2 Introduction to Logic Gates

The basic building blocks for digital circuits are logic gates. Most logic gates are binary logic, i.e. have two states of 0 or 1. The input or output of these logic gates can only exist in one of these states, where a positive logic system treats 0 as FALSE value and 1 as TRUE value and conversely for the negative logic system. Figure 2.1 shows a logic waveform that is logic 1 between time *t*1 and *t*2 and is logic 0 at other times. Positive logic will be assumed throughout the book except where denoted otherwise.

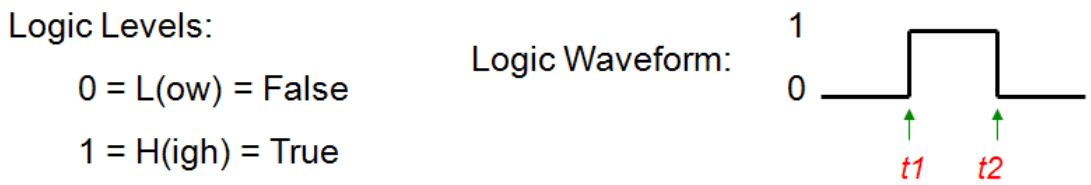


Figure 2.1: Positive logic waveform.

Figures 2.2 and 2.3 show the input and output voltage ranges for logic 0 and 1 for a common logic gate[2] used in digital devices. Transition region is the range where the voltage is not defined and hence, the input or output voltage from the device should not fall in this region as the logic value can be either 0 or 1. The output ranges are smaller as compared to input ranges, which is useful to reduce noise interference. The difference between the input-output ranges is known as noise margin. While it is usual to have a noise margin that is the same for both logical values, this does not have to be the case all the time.

To illustrate the usefulness of this noise margin, consider an example where there is noise interference in between two devices. Suppose the output voltage from the first digital device is 4.6 V (i.e. digital logic 1) and a spike (noise) of -0.5V enters as interference. The value of input voltage to the second device will be 4.1 V and the input digital level will still be 1. Without this noise margin, the digital level input to the second device will be unpredictable as it will fall within the transition region. The difference between input and output ranges for a given logic value is known as guaranteed noise immunity, which is 1 V in this case. It should also be obvious that the transition region for output voltage will be wider than for the input voltage because of this noise margin.

2 The gate is actually a CMOS type NAND gate. NAND gates will be discussed later in the chapter.

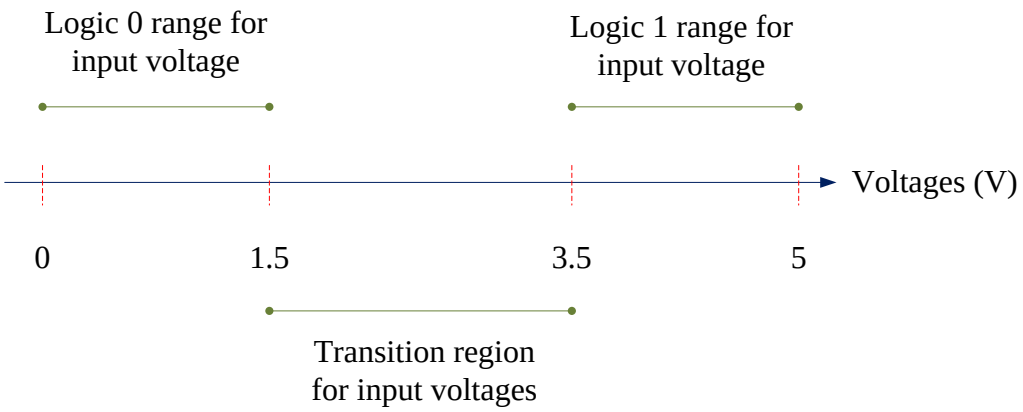


Figure 2.2: Input logic related to actual voltages.

