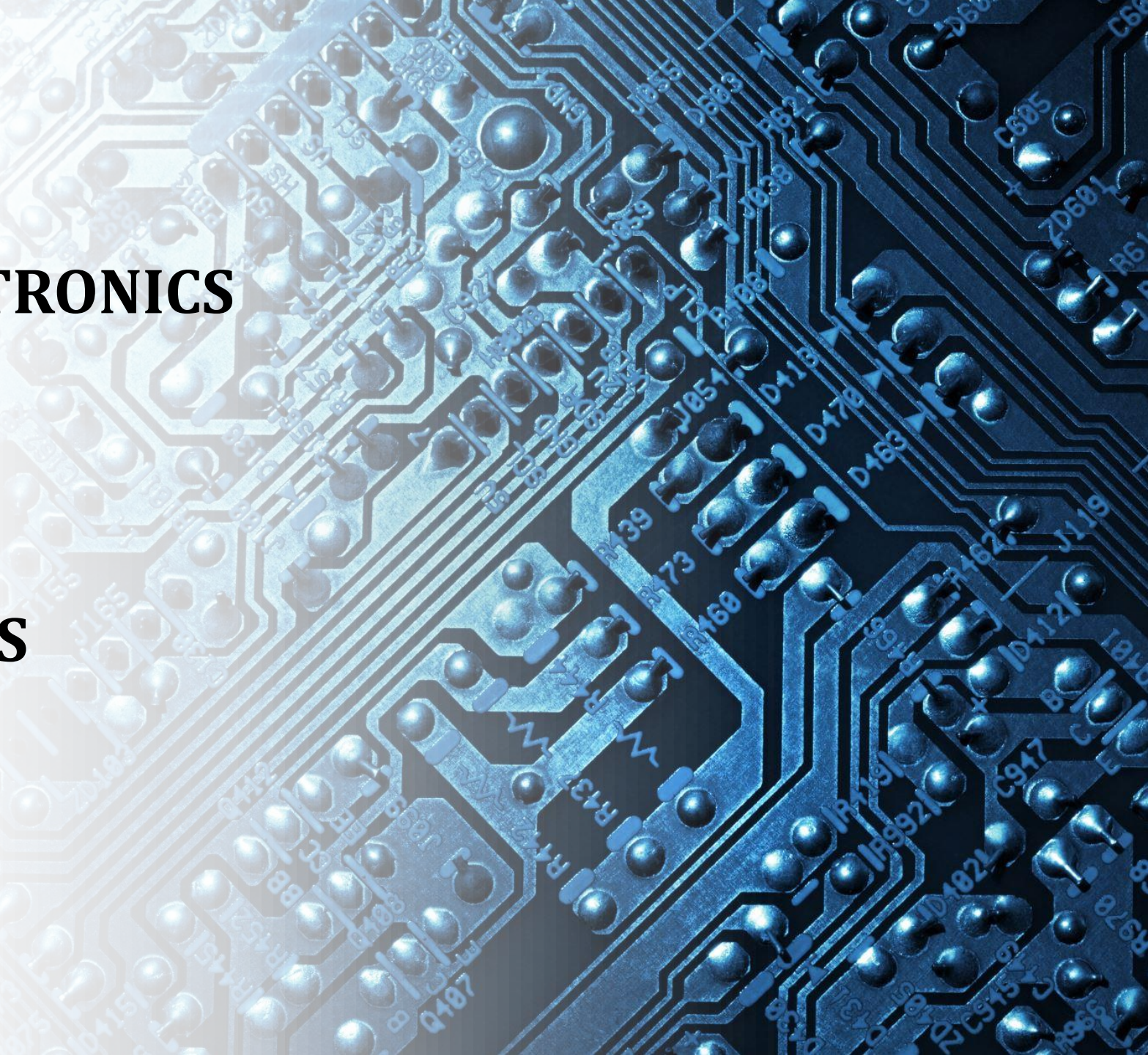
104010 : BASIC ELECTRONICS ENGINEERING

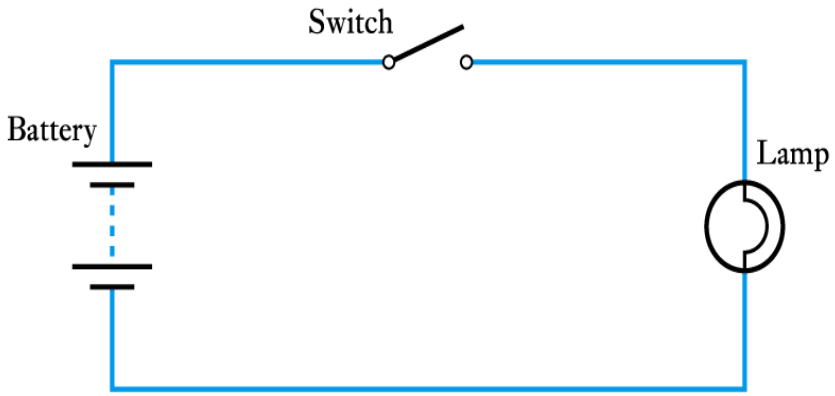
UNIT III
DIGITAL ELECTRONICS

Logic Gates



Concept of Binary Logic

- A binary quantity is one that can take only 2 states.



S	L
OPEN	OFF
CLOSED	ON

S	*L*
0	0
1	1

Binary Logic

- In Binary logic, the addition is,

  0 + 0= 0

  0 + 1= 1

  1 + 0= 1

  1 + 1= 1

- The Multiplication of two bits is,

  0.0=0

  0.1=0

  1.0=0

  1.1=1

Logic Gates

- The building blocks used to create digital circuits are **logic gates.**
- There are three elementary logic gates and a range of other simple gates.
- Each gate has its own **logic symbol** which allows complex functions to be represented by a logic diagram.
- The function of each gate can be represented by a **truth table** or using **Boolean notation.**

AND Gate

- The output state of a digital logic AND gate only returns “LOW” again when ANY of its inputs are at a logic level “0”. In other words, for a logic AND gate, any LOW input will give a LOW output.
- The logic or Boolean expression given for a digital logic AND gate is that for Logical Multiplication which is denoted by a single dot or full stop symbol, (.) giving us the Boolean expression of: A.B = Q.

(a) Circuit symbol

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

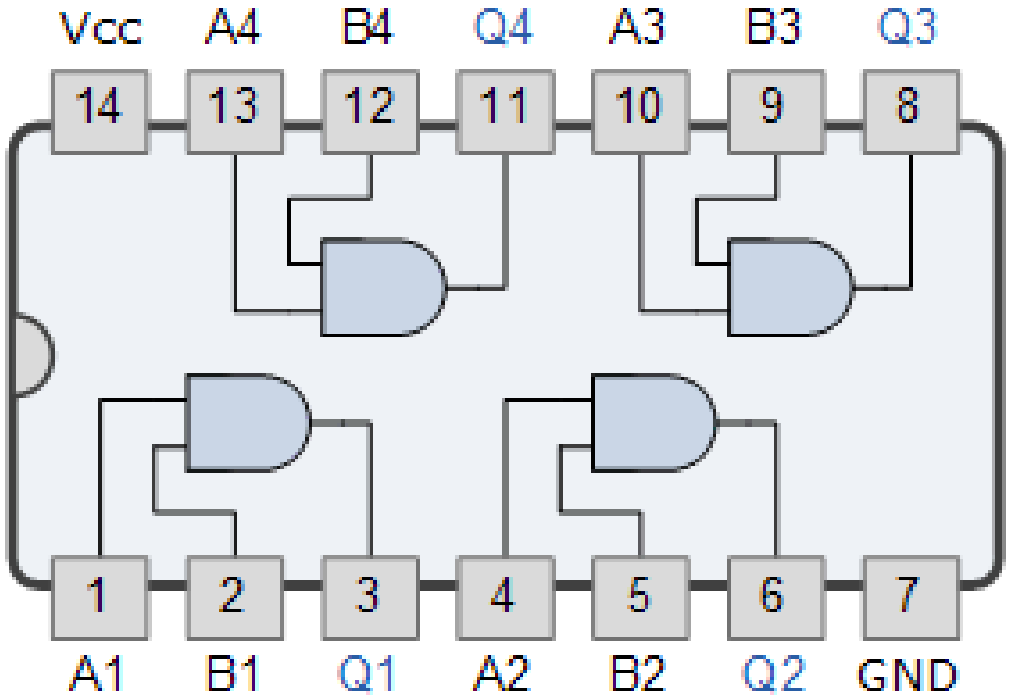
(b) Truth table

$$C = A \cdot B$$

(c) Boolean expression

AND Gate IC: 7408

7408 Quad 2-input AND Gate



OR Gate

- The output, Q of a “Logic OR Gate” only returns “LOW” again when ALL of its inputs are at a logic level “0”. In other words, for a logic OR gate, any “HIGH” input will give a “HIGH”, logic level “1” output.
- The logic or Boolean expression given for a digital logic OR gate is that for Logical Addition which is denoted by a plus sign, (+) giving us the Boolean expression of: A+B = Q.

(a) Circuit symbol

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

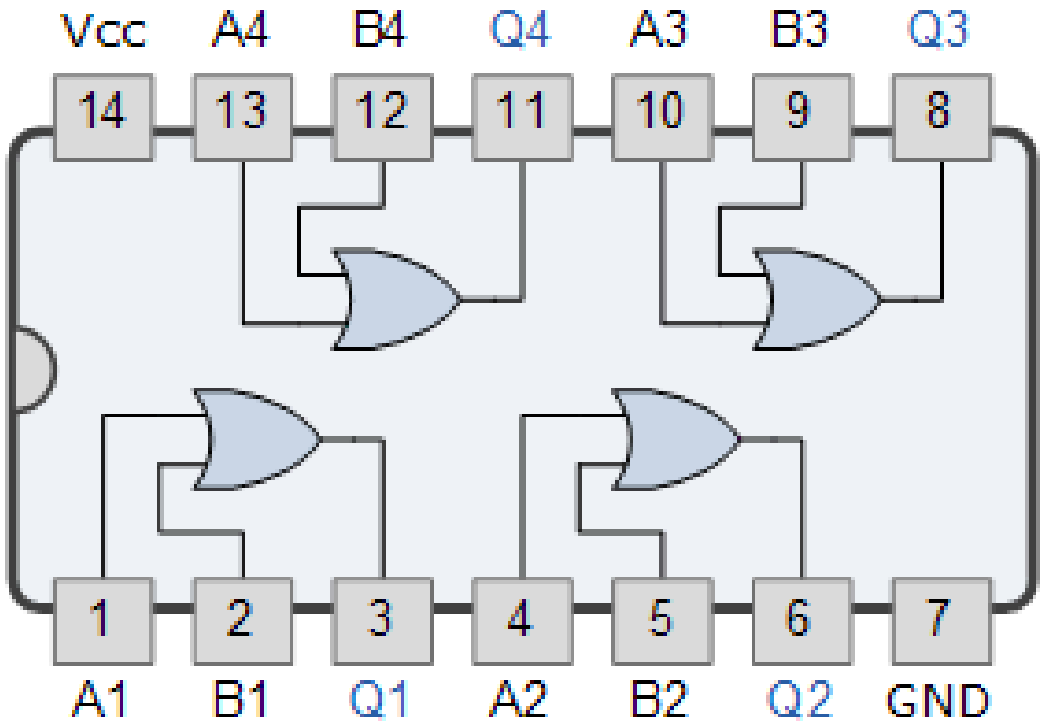
(b) Truth table

$C = A + B$

(c) Boolean expression

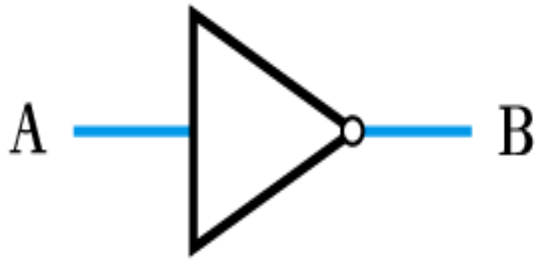
OR Gate IC: 7432

7432 Quad 2-input OR Gate



NOT Gate

- Inverting NOT gates are single input device which have an output level that is normally at logic level “1” and goes “LOW” to a logic level “0” when its single input is at logic level “1”, in other words it “inverts” (complements) its input signal.
- The output from a NOT gate only returns “HIGH” again when its input is at logic level “0” giving us the Boolean expression of: $\overline{A} = Q$.



