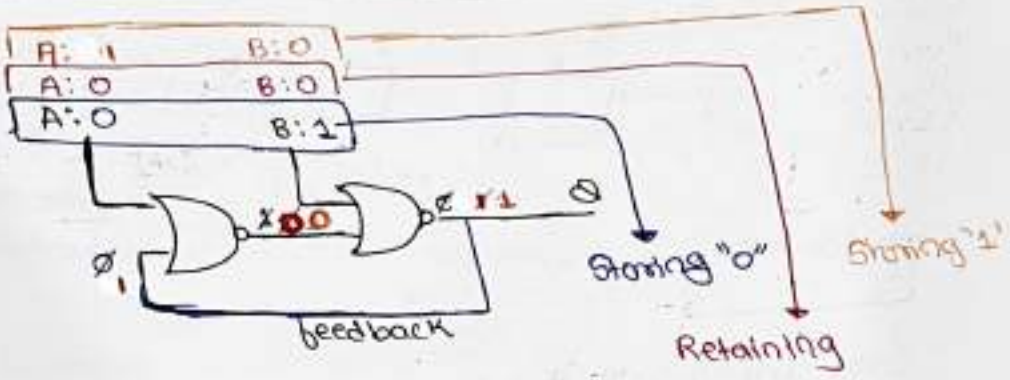
① 

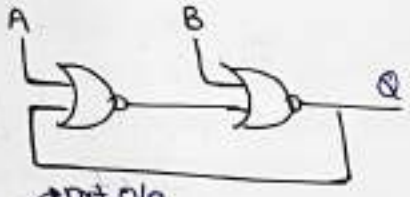
} can we call it store Device? no

To store 0, continuously x=1 I/p must be applied.



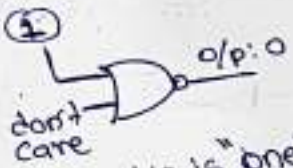
— box





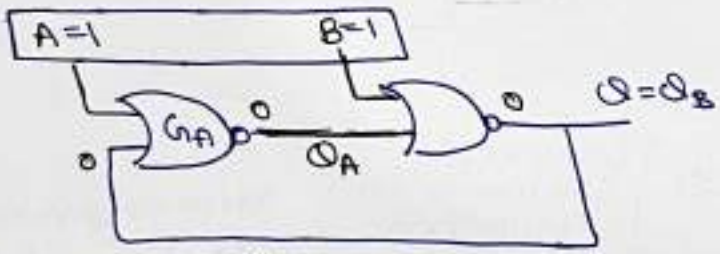
Store 1: A=1, B=0
Store 0: A=0, B=1
Retain: A=0, B=0

A	B	Q_n	time	
0	1	0	0	Storing '0'
0	0	Q=0	1	Retaining
1	0	1	2	Storing '1'
0	0	1	3	Retaining



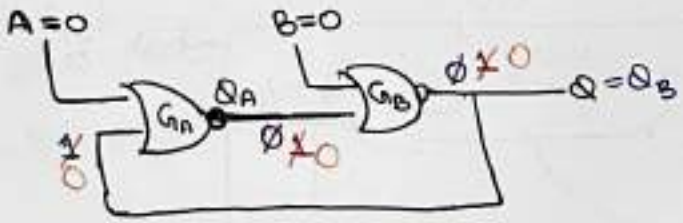
if one i/p is "one" the o/p will "0" for ~~XOR~~ NOR

what happens when A=1 & B=1 ?



o/p: Q=0 (Stable o/p)

After 1 Sec

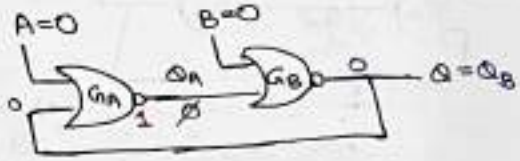


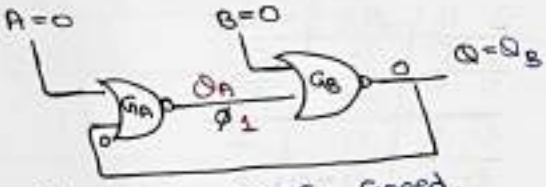
If G_A, G_B have equal speed.

If G_A, G_B have equal speed:

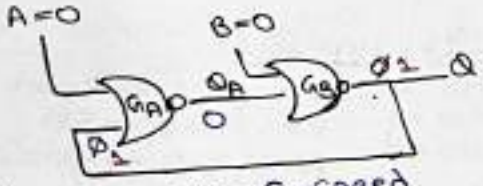
Q_A	Q_B
0	0
1	1
0	0
1	1

not stable O/P

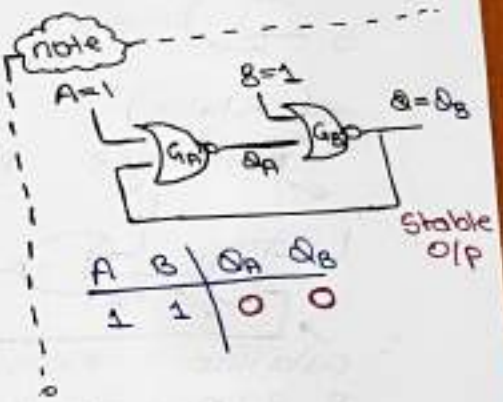




if G_A Speed > G_B Speed.



if G_B Speed > G_A Speed



A	B	Q_A	Q_B
1	1	0	0

But After A=B=1
if you Remove I/p signal
(If you want to Retain)
(If you make A=0, B=0)
then we Don't know what will happen.

Analogy

Dinner: milk + Curd (forbidden)

↓

next day: uncertain behaviour of stomach

So A=1, B=1 is forbidden

⇓

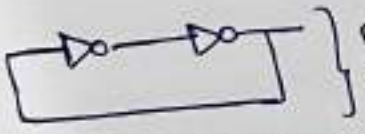
O/p = Stable

Q_A	Q_B
0	0

Problem:

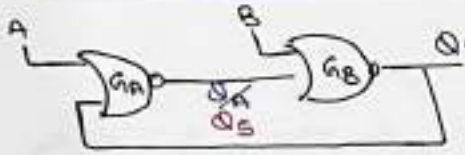
A=1, B=1 ⟹ A=0, B=0

uncertain O/p.



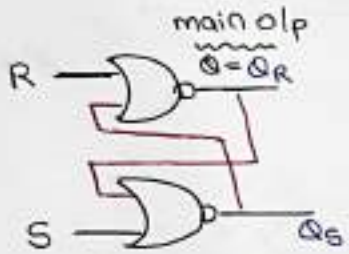
no user control

one bit storage device



A	B	Q_n → nxt o/p
0	1	0 — Reset
1	0	1 — Set
0	0	Q (Retain/unchange) current o/p
1	1	0 — forbidden

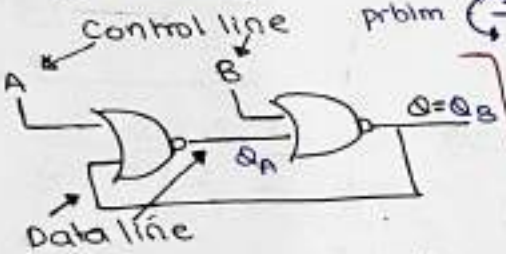
(Set-Reset) SR latch



(SR latch)

S	R	Q_n	time
1	0	1	0
0	0	Q=1	1
0	1	0	2
0	0	Q=0	3
1	1	0	4
0	0	Undefⁿ	5

Prblm ↺ problem



Analysis
A=0, B=1 ~ Reset
A=1, B=0 ~ Set
A=0, B=0 ~ Retain

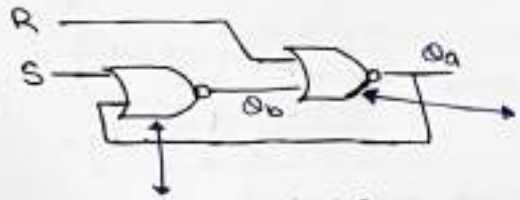
S	R	Q_{t+1} → nxt o/p
0	1	0
1	0	1
0	0	Q_t
1	1	0 ---→ But is forbidden.

ex: why S=1, R=1 is forbidden in SR latch?

Reason
after, S=1, R=1
o/p = 0
stable
if we make S=0 & R=0
then uncertain o/p

(a) uncertain o/p
(b) unstable op
(c) none

• Analyzing the Basic latch



S	Q_a	Q_b=NOR(S,Q_a)	
0	0	1	$\overline{Q_a}$
0	1	0	
1	0	0	0
1	1	0	

S	Q_b
0	$\overline{Q_a}$
1	0

R	Q_b	Q_a=NOR(R,Q_b)	
0	0	1	$\overline{Q_b}$
0	1	0	
1	0	0	0
1	1	0	

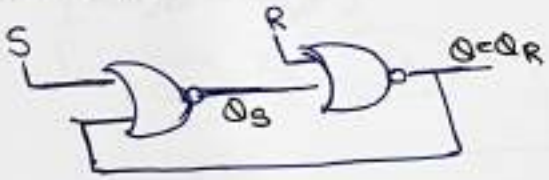
R	Q_a
0	$\overline{Q_b}$
1	0

S	R	$Q_a(t+1)$	$Q_b(t+1)$
0	0	$Q_a(t)$	$Q_b(t)$ → Latch
0	1	0	1 → Reset
1	0	1	0 → set
1	1	0	0 → undesirable

note

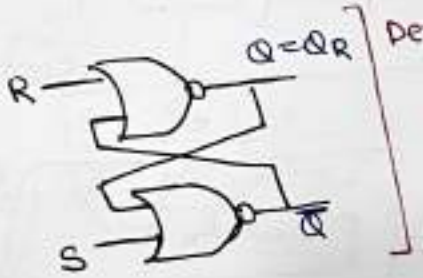
① for all Allowed I/p combination

$Q_S \neq Q_R$; $Q_S = \overline{Q_R}$



S	R	Q_S	$Q=Q_R$	
1	1	0	0	forbidden
				invalid I/p comb.
0	1	1	0	$Q_S=\overline{Q_R}$
1	0	0	1	
0	0	$\overline{Q_t}$	Q_t	

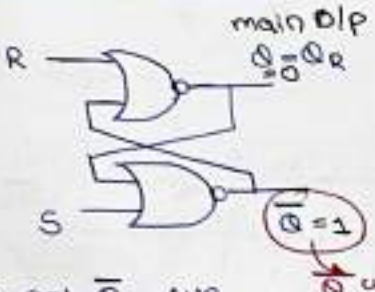
note if $Q_S=Q_R$ ⇓ if we try to Retain] uncertain behaviour.



] Desired.

SR latch ≡ SR latch using NOR

(2)



SR latch.

Storing 0 | are same
$Q = Q_R = 0$ |
State 0 |
O/p = 0 |

To get $\bar{Q}$, we Don't need inverter → $\bar{Q}$ we are getting for free

R	S	$Q = Q_R$	Q_S
1	1	0	0
0	0	uncertain / undefined / Oscillations (when both NOR gate have same speed)	

If both NOR gate have same speed:

R	S	$Q = Q_R$	Q_S
1	1	0	0
0	0	$\not\phi_1$ ...	$\not\phi_1$ ...

