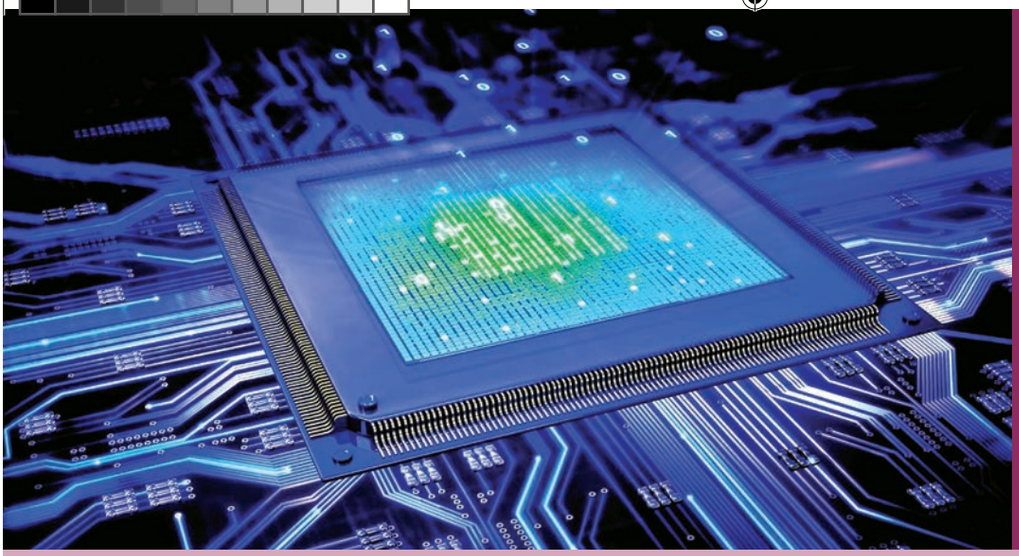


CHAPTER 1

Digital Circuit Applications

LEARNING OBJECTIVE

Through this chapter the students can learn about the following:

- Construction of Combinational Gates and its applications
- Classification of Logic gates – Arithmetic circuits like,Adder and Subtractor
- How the Digital signals are Decoded and Encoded
- The way of Multiplexing and De-Multiplexing
- Construction and working Flip-flops(Memory)
- Construction of Binary Counters and Registers and its Applications

CONTENT

Introduction

Though you are little familiar with digital circuits, i.e., particularly the basic gates (AND, OR, NOT), the utility of these circuits in constructing many discrete circuits for instrumentation application are discussed in detail in this Chapter. Further, we are going to discuss about the combinational gates.

1.1 Application of Basic Gates

NOT – Gate Application

By using the inverter (NOT) gate, we can get the 1's complement value of any given number because the output of NOT- gate is complement of the input. Figure 1.1 illustrates an example of 1's Complement digital circuit.

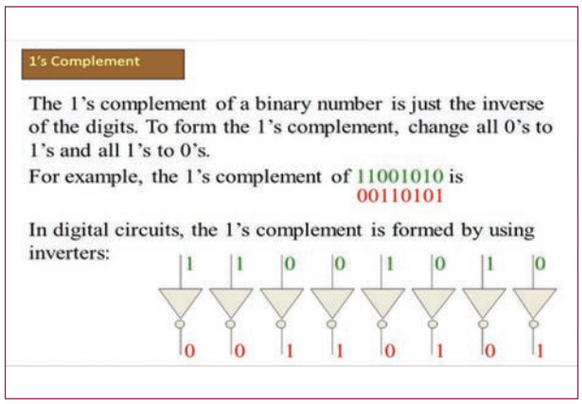


FIGURE 1.1 Example of 1's Complement circuit using inverters

AND – Gate Application

By using AND gate very simple but important application can be executed. An AND gate is used in a simple automobile seat alarm system to detect, when ignition switch is ON and the seat belt is unbuckled. If the ignition switch is ON, a HIGH is produced on input A of the AND gate. If the seat belt is not properly locked, HIGH is produced in input B of the AND gate. Also, when the ignition switch is turned on, a timer is started that produces a HIGH on input C for 30 seconds. If all three conditions exist- that is, if the ignition is ON and that seat belt is unbuckled and the timer is running - the output of the AND gate is HIGH and an audible alarm is energised to remind the driver. Figure 1.2 shows a simple car-seat belt alarm circuit using AND gate.

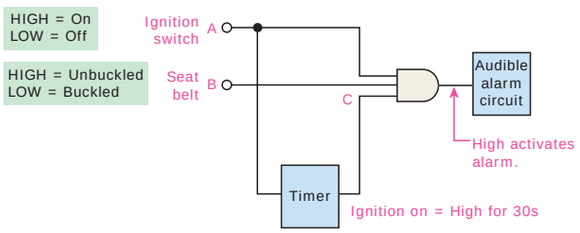


FIGURE 1.2 A simple Car-seat belt alarm circuit using AND Gate

OR- Application

A simplified portion of an intrusion detection and alarm system is shown in Figure 1.3. This system can be used for one room in a house which has two windows and one door (3 input OR gate is used). The sensors are magnetic switches that produce a HIGH output, when open and a Low output when closed. As along as the windows and doors are secured, the switches are closed and all three of the OR gate inputs are low. When one of the windows or door is opened a HIGH is produced on that input to the OR gate and the gate output goes HIGH. It then activates and latches an alarm circuit to warn about the intrusion.

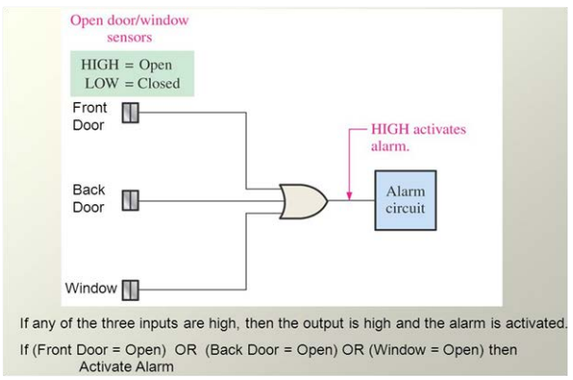


FIGURE 1.3 Simple intrusion detection alarm system using OR-gate

This shows a simple basic gate can be employed in few of the very common and important applications in day today life.

1.2 Combinational Gates

Now, let us see some of the other gates constructed by using these basic gates, which are called as combinational gates. The following are some of the important gates.

1. NAND
2. NOR
3. EX-OR
4. EX-NOR

NAND gate

The term NAND is derived from NOT-AND gates. It is nothing but complemented output of AND gate. The standard logic symbol for 2-input NAND gate is shown Figure 1.4.

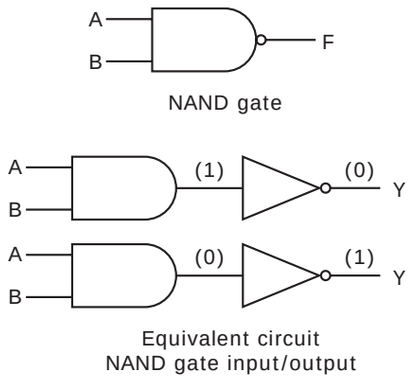


FIGURE 1.4 Symbol and Equivalent circuit of 2-input NAND gate

Operation of a NAND – gate

A NAND gate produces low output when all the inputs are high. When any of the input is low, the output will be HIGH. Figure 1.4 shows a 2-input NAND gate with the inputs labelled as A and B and the output is labelled as Y. The operation can be stated as follows.

For a 2-input NAND gates, the output Y is low only when, inputs A and B are HIGH. Output 'Y' is HIGH when either A or B is low, or when both inputs A and B are low.

The operation of a NAND gate is opposite to that of a AND gate. In a NAND gate, low-level (0) is the active output level, as indicated by the bubble on the output. Table 1.1 shows the logical operation of the 2 input NAND gate. The logic expression for two input NAND gate is $Y = \overline{A \cdot B}$.

TABLE 1.1 Truth table of NAND gate

A	B	$Y = \overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

The operation of NAND gate can also be explained with waveform. Let us look at the pulse waveform operation of a NAND gate. With reference to the truth table the output of NAND gate is low, when all of the inputs are HIGH.

EXAMPLE 1.1

If the two waveforms A and B shown in Figure 1.5 are applied to the NAND gate inputs, determine the resulting output waveform. Bubble indicates an active-LOW output.

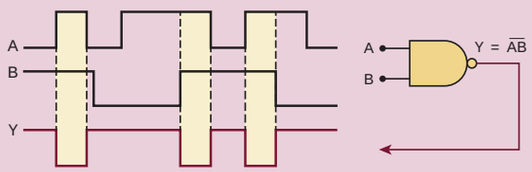


FIGURE 1.5 Waveforms of 2-input NAND-gate

Solution: Output waveform Y is LOW only during the four time intervals, when both input waveforms A and B are HIGH as shown in the timing diagram (Figure 1.5).

Activity

Determine the output waveform and show the timing diagram if input waveform B is inverted.

Application Sample (NAND gate)

In a house there are two water tanks. Each tank has a sensor that detects when the water level drops to 25% of full level. The sensor produces a HIGH level of 5 V when the tanks are more than one quarter-full. When the volume of water in a tank drops to one quarter-full, the sensor senses and gives an output Low Level (0V).

Solution: Figure 1.6 shows a NAND gate with its two inputs connected to the tank level sensor and its output connected to the indicator panel. The operation can be stated as follows: if the tank A and B are above the one–quarter full, the LED is ON.

As long as both the sensors output are HIGH (5 V), indicating that both tanks are more than one-quarter full, the NAND gate output is low (0 V), the green LED circuit is arranged so that low voltage turns it on. The resistor limits the LED current.

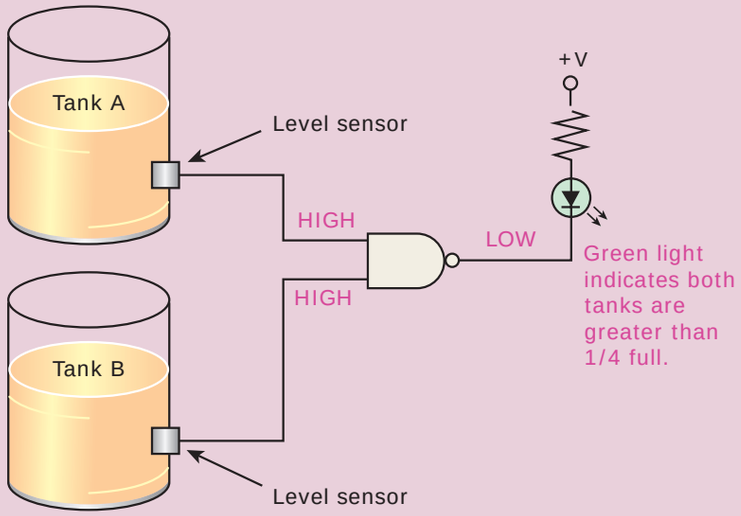


FIGURE 1.6: NAND gate with its two inputs connected to the tank level sensor

NOR - gate

The NOR–gate is derived from the combination of NOT-OR gate. It is nothing but complemented output of OR gate. The standard logic symbols for 2-input NAND gate is shown in Figure 1.7.

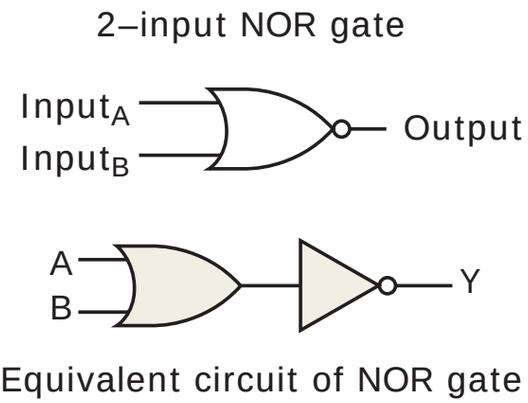


FIGURE 1.7 Symbol and Equivalent Circuit of NOR gate

Operation of a NOR-gate

A NOR-gate produces a LOW output when any of its input is HIGH. The output will be HIGH when only all of the inputs are LOW. Figure 1.7 shows the NOR gate labelled A and B are inputs and Y as the output. The operations can be stated as follows.

