

# Unit 1

## Binary Basics

### *Unit Objectives*

After completion of this unit you will be able to identify and describe the following concepts:

1. Numbering systems—base 10 (decimal), base 2 (binary) and base 16 (hexadecimal)
2. Conversions among various number systems—binary, hexadecimal and decimal
3. Math functions (addition, subtraction, multiplication and division) with binary numbers
4. Recognize various logic gates and their functions and develop truth tables for them
5. Develop truth tables for various logic circuits

### *1-1 Numbering Systems and their uses*

The numbering system in general use is the base 10 system, probably because we have used our 10 fingers for counting since the early days of man. Digital computers operate on the principles of semiconductor circuitry switching between 2 states—on and off. These two states can be represented as a “1” or a “0”. Since there are only two states to be considered, a base 2 system is used. This base 2 numbering system is referred to as the binary system and is the language of computers.

### *1-1.2 Notation Review—Exponents and Subscripts*

When dealing with large values, such as 10,000,000 ohms of resistance, it is easier to write the value as  $10^6$  ohms. This form of number shorthand is called *scientific notation*. Using *scientific notation* requires an understanding of the *powers of ten*. The number 10,000,000 can be seen as the value of 10 raised to the 6<sup>th</sup> power, or  $10 \cdot 10 \cdot 10 \cdot 10 \cdot 10 \cdot 10$  or  $10^6$ . So,  $10^N$  is 10 multiplied by itself N times. The following table illustrates how a value is converted to scientific notation. The number used in the following example is 832,129.

Number	8	3	2	1	2	9
Power of Ten	$10^5$ (100,000)	$10^4$ (10,000)	$10^3$ (1,000)	$10^2$ (100)	$10^1$ (10)	$10^0$ (1)

As we move from right (starting at  $10^0$  or 1) to left, the powers of ten increase. The highest power of ten in this case is  $10^5$ , so we move the decimal point 5 places to the left.



We now re-write the value in scientific notation:  $8.32129 \cdot 10^5$ .

## Unit 1

Actually, raising any number to a power is a statement of the number of times that a number is multiplied times itself. Scientific notation uses the **base 10** number system. Since we will also be using a **base 2** system, let's consider powers of 2. The following table shows values for powers of 2 from  $2^0$  to  $2^8$ .

Power of Two	$2^8$ (256)	$2^7$ (128)	$2^6$ (64)	$2^5$ (32)	$2^4$ (16)	$2^3$ (8)	$2^2$ (4)	$2^1$ (2)	$2^0$ (1)
--------------	-------------	-------------	------------	------------	------------	-----------	-----------	-----------	-----------

As with powers of ten,  $2^N$  is 2 times itself N times.

Example:  $2^6 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 64$

As you have seen, the power a value is raised to is illustrated with a superscript—value<sup>N</sup>. Another form of notation is the subscript—value<sub>N</sub>. The subscript is often used in schematic drawings to represent specific components: R<sub>1</sub>, C<sub>5</sub>, etc. It can also be used to represent the base of a numbering system. An example of this would be the number 259. When you see this number you assume that it is a base 10 (Decimal) value, but it could also be a base 16 (Hexadecimal) value. To make sure which number system is being used, a subscript is attached to the value. As illustrated below, the use of this notation avoids confusion when using different numbering systems.

**$259_{10}$  = Base 10 value or  $259_{16}$  = Base 16 value**

### 1-1.3 The Decimal (base 10) Numbering system

We are all familiar with the base 10 numbering system, so we will start with it as the basis for explaining other numbering systems. The base 10 numbering system consists of ten possible digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9. When placed together in a multi-digit number, each digit has a weighting factor based on a power of 10 ( $10^x$ ). The lowest value of weighting factor is the right-most number, and the highest value of weighting factor is the left-most number. Consider the number "36258". The right-most value of "8" has a weighting factor of  $10^0$  and the left-most value has a weighting factor of  $10^4$ . Let's consider the weighting factors for each digit position from MSD (Most Significant Digit) to LSD (least Significant Digit).

Place Value	MSD	4SD	3SD	2SD	LSD
Number	3	6	2	5	8
Weighting Factor	$10^4$ (10,000)	$10^3$ (1,000)	$10^2$ (100)	$10^1$ (10)	$10^0$ (1)
Number • Weighting	30,000	6,000	200	50	8

Note: The weighting factor values in parentheses are the actual values of 10 raised to the power shown.

In order to determine the value we must now add the weighted numbers together.

$$30,000 + 6,000 + 200 + 50 + 8 = 36,258$$

We normally don't go through this effort when using the decimal system because it is our normal numbering system and we automatically understand that the left-most value is the greatest and the right-most value is the least. This weighting example using the familiar decimal numbering system is provided because the same concept applies to other numbering systems as well.

### ***1-1.4 The Binary (base 2) Numbering System***

As previously mentioned, the binary numbering system is used in digital electronics because there are only two digits to be considered: 0 and 1. These two values are represented by the *on/off*, *voltage/no voltage* conditions in electronic circuits. Logically they can represent *yes/no* or *true/false* conditions.

Why use binary numbers? Because there are only two values, binary numbers are *easy to store* and *simple to transport* over long distances. When processed, binary information can represent instructions or values. Although binary values can be of any length, we often see them in groups of eight (referred to as “bytes” or “octets”).

When using binary numbers it is important to note that the maximum number of combinations of 1s and 0s is a function of the number of places in the group of binary numbers. When binary numbers are grouped into octets, for example, the maximum number of values that can be represented will be  $2^8$  or 256 values ranging from 00000000 to 11111111. The following is an example of bits arranged into octets:

**00110100 10111011 01011001 10101011 01101110**

Being human, these numbers hold no significance for us. They could represent amplitude values for digitized voice or instructions for processing information or alphabet characters to be displayed on a computer screen.

### ***1-1.5 The Hexadecimal (base 16) Numbering System***

We have seen the decimal numbering system and the binary numbering system. Why do we need another numbering system? The hexadecimal system is actually an extension of the binary numbering system that makes reading and interpreting long strings of binary numbers simpler for humans. Computers still read binary values, but we use hexadecimal as a sort of shorthand when we have to read binary values in 8, 16 and 32 bit sequences. . The base 16 numbering system consists of sixteen possible digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Notice that since this is a base 16 numbering system, when 9 is reached there are still six more values to be counted before moving to the left and repeating the lower values. The letters A through F were chosen to represent the last six values in a hexadecimal numbering system. The following is an example of hexadecimal numbers representing the eight bit sequences shown in section 1-1.3:

**34 BB 59 AB 6E**

Hexadecimal, as you can see, takes up less space than binary values. And, believe it or not, they are easier for humans to read once you get the hang of it. We are often given instructions for programming physical switches, or setting other equipment parameters, in hexadecimal (typically referred to as just “Hex”) form.

