PREV
15. Rectifiers and Other Di

NEXT
17. Integrated Circuits

16

Digital Electronics

TOPICS DISCUSSED

- Number systems
- Logic gates
- Boolean algebra
- De Morgan's theorem
- Combinational circuits
- Boolean expressions
- Universal gates
- Flip-flops
- Shift registers
- Arithmetic circuits
- Memory function and data storage
- Digital systems

16.1 INTRODUCTION

diodes, resistors, etc. Digital electronics is an interesting subject of study.

Digital electronics includes electronic systems that use digital signals. Digital electronics or any digital circuit are usually made from large assemblies of logic gates. Digital electronics, digital circuits, and digital systems are often used interchangeably. Digital circuits and systems used in computers and electronic controls are *logic gates* and *multivibrators*. Logic gates are combinational (i.e., with no memory) and sequential (i.e., with memory). Combinational gates are AND, NOT, OR, NAND and NOR gates. Sequential gates include flip-flops (bistable multivibrators) which are available as registers and counters.

In this chapter brief descriptions of the above mentioned digital circuits and systems will be provided. Electrical signals that are continuous and can have any value over a given range are called analog signals. Electronic circuits that are used to process (i.e., amplify or modify) analog signals are called analog circuits. In a digital circuit, a signal is represented in one of two states i.e., low or high (on or off). These two voltage levels are called logic levels. Thus, a digital circuit functions in two states, i.e., in a binary manner.

Signals of two discrete values or levels, high and low, are called digital signals. The circuits which process digital signals are called digital circuits. Digital electronics, or any digital circuit, are made from large assemblies of logic gates which are simple electronic representation of boolean logic functions.

Digital circuits have a number of advantages over analog circuits. For example, signals represented digitally can be transmitted without degradation of quality. That is why analog signals are converted to digital signals through A to D converters and the digital signals are transmitted. These are again converted through D to A converters and the actual analog signal is recovered. For example, a continuous audio signal can be transmitted as a sequence of 1s and 0s and can be reconstructed without much error. An hour of music can be stored in a compact disc using digital signals which may be a few billion binary digits (1s and 0s). All systems comprising light, temperature, conductivity, pressure, magnetic fields, etc., are analog systems. The continuous signals of these are converted into discrete digital signals.

Before discussing logic gates and some digital circuits we will discuss data representation using number systems. The number system used to represent data in a computer is known as the binary number system.

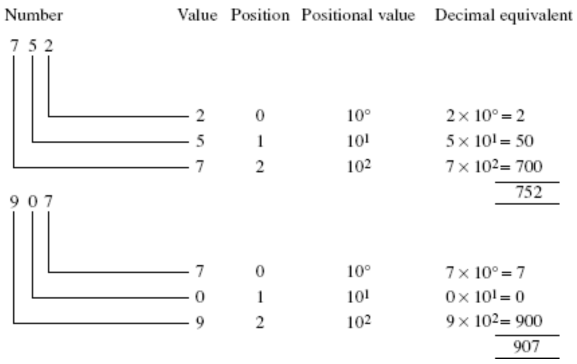
16.2 NUMBER SYSTEMS

A number system is a set of rules of symbols used to represent numbers. There are number systems like decimal, binary, octal, and hexadecimal. The knowledge of number systems is very important for the design of digital circuits and systems.

16.2.1 Decimal Number System

We are familiar with the decimal number system which uses 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, i.e., ten symbols. It is a base-10 counting because the total number of symbols used is ten. When we represent a number

Decimal 692 = $6\times10^{2} + 9\times10^{1} + 2\times10^{0}$



Number	Value	Position	Positional value	Decimal equivalent
7 5 2	2	0	10°	$2\times10^{\circ}=2$
	5	1	10^{1}	$5\times10^{1}=50$
	7	2	10^{2}	$7\times10^{2}=700$
				752
9 0 7	7	0	10°	$7\times10^{\circ}=7$
	0	1	10^{1}	$0\times10^{1}=0$
	9	2	10^{2}	$9\times10^{2}=900$
				907

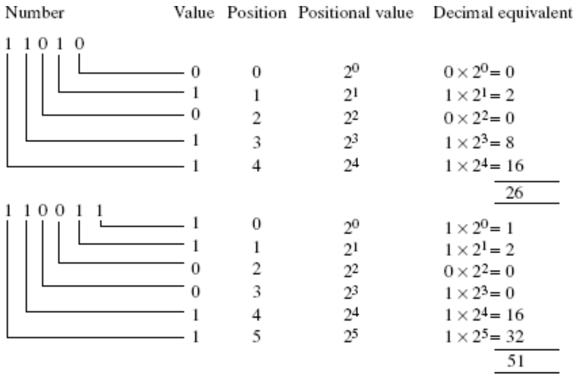
A decimal number of say, 824.48 is represented as

Decimal 824.48 = $8\times10^{2} + 2\times10^{1} + 4\times10^{6} \cdot 4\times10^{-1} + 8\times10^{-2}$

The multiples of 10 used are called weighted values. The decimal number 824.48 is represented as $(824.24)_{10}$ as 10 is the base.

16.2.2 Binary Number System

In the binary system a number is expressed by two symbols or digits, 0 and 1. Since the total number of symbols or digits used is two, it has a base 2. The binary symbols 0 and 1 are called binary digits or bits. A group of eight bits is called a byte. In this system the positional value is used in the same way as the decimal system. For example, two binary numbers, 11010 and 110011 are represented as their decimal equivalent as



Number	Value	Position	Positional value	Decimal equivalent
1 1 0 1 0	0	0	2^{0}	$0\times2^{0}=0$
	1	1	2^{1}	$1\times2^{1}=2$
	0	2	2^{2}	$0\times2^{2}=0$
	1	3	2^{3}	$1\times2^{3}=8$
	1	4	2^{4}	$1\times2^{4}=16$
				26
1 1 0 0 1 1	1	0	2^{0}	$1\times2^{0}=1$
	1	1	2^{1}	$1\times2^{1}=2$
	0	2	2^{2}	$0\times2^{2}=0$
	0	3	2^{3}	$1\times2^{3}=0$
	1	4	2^{4}	$1\times2^{4}=16$
	1	5	2^{5}	$1\times2^{5}=32$
				51

16.2.3 Conversion of Binary to Decimal

A binary number can be converted into its decimal equivalent number by noting that successive digits from the extreme right of a binary number are the coefficients of ascending power of 2 beginning with zeroeth power of 2. A few examples will illustrate this conversion.

Example 16.1 Binary number is 1100101. Convert this number into its decimal equivalent

1	1	0	0	1	0	1
1×2^6	1×2^5	0×2^4	0×2^3	1×2^2	0×2^1	1×2^0

64 + 32 + 0 + 0 + 4 + 0 + 1 = 101

Decimal number is 101. Therefore, $(1100101)_2 = (101)_{10}$

Example 16.2 Binary number is 11010. Convert this number into its decimal equivalent.

Solution:

1	1	0	1	0
1×2^4	1×2^3	0×2^2	1×2^1	0×2^0

16 + 8 + 0 + 2 + 0 = 26

Decimal number is 26. Therefore, $(11010)_2 = (26)_{10}$

Example 16.3 Binary number is 101101. Convert this number into its decimal equivalent.

Solution:

1	0	1	1	0	1
1×2^5	0×2^4	1×2^3	1×2^2	0×2^1	1×2^0

32 + 0 + 8 + 4 + 0 + 1 = 45

Therefore, $(101101)_2 = (45)_{10}$

Fractional binary number to fractional decimal number

The decimal equivalent of a binary fraction is determined by multiplying each digit in the fraction by 2^{-1}, 2^{-2} 2^{-3}, 2^{-4}, ... etc., beginning with the first digit after the binary point. The products are then added to get the decimal fraction. A few examples will illustrate this procedure

Example 16.4 Convert the fractional binary number 0.1011 into decimal fraction.

Solution:

The number 0.1011 is converted into decimal number as

$$\begin{aligned}&1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}\\&= 0.5 + 0 + 0.125 + 0.0625\\&= 0.6875\end{aligned}$$

Example 16.5 Convert the fractional binary number 0.1101 into decimal fraction

$$
\begin{aligned}
&1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3} + 1 \times 2^{-4} \\
&= 0.5 + 0.25 + 0 + 0.625 \\
&= 0.\ 8125
\end{aligned}
$$

16.2.4 Conversion of Decimal to Binary

To convert a decimal number into a binary equivalent number, the decimal number is expressed as a sum of ascending power of 2. The positional values of the binary number system are

$$
\begin{array}{ccccccccc}
\cdots & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
 & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1
\end{array}
$$

Suppose we are to find the binary equivalent of the decimal number 45. The number 45 is arrived at by adding 32 + 8 + 4 + 1. The 1S and 0S are placed as shown

$$
\begin{array}{ccccccccccc}
2^5 & & 2^4 & & 2^3 & & 2^2 & & 2^1 & & 2^0 \\
32 & + & 0 & + & 8 & + & 4 & + & 0 & + & 1 = 45
\end{array}
$$

The binary number is 101101

The binary equivalent of 19 is arrived as

16 + 2 + 1

$$
\begin{array}{cccccccc}
2^4 & & 2^3 & 2^2 & & 2^1 & & 2^0 \\
16 & + & 0 + & 0 & + & 2 & + & 1 = 19
\end{array}
$$

The binary number is 10011

The binary equivalent of decimal numbers 7, 9, 11, respectively are

Example 1

7 = 4 + 2 + 1
$2^2 + 2^1 + 2^0$

The binary number of 7 is 111

Example 2

9 = 8 + 0 + 0 + 1
$2^3 + 0 + 0 + 2^0$

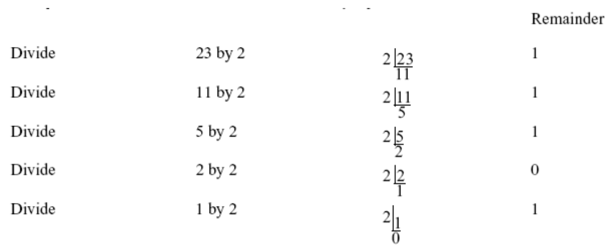
The binary number of 9 is 1001

Example 3

11 = 8 + 0 + 2 + 1

We can use an alternative method of converting a decimal number into a binary equivalent by dividing the decimal number progressively by 2 till the quotient is zero. The remainders of the successive divisions, expressed in the reverse order, gives the binary equivalent number. A few examples will illustrate the procedure.