(a) Circuit symbol

A	B
0	1
1	0

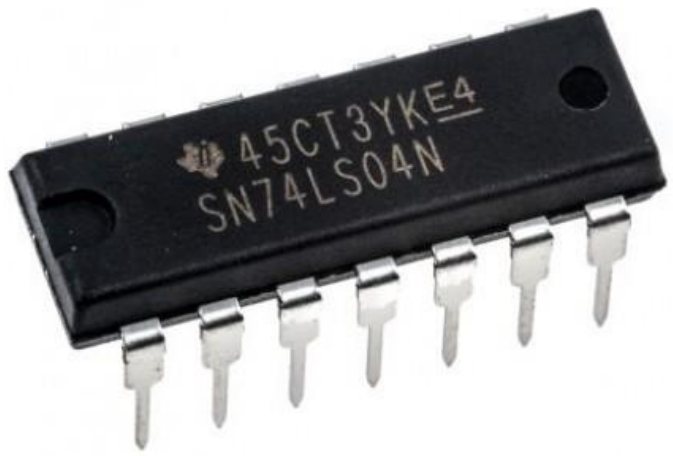
(b) Truth table

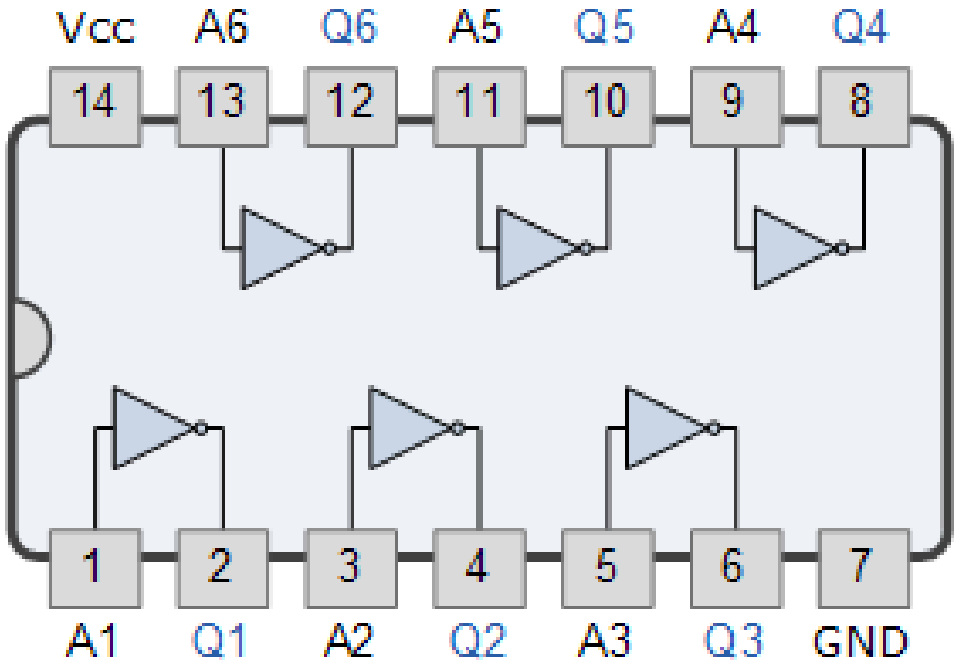
$B = \overline{A}$

(c) Boolean expression

NOT Gate IC: 7404

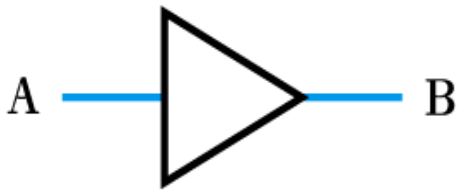
7404 NOT Gate or Inverter





Logic Buffer Gate

- The digital buffer is a “non-inverting” device and will therefore give us the Boolean expression of: Q = A.



(a) Circuit symbol

A	B
0	0
1	1

(b) Truth table

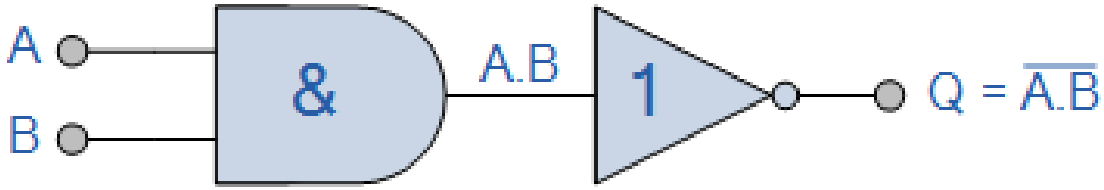
$B = A$

(c) Boolean expression

NAND Gate

- The NAND (Not – AND) gate has an output that is normally at logic level “1” and only goes “LOW” to logic level “0” when ALL its inputs are at logic level “1”. The Logic NAND Gate is the reverse or “Complementary” form of the AND gate we have seen previously.

- Logic NAND Gate Equivalence:



2-input "AND" gate plus a "NOT" gate

NAND Gate

- The logic or Boolean expression given for a logic NAND gate is that for Logical Addition, which is the opposite to the AND gate, and which it performs on the complements of the inputs. The Boolean expression for a logic NAND gate is denoted by a single dot or full stop symbol, (.) with a line or Overline, (¯) over the expression to signify the NOT or logical negation of the NAND gate giving us the Boolean expression of: $\overline{A.B} = Q$

(a) Circuit symbol

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

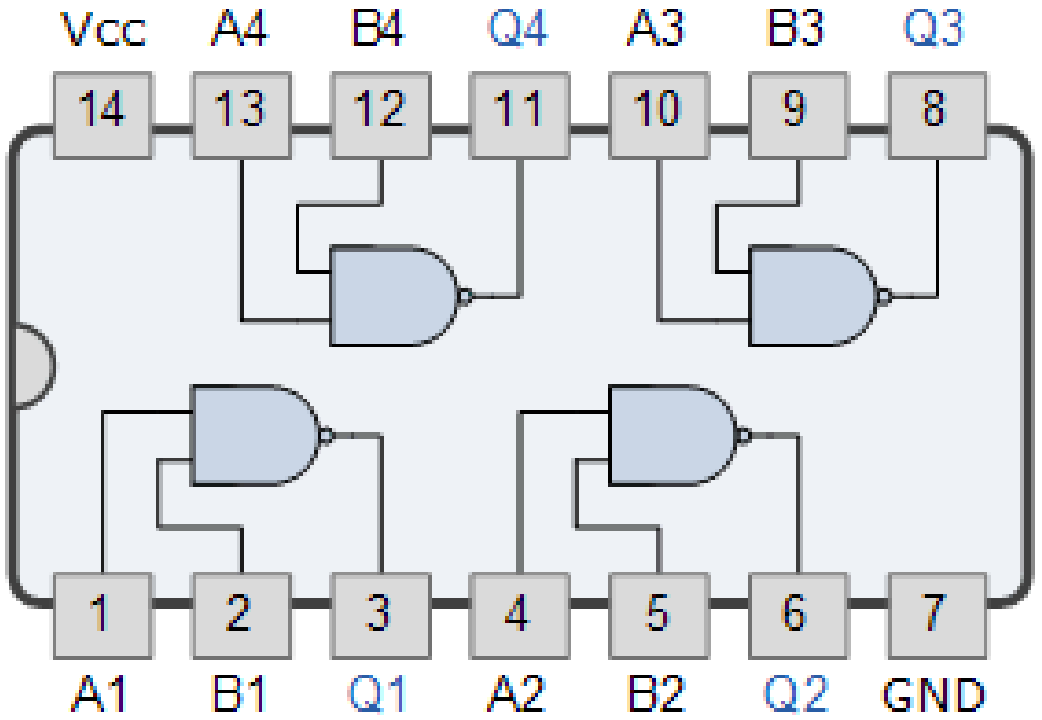
(b) Truth table

$C = \overline{A \cdot B}$

(c) Boolean expression

NAND Gate IC: 7400

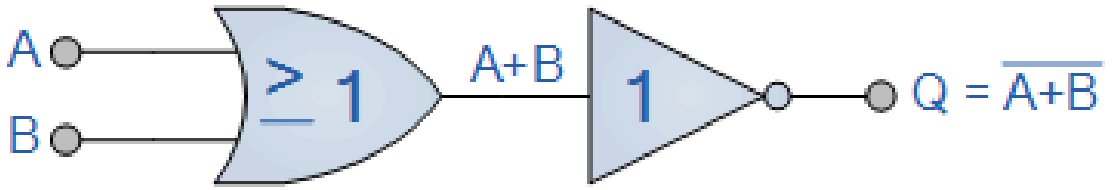
7400 Quad 2-input NAND Gate



NOR Gate

- The inclusive NOR (Not-OR) gate has an output that is normally at logic level “1” and only goes “LOW” to logic level “0” when ANY of its inputs are at logic level “1”. The Logic NOR Gate is the reverse or “Complementary” form of the inclusive OR gate we have seen previously.

- Logic NOR Gate Equivalence:



2-input "OR" gate plus a "NOT" gate

NOR Gate

- The logic or Boolean expression given for a logic NOR gate is that for Logical Multiplication which it performs on the complements of the inputs. The Boolean expression for a logic NOR gate is denoted by a plus sign, (+) with a line or Overline, (¯) over the expression to signify the NOT or logical negation of the NOR gate giving us the Boolean expression of: $\overline{A + B} = Q$

(a) Circuit symbol

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

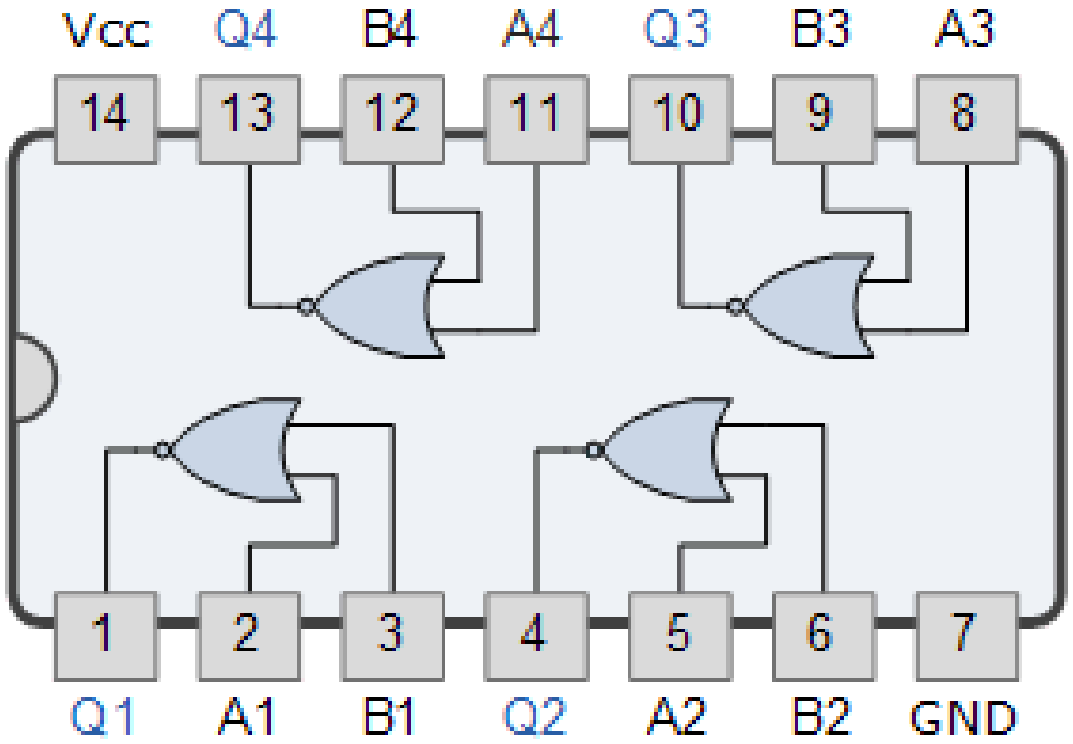
(b) Truth table

$C = \overline{A + B}$

(c) Boolean expression

NOR Gate IC: 7402

7402 Quad 2-input NOR Gate



Ex-OR Gate

- A logic output “1” is obtained when ONLY A = “1” or when ONLY B = “1” but NOT both together at the same time, giving the binary inputs of “01” or “10”, then the output will be “1”. This type of gate is known as an Exclusive-OR function or more commonly an Ex-Or function for short. This is because its boolean expression excludes the “OR BOTH” case of Q = “1” when both A and B = “1”.

