Basic and Derived Logic Gates

Introduction

- **Digital (logic) circuits operate in the binary mode where each input &output voltage is either a 0 or 1; the 0 and 1 designations represent predefined voltage ranges.**
- **This characteristic of logic circuits allows us to use *Boolean algebra* as a tool for the analysis and design of digital systems.**
- **In this chapter we will study *logic gates*, which are the most fundamental logic circuits, and we will see how their operation can be described using Boolean algebra.**
- **The interconnection of gates to perform a variety of logical operations is called logic design.**
- **We will also see how logic gates can be combined to produce logic circuits, and how these circuits can be described and analyzed using Boolean algebra.**

Basic Logic Gates

Boolean algebra

Boolean Constants and Variables

- Boolean algebra differs in a major way from ordinary algebra in that Boolean constants and variables are allowed to have only two possible values, *0 or 1*.
- Boolean Variable is a quantity that may, at different times, be equal to either 0 or 1 .
- The Boolean Variables are often used to represent the voltage level represent on a wire or input/output terminal of a circuit.
- For example: in a certain digital system the Boolean Value of 0 might be assigned to any voltage in the range from 0 to 0.8V while the Boolean value of 1 might be assigned to any voltage in the range of 2 to 5V.
- Voltage b/w 0.8 and 2V are undefined (neither 0 nor 1) and under normal circumstances should not occur.

- **Thus, Boolean 0 and 1 do not represent actual numbers but instead represents the state of voltage variable, or what is called its *logic level*.**
- **A voltage in a digital circuit is said to be at logic level 0 or the logic level 1, depending on its actual numerical value.**
- **In table below, some of the more common terms in the digital logic field are shown.**

LOGIC 0	LOGIC 1
False	True
Off	On
Low	High
No	Yes
Open switch	Closed switch

- **In all our works to follow we shall use *letter symbols* to represent *logic variables*.**
- **Because only two values are possible, Boolean algebra is relatively easy to work with as compared with ordinary algebra.**
- **In Boolean algebra there are no *decimals*, *fractions*, *negative numbers*, *square root*, *logarithms*, *imaginary number*, and so on.**
- **In Boolean algebra there are only three basic operation; *OR*, *AND* and *NOT*.**
- **These basic operations are called *logic operations***
- **Digital circuits called logic gates can be constructed from *diode, transistors*, and *resistors* connected in such a way that the circuit output is the result of a basic logic operation (OR, AND, NOT) performed on the inputs.**

Truth Tables

- Many logic circuits have more than one input and only one output.
- A truth table shows how the logic circuit's output responds to the various combinations of logic levels at the inputs.
- The format for two-, three-, and four-input truth tables is shown in Table below
- The number of input combinations will equal 2^N for an N-input truth table.

Inputs		Output
A	B	Y
0	0	?
0	1	?
1	0	?
1	1	?

Truth table for two-input

Inputs			Output
A	B	C	Y
0	0	0	?
0	0	1	?
0	1	0	?
0	1	1	?
1	0	0	?
1	0	1	?
1	1	0	?
1	1	1	?

Truth table for three-input

Inputs				Output
A	B	C	D	Y
0	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	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	?

Truth table for four-input

OR OPERATION

- Let A and B represents two independent logic variables.
- When A and B are combined using the OR operation the result Y can be expressed as

$$Y = A+B$$

- In this expression the + sign does not stand for ordinary addition; it stands for the OR operation, whose rules are given in the truth table shown below

A	B		Y= A+B
0	0		0
0	1		1
1	0		1
1	1		1

- **It should be apparent from the truth table that *except* for the case where A=B=1, the OR operation is the same as ordinary operation.**
- **However, for A=B=1 the OR sum is 1 (not 2 as in ordinary addition).**
- **This is easy to remember if we recall that only 0 and 1 are possible values in Boolean algebra, so that the largest value we can get is 1.**
- **This same result is true if we have Y= A+B+C, for the case where A=B=C=1 that is**

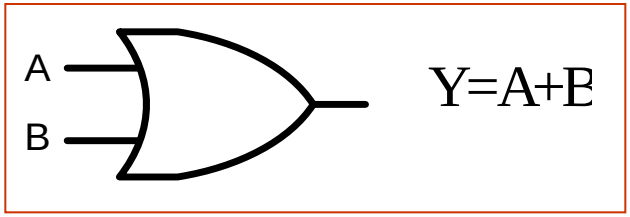
$$Y = 1+1+1 = 1$$

- **We can therefore say that the OR operation result will be 1 if any one or more variables is a 1**
- **The expression Y= A+B is read as**

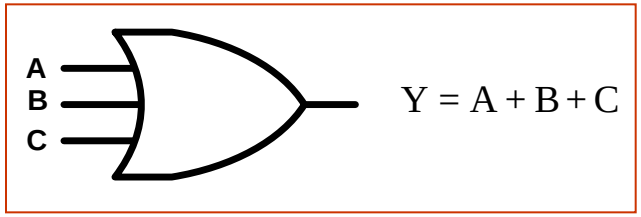
''Y equals A OR B''

OR Gate

- A gate is logic circuit with one output and one or more inputs; an output signal occurs **only for certain combinations of input signals.**
- In digital circuitry an OR gate is a circuit that has two or more inputs and whose output is equal to the OR sum of the inputs.
- This is the logic symbol for a two-input OR gate
- The OR gate operates in such way that its output is HIGH (Logic 1) if either input A or B or both are at a logic 1 level.
- The OR gate output will be Low (Logic 0) only if all its inputs are at logic 0.This same idea can be extended to more than two inputs.
- Figure below shows three inputs OR gate and its truth table.