For a 2-input NOR gate, output Y is LOW, when either input A or input B is HIGH or the output Y is HIGH only when both inputs are LOW.

The operation of NOR gate is opposite to that of OR gate. In a NOR gate, the low output is the active output level as indicated by the bubble on the output. Table 1.2 shows the logical operation of the 2-input NOR gate. The logic expression for two input NOR gate is $Y = \overline{A + B}$.

TABLE 1.2 Truth Table of NOR gate

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

The operation of NOR- gate can also be explained with waveforms. Let we look at the pulse waveform operation of a NOR gate (Figure 1.8) with reference to the truth table the output of NOR-gate is HIGH, only when both the inputs are LOW.

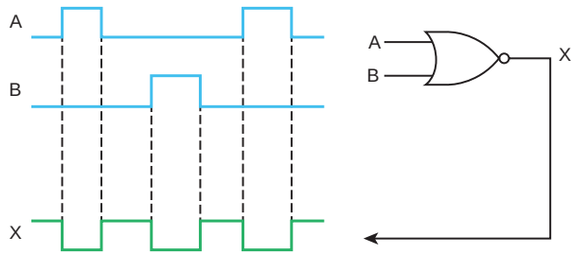


FIGURE 1.8 Waveforms of NOR gate

If the two waveforms shown in Figure 1.8 are applied to a NOR gate, what is the resulting output waveform?

Solution

Whenever any input of the NOR gate is HIGH, the output is LOW as shown by the output waveform Y in the timing diagram.

Activity

Invert input B and determine the output waveform in relation to the inputs.

Universal gate

NAND and NOR gates are termed as universal gates. Because by using these gates (either NAND or NOR), we can derive the operations of any other gates function.

Application Sample (NOR- gate)

Figure 1.9 shows how different functions of an aircraft are combined together to get information monitoring an aircraft.

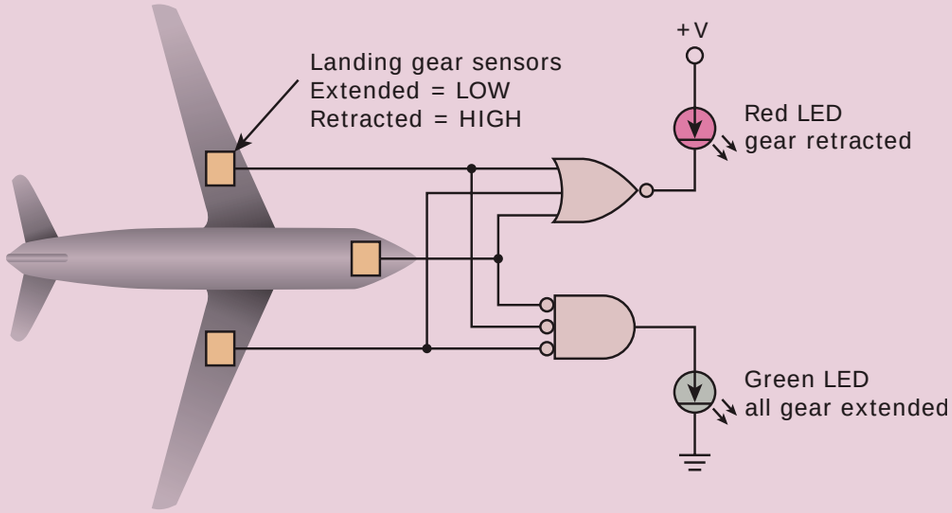


FIGURE 1.9 Aircraft functional monitoring system

Exclusive-OR and Exclusive NOR-gate

Exclusive-OR and Exclusive-NOR gate are formed by a combination of the earlier gates that we discussed so far. These gates are used in many fundamental applications and treated as basic logic elements. Standard symbol for Exclusive OR (XOR as short) gate is shown in Figure 1.10.

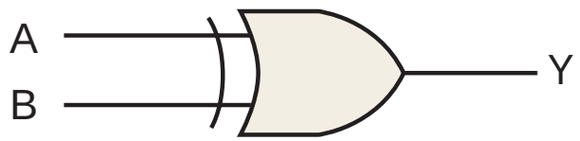


FIGURE 1.10 Symbol of Ex-OR gate

The XOR gate has only two inputs. The output of XOR gate is HIGH, only when the inputs of the gates are in opposite logic levels. The output of the gate is low when the inputs are identical i.e., both are LOW or HIGH.

> For an XOR gate the output Y is HIGH only when inputs A is low and B is HIGH and vice versa. It will be LOW on other conditions. The unique characteristic of XOR gate is that it produces HIGH output only when an odd number of HIGH inputs are present.

Table 1.3 shows the logical operation of two-input XOR-gate.

TABLE 1.3 Truth table of XOR gate

A	B	$Y = (\overline{A} \cdot B) + (A \cdot \overline{B})$
0	0	0
0	1	1
1	0	1
1	1	0

The logical expression for XOR-gate $Y = (\overline{A} \cdot B) + (A \cdot \overline{B})$. This can be often shortened and given as $Y = A \oplus B$. It is also called as "Inequality Comparator".

Operation with Waveform Inputs

As we have done with the other gates, let us examine the operation of XOR and XNOR gates with pulse waveform inputs. As before, we apply the truth table operation during each distinct time interval of the pulse waveform inputs as illustrated in Figure 1.11(a) for an XOR gate. We can see that the input waveforms A and B are at opposite levels during time intervals t_2 and t_4. Therefore, the output Y is HIGH during these two times. Since both inputs are at the same level, either both HIGH and both LOW, during time intervals t_1 and t_3, the output is LOW as shown in the timing diagram.

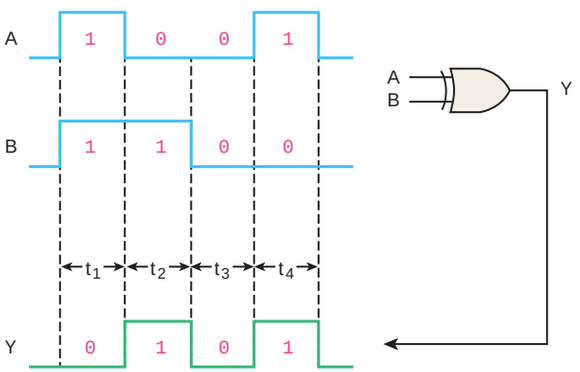


FIGURE 1.11(a) Exclusive-OR gate Operation with pulse waveform inputs

Application Sample (EX- OR gate)

A certain system contains two identical circuits operating in parallel. As long as both are operating properly, the outputs of the circuits are always the same. If one of the circuits fails, the outputs will be at opposite level at some time. Derive a way to detect that a failure occurred in one of the circuits.

Solution

The outputs of the circuits are connected to the inputs of an XOR gate as shown in Figure 1.11(b). A failure in either one of the circuits produces differing outputs,

which cause the XOR inputs to be at opposite levels. This condition produces a HIGH on the output of the XOR gate, indicating a failure in one of the circuits.

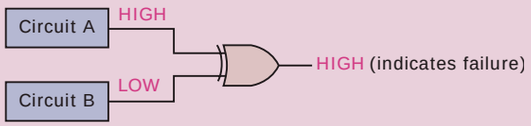


FIGURE 1.11(b) Application of XOR gate

Exclusive–NOR GATE

The standard symbol for Exclusive–NOR (X-NOR) gate is shown in Figure 1.12. The bubble on the output of the X-NOR symbol indicates that its output is opposite to that of the XOR gate, i.e., the output is complemented XOR gate.

For an exclusive-NOR gate (X-NOR), output Y is low when input A is LOW and input B is HIGH, or when A is HIGH and B is LOW. Y is HIGH only when both A and B are HIGH or both LOW.

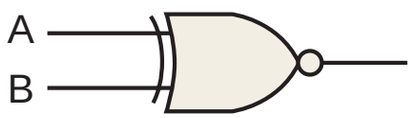


FIGURE 1.12 Symbol of X-NOR

Table 1.4 shows the logical operation of a two-input X-NOR gate. The logical expression for X-NOR gate is $Y = (\overline{A}.B) + (A.\overline{B})$. This can be often standard and given as $Y = \overline{A \oplus B}$.

TABLE 1.4 Truth table of Ex-NOR gate

A	B	$Y = \overline{A \oplus B}$
0	0	1
0	1	0
1	0	0
1	1	1

Application-Sample

EX-NOR gate

Observe the output waveforms for the XOR and XNOR gate. Determine the output waveforms for the XOR gate and for the X-NOR gate, given the input waveforms, A and B, as shown in Figure 1.13.

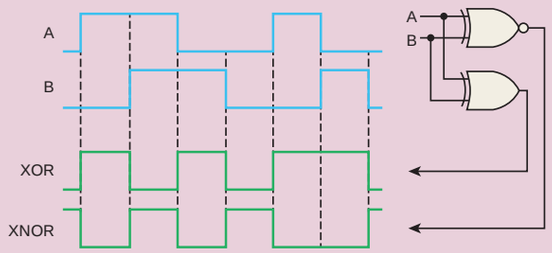


FIGURE 1.13 Output waveforms of Ex-OR and EX-NOR Gates

Solution

The output waveforms are shown is the Figure 1.13. We can observe that the XOR output is HIGH only when both inputs are at opposite levels. Note that the XNOR output is HIGH only when both inputs are the same. Therefore it is termed as equality comparator. Yet another simple example of XOR gate is adder, which is used to add two bits.

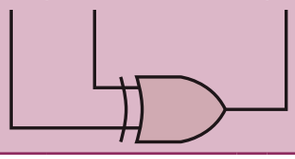
An Application

An Exclusive-OR gate can be used as a two-bit modulo-2 adder. We know the basic rules for binary addition are as follows: 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 10. An examination of the truth table for an XOR gate shows that its output is the binary sum of the two input bits. In the case where the inputs are both 1s, the output is the sum 0, but you lose the carry of 1. We will see how XOR gates are combined to make complete adding circuits. Table 1.5 illustrates an XOR gate used as a modulo-2 adder.

TABLE1.5 Application of XOR as Adder

An XOR gate used to add two bits

A	B	Σ
0	0	1
0	1	0
1	0	0
1	1	1



Though NAND and NOR gates are called as universal gates, among NAND gate is more versatile. So far, we have learned seven types of gate circuits consisting AND, OR, NOT, NAND, ENOR, XOR and EX-NOR. We can buy IC's that perform any of these seven basic functions. But, in the market NAND gate is the more widely available IC.

Activity

Write the Boolean expression for a three input NAND gate.

1.3 Boolean algebra

Though you have studied three important basic Boolean operations Addition (OR), multiplication (AND), complementation or inversion (NOT). Other than these, there are three important basic laws as like in mathematical algebra. They are,

1. Commutative law
2. Associative law
3. Distributive law

The Boolean is used to simplify the gate (digital) circuits

Boolean algebra was derived by George Boole in 1854

1. Commutative law

The law by addition and multiplication say that the order in which variable are OR-ed (or) AND-ed makes no different as the sum assured is arrived at either way. These laws of addition and multiplication for two variables are written algebraically as follows.

Commutative law of addition of two variables

$$A+B = B+A$$

Commutative law of multiplication for two variables

$$A \cdot B = B \cdot A$$

Figure 1.14 and Figure 1.15 illustrate the commutative law applied to the OR gate and the AND gate.