- In other words the output of an Exclusive-OR gate ONLY goes “HIGH” when its two input terminals are at “DIFFERENT” logic levels with respect to each other.

Ex-OR Gate

- An odd number of logic “1’s” on its inputs gives a logic “1” at the output. These two inputs can be at logic level “1” or at logic level “0” giving us the Boolean expression of: $Q = \overline{A}.B + A.\overline{B}$.

(a) Circuit symbol

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

(b) Truth table

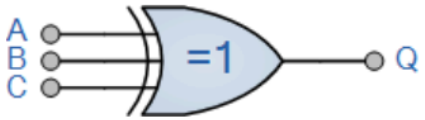
$C = A \oplus B$

(c) Boolean expression

Ex-OR Gate

The 3-input Logic Ex-OR Gate

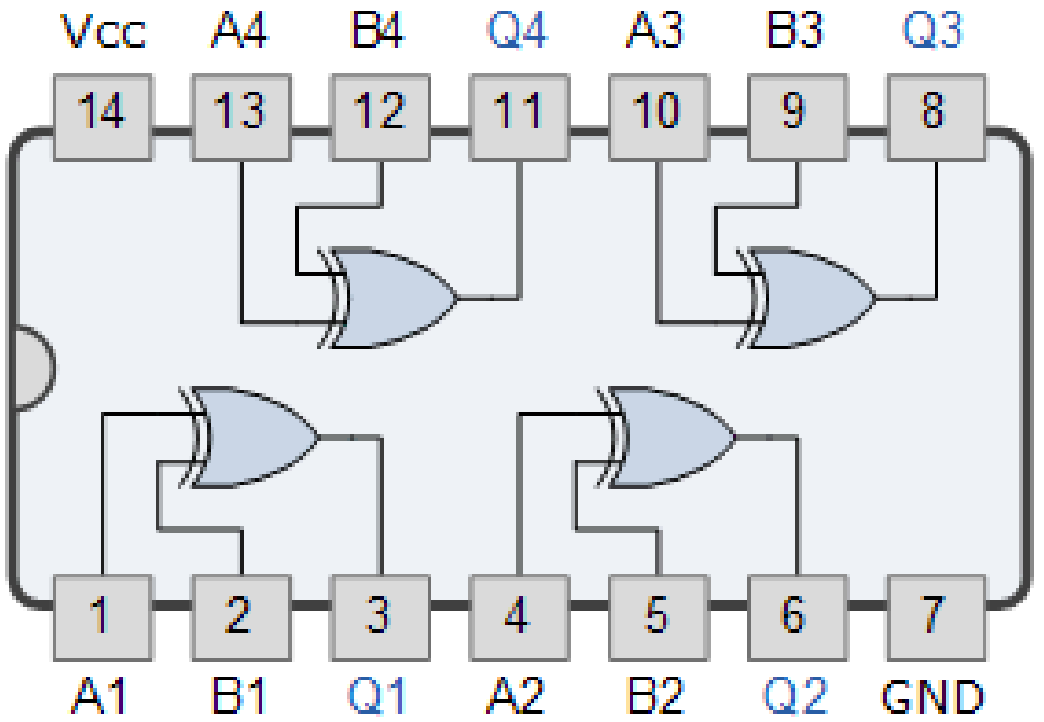
$Q = A \oplus B \oplus C$

Symbol	Truth Table			
 3-input Ex-OR Gate	C	B	A	Q
	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
Boolean Expression $Q = A \oplus B \oplus C$	"Any **ODD** Number of Inputs" gives Q			

→ When no. of '1's in input is odd, the o/p of Ex-OR is '1'.

Ex-OR Gate IC: 7486

7486 Quad 2-input Ex-OR Gate



Ex-NOR Gate

- A logical output “1” is only obtained if BOTH of its inputs are at the same logic level, either binary “1” or “0”. For example, “00” or “11”. This input combination would then give us the Boolean expression of: $Q = A.B + \overline{A}.\overline{B}$

- An even number of logic “1’s” on its inputs gives a logic “1” at the output, otherwise is at logic level “0”.

(a) Circuit symbol

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

(b) Truth table

$C = \overline{A \oplus B}$

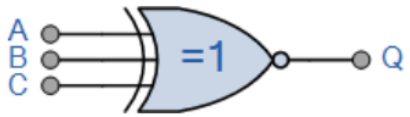
$= A \odot B$

(c) Boolean expression

Ex-NOR Gate

The 3-input Logic Ex-NOR Gate

Symbol	Truth Table			
	C	B	A	Q
	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
Boolean Expression $Q = \overline{A \oplus B \oplus C}$	Read as "any **EVEN** number of Inputs" gives Q			

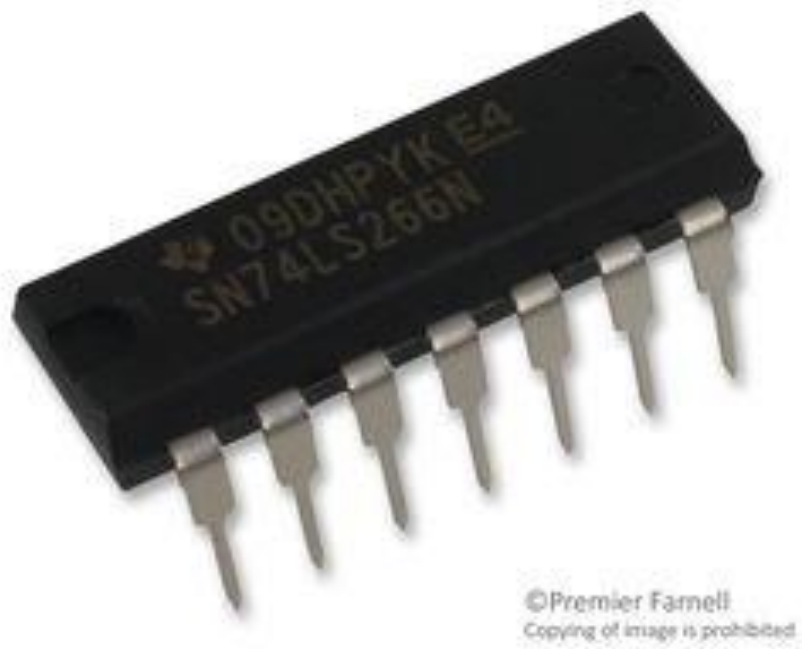


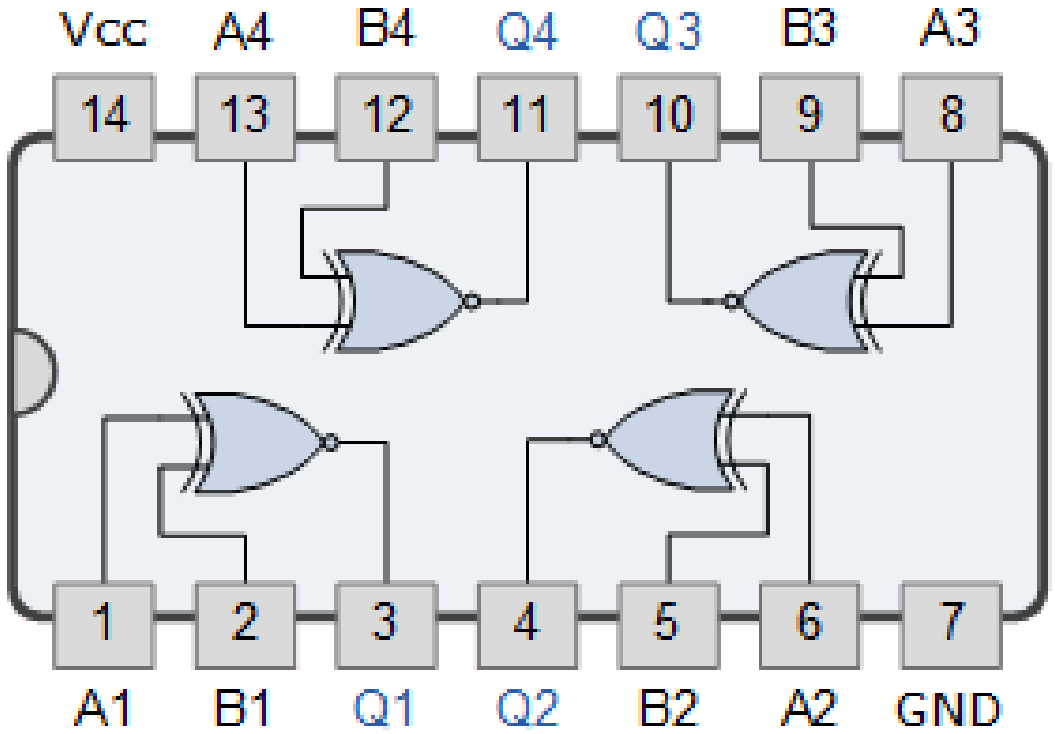
3-input Ex-NOR Gate

→ When no. of 1's in input is even, the o/p of Ex-NOR is 1.

Ex-NOR Gate IC: 74266

74266 Quad 2-input Ex-NOR Gate





Universal Gates

- A Universal Gate is a gate which can implement any Boolean function without need to use any other gate type.

- The NAND and NOR gates are universal gates.

- In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

Boolean Algebra

- Boolean algebra is used to express the effects that various digital circuits have on logic inputs, and to manipulate logic variable for the purpose of the determination of the best method for performing a given circuit function.
  - In formal logic, these values are "true" and "false."
  - In digital systems, these values are "on" and "off," 1 and 0, or "high" and "low."

Boolean Algebra (Ctd…)

- In Boolean algebra, there are only three basic operations:

  1. *Logical Addition*, also called as *OR addition* or simply the *OR operation*, symbolized as (+) .

  2. *Logical Multiplication*, also called as *AND multiplication* or simply the *AND operation*, symbolized as (.).

  3. *Logical Complementation or inversion*, also called as *NOT operation* and symbolized as (¯). (′)

Boolean Algebra Rules

1. Commutative Law:

- The *commutative law of addition* for two variables is written as:

$$A+B = B+A$$

- The *commutative law of multiplication* for two variables is written as:

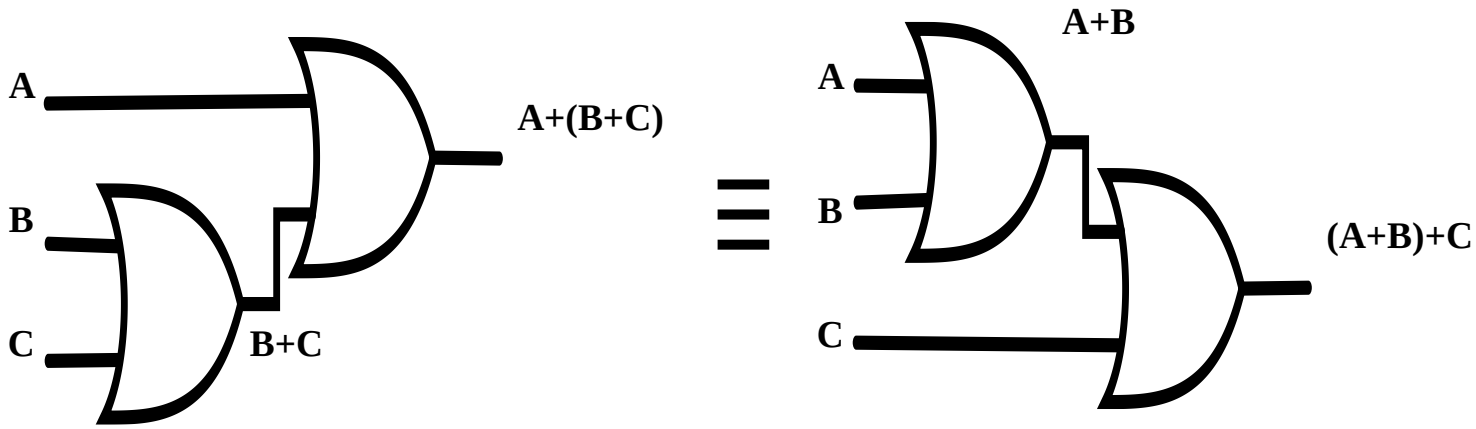
$$AB = BA \quad [A \cdot B = B \cdot A]$$

Boolean Algebra Rules

2. Associative Law:

- The *associative law of addition* for 3 variables is written as:

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

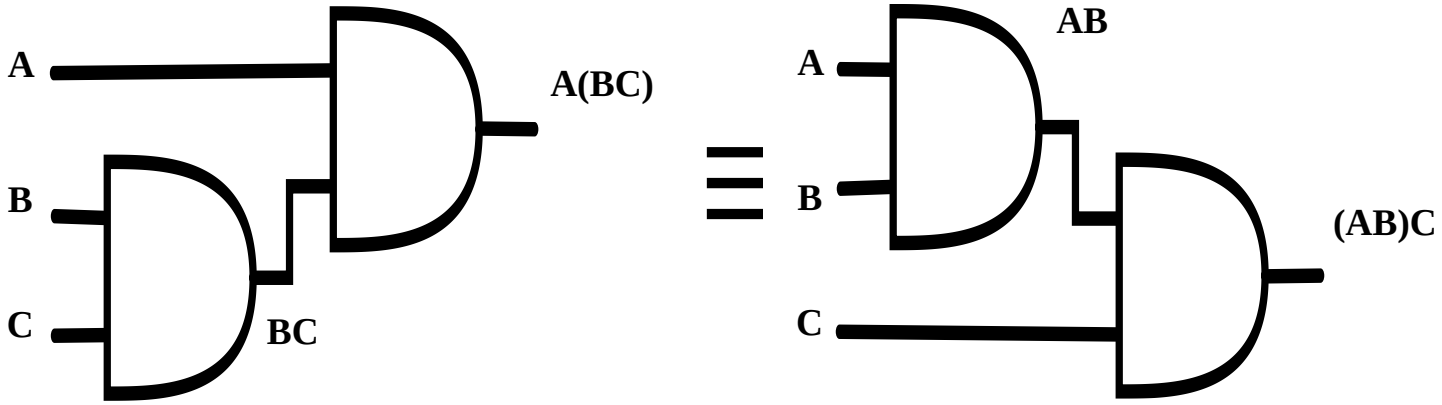


Boolean Algebra Rules

- The *associative law of multiplication* for 3 variables is written as:

$$A(BC) = (AB)C$$

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

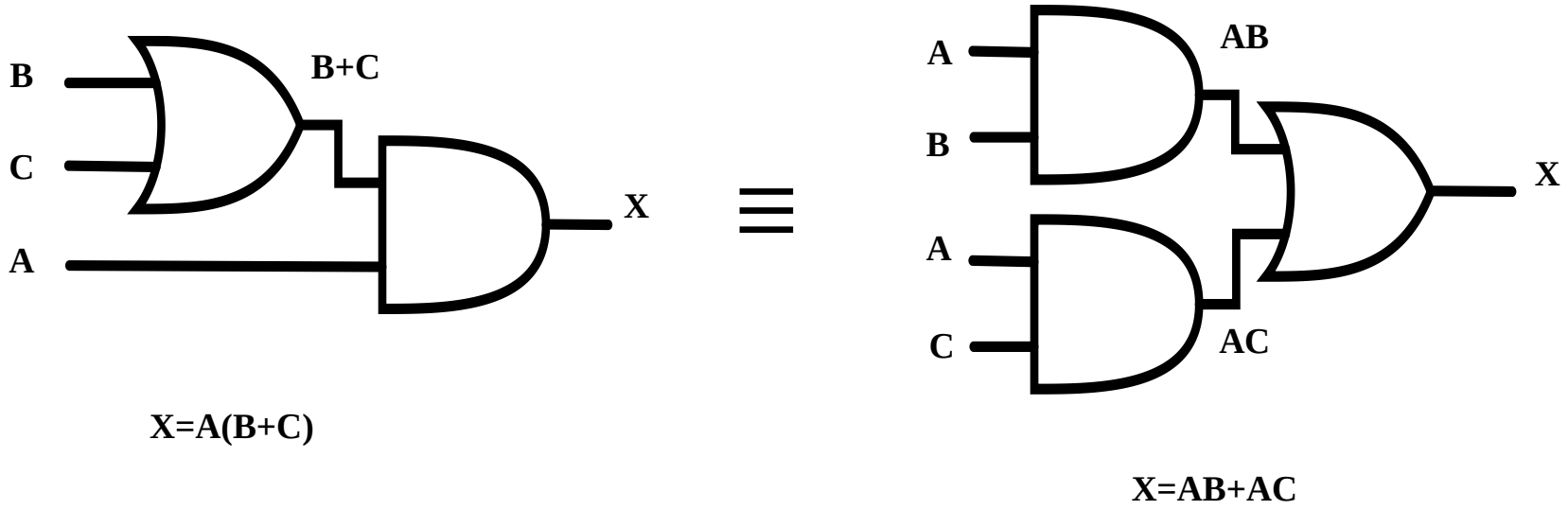


Boolean Algebra Rules

3. Distributive Law:

- The *distributive law* is written for 3 variables as follows:

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



Rules of Boolean Algebra

- Basic rules that are useful in simplification of boolean algebra expressions are as follows:

1. $A+0=A$
2. $A+1=1$
3. $A \bullet 0=0$
4. $A \bullet 1=A$
5. $A+A=A$
6. $A+\overline{A}=1$
7. $A \bullet A=A$
8. $A \bullet \overline{A}=0$
9. $\overline{\overline{A}}=A$
10. $A+AB=A$
11. $A+\overline{A}B=A+B$
12. $(A+B)(A+C)=A+BC$

1. $1+0=1$, $0+0=0$ $\}$ $A+0=A$

2. $1+1=1$, $0+1=1$ $\}$ $A+1=1$

3. $1 \cdot 0=0$, $0 \cdot 0=0$ $\}$ $A \cdot 0=0$

4. $1 \cdot 1=1$, $0 \cdot 1=0$ $\}$ $A \cdot 1=A$

5. $1+1=1$, $0+0=0$ $\}$ $A+A=A$

6. $1+0=1$, $0+1=1$ $\}$ $A+\overline{A}=1$

7. $1 \cdot 1=1$, $0 \cdot 0=0$ $\}$ $A \cdot A=A$

8. $1 \cdot 0=0$, $0 \cdot 1=0$ $\}$ $A \cdot \overline{A}=0$

9. $A=1, \overline{A}=0, \overline{\overline{A}}=1$; $A=0, \overline{A}=1, \overline{\overline{A}}=0$ $\}$ $\overline{\overline{A}}=A$

10. $A+AB=A(1+B)$
$=A(1)$
$=A$

11. $A + \bar{A}B$

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

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

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

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

$= A + B(1)$

$= A + B$

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

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

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

$= A + AC + AB + BC$

$= A(1+C) + AB + BC$

$= A + AB + BC$

$= A(1+B) + BC$

$= A + BC$

$\therefore \quad (A+B)(A+C) = A + BC$

De-Morgan's Theorem

- De-Morgan's Theorems are two additional simplification techniques that can be used to simplify Boolean expressions. Again, the simpler the Boolean expression, the simpler the resulting logic.

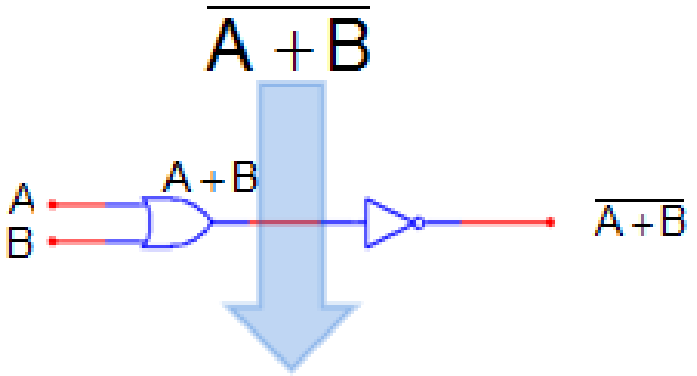
$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

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

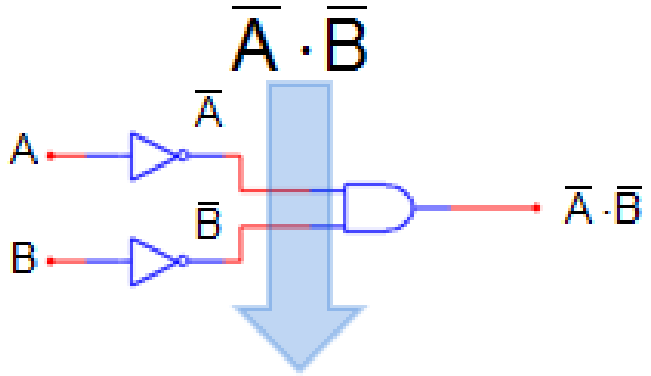
De-Morgan's Theorem 1

- Proof:

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



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



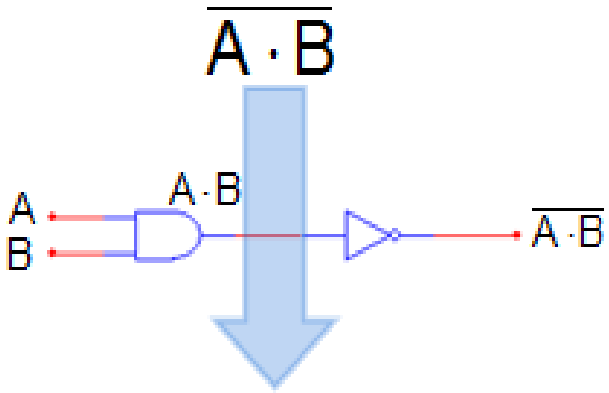
A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot \overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

The truth-tables are equal; therefore, the Boolean equations must be equal.

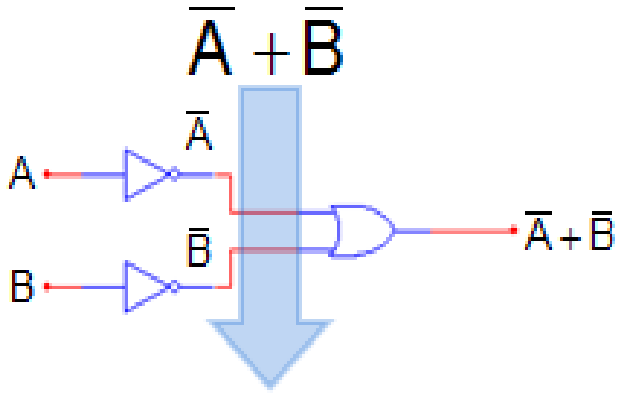
De-Morgan's Theorem 2

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

- Proof:



A	B	A·B	$\overline{A \cdot B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0



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

The truth-tables are equal; therefore, the Boolean equations must be equal.

Duality Theorems

- Duality underlies all Boolean algebra. Each boolean expression has its dual which is true as the original expression. The dual of a given Boolean expression can be obtained as:
  1. Change each OR sign to and AND sign
  2. Change each AND sign to and OR sign
  3. Complement any 0 or 1 appearing in the expression

Example -

A + 0 = A
A · 1 = A

} Duals of each other

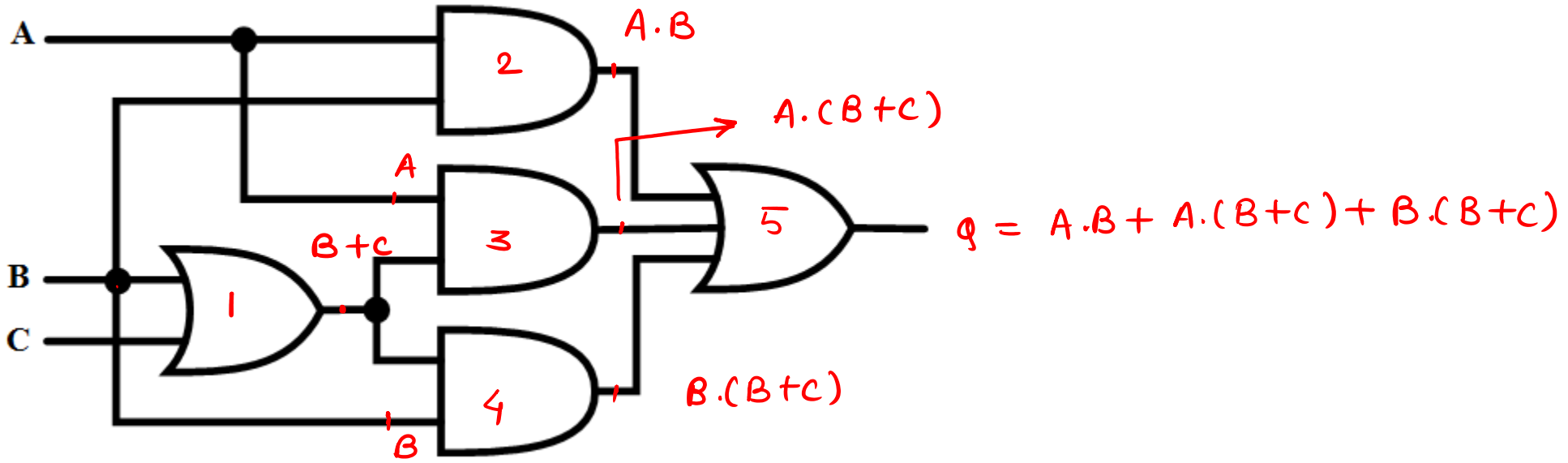
- Examples:

A+0=A	=>	A.1=A
A+ 1=1 A·0=0	=>	A.0=0
A+B=B+A	=>	A.B=B.A
A+AB=A	=>	A(A+B)=A
A(B+C)=AB+AC	=>	A+B.C=(A+B).(A+C)

Simplification of Boolean Expressions

- A simplified Boolean expression uses the fewest gates possible to implement a given expression.
- Example#1:

  Consider this expression:



Example #1(Ctd…)

- Simplify the expression using Boolean algebra.

