Digital Electronics

Lecture-1/Digital Electronics/Unit-4

Topics to be covered

*Introduction

*Analog & Digital Circuits

*Radix Or Base

*Digital Number System

*Binary Number System

*Octal Number System

*Hexadecimal Number System

Digital Electronics

- Uses for performing numerical computations quickly and accurately.
- Uses Diodes and Transistors as switches to chang from one voltage level to another.
- Uses two basic symbols (0 and 1) to represent information.
- e.g.. CD players,Stereos,TVs, Telephones, etc.

Analog Circuits

- voltages and currents vary continuously throughout some range.
- The output could vary through an infinite no. of possible values .
- e.g. Sine wave generators, power supplies, radio receivers, etc.

Digital Circuits

* The output voltage can have only two values or states.

  ON or OFF

  HIGH or LOW

  TRUE or FALSE

  1 or 0

* e.g. CD players, Stereos, TVs, etc.

Radix Or Base

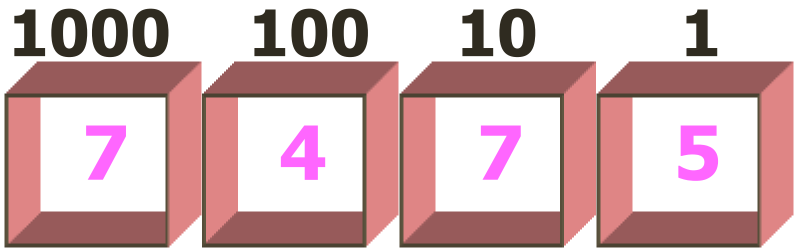
The number of digits used in the number system is called the **radix** or the **base** of the number system.

• Decimal	Base = 10
• Binary	Base = 2
• Octal	Base = 8
• Hexadecimal (Hex)	Base = 16

Decimal Number System

Base (Radix)	10
Digits	0,1,2,3,4,5,6,7,8,9
e.g.	7475_{10}

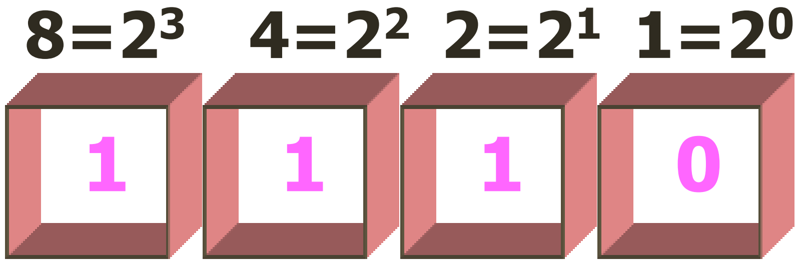
The magnitude represented by a digit is decided by the position of the digit within the number.



For example the digit 7 in the left- most position of 7475 counts for 7000 and the digit 7 in the second position from the right counts for 70.

Binary Number System

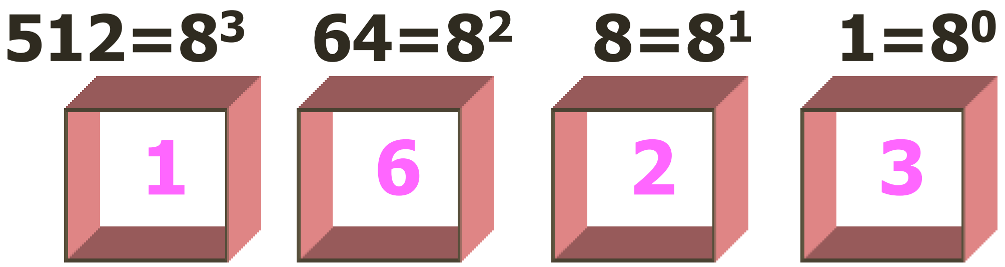
Base (Radix)	2
Digits	0, 1
e.g.	1110_2



The digit 1 in the third position from the right represents the value 4 and the digit 1 in the fourth position from the right represents the value 8.

Octal Number System

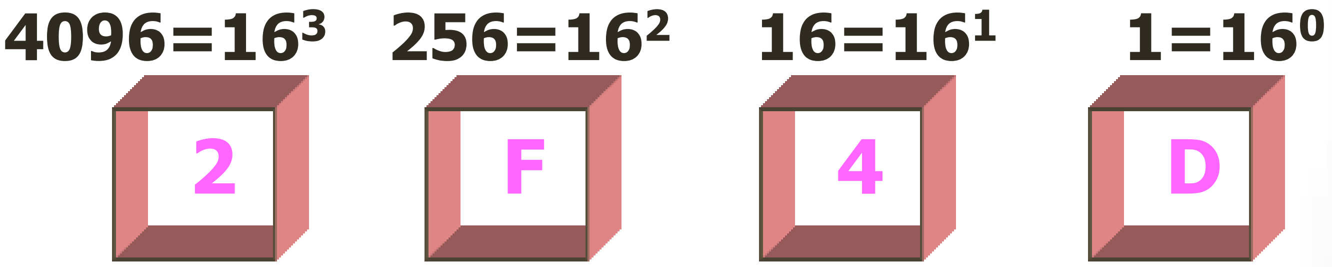
Base (Radix)	8
Digits	0, 1, 2, 3, 4, 5, 6, 7
e.g.	1623_8



The digit 2 in the second position from the right represents the value 16 and the digit 1 in the fourth position from the right represents the value 512.

Hexadecimal Number System

Base (Radix)	16
Digits	0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F
e.g.	$2F4D_{16}$



The digit F in the third position from the right represents the value 3840 and the digit D in the first position from the right represents the value 1.

Binary Arithmetic

Lecture No. -2/Digital Electronics/Unit-4

Topics to be covered

*Binary Addition

*Binary Complements

* Binary Subtraction

•Binary Multiplication

*Binary Division

Binary Addition

(a)
$$\begin{array}{r} 0 \\ +\;0 \\ \hline 0 \\ \hline \end{array}$$

(b)
$$\begin{array}{r} 0 \\ +\;1 \\ \hline 1 \\ \hline \end{array}$$

(c)
$$\begin{array}{r} 1 \\ +\;0 \\ \hline 1 \\ \hline \end{array}$$

(d)
$$\begin{array}{r} 1 \\ +\;1 \\ \hline 1\,0 \\ \hline \end{array}$$

Carry Bit

Binary Addition Examples

(a)
```
   1011
 + 1100
 ------
  10111
```

(b)
```
   1010
 +  100
 ------
   1110
```

(c)
```
   1011
 +  101
 ------
  10000
```

(d)
```
    101
 + 1001
 ------
   1110
```

(e)
```
   10011001
 +   101100
 ----------
   11000101
```

Binary Complement (1's Complement) Operation

$$1 \longrightarrow 0$$

$$0 \longrightarrow 1$$

Example

$$\begin{array}{ccccccccc} 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}$$

Two's Complement

The Two's complement of a binary number is obtained by first complementing the number and then adding 1 to the result.

```
 1001110

 0110001  ⟵  One's Complement
+      1
―――――――――
 0110010  ⟵  Two's Complement
```

Binary Subtraction

Binary subtraction is implemented by adding the Two's complement of the number to be subtracted.

Example

$$\left.\begin{array}{r} 1101 \\ -1001 \end{array}\right\} \qquad \begin{array}{r} 1101 \\ +0111 \\ \hline 10100 \end{array}$$

Two's complement of 1001

If there is a carry then it is ignored. Thus, the answer is 0100.

Binary Multiplication

(a)
$$\begin{array}{r} 0 \\ \times\, 0 \\ \hline 0 \\ \hline \end{array}$$

(b)
$$\begin{array}{r} 0 \\ \times\, 1 \\ \hline 0 \\ \hline \end{array}$$

(c)
$$\begin{array}{r} 1 \\ \times\, 0 \\ \hline 0 \\ \hline \end{array}$$

(d)
$$\begin{array}{r} 1 \\ \times\, 1 \\ \hline 1 \\ \hline \end{array}$$

Binary Multiplication

Multiplication of multibit binary numbers involves forming the partial products, shifting each successive partial product left one place, then adding all the partial products.

Example

```
  10101
  × 101
-------
1101001
```

Binary Division

Binary division is performed in the same manner as with decimal numbers.

Example

```
101)1101001( 10101 Quotient
    101
    -------
      110
      100
      -------
       001
       000
       -------
         010
         000
         -------
          100
          100
          -------
            0
```

Octal Arithmetic

* Addition
* Subtraction
* Multiplication
* Division

Octal Addition

The sum of two octal digits is the same as their decimal sum, provided the decimal sum is less than 8. If the sum is 8 or greater , we subtract 8 from it to obtain the octal digit. A carry 1 is generated.

Octal Addition

(a)
$$\begin{array}{r} 4 \\ +\ 2 \\ \hline 6 \\ \hline \end{array}$$

(b)
$$\begin{array}{r} 6 \\ +\ 7 \\ \hline 1\ 5 \\ \hline \end{array}$$

Carry Bit

(c)
$$\begin{array}{r} 5 \\ +\ 3 \\ \hline 1\ 0 \\ \hline \end{array}$$

Carry Bit

Carry Bit

(d)
$$\begin{array}{r} 11 \\ 256 \\ +\ 437 \\ \hline 715 \\ \hline \end{array}$$

Octal Subtraction

Octal subtraction is the same as that of decimal subtraction when we subtract a number less than 8. But if the number is 8 or greater, borrow 1 from next most significant digit, then convert the octal no into decimal and then perform the required operation.

Octal Subtraction

(a)

$$\begin{array}{r} 4 \\ -\ 2 \\ \hline 2 \\ \hline \end{array}$$

(b)

1 ← Borrow Bit

$$\begin{array}{r} 10 \\ -7 \\ \hline 1 \\ \hline \end{array}$$

(c)

1 ← Borrow Bit

$$\begin{array}{r} 76 \\ -67 \\ \hline 07 \\ \hline \end{array}$$

(d)

1 ← Borrow Bit

$$\begin{array}{r} 256 \\ -\ 127 \\ \hline 127 \\ \hline \end{array}$$

Octal Multiplication

Multiplication of octal numbers is performed by converting decimal partial products to octal.

Example

```
   15
 x 24
 ----
   64  (20-16=4, carry 2)
  32   (10-8=2, carry 1)
 ----
  404  (6+2=8-8=0, carry 1)
```

Octal Division

- Octal division is the same as that of decimal division.
- Partial product should be converted into Octal equivalent.

Octal Division

$(36)_8$ / (5)8 = $(6)_8$

```
5) 36 ( 6
   36      ( 30 - 24 = 6 carry 3 )
   --
   00
```

Therefore,

$(36)_8 / (5)_8 = (6)_8$

Octal Division

$(60236002)_8$ / $(675)_8$

```
675 ) 60236002 ( 67452
      5156
      ------
      06456
       6053
       ------
        4030
        3364
        ------
         4440
         4261
         ------
          1572
          1572
          ------
          0000
```

(60236002) / (675)8

=

(67452)8

Hexadecimal Arithmetic

* Addition
* Subtraction
* Multiplication
* Division

Hexadecimal Addition

The sum of two Hexadecimal digits is the same as their decimal sum, provided the decimal sum is less than 16. If the sum is 16 or greater , we subtract 16 from it to obtain the hexadecimal digit. A carry 1 is generated.

