Digital Electronics
ECE 2203

Professor Dr. Md. Shamim Ahsan
Electronics and Communication Engineering Discipline
Khulna University.

Lecture 2
Outline

Boolean Algebra & Combinational Logic Circuits

- Boolean Algebra
- Logic Gates and Operation:
  - AND, OR, NOT, NOR, NAND, XOR, XNOR
- Describing Logic Circuits Algebraically
- Determining Output Level from a Diagram
- Implementing Circuits from Boolean Expressions
Boolean Algebra & Combinational Logic Circuits

- **Boolean Algebra:**
  - Digital circuits operate in the binary mode where each input and output voltage is either a 0 or a 1; 0 and 1 designates represent predefined voltage ranges.
  - Boolean algebra is a means for expressing the relationship between a logic circuit’s inputs and outputs.

- **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 0 and 1 do not present actual numbers but instead represent the state of a voltage variable, or what is called its logic level. A voltage in a digital circuit is said to be at the logic 0 level or logic 1 level, depending on the actual numerical value.
  - In fact, in Boolean algebra there are only three basic operations: **OR, AND, and NOT**. These basic operations are called logic operations.
  - Digital circuits called logic gates can be constructed from diodes, 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:

- A Truth Table is a means for describing how a logic circuit’s output depends on the logic levels present at the circuit inputs. Figure 1 illustrates a truth table for one type of two-input logic circuit.

![Truth Table](image)

**Figure 1.** Example of Truth Table for two-input.

- The table lists all possible combinations of logic levels present at inputs A and B along with the corresponding output level x.
**OR Operation:**

- The Truth Table in Figure 2(a) shows what happens when two logic inputs, A and B, are combined using the **OR Operation** to produce the output x. The table shows that x is a logic 1 for every combination of input levels where one or more inputs are 1. The only case where x is a logic 0 is when both inputs are 0.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>x = A + B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 2. (a) Truth table defining the OR operation; (b) circuit symbol for a two-input OR gate.*

- The Boolean expression for the **OR Operation** is ~

\[ x = A + B \quad (1) \]

- In the operation, the ‘+’ sign does not stand for ordinary addition; it stands for the operation.
**OR Gate:**

- In digital circuitry an **OR Gate** is a circuit that has two or more inputs and whose output is equal to the **OR** combination of the inputs. **Figure 2(b)** is the logic symbol for a two-input **OR** gate.
- The inputs A and B are logic voltage levels, and the output x is a logic voltage level whose value is the result of **OR** operation on A and B; that is \( x = A + B \).
- **Figure 3** shows a three-input **OR** gate and its Truth Table. The Boolean expression for the three-input **OR** operation is

\[
x = A + B + C
\]

(2)

**Figure (3).** Symbol and Truth Table for a three-input **OR** gate.
OR Gate:

- The examination of the Truth Table shows again that the output will be 1 for every case where one or more inputs are 1. This general principle is the same for all OR gates with any number of inputs.
- Figure 4 shows the standard TTL SSI two input OR gate.

Figure (4). Quadruple two-input OR-gates.
OR Gate:

Example (1): Determine the OR gate output in Figure 5. The OR gate inputs A and B are varying according to the timing diagrams shown. For example, A starts out LOW at time $t_0$, goes HIGH at $t_1$, back LOW at $t_3$, and so on.

Solution:

Between time $t_0$ and $t_1$, both inputs are LOW; so output = LOW. At $t_1$, input A goes HIGH while B remains LOW. Thus output goes HIGH at $t_1$ and stay HIGH until $t_4$ since during this interval one or both inputs are HIGH. At $t_4$, input B goes from HIGH to LOW so that now both inputs are LOW; that’s why, output returns back to LOW. At $t_5$, A goes HIGH sending output back to HIGH where it stays rest of the time.
AND Operation:

- The Truth Table in Figure 6(a) shows what happens when two logic inputs, A and B, are combined using the **AND Operation** to produce output $x$. The Table shows that $x$ is a **logic 1** only when both inputs (A and B) are at the **logic 1** level. For any case where one of the inputs is 0, the output is 0.

\[
\begin{array}{ccc}
A & B & x = A \cdot B \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

**Figure 6.** (a) Truth Table for AND Operation; (b) AND gate symbol.

- The Boolean expression for the **AND Operation** is ~

\[
x = A \cdot B
\]  

