CHAPTER – 1

Digital Logic Circuit

Meaning of Computer Organisation and Architecture

- Computer Organisation refers to the operational units and their interconnection that realize the architectural specifications.

- Computer Architecture refers to those attributes of a system that are visible to a programmer, or in other words, those attributes that have a direct impact on a logical execution of a program.

Introduction :

- Logical gates are important building blocks in digital circuits. So study of logic gates is very important.
- There are three types of basic gates. AND, OR and NOT gates. Other gates are NAND, NOR, EX-OR, EX-NOR etc.
- English mathematician George Boole invented symbolic logic in 1854, which is known as Boolean Algebra.
- In this chapter we will learn Logic Gates, Laws and theorems on Boolean Algebra, De Morgan’s theorems, Karnaugh map etc.

Logic Gates :

- Before we study Logic gates, let us first understand what are logic levels?
- There are two types of Logic levels : 0 and 1. These show quite different situation as shown in the following table.

Sr.No	Device	Logic 0	Logic 1
1	Switch	Off	On
2	Door	Closed	Open
3	Lamp	Off	On
4	Level	Low	High

Logic Gates :

- **Gates** is an electronic ckt with one or more inputs but only one output. Logic gates process signals which represent **true** or **false.**
- Logic gates are blocks of hardware that produce a logical 1 of logical 0 output signal depending on input signal to logic gate.
- They are also known as logic circuit because with the proper input they establish logical manipulation path.

Logic Gates :

- They also categorized as below :

(1) Basic gates - **(AND, OR, NOT)**

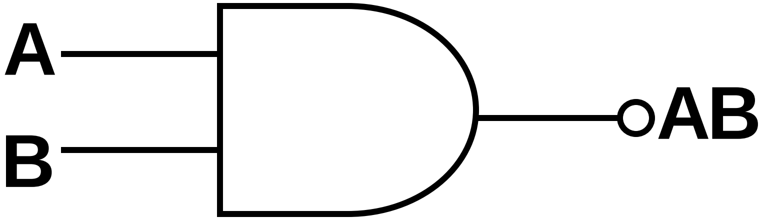
(2) Universal gates - **(NAND, NOR)**

(3) Exclusive gates - **(EX-OR, EX-NOR)**

❖ Basic Gates :

1) AND Gate :

- The AND gate is an electronic circuit that gives a high output (1) only if all its inputs are high.
- A dot (.) is used to show the AND operation i.e. A.B.



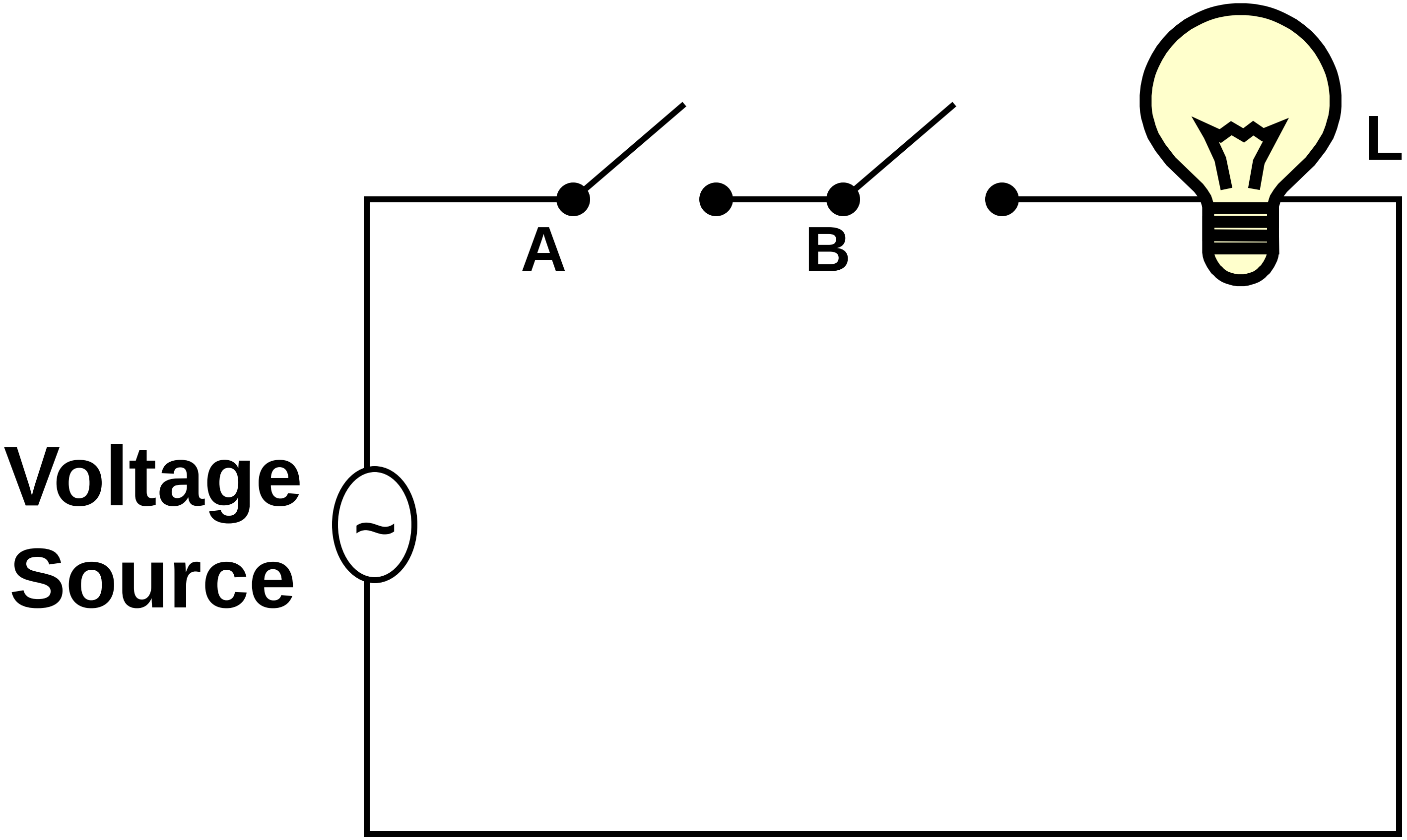
2 input AND gate		
A	**B**	**A.B**
0	0	0
0	1	0
1	0	0
1	**1**	**1**

Truth Table

1) AND Gate :

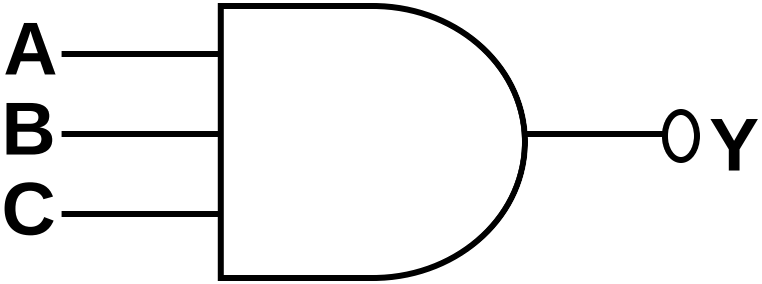
- It means only one time output remain high rest of time output remain low.
- **2^n-1** times output is low where n is number of input.
- Suppose three input AND gate than **$2^3-1=7$** times result is false (low).
- AND gate can be easily explained with following circuit diagram. As shown the circuit if switches A and B both are closed then lamp L glows. If any one switch is open or both the switch are open then lamp L will not glow.

1) AND Gate Example :



Boolean expression is Y=A.B

1) AND Gate : (Three input AND gate)



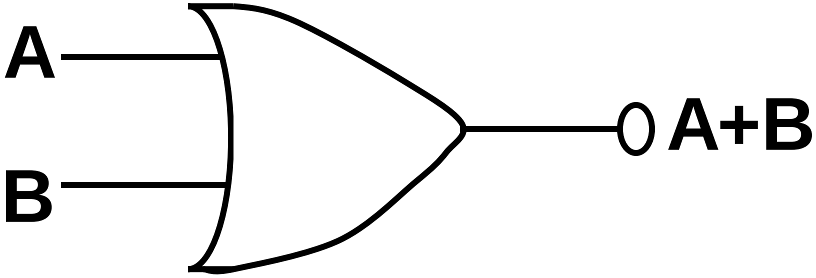
Three input AND gate

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	**1**	**1**	**1**

Truth Table

2) OR Gate :

- The OR gate is an electronic circuit that gives a high output (1) if one or more of its inputs are high.
- A plus (+) sign is used to show the OR operation.



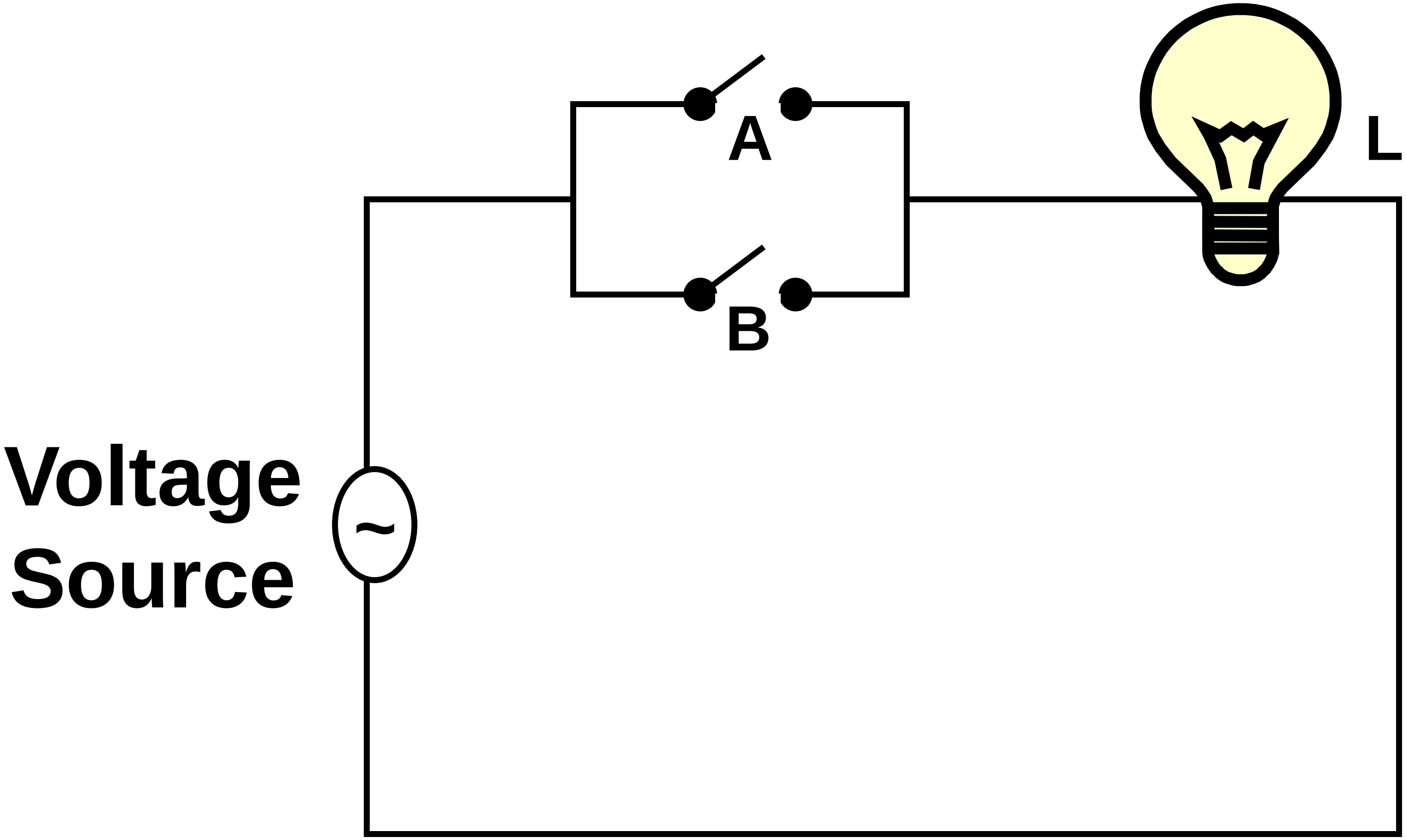
2 input OR gate		
A	**B**	**A+B**
0	**0**	**0**
0	1	1
1	0	1
1	1	1

Truth Table

2) OR Gate :

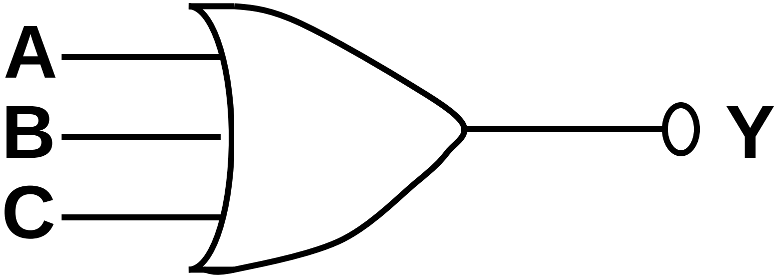
- It means only one time output remain low rest of time output remain high.
- **2^n-1** times result is high where n is number of input.
- Suppose three input OR gate than **2^3-1=7** times result is true (high).
- OR gate can be easily explained with following circuit diagram.
- As shown in this circuit if switches A and B both are open then lamp L not glows, otherwise in other state lamp will glow.

2) OR Gate :



Boolean expression is Y=A+B

2) OR Gate : (Three input OR gate)



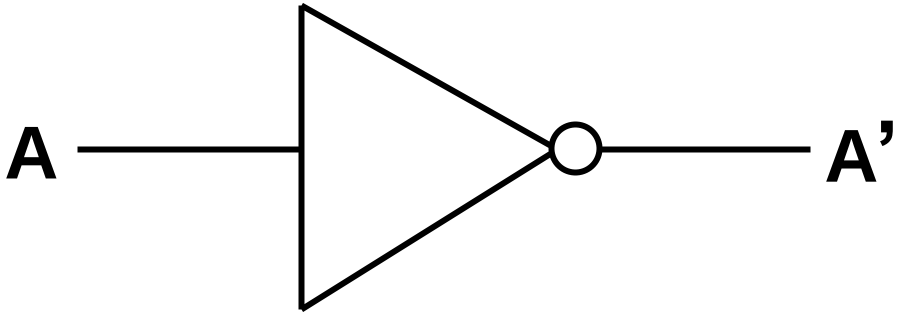
Three input OR gate

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Truth Table

3) NOT Gate :

- The **NOT** gate is an electronic circuit that produces an inverted version of the input at its output.
- It is also known as an ***inverter.*** If the input variable is A, the inverted output is known as NOT A. There is also shown as **A’** or A with a bar over the top as shown as the outputs.



NOT gate	
A	**A’**
0	1
1	0

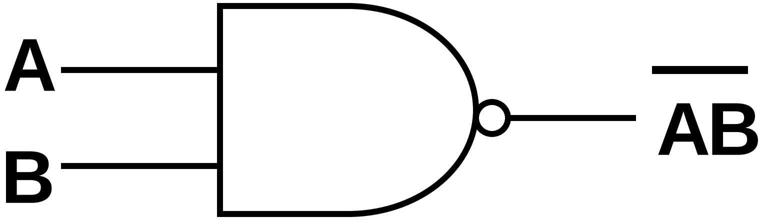
3) NOT Gate :

- When input at logic 0, output is 1 and when input at logic 1 then output is 0.

❖ Universal Gates :

1) NAND Gate :

- NAND gate means NOT-AND gate which is equal to an AND gate followed by a NOT gate.
- The output of all NAND gates are high if **any** of inputs are low. The symbol is an AND gate with a small circle on the output.



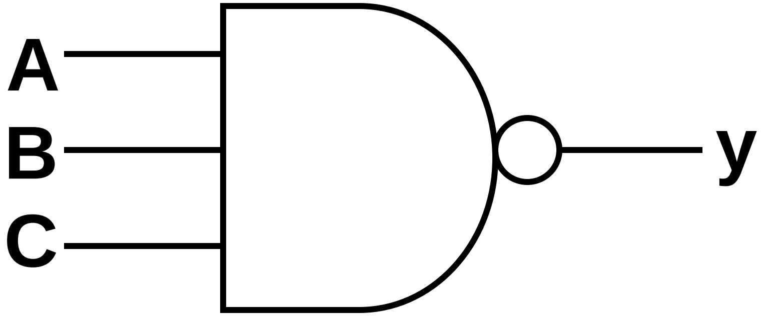
Truth Table

2 input NAND gate		
A	**B**	**A.B**
0	0	1
0	1	1
1	0	1
1	**1**	**0**

❖ NAND Gate : (Three input NAND gate)

1) NAND Gate :

- When all input are high, output remain LOW.



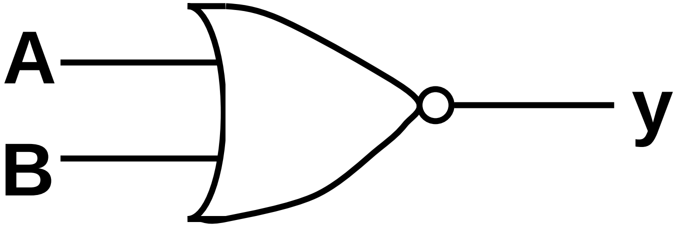
Truth Table

A	B	C	Y
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	1
1	**1**	**1**	**0**

❖ Universal Gates :

2) NOR Gate :

- NOR gate means NOT-OR gate which is equal to an OR gate followed by a NOT gate.
- The outputs of all NOR gates are low if **any** of the inputs are high.
- The symbol is an OR gate with a small circle on the output. The small circle represent inversion.

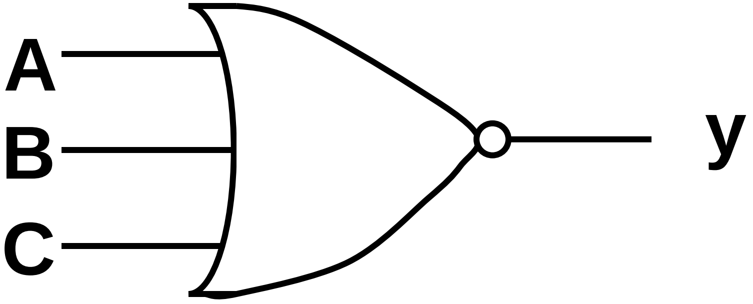


Truth Table

2 input NOR gate		
A	B	$\overline{A+B}$
0	**0**	**1**
0	1	0
1	0	0
1	1	0

❖ NOR Gate : (Three input NOR gate)

2) NOR Gate :



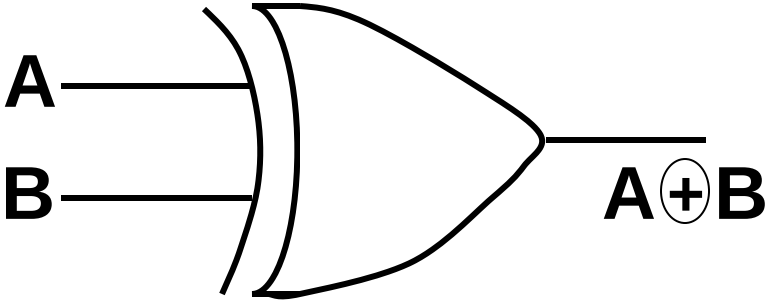
Truth Table

A	B	C	Y
0	**0**	**0**	**1**
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

❖ Exclusive Gates :

1) Ex-OR Gate :

- The **'Exclusive-OR'** gate is a circuit which will give a high output if either, but not both, of its two inputs are high.
- An encircle plus sign ⊕ is used to show the Ex-OR operation.

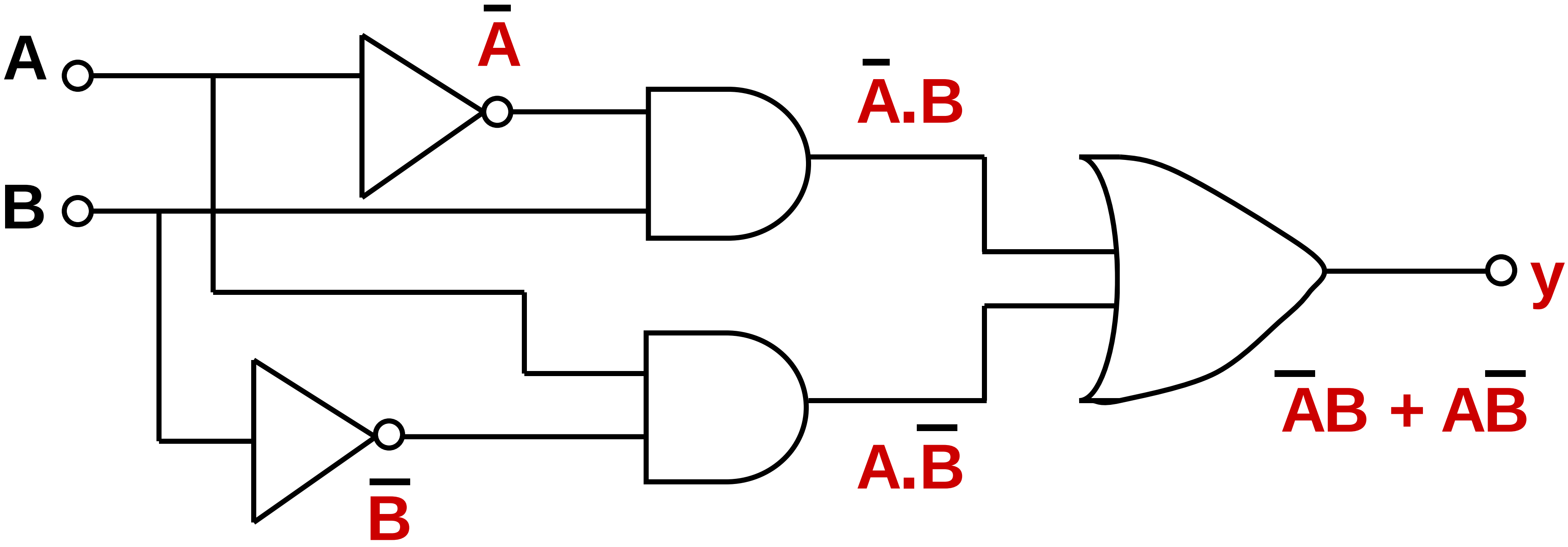


EOR

2 input Ex-OR gate		
A	**B**	**A ⊕ B**
0	**0**	**0**
0	1	1
1	0	1
1	**1**	**0**

Truth Table

1) Ex-OR Gate :

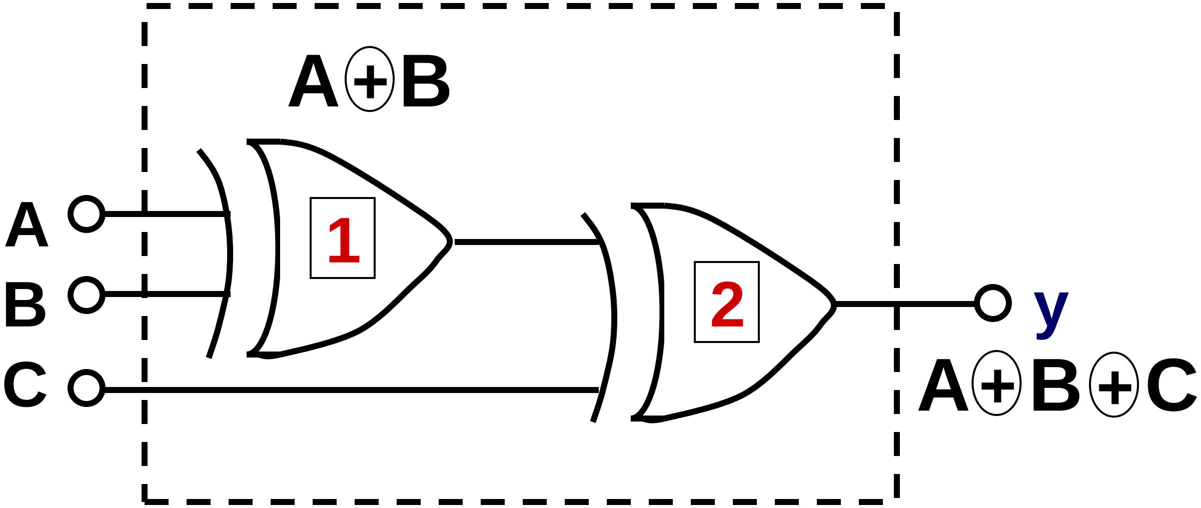


1) Ex-OR Gate : (Three input Ex-OR gate)

Truth Table

A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

1) Ex-OR Gate : (Three input Ex-OR gate)

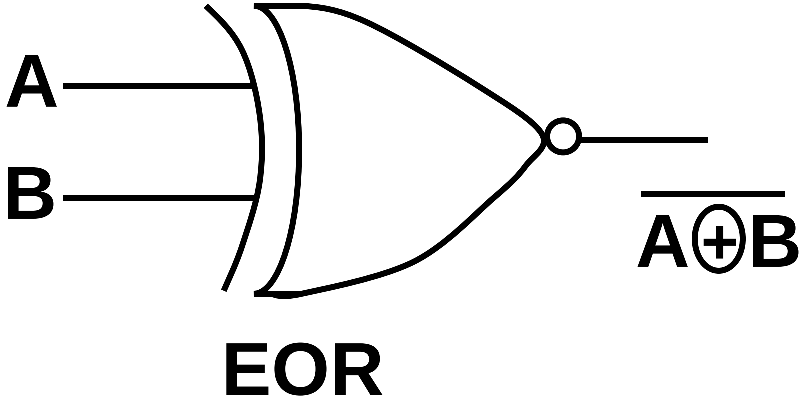


Three input Ex-OR gate

❖ Exclusive Gates :

2) Ex-NOR Gate :

- The **‘Exclusive-NOR’** gate circuit does the opposite to the Ex-OR gate.
- It will give low output if **either, but not both** of its two inputs are high.
- The symbol is an EX-NOR gate with a small circle on the output. The small circle represent inversion.



