Digital Design with CPLD Applications and VHDL

Dueck

CHAPTER 1

Basic Principles of Digital Systems

OUTLINE

CHAPTER OBJECTIVES

Upon successful completion of this chapter, you will be able to:

- Describe some differences between analog and digital electronics.
- Understand the concept of HIGH and LOW logic levels.
- Explain the basic principles of a positional notation number system.
- Translate logic HIGHs and LOWs into binary numbers.
- Count in binary, decimal, or hexadecimal.
- Convert a number in binary, decimal, or hexadecimal to any of the other number bases.
- Calculate the fractional binary equivalent of any decimal number.
- Distinguish between the most significant bit and least significant bit of a binary number.
- Describe the difference between periodic, aperiodic, and pulse waveforms.
- Calculate the frequency, period, and duty cycle of a periodic digital waveform.
- Calculate the pulse width, rise time, and fall time of a digital pulse.

Digital electronics is the branch of electronics based on the combination and switching of voltages called logic levels. Any quantity in the outside world, such as temperature, pressure, or voltage, can be symbolized in a digital circuit by a group of logic voltages that, taken together, represent a binary number. ■

Each logic level corresponds to a digit in the binary (base 2) number system. The *b*inary dig*its*, or bits, 0 and 1, are sufficient to write any number, given enough places. The hexadecimal (base 16) number system is also important in digital systems. Since every combination of four binary digits can be uniquely represented as a hexadecimal digit, this system is often used as a compact way of writing binary information.

Inputs and outputs in digital circuits are not always static. Often they vary with time. Time-varying digital waveforms can have three forms:

1. Periodic waveforms, which repeat a pattern of logic 1s and 0s
2. Aperiodic waveforms, which do not repeat
3. Pulse waveforms, which produce a momentary variation from a constant logic level

1.1 Digital Versus Analog Electronics

KEY TERMS

Continuous Smoothly connected. An unbroken series of consecutive values with no instantaneous changes.

Discrete Separated into distinct segments or pieces. A series of discontinuous values.

Analog A way of representing some physical quantity, such as temperature or velocity, by a proportional continuous voltage or current. An analog voltage or current can have any value within a defined range.

Digital A way of representing a physical quantity by a series of binary numbers. A digital representation can have only specific discrete values.

The study of electronics often is divided into two basic areas: **analog** and **digital** electronics. Analog electronics has a longer history and can be regarded as the "classical" branch of electronics. Digital electronics, although newer, has achieved greater prominence through the advent of the computer age. The modern revolution in microcomputer chips, as part of everything from personal computers to cars and coffee makers, is founded almost entirely on digital electronics.

The main difference between analog and digital electronics can be stated simply. Analog voltages or currents are **continuously** variable between defined values, and digital voltages or currents can vary only by distinct, or **discrete,** steps.

Some keywords highlight the differences between digital and analog electronics:

Analog	**Digital**
Continuously variable	Discrete steps
Amplification	Switching
Voltages	Numbers

An example often used to illustrate the difference between analog and digital devices is the comparison between a light dimmer and a light switch. A light dimmer is an analog device, since it can make the light it controls vary in brightness anywhere within a defined range of values. The light can be fully on, fully off, or at some brightness level in between. A light switch is a digital device, since it can turn the light on or off, but there is no value in between those two states.

The light switch/light dimmer analogy, although easy to understand, does not show any particular advantage to the digital device. If anything, it makes the digital device seem limited.

One modern application in which a digital device is clearly superior to an analog one is digital audio reproduction. Compact disc players have achieved their high level of popularity because of the accurate and noise-free way in which they reproduce recorded music. This high quality of sound is possible because the music is stored, not as a magnetic copy of the sound vibrations, as in analog tapes, but as a series of numbers that represent amplitude steps in the sound waves.

Figure 1.1 shows a sound waveform and its representation in both analog and digital forms.

The analog voltage, shown in Figure 1.1b, is a copy of the original waveform and introduces distortion both in the storage and playback processes. (Think of how a photocopy deteriorates in quality if you make a copy of a copy, then a copy of the new copy, and so on. It doesn't take long before you can't read the fine print.)

A digital audio system doesn't make a copy of the waveform, but rather stores a code (a series of amplitude numbers) that tells the compact disc player how to re-create the original sound every time a disc is played. During the recording process, the sound waveform

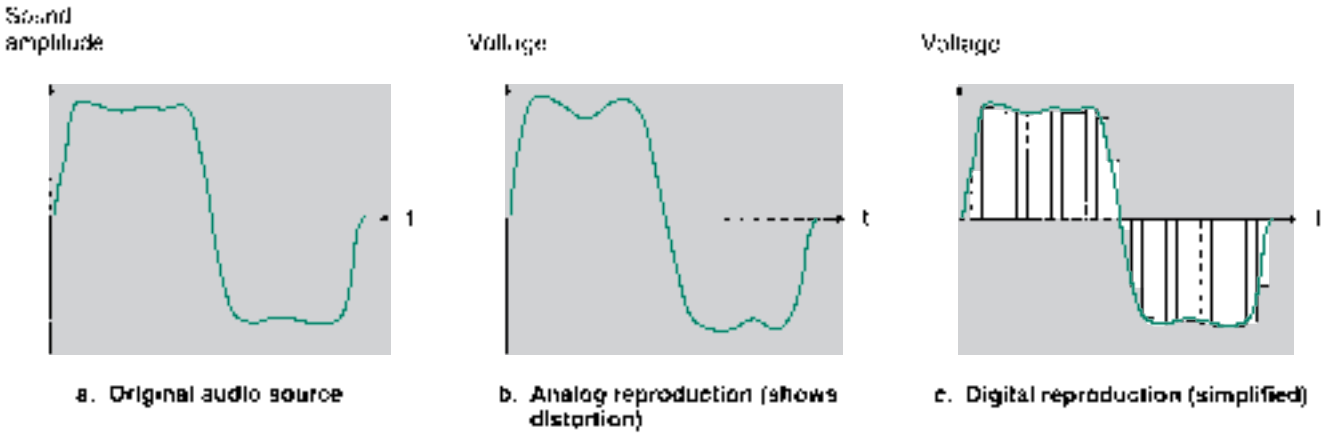


FIGURE 1.1
Digital and Analog Sound Reproduction

is "sampled" at precise intervals. The recording transforms each sample into a digital number corresponding to the amplitude of the sound at that point.