(3)

- In this expression, the ‘.’ sign stands for the Boolean **AND** operation and not the multiplication operation.
**Boolean Algebra & Combinational Logic Circuits**

- **AND Gate:**
  - The logic symbol for a two-input AND gate is shown in Figure 6(b). The AND gate output is equal to the AND product of the logic inputs; that is \( x = AB \). 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 characteristics of AND gates with more than two inputs. For example, a three-input AND gate and its accompanying Truth Table are shown in Figure 7.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>( x = ABC )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

*Figure 7. Truth Table and symbol for a three-input AND gate.*
Boolean Algebra & Combinational Logic Circuits

**AND Gate:**

**Example (2):** Determine the output $x$ from the AND gate in Figure 8 for the given input waveforms.

![Waveforms and AND gate diagram](image)

**Solution:**

For the given input waveforms, the output becomes HIGH only during the intervals $t_2 - t_3$ and $t_6 - t_7$. At all other times, one or more of the inputs are 0, thereby producing a LOW output.
AND Gate:

- Figure 9 shows standard TTL Small Scale Integration (SSI) circuit for a two-input AND gates.

![7408 Quadraple two-input AND gates](image)

**Figure 9.** Quadraple two-input AND gates.
**NOT Operation:**

- **NOT Operation** performs on a single input variable. For example, if the variable A is subjected to the **NOT Operation**, the result x can be expressed as
  \[ x = \overline{A} \quad (4) \]

- Where the over bar represents the **NOT Operation**. Expression of Equation (4) is read as ‘x equals NOT A’ and indicates that the logic value of \( x = \overline{A} \) is opposite to the logic value of A. The Truth Table in **Figure 10(a)** clarifies **NOT Operation** for the two cases: A = 0 and A = 1. That is:

  \[ 0 = \overline{1} \quad \text{because 0 is not 1} \]

  and

  \[ 1 = \overline{0} \quad \text{because 1 is not 0} \]

**Figure 10.** (a) Truth Table for NOT Operation; (b) Symbol of NOT gate; (c) sample waveforms.
Boolean Algebra & Combinational Logic Circuits

- **NOT Operation:**
  - The **NOT Operation** is also referred to as inversion or complementation. Another indication for inversion is the prime symbol ‘/’. That is $A’ = \bar{A}$.

- **NOT Circuit (INVERTER):**
  - **Figure 10(b)** shows the symbol for a **NOT** circuit, which is most 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.
  - **Figure 10(c)** shows how the **INVERTER** affects an input signal. It inverts the input signal at all points of the waveform.
NOT Gate:

- Figure 11 shows standard TTL (SSI) circuit for hex INVERTERs/NOT gates.

**Figure 11.** Hex INVERTERs/NOT gates.
NOR Gate:

- The NOR gate operates like an OR gate followed by an INVERTER.

- The Boolean Expression for two input (A and B) NOR gate is:

  \[ x = \overline{A + B} \]  

- The symbol for a two-input NOR gate is shown in Figure 12(a) and the corresponding truth table is shown in Figure 12(b).

- The equivalent circuit of NOR gate is shown in Figure 12(c).

- The truth table of Figure 12(b) shows that the NOR gate output is the exact inverse of the OR gate output for all possible input conditions.

- The same operation can be extended to NOR gates with more than two inputs.
NOR Gate:

**Example (3):** Determine the waveform at the output of a NOR gate for the input waveforms shown in Figure 13.

![NOR Gate Diagram]

**Solution:**

One way to determine the NOR output waveform is to find first the OR output waveform and then invert it. Another way utilizes the fact that a NOR gate output will be **HIGH** only when all inputs are **LOW**. Thus, we can examine the input waveforms, find those time intervals where they are all **LOW**, and make the NOR output **HIGH** for those intervals. The NOR output will be **LOW** for all other time intervals. The resultant waveform is shown in the figure.
NAND Gate:

- The symbol of a two-input NAND gate is shown in Figure 14(a).
- The NAND operates like an AND gate followed by an INVERTER which is shown in Figure 14(b).
- The truth table for two-input NAND gate is shown in Figure 14(c).
- The output expression for the NAND gate is \( x = \overline{AB} \) (6)
- The truth table in Figure 14(c) shows that the NAND gate output is the exact inverse of the AND gate for all possible input conditions.
- This same characteristics is true for NAND gates having more than two inputs.

