

Adders

OUTLINE

- 5.1 Half Adder
- 5.2 Full Adder
- 5.3 Binary 1's Complement Subtraction
- 5.4 1's Complement Adder/Subtractor Circuit
- 5.5 Binary 2's Complement Subtraction
- 5.6 2's Complement Adder/Subtractor Circuit
- 5.7 Signed 2's Complement Numbers
- 5.8 Binary-Coded-Decimal Addition
- 5.9 Binary-Coded-Decimal Adder Circuit
- 5.10 Arithmetic Logic Unit (ALU)
- 5.11 Programming a GAL
- 5.12 Troubleshooting Adder Circuits

Digital Application Floating-Point Unit (FPU)

LAB 5A Adders

LAB 5B Adder Circuits

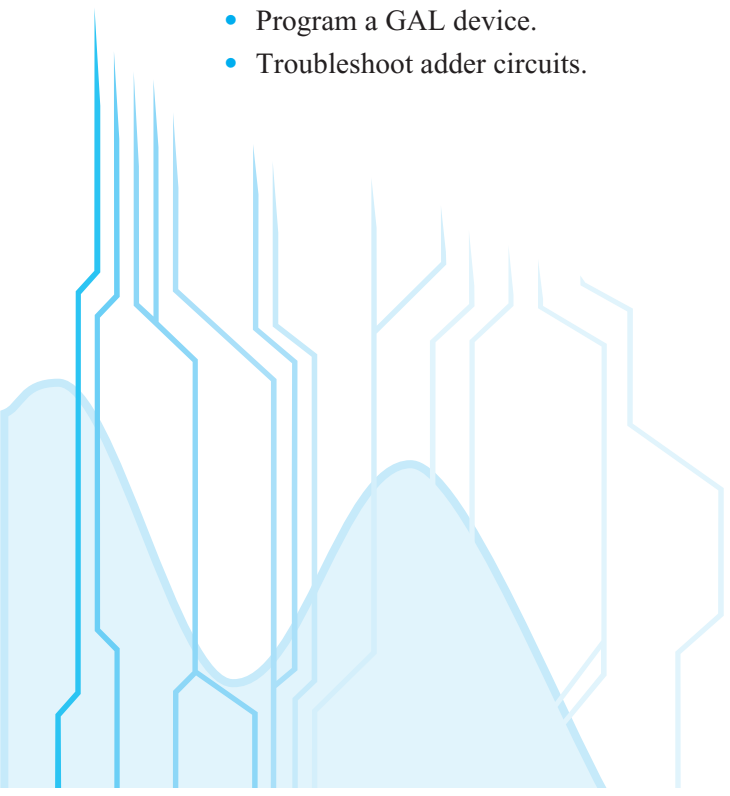
KEY TERMS

arithmetic logic unit (ALU)	look ahead carry
binary-coded decimal (BCD)	1's complement
end-around carry (EAC)	overflow
fast carry	signed 2's complement
full adder	2's complement



OBJECTIVES

After completing this chapter, you should be able to:

- Define half adder and draw the block diagram and truth table.
 - Develop the logic circuitry and construct a half adder.
 - Define full adder and draw the block diagram and truth table.
 - Develop the logic circuitry and construct a full adder.
 - Subtract binary numbers using 1's complement method.
 - Design the circuitry required to use a full adder as a 1's complement adder/subtractor.
 - Subtract binary numbers using 2's complement method.
 - Design the circuitry required to use a full adder as a 2's complement adder/subtractor.
 - Convert from decimal to signed 2's complement and signed 2's complement to decimal.
 - Add and subtract signed 2's complement numbers.
 - Convert from decimal to BCD and BCD to decimal.
 - Add BCD numbers.
 - Design the circuitry required to use a full adder as a BCD adder.
 - Combine numbers using the logic and arithmetic functions of an arithmetic logic unit.
 - Program a GAL device.
 - Troubleshoot adder circuits.
- 



5.1 HALF ADDER

A **half adder** is a circuit that has two inputs, A and B , and two outputs, sum and carry. It adds A and B according to the rules for binary addition and outputs the sum and carry. The block diagram and truth table for a half adder are shown in Figure 5-1. The truth table follows the rules for binary addition. The last line shows 1 plus 1 is 10, as it must be.

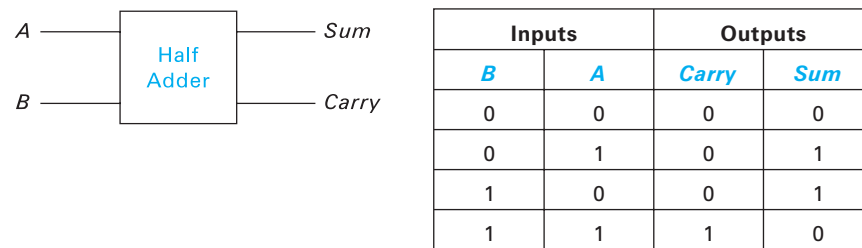


FIGURE 5-1 Half adder

The sum output has a truth table identical to the exclusive-OR, and the carry output has a truth table identical to an AND gate. One way to construct a half adder is shown in Figure 5-2. Another way to construct a half adder is shown in Figure 5-3. In Figure 5-3, the exclusive-OR is constructed from one AND gate and two NOR gates as discussed in Chapter 4. A is ANDed with B as part of the exclusive-OR and can be used as the carry signal.

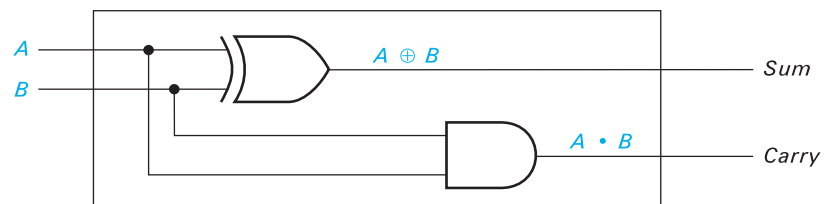


FIGURE 5-2 Logic diagram of a half adder

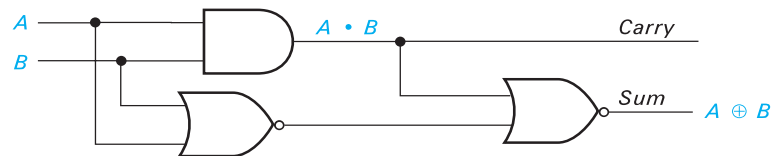


FIGURE 5-3 Using an AND gate and two NOR gates to construct a half adder

EXAMPLE 5-1

Add $A = 1, B = 1$.

Solution See Figure 5-4.

1 plus 1 has a sum of 0 and a carry of 1.

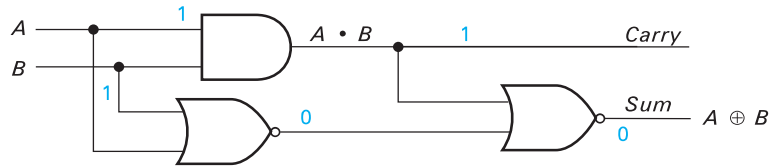


FIGURE 5-4



5.2 FULL ADDER

Whereas the half adder added two inputs, A and B , the **full adder** adds three inputs together, A , B , and a carry from a previous addition, and outputs a sum and carry. The truth table follows the rules for binary addition. The block diagram and truth table for a full adder are shown in Figure 5-5.

The sum is 1 each time the total number of 1s on inputs A , B , and carry-in is odd. This is analogous to an even-parity generator as shown in Figure 5-6. The output is 1 for an odd number of 1s in.

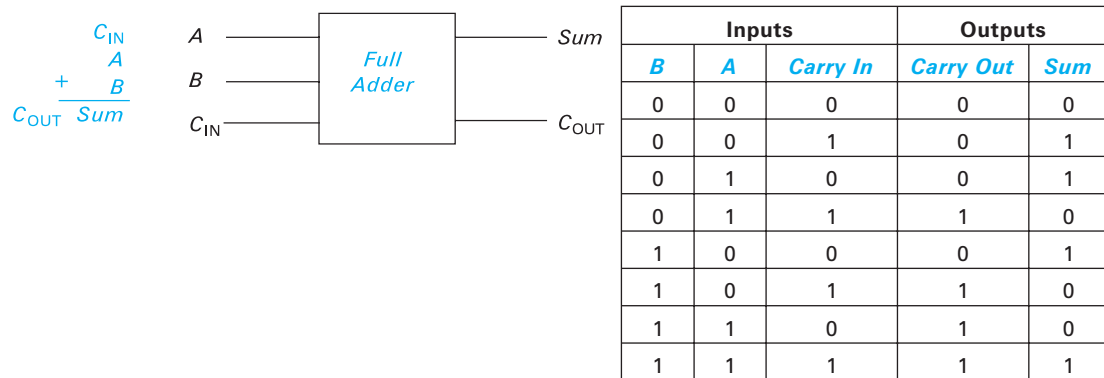


FIGURE 5-5 Full adder

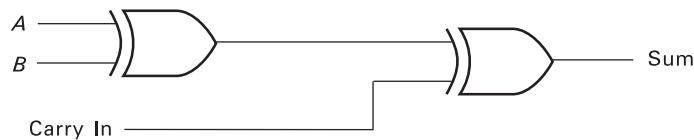


FIGURE 5-6 Full adder—sum

EXAMPLE 5-2

Add $A = 1, B = 1, \text{Carry-in} = 1$.

Solution See Figure 5-7.

1 plus 1 plus 1 has a sum of 1.

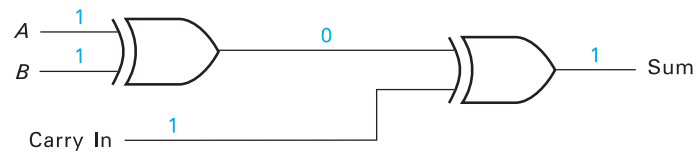


FIGURE 5-7

Each of the exclusive-OR gates in Figure 5-6 can be replaced with two NOR gates and an AND gate, Figure 5-8.

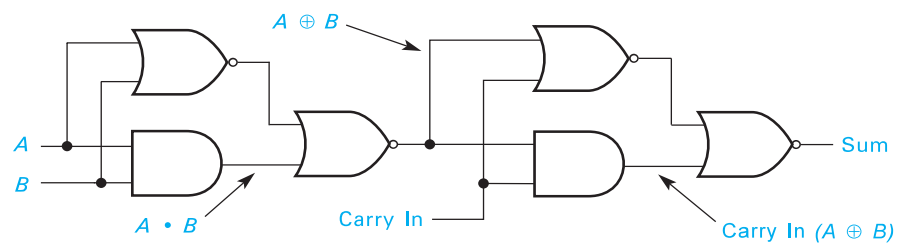


FIGURE 5-8 A second construction for a full adder—sum

EXAMPLE 5-3

Add $A = 1, B = 0, \text{Carry-in} = 1$.

Solution See Figure 5-9.

1 plus 0 plus 1 has a sum of 0.

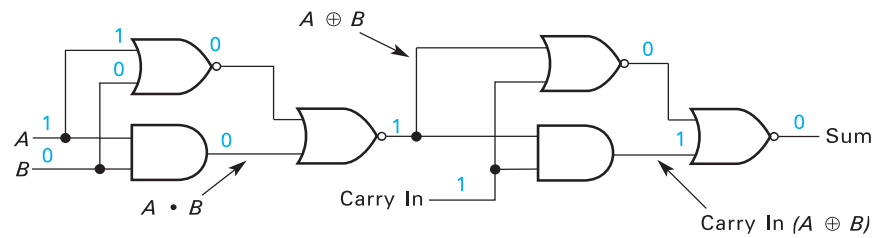


FIGURE 5-9

The carry output is 1 for the conditions on lines 4, 6, 7, and 8 of the truth table in Figure 5-5.

$$\begin{aligned}
 C_{OUT} &= \bar{B}AC_{IN} + B\bar{A}C_{IN} + BA\bar{C}_{IN} + BAC_{IN} \\
 &= C_{IN}(\bar{B}A + B\bar{A}) + BA(\bar{C}_{IN} + C_{IN}) \\
 &= C_{IN}(\bar{B}A + B\bar{A}) + BA \\
 &= C_{IN}(B \oplus A) + BA
 \end{aligned}$$

In Figure 5-8, A has already been exclusive-ORed with B , and the result has been ANDed with C_{IN} . Also, A has been ANDed with B . A two-input OR gate will combine these two signals to produce the carry-out signal as shown in Figure 5-10. The full adder in Figure 5-10 is constructed from two half adders and an OR gate. Each half adder is enclosed in broken lines.

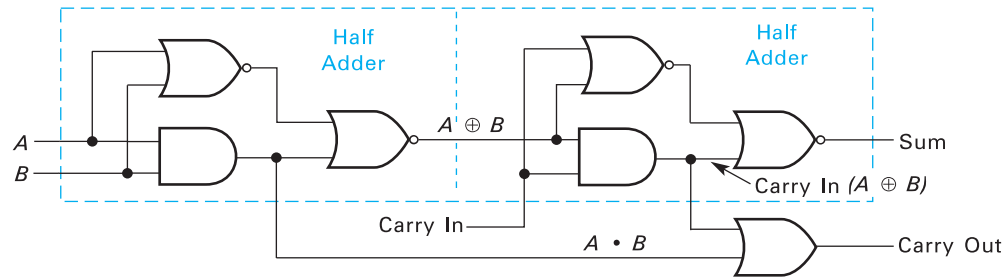


FIGURE 5-10 Full adder—sum and carry

EXAMPLE 5-4

Add $A = 1$, $B = 1$, Carry-in = 0.

Solution See Figure 5-11.

Sum = 0; Carry = 1

1 plus 1 plus 0 = 10

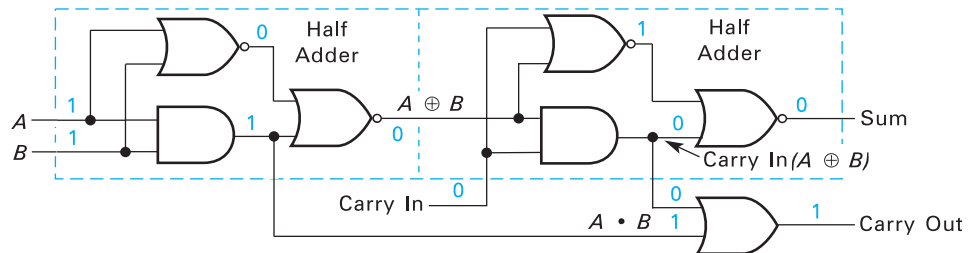


FIGURE 5-11

Table 5-1 lists some of the available 4-bit full adder ICs. The internal circuitry contains enough components for the IC to be classified as medium-scale integration.

The 7483 was used in Lab 1A to add two 4-bit numbers, A and B , and a carry-in, called C_0 . The outputs are a 4-bit sum and carry-out, C_4 , as shown in Figure 5-12. Carries C_1 , C_2 , and C_3 are handled internally and do not appear on the pins of the IC.