AB+A(B+C)+B(B+C)

= AB+AB+AC+BB+BC

(Distributive Law)

= AB+AB+AC+**B**+BC

(BB=B)

= **AB**+AC+B+BC

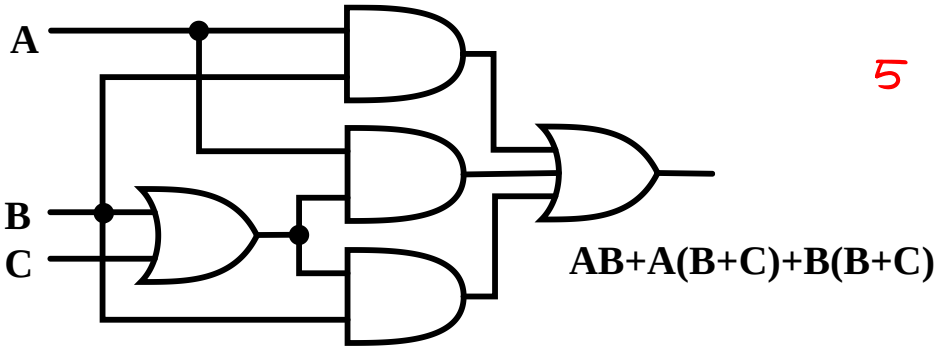
(AB+AB=AB)

= AB+AC+**B**

(B+BC=B)

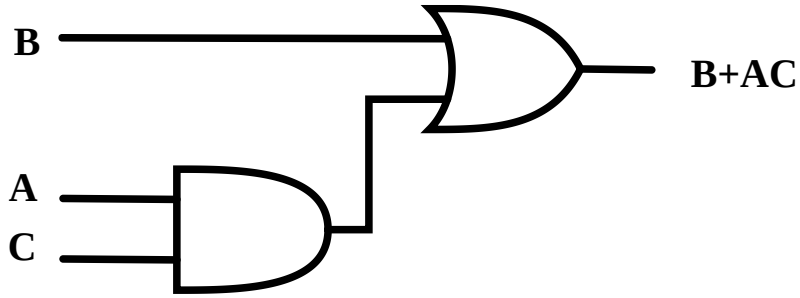
= **B**+AC

(AB+B=B)



5 Gates

Reduction in hardware



2 Gates

Example#2

$$\overline{(AB)}\,(\overline{A} + B)\,(\overline{B} + B)$$

Expression	Rule(s) Used
$\overline{(AB)}\,(\overline{A} + B)\,(\overline{B} + B)$	Original Expression
$\overline{(AB)}\,(\overline{A} + B)\,(1)$	Compliment law, Identity law
$(\overline{A} + B)\,(\overline{A} + B)$	De Morgan's Law (AND with 1 was dropped because A*1=A)
$\overline{A}\,\overline{A} + \overline{A}\,B + \overline{B}\,\overline{A} + \overline{B}\,B$	Distributive Law
$\overline{A} + \overline{A}(B + \overline{B}) + 0$	Redundancy (on $\overline{A}$s), Identities
$\overline{A} + \overline{A}(1)$	
$\overline{A}$	

Example#3

$$(A + C)(AD + \overline{AD}) + AC + C$$

Expression	Rule(s) Used
$(A + C)(AD + \overline{AD}) + AC + C$	Original Expression
$(A + C)A(D + \overline{D}) + AC + C$	Distributive.
$(A + C)A + AC + C$	Compliment, Identity.
$AA + AC + AC + C$	Distributive.
$A + AC + C$	Redundancy
$A + C + C$	Identity $(A + AC = A + C)$
$A + C$	Redundancy

Example#4

$$A(A + \overline{BC}) + A(\overline{B} + C)$$

Expression

$$A(A + \overline{BC}) + A(\overline{B} + C)$$

$$AA + A\overline{BC} + A\overline{B} + AC$$

$$A + A\overline{BC} + A\overline{B} + AC$$

$$A + A\overline{B}(C + 1) + AC$$

$$A + A\overline{B} + AC$$

$$A\ (1 + \overline{B} + C)$$

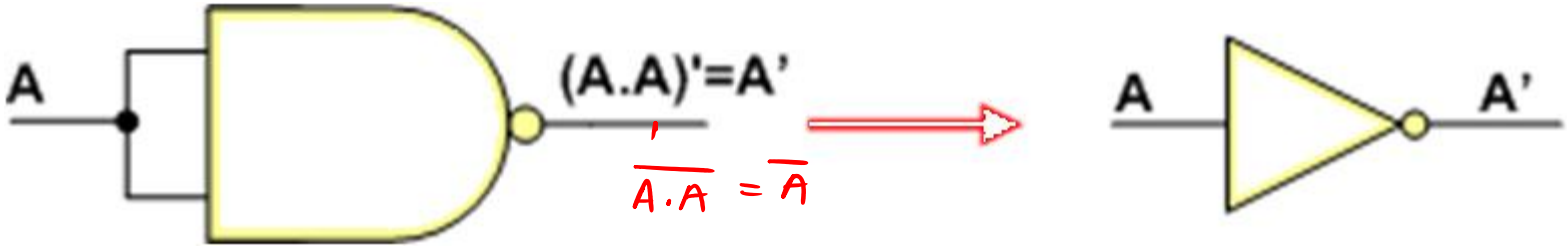
$$A$$

NAND as Universal Gate

$Q = \overline{A \cdot B}$

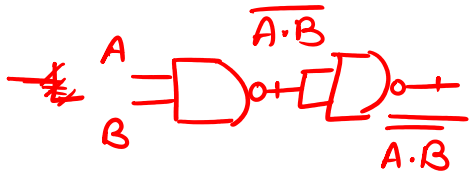
- Implementing Inverter with only NAND gates: (NOT) $Q = \overline{A}$

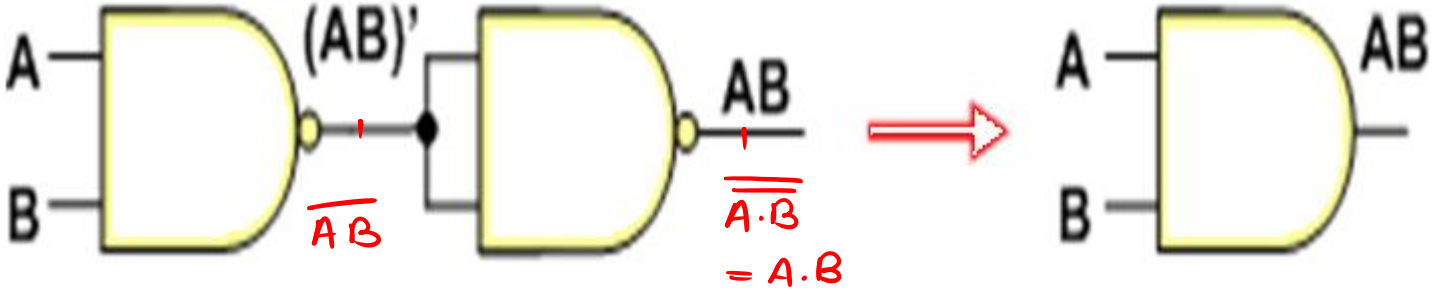
$Q = \overline{A \cdot A} = \overline{A}$



$\overline{\overline{A}} = A$

- Implementing AND gate with only NAND gates: $Q = \overline{\overline{A \cdot B}} = (\overline{A \cdot B})$

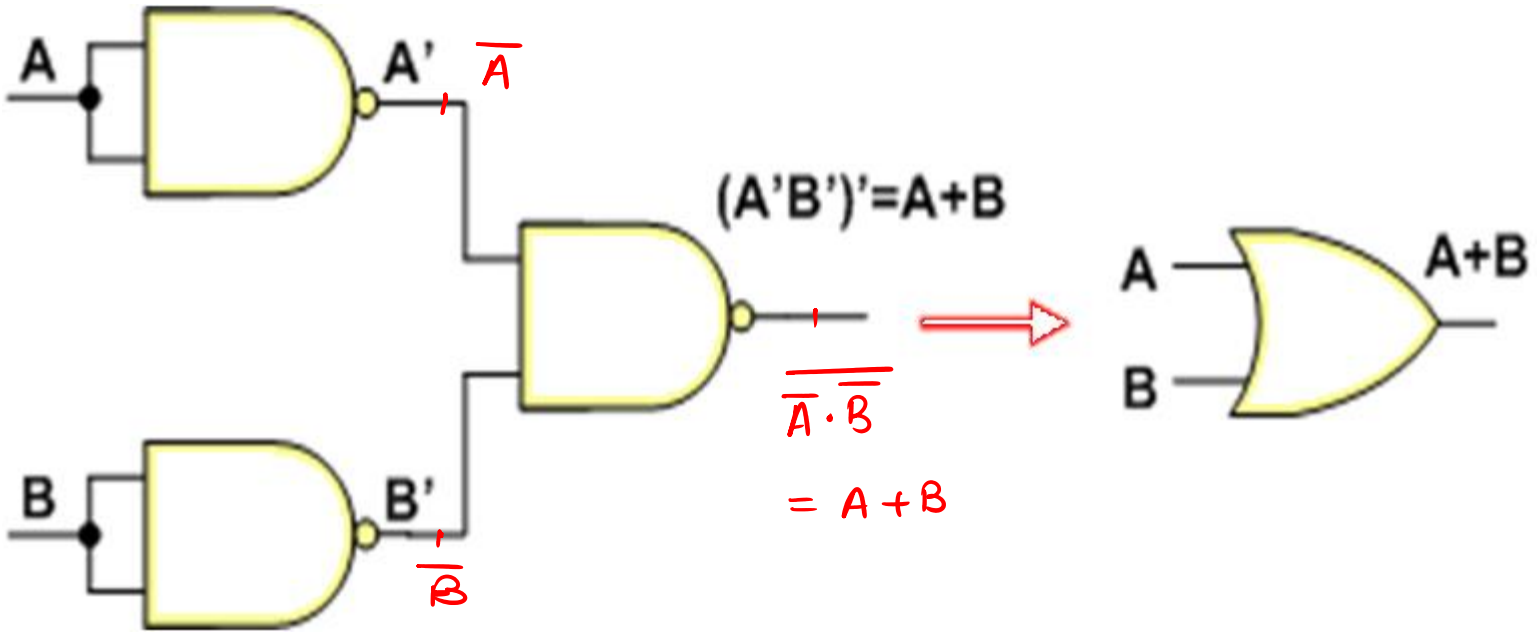




NAND as Universal Gate

- Implementing OR gate with only NAND gates

$Q = A + B = \overline{(\overline{A+B})} = \overline{\overline{A}\cdot\overline{B}}$ (3; 1, 2)