Figure 14. (a) NAND Symbol; (b) equivalent circuit; (c) truth table.
Example (4): Determine the output waveform of a NAND gate having the inputs shown in Figure 15.

Solution:

One way is to draw first the output waveform for an AND gate and then invert it. Another way utilizes the fact that a NAND output will be LOW only when all inputs are HIGH. Thus, you can find those time intervals during which the inputs are all HIGH, and make the NAND output LOW for those intervals. The output will be HIGH at all other times.
NOR and NAND Gates:

- Figure 16 shows standard TTL (SSI) circuit for quadruple two-input NOR and NAND gates.

Figure 16. (a) Quadruple two-input NOR gates; (b) quadruple two-input NAND gates.
Exclusive-OR (XOR) Gate:

- Boolean expression for two-input XOR operation is
  \[ x = A \oplus B = \overline{AB} + A\overline{B} \]  (7)
- The standard logic symbol and truth table for XOR gate are shown in Figures 17(a) and 17(b).
- From the truth table of Figure 17(b) we find that, the output of the XOR gate is HIGH or 1 if and only if its inputs are not simultaneously HIGH or LOW.
- When all the inputs are HIGH or LOW, the output is LOW or 0.
XOR Gate:

- **Figure 18** shows standard **TTL (SSI)** circuit for quadruple two-input XOR gate.

![Quadraple two-input XOR gate](image)

**Figure 18.** Quadruple two-input XOR gate.
Exclusive-NOR (XNOR) Gate:

- **XNOR** is the complement of **XOR**. The Boolean expression for two-input XNOR operation is:

\[
x = \overline{A \oplus B} = A \odot B = AB + \overline{A\overline{B}}
\]  

(8)

- The standard logic symbol and truth table for XNOR gate are shown in Figures 19(a) and 19(b).

- From the truth table of Figure 19(b) we find that, the output of the XNOR gate is **HIGH** or 1 if and only if its inputs are simultaneously **HIGH** or **LOW**.

- When all the inputs are not simultaneously **HIGH** or **LOW**, the output is **LOW** or 0.

**Figure 19.** (a) XNOR Symbol; (b) truth table.
XNOR Gate:

- Figure 20 shows standard TTL (SSI) circuit for quadruple two-input XNOR gate.

![XNOR Gate Diagram]

**Figure 20.** Quadruple two-input XNOR gate.
Describing Logic Circuits Algebraically:

- Any logic circuit, no matter how complex, can be completely described using the three basic Boolean operations (AND, OR, and NOT).
- For the circuit of Figure 21, the Boolean expression becomes:

\[ x = A \cdot B + C \]

Figure 21. Logic circuit with its Boolean expression.
Describing Logic Circuits Algebraically:

Example (5) (Tocci-Exercise-B: 3-12(b)): Write the Boolean expression for output x in Figure 22. Determine the value of x for all possible input conditions, and list the values in a truth table.

Solution:

The output from the upper, middle, and lower AND gates are: \( \overline{A}BC \), \( ABC \), and \( ABD \)

So the final expression is \( \overline{x} = \overline{A}BC + ABC + ABD \).
Describing Logic Circuits Algebraically:

Example (5) (Tocci-Exercise-B: 3-12(b)):

Solution (Continued):

```
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>AB,C</th>
<th>AB,C</th>
<th>ABD</th>
<th>x = AB,C + AB,C + ABD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Figure 23. The truth table for the circuit of Figure 22.
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.
- Suppose the inputs to **Figure 24** are, \( A = 0 \), \( B = 1 \), \( C = 1 \), and \( D = 1 \). The procedure is to start from the inputs and to proceed through each INVERTER and gate, writing down each of their outputs in the process until the final output is reached.

**Figure 24.** Logic circuit with its Boolean expression.
Implementing Circuits from Boolean Expressions:

- When the operation of a circuit is defined by a Boolean expression, we can draw a logic-circuit diagram directly.
- Suppose that we want to construct a circuit whose output is \(~ y = AC + B\overline{C} + \overline{A}BC\).
- The circuit diagram is shown in Figure 25.

Figure 25. Logic circuit with its Boolean expression.
References