FIGURE 1.14 Commutative law using OR gate

FIGURE 1.15 Commutative law using AND gate

2. Associative law

These law of addition and multiplication say that in the ORing or ANDing of several variables (more than two), grouping of the variables is immaterial and the addition results obtained are the same. These laws of addition and multiplication for three variables are written algebraically as follows. Figure 1.16 and Figure 1.17 illustrate the associative law as applied to OR and AND gates.

Associative law of addition of three variables

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

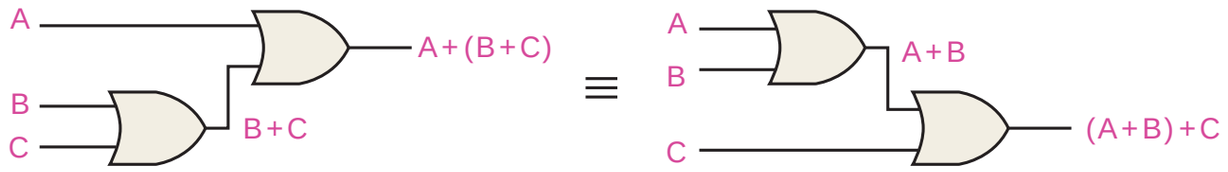


FIGURE 1.16 Associate law of three variables using OR gates

Associative law of multiplication of three variables $A \cdot (B \cdot C) = (A \cdot B) \cdot C$

FIGURE 1.17 Associate law of three variables using AND gates

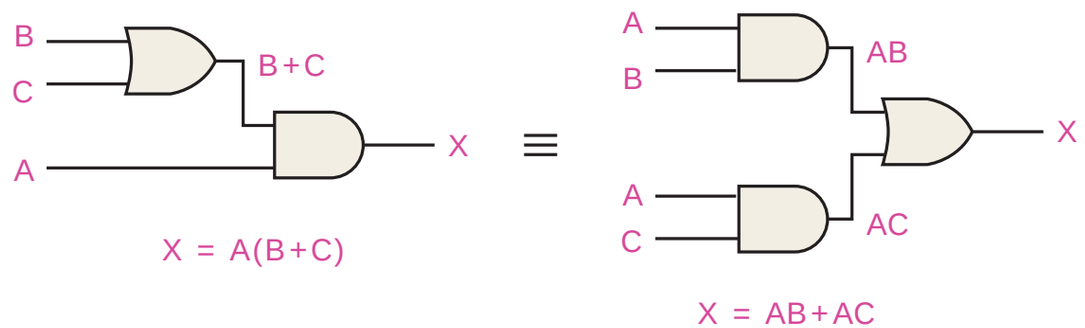


FIGURE 1.18 Distributive law using AND and OR gates

3. Distributive law

This law states that ORing several variables and ANDing the result with the single variable is equivalent to ANDing the single variable with each of several variables and the ORing the products. The law is algebraically written as follows.

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

Figure 1.18 shows gate implementation of distributed law.

Boolean Algebra Rules

Though we discussed this in 11th Standard, it is better to recapture the basic rules that are useful in manipulation and simplification of Boolean algebra expressions and are calculated using Table 1.6.

TABLE 1.6 Boolean Algebra Rules

1. $A + 0 = A$	7. $A \cdot A = A$
2. $A + 1 = 1$	8. $A \cdot \overline{A} = 0$
3. $A \cdot 0 = 0$	9. $\overline{\overline{A}} = A$
4. $A \cdot 1 = A$	10. $A + AB = A$
5. $A + A = A$	11. $A + \overline{A}B = A + B$
6. $A + \overline{A} = 1$	12. $(A+B)(A+C) = A + BC$

1.4 Classification of Logic circuit

Logic circuit may be classified into two broad categories:

1. Combinational logic circuits
2. Sequential logic circuits.

A combinational logic circuit contains logic gates only but does not contain storage elements. Sequential logic circuit contains storage elements in addition to logic gates. When logic gates are connected together to provide a specified output for certain specified combination of input variables without any storage, the resulting network is known as combinational logic circuit. The block diagram of logic combinational circuit is shown in Figure 1.19.

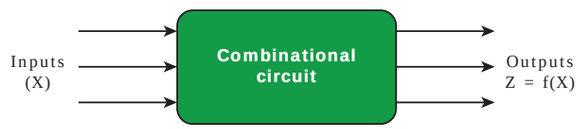


FIGURE 1.19 Combinational Logic Circuit

Sequential Logic Circuit

It accepts input binary variables and generates output variables depending on the logical combination of logic gates. The combinational logic circuits with memory element is called as sequential logic circuit, which is shown in Figure 1.20.

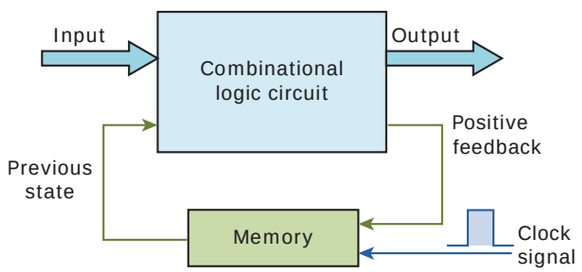


FIGURE 1.20 Block Diagram of Sequential Logic Circuit

A combinational circuit connected with feedback path termed as memory elements. The memory elements are device, capable of storing binary information within them.

Arithmetic Circuits

One of the essential functions of most computers and calculators is the performance of manipulating the arithmetic operations. The logic gates discussed so for can be used to perform arithmetic operations such as addition, subtraction, multiplication and division, which is used in electronic calculators and digital instruments. Since these circuits are electronic, they are very fast. Performing an addition takes less than 1 μs.

Now we will discuss some of the arithmetic operating circuits such as Half–adder, full-adder, parallel binary adder, half-subtractor and full-subtractor. The logic functions that are commonly used are OR, AND, and EX-OR gates.

Half-Adder

A logic circuits used for the addition of two single bit numbers is referred as a Half-Adder. When we add two binary numbers, we start with the least significant column. This means that we have to add two bits with the possibility of a carry. The circuit of a half-adder is shown Figure 1.21(a). Note in the Figure that the output sum is denoted by the mathematical symbol **Σ**.

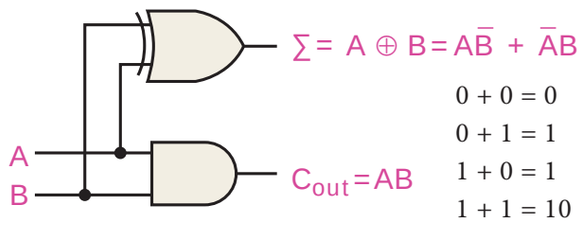


FIGURE 1.21(a) Half-Adder

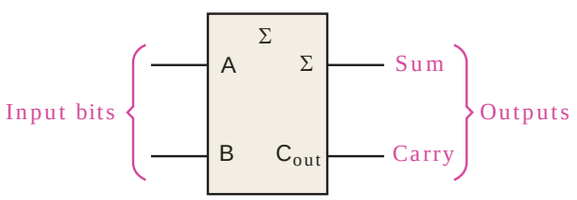


FIGURE 1.21(b) Logic symbol for a half-adder

It consists of an EX-OR gate and an AND gate. The output of an EX-OR gate is called SUM, while the output of the AND gate is called as CARRY. As the AND gate generates a high output only, when both inputs are high i.e., the carry as 1.

When both inputs of EX-OR gate is high or low the output i.e., the sum is low (0). When either of the input is high the output is high. Thus its the bianary addition. The logic symbol of Half-adder is shown in Figure 1.21(b). Truth table for a half adder is given in the Table 1.7.

TABLE 1.7 Truth-Table of Half-Adder

A	B	Σ	C_{out}
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Σ = sum
C_{out} = output carry
A and B = input variables (operands)

From the truth table a half-adder, the logical equations for CARRY and SUM can be written as,

CARRY $C = A \cdot B$

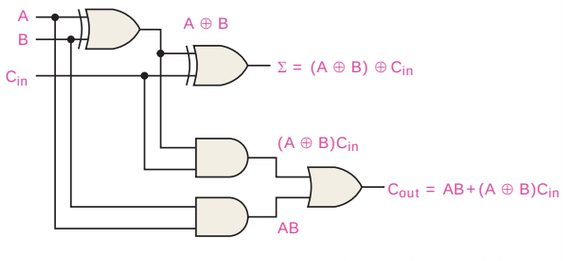
SUM $S = \overline{A} \cdot B + A \cdot \overline{B} = A \oplus B$

This Circuit is called as half-adder, because it cannot accept a CARRY-IN from previous additions. This is the reason that half-adder circuits can be used for binary additions of lower cost bit only. For higher order columns, we use 3-input adder called full-adder.

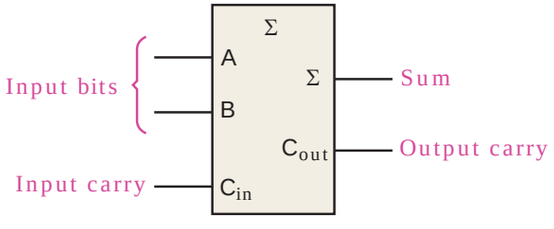
Full Adder

Full adder circuit is nothing but two half–adder circuits connected to an OR gate. As we seen in half -adder circuit, it has only two inputs and there is no provision to add CARRY coming from the lower-bit order when multi-bit addition is performed. For this purpose, we use a logic circuit, which can add three bits.

The third-bit is the CARRY from a lower column. This shows that we used a logic circuit with 3-inputs and 2-outputs. Such a circuit is called full-adder. Hence, full adder may be defined as logic circuits that add 3-bits, i.e., two bits to be added and CARRY-bit from lower-bit order, which results in SUM and CARRY. Figures 1.22(a) and (b) show the logic circuit and logic symbol of full-adder circuit, respectively. It has two inputs called A and B plus a third input (C_{IN}), called the CARRY IN and two outputs SUM and CARRY OUT(C_{OUT}).



(a) Complete logic circuit for a full-adder



(b) Logic symbol of full-adder

(each half-adder is enclosed by a shaded area)

FIGURE 1.22 A simple circuit of full adder

Truth table of full-adder for all possible inputs/outputs is given in Table 1.8 and can be easily checked for its validity. From the Figure 1.22(a), we can observe that the output CARRY is high when two or more number of inputs are high. Yet another output SUM will get high output, when an odd number of inputs are high. This can be verified from Table 1.8. The full adder can do more than a million additions per second.

TABLE 1.8 Truth Table of Full-Adder

A	B	C_{in}	Σ	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

C_{in} = input carry, sometimes designated as *CI*
C_{out} = output carruy, sometimes designated as *Co*
Σ = sum
A and *B* = input variables (operands)

Parallel Binary Adder

The parallel binary adder is a combinational circuit of various full-adders in parallel structure. When more than one 1-bit numbers are to be added, there can be full-adder circuit for every column to perform the addition .The number of full-adder in a parallel binary adder depends on the number of bits present in the number for the addition. If 4-bit numbers are to be added, then there will be 4-full adder in the parallel binary adder. The parallel binary adder can be designed with the help of basic logic gates. The sub-module in the logic circuit will resemble the logic gate of half-adder and full-adder to understand it clearly. Let us put light on designing and working of the 2-bit parallel binary adder.

Logic Circuit of 2-Bit parallel Binary Adder

The 2-Bit parallel binary adder can be designed with the help of Ex-OR gate and AND gate. If you carefully observe the logic circuit of 2-bit parallel binary adder, you can notice that 2-full adder circuits are connected in a parallel manner. Now, we easily guess and understand the working of this.