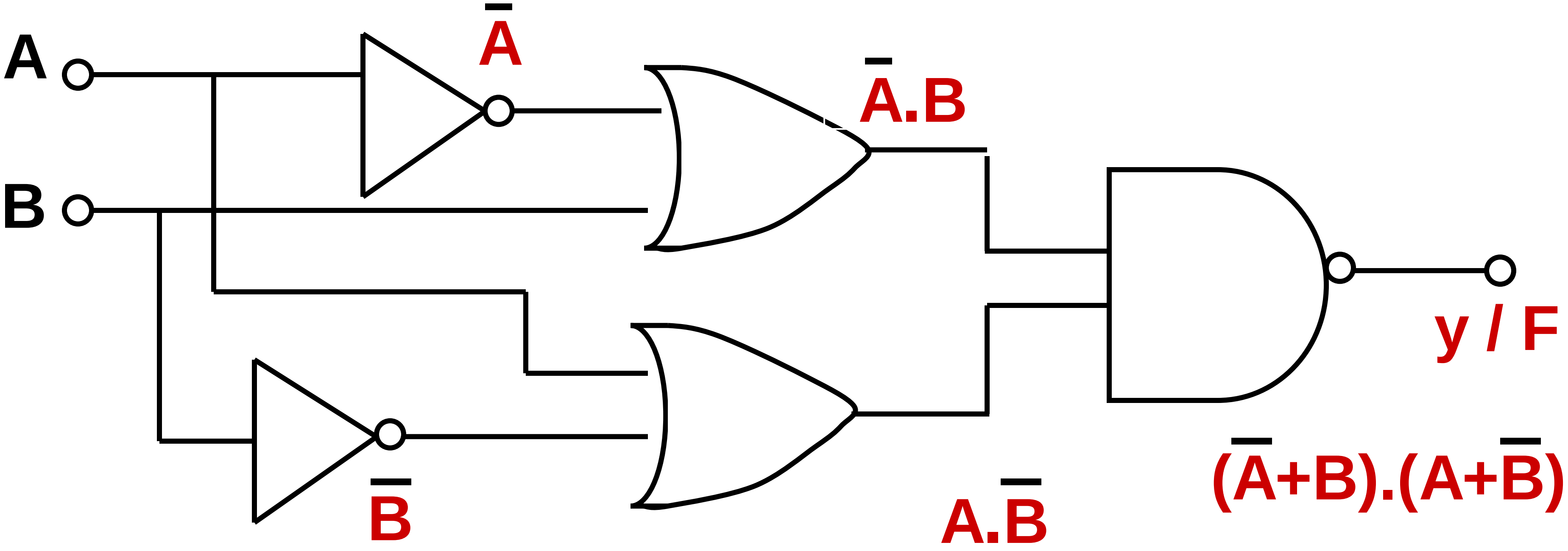
❖ Exclusive Gates :

2) Ex-NOR Gate :

2 input Ex-NOR gate		
A	**B**	$\overline{A \oplus B}$
0	**0**	**1**
0	1	0
1	0	0
1	**1**	**1**

Truth Table

1) Ex-NOR Gate :



Ex-NOR

❖ Exclusive Gates :

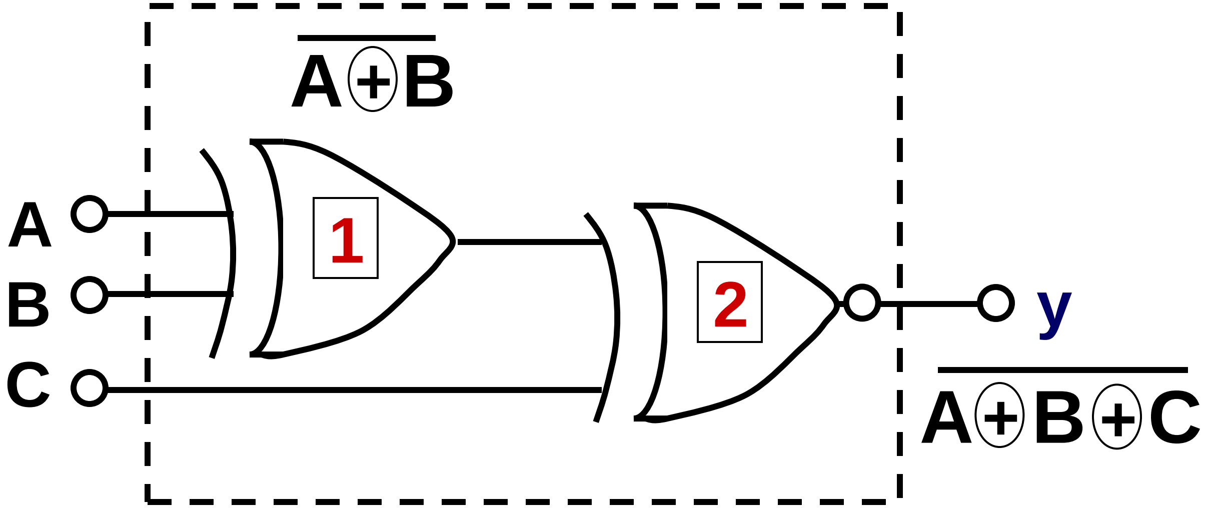
2) Ex-NOR Gate :

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

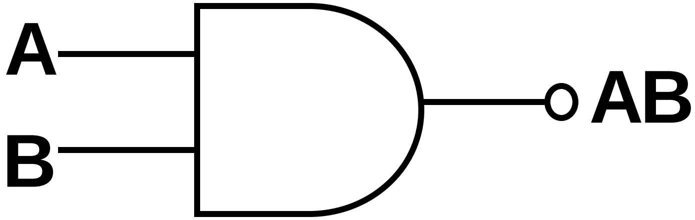
Truth Table

❖ Exclusive Gates :

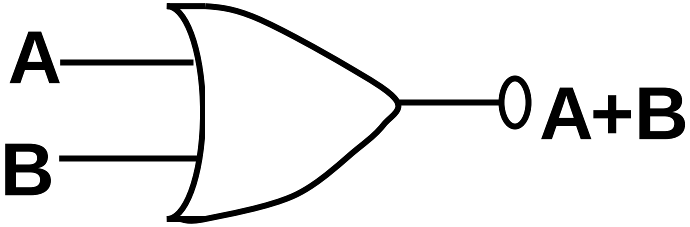
2) Ex-NOR Gate :



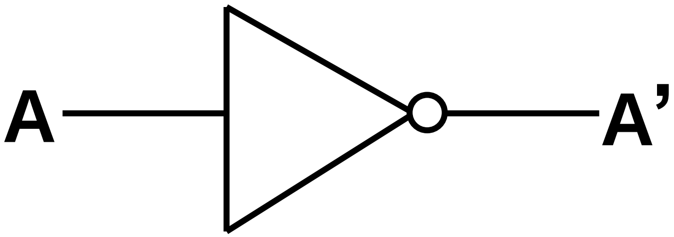
❖ Summary of Logic gates :

<table>
<tr><th>No</th><th>Name of gate</th><th>Logic Diagram</th><th>Truth Table</th></tr>
<tr><td>1</td><td>AND</td><td> </td><td>
<table>
<tr><th colspan="3">2 input AND gate</th></tr>
<tr><th>A</th><th>B</th><th>A.B / F</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

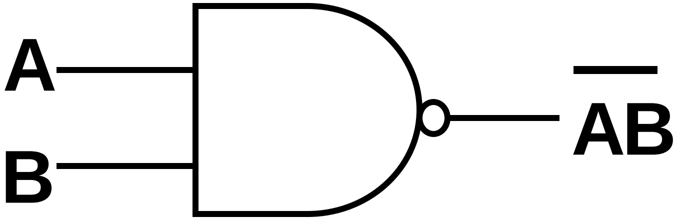
❖ Summary of Logic gates :

<table>
<tr><th>No</th><th>Name of gate</th><th>Logic Diagram</th><th>Truth Table</th></tr>
<tr><td>2</td><td>OR</td><td>
</td><td>
<table>
<tr><th colspan="3">2 input OR gate</th></tr>
<tr><th>A</th><th>B</th><th>A+B / F</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

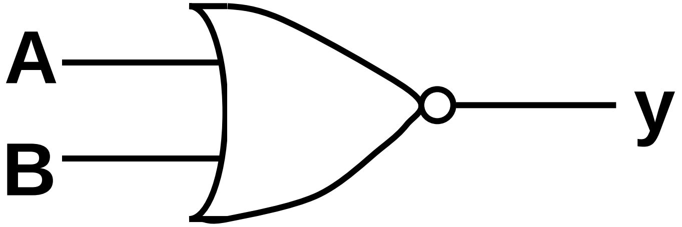
❖ Summary of Logic gates :

<table>
<tr><th>No</th><th>Name of gate</th><th>Logic Diagram</th><th>Truth Table</th></tr>
<tr><td>3</td><td>NOT</td><td>
</td><td>
<table>
<tr><th colspan="2">NOT gate</th></tr>
<tr><th>A</th><th>A’/$\bar{A}$</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

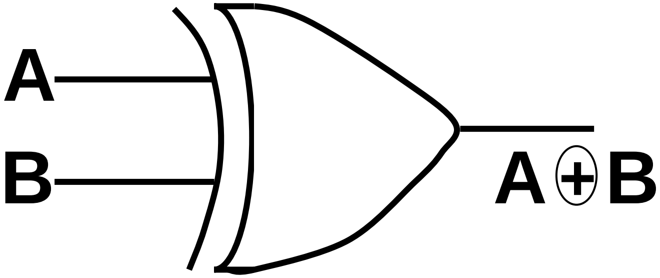
❖ Summary of Logic gates :

<table>
<tr><th>No</th><th>Name of gate</th><th>Logic Diagram</th><th>Truth Table</th></tr>
<tr><td>4</td><td>NAND</td><td>
</td><td>
<table>
<tr><th colspan="3">2 input NAND gate</th></tr>
<tr><th>A</th><th>B</th><th>$\overline{A.B}$</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

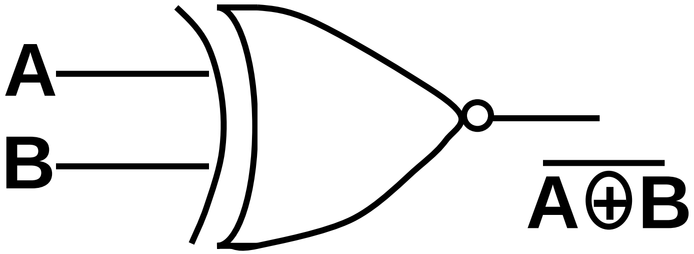
❖ Summary of Logic gates :

<table>
<tr><th>No</th><th>Name of gate</th><th>Logic Diagram</th><th>Truth Table</th></tr>
<tr><td>5</td><td>NOR</td><td>
</td><td>
<table>
<tr><th colspan="3">2 input NOR gate</th></tr>
<tr><th>A</th><th>B</th><th>$\overline{A+B}$</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

❖ Summary of Logic gates :

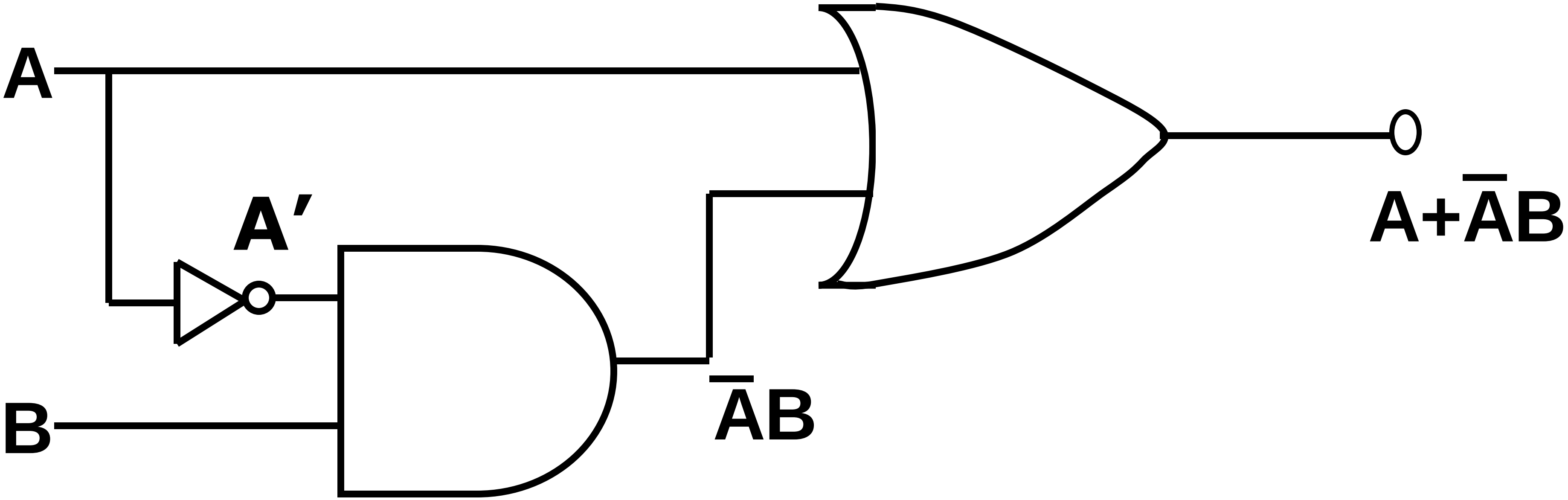
<table>
<tr><th>No</th><th>Name of gate</th><th>Logic Diagram</th><th>Truth Table</th></tr>
<tr><td>6</td><td>EX
-
OR</td><td>
</td><td>
<table>
<tr><th colspan="3">2 input Ex OR gate</th></tr>
<tr><th>A</th><th>B</th><th>A ⊕ B</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td></tr>
</table>
</td></tr>
</table>

❖ Summary of Logic gates :

<table>
<tr><th>No</th><th>Name of gate</th><th>Logic Diagram</th><th>Truth Table</th></tr>
<tr><td>7</td><td>EX - NOR</td><td>
</td><td>
<table>
<tr><th colspan="3">2 input Ex NOR</th></tr>
<tr><th>A</th><th>B</th><th>$\overline{A \oplus B}$</th></tr>
<tr><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>
</td></tr>
</table>

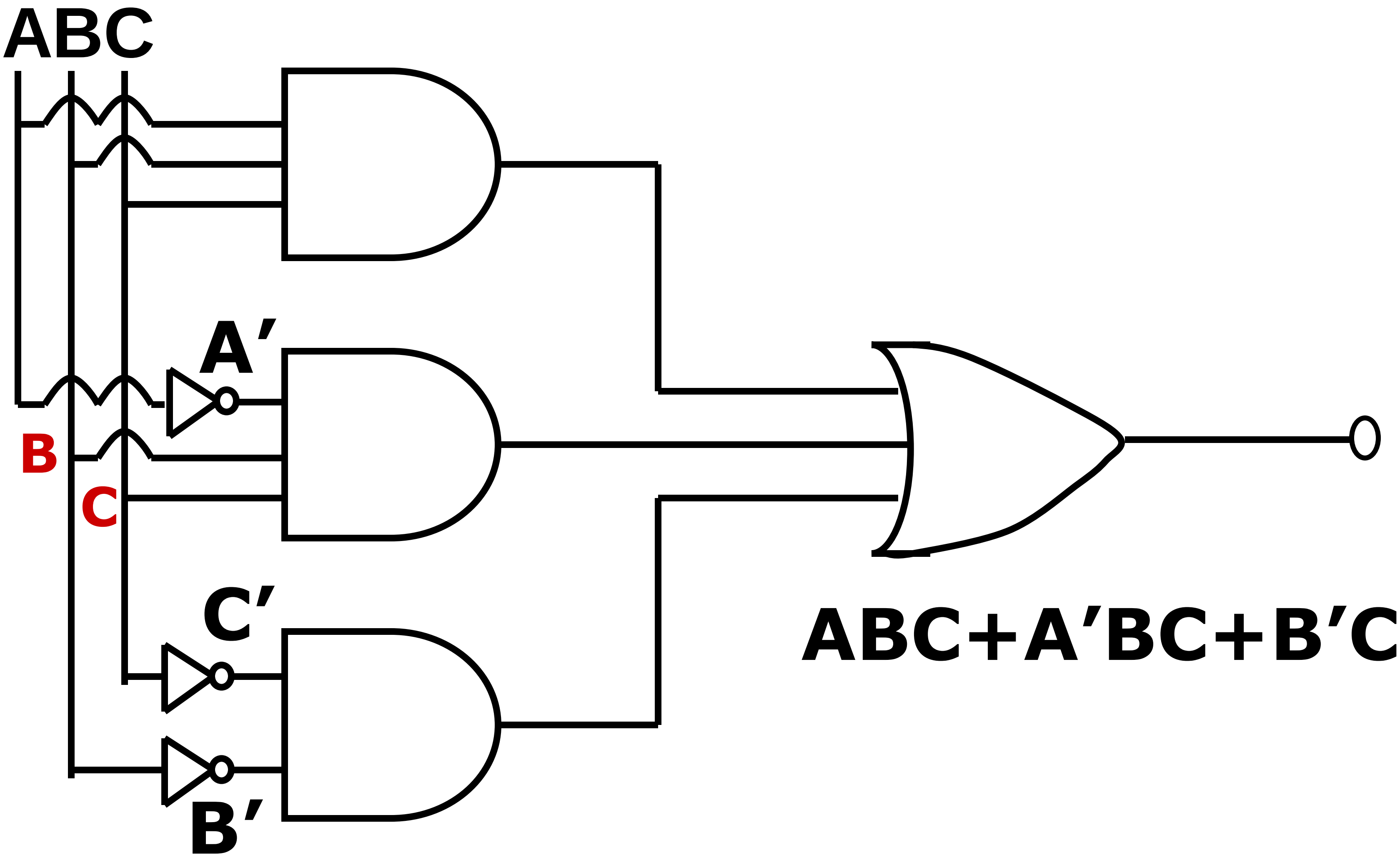
❖ Implementation Of Boolean Equation Using Logic Gates :