Hexadecimal Addition

(a)

$$\begin{array}{rcr} C & \rightarrow & 12 \\ +\;3 & \rightarrow & +\;\;3 \\ \hline F & & 15 \end{array}$$

(b)

$$\begin{array}{r} A \\ +\;2 \\ \hline C \end{array}$$

(c)

$$\begin{array}{r} E \\ +\;D \\ \hline 1\,B \end{array}$$

Carry Bit

(d)

$$\begin{array}{r} 11 \\ AC2 \\ +\;\;7BD \\ \hline 127F \end{array}$$

Carry Bit

Hexadecimal Subtraction

Hexadecimal subtraction is the same as that of decimal subtraction when we subtract a number less than 16. If the number is 16 or greater, borrow 1 from next most significant digit.

Hexadecimal Subtraction

(a)

$$\begin{array}{r} A \\ -\ 3 \\ \hline 7 \\ \hline \end{array}$$

(b)

$$\begin{array}{r} \overset{1}{A}2 \\ -\ F \\ \hline 93 \\ \hline \end{array}$$

$\leftarrow$ Borrow Bit

Hexadecimal Multiplication

Multiplication of hexadecimal numbers is performed by converting decimal partial products to hexadecimal

Example

```
   A
  x3
______
  1E (30-16=14, carry 1)
```

Hexadecimal Division

- Hexadecimal division is the same as that of decimal division.
- Partial product should be converted into Hexadecimal equivalent.

Hexadecimal Division

$(34)_{16} / (4)_{16} = (D)_{16}$

```
4) 34 ( D
    34      ( 52 - 48 = 4 carry 3 )
   ----
    00
```

Therefore,

$(34)_{16} / (4)_{16} = (D)_{16}$

Hexadecimal Division

Convert in Hexadecimal :

$(45B)_{16}$ / $(5)_{16}$

Answer is

$(DF)_{16}$

Conversions Between Number systems

Lecture No. -3/Digital Electronics/Unit-4

Topics to be covered

*Binary to Decimal Conversion

*Decimal to Binary Conversion

*Octal to Decimal Conversion

*Decimal to Octal Conversion

*Octal to Binary

*Binary to Octal Conversion

Binary to Decimal Conversion

- The Decimal equivalent of the Binary number is the sum of all its bits multiplied by the weights of their respective positions.

Binary Number :	1	1	0	1	0
Position Weights :	2^4	2^3	2^2	2^1	2^0

Thus ,

$$11010 = (1*2^4) + (1*2^3) + (0*2^2) + (1*2^1) + (0*2^0)$$

$$= 16 + 8 + 0 + 2 + 0$$

$$= (26)_{10}$$

Binary to Decimal Conversion Cont…

To convert the binary fraction into its decimal equivalent, we multiply each digit in the fraction successively by

2^{-1} 2^{-2} 2^{-3}…..etc

then add the products.

Binary Number	: 1	0	.	1	0	1
Position Weights :	2^{1}	2^{0}		2^{-1}	2^{-2}	2^{-3}

Thus ,

$$10.101 = (1 * 2^{1}) + (0 * 2^{0}) + (1 * 2^{-1}) + (0 * 2^{-2}) + (1 * 2^{-3})$$

$$= 2 + 2.5 + 0$$

$$= (2.625)_{10}$$

Decimal to Binary Conversion

- Only final bit determines if even/odd
  - If number is odd, final digit is 1
  - If number is even, final digit is 0
- To discover final binary digit, divide by 2
  - remainder (0 or 1) is binary digit
  - quotient can be used to repeat procedure
    - $2^1/2 = 1$, $2^8/2 = 128$, $2^{15}/2 = 16384$, ...

Decimal to Binary to Conversion Cont..

	106_{10}	
÷ 2 =	53	rem 0
÷ 2 =	26	rem 1
÷ 2 =	13	rem 0
÷ 2 =	6	rem 1
÷ 2 =	3	rem 0
÷ 2 =	1	rem 1
÷ 2 =	$\boxed{0}$	rem 1

read up

result is 1101010_2

stop when this result zero

Decimal to Binary to Conversion cont..

right hand bit is called the least significant bit (LSB)

$106_{10} = 1101010_2$

left hand bit is called the most significant bit (MSB)

Decimal to Binary Conversion Cont..

To convert the fraction part , repeatedly multiplying by 2 until the fractional part exactly equals zero or a sufficient number of bits have been obtained.

$(0.3125)_{10} = (?)_2$

0.3125	Whole Part
*2	
0.6250	0
*2	
1.2500	1
*2	
0.5000	0
*2	
1.0000	1

Read down

result is $.0101_2$

Octal to Decimal Conversion

- The Decimal equivalent of the Octal number is the sum of all its digits multiplied by the weights of their respective positions.

Octal Number :	1	0	3	3
Position Weights :	8^3	8^2	8^1	8^0

Thus ,

$$1033 = (1 * 8^3) + (0 * 8^2) + (3 * 8^1) + (3 * 8^0)$$

$$= 512 + 0 + 24 + 3$$

$$= (539)_{10}$$

Decimal to Octal Conversion

- To discover octal digit, repeatedly divide the no. by 8 write down the remainders after each division. By taking the remainders in reverse order,we obtain octal number.
- For decimal fractions, multiplied repeatedly by 8 and carry digits are written in integer position.

Decimal to Octal Conversion Cont..

357_{10}

$\div 8 = 48$ rem 7

$\div 8 = 5$ rem 6

$\div 8 = \boxed{0}$ rem 5

read up

result is 567_8

stop when this result zero

Decimal to Octal Conversion Cont..

To convert the fraction part , repeatedly multiplying by 8 until the fractional part exactly equals zero or until a sufficient number of bits have been obtained.

$(0.23)_{10} = (?)_8$

```
0.23
  *8        0.84   carry  1
------
1.84
0.84
  *8        0.72   carry  6
------
6.72
0.72
  *8        0.76   carry  5
------
5.76
```

result is 165_8

Read down

Octal to Binary Conversion

- Base of octal number system =8 =2^3
- Base of binary number system =2
- There is direct relation between 3-bit groups in a binary number and the octal digits.
- Octal digit can be replaced by a 3-bit binary equivalent.

Octal to Binary Conversion cont..

$(257)_8 = (?)_2$

$$\begin{tikzcd} 2 \arrow[d, dash] & 5 \arrow[d, dash] & 7 \arrow[d, dash] \\ 010 & 101 & 111 \end{tikzcd}$$

Therefore, $(257)_8 = (10101111)_2$

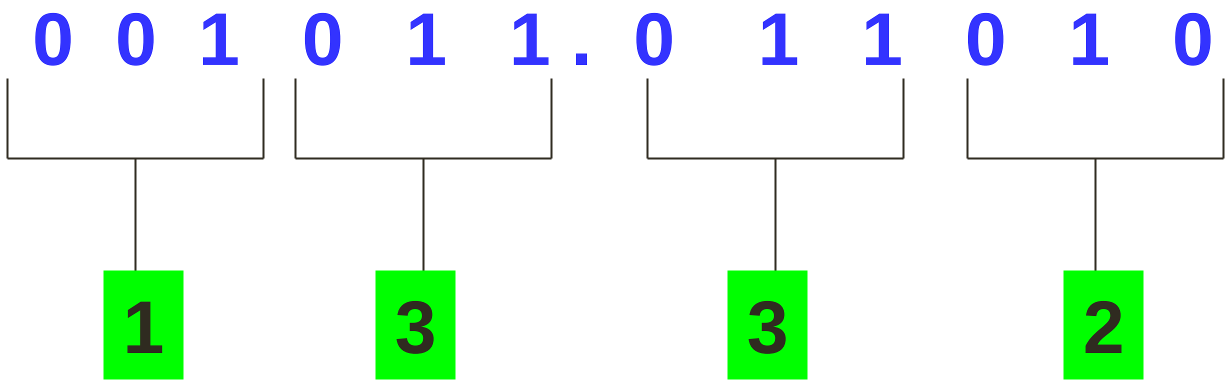
Octal to Binary Equivalent

Octal Digit	Binary Bits
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Binary to Octal Conversion

The Binary bits are partitioned into 3-bits and each group is replaced by its octal equivalent.

Example: $(1011.01101)_2 = (?)_8$



Therefore, $(1011.01101)_2 = (13.32)_8$

Hexadecimal to Decimal Conversion

The Decimal equivalent of the Hexadecimal number is the sum of all its digits multiplied by the weights of their respective positions.

Hexadecimal Number	: 4	7	3
Position Weights	: 16^2	16^1	16^0

Thus ,

$$473 = (4*16^2)+(7*16^1)+(3*16^0)$$
$$= 1024 + 112 + 3$$
$$= (1139)_{10}$$

Decimal to Hexadecimal Conversion

- To discover Hexadecimal digit, repeatedly divide the no. by 16 write down the remainders after each division. By taking the remainders in reverse order,we obtai Hexadecimal number.
- For decimal fractions, multiplied repeatedly by 16 and carry digits are written in integer position.

Decimal to Hexadecimal Conversion Cont..

$(379)_{10}=(?)_{16}$

	379	
÷ 16 =	23	rem B
÷ 16 =	1	rem 7
÷ 16 =	$\boxed{0}$	rem 1

stop when this result zero

read up

result is $17B_{16}$

Thus,

$(379)_{10} = (17B)_{16}$

Hexadecimal to Binary Conversion

* Base of Hexadecimal number system =16 =2^4
* Base of binary number system =2
* There is direct relation between **4-Bit groups** in a binary number and the **Hexadecimal digits**.

* **Hexadecimal digit** can be replaced by a **4-Bit** binary equivalent.

Hexadecimal to Binary Equivalent

Hexadecimal Digit	Binary Bits	Hexadecimal Digit	Binary Bits
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Hexadecimal to Binary Conversion cont..