Logic symbol of
Two-input OR gate



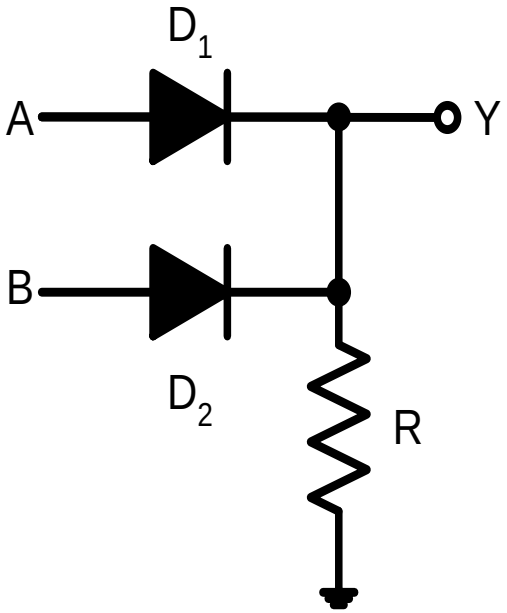
Logic symbol of
three-input OR gate

A	B	C	Y= A+B+C
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

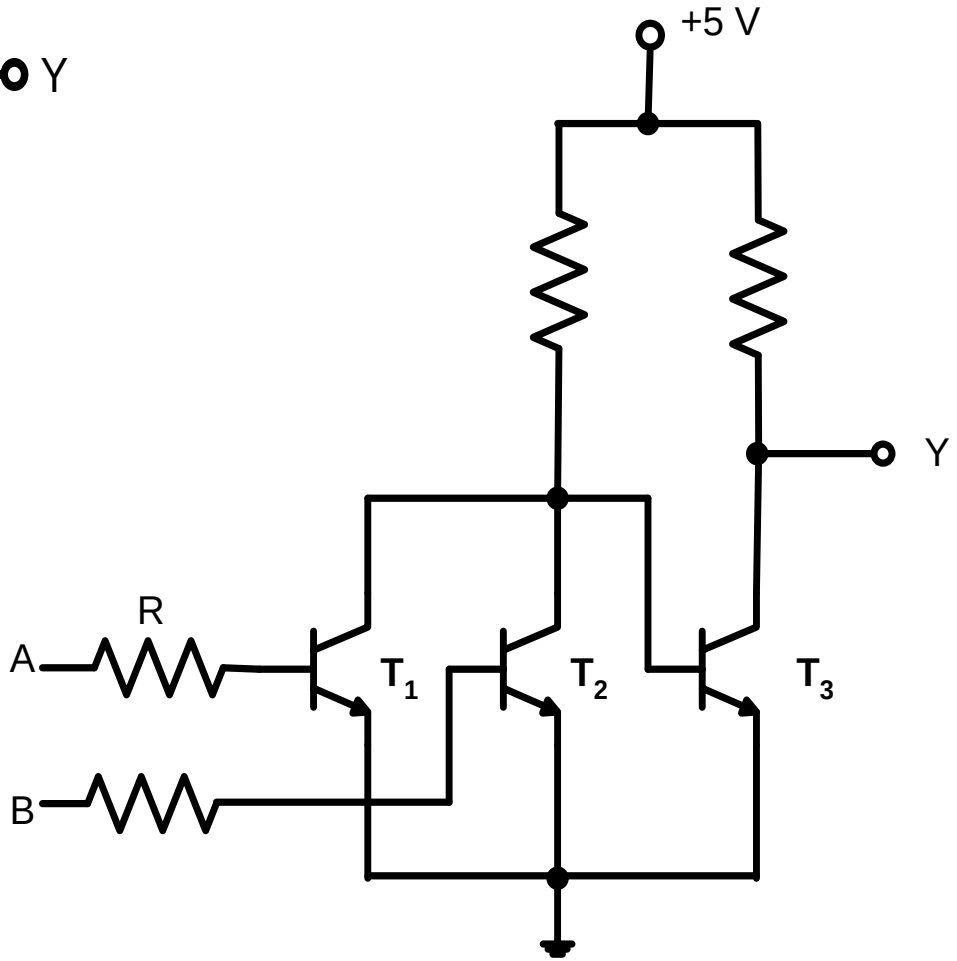
Summary of the OR Operation

- The OR operation produces a result of 1 when any of the input variables is 1
- The OR operation produces a result of 0 only when all the input variables are 0.
- With the OR operation, 1+1 =1, 1+1+1 = 1, and so on.

Discrete **OR** Gate



Two-input
diode OR Gate

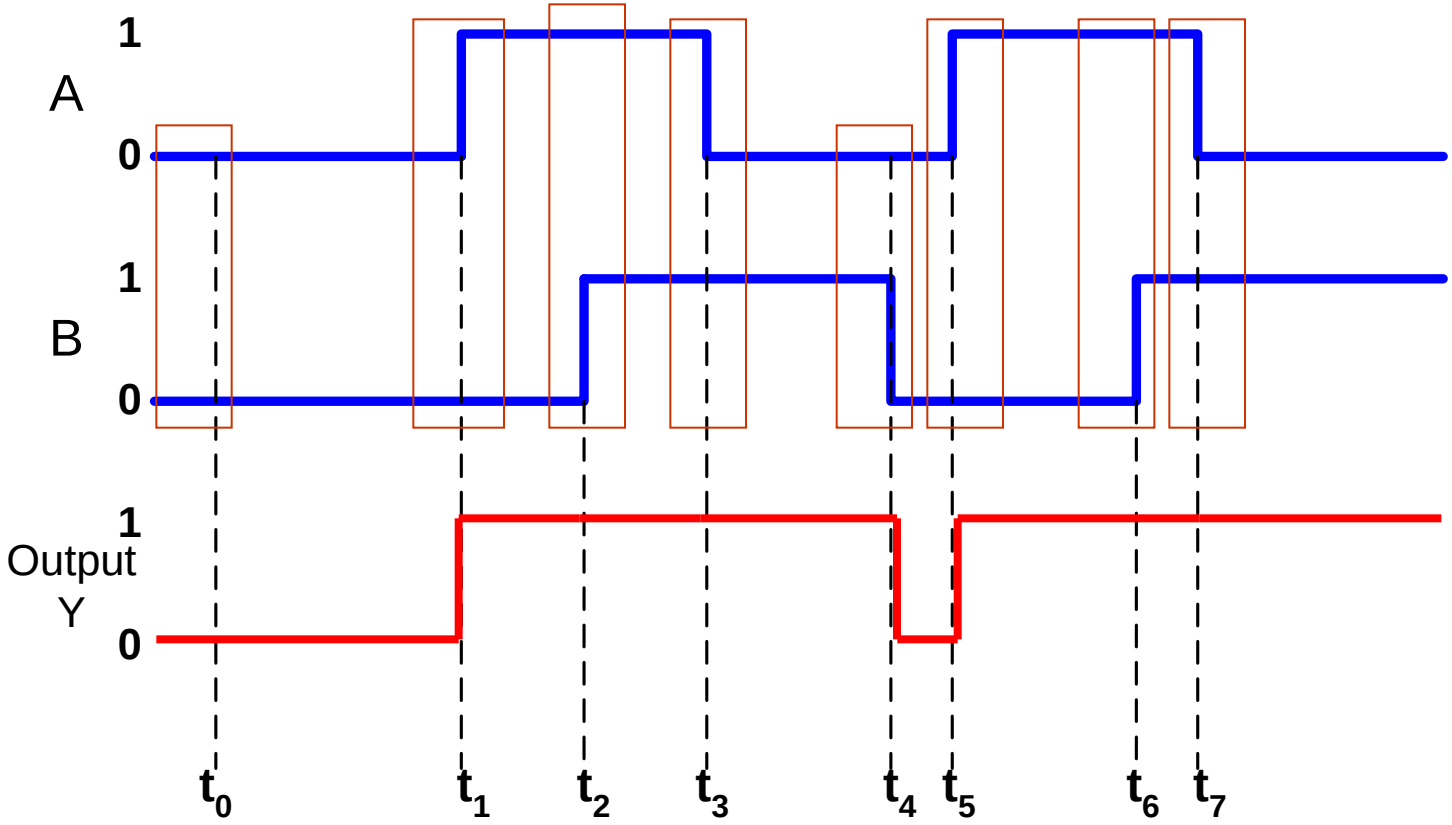


Two-input
Transistor OR gate

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

Truth Table

Example: Determine the OR gate output in Figure below. The OR gate inputs A and B are vary-ing according to the timing diagrams shown. For example, A starts out LOW at time to, goes HIGH at t1, back LOW at t3, and so on.