1) F=A+A’B



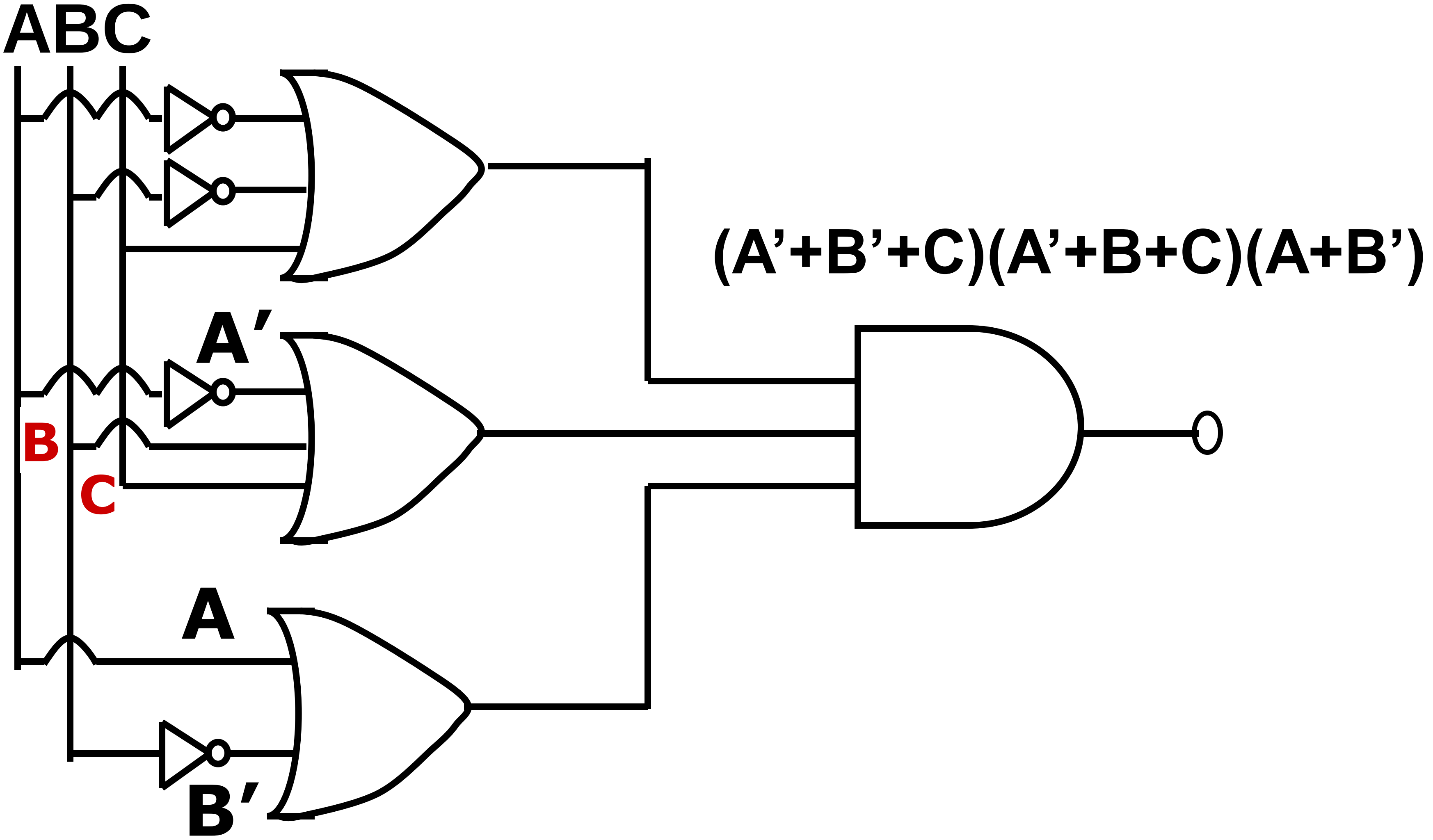
❖ Implementation Of Boolean Equation Using Logic Gates :

2) F= ABC+A’BC+B’C’



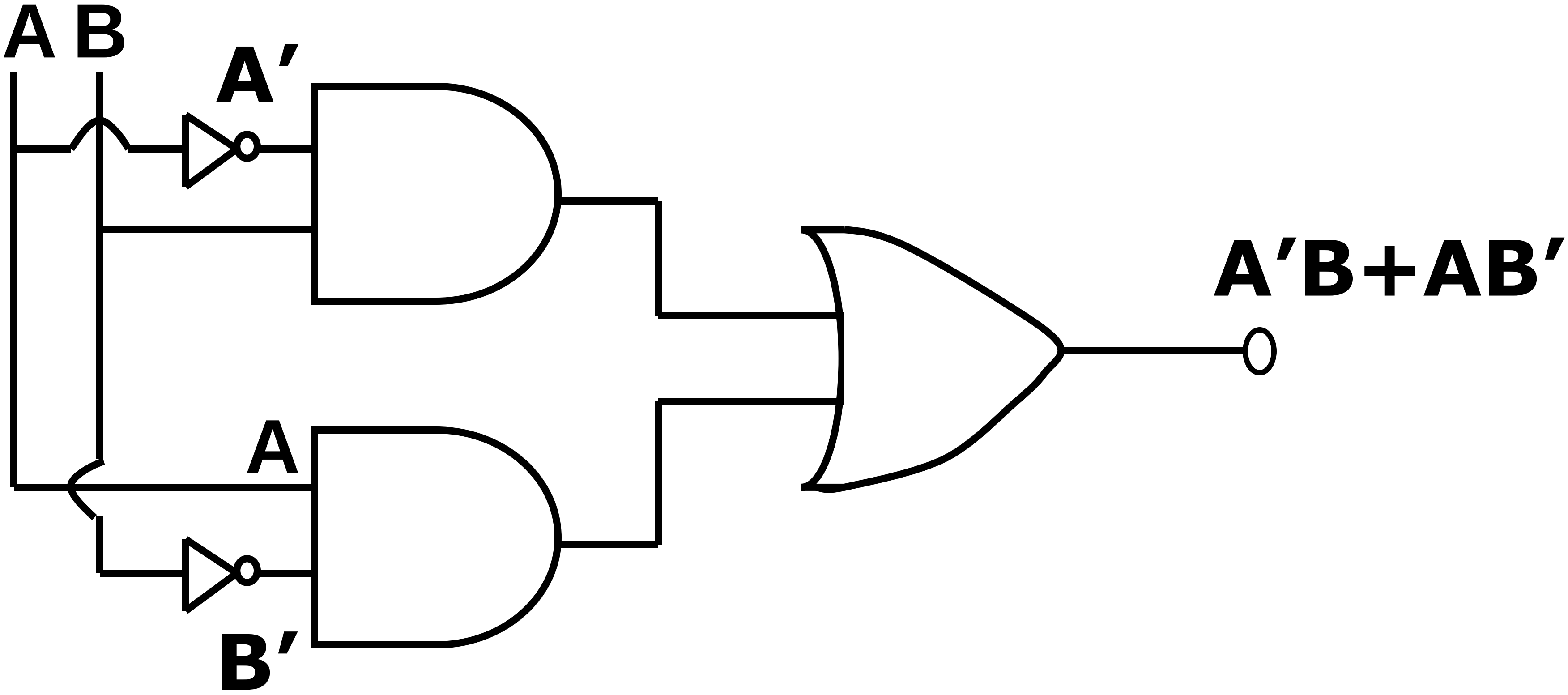
❖ Implementation Of Boolean Equation Using Logic Gates :

3) F= (A’+B’+C)(A’+B+C)(A+B’)



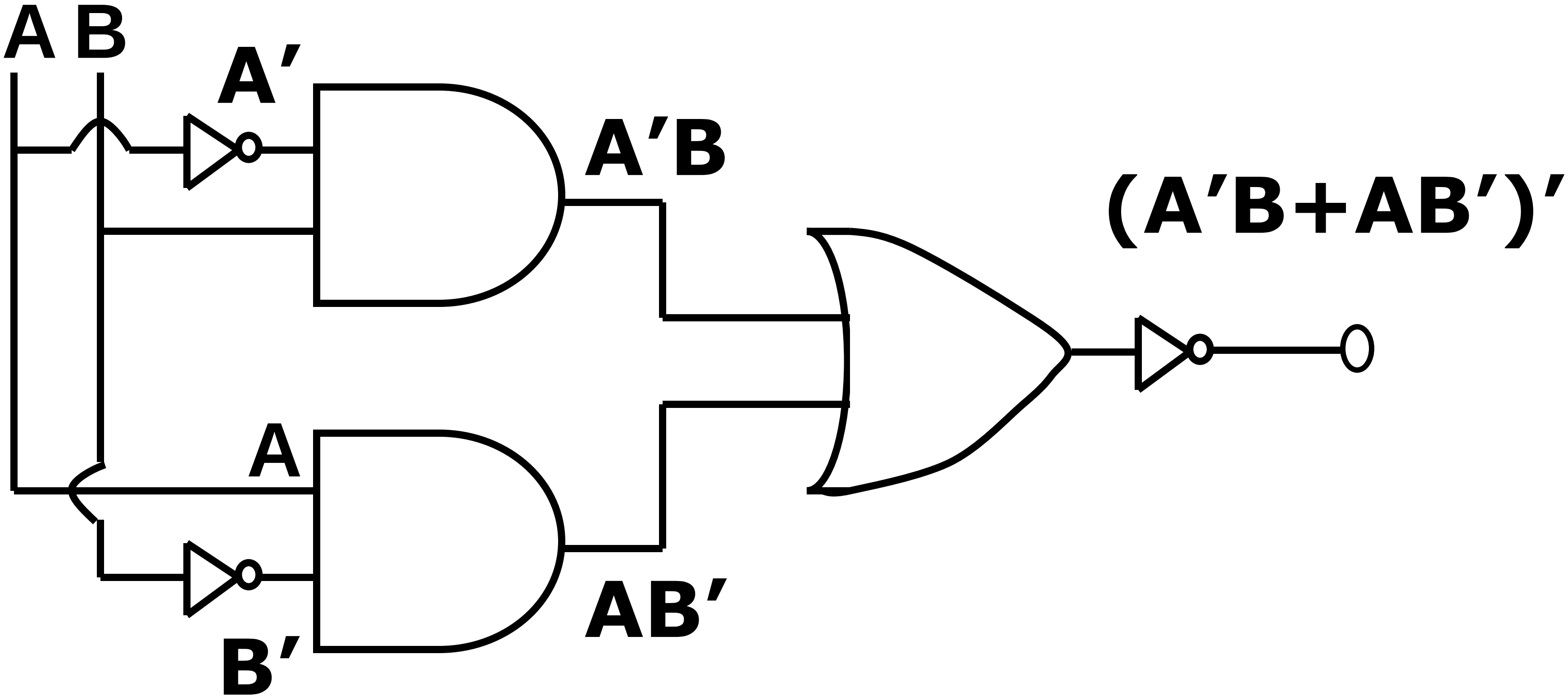
❖ Implementation Of Boolean Equation Using Logic Gates :

4) F=A’B+AB’



❖ Implementation Of Boolean Equation Using Logic Gates :

5) F=(A’B+AB’)’



Boolean Logic :

- Boolean Logic is a complete system for logical operations, used in many systems.
- It was named after George Boole, who first defined an algebraic system of logic in the mid 19th century.
- Boolean logic has many applications in electronics, computer hardware and software, and is the base of all modern digital electronics.

Postulates of Boolean Algebra :

- Postulates of Boolean Algebra means it require no proof.

1) Commutative Postulates

$$A+B=B+A$$

$$A.B=B.A$$

2) Identity Postulates

$$A+0=A$$

$$A.1=A$$

$$A+A'=1$$

$$A.A'=0$$

Postulates of Boolean Algebra :

- Postulates of Boolean Algebra means it require no proof.

3) Distributive Postulates

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

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

Properties of Boolean Algebra :

1) Commutative Property :

- Boolean addition is commutative where order of variable can be interchange.

$$A+B=B+A$$

- This is a OR property, where sequence of A and B can be interchange.

Properties of Boolean Algebra :

2) Associative Property :

- In OR property grouped of variable can be order any way.

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

- Here in this example left hand side group of variables B and C is made and A is added to it. If group of A and B is made and C is added to it there is no change in the result.
- Associative property is applicable to AND operation :

$$A.(B.C)=(A.B).C$$

Properties of Boolean Algebra :

3) Distributive Property :

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

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

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

$$A+BC=A.1+B.C$$

Properties of Boolean Algebra :

- Absorption Property / Absorption Theorems:

$A+0=A$

$A.1=A$

$A+A=A$

$A.A=A$

$A+1=1$

$A.0=0$

$A+\bar{A}=1$

$A-\bar{A}=0$

$\bar{A}=A$

Properties of Boolean Algebra :

Absorption Property / Absorption Theorems:

1) A.(A+B)=A

L.H.S=A.A+A.B

=A+AB || A=A.A

=A(1+B)

=A.1 ||1=1+B

=A

L.H.S=R.H.S

Properties of Boolean Algebra :

Absorption Property / Absorption Theorems:

2) $A+AB=A$

$$
\begin{aligned}
\text{L.H.S} &= A+AB \\
&= A(1+B) \qquad || 1+B=1 \\
&= A.1 \\
&= A
\end{aligned}
$$

L.H.S=R.H.S

Properties of Boolean Algebra :

Absorption Property / Absorption Theorems:

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

L.H.S $=A+B$

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

$=1.(A+B)$

$=A+B$

L.H.S=R.H.S

Properties of Boolean Algebra :

Absorption Property / Absorption Theorems:

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

$L.H.S=A.\bar{A}+AB$

$=0+AB$ || $A.\bar{A}=0$

$=AB$

$L.H.S=R.H.S$

Properties of Boolean Algebra :

Absorption Property / Absorption Theorems:

5) $X+X.Y=X$

$=X+X.Y$

$=X(1+Y)$ || $X.X=1$

$=X.1$

$=X$

L.H.S=R.H.S

Properties of Boolean Algebra :

Absorption Property / Absorption Theorems:

6) $X.(X+Y)=X$

L.H.S$=X.(X+Y)$

$=X.X+X.Y$ || remove bracket

$=X+X.Y$

$=X(1.Y)$ ||$(1+Y)=1$

$=X.1$

$=X$

L.H.S=R.H.S

CANONICAL OR STANDARD FORMS:

- There are two ways to represent Boolean function, one is standard (canonical) Sum Of Product form, and another is the standard Product Of Sum.

SUM OF PRODUCT : (SOP)

- It is an expression in which one or more product terms are logically added.
- Sum of Product is also known as Product of Minterms.
- Minterms and Maxterms are complement each other.

SUM OF PRODUCT : (SOP)

- ❑ **Min terms:**
- ❖ Sum Of Product is also known as Minterms.
- ❖ Each minterm obtained from an AND term of the n variable, with each variable being primed if the bit of binary number is 0 and unprimed if a 1.
- ❖ Symbol of minterm is m_j where j=0,1,2,… The three variable Minterms are shown in the table.

❑ Min terms:

Input			Min Terms	Min Term
X	Y	Z	(Standard Product Terms)	Designation
0	0	0	X’ Y’ Z’	m_0
0	0	1	X’ Y’ Z	m_1
0	1	0	X ’Y Z’	m_2
0	1	1	X ’Y Z	m_3
1	0	0	X Y’ Z’	m_4
1	0	1	X Y’ Z	m_5
1	1	0	X Y Z’	m_6
1	1	1	X Y Z	m_7

CANONICAL OR STANDARD FORMS:

PRODUCT OF SUM: (POS)

- It is an expression in which one or more product terms are logically multiplied.
- Product Of Sum is also known as Product of Maxterms.
- Minterms and Maxterms are complement each other.

PRODUCT OF SUM: (POS)

❑ Max terms:

- ❖ Product Of Sum is also known as Maxterms.
- ❖ Each maxterm obtained from an OR term of the n variable, with each variable being primed if the bit of binary number is 1 and unprimed if a 0.
- ❖ Symbol of maxterm is M_j where j=0,1,2,… The three variable Minterms are shown in the table.

❑ Max terms:

Input			Max Terms	Max Term
X	**Y**	**Z**	(Standard Product Terms)	Designation
0	0	0	$X+Y+Z$	M_0
0	0	1	$X+Y+Z'$	M_1
0	1	0	$X+Y'+Z$	M_2
0	1	1	$X+Y'+Z'$	M_3
1	0	0	$X'+Y+Z$	M_4
1	0	1	$X'+Y+Z'$	M_5
1	1	0	$X'+Y'+Z$	M_6
1	1	1	$X'+Y'+Z'$	M_7

DE-MORGAN'S THEOREMS :

- DE-MORGAN was great logician and mathematician.
- He discovered the two important theorems as follows:

- Theorem-1 : The complement of a sum equals to the products of the complements.

$$\overline{X+Y}=\overline{X}.\overline{Y}$$

- Theorem-2 : The complement of a product equals the sum of the complements.

$$\overline{X.Y}=\overline{X}+\overline{Y}$$

DE-MORGAN'S THEOREMS :

- Proof of De-Morgan's theorem - 1
- **Theorem-1 :**

$$\overline{X+Y}=\overline{X}.\overline{Y}$$

X	Y	$\overline{X}$	$\overline{Y}$	$X+Y$	$\overline{X+Y}$	$\overline{X}.\overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

DE-MORGAN'S THEOREMS :

- ❑ Proof of De-Morgan's theorem - 1
- ❑ **Theorem-2 :**

$$\overline{X.Y}=\overline{X}+\overline{Y}$$

X	Y	$\overline{X}$	$\overline{Y}$	$X.Y$	$\overline{XY}$	$\overline{X}+\overline{Y}$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

KARNAUGH MAP (K-MAP) :

- The boolean function may be simplified using algebra method, but this method is sometimes difficult because it lacks specific rules.
- This method is also known as a **KARNAUG MAP** or **Veitch Diagram**.
- The map is a diagram made of squares and each square represent one minterm.

❑ How To plot a KARNAUGH MAP?

❑ **K- MAP for 2 variables :**

X \ Y	0	1
0	X’Y’	X’Y
1	XY’	XY

❑ K- MAP for 2 variables :

- For any square see the variable in the both the row and the column.
- For first square of the 1st row, variables are X’ and Y’, and hence it will represent the product term X’Y’.
- For the 2nd square of the 1st row, the variables are X’ and Y, so it represents X’Y.
  - 0 and 1 are written at the top of the map shown in the above figure. They indicate 0 and 1 for variable.
  - It means 0 represents variable in complemented from (Y’) and 1 represents variable in uncomplemented from (Y).
  - Similarly on the extreme left side of the map 0 and 1 are written and they represents X’ and X respectively.

❑ K- MAP for 2 variables :

(1) F=X’Y+XY’

X \ Y	0	1
0		1
1	1	

❑ K- MAP for 3 variables :

- A three variable K-Map has eight square or cells and each square represent different product terms.
- For example: **X’Y’Z’, X’Y’Z, X’YZ, X’YZ’, XY’Z’, XY’Z, XYZ, XYZ’.**

X \ YZ	00	01	11	10
0	**X’Y’Z’**	**X’Y’Z**	**X’YZ**	**X’YZ’**
1	**XY’Z’**	**XY’Z**	**XYZ**	**XYZ’**

K- MAP for 4 variables :

- A four variable K-Map has sixteen square or cells and each square represent different product terms.
- For example: **W’X’Y’Z’,…… WX’YZ’.**

YZ WX	00	01	11	10
0	X’Y’Z’	X’Y’Z	X’YZ	X’YZ’
1	XY’Z’	XY’Z	XYZ	XYZ’

❑ Simplification using K-MAP :

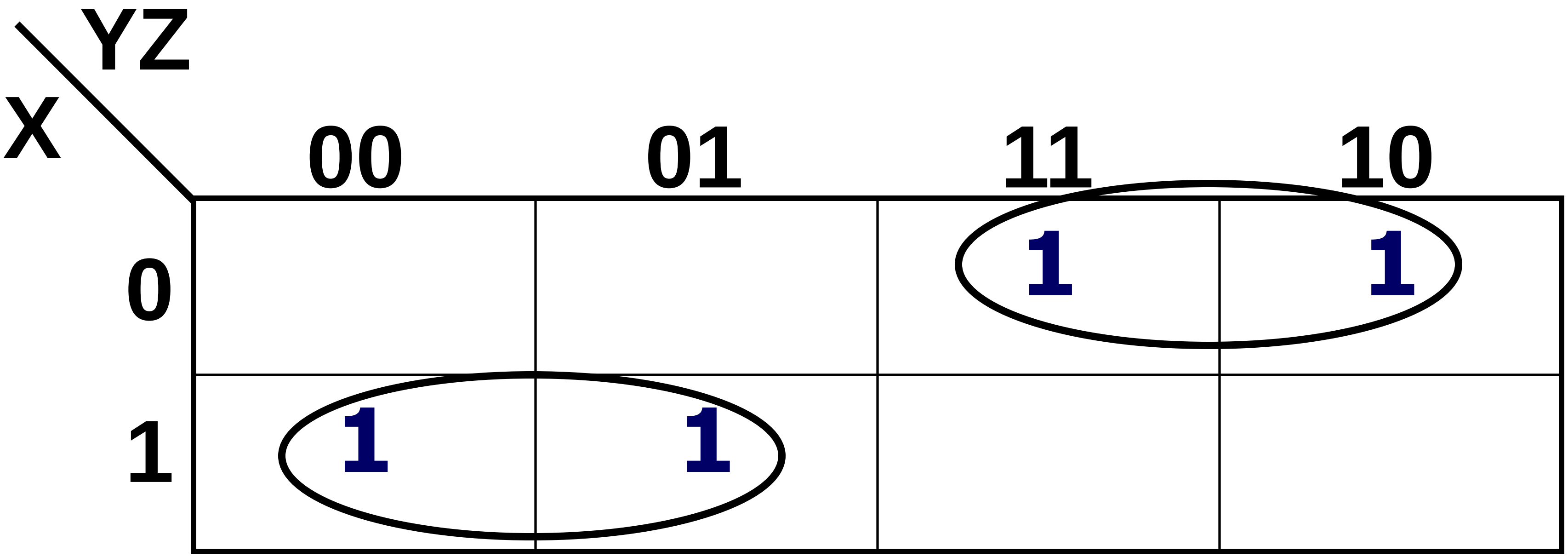
❖ **[A] $F=(X'YZ)+(X'YZ')+(XY'Z')+(XY'Z)$**

$$= X'Y(Z+Z')+XY'(Z'+Z)$$

$$= (X'Y+XY')+(Z+Z')(Z'+Z)$$

$$= (X'Y+XY').1$$

$$= X'Y+XY'$$



X \ YZ	00	01	11	10
0			1	1
1	1	1		

❑ Simplification using K-MAP :

- **Simplified the equation**

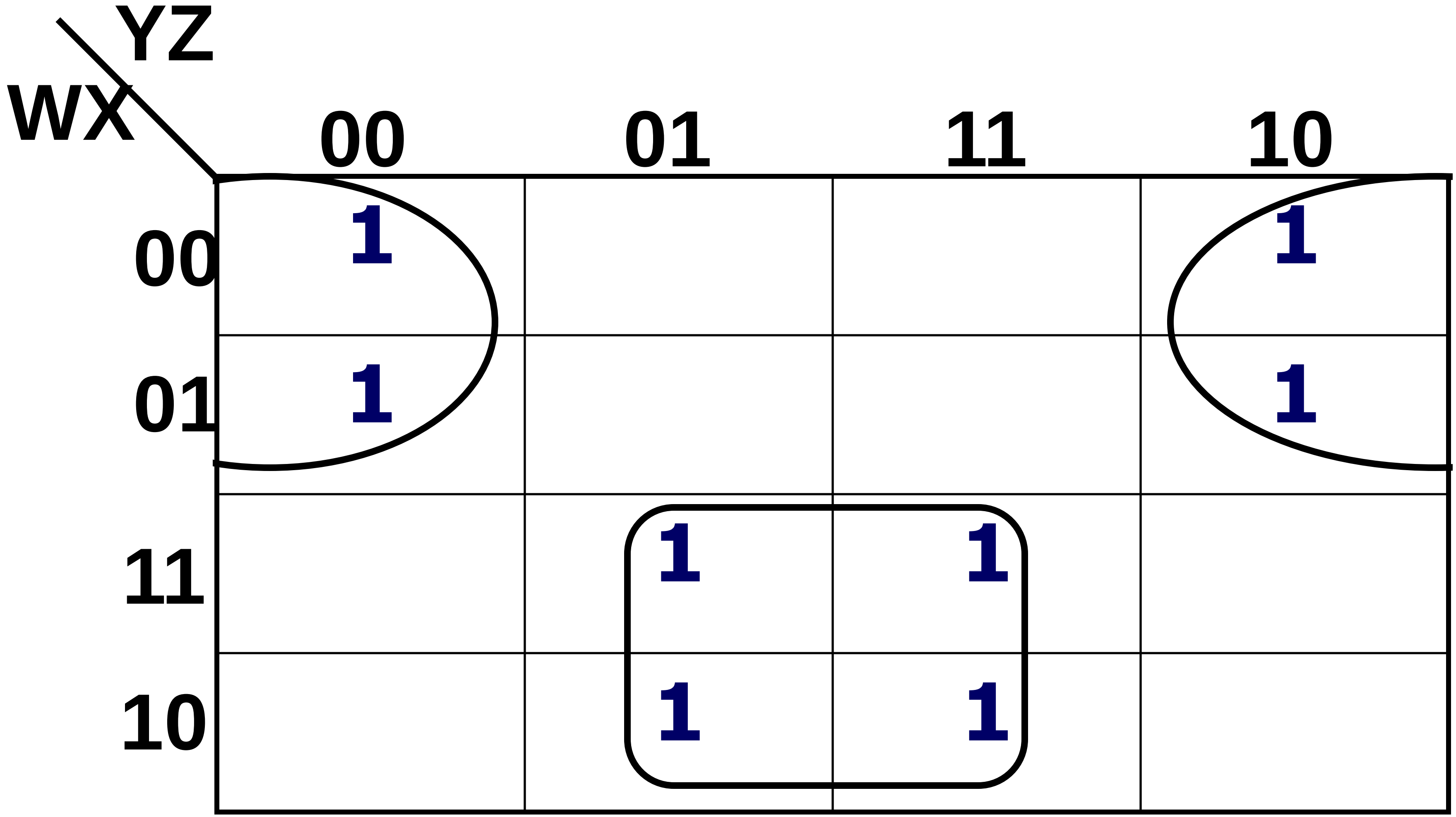
F(W,X,Y,Z)= Σ(0,2,4,6,9,11,13,15)

YZ WX	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

❑ Simplification using K-MAP :

- Simplified the equation

  F(W,X,Y,Z)= Σ(0,2,4,6,9,11,13,15)



WX \ YZ	00	01	11	10
00	1			1
01	1			1
11		1	1	
10		1	1	

❑ Simplification using K-MAP :

❖ **Simplified the equation**

F(W,X,Y,Z)= Σ(0,1,2,5,8,9,10)

WX \ YZ	00	01	11	10
00	1	1		1
01		1		
11				
10	1	1		1

❑ Don’t care condition :

- In K-MAP every cell represent a minterm or maxterm.
- Sometime may be possible that any 1s or 0s does not matter.
- We say that don’t care what the function output is to be for this minterm.
- Minterms that may produce either 0 or 1 for the function are said to be **do not care** and are marked with × in the map.
- The don’t care condition can be used to provide further simplification of the algebraic expression.