## Unit 1

### 1-2 Converting between numbering systems

When dealing with any of the three numbering systems we have been considering, we must sooner or later have to convert from one to the other. We will take a look at the processes involved in this section.

#### 1-2.2 Binary to Decimal Conversion

Converting from binary to decimal is a fairly simple task. It is essentially a three-step process similar to the decimal example (section 1-1.2) only with  $2^x$  instead of  $10^x$ :

1. Weight the values of each position with powers of 2—right-most position being LSD and left-most position being MSD
2. Multiply weighted values by the binary value at that position (1 or 0)
3. Add the values together

Let's try the process to convert the binary value of **10100010** to decimal form.

Place Value	MSD	7SD	6SD	5SD	4SD	3SD	2SD	LSD
Number	1	0	1	0	0	0	1	0
Weighting Factor	$2^7$ (128)	$2^6$ (64)	$2^5$ (32)	$2^4$ (16)	$2^3$ (8)	$2^2$ (4)	$2^1$ (2)	$2^0$ (1)
Number • Weighting	128	0	32	0	0	0	2	0

Note: The weighting factor values in parentheses are the actual values of 2 raised to the power shown.

In order to determine the decimal value we must now add the weighted numbers together.

$$128 + 0 + 32 + 0 + 0 + 0 + 2 + 0 = 162_{10}$$

So, we can see that the binary value of 10100010 converts to a decimal value of 162. You may have noticed that the multiplication step is not necessary when dealing with binary values. A simpler way to convert might be to binary weight the positions and add the weighted values together of any position with a binary value of 1.

Binary to Decimal Conversion Examples:

$$1011 = 2^3 + 0 + 2^1 + 2^0 = 8 + 0 + 2 + 1 = 11$$

$$0110 = 0 + 2^2 + 2^1 + 0 = 0 + 4 + 2 + 0 = 6$$

$$1111 = 2^3 + 2^2 + 2^1 + 2^0 = 8 + 4 + 2 + 1 = 15$$

#### Exercise 1-1: Binary to Decimal Conversion

1. 10110111
2. 100111
3. 01110
4. 011010
5. 1011110010
6. 1001
7. 11100100
8. 100001

**1-2.3 Decimal to Binary Conversion**

Converting from decimal to binary is a different process. Decimal to binary conversion is accomplished by successive division of the original decimal value by 2. The binary value is determined by a remainder test after each division.

Let's look at the steps involved in conversion from decimal to binary.

1. Divide the number to be converted by 2
2. Test for a remainder of the division: yes = 1, no = 0
3. Divide the quotient of the previous division by 2
4. Continue steps 2 and 3 until  $1 \div 2$  is reached
5. Determine the binary value of the number:  
     MSD = remainder test of first division  
     LSD = remainder test of last division

Using the decimal value of **162** from the binary to decimal conversion section we will now perform a decimal to binary conversion.

Operation	Result	Test for Remainder (yes = 1, no = 0)
$162 \div 2$	81	0 LSD
$81 \div 2$	40	1
$40 \div 2$	20	0
$20 \div 2$	10	0
$10 \div 2$	5	0
$5 \div 2$	2	1
$2 \div 2$	1	0
$1 \div 2$	0	1 MSD

Reading the remainder test from bottom (MSD) to top (LSD) we get **10100010**. Another technique when converting smaller values is to add binary weight values together to reach the decimal value and then replace them with a 1 in the proper position.

Example: Converting decimal value 25 to a binary value using by adding  $2^x$  values

$$25_{10} = 16 + 8 + 1 = 2^4 + 2^3 + 2^0 = 2^4 + 2^3 + 0 + 0 + 2^0 = 11001$$

**Exercise 1-2: Decimal to Binary Conversion**

- |       |        |       |        |
|-------|--------|-------|--------|
| 1. 12 | 2. 154 | 3. 86 | 4. 319 |
| 5. 27 | 6. 63  | 7. 79 | 8. 36  |

## Unit 1

### 1-2.4 Binary to Hexadecimal Conversion

As previously stated, reading and interpreting groups of binary numbers is a difficult task for humans. Therefore we commonly see these binary values after they have been converted to something easier to read. The hexadecimal numbering system is quite often used for this purpose, but upon occasion we still have to perform conversion from binary to hexadecimal all by ourselves.

The process of converting from binary to hex is simple and straightforward. It is a three-step process:

1. Break the binary value into groups of four (4) bits from right to left
2. Binary weight values ( $2^3, 2^2, 2^1, 2^0$ ) of each individual 4-bit group
3. Add the binary weighted values together to determine which of the sixteen possible hex values (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) is represented

Let's look at a long binary value and convert to hex: **10000111010100100101101**

Notice that converting this value to decimal would be a tedious task. So, let's follow the three-step process.

Binary Number	10000111010100100101101					
1. 4-bit groups	0100	0011	1010	1001	0010	1101
2. Binary weighting	0+4+0+0	0+0+2+1	8+0+2+0	8+0+0+1	0+0+2+0	8+4+0+1
3. Add to Hex Value	4	3	A	9	2	D

Note: Leading 0s are added to the left-most group to fill it out to four bits.

$$10000111010100100101101 = 43A92D_{16}$$

### Exercise 1-3: Binary to Hexadecimal Conversion

1. 10101011
2. 001011
3. 0110110
4. 1010110010
5. 10101010110
6. 11011101
7. 111001010
8. 1011010101

### 1-2.5 Hexadecimal to Binary Conversion

As you may have guessed, going from hexadecimal to binary is basically the reverse of the binary to hex process.

Let's look at a hexadecimal value and convert it to binary: A4893D

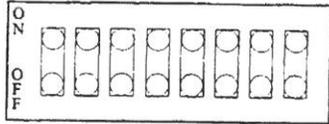
Hex Value	A	4	8	9	3	D
1. Binary Weighting	8+0+2+0	0+4+0+0	8+0+0+0	8+0+0+1	0+0+2+1	8+4+0+1
2. 4-bit groups	1010	0100	1000	1001	0011	1101
3. Combine to binary value	101001001000100100111101					

$$A4893D_{16} = 101001001000100100111101$$

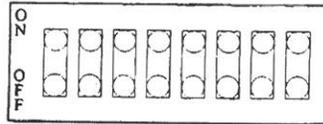
**Exercise 1-4: Hexadecimal to Binary Conversion**

Program the following switches with information supplied in Hex format. A binary 1 is an **on** position and a binary 0 is an **off** position. Fill-in the circle at the proper switch position.