AND OPERATION

- If two logic variables A and B are combined using the AND operation, the result, Y, can be expressed as

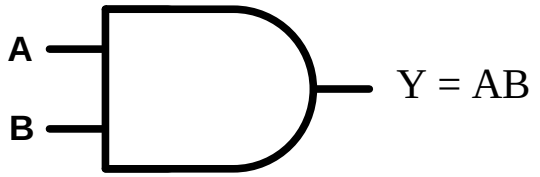
$$Y = A \cdot B$$

- In this expression the · sign stands for the Boolean AND operation whose rules are given in the truth table shown in figure below.
- It should be apparent from the table that the ***AND operation*** *is exactly the same as ordinary multiplication.*
- Whenever A or B is 0, their product is zero, when both A and B are 1, their product is 1. We can therefore say that in the AND operation the result will be 1 *only if* all the inputs are 1: for all other cases the result is 0.
- The expression Y= A.B is read ‘’*Y equals A AND B*’’
- The multiplication sign is generally omitted as in ordinary algebra, so that the expression becomes Y=AB

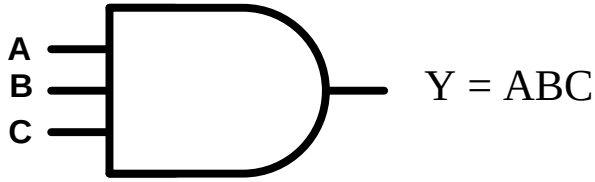
A	B	Y= A.B
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate

- The logic symbol for a two- input AND gate is shown in fig. Below
- The AND gate output is equal to the AND product of the logic inputs i.e. Y= AB.
- In other words, the AND gate is a circuit that operates in such a way that its output is HIGH only when all its inputs are HIGH.
- For all other cases the AND gate output is LOW.
- This same operation is Characteristic of AND gates with more than two inputs.
- A three-input AND gate and its accompanying truth table are shown in fig. Below.



Logic symbol of AND gate



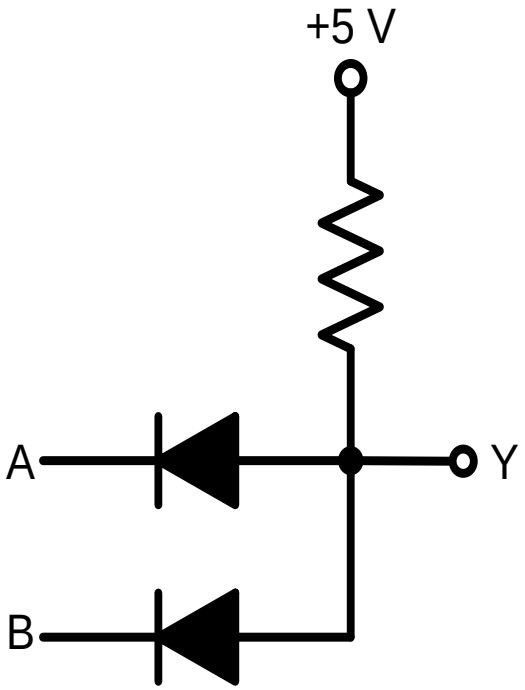
Logic symbol of three-input AND gate

Summary of the AND operation

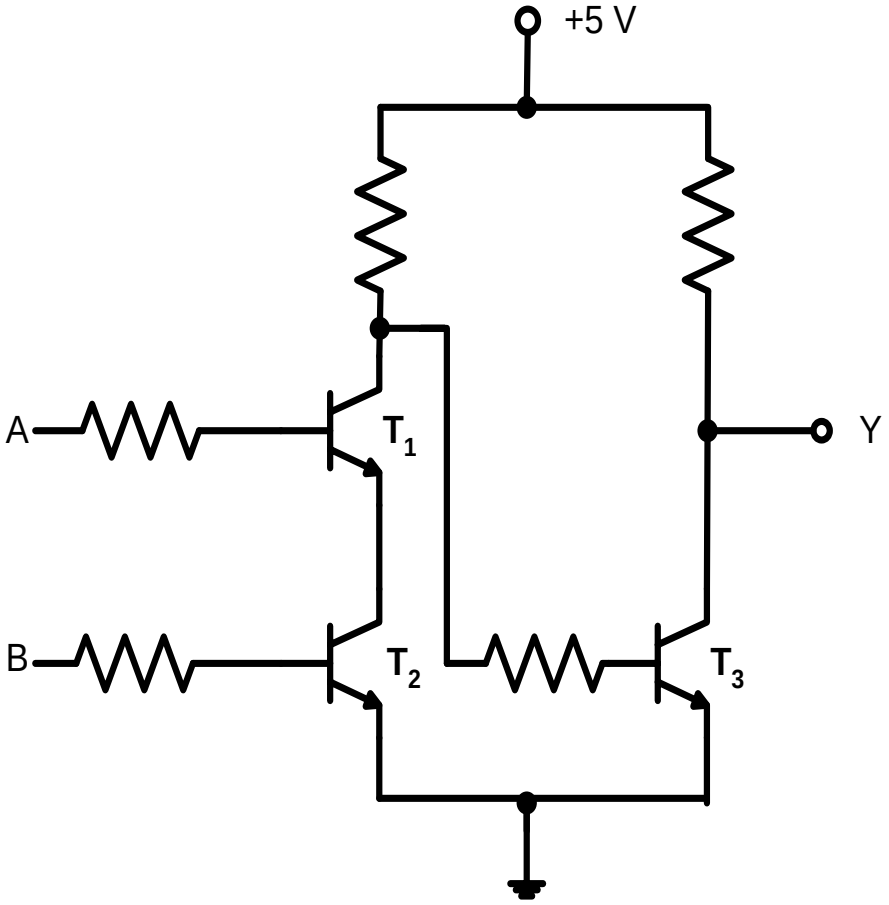
- **The AND operation is performed exactly like ordinary multiplication 1s and 0s**
- **The output equal to 1 occurs only for the single case where all inputs are 1**
- **The output is 0 for any case where one or more inputs are 0.**