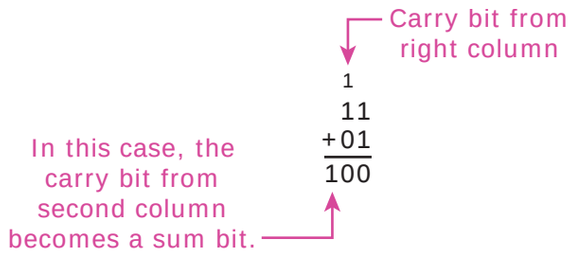


FIGURE 1.23(a) Addition of 2 parallel bits

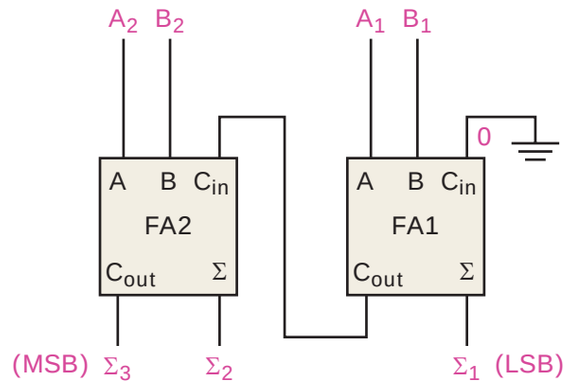


FIGURE 1.23(b) Block Diagram of 2-bit Parallel Adder

Figure 1.23(a) shows the method of 2-bit parallel addition and Figure 1.23(b) shows the schematic block diagram of the same. As we know, from the difference between the half-adder and full-adder, that the half-adder is a logic circuit which adds two 1-bit circuits but, does not add carry from previous addition. Therefore, full-adders came into action. A full-adder can add two 1-bit numbers along with the carry from previous addition.

Coming back to the parallel binary adder, it also has two full-adders. When we start add two numbers, the first step we follow is the addition of LSB (Least Significant Bit) of two numbers. After this, if we have any carry, we forward it to higher order columns. Now, the adder performs the similar task. It adds the LSBs of both the numbers and if any carry bit is there, it passes it to the carry-in terminal of another.

We may use half-adder for the addition of LSBs of both numbers as for the addition of LSBs there is no previous carry from previous addition. But, for the

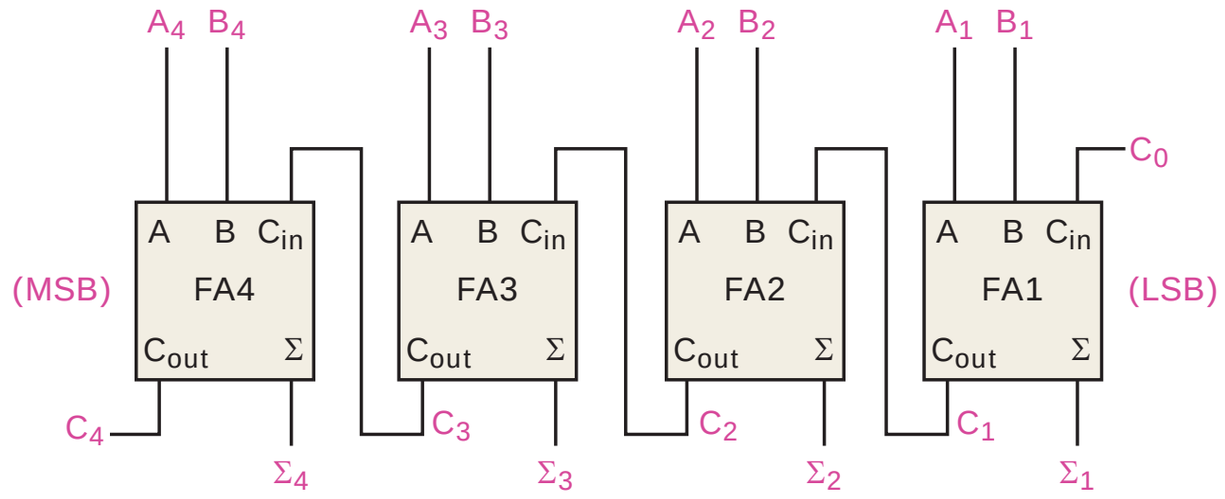


FIGURE 1.24(a) Four-Bit Parallel Binary Adder Block Diagram

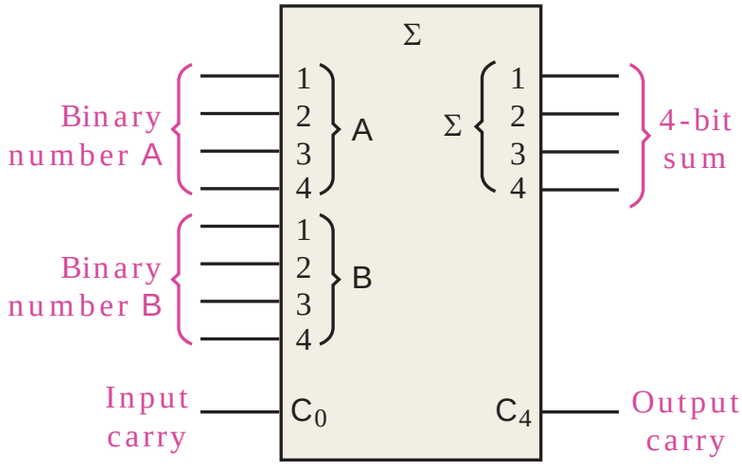


FIGURE 1.24(b) Logic Symbol

addition of bits present in higher order column, we must use full-adder because there may be or may not be a carry from previous addition.

With this, we complete the discussion of about 2-Bit parallel binary adder. As like, it is also better to know how two numbers with 4-bits are to be added. To perform this task, we definitely need 4-bit parallel binary adder.

Let us focus on the block diagram given in Figure 1.24(a), which represents a 4-bit parallel binary adder. It consists of 4-full adders, each of the 4-full adders have 3-input terminals and 2-output terminals. The input terminals are available for entering two numbers to be added and one input terminal is used for entering the previous carry.

The carry generated from the addition will be generated from C_{out} terminal. The sum of the addition will be generated from the sum bit of the adder. It must be noted here that C_{out} stands for carry-out and C_{in} stands for carry-in. The connection will be such that the C_{out} terminal to the C_{in} terminal of next full-adder used for high order column.

For addition of LSBs, we have a choice of either to use half-adder or to use full-adder. This is because we don't have previous carry, so half adder can also be used. It we want to use full adder, then C_{in} terminal of the full adder can be grounded.

For other full adders connected to higher order column, this will not be a major issue, because the C_{out} terminal of the previous adder can be connected to the carry-in of adders connected to higher order columns.

Significance of Parallel Binary Adder

With the help of full-adder, we cannot add numbers of more than 1-Bit. As the number of bits increases in a number, the column of addition also increases. A full-adder can add only one column, thus for each column we used a full-adder. This combined design of all full adder results in a combinational circuit, which is called parallel binary adder.

Half-Subtractor

Half Subtractor is a digital circuit which process the subtraction of two 1-bit (0, 1) numbers. In this, the two numbers involved are called as Minuend and Subtrahend nothing but the inputs, named as X and Y. X is the Minuend and Y is the Subtrahend. There are two outputs named as D (differences) and B (Borrow). The word 'HALF' before the subtractor signifies that it deals with only two 1-bit numbers, it has nothing to do with the borrow from the previous stage. Figure 1.25 clearly elaborates the subtraction rule of binary numbers. The logic circuit of the Half-Adder is shown in Figure 1.26(a) and the symbol is shown in Figure 1.26(b). The operation of this logic circuit is based on the rules of binary subtraction given in truth table (Table 1.9) reproduce on the basis of subtraction process.

Procedure for Subtraction

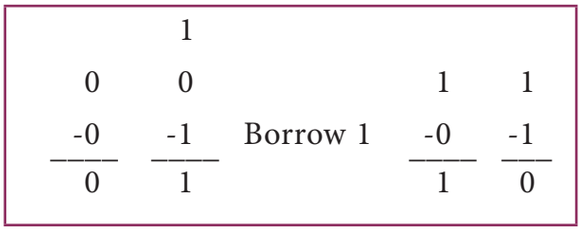


FIGURE 1.25 Binary Subtraction Rules

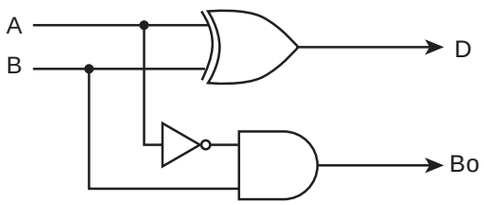


FIGURE 1.26 (a) Half Subtractor Circuit

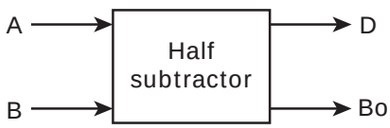


FIGURE 1.26 (b) Symbol of Half Subtractor

Circuit of Half-Subtractor

The logic circuit of Half-Subtractor involves usage of logic circuits. In order to design logic circuit, we should understand two concepts. First, the difference operation of half-subtractor resembles operation of EX-OR gate. Thus, we can easily utilise the EX-OR gate for generating difference bit. Similarly, the borrow generated by half-subtractor can be easily obtained by using the combination of NOT gate and AND gate.

TABLE 1.9 Truth Table of Half-Subtractor

Input		Output	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Truth Table of Half-Subtractor

In case of half subtractor there are two inputs. Thus the number of possible combinations will be 4. The resultant of all the 4 inputs will be described as outputs. The output of half-subtractor is described in two columns. One will signify the difference bit and another will signify the borrow bit. To derive the truth table, just use the EX-OR operation of two inputs for generating difference and NOT followed by AND operation for generating the borrow bit.

Full-Subtractor

The binary subtraction half-subtractor can handle only 2-bits at a time and can be used for the least significant column of a subtraction problem. Just like a full-adder, a full-subtractor circuit is required to perform a multi-bit subtraction, where a borrow from the previous bit position may also be there.

It has 3-inputs viz., X (minuend), Y (subtrahend) and B_{in} (borrow from the previous stage). It has two outputs such as D (difference) and B_{out} (borrow) as shown in the symbol given in Figure 1.27(a)

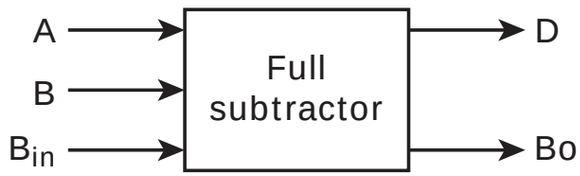


FIGURE 1.27 (a) Symbol of Full-Subtractor

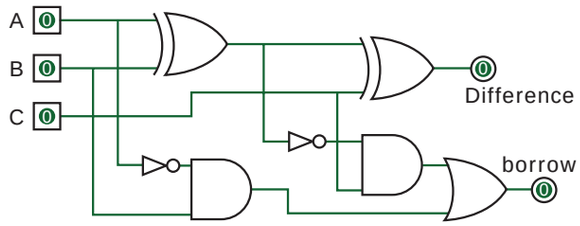


FIGURE 1.27 (b) Circuit diagram of Full Subtractor

Full-Subtractor is formed by using two half-subtractors and one OR gate. Figure 1.27(b) shows the circuit diagram of Full-Subtractor. For subtraction of n-bit numbers directly, we have to cascade n-full-subtractors. Truth table for full-subtractor is given in Table 1.10.

TABLE 1.10 Truth Table of Full-Subtractor

A	B	C	Difference	Borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

1.5 Comparators

The basic function of comparator is to compare the magnitudes of two binary quantities to determine the relationship of those quantities.

Equality

The Exclusive NOR gate can be used as a basic comparator, because its output is 0, if the two input bits are not equal and 1, if the input bits are equal. Figure 1.28 shows the Exclusive-NOR gate as 2-bit comparator.