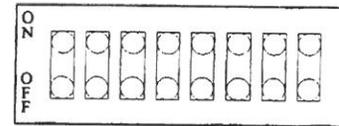
1. A6



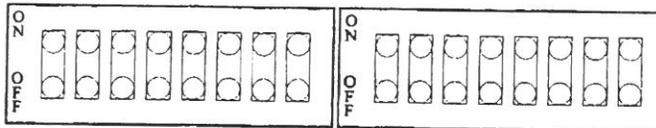
2. 38



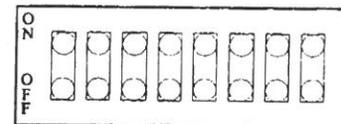
3. 4F



4. A5D3



5. 7E



**1.2-6 Converting between Hexadecimal and Decimal**

The last number conversion process we will consider is going from Hex-to-Decimal or Decimal-to-Hex. When performing these conversions, you must first convert to binary and then convert from binary to the other number system. Let's use the number  $36_{10}$  and convert to its Hexadecimal equivalent, and then  $36_{16}$  and convert to its Decimal equivalent:

$$36_{10} = 100100_2 = 24_{16}$$

and

$$36_{16} = 00110110_2 = 54_{10}$$

You can see how important the subscript can be at times to avoid confusion when using different number systems. The following table shows the relationship of the number systems we have been considering. Notice that for values 0 through 9 both Decimal and Hexadecimal systems are the same.

**Conversion Table: Decimal, Binary and Hexadecimal**

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

**1-3 Binary math functions**

In this section you will learn to perform basic binary math—addition, subtraction and multiplication.

**1-3.1 Addition of binary numbers**

When adding binary numbers the method is the same as that used in other bases. Consider the base 10 system. When we add two numbers we start at the right side and add the column values. If the sum exceeds the single digit value of the base, a carry is generated to the next column. Let's add the decimal values 5678 and 2345.

- Step 1.  $8 + 5 = 13$  so we place a 3 in the right column and carry the 1 into the next column
- Step 2.  $1 + 7 + 4 = 12$  so we will place 2 in the column and carry the 1 to the next column
- Step 3.  $1 + 6 + 3 = 10$  so we will place 0 in the column and carry the 1 to the next column
- Step 4.  $1 + 5 + 2 = 8$  so the answer will be 8023

When adding binary values we do the same thing, only in the base 2 system the highest value in a column is 1. Before we perform the binary addition let's introduce the **rules of addition**.

- 1.  $0 + 0$  always equals 0
- 2.  $0 + 1$  always equals 1
- 3.  $1 + 1$  always equals 0 and a carry
- 4.  $1 + 1 + \text{a carry}$  always equals 1 and a carry

Now let's perform the binary addition of 1101 and 0111.

- Step 1.  $1 + 1 = 0$  and a carry so place a 0 in the first column and carry the 1 to the next column
- Step 2.  $1 + 0 + 1 = 0$  and a carry so place a 0 in that column and carry the 1 to the next column
- Step 3.  $1 + 1 + 1 = 1$  and a carry so place a 1 in that column and carry the 1 to the next column
- Step 4.  $1 + 1 + 0 = 0$  and a carry so place a 0 in that column and carry the 1 to the last column

Decimal Addition	Binary Addition
carry 1 1 1 5 6 7 8 + 2 3 4 5 <hr style="width: 100%;"/> 8 0 2 3	carry 1 1 1 1 1 0 1 + 0 1 1 1 <hr style="width: 100%;"/> 1 0 1 0 0

There is another way to perform binary addition. You just convert to decimal and add them, then convert back to binary. Let's use the previous addition example. Convert 1101 and 0111 to decimal by binary weighting the positions:  $8 + 4 + 0 + 1 = 13_{10}$  and  $0 + 4 + 2 + 1 = 7_{10}$ . Now add  $7_{10}$  and  $13_{10}$  together to get  $20_{10}$ . Next convert  $20_{10}$  to its binary value: **10100**

**Exercise 1-5: Addition of binary numbers**

- 1. 0111 + 1011
- 2. 01101 + 101
- 3. 100110 + 11011
- 4. 10111 + 00111
- 5. 11011 + 1001
- 6. 11101 + 101001

**1-3.2 Modulo-2 Addition**

Another type of binary addition is **modulo-2 addition**. This technique is performed in the same way as binary addition **except there is no carry performed**. Consider the following example comparing binary addition and modulo-2 addition of the same values.

Binary Addition	Modulo-2 Addition	<u>Rules of Modulo-2 Addition</u>
carry 1 1 1 1 1 0 1 + 0 1 1 1 <hr style="width: 100px; margin-left: 0;"/> 1 0 1 0 0	1 1 0 1 + 0 1 1 1 <hr style="width: 100px; margin-left: 0;"/> 1 0 1 0	1. 0 + 0 always equals 0 2. 0 + 1 always equals 1 3. 1 + 0 always equals 1 4. 1 + 1 always equals 0

We will consider modulo-2 addition further in the logic gates section (section 1-4).

**1-3.3 Subtraction of Binary Numbers**

There are several methods for subtracting binary numbers. A simple approach is the **one's complements** method. The one's complement of a binary number is simply the inverted value of the binary number: 1s become 0s and 0s become ones. The ones complement of 10110010 would be 01001101. When subtracting binary values using the ones complement method, the subtrahend is inverted and then binary addition is performed according to the rules provided in the last section. If a carry is performed in the left-most position, the carry will be brought down (as illustrated) and a second addition will be performed. Let's try it. Subtract 0111 from 1101.

Step 1. Perform the one's complement of the subtrahend

Step 2. Perform addition according to the stated rules of binary addition

Step 3. **IF** there is a carry from the left-most column, add it to the sum obtained in steps 1 and 2

	Step 1	Step 2
(minuend) 1 1 0 1	1 1 0 1	carry 1 1 1 0 1 + 1 0 0 0 <hr style="width: 100px; margin-left: 0;"/> 0 1 0 1 + 1 <hr style="width: 100px; margin-left: 0;"/> 0 1 1 0
(subtrahend) - 0 1 1 1	- 1 0 0 0	
	<hr style="width: 100px; margin-left: 0;"/>	
	<hr style="width: 100px; margin-left: 0;"/>	
		Step 3

As with addition, another method would be to convert from binary to decimal values and perform subtraction then convert the answer back to binary. Either approach is acceptable.