The "samples" (the voltages represented by the vertical bars) of the digitized audio waveform shown in Figure 1.1c are much more widely spaced than they would be in a real digital audio system. They are shown this way to give the general idea of a digitized waveform. In real digital audio systems, each amplitude value can be indicated by a number having as many as 16,000 to 65,000 possible values. Such a large number of possible values means the voltage difference between any two consecutive digital numbers is very small. The numbers can thus correspond extremely closely to the actual amplitude of the sound waveform. If the spacing between the samples is made small enough, the reproduced waveform is almost exactly the same as the original.

SECTION 1.1 REVIEW PROBLEM

1.1 What is the basic difference between analog and digital audio reproduction?

1.2 Digital Logic Levels

KEY TERMS

Logic level A voltage level that represents a defined digital state in an electronic circuit.

Logic HIGH (or **logic 1**) The higher of two voltages in a digital system with two logic levels.

Logic LOW (or **logic 0**) The lower of two voltages in a digital system with two logic levels.

Positive logic A system in which logic LOW represents binary digit 0 and logic HIGH represents binary digit 1.

Negative logic A system in which logic LOW represents binary digit 1 and logic HIGH represents binary digit 0.

Digitally represented quantities, such as the amplitude of an audio waveform, are usually represented by binary, or base 2, numbers. When we want to describe a digital quantity electronically, we need to have a system that uses voltages or currents to symbolize binary numbers.

The binary number system has only two digits, 0 and 1. Each of these digits can be denoted by a different voltage called a **logic level.** For a system having two logic levels, the

lower voltage (usually 0 volts) is called a **logic LOW** or **logic 0** and represents the digit 0. The higher voltage (traditionally 5 V, but in some systems a specific value such as 1.8 V, 2.5 V or 3.3 V) is called a **logic HIGH** or **logic 1,** which symbolizes the digit 1. Except for some allowable tolerance, as shown in Figure 1.2, the range of voltages between HIGH and LOW logic levels is undefined.

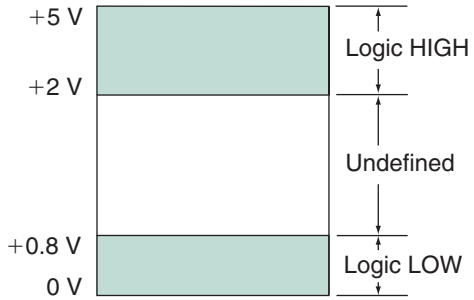


FIGURE 1.2
Logic Levels Based on +5 V and 0 V

NOTE

For the voltages in Figure 1.2:

+5 V = Logic HIGH = 1
0 V = Logic LOW = 0

The system assigning the digit 1 to a logic HIGH and digit 0 to logic LOW is called **positive logic.** Throughout the remainder of this text, logic levels will be referred to as HIGH/LOW or 1/0 interchangeably.

(A complementary system, called **negative logic,** also exists that makes the assignment the other way around.)

1.3 The Binary Number System

KEY TERMS

Binary number system A number system used extensively in digital systems, based on the number 2. It uses two digits, 0 and 1, to write any number.

Positional notation A system of writing numbers where the value of a digit depends not only on the digit, but also on its placement within a number.

Bit *B*inary dig*it*. A 0 or a 1.

Positional Notation

The **binary number system** is based on the number 2. This means that we can write any number using only two binary digits (or **bits**), 0 and 1. Compare this to the decimal system, which is based on the number 10, where we can write any number with only ten decimal digits, 0 to 9.

The binary and decimal systems are both **positional notation** systems; the value of a digit in either system depends on its placement within a number. In the decimal number 845, the digit 4 really means 40, whereas in the number 9426, the digit 4 really means 400 (845 = 800 + 40 + 5; 9426 = 9000 + 400 + 20 + 6). The value of the digit is determined by *what* the digit is as well as *where* it is.

In the decimal system, a digit in the position immediately to the left of the decimal point is multiplied by 1 (10^0). A digit two positions to the left of the decimal point is mul-

tiplied by 10 (10^1). A digit in the next position left is multiplied by 100 (10^2). The positional multipliers, as you move left from the decimal point, are ascending powers of 10.

The same idea applies in the binary system, except that the positional multipliers are powers of 2 ($2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 32, \ldots$). For example, the binary number 101 has the decimal equivalent:

$$\begin{aligned} & (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ = & (1 \times 4) + (0 \times 2) + (1 \times 1) \\ = & \quad 4 \quad + \quad 0 \quad + \quad 1 \\ = & \quad 5 \end{aligned}$$

EXAMPLE 1.1

Calculate the decimal equivalents of the binary numbers 1010, 111, and 10010.

SOLUTIONS

$$\begin{aligned} 1010 &= (1\times2^3) + (0\times2^2) + (1\times2^1) + (0\times2^0) \\ &= (1\times8) + (0\times4) + (1\times2) + (0\times1) \\ &= 8 + 2 = 10 \\ 111 &= (1\times2^2) + (1\times2^1) + (1\times2^0) \\ &= (1\times4) + (1\times2) + (1\times1) \\ &= 4 + 2 + 1 = 7 \\ 10010 &= (1\times2^4) + (0\times2^3) + (0\times2^2) + (1\times2^1) + (0\times2^0) \\ &= (1\times16) + (0\times8) + (0\times4) + (1\times2) + (0\times1) \\ &= 16 + 2 = 18 \end{aligned}$$

Binary Inputs

KEY TERMS

Most significant bit The leftmost bit in a binary number. This bit has the number's largest positional multiplier.

Least significant bit The rightmost bit of a binary number. This bit has the number's smallest positional multiplier.

A major class of digital circuits, called combinational logic, operates by accepting logic levels at one or more input terminals and producing a logic level at an output. In the analysis and design of such circuits, it is frequently necessary to find the output logic level of a circuit for all possible combinations of input logic levels.