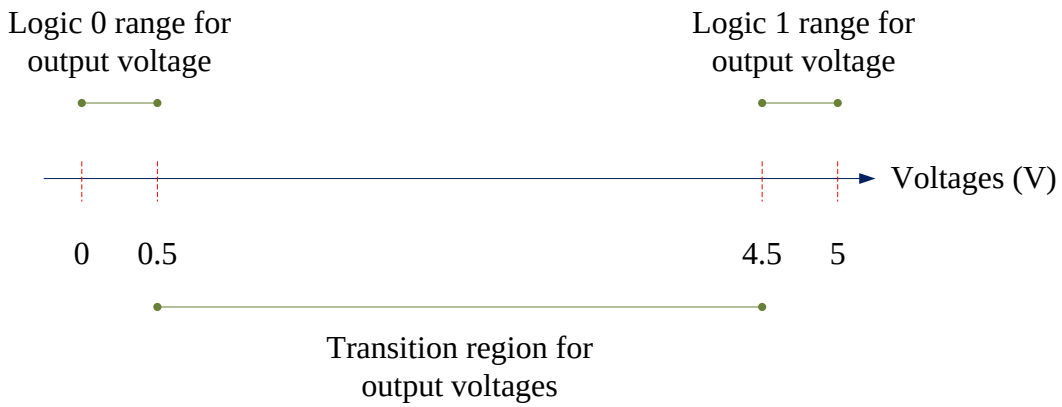


Figure 2.3: Output logic related to actual voltages.

Actual pulse waveform does not resemble the form shown in Figure 2.1, but is rather like the one shown in Figure 2.4[3] where there is a period of time required for the pulse to rise and fall and these are known as rise and fall times, respectively. The time taken for the pulse to rise from 10% to 90% of the amplitude is rise time while the fall time is the time taken for the amplitude value to drop to 10% from 90%. The actual rise and fall times for a digital device depends on its specifications; costly devices have smaller times. The pulse width is measured using 50% of the rise and fall amplitude values as shown in the figure.

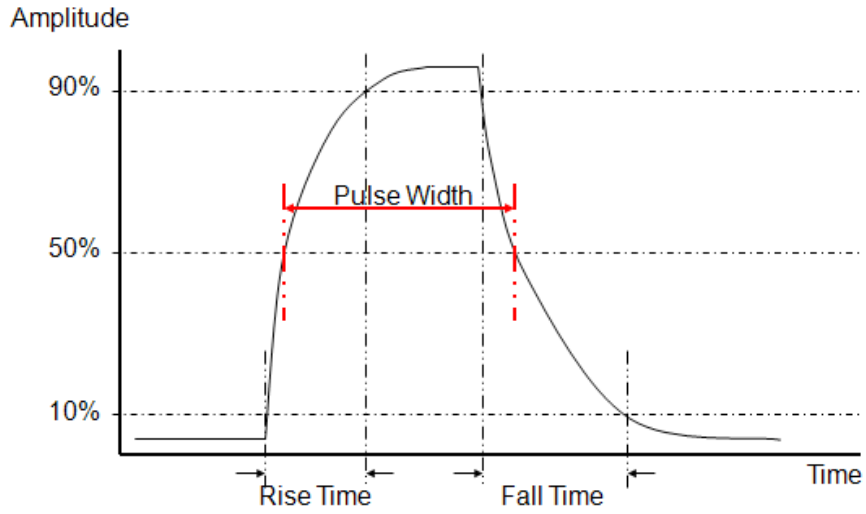


Figure 2.4: An example of actual pulse waveform.

3 Even this figure is simplified for ease of understanding. Actual waveform will have lots of spikes.

2.1 AND gate

Basically AND gate is composed of two inputs and a single output as shown in Figure 2.5 with algebraic representation[4] $F = A \cdot B$ or simply . $F = AB$ The traditional symbol shown in Figure 2.5(a) is more commonly employed in text books. However, the IEEE/ANSI symbol as shown in Figure 2.5(b) is gaining popularity and has the advantage of containing qualifying symbols inside the logic-symbol that describes the operation of the gate. The truth table that gives the output *F* for inputs *A* and *B* is given in Table 2.1. It can be seen that the output is LOW (FALSE) when any one of the inputs is LOW (FALSE) and the output is only HIGH (TRUE) when all the inputs are HIGH (TRUE).