**Exercise 1-6: Subtraction of binary numbers**

- |                  |                 |                    |
|------------------|-----------------|--------------------|
| 1. 1011 - 0011   | 2. 1101 - 0101  | 3. 101110 - 01011  |
| 4. 10111 - 00101 | 5. 11011 - 1001 | 6. 111101 - 101001 |

Unit 1

**1-3.4 Multiplication of Binary Numbers**

When multiplying binary numbers the best approach is to convert the binary numbers to decimal values, perform multiplication and then convert them back to binary values. Let's multiply 1011 times 0110.

Step 1. Convert both binary values to decimal values as shown in section 1-2.2

Step 2. Perform decimal multiplication

Step 3. Convert the decimal answer to a binary value as shown in section 1-2.3

$\begin{array}{r} 1011 \\ \times 0110 \\ \hline \end{array}$	<p><b>Step 1</b></p> $8 + 0 + 2 + 1 = 11$ $0 + 4 + 2 + 0 = 6$	<p><b>Step 2</b></p> $\begin{array}{r} 11 \\ \times 6 \\ \hline 66 \end{array}$
--	---	---

<b>Step 3</b>		
Operation	Result	Test for Remainder (yes = 1, no = 0)
$66 \div 2$	33	0      LSD
$33 \div 2$	16	1
$16 \div 2$	8	0
$8 \div 2$	4	0
$4 \div 2$	2	0
$2 \div 2$	1	0
$1 \div 2$	0	1      MSD
<b>Decimal 66 = Binary 1000010</b>		

**Exercise 1-7: Multiplication of binary numbers**

1. 1011 · 0110      2. 0111 · 1000      3. 1111 · 0001      4. 101 · 1110

**1-4 Logic Gates**

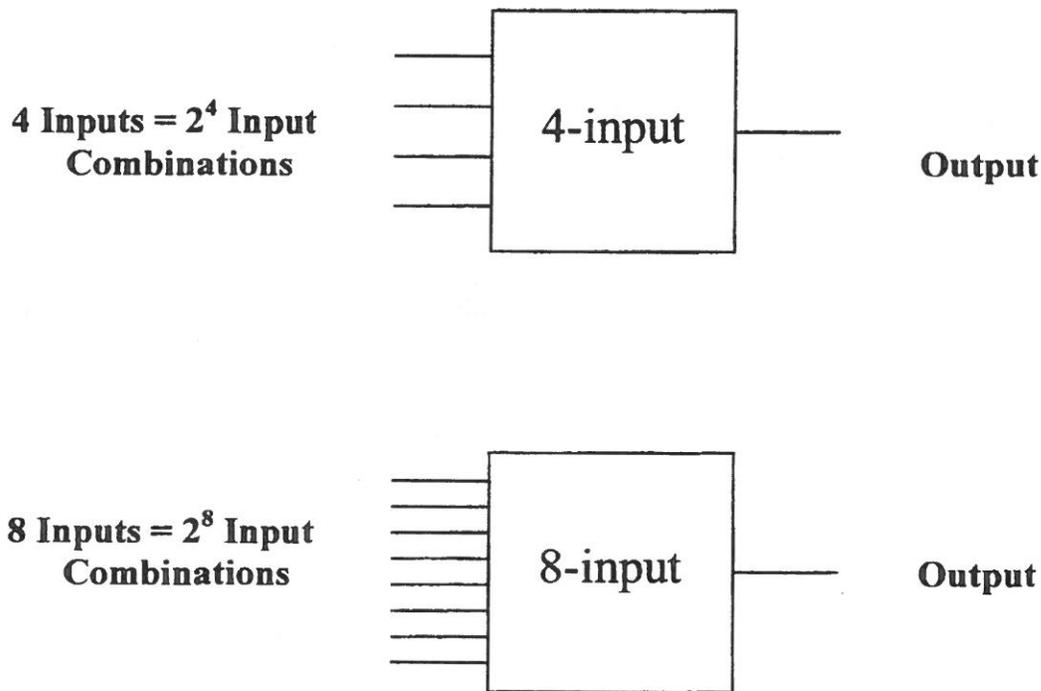
Logic gates are the fundamental building blocks of digital circuits. Complex digital circuits such as flip-flops, shift registers encoders and decoders can be constructed using the basic gate circuits presented in this section. This course does not address the complex digital circuits, only the basic logic gates.

Remember, in the digital world we define only two states: **on** and **off**. So, before we look at the gate circuits, let's consider some of the terms used to define on and off. The following table shows notations used with logic circuits.

<b>Logic Circuit Notation</b>	
<b>ON</b>	<b>OFF</b>
Closed	Open
1	0
True	False
High	Low
+ Voltage	0 Voltage

Each type of logic gate has its own truth table associated with it. A truth table is a table that shows all possible inputs to a logic gate and the output for a given set of input levels.

Logic gates can, and often do, have more than two inputs. As the number of inputs increases, the number of possible combinations of inputs increases by  $2^x$  (x = number of inputs). A three input gate would have  $2^3$  (eight) possible combinations of inputs and thus a larger truth table. Consider the generic gates in figure 1.1.



**Figure 1.1 - Generic Gates**

**1-4.2 The AND Gate**

The first gate circuit to be considered is the AND gate. The AND function is shown below in Figure 1.2 using a simple circuit consisting of a battery, switches and a lamp. Notice that in order for the lamp to be on, both switches must be closed. Adding more switches in series in the AND function circuit is equivalent to adding more inputs to an AND gate. The truth table shows all of the possible combinations of the two switches and the resulting lamp status. The AND function can then be stated in the following way:

In order for the lamp to be on, **both** the A switch AND the B switch must be closed.

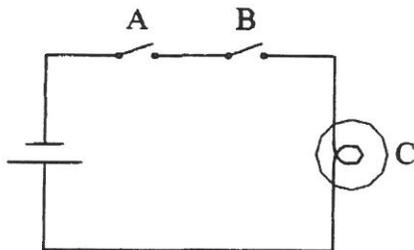


Figure 1.2 - The AND function

Switch A	Switch B	Lamp C
Open	Open	Off
Open	Closed	Off
Closed	Open	Off
Closed	Closed	On

Truth Table for AND Function

The symbol for an AND gate is shown in Figure 1.3 below. A and B represent the input and C is the output. Notice that because this is a 2-input AND gate the number of possible input combinations is four.