❑ Simplification using Don’t Care condition :

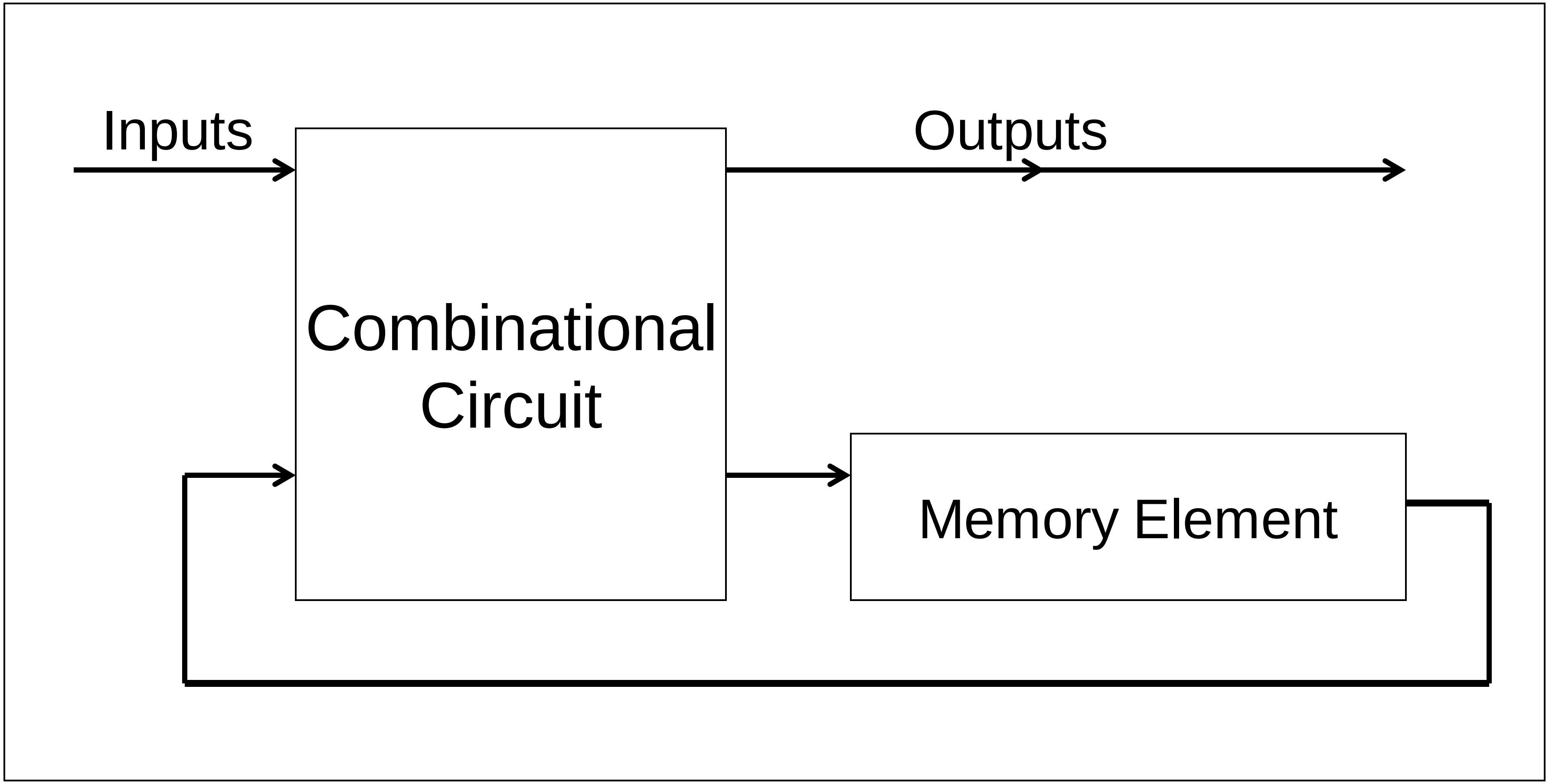
- Simplified the equation $F(W,X,Y,Z)=\Sigma(4,5,6,8,9,12,13)+d(3,7,10,11,15)$

WX \ YZ	00	01	11	10
00	0	1	× 3	2
01	1 4	1 5	× 7	1 6
11	1 12	1 13	× 15	14
10	1 8	1 9	× 11	× 10

❑ What is Combinational Circuits? :

- ❖ It is logical circuits, the output at any time depends on the logic levels at the input at that instant only.
- ❖ It does not depend on the past condition.
- ❖ A combinational circuit transforms binary information from the given output data to the required output data.

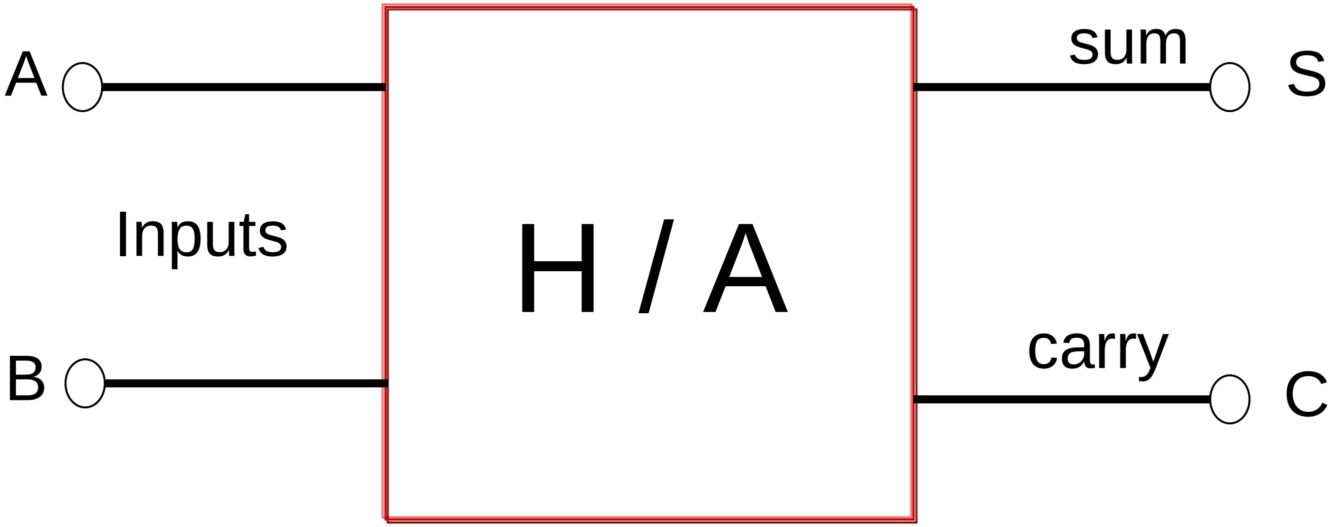
❑ Combinational Circuits:



Block diagram of Combinational Circuit

❑ Half Adder :

- A half adder is a combinational circuit adds two binary bits.
- Block diagram of half adder is as given below.



Block diagram of Half Adder

❑ Half Adder :

- There are two input terminals which are marked as A and B.
- Binary numbers the sum of which has to be made are applied here.
- There are two output terminals. One terminal is for sum and the other is the carry bit C.
- Truth table of half adder is shown below.

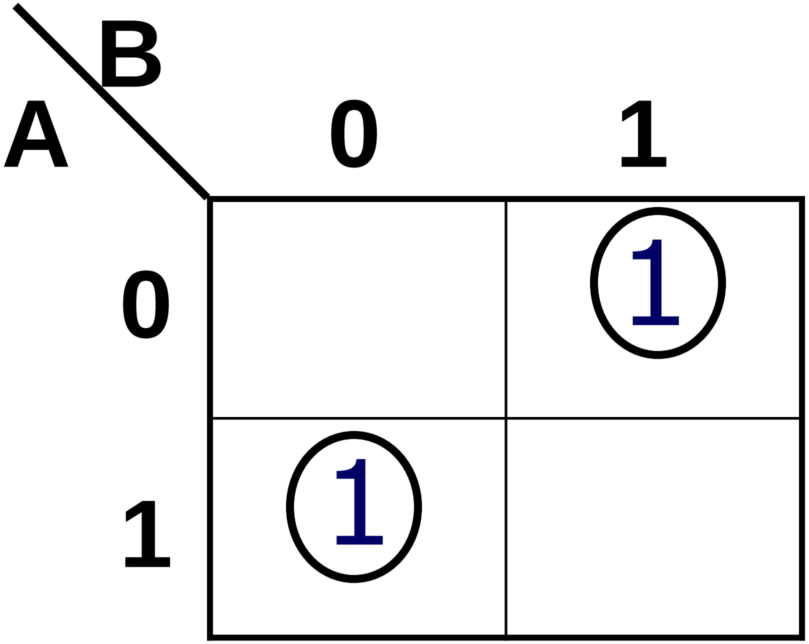
❑ Half Adder’ truth table :

Input		Output	
A	**B**	**SUM**	**Carry**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

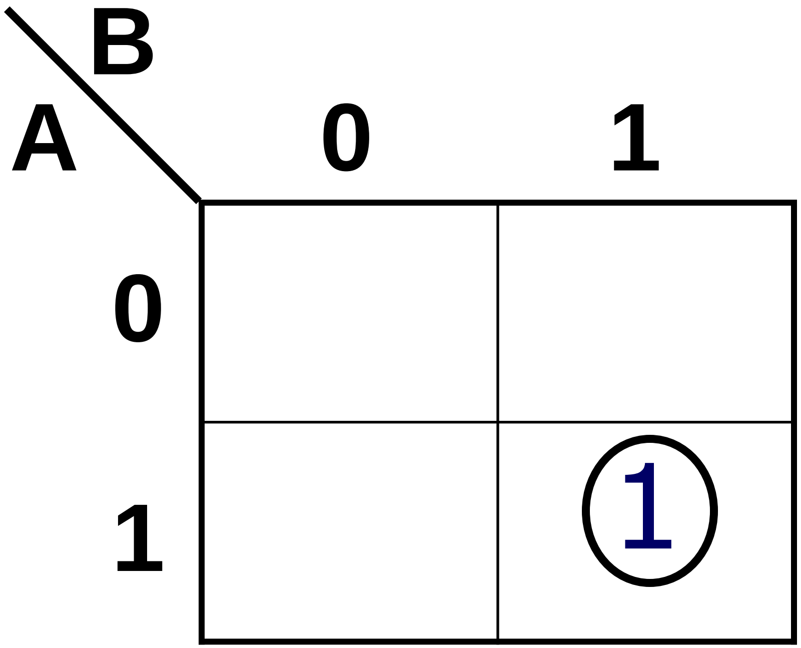
$S=A'B+AB'$

$C=AB$

❑ K-MAP for Half Adder:



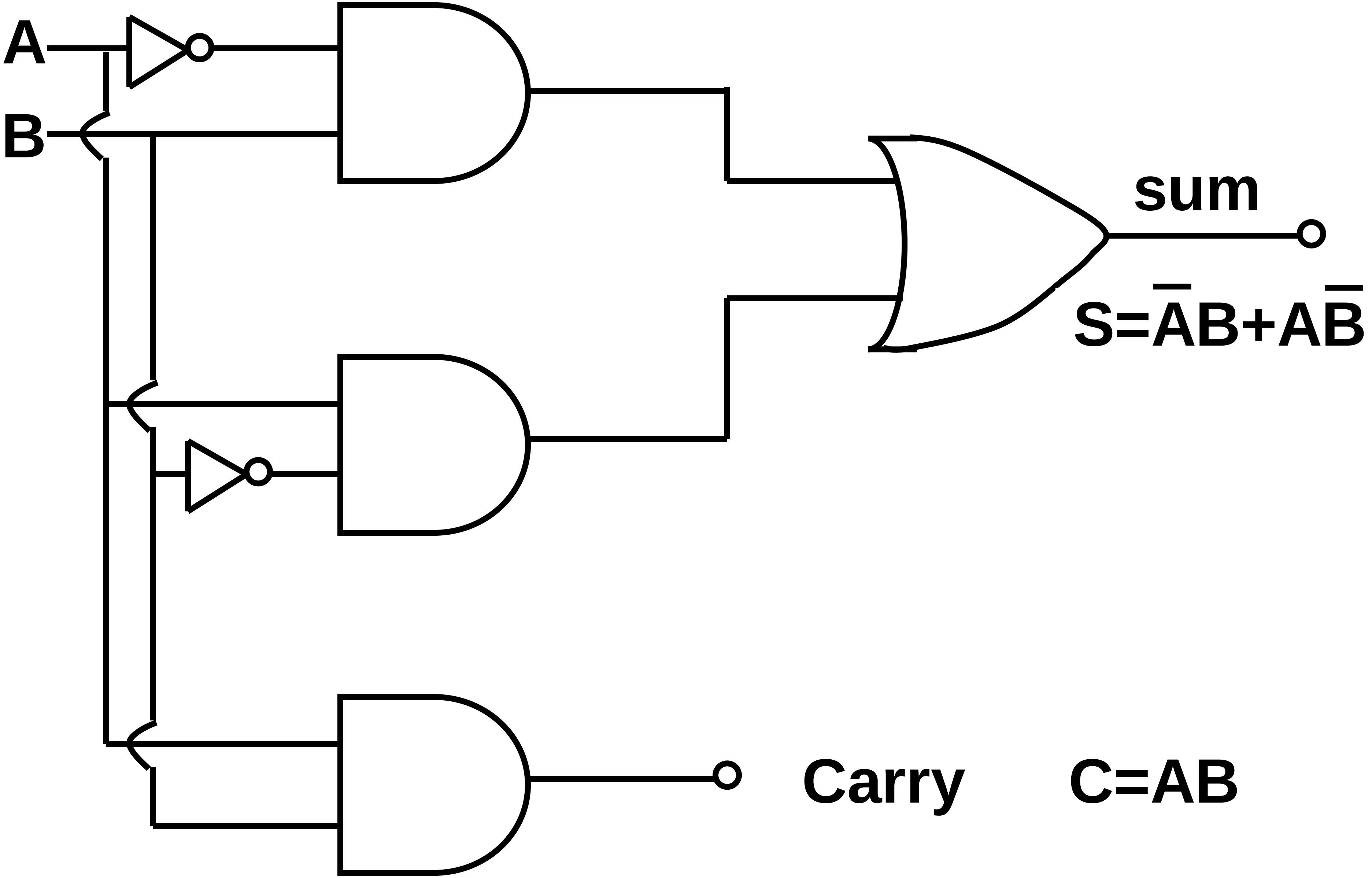
A \ B	0	1
0		1
1	1	



A \ B	0	1
0		
1		1

❖ From the truth table let us construct the K-MAP to find Boolean expression for the sum S and carry C.

□ K-MAP for Half Adder:



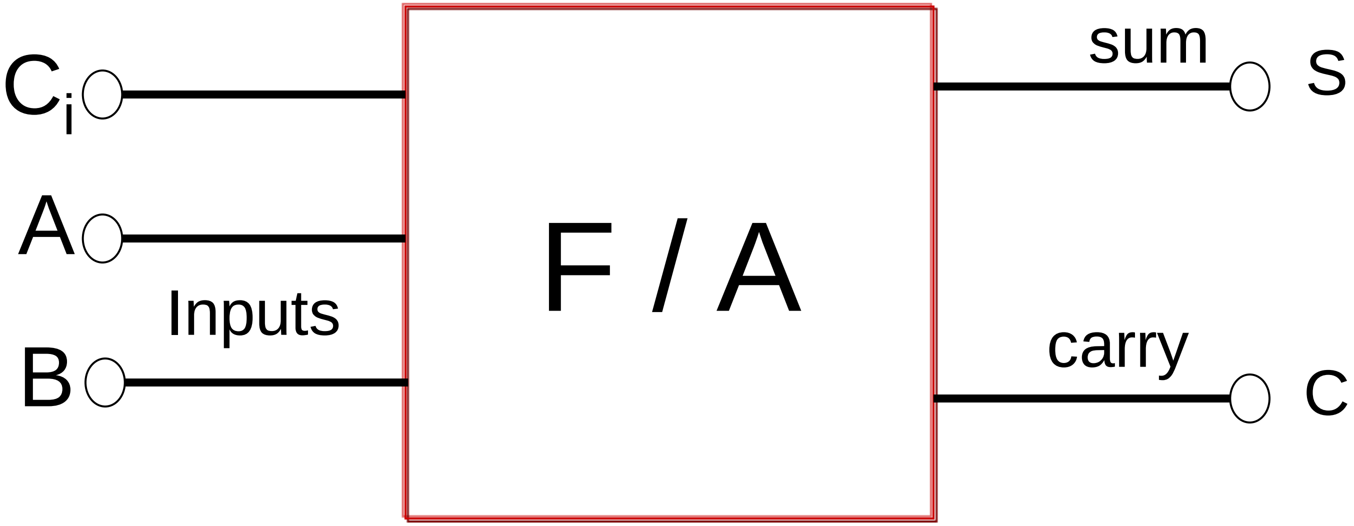
$S=\bar{A}B+A\bar{B}$

$C=AB$

❑ Full Adder :

- A full adder is a combinational circuit that performs the arithmetic sum of three input bits.
- It consists three inputs and two outputs.
- When we want to add two binary numbers each having two or more bits the LSB (Least Significant Bit) can be added by using a half adder.
- Block diagram of full adder is as given below:

❑ Full Adder Diagram :



Block diagram of Full Adder

- In this there are three input terminal. One output is C_i which is carry from the previous stage.
- A and B are two input terminals. There are two output terminals. One is final sum S and the other is final carry C.