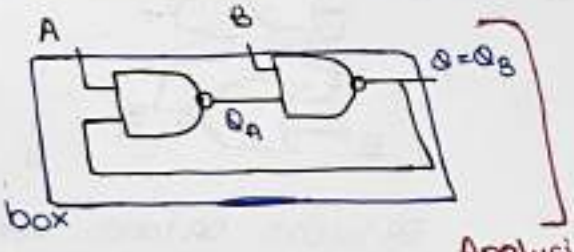
Oscillations

But in Reality; Both NOR gate have Different Speed

$Q_R = 0, Q_S = 1$ / $Q_R = 1, Q_S = 0$ } uncertain

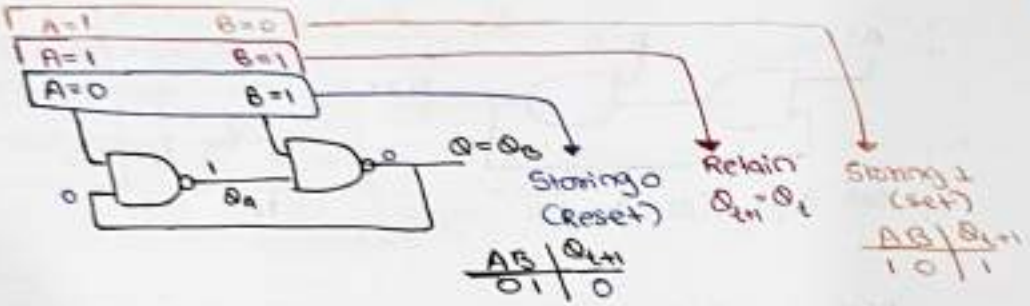
SR Latch with NAND Gate $\bar{S}\bar{R}$ latch

no user control

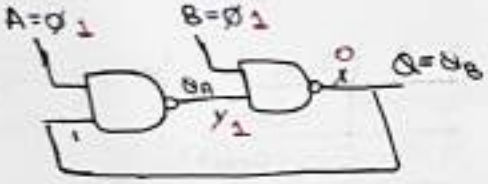


Analysis

if one i/p is '0' in NAND the o/p will be 1



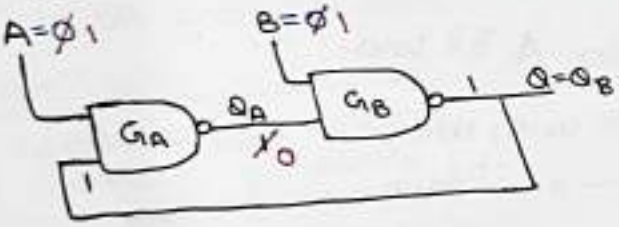
A	B	$Q=Q_B$	Q_A
0	1	0	1 → Reset
1	0	1	0 → Set
1	1	Retain	Retain → Retain
0	0	?	?



A	B	Q_A	Q_B
0	0	1	1
1	1	$Ø_1$	$Ø_1$
		$Ø_0$	$Ø_0$

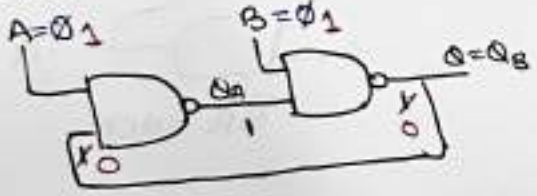
- if $G_A = G_B$ speed → oscillation

- if $G_A > G_B$ speed



A	B	Q_A	Q_B
0	0	1	1
1	1	0	1

- if $G_B > G_A$ speed



A	B	Q_A	Q_B
0	0	1	1
1	1	1	0

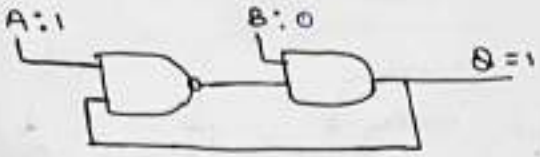
Problem:

A	B
0	0

$\Rightarrow$

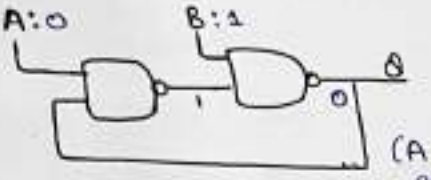
A	B
1	1

] then uncertain behaviour



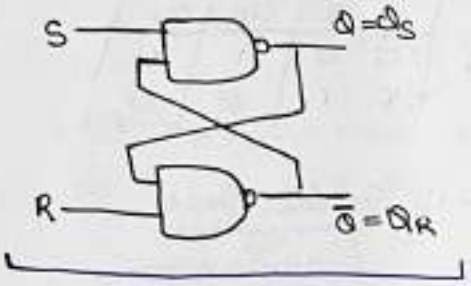
B ⇒ working for Set (time)
But Active low wise

• Set it? (Q=1)



(A is working as Reset but Active low wise)

• Reset it? (Q=0)



$\bar{S}\bar{R}$ latch

S	R	Q_{t+1}
0	1	1 (Set)
1	0	0 (Reset)
1	1	Q_t (Retain)
0	0	!

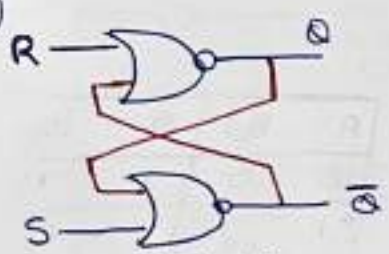
! → forbidden

• Comparison of SR latch & $\bar{S}\bar{R}$ latch

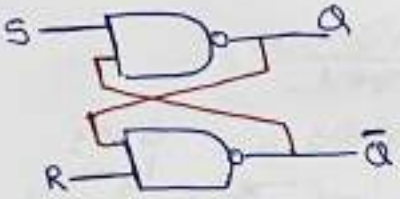
(1) SR latch = SR latch with NOR

$\bar{S}\bar{R}$ latch = —"— —"— NAND

(2)



SR latch



$\bar{S}\bar{R}$ latch

(3) SR latch is Active high

↳ (SR latch with NOR)

$\bar{S}\bar{R}$ latch is Active low

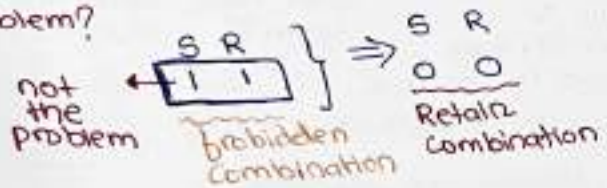
S	R	SR latch $Q=Q_R$	SR latch Q_S	$\bar{S}\bar{R}$ latch $Q=Q_S$	$\bar{S}\bar{R}$ latch Q_R
1	0	1 (set)	0	0 (Reset)	1
0	1	0 (Reset)	1	1 (Set)	0
0	0	Retain	Retain	1	1 (forbidden)
1	1	0	0	Retain	Retain

forbidden

note

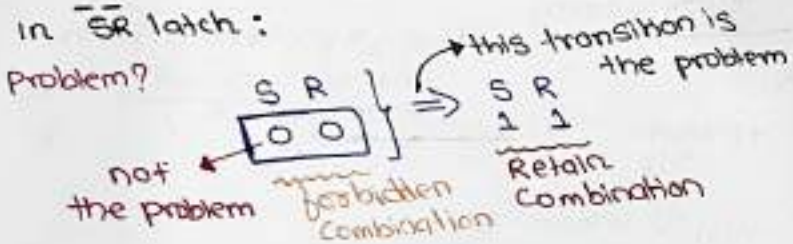
In SR latch

Problem?



In $\bar{S}\bar{R}$ latch:

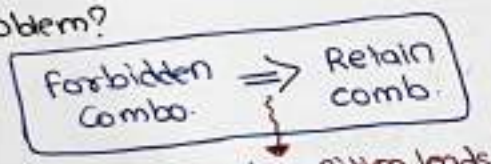
Problem?



note

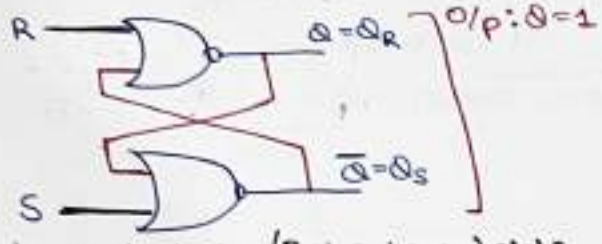
In SR latch, $\bar{S}\bar{R}$ latch

Problem?



this transition leads to uncertain behaviors.

- SR latch with Cross coupled NOR gate's



Block Diagram.

- if we want $\bar{Q}$, then one need to use extra inverters. Just take connection from $\bar{Q}$

- Characteristic / Behaviour table

S	R	Q_{t+1} → O/p Q at t+1 time
0	0	Q_t (Retain / unchange)
0	1	0 (Reset / Resetting the o/p state)
1	0	1 (set / setting the o/p state)
1	1	0 (forbidden)

why these is not a truth table b/c it should satisfy these condition

$Q_{t+1} = f(R, S, Q_t)$

Q_{t+1} → nxt O/p

Q_t → present O/p

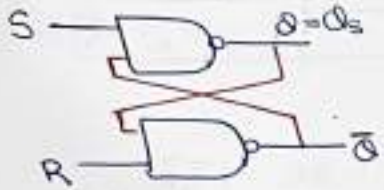
In truth table of Q_{t+1} ⇒ #Row = 8 Rows.

- Truth table.

R	S	Q_t	Q_{t+1}	
0	0	0	0	Retain/latch
0	0	1	1	
0	1	0	1	set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	0	forbidden
1	1	1	0	

$\bar{S}\bar{R}$ latch (SR latch with cross-coupled NAND gate)

Active low latch



- characteristic table

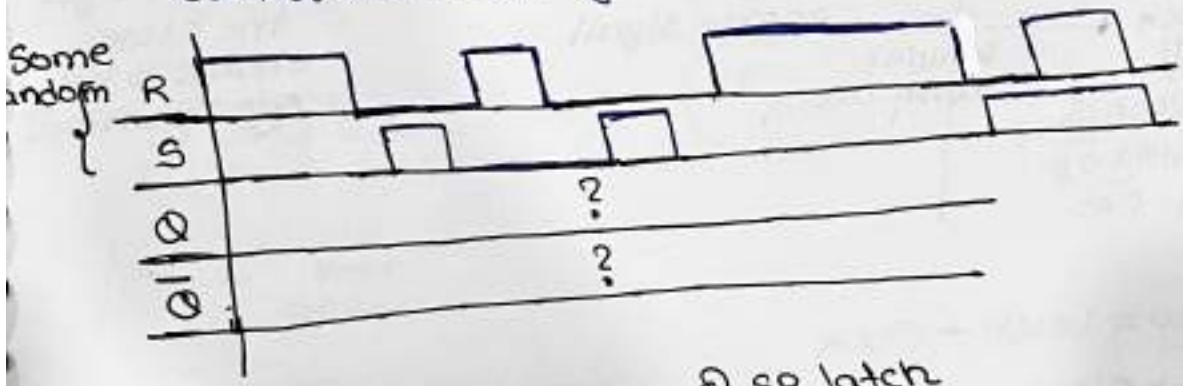
S	R	Q_{t+1}
0	0	1 (forbidden)
0	1	1 (set)
1	0	0 (Reset)
1	1	Q_t (Retain)

- Truth table

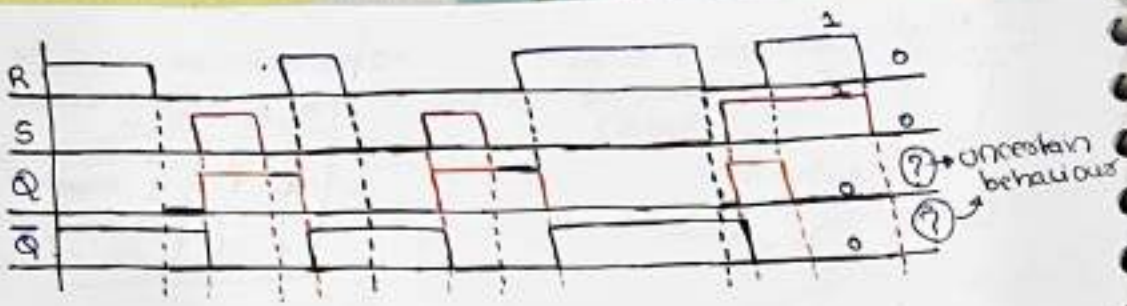
R	S	Q_t	Q_{t+1}	
0	0	0	1	forbidden
0	0	1	1	forbidden
0	1	0	0	Reset
0	1	1	0	Reset
1	0	0	1	set
1	0	1	1	set
1	1	0	0	latch/Retain/ memory/ unchange
1	1	1	1	latch/Retain/ memory/ unchange

- Problem with SR (or $\bar{S}\bar{R}$) latch:
  As soon as you apply/change I/p, O/p will change

SR latch: no controlling I/p to control it's working.