In order for the output to be **high**, **both** the A input AND the B input must be **high**.

Truth Table for AND Gate

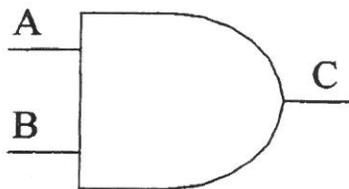


Figure 1.3 - The AND Gate Symbol

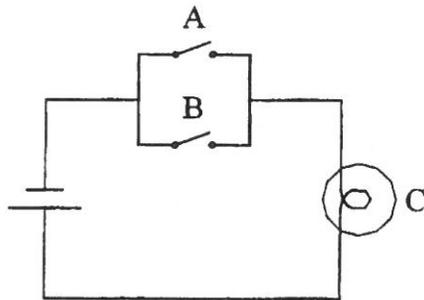
Input A	Input B	Output C
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table for AND Gate

**1-4.3 The OR Gate**

As with the AND gate, the function of the **OR** gate can be represented by a circuit consisting of a battery, two switches and a lamp (Figure 1.4). Adding more switches **in parallel** in the OR function circuit is equivalent to adding more inputs to an OR gate. The truth table shows all of the possible combinations of the two switches and the resulting lamp status. The OR function can then be stated in the following way:

In order for the lamp to be **on**, **either** the A switch **OR** the B switch **or both** must be closed.



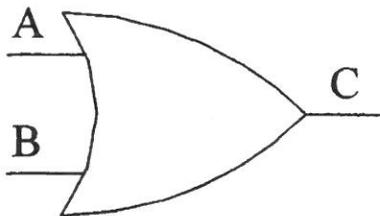
**Figure 1.4 - The OR function**

Switch A	Switch B	Lamp C
Open	Open	Off
Open	Closed	On
Closed	Open	On
Closed	Closed	On

**Truth Table for OR Function**

The symbol for an OR gate is shown in Figure 1.5 below. A and B represent the input and C is the output. Notice that because this is a 2-input OR gate the number of possible input combinations is four.

In order for the output to be **high**, **either** the A input **OR** the B input **or both** must be **high**.



**Figure 1.5 - The OR Gate Symbol**

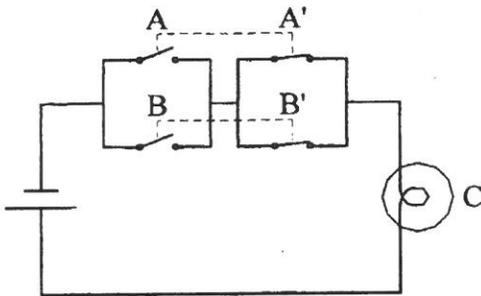
Input A	Input B	Output C
0	0	0
0	1	1
1	0	1
1	1	1

**Truth Table for OR Gate**

**1-4.4 The X-OR (Exclusive-OR) Gate**

The Exclusive-OR gate is an OR gate that only provides a high output **if one and only one** of the gates is high. As with the previous gates, the X-OR function can be represented by a circuit consisting of a battery, switches and a lamp (Figure 1.6). The circuit shown only illustrates the operation of a 2-input X-OR function. Since an X-OR gate is more complex than the AND or the OR gate, the configuration of the circuit is a bit more complex. Notice that instead of just an A switch and a B switch there is also an A' and a B' switch. The dashed lines connecting the switches together indicate that they work together. In other words, closing the A switch will cause the A' switch to open. The truth table shows all of the possible combinations of the switches and the resulting lamp status. The X-OR function can then be stated in the following way:

In order for the lamp to be **on**, *only* the A switch **OR** the B switch must be **closed**—*not both*.

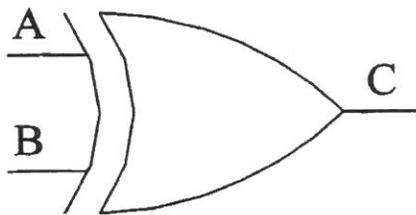


Switch A	Switch B	Lamp C
Open	Open	Off
Open	Closed	On
Closed	Open	On
Closed	Closed	Off

**Truth Table for X-OR Function**

The symbol for an X-OR gate is shown in Figure 1.7 below. A and B represent the input and C is the output. Notice that because this is a 2-input X-OR gate, the number of possible input combinations is four. Since an X-OR gate can have more than two inputs the following statement applies to **only** an X-OR gate with **two** inputs.

In order for the output to be **high**, *only* the A input **OR** the B input must be **high**—*not both*.



Input A	Input B	Output C
0	0	0
0	1	1
1	0	1
1	1	0

**Figure 1.7 - The X-OR Gate Symbol**

**Truth Table for X-OR Gate**

If you compare the truth table for the X-OR gate to the rules stated for **Modulo-2 Addition** (section 1-3.2), you will notice that modulo-2 addition of binary values is the same as performing an Exclusive OR of the values.

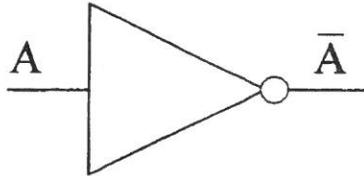
So far we have just considered 2-input X-OR gates. The following is a more general statement regarding the X-OR function for **any number of inputs**.

In order for the output of an X-OR gate to be high, **one and only one** of the inputs must be high.

**1-4.5 The Inverter**

The **Inverter**, also referred to as the **NOT** circuit, inverts or complements an input. The symbol for an inverter is shown in Figure 1.8. The Inverter function can then be stated in the following way:

If the input is **A** the output is **NOT A** and conversely, if the input is **NOT A** the output is **A**.