The digital circuit in the black box in Figure 1.3 has three inputs. Each input can have two possible states, LOW or HIGH, which can be represented by positive logic as 0 or 1. The number of possible input combinations is $2^3 = 8$. (In general, a circuit with n binary inputs has 2^n input combinations, ranging from 0 to $2^n - 1$.) Table 1.1 shows a list of these combinations, both as logic levels and binary numbers, and their decimal equivalents.

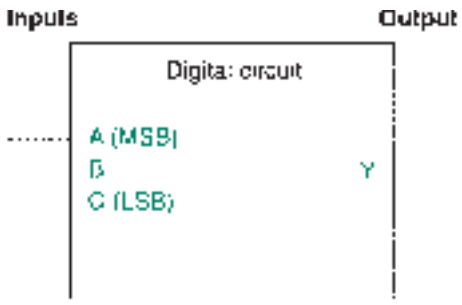


FIGURE 1.3
3-Input Digital Circuit

TABLE 1.1 Possible Input Combinations for a 3-Input Digital Circuit

Logic Level			Binary Value			Decimal Equivalent
A	*B*	*C*	*A*	*B*	*C*	
L	L	L	0	0	0	0
L	L	H	0	0	1	1
L	H	L	0	1	0	2
L	H	H	0	1	1	3
H	L	L	1	0	0	4
H	L	H	1	0	1	5
H	H	L	1	1	0	6
H	H	H	1	1	1	7

A list of output logic levels corresponding to all possible input combinations, applied in ascending binary order, is called a truth table. This is a standard form for showing the function of a digital circuit.

The input bits on each line of Table 1.1 can be read from left to right as a series of 3-bit binary numbers. The numerical values of these eight input combinations range from 0 to 7 (2^n possible input combinations, having decimal equivalents ranging from 0 to 2^n-1) in decimal.

Bit *A* is called the **most significant bit** (MSB), and bit *C* is called the **least significant bit** (LSB). As these terms imply, a change in bit *A* is more significant, since it has the greatest effect on the number of which it is part.

Table 1.2 shows the effect of changing each of these bits in a 3-bit binary number and compares the changed number to the original by showing the difference in magnitude. A change in the MSB of any 3-bit number results in a difference of 4. A change in the LSB of any binary number results in a difference of 1. (Try it with a few different numbers.)

TABLE 1.2 Effect of Changing the LSB and MSB of a Binary Number

	A	*B*	*C*	Decimal	
Original	0	1	1	3	
Change MSB	1	1	1	7	Difference = 4
Change LSB	0	1	0	2	Difference = 1

EXAMPLE 1.2

Figure 1.4 shows a 4-input digital circuit. List all the possible binary input combinations to this circuit and their decimal equivalents. What is the value of the MSB?

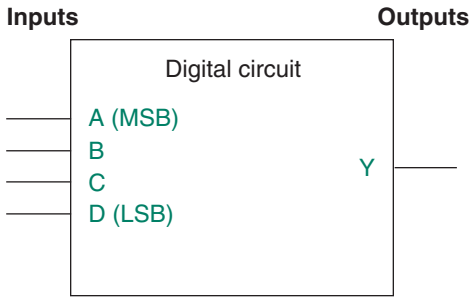


FIGURE 1.4
Example 1.2: 4-Input Digital Circuit

SOLUTION Since there are four inputs, there will be $2^4 = 16$ possible input combinations, ranging from 0000 to 1111 (0 to 15 in decimal). Table 1.3 shows the list of all possible input combinations.

The MSB has a value of 8 (decimal).

TABLE 1.3 Possible Input Combinations for a 4-Input Digital Circuit

A	*B*	*C*	*D*	Decimal
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Knowing how to construct a binary sequence is a very important skill when working with digital logic systems. Two ways to do this are:

1. *Learn to count in binary.* You should know all the binary numbers from 0000 to 1111 and their decimal equivalents (0 to 15). *Make this your first goal in learning the basics of digital systems.*

   Each binary number is a unique representation of its decimal equivalent. You can work out the decimal value of a binary number by adding the weighted values of all the bits.

   For instance, the binary equivalent of the decimal sequence 0, 1, 2, 3 can be written using two bits: the 1's bit and the 2's bit. The binary count sequence is:

   $$00\ (= 0 + 0)$$
   $$01\ (= 0 + 1)$$
   $$10\ (= 2 + 0)$$
   $$11\ (= 2 + 1)$$

   To count beyond this, you need another bit: the 4's bit. The decimal sequence 4, 5, 6, 7 has the binary equivalents:

   $$100\ (= 4 + 0 + 0)$$
   $$101\ (= 4 + 0 + 1)$$
   $$110\ (= 4 + 2 + 0)$$
   $$111\ (= 4 + 2 + 1)$$

   The two least significant bits of this sequence are the same as the bits in the 0 to 3 sequence; a repeating pattern has been generated.

The sequence from 8 to 15 requires yet another bit: the 8's bit. The three LSBs of this sequence repeat the 0 to 7 sequence. The binary equivalents of 8 to 15 are:

$$1000\ (=8+0+0+0)$$
$$1001\ (=8+0+0+1)$$
$$1010\ (=8+0+2+0)$$
$$1011\ (=8+0+2+1)$$
$$1100\ (=8+4+0+0)$$
$$1101\ (=8+4+0+1)$$
$$1110\ (=8+4+2+0)$$
$$1111\ (=8+4+2+1)$$

Practice writing out the binary sequence until it becomes familiar. In the 0 to 15 sequence, it is standard practice to write each number as a 4-bit value, as in Example 1.2, so that all numbers have the same number of bits. Numbers up to 7 have leading zeros to pad them out to 4 bits.

This convention has developed because each bit has a physical location in a digital circuit; we know a particular bit is logic 0 because we can measure 0 V at a particular point in a circuit. A bit with a value of 0 doesn't go away just because there is not a 1 at a more significant location.

While you are still learning to count in binary, you can use a second method.