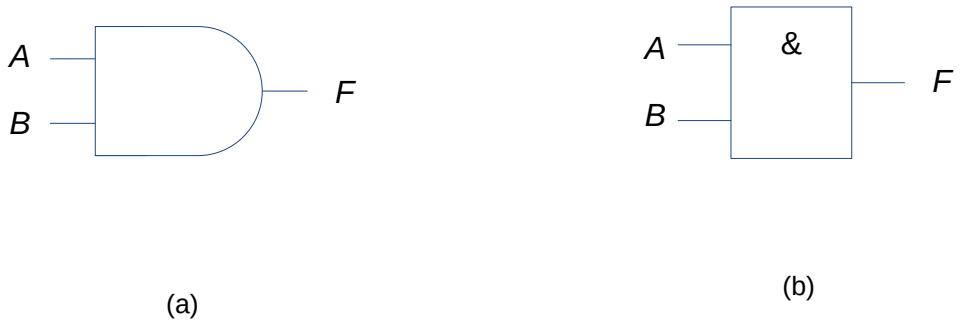


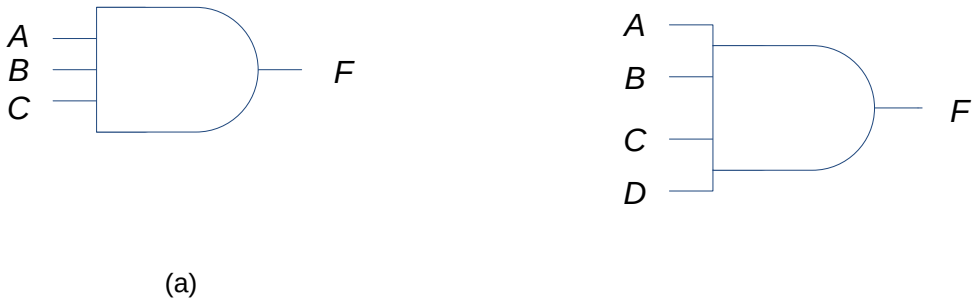
Figure 2.5: AND gate logic symbols (a) traditional (b) IEEE/ANSI standard.

Table 2.1: Truth table for two-input AND gate

A	*B*	*F*
0	0	0
0	1	0
1	0	0
1	1	1

AND gate inputs do not have to be limited to two; there can be any number of inputs greater than one as shown in Figure 2.6.

4 Also known as Boolean or logic expression.



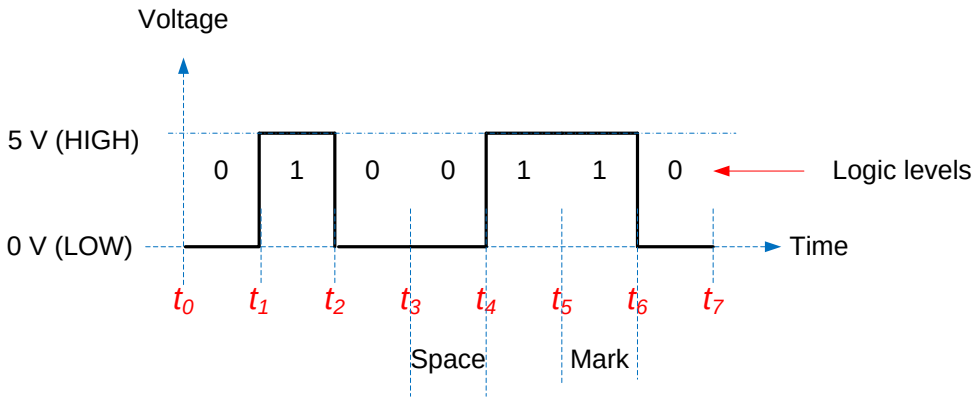
(a) (b)

Figure 2.6: Three and four input AND gates: (a) $F = A \cdot B \cdot C$ (b) $F = A \cdot B \cdot C \cdot D$.