Input	Output
1	0
0	1

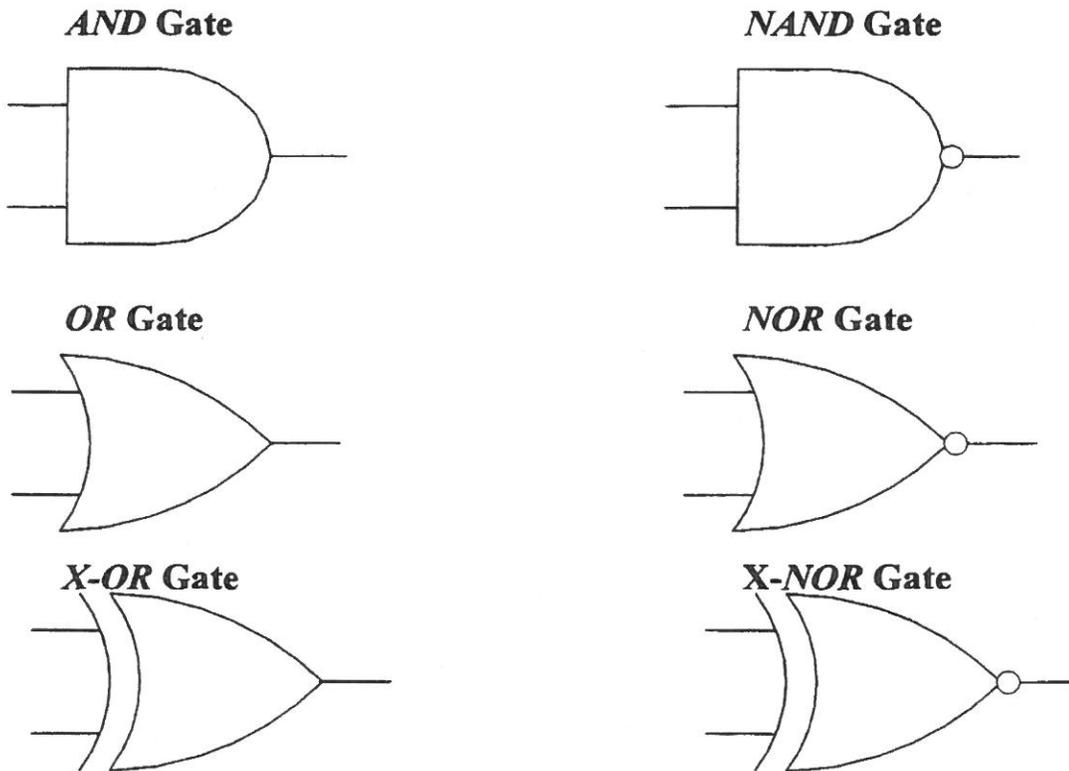
**Figure 1.8 - The Inverter Symbol**

**Truth Table for an Inverter**

The line above the output **A** indicates that it is the inverse of the input **A**.

The inverter is used to change the logic states on the input and/or output of logic gates. When the inverter is added to a logic gate output it becomes a different logic gate. For example: An inverter at the output of an **AND** gate will turn it into a **NAND** gate. When this is done at the output of a gate the only portion of the inverter symbol that is used is the small circle at the output point.

Consider the logic gates in Figure 1.9 and the effect of inverting their outputs:

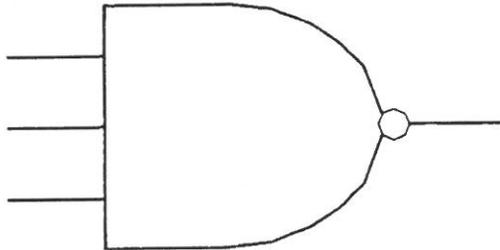


**Figure 1.9 Basic Logic Gates**

Unit 1

**Exercise 1-8: Logic Functions**

1. What is the type of gate illustrated below? \_\_\_\_\_
2. Fill-in the associated truth table.



Logic Gate for questions 1 and 2

Truth Table			
Inputs			Output
A	B	C	D
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

3. Inverting the inputs of a 2-input AND gate will change it into another the type of gate.  
What type of gate will it effectively become? \_\_\_\_\_

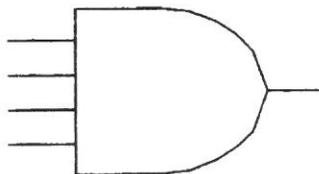
4. Perform the **AND** function on the following binary numbers. Treating each column separately as A and B inputs, **AND** them together to a C output.

A 10011100  
 B 10001101  
 C \_\_\_\_\_

5. Perform **Modulo-2 Addition (X-OR)** on the following binary numbers—A and B.

A 10011100  
 B 10001101  
 C \_\_\_\_\_

6. Refer to the following logic gate.



How many input combinations will provide a **high** output state? \_\_\_\_\_

How many input combinations are possible for this gate? \_\_\_\_\_

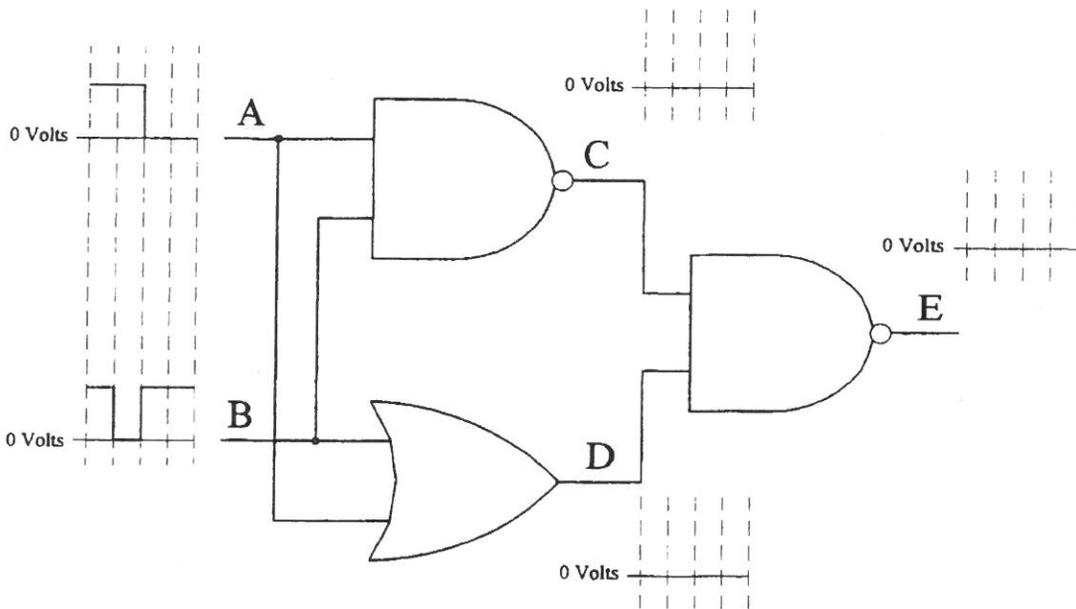
Unit 1

**1-5 Truth tables and logic circuits**

As previously mentioned, logic gates can be connected together to perform many functions in digital circuits. In troubleshooting, it is important to be able to follow a signal through these digital circuits and accurately predict the proper output for a given input pattern.