NAND as Universal Gate

- Implementing Ex-OR gate with only NAND gates

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

① ②

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

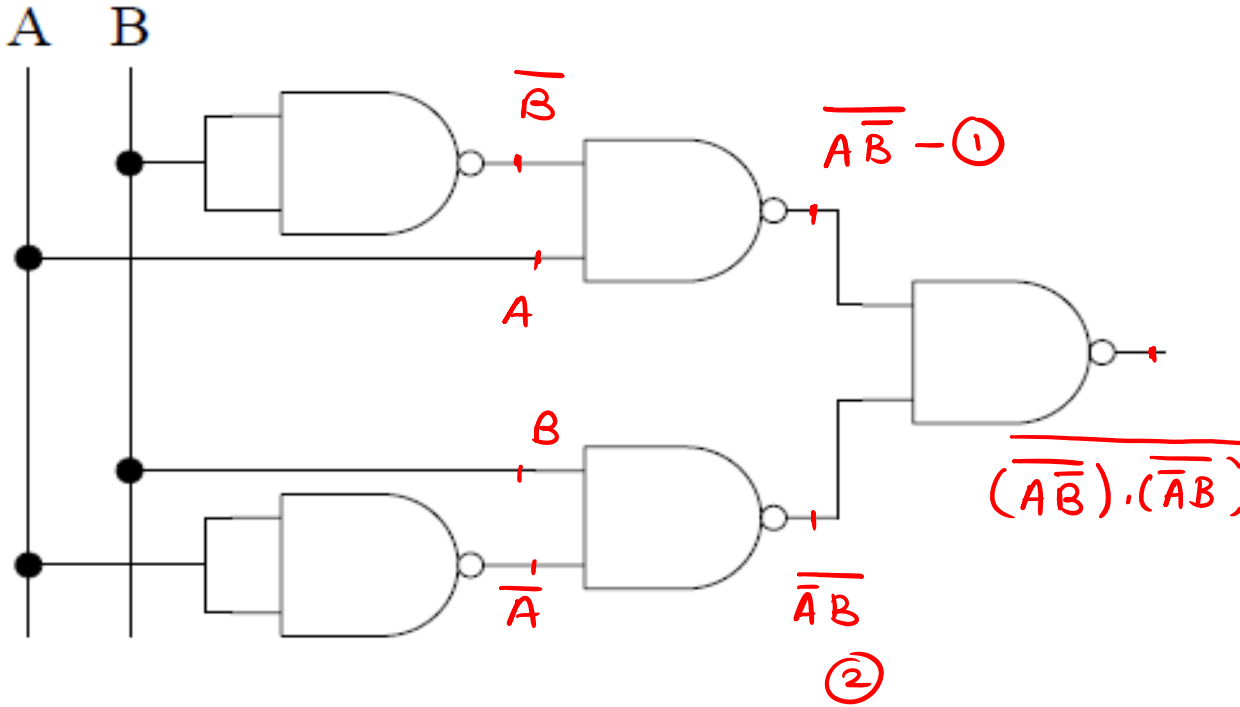
$= \overline{\overline{\overline{A}B + A\overline{B}}}$

$= \overline{(\overline{\overline{A}B}) \cdot (\overline{A\overline{B}})}$

De-Morgan's Thm. → 5

1 3 2 4

5 NAND gates



NAND as Universal Gate

- Implementing Ex-NOR gate with only NAND gates

$$Q = AB + \overline{A} \cdot \overline{B}$$

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

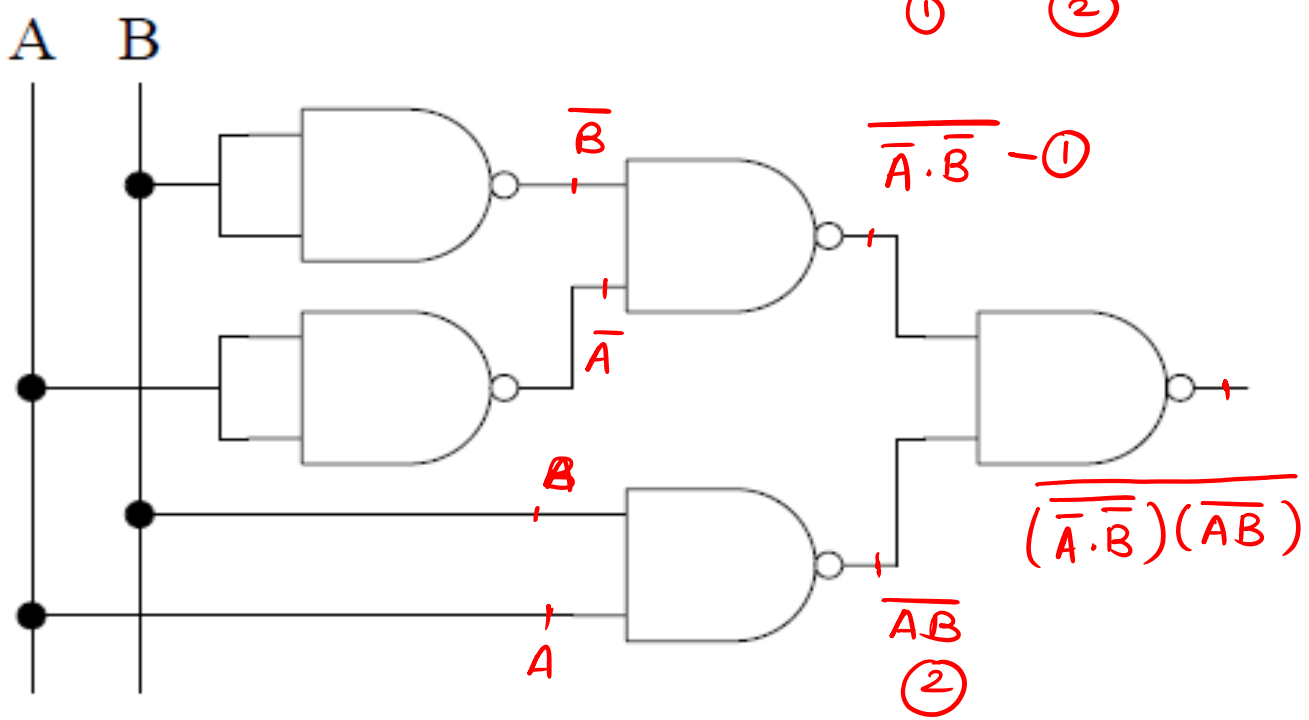
$$= \overline{(\overline{AB}) \cdot (\overline{\overline{A} \cdot \overline{B}})}$$

(3) (1) (2) (4) (5)

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

$$= \overline{\overline{\overline{A}\,\overline{B}} \cdot \overline{A\,B}}$$

① ②



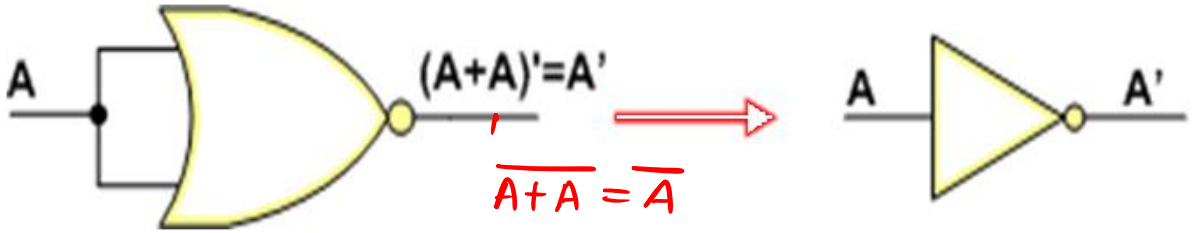
NOR as Universal Gate

$\overline{A+B}$

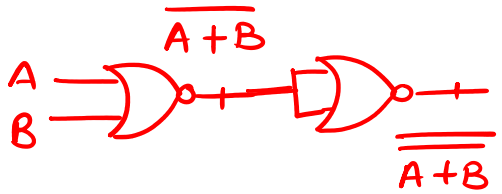
(NOT)

- Implementing Inverter with only NOR gates: $Q = \overline{A}$

$Q = \overline{A+A} = \overline{A}$



- Implementing OR gate with only NOR gates: $Q = A+B = \overline{\overline{A+B}}$



NOR as Universal Gate

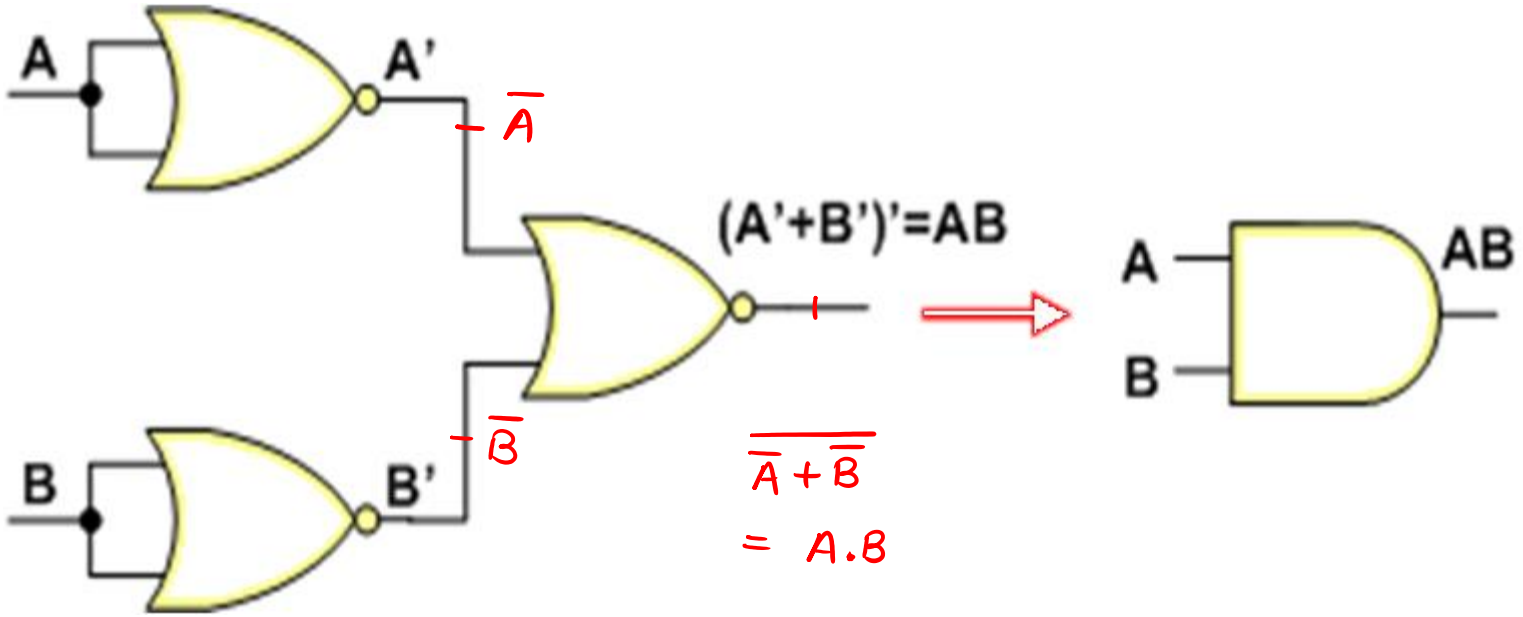
- Implementing AND gate with only NOR gates

$Q = A \cdot B$

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

$= \overline{\overline{A} + \overline{B}}$ → 3

(1) (1) (1) (2)