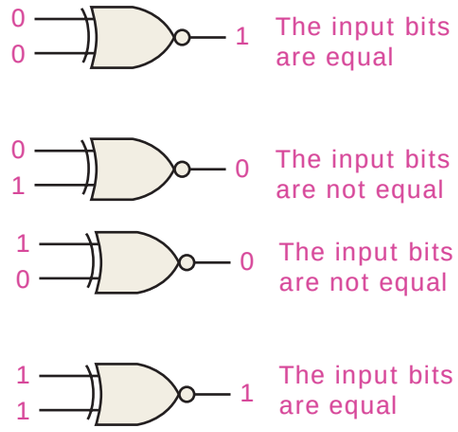


FIGURE 1.28 Basic Comparator Operation

To compare two binary-bits containing two bits each, an additional Exclusive-NOR gate is necessary. The two least significant bits (LSBs) of the two numbers are compared by gate G1, and the two most significant bits (MSBs) are compared by gate G2, as shown in Figure 1.29. If the two numbers are equal, their corresponding bits are the same and the output of each Exclusive-NOR gate is 1. If the corresponding sets of bits are not equal, a 0 occurs on that Exclusive-NOR gate output.

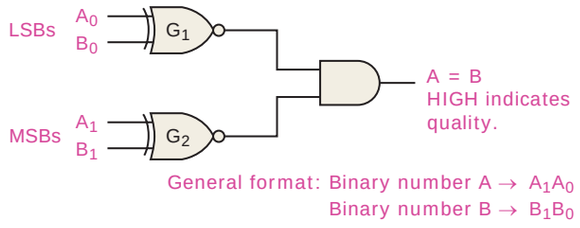
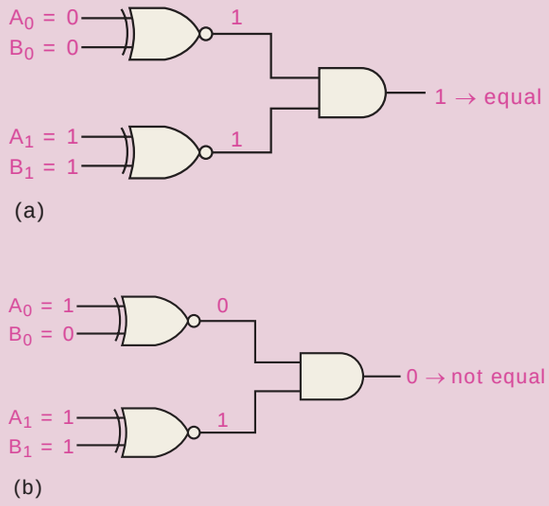


FIGURE 1.29 Logic diagram for equality comparison of two 2-bit numbers

In order to produce a single output indicating an equality or inequality of two numbers, an AND gate can be combined with XNOR gates as shown in Figure 1.29. The output of each Exclusive-NOR gate is applied to the AND gate input. When the two input bits for each Exclusive-NOR gates are equal, the corresponding bits of the numbers are equal and a 0 appears on at least one input of the AND gate to produce a 1 on its output. Thus, the output of the AND gate indicates equality (1) or inequality (0) of the two numbers. The following example clearly explains this operation for two specific cases.

EXAMPLE 1.2

Apply each of the following sets of binary numbers to the comparator inputs in Figures1.30(a) & (b) and determine the output by the following the logic levels through the circuit.



FIGURES 1.30 (a)&(b) Application of Comparators

Solution

(a) The output is 1 for inputs 10 and 10 as shown in Figure 1.30(a).

(b) The output is 0 for inputs 11 and 10, as shown in Figure 1.30(b).

Activity

Repeat the process for binary inputs of 01 and 10.

Note

The basic comparator can be expanded to any number of bits. The AND gate sets the condition that all corresponding bits of the two numbers must be equal if the two numbers themselves are equal.

Inequality

In addition to the equality output, many IC comparators provide additional outputs that indicate which of the two binary numbers being compared is larger. That is, there is an output that indicates when number A is greater than number B (A > B) and an output that indicates when number A is less than number B (A < B), as shown in Figure 1.31.

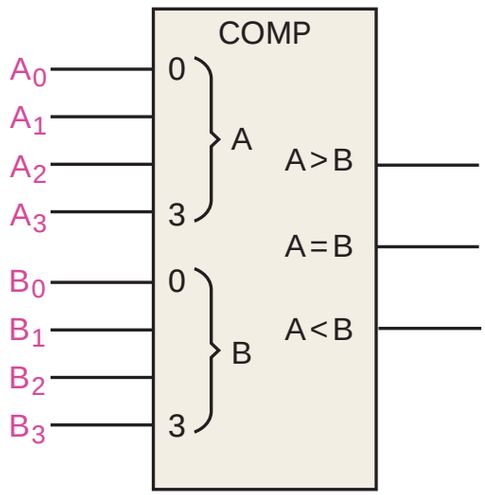


FIGURE 1.31 Logic symbol for a 4-bit comparator with inequality operation

To determine the inequality of binary numbers A and B, we first examine the highest order bit in each number. The following conditions are possible:

1. If $A_3 = 1$ and $B_3 = 0$, number A is greater than number B.
2. If $A_3 = 0$ and $B_3 = 1$, number A is less than number B.
3. IF $A_3 = B_3$, then you must examine the next lower bit position for an inequality.

These three operations are valid for each-bit position in the numbers. The general procedure used in a comparator is to check for an inequality in a bit position, starting with highest order bits (MSBs). When such an inequality is found, the relationship of the two numbers is established and any other inequalities in lower-order bit positions must be ignored because it is possible for an opposite indication to occur, the highest-order indication must take precedence. This can be explained through an example.

EXAMPLE 1.3

Determine the input numbers for the outputs A = B, A > B, and A < B shown in Figure 1.32.

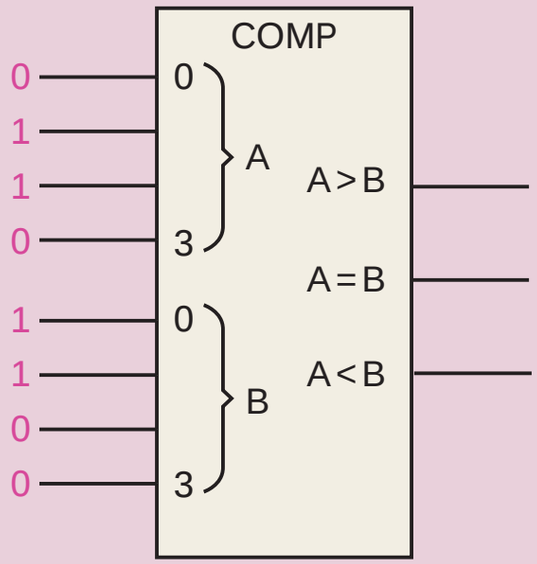


FIGURE 1.32 Application of Comparator

Solution

The number on the A inputs is 0110, the number on the B inputs is 0011. Then, the A > B output is HIGH and the other outputs are LOW.

Activity

What are the outputs when $A_3A_2A_1A_0 = 1001$ and $B_3B_2B_1B_0 = 1010$?

1.6 Decoders

An electronic device that converts signals from one form to another i.e., code into set of signals. Decoding is the process of converting code into plain text or any format that is useful for subsequent processes. It does the reverse of encoding. It converts encoded data communicated during transmission (like TV signals from satellite and Computer e-mails) and files to their original states.

In digital electronics, a binary decoder is a combinational logic circuit that converts binary information from the 'n' coded inputs to a maximum of 2^n

unique outputs. They are used in wide variety of applications, including data demultiplexing, seven segment displays and memory address decoding.

Basic Binary Decoder

We need to determine when a binary 1001 occurs on the inputs of a digital circuit. An AND can be used as the basic decoding element because it produces a HIGH output only when all of its inputs are HIGH. Therefore, we must make sure that all of the inputs to the AND gate are HIGH when the binary number 1001 occurs. This can be done by inverting the two middle inputs (the 0s), as shown in Figure 1.33.

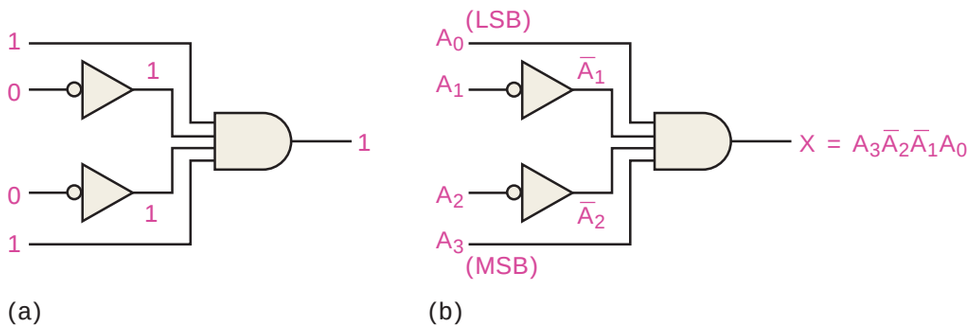


FIGURE 1.33 Decoding logic for the binary code 1001 with an active-HIGH output

EXAMPLE 1.4

Determine the logic required to decode the binary number 1011 by producing a HIGH level on the output.

Solution

The decoding function can be formed by complementing only the variables that appear as 0 in the desired binary number as follows:

$$X = A_3 A_2 A_1 A_0 \quad (1011)$$

This function can be implemented by connecting the true (un-complemented) variables A_0, A_1 and A_3 directly to the inputs of an AND gate and inverting the variables A_2 before applying it to the AND gate input. The decoding logic is shown in Figure 1.34.

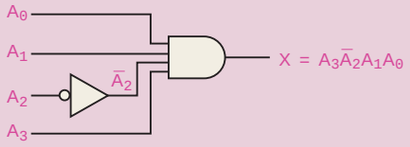


FIGURE 1.34 Decoding logic for producing a HIGH output when 1011 is on the inputs.

1.7 Encoders

Encoder is a device, circuit, transducer, software program, algorithm or person that converts information from one format or code to another for the purpose of standardisation or compression. An encoder is a combinational logic circuit that essentially performs a "reverse" decoder function. An encoder accepts an active level on one of its inputs representing a digit, such as a decimal or octal digit and converts it to a coded output, such as BCD or binary. Encoders can also be devised to encode various symbols and alphabetic characters. The process of converting from familiar symbols or numbers to a coded format is called encoding.

Decimal-to-BCD Encoder

This type of encoder has ten inputs. One for each decimal digit and four outputs corresponding to the BCD code as shown in the Figure1.35. This is a basic 10-line-to-4-line encoder.

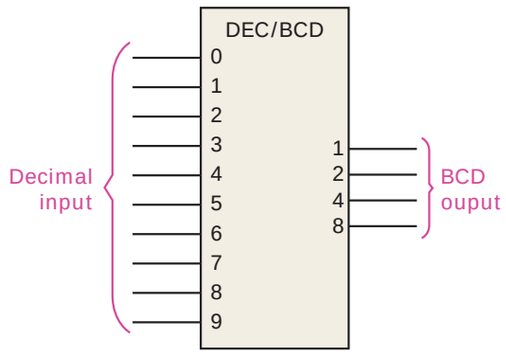


FIGURE1.35 Logic symbol for a Decimal-to-BCD encoder

The BCD (8421) code is listed in Table1.11. From this table, you can determine the relationship between each BCD bit and the decimal digits in order to analyse the logic. For instance, the most significant bit of the BCD code, A_3, is always 1 for decimal digit 8 or 9. An OR expression for bit A_3 in terms of the decimal digits can therefore be written as $A_3 = 8 + 9$.

TABLE 1.11 Decimal to BCD Encoder

	BCD Code			
DECIMAL DIGIT	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Bit A_2 is always 1 for decimal digit 4, 5, 6 or 7 and can be expressed as an OR function as follows:

$$A_2 = 4 + 5 + 6 + 7$$

Bit A_1 is always 1 for decimal digit 2, 3, 6 or 7 and can be expressed as

$$A_1 = 2 + 3 + 6 + 7$$

Finally, A_0 is always for decimal digit 1, 3, 5, 7 or 9

$$A_0 = 1 + 3 + 5 + 7 + 9$$

Now, let us implement the logic circuitry required for encoding each decimal digit to a BCD code by using the logic expressions just developed. It is simply a matter of ORing the appropriate decimal input lines to form each BCD output. The basic encoder logic resulting from these expressions is shown in Figure 1.36.