- Timing diagram of SR latch



- Timing diagram of SR latch

$Q = Q_R$ & $\bar{Q} = Q_S$

SR latch | $\bar{S}\bar{R}$ latch:

problem: no controlling i/p.

SR latch:

S R	$Q = Q_R$	$Q_S = \bar{Q}$	
0 0	Q_t	$\bar{Q}_t$	} valid cases.
0 1	0	1	
1 0	1	0	
1 1	0	0	(forbidden)

Flip-Flop motivation.

- the basic latch change it's state when the i/p signal change
- it's hard to control when these i/p signal will change & thus it's hard to know when the latch may change it's state
- we want to have something like an enable i/p (In this case it's called the clock i/p b/c it's desirable for the state change to be synchronized)

- For Seq. ckts ⇓ clock ⇓ will control working of Seq. ckt.
- For comb. ckts. ⇓ enable signal

similiar behaviour

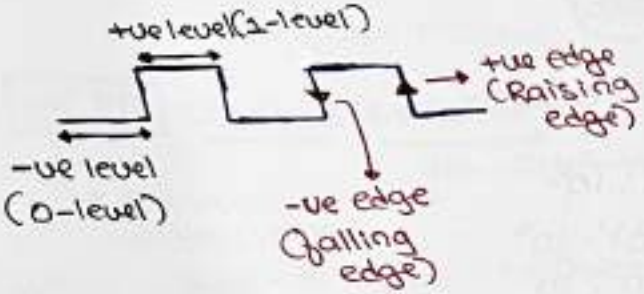
Flipflop = Latch + Clock
- → clocked latch
- → Latch controlled by clock

Clock

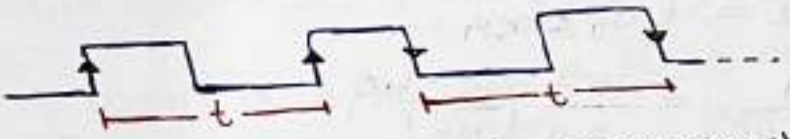
System Clock

- Clock signal control the o/p of seq. circuit that is it determine when & how the memory elements change their o/p.

- if a seq. circuit is not having any clock signal as i/p the o/p of the circuit will change randomly

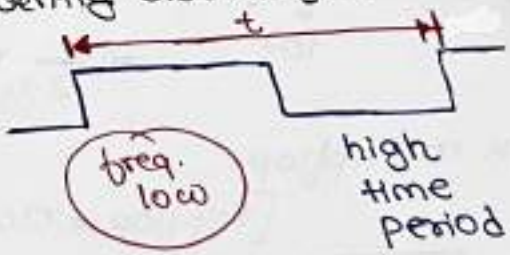


Time period (t): time duration blw Consecutive Risingedge or falling edge



frequency: How frequently clock signal is changing

freq. high — low time period.



freq. (Vs) time period:

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

$t \uparrow \; f \downarrow$

$t \downarrow \; f \uparrow$

note ---------- Unit of time period (seconds (min / hour))

$t = 5ns, 2\mu s \ldots$

Unit of frequency: Hz (hertz)

Hz = per second $\left(\frac{1}{sec} = Hz\right)$

ex: $t = 5ns \quad f = ?$

$$f = \frac{1}{5ns} = \frac{10^9}{5} \cdot \left(\frac{1}{sec}\right)^{Hz}$$

($5ns \downarrow 10^{-9}$)

$\Rightarrow 0.2GHz$

nano : 10^{-9}	Giga : 10^9
micro : 10^{-6}	mega : 10^6
milli : 10^{-3}	Kilo : 10^3

$5\mu s = 5 \times 10^{-6}$ sec $\qquad 2gHz = 2 \times 10^9 Hz$

ex: $f = 2.4GHz \Rightarrow t$ of system clock

$$t = \frac{1}{f} = \frac{1}{2.4GHz} = \frac{10^{-9}}{2.4} \left(\frac{1}{Hz}\right)^{sec}$$

($2.4GHz \downarrow 10^9$)

$\Rightarrow \frac{1}{2.4} nsec = 0.416nsec.$

- Back to Flipflop

[Flipflop = clocked latch]

Latch

Standard FF

[SR latch, $\overline{S}\overline{R}$ latch]

Flipflop $\Rightarrow$ clocked latch

Standard FF

{SR FF, D FF, T FF, JK FF}

① SR flipflop ⟶ Store a bit
(clocked SR latch) ⟶ one bit storage device

- Behaviour / characteristic of SR FF:

Clock	S	R	Q_{t+1} (nxt o/p)
0	x	x	Q_t (no change)
1	0	0	Q_t (Retain / unchanged)
1	0	1	0 Reset
1	1	0	1 Set
1	1	1	x ⟶ forbidden

never occurs.

SR FF — Clocked SR latch (Set-Reset latch)
↳ SR latch with clock

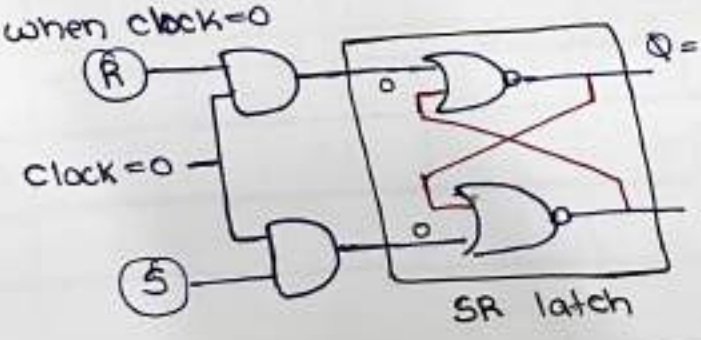
when clock=0 then FF Doesn't Respond to change in I/P

f(a,b) = X (Don't Care)
iff
① I/p Doesn't occur / I/p never occurs
↳ Don't care combination

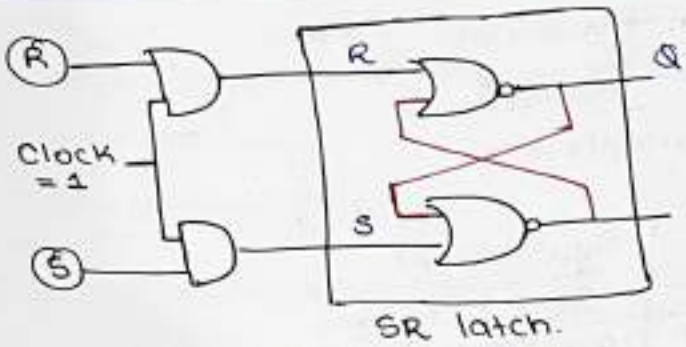
② I/p can occur, But function value Doesn't matter

SR latch (By default)
- SR latch with NOR
- SR latch with NAND

- SR latch ⟹ (+clock) SR flipflop
- when clock=0



Q = unchanged whatever R,S



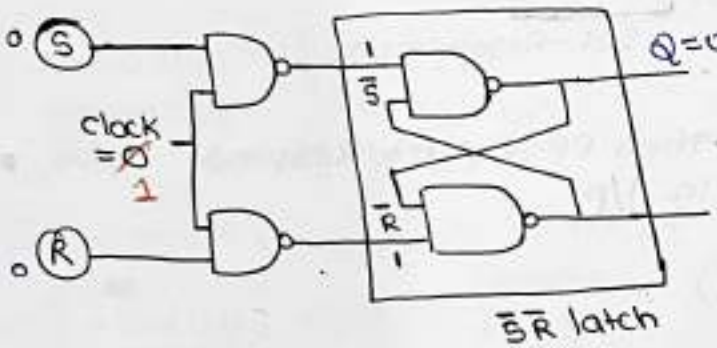
SR latch.

if clock = 1

R	S	Q_{t+1}
0	0	Q_t
0	1	1
1	0	0
1	1	x

- SR flipflop using SR latch (using NOR)

$\bar{S}\bar{R}$ latch $\Rightarrow$ SR flipflop



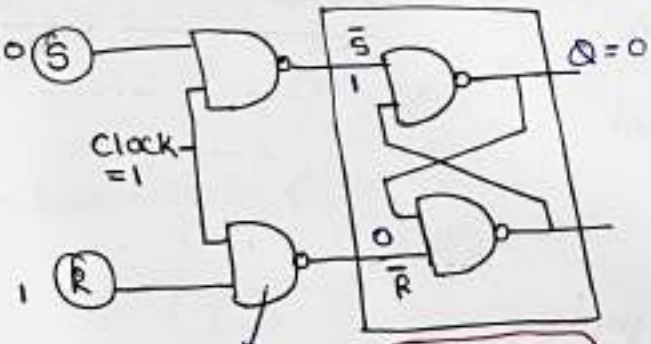
Q = unchange

unchange R, S

$Q = Q_t$

$\bar{S}\bar{R}$ latch

S	R	Q_{t+1}
0	0	Q_t



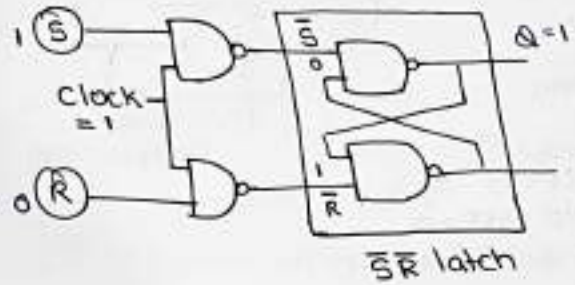
S	R	Q_{t+1}
0	0	Q_t
0	1	0

$\bar{S}R$ latch

Active low

we are using NAND gate b/c we want $\bar{R}$

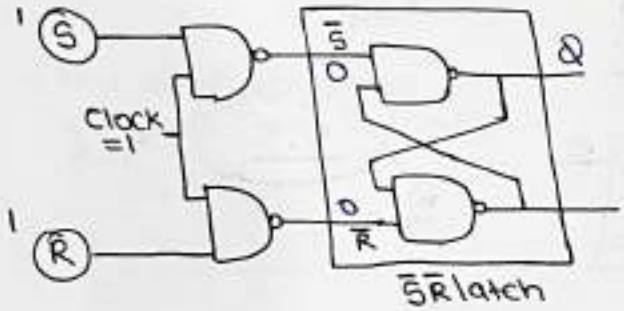
$\bar{S}\bar{R}$ latch $\Rightarrow$ SR flipflop



