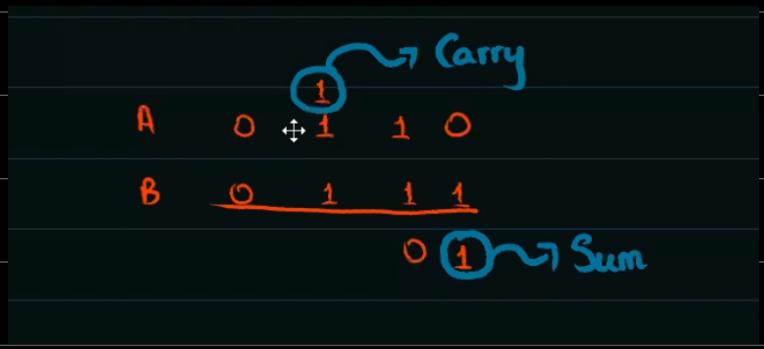
Logic Gates

ADDERS

- Adders are used in electronics
- Digital circuit that performs addition of numbers
- Present in ALU (Arithmetic Logic Unit)

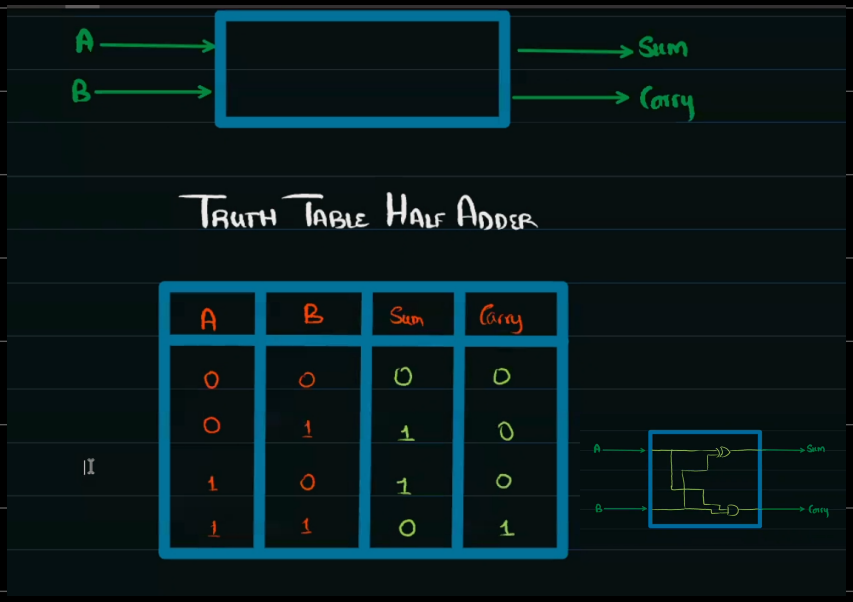


- If we are adding only two bits, then we use half adder.
- If we are adding multiple bits, then we use full adder.

Half Adders

(1) carry
19
13
(2 2) Result of half adder

- Addition of two bits
- Half Adder will not add carry

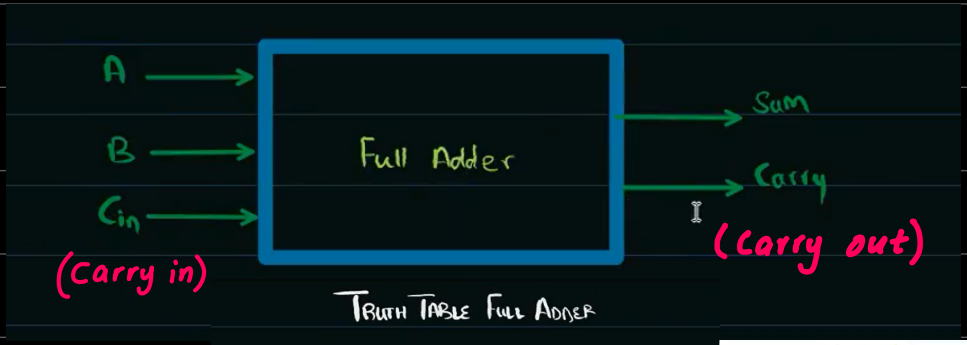


Problem in Half Adder

- Basically we have to consider carry while adding so that's not possible in Half adder
- For that purpose, you have to use full adder.