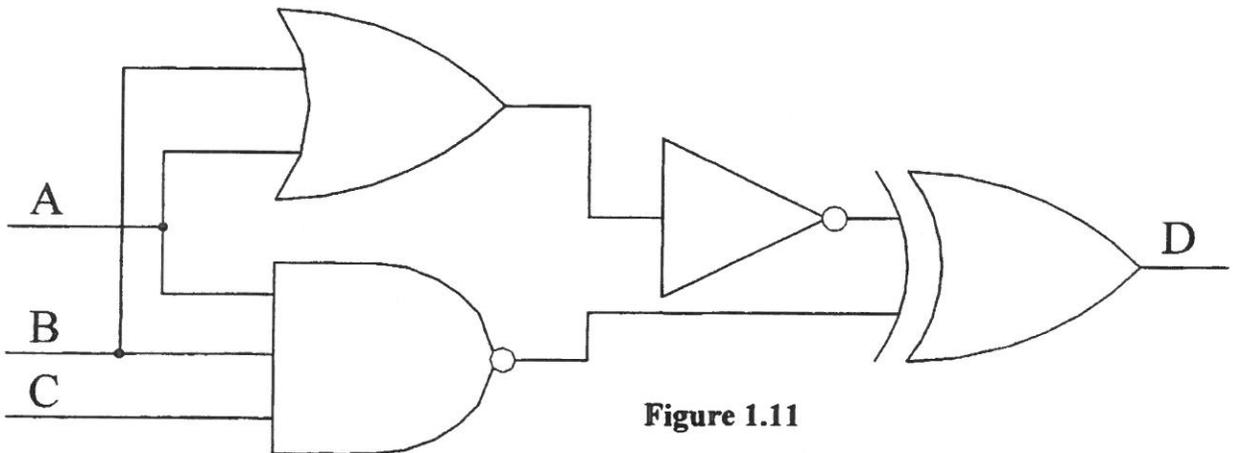
In this section you will be performing signal tracing exercises with circuits that combine all of the logic gates previously discussed. You will determine the output values at various points along the circuit path for given input values. The solution will be presented on the next page.

**Circuit 1** Refer to Figure 1.10 below. Inputs A and B have two separate signal patterns at their input. As each bit flows into the A and B inputs determine what the level is at outputs C, D and E. Remember: a zero level represents a binary 0 and a plus level represents a binary 1.



**Figure 1.10**

**Circuit 2** Develop a truth table for the circuit in Figure 1.11 showing all possible inputs at A, B and C and the corresponding output at D.



**Figure 1.11**

Unit 1

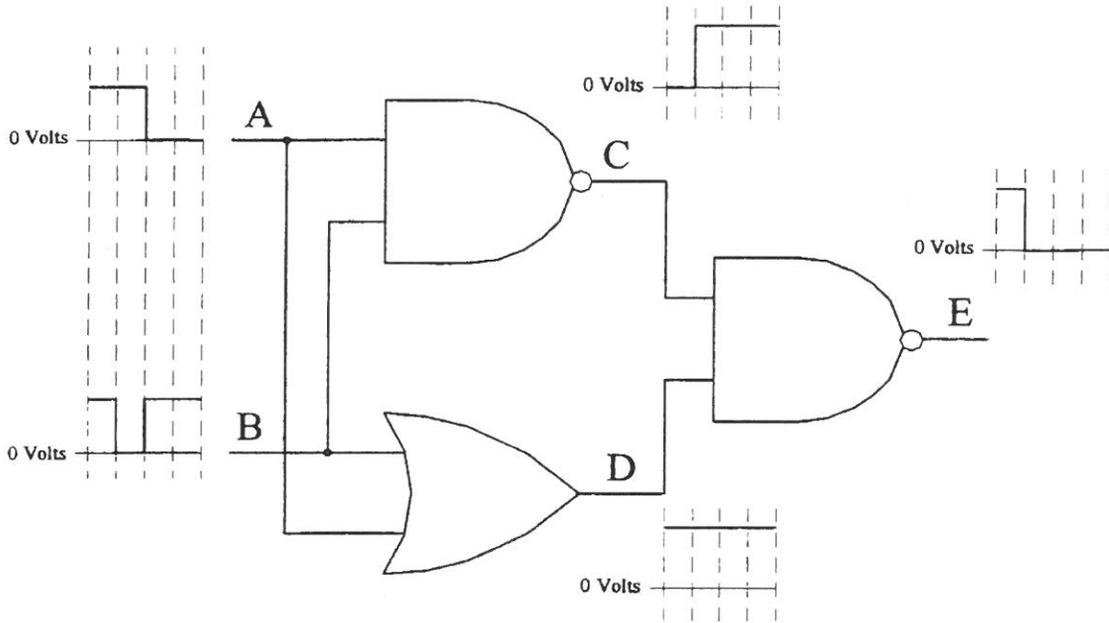
**Circuit 1 Solution:** Perform the logic function at each point of the circuit as shown.

Input A 1100  
**NAND** Input B 1011  
 = Output C 0111

Input A 1100  
**OR** Input B 1011  
 = Output D 1111

Input C 0111  
**NAND** Input D 1111  
 = Output E 1000

In the drawing below, the binary values calculated for each gate output are represented by voltage levels. For example: Output E is 1,0,0,0 or high, low, low, low



**Circuit 2 Solution:** Since there are three inputs—A, B and C—there will be eight possible input combinations from 000 to 111.

**Things to consider in this circuit**

1. There is only one time that the NAND gate output will change to a low—when A, B and C are all **high**
1. Since the X-OR input will have a high from the NAND gate output most of the time, the output of the OR gate will control the output at D for seven of the eight conditions.

Truth Table			
Inputs			Output
A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

## Unit 1 Summary

- ✓ There are three common numbering systems: Base 10 (Decimal), Base 2 (binary) and Base 16 (Hexadecimal)
- ✓ The base 10 numbering system consists of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. The Least Significant Digit (LSD) is the right-most digit.
- ✓ The base 2 numbering system consists of 0 and 1. The Least Significant Digit (LSD) is the right-most digit.
- ✓ The base 16 numbering system consists of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. It is a shorthand technique for representing binary values.
- ✓ There are techniques for converting among any of these three numbering systems.
- ✓ There are techniques for addition, subtraction, multiplication and division of these number systems.
- ✓ There are three basic logic gates: AND, OR and X-OR.
- ✓ All inputs to an AND gate must be high (binary 1) in order to have a high output.
- ✓ If any of the inputs to an OR gate are high, the output will be high.
- ✓ An X-OR gate output will be high if one, and only one, input is high.
- ✓ The number of possible combinations in a truth table for a given gate is a function of the number of inputs to the gate.