2. *Follow a simple repetitive pattern.* Look at Tables 1.1 and 1.3 again. Notice that the least significant bit follows a pattern. The bits alternate with every line, producing the pattern 0, 1, 0, 1, ... The 2's bit alternates every two lines: 0, 0, 1, 1, 0, 0, 1, 1, ... The 4's bit alternates every four lines: 0, 0, 0, 0, 1, 1, 1, 1, ... This pattern can be expanded to cover any number of bits, with the number of lines between alternations doubling with each bit to the left.

Decimal-to-Binary Conversion

There are two methods commonly used to convert decimal numbers to binary: sum of powers of 2 and repeated division by 2.

Sum of Powers of 2

You can convert a decimal number to binary by adding up powers of 2 by inspection, adding bits as you need them to fill up the total value of the number. For example, convert 57_{10} to binary.

$$64_{10} > 57_{10} > 32_{10}$$

- We see that 32 ($=2^5$) is the largest power of two that is smaller than 57. Set the 32's bit to 1 and subtract 32 from the original number, as shown below.

$$57 - 32 = 25$$

- The largest power of two that is less than 25 is 16. Set the 16's bit to 1 and subtract 16 from the accumulated total.

$$25 - 16 = 9$$

- 8 is the largest power of two that is less than 9. Set the 8's bit to 1 and subtract 8 from the total.

$$9 - 8 = 1$$

- 4 is greater than the remaining total. Set the 4's bit to 0.
- 2 is greater than the remaining total. Set the 2's bit to 0.

- 1 is left over. Set the 1's bit to 1 and subtract 1.

$$1 - 1 = 0$$

- Conversion is complete when there is nothing left to subtract. Any remaining bits should be set to 0.

32	16	8	4	2	1	
1						57 − 32 = 25
1	1					57 − (32 + 16) = 9
1	1	1				57 − (32 + 16 + 8) = 1
1	1	1	0	0	1	57 − (32 + 16 + 8 + 1) = 0

$$57_{10} = 111001_2$$

EXAMPLE 1.3 Convert 92_{10} to binary using the sum-of-powers-of-2 method.

SOLUTION $128 > 92 > 64$

64	32	16	8	4	2	1	
1							92 − 64 = 28
1	0	1					92 − (64 + 16) = 12
1	0	1	1				92 − (64 + 16 + 8) = 4
1	0	1	1	1	0	0	92 − (64 + 16 + 8 + 4) = 0

$$92_{10} = 1011100_2$$

Repeated Division by 2

Any decimal number divided by 2 will leave a remainder of 0 or 1. Repeated division by 2 will leave a string of 0s and 1s that become the binary equivalent of the decimal number. Let us use this method to convert 46_{10} to binary.

1. Divide the decimal number by 2 and note the remainder.

$$46/2 = 23 + \text{remainder } 0 \text{ (LSB)}$$

   The remainder is the least significant bit of the binary equivalent of 46.

2. Divide the quotient from the previous division and note the remainder. The remainder is the second LSB.

$$23/2 = 11 + \text{remainder } 1$$

3. Continue this process until the *quotient* is 0. The last remainder is the most significant bit of the binary number.

$$11/2 = 5 + \text{remainder } 1$$
$$5/2 = 2 + \text{remainder } 1$$
$$2/2 = 1 + \text{remainder } 0$$
$$1/2 = 0 + \text{remainder } 1 \qquad \text{(MSB)}$$

To write the binary equivalent of the decimal number, read the remainders from the bottom up.

$$46_{10} = 101110_2$$

EXAMPLE 1.4

Use repeated division by 2 to convert 115_{10} to a binary number.

SOLUTION

$$115/2 = 57 + \text{remainder } 1 \text{ (LSB)}$$
$$57/2 = 28 + \text{remainder } 1$$
$$28/2 = 14 + \text{remainder } 0$$
$$14/2 = 7 + \text{remainder } 0$$
$$7/2 = 3 + \text{remainder } 1$$
$$3/2 = 1 + \text{remainder } 1$$
$$1/2 = 0 + \text{remainder } 1 \text{ (MSB)}$$

Read the remainders from bottom to top: 1110011.

$$115_{10} = 1110011_2$$

In any decimal-to-binary conversion, the number of bits in the binary number is the exponent of the smallest power of 2 that is larger than the decimal number.

For example, for the numbers 92_{10} and 46_{10},

$2^7 = 128 > 92$ 7 bits: 1011100
$2^6 = 64 > 46$ 6 bits: 101110

Fractional Binary Numbers

KEY TERMS

Radix point The generalized form of a decimal point. In any positional number system, the radix point marks the dividing line between positional multipliers that are positive and negative powers of the system's number base.

Binary point A period (".") that marks the dividing line between positional multipliers that are positive and negative powers of 2 (e.g., first multiplier right of binary point = 2^{-1}; first multiplier left of binary point = 2^0).

In the decimal system, fractional numbers use the same digits as whole numbers, but the digits are written to the right of the decimal point. The multipliers for these digits are negative powers of 10—10^{-1} (1/10), 10^{-2} (1/100), 10^{-3} (1/1000), and so on.

So it is in the binary system. Digits 0 and 1 are used to write fractional binary numbers, but the digits are to the right of the **binary point**—the binary equivalent of the decimal point. (The decimal point and binary point are special cases of the **radix point,** the general name for any such point in any number system.)

Each digit is multiplied by a positional factor that is a negative power of 2. The first four multipliers on either side of the binary point are:

				binary point				
2^3	2^2	2^1	2^0	·	2^{-1}	2^{-2}	2^{-3}	2^{-4}
$= 8$	$= 4$	$= 2$	$= 1$		$= 1/2$	$= 1/4$	$= 1/8$	$= 1/16$

EXAMPLE 1.5 Write the binary fraction 0.101101 as a decimal fraction.