Example 16.6 Convert decimal 23 into its binary equivalent number.

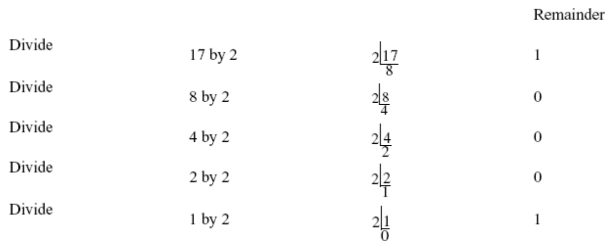


			Remainder
Divide	23 by 2	2 | 23 11	1
Divide	11 by 2	2 | 11 5	1
Divide	5 by 2	2 | 5 2	1
Divide	2 by 2	2 | 2 1	0
Divide	1 by 2	2 | 1 0	1

Now we will arrange the remainders in the reverse order, i.e., from bottom upwards. The binary equivalent number is 10111

Example 16.7 Determine the binary equivalent of decimal number 17.

Solution:

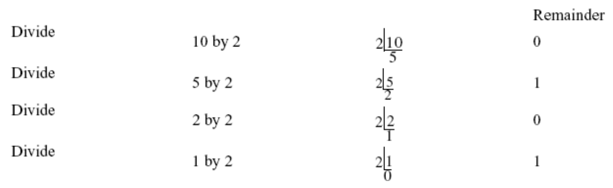


			Remainder
Divide	17 by 2	2 | 17 8	1
Divide	8 by 2	2 | 8 4	0
Divide	4 by 2	2 | 4 2	0
Divide	2 by 2	2 | 2 1	0
Divide	1 by 2	2 | 1 0	1

The remainder when put in reverse order give the binary equivalent a 10001.

Example 16.8 Determine the binary equivalent of decimal number 10.

Solution:



			Remainder
Divide	10 by 2	2 | 10 5	0
Divide	5 by 2	2 | 5 2	1
Divide	2 by 2	2 | 2 1	0
Divide	1 by 2	2 | 1 0	1

Conversion of fractional decimal number to fractional binary number

When the decimal number is a fraction, the conversion can be carried out through the following steps.

First, multiply the decimal fraction by 2. The result contains an integer part and a fractional part. Separate the integer part and write it in a column. The fractional part becomes the new fraction. Repeat these steps till the fractional part becomes zero or the desired number of binary places are obtained. The 1s and 0s separated placed together gives the required fractional binary number. A few examples will illustrate the procedure.

Example 16.9 Convert decimal 0.8125 into its equivalent binary number.

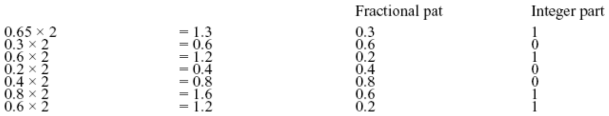
Solution:

		Fractional part	Integer part
0.8125 × 2	= 1.6250	0.6250	1
0.6250 × 2	= 1.2500	0.2500	1
0.2500 × 2	= 0.5000	0.5000	0
0.5000 × 2	= 1.0000	0	1

The binary decimal number is 0.1101

Example 16.10 Convert 0.65 into its binary fraction.

Solution:



		Fractional pat	Integer part
0.65 × 2	= 1.3	0.3	1
0.3 × 2	= 0.6	0.6	0
0.6 × 2	= 1.2	0.2	1
0.2 × 2	= 0.4	0.4	0
0.4 × 2	= 0.8	0.8	0
0.8 × 2	= 1.6	0.6	1
0.6 × 2	= 1.2	0.2	1

In the last step we get back 0.2 as the fractional part. To get the approximate result we can terminate the process at this stage. Thus, the approximate binary fraction is 0.1010011.

16.2.5 Binary Addition

The binary addition is performed by using the following rules.

- 0 + 0 = 0
- 1 + 0 = 1
- 0 + 1 = 1
- 1 + 1 = 10 (decimal equivalent of 10 is 2)

Examples

- Add $$\begin{array}{r} 10 \\ +11 \\ \hline 101(=5) \end{array} \qquad \begin{array}{c} 10+11=101 \\ 2+3=5 \end{array}$$

$111(=7)$

Here the first column gives 1 + 1 = 10. We put 0 below this column and add the carry 1 to the second column. In the second column 1 + 1 = 10. We add the carry 1 with 0 to get 1 and carry 1 to the third column. In the third column 1 + 1 = 10. We add the carry 1 to 0 to get 1 and the carry 1 is placed in the fourth column.

- Add $$\begin{array}{r} 10001(=17) \\ +111(=\ 7) \\ \hline 11000(=24) \end{array}$$

First column 1 + 1 = 10, place 0 and carry 1

Second column, 1 + 0 = 1, 1 + 1 = 10, place 0 and carry 1

Third column, 1 + 0 = 1, 1 + 1 = 10, place 0 and carry 1

Fourth column, 1 + 0 = 1, place 1 and carry 0

Fifth column, 1 + 0 = 1, place 1

16.2.6 Binary Subtraction

The following rules are used for binary subtraction

- 0 – 0 = 0
- 1 – 0 = 1
- 1 – 1 = 0
- 10 – 1 = 1 (decimal equivalent of 10 is binary 2)

Examples

- Subtract $$\begin{array}{r} 1011(=11) \\ -1000(=\ 8) \\ \hline 0011(=3) \end{array}$$
- Subtract $$\begin{array}{r} 110(=6) \\ -11(=3) \\ \hline 011(=3) \end{array}$$

In the second example, the first column is 10 – 1 = 1 (after borrowing 1 from the second column). The borrow is carried and added to the lower number of second column to get 1 + 1 = 10. Therefore, the second column gives 1 – 0 = 1 and the third column gives 1 – 1 = 0.

16.2.7 Binary Multiplication

Binary multiplication is carried out using the following rules.

- 0 × 0 = 0
- 1 × 0 = 0
- 0 × 1 = 0
- 1 × 1 = 1

Example

$$
\begin{array}{cccccccl}
 & & & & 1 & 1 & 1 & (=7) \\
 & & & & 1 & 0 & 1 & (=5) \\
\cline{4-7}
 & & & & 1 & 1 & 1 & \\
 & & & 0 & 0 & 0 & \times & \\
 & & 1 & 1 & 1 & \times & \times & \\
\cline{2-7}
 & 1 & 0 & 0 & 0 & 1 & 1 & (=35) \\
 & \multicolumn{2}{c}{1+1} & 1+1 & 1+1 & 1+0 & 1+0 & \\
 & \multicolumn{2}{c}{=10} & =10 & =10 & =1 & =1 & \\
 & & & \text{carry } 1 & \text{carry } 1 & & &
\end{array}
$$

16.3 OCTAL NUMBER SYSTEM

Octal system has a base of 8 counting from 0 to 7. To convert a decimal number into octal equivalent of decimal number we divide the integer repeatedly by 8 till the quotient is zero. The remainders written in reverse order gives the octal equivalent number.

For example let us find the octal equivalent of decimal number 756.

756 ÷ 8 = 94	Remainder = 4
94 ÷ 8 = 11	Remainder = 6
11 ÷ 8 = 1	Remainder = 3
1 ÷ 8 = 0	Remainder = 1

The octal equivalent of 756 is 1364.

To convert an octal to binary each octal symbol is replaced by a 3-bit binary equivalent as shown

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Conversion of binary number into an equivalent octal number can be done in the following way.

Example (i)	
	Convert binary 11001 into octal equivalent
	The binary number is grouped as 011 001
	Replacing each group by its octal equivalent we get the number 31
Example (ii)	
	Convert binary 0.111 001110 into its equivalent octal
	The binary number is grouped into 3-bit form as
	0.111 001 110
	The equivalent is 0.716

To convert an octal number into its decimal equivalent the following steps be followed.