The purpose of this unit is to provide a general understanding of the subject areas addressed. For more information on the topics covered in this unit, refer to the Web sites and reference books listed in the **Study Guide for the Digital Communications and Computer Literacy Test.**

## ANSWERS TO UNIT 1 EXERCISES

### Exercise 1-1: Binary to Decimal Conversion Answers

1.  $10110111 = \underline{183}$       2.  $100111 = \underline{39}$       3.  $01110 = \underline{14}$       4.  $011010 = \underline{26}$   
 5.  $1011110010 = \underline{754}$       6.  $1001 = \underline{9}$       7.  $11100100 = \underline{228}$       8.  $100001 = \underline{33}$

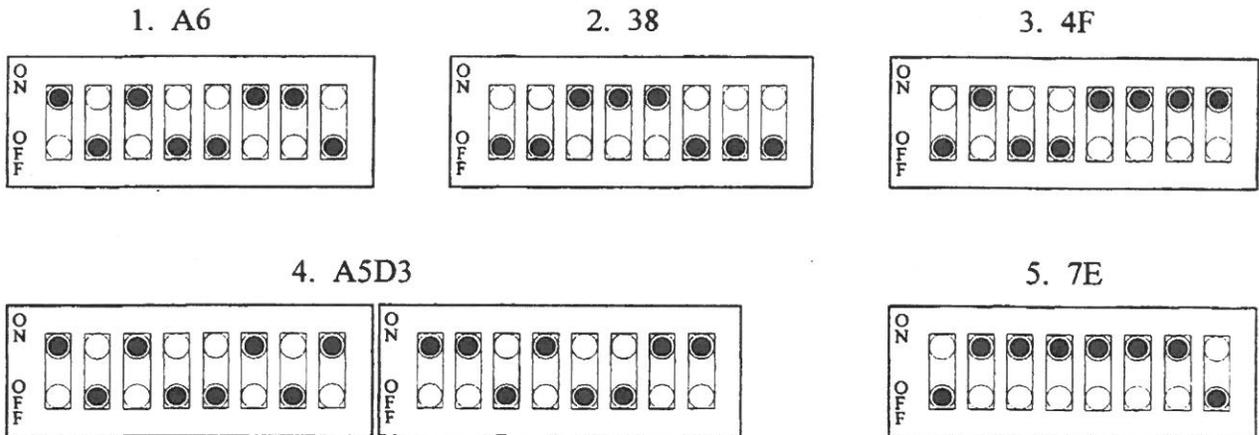
### Exercise 1-2: Decimal to Binary Conversion Answers

1.  $12 = \underline{1100}$       2.  $154 = \underline{10011010}$       3.  $86 = \underline{1010110}$       4.  $319 = \underline{100111111}$   
 5.  $27 = \underline{11011}$       6.  $63 = \underline{111111}$       7.  $79 = \underline{1001111}$       8.  $36 = \underline{100100}$

### Exercise 1-3: Binary to Hexadecimal Conversion Answers

1.  $10101011 = \underline{A6}$       2.  $001011 = \underline{B}$       3.  $0110110 = \underline{36}$       4.  $1010110010 = \underline{2B2}$   
 5.  $10101010110 = \underline{556}$       6.  $11011101 = \underline{DD}$       7.  $111001010 = \underline{1CA}$       8.  $1011010101 = \underline{2D5}$

### Exercise 1-4: Hexadecimal to Binary Conversion Answers



### Exercise 1-5: Addition of binary numbers Answers

1.  $0111 + 1011 = \underline{10010}$       2.  $01101 + 101 = \underline{10010}$       3.  $100110 + 11011 = \underline{1000001}$   
 4.  $10111 + 00111 = \underline{11110}$       5.  $11011 + 1001 = \underline{100100}$       6.  $11101 + 101001 = \underline{1000110}$

## ANSWERS TO UNIT 1 EXERCISES (cont.)

### *Exercise 1-6: Subtraction of binary numbers Answers*

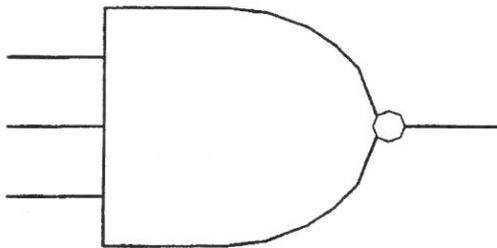
1.  $1011 - 0011 = \underline{1000}$       2.  $1101 - 0101 = \underline{1000}$       3.  $101110 - 01011 = \underline{100011}$   
 4.  $10111 - 00101 = \underline{10010}$       5.  $11011 - 1001 = \underline{10010}$       6.  $111101 - 101001 = \underline{10100}$

### *Exercise 1-7: Multiplication of binary numbers Answers*

1.  $1011 \cdot 0110 = \underline{1000010}$       2.  $0111 \cdot 1000 = \underline{111000}$   
 3.  $1111 \cdot 0001 = \underline{1111}$       4.  $101 \cdot 1110 = \underline{1000110}$

### *Exercise 1-8: Logic Functions Answers*

1. What is the type of gate illustrated below? 3-input NAND Gate  
 2. Fill-in the associated truth table.



Truth Table			
Inputs			Output
A	B	C	D
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

3. Inverting the inputs of a 2-input AND gate will change it into another the type of gate.  
 What type of gate will it effectively become? 2-input NOR Gate  
 4. Perform the *AND* function on the following binary numbers. Treating each column separately as A and B inputs, *AND* them together to a C output.

A 10011100  
 B 10001101  
 C 10001100

## ANSWERS TO UNIT 1 EXERCISES (cont.)

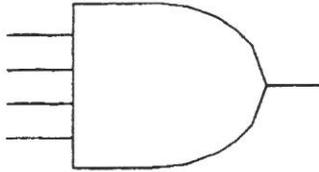
5. Perform **Modulo-2 Addition (X-OR)** on the following binary numbers—A and B.

A 10011100

B 10001101

C 00010001

6. Refer to the following logic gate.



How many input combinations will provide a **high** output state? 1

How many input combinations are possible for this gate? 16