$(7AB)_{16} = (?)_2$

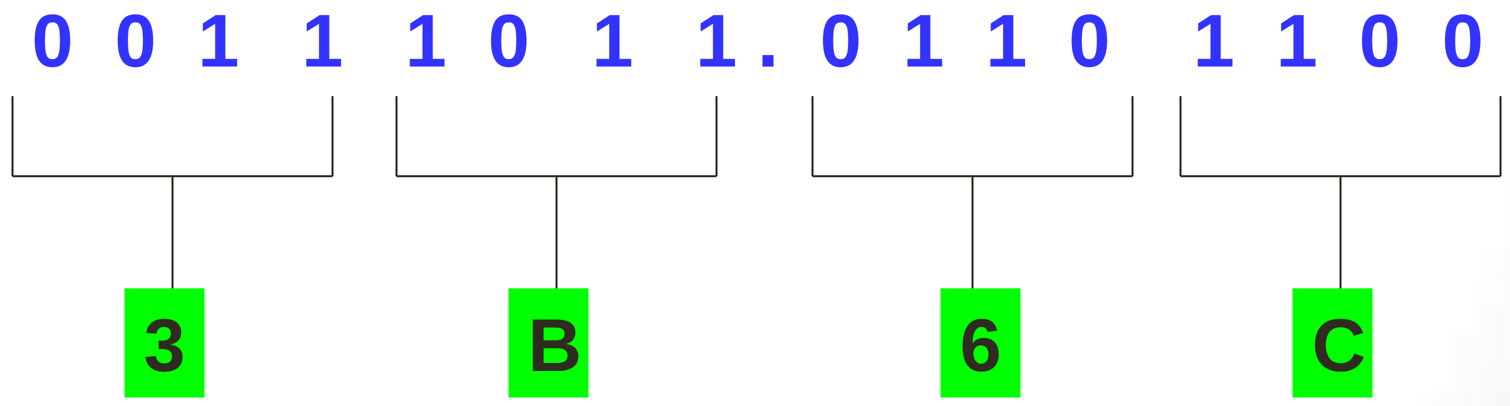
$$\begin{tikzcd}
7 \arrow[d, dash] & A \arrow[d, dash] & B \arrow[d, dash] \\
0111 & 1010 & 1011
\end{tikzcd}$$

Therefore, $(7AB)_{16} = (011110101011)_2$

Binary to Hexadecimal Conversion

The Binary bits are partitioned into 4-bits and each group is replaced by its Hexadecimal equivalent.

Example: $(111011.011011)_2 = (?)_{16}$



Therefore, $(111011.011011)_2 = (3B.6C)_{16}$

Tutorial Class-1/Digital Electronics

Do as directed:

* $(100111)_2+(1010)_2$
* (1101)2-(111)2
* $(101)_2*(110)_2$
* $(43.3125)_{10}=(?)_2$
* $(563)_8+(263)_8$
* $(26)_8*(34)_8$
* $(A34F)_{16}+(34BD)_{16}$

1's and 2's Complements

Topics to be covered

*Subtraction using 1's complement

*Subtraction using 2's complement

Subtraction Using 1's Complement

- Find 1's complement of the number to be subtracted .
- Add it to the number from which the subtraction is to be made.
- When there is a carry in the last position, add it to the remaining number.
- If Carry 1 is not obtained in the last position , the result is negative number and it is in its 1's complement form.

Subtraction Using 1's Complement

Subtraction of 10001 from 10011

```
                           1 0 0 1 1
                           0 1 1 1 0 (1's Complement of 10001)
                         -----------
1 Carry                  1 0 0 0 0 1
1 Carry is removed         0 0 0 0 1
1 Carry is Added                   1
                         -----------
                           0 0 0 1 0  <-- Result
```

Subtraction Using 1's Complement

Subtraction of 10011 from 10001

```
              1 0 0 0 1
              0 1 1 0 0 (1's Complement of 10011)
             -----------
No Carry      1 1 1 0 1
```

Result :

-00010 (1's Complement of 11101 with –ve sign)

Subtraction Using 2's Complement

- Find 2's complement of the number to be subtracted .
- Add it to the number from which the subtraction is to be made.
- When there is a carry in the last position, it is ignored.
- If Carry 1 is not obtained in the last position , the result is negative number and it is in its 2's complement form.

Subtraction Using 2's Complement

Subtraction of 00111 from 01110

```
          0  1  1  1  0
          1  1  0  0  1 (2's Complement of 00111)
        ---------------
1 Carry  ~~1~~  0  0  1  1  1 ( Carry Discarded)
```

Result : 0 0 1 1 1

Subtraction Using 2's Complement

Subtraction of 10011 from 10001

```
          1 0 0 0 1
          0 1 1 0 1 (2's Complement of 10011)
          ---------
No Carry  1 1 1 1 0
```

Result :

-00010 (2's Complement of 11110 with –ve sign)

Logic Gates

Topics to be covered

*Introduction

*Types of Logic Gates

*Truth Table

*Logic Symbol

*Logic Expression

Logic Gates

- A logic gate is an electronic circuit which has the ability to make logical decision.
- A logic gate is similar to that of a switching circuit .
- Logical expressions can be described by logical functions using the two binary symbols 1 and 0.

Logic Gates Cont.

- Two different electrical voltage levels such as 5 volts and 0 volt may be used to represent binary 1 and 0.
- Binary logic deals with binary variables and with operations that assume a logical meaning.
- A particular logic operation can be described in an algebraic or tabular form called truth table.

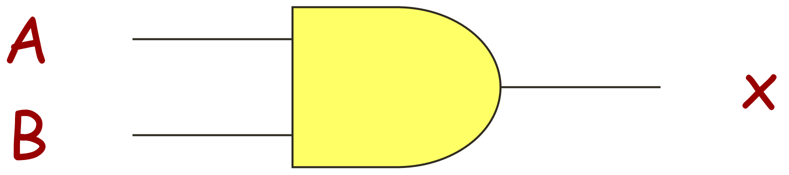
Types of Logic Gates

AND
OR
NOT (Inverter)
NAND (Not AND)
NOR (Not OR)
XOR (Exclusive-OR)
X-NOR (Exclusive-NOR)

AND Logic Gate

Output is 1 if and only if all its inputs are 1

AND Logic Gate



$$\boxed{x = A \cdot B}$$

A, B =Binary Input Variables

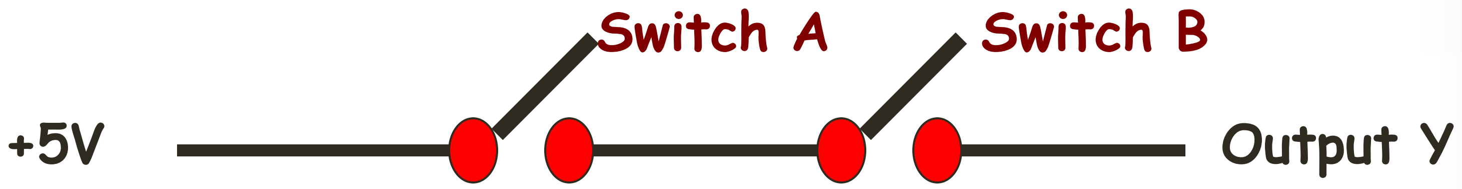
x =Binary Output Variable

Truth Table

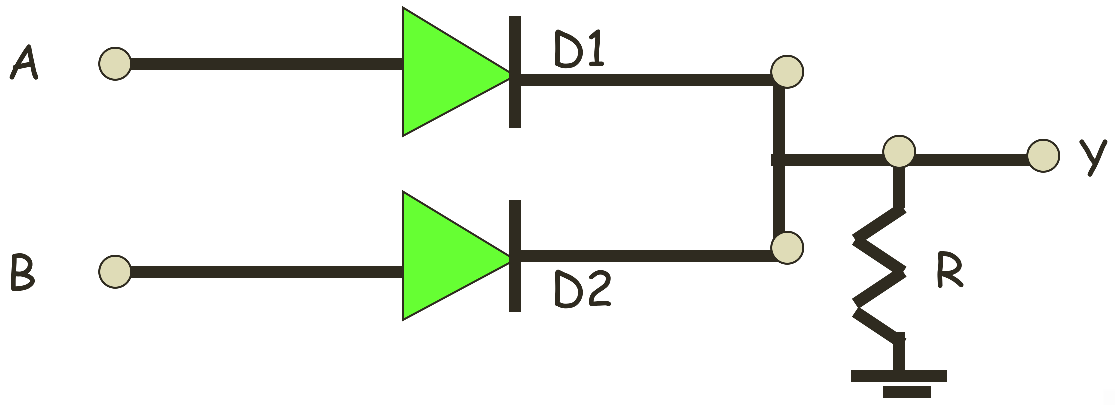
A	B	x
0	0	0
0	1	0
1	0	0
1	1	1

AND Logic Gate

AND Switching Circuit



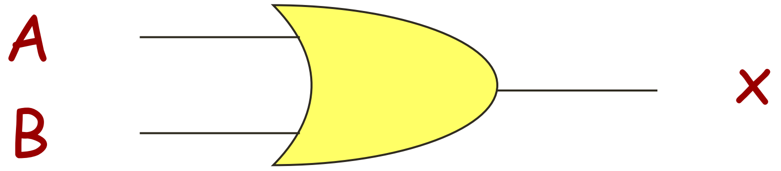
AND Circuit Diagram



OR Logic Gate

Output is 1 if at least one of its inputs is 1.

OR Logic Gate



$$\boxed{x = A + B}$$

Truth Table

A	B	x
0	0	0
0	1	1
1	0	1
1	1	1

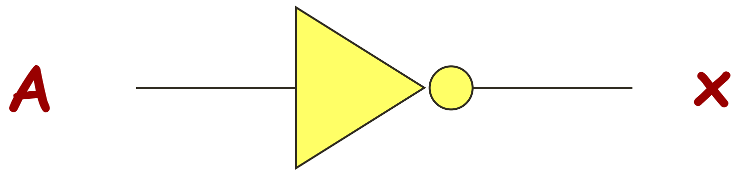
A, B =Binary Input Variables

x =Binary Output Variable

NOT Logic Gate

Output is 1 when its input is 0 and whose output is 0 when its input is 1.

NOT Logic Gate (Inverter) **Truth Table**



$$\boxed{x = \bar{A}}$$

A	x
0	1
1	0

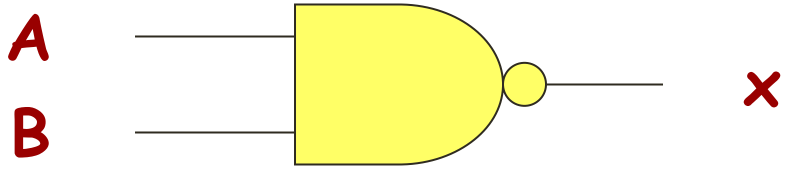
A =Binary Input Variable

x =Binary Output Variable

NAND Logic Gate

Output is 1 if at least one of its input is 0.