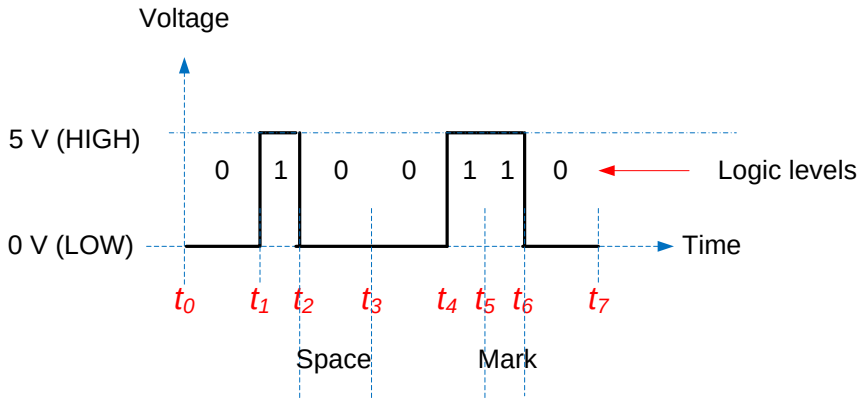
2.1.1 Timing diagram

Timing diagram is useful in describing the relationship between the inputs and output of a logic gate. The inputs of a digital logic gate can be shown diagrammatically as a waveform that represents the changing values over time. A waveform corresponding to the changing values of the inputs over time will be generated at the output of the logic gate. Figure 2.7 show examples of timing diagram waveform for equal and unequal mark-space cycles. The mark represents the time for logic level HIGH, while the space represents the time for logic level LOW. Equal mark-space requires periodic clock pulse[5]. All the discussion in this book will be using equal mark-space timing waveforms only.

5 Clock pulses will be discussed in later chapters.



(a)



(b)

Figure 2.7: Example of timing diagram waveforms: (a) equal mark-space (b) unequal mark-space.

2.1.2 Timing diagram example for AND gate

Figure 2.8 shows an example of a timing diagram for a two-input AND gate. At each time block, the inputs *A* and *B* affect the output *F*. For example, in time block t_0 to t_1, both inputs are LOW, so the output is also LOW. Similarly, the entire timing waveform for the output can be obtained using AND operation of inputs in each time block.

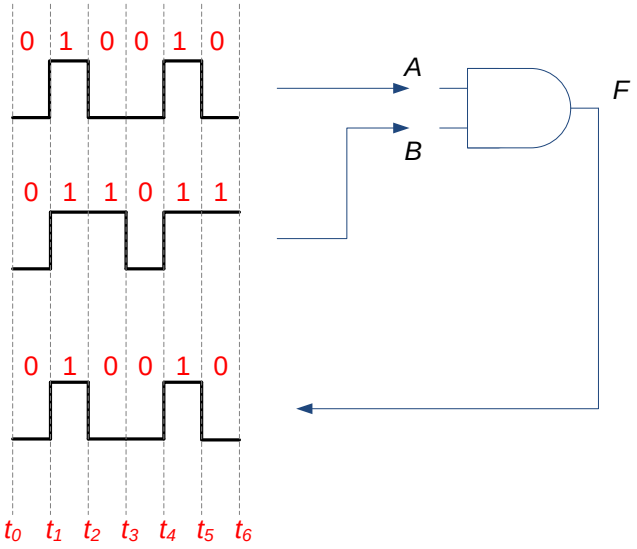


Figure 2.8: Timing diagram waveform for a two-input AND gate.

2.2 OR gate

OR gate as shown in Figure 2.9 has algebraic representation, $F = A + B$. The truth table that gives the output F for inputs A and B is given in Table 2.2. It can be seen that the output is HIGH when any one of the inputs is HIGH and the output is only LOW when all the inputs are LOW.

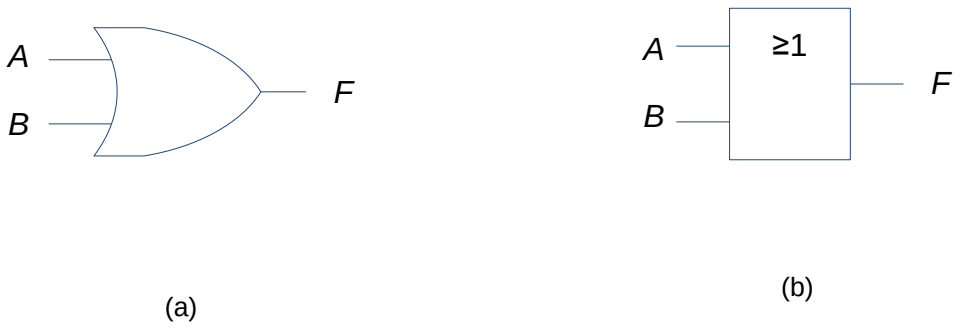


Figure 2.9: OR gate logic symbols: (a) traditional (b) IEEE/ANSI standard.

Table 2.2: Truth table for two-input OR gate

A	*B*	*F*
0	0	0
0	1	1
1	0	1
1	1	1

Similar to AND gate, there can be any number of inputs greater than one as shown in Figure 2.10.

Figure 2.10: Three and four input OR gates: (a) $Y = A + B + C$ (b) $Y = A + B + C + D$.

