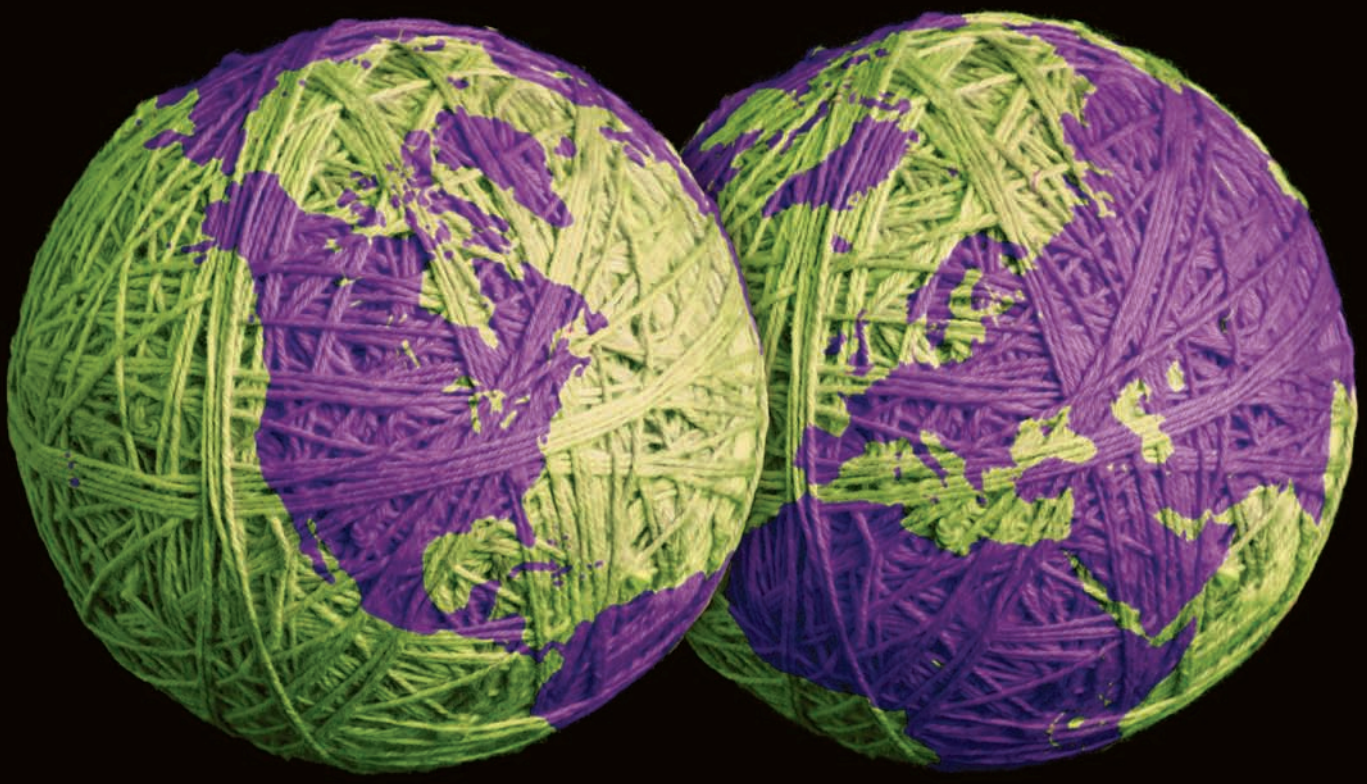
Pearson New International Edition

Digital Electronics
A Practical Approach with VHDL

William Kleitz

Ninth Edition



ALWAYS LEARNING

PEARSON

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England and Associated Companies throughout the world

Visit us on the World Wide Web at: www.pearsoned.co.uk

ISBN 10: 1-292-02561-1
ISBN 13: 978-1-292-02561-2

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Printed in the United States of America

2 Two's-Complement Representation

The most widely used method of representing binary numbers and performing arithmetic in computer systems is by using the **two's-complement method**. With this method, both positive and negative numbers can be represented using the same format, and binary subtraction is greatly simplified.

All along we have seen representing binary numbers in groups of eight for a reason. Most computer systems are based on 8- or 16-bit numbers. In an 8-bit system, the total number of different combinations of bits is 256 (2^8); in a 16-bit system, the number is 65,536 (2^{16}).

To be able to represent both positive *and* negative numbers, the two's-complement format uses the most significant bit (MSB) of the 8- or 16-bit number to signify whether the number is positive or negative. The MSB is therefore called the **sign bit** and is defined as 0 for positive numbers and 1 for negative numbers. *Signed two's-complement* numbers are shown in Figure 4.