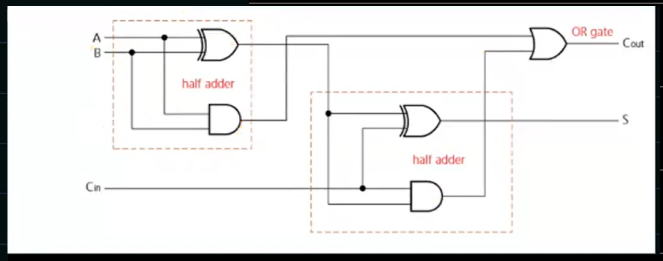
Full Adder

- ADDs three bits
- Not able to add complete nibble



TRUTH TABLE FULL ADDER

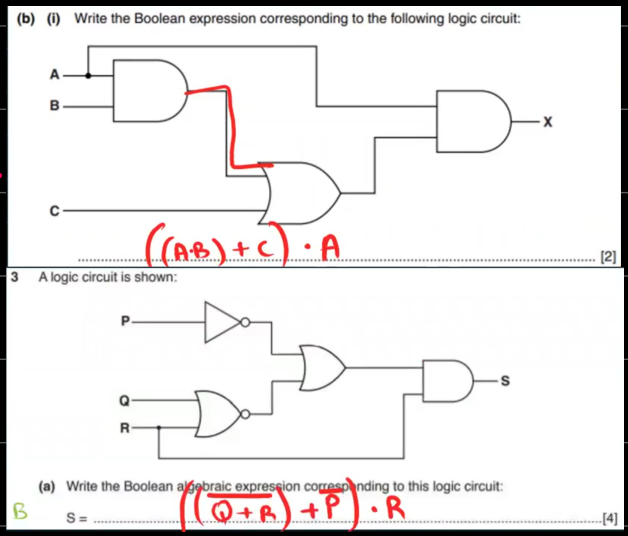
A	B	Cin	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Boolean Expression

- A AND B : $A.B$
- A NAND B : $\overline{A.B}$
- A OR B : $A + B$
- A NOR B : $\overline{A+B}$
- NOT A : $\overline{A}$
- A XOR B : $A \oplus B$

Note : mathematical symbols only represent gates.



Karnaugh Maps (k-Maps)

- It is a graphical representation that provides a systematic method for simplifying the boolean expression

CASE 1 : Writing Expression by Trace Table as Sum of Product

1 = A

0 = $\bar{A}$

1 = B

0 = $\bar{B}$

INPUT			OUTPUT
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Step 1 : Mark Rows with output 1

Step 2: Now write A or $\bar{A}$ / B or $\bar{B}$ / C or $\bar{C}$

Boolean Expression : $\bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + \bar{A}.B.C + A.\bar{B}.C + A.B.C$

Boolean Expression : $\bar{A}.\bar{B}.\bar{C} + \bar{A}.\bar{B}.C + \bar{A}.B.\bar{C} + \bar{A}.B.C + A.\bar{B}.\bar{C} + A.\bar{B}.C$

3 (a) A Boolean algebraic expression produces the following truth table.