2.2.1 Timing diagram example for OR gate

Figure 2.11 shows an example of a timing diagram for a two-input OR gate. At each time block, the inputs *A* and *B* affect the output *F*. For example, in time block t_5 to t_6, one input is HIGH, so the output is HIGH. Similarly, the entire timing waveform for the output can be obtained using OR operation of inputs in each time block.

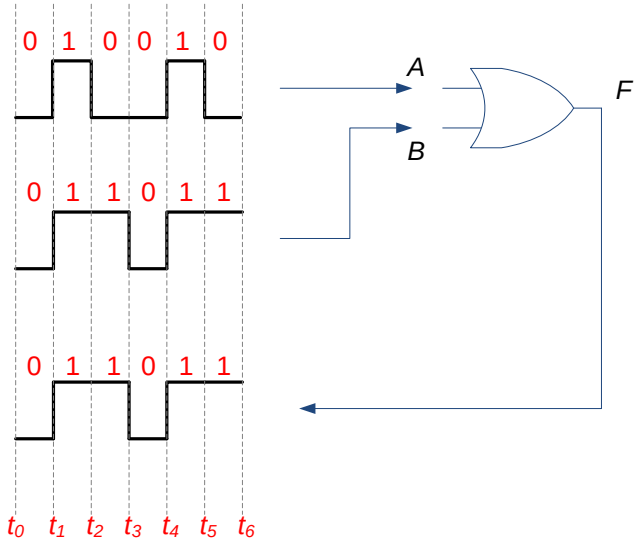


Figure 2.11: Timing diagram waveform for a two-input OR gate.

2.3 NOT gate

NOT gate is also known as INVERTER as it inverts (complements) the input logic level. It is shown in Figure 2.12 and has only one input and one output with algebraic representation of $F = \overline{A}$ or $F = A'$. The bubble in the symbol denotes inversion (without it, the symbol will represent a buffer gate that does not alter the logic level; in IEEE/ANSI standard, the bubble is replaced by a triangle). The truth table for NOT gate is given in Table 2.3.

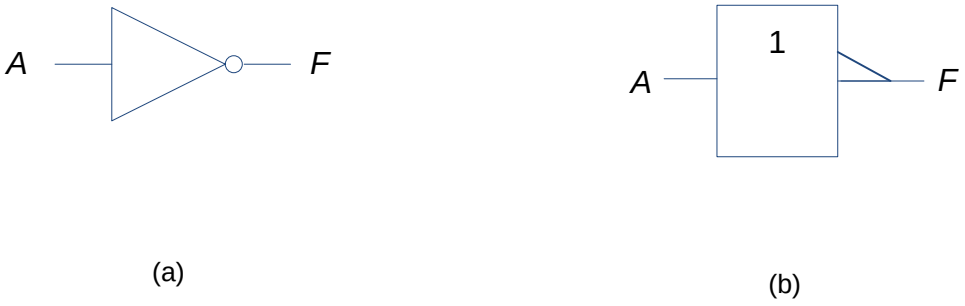


Figure 2.12: NOT gate logic symbols: (a) traditional (b) IEEE/ANSI standard.

Table 2.3: Truth table for NOT gate

A	*F*
0	1
1	0

NOT gate can also be connected in cascade and a few examples are shown in Figure 2.13. It should be obvious that odd number of NOT gate connections give output logic level that is complement to the input logic level and an even number of NOT gates connections give output logic level that is the same as the input logic level.

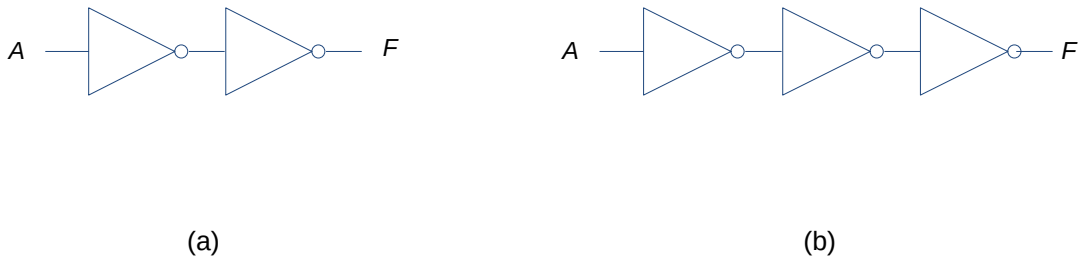


Figure 2.13: Cascade connection of NOT gates: (a) $F = \overline{\overline{A}} = A$ (b) $F = \overline{\overline{\overline{A}}} = \overline{A}$.