$\bar{S}\bar{R}$ latch

SR	Q_{t+1}
0 0	Q_t
0 1	0
1 0	1

$\bar{S}\bar{R}$ latch $\Rightarrow$ SR flip-flop



$\bar{S}\bar{R}$ latch

S R	Q_{t+1}
0 0	Q_t
0 1	0
1 0	1
1 1	X

SR $\Rightarrow$ fixed behaviour

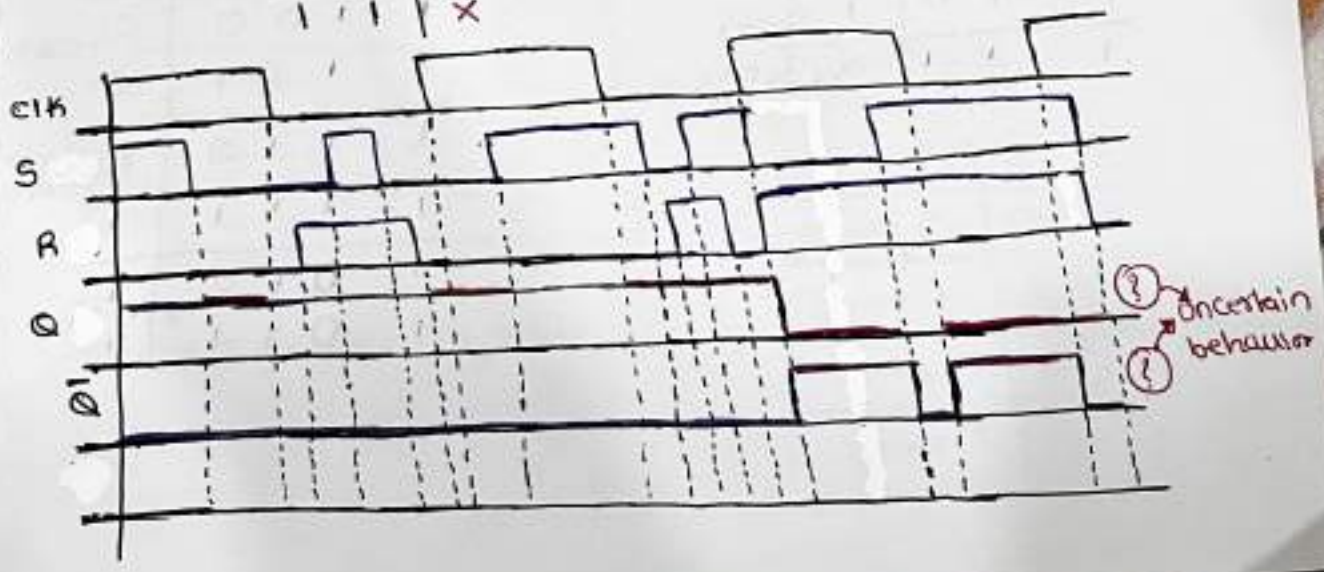
$\Downarrow$

S R	Q_{t+1}
0 0	Q_t
1 0	1
0 1	0
1 1	X

when {clock=0}
In SR flipflop

S R 1 1 $\Rightarrow$ S R 0 0 — no problem

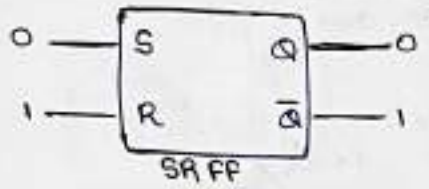
FF Doesn't Respond to i/p change



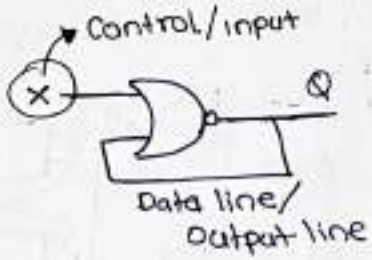
① SR Flip Flop

IN Seq. circuit.

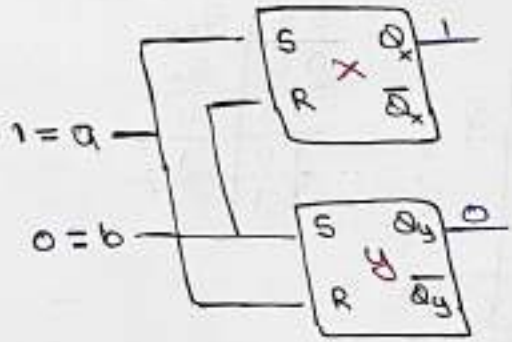
① State = O/p = (Stored) Data



State of FF = 0
O/p of FF = 0
Data (stored) = 0



Circuit.



State of circuit ⇒ 10

O/p

- Characteristic table (behaviour)

Clock	S	R	Q_{t+1}
0	x	x	Q_t (unchange)
1	0	0	Q_t (Retain)
1	0	1	0 (Reset)
1	1	0	1 (Set)
1	1	1	x (don't care)

- Truth table

$Q_{t+1} = f(R, S, Q_t, \text{Clock})$

Clock	R	S	Q_t	Q_{t+1}	
0	x	x	x	Q_t	
1	0	0	0	0	retain
1	0	0	1	1	
1	1	0	0	0	reset
1	1	0	1	0	
1	1	1	0	x	don't care
1	1	1	1	x	
1	0	1	0	1	set
1	0	1	1	1	

$Q_{t+1} = f(S, R, Q_t)$

Q_t \ SR	$S\bar{R}$ (retain)	$\bar{S}R$ (reset)	SR (don't care)	$S\bar{R}$ (set)
$\bar{Q}_t$	0	0	X	1
Q_t	1	0	X	1

S $\quad$ $Q_t\bar{R}$

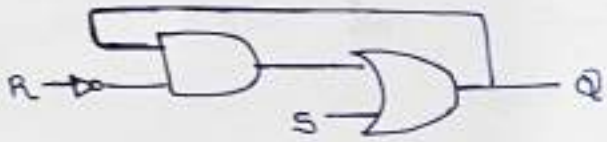
- minimum SOP

$$Q_{t+1} = S + Q_t\bar{R}$$

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

Q_n → next state; Q → present state

- 3rd Implementation of SR FF : Feedback (prev. o/p feeding back)



- Characteristic table (from i/p point of view)

S	R	Q_{t+1}
1	0	1
0	1	0
0	0	Q_t

- Excitation table (from o/p pov)

present Q_t → next Q_{t+1}	S	R
0 → 0	Reset/Retain	
0 → 1	Set	
1 → 0	Reset	
1 → 1	Set/Retain	

- Excitation table (From o/p pov)

present Q_t → next Q_{t+1}	S	R
0 → 0	0	1
	0	0
0 → 1	1	0
1 → 0	0	1
1 → 1	0	0
	1	0

- if o/p change's $Q_t \rightarrow Q_{t+1}$

If SR FF

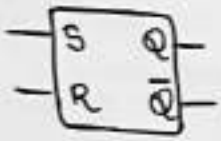
SR (11) ⇒ forbidden → never occurs.

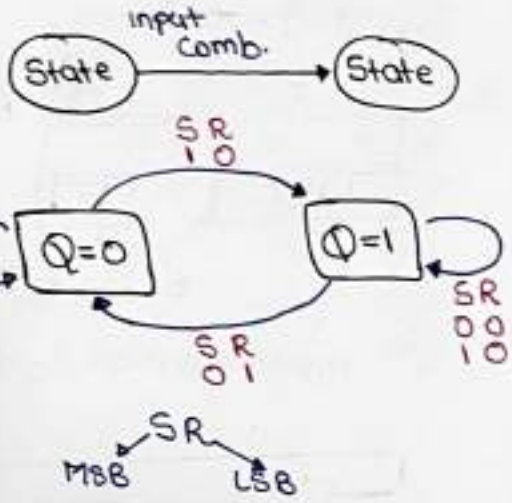
⇒ Excitation table (from o/p pov)

Q_t → Q_{t+1}	S	R
0 → 0	0	X
0 → 1	1	0
1 → 0	0	1
1 → 1	X	0

- State diagram

SR FF : 2 State — o/p





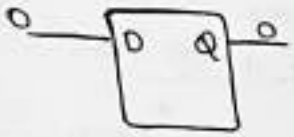
② D Flip Flop → Idea:

Data (Storage) FF

To store a bit, just provide that bit on I/p, we will store it.

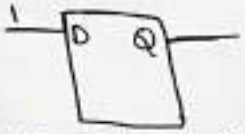
no need of changing 2 signals like SR

Store 0: ≡ Reset ≡ O/p 0



To make $Q_{next}=0$; Present I/p D=0

Store 1: ≡ Set ≡ O/p 1



To make $Q_{nxt}=1$; Present I/p : D=1

- nothing fancy, just store your Data.

D-ff
↳ Data

D	Q_{t+1}
0	0
1	1

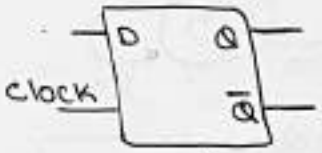
- Characteristic table: Stable table

Clock	D	Q_{t+1} → nxt O/p
1	0	0
1	1	1
0	X	Q_t

when Clock = 0

FF Doesn't Respond to I/p change

- Block diagram



State eq. / Characteristic eq.

$Q_{t+1} = f(D, Q_t)$

$Q_n = f(D, Q) \rightarrow$ present O/p

nxt O/p ; present I/p

Char. equation.

$Q_{t+1} = D$ ⇒ whatever I/p you apply now, will be the nxt O/p (present State Doesn't matter)

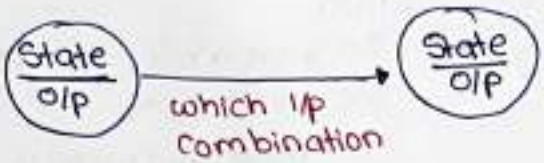
- Truth table $Q_n = f(D, Q)$

D	Q	Q_n
0	0	0
0	1	1
1	0	0
1	1	1

Dummy variable in D-FF

$Q_n = D$

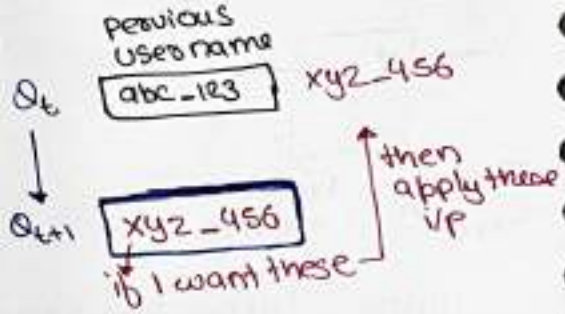
• Excitation table (from o/p POV)



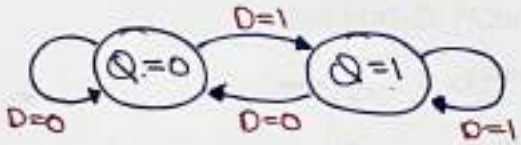
Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

What even nxt state you want; just Apply that as I/p.