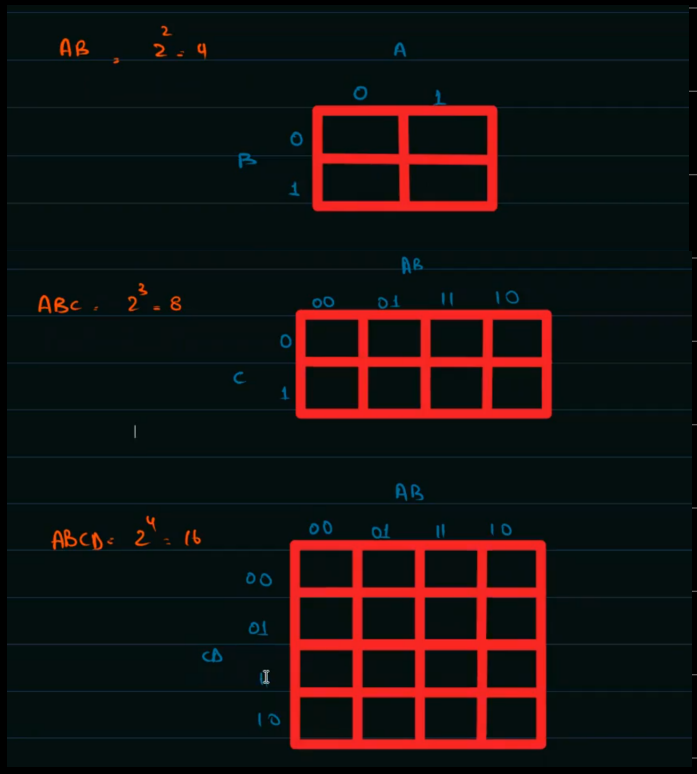
INPUT			OUTPUT
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(iii) Write the simplified sum-of-products Boolean expression for the truth table.

X = ... [2]

CASE 2 : Filling / Constructing k-MAP by Trace Table

- Number of cells depends on the number of inputs
- 2^n = cells where 'n' is the number of inputs