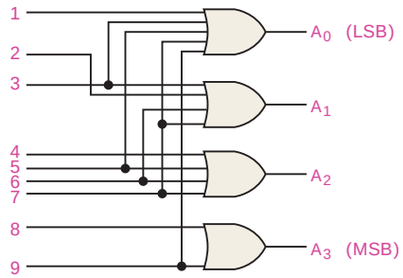


FIGURE 1.36 Basic logic diagram of Decimal-to-BCD encoder.

Note

A 0-digit input is not needed because the BCD outputs are all low, when there are no HIGH inputs.

The basic operation of the circuit shown in Figure 1.36 is briefly described as follows: When a HIGH appears on one of the decimal input lines, the appropriate levels occur on the four BCD output lines. For instance, if input line 9 is HIGH (assuming all other input lines are LOW), this condition will produce a HIGH on outputs A_0 and A_3 and LOWs on outputs A_1 and A_2, which is the BCD code (1001) for decimal 9.

1.8 Multiplexer

A multiplexer (MUX) is a device allowing one or more low speed analog or digital signal to be selected, combined and transmitted at a higher speed on a single shared medium or within a single shared device. A MUX function is a multiple-input, single-output switch.

Multiple signals share one device or transmission conductor such as copper or fibre optic cable. In telecommunication, the analog or digital signals transmitted on several communication channels by a multiplex method. These signals are single-output higher-speed signals. A 4-to-1 multiplexer contains four input signals and 2-to-1 multiplexer has two input signals and one output signal.

A logic symbol for a 4-input multiplexer (MUX) is shown in Figure 1.37. Notice that there are two data-select lines because with two select bits, any one of the four data-input lines can be selected.

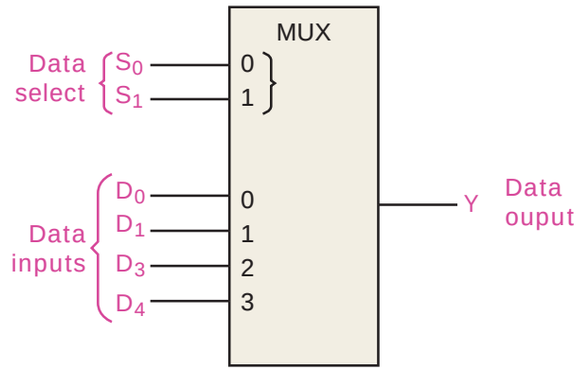


FIGURE 1.37 Logic symbol for a 1-of-4 data selector/ multiplexer

Multiplexing

The technique of transmitting multiple signals over a single medium is defined as Multiplexing. This technique is widely used in the Open System Interconnection (OSI) model. The different types of multiplexing technologies are:

- Wavelength Division Multiplexing (WDM)
- Frequency Division Multiplexing (FDM)
- Dense Wavelength Division Multiplexing (DWDM)
- Conventional Wavelength Division Multiplexing (CWDM)
- Reconfigurable Optical Add-Drop Multiplexer (ROADM)
- Orthogonal Frequency Division Multiplexing (OFDM)
- Add/Drop Multiplexing (ADM)
- Inverse Multiplexing (IMUX)

Multivibrators

Though we studied about Multivibrators in eleventh standard, it is essential to recollect the working of Bi-stable Multivibrators before entering into the Flip-flop.

Bi-stable or flip-flop multivibrator

The bistable multivibrator has both the states (HIGH, LOW) at stable condition. It requires an external triggering pulse to change the operation from either one state to the other. Thus, one pulse is used to generate half-cycle of square wave and another pulse to generate the next half-cycle of square wave. It is also known as a flip-flop multivibrator because of its assured two possible states. So that, it can store one bit of information and is widely used in digital logic and computer memory. Hence, a flip-flop is nothing but storage (memory) device, which can store one-bit at a time.

1.9 Flip-Flops

The output of the digital circuits studied in previous chapters are dependent entirely on the input, i.e., if the input changes, the output also changes. However, there are requirements for a digital device or circuit whose output will remain unchanged, once set, even if there is a change in input. Such a device can be used to store a binary number. A flip-flop is one such circuit.

Definition

A flip-flop is a bi-stable circuit made up of logic gates. A bi-stable circuit can exist in either of the two stable states indefinitely

and can be made to change its state by means of some external signal. The most important use of this property is that a flip-flop can "store" binary information. We have seen that a logic gate can make a logical decision based on the immediate conditions at the input terminals. However, the gates normally do not have a memory characteristic to retain the input data. On the other hand, flip-flops have the valuable feature of remembering. The reason is that a flip-flop circuit is bi-stable.

Because the flip-flop's output remains at 0 or 1 depending on the last input signal, the flip-flop can be said to be in "remember" condition. Another name for the flip-flop is bi-stable multivibrator. We shall discuss three important types of flip-flops viz. (i) R–S flip-flop and (ii) J–K flip-flop (iii) D-flip-flop.

Flip-Flop or Latch

Any device or circuit that has two stable states is said to be bi-stable. For instance, a toggle switch has two stable states. It is either up or down, depending on the position of the switch as shown in Figure 1.38(a). The switch is also said to have memory since it will remain as set until someone changes its position. A flip-flop is a bi-stable electronic circuit that has two stable states-that, i.e., its output is either 0 or +5 V DC as shown in Figure 1.38(b).

The flip-flop also has memory, since its output will remain as set until something is done to change it. As such, the flip-flop (or the switch) can be regarded as a memory device. In fact, any bistable device can be used to store one binary digit (bit). For instance, when the flip-flop has its output set at 0 V DC, it can be regarded as storing a logic 0 and when its output is set at + 5 V DC, as storing a logic 1. The flip-flop is often called a latch, since it will hold, or latch, in either stable state.

Basic Idea

One of the easiest ways to construct a flip-flop is to connect two inverters in series as shown in Figure 1.39(a). The line connecting the output of inverter B (*INV B*) back to the input of inverter A (*INV A*) is referred to as the feedback line.

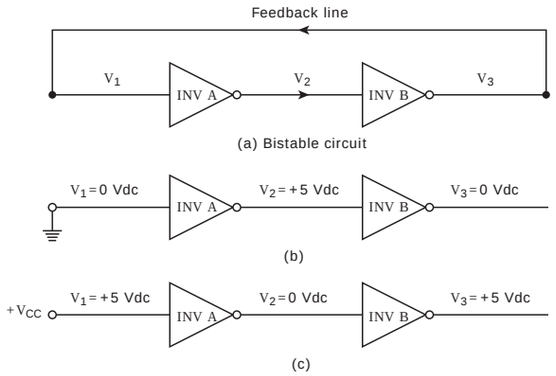


FIGURE 1.39 Flip-Flop using inverters

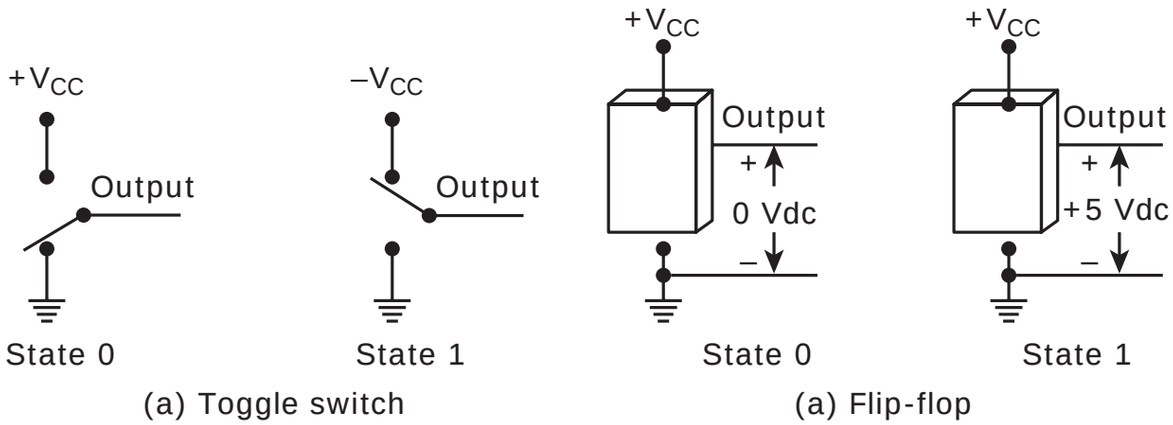


FIGURE 1.38 Flip-Flop Operations

For the moment, remove the feedback line and consider V_1 as the input and V_3 as the output as shown in Figure 1.39(b). There are only two possible signals in a digital system, and in this case, we will define L = 0 = 0 V DC and H = 1 = + 5 V DC. If V_1 is set to 0 V DC, then V_3 will also be 0 V DC. Now, if the feedback line shown in Figure 1.39(a) is reconnected, the ground can be removed from V_1 and V_3 will remain at 0 V DC. This is true since, once the input of INV A is grounded, the output of INV B will go low and can then be used to hold the input of INV A low by using the feedback line. This is one stable state V_3 = 0 V DC. Conversely, if V_1 is +5 V DC, V_3 will also be +5 V DC as seen in Figure 1.39(c). The feedback line can again be used to hold V_1 at + 5 V DC since V_3 is also at + 5 V DC. This is the second stable state V_3 = + 5 V DC.

NOR-Gate latch

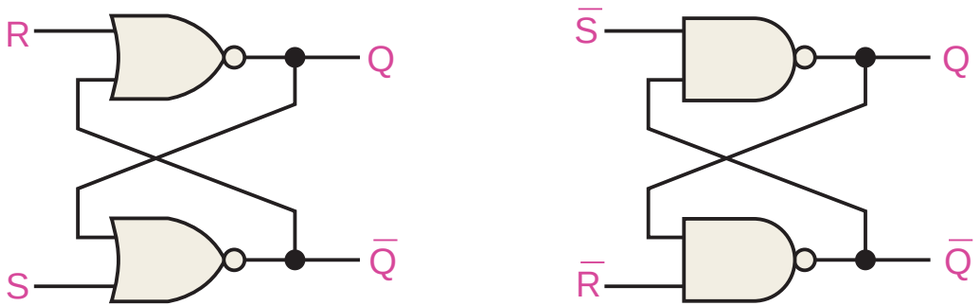
The basic flip-flop shown in Figure 1.39(a) can be improved by replacing the inverters with either NOR or NAND gates. The additional inputs on these gates provide a convenient means for application of input signals to switch the flip-flop from one stable state to the other. Two 2-input NOR gates are connected as shown in Figure 1.40(a) to form a flip-flop. Notice that, if the two inputs labelled R and S are ignored, this circuit will function exactly as the one shown in Figure 1.39(a).

This circuit is redrawn in a more conventional form as shown in Figure 1.40. From this, we will study the function of NOR-latch. The flip-flop actually has two outputs, defined in more general terms as Q and $\overline{Q}$. It should be clear that regardless of the value of Q, its complement is $\overline{Q}$. There are two inputs to the flip-flop defined as R and S. The input/output possibilities for this RS flip-flop are summarized in the truth table in Table 1.12.

TABLE 1.12 Truth Table of RS flip-flop

R	S	Q	Action
0	0	Last State	No Change
0	1	1	SET
1	0	0	RESET
1	1	?	Forbidden

1. The first input condition in the truth table is $R = 0$ and $S = 0$. Since 0 at the input of a NOR gate has no effect on its output, the flip-flop simply remains in its present state; that is, Q remains unchanged.