Consider the octal number from the extreme right as the coefficient of the ascending power of 8, the starting power of 8 at the extreme right has to be taken as zero. If the octal number has a fraction, the fractional numbers have to be considered as the coefficient of ascending power of 8^{-1}.

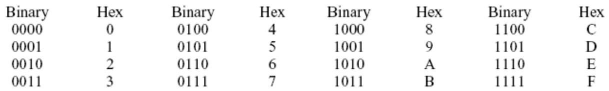
Examples

$$\begin{aligned}\text{(i) } 415.2 &= 4 \times 8^2 + 1 \times 8^1 + 5 \times 8^0 + 2 \times 8^{-1}\\ &= 256 + 8 + 5 + 0.25\\ &= 269.25\end{aligned}$$

$$\begin{aligned}\text{(ii) } 732.14 &= 7 \times 8^2 + 3 \times 8^1 + 2 \times 8^0 + 1 \times 8^{-1} + 4 \times 8^{-2}\\ &= 448 + 24 + 2 + 0.125 + 0.0625\\ &= 474.1875\end{aligned}$$

16.4 HEXADECIMAL NUMBER SYSTEM

In hexadecimal system we use 16 symbols, viz 0 to 9 and the capital letters A B C D E F for 10 to 15. Hence, the hexadecimal number sysend for integral numbers and from the left end for fractions. Each group is replaced by its equivalent.



Binary	Hex	Binary	Hex	Binary	Hex	Binary	Hex
0000	0	0100	4	1000	8	1100	C
0001	1	0101	5	1001	9	1101	D
0010	2	0110	6	1010	A	1110	E
0011	3	0111	7	1011	B	1111	F

Examples

(i) Convert 1111 1110 0011 into hexadecimal equivalent

Group the binary as (from right to left) 1111 1110 0011

Replace the group by their equivalent as FE3

(ii) Convert 1111 10001.100 1100 1101 into hexadecimal equivalent

Group the binary as

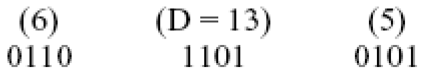
0001 1111 0001 . 1001 1001 1010

Replace the groups by their equivalent to get

1F1.99A

(iii) Convert Hex number 6D5 into its binary equivalent.

Starting from right we replace binary equivalent of 5, D, and 4 as



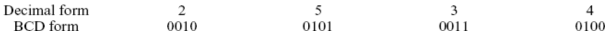
(6)	(D = 13)	(5)
0110	1101	0101

It is seen from the above examples that conversion between Hex and binary is very quick.

For conversion of decimal number into binary equivalent, or useful system, a binary coded decimal system, or BCD system is often used.

It is a code in which individual decimal digits are replaced by a group of 4 binary bits in a row. The weights of the 4 bits from left to right of the binary string are 8,4,2, and 1, respectively. The BCD code is also called eight four two one code.

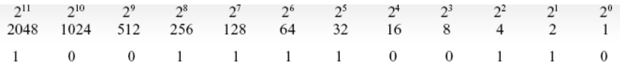
Example Express 2534 in BCD form.



Decimal form	2	5	3	4
BCD form	0010	0101	0011	0100

Thus, $(2534)_{10}$ = $(0010010100110100)_{BCD}$.

It is to be noted that this system is totally different from the method of conversion of decimal numbers into binary numbers.



2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
2048	1024	512	256	128	64	32	16	8	4	2	1
1	0	0	1	1	1	1	0	0	1	1	0

2048 + 0 + 0 + 256 + 128 + 64 + 32 + 0 + 0 + 4 + 2 + 0 = 2534

Therefore, $(2534)_{10} = (100111100110)_2$

16.4.1 Application of Binary Numbers in Computers

The computer works with binary numbers, and hence all numbers, letters, symbols, etc. are to be converted into binary equivalents. Computers are made using very-large-scale integrated digital circuits. The main circuit element being transistors which work in two modes, i.e., on or off, i.e., conducting or non conducting. These transistors, in fact, work like a switch. The term binary or bistable means two states of operation either conducting or non-conducting giving rise to voltage level high or voltage level low, i.e., either 1 or 0. The computer works on the basis of binary logic circuits made of logic gates. Various logic gates are discussed in the following section.

16.5 LOGIC GATES

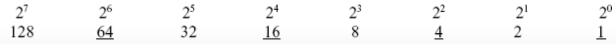
A logic gate is a simple device used to make digital integrated circuits. Diodes and transistors are used to perform switching functions in logic gates. Logic gates basically have one or more inputs and only one output. The output of the gate depends upon the way the inputs are applied and they are named as AND gate, OR gate, NOT gate, etc. Logic gates may be classified into two categories; combinational type (with no memory) and sequential type (with memory).

In digital circuits or systems there are two possible states or conditions. A switch may be closed or open, i.e., on or off. These two conditions may be represented by 1 and 0, respectively. The two logic levels, i.e., 1 and 0 represent, respectively, voltage high or voltage low. Logic systems can be positive or negative. In positive logic voltage high is represented by 1 and voltage low is represented by 0. In negative logic, voltage high is 0 and voltage low is 1. Binary number system may be used to represent the states. In a digital system, analog quantities are first converted to digital forms using the binary code. After the signal is processed, it is again converted into analog form for display.

The possible states or conditions in digital systems are expressed as 1 and 0. The closed switch or the existence of a signal may be called 1 and open switch or non existence of a signal may be called 0. 1 and 0 are referred to as binary numbers. As mentioned earlier, in decimal system we have 0 to 9 as ten numbers. Using these numbers we can represent any number, e.g., year 2009 is a representation of

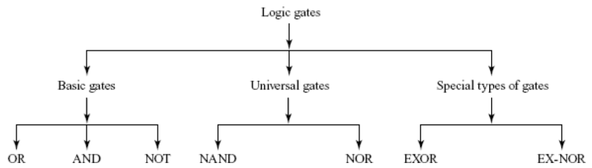
$2 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 9 \times 10^0$

Here, the base is 10 as we use 10 digits. In the binary system we only use 1 and 0 with 2 as the base. The number 82 is represented as



2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Logic circuits or logic gates provide an output of logical 1 or logical 0 with a certain logical input. The logic gates are classified into two categories viz three logic gates and two universal gates. There are also two special types of gates. These are shown below.



16.5.1 NOT Gate

A NOT gate works as an inverter. It has a single input and a single output. The output signal is the negative of the input signal. The output is not the same as input, the output negates the input. If input is A, the output is A (A-bar).

Figure 16.1 shows a transistor circuit. When input A is 0 V, the transistor is off because V_{BE} = 0. The output voltage is the same as V_{CC}, i.e., 5 V which is referred to as 1. If input A is 1, i.e., 5 V, the transistor goes into saturation and the output w ill be low which is referred to as 0. The logic symbol and truth table for a NOT gate have also been shown in Fig. 16.1 (b) and (c), respectively.

Thus, the NOT gate is a logic gate that gives an output that is opposite to the state of its input.

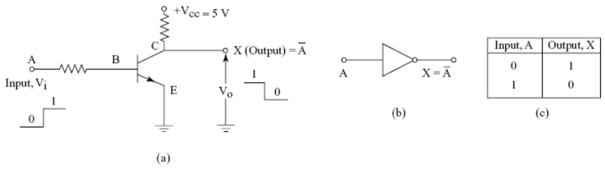


Figure 16.1 (a) Circuit for a positive logic NOT gate; (b) symbol of a NOT gate; (c) truth table of a NOT circuit

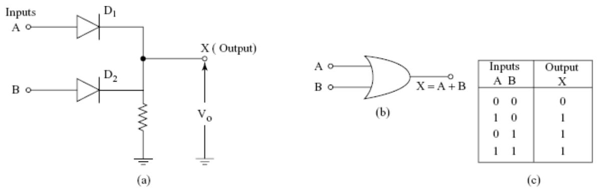


Figure 16.2 (a) Logic OR gate circuit using diodes; (b) symbol of an OR gate; (c) truth table

16.5.2 OR Gate

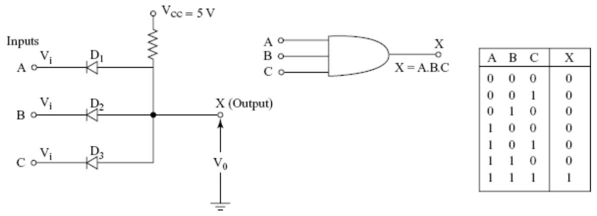
The OR gate provides an output 1 when one of its inputs is in 1 state.

Output will occur when there is input in any one of the inputs A or B or at both. If A = B = 0, then the two diodes D_1 and D_2 will be reverse biased, and hence X output, X will be zero. If A = 1 and B = 0 or if A = 0 and B = 1, i.e., when any one of the inputs is high, the output, X will be 1. If A = 1 and B = 1, i.e., when both the diodes output X will be 1. If A = 1 and B = i.e., when will be in forward biased Condition, the output X will be 1.

Although two diodes have been used to show an OR gate, in actual practice OR function is obtained using a transistor–transistor logic (TTL). The 2-input OR gates is available in the TTL 74 xx series. 7432 is a 2-input TTL OR gate integrated circuit. The IC chip has four OR gates.

