2

Logic Gates

2.1. Introduction

Logic functions provide ways to combine different digital signals – or signals that can only take one of two possible levels: low level (0) and high level (1) – based on the laws of Boolean algebra. These laws are applied using logic gates, which can be classified according to the number of available inputs.

Each logic gate has an equivalent electric circuit. However, an electronic logic gate is very different from its electrical equivalent. It is much faster, smaller, and consumes less electric energy.

Figure 2.1 shows the electric circuit that corresponds to the NOT gate. The lightemitting diode comes on when the switch S_1 is opened and goes off when the switch S_1 is closed.

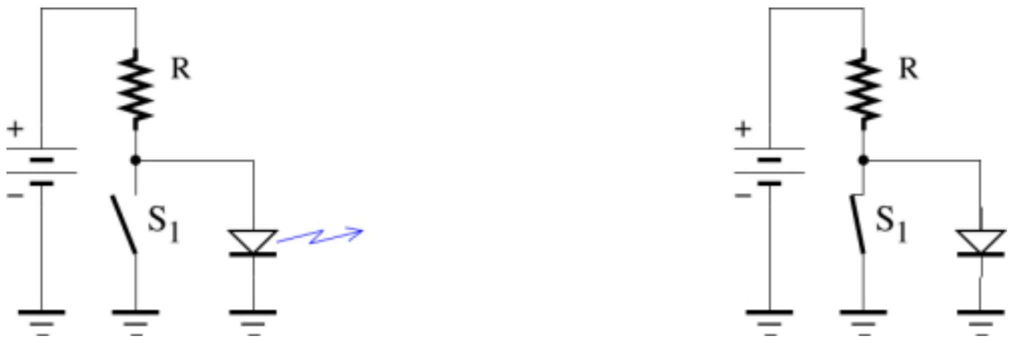

Figure 2.1. *Electric circuit that is the equivalent of the NOT gate*

The electric circuit shown in Figure 2.2 operates as an AND gate. The diode lights up if and only if both switches S_1 and S_2 are closed.

Figure 2.3 shows the electric circuit for the OR gate. The diode comes on if at least one of the switches (S_1 or S_2) is closed.

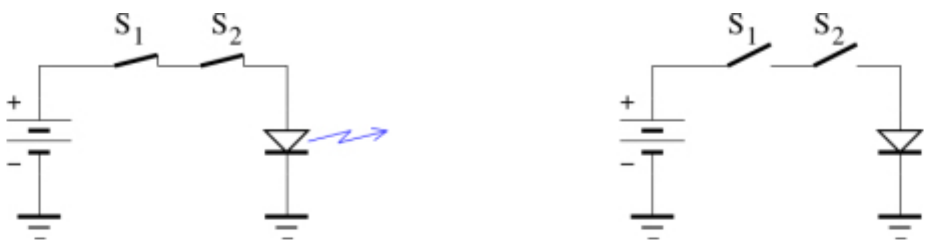

Figure 2.2. *Electric circuit corresponding to the AND gate*

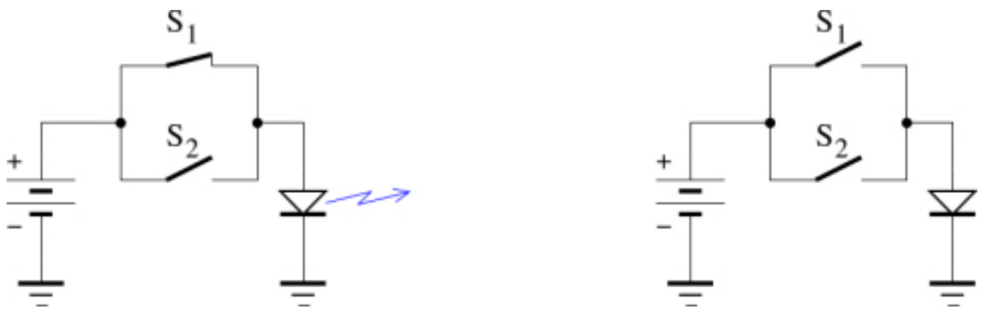

Figure 2.3. *Electric circuit corresponding to the OR gate*

The electric circuit corresponding to the XOR gate is illustrated in Figure 2.4. The diode emits visible light when either the switch S_1 or the switch S_2 is closed.

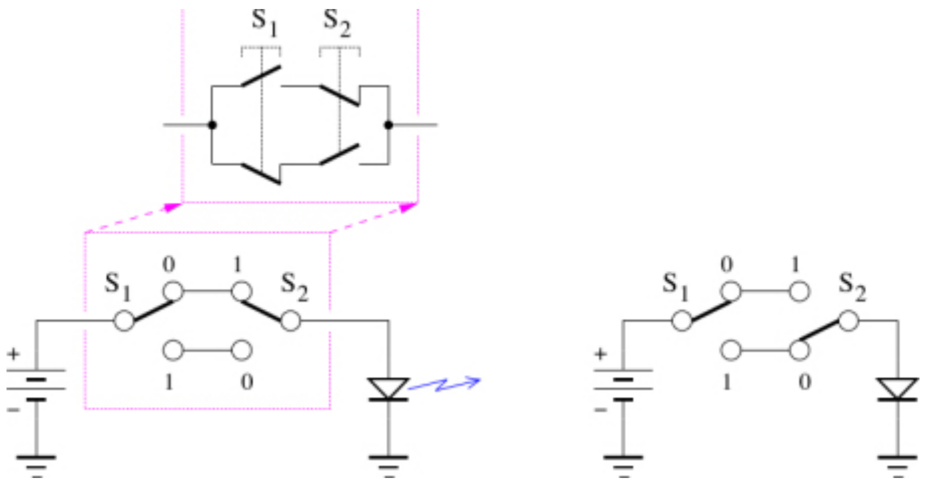

Figure 2.4. *Electric circuit corresponding to the XOR gate, where pressure on either push button S_1 or push button S_2 turns on the diode, but pressure on push button S_1 and push button S_2 turns off the diode*

2.2. Logic gates

Logic gates can be used to combine digital signals based on basic Boolean functions.