NAND Logic Gate



$$\boxed{x = \overline{A . B}}$$

A,B =Binary Input Variables

x =Binary Output Variable

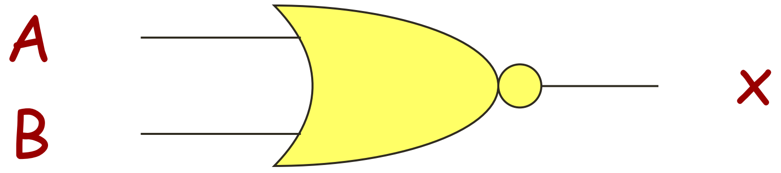
Truth Table

A	B	x
0	0	1
0	1	1
1	0	1
1	1	0

NOR Logic Gate

Output is 1 if all the inputs are 0 and it is 0 if at least one of its inputs is 1

NOR Logic Gate



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

Truth Table

A	B	x
0	0	1
0	1	0
1	0	0
1	1	0

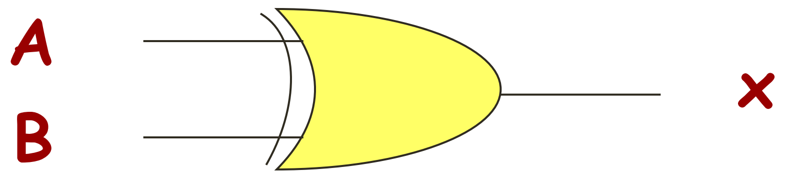
A,B =Binary Input Variables

x =Binary Output Variable

XOR Logic Gate

Output is 1 if any input is 1 but not if all inputs are 1

XOR Logic Gate



$\boxed{x = A \oplus B}$

A,B =Binary Input Variables

x =Binary Output Variable

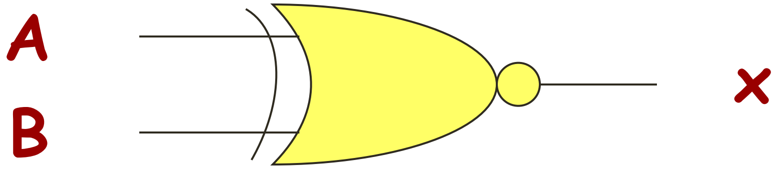
Truth Table

A	B	x
0	0	0
0	1	1
1	0	1
1	1	0

X-NOR Logic Gate

Output is 1 if inputs are the same either 0 or 1.

Exclusive-NOR Logic Gate



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

A,B =Binary Input Variables

x =Binary Output Variable

Truth Table

A	B	x
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Algebra

Topics to be covered

*Introduction

*Laws of Boolean Algebra

*Rules for Boolean Algebra

Boolean Algebra

- provides the operations, laws, rules and theorems for working with the set {0, 1}.

- Logical expression can be expressed symbolically in equation form.

- Equations can be manipulated mathematically.

Boolean Operations

- Sum / Or
  - Denoted by **+**
  - 1+1 = 1, 1+0 = 1, 0+1 = 1, 0+0 = 0
- Product / And
  - Denoted by .
  - 1.1 = 1, 1.0 =0, 0.1 = 1, 0.0 = 0
- Compliment / Negation
  - Denoted by `
  - 1` = 0, 0` = 1

Truth Tables

Sum/Or

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

Product/And

x	y	x . y
0	0	0
0	1	0
1	0	0
1	1	1

Compliment/Negation

x	x`
0	1
1	0

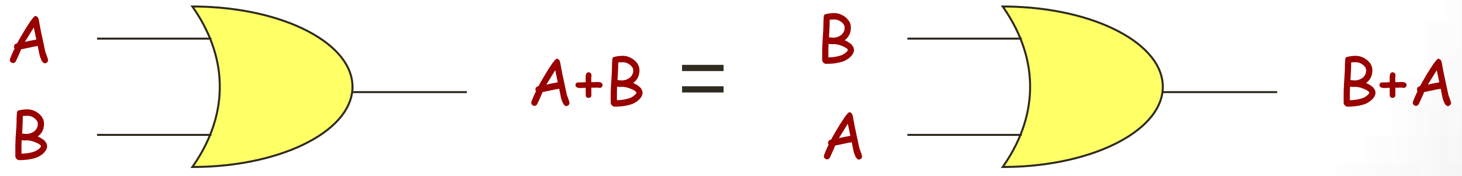
Laws of Boolean Algebra

The Commutative Law of Addition

$A + B = B + A$

Statement:

The order in which the variables are ORed makes no difference.



Laws of Boolean Algebra

Truth Table for Commutative Law of Addition

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

=

A	B	B + A
0	0	0
0	1	1
1	0	1
1	1	1

Truth Tables are identical

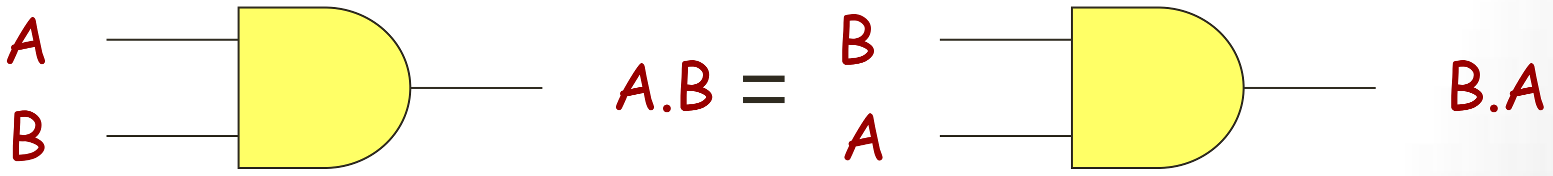
Laws of Boolean Algebra

The Commutative Law of Multiplication

$A \, . \, B = B \, . \, A$

Statement:

The order in which the variables are ANDed makes no difference.



Laws of Boolean Algebra

Truth Table for Commutative Law of Multiplication

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

=

A	B	B . A
0	0	0
0	1	0
1	0	0
1	1	1

Truth Tables are identical

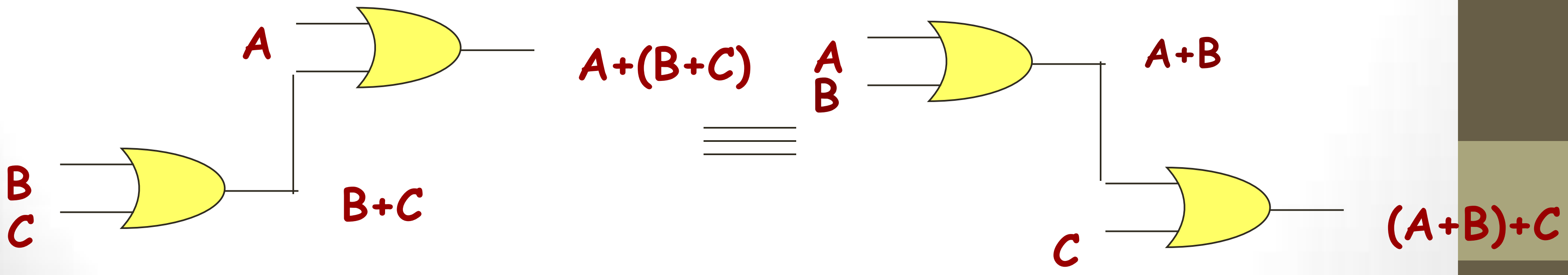
Laws of Boolean Algebra

The Associative Law of Addition

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

Statement:

The ORing of several variables, result is the same irrespective of the way the variable are grouped.



Laws of Boolean Algebra

Truth Table for Associative Law of Addition

A	B	C	B+C	A+(B+C)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	1
1	0	0	0	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	A+B	(A+B)+C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Truth Tables are identical

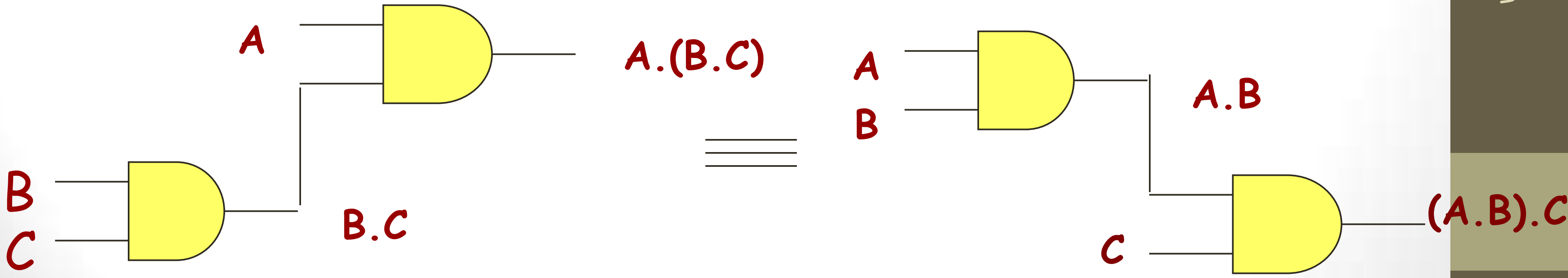
Laws of Boolean Algebra

The Associative Law of Multiplication

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

Statement:

The ANDing of several variables, result is the same irrespective of the way the variable are grouped.



Laws of Boolean Algebra

Truth Table for Associative Law of Multiplication

A	B	C	B.C	A(B.C)
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

=

A	B	C	A.B	(A.B).C
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

Truth Tables are identical

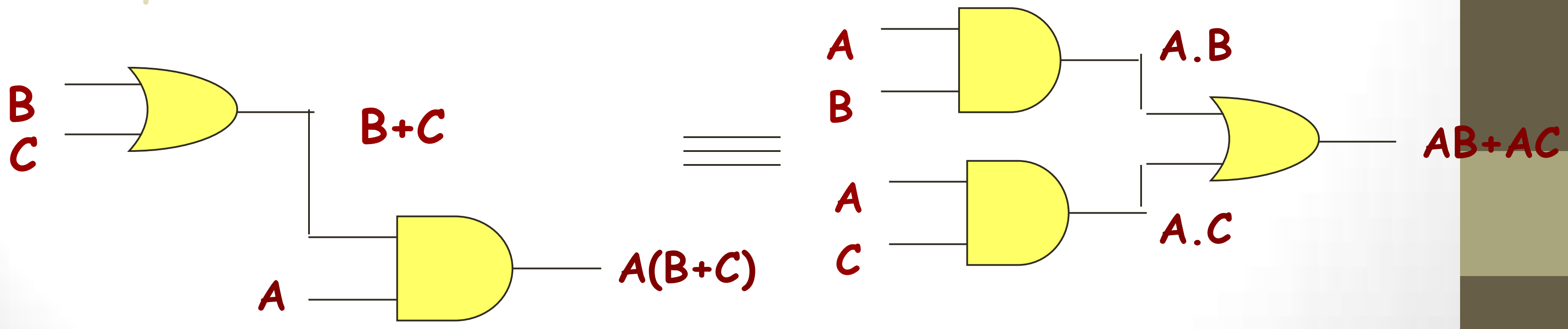
Laws of Boolean Algebra

The Distributive Law

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

Statement:

ORing several variables and ANDing the result with a single variable is equivalent to ANDing the single variable with each of the several variables and ORing the products.



Laws of Boolean Algebra

Truth Table for Distributive Law

A	B	C	B+C	A(B+C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

A	B	C	A.B	A.C	A.B+A.C
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Truth Tables are identical

Rules for Boolean Algebra

Useful in manipulating and simplifying Boolean Expression.