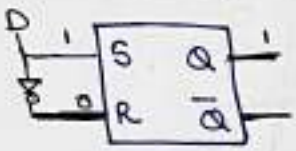
Analogy:
Instagram



State Diagram: State ≡ O/p ≡ Store value



Implementation: "D" using SR FF

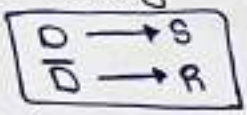


D	Q_{t+1}
0	0
1	1

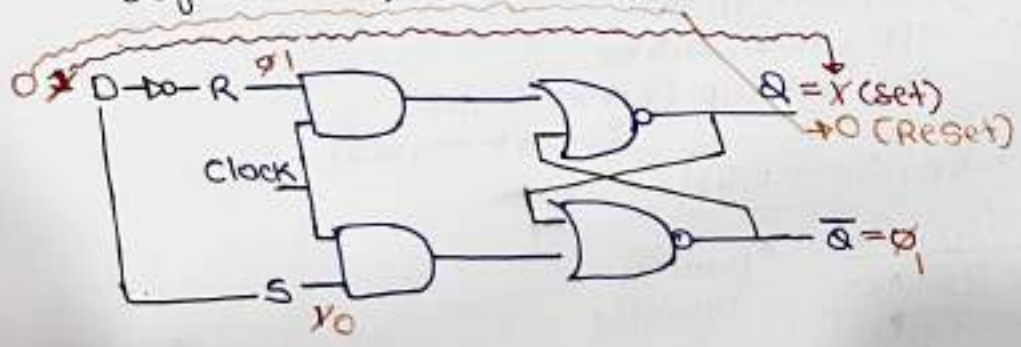
• Implementation:

S	R	Q_{t+1}	D
0	1	0 (Stored)	0
1	0	1 (Store 1)	1

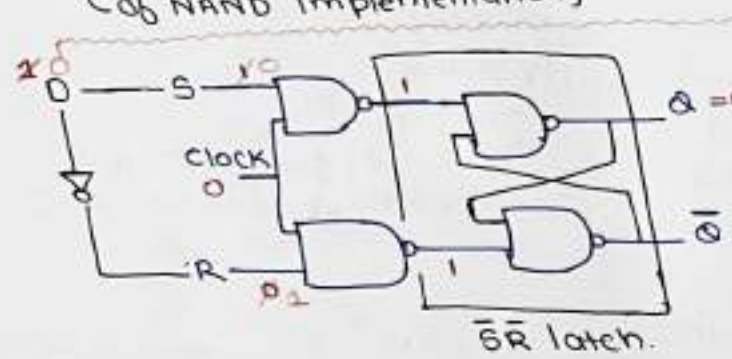
• Implementation "D" using SR



① "D" using SR Flip flop (of NOR Implementation)



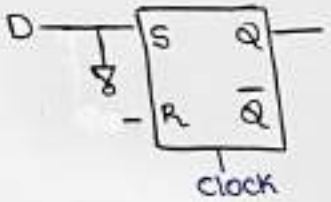
② D using SR flipflop
(& NAND implementation)



Q = unchanged whatever "D" applied

Q = Reset

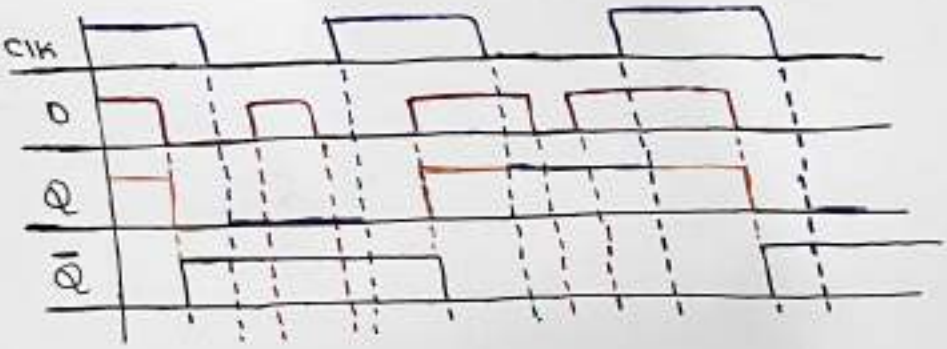
*D using SR



S=R never possible now

when use Implement D using SR

Timing Diagram



Level - Sensitive

Edge - Triggered

Flipflop

2 way to design

level - Triggered FF
(FF Responds to level of clocks)

Edge - triggered FF
(FF Respond to edge of clock)

① +ve level triggered FF

② -ve level triggered FF

① Rising edge triggered FF

② falling edge triggered FF

③ **JK Flipflop**

S R
J K

Similar to SR, but 'i' i/p combination usable

Flipflop: one bit storage device

⇓

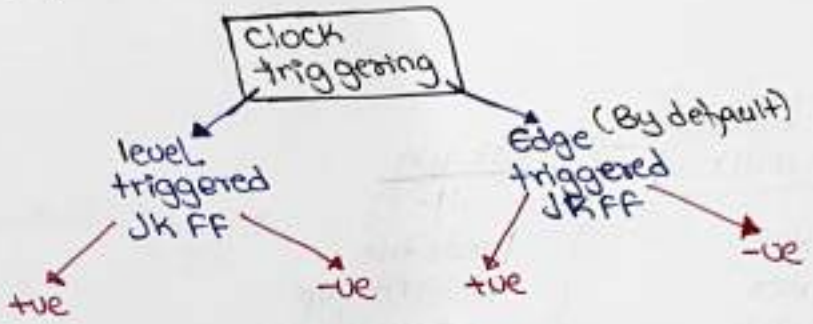
Clocked latch

- Store 0
- Store 1
- Unchanged
- Complement

• Characteristic table (Stable)

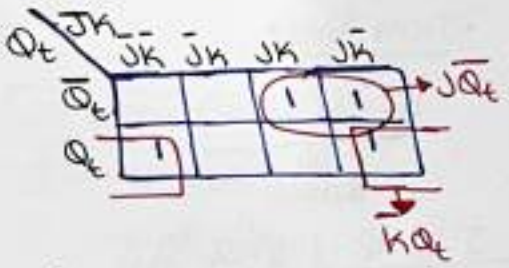
J K
S R

J(Set)	K(Reset)	Q_{t+1}
0	0	Q_t (Retain/ no change)
0	1	0 (Reset)
1	0	1 (Set)
1	1	$\overline{Q_t}$ (Complm^n/ Toggle)



$Q_{t+1} = f(J, K, Q_t)$

K-map of Q_{t+1}

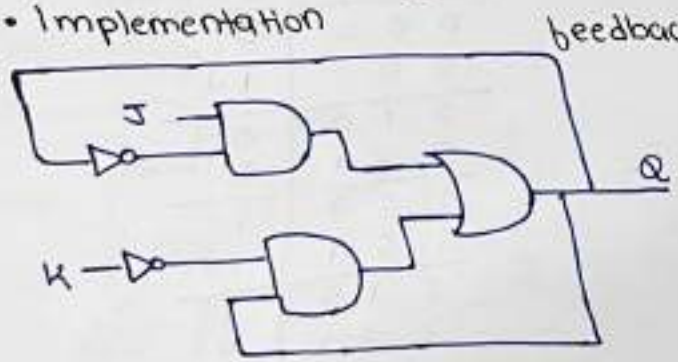


• State eqn (O/p eqn) (Characteristic eqn)

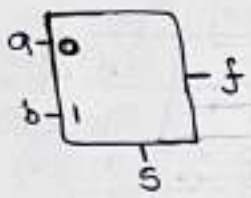
$Q_{t+1} = J\bar{Q}_t + \bar{K}Q_t$

$\boxed{Q_n = J\bar{Q} + \bar{K}Q}$

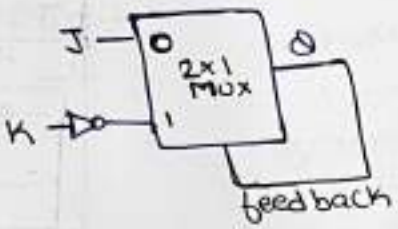
• Implementation



Mux eqn:
$f = \bar{s}a + sb$



JK FF:
$Q_n = \bar{Q}J + Q\bar{K}$



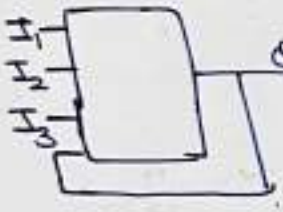
Q_n, Q → not diff. o/p line → same o/p line

Sequential Circuit ⇒ feedback ⇓ feeds the present o/p back as i/p

- flip flop
- Counter
- Register
- finite state machine

- Sequential ckt

o/p $Q_n = f(I, Q)$ (Present i/p, o/p)



- Truth table ⇓ $Q_n = f(J, K, Q)$ ⇓ 8 rows

J	K	Q	$Q^+ = Q_n = Q_{t+1}$	
0	0	0	0	Retain
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

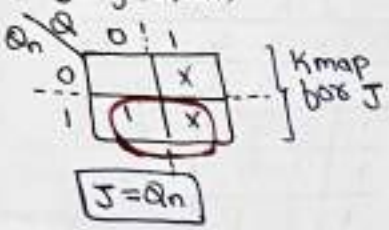
• Excitation Table

$Q \to Q_n$	J	K	
0 → 0			Retain/Reset
0 → 1			Set/Toggle
1 → 0			Reset/Toggle
1 → 1			Retain/Set

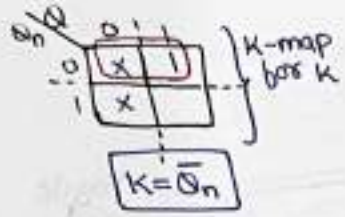
• Excitation table

$Q \to Q_n$	J	K	J	K
0 → 0	0	0	0	x
0 → 1	1	0	1	x
1 → 0	0	1	x	1
1 → 1	0	0	x	0

$J = f(Q, Q_n)$

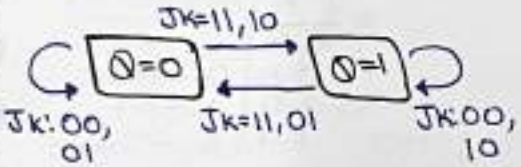


$K = f(Q, Q_n)$



State diagram.

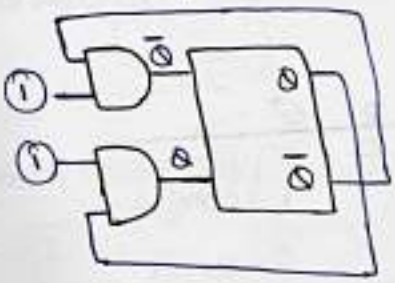
JK-FF ⇒ Two state 0,1



Implementation using SR.

Idea:

J	K	Q_n
1	1	$\bar{Q}$



Q	Q̄
0	1
1	0

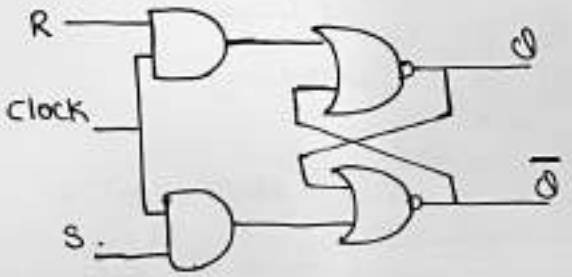
wanted ✓

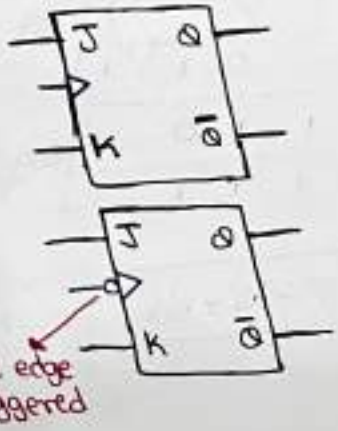
SR to JK:

Q ⟶ with R

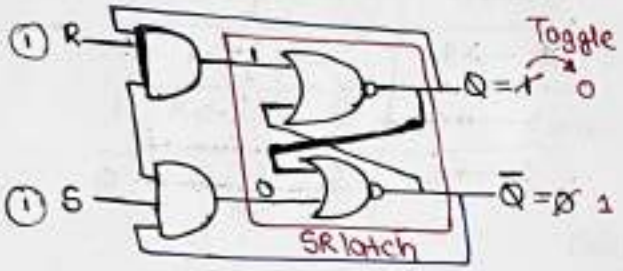
Q̄ ⟶ with S

SR (of NOR)





JK Implementation using SR (ob NOR)



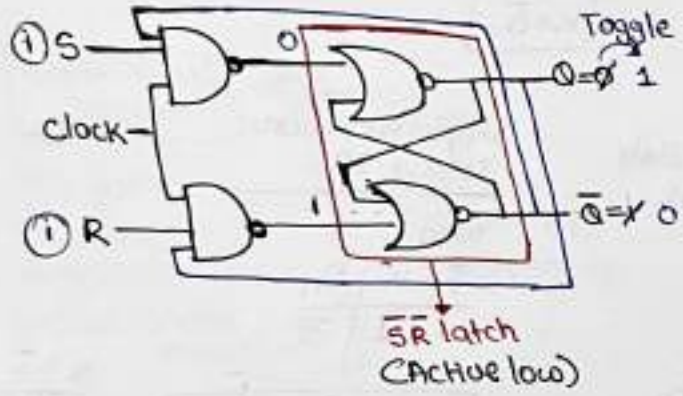
SR to JK :

Q → with R (feedback)

$\bar{Q}$ → with S (feedback)

CIK	J	K	Q_{t+1}	
0	x	x	Q_t	Retain
1	0	0	Q_t	Retain
1	0	1	0	Reset
1	1	0	1	set
1	1	1	$\bar{Q}_t$	complement

JK Implementation using SR (ob NAND)



truth table

J	K	Q_{t+1}	
0	0	Q_t	– Hold
0	1	0	– Reset
1	0	1	– set
1	1	$\bar{Q}_t$	– Toggle

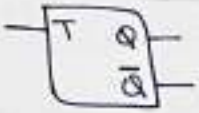
J	K	Q	Q^+	
0	0	0	0	do nothing
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	1	toggle
1	1	1	0	

$$Q^+ = JQ' + K'Q$$

④ **T Flipflop**

Toggle FF

If (T=1) $Q_n = \bar{Q}$, $Q_n = Q$



T=1 makes state (Toggle/complement)

- Truth table

T	Q	Q_n	
T=0	0	0 → 0	Retain/ no change
	0	1 → 1	
T=1	1	0 → 1	
	1	1 → 0	

- State eqn.