SOLUTION

$$1 \times 1/2 = 1/2$$
$$0 \times 1/4 = 0$$
$$1 \times 1/8 = 1/8$$
$$1 \times 1/16 = 1/16$$
$$0 \times 1/32 = 0$$
$$1 \times 1/64 = 1/64$$

$$\begin{aligned} 1/2 + 1/8 + 1/16 + 1/64 &= 32/64 + 8/64 + 4/64 + 1/64 \\ &= 45/64 \\ &= 0.703125_{10} \end{aligned}$$

Fractional-Decimal-to-Fractional-Binary Conversion

Simple decimal fractions such as 0.5, 0.25, and 0.375 can be converted to binary fractions by a sum-of-powers method. The above decimal numbers can also be written $0.5 = 1/2$, $0.25 = 1/4$, and $0.375 = 3/8 = 1/4 + 1/8$. These numbers can all be represented by negative powers of 2. Thus, in binary,

$$0.5_{10} = 0.1_2$$
$$0.25_{10} = 0.01_2$$
$$0.375_{10} = 0.011_2$$

The conversion process becomes more complicated if we try to convert decimal fractions that cannot be broken into powers of 2. For example, the number $1/5 = 0.2_{10}$ cannot be exactly represented by a sum of negative powers of 2. (Try it.) For this type of number, we must use the method of repeated multiplication by 2.

Method:

1. Multiply the decimal fraction by 2 and note the integer part. The integer part is either 0 or 1 for any number between 0 and 0.999. . . . The integer part of the product is the first digit to the left of the binary point.

$$0.2 \times 2 = 0.4 \qquad \text{Integer part: } 0$$

2. Discard the integer part of the previous product. Multiply the fractional part of the previous product by 2. Repeat step 1 until the fraction repeats or terminates.

$$0.4 \times 2 = 0.8 \qquad \text{Integer part: } 0$$
$$0.8 \times 2 = 1.6 \qquad \text{Integer part: } 1$$
$$0.6 \times 2 = 1.2 \qquad \text{Integer part: } 1$$
$$0.2 \times 2 = 0.4 \qquad \text{Integer part: } 0$$

(Fraction repeats; product is same as in step 1)

Read the above integer parts from top to bottom to obtain the fractional binary number. Thus, $0.2_{10} = 0.00110011\ldots_2 = 0.\overline{0011}_2$. The bar shows the portion of the digits that repeats.

EXAMPLE 1.6

Convert 0.95_{10} to its binary equivalent.

SOLUTION

$0.95 \times 2 = 1.90$ Integer part: 1
$0.90 \times 2 = 1.80$ Integer part: 1
$0.80 \times 2 = 1.60$ Integer part: 1
$0.60 \times 2 = 1.20$ Integer part: 1
$0.20 \times 2 = 0.40$ Integer part: 0
$0.40 \times 2 = 0.80$ Integer part: 0
$0.80 \times 2 = 1.60$ Fraction repeats last four digits

$$0.95_{10} = 0.\overline{111100}_2$$

SECTION 1.3 REVIEW PROBLEMS

1.2. How many different binary numbers can be written with 6 bits?
1.3. How many can be written with 7 bits?
1.4. Write the sequence of 7-bit numbers from 1010000 to 1010111.
1.5. Write the decimal equivalents of the numbers written for Problem 1.4.

1.4 Hexadecimal Numbers

After binary numbers, hexadecimal (base 16) numbers are the most important numbers in digital applications. Hexadecimal, or hex, numbers are primarily used as a shorthand form of binary notation. Since 16 is a power of 2 ($2^4 = 16$), each hexadecimal digit can be converted directly to four binary digits. Hex numbers can pack more digital information into fewer digits.

Hex numbers have become particularly popular with the advent of small computers, which use binary data having 8, 16, or 32 bits. Such data can be represented by 2, 4, or 8 hexadecimal digits, respectively.

TABLE 1.4 Hex Digits and Their Binary and Decimal Equivalents

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

Counting in Hexadecimal

The positional multipliers in the hex system are powers of sixteen: $16^0 = 1$, $16^1 = 16$, $16^2 = 256$, $16^3 = 4096$, and so on.

We need 16 digits to write hex numbers; the decimal digits 0 through 9 are not sufficient. The usual convention is to use the capital letters A through F, each letter representing a number from 10_{10} through 15_{10}. Table 1.4 shows how hexadecimal digits relate to their decimal and binary equivalents.

NOTE

Counting Rules for Hexadecimal Numbers:

1. Count in sequence from 0 to F in the least significant digit.
2. Add 1 to the next digit to the left and start over.
3. Repeat in all other columns.

For instance, the hex numbers between 19 and 22 are 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, 22. (The decimal equivalents of these numbers are 25_{10} through 34_{10}.)

EXAMPLE 1.7

What is the next hexadecimal number after 999? After 99F? After 9FF? After FFF?

SOLUTION The hexadecimal number after 999 is 99A. The number after 99F is 9A0. The number after 9FF is A00. The number after FFF is 1000.

EXAMPLE 1.8

List the hexadecimal digits from 190_{16} to 200_{16}, inclusive.

SOLUTION The numbers follow the counting rules: Use all the digits in one position, add 1 to the digit one position left, and start over.

For brevity, we will list only a few of the numbers in the sequence:

190, 191, 192, . . . , 199, 19A, 19B, 19C, 19D, 19E, 19F,
1A0, 1A1, 1A2, . . . , 1A9, 1AA, 1AB, 1AC, 1AD, 1AE, 1AF,
1B0, 1B1, 1B2, . . . , 1B9, 1BA, 1BB, 1BC, 1BD, 1BE, 1BF,
1C0, . . . , 1CF, 1D0, . . . , 1DF, 1E0, . . . , 1EF, 1F0, . . . , 1FF, 200

SECTION 1.4A REVIEW PROBLEMS

1.6. List the hexadecimal numbers from FA9 to FB0, inclusive.

1.7. List the hexadecimal numbers from 1F9 to 200, inclusive.

Hexadecimal-to-Decimal Conversion

To convert a number from hex to decimal, multiply each digit by its power-of-16 positional multiplier and add the products. In the following examples, hexadecimal numbers are indicated by a final "H" (e.g., 1F7H), rather than a "16" subscript.

EXAMPLE 1.9

Convert 7C6H to decimal.