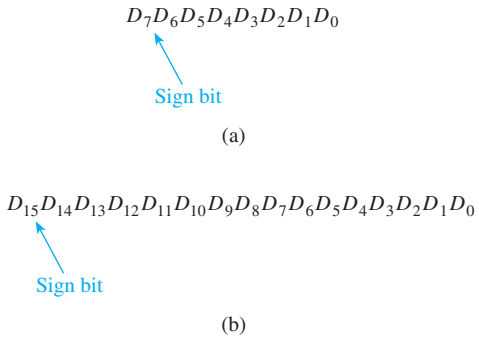


Figure 4 Two's-complement numbers: (a) 8-bit number; (b) 16-bit number.

The *range of positive numbers* in an 8-bit system is 0000 0000 to 0111 1111 (0 to 127). The *range of negative numbers* is 1111 1111 to 1000 0000 (−1 to −128). In general, the maximum positive number is equal to $2^{N-1} - 1$, and the maximum negative number is $-(2^{N-1})$, where N is the number of bits in the number, including the sign bit (e.g., for an 8-bit positive number, $2^{8-1} - 1 = 127$).

A table of two's-complement numbers can be developed by starting with some positive number and continuously subtracting 1. Table 3 shows the signed two's-complement numbers from +7 to −8.

Converting a decimal number to two's complement, and vice versa, is simple and can be done easily using logic gates, as we will see later in this chapter. For now, let's deal with 8-bit numbers; however, the procedure for 16-bit numbers is exactly the same.

Steps for Decimal-to-Two's-Complement Conversion

1. If the decimal number is positive, the two's-complement number is the true binary equivalent of the decimal number (e.g., +18 = 0001 0010).
2. If the decimal number is negative, the two's-complement number is found by
   - **(a)** Complementing each bit of the true binary equivalent of the decimal number (this is called the **one's complement**).
   - **(b)** Adding 1 to the one's-complement number to get the magnitude bits. (The sign bit will always end up being 1.)

Team Discussion

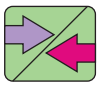
Try to represent the number 160_{10} in two's-complement for an 8-bit system. Why doesn't it work?

TABLE 3	Signed Two's-Complement Numbers +7 Through −8
Decimal	**Two's Complement**
+7	0000 0111
+6	0000 0110
+5	0000 0101
+4	0000 0100
+3	0000 0011
+2	0000 0010
+1	0000 0001
0	0000 0000
−1	1111 1111
−2	1111 1110
−3	1111 1101
−4	1111 1100
−5	1111 1011
−6	1111 1010
−7	1111 1001
−8	1111 1000

Steps for Two's-Complement-to-Decimal Conversion

1. If the two's-complement number is positive (sign bit = 0), do a regular binary-to-decimal conversion.
2. If the two's-complement number is negative (sign bit = 1), the decimal sign will be −, and the decimal number is found by
   - **(a)** Complementing the entire two's-complement number, bit by bit.
   - **(b)** Adding 1 to arrive at the true binary equivalent.
   - **(c)** Doing a regular binary-to-decimal conversion to get the decimal numeric value.

The following examples illustrate the conversion process.



Common Misconception

As soon as some students see the phrase "convert to two's complement," they go ahead with the procedure for negative numbers whether the original number is positive or negative.

EXAMPLE 5

Convert $+35_{10}$ to two's complement.

Solution:

True binary = 0010 0011
Two's complement = 0010 0011 ***Answer***

EXAMPLE 6

Convert -35_{10} to two's complement.

Solution:

True binary = 0010 0011
One's complement = 1101 1100
Add 1 = +1
Two's complement = 1101 1101 ***Answer***

EXAMPLE 7

Convert 1101 1101 two's complement back to decimal.

Solution: The sign bit is 1, so the decimal result will be negative.

Two's complement = 1101 1101
Complement = 0010 0010
Add 1 = +1
True binary = 0010 0011
Decimal complement = −35 ***Answer***

EXAMPLE 8

Convert -98_{10} to two's complement.

Solution:

True binary = 0110 0010
One's complement = 1001 1101
Add 1 = +1
Two's complement = 1001 1110 ***Answer***

EXAMPLE 9

Convert 1011 0010 two's complement to decimal.

Solution: The sign bit is 1, so the decimal result will be negative.