TABLE 5-1 Medium Scale Integration Adder Circuits

Device No.	Family	Description
7483	TTL	4-bit binary adder with fast carry
74C83	CMOS	4-bit binary adder with fast carry
4008	CMOS	4-bit full adder with fast carry

$$\begin{array}{r}
 C_3 \ C_2 \ C_1 \ C_0 \\
 A_4 \ A_3 \ A_2 \ A_1 \\
 + \ B_4 \ B_3 \ B_2 \ B_1 \\
 \hline
 C_4 \ \Sigma_4 \ \Sigma_3 \ \Sigma_2 \ \Sigma_1
 \end{array}$$

FIGURE 5-12 4-bit full adder inputs and outputs

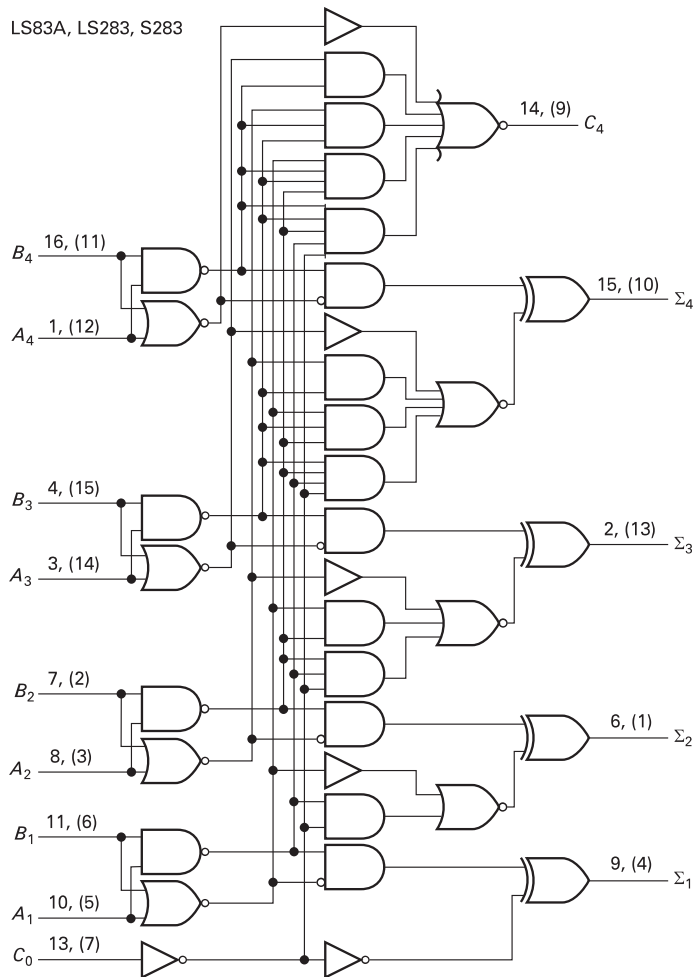


FIGURE 5-13 7483 logic diagram

When working the addition problem longhand, C_4 is not determined until each of the columns has been added. The carry has to “ripple-through” four stages of addition. The logic diagram in Figure 5-13 shows how a 7483 produces C_4 from the inputs without waiting for the “ripple-effect” to take place. This results in a **fast carry** or **look ahead carry**. The result is a faster operation; in fact, C_4 appears before the Σ outputs are established.

One of the gates in the logic diagram shown in Figure 5-13 is not a basic gate, but is a combination of basic gates. The Boolean expression for the gate, its equivalent logic diagram, and its truth table are shown in Figures 5-14A and 5-14B. When A is 0 AND when B is 1 the output is 1.

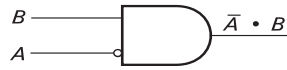


FIGURE 5-14A

B	A	Y
0	0	0
0	1	0
1	0	1
1	1	0

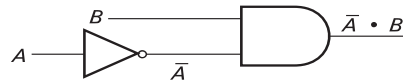


FIGURE 5-14B

EXAMPLE 5-5

For the gate shown in Figure 5-15A, write the Boolean expression, the equivalent logic diagram, and its truth table.

Solution When A is 1 AND when B is 0 the output is 0. See Figure 5-15B.

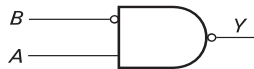


FIGURE 5-15A

B	A	Y
0	0	1
0	1	0
1	0	1
1	1	1

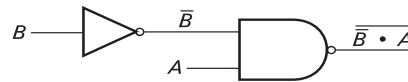


FIGURE 5-15B

EXAMPLE 5-6

Add these numbers using a 7483. Follow the logic levels through the logic diagram.

$A = 1001$, $B = 1010$, and $C_0 = 1$

Solution See Figure 5-16.

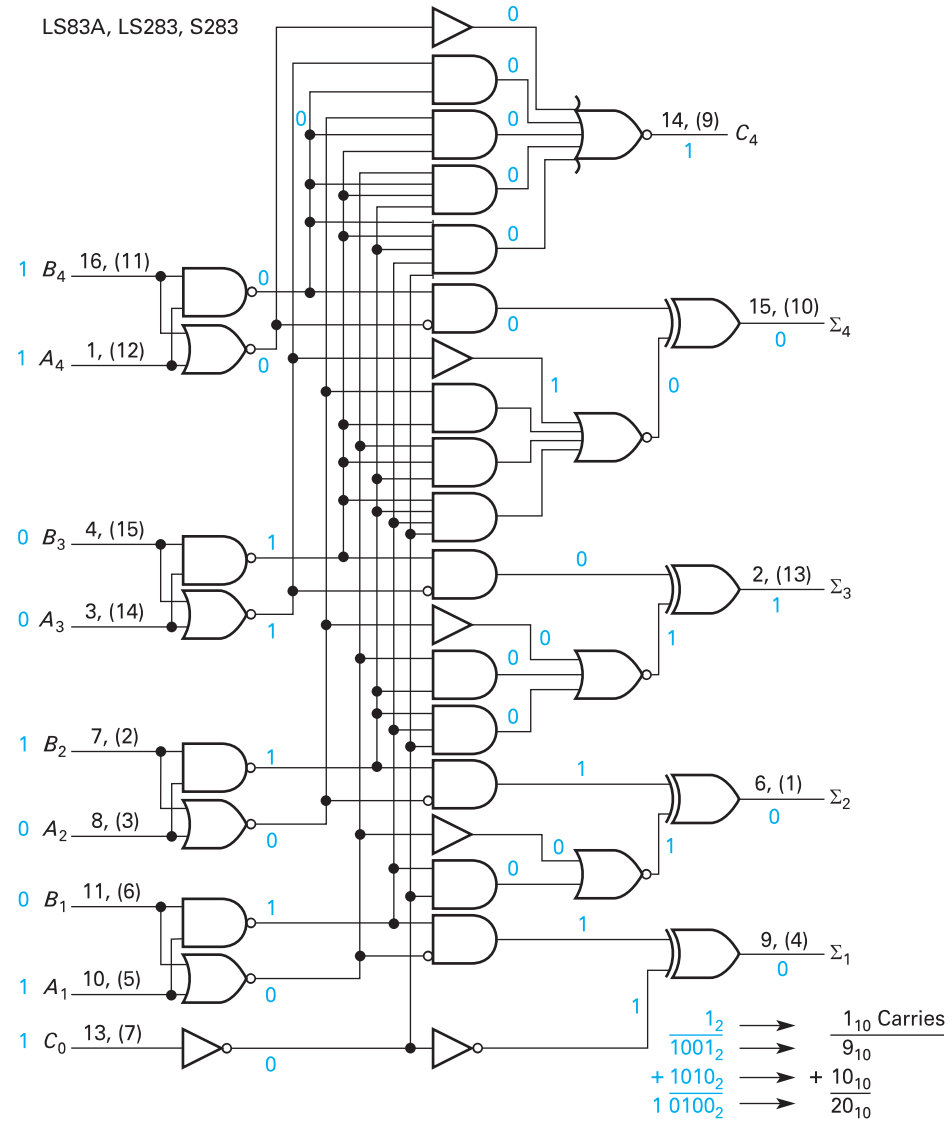
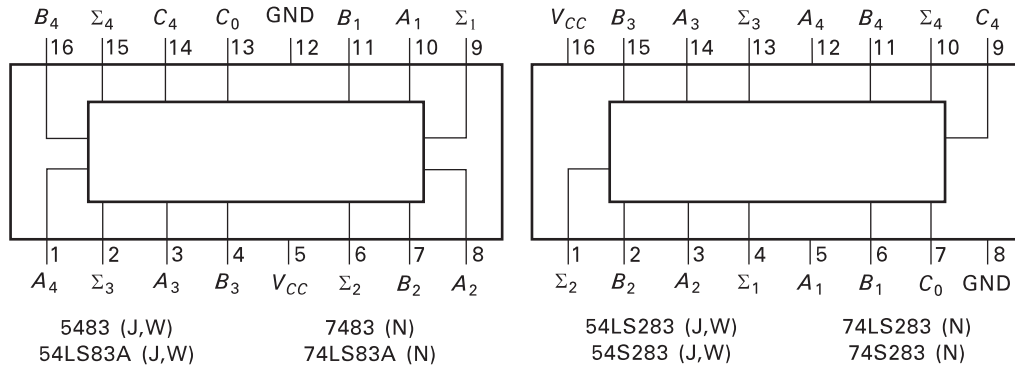


FIGURE 5-16

The truth table and connection diagram (pinout) for the 7483 and 74S283 are shown in Figure 5-17. Note that the 7483 is not “corner powered.” V_{CC} is pin 5 and ground is on pin 12.

Writing a truth table for nine inputs creates a table of 512 lines (2^9). The truth table shown has been reduced to 16 lines. The note below the truth table explains that the

table is used in two steps. $A_1, B_1, A_2, B_2,$ and C_0 determine the outputs $\Sigma_1, \Sigma_2,$ and C_2 that are internal to the IC. C_2 is then used with $A_3, B_3, A_4,$ and B_4 to determine $\Sigma_3, \Sigma_4,$ and C_4 .



Input				Output					
				When $C_0 = L$			When $C_0 = H$		
A_1	B_1	A_2	B_2	Σ_1	Σ_2	C_2	Σ_1	Σ_2	C_2
A_3	B_3	A_4	B_4	Σ_3	Σ_4	C_4	Σ_3	Σ_4	C_4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	L	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

H = HIGH Level, L = LOW Level

Note: Input conditions at A_1, B_1, A_2, B_2 and C_0 are used to determine outputs Σ_1 and Σ_2 and the value of the internal carry C_2 . The values at $C_2, A_3, B_3, A_4,$ and B_4 are then used to determine outputs $\Sigma_3, \Sigma_4,$ and C_4 .

FIGURE 5-17 Truth table and connection diagrams

EXAMPLE 5-7

Use the 7483 to add 0110 to 1101 with $C_0 = 1$.

A_4	A_3	A_2	A_1
0	1	1	0

B_4	B_3	B_2	B_1
1	1	0	1

C_0
1

Solution

Step 1.

A_1	B_1	A_2	B_2
-------	-------	-------	-------

 =

L	H	H	L
---	---	---	---

 (line 7)

with $C_0 = H, \Sigma_1 = L, \Sigma_2 = L, C_2 = H$

Step 2.

A_3	B_3	A_4	B_4
-------	-------	-------	-------

 =

H	H	L	H
---	---	---	---

 (line 12)

with $C_2 = H, \Sigma_3 = H, \Sigma_4 = L, C_4 = H$

Σ_4	Σ_3	Σ_2	Σ_1
------------	------------	------------	------------

 =

L	H	L	L
---	---	---	---

 = 0100

with $C_4 = H = 1$.

$0110 + 1101 + 1 = 10100$

$6 + 13 + 1 = 20$

The IEC logic symbol for a 74LS283 is shown in Figure 5-18. A capital sigma, Σ , is used to denote addition. This symbol uses P_3, P_2, P_1, P_0 and Q_3, Q_2, Q_1, Q_0 to represent the two 4-bit numbers to be added and $\Sigma_3, \Sigma_2, \Sigma_1, \Sigma_0$ to represent the result. Note that C_1 is used for carry-in and C_0 for carry-out.

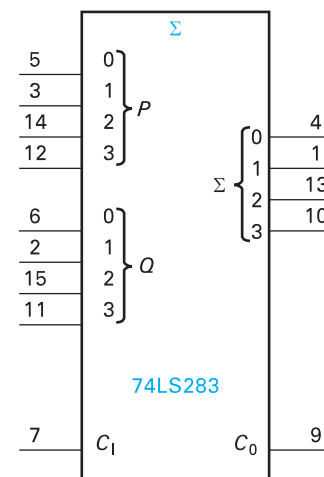


FIGURE 5-18 IEC logic symbol 74LS283


SELF-CHECK 1

1. How does a half adder differ from a full adder?
2. Draw the block diagram and truth table for a half adder.
3. Draw the block diagram and truth table for a full adder.
4. Add these numbers using a 7483. Follow the logic levels through Figure 5-13.

a. 1 ($C_0 = 1$) 1001 + 0110 ———	b. 0 ($C_0 = 0$) 0111 + 0110 ———
---	---

Although circuits that subtract numbers can be designed and constructed, computers use a complement method of subtraction that converts the problem into addition. Then, adder circuits can be used to work subtraction problems. We will study two complement systems, 1's complement and 2's complement. Each system uses the concept of **overflow**. Overflow occurs when the sum of the most significant (left-most) column yields a carry. For example:

$\begin{array}{r} 872 \\ +345 \\ \hline 1\ 217 \end{array}$	$\begin{array}{r} 7326 \\ +0074 \\ \hline 0\ 7400 \end{array}$
Overflow $\xrightarrow{\quad}$ \uparrow	No Overflow $\xrightarrow{\quad}$ \uparrow

In the second example, leading zeros are added to block in the two numbers.



5.3 BINARY 1'S COMPLEMENT SUBTRACTION

To take the **1's complement** of a binary number, simply change each bit. The 1's complement of 1 is 0 and vice versa. The 1's complement of 1001010 is 0110101. To subtract using 1's complement:

1. Take the 1's complement of the subtrahend (bottom number).
2. Add the 1's complement to the minuend (top number).
3. Overflow indicates that the answer is positive. Add the overflow to the least significant bit. This operation is called **end-around carry (EAC)**.
4. If there is no overflow then the answer is negative. Take the 1's complement of the original sum to obtain the true magnitude of the answer.

EXAMPLE 5-8

Subtract. $11001_2 - 10001_2$

Solution

$$\begin{array}{r}
 11001 \longrightarrow 11001 \\
 -10001 \quad \quad \quad +01110 \\
 \hline
 1\ 00111 \longrightarrow 00111 \\
 +\quad 1 \longleftarrow \text{EAC} \\
 \hline
 1000
 \end{array}$$

Overflow \longleftarrow

The answer is +1000.

Check.

$$25_{10} - 17_{10} = 8_{10}$$

EXAMPLE 5-9

Subtract. $1011_2 - 101_2$

Solution

$$\begin{array}{r}
 1011 \longrightarrow 1011 \\
 -\ 101 \quad \quad \quad +1010 \\
 \hline
 1\ 0101 \longrightarrow 0101 \\
 +\quad 1 \longleftarrow \text{EAC} \\
 \hline
 0110
 \end{array}$$

Overflow \longleftarrow

Note that the leading 0 becomes a 1. The answer is +110.

Check.

$$11_{10} - 5_{10} = 6_{10}$$

The same process is used when the subtrahend is larger than the minuend.

EXAMPLE 5-10

Subtract. $101_2 - 11000_2$

Solution

$$\begin{array}{r}
 101 \longrightarrow 101 \\
 -11000 \quad \quad \quad +00111 \\
 \hline
 01100
 \end{array}$$

No Overflow \longleftarrow

The answer is negative. The true magnitude is the 1's complement of 01100 or 10011. The answer is -10011.

Check.

$$5_{10} - 24_{10} = -19_{10}$$

EXAMPLE 5-11Subtract. $10000_2 - 11101_2$ **Solution**

$$\begin{array}{r}
 10000 \\
 -11101 \\
 \hline
 \end{array}
 \longrightarrow
 \begin{array}{r}
 10000 \\
 +00010 \\
 \hline
 10010
 \end{array}$$

No Overflow

The answer is negative. The true magnitude is the 1's complement of 10010 or 01101. The answer is -01101 .

Check.

$$16_{10} - 29_{10} = -13_{10}$$



5.4 1'S COMPLEMENT ADDER/SUBTRACTOR CIRCUIT

Design a circuit that will use a 7483 to add the 4-bit number B_4, B_3, B_2, B_1 to the 4-bit number A_4, A_3, A_2, A_1 or subtract B_4, B_3, B_2, B_1 from A_4, A_3, A_2, A_1 . Use the 1's complement method for subtraction.