A	B	C	Y=ABC
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

Discrete **AND** Gate



Two-input
diode AND Gate



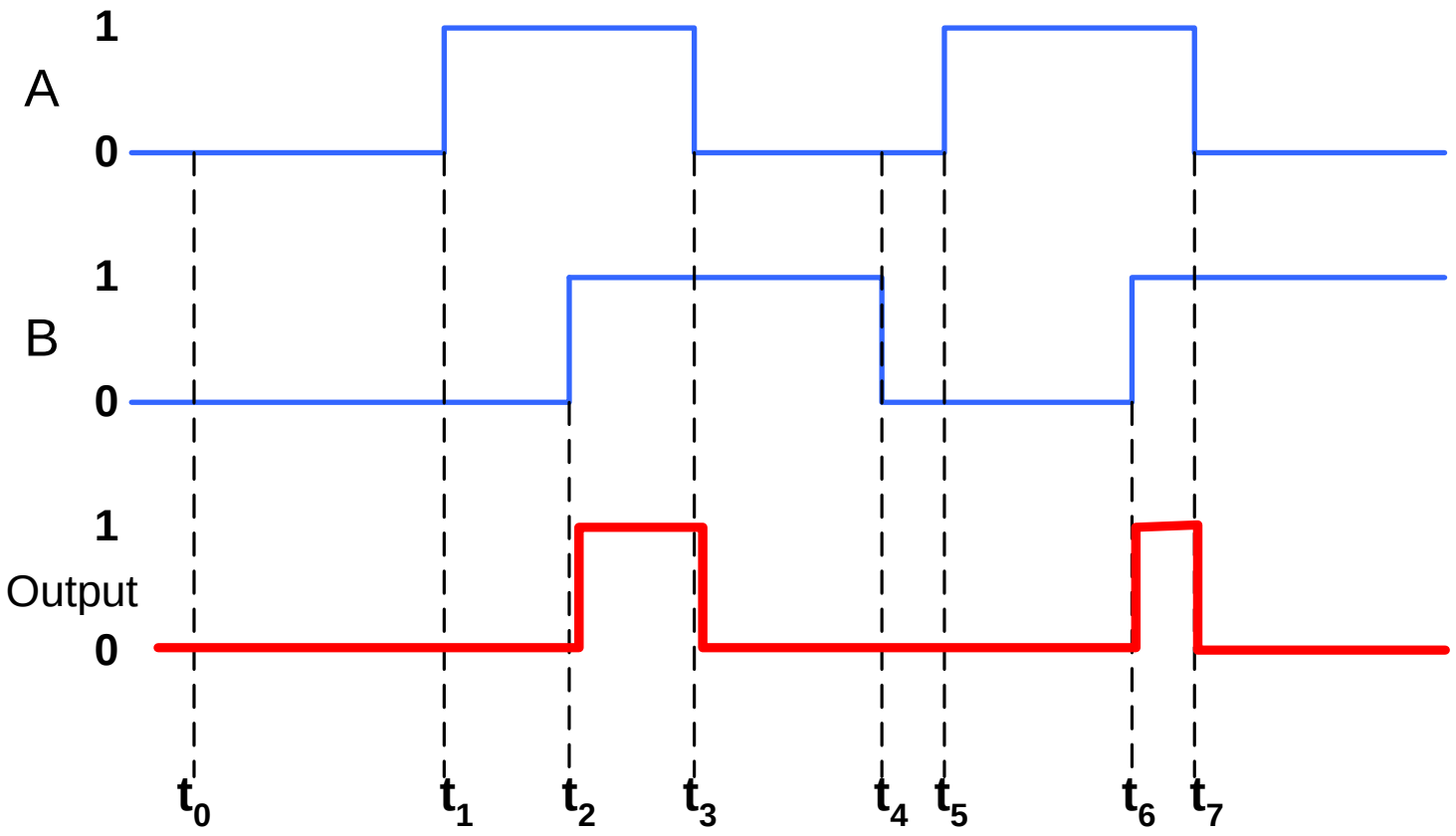
Two-input
Transistor AND gate

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

Example: Determine the output from the AND gate in Figure below for the given input waveforms.

Solution



NOT OPERATION

- The NOT operation is unlike the OR and AND operation in that it can be performed ***on a single input variable***.
- Example: If the variable A is subjected to the NOT operation, the result Y can be expressed as

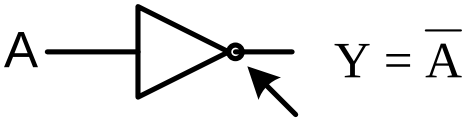
$$Y = \bar{A}$$

- Where the over bar represents the not operation.
- This expression is read as
- *''Y equals NOT A''* or
- *''Y equals the inverse of A ''* or
- *''Y equals the complement of A''.*
- All indicate that the logic value of $Y = \bar{A}$ is opposite to the logic value of A.
- The truth table below clarifies this for the two cases

A	$Y = \bar{A}$
0	1
1	0

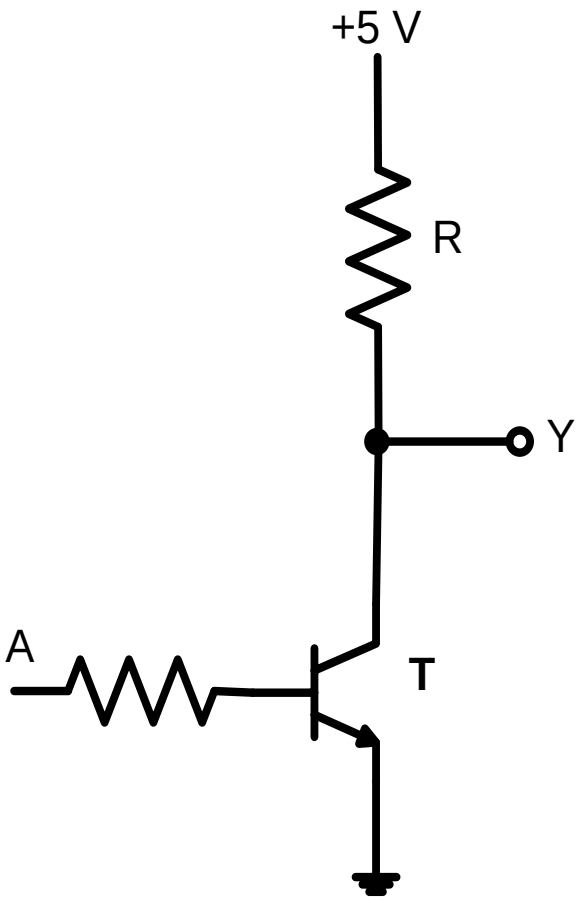
NOT Circuit (INVERTER)

- Fig below, shows the symbol for a not circuit, which is more commonly called an INVERTER.
- This circuit always has only a single input and its output logic level is always opposite to the logic level of this input.



Presence of small circle always denotes inversion

Discrete **NOT** Gate



Transistor as Inverter

A	Y
0	1
1	0

Truth Table

Summary of Boolean operation

- **The rules for the OR, AND, and NOT operation may be summarized as follows:**

OR	AND	NOT
0+0 = 0	0·0 = 0	$\bar{0} = 1$
0+1 = 1	0·1 = 0	$\bar{1} = 0$
1+0 = 1	1·0 = 0	
1+1 = 1	1·1 = 1	

Derived Logic Gates

NOR GATES & NAND GATES

- Two other types of logic gates, NOR gates and NAND gates, are used extensively in digital circuitry.
- These gates are derived from combination of the basic AND, OR and NOT gates, which make it relatively easy to describe them using the Boolean algebra operations learned previously.

NOR Gate

- The NOR gate is actually a NOT OR gate. In other words, the output of an OR gate is inverted to form a NOR gate.
- The logic symbol for the NOR gate is diagramed in fig (a).
- Note that the NOR symbol is an OR symbol with a small invert bubble (small circle) on the right side
- The NOR function is being performed by an OR gate and an INVERTER in fig (b)
- The Boolean expression for the final NOR function is $Y = \overline{A + B}$.
- The truth table in fig below shows that the NOR gate output is the exact inverse of the OR gate output for all possible input conditions.
- The OR gate output goes HIGH when any input is HIGH, while the NOR gate output goes LOW when any input is HIGH.