Two's complement = 1011 0010
Complement = 0100 1101
Add 1 = +1
True binary = 0100 1110
Decimal complement = −78 ***Answer***

Review Questions

4. Which bit in an 8-bit two's-complement number is used as the sign bit?

5. Are the following two's-complement numbers positive or negative?

(a) 1010 0011

(b) 0010 1101

(c) 1000 0000

3 Two's-Complement Arithmetic

All four of the basic arithmetic functions involving positive *and* negative numbers can be dealt with very simply using two's-complement arithmetic. Subtraction is done by

adding the two two's-complement numbers. Thus, the same digital circuitry can be used for additions *and* subtractions, and there is no need always to subtract the smaller number from the larger number. We must be careful, however, not to exceed the *maximum range* of the two's-complement number: +127 to −128 for 8-bit systems, and +32,767 to −32,768 for 16-bit systems ($+2^{N-1} - 1$ to -2^{N-1}).

When *adding* numbers in the two's-complement form, simply perform a regular binary addition to get the result. When *subtracting* numbers in the two's-complement form, convert the number being subtracted to a *negative* two's-complement number and perform a regular binary addition [e.g., 5 − 3 = 5 + (−3)]. The result will be a two's-complement number, and if the result is negative, the sign bit will be 1.

Work through the following examples to familiarize yourself with the addition and subtraction procedure.

EXAMPLE 10

Add 19 + 27 using 8-bit two's-complement arithmetic.

Solution:

$$\begin{aligned} 19 &= 0001\ 0011 \\ 27 &= \underline{0001\ 1011} \\ \text{Sum} &= 0010\ 1110 = 46_{10} \end{aligned}$$

EXAMPLE 11

Perform the following subtractions using 8-bit two's-complement arithmetic.

(a) 18 − 7;

(b) 21 − 13;

(c) 118 − 54;

(d) 59 − 96.

Solution:

(a) 18 − 7 is the same as 18 + (−7), so just add 18 to negative 7.

$$\begin{aligned} +18 &= 0001\ 0010 \\ -7 &= \underline{1111\ 1001} \\ \text{Sum} &= 0000\ 1011 = 11_{10} \end{aligned}$$

Note: The carry-out of the MSB is ignored. (It will always occur for positive sums.) The 8-bit answer is 0000 1011.

(b)
$$\begin{aligned} +21 &= 0001\ 0101 \\ -13 &= \underline{1111\ 0011} \\ \text{Sum} &= 0000\ 1000 = 8_{10} \end{aligned}$$

(c)
$$\begin{aligned} +118 &= 0111\ 0110 \\ -54 &= \underline{1100\ 1010} \\ \text{Sum} &= 0100\ 0000 = 64_{10} \end{aligned}$$

(d)
$$\begin{aligned} +59 &= 0011\ 1011 \\ -96 &= \underline{1010\ 0000} \\ \text{Sum} &= 1101\ 1011 = -37_{10} \end{aligned}$$

Review Questions

6. Which of the following decimal numbers cannot be converted to 8-bit two's-complement notation?

(a) 89

(b) 135

(c) −107

(d) −144

7. The procedure for subtracting numbers in two's-complement notation is exactly the same as for adding numbers. True or false?

8. When subtracting a smaller number from a larger number in two's complement, there will always be a carry-out of the MSB, which will be ignored. True or false?

4 Hexadecimal Arithmetic*

Hexadecimal representation is a method of representing groups of 4 bits as a single digit. Hexadecimal notation has been widely adopted by manufacturers of computers and microprocessors because it simplifies the documentation and use of their equipment. Eight- and 16-bit computer system data, program instructions, and addresses use hexadecimal to make them easier to interpret and work with than their binary equivalents.

Hexadecimal Addition

Remember, hexadecimal is a base 16 numbering system, meaning that it has 16 different digits (as shown in Table 4). Adding 3 + 6 in hex equals 9, and 5 + 7 equals C. But, adding 9 + 8 in hex equals a sum greater than F, which will create a carry. The sum of 9 + 8 is 17_{10}, which is 1 larger than 16, making the answer 11_{16}.

TABLE 4 Hexadecimal Digits with Their Equivalent Binary and Decimal Values

Hexadecimal	Binary	Decimal
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
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

*Most scientific calculators perform number base conversions and arithmetic. This allows you to enter binary, octal, decimal, or hexadecimal numbers and perform any of the arithmetic operations. In this chapter we will learn the step-by-step procedures for performing these operations by hand, but as the numbers get more complex it is best to use your calculator for these functions.