INPUT			OUTPUT
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

AB

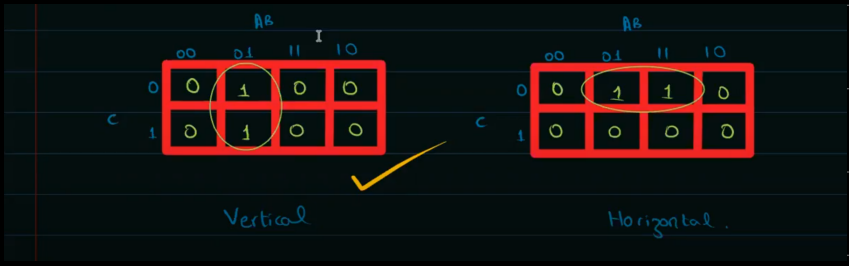
C \ AB	00	01	11	10
0	1	0	0	0
1	1	1	1	1

CASE 3 : Simplifying Boolean Expression

- Looping → adjacent 1
- Equation using these loops

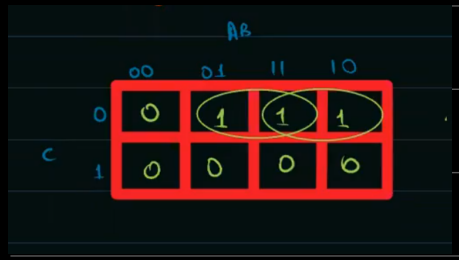
Rules For Looping

① Adjacent 1's are enclosed in a circle

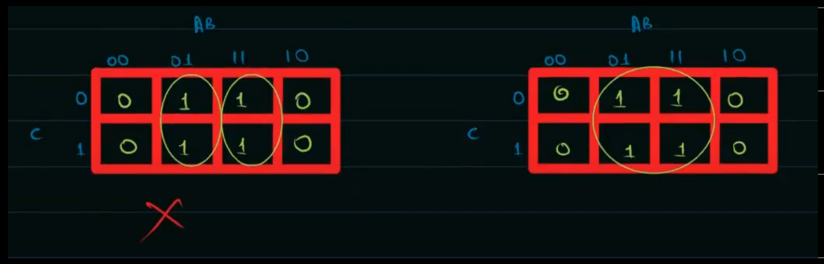


② Number of 1's in a particular loop

③ Overlapping Rule



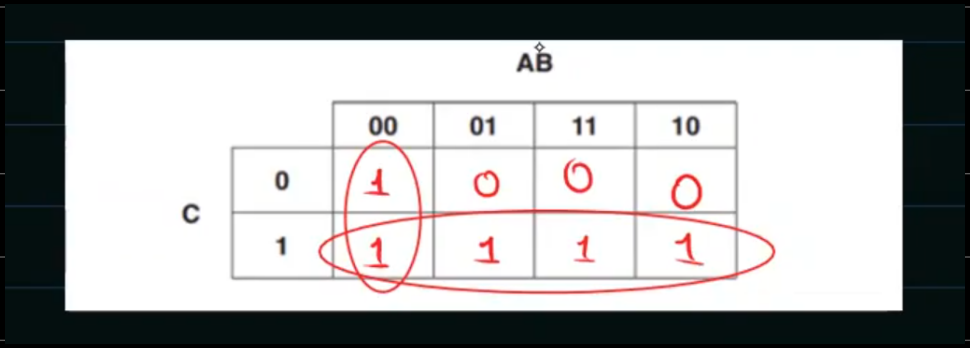
④ Group / loop of 1 should be as large as possible



⑤ 1's in corner are considered as adjacent.

⑥ Diagonal looping is not allowed.

Equation



Note: Loops are getting added

Look for a constant value

Equation → $\bar{A}.\bar{B} + C$

C \ AB	00	01	11	10
0	1	1	0	1
1	1	1	0	1

Equation : $\bar{A} + \bar{B}$

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Equation : $B.C + A.B + A.C$

Boolean Algebra

Laws of Boolean Algebra

* Link symbols and equations with a logic circuit.

① Commutative Law:

$A+B = B+A$ Note: Order does not matter

$A \cdot B = B \cdot A$

② Assosciative Law:

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

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

③ Distributive Law:

- $A \cdot (B+C) = (A \cdot B) + (A \cdot C)$
- $(A+B) \cdot (A+C) \longrightarrow$ Common will come out and will be added into the product of remaining two.
  ↳ $A + (B \cdot C)$

④ Idempotent Law :

$A \cdot A = A$

$A + A = A$

Q- $A+B+C + A+B+C + A+B+C + A+B+C$

$= A+B+C$

⑤ Null Law :

$0 \cdot A = 0$

$1 + A = 1$

Q- $1 + \bar{B} = 1$

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

⑥ Identity Law:

$1 \cdot A = A$

$0 + A = A$

⑦ Inverse Law:

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

$A + \bar{A} = 1$

(8) Absorbtion Law:

- $A + (A \cdot B) = A$
- $A \cdot (A + B) = A$
- $A + (\bar{A} \cdot B) = A + B$
- $\bar{A} + (\bar{\bar{A}} \cdot B) = \bar{A} + B$ ($\bar{\bar{A}} \rightarrow A$)

(9) Demorgan's Law:

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

(10) Double Compliment Law:

$\bar{\bar{A}} = A$

Simplifying Boolean Expressions

CASE 1 : $A + B + \bar{A} + \bar{B}$

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

$= 1 \oplus 1 = 1$ OR gate

CASE 2: $\bar{A} \cdot A + \bar{A} \cdot B + A \cdot B + B \cdot \bar{B} + A \cdot A \cdot A + A \cdot A \cdot \bar{B}$

$= 0 + \bar{A} \cdot B + A \cdot B + 0 + A \cdot A \cdot A + A \cdot A \cdot \bar{B}$ (Inverse Law)

$= 0 + \bar{A} \cdot B + A \cdot B + 0 + A + A \cdot \bar{B}$ (Idempotent Law)

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

$= B \cdot (\bar{A} + A) + A \cdot (1 + \bar{B})$

$= B(1) + A(1)$

$= B + A$

CASE 3: $A \cdot B \cdot C + \bar{A} + A \cdot \bar{B} \cdot C$

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

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

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

$= A \cdot C + \bar{A}$

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

$= \bar{A} + C$

CASE 4: $A.\bar{B}.\bar{C} + A.B.\bar{C} + A.B.C$

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

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

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

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

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

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

* can take common of two terms only as well.