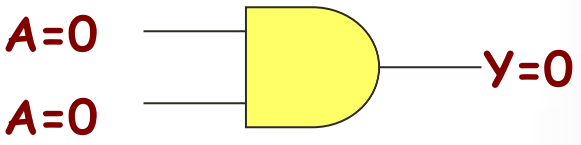
Rules for Boolean Algebra

The Idempotent Property

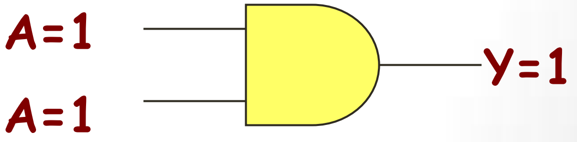
1. $A \cdot A = A$

Rule 1: If a variable is ANDed with itself, the result is equal to the variable.

If $A=0$, then $A.A = 0.0 = 0 = A$



If $A=1$, then $A.A = 1.1 = 1 = A$



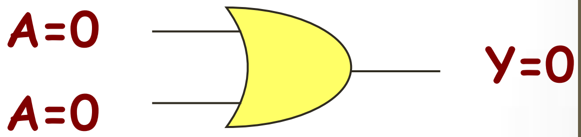
Rules for Boolean Algebra

The Idempotent Property

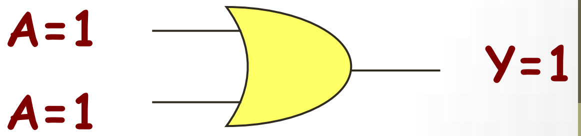
2. $A + A = A$

Rule 2: If a variable is ORed with itself, the result is equal to the variable.

If $A=0$, then $A+A = 0 + 0 = 0 = A$



If $A=1$, then $A+A = 1 + 1 = 1 = A$



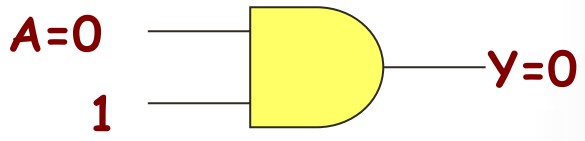
Rules for Boolean Algebra

The Identity Property

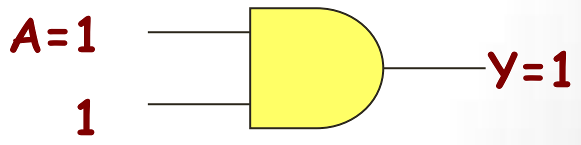
3. $A \cdot 1 = A$

Rule 3: If a variable is ANDed with 1, the result is equal to the variable.

If $A=0$, then $A \cdot 1 = 0 \cdot 1 = 0 = A$



If $A=1$, then $A \cdot 1 = 1 \cdot 1 = 1 = A$



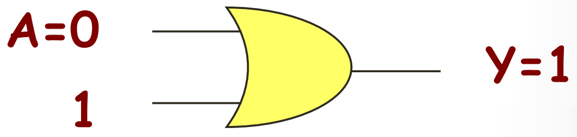
Rules for Boolean Algebra

The Identity Property

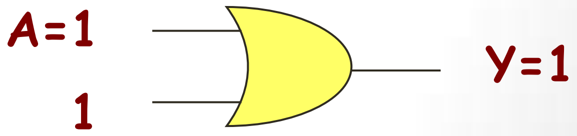
4. $A + 1 = 1$

Rule 4: If a variable is ORed with 1, the result is equal to 1.

If $A=0$, then $A + 1 = 0 + 1 = 1$



If $A=1$, then $A + 1 = 1 + 1 = 1$



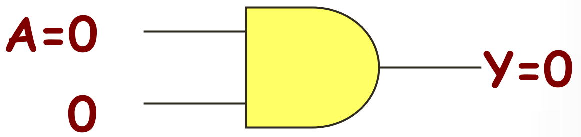
Rules for Boolean Algebra

The Null Property

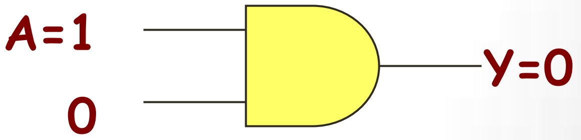
5. $A \cdot 0 = 0$

Rule 5: If a variable is ANDed with 0, the result is equal to 0.

If $A=0$, then $A \cdot 0 = 0 \cdot 0 = 0$



If $A=1$, then $A \cdot 0 = 1 \cdot 0 = 0$



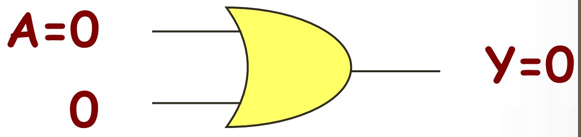
Rules for Boolean Algebra

The Null Property

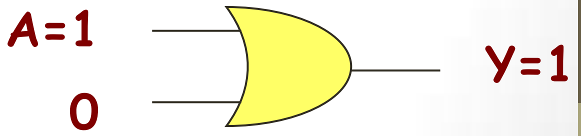
6. $A + 0 = A$

Rule 6: If a variable is ORed with 0, the result is equal to the variable.

If $A=0$, then $A + 0 = 0 + 0 = 0 = A$



If $A=1$, then $A + 0 = 1 + 0 = 1 = A$



Rules for Boolean Algebra

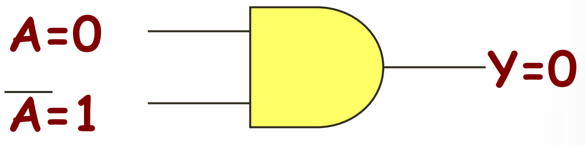
The Negation Property

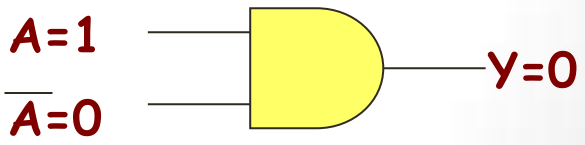
7. $A \cdot \overline{A} = 0$

Rule 7: If a variable is ANDed with its complement, the result is equal to 0 .

If $A=0$, then $\overline{A} = 1$ and $A.\overline{A} = 0.1 = 0$

If $A=1$, then $\overline{A} = 0$ and $A.\overline{A} = 1.0 = 0$





Rules for Boolean Algebra

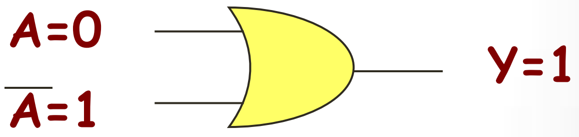
The Negation Property

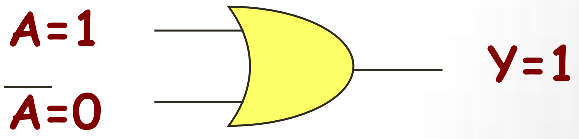
8. $A + \overline{A} = 1$

Rule 8: If a variable is ORed with its complement, the result is equal to 1

If $A=0$, then $\overline{A} = 1$ and $A+\overline{A} = 0+1 = 1$

If $A=1$, then $\overline{A} = 0$ and $A+\overline{A} = 1+0 = 0$





Rules for Boolean Algebra

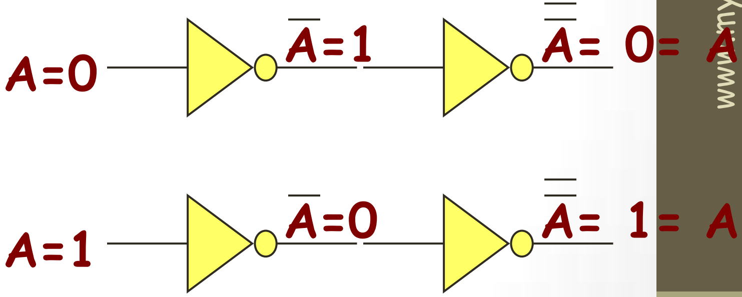
The Double Negation Property

9. $\overline{\overline{A}} = A$

Rule 9: If a variable is complemented twice, the result is the variable itself .

If $A=0$, then $\overline{A} = 1$ and $\overline{\overline{A}} = 0 = A$

If $A=1$, then $\overline{A} = 0$ and $\overline{\overline{A}} = 1 = A$



$A=0$
$\overline{A}=1$
$\overline{\overline{A}}= 0= A$
$A=1$
$\overline{A}=0$
$\overline{\overline{A}}= 1= A$

Rules for Boolean Algebra

The Absorption Property

10. $A + AB = A$

L.H.S.

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

$= A.1$ (From identity property $(A+1)=1$)

$= A$

R.H.S.

Rules for Boolean Algebra

The Absorption Property

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

L.H.S.

$A + \overline{A}B = A + AB + \overline{A}B$ (From Rule 10)

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

$= A + 1.B$ (From Rule 8)

$= A + B$ (From Rule 3)

R.H.S.

Rules for Boolean Algebra

The Absorption Property

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

L.H.S.

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

$= A + A.B$ (From Rule 1)

$= A$ (From Rule 10)

R.H.S.

DeMorgan's Theorems

Topics to be Covered

*Introduction

*Theorem - 1

* Theorem - 2

DeMorgan's Theorems

DeMorgan's theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted.

DeMorgan's Theorems

Theorem – 1

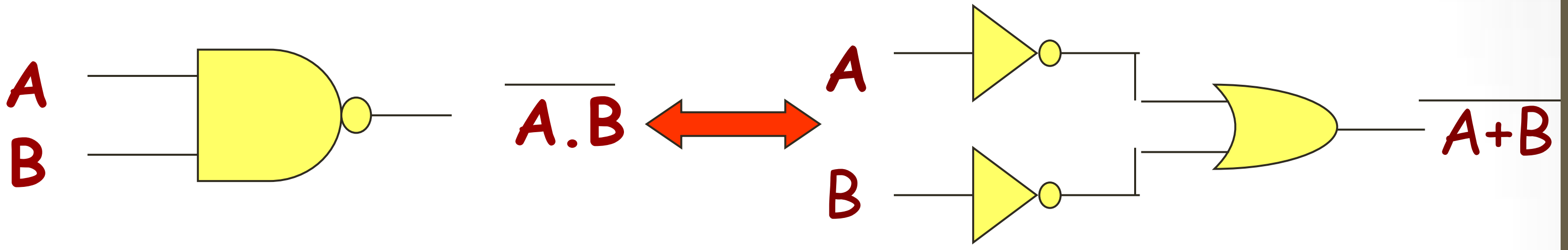
$$\overline{A.B} = \overline{A} + \overline{B}$$

Statement :
The Complement of two or more variables ANDed is the same as OR of the Complement of each individual variable.

OR

NAND gate performs the same operation as an OR gate whose inputs are inverted.