To use a 7483 4-bit full adder as a 1's complement adder/subtractor, the following details must be considered.

1. Refer to Figure 5-19. Leave the number B_4, B_3, B_2, B_1 unaltered for an addition problem, but take the 1's complement of the subtrahend for a subtraction problem. An exclusive-OR inverts data (1's complement) when the control input is HIGH. Exclusive-OR gates will be used to invert B_4, B_3, B_2, B_1 for subtraction. A control signal is needed that will be 1 for subtraction and 0 for addition. A_4, A_3, A_2, A_1 will be fed directly into the 7483.
2. Refer to Figure 5-20. If the problem is subtraction and if there is overflow ($C_4 = 1$), perform an EAC (end-around carry). To detect when subtraction and overflow occur, AND the control line with C_4 . The output of the AND gate number 1 is 1 when an EAC results. But in this case, the output of AND gate number 1 can be fed directly into C_0 .
3. Refer to Figures 5-21 and 5-22. If the problem is subtraction and if there is no overflow ($C_4 = 0$), indicate the answer is negative and take the 1's complement of the result to obtain the true magnitude of the answer. If C_4 is inverted,

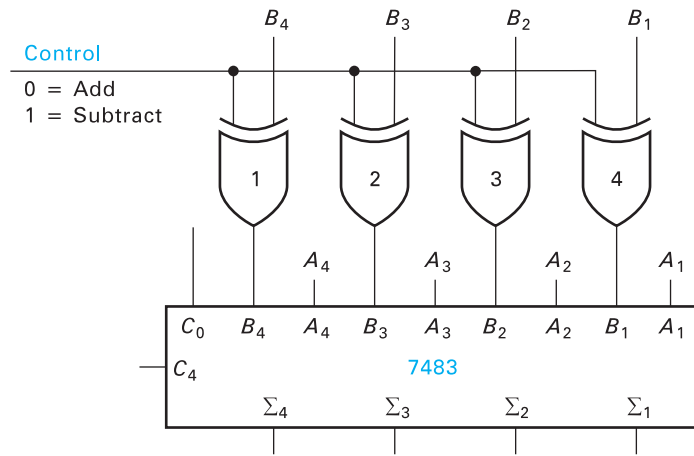


FIGURE 5-19

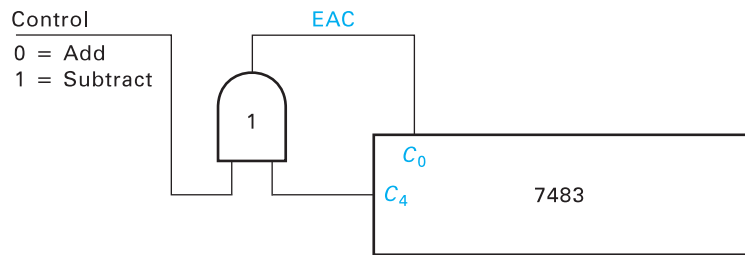


FIGURE 5-20

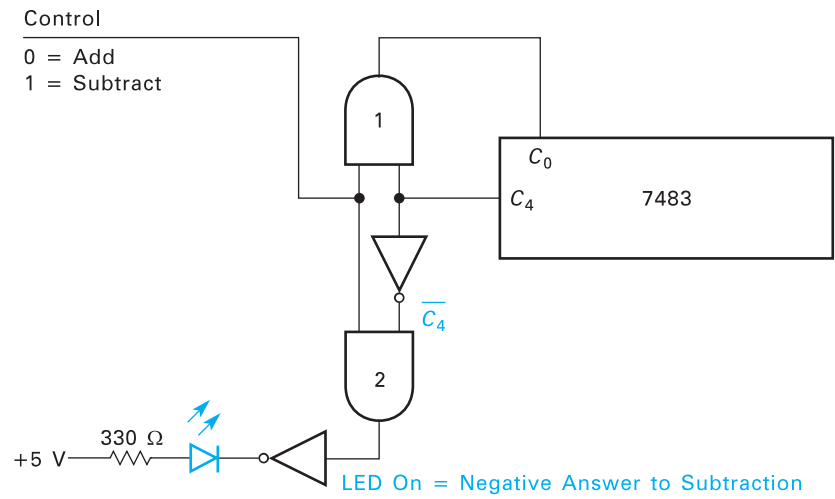


FIGURE 5-21

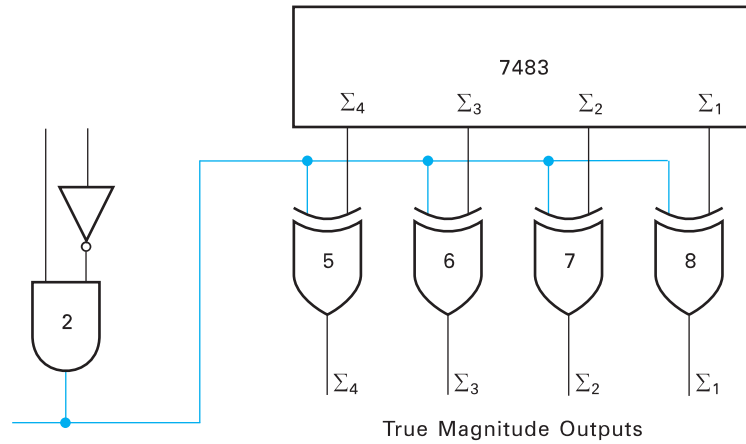


FIGURE 5-22

then an AND gate can be used to detect when a subtraction process is to be performed and when C_4 is 0. The control input and C_4 are fed into AND gate number 2. A HIGH output indicates a subtraction problem is being performed, and the answer is negative. This signal could be used to light an LED to indicate the answer is negative. The LED requires approximately 12 mA to burn brightly. As will be seen in Chapter 6, TTL can handle more current in the 0 mode than in the 1 mode. This signal will be inverted to drive the LED in the active LOW mode. A red LED drops about 1.6 V or 1.7 V when lit (LED voltage drops vary greatly with different colors). This leaves $5\text{ V} - 1.7\text{ V}$ or 3.3 V to be dropped across the resistor. Ohm's Law dictates the resistor should be about

$$\frac{3.3\text{ V}}{12 \times 10^{-3}\text{ A}} = 275\ \Omega$$

A 330- Ω resistor, the nearest standard size, will be used to limit the current through the LED. The 7404 inverter can handle 16 mA in the zero mode which is more than enough to light the LED as designed. As shown in Figure 5-22, the output of AND gate number 2 can be used to control four exclusive-OR gates to invert (take the 1's complement) when the answer to a subtraction problem is negative. Refer to Figure 5-22. The full schematic is drawn in Figure 5-23.

EXAMPLE 5-12

Add 1011 plus 0010.

Solution See Figure 5-24.

EXAMPLE 5-13

Subtract 0110 from 1001.

Solution See Figure 5-25.

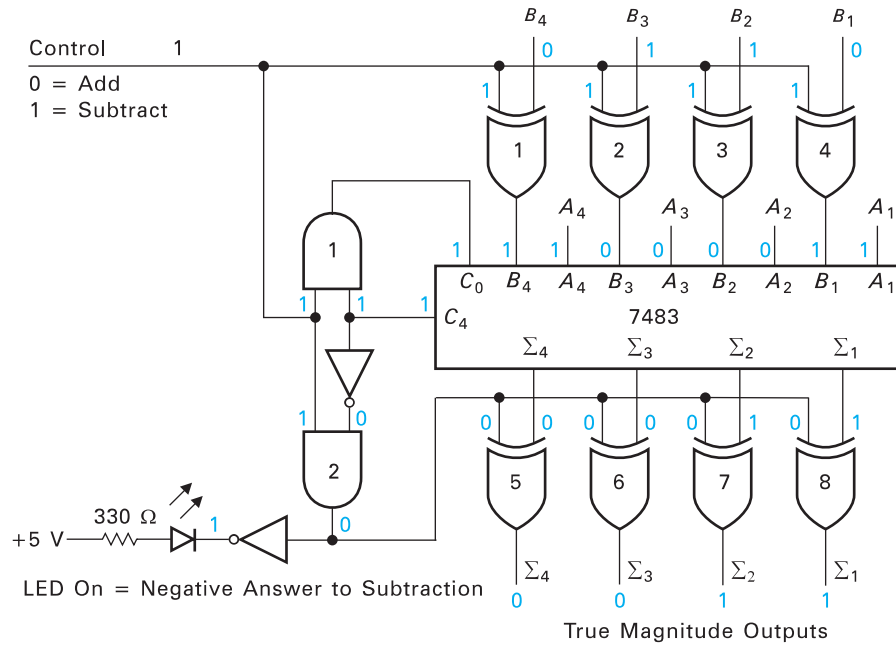


FIGURE 5-25

EXAMPLE 5-14

Subtract 1010 from 0011.

Solution See Figure 5-26 (page 223).

SELF-CHECK 2

1. Subtract using the 1's complement method. Follow the logic levels through Figure 5-23.

a. $\begin{array}{r} 0101 \\ -1010 \end{array}$	b. $\begin{array}{r} 11 \\ -1000 \end{array}$
---	---
2. Subtract using the 1's complement method. Follow the logic levels through Figure 5-23.

a. $\begin{array}{r} 1101 \\ -0110 \end{array}$	b. $\begin{array}{r} 1100 \\ -111 \end{array}$
---	--

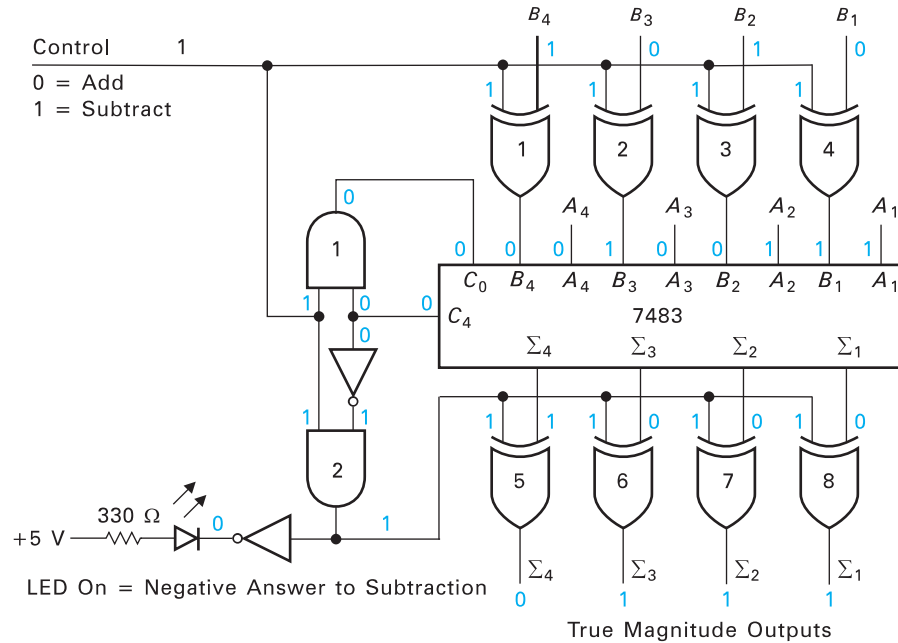


FIGURE 5-26



5.5 BINARY 2'S COMPLEMENT SUBTRACTION

To form the **2's complement** of a number, first take the 1's complement and then add 1. The 2's complement of 10110 is $01001 + 1 = 01010$. A shorter method is to start at the least significant bit and, moving to the left, leave each bit the same until the first 1 is passed. Then change each bit thereafter.

EXAMPLE 5-15

Find the 2's complement of 101101000.

Solution Change each bit to the left of the first 1.

$$\begin{array}{r} \text{Number} \quad \quad = 101101000 \\ \text{2's complement} = 010011000 \end{array}$$

EXAMPLE 5-16

Find the 2's complement of 1011011.

Solution

Method 1.

$$\begin{array}{r} \text{1's complement} \quad 0100100 \\ \text{Add 1} \quad \quad \quad \quad + \quad \quad \quad 1 \\ \hline \text{2's complement} \quad 0100101 \end{array}$$

Method 2.
Change each bit to the left of the first 1.

1011011
0100101

EXAMPLE 5-17

Find the 2's complement of 101000000.

Solution

Method 1.

1's complement	010111111
Add 1	+ <u>1</u>
2's complement	011000000

Method 2.

Change each bit to the left of the first 1.

101000000
011000000

To subtract using the 2's complement:

1. Take the 2's complement of the subtrahend (bottom number).
2. Add it to the minuend (top number).
3. Overflow indicates that the answer is positive. Ignore the overflow (no end-around carry).
4. No overflow indicates that the answer is negative. Take the 2's complement of the original sum to obtain the true magnitude of the answer.

EXAMPLE 5-18

Subtract. $1011_2 - 100_2$

Solution

1011	→	1011
$- 100$		$+ 1100$
		<u>1 0111</u>
		↑
Overflow		

1's complement =	1011
	+ <u>1</u>
2's complement =	1100

The answer is positive 111.

Check.

$$11_{10} - 4_{10} = 7_{10}$$

EXAMPLE 5-19

Subtract. $10011_2 - 10010_2$

Solution

$$\begin{array}{r} 10011 \\ -10010 \\ \hline \end{array} \longrightarrow \begin{array}{r} 10011 \\ +01110 \\ \hline 1\ 00001 \end{array}$$

Overflow \longleftarrow

The answer is positive 1.

Check.

$$19_{10} - 18_{10} = 1_{10}$$

The process is the same when the subtrahend is larger than the minuend.

EXAMPLE 5-20

Subtract. $10010_2 - 11000_2$

Solution

$$\begin{array}{r} 10010 \\ -11000 \\ \hline \end{array} \longrightarrow \begin{array}{r} 10010 \\ +01000 \\ \hline 11010 \end{array}$$

No Overflow \longleftarrow

The answer is negative. The true magnitude is the 2's complement of 11010 or 110. The answer is -110 .

Check.

$$18_{10} - 24_{10} = -6_{10}$$

EXAMPLE 5-21

Subtract. $1001_2 - 10101_2$

Solution

$$\begin{array}{r} 1001 \\ -10101 \\ \hline \end{array} \longrightarrow \begin{array}{r} 01001 \\ +01011 \\ \hline 10100 \end{array}$$

No Overflow \longleftarrow

The answer is negative. The true magnitude is the 2's complement of 10100 or 1100. The answer is -1100 .

Check.

$$9_{10} - 21_{10} = -12_{10}$$

Two advantages of subtraction by a complement system are:

1. The procedure is the same whether the subtrahend is larger or smaller than the minuend. This saves the extra time or circuitry for a digital machine to decide if one number is larger or smaller than another.
2. The subtraction problem is converted to an addition problem. The same circuitry could be used for both processes.



5.6 2'S COMPLEMENT ADDER/SUBTRACTOR CIRCUIT

Design a circuit that will use a 7483 to add the 4-bit number B_4, B_3, B_2, B_1 to the 4-bit number A_4, A_3, A_2, A_1 and to subtract B_4, B_3, B_2, B_1 from A_4, A_3, A_2, A_1 . Use the 2's complement method for subtraction.

To use the 7483 4-bit full adder as a 2's complement adder/subtractor, the following details must be considered.

1. Refer to Figure 5-27. Leave the number B_4, B_3, B_2, B_1 unaltered for an addition problem, but take the 2's complement of the subtrahend for a subtraction problem. The 2's complement can be formed by taking the 1's complement and adding 1. The 1's complement can be formed by using exclusive-OR gates as we did in the 1's complement subtractor. 1 can be added to form the 2's complement by wiring the control signal directly to C_0 .
2. Refer to Figure 5-28. If the problem is subtraction and if there is no overflow ($C_4 = 0$), indicate the answer is negative and take the 2's complement of the result to obtain the true magnitude of the answer. As in the 1's complement subtraction circuit, C_4 can be inverted to form \bar{C}_4 . \bar{C}_4 can be ANDed with the control signal. A HIGH out of the AND gate indicates that a subtraction problem is being performed and the answer is negative. This signal will be inverted to drive an LED in the active LOW mode. Refer to Figure 5-29. A HIGH output of the AND gate also indicates the result of the addition should be 2's complemented to obtain the true magnitude of the answer. The 1's complement can be formed by exclusive-ORing the results with the output of the AND gate. To add 1 to form the 2's complement, another 7483 must be used. The output of the AND can be fed directly into C_0 of 7483-2 to complete the 2's complement process. The true magnitude outputs appear at $\Sigma_4, \Sigma_3, \Sigma_2, \Sigma_1$ of 7483-2.