(a) Active-HIGH input S-R latch (b) Active-LOW input $\overline{S}$-$\overline{R}$ latch

FIGURE 1.40 Latch using NOR or NAND gates

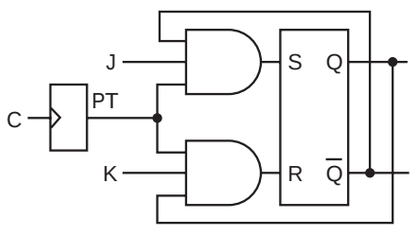
2. The second input condition R = 0 and S = 1 forces the output of NOR gate B low. Both inputs to NOR-gate A are now low, and the NOR-gate output must be high. Thus, a 1 at the S input is said to SET the flip-flop and it switches to the stable state where Q = 1.
3. The third input condition is R = 1 and S = 0 forces the output of NOR gate A low, and hence both the inputs to NOR gate B are now low, the output must be high. Thus, a 1 at the R input is said to RESET the flip-flop and it switches to the stable state where Q = 0 (or Q = 1).
4. The last input condition R = 1 and S = 1 is forbidden, as this forces the outputs of both NOR gates to the low state. In other words, both Q = 0 and $\overline{Q}$ 0 at the same time. But, this violates the basic definition of a flip-flop that requires Q to be the complement of $\overline{Q}$, and so it is generally agreed never to impose this input condition. Incidentally, if this condition is for some reason imposed and the next input is R = 0, S = 0, then the resulting state Q depends on propagation delays of two NOR gates. If delay of gate A is less, i.e., it acts faster, then Q = 1, else it is 0. Such dependence makes the job of a design engineer difficult, as any replacement of a NOR gate will make Q unpredictable. That's why R = 1, S = 1 is forbidden and truth table entry is a question mark (?). It is also important to remember that TTL gate inputs are quite noise-sensitive and therefore should never be left unconnected (floating). Each input must be connected either to the output of a prior circuit, or if unused, to GND or + V_{cc}.

Edge-Triggered Jk Flip-Flop

Setting R = S = 1 with an edge-triggered RS flip-flop forces both Q and $\overline{Q}$ to the same logic level. This is an illegal condition, and it is not possible to predict the final state of Q. The JK flip-flop accounts for this illegal input and is therefore a more versatile circuit. Among other things, flip-flops can be used to build counters. Counters can be used to count the number of PTs or NTs of a clock. For the purpose of counting, the JK flip-flop is an ideal element to use. There are many commercially available edge-triggered JK flip-flops. Let's see how they function.

Positive-Edge-Triggered J K Flip-Flops

In Figure 1.41, the pulse-forming box changes the clock into a series of positive pulses, and thus this circuit will be sensitive to PTs of the clock. The basic circuit is identical to the previous positive-edge-triggered RS flip-flop with two important conditions:



(a) One way to implement a JK flip-flop

FIGURE 1.41 JK-flip-flops with Positive-Edge

TABLE 1.13 Truth Table of JK Flip-flop

C	J	K	$Q_n + 1$	Action
↑	0	0	Q_n Last State	No Change
↑	0	1	0	RESET
↑	1	0	1	SET
↑	1	1	$\overline{Q}_n$(toggle)	Toggle

1. Q output is connected back to the input of the lower AND gate.
2. $\overline{Q}$ output is connected back to the input of the upper AND gate.

This cross-coupling from outputs to inputs changes the RS flip-flop into a JK flip-flop.

D Flip-Flop or D Latch

D Flip-flops are used as a part of memory storage elements and data processors as well. D flip-flop can be built using NAND gate or with NOR gate. Due to its versatility they are available as IC packages. The major applications of D flip-flop are to introduce delay in timing circuit, as a buffer, sampling data at specific intervals. D flip-flop is simpler in terms of wiring connection compared to JK flip-flop. Figures 1.42(a) & (b) show the D flip-flop symbol and D flip-flop using NAND gates, respectively. Whenever the clock signal is LOW, the input is never going to affect the output state. The clock has to be high for the inputs to get active. Thus, D flip-flop is a controlled Bi-stable latch where the clock signal is the control signal. Again, this gets divided into positive edge triggered D flip-flop and negative edge triggered D flip-flop. Thus, the output has two stable states based on the inputs as summarized in Table 1.14.

TABLE 1.14 Truth Table of D Flip-Flop

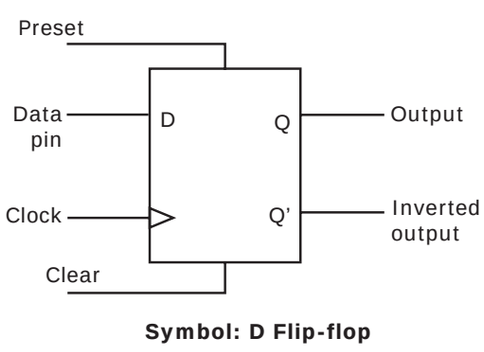
Clock	INPUT	OUTPUT	
	D	Q	Q'
LOW	X	0	1
HIGH	0	0	1
HIGH	1	1	0

1.10 Counters

As you learned in previous Section, flip-flops can be connected together to perform counting operations. Such a group of flip-flops is a counter, which is a type of finite state machine. The number of flip-flops used and the way in which they are connected determine the number of states (called the modulus) and also the specific sequence of states that the counter goes through during each complete cycle. Counters are classified into two broad categories according to the way they are clocked: asynchronous and synchronous. In asynchronous counters, (commonly called ripple counters), the first flip-flop is clocked by the external clock pulse and then each successive flip-flop is clocked by the output of the preceding flip-flop. In synchronous counters, the clock input is connected to all of the flip-flops so that they are clocked simultaneously. Within each of these two categories, counters are classified primarily by the type of sequence, the number of states, or the number of flip-flops in the counter.

Asynchronous Counters

The term asynchronous refers to events that do not have a fixed time relationship with each other and generally, do not occur at the same time. An asynchronous counter is one in which the flip-flops (FF) within the counter do not change states at exactly the same time because they do not have a common clock pulse.



Symbol: D Flip-flop

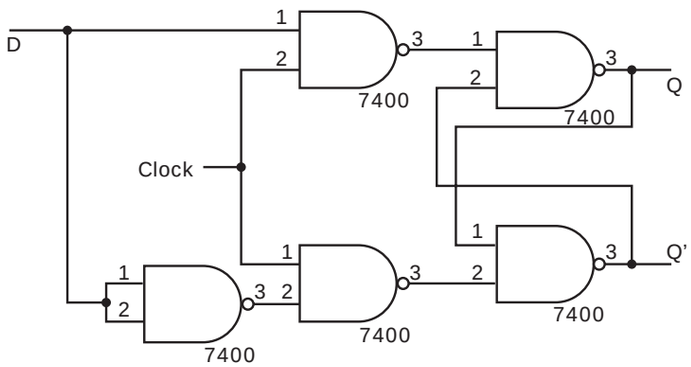


FIGURE 1.42(a) D Flip-Flop Symbol (b) D Flip-Flop using NAND gates

A 2-Bit Asynchronous Binary Counter

Figure 1.43 shows a 2-bit counter connected for asynchronous operation. Notice that the clock (CLK) is applied only to the clock input (C) of the first flip-flop FF0, which is always the least significant bit (LSB). The second flip-flop FF1 is triggered by the $\overline{Q}_0$ output of FF0. FF0 changes state at the positive-going edge of each clock pulse, but FF1 changes only when triggered by a positive-going transition of the $\overline{Q}_0$ output of FF0. Because of the inherent propagation delay time through a flip-flop, a transition of the input clock pulse (CLK) and a transition of the $\overline{Q}_0$ output of FF0 can never occur at exactly the same time. Therefore, the two flip-flops are never simultaneously triggered, so the counter operation is asynchronous.

From the Timing Diagram shown in Figure 1.44, let us examine the basic operation of the asynchronous counter by applying four clock pulses to FF0 and observing the $\overline{Q}_0$ output of each flip-flop. Figure 1.44 illustrates the changes in the state of the flip-flop outputs in response to the clock pulses. Both flip-flops are connected for toggle operation (D = Q) and are assumed to be initially RESET (Q LOW). The positive-going edge of CLK1 (clock pulse 1) causes the $\overline{Q}_0$ output of FF0 to go HIGH as shown in Figure 1.44. At the same time the $\overline{Q}_0$ output goes LOW, but it has no effect on FF1 because a positive-going transition must occur to trigger the flip-flop.

After the leading edge of CLK1, Q_0 = 1 and Q_1 = 0. The positive-going edge of CLK2 causes Q_0 to go LOW. Output $\overline{Q}_0$ goes HIGH and triggers FF1, causing Q_1 to go HIGH. After the leading edge of CLK2, Q_0 = 0 and Q_1 = 1. The positive-going edge of CLK3 causes Q_0 to go HIGH again. Output Q0 goes LOW and has no effect on FF1. Thus, after the leading edge of CLK3, Q_0 = 1 and Q_1 = 1. The positive-going edge of CLK4 causes Q_0 to go LOW, while Q_0 goes HIGH and triggers FF1, causing Q_1 to go LOW. After the leading edge of CLK4, Q_0 = 0 and Q_1 = 0. The counter has now recycled to its original state (both flip-flops are RESET). In the timing diagram, the waveforms of the Q_0 and Q_1 outputs are shown relative to the clock pulses as illustrated in Figure 1.44.

For simplicity, the transitions of Q_0, Q_1 and the clock pulses are shown as simultaneous even though this is an asynchronous counter. There is, of course, some small delay between the CLK and the Q_0 transition and between the Q_0 transition and the Q_1 transition. Note in Figure 1.44 that the 2-bit counter exhibits four different states, as you would expect with two flip-flops (2^2 = 4). Also, notice that if Q_0 represents the least significant bit (LSB) and Q_1 represents the most significant bit

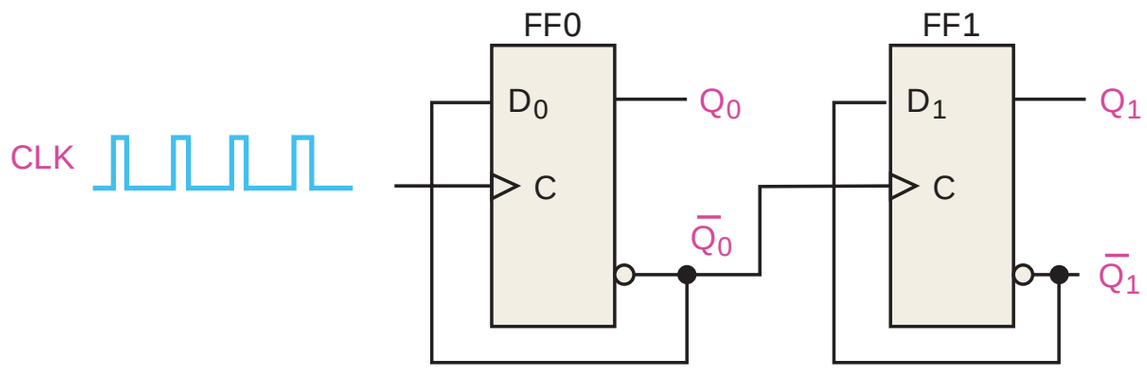


FIGURE 1.43 A 2-bit asynchronous binary counter

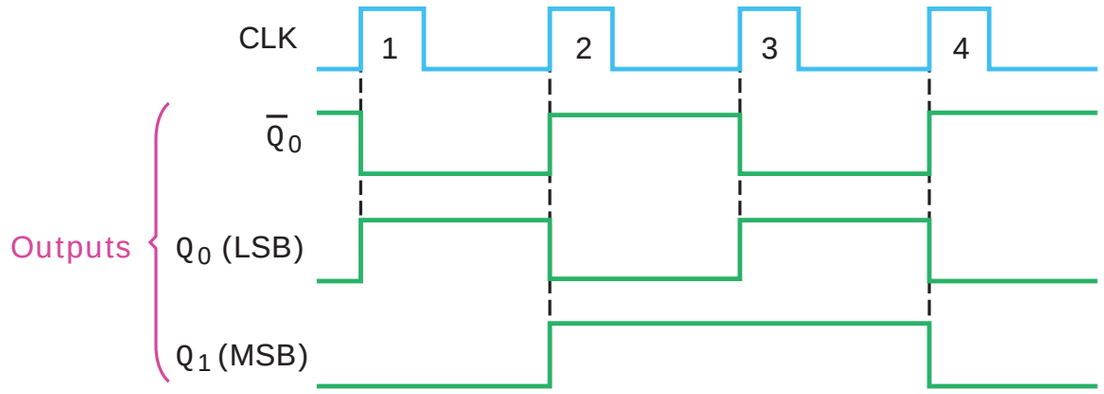


FIGURE 1.44 Timing diagram for the counter given in Figure 1.43

(MSB), the sequence of counter states represents a sequence of binary numbers as listed in Table 1.15.