DeMorgan's Theorems



A	B	A.B	$\overline{A.B}$
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

DeMorgan's Theorems

Theorem – 2

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

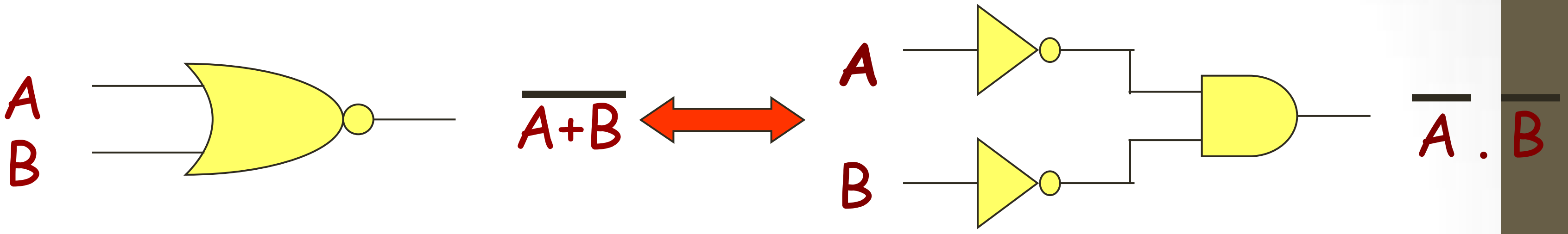
Statement :

The Complement of two or more variables ORed is the same as AND of the Complement of each individual variable.

OR

NOR gate performs the same operation as an AND gate whose inputs are inverted.

DeMorgan's Theorems

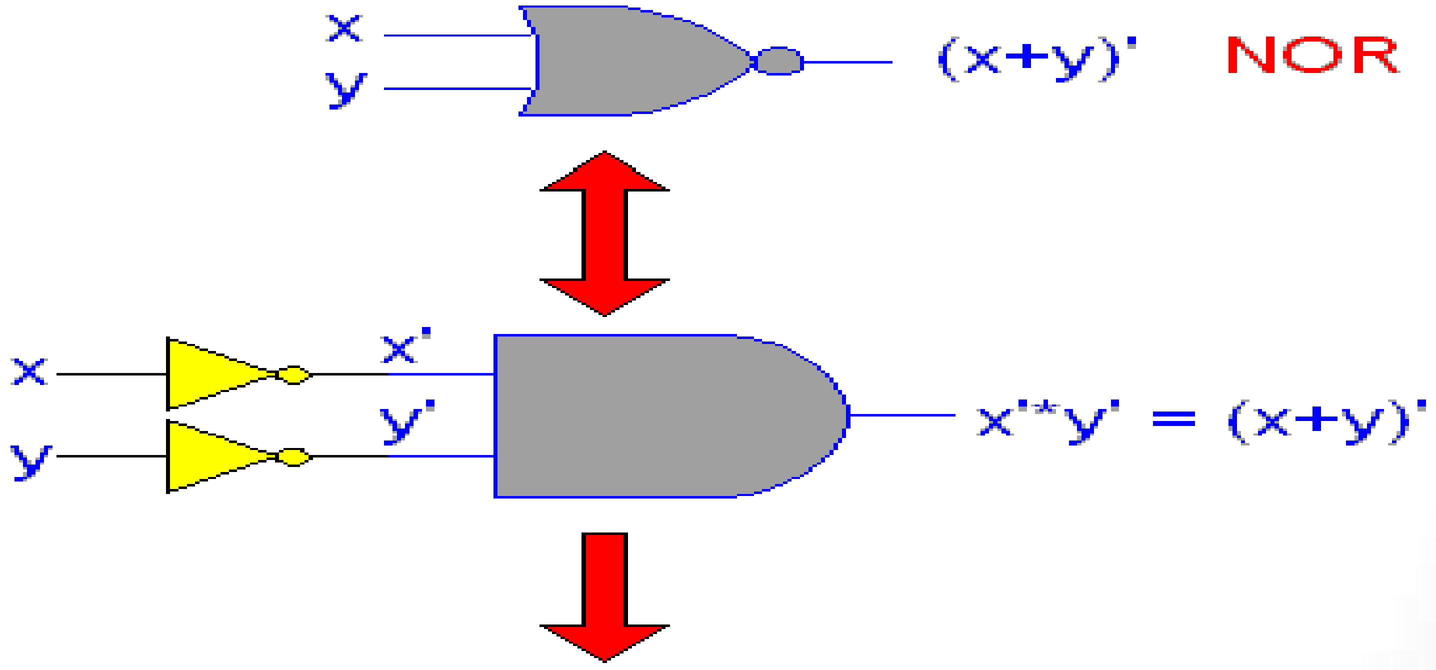


A	B	A+B	$\overline{A+B}$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

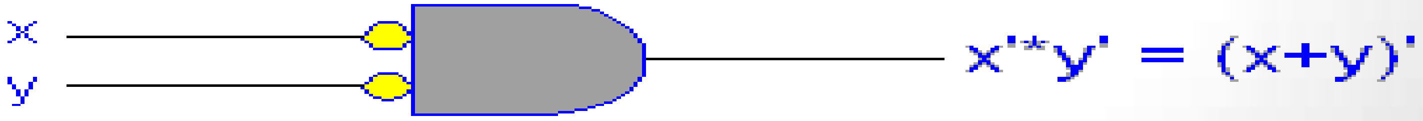
A	B	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

Implication of DeMorgan's Theorems

For (1): (x+y)' = x' * y'

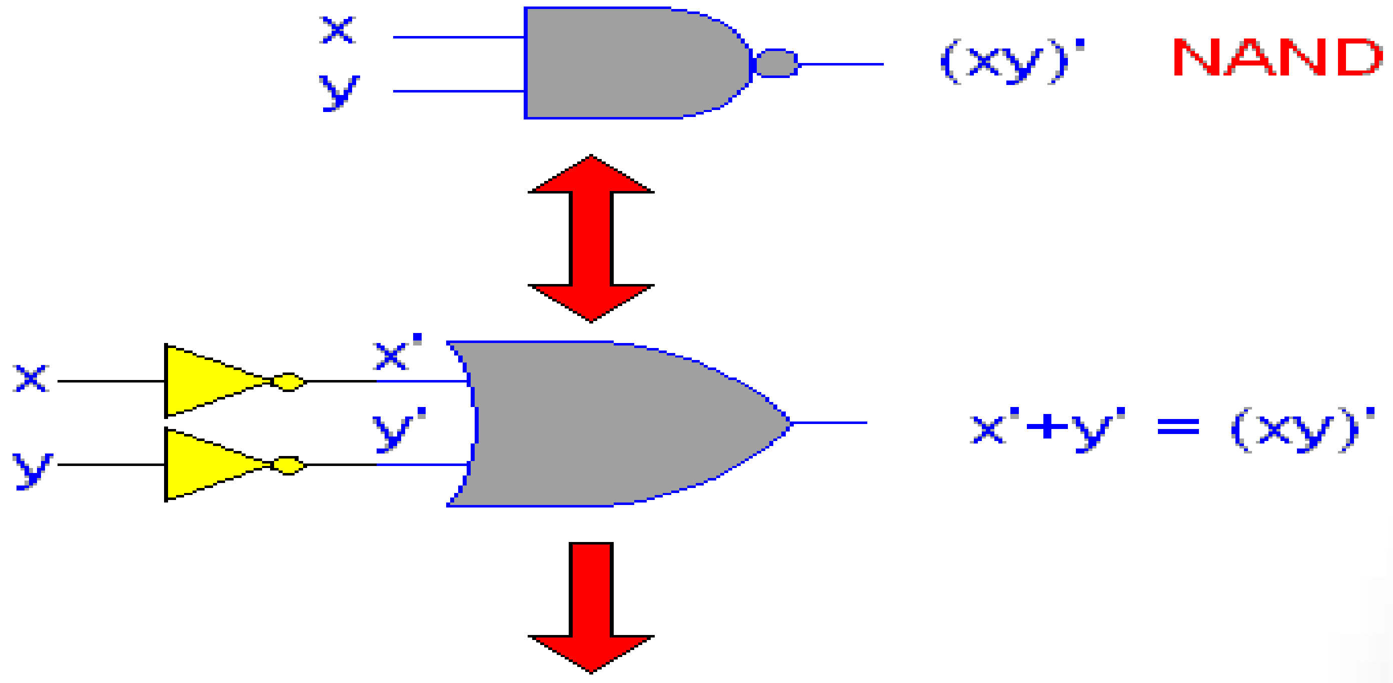


Alternative symbol for the NOR function

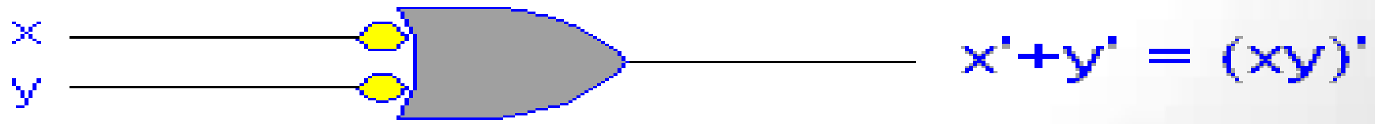


Implication of DeMorgan's Theorems

For (2): (x*y)' = x' + y'



Alternative symbol for the NAND function



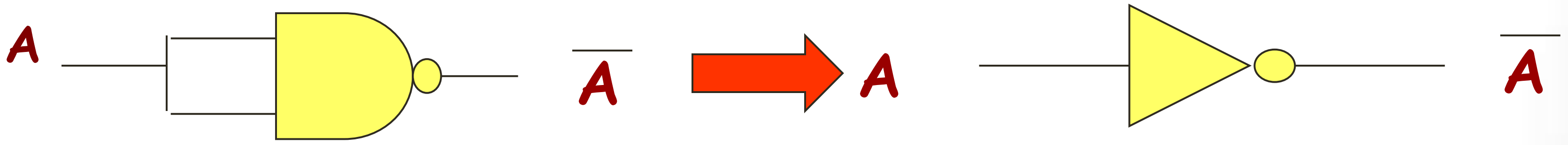
Universality of NAND & NOR Gates

It is possible to implement any logic expression using only NAND gates or NOR Gates and no other type of gate. This is because NAND and NOR gates, in the proper combination, can be used to perform each of the Boolean operations OR, AND, and INVERT.

Universality of NAND Gate

Not gate from NAND Gate

Inputs of two NAND gates are connected together , acts as a NOT gate.



As $\overline{A.A} = \overline{A} + \overline{A} = \overline{A}$

(From DeMorgan's theorem)

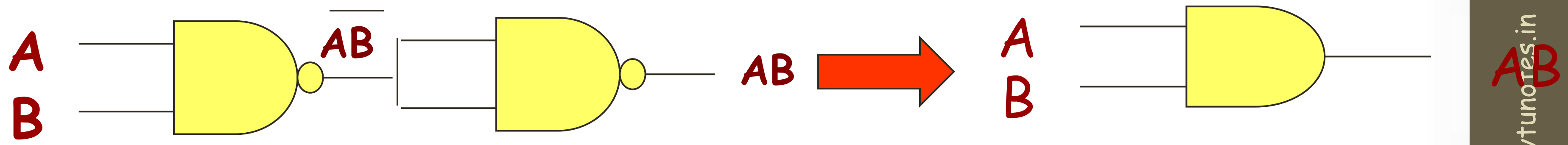
Universality of NAND Gate

AND gate from NAND Gate

- Two NAND gates requires.
- Output of one NAND gate is given to another NAND gate.
- Second NAND gate acting as an inverter, resulting the AND gate.