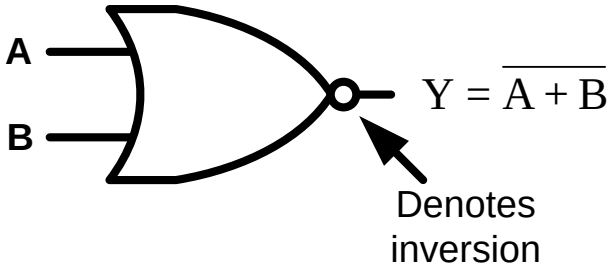


Fig (a)

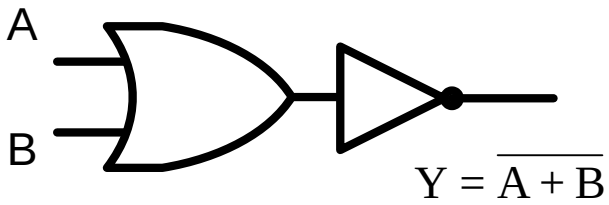
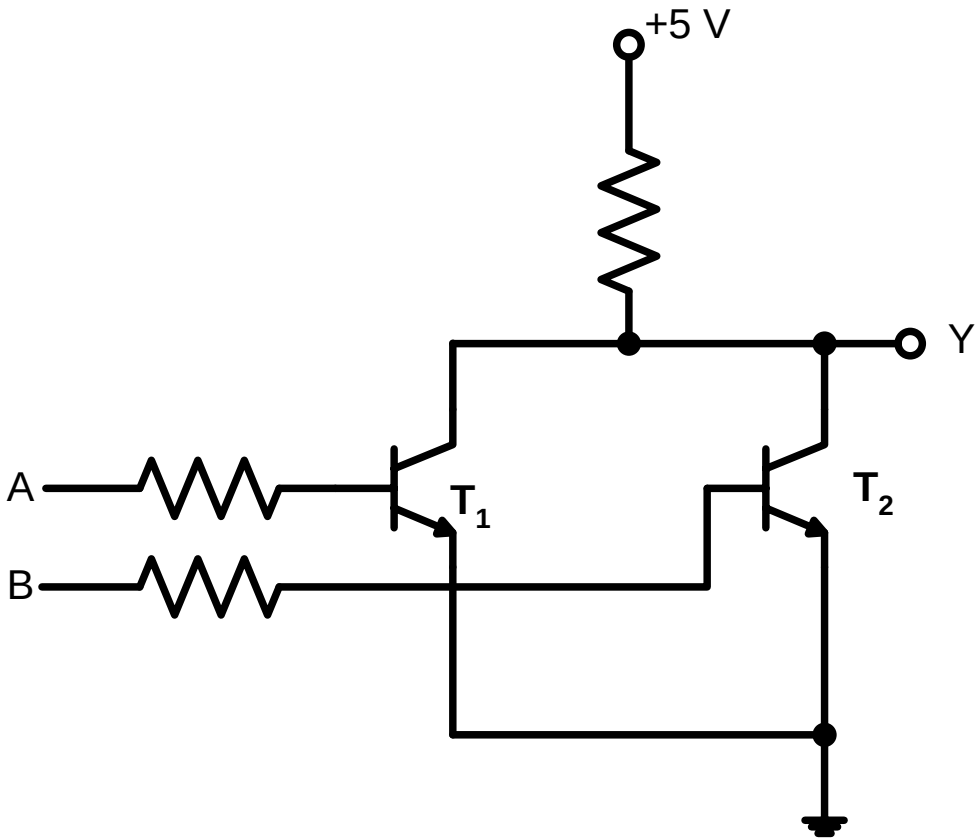


Fig (b)

INPUTS		OUTPUTS	
A	B	OR	NOR
0	0	0	1
1	0	1	0
0	1	1	0
1	1	1	0

Fig (c)
Truth table for
OR and NOR gates

Discrete **NOR** Gate



Discrete two-input NOR gate

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

Truth Table

NAND Gate

- The NAND gate is a NOT AND, or an inverted AND function.
- The Standard logic symbol for the NAND gate is diagramed in fig (a).
- The little invert bubble (small circle) on the right end of the symbol means to invert the AND.
- Fig (b) shows a separate AND gate and inverter being used to produce the NAND logic function.
- The truth table in fig. (c) shows that the NAND gate output is the exact inverse of the AND gate for all possible input conditions
- The AND output goes HIGH only when all inputs are HIGH, while the NAND output goes LOW only when all input are HIGH.

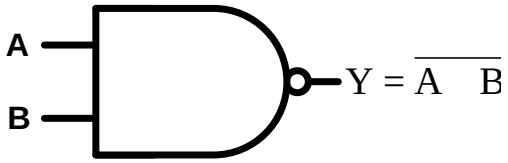


Fig (a)

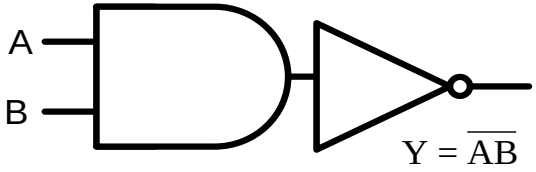
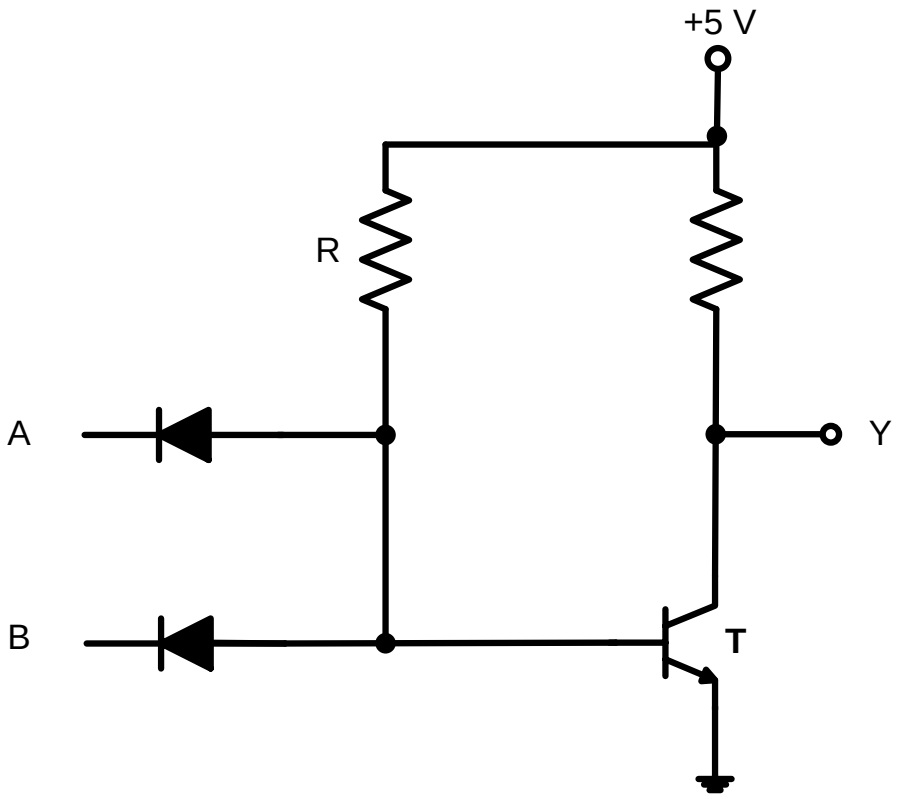