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

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

$= A.[\bar{C}(1)] + A.B.C$

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

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

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

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

CASE 5: $A.\bar{B}.\bar{C} + A.B.\bar{C} + A.B.C$

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

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

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

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

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

Flip-Flops

Combination Circuits

- Output depends on input

Sequential Circuits

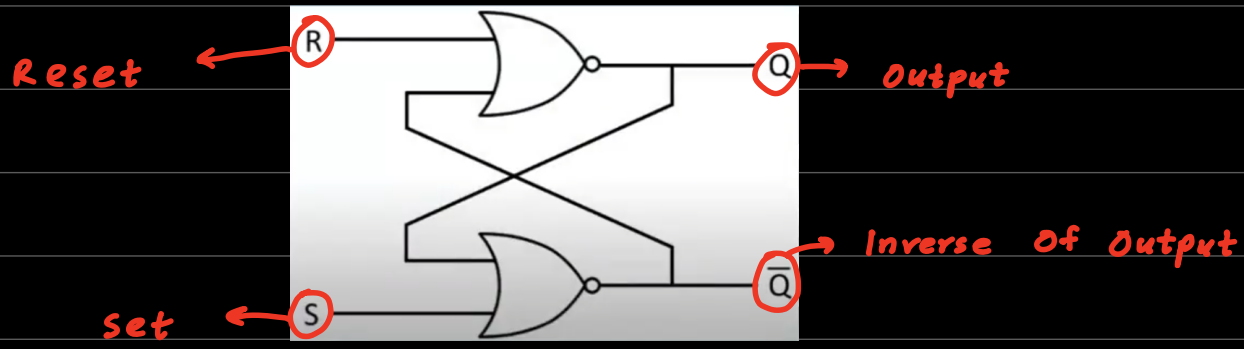
- output depends on the output of previous value.

- Flip-flops are circuits used in digital electronics to store information / binary data

Q- Describe the role of flip-flops in computer?

- A flip-flop can either store a '0' or a '1'
- Computers use bits to store data.
- Flip-Flops can therefore be used to store bits of data
- Memory can be created from flip-flops

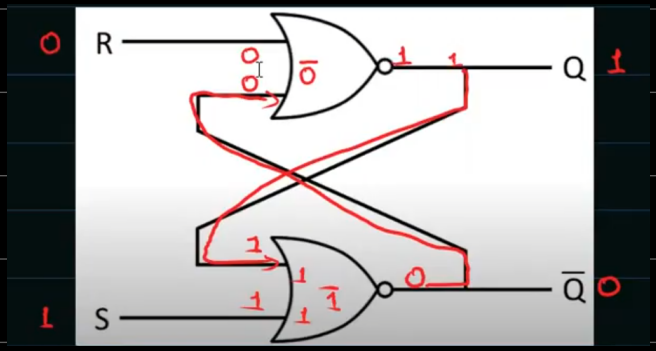
Type 1 : SR Flip-Flops



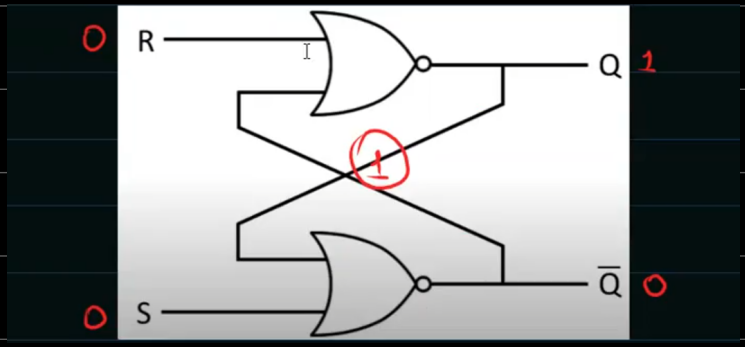
- The data is trapped => output of one gate becomes input of another gate.