SOLUTION

$$\begin{aligned} 7 \times 16^2 &= 7_{10} \times 256_{10} = 1792_{10} \\ \mathrm{C} \times 16^1 &= 12_{10} \times 16_{10} = 192_{10} \\ 6 \times 16^0 &= 6_{10} \times 1_{10} = 6_{10} \\ &\qquad\qquad\qquad\qquad 1990_{10} \end{aligned}$$

EXAMPLE 1.10

Convert 1FD5H to decimal.

SOLUTION

$$\begin{aligned} 1 \times 16^3 &= 1_{10} \times 4096_{10} = 4096_{10} \\ \mathrm{F} \times 16^2 &= 15_{10} \times 256_{10} = 3840_{10} \\ \mathrm{D} \times 16^1 &= 13_{10} \times 16_{10} = 208_{10} \\ 5 \times 16^0 &= 5_{10} \times 1_{10} = 5_{10} \\ &\qquad\qquad\qquad\qquad 8149_{10} \end{aligned}$$

SECTION 1.4B REVIEW PROBLEM

1.8 Convert the hexadecimal number A30F to its decimal equivalent.

Decimal-to-Hexadecimal Conversion

Decimal numbers can be converted to hex by the sum-of-weighted-hex-digits method or by repeated division by 16. The main difficulty we encounter in either method is

remembering to convert decimal numbers 10 through 15 into the equivalent hex digits, A through F.

Sum of Weighted Hexadecimal Digits

This method is useful for simple conversions (about three digits). For example, the decimal number 35 is easily converted to the hex value 23.

$$35_{10} = 32_{10} + 3_{10} = (2 \times 16) + (3 \times 1) = 23\text{H}$$

EXAMPLE 1.11

Convert 175_{10} to hexadecimal.

SOLUTION

$$256_{10} > 175_{10} > 16_{10}$$

Since $256 = 16^2$, the hexadecimal number will have two digits.

$$(11 \times 16) > 175 > (10 \times 16)$$

16	1
A	

$$175 - (\text{A} \times 16) = 175 - 160 = 15$$

16	1
A	F

$$175 - ((\text{A} \times 16) + (\text{F} \times 1))$$
$$= 175 - (160 + 15) = 0$$

Repeated Division by 16

Repeated division by 16 is a systematic decimal-to-hexadecimal conversion method that is not limited by the size of the number to be converted.

It is similar to the repeated-division-by-2 method used to convert decimal numbers to binary. Divide the decimal number by 16 and note the remainder, making sure to express it as a hex digit. Repeat the process until the quotient is zero. The last remainder is the most significant digit of the hex number.

EXAMPLE 1.12

Convert 31581_{10} to hexadecimal.

SOLUTION

31581/16 = 1973 + remainder 13 (D) (LSD)
1973/16 = 123 + remainder 5
123/16 = 7 + remainder 11 (B)
7/16 = 0 + remainder 7 (MSD)

$$31581_{10} = 7\text{B5DH}$$

SECTION 1.4C REVIEW PROBLEM

1.9 Convert the decimal number 8137 to its hexadecimal equivalent.

Conversions Between Hexadecimal and Binary

Table 1.4 shows all 16 hexadecimal digits and their decimal and binary equivalents. Note that for every possible 4-bit binary number, there is a hexadecimal equivalent.

Binary-to-hex and hex-to-binary conversions simply consist of making a conversion between each hex digit and its binary equivalent.

EXAMPLE 1.13 Convert 7EF8H to its binary equivalent.

SOLUTION Convert each digit individually to its equivalent value:

$$7H = 0111_2$$
$$EH = 1110_2$$
$$FH = 1111_2$$
$$8H = 1000_2$$

The binary number is all the above binary numbers in sequence:

$$7EF8H = 111111011111000_2$$

The leading zero (the MSB of 0111) has been left out.

SECTION 1.4D REVIEW PROBLEMS

1.10 Convert the hexadecimal number 934B to binary.

1.11 Convert the binary number 11001000001101001001 to hexadecimal.

1.5 Digital Waveforms

KEY TERM

Digital waveform A series of logic 1s and 0s plotted as a function of time.

The inputs and outputs of digital circuits often are not fixed logic levels but **digital waveforms,** where the input and output logic levels vary with time. There are three possible types of digital waveform. *Periodic* waveforms repeat the same pattern of logic levels over a specified period of time. *Aperiodic* waveforms do not repeat. *Pulse* waveforms follow a HIGH-LOW-HIGH or LOW-HIGH-LOW pattern and may be periodic or aperiodic.

Periodic Waveforms

KEY TERMS

Periodic waveform A time-varying sequence of logic HIGHs and LOWs that repeats over a specified period of time.

Period *(T)* Time required for a periodic waveform to repeat. Unit: seconds *(s)*.

Frequency *(f)* Number of times per second that a periodic waveform repeats. $f = 1/T$ Unit: Hertz (Hz).

Time HIGH *(t_h)* Time during one period that a waveform is in the HIGH state. Unit: seconds *(s)*.

Time LOW *(t_l)* Time during one period that a waveform is in the LOW state. Unit: seconds *(s)*.

Duty cycle *(DC)* Fraction of the total period that a digital waveform is in the HIGH state. $DC = t_h/T$ (often expressed as a percentage: $\%DC = t_h/T \times 100\%$).

Periodic waveforms repeat the same pattern of HIGHs and LOWs over a specified period of time. The waveform may or may not be symmetrical; that is, it may or may not be HIGH and LOW for equal amounts of time.

EXAMPLE 1.14

Calculate the **time LOW, time HIGH, period, frequency,** and **percent duty cycle** for each of the periodic waveforms in Figure 1.5.

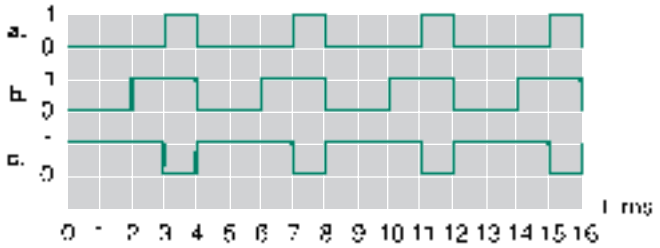


FIGURE 1.5
Example 1.14: Periodic Digital Waveforms

How are the waveforms similar? How do they differ?