2.4 AND implementation with OR gate and vice versa

It is useful to know that AND gate logic can be easily implemented using OR gate and vice versa through a simple process using additional NOT gates. For example, an AND gate equivalent can be constructed with an OR gate with both the inputs and outputs inverted through NOT gates. Figure 2.14 shows an example with equivalent truth table in Table 2.4. This is actually DeMorgan's first theorem, which will be discussed in detail in Chapter Three. It is mentioned here so that the reader is aware that it is possible to implement one gate logic with another gate(s).

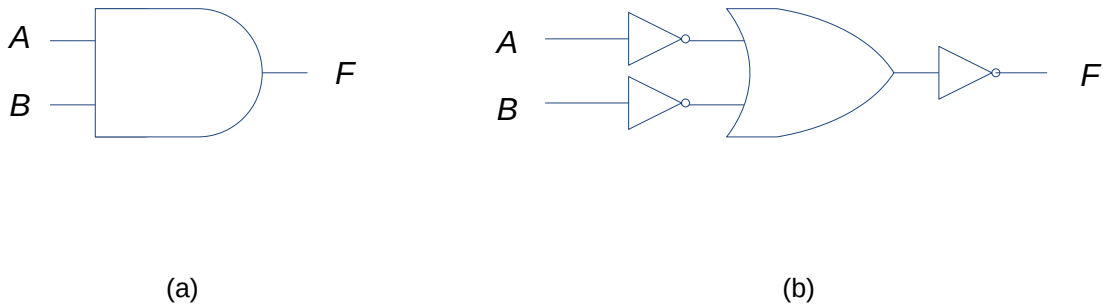


Figure 2.14: AND gate implementation with OR gate: (a) $F = AB$ (b) $F = \overline{\overline{A} + \overline{B}} = AB$.

Table 2.4: Truth table illustrating AND gate implementation using OR and NOT gates

A	B	$F = AB$	$\overline{A}$	$\overline{B}$	$F = \overline{A} + \overline{B}$	$F = \overline{\overline{A} + \overline{B}}$
0	0	0	1	1	1	0
0	1	0	1	0	1	0
1	0	0	0	1	1	0
1	1	1	0	0	0	1

2.5 NAND gate

NAND and NOR gates that will be discussed in the following section are known as universal gates as combinations of these gates are sufficient to obtain equivalent operation of OR, AND or NOT gates. However, this is different to the implementation discussed in Section 2.4 as either NAND or NOR gates on their own will be sufficient to implement logic function of any of the other gates. NAND gate logic symbol is shown in Figure 2.15 (note the addition of a bubble when compared to AND gate) and its truth table is shown in Table 2.5. A NAND gate operation can also be obtained through cascade operation of AND and NOT gates as shown in Figure 2.16. Algebraically, the operation can be defined as. $F = \overline{AB}$.

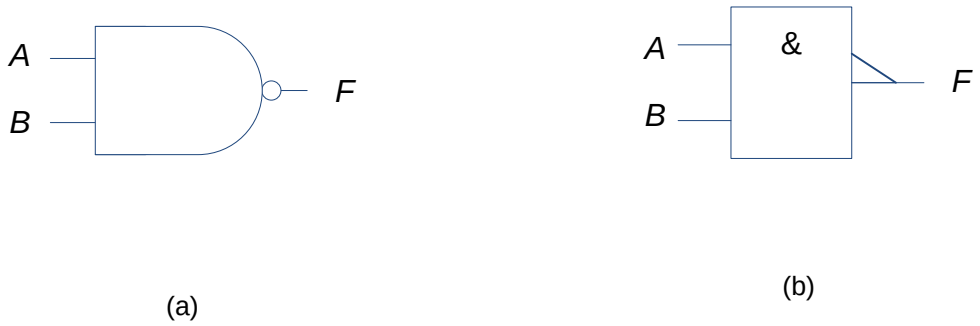


Figure 2.15: NAND gate logic symbols: (a) traditional (b) IEEE/ANSI standard.