Fig (b)

INPUTS		OUTPUTS	
A	B	AND	NAND
0	0	0	1
1	0	0	1
0	1	0	1
1	1	1	0

Fig (c)

Discrete NAND Gate



Discrete two-input NAND gate

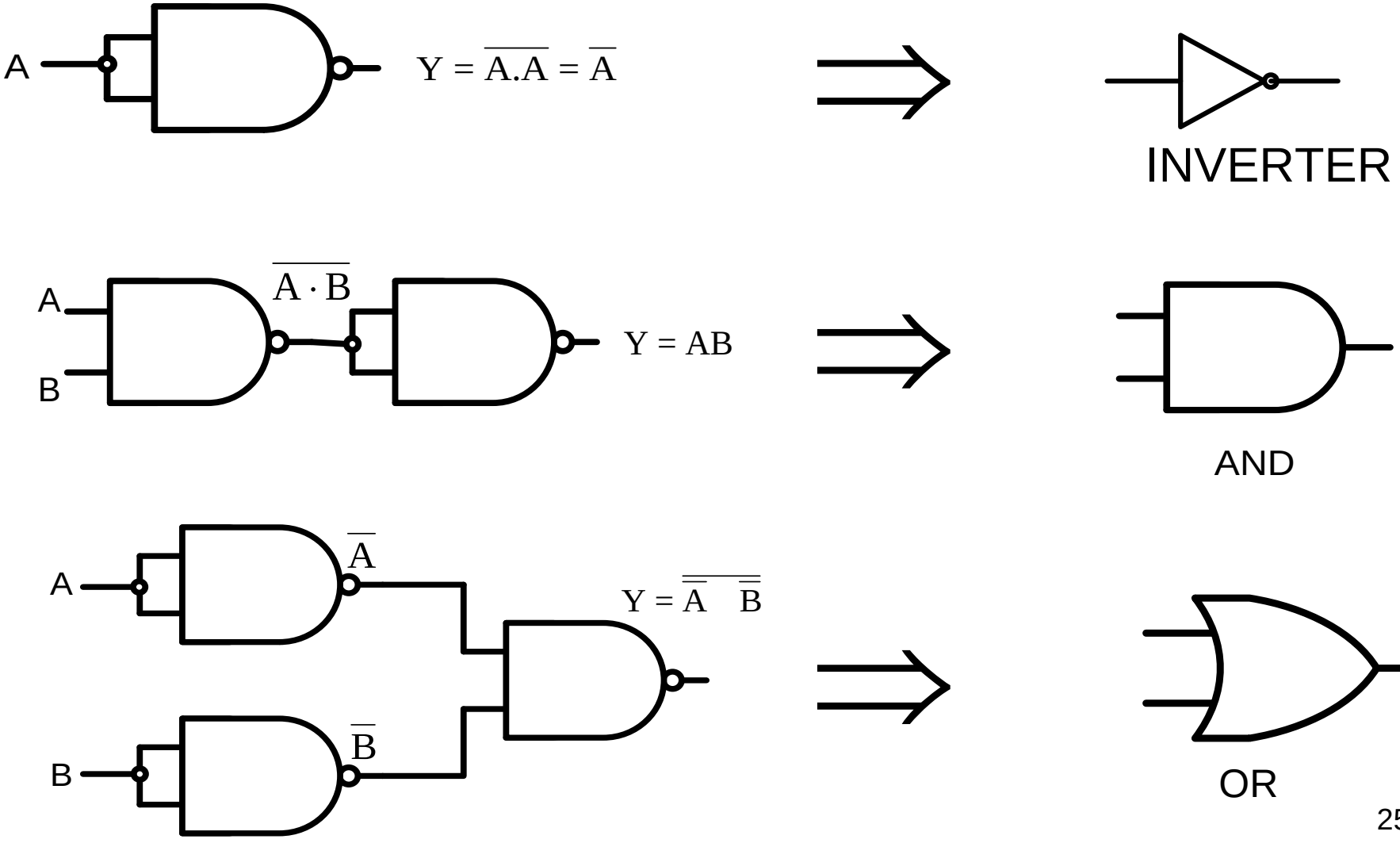
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

Universality of NAND Gates and NOR gates

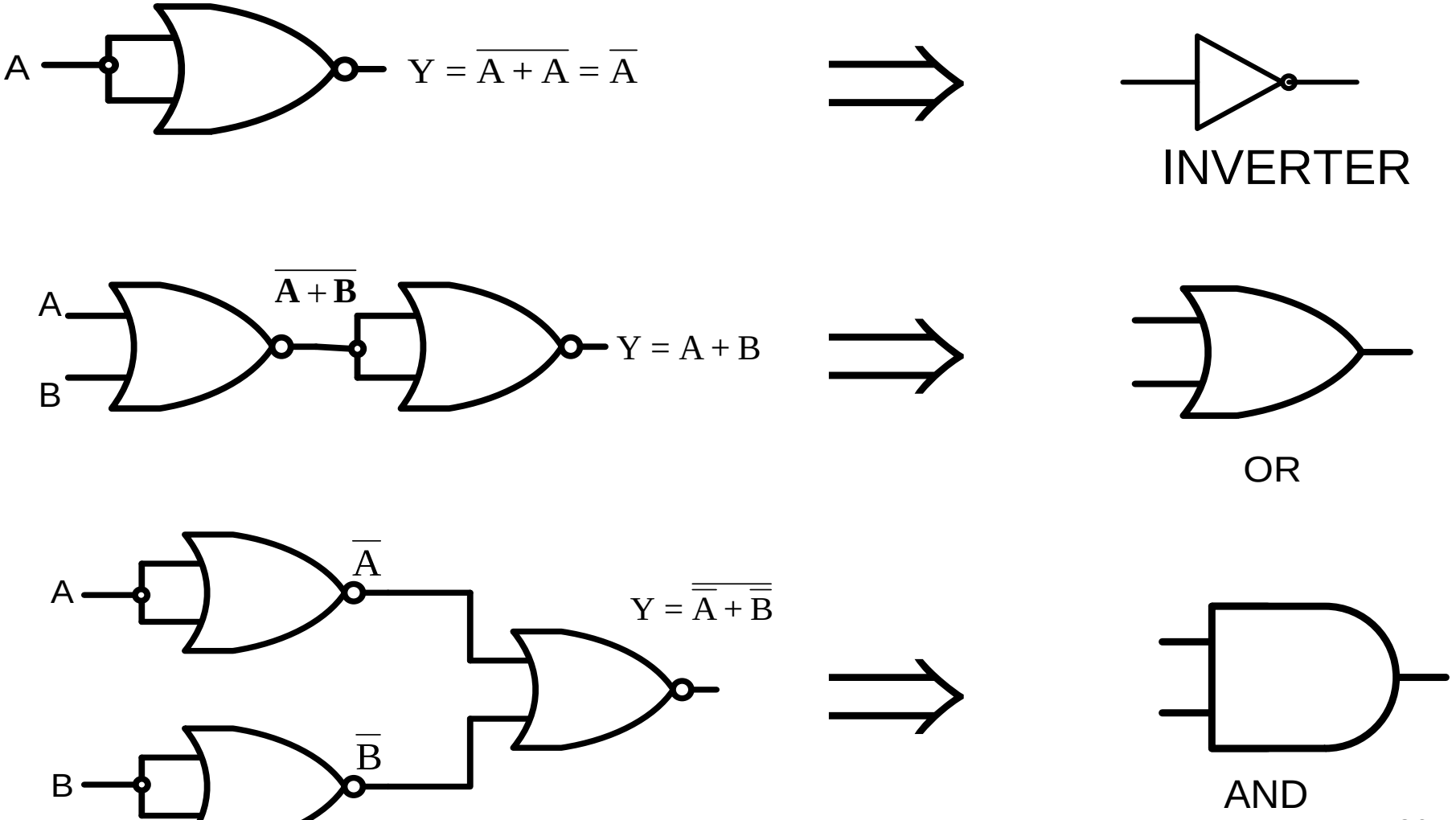
- All Boolean expressions consist of various combinations of the basic operations of OR, AND, and INVERTER.
- Therefore, any expression can be implemented using combinations of OR gates, AND gates and INVERTERS.
- It is possible, however to implement any logic expression using only NAND gates and no other type of gate.
- This is because NAND gates, in the proper combination, can be used to perform each of the Boolean operations OR, AND, and INVERTER.
- This is demonstrated in fig below.