TABLE 1.15 Binary state sequence for the counter shown in Fig 1.43

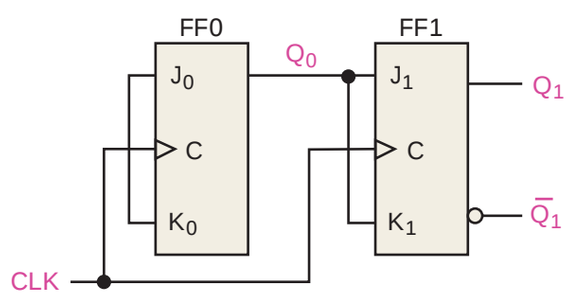
Clock Pulse	Q_1	Q_0
Initially	0	0
1	0	1
2	1	0
3	1	1
4 (recycle)	0	0

Since it goes through a binary sequence, the counter in Figure 1.43 is a binary counter. It actually counts the number of clock pulses up to three, and on the fourth pulse it recycles to its original state ($Q_0 = 0$, $Q_1 = 0$). The term recycle is commonly applied to counter operation, since it refers to the transition of the counter from its final state back to its original state.

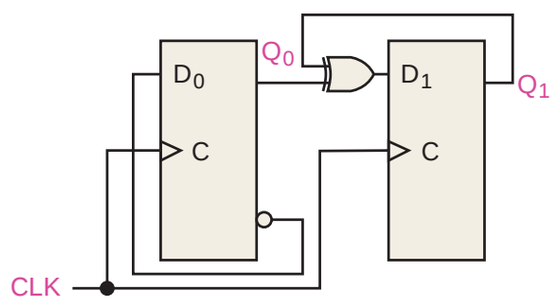
Synchronous Counters

The term synchronous refers to events that have a fixed time relationship with each other. A synchronous counter is one in which all the flip-flops in the counter are clocked at the same time by a common clock pulse. J-K flip-flops are used to illustrate most synchronous counters. D flip-flops can also be used but generally require more logic because of having no direct toggle or no-change states.

A 2-Bit Synchronous Binary Counter is shown in Figure 1.45(a). Notice that an arrangement different from that for the asynchronous counter must be used for the J_1 and K_1 inputs of FF1 in order to achieve a binary sequence. A D flip-flop implementation is shown in Figure 1.45(b).



(a) J-K flip-flop



(b) D flip-flop

FIGURE 1.45 2-bit synchronous binary counters

1.11 Registers (shift)

Shift registers are a type of sequential logic circuit used primarily for the storage of digital data and typically do not possess a characteristic internal sequence of states.

Shift Register Operations

Shift registers consist of arrangements of flip-flops and are important in applications involving the storage and transfer of data in a digital system. A register has no specified sequence of states, except in certain very specialized applications.

A register in general, is used solely for storing and shifting data (1s and 0s) entered into it from an external source and typically possesses no characteristic internal sequence of states.

A register is a digital circuit with two basic functions: data storage and data movement.

The storage capability of a register makes it an important type of memory device. Figure 1.49 illustrates the concept of storing a 1 or a 0 in a D flip-flop. A 1 is applied to the data input as shown, and a clock pulse is applied that stores the 1 by setting the flip-flop. When the 1 on the input is removed, the flip-flop remains in the SET state, thereby storing the 1. A similar procedure applies to the storage of a 0 by resetting the flip-flop, as illustrated in Figure 1.46.

The storage capacity of a register is the total number of bits (1s and 0s) of digital data it can retain. Each stage (flip-flop) in a shift register represents one bit of storage capacity; therefore, the number of stages in a register determines its storage capacity. The shift capability of a register permits the movement of data from stage to stage within the register or into or out of the register upon the application of clock pulses. Figure 1.47 illustrates the types of data movement in shift registers. The block represents any arbitrary 4-bit register, and the arrows indicate the direction of data movement.

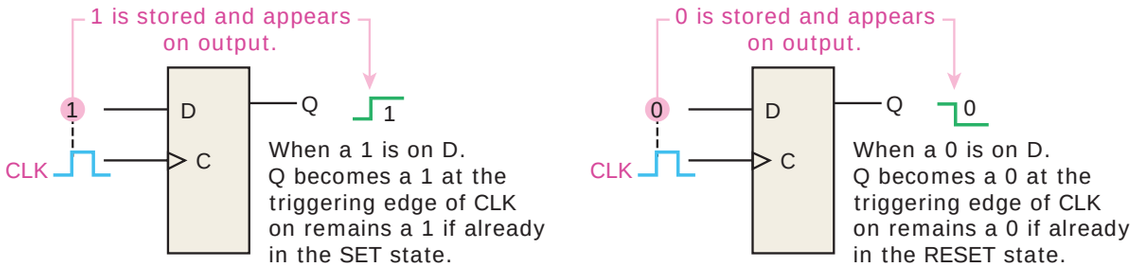


FIGURE 1.46 The flip-flop as a storage element.

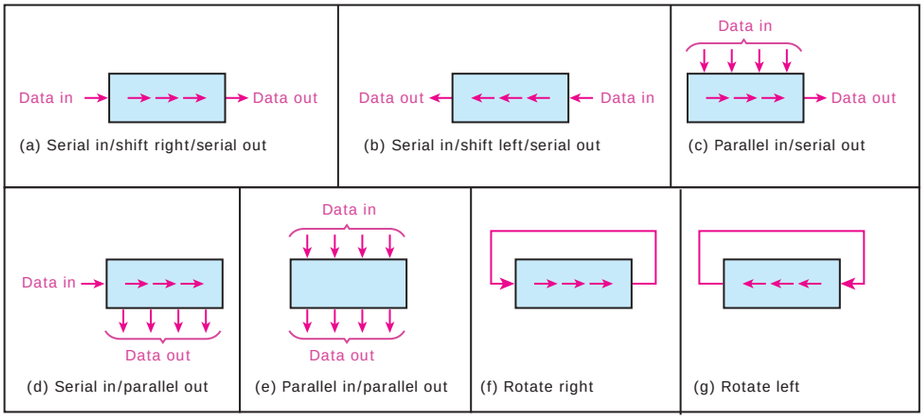


FIGURE 1.47 Basic data movement in shift registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

LEARNING OUTCOME

At the end of this chapter the students would have learned about the following:

- Construction of Combinational Gates and its applications
- Working of Arithmetic circuits like, Half, Full-adder and Half & Full Subtractor
- The way of Decoding and Encoding of Signals
- Multiplexing and De-Multiplexing
- Basic working of Flip-flops (Memory)
- Construction of Binary Counters and Registers and its Applications

GLOSSARY

NAND gate	A logic gate that produces a LOW output only when all the inputs are HIGH.
NOR gate	A logic gate in which the output is LOW when one or more of the inputs are HIGH.
OR gate	A logic gate that produces a HIGH output when one or more inputs are HIGH.
Sequential circuit	A digital circuit whose logic states follow a specified time sequence.
Propagation delay time	The time interval between the occurrence of an input transition and the occurrence of the corresponding output transition in a logic circuit.
Truth table	A table showing the inputs and corresponding output(s) of a logic circuit.
Adder	Digital circuit used to add binary digits.
Subtractor	Digital circuit used to subtract binary digits.
Encoder	Encoder is a device that converts information from one format or code to another.
Decoder	The process of converting code into plain text or any format. It's a reverse process of Encoding.
Multiplexer	A Multiple input and Single output switch.
Flip-flop	A Bi-stable digital circuit which can store a binary digit at a time.
Counter	Flip-flops connected together to perform counting operations.
Registers	A type of sequential logic circuit used primarily for the storage of digital data.

QUESTIONS

I Multiple choice Questions

1. When both input of the NAND gate goes HIGH, what will be the output?

(a) 1 (b) 0 (c) 10 (d) 01

2. The Complement of the variable is

(a) 0
(b) 1
(c) equal to the variable
(d) the inverse of the variable

3. According to Commutative law of addition

(a) AB = BA
(b) A = A + A
(c) A +(B+C) = (A+B) +C
(d) A + B = B + A

4. According to Distributive law

(a) A(B + C) = AB + AC
(b) A(BC) = ABC
(c) A(A + 1) = A
(d) A + AB = A

5. Which one of the rule is not a valid rule of Boolean algebra?

(a) A + 1 = 1
(b) A = A
(c) AA = A
(d) A + 0 = A

59XJ4

6. An Exclusive OR function is expressed as

(a) $A \cdot B + A \cdot B$
(b) $\left(\overline{A} \cdot B + A \cdot \overline{B}\right)$
(c) $\left(\overline{A} + B\right)\left(A + \overline{B}\right)$
(d) (A + B) (A + B)

7. The AND operation can be produced with

(a) two NAND gates
(b) three NAND gates
(c) one NOR gate
(d) three NOR gates

8. A Half-adder is characterised by

(a) two inputs and two outputs
(b) three inputs and two outputs
(c) two inputs and three outputs
(d) two inputs and one output

9. A 4-Bit parallel adder can add

(a) two 4-bit Binary numbers
(b) two 2-bit Binary numbers
(c) four bits at a time
(d) four bits in sequence

10. In general, a multiplexer has

(a) one data input, several data outputs and selection inputs
(b) one data input, one data outputs and one selection input
(c) several data inputs, several data outputs and selection inputs
(d) several data inputs, one data output and selection inputs

11. The flip-flop belongs to a category of logic circuits known as

(a) monostable multivibrator
(b) bistable multivibrators
(c) astable multivibrators
(d) one-shots

12. Asynchronous counters are known as

(a) ripple counters
(b) multiple clock counters
(c) decade counter
(d) modulus counters

13. An Asynchronous counter differs from a synchronous counter in
    (a) the number of states in its sequence
    (b) the method of clocking
    (c) the type of flip-flops used
    (d) the value of the modulus

14. To serially shift a byte of data into a shift register, there must be
    (a) one clock pulse
    (b) one load pulse
    (c) eight clock pulses
    (d) one clock pulse for each 1 in the data

II Answer in one or two sentences

1. Write the any three names of combinational gates.
2. Draw the construction of NAND gate with truth table.
3. Construct OR gate using NAND gate (diagram).
4. Define encoder
5. Write shortly about Multiplexer(MUX)
6. Draw the circuit of Half-adder.
7. Write the truth table of Full-Adder.
8. If a bit is to be stored, how it can be?
9. Write briefly about asynchronous counter.
10. Write about decoder

III Answer in a paragraph

1. Explain in detail the construction of Ex-OR and Ex-NOR gate with truth table.
2. Why NAND & NOR gates are called as Universal gates? Explain with an example.
3. If any 2 bits are to be added, how it can be done through a logic gate circuit? Justify with necessary diagrams?
4. Is it possible to perform subtraction in logic gates? Prove with circuit and table.
5. Define Multiplexer.

Part – D (10 Marks)

IV Answer in One Page (Essay type Question)

1. What are the three basic Boolean laws. Define each with example.
2. Construct full adder and half subtractor circuits. Prove with truth table.
3. Explain the working of JK-flip-flop.
4. Write about shift register.

Answers

1. (b)	2 (d)	3. (d)	4. (a)	5. (b)
6. (b)	7 (a)	8. (a)	9. (b)	10. (d)
11. (b)	12 (a)	13 (b)	14. (a)	