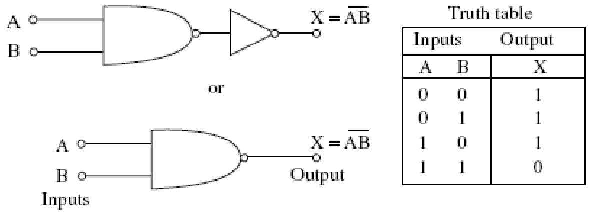
16.5.3 AND Gate

This gate has two or more inputs but only one output. The AND gate is a logic gate that gives an output of 1 only when all of its inputs are 1. Thus, its output will be 0 when at least one of its inputs is 0. Figure 16.3 shows a 3-input AND gate circuit using three diodes. The AND gate may be considered equivalent to a circuit in which a number of switches are connected in series. Only when all the switches are on, will there be an output.



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

Figure 16.3 (a) Logic AND gate circuit; (b) symbol; (c) truth table



Truth table

Inputs		Output
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Figure 16.4 Standard symbol and truth table of a NAND gate

As in Fig. 16.3 (a), when inputs A = B = C = 0, all the diodes will be forward biased and will conduct, the output will be zero.

When A = B = C = 1, all the diodes will be reverse biased and will remain off. The output will be high, i.e., equal to 1.

When any of the inputs is high, the corresponding diode will

There are such four AND gates in the IC chip.

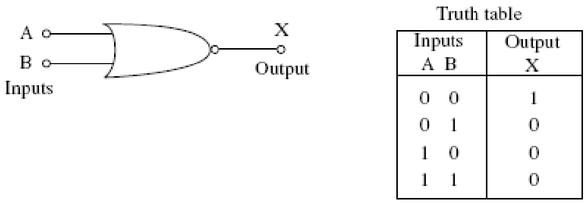
16.5.4 NAND Gate

The NAND gate is an AND gate with a NOT gate at its end. For the same combination of inputs, the output of a NAND gate will be opposite to that of an AND gate. The combination of AND and NOT gate can be made using a diode transistor logic circuit with minor modification. The symbol of a 2-input NAND gate along with its truth table have been shown in Fig. 16.4.

16.5.5 NOR Gate

The NOR gate is an OR gate with a NOT gate at its end. For the same combination of inputs, the output of a NOR gate will be opposite to that of the output of a NOR gate. The symbolic representation and truth table of a NOR gate have been shown in Fig. 16.5.

A truth table describes the behaviour of a logic gate. It shows the value of the output for every possible combination of inputs. The simplest form of electronic logic for AND and OR gates can be built using diodes. But a NOT gate cannot be made using diodes. To build a complete logic system a resistor–transistor logic, or RTL are used. RTL gates can be cascaded indefinitely to produce complex logic function. RTL gates were used in early integrated circuits. For achieving higher speed of functioning, diode–transistor logic or DTL were used. To reduce the space required, transistor–transistor logic (TTL) was created in the fabrication of ICs. Complementary metal oxide semiconductor (CMOS) logic has been replacing TTL logic to reduce size and power consumption. For small-scale logic, TTL 7400 series and CMOS 4000 series of ICs are used. Gradually, the fixed function logic gates mentioned earlier are being replaced by programmable logic devices, which permit a large number of fixed function logic gates packed into an integrated circuit.



Truth table

Inputs A B	Output X
0 0	1
0 1	0
1 0	0
1 1	0

Figure 16.5 Standard symbol and truth table of a NOR gate

16.6 BOOLEAN ALGEBRA

In 1854, George Boole developed a new form of algebra which has come to be known as Boolean algebra. Boolean algebra is being applied for the solution of logical problems.

The operation of logic gates in relation to one another may be represented and analysed using a branch of mathematics called Boolean algebra. Expressions used in Boolean algebra are called Boolean expressions. In Boolean algebra only capital letters like A, B, C, D, etc. are used.

The OR operation is represented by addition because the only way to obtain a result of an addition operation equal to 0 is to make all inputs equal to 0, which basically describes an OR operation. The Boolean expression for A or B is A + B.

NOT operation means inversion. The Boolean expression for NOT operation is A = NOT(A) = A. NAND operation is an AND operation followed by a NOT operation. The NOR operation is an OR operation followed by a NOT operation. Boolean expressions for the three basic gates are given below.

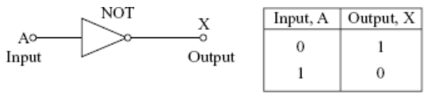
16.6.1 Boolean Expressions

Boolean expressions are equivalent expressions of logic states of gates. For example, the Boolean expressions for the basic gates are

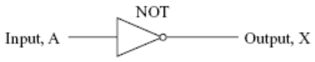
a NOT gate with input A and output C, C = NOT A, or C = A

The NOT gate

The output of a NOT gate is the inverse of its input. If the input is low, the output is high, and if the input is high, the output is low. NOT gates can have only one input and one output. A NOT gate is also called an inverter. The symbolic representation along with its Boolean expression and truth table are shown below.



Input, A	Output, X
0	1
1	0



X NOT A

or X = A

0 1

1 0

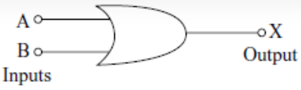
The OR gate

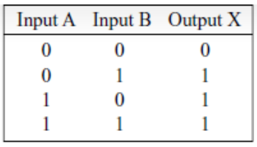
The output of an OR gate is high if any of its inputs is high. An OR gate has two or more inputs and one output. The Boolean expression for an OR gate is written as

X = A or B

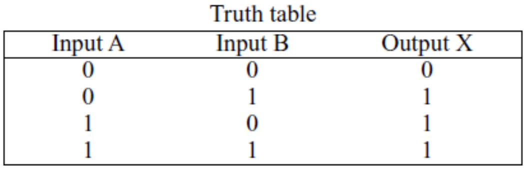
or, X = A + B

The symbol and truth tables are shown below.





Input A	Input B	Output X
0	0	0
0	1	1
1	0	1
1	1	1

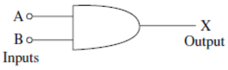


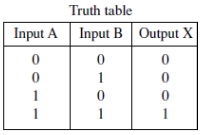
Truth table

Input A	Input B	Output X
0	0	0
0	1	1
1	0	1
1	1	1

The AND gate

The output of an AND gate is high if all inputs are high, otherwise the output is low. The AND gate has two or more inputs and one output. The Boolean expression, truth table, and symbolic representation of an AND gate are shown below.

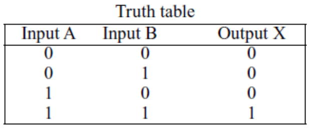




Truth table

Input A	Input B	Output X
0	0	0
0	1	0
1	0	0
1	1	1

X = A and B
or, X = A.B



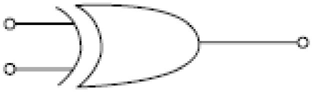
Truth table

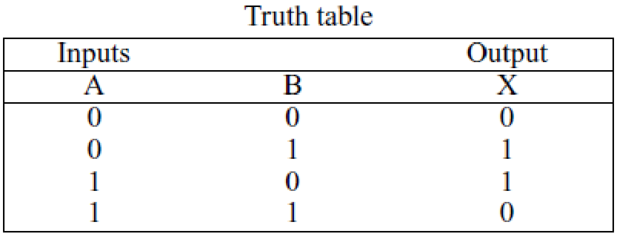
Input A	Input B	Output X
0	0	0
0	1	0
1	0	0
1	1	1

The EXOR gate

The EXOR gate stands for Exclusive OR gate. The EXOR gate is a logic gate that gives an output 1 when only one of its inputs is 1. The symbol and truth table of an EXOR gate have been shown in Fig. 16.6.

Symbol





Truth table

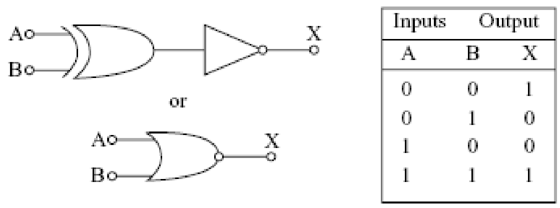
Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Figure 16.6 Standard symbol and truth table of an EXOR gate

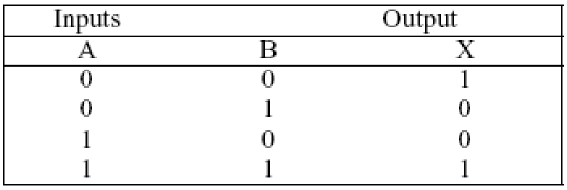
gate as such is also called an inclusive OR gate whereas EXOR gate is an exclusive OR gate.

EX-NOR gate

The operation of an EX-NOR gate is just opposite to the operation of an EX OR gate. Inputs are two or more and output is one. The output will be high for similar types of inputs only. The symbol and truth table have been shown.



Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1



Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Thus, we have seen seven types of gates. Three basic gates are NOT, OR, and AND gates and the combinational gates are NAND, NOR, EXOR and EX-NOR gates.

NAND and NOR gates are called the universal gates as all other gates can be created from a suitable network of just NAND or just NOR gates.

A logic gate performs a logical operation on one or more logic inputs and produces a single logic output. Logic gates are primarily implemented using resistors, diodes, and transistors.