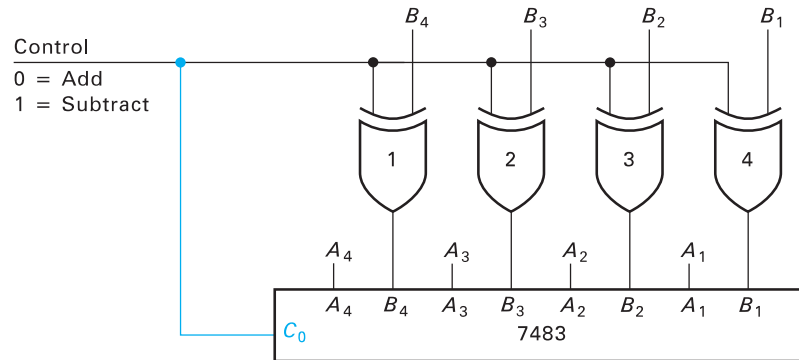


FIGURE 5-27

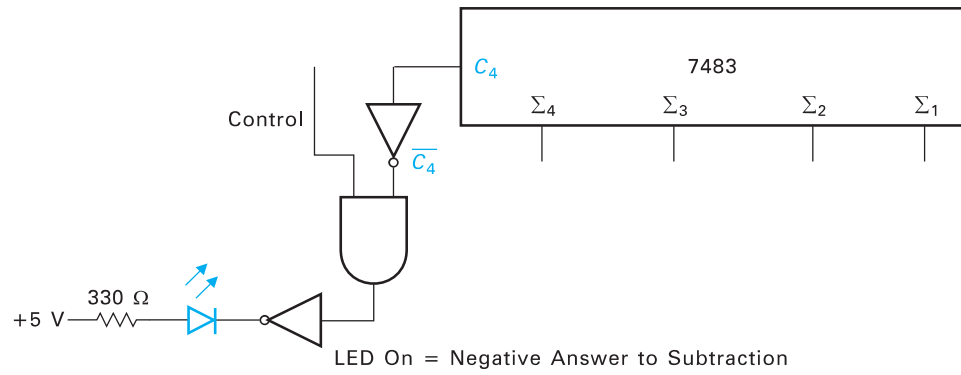


FIGURE 5-28

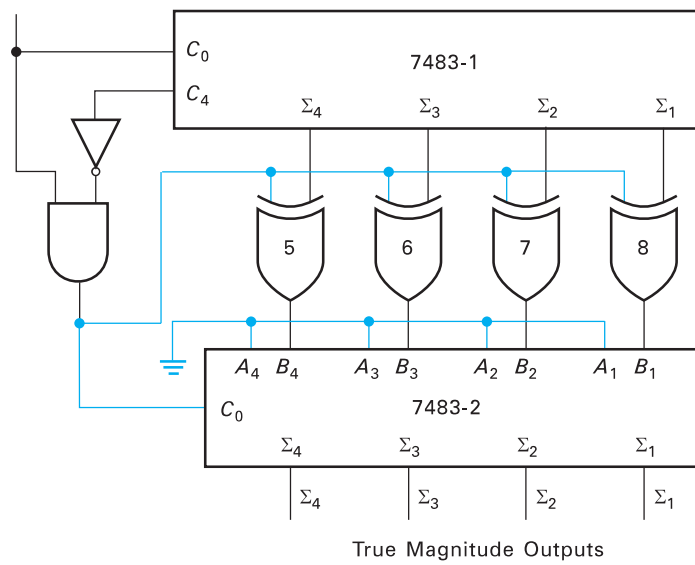


FIGURE 5-29

EXAMPLE 5-22

Add 1001 to 0101.

Solution See Figure 5-31.

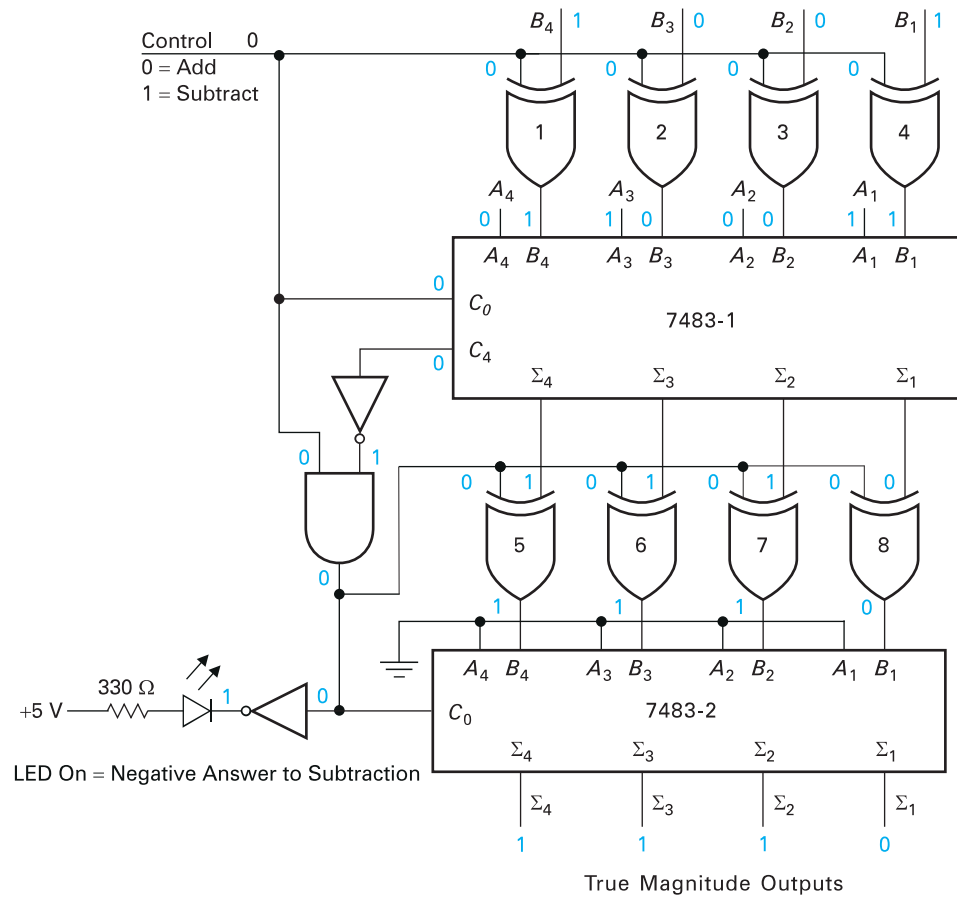


FIGURE 5-31

EXAMPLE 5-23

Subtract 0101 from 1001.

Solution See Figure 5-32.

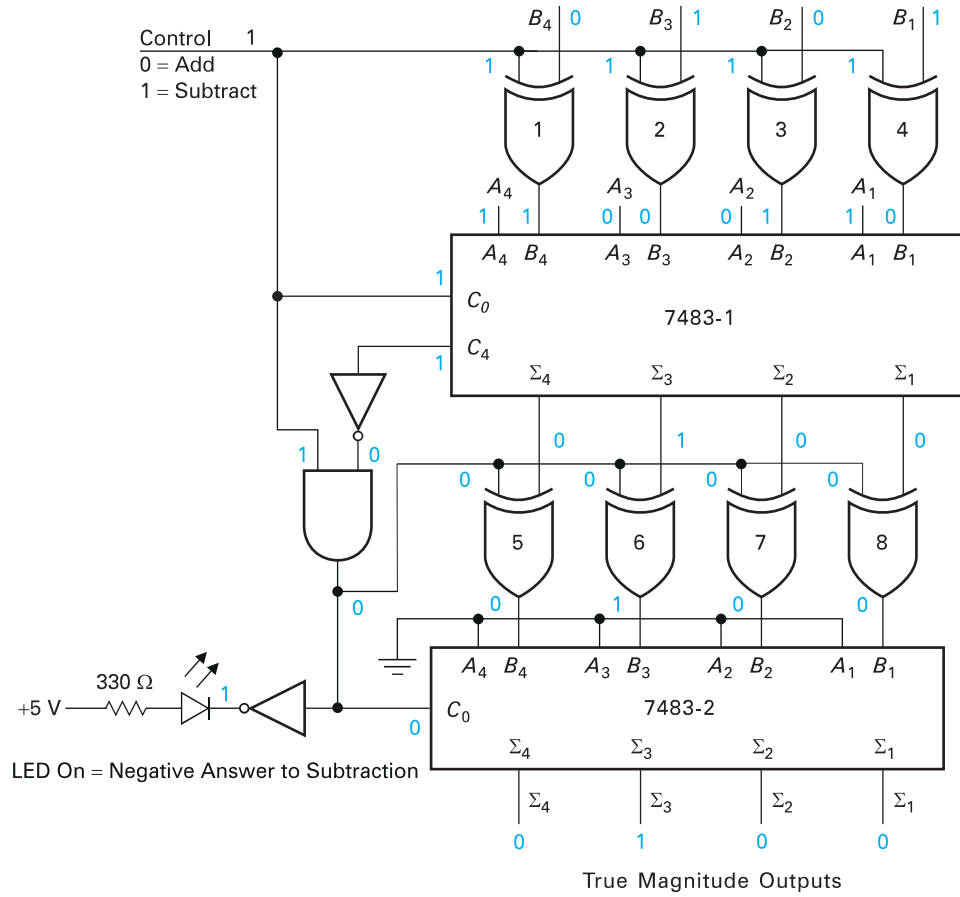


FIGURE 5-32

EXAMPLE 5-24

Subtract 1001 from 0101.

Solution See Figure 5-33.

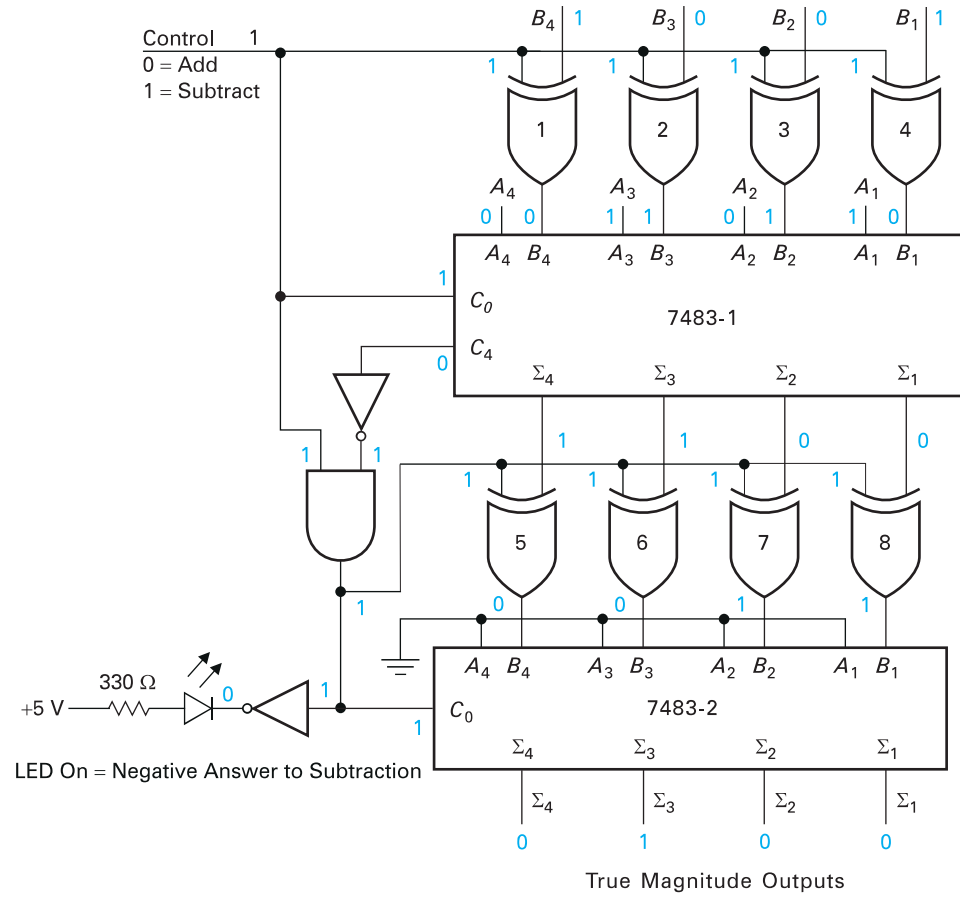


FIGURE 5-33

SELF-CHECK 3

1. Subtract using the 1's complement method. Follow the logic levels through Figure 5-23.

a. $\begin{array}{r} 0101 \\ -1010 \\ \hline \end{array}$	b. $\begin{array}{r} 11 \\ -1000 \\ \hline \end{array}$
---	---
2. Subtract using the 2's complement method. Follow the logic levels through Figure 5-30.

a. $\begin{array}{r} 1001 \\ -0110 \\ \hline \end{array}$	b. $\begin{array}{r} 1100 \\ -111 \\ \hline \end{array}$
---	--



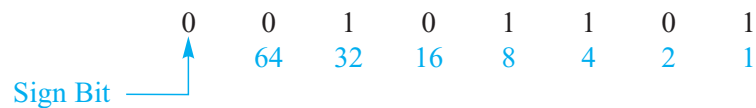
5.7 SIGNED 2'S COMPLEMENT NUMBERS

Microcomputers sometimes use one bit of a binary number to indicate the sign of the number and the remaining bits to indicate the magnitude. Negative numbers are stored in memory in 2's complement form. This system is called **signed 2's complement**. In signed 2's complement numbers, the most significant bit is used as the sign bit. A zero in the sign bit usually indicates that the number is positive and the remaining bits express the number in true magnitude form.

EXAMPLE 5-25

Convert 00101101 in a signed 2's complement system to a decimal number.

Solution



The true magnitude is $32 + 8 + 4 + 1 = 45$. The number is positive. The answer is 45_{10} .

EXAMPLE 5-26

What is the highest positive number that can be represented in an 8-bit signed 2's complement system?

Solution The highest positive number that can be represented in an 8-bit signed 2's complement system is 01111111.



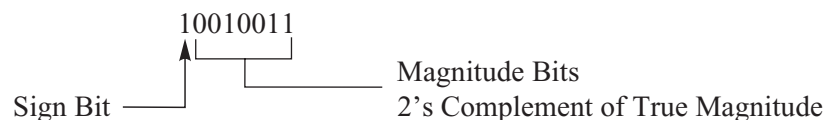
The true magnitude is $64 + 32 + 16 + 8 + 4 + 2 + 1 = 127$. The number is positive. The positive number 127_{10} is the highest decimal number that can be represented in this system.

A 1 in the sign bit usually indicates that the number is negative. The remaining bits express the number in 2's complement form.

EXAMPLE 5-27

Convert 10010011 in a signed 2's complement system to a decimal number.

Solution



To find the true magnitude, take the 2's complement of the complete number, including the sign bit.

$$2\text{'s complement} = 10010011$$

$$\text{True magnitude} = 01101101$$

0	1	1	0	1	1	0	1
128	64	32	16	8	4	2	1

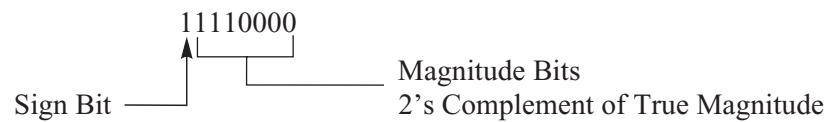
$$64 + 32 + 8 + 4 + 1 = 109$$

The number is negative. The answer is -109_{10} .