One-Bit Memory

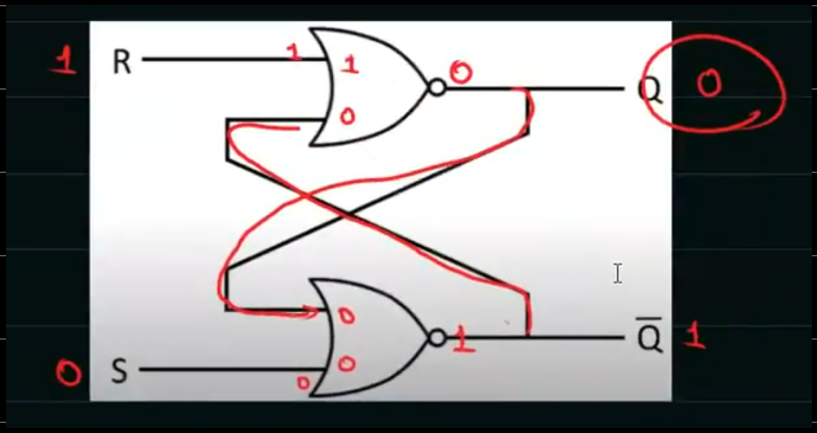
(1)



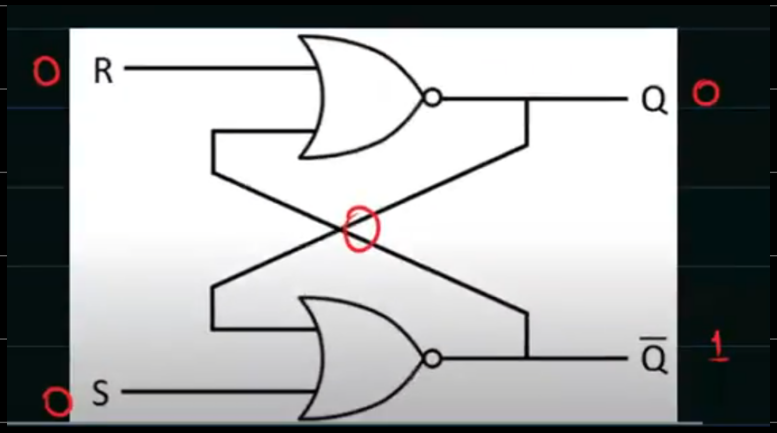
(2) If I want to latch (Trap) it then set and reset value should be 0.



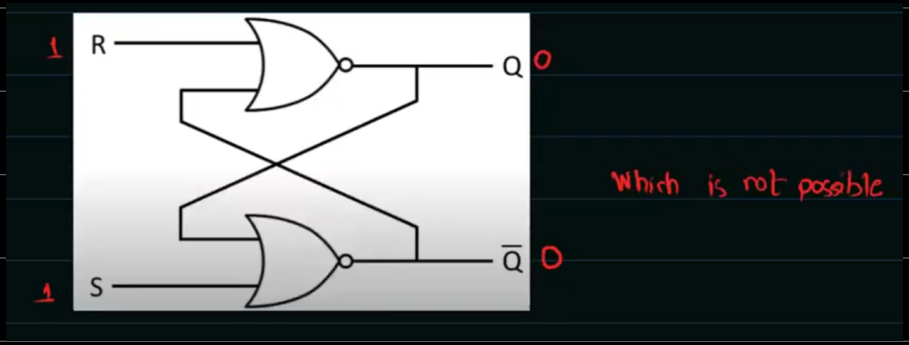
(3)