16.7 DE MORGAN'S THEOREM

The theorem states how an AND operation can be converted into an OR operation, as long as a NOT operation is available. The theorem is usually expressed in two equations as

$$\overline{A+B} = \overline{A} \bullet \overline{B}$$

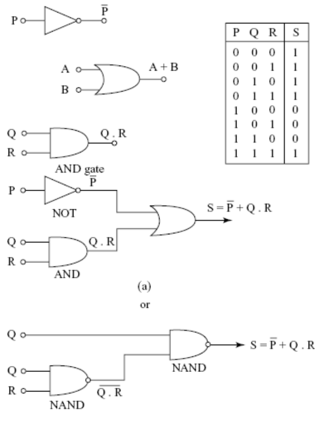
$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

and

De Morgan's theorem has a practical use in digital electronics. Any logic circuit may be implemented by the basic gates described earlier. A designer may eliminate the use of more ICs by substituting gates with the equivalent combination of other gates wherever possible. De Morgan's theorem basically implies that any Boolean

Combinational logic circuits are constructed by interconnecting different logic gates. A combinational circuit with its truth table has been shown in Fig. 16.7. The same circuit can also be implemented using NAND gates only as shown in Fig. 16.7 (b).

The logic gates and related Boolean equations are shown in tabular form in Table 16.1.



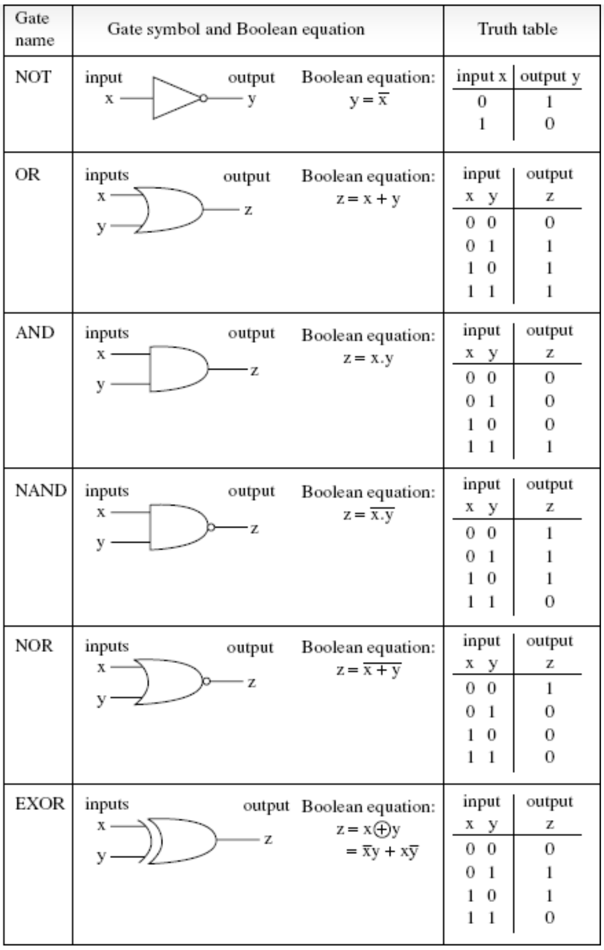
P	Q	R	S
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Output $S = \overline{P \bullet \overline{Q \bullet R}}$

$= \overline{P} + Q \bullet P$ (according to De Morgan's theorem)

Figure 16.7 (a) Truth table for a combinational circuit; (b) NAND gate implementation of the same logic

Table 16.1 Logic Gates and Their Truth Tables



Gate name	Gate symbol and Boolean equation	Truth table
NOT	input x, output y Boolean equation: $y = \overline{x}$	input x | output y 0 | 1 1 | 0
OR	inputs x, y, output z Boolean equation: $z = x + y$	input x y | output z 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
AND	inputs x, y, output z Boolean equation: $z = x.y$	input x y | output z 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
NAND	inputs x, y, output z Boolean equation: $z = \overline{x.y}$	input x y | output z 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR	inputs x, y, output z Boolean equation: $z = \overline{x + y}$	input x y | output z 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
EXOR	inputs x, y, output z Boolean equation: $z = x \oplus y$ $= \overline{x}y + x\overline{y}$	input x y | output z 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Example 16.11 Write the Boolean expression and truth table for the logic gate circuit shown in Fig. 16.8.

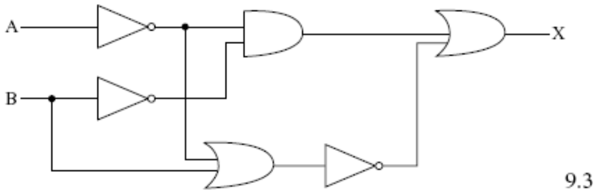


Figure 16.8:

Solution:

The Boolean expression is developed as shown below.

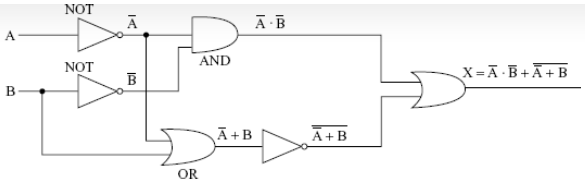


Figure 16.9:

The Boolean expression is

$$X = \bar{A}.\bar{B} + \overline{\bar{A} + B}$$
$$= Y + Z$$

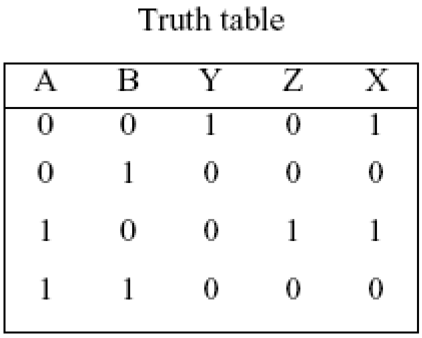
where

$$Y = \bar{A}.\bar{B}$$

and

$$Z = \overline{\bar{A} + B}$$

Now let us write the truth table. We first write the various combinations of 0 and 1 for the two inputs, A and B. There are $2^N = 2^2 = 4$ combinations. For each row of inputs, we will calculate Y and Z, respectively. Then we would add Y and Z, to get X.



Truth table

A	B	Y	Z	X
0	0	1	0	1
0	1	0	0	0
1	0	0	1	1
1	1	0	0	0

Example 16.12 For the logic circuit shown in Fig. 16.10, write the Boolean expression and construct the truth table.

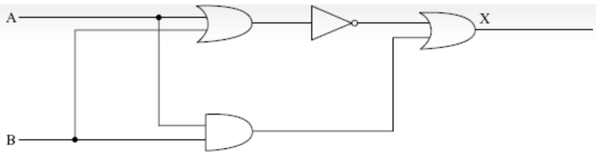


Figure 16.10:

Solution:

The Boolean expression is developed from Fig. 16.11 as

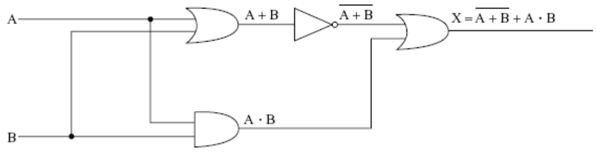


Figure 16.11:

The Boolean expression is

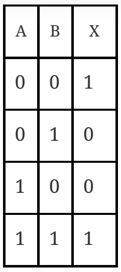
$$X = \overline{A + B} + A.B$$
$$= Y + Z$$
$$\text{where } Y = \overline{A + B} \text{ and } Z = A.B$$

For constructing the truth table, we will first write the various combinations of the inputs at A and B in the form of binary numbers (the total number of possible combinations being Z^N where N is the number of inputs). We then calculate Y and Z, and then X is calculated.

Truth table

A	B	Y	Z	X = Y + Z
0	0	1	0	1
0	1	0	0	0
1	0	0	0	0
1	1	0	1	1

Boolean expression for any logic function can be obtained from the truth table. This is done by taking the sum of all terms which correspond to all those combinations of inputs for which the output attains



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

The first row and fourth row have to be considered where output is high, i.e., 1.

Inputs A and B are to be considered 1. The complement $\overline{A}$ and $\overline{B}$ therefore 0.

For row 1

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

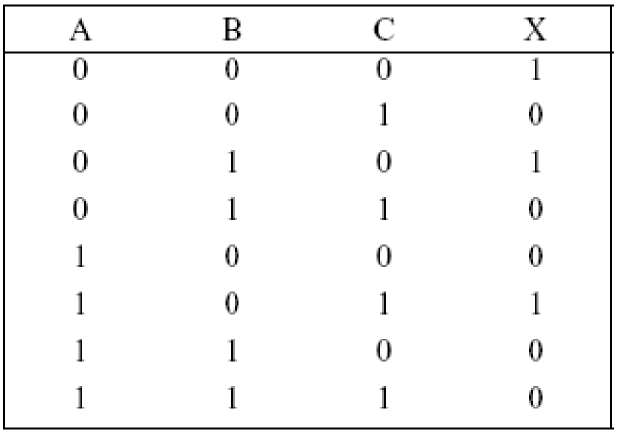
For row 4

Z = A.B

The Boolean expression for the output is

$$X = Y + Z = \overline{A + B} + A.B$$

Example 16.13 The truth table for three variable inputs and single output logic circuit is shown below. Note that for three inputs the combinations are 2^3 = 8. Construct the logic circuit using AND and OR gates.