$Q_n = T \oplus Q$

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

Clk	T	Q_{t+1}	
not Triggers	X	Q_t	
	0	Q_t	Retain
	1	$\bar{Q}_t$	(Toggle)

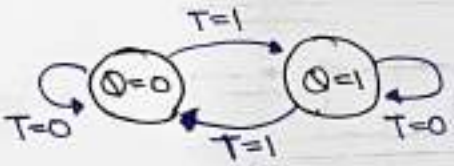
- Excitation table

Q →	Q_n	T	
0 →	0	0	
Toggle 0 →	1	1	
Toggle 1 →	0	1	
1 →	1	0	

$T = f(Q, Q_n)$

$T = Q \oplus Q_n$

State Diagram



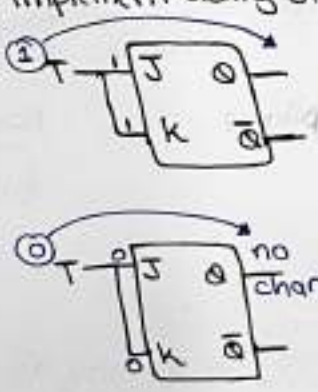
Implementation ①:

$Q_n = Q \oplus T$

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

- Implementation 2 (using MUX)

- Implement using JK



J	K	Q_n	
0	0	Q	
0	1	0	
1	0	1	
1	1	$\bar{Q}$	

relevant to T

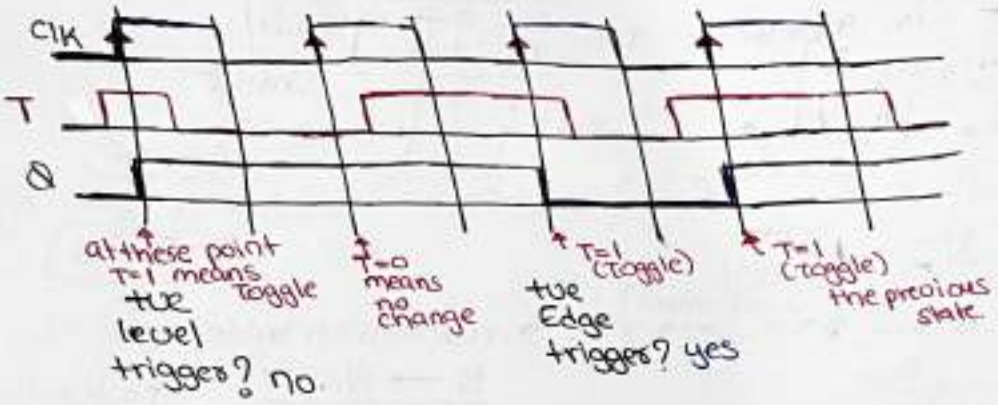
- Truth table

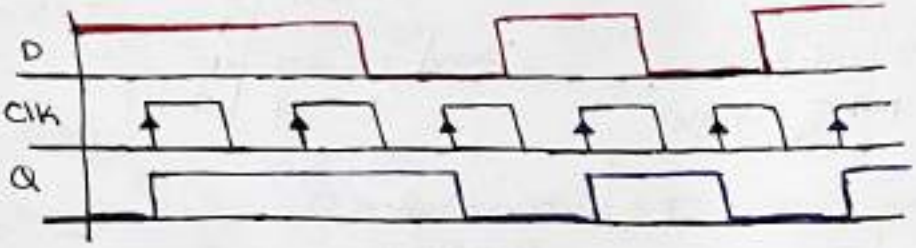
T	Q_{t+1}	
0	Q_t	→ no change
1	$\bar{Q}_t$	→ complement

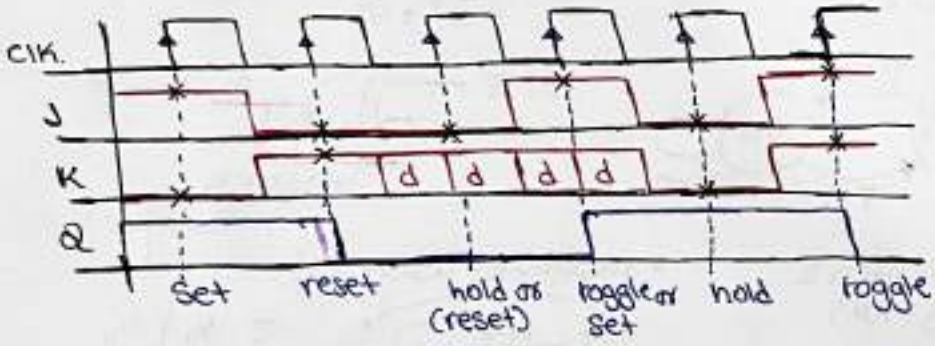
- A slight modification of D-FF for nice application (Ex: counter)

• T-FF
(Timing diagram)

• +ve edge triggering







d: don't care

Author 1: (ve pollao)

Flipflop: clocked latch

Author 2:

flipflop: Edge triggered flipflop

Latch: Level triggered FF b/c as long as I/p are applied O/p will keep changing

→ no concept of clock

ex. Clock triggering
(+ve edge triggered ff, -ve level Triggered ff)

is property of?

① Clock ✓ ② flipflop.

is fixed for all "FF"

+ve edge triggered clock ✗ (non sense)

+ve edge triggered "FF" ✓

means FF is triggered by +ve edge of clock

ex: gate IT 2007

- Cross coupled RS flipflop realized using 2 NAND gate may lead to oscillation?

  it's a latch (not a FF)

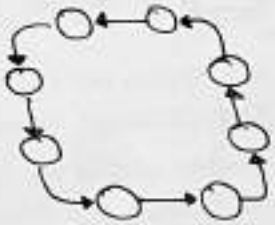
  $\overline{S}\overline{R}$ latch

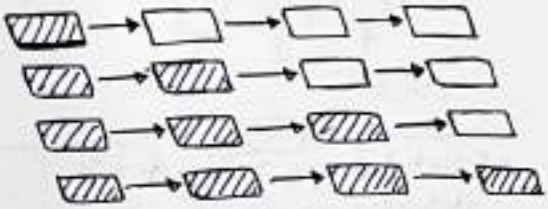
  oscillation condition is 00, 11

Counters

- A counter is essentially a register that goes through a predetermined sequence of binary states.
- Counters are a special type of registers
- A counter is usually constructed from 2 or more FF which change state in a prescibed seq. when i/p pulse are received.

Counters (VS) Shift Registers

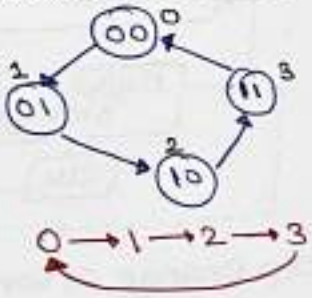




- A counter that follows the binary no. sequence is called a binary counter

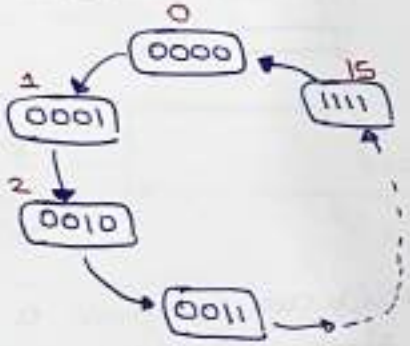
'n'bit counter consist 'n' flipflop & can count 0 to $(2^n - 1)$

2bit binary counters

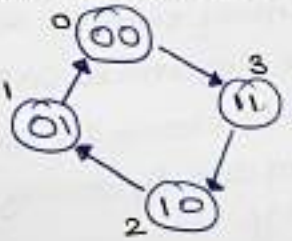


0 → 1 → 2 → 3

4bit binary counter



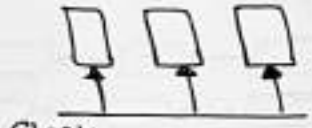
2bit binary count-down counter



3 → 2 → 1 → 0

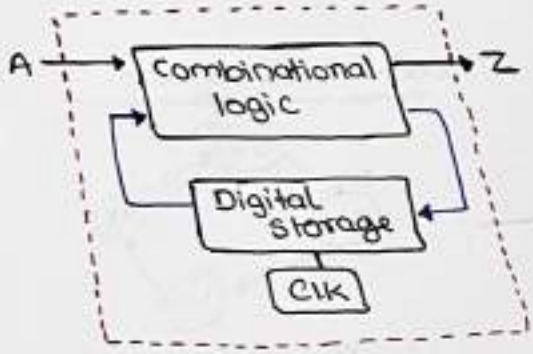
Type of counter

① Synchronous counter
has it's FF (flipflop) clocked at the same time



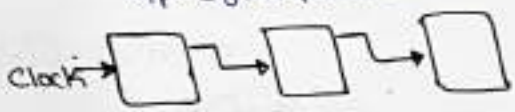
note

- Modulus of a counter is defined as the no. of unique state that a counter will sequence through

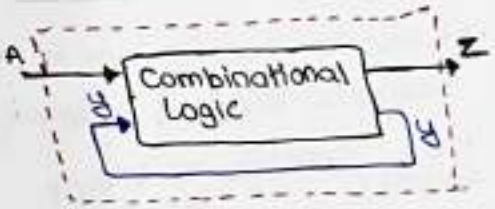


- o/p change at same time

② Asynchronous counter (Ripple counter)
clk of a FF is fed from the o/p of the previous FF



- As soon as i/p change, o/p change immediately w/o waiting for any clock or something



- o/p change at diff. time

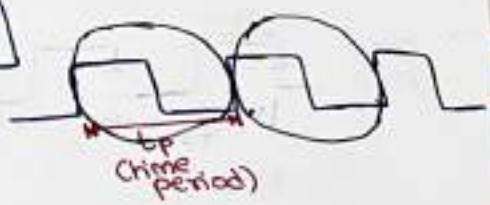
Flipflop

- memory element used in clk sequential circuit
- These circuit are binary cells capable of storing one bit of info.
- A FF circuit can maintain a binary state indefinitely until directed by an i/p signal to switch state
- Bi stable device

- Application of FF
① Counter ② Shift Register ③ Storage Register ④ freq. Divider.