❑ Full Adder Diagram :

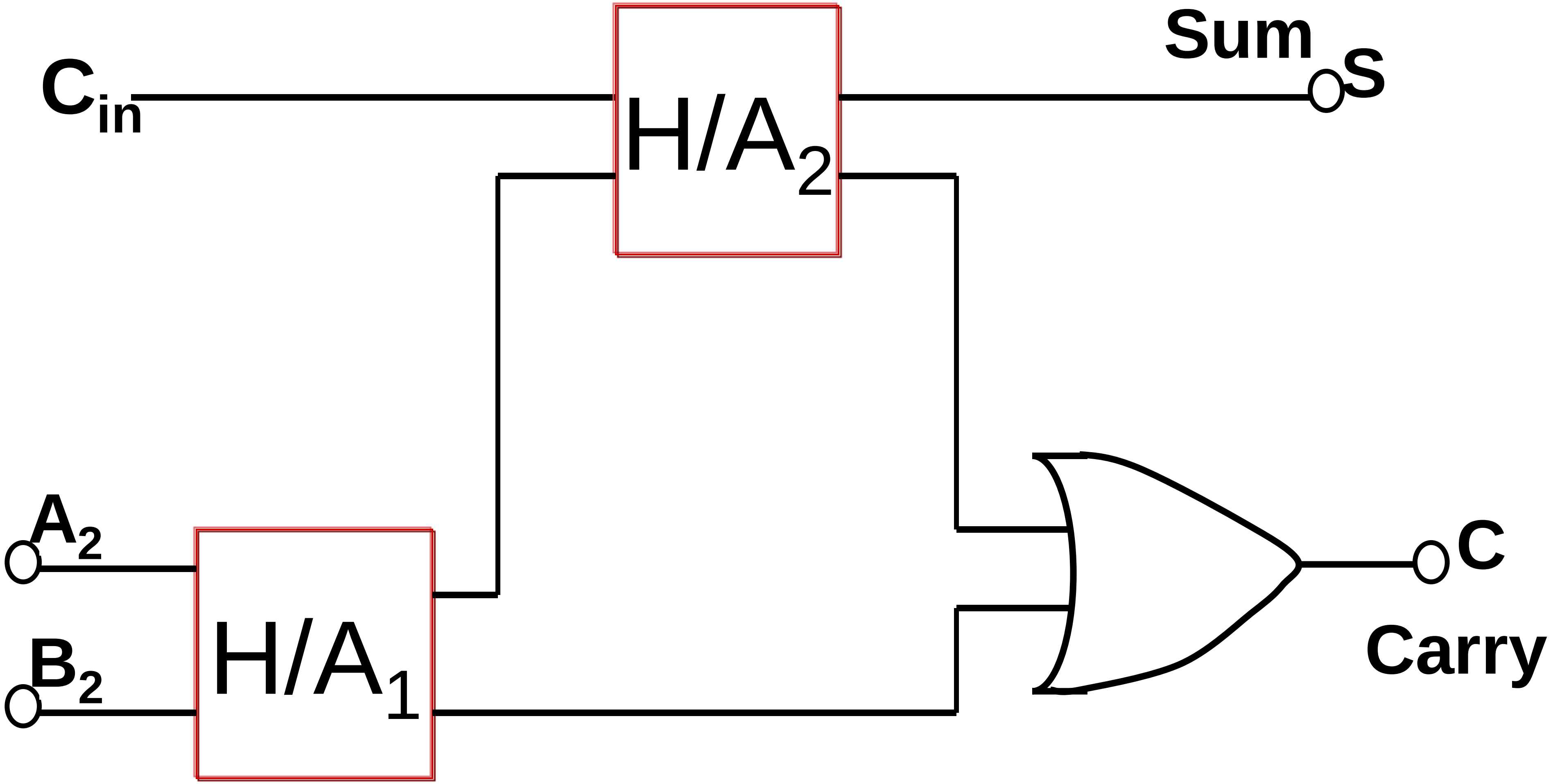
Input			Output	
A	**B**	Carry from Previous stage **C**	Final Sum **S**	Final Carry **C**
0	0	0	0	0
0	1	0	1	0
1	0	0	1	0
1	1	0	0	1
0	0	1	1	0
0	1	1	0	1
1	0	1	0	1
1	1	1	1	1

❑ Full Adder Diagram :

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

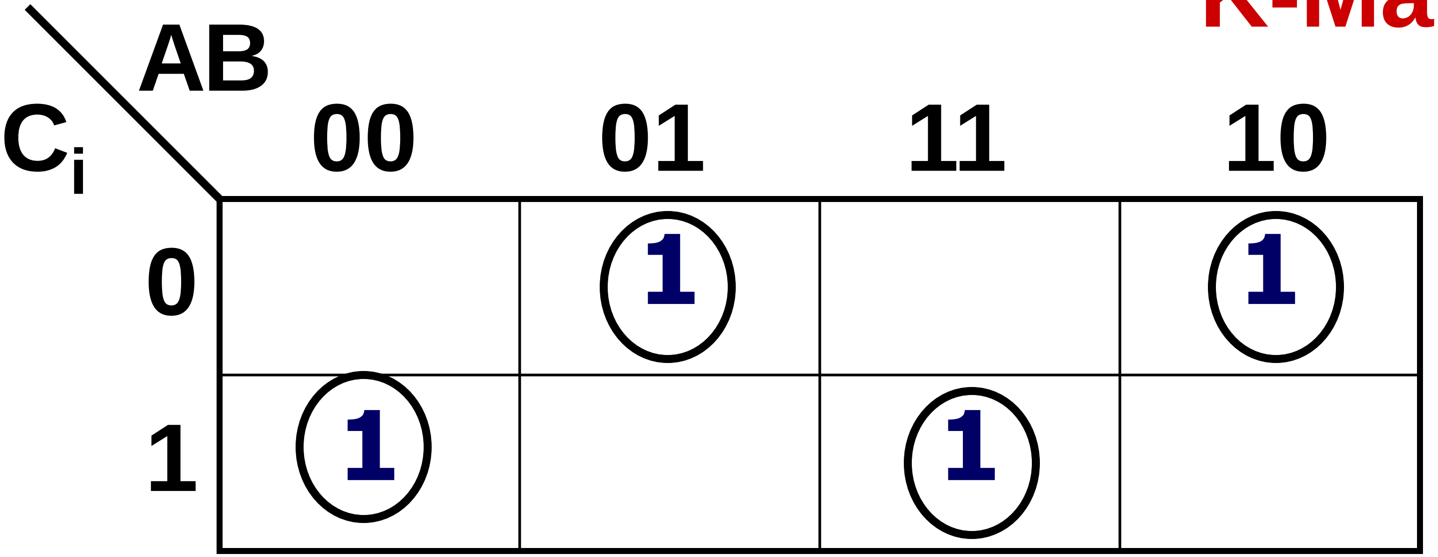
$C=AB\bar{C}+\bar{A}BC+A\bar{B}C+\bar{A}\bar{B}\bar{C}$

❑ Full Adder circuit using two half adders:



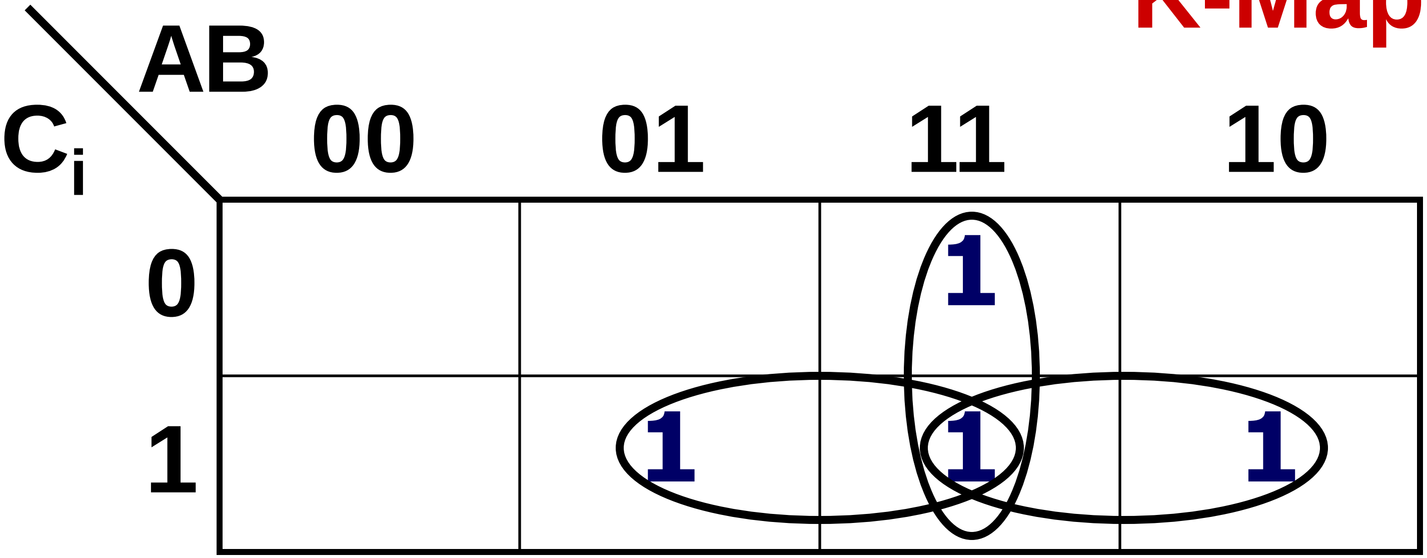
❑ K-MAP for Full Adder:

K-Map for Sum



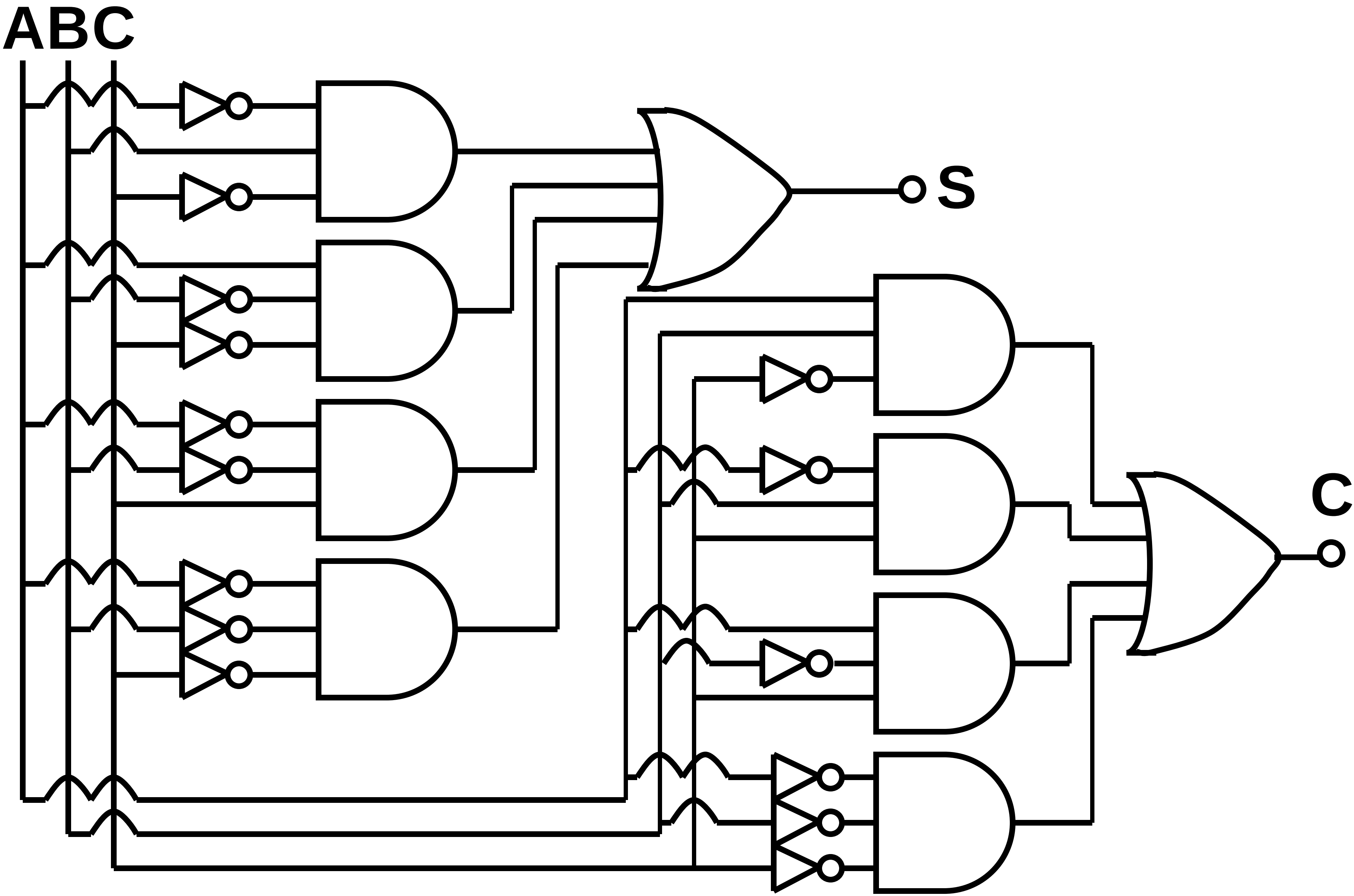
C_i \ AB	00	01	11	10
0		1		1
1	1		1	

K-Map for Carry



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

❑ K-MAP for Full Adder:



❑ Comparison between Half Adder and Full Adder

Half Adder	Full Adder
1. It is used for 2 bit addition.	**1.** It is used for Multi bit addition.
2. One Ex-OR/OR gate and one AND gate are used.	**2.** Two Ex-OR/OR gates and Multiple AND gates are used.
3. Output is the sum of two signals.	**3.** Output is the sum of three signals.
4. Circuit is simple.	**4.** Circuit is complicated.

❑ Comparison between Half Adder and Full Adder

Half Adder	Full Adder
5. There are two input and two output terminal.	**5.** There are three input and two output terminal.
6. Two half adder makes one full adder.	**6.** Two full adder does not make one half adder.

❑ BCD Adder :

- BCD stands for Binary Coded Decimal.
- A BCD adder is a circuit which adds two BCD digits in parallel and produces the sum in BCD, means from 0 to 9 Decimal numbers are represent in BCD form with 4 Binary digit.

❑ BCD Adder :

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

BCD Adder :

- In BCD addition when two BCD numbers get addition, after addition if value more than 9 then it will selected by adding 6 to answer.
- Example: If we make sum of decimal digit 7 and 7 in binary form, it will be

$$\begin{array}{ccccc} & & \mathbf{1} & \mathbf{1} & \\ & 0 & 1 & 1 & 1 \\ + & 0 & 1 & 1 & 1 \\ \hline & 1 & 1 & 1 & 0 \end{array}$$

❑ BCD Adder :

- The result is 1110 equivalent 14 of decimal which is not valid BCD form.
- In order to get the answer in the BCD form we have to skip six forbidden group.
- To be this we have to add 0110 to the answer, so in example let us add 0110 to the answer.

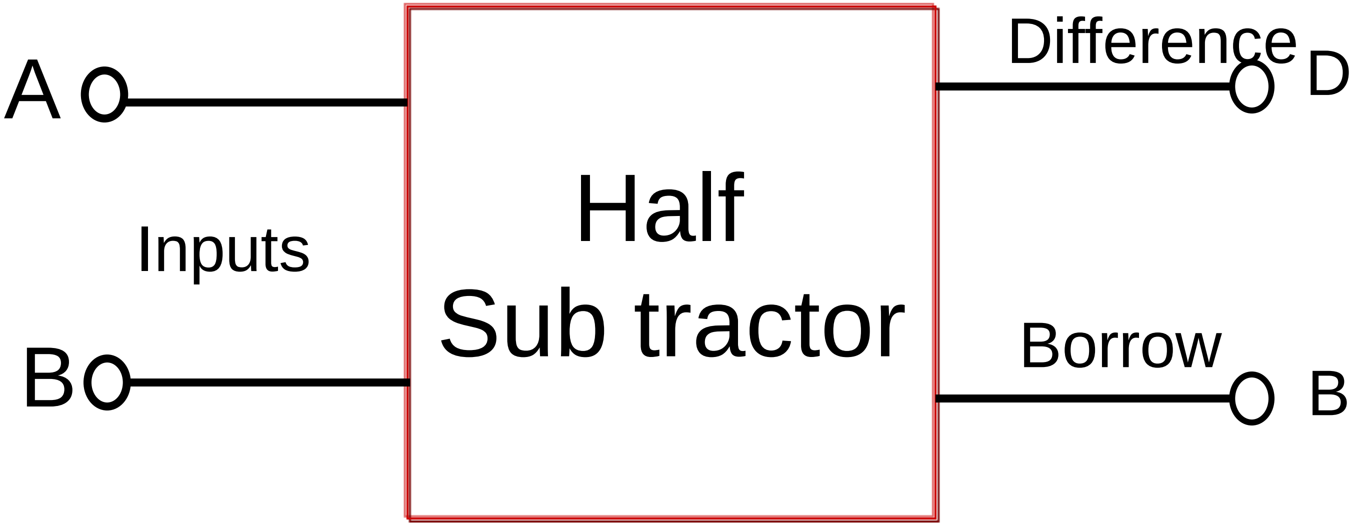
$$\begin{array}{ccccc} & \mathbf{1} & \mathbf{1} & & \\ & 1 & 1 & 1 & 0 \\ \mathbf{+} & 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 1 & 0 & 0 \end{array}$$

Which is again
group line **0001=1**
and **0100=4**

Correct BCD for is **14 (00010100)**

❑ Half Sub tractor :

❖ Binary sub tractor can be made using half sub tractor. Block diagram is shown below:



Block diagram of Half Sub tractor

❑ Half Sub tractor :

- There are two input terminals A and B bits to be subtracted are applied here.
- There are two output terminals. One is for the difference signal and the other is for borrow signal. Truth table is as given below:

❑ Half Sub tractor’ Truth table:

Input		Output	
A	**B**	Difference **D**	Borrow **B**
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

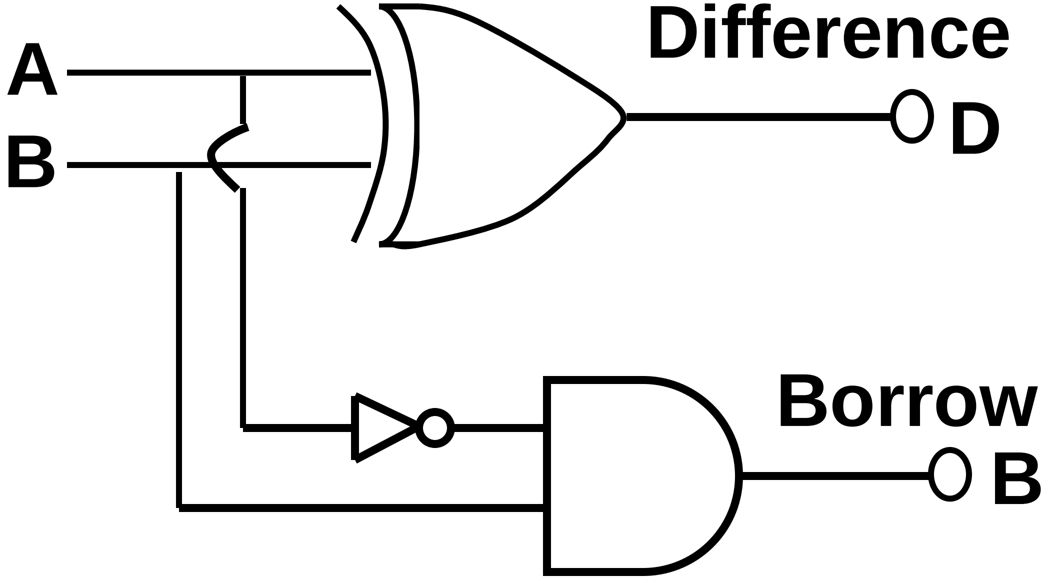
- D=A′B+AB′
- B=A′B

❑ Half Sub tractor' Truth table:

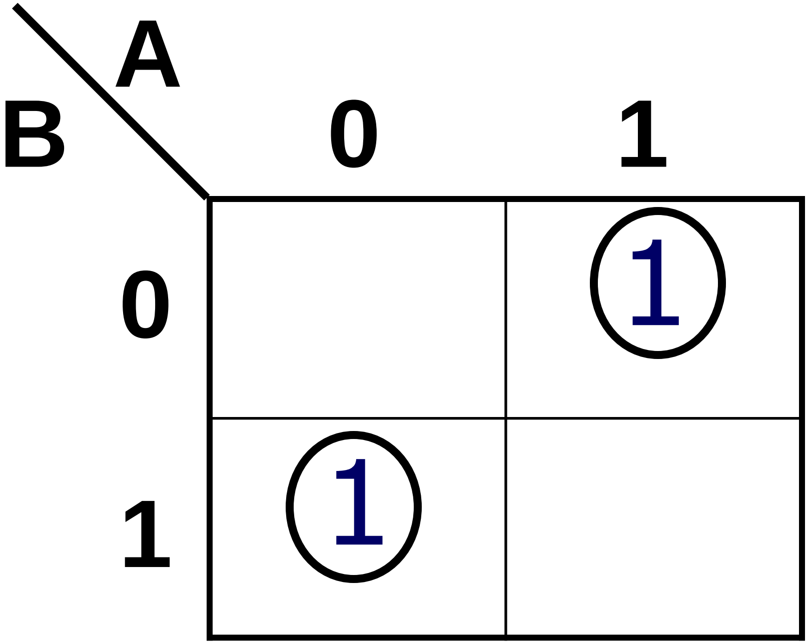
- From the truth table we can write the sum of product expression for difference D and borrow B.
- Half sub tractor using Ex-OR gate.

D=A′B+AB′

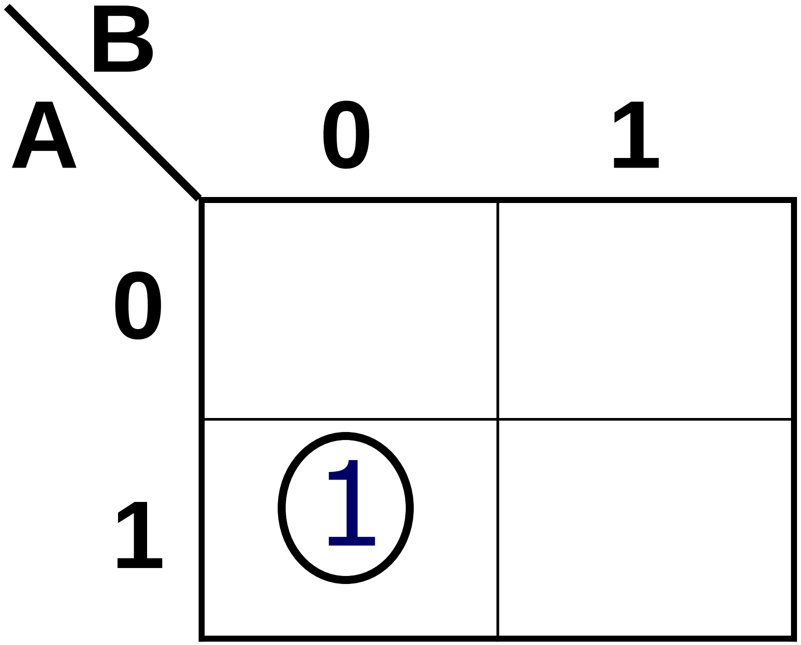
B=A′B



❑ K-MAP for Half Subtractor:



B \ A	0	1
0		1
1	1	

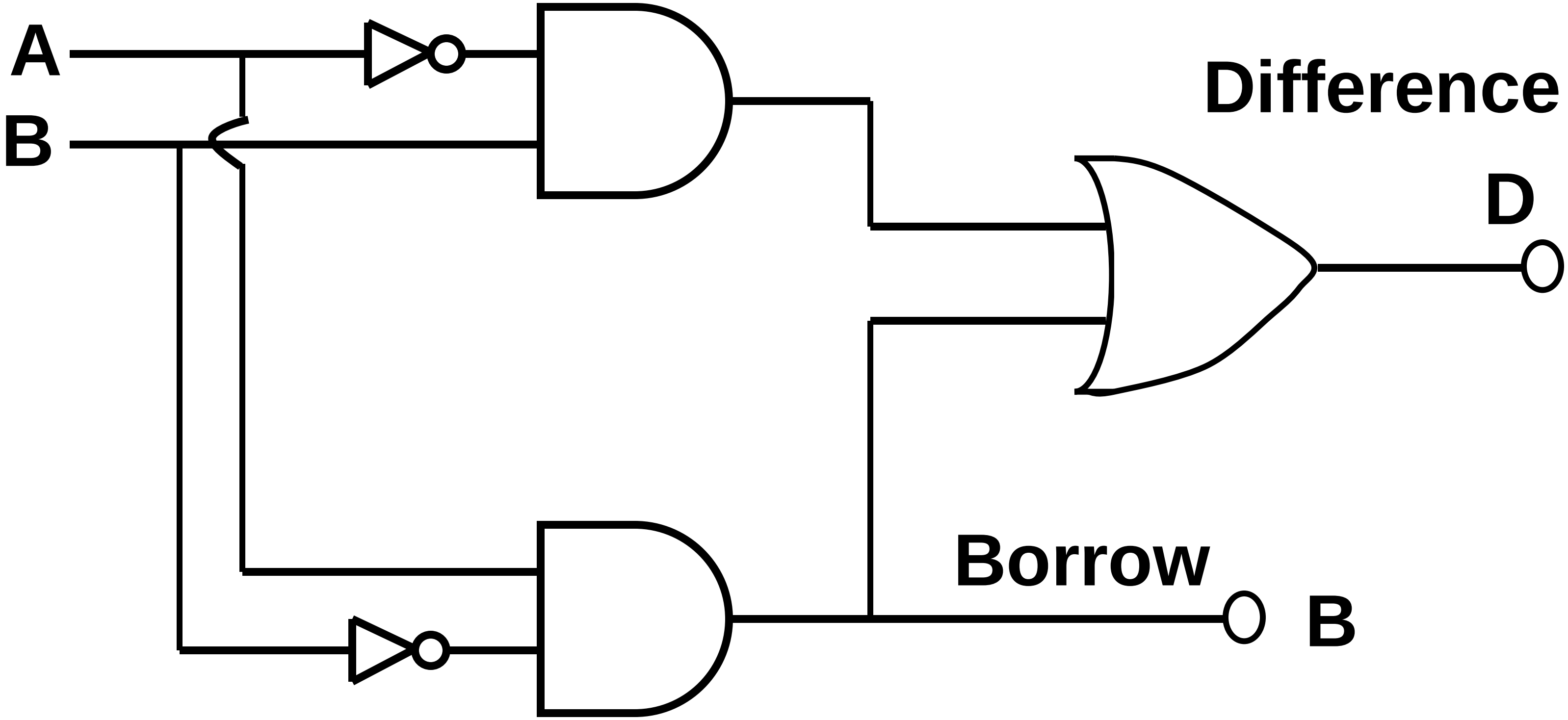


A \ B	0	1
0		
1	1	

D=A’B+AB’

B=A’B

❑ K-MAP for D/ B :

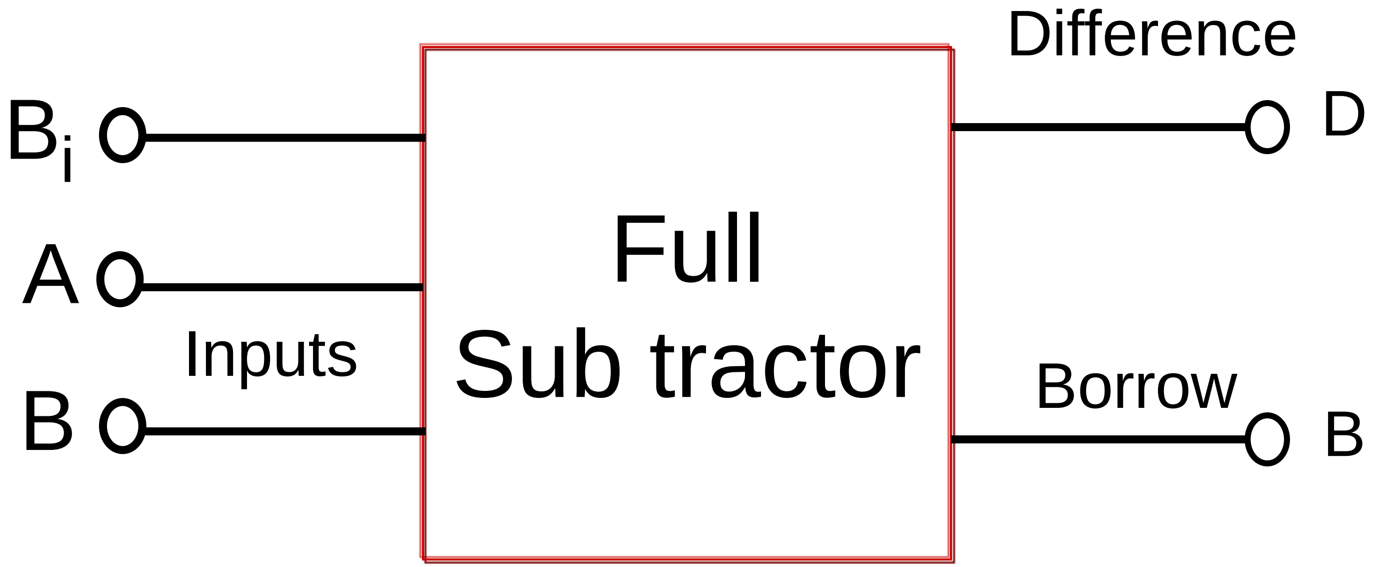


D=A’B+AB’

B=A’B

❑ Full Sub tractor :

❖ Block diagram is shown below:



Block diagram of Full Sub tractor

❑ Full Sub tractor :

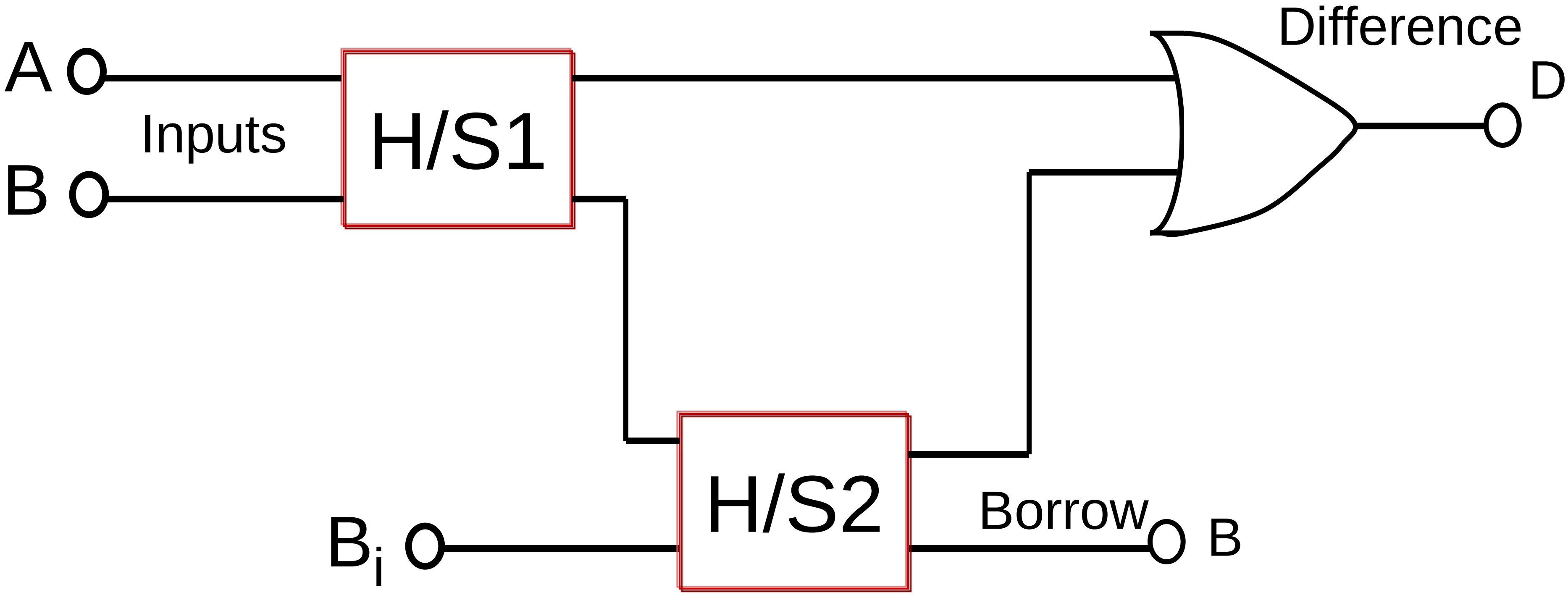
- There are three input terminals and two output terminals.
- One input is A from which the second input B has to be subtracted. B_i is the borrow from previous stage.
- One output is difference D and the other output is borrow B_o here table is given below:

❑ Full Sub tractor Diagram :

Input			Output	
A	**B**	$\mathbf{B_i}$	Differen ce **D**	Borrow **B**
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

❑ Full Sub tractor using two half sub tractor :

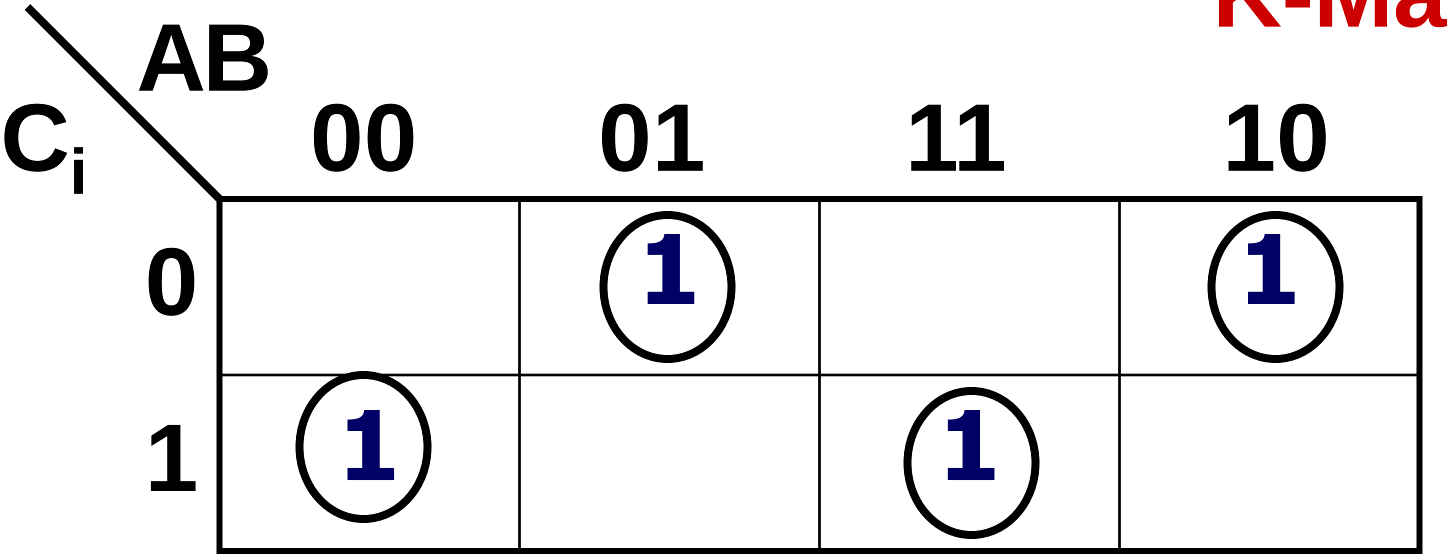
- Block diagram of full sub tractor using two half sub tractor is shown below:



Block diagram of Full Sub tractor using two half sub tractor

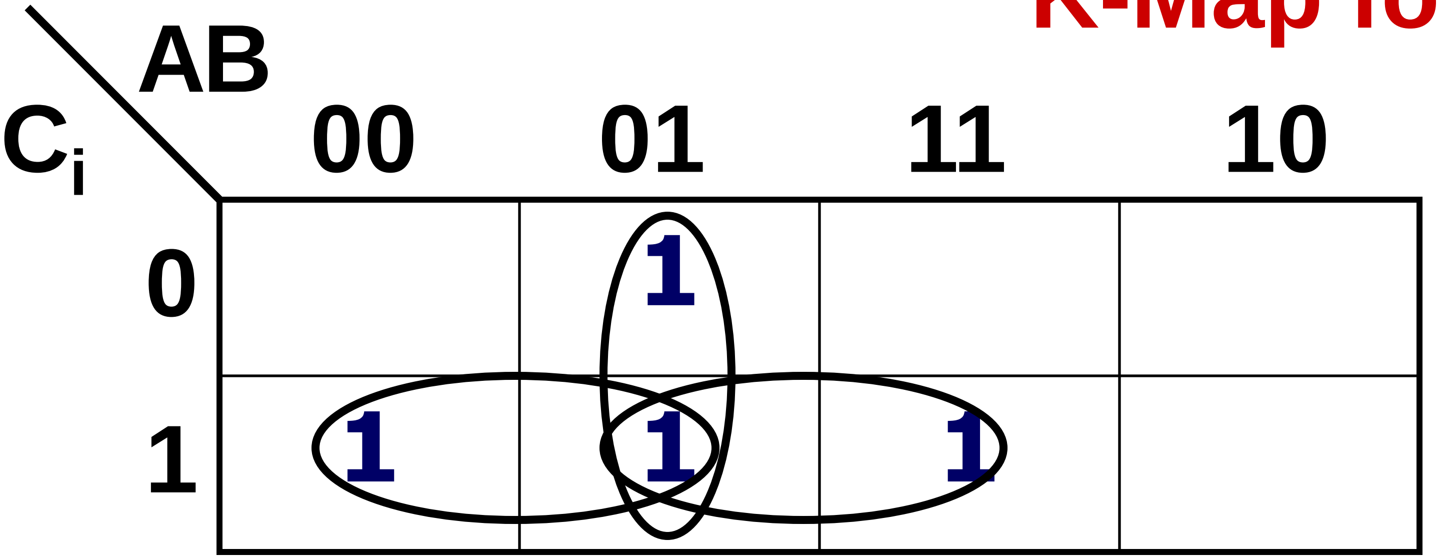
□ K-MAP for Full Subtractor:

K-Map for Sub



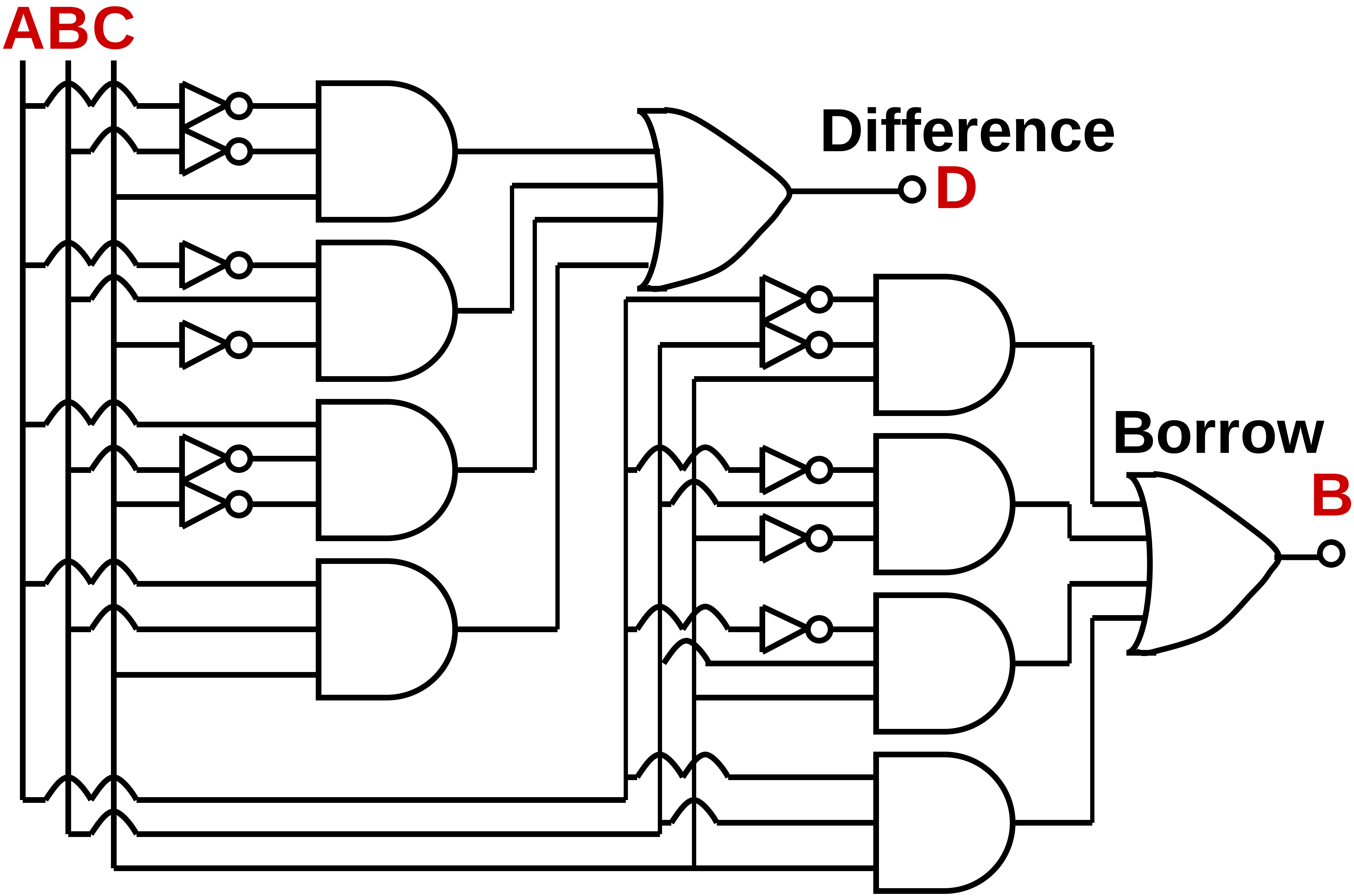
C_i \ AB	00	01	11	10
0		1		1
1	1		1	

K-Map for Borrow



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

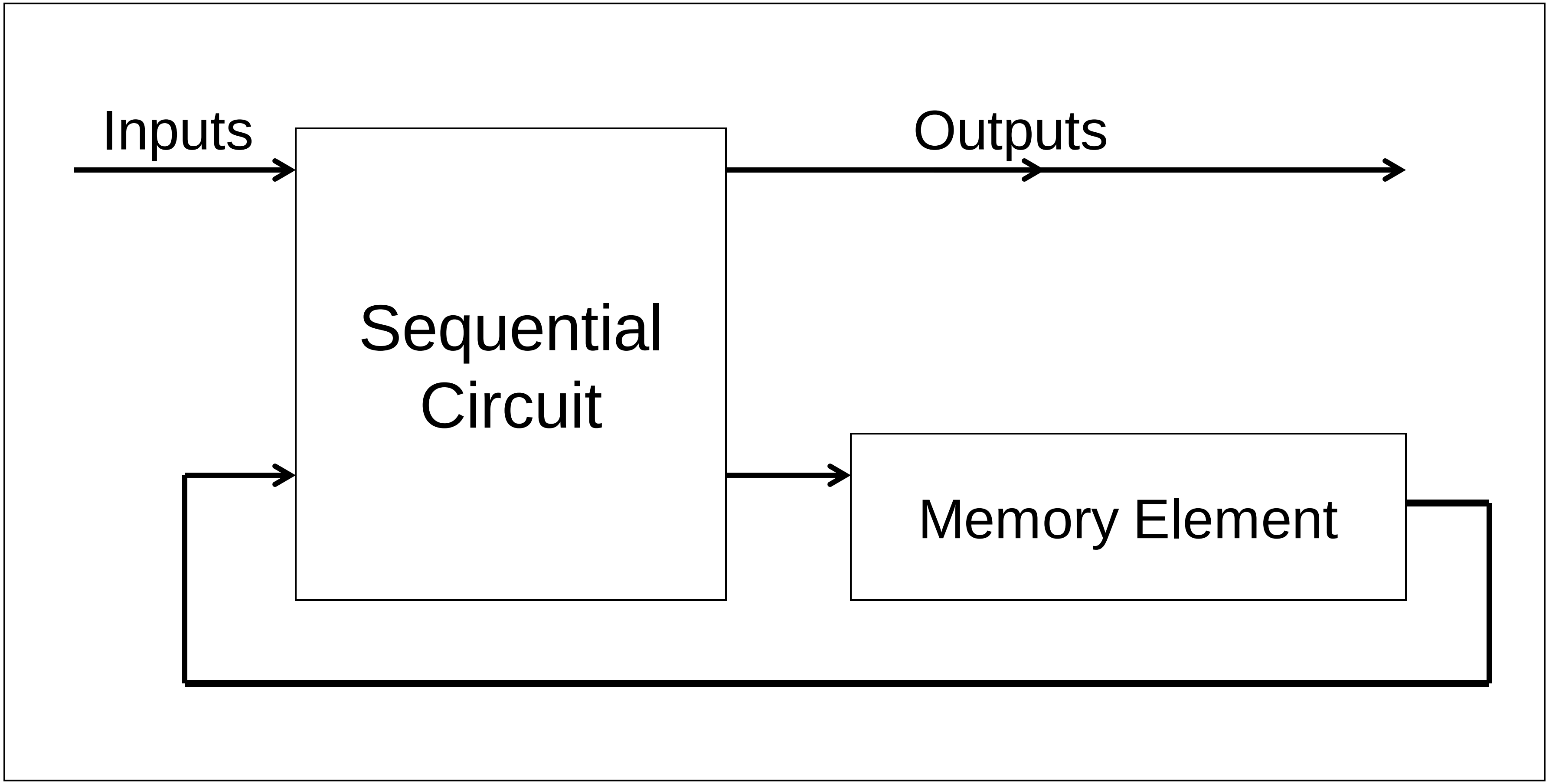
❑ K-MAP for Full Subtractot:



❑ Sequential Circuit :

- ❖ Digital electronics is classified into combinational logic and sequential logic.
- ❖ Combinational logic output depends on the input levels whereas sequential logic output depends on sorted level and also the input levels.
- ❖ The memory element are devices capable of storing binary info.
- ❖ The binary info stored in the memory elements at any given time defines the state of the sequential circuit.

❑ Sequential Circuits:



Block diagram of Sequential Circuit

❑ Sequential Circuit :

- There are two types of sequential circuit. Their classification depends on the timing of their signals.
  - *Synchronous sequential circuits*
  - *Asynchronous sequential circuits*
- In order to build sophisticated digital logic circuits, including computers, we need more a powerful model. We need circuits whose output depends upon both the input of the circuit and its previous state. In other words we need circuits that have *memory*.

❑ Flip-Flop :

- A Flip-Flop is a binary cell capable for storing just one bit of information. i.e. they contain either 0 or 1.
- It is kind of an electronic circuit which has two stable states, therefore it is also known as *bistable multivibrator,* and thereby is capable of serving as one bit of memory.
- In other words a sequential circuit is an interconnection of flip-flop and gates.
- The gates by themselves constitute a combinational circuit, but when included with flip-flop the overall circuit is classified as a sequential circuit.