Table 2.5: Truth table for NAND gate

A	*B*	*AB*	*F*
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

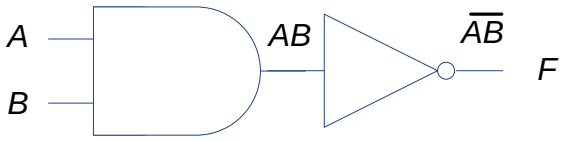


Figure 2.16: NAND gate logic using AND and NOT gates.

Figure 2.17 shows an example for implementing an AND gate using NAND gates only. The blue shaded tiny bubble represents branch-off of the signal and should not be confused with the *empty* bubble that is used to represent inversion operation. Similarly, other gates such as OR and NOT can be implemented using NAND gates and these are left as exercises for the reader.

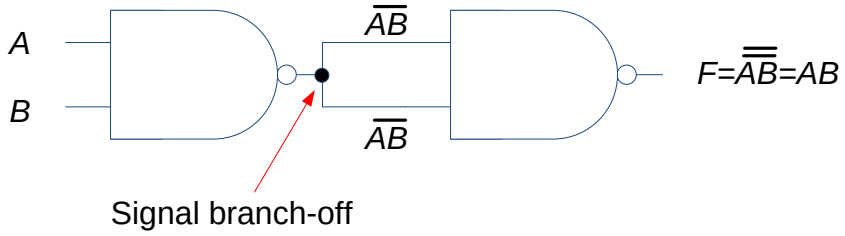


Figure 2.17: AND gate implementation using two NAND gates.

2.6 NOR gate

NOR gate is basically an OR gate with the output inverted. Figure 2.18 shows the logic symbol with truth table shown in Table 2.6. Algebraically, the operation can be defined as $F = \overline{A + B}$. Similar to NAND gate, several NOR gates can be used to implement AND, OR or NOT gates. An example of this is shown in Figure 2.19 and the reader can easily verify through the use of truth tables that $F = AB$.

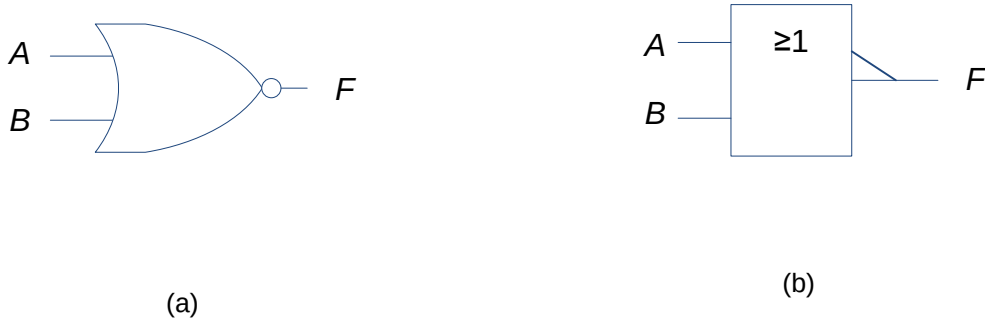


Figure 2.18: NOR gate logic symbols: (a) traditional (b) IEEE/ANSI standard.

Table 2.6: Truth table for NOR gate

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

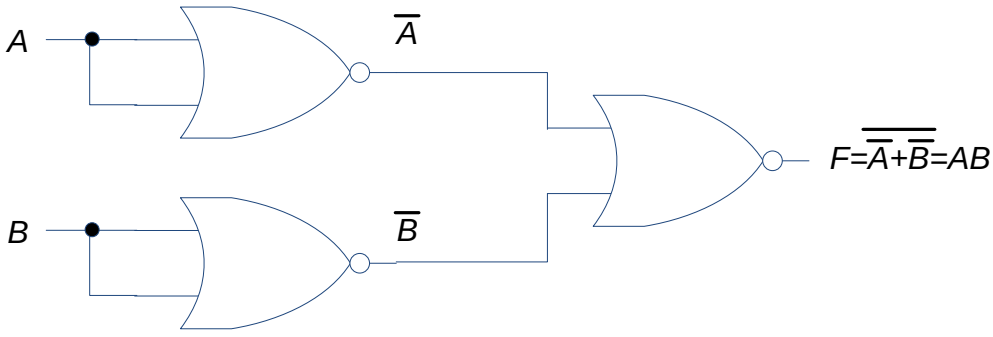


Figure 2.19: AND gate logic implementation using NOR gates.