Universality of NAND Gates



Universality of NOR Gates

- **Similarly, it can be shown that NOR gates can be arranged to implement any of the Boolean operations. See fig. below**



XOR and XNOR Logic Circuits

- Two special logic circuits that occur quite often in digital systems are the exclusive-OR and exclusive NOR circuits.

Exclusive OR

- Consider the logic circuit of fig (a) .
- The output expression of this circuit is $Y = \overline{A}B + A\overline{B}$
- The accompanying truth table shows that x= 1 for two cases:

  A= 0, B= 1 (the $\overline{A}B$ term) and

  A= 1, B= 0 (the $A\overline{B}$ term).
- In other words, this circuit produces a HIGH output whenever the two inputs are at opposite levels.
- This is the exclusive OR circuit, which will hereafter be abbreviated XOR.
- This particular combination of logic gates occurs quite often and is very useful in certain applications.

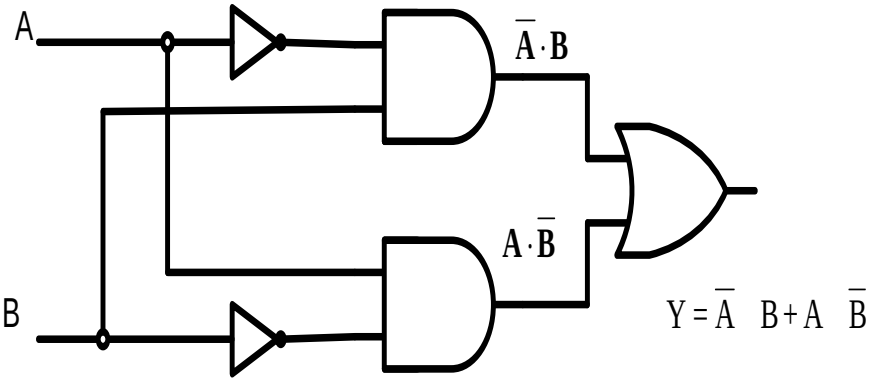
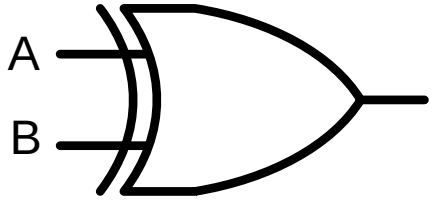


Fig. (a)

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

- In fact, the XOR circuit has been given a symbol of its own as shown in fig below
- This XOR circuit is commonly referred to as an XOR gate, and we consider it as another type of logic gate.
- An XOR gate has only two inputs; there are no three-input or four- input XOR gates.
- A short hand way that is sometimes used to indicate the XOR output expression is Y= A⊕B.
- Where, the symbol ⊕ represents the XOR gate operation.
- The characteristics of an XOR gate are summarized as follows:



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

1. It has only two inputs and its output is

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

2. Its output is HIGH only when the two inputs are at different levels

Exclusive –NOR

- The exclusive-NOR circuit (Abbreviated XNOR) operates completely opposite to the XOR circuit.
- The fig (a) drawn below, shows an X-NOR circuit and its accompanying truth table.
- The output expression is $Y = \overline{A}\ \overline{B} + A\ B$
- Which indicates along with the truth table that Y will be 1 for two cases:
- A=B=1 (the AB term) and
- A=B=0 (the $\overline{A} \cdot \overline{B}$ term).
- In other words, this circuit produces a HIHG output whenever the two inputs are at the same level.
- In should be apparent that the output of the XNOR circuit is the exact inverse of the output of the XOR circuit.

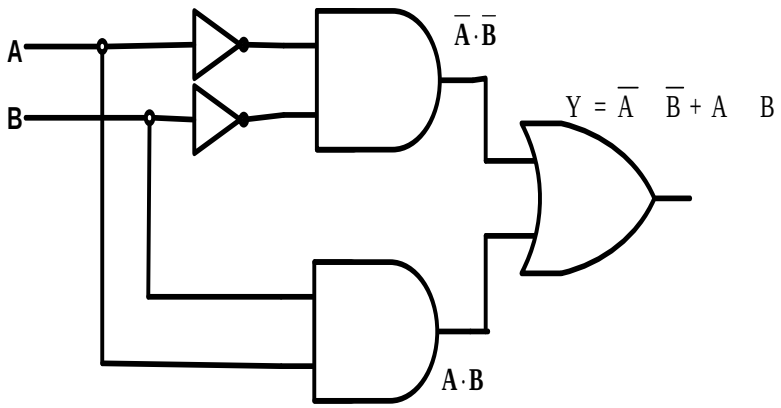
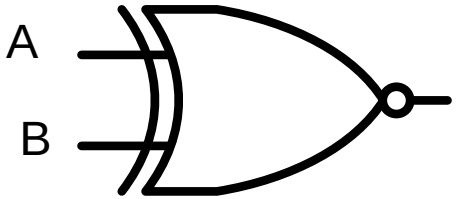


Fig.(a)

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

- **The traditional symbol for an XNOR gate is obtained by simply adding a small circle at the out put of the XOR symbol.**