❑ Flip-Flop :

- It has two output terminals which are complement of each other. Which are **Q** and **Q’.**
- When Q is at logic 0, flip flop is at **reset** and when Q is at logic = 1 it is in **set** mode.
- They are various FF available which are…

(1) SR or RS FF

(2) D FF

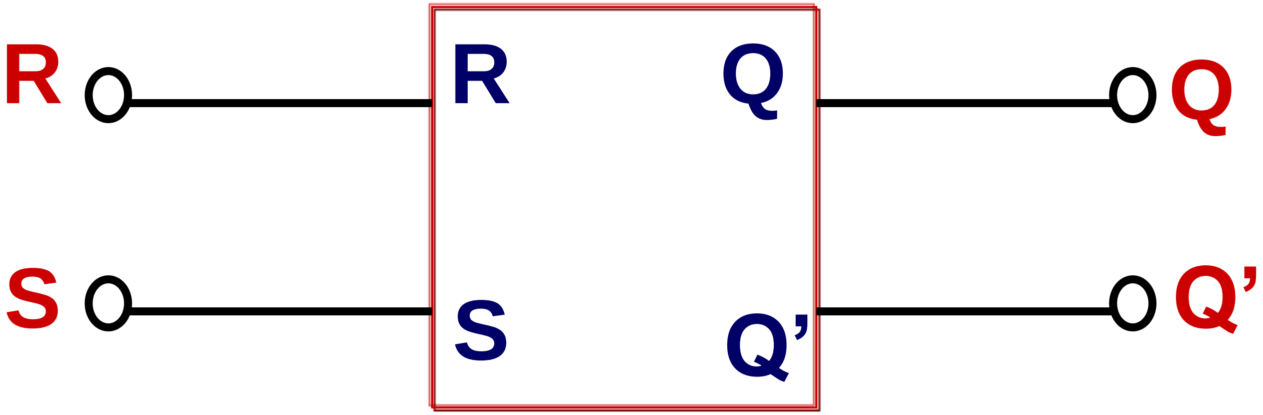
(3) T FF

(4) JK FF

(5) Master slave JK FF

❑ (1) SR or RS FF :

- SR FF means set reset flip flop.
- It also known as reset set flip flop. Block diagram of the flip flop is shown as below.
- Here R and S are input terminals whereas Q and Q′ are complement of each other.

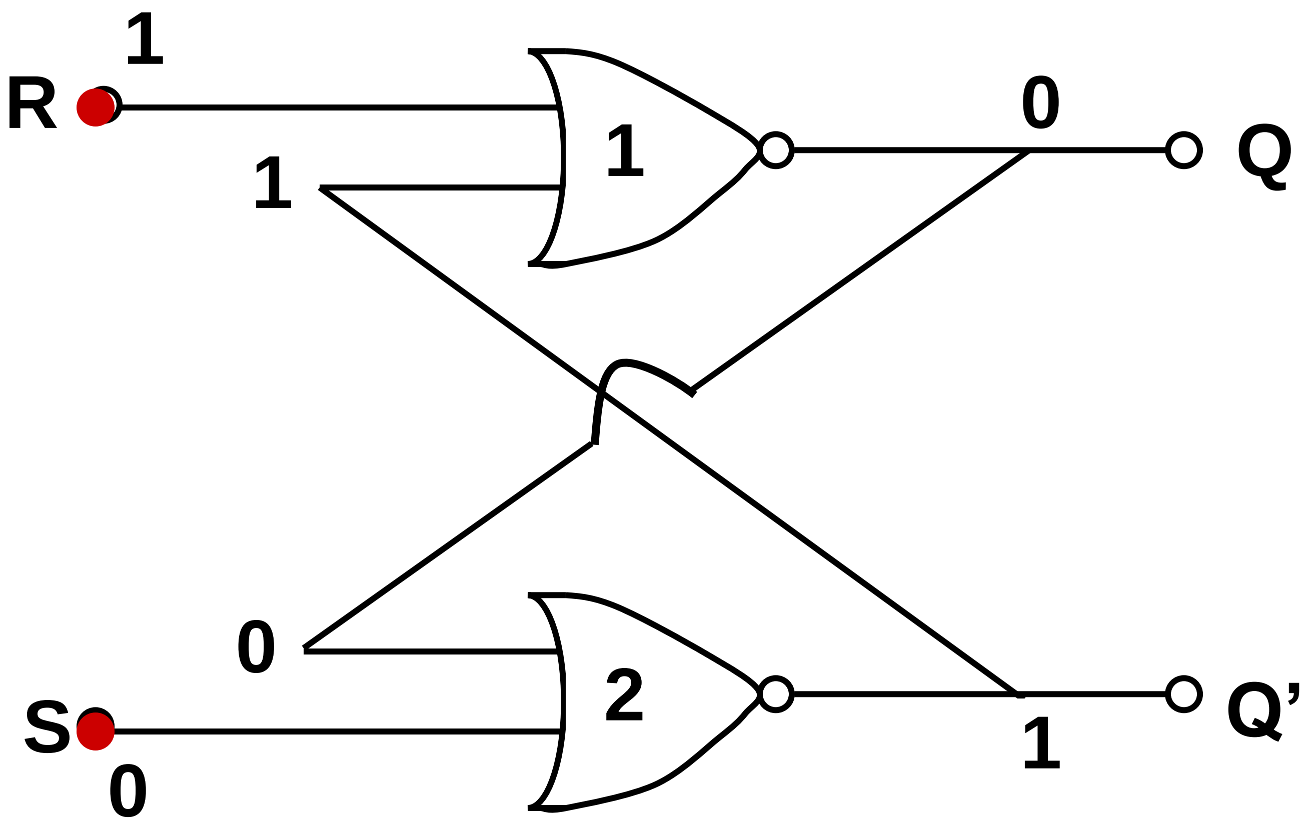


❑ (1) SR or RS FF :

- The most fundamental flip flop is the simple RS flip flop or (RS latch) where R and S stand reset and set.
- RS flip flop using two NOR gate is displayed.

R	S	Q	Comments
0	0	1	No change
0	1	1	Set
1	0	0	Reset
1	1	?	Forbidden

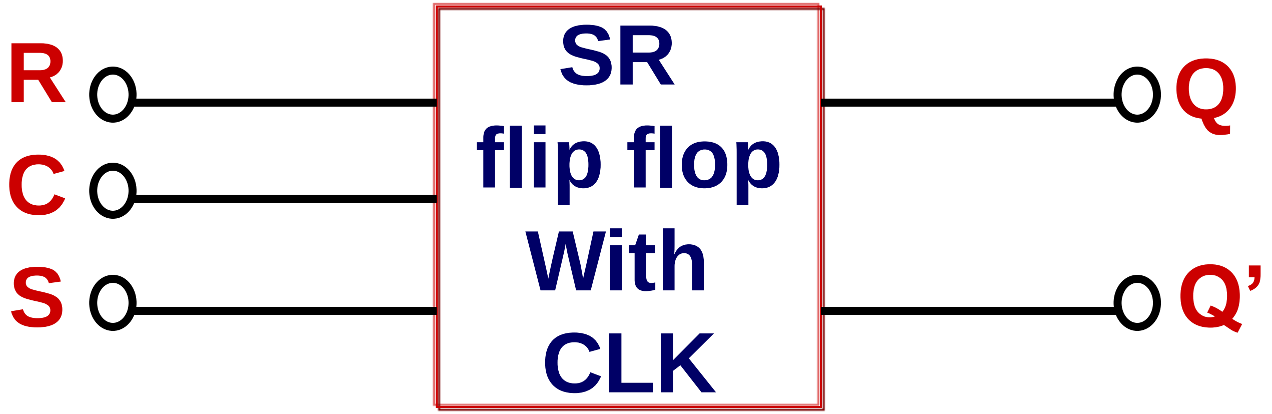
❑ (1) SR or RS FF (Logical Diagram) :



❑ Clocked RS FF :

- In the simple R-S flip flop the output is depends on input condition that is at any time the input conditions are changed, output is also changed. They are called as ***asynchronous*** flip flop.
- The clocked R-S flip flop requires a clocked (Enabled) input.
- It means that this type of flip flop has three inputs, named as **S (set), R (Reset),** and **C (Clock)**.
- Its R and S inputs will control the state of the flip flop only when the clock input is high.
- When the clock is low, the input becomes ineffective and no change of state can take place.
- This flip flop is known as Gated R-S flip flop or ***Synchronous*** flip flop.

❑ Clocked RS FF :



Block diagram of RS FF with clock

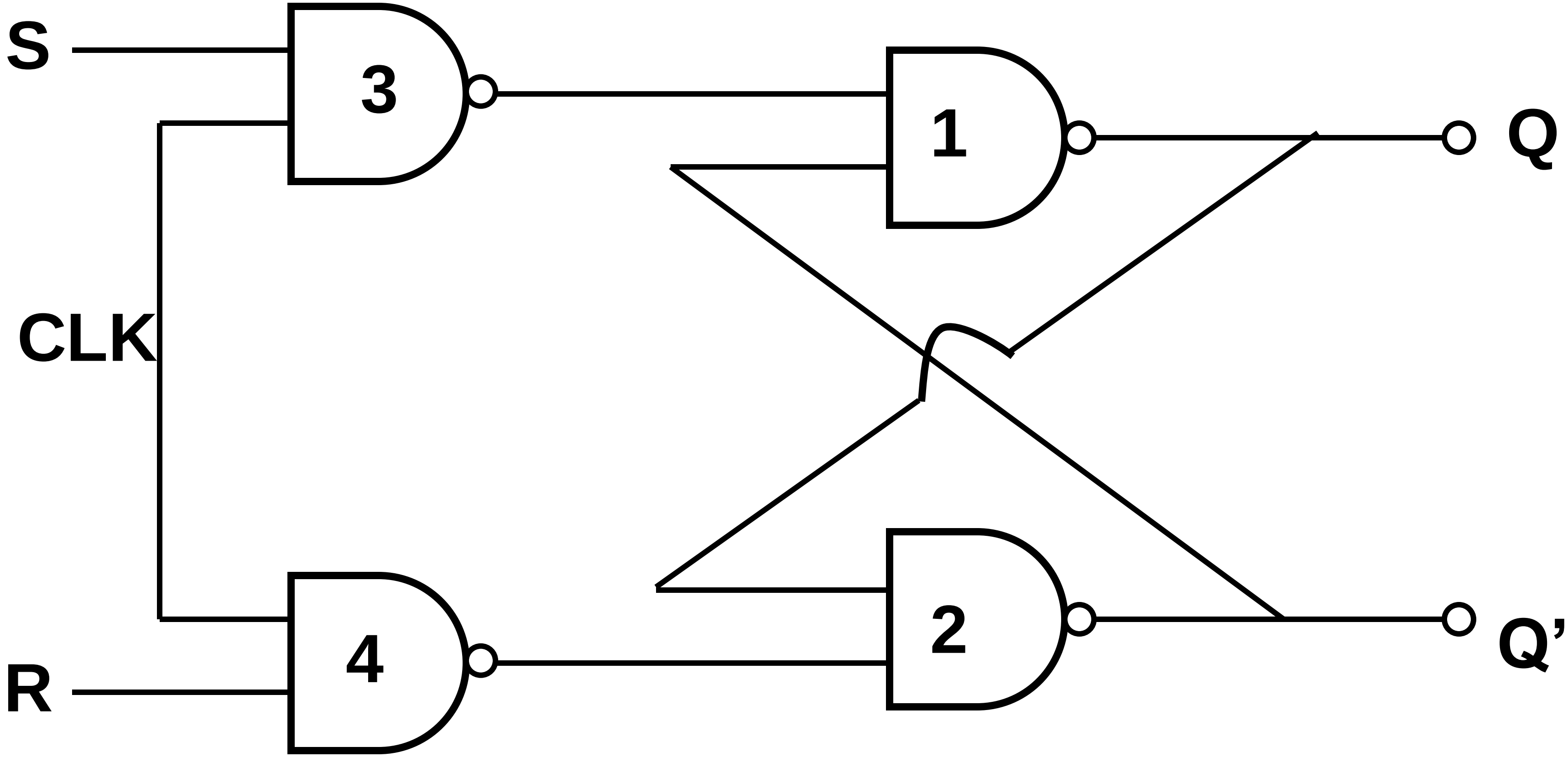
Truth Table

R	S	Q	Comments
0	0	1(High)	No change
0	1	1(High)	Set
1	0	1(High)	Reset
1	1	1(High)	Indeterminate

❑ Clocked RS FF :

- If both inputs R and S equal to zero during the clock transition the output does not change, that is Q remain unchanged.
- The second input condition R=0 and S=1 forces of output of first NAND gate to be high, that is Q=1. Thus Q=1 is said to SET.
- The third input condition in the truth R=1 and S=0 forces output of second NAND gate to be high, that is Q’=1 and Q=0 is said to RESET.

❑ (1) SR or RS FF with clock (Logical Diagram) :



❑ (2) D Flip Flop / Delay flip flop :

- In D type flip flop there is one input called D input (or data input) in addition to clock input.
- The main disadvantages of R-S flip flop is when both the inputs at high level it produced forbidden condition.
- To eliminated this condition the new kind of flip flop is introduced with modification and it is also known as Delay flip flop or D flip flop.
- An inverter is connected as shown in the input so that both that both the input terminals do not go to same state simultaneously. So forbidden condition does not arise.

(2) D Flip Flop / Delay flip flop :

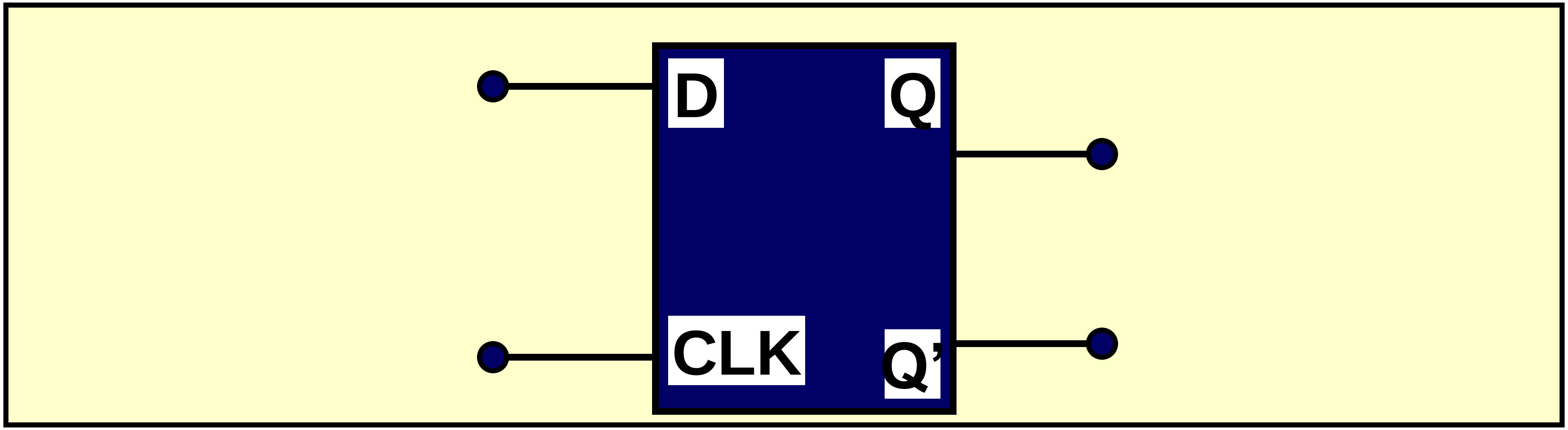
- Truth table of D flip flop:

Q(t) Present state	D	Q(t+1) Next State
0	0	0
0	1	1
1	0	0 Reset
1	1	1 Set

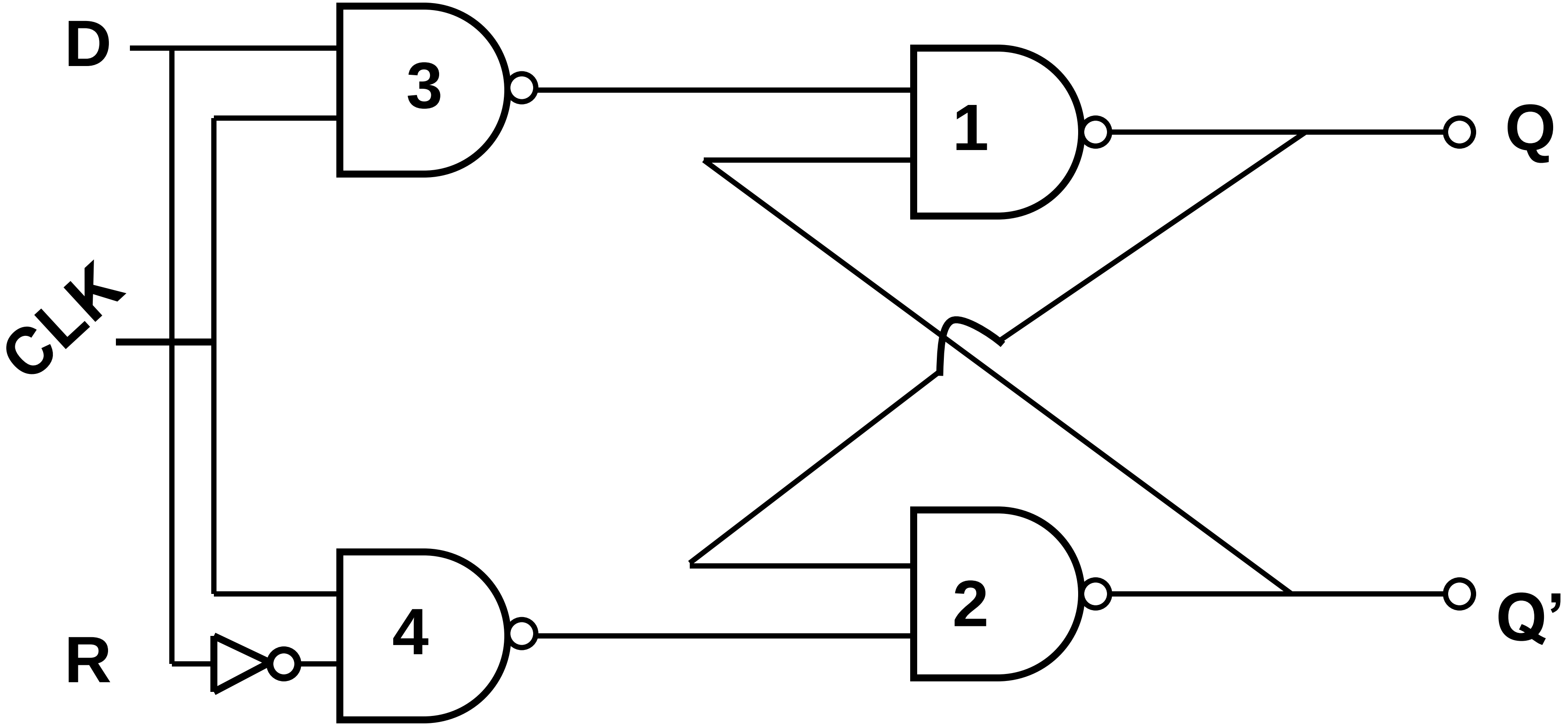
Output =D’Q+DQ

- When D=0 and clock is high then it makes next state Q(t+1) low, means reset the flip flop.
- When D value is high with high clock pulses it causes the flip flop to set. It does not have no change condition.

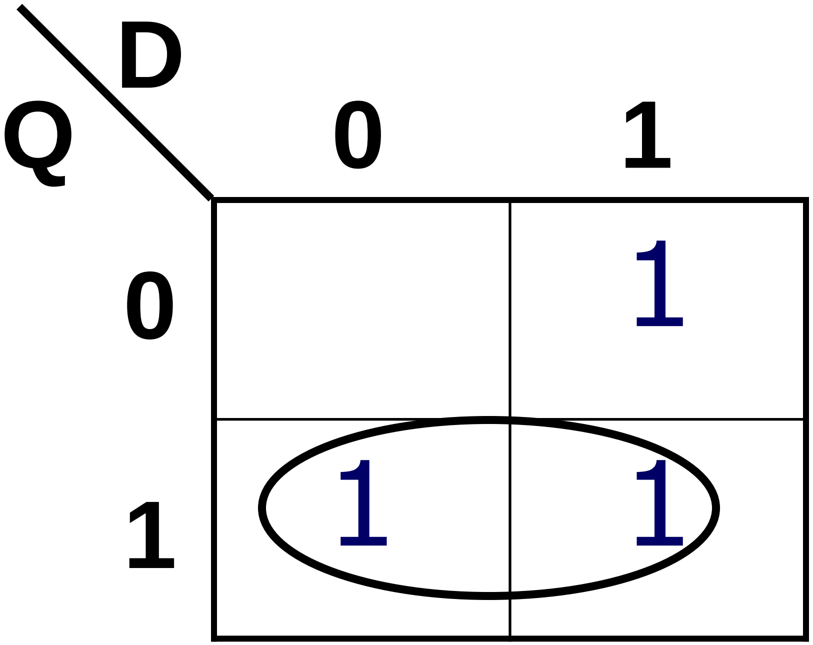
❑ (2) D Flip Flop / Delay flip flop :



❑ (2) D Flip Flop / Delay flip flop :



K-MAP for D flip flop:



Q \ D	0	1
0		1
1	1	1

$$Q(t+1)=D'Q+DQ$$

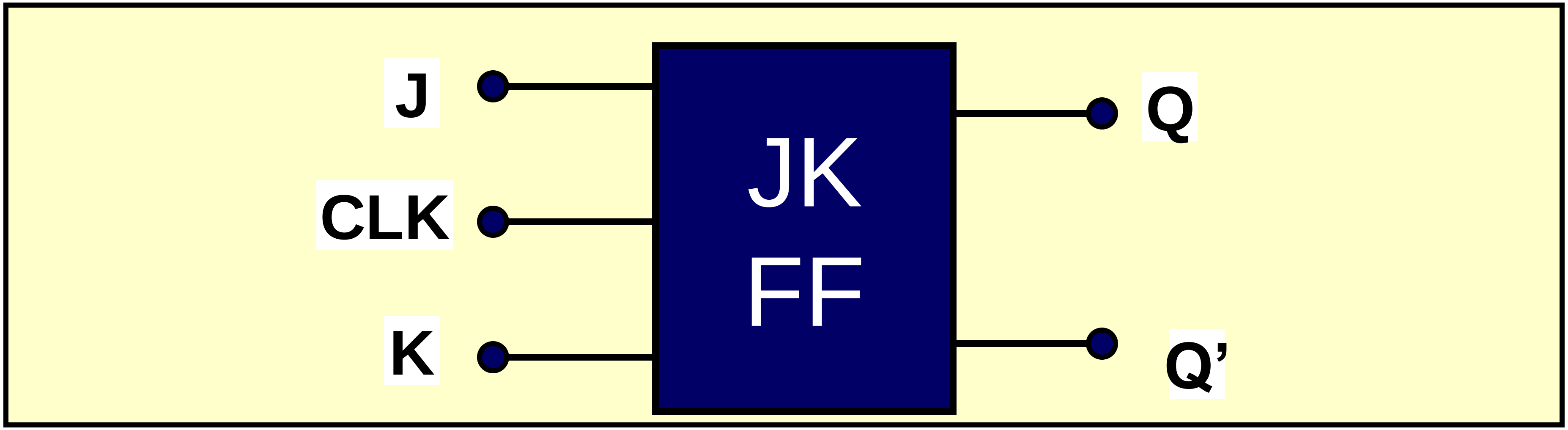
❑ (3) J-K (Jump & Kick) flip flop :

- ❖ The JK flip flop is called a universal flip flop because the other flip flop like D, RS and T can be derived from it.
- ❖ The JK flip flop is very versatile and most widely used.
- ❖ The block diagram and circuit diagram shown in figure.
- ❖ The J k inputs are equal to S (Set) and R (Reset) inputs of RS flip flop.