2.7 Integrated circuits

All the gates that we have discussed in this chapter are manufactured as integrated circuit (IC) with several gates in one IC. For example, 74LS00 is a transistor-transistor logic (TTL) technology based IC that has four (quad) two-input NAND gates. Complementary Metal-Oxide Semiconductor (CMOS) is another technology that is widely used for manufacturing IC but TTL devices are more commonly employed for laboratory experiments as they are more robust to electrostatic noise. Figure 2.20 shows the pin configuration of 74LS00 and Figure 2.21 shows an example of pin configurations to implement NOT operation. Pin 14 is connected to the power supply while pin 7 is the ground pin. It should be obvious that the LED will only light-up (i.e. the output will be HIGH) if switch *A* is turned OFF (i.e. made to logic level LOW) – similar to the input and output values as in the truth table shown in Table 2.3.

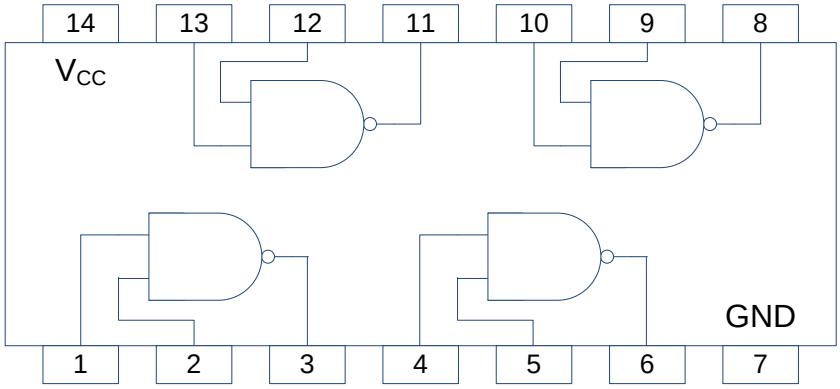


Figure 2.20: 74LS00 - Quad NAND IC.

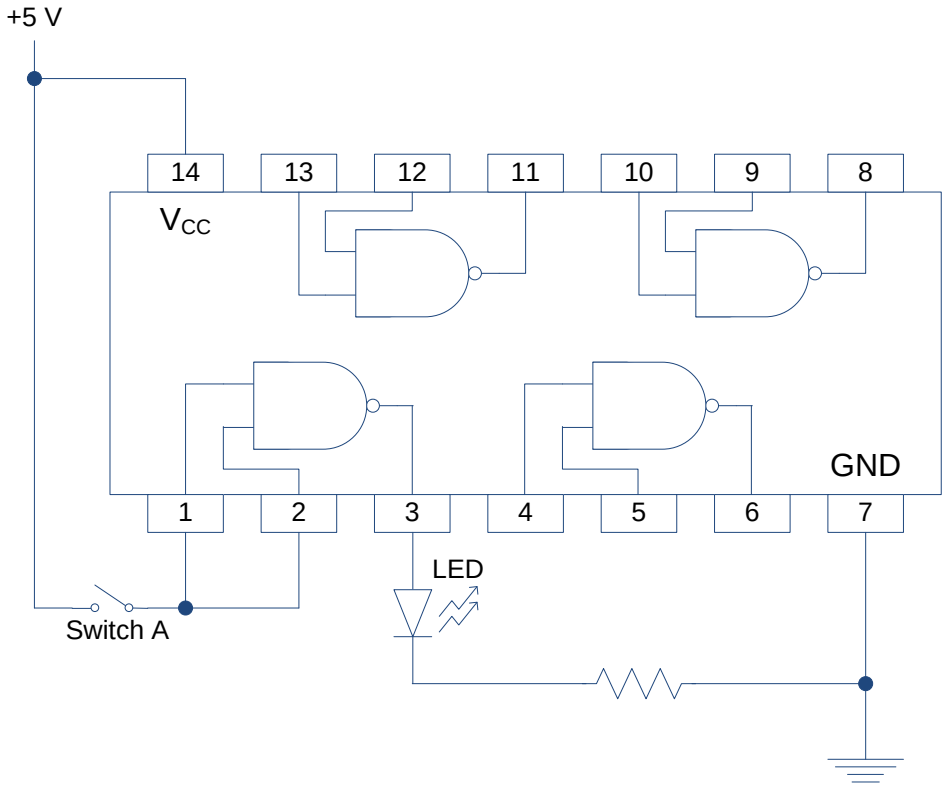


Figure 2.21: NOT gate implementation example using 74LS00.