2.2.1. *NOT gate*

The NOT function provides the complementary state to a given variable. The function is represented by a bar placed above the input variable and implemented by a NOT gate (or logic inverter).

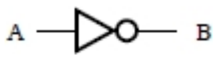

Figure 2.5. *NOT gate.* $B = \overline{A}$

Table 2.1. *Truth table. Input: A; Output: B*

A	B
0	1
1	0

Figure 2.6 depicts the symbol for a NOT gate. The logic level of the output variable is obtained by taking the complement of the input variable, as shown in the truth table given in Table 2.1. Thus, if an input is at logic level 0, the output will be at logic level 1, and vice versa.

2.2.2. *AND gate*

The AND function, which is also called logic product, is represented by a dot (·).

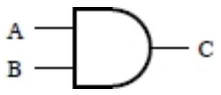

Figure 2.6. *AND gate* $C = A \cdot B$

The AND gate can have two inputs, as illustrated in Figure 2.6, the output variable takes the high logic level (or the value 1) if and only if the input variables are both at the high logic level (or the value 1). In all other cases, the output is set to the low logic level (or the value 0). Table 2.2 shows the truth table for the AND gate.

Table 2.2. *Truth table. Inputs: A, B; Output: C*

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

2.2.3. *OR gate*

The OR function, which is also called logic addition, is represented by a plus (+).

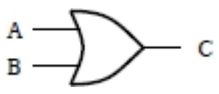

Figure 2.7. *OR gate C = A + B*

Table 2.3. *Truth table. Inputs: A, B; output: C*

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

Figure 2.7 depicts the OR gate, which has two gates. As illustrated by the truth table in Table 2.3, the output takes the logic level 1 if at least one of the two inputs is at the logic level 1, it takes logic level 0 if both the inputs are at the logic level 0.

2.2.4. *XOR gate*

The XOR (exclusive OR) function is represented by a plus within a circle (⊕).

Figure 2.8 depicts the symbolic representation of an XOR gate having two inputs. According to the truth table shown in Table 2.4, the output takes either logic level 1, when only one of the inputs is at logic level 1, or logic level 0, when both inputs are either at logic level 0 or at logic level 1.

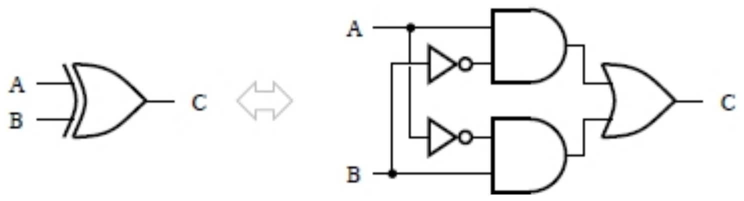

Figure 2.8. *XOR gate (exclusive OR)* $C = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$

Table 2.4. *Truth table. Inputs: A, B; Output: C*

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

2.2.5. *Complementary logic gates*

The NAND (NOT AND), NOR (NOT OR), and XNOR (NOT exclusive OR or inclusive AND) gates are said to be complementary and correspond, respectively, to the AND, OR and XOR gates when followed by a NOT gate. They are characterized by the following logic equations:

– NAND gate: $C = \overline{A \cdot B}$;

– NOR gate: $C = \overline{A + B}$;

– XNOR gate: $C = \overline{A \oplus B} = A \cdot B + \overline{A} \cdot \overline{B} = A \odot B$.

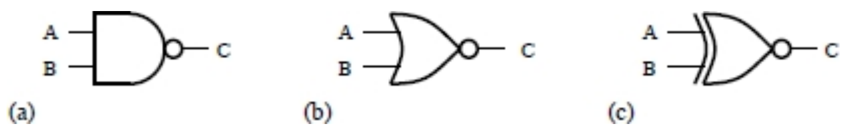

Figure 2.9. *NAND (NOT AND) (a), NOR (NOT OR) (b) and XNOR (NOT exclusive OR) (c) gates*

The NAND and NOR gates are considered to be universal gates. This means that any logic function can be implemented using just NAND or NOR gates. It must be noted that neither the XOR gate nor the XNOR gate is universal.

2.3. Three-state buffer

A three-state buffer works as a signal-controlled switch. An enable signal is used to control whether the input signal is transferred toward the output or isolated from the output, which is then held in a high-impedance state.

The output from the circuit shown in Figure 2.10 can take any of the following three states: high (1), low (0) and high impedance (*z*).

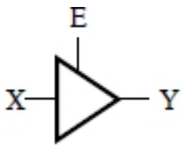

Figure 2.10. *Three-state buffer*

When E = 0, the output is held in the high impedance state.

When E = 1, the output