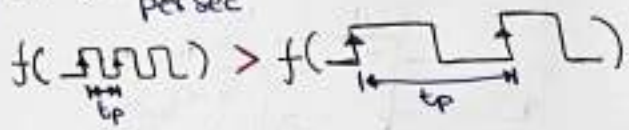
ex: clk signal of frequency "f" Hz
Generate another clk freq "$\frac{f}{2}$ Hz

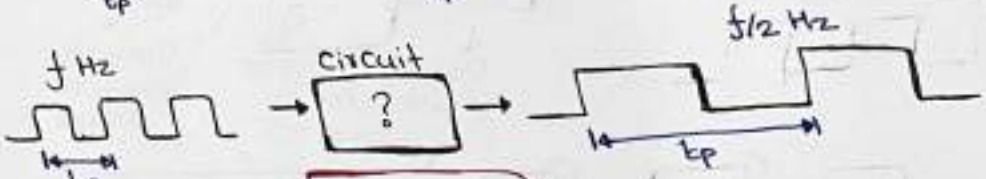
$f = \frac{1}{t_p}$

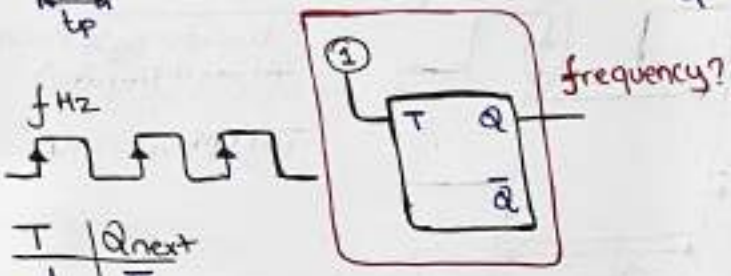


Frequency

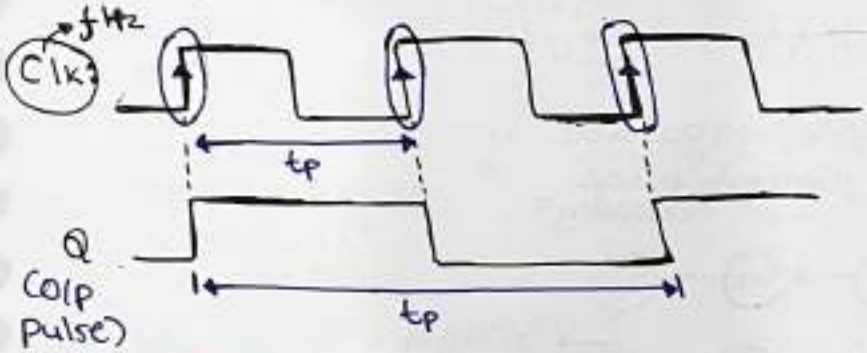
f_{Hz} = f oscillations per sec



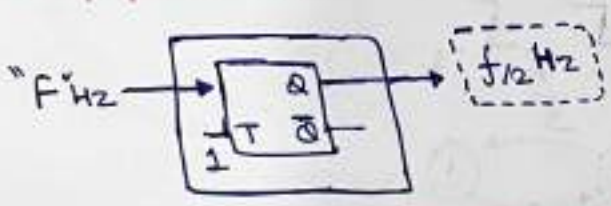




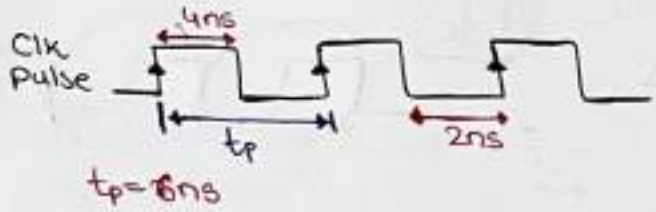
T	Q_{next}
1	$\bar{Q}$



↳ frequency (o/p pulse) = f/2



T-FF Divides the clk freq. by 2.



Duty cycle : $\frac{\text{on period}}{t_p} = \frac{4}{6} = \frac{2}{3}$

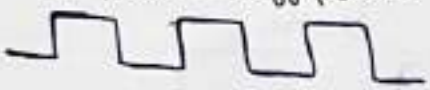
$\Rightarrow \frac{2}{3} \times 100\%$

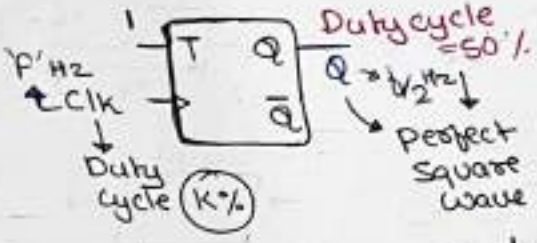
$\Rightarrow 66.66\%$

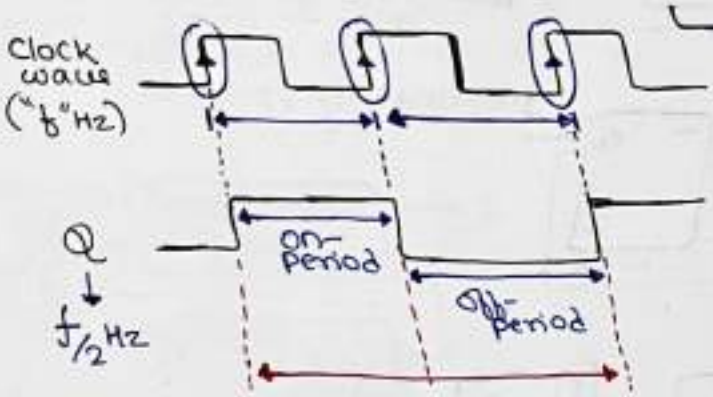
° perfect square wave:

Duty cycle = 50%

on-period = off period







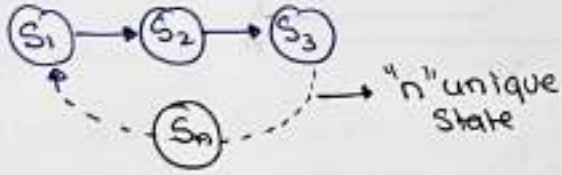
Divide by 2 counters (freqⁿ Dividers)

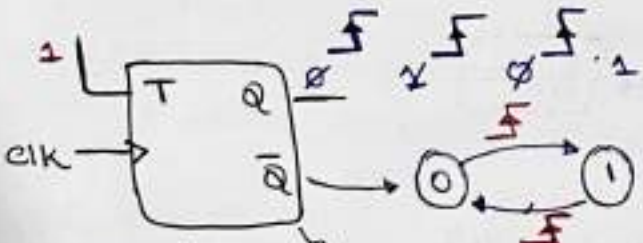
T-flipflop is known as

Clearly there some division going on..

how do we interpret this as counting?

Counter: count → something

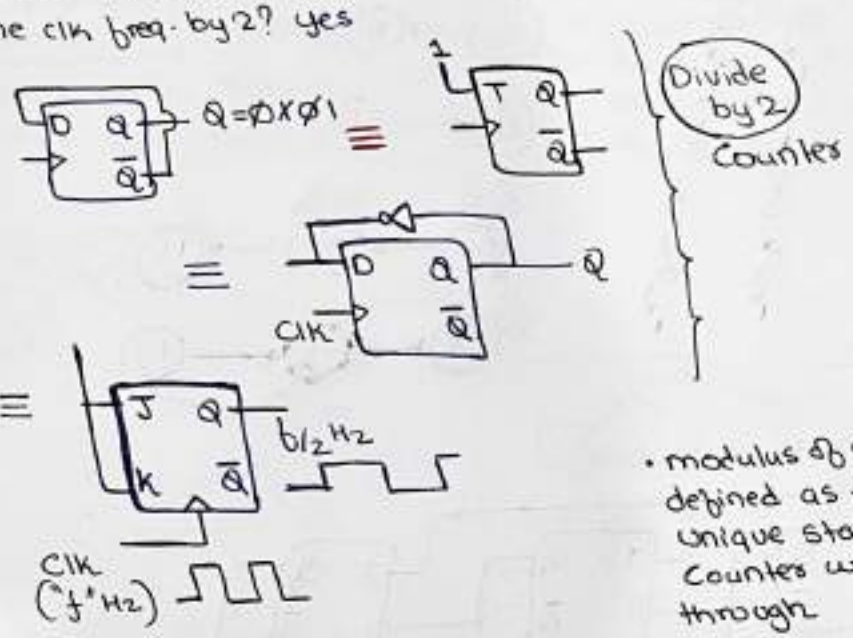




T	Qnext
1	$\bar{Q}$

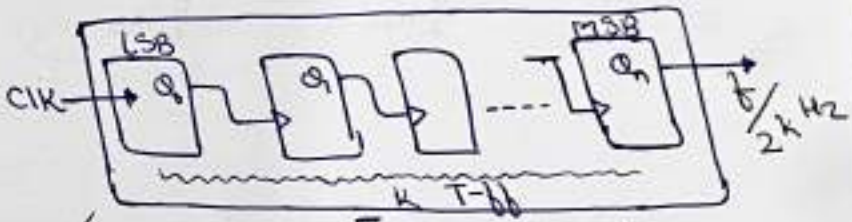
modulo-2 counter

ex: Can we use D-ff to Divide the clk freq. by 2? yes



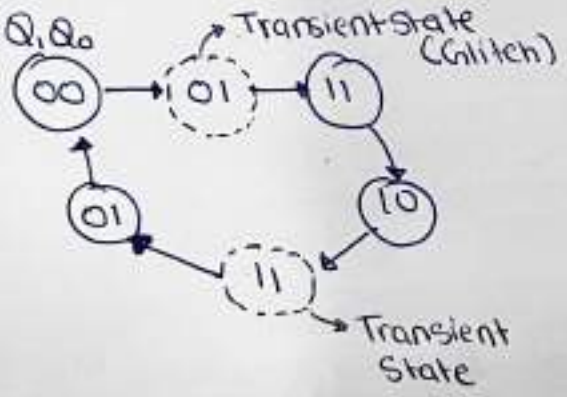
- modulus of a counter is defined as the no. of unique state that a counter will sequence through

ex. CLK freqn of 'f' Hz
How to generate another clk of freqn $b/2^k$

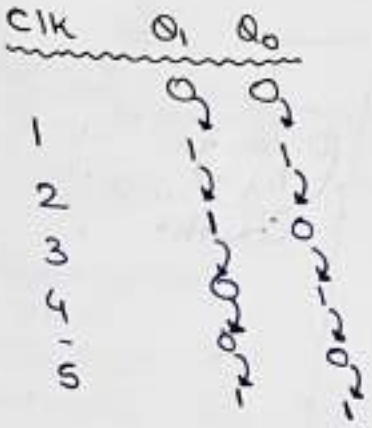


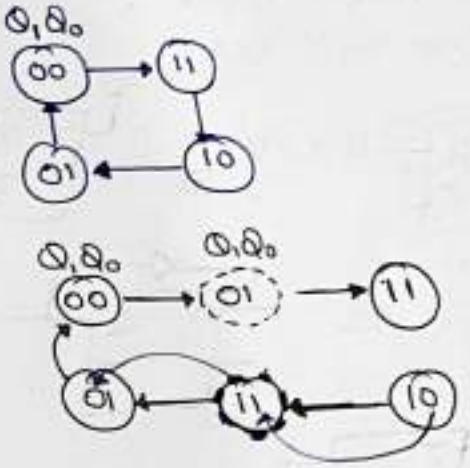
when clk ↓ then Q_0 toggle
& when Q_0 ↓ then Q_1 toggle

assume we have only 2 FF Q_0 & Q_1

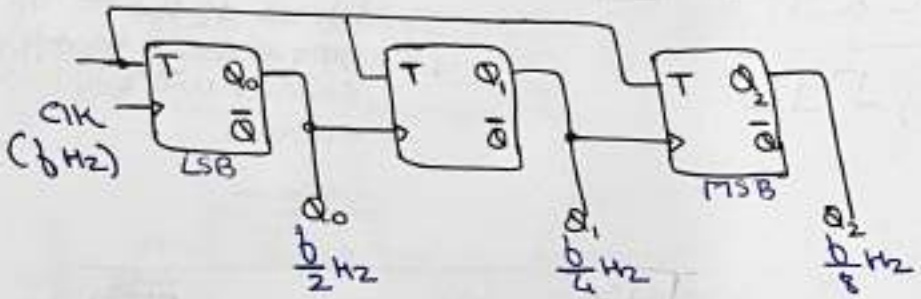


Blc of P_d of ff, we get transient State (Glitches)





- A 3bit down counter

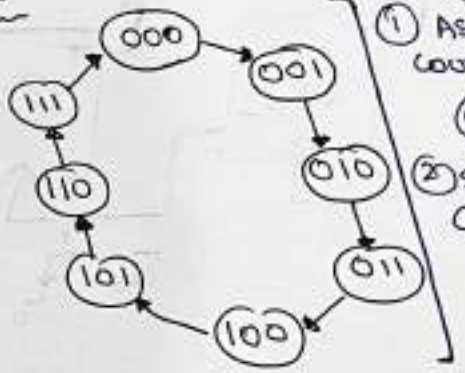


Asynchronous counter

1. Binary Ripple counter
   - Binary counter
- binary counter is a set of FF who's state change on response of pulse applied
- Combined state of the FF at any time is the binary equivalent of the total no of pulse that have been applied.