NOR as Universal Gate

- Implementing Ex-OR gate with only NOR gates

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

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

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

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

$= \overline{(\overline{A}B)\cdot(A\overline{B})}$

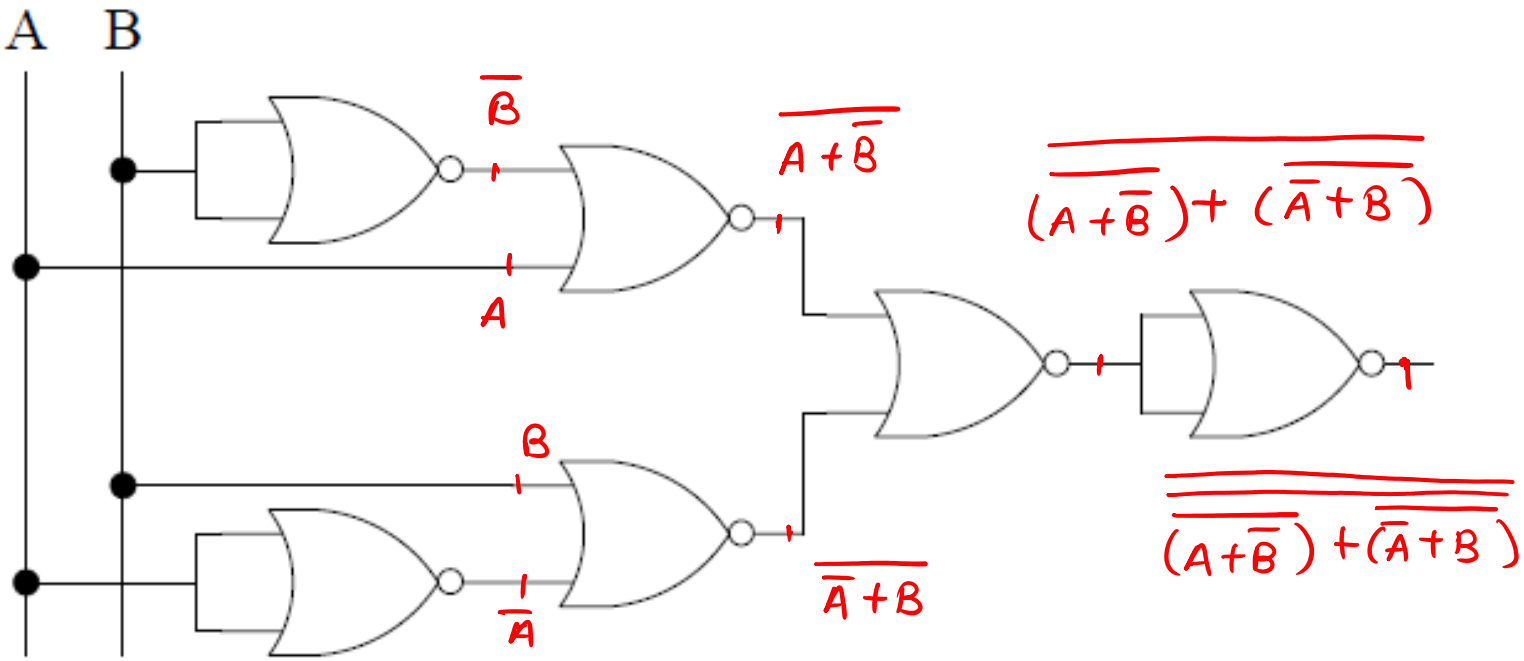
$= \overline{(A+\overline{B})\cdot(\overline{A}+B)}$

$= \overline{(A+\overline{B})} + \overline{(\overline{A}+B)}$ → 4

(3, 2, 1, 1)

NOR → 6

$= \overline{\overline{(A+\overline{B})} + (\overline{A}+B)}$ ← 6, ← 5



NOR as Universal Gate

Example :
F = AB + CD
NAND = 3
NOR = 8

$F = \overline{AB + CD}$

$= \overline{(\overline{AB})(\overline{CD})}$

$= (\overline{A} + \overline{B})(\overline{C} + \overline{D})$

$= \overline{(\overline{A} + \overline{B})} + \overline{(\overline{C} + \overline{D})}$

- Implementing Ex-NOR gate with only NOR gates

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

$Q = AB + \overline{A}\,\overline{B}$

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

$= \overline{(\overline{AB}) \cdot (\overline{\overline{A}\,\overline{B}})}$

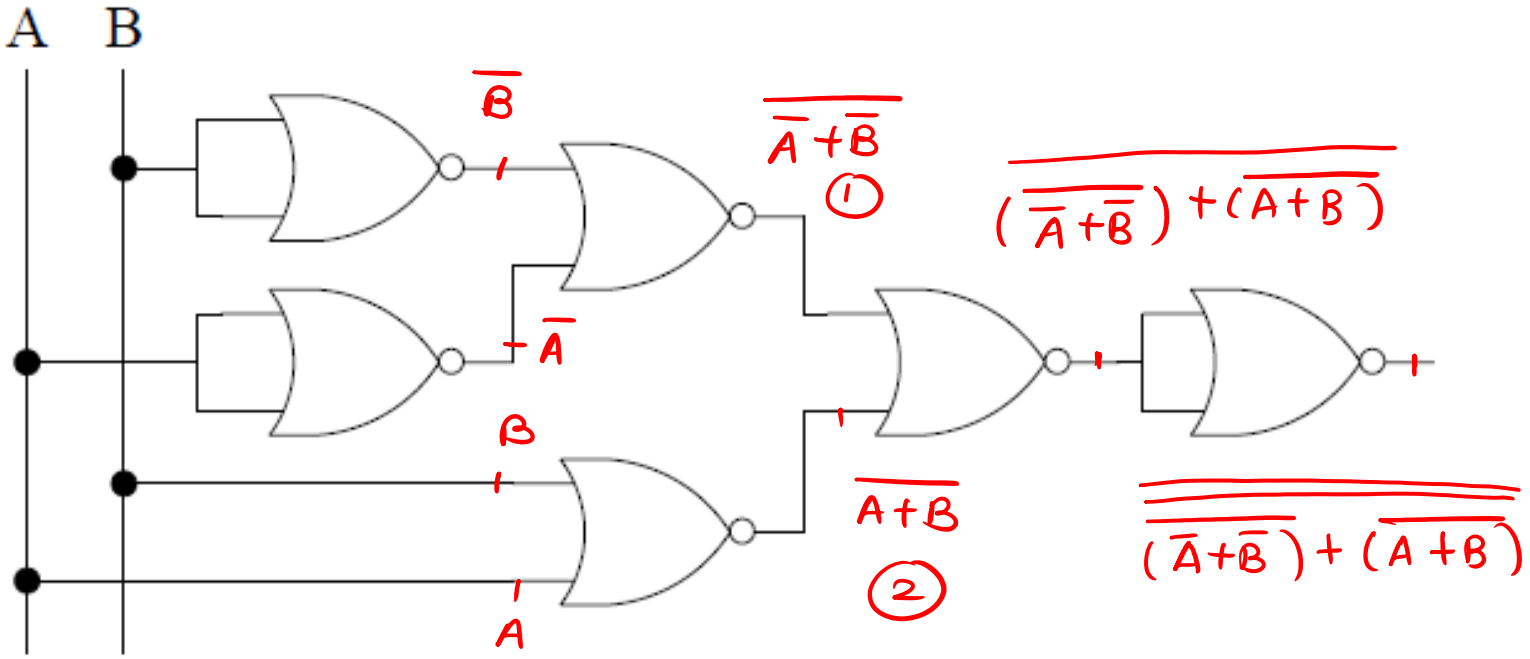
$= \overline{(\overline{A} + \overline{B}) \cdot (A + B)}$

$= (\overline{\overline{A} + \overline{B}}) + (\overline{A + B})$

$= \overline{(\overline{A} + \overline{B})} + \overline{(A + B)}$

3 — 4 — 5 — 6

1 1 2 — 2 — 1



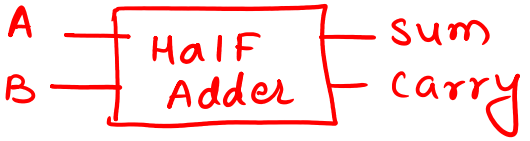
6 NOR Gates

Half Adder

Digital Circuits-
1] Combinational ckt : The o/p depends only on present i/ps
2] Sequential ckt : The o/p depends on present and past i/ps
↳ Memory.

- Half Adder is a combinational circuit that performs the addition of two bits.

- This circuit needs two binary inputs and two binary outputs.



A B S C

- The half adder takes two single bit binary numbers and produces a sum and a carry–out, called “carry”.

Half Adder

- Truth Table of half adder is

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

Ex-OR

AND

- The equations for Sum and Carry are

$$\text{Sum} = A \oplus B$$

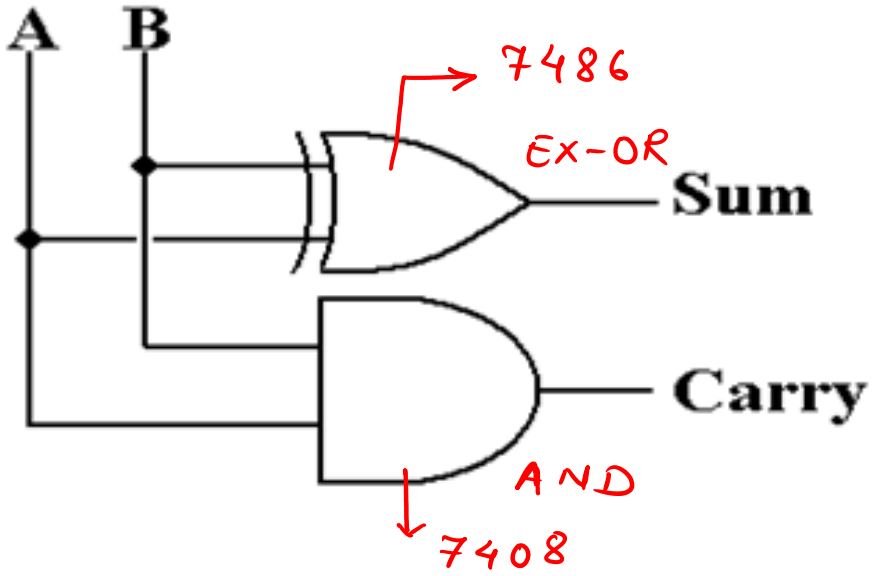
$$\text{Carry} = A.B$$

$$\text{Sum} = A \oplus B = \bar{A}B + A\bar{B}$$

$$\text{Carry} = A \cdot B$$

Implementation of Half Adder

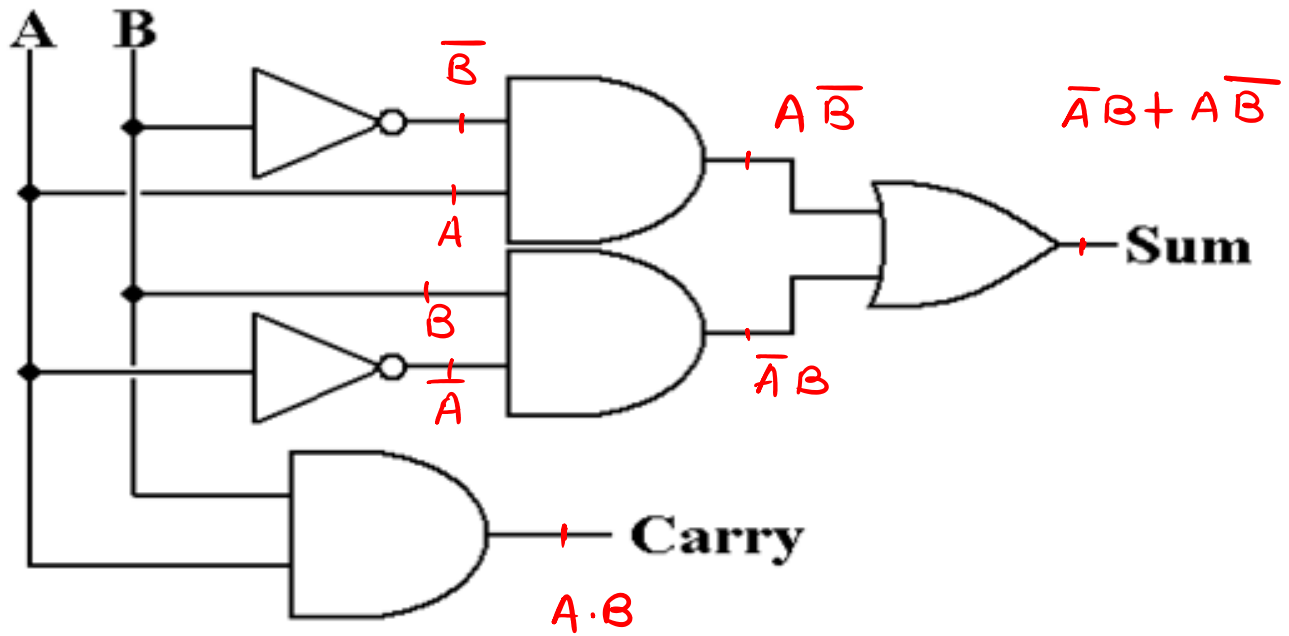
- Using Ex-OR and AND gates:



Implementation of Half Adder

- Using basic logic gates

AND, NOT, OR



* Implement Half Adder using only NAND gates.