Universality of NAND Gate

AND gate from NAND Gate



As A NAND B = $\overline{AB}$ and $\overline{\overline{AB}}$ = AB (From $\overline{\overline{A}}$ = A)

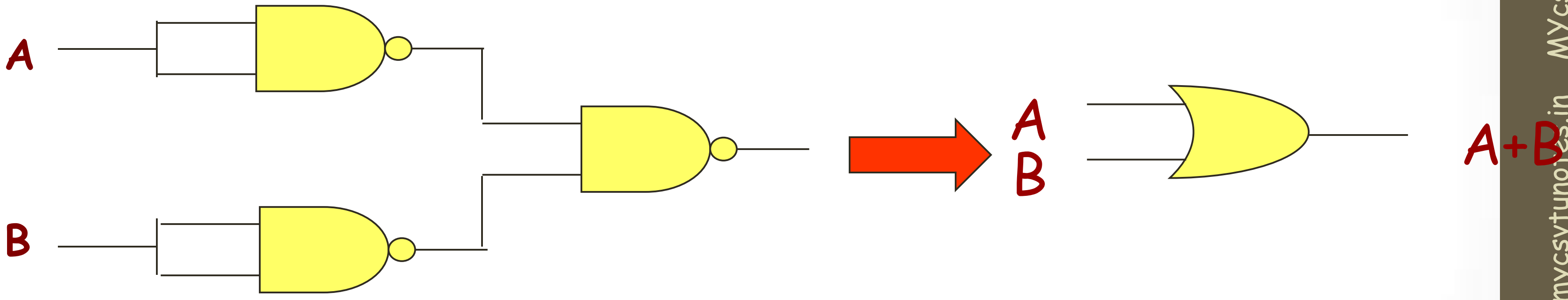
Universality of NAND Gate

OR gate from NAND Gate

- Three NAND gates requires.
- First two NAND gates operated as NOT gates by connecting together the inputs of each NAND gate.
- Outputs of these NOT gates are given to the third NAND gate.

Universality of NAND Gate

OR gate from NAND Gate



As $\overline{A.A} = \overline{A}$ and $\overline{B.B} = \overline{B}$ (From $\overline{AA}$=A)

$\overline{\overline{A}}\,.\overline{\overline{B}} = \overline{\overline{A}} + \overline{\overline{B}}$ (From DeMorgan's Theorem)

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

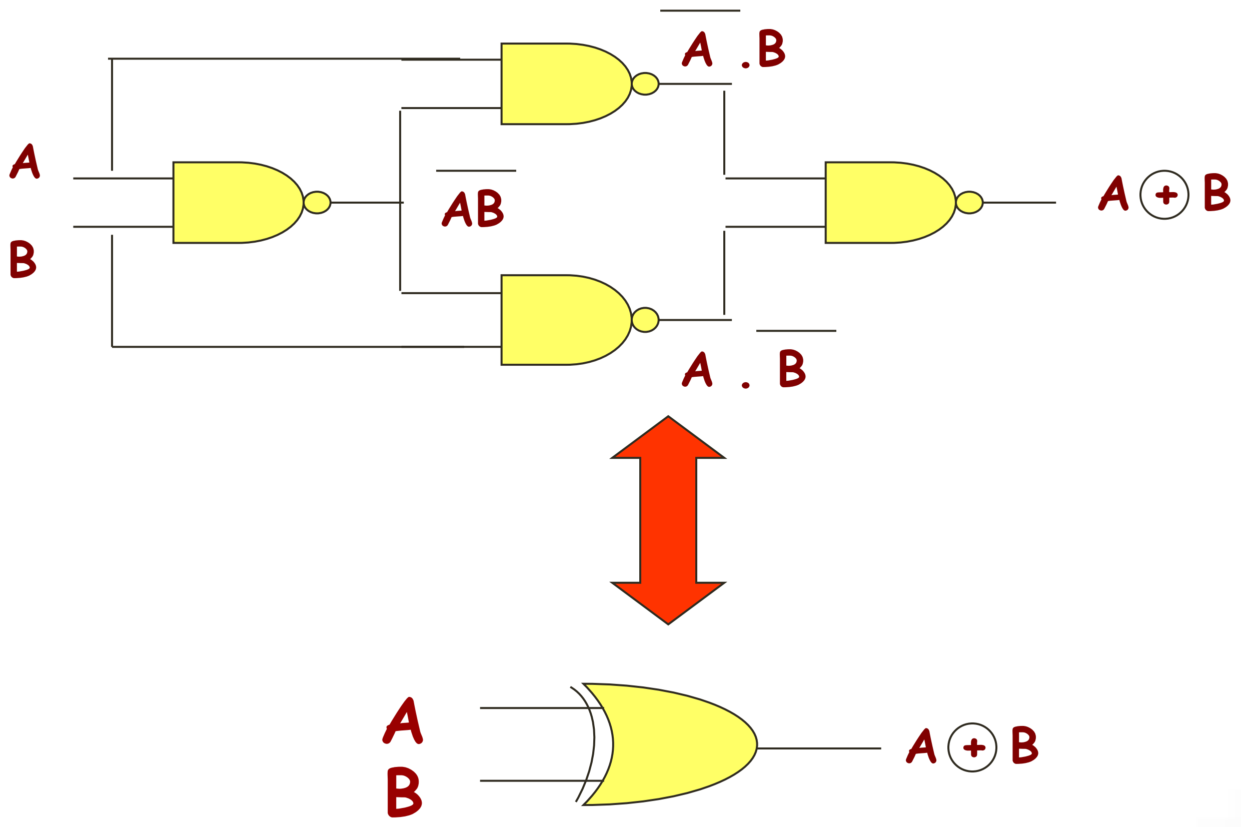
Universality of NAND Gate

X-OR gate from NAND Gate

- Four NAND gates requires.
- First NAND gate gives output $\overline{AB}$.
- Outputs of Second NAND gate is $\overline{A} + B$.
- Output of Third NAND gate is $A + \overline{B}$.
- Output of last NAND gate gives $A\overline{B} + \overline{A}B$.

Universality of NAND Gate

X-OR gate from NAND Gate



Universality of NAND Gate

X-OR gate from NAND Gate

As $\overline{A.\overline{AB}} = \overline{A} + AB = \overline{A} + B$ (From DeMorgan's Theorem)

&

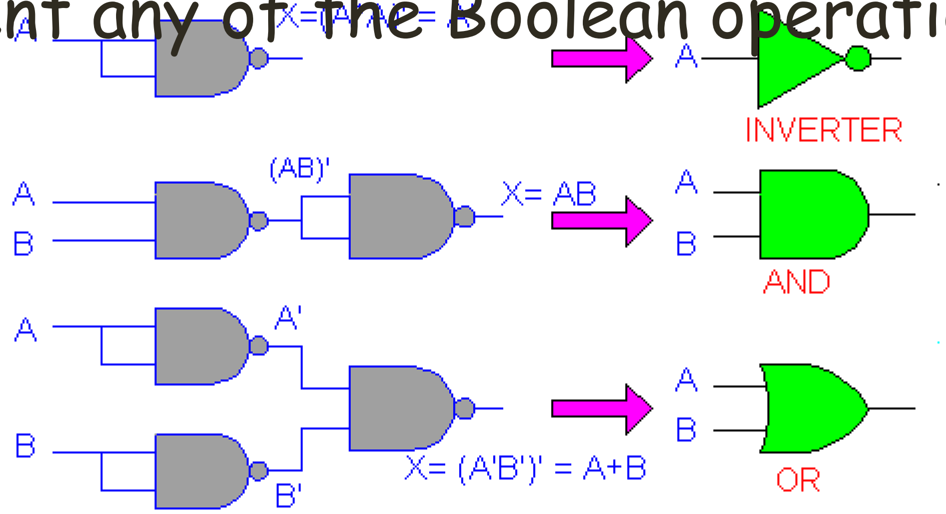
$\overline{B.\overline{AB}} = \overline{B} + AB = \overline{B} + A$ (From DeMorgan's Theorem)

Output $[(\overline{A} + B).(\overline{B} + A)] = \overline{A}B + B\overline{A}$

$= A \oplus B$

Universality of NAND Gate

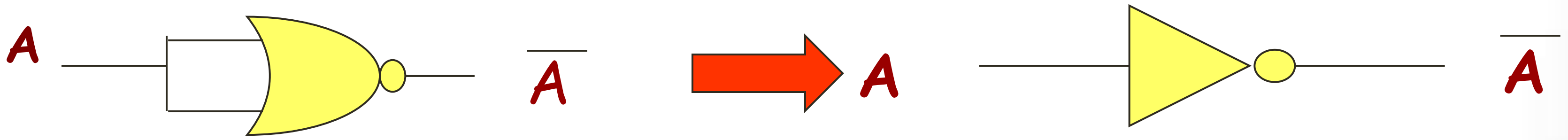
It can be shown that NAND gates can be arranged to implement any of the Boolean operations.



Universality of NOR Gate

Not gate from NOR Gate

Inputs of two NOR gates are connected together , acts as a NOT gate.



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

(From DeMorgan's theorem)

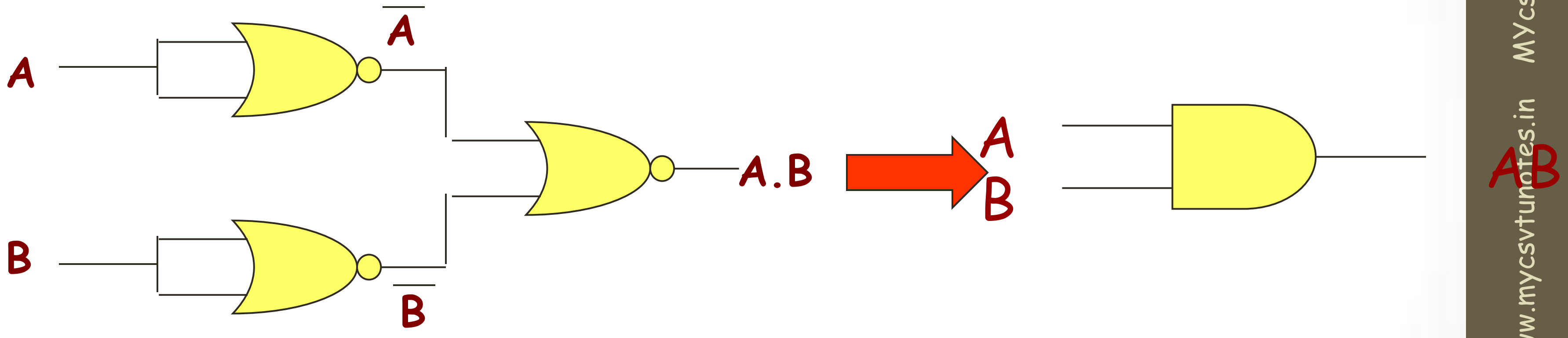
Universality of NOR Gate

AND gate from NOR Gate

- Three NOR gates requires.
- Two NOR gates acting as inverter.
- Third NOR gate resulting the AND gate.