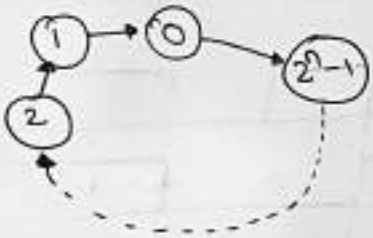
① Binary up counter

3bit Binary counter

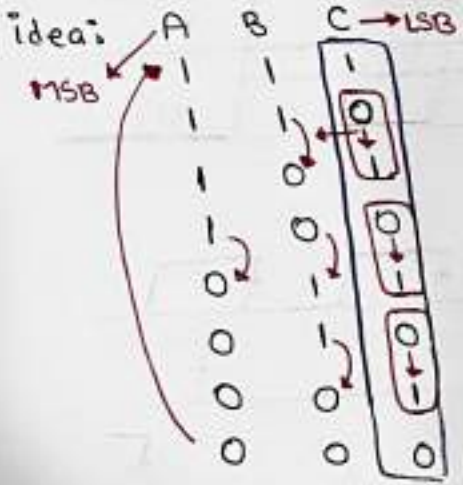


Implement
↓
① Async. counter
or
② Synch. Counter

② 3bit Binary down counter



idea:

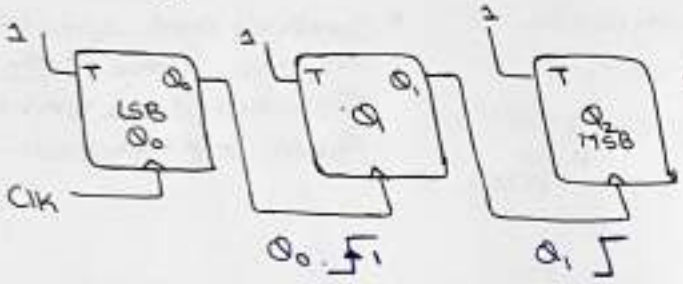


① LSB changes in every cycle
↳ Connect to clock

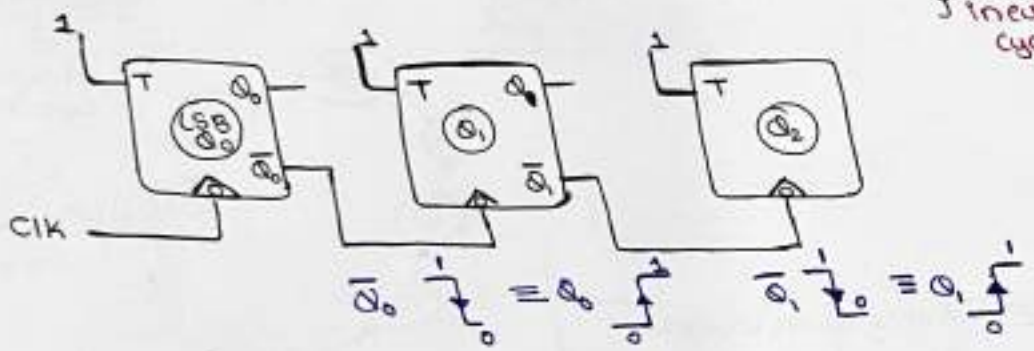
② 2nd bit change when LSB 0 → 1
↳ If ff's are +ve edge triggered: Q_0 to clk of B
↳ If ff are -ve edge triggered $\overline{Q_0}$ to clk of B

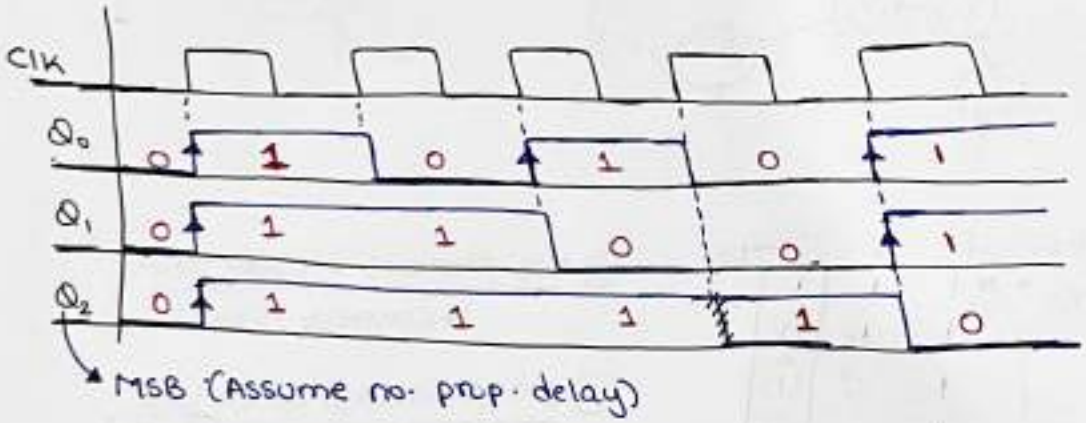
③ 3rd bit change when B 0 → 1
↳ If ff's are +ve edge Triggered: Q_1 to clk of A
↳ If ff's are -ve ---- : $\overline{Q_1}$ to clock of A

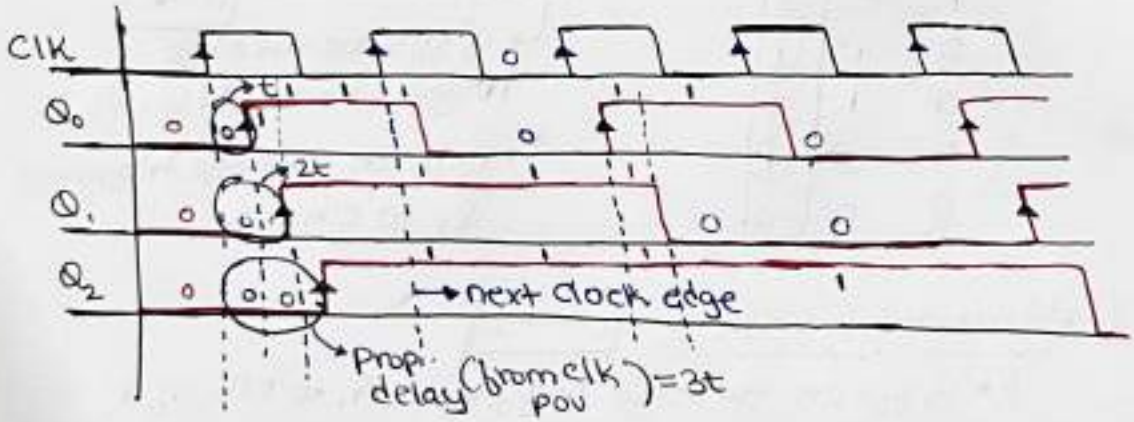
idea: Binary down counter



$Q_2 Q_1 Q_0$ } change in every cycle



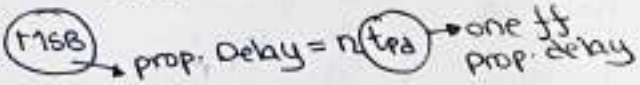


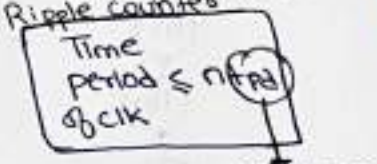


If prop. delay of FF ≠ 0

note n-bit Ripple Counter.

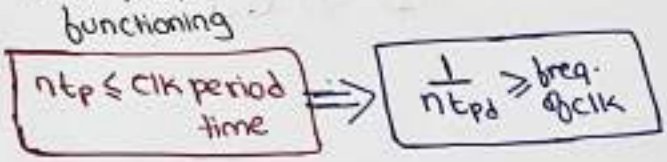
max. no. of ff's we can have in Ripple counters





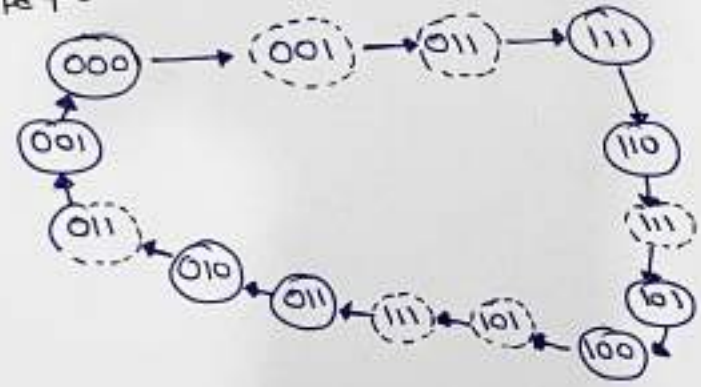
• In Any Ripple counter on n-ffs

For proper functioning



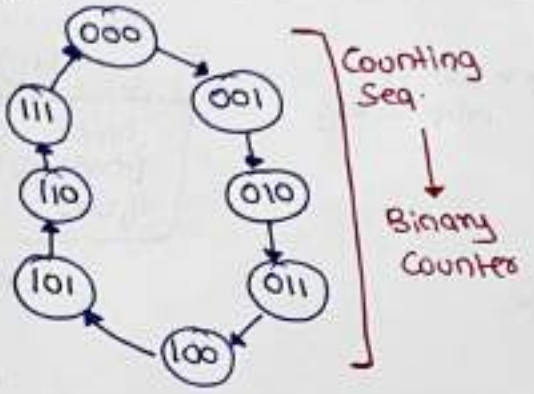
• 3bit Binary down counter (down counter)

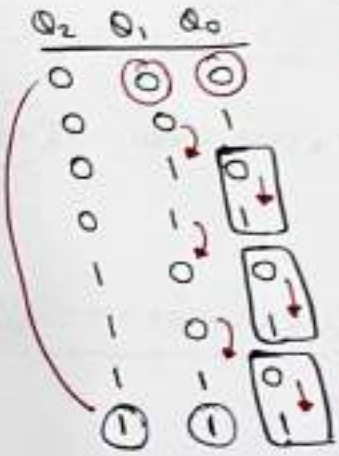
If Tpd ≠ 0



Q_2	Q_1	Q_0
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

2. Binary up counter





? redraw.