EXAMPLE 5-28

Convert 11110000 in a signed 2's complement system to a decimal number.

Solution



$$2\text{'s complement} = 11110000$$

$$\text{True magnitude} = 00010000$$

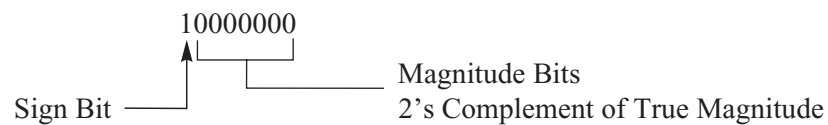
0	0	0	1	0	0	0	0
128	64	32	16	8	4	2	1

The number is negative. The answer is -16_{10} .

EXAMPLE 5-29

What is the most negative number that can be represented in an 8-bit signed 2's complement system?

Solution The most negative number that can be represented in an 8-bit signed 2's complement system is 10000000.



$$2\text{'s complement} = 10000000$$

$$\text{True magnitude} = 10000000$$

1	0	0	0	0	0	0	0
128	64	32	16	8	4	2	1

The negative number -128_{10} is the most negative number that can be represented in this system.

In an 8-bit signed 2's complement system, the numbers can range from -128_{10} to $+127_{10}$.

To express a negative decimal number in signed 2's complement form, convert the magnitude into binary and then take the 2's complement.

EXAMPLE 5-30

Express -78_{10} as an 8-bit signed 2's complement number.

Solution

$$78_{10} =$$

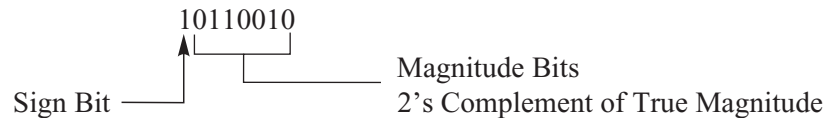
0	1	0	0	1	1	1	0
128	64	32	16	8	4	2	1

$$\text{True magnitude} = 01001110$$

$$\text{2's complement} = 10110010$$

$$-78_{10} = 10110010 \text{ (signed 2's complement)}$$

Check.



$$\text{2's complement} = 10110010$$

$$\text{True magnitude} = 01001110$$

0	1	0	0	1	1	1	0
128	64	32	16	8	4	2	1

The number is negative. The 8-bit signed 2's complement number 10110010 is equal to $-(64 + 8 + 4 + 2) = -78_{10}$.

Numbers in signed 2's complement form can be added using straight binary addition and subtracted by taking the 2's complement of the subtrahend and adding. The sign bit will indicate the sign of the answer. Positive answers will be in true magnitude form. Negative answers will be in 2's complement form.

EXAMPLE 5-31

Add these 8-bit signed 2's complement numbers.

$$01011001 + 10101101$$

Solution

0 1 0 1 1 0 0 1	(+89)
+ 1 0 1 0 1 1 0 1	(-83)
1 0 0 0 0 0 1 1 0	(+ 6)

Ignore Overflow ↑

EXAMPLE 5-32

Add these 8-bit signed 2's complement numbers. Express the answer in decimal form.

$$11011001 + 10101101$$

Solution

$$\begin{array}{r}
 11011001 \quad (-39) \\
 +10101101 \quad (-83) \\
 \hline
 110000110 \quad (-122)
 \end{array}$$

Ignore Overflow \rightarrow (points to the carry bit)
 Sign Bit \rightarrow (points to the leftmost bit)
 Magnitude Bits \rightarrow (points to bits 1-7)
 2's Complement of True Magnitude \rightarrow (points to bits 1-7)

2's complement = 10000110

True magnitude = 01111010

0	1	1	1	1	0	1	0
128	64	32	16	8	4	2	1

The number is negative. The answer is

$$-(64 + 32 + 16 + 8 + 2) = -122$$

Check.

$$-39 + (-83) = -122$$

EXAMPLE 5-33

Subtract these 8-bit signed 2's complement numbers. Express the answer in decimal form.

$$01011011 - 11100101$$

Solution To subtract, take the 2's complement of the subtrahend and add.

$$\begin{array}{r}
 01011011 \longrightarrow 01011011 \\
 -11100101 \longrightarrow +00011011 \\
 \hline
 01110110
 \end{array}$$

No Overflow \rightarrow (points to the carry bit)
 Sign Bit \rightarrow (points to the leftmost bit)

1	1	1	0	1	1	0
64	32	16	8	4	2	1

The answer is positive. The answer is

$$64 + 32 + 16 + 4 + 2 = 118$$

Check.

$$91 - (-27) = 118$$

EXAMPLE 5-34

Subtract these 8-bit signed 2's complement numbers. Express the answer in decimal form.

$$10001010 - 11111100$$

Solution To subtract, take the 2's complement of the subtrahend and add.

$$\begin{array}{r} 10001010 \\ -11111100 \\ \hline \end{array} \longrightarrow \begin{array}{r} 10001010 \\ +00000100 \\ \hline 10001110 \end{array}$$

No Overflow
Sign Bit
Magnitude Bits
2's Complement of
True Magnitude

$$2's \text{ complement} = 10001110$$

$$\text{True magnitude} = 01110010$$

0	1	1	1	0	0	1	0
128	64	32	16	8	4	2	1

The number is negative. The answer is

$$-(64 + 32 + 16 + 2) = -114$$

Check.

$$-118 - (-4) = -114$$

In each of the examples of signed 2's complement mathematics presented so far, the result has been correct. To ensure that the result is correct, the carry from column 7 into the sign bit and the overflow must be monitored and the following rules observed.

1. If there is a carry from column 7 into the sign bit and an overflow, the answer is correct.
2. If there is no carry from column 7 into the sign bit and no overflow, the answer is correct.
3. If there is no carry from column 7 into the sign bit and overflow occurs or vice versa, the answer is not correct.

Systems that use signed 2's complement mathematics must monitor the carry from column 7 into the sign bit and the overflow to signal whether or not an error has occurred.

In the following two examples, the results are not correct.

EXAMPLE 5-35

Subtract these 8-bit signed 2's complement numbers. Express the answer in decimal form.

4. Subtract these signed 2's complement numbers. State whether the result is correct or incorrect.

$$\begin{array}{r} 00110110 \\ - 10101110 \\ \hline \end{array} \qquad \begin{array}{r} 10001111 \\ - 10101101 \\ \hline \end{array}$$



5.8 BINARY-CODED-DECIMAL ADDITION

Recall that BCD uses four bits to represent a decimal number as shown in Figure 5-34. Although legitimate BCD numbers must stop at nine, there are six more counts before all four columns are full. These six steps are not legitimate BCD numbers. In BCD addition, care must be taken to compensate for these six forbidden states. If overflow occurs during an addition, or if one of the forbidden states occurs as a result of an addition, then six must be added to the result to “flip through” the unwanted states. In Figure 5-34, start at 7 and add 5. The result is 1100. To flip out of the forbidden states, count six more. The answer is 0010 or 2 with a carry to the next column. When you reach 1111 the next count is 0000 and a carry has occurred.

$$7 + 5 = 12$$

Legitimate BCD Numbers	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
Forbidden Numbers	1	0	1	0	
	1	0	1	1	
	1	1	0	0	
	1	1	0	1	
	1	1	1	0	
	1	1	1	1	

FIGURE 5-34 BCD numbers

EXAMPLE 5-37

Add 3 plus 5.

Solution

$$\begin{array}{r} 0011 \\ +0101 \\ \hline 1000 \end{array}$$

There is no overflow, and the result is a legitimate BCD number, so it is correct. The answer is 8.

EXAMPLE 5-38

Add 8 plus 5.

$$\begin{array}{r} \text{Solution} \quad 1000 \\ + 0101 \\ \hline 1101 \end{array}$$

There is no overflow, but the result is not a legitimate number. Six must be added to compensate for the six forbidden numbers.

$$\begin{array}{r} 1101 \\ + 0110 \\ \hline 10011 \end{array}$$

The answer is 13.

EXAMPLE 5-39

Add 8 plus 9.

$$\begin{array}{r} \text{Solution} \quad 1000 \\ + 1001 \\ \hline 10001 \end{array}$$

The result is a legitimate BCD number, but there is overflow. Six must be added to compensate for the forbidden states.

$$\begin{array}{r} 10001 \\ + 0110 \\ \hline 10111 \end{array}$$

The answer is 17.

EXAMPLE 5-40

Add 167 plus 396.

Solution	<u>1</u>	<u>1</u>	<u> </u>	Carries
	0001	0110	0111	
	+ 0011	<u>1001</u>	<u>0110</u>	
	0101	0000	1101	

Six is added to the least significant digit because the result is not a legitimate BCD number. Six must also be added to the middle digit because of the overflow. The most significant digit result produced no overflow, and it is a legitimate BCD number, so it is not necessary to add six.

0101	0000	1101
+ 0	<u>0110</u>	<u>0110</u>
0101	0110	0011

The answer is 563.



5.9 BINARY-CODED-DECIMAL ADDER CIRCUIT

To convert a binary adder into a BCD adder, logic must be provided that will produce a signal to indicate whether six should be added to the result of an addition. The carry out of the binary adder can be monitored to see if overflow resulted. But how do you distinguish a legitimate BCD number from a forbidden one? See Figure 5-34.

All 4-bit numbers above nine have 1s in the eight's column and a 1 in the four's column or two's column. Written in Boolean, this is $8(4 + 2)$. This signal can be produced with a two-input OR gate and a two-input AND gate. If this signal is 1, or if overflow occurs ($C_4 = 1$), six must be added to compensate for the six forbidden states. $8(2 + 4)$ is ORed with C_4 to produce the ADD 6 signal. Another 7483 will be used to add six to the result from 7483-1 when the ADD 6 signal is 1. Since 6 is 0110 in binary, A_4 and A_1 will be grounded, while the ADD 6 signal will be wired to A_3 and A_2 . When ADD 6 is 0, A_4, A_3, A_2, A_1 will be 0000. When ADD 6 is 1, A_4, A_3, A_2, A_1 will be 0110 or 6. C_0 of 7483-2 must be grounded, or seven will be added instead of six. The complete schematic is shown in Figure 5-35.

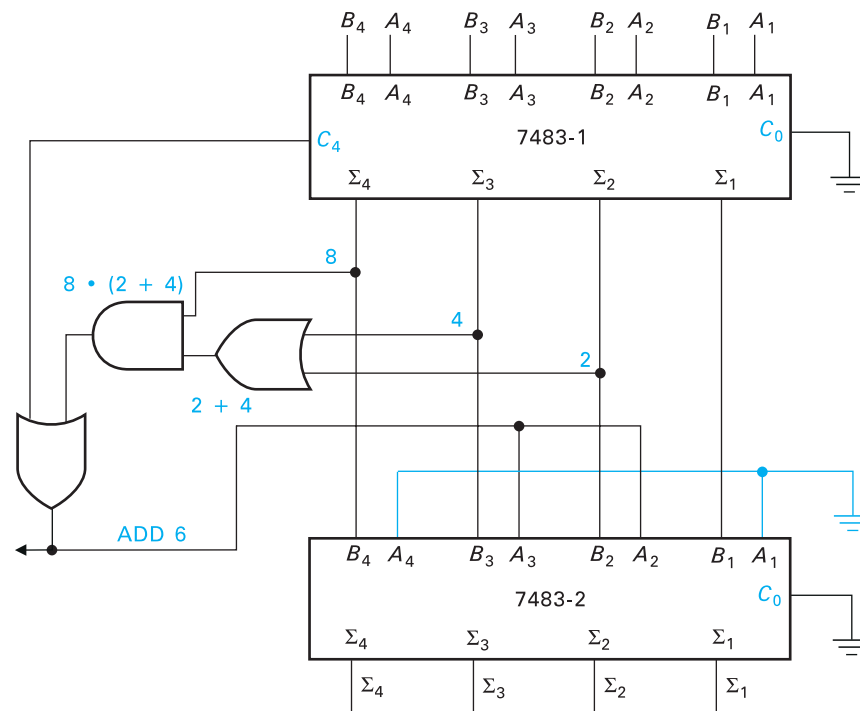


FIGURE 5-35 BCD adder

EXAMPLE 5-41

Use the BCD adder to add 9 and 3.

Solution See Figure 5-36. The sum from the first adder is 1100, which generates an ADD 6 signal and a carry to the next stage. The second adder adds 1100 + 0110 for a sum of 0010 or 2. The answer is 12.

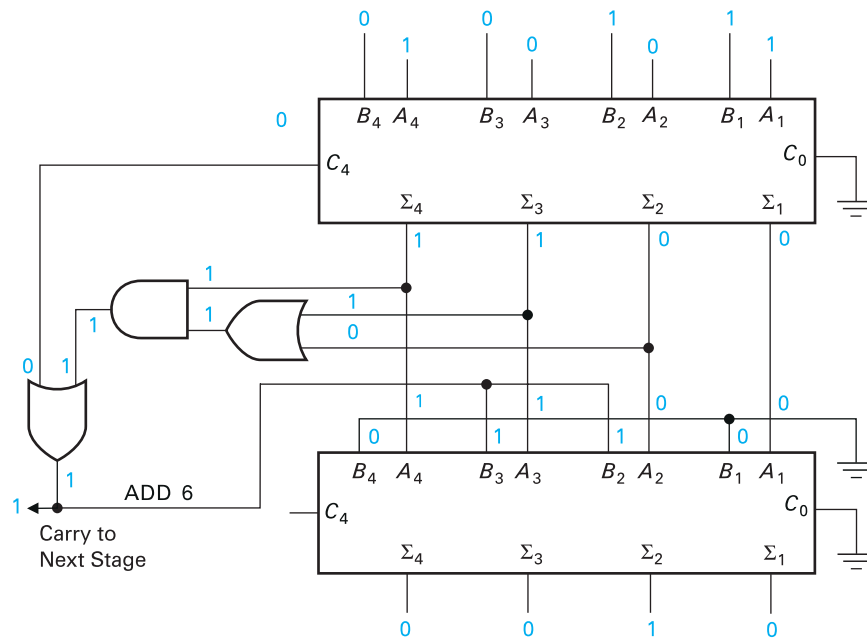


FIGURE 5-36

EXAMPLE 5-42

Use the BCD adder to add 9 and 7.

Solution See Figure 5-37. The sum from the first adder is 0000 with a 1 out on C_4 . This time C_4 generates the ADD 6 signal and the carry to the next stage. The answer is 16.

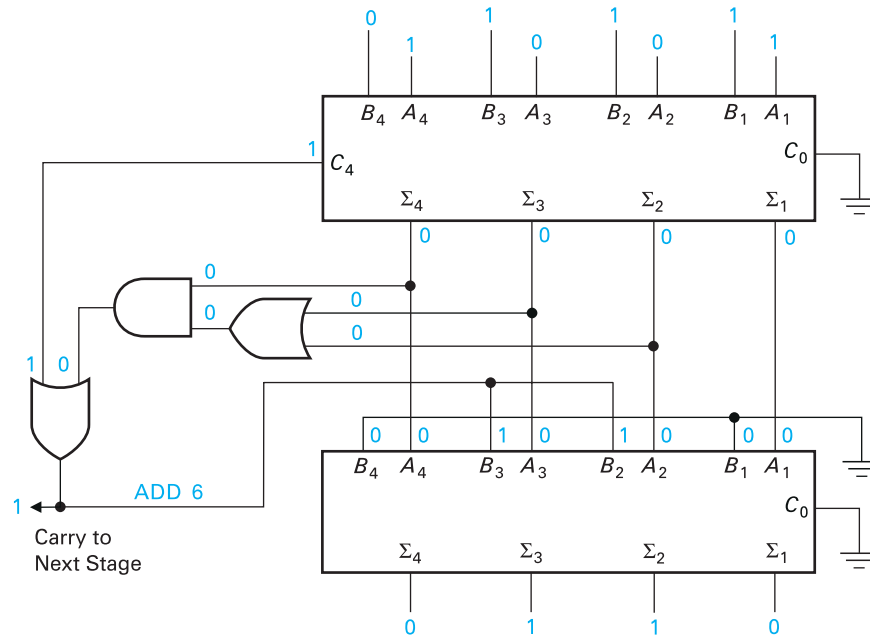


FIGURE 5-37

Follow several examples through until you understand the functioning of the ADD 6 circuit.