Universality of NOR Gate

AND gate from NOR Gate



As $\overline{A+A} = \overline{A}$ and $\overline{B+B} = \overline{B}$ (From $\overline{A+A} = \overline{A}$)

$\overline{\overline{A}+\overline{B}} = \overline{\overline{A}} \,.\, \overline{\overline{B}}$ (From DeMorgan's Theorem)

$= A \,.\, B$ (From $\overline{\overline{A}} = A$)

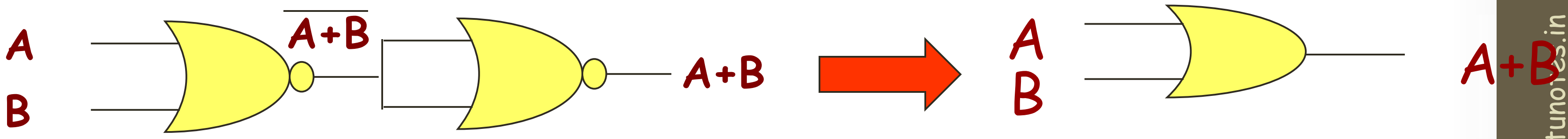
Universality of NOR Gate

OR gate from NOR Gate

- Two NOR gates requires.
- Output of one NOR gate is given to another NOR gate.
- Second NOR gate acting as an inverter, resulting the OR gate.

Universality of NOR Gate

OR gate from NOR Gate



As A NOR B = $\overline{A + B}$ and $\overline{\overline{A+ B}}$ =A + B

(From $\overline{\overline{A}}$ = A)

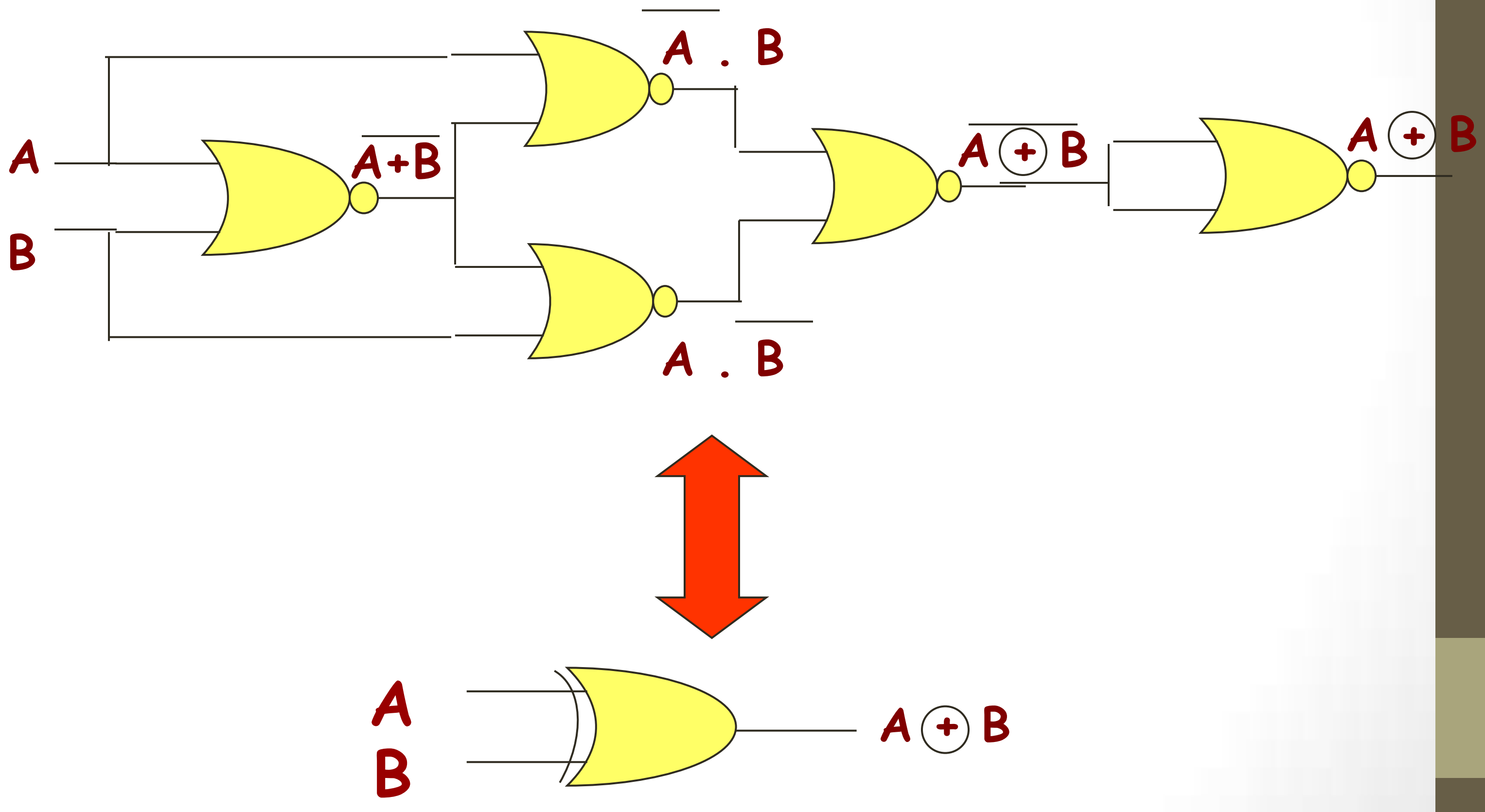
Universality of NOR Gate

X-OR gate from NOR Gate

- Five NOR gates requires.
- First NAND gate gives output $\overline{A+B}$.
- Outputs of Second NAND gate is $\overline{A}.\ B$.
- Output of Third NAND gate is $A\ .\ \overline{B}$.
- Output of last NAND gate gives $\overline{A}B + A\overline{B}$.

Universality of NOR Gate

X-OR gate from NOR Gate



Universality of NOR Gate

X-OR gate from NOR Gate

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

(From DeMorgan's Theorem)

&

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

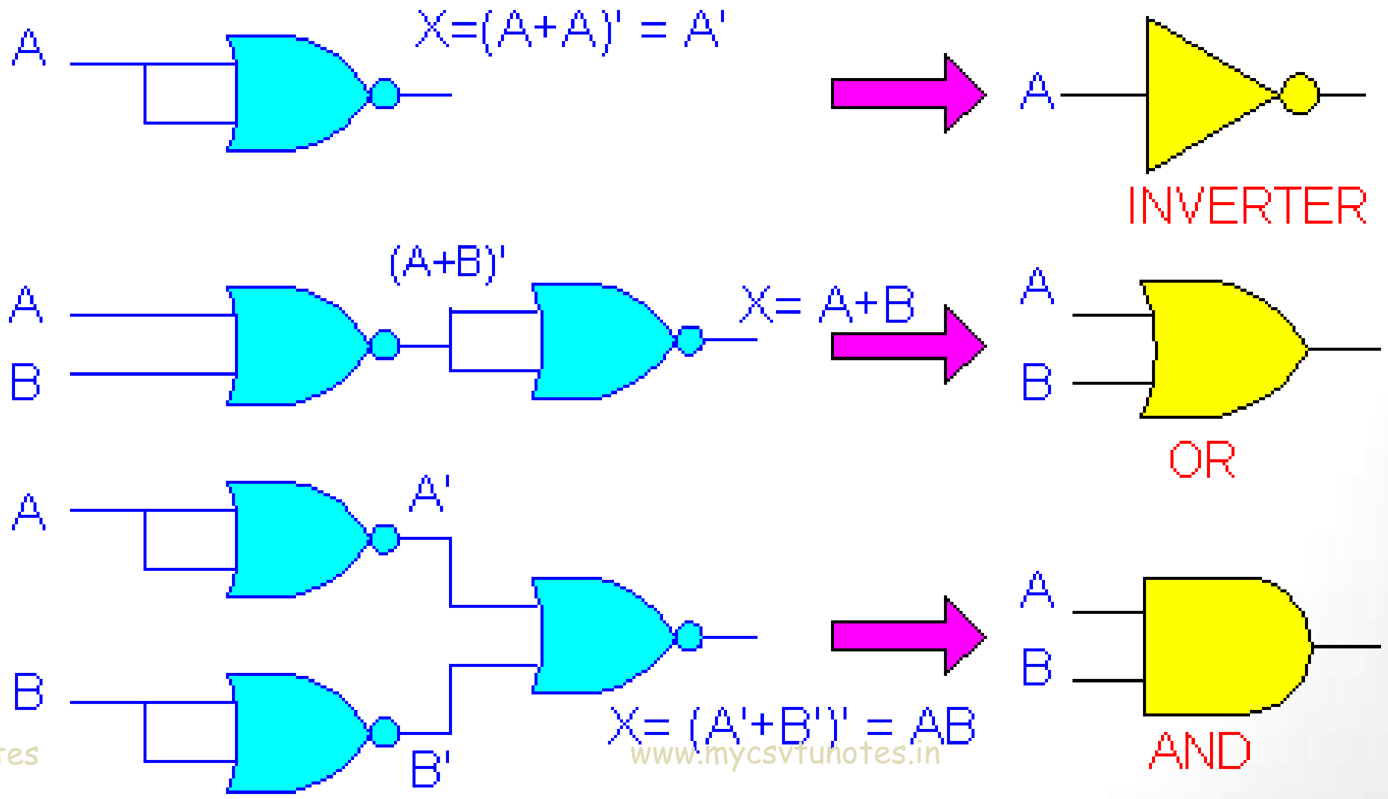
(From DeMorgan's Theorem)

Output $\overline{[(\overline{A}.B)+(\overline{B}.A)]} = \overline{A \oplus B}$

$= A \oplus B$

Universality of NOR Gate

In a similar manner, it can be shown that NOR gates can be arranged to implement any of the Boolean operations.



Boolean Theorems

Prove That:

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

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

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

(4) $AB + \overline{A}C + BC = AB + \overline{A}C$

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

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

(7) $(A+B)(\overline{A}+C) = AC + \overline{A}\,B$

(8) $A.\overline{B} + \overline{A}\,.\,B = (A+B).(\overline{A} + \overline{B})$

(9) $A.B + \overline{A}\,.\,\overline{B} = (A + \overline{B})\,(A + \overline{B})$

Boolean Theorems

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

Boolean Theorems

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

Boolean Theorems

Boolean Theorems

Boolean Theorems

Boolean Theorems