(4)



⑤ Error



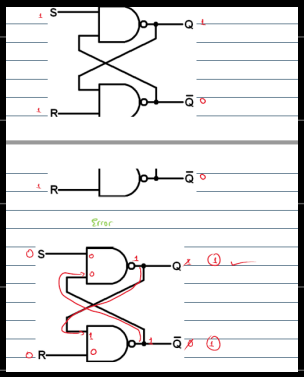
Trace Table of SR NOR Gate Flip-Flop

	S	R	Q	$\bar{Q}$
Initially	1	0	1	0
S changed to 0	0	0	1	0
R changed to 1	0	1	0	1
R changed to 0	0	0	0	1
S and R changed to 1	1	1	0	0

- Whenever the input of S and R is 1 at the same time, then the value of Q and $\bar{Q}$ will be 0 which is not possible as $\bar{Q}$ is the inverse of Q.

NAND Gate SR Flip-Flop

Working



S=1, R=1 → value retained

	S	R	Q	$\bar{Q}$
Initially	1	0	0	1
R changed to 1	1	1	0	1
S changed to 0	0	1	1	0
S changed to 1	1	1	1	0
S and R changed to 0	0	0	1	1

(ii) One of the combinations in the truth table should not be allowed to occur.

State the values of S and R that should not be allowed. Justify your choice.

S = 0 R = 0

Q and $\bar{Q}$ have the same output and they should be compliment of each other due to which flip flops becomes unstable.

[3]

Note: See the pattern in trace table.

Q- What are the two problems in S-R Flip Flop?

- One combination of S and R is there on which Q and $\bar{Q}$ have the same value.
- Input may not arrive at the same time.

Note: Whenever the values of Q and $\bar{Q}$ becomes same the flip-flop becomes unstable

- SR Flip-flops have only two stable states i.e: "0" or "1"

J-k Flip Flop

Q- Explain why the J-k Flip-Flop are an improvement on the S-R Flip-Flop?

· SR Flip-Flops have an invalid combination of S and R

→ J-k flip-flop do not allow for Q and $\bar{Q}$ to have the same value. All four combinations of values of J and k are valid.

· S-R flip-flops inputs may not arrive at the same time

→ J-k flip-flop incorporates a clock pulse for synchronisation

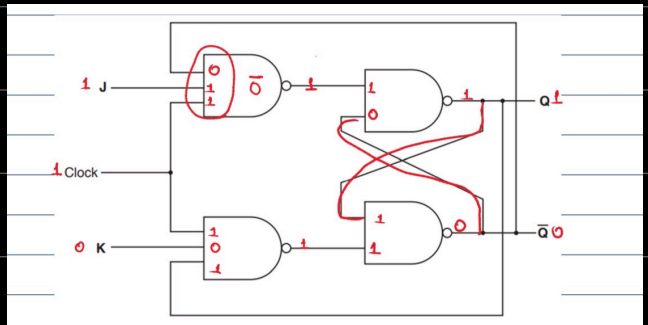
Q- What are the advantages of J-k flip-flop?

· All four possibilities are valid.

· Unstable state is avoided

· Flip-flop is stable.

Note: Always remember that Q and $\bar{Q}$ can not have the same value in J-k Flip-Flop. Consider inputs of other gate if problem arises.



(i) Complete this truth table for the JK flip-flop.

J	K	Clock	Working space	Initial values Q	Initial values $\bar{Q}$	Final values Q	Final values $\bar{Q}$
0	0	1		1	0	1	0
0	0	1		0	1	0	1
0	1	1		1	0	0	1
0	1	1		0	1	0	1
1	0	1		1	0	1	0
1	0	1		0	1	1	0
1	1	1		1	0	0	1
1	1	1		0	1	1	0

Benefits of karnaugh Maps:

- Minimises number of Boolean Expressions
- Minimises number of logic Gates used, making the circuit more efficient.