SOLUTION

a. Time LOW: $t_l = 3$ ms
Time HIGH: $t_h = 1$ ms
Period: $T = t_l + t_h = 3$ ms $+ 1$ ms $= 4$ ms
Frequency: $f = 1/T = 1/(4\text{ ms}) = 0.25$ kHz $= 250$ Hz
Duty cycle: $\%DC = (t_h/T) \times 100\% = (1\text{ ms}/4\text{ ms}) \times 100\%$
$= 25\%$
(1 ms = 1/1000 second; 1 kHz = 1000 Hz.)

b. Time LOW: $t_l = 2$ ms
Time HIGH: $t_h = 2$ ms
Period: $T = t_l + t_h = 2$ ms $+ 2$ ms $= 4$ ms
Frequency: $f = 1/T = 1/(4\text{ ms}) = 0.25$ kHz $= 250$ Hz
Duty cycle: $\%DC = (t_h/T) \times 100\% = (2\text{ ms}/4\text{ ms}) \times 100\%$
$= 50\%$

c. Time LOW: $t_l = 1$ ms
Time HIGH: $t_h = 3$ ms
Period: $T = t_l + t_h = 1$ ms $+ 3$ ms $= 4$ ms
Frequency: $f = 1/T = 1/(4\text{ ms}) = 0.25$ kHz and 250 Hz
Duty cycle: $\%DC = (t_h/T) \times 100\% = (3\text{ ms}/4\text{ ms}) \times 100\%$
$= 75\%$

The waveforms all have the same period but different duty cycles. A square waveform, shown in Figure 1.5b, has a duty cycle of 50%.

Aperiodic Waveforms

KEY TERM

Aperiodic waveform A time-varying sequence of logic HIGHs and LOWs that does not repeat.

An **aperiodic waveform** does not repeat a pattern of 0s and 1s. Thus, the parameters of time HIGH, time LOW, frequency, period, and duty cycle have no meaning for an aperiodic waveform. Most waveforms of this type are one-of-a-kind specimens. (It is also worth noting that most digital waveforms are aperiodic.)

Figure 1.6 shows some examples of aperiodic waveforms.

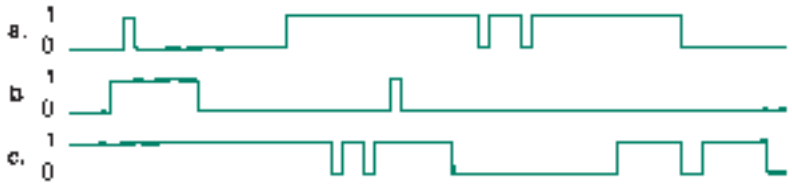


FIGURE 1.6
Aperiodic Digital Waveforms

EXAMPLE 1.15

A digital circuit generates the following strings of 0s and 1s:

a. 00111111011010110100000110000
b. 00110011001100110011001100110011
c. 00000000111111110000000001111
d. 10111011101110111011101110111011

The time between two bits is always the same. Sketch the resulting digital waveform for each string of bits. Which waveforms are periodic and which are aperiodic?

SOLUTION Figure 1.7 shows the waveforms corresponding to the strings of bits above. The waveforms are easier to draw if you break up the bit strings into smaller groups of, say, 4 bits each. For instance:

a. 0011 1111 0110 1011 0100 0011 0000

All of the waveforms except Figure 1.7a are periodic.

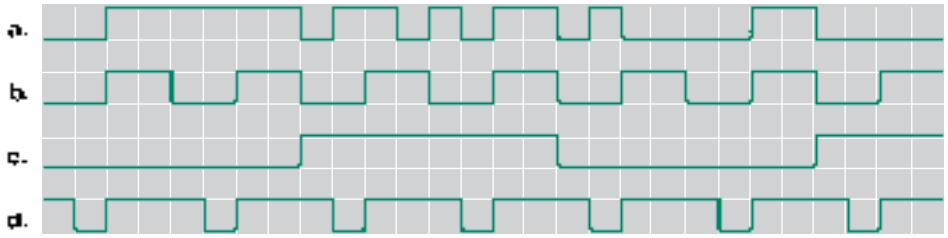


FIGURE 1.7
Example 1.15: Waveforms

Pulse Waveforms

KEY TERMS

Pulse A momentary variation of voltage from one logic level to the opposite level and back again.

Amplitude The instantaneous voltage of a waveform. Often used to mean maximum amplitude, or peak voltage, of a pulse.

Edge The part of the pulse that represents the transition from one logic level to the other.

Rising edge The part of a pulse where the logic level is in transition from a LOW to a HIGH.

Falling edge The part of a pulse where the logic level is a transition from a HIGH to a LOW.

Leading edge The edge of a pulse that occurs earliest in time.

Trailing edge The edge of a pulse that occurs latest in time.

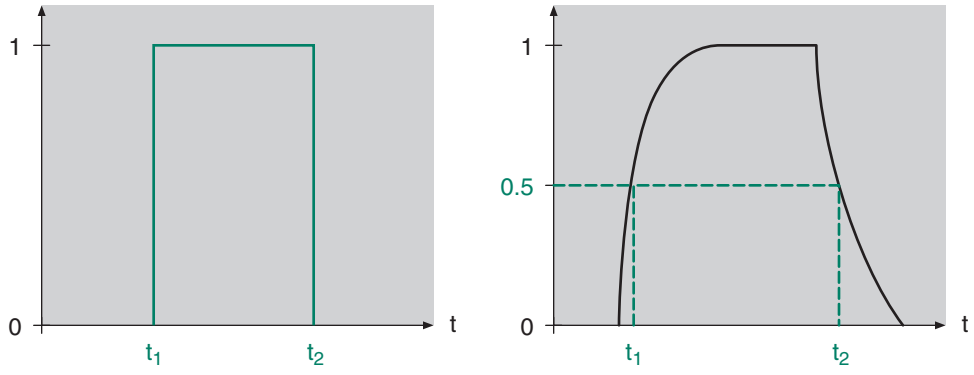
Pulse width (t_w) Elapsed time from the 50% point of the leading edge of a pulse to the 50% point of the trailing edge.