SELF-CHECK 5

1. List the forbidden numbers in the BCD number system.
2. List two conditions that cause six to be added to the preliminary sum in a BCD adder.
3. Add these BCD numbers. Follow the logic levels through Figure 5-35.

a. $\begin{array}{r} 0111 \\ + 1000 \\ \hline \end{array}$	b. $\begin{array}{r} 1001 \\ + 1001 \\ \hline \end{array}$
--	--



5.10 ARITHMETIC LOGIC UNIT (ALU)

An **arithmetic logic unit (ALU)** performs addition and subtraction as well as logical operations like AND and OR on the input data. The 74181 4-bit ALU has four Function Select inputs, S_3 – S_0 , that select sixteen different arithmetic operations or sixteen different logic operations. When the mode control input M is HIGH, all internal carries are inhibited, and the ALU performs logic operations, such as AND, OR, NAND, NOR, ex-OR, and ex-NOR. When M is LOW, the carries are enabled, and the ALU performs arithmetic operations, such as add, subtract, compare, and double. Table 5-2 defines the sixteen arithmetic functions and logic functions performed by the 74181.

TABLE 5-2 74181 Function Table

S_3	S_2	S_1	S_0	($M = 1$) Logic	($M = 0$) Arithmetic
L	L	L	L	\bar{A}	$A + 1$
L	L	L	H	$\bar{A} + \bar{B}$	$A + B$
L	L	H	L	$\bar{A} B$	$A + \bar{B}$
L	L	H	H	0	0
L	H	L	L	$\bar{A} \bar{B}$	A plus $A \bar{B}$
L	H	L	H	\bar{B}	$A \bar{B}$ plus $(A + B)$
L	H	H	L	$A + B$	A minus B
L	H	H	H	$A \bar{B}$	$A \bar{B}$
H	L	L	L	$\bar{A} + B$	$A B$ plus A
H	L	L	H	$A + B$	A plus B
H	L	H	L	B	$A B$ plus $(A + \bar{B})$
H	L	H	H	$A B$	$A B$
H	H	L	L	1	A plus A
H	H	L	H	$A + \bar{B}$	A plus $(A + B)$
H	H	H	L	$A + B$	A plus $(A + \bar{B})$
H	H	H	H	A	$A - 1$

Arithmetic operations are performed on two 4-bit words A_3, A_2, A_1, A_0 and B_3, B_2, B_1, B_0 . The result appears on F_3, F_2, F_1, F_0 . The 74181 is a full adder. The carry-in is called C_n and the carry-out is called C_{n+4} . C_n and C_{n+4} are active low. Place a LOW on C_n to represent a carry-in and a HIGH to represent no carry-in. Negative results are presented in 2's complement form.

EXAMPLE 5-43

If M is LOW, $S = 1001$, $A = 1011$, $B = 1000$, and C_n is HIGH, predict the outputs F and C_{n+4} .

Solution With M LOW, arithmetic functions are selected. $S = 1001$ selects the function A plus B . $C_n = \text{HIGH}$ means no carry-in. The 74181 will add A and B with no carry-in. $C_{n+4} = \text{LOW}$ (signifying carry-out) and $F = 0011$.

Check.

$$11 + 8 + 0 = 19$$

EXAMPLE 5-44

If S is changed to 1100, predict the outputs.

Solution 1100 selects the function A plus A . $C_{n+4} = \text{LOW}$ signifying carry-out and $F = 0110$.

Check.

$$11 \text{ plus } 11 = 22.$$

Logic operations are performed on individual pairs of bits. The function $A + B$ ORs A_0 with B_0 and the result appears on F_0 . Likewise, A_1 is ORed with B_1 and the result appears on F_1 and so on. For example, when 1010 is ORed with 1001 the result is 1011. The carries C_n and C_{n+4} are disabled during logical operations.

EXAMPLE 5-45

If $M = \text{HIGH}$, $S = 1011$, $A = 0110$, and $B = 1100$, predict the output F .

Solution $M = \text{HIGH}$ selects the logic functions. $S = 1011$ selects AB ($A \text{ AND } B$). C_n and C_{n+4} are disabled. Corresponding bits of A and B are ANDed. $F = 0100$.

EXAMPLE 5-46

If S is changed to 0001, predict the output.

Solution $S = 0001$ selects $\bar{A} + \bar{B}$. Both A and B must be complemented and then corresponding bits are ORed together.

$$\bar{A} = 1001, \bar{B} = 0011, \text{ and } F = 1011.$$

Here are some other arithmetic logic unit ICs:

74381	4-bit ALU
74382	4-bit ALU with overflow output for 2's complement
74881	4-bit ALU
74582	4-bit BCD ALU
74583	4-bit BCD adder
74882	32-bit look ahead carry generator

The GAL is built to implement sum of product expressions, so the first task is to write Boolean expressions for the three outputs. Here is the output expression for C_2 .

$$\begin{aligned}
 C_2 = & \\
 & \overline{A_2} \overline{A_1} B_2 B_1 C_0 + \overline{A_2} A_1 B_2 \overline{B_1} C_0 + \overline{A_2} A_1 B_2 B_1 \overline{C_0} + \\
 & \overline{A_2} A_1 B_2 B_1 C_0 + A_2 \overline{A_1} \overline{B_2} B_1 C_0 + A_2 \overline{A_1} B_2 \overline{B_1} \overline{C_0} + \\
 & A_2 \overline{A_1} B_2 \overline{B_1} C_0 + A_2 \overline{A_1} B_2 B_1 \overline{C_0} + A_2 \overline{A_1} B_2 B_1 C_0 + \\
 & A_2 A_1 \overline{B_2} \overline{B_1} C_0 + A_2 A_1 \overline{B_2} B_1 \overline{C_0} + A_2 A_1 \overline{B_2} B_1 C_0 + \\
 & A_2 A_1 B_2 \overline{B_1} \overline{C_0} + A_2 A_1 B_2 \overline{B_1} C_0 + A_2 A_1 B_2 B_1 \overline{C_0} + \\
 & A_2 A_1 B_2 B_1 C_0
 \end{aligned}$$

This expression has sixteen product terms, as do the expressions for Σ_2 and Σ_1 . Each product term has five variables ANDed together. Can the GAL16V8 handle these three equations? Figure 5-38A shows the GAL16V8 configured in the simple mode. Each of the eight cells has an OR gate fed by eight AND gates as shown in Figures 5-38A and B. Each AND gate has 32 inputs. One cell of the GAL16V8 can OR eight of the terms and that output can be fed into another cell along with seven more product terms. By feeding the output of one cell into another we can process fifteen product terms. Each of our output expressions have sixteen terms. So, to implement each output without reducing the equations, the output of the second cell can be fed into the third. Each output C_2 , Σ_2 , Σ_1 will require three cells in the GAL. The GAL16V8 only contains eight cells; we have to find a larger GAL or reduce the equations. C_2 reduces to:

$$\begin{aligned}
 & B_2 B_1 C_0 + A_2 B_1 C_0 + A_1 B_2 B_1 + \\
 & A_2 A_1 C_0 + A_1 B_2 C_0 + A_2 A_1 B_2 + \\
 & A_2 B_2 \overline{B_1} + A_2 B_2 \overline{C_0} + A_2 A_1 B_1
 \end{aligned}$$

This will take two cells to implement.

Σ_1 looks like this:

$$\begin{aligned}
 & \overline{A_2} \overline{A_1} \overline{B_2} \overline{B_1} C_0 + \overline{A_2} \overline{A_1} \overline{B_2} B_1 \overline{C_0} + \overline{A_2} \overline{A_1} B_2 \overline{B_1} C_0 + \\
 & \overline{A_2} \overline{A_1} B_2 B_1 \overline{C_0} + \overline{A_2} A_1 \overline{B_2} \overline{B_1} \overline{C_0} + \overline{A_2} A_1 \overline{B_2} B_1 C_0 + \\
 & \overline{A_2} A_1 B_2 \overline{B_1} \overline{C_0} + \overline{A_2} A_1 B_2 B_1 C_0 + A_2 \overline{A_1} \overline{B_2} \overline{B_1} C_0 + \\
 & A_2 \overline{A_1} \overline{B_2} B_1 \overline{C_0} + A_2 \overline{A_1} B_2 \overline{B_1} C_0 + A_2 \overline{A_1} B_2 B_1 \overline{C_0} + \\
 & A_2 A_1 \overline{B_2} \overline{B_1} \overline{C_0} + A_2 A_1 \overline{B_2} B_1 C_0 + A_2 A_1 B_2 \overline{B_1} \overline{C_0} + \\
 & A_2 A_1 B_2 B_1 C_0
 \end{aligned}$$

Σ_1 reduces to this:

$$\overline{A_1} \overline{B_1} C_0 + \overline{A_1} B_1 \overline{C_0} + A_1 \overline{B_1} \overline{C_0} + A_1 B_1 C_0$$

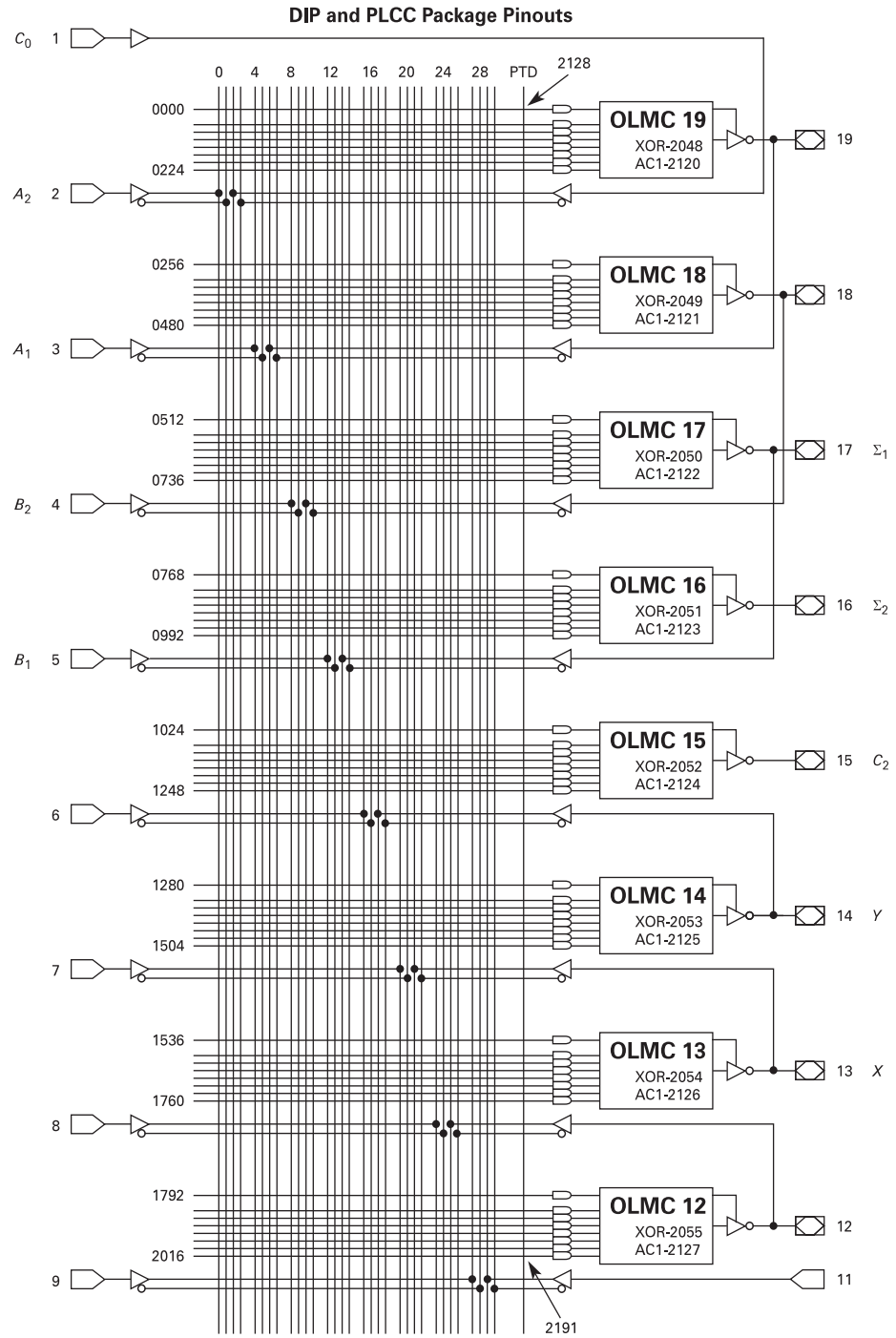


FIGURE 5-38A Input and output variables (Courtesy of Lattice Semiconductor Corporation)

This expression makes sense because A_1 , B_1 , and C_0 determine Σ_1 . Σ_1 will be 1 when exactly one of those three is HIGH (first three terms) or when all three are HIGH (last term).

Σ_2 looks like this:

$$\begin{aligned} &\bar{A}_2 \bar{A}_1 \bar{B}_2 B_1 C_0 + \bar{A}_2 \bar{A}_1 B_2 \bar{B}_1 \bar{C}_0 + \bar{A}_2 \bar{A}_1 B_2 \bar{B}_1 C_0 + \\ &\bar{A}_2 \bar{A}_1 B_2 B_1 \bar{C}_0 + \bar{A}_2 A_1 \bar{B}_2 \bar{B}_1 C_0 + \bar{A}_2 A_1 \bar{B}_2 B_1 \bar{C}_0 + \\ &\bar{A}_2 A_1 \bar{B}_2 B_1 C_0 + \bar{A}_2 A_1 B_2 \bar{B}_1 \bar{C}_0 + A_2 \bar{A}_1 \bar{B}_2 \bar{B}_1 \bar{C}_0 + \\ &A_2 \bar{A}_1 \bar{B}_2 \bar{B}_1 C_0 + A_2 \bar{A}_1 \bar{B}_2 B_1 \bar{C}_0 + A_2 \bar{A}_1 B_2 B_1 C_0 + \\ &A_2 A_1 \bar{B}_2 \bar{B}_1 \bar{C}_0 + A_2 A_1 B_2 \bar{B}_1 C_0 + A_2 A_1 B_2 B_1 \bar{C}_0 + \\ &A_2 A_1 B_2 B_1 C_0 \end{aligned}$$

Σ_2 reduces to this:

$$\begin{aligned} &\bar{A}_2 \bar{A}_1 B_2 \bar{B}_1 + \bar{A}_2 \bar{A}_1 B_2 \bar{C}_0 + \bar{A}_2 A_1 \bar{B}_2 C_0 + \\ &\bar{A}_2 A_1 \bar{B}_2 B_1 + \bar{A}_2 \bar{B}_2 B_1 C_0 + \bar{A}_2 B_2 \bar{B}_1 \bar{C}_0 + \\ &A_2 \bar{A}_1 \bar{B}_2 \bar{B}_1 + A_2 \bar{A}_1 \bar{B}_2 \bar{C}_0 + A_2 \bar{B}_2 \bar{B}_1 \bar{C}_0 + \\ &A_2 A_1 B_2 C_0 + A_2 A_1 B_2 B_1 + A_2 B_2 B_1 C_0 \end{aligned}$$

These twelve product terms can be implemented by two GAL16V8 cells. The next step is to write a source program using an ASCII editor. Our program will be called 2bitadd.PLD. The PLD extension is necessary for the CUPL compiler. First let's define which input pins in Figure 5-38A to use for each input variable.