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

Solution:

By writing the sum of all terms for which the output attains a high

$$X = \overline{ABC} + \overline{A}B\overline{C} + A\overline{B}C$$

Using the above Boolean expression, the logic circuit is drawn as in Fig. 16.12.

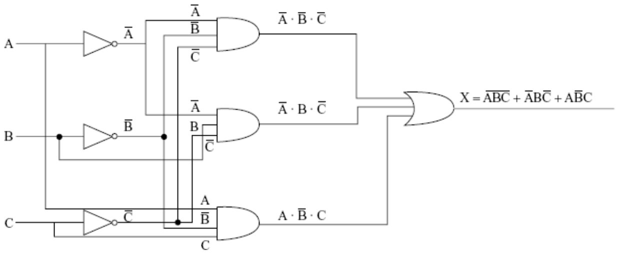

A
B
C
$\overline{A}$
$\overline{B}$
$\overline{C}$
$\overline{A}$
$\overline{B}$
$\overline{C}$
$\overline{A}$
B
$\overline{C}$
A
$\overline{B}$
C
$\overline{A} \cdot \overline{B} \cdot \overline{C}$
$\overline{A} \cdot B \cdot \overline{C}$
$A \cdot \overline{B} \cdot C$
$X = \overline{ABC} + \overline{A}B\overline{C} + A\overline{B}C$

Figure 16.12

Example 16.14 Draw the logic circuit for the Boolean expression expressed in two forms. Also write the truth table

$$Y = \overline{A}\,\overline{C} + B\,\overline{C} \qquad \text{(i)}$$

and

$$Y = \overline{C}\,(\overline{A} + B) \qquad \text{(ii)}$$

Solution:

for form (i)

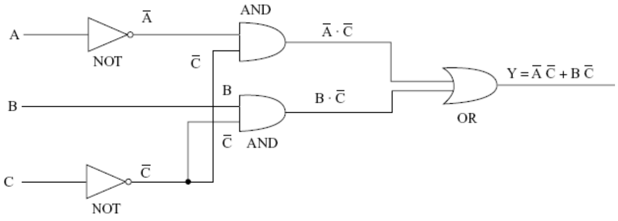


Figure 16.13:

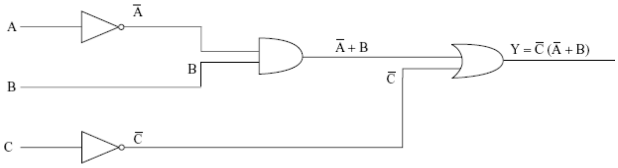


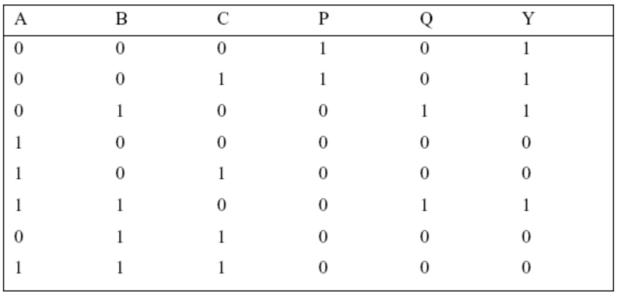
Figure 16.14:

The truth table for

$$Y = \overline{AC} + B\overline{C}$$

= P + Q

is shown as



A	B	C	P	Q	Y
0	0	0	1	0	1
0	0	1	1	0	1
0	1	0	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	1
0	1	1	0	0	0
1	1	1	0	0	0

16.9 SIMPLIFICATION OF BOOLEAN EXPRESSIONS USING DE MORGAN'S THEOREM

De Morgan's theorem are very useful in simplifying expressions in which a product or a sum of variables is inverted. The De Morgan's theorem is reproduced as two forms

(i) Complement of a sum is equal to the product of complements, e.g.

$$\overline{A+B+C} = \overline{A}.\overline{B}.\overline{C}$$

(ii) Complement of a product is equal to sum of complements, e.g.

$$\overline{A.B.C} = \overline{A}+\overline{B}+\overline{C}$$

We will include some more important theorems which help in the simplification of Boolean logic expression.

A + AB = A (here term A is present in both the terms, and hence the other term is neglected)

$$AB + ABC + A\overline{B} = AB + A\overline{B}$$

= A

AC + ABC + ACD = AC, (here the factor AC is present in all the terms, and hence the other terms are neglected)

Consensus theorem: If in a Boolean expression, one term contains a variable and the second term contains the complement of this variable, then in the expression a third term can be added without effecting the value of the expression, e.g. if X=AC+B$\overline{C}$ where

$\overline{C}$ is the complement of C. We can add another term in the expression as

$$X = AC + B\overline{C} + AB$$

To prove we write

$$\begin{aligned} X &= AC + B\overline{C} + AB(C + \overline{C}) \\ &= AC + B\overline{C} + ABC + AB\overline{C} \\ &= AC(B+1) + B\overline{C}(A+1) \\ &= AC + B\overline{C} \end{aligned}$$

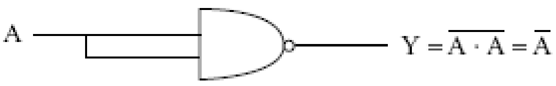
16.10 UNIVERSAL GATES

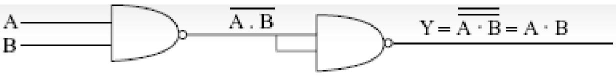
As mentioned earlier NAND and NOR gates are known as universal gates because they can be used to build any logic circuit. The logic function of NOT, OR, and AND can be performed using either NAND or NOR gates. First we shall see how a NAND gate, functions as NOT, AND, and OR gates respectively.

A NAND gate with its two inputs tied together is a single input NAND gate.

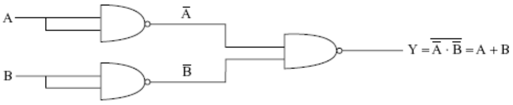
16.10.1 Use of NAND Gate to Form the Three Basic Gates

(i) NAND gate as NOT gate





(iii) NAND gate as OR gate



(∵ According to De Morgan's second law. $\overline{\bar{A}.\bar{B}} = \bar{\bar{A}} + \bar{\bar{B}} = A + B$)

Figure 16.15 Three basic gates using NAND gates

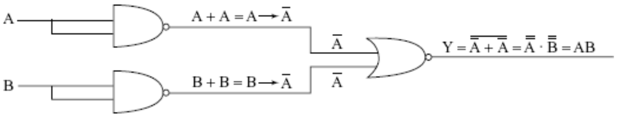
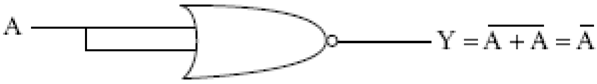

A
B
$A + A = A \longrightarrow \bar{A}$
$B + B = B \longrightarrow \bar{A}$
$\bar{A}$
$\bar{A}$
$Y = \overline{\bar{A} + \bar{A}} = \bar{\bar{A}} \cdot \bar{\bar{B}} = AB$

Figure 16.16 Three basic gates using NOR gates

16.10.2 Use of NOR Gate to Form the Three Basic Gates

Now we shall see the use of a NOR gate to form the three basic gates.

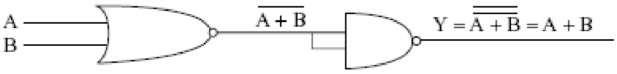
(i) NOR gate as NOT gate



If all the inputs of the NOR gate are connected together, we obtain a NOT gate as shown above.

(ii) NOR gate as OR gate

The OR operation is obtained by a NOR gate followed by a single input NOR gate as shown.



(iii) NOR gate as AND gate

The AND operation by NOR gates is achieved as shown. The inputs at A and B are inverted by 1-input NOR gates. The inverted inputs are old to a NOR gate.

that time. The previous records of inputs does not effect the output at all.

The output of a flip-flop is either a low voltage (0) or a high voltage (1). The output remains in one of these states indefinitely unless an external trigger is applied to change that state. Thus a flip-flop will retain or remember a state indefinitely after the removal of the input trigger. A flip-flop is considered a storing device of 1-bit memory. Since a flip-flop will have two stable states (0 or 1), it is also referred to as *bistable multivibrator*.

In most flip-flop circuits the change of state of the flip-flop is of a definite rate, and hence they are of clocked type.

16.11.1 RS Flip-flop

The RS flip-flop is used to temporarily hold or store information until it is required. A single RS circuit will store one binary digit, i.e., either 1 or 0. For storing a six digit binary number, six RS flip-flops will be required. As shown in Fig. 16.17, a RS flip flop which is a one-bit memory device has two inputs R and S. R stands for 'RESET' and S stands for 'SET'. The two output terminals are marked Q and $\overline{Q}$ One input will set the device and another will reset the device back to its original state. The output will either be at logic level 1 or 0 depending upon the set/reset condition. The simplest way to make a 1-bit set/reset RS flip-flop is to connect together a pair of NAND gates which are cross-coupled as shown in Fig. 16.17. The output $\overline{Q}$ is the inverse or complement of Q.