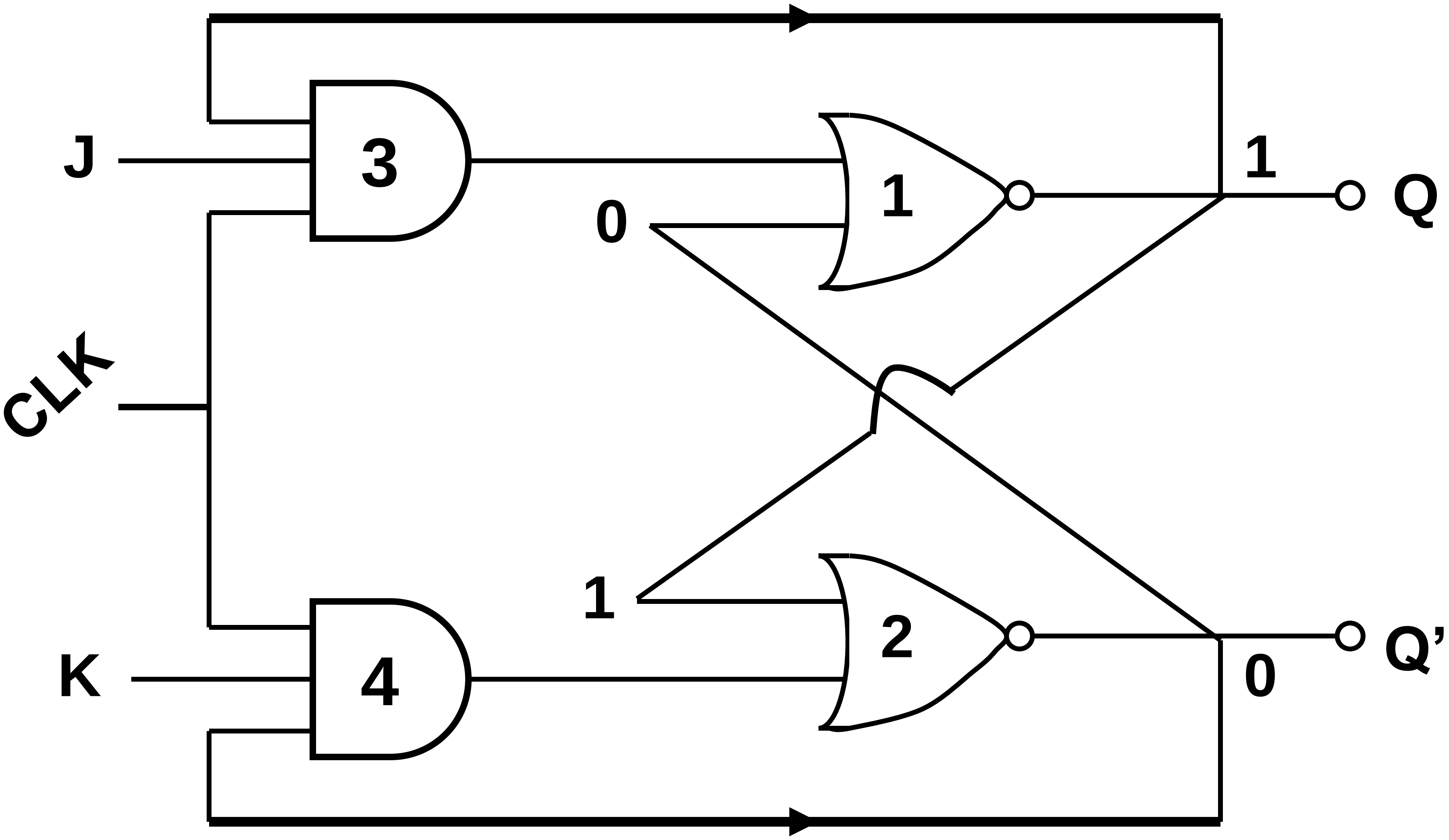
❑ Working of J-K flip flop :

- The functioning of J K flip flop is similar to that of the R-S flip flop.
- Inputs J and K behave like inputs S and R to set and reset flip flop.
- As shown in the truth table if both the inputs J and K equal to zero then no change of state take place even if a clock pulse is applied.
- The second condition J=0 and K=1 causes flip flop reset.
- When J=1 and K=0 the flip flop sets.
- When both inputs J=K=1 the flip flop switches to its complement state.

❑ (3) Working of J-K flip flop :



❑ (3) Working of J-K flip flop :

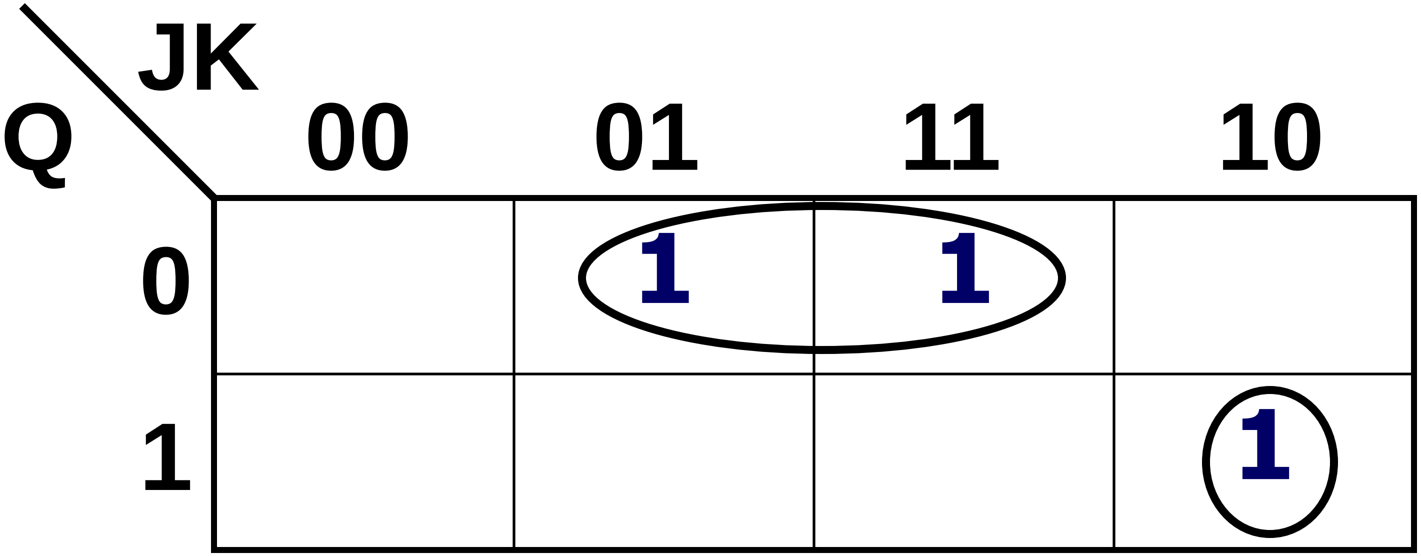


❑ (3) Working of J-K flip flop :

J	K	Q(t+1)
0	0	No Change
0	1	0 Reset
1	0	1 Set
1	1	Q’ Complement

$$Q(t+1)=J'KQ'+JK'Q+JKQ'$$

❑ K-MAP for J-K flip flop :



Q \ JK	00	01	11	10
0		1	1	
1				1

K-MAP for

Reset

Set

Complement

$$Q(t+1)=J'KQ'+JK'Q+JKQ'$$

❑ (4) T flip flop (Toggle(Two input) flip flop) :

- ❖ This flip flop has a single control input, labeled as T for Toggle.
- ❖ This flip flops are not widely available but it is easy to construct from J K flip flop.
- ❖ T flip flop is obtained by combining J and K inputs of JK flip flop.
- ❖ T FF has only two condition when T=0(J=K=0) clock transition does not change the state of flip flop but when T=1, clock transition complements or toggles the state of flip flop.

❑ T flip flop :

Q(t)	T	Q(t+1)
0	0	0 No Change
0	1	1
1	0	1
1	1	0 Complement

$Q(t+1)=Q'T+QT'$

□ K-MAP for T flip flop :

Q \ T	0	1
0		1
1	1	

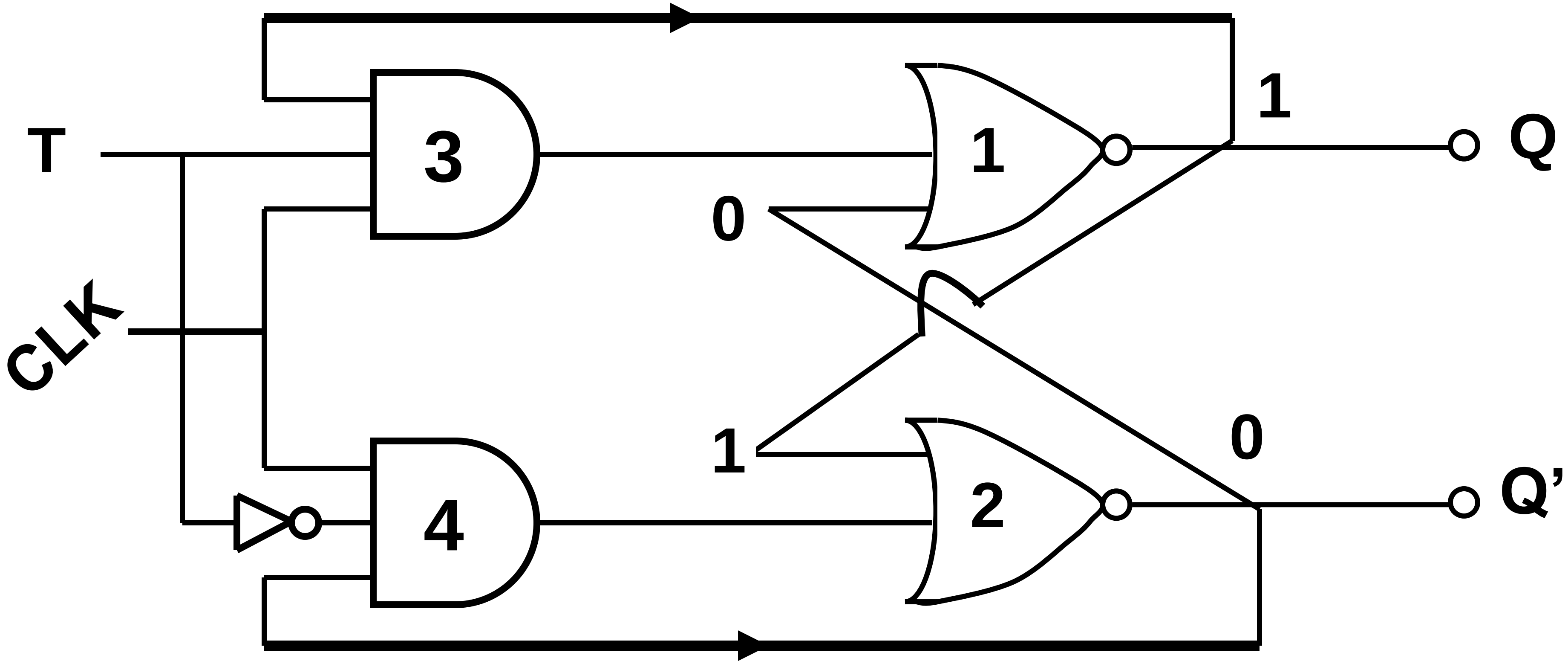
$$Q(t+1)=Q'T+QT'$$

❑ (4) T flip flop (Toggle(Two input) flip flop) :

- It means when

  T=0 : Q(t+1)=Q(t)   Q

  T=1 : Q(t+1)=Q'(t)   Q'



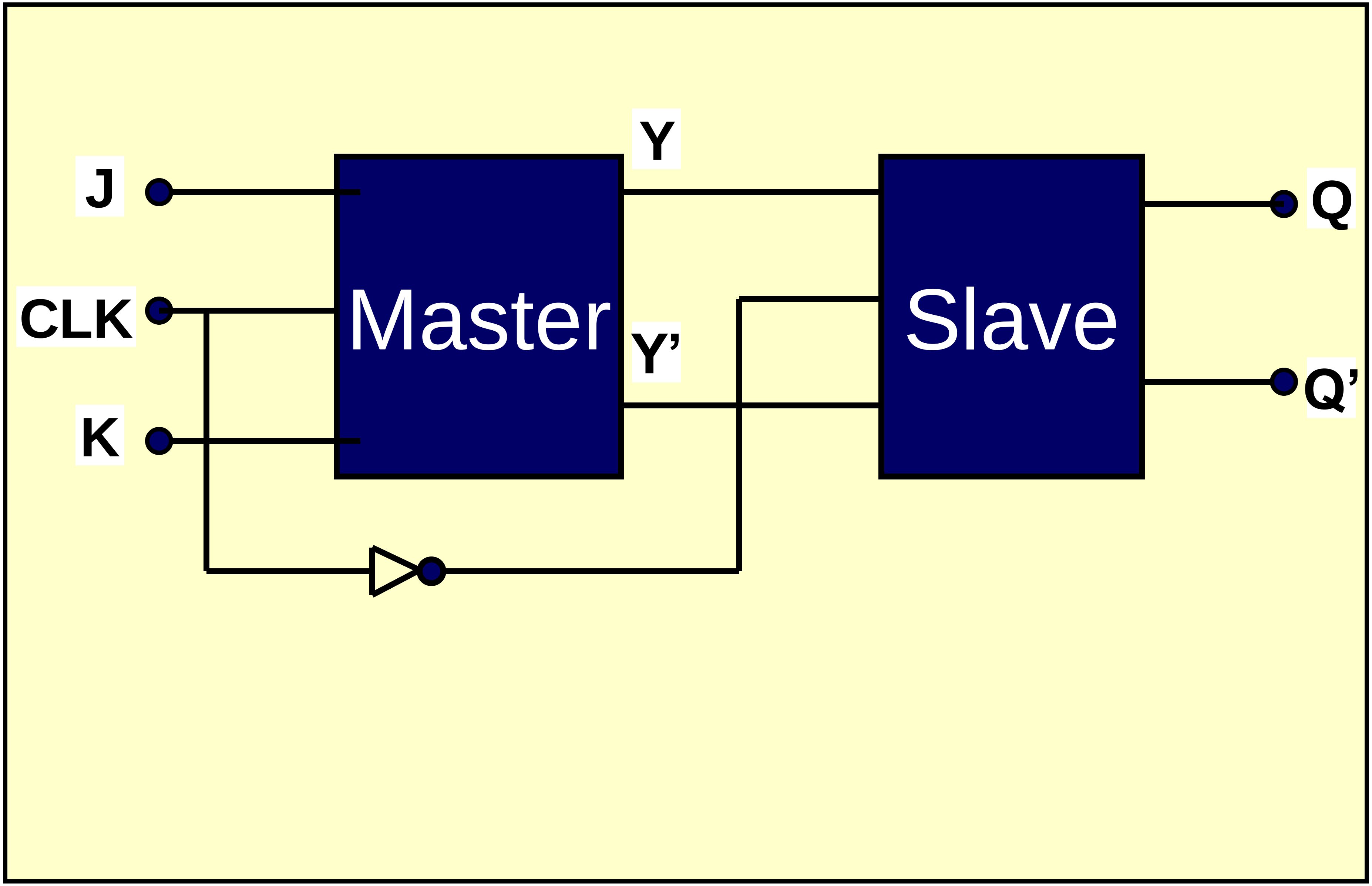
(5) Master Slave flip flop) :

- A master slave flip flop is constructed with the help of two separate circuits.
- First part of the circuit serves as a master and second part of the circuit serves as a slave flip flop.
- **Operation :** When clock pulse CP=0 the output of the inverter 1. This is applied to Slave flip flop.
- Since the clock input of the slave in now 1. The flip flop is enabled and output Q is equal to Y, while Q’ is equal to Y’.

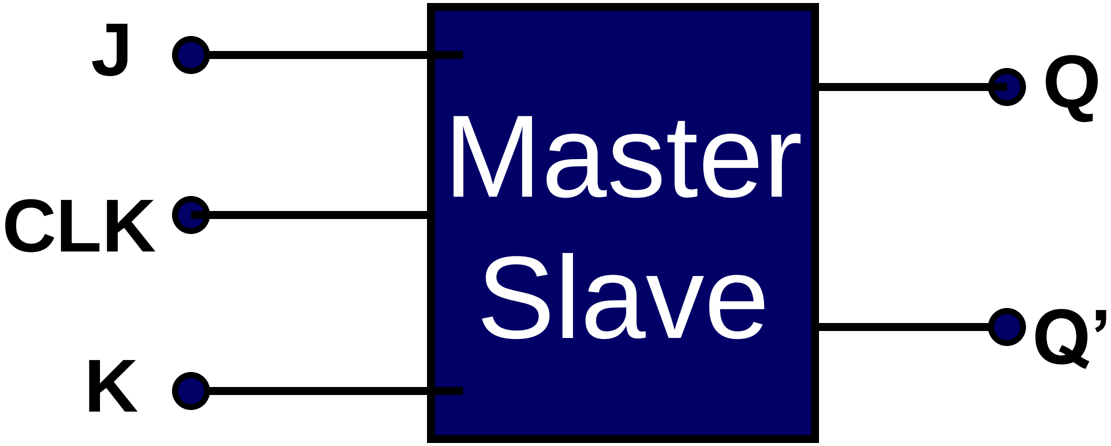
(5) Master Slave flip flop) :

- When the clock pulse again goes to 1, the information then at the external J & K inputs is transmitted to the master flip flop and thus Y and Y' get the values to the inputs of J-K values.
- The slave flip flop however is isolated as long as the clock pulse is at its 1 level because the output of the inverter is zero.
- When the clock input of the slave is 1, the flip flop enabled and output Q is equal to Y while Q' is equal to Y'.

❑ (3) Working of J-K flip flop :

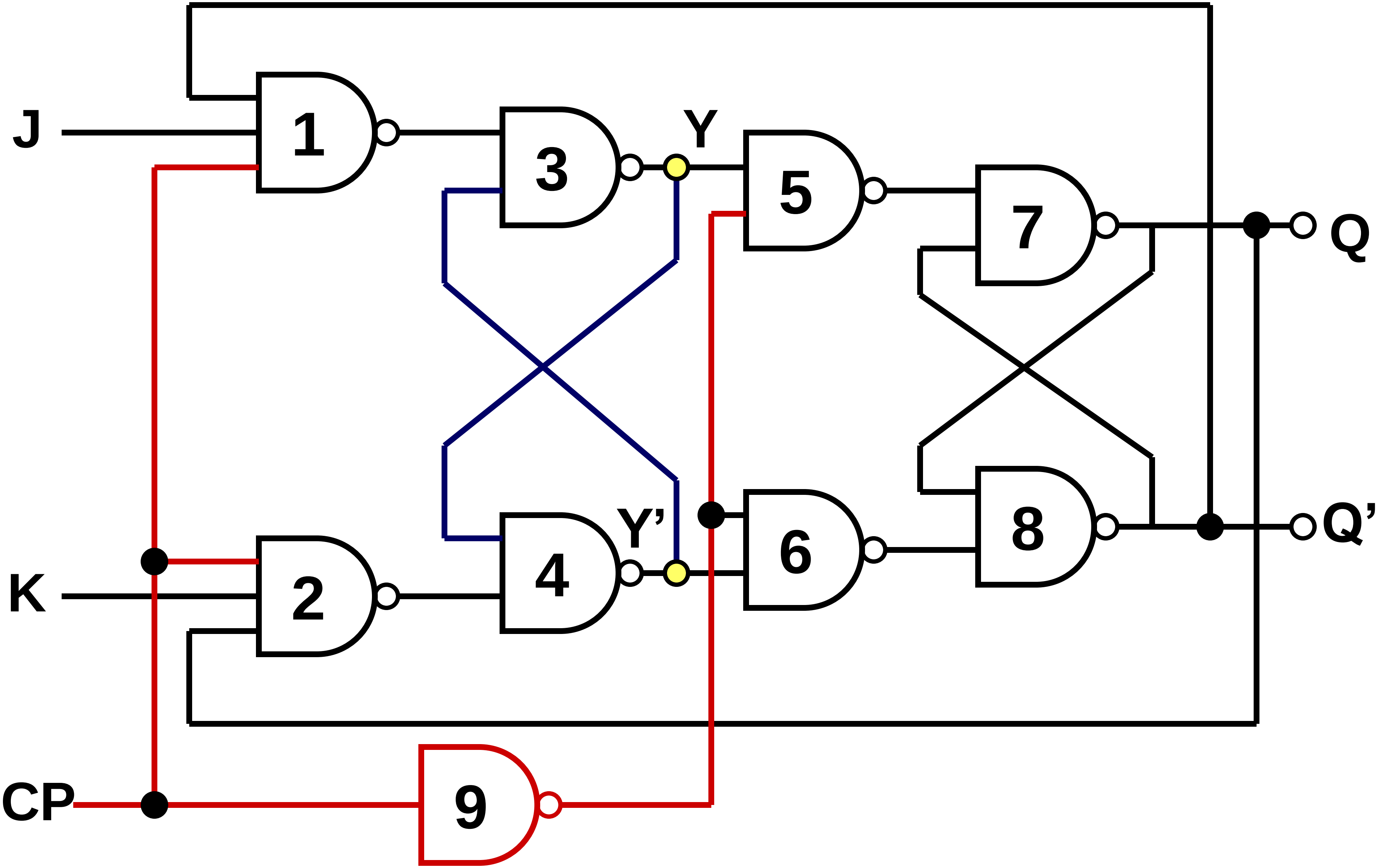


❑ (3) Working of J-K flip flop :



- There are three basic types of master slave flip flop : RS flip flop, D flip flop and JK flip flop.
- The master slave combination can be constructed for any type of flip flop by adding a clocked RS flip flop with an inverted clock to form the slave.
- An example of s master slave JK flip flop constructed with NAND gate is shown in figure.

❑ (3) Working of J-K flip flop :



❑ Difference Between Sequential and Combinational Circuit :

Combinational Circuit	Sequential Circuit
1. In this only logic gates are used. No memory element is used.	1. In this circuit memory element used addition with gates.
2. Output at any instant depends only on the input condition.	2. In this output at any instant is dependent also on the past condition.

❑ Difference Between Sequential and Combinational Circuit :

Combinational Circuit	Sequential Circuit
3. Design is simple due to absence of the memory element.	3. Design is difficult due to memory element.
4. More hardware required.	4. Less hardware needed.
5. Cost is more as more hardware needed.	5. Cost is less.