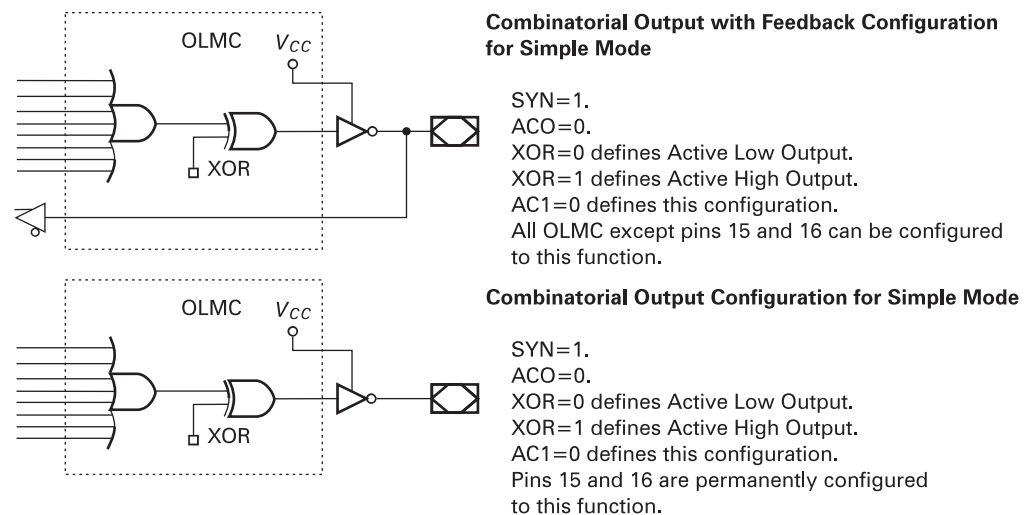


FIGURE 5-38B Output logic macro cell (OLMC) (Courtesy of Lattice Semiconductor Corporation)

The input part of the program will look like this. Comment lines or groups of lines begin with /* and end with */. Each non-comment line ends with a semicolon. Subscripts and Greek letters cannot be used. E will be used for Σ .

```

***** Define inputs *****/
Pin 1 = C0;
Pin 2 = A2;
Pin 3 = A1;
Pin 4 = B2;
Pin 5 = B1;

```

Define the pins in Figure 5-38 to be used for each output variable. Σ_2 and C_2 have more than eight product terms and will use more than one cell. Intermediate outputs X and Y are defined for that purpose.

```

***** Define outputs *****/
Pin 17 = E1;
Pin 16 = E2;
Pin 15 = C2;
Pin 13 = X;
Pin 14 = Y;

```

Use Boolean expressions to define the outputs in terms of the inputs. The parentheses used in the following statements are not necessary, but they make the statements more readable. The first seven terms of C_2 are ORed together in cell 13 to produce output X . X is used as an input to cell 15 to produce C_2 . The first seven terms of Σ_2 are ORed together in cell 14 to produce output Y . Y is used as an input to cell 16 to produce Σ_2 .

```

*** Logic:      ! = NOT      # = OR      & = AND      ***/
X = (B2 & B1 & C0) # (A2 & B1 & C0) # (A1 & B2 & B1) # (A2 & A1 & C0) #
(A1 & B2 & C0) # (A2 & A1 & B2) # (A2 & B2 & !B1);

C2 = X # (A2 & B2 & !C0) # (A2 & A1 & B1);

Y = (!A2 & !A1 & B2 & !B1) # (!A2 & !A1 & B2 & !C0) # (!A2 & A1 & !B2 & C0) #
(!A2 & A1 & !B2 & B1) # (!A2 & !B2 & B1 & C0) # (!A2 & B2 & !B1 & !C0) #
(A2 & !A1 & !B2 & !B1);

E2 = Y # (A2 & !A1 & !B2 & !C0) # (A2 & !B2 & !B1 & !C0) # (A2 & A1 & B2 & C0) #
(A2 & A1 & B2 & B1) # (A2 & B2 & B1 & C0);

E1 = (!A1 & !B1 & C0) # (!A1 & B1 & !C0) # (A1 & !B1 & !C0) # (A1 & B1 & C0);

```

The completed program is shown in Figure 5-39. Each line of the opening comment section is mandatory, even if no useful information is entered.

```

Name          2bitadd;
Partno        0000002;
Revision      01;
Date          6/26/99;
Designer      RLDonovan;
Company       MCC;
Location      None;
Assembly      None;
Device        G16V8AS;

/*****      Target device: GAL16V8      *****/

/*****      Define inputs      *****/

Pin 1 = C0;
Pin 2 = A2;
Pin 3 = A1;
Pin 4 = B2;
Pin 5 = B1;

/****Define outputs*****/

Pin 17 = E1;
Pin 16 = E2;
Pin 15 = C2;
Pin 14 = Y;
Pin 13 = X;

/*** Logic:      ! = NOT      # = OR      & = AND      ***/

X = (B2 & B1 & C0) # (A2 & B1 & C0) # (A1 & B2 & B1) # (A2 & A1 & C0) #
(A1 & B2 & C0) # (A2 & A1 & B2) # (A2 & B2 & !B1);

C2 = X # (A2 & B2 & !C0) # (A2 & A1 & B1);

Y = (!A2 & !A1 & B2 & !B1) # (!A2 & !A1 & B2 & !C0) # (!A2 & A1 & !B2 & C0) #
(!A2 & A1 & !B2 & B1) # (!A2 & !B2 & B1 & C0) # (!A2 & B2 & !B1 & !C0) #
(A2 & !A1 & !B2 & !B1);

E2 = Y # (A2 & !A1 & !B2 & !C0) # (A2 & !B2 & !B1 & !C0) # (A2 & A1 & B2 & C0)
# (A2 & A1 & B2 & B1) # (A2 & B2 & B1 & C0);

E1 = (!A1 & !B1 & C0) # (!A1 & B1 & !C0) # (A1 & !B1 & !C0) # (A1 & B1 & C0);

```

FIGURE 5-39 2bitadd.PLD

The .PLD file is now compiled using a program called Universal Compiler for Programmable Logic (CUPL). CUPL produces a JEDEC file, in this case, 2bitadd.JED. The JEDEC file contains a chart of 1s and 0s for each programmable fuse in the device. 2bitadd.JED is shown in Figure 5-40. The numbers at the beginning of each line

correspond to a fuse number in the array in Figure 5-38. A 0 indicates the connection is to remain; a 1 indicates that the connection will be erased. The JEDEC file is sent to the programming hardware to program the GAL16V8.

```

CUPL(TD)    4.8a Serial# ED-20001886
Device      g16v8as Library DLIB-h-37-2
Created     Sun Jun 28 17:27:45 1999
Name        2bitadd
Partno      0000002
Revision    01
Date        6/26/99
Designer    RLDonovan
Company     MCC
Assembly    None
Location    None
*QP20
QF2194
*G0
*F0
*L00512 11011011111101111111111111111111
*L00544 11101011111101111111111111111111
*L00576 11100111111101111111111111111111
*L00608 11010111111101111111111111111111
*L00768 11111111111111101111111111111111
*L00800 01101011101111111111111111111111
*L00832 01101111101110111111111111111111
*L00864 01010111011111111111111111111111
*L00896 01110111011101111111111111111111
*L00928 01011111011101111111111111111111
*L01024 11111111111111111111110111111111
*L01056 01101111101111111111111111111111
*L01088 01110111111011111111111111111111
*L01280 10111011011110111111111111111111
*L01312 10101011011111111111111111111111
*L01344 10010111101111111111111111111111
*L01376 10110111101101111111111111111111
*L01408 10011111101101111111111111111111
*L01440 10101111011110111111111111111111
*L01472 01111011101110111111111111111111
*L01536 11011111011101111111111111111111
*L01568 01011111111011111111111111111111
*L01600 11110111011101111111111111111111
*L01632 01010111111111111111111111111111
*L01664 11010111011111111111111111111111
*L01696 01110111011111111111111111111111

```

FIGURE 5-40 2bitadd.JED

```

*L01728 01111111011110111111111111111111
*L02048 00111110001100000011000000110000
*L02080 00110000001100000011000000110010
*L02112 00000000110000011111111111111111
*L02144 11111111111111111111111111111111
*L02176 111111111111111110
*C715C
*□4811

```

FIGURE 5-40 (continued)

SELF-CHECK 7

1. Apply the JEDEC file in Figure 5-40 to the GAL16V8 array in Figure 5-38 to determine what expressions will be programmed into the IC. (Match the zeros in the JEDEC file with the intersections in Figure 5-38.)
2. Write the PLD file for a half adder.
3. Write the PLD file for a full adder.



5.12 TROUBLESHOOTING ADDER CIRCUITS

In the Chapter 5 labs you will be working with the adder circuits discussed in this chapter. Chapter 1 discussed problems that can be encountered with a 4-bit full adder. Review those concepts. The adder circuits in this chapter involve several components. A systematic approach to troubleshooting these circuits needs to be taken.

1. Set the inputs for a problem that will test a particular aspect of the circuit. For example, in a BCD adder three test cases should be considered: a problem that requires no “add 6” correction, a problem that requires the “add 6” because an unused result occurred in the preliminary addition, and a problem that requires the “add 6” because C_4 occurred. $0 + 0 = 0$ is a good first test. If the circuit gives an answer of 6, check out the “add 6” circuit to see how it is being enabled.
2. Determine whether the final output is correct.
3. If not, do these preliminary checks before spending time searching for a problem in the hardware.
 - a. Check the power and ground connections on each IC.
 - b. Check the inputs to the first adder IC to ensure that the adder circuit is working the correct problem.
4. If power connections and inputs are correct and the output is not, check voltages in the middle of the circuit to “divide and conquer.” If the voltages in the

middle are correct, the fault is in the second half of the circuit. If the voltages in the middle are not correct, the fault is in the first half of the circuit. Continue to divide the circuit until the general area of the fault is located.

5. If the inputs to a device are correct and the outputs are not, there are two possibilities.
 - a. The device is bad.
 - b. Something connected to the output of the device is loading it down (keeping it from assuming its proper voltage level). This condition is often overlooked.

EXAMPLE 5-47

Troubleshoot the BCD adder circuit in Figure 5-41.

Solution

Step 1. A test problem has been entered. $0000 + 0000 = 1000$

Step 2. Check the preliminary sum from the first adder. 0000 is correct. The problem lies in the second half of the circuit. Since the result is 8 and not 6, the “ADD 6” circuit is probably not the culprit.

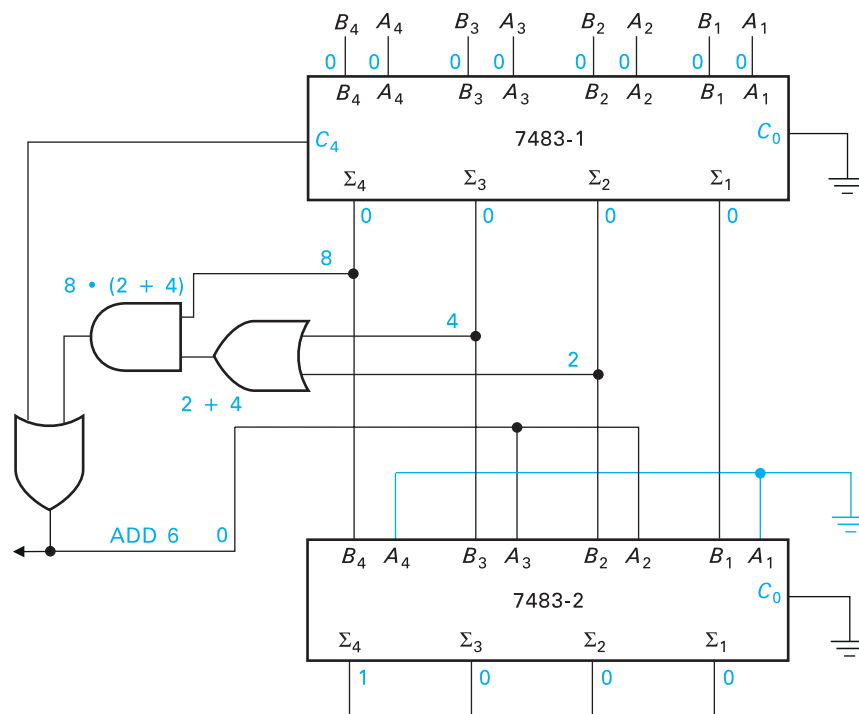


FIGURE 5-41 Troubleshooting a BCD adder

Step 3. Check the inputs to 7483-2. All inputs measure 0 except B_4 . B_4 measures 1.7 V which indicates a floating TTL input. A cold solder joint has left B_4 floating, and the 7483-2 is taking it as a 1.

Lab 5B has circuit files 5B-4.ewb, 5B-5.ewb, and 5B-6.ewb to troubleshoot.

DIGITAL APPLICATION

Floating-Point Unit (FPU)

Pentium central processing units (CPUs) have floating-point units (FPUs) built into them. The primary CPU is equipped to handle addition, subtraction, multiplication, and division of integer numbers. More complicated math problems such as logarithms, trigonometry and inverse trig functions, and floating-point math are handled by the FPU. Math-intensive computer application programs such as computer-aided design (CAD) and other graphics programs and spreadsheet and database programs that run on the Pentium use FPU instructions to handle their math needs efficiently. Intel processors prior to the 80486 used a separate math coprocessor IC to handle higher math functions. The 80386 processor worked with the 80387 math coprocessor. Moving the coprocessor onboard increased communication speed and lowered execution time.

SUMMARY

- A half adder adds two inputs and produces a sum and a carry.
- A full adder adds three inputs, two bits, and a carry-in, and produces a sum and a carry.
- A fast carry or look ahead carry is produced without waiting for the result to “ripple through” each stage of addition.
- To subtract using a complement method, add the complement of the subtrahend to the minuend.
- The 1’s complement of a binary number is formed by changing each bit.
- To produce a 1’s complement adder/subtractor from a full adder IC:
 1. Exclusive-ORs are used to complement the subtrahend for subtraction.
 2. An AND gate produces an EAC when overflow is produced during subtraction.
 3. An inverter and gate detect no overflow on subtraction.
 4. When no overflow on subtraction is detected, exclusive-ORs take the 1’s complement to produce true magnitude outputs.

- The 2's complement can be formed by two methods:
 1. Change every bit and add 1.
 2. Starting at the LSB and moving to the left, leave each bit unchanged until the first 1 is passed. Thereafter change each bit.
- To produce a 2's complement adder/subtractor from a full adder IC:
 1. Exclusive-ORs are used to take the 1's complement of the subtrahend for subtraction, and a 1 is input to C_0 to produce the 2's complement.
 2. An inverter and gate detect no overflow on subtraction.
 3. When no overflow is detected, the 2's complement must be taken to produce the true magnitude output. Exclusive-ORs take the 1's complement of the preliminary sum, and a second full adder is used to add 1 to produce the 2's complement.
- In a signed 2's complement system, the MSB indicates the sign of the number. 0 indicates positive, and 1 indicates negative.
- In a signed 2's complement system, negative numbers are written in 2's complement form.
- In a signed 2's complement system, the carry-in to the sign bit and the overflow are monitored to determine whether or not the result is correct.
- In binary-coded decimal (BCD), each decimal digit is represented with four bits according to the weighted 8, 4, 2, 1 system.
- In the BCD number system there are six forbidden numbers: 1010, 1011, 1100, 1101, 1110, 1111.
- If one of the six forbidden numbers appears as a result of a BCD addition, six must be added to leap over the forbidden states.
- If a carry out is generated as the result of a BCD addition, six must be added to compensate for the six forbidden numbers.
- To produce a BCD adder from a full adder IC:
 1. An OR gate and an AND gate watch for an 8 AND a 4 OR 2, which indicates that a forbidden number has been produced as a preliminary sum.
 2. The forbidden number OR a carry out on C_4 of the adder indicates that six must be added to the original sum.
 3. A second full adder is used to add six when one of the above conditions is detected.
- To program, a PLD a source file with an extension of .PLD is written. The .PLD file is compiled to create a .JED file. The .JED file is sent to the IC programmer hardware to actually program the PLD.