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

- **A shorthand way to indicate the output expression of the XNOR is**

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

- **This is simply the inverse of the XOR operation.**
- **The XNOR gate is summarized as follows:**

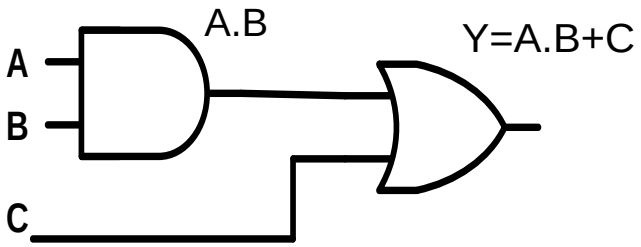
1. **It has only two inputs and its output is**

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

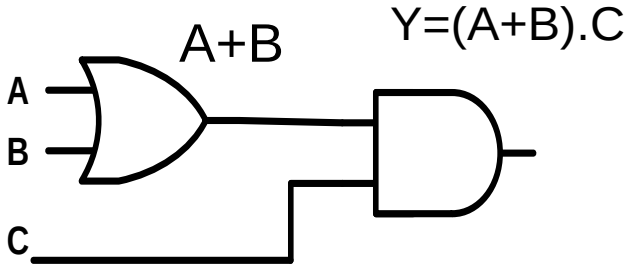
2. **Its output is HIGH only when the two input are the same level.**

DESCRIBING LOGIC CIRCUITS ALGEBRAICALLY

- Any logic circuit, no matter how complex, may be completely described using the Boolean operations previously defined, because the OR gate, AND gate, and NOT circuit are the basic building blocks of digital systems.
- For example, consider the circuit in Figure (a)
- This circuit has three inputs, A, B, and C and a single output, Y.
- Utilizing the Boolean expression for each gate, we can easily determine the expression for the output.
- Occasionally, there may be confusion as which operation to be performed first. The expression Y=A·B+C can be interpreted in two different ways:
  - A. B is OR$_{ed}$ with C
  - A is AND$_{ed}$ with the term B+C
- To avoid this confusion, it will be understood that if an expression contains both AND and OR operations the AND operations are performed first, unless there are *parentheses* in the expression, in which case the operation inside the parentheses is to be performed first.
- Try to write the expression to Figure (b)



(a) Logic circuit which does not require parenthesis



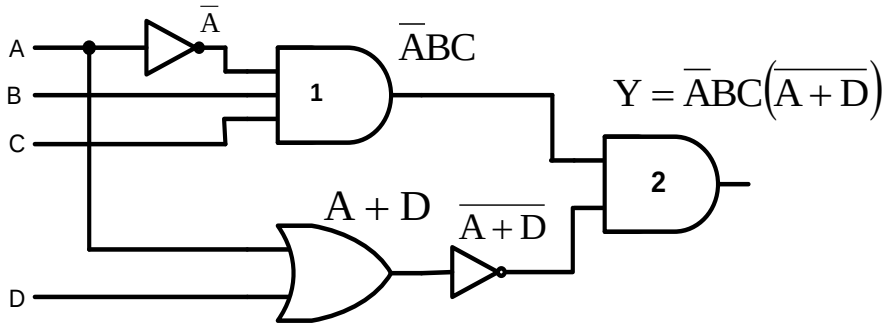
(b) Logic circuit requires parenthesis

EVALUATING LOGIC–CIRCUIT OUTPUTS

- Once the Boolean expression for a circuit output has been obtained, the output logic level can be determined for any set of input levels
- For example: suppose that we want to know the logic level of the output Y for the circuit in Figure (a) for the case where A=0, B=1, C=1, and D=1.
- As in ordinary algebra, the value of Y can be found by '***plugging***' the values of the variables into the expression

In general, the following rules must always be followed when evaluated a Boolean expression

1. First ,perform all inversions of single terms; that is
$$\bar{0} = 1 \text{ or } \bar{1} = 0$$
2. Then perform all operations within parentheses.
3. Perform an AND operation before an OR operation unless parentheses indicate otherwise
4. If an expression has a bar over it, perform the operations of the expression first and then invert the result.

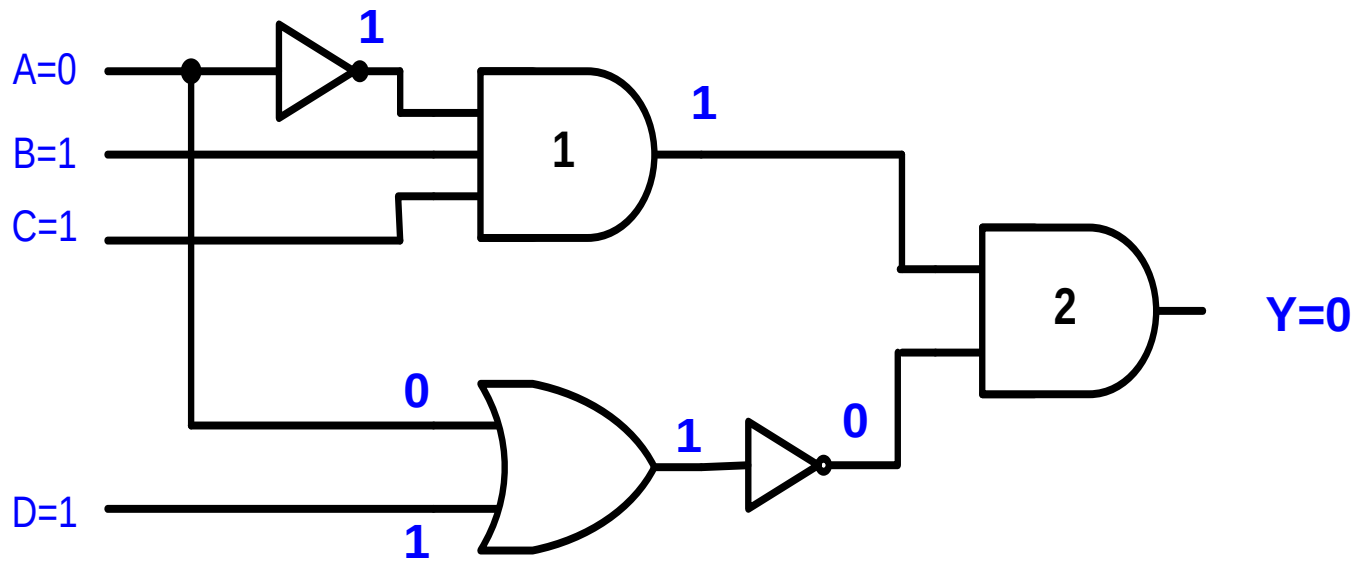


(a)

$$Y = \bar{A}BC\overline{(A + D)}$$
$$= \bar{0}.1.1.\overline{(0 + 1)}$$
$$= 1.1.1.\bar{1}$$
$$= 0$$

Determining output level from a Diagram

- The output logic level for given input levels can also be determined directly from the circuit diagram *without* using the Boolean expression.
- Technicians often use this technique during the troubleshooting or testing of logic system since it also tells them what each gate output is supposed to be as well as the final output.



Thank You!!!

Any Question???