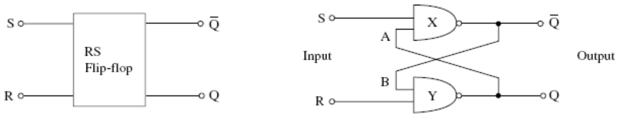
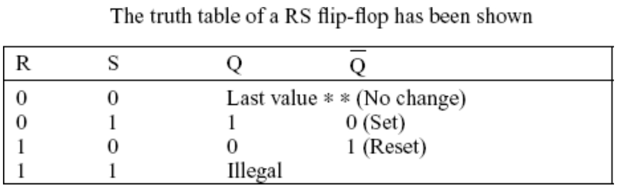


Figure 16.17 RS flip-flop using NAND gates



The truth table of a RS flip-flop has been shown

R	S	Q	$\overline{Q}$
0	0	Last value * *	(No change)
0	1	1	0 (Set)
1	0	0	1 (Reset)
1	1	Illegal	

Application of a positive trigger at the input S will be setting the flip-flop. The output at Q will be 1 and at $\overline{Q}$ will be 0. The application of positive trigger at R will be resetting the flip-flop. The output as $\overline{Q}$ will be 1 and at Q will be 0.

For the set state, the input at R is at logic level 0 and input at S is at logic level 1, the output of NAND gate Y has atleast one of its inputs

For reset state, R = 1 and S = 0. The NAND gate X will have one of its inputs at level 0, and hence output at $\overline{Q}$ is 1. The flip-flop will function as follows.

- (a) For R = 0, S = 0, i.e., with no input at the input terminals, the output will retain its previous state, i.e., output at Q may be 0 or 1.
- (b) For R = 0, S = 1, i.e., trigger input is provided in the set input terminal, the output at Q will be 1.
- (c) For R = 1, S = 0, Q = 0
- (d) For R = 1, S = 1 the output is forced to become 0 and 1 simultaneously. Such input condition is called illegal or invalid forbidden input condition.

An RS flip-flop can also be created by two single input NAND gates and two 2-input NAND gates connected in a fashion as shown in Fig. 16.18.

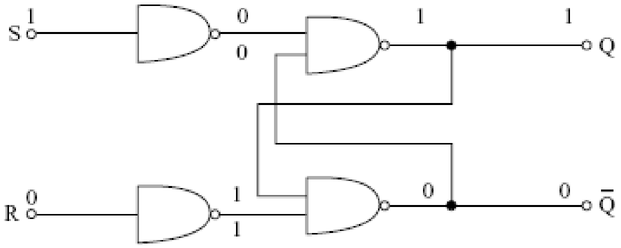
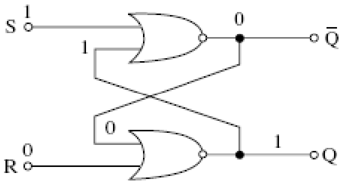


Figure 16.18 RS flip-flop using combination of 1-input and 2-input NAND gates

Flip-flop can also be formed by using 2-input NOR gates as shown in Fig. 16.19.



S	R	Q	Q̄
0	0	No change	
0	1	0	1 Reset
1	0	1	0 Set
1	1	0	0
		In valid	

Figure 16.19 Flip-flop using NOR gates

The circuit works in a similar way as in the case of NAND gate circuit. When both the inputs are at logic level 1, an invalid condition is created.

16.11.2 Gated or Clocked RS Flip-flop

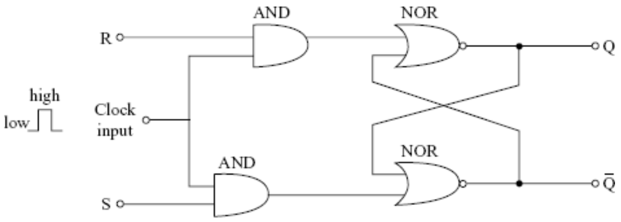
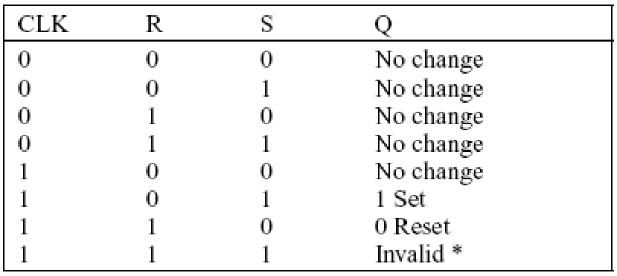


Figure 16.20 Clocked RS flip-flop

When the clock input is high (i.e., 1) the flop-flip will be set for R = 0 and S = 1 and will be reset for R = 1 and S = 0 as has been shown in the truth table.

Truth table for a clocked RS flip-flop



CLK	R	S	Q
0	0	0	No change
0	0	1	No change
0	1	0	No change
0	1	1	No change
1	0	0	No change
1	0	1	1 Set
1	1	0	0 Reset
1	1	1	Invalid *

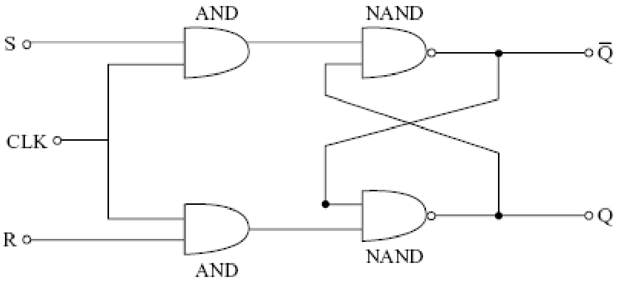


Figure 16.21 Clocked SR flip-flop

Clocked RS flip-flop can also be made using 2-input gates as shown in Fig. 16.21.

When the CLK input is at 0 level, the outputs of the two AND gates are also at logic level 0. The output of the Flip-flop at Q and $\overline{Q}$ will be uneffected and will be latching at the last known state. With a high clock input (i.e., 1) the flip-flop is set and reset as has been shown in the truth table. Synchronization of the clock timing signal to the flip-flop creates what is sometimes called a clocked SR flip-flop.

RS flip-flop). The two inputs of RS flip-flop of Fig. 16.17 is now replaced by 3-input AND gates. The third input of each gate is the feedback from the output Q and $\overline{Q}$ as shown in Fig. 16.22. In RS flip-flop the invalid state occurred due to the activation of both the inputs. Here, the two inputs are interlocked so that they cannot be activated simultaneously.

16.11.4 D Flip-flops

The output Q and $\overline{Q}$ of a D flip-flop change only on the positive going edge of the incoming clock pulse as shown in Fig. 16.23. If D = 1 and the positive going clock edge appears, then output Q = 1 and $\overline{Q}$ When D = 0 and positive going pulse appears, then output Q = 0 and $\overline{Q}$ For negative going edge of the incoming clock pulse, the flip-flop is inactive don't care.

For edge-triggered flip-flops, the clock signal is applied in the form of sharp positive and negative spikes instead of in the form of square or rectangular pulse train. Such sharp spikes are obtained from the rectangular pulse with the help of a passive differentiator circuit as has been shown in Fig. 16.23. Edge-trigger flip-flops can be of two types, viz positive edge triggered and negative edge triggered.

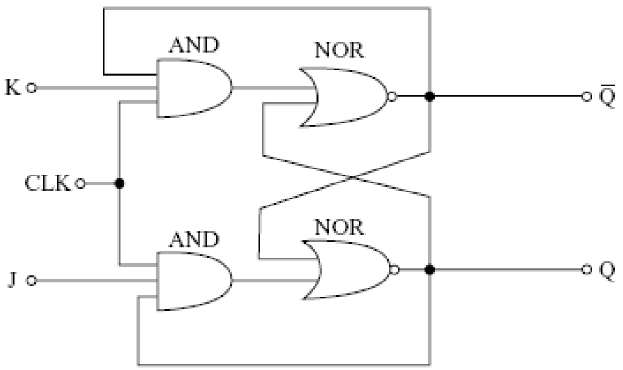


Figure 16.22 JK flip-flop

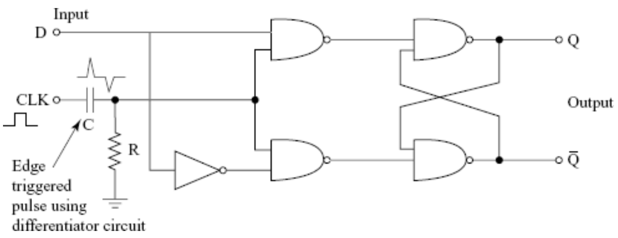


Figure 16.23 D flip-flop using edge-triggered NAND gates

16.11.5 T Flip-flops (Toggle Flip-flop)

If T = 0, J = 0 and K = 0, the output Q and $\overline{Q}$ will remain unchanged. When T = 1, J = 1 and K = 1, the output will toggle corresponding to each and every edge of the clock signal.