Implementation of Half Adder

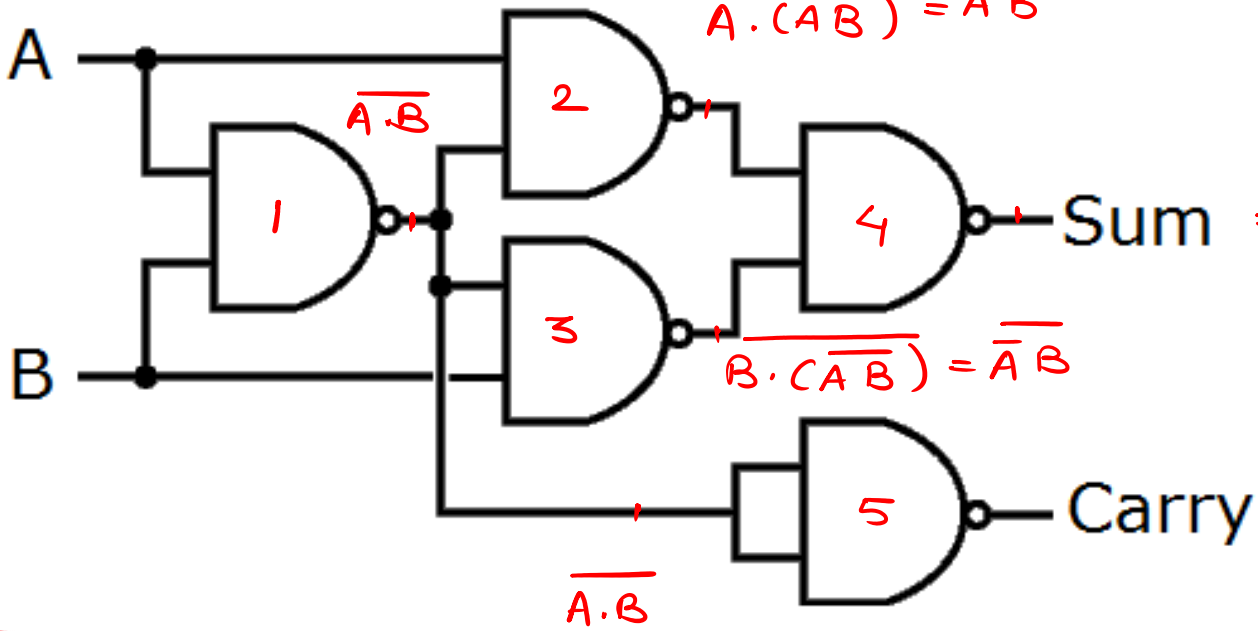
$Q = \overline{\overline{A}B + A\overline{B}} = (\overline{\overline{A}B})(\overline{A\overline{B}})$

- Using NAND gates

NAND gate 1
$= \overline{A \cdot B}$

NAND gate 2
$= \overline{A \cdot (\overline{A}B)}$
$= \overline{A \cdot (\overline{A} + \overline{B})}$
$= \overline{A \cdot \overline{A} + A\overline{B}}$
$= \overline{A\overline{B}}$ (0)

NAND gate 3
$= \overline{B \cdot (\overline{A}B)} = \overline{B \cdot (\overline{A} + \overline{B})}$
$= \overline{\overline{A}B + B\overline{B}}$ ($B\overline{B}$ = 0)
$= \overline{\overline{A}B}$



NAND gate 4
$= (\overline{A\overline{B}}) \cdot (\overline{\overline{A}B})$

$Sum = A \oplus B$

NAND gate 5
$= \overline{A \cdot B}$
$= A \cdot B$

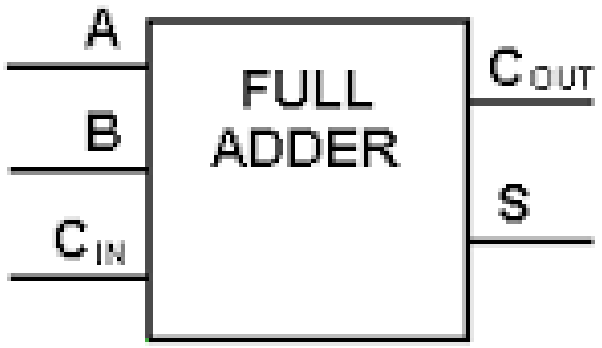
$Carry = A \cdot B$

- Minimum NAND gates for Ex-OR = 4
- Minimum NAND gates for Half Adder = 5

Full Adder

- Half adder has only two inputs and there is no provision to add ‘Carry’ coming from lower the lower bit order when multi-bit addition is performed.
- Full adder is a logic circuit that adds 3 bits- two bits to be added and a carry bit from lower order, which results in Sum and Carry as shown in figure below:

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



Carry out

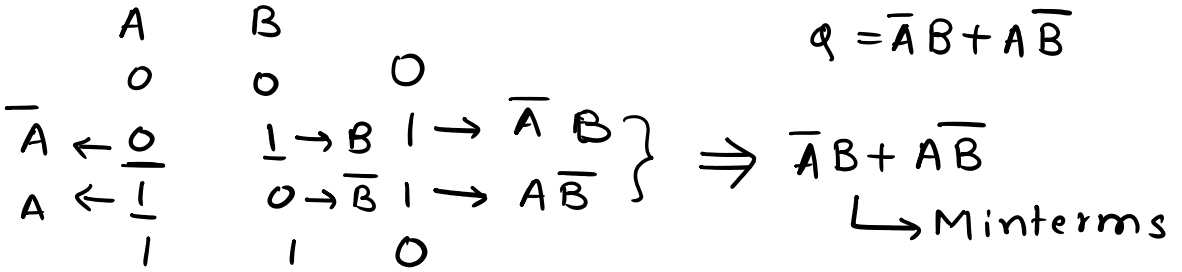
Sum

Cin

Cin = Carry from lower bit

Full Adder

- Truth Table of Full Adder is



Half Adder:

Sum = A ⊕ B

Cout = A · B

Full Adder:

Sum = A ⊕ B ⊕ Cin

Cout = $\bar{A}$BCin + A$\bar{B}$Cin + AB$\bar{Cin}$ + ABCin

= AB + BCin + ACin

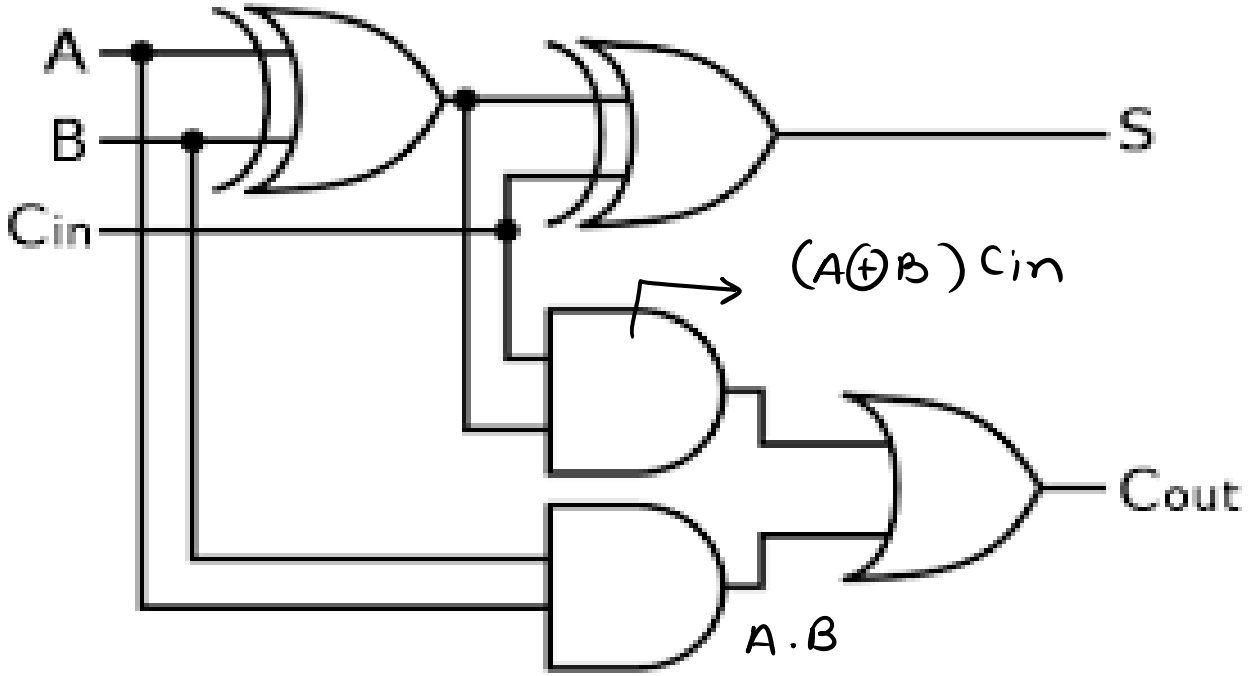
A	B	C	Sum	Carry	
0	0	0	0	0	
0	0	1	1	0	
0	1	0	1	0	
0 $\bar{A}$	1 B	1 Cin	0	1	→ $\bar{A}$BCin
1	0	0	1	0	
1	0	1	0	1	→ A$\bar{B}$Cin
1	1	0	0	1	→ AB$\bar{Cin}$
1	1	1	1	1	→ ABCin

Implementation of Full Adder

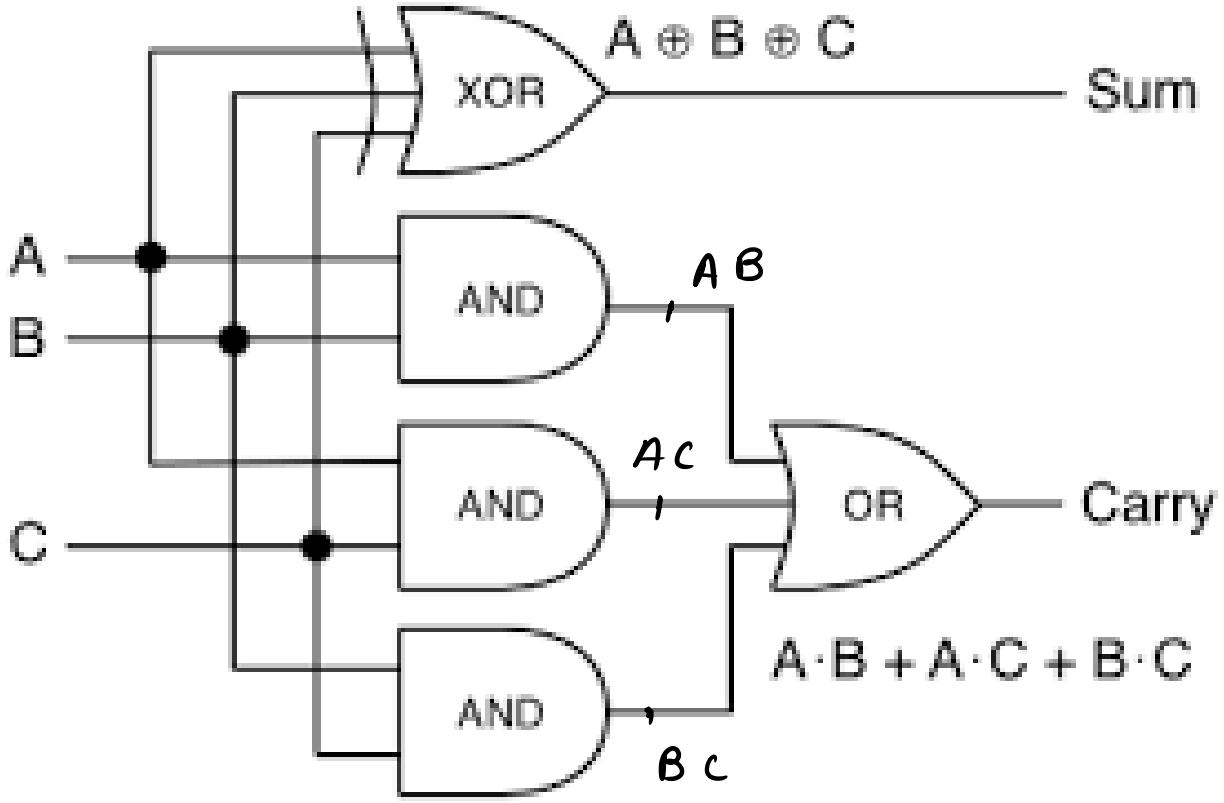
- The expressions for Sum and Carry are

$$Sum = A \oplus B \oplus Cin$$

$$Cout = A.B + B.Cin + A.Cin$$



Implementation of Full Adder



Acknowledgements

Modern Digital Electronics: R P Jain

Web sources

Thank you!