Rise time (t_r) Elapsed time from the 10% point to the 90% point of the rising edge of a pulse.

Fall time (t_f) Elapsed time from the 90% point to the 10% point of the falling edge of a pulse.

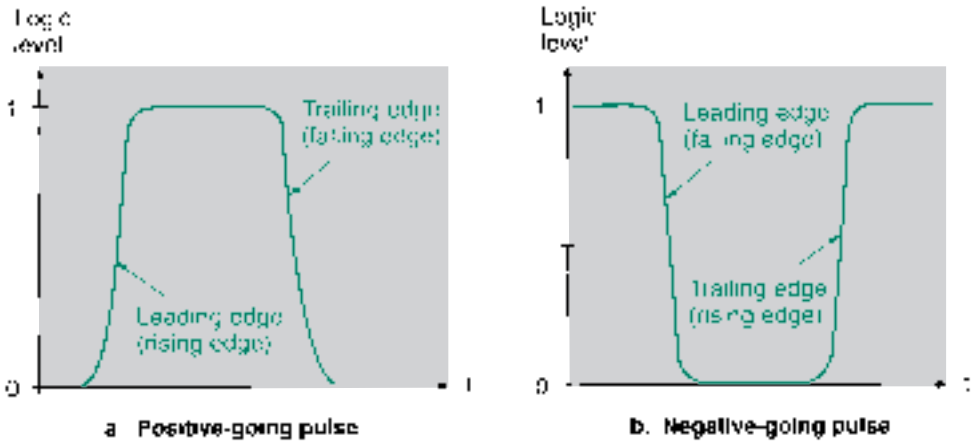
Figure 1.8 shows the forms of both an ideal and a nonideal **pulse.** The **rising and falling edges** of an ideal pulse are vertical. That is, the transitions between logic HIGH and LOW levels are instantaneous. There is no such thing as an ideal pulse in a real digital circuit. Circuit capacitance and other factors make the pulse more like the nonideal pulse in Figure 1.8b.

Pulses can be either positive-going or negative-going, as shown in Figure 1.9. In a positive-going pulse, the measured logic level is normally LOW, goes HIGH for the duration



a. Ideal pulse (instantaneous transitions) **b. Nonideal pulse**

FIGURE 1.8
Ideal and Nonideal Pulses



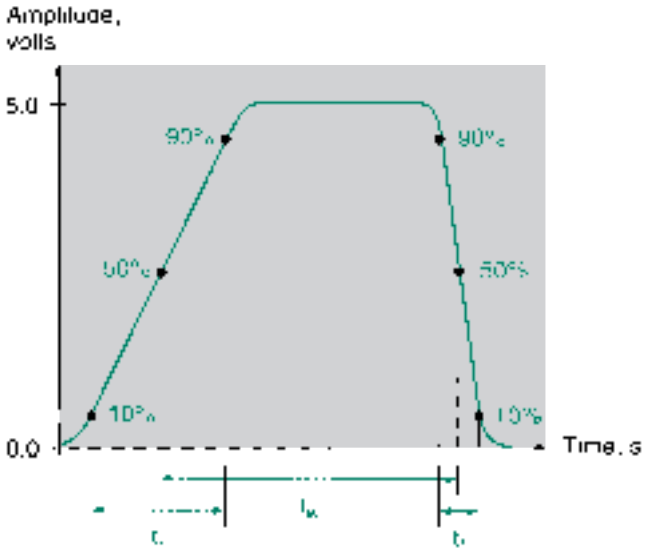
a Positive-going pulse **b. Negative-going pulse**

FIGURE 1.9
Pulse Edges

of the pulse, and returns to the LOW state. A negative-going pulse acts in the opposite direction.

Nonideal pulses are measured in terms of several timing parameters. Figure 1.10 shows the 10%, 50%, and 90% points on the rising and falling edges of a nonideal pulse. (100% is the maximum **amplitude** of the pulse.)

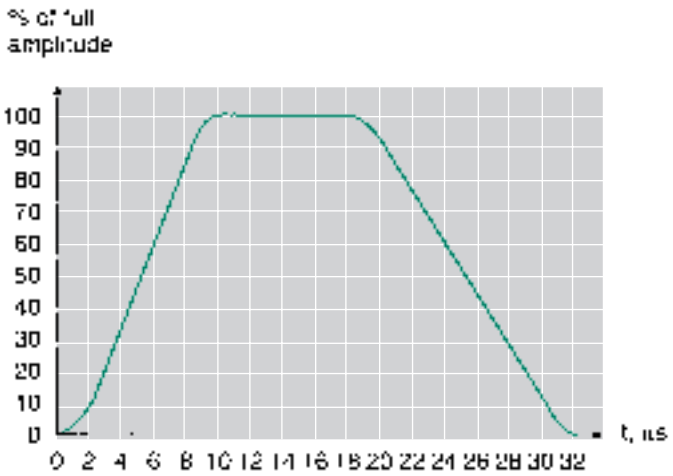
FIGURE 1.10
Pulse Width, Rise Time, Fall Time



The 50% points are used to measure **pulse width** because the edges of the pulse are not vertical. Without an agreed reference point, the pulse width is indeterminate. The 10% and 90% points are used as references for the **rise and fall times,** since the edges of a nonideal pulse are nonlinear. Most of the nonlinearity is below the 10% or above the 90% point.

EXAMPLE 1.16 Calculate the pulse width, rise time, and fall time of the pulse shown in Figure 1.11.

FIGURE 1.11
Example 1.16: Pulse



SOLUTION From the graph in Figure 1.11, read the times corresponding to the 10%, 50%, and 90% values of the pulse on both the **leading and trailing edges.**

Leading edge:	10%:	2 µs	*Trailing edge:*	90%:	20 µs
	50%:	5 µs		50%:	25 µs
	90%:	8 µs		10%:	30 µs