16.11.6 Master–Slave JK Flip-flop

Master-slave JK flip-flop as shown in Fig. 16.25 is a cascade connection of two flip-flops. The master is a clocked JK flip-flop and the slave is a RS flip-flop. The output of the second flip-flop is fed back to the input of the first. Positive clock pulses are applied to the first, called the master and same clock pulses after inversion are applied to the slave Flip-flop. When the clock pulse is at 1 (positive level) the master is active and the slave is inactive. When clock is at 0 (low level) the slave gets active and the master stays inactive

16.11.7 Counters and Shift Registers

Flip-flops are wired together that form circuits to do counting functions. Manufacturers make self-contained counters in IC form.

A register is a group of flip-flops used to store binary numbers. The output on each flip-flop is connected to the input of the adjacent input to form a register. The connection of the flip-flops are made in such a way that each data bits get transferred or shifts one flip-flop to the left or to the right. Flip-flops are classified the way data are entered and retrieved. For example, in serial-in serial-out registers, input data are applied one bit at a time to the first flip-flop in the form of a chain and is retrieved or read out from the last flip-flop in the form of a chain, one bit at a time. You must have seen the use of shift registers in your calculators. As we enter any digit on the key board, the numbers shift to the left on the display.

Taken, for example, what happens when we enter 65. We press 6 on the key board. 6 appears on the extreme right on the display. Next we press 5 on the key board. The number 6 gets shifted to the left creating space for number 5. This way the shift register carries out two functions, viz it holds or remembers the number 6 even after the pressure on the key board is released as it has a temporary memory, then it shifts the number to the left on the display each time we press a new digit or a character on the key board. The short time memory and shifting characteristics have made shift registers extremely useful in digital electronic systems.

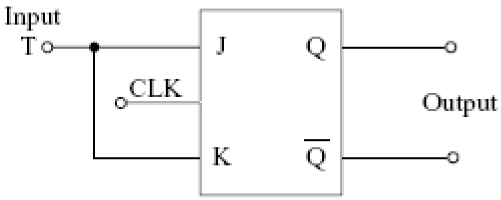


Figure 16.24 A JK flip-flop converted into T flip-flop

Figure 16.25 Master—slave JK flip-flop

16.11.8 Arithmetic Circuits

Combinational logic circuits are used to make adders and subtractors. In the central processing unit (CPU) of a computer, arithmetic functions are performed in a section called arithmetic logic unit (ALU). This section can perform functions like add, subtract, multiply, dvide, compare, complement, shift, and perform logic functions like AND, OR, and XOR.

16.11.9 Memory Function or Data Storage

You must have seen floppy disks and CD-ROMs. CD-ROM (compact disk read-only memory) is an optical storage device which can store many thousands of typed pages of information. Such optical disks can store much more data than floppy disks. Flip-flops form the basic memory cell.

Some semiconductor memory devices are RAM (random-axis memory), ROM (read-only memory), PROM (programmable read-only memory), EPROM (erasable PROM), etc. The hard disk drive is currently the most important large capacity storage (bulk storage) memory device used in computers these days.

16.11.10 Digital Systems

Like any other system a digital system consists of a number of sub-systems assembled together to perform some specific task. In a digital system, the sub-system may be adders, subtractors, counters, shift registers, RAMs or ROMs. Subsystems are manufactured on a single chip of IC. Even the whole system is made available on a single IC chip. Such digital integrated systems are classified as SSI (small-scale integration), MSI (medium-scale integration), LSI (large-scale integration), VLSI (very-large- scale integration), and ULSI (ultra-large-scale integration). A VLSI contains logic gates ranging from 10,000 to 99,999. Your digital wrist watch, for example, is an assembly of a number of digital subsytems on a single chip. The calculator we use, if we open it, we will notice, contains a small pencil cell, a mini read out display, key board from which few wires are attached to an IC chip. The IC is an LSI chip which is intended to perform thousands of logic functions consisting of storage, processing, and control that are required in a calculating system.

16.12 REVIEW QUESTIONS

A. Short Answer Type Questions

1. State the advantages of digital signals over analog signals.
2. Differentiate between analog and digital signals. What are the disadvantages of analog systems?
3. What are the different types of number systems? Explain each of them.
4. What is a binary number system. Mention its applications
5. Explain with example how binary numbers can be converted into decimal numbers.
6. Illustrate how decimal numbers can be converted into binary numbers.

    3. 0.578;
    4. 110.58;
    5. 109;
    6. 18.33;
    7. 212.

8. Convert the following fractional decimal numbers into binary numbers:
    1. 0.782;
    2. 0.359;
    3. 0.568;
    4. 0.075;
9. Convert the following binary numbers into decimal numbers:
    1. 1101;
    2. 1111;
    3. 1001;
    4. 1100101;
10. Convert the following fractional binary numbers to fractional decimal numbers:
    1. 0.1111;
    2. 0.1001;
    3. 0.0101;
11. Add the following binary numbers:
    1. 1011+1110;
    2. 101010+111011
12. Explain the difference between hexadecimal and octal systems.
13. Perform the following binary multiplication:
    1. $(111)_2 \times (1.2)_2$;
    2. $(10101)_2 \times (1.1)_2$;
    3. $(10100)_2 \times (0.1)_2$.
14. State De Morgan’s theorem
15. Write down truth tables for the following gates:
    1. NAND gate;
    2. AND gate;
    3. NOR gate.
16. Explain a logic OR gate using diodes. Also write the truth table.
17. What are universal gates? Explain with an example.
18. What ate flip-flops? Explain one of them.
19. Explain RS flip-flop. What is D flip-flop?
20. Explain JK flip-flop. What is T flip flop?
21. Draw and explain the circuit for master–slave JK flip-flop.
22. What is a digital system? Give an example of a digital system and name its subsystems.

[Ans 9, 10011]

24. Convert decimal numbers 7547 and 100.625 into hexadecimal numbers.

[Ans $(7547)_{10}$ = $(107\ B)_{16}$; $(100.625)_{10}$ = $(64.A)_{16}$]

25. Convert the following hexadecimal numbers to binary.

- AEO;
- 75 F;
- E 25.

[Ans (a) 101011100000; (b) 011101101111 (c) 111000100101]

26. Convert the following octal into binary.

- 71;
- 560;
- 173.

[Ans (a) 111001; (b) 101110000; (c) 001111011]

27. Perform the following binary operations.

- $(1011)_2 \times (1001)_2$;
- $(111)_2 \times (190)_2$.

[Ans (a) 1100011; (b) 101010]

28. Express the decimal number 9 into (a) binary; (b) BCD; (c) octal; (d) hexadecimal.

[Ans (a) 1001; (b) 1001; (c) 11; (d) 9]

29. Add hexadecimal numbers 3E91 and 2F93.

[Ans 6E 24]

30. Solve the following conversion.

- $(110001)_2 = (?)_{10}$;
- $(23.6)_{10} = (?)_2$;
- $(BCA\ 3.AD)_{16} = (?)_2$.

1. inverters connected to the output of an AND gate
2. an inverter connected to the inputs of a NAND gate
3. inverters connected to the inputs of an OR gate
4. an inverter connected to the output of an OR gate.

2. The sum of binary 10011 and 0111 is
   1. 01011
   2. 11010
   3. 11101
   4. 10111.
3. Which of the following statements is true according to De Morgan’s theorems?
   1. $\overline{A + B} = \overline{A} \cdot \overline{B}$
   2. $\overline{A} + \overline{B} = \overline{A} \cdot \overline{B}$
   3. $A + B = \overline{A} \cdot \overline{B}$
   4. $\overline{A \cdot B} = \overline{A} \cdot \overline{B}$.
4. Connecting of inverters at all the inputs of an AND gate produces a
   1. NAND gate
   2. OR gate
   3. NOR gate
   4. XOR gate.
5. Which of the following gates is represented by the Boolean expression: A + B + C + D = Y
   1. 4-input AND gate
   2. 4-input OR gate
   3. 4-input NAND gate
   4. 4-input NOR gate.
6. Which of the following is the Boolean expression for a 3-input NAND gate?
   1. $\overline{A \cdot B \cdot C} = Y$
   2. $\overline{A + B + C} = Y$
   3. $\overline{A} \cdot \overline{B} \cdot \overline{C} = Y$
   4. A.B.C=Y.
7. Which flip-flop has only one data input?
   1. RS flip-flop
   2. D flip-flop
   3. JK flip-flop
   4. Triggeringflip-flop.
8. Which of the following can be called an universal flip flop?
   1. JK flip-flop
   2. D flip-flop

9. In a JK flip-flop, repeated clock pulses causes the output to turn on-off-on-off—when
    1. both inputs J and K are at 0
    2. J is at 0 and K is at 1
    3. K is at 0 and J at 1
    4. both inputs J and K are at 1.
10. The master–slave JK flip-flop is an example of
    1. level-triggered device
    2. positive edge-triggered device
    3. negative edge-triggered device
    4. pulse-triggered device.
11. An astable multivibrator
    1. generates a continuous flow of pulses
    2. is a flip-flop
    3. generates a single short pulse
    4. always is in one of the two stable states.
12. A monstable multivibrator is also called a
    1. (a) flip-flop
    2. clock
    3. one shot multivibrator
    4. free running multivibrator.

Answers to Multiple Choice Questions

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

Resource Centers / Playlists / History / Topics / Settings / Get the App /

PREV
15. Rectifiers and Other Diode Circuits

NEXT
17. Integrated Circuits