- To troubleshoot an adder circuit follow these steps:
 1. Establish the inputs.
 2. Determine whether or not the result is correct.
 3. Confirm that power supply voltages, ground connections, and inputs are correct.
 4. Test voltages at a midpoint to divide the circuit and determine which part contains the fault.
 5. If the output of a device does not measure correctly, determine whether it is faulty or if it is loaded down by a following device.

QUESTIONS AND PROBLEMS

1. Draw the truth table and logic diagram of a half adder. Show IC numbers and pin numbers.
2. Draw the truth table and logic diagram of a full adder. Show IC numbers and pin numbers.
3. Subtract using 1's complement.

a. $\begin{array}{r} 1010_2 \\ -1000_2 \\ \hline \end{array}$	b. $\begin{array}{r} 10001_2 \\ -11101_2 \\ \hline \end{array}$
---	---
4. Subtract using 1's complement

a. $\begin{array}{r} 1101_2 \\ -100_2 \\ \hline \end{array}$	b. $\begin{array}{r} 1001_2 \\ -1100_2 \\ \hline \end{array}$
--	---
5. Work the following problems. Use the 1's complement method on the subtraction problems. Confirm Figure 5-23 by following these problems through the circuit.

a. $\begin{array}{r} 0111 \\ +1000 \\ \hline \end{array}$	b. $\begin{array}{r} 1010 \\ -0111 \\ \hline \end{array}$	c. $\begin{array}{r} 0011 \\ -1000 \\ \hline \end{array}$
---	---	---
6. Subtract using 2's complement.

a. $\begin{array}{r} 11010_2 \\ -1100_2 \\ \hline \end{array}$	b. $\begin{array}{r} 10010_2 \\ -11110_2 \\ \hline \end{array}$
--	---
7. Subtract using 2's complement

a. $\begin{array}{r} 100101_2 \\ -1001_2 \\ \hline \end{array}$	b. $\begin{array}{r} 10101_2 \\ -11000_2 \\ \hline \end{array}$
---	---

8. Work the following problems. Use the 1's complement method on the subtraction problems. Confirm Figure 5-23 by following these problems through the circuit.
- | | | |
|---|---|---|
| a. $\begin{array}{r} 0101 \\ +1011 \\ \hline \end{array}$ | b. $\begin{array}{r} 1000 \\ +0110 \\ \hline \end{array}$ | c. $\begin{array}{r} 0100 \\ -1100 \\ \hline \end{array}$ |
|---|---|---|
9. Work the following problems. Use the 2's complement method on the subtraction problems. Confirm Figure 5-30 by following these problems through the circuit.
- | | | |
|---|---|---|
| a. $\begin{array}{r} 0101 \\ +1000 \\ \hline \end{array}$ | b. $\begin{array}{r} 1001 \\ -0111 \\ \hline \end{array}$ | c. $\begin{array}{r} 0011 \\ -1000 \\ \hline \end{array}$ |
|---|---|---|
10. Work the following problems. Use the 2's complement method on the subtraction problems. Confirm Figure 5-30 by following these problems through the circuit.
- | | | |
|---|---|--|
| a. $\begin{array}{r} 0110 \\ +1000 \\ \hline \end{array}$ | b. $\begin{array}{r} 1011 \\ - 101 \\ \hline \end{array}$ | c. $\begin{array}{r} 101 \\ -1010 \\ \hline \end{array}$ |
|---|---|--|
11. Work the following BCD problems. Confirm Figure 5-35 by following these problems through the circuit.
- | | | |
|---|---|---|
| a. $\begin{array}{r} 0100 \\ +0101 \\ \hline \end{array}$ | b. $\begin{array}{r} 1001 \\ +0110 \\ \hline \end{array}$ | c. $\begin{array}{r} 1001 \\ +0111 \\ \hline \end{array}$ |
|---|---|---|
12. Work the following BCD problems. Confirm Figure 5-35 by following these problems through the circuit.
- | | | |
|--|---|---|
| a. $\begin{array}{r} 101 \\ +1001 \\ \hline \end{array}$ | b. $\begin{array}{r} 1001 \\ +1000 \\ \hline \end{array}$ | c. $\begin{array}{r} 0101 \\ +0010 \\ \hline \end{array}$ |
|--|---|---|
13. Draw the logic diagram of an 8-bit 1's complement adder/subtractor.
14. Draw the logic diagram of an 8-bit 2's complement adder/subtractor.
15. Draw the logic diagram of an 8-bit BCD adder.
16. Find a CMOS adder IC in a data book. Draw the logic diagram. Describe in your own words its function.
17. Work the following problems. Use the 1's complement method. Confirm your design by following the problem through the circuit that you designed in number 13.
- | | | |
|---|---|---|
| a. $\begin{array}{r} 01101100 \\ +00111010 \\ \hline \end{array}$ | b. $\begin{array}{r} 10101011 \\ -01001100 \\ \hline \end{array}$ | c. $\begin{array}{r} 00011100 \\ -10110101 \\ \hline \end{array}$ |
|---|---|---|
18. Work the following problems. Use the 1's complement method. Confirm your design by following the problem through the circuit that you designed in number 13.
- | | | |
|---|---|---|
| a. $\begin{array}{r} 10000001 \\ +00111010 \\ \hline \end{array}$ | b. $\begin{array}{r} 10001111 \\ -10000111 \\ \hline \end{array}$ | c. $\begin{array}{r} 00111111 \\ -01000110 \\ \hline \end{array}$ |
|---|---|---|

19. Work the following problems. Use the 2's complement method on the subtraction problems. Confirm your design by following the problem through the circuit that you designed in number 14.

a. $\begin{array}{r} 01010001 \\ +01111010 \\ \hline \end{array}$	b. $\begin{array}{r} 11010011 \\ -00101101 \\ \hline \end{array}$	c. $\begin{array}{r} 11000000 \\ -11000001 \\ \hline \end{array}$
---	---	---

20. Work the following problems. Use the 2's complement method on the subtraction problems. Confirm your design by following the problem through the circuit that you designed in number 14.

a. $\begin{array}{r} 10011000 \\ +01100110 \\ \hline \end{array}$	b. $\begin{array}{r} 01111101 \\ -00110010 \\ \hline \end{array}$	c. $\begin{array}{r} 01101100 \\ -10010010 \\ \hline \end{array}$
---	---	---

21. Work the following problems. Confirm your design by following the problem through the circuit that you designed in number 15. All the numbers are BCD.

a. $\begin{array}{r} 10000010 \\ +00000111 \\ \hline \end{array}$	b. $\begin{array}{r} 00101000 \\ +01001001 \\ \hline \end{array}$	c. $\begin{array}{r} 10010101 \\ +01010001 \\ \hline \end{array}$
---	---	---

22. Work the following problems. Confirm your design by following the problem through the circuit you designed in problem 15. All the numbers are BCD.

a. $\begin{array}{r} 01110100 \\ +00111001 \\ \hline \end{array}$	b. $\begin{array}{r} 10000110 \\ +01111000 \\ \hline \end{array}$	c. $\begin{array}{r} 01010001 \\ +00110100 \\ \hline \end{array}$
---	---	---

These questions refer to the 1's complement subtractor circuit in Figure 5-23.

23. How is the end-around carry accomplished?
24. Explain the function of exclusive-OR gates 1, 2, 3, and 4.
25. Explain the function of exclusive-OR gates 5, 6, 7, and 8.
26. What is the function of AND gate 2?

These questions refer to the 2's complement adder/subtractor in Figure 5-30.

27. What is the function of exclusive-OR gates 5, 6, 7, and 8?
28. When is the C_0 input on 7483-2 a 1 level?
29. What are the four pins that are grounded on 7483-2?
30. Why are they grounded?
31. When is the output of the AND gate a 1?

These questions refer to the BCD adder in Figure 5-35.

32. What is the function of the 7483-2?
33. Why does the C_4 output of 7483-1 need to be included in the development of the ADD 6 signal?

34. What would happen if C_0 on the 7483-2 were not grounded?
35. Draw the IEC symbol for a 4-bit full adder.
36. Express the following decimal numbers in 8-bit signed 2's complement form.
- | | | |
|--------|--------|---------|
| a. -38 | c. -12 | e. -100 |
| b. +57 | d. +12 | f. +60 |
37. Express the following decimal numbers in 8-bit signed 2's complement form.
- | | | |
|--------|-------|---------|
| a. -50 | c. -2 | e. -120 |
| b. +43 | d. +8 | f. +83 |
38. Add these 8-bit signed 2's complement numbers. Use the carry from column 7 and the overflow to tell whether the answer is correct.
- | | |
|------------------------|------------------------|
| a. 00011110 + 00111000 | c. 11100011 + 10000001 |
| b. 01011101 + 00111100 | d. 00110011 + 11001100 |
39. Add these 8-bit signed 2's complement numbers. Use the carry from column 7 and the overflow to tell whether the answer is correct.
- | | |
|------------------------|------------------------|
| a. 00111101 + 11010110 | c. 10011100 + 10011011 |
| b. 01100111 + 11001001 | d. 01111111 + 01111111 |
40. Use the 2's complement method to subtract these 8-bit signed 2's complement numbers. Use the carry from 7 and the overflow to indicate whether the answer is correct.
- | | |
|------------------------|------------------------|
| a. 00101010 - 01101101 | c. 10001111 - 10100000 |
| b. 01111111 - 10000000 | d. 10000000 - 10000000 |

LAB
5A

Adders

OBJECTIVES

After completing this lab, you should be able to:

- Draw the logic diagram for a BCD adder.
- Construct and use a BCD adder.
- Draw the logic diagram for a 1's complement adder/subtractor.
- Construct and use a 1's complement adder/subtractor.

COMPONENTS NEEDED

- 1 7408 IC
- 1 7432 IC
- 2 7483 ICs
- 2 7486 ICs
- 5 LEDs
- 5 330- Ω resistors

PREPARATION

Use a BCD adder as follows:

1. Put 0000 and 0011 into the BCD adder input. Check to see if the output of the first 7438 IC is 0011. If it is, check the second 7483 IC for the sum of 0011.
2. If the adder works with sums of 9 or less but does not work with sums of 10 or more, then the ADD 6 part of the adder is not functioning.
3. If the sum of the two inputs 0000 and 0011 is 1001, then the ADD 6 part of the BCD adder is turned on when it should be off.

Troubleshoot the rest of Lab 5A if it is needed. It is a good idea to write the steps you use to troubleshoot a circuit in a notebook for reference during the troubleshooting procedure and for use at a later time.

Review the lab safety rules under SAFETY ADVICE in the PREPARATION section of Lab 1A, Chapter 1.

PROCEDURE

1. Draw the logic diagram of a BCD adder. Show pin numbers. Use two 7483s and additional gates as needed. Use LEDs to monitor the outputs. Have your instructor approve your drawing.
2. Construct the circuit. Let $A = 0101$ and $B = 0011$. What is the sum? Is there a carry-out of the first 7483?
3. Let $A = 0101$ and $B = 1001$. What is the sum? Is there a carry-out of the first 7483?
4. What is the purpose of the ADD 6 signal?
5. Add these combinations:

a. $\begin{array}{r} 0110 \\ +0110 \\ \hline \end{array}$	b. $\begin{array}{r} 1001 \\ +1001 \\ \hline \end{array}$	c. $\begin{array}{r} 1001 \\ +0001 \\ \hline \end{array}$
---	---	---
6. Draw the logic diagram of a 1's complement adder/subtractor. Use one 7483 and additional gates as needed. Use LEDs to monitor the outputs. Have your instructor approve your drawing.
7. Construct the circuit. Add $A = 0101$ and $B = 0011$. What is the sum? Is there a carry-out? Does the first set of exclusive-OR gates complement B ? Does the last set invert the answer? Is an EAC performed?
8. Let $A = 0101$ and $B = 1001$. What is the difference ($A - B$)? Is the result of the addition complemented by the second set of exclusive-OR gates? Is B complemented by the first set of exclusive-OR gates? Is an EAC performed?
9. Let $A = 1001$ and $B = 0101$. What is the difference ($A - B$)? Is B complemented by the first set of exclusive-OR gates? Is the result of the addition complemented by the second set of exclusive-OR gates? Is an EAC performed?
10. Try these combinations:

a. $\begin{array}{r} 0110 \\ -0110 \\ \hline \end{array}$	b. $\begin{array}{r} 1001 \\ -1010 \\ \hline \end{array}$	c. $\begin{array}{r} 1001 \\ -0001 \\ \hline \end{array}$
---	---	---

If your circuit does not work properly, consider these points:

1. Check all power supply voltages and connections.
2. Check all input and output voltages for proper voltages.
3. Troubleshoot the circuit using the following basic steps. These steps can be used to troubleshoot any electronic circuit.
 - a. First understand the electronic circuit's operation theory.
 - b. Determine or set the input value to the circuits.

- c. Measure the circuit's output value.
- d. Based on the input values, output values, and your understanding of how the circuits work, determine which part of the circuit is faulty. Then test the part you suspect.

LAB
5B

Adder Circuits

OBJECTIVES

After completing this lab, you should be able to use Electronics Workbench to:

- Troubleshoot a BCD adder circuit.
- Troubleshoot a 1's complement circuit.
- Troubleshoot a 2's complement circuit.

PREPARATION

In each part of this lab, you are asked to troubleshoot one of the three adder/subtractor circuits covered in this chapter. In each circuit the word generator is used to supply inputs to the circuits. The first three addresses of the generator contain test problems. If you cannot determine the fault by studying the test problems, create some tests of your own. Use the lamp indicators, volt indicators, or voltmeters to check critical voltages. Keep a log of your procedure and write a paragraph discussing your conclusion.

PROCEDURE

Part 1

In Electronics Workbench open circuit file 5B-1.ewb and answer these questions.

1. What is the function of this circuit?
2. Which bits of the word generator are supplying the B inputs? A inputs?
3. The first three lines of the word generator contain test problems for the circuit. What are the three test problems?
4. Is the final result correct in each case?
5. Is the preliminary sum (output of the first 7483) correct in each case?

6. In each situation, should six be added to correct the preliminary result?
7. If so, is six being added correctly?
8. From your answers to questions 4 through 7, if you have detected a problem, what is the most probable fault? Write a paragraph explaining your decision.

Part 2

In Electronics Workbench open circuit file 5B-2.ewb and answer these questions.

1. What is the function of this circuit?
2. Which bits of the word generator are supplying the B inputs? A inputs? Add/subtract control signal?
3. The first three lines of the word generator contain test problems for the circuit. What are the three test problems?
4. Is the final result correct in each case?
5. Is the preliminary sum (output of the first 7483) correct in each case?
6. In each situation, should an end-around carry be added into the preliminary sum? If so, is it being added correctly?
7. In each situation, should the preliminary sum be complemented? If so, is it being complemented correctly?
8. From your answers to questions 3 through 7, if you have detected a problem, what is the most probable fault? Write a paragraph explaining your decision.

Part 3

In Electronics Workbench open circuit file 5B-3.ewb and answer these questions.

1. What is the function of this circuit?
2. Which bits of the word generator are supplying the B inputs? A inputs? Add/subtract control signal?
3. The first three lines of the word generator contain test problems for the circuit. What are the three test problems?
4. Is the final result correct in each case?
5. Is the preliminary sum (output of the first 7483) correct in each case?
6. In each situation, should the preliminary sum be complemented? If so, is it being complemented correctly?
7. From your answers to questions 3 through 7, if you have detected a problem, what is the most probable fault? Write a paragraph explaining your decision.

Part 4

Troubleshoot circuit files 5B-4.ewb, 5B-5.ewb, and 5B-6.ewb. Keep a log of your discoveries and conclusions.

