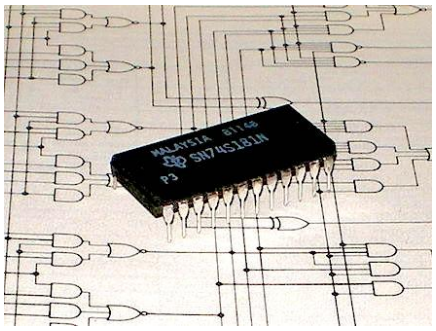

The chip shown in the photograph is the first complete arithmetic logic unit on a single chip. It was used as the arithmetic/logic core in the CPUs of many historically significant minicomputers. In this unit, you will learn about the basic circuits used for adding and subtracting binary numbers.

Source:
https://en.wikipedia.org/wiki/File:SN74S181N.JPG
Transferred from en.wikipedia to Commons
by audriusa

UNIT 9

APPLICATIONS OF LOGIC CIRCUITS

Structure

STUDY GUIDE

In this unit, you will learn about the applications of logic circuits in binary addition and subtraction. These are the basic operations in computing systems and digital technology. You have learnt binary arithmetic in Unit 6. So, you should be familiar with both operations. We suggest that you revise Sec. 6.6 of Unit 6. In this unit, you will learn about half adder and full adder circuits that carry out these operations. You will also learn about 2's complements used for binary subtraction. To study this unit well, you should know the concepts of Units 6, 7 and 8 very well. You will need to practice drawing the logic circuits for various applications, and explaining their operation. Solve all SAQs and Terminal Questions on your own to learn the applications of logic circuits well.

"We cannot solve our problems with the same thinking we used when we created them."

Albert Einstein

9.1 INTRODUCTION

In Units 7 and 8, you have learnt about logic circuits and how to simplify them using Boolean algebra. Several logic gates can be combined to design a digital circuit for a desired application. This application involving several logic gates may be a simple or complex one. In this unit, we consider a few simple applications of logic circuits in binary addition and subtraction.

You have learnt the basic rules for adding and subtracting binary numbers in Unit 6. We will explain another method of subtracting binary numbers, which can be used to create simpler digital circuits containing logic gates to carry out these operations.

In Secs. 9.2 and 9.3, you will learn how binary addition and subtraction are carried out using the circuits of half adder and full adder. In Sec. 9.4, we explain the binary adder used to add 4-bit binary numbers. Finally, In Sec. 9.5, you will learn about the 4-bit binary adder-subtractor that uses the 2's complements of binary numbers for subtraction. In the next block you will learn about analog circuits such as amplifiers, oscillators and power supplies that form the backbone of any electronic system.

Expected Learning Outcomes

After studying this unit, you should be able to:

- draw the circuits of half adder and full adder and describe their operation for binary addition;
- draw the circuit of a 4-bit binary adder and describe its operation for binary addition;
- calculate the 1's complement and 2's complement of binary numbers; and
- draw the circuit of a 4-bit binary adder-subtractor and describe its operation for binary addition and subtraction.

9.2 BINARY ADDITION: HALF ADDER

In Sec. 6.6.1 of Unit 6, you have learnt the method of adding binary numbers. Let us quickly revise the method to carry out addition of one-bit binary numbers so that you may understand the design of a half adder. It is a logic circuit that carries out addition of one-bit binary numbers.

Let us write the rules for binary addition of single digit numbers that you have learnt in Sec. 6.6.1:

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 1$$

Using these rules, we can write the binary addition of two one-bit binary numbers as follows:

0	0	1	1
+ 0	+ 1	+ 0	+ 1
00	01	01	10

In this example of binary addition, the bit on the right hand side is the sum while the bit on the left hand side is the carry. This can be realized through a logic circuit known as half adder. We can put this in the form of a truth table as shown in Table 9.1.

Table 9.1: Truth table for half adder

A	*B*	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

This application has two outputs, one for 'sum' and another for 'carry'. Therefore, we have to obtain two Boolean expressions for the two outputs. Can you identify the operation that leads to the output given in the Carry column? It is the AND operation. Therefore, the Boolean expression for carry is:

$$\text{Carry} = AB$$

Which operation will give the ouput for Sum in Table 9.1? Check the following Boolean expression out:

$$\text{Sum} = \bar{A}B + A\bar{B} = A \oplus B$$

You may also like to recall the truth table given in Sec. 7.3.2 of Unit 7. Is the output not the same as the Sum in Table 9.1? So, the Sum is the output of the XOR gate. The two gates, AND and XOR are connected as shown in Fig. 9.1 to give the two outputs. This circuit is known as **half adder**.

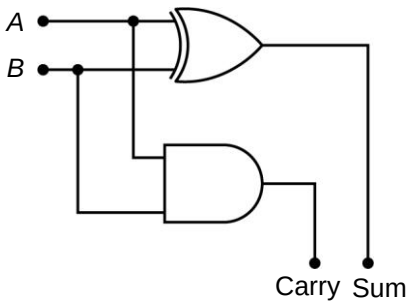

Fig. 9.1: **Circuit for half adder.**

The symbol of half adder is shown in Fig. 9.2.

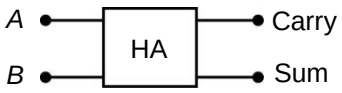

Fig. 9.2: **Symbol of half adder.**

Recall that an OR gate gives 1 + 1 = 1, which is Boolean addition. It does not give the ouput as 1 + 1 = 10, which is binary addition. So, we could not use the OR gate for this binary addition operation. This problem is taken care of by designing the half adder circuit using XOR gate.

However, using a half adder, we can perform binary addition of only one-bit binary numbers and add 2 bits at a time. What do we do when we need to perform the operation of binary addition on three bits at a time? We use the full adder circuit about which you will now learn.

9.3 FULL ADDER

The full adder can add three single-bit binary numbers. The binary addition of three single-bit binary numbers is done as follows:

0	0	0	0	1	1	1	1
+ 0	+ 0	+ 1	+ 1	+ 0	+ 0	+ 1	+ 1
+ 0	+ 1	+ 0	+ 1	+ 0	+ 1	+ 0	+ 1
00	01	01	10	01	10	10	11

The right hand bits in each of these additions represent the sum and the left hand bits represent the carry. These eight possible combinations of three single-bit binary numbers can be presented in the form of a truth table given in Table 9.2.

Table 9.2: Truth table for full adder

A	*B*	*C*	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

In order to design the logic circuit for a full adder, we write the Boolean expressions for both the sum and carry in Table 9.2 and simplify those in MSP form. We use the sum of product (SOP) method and write:

$$\text{Sum} = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$= \bar{A}(\bar{B}C + B\bar{C}) + A(\bar{B}\bar{C} + BC)$$

$$= \bar{A}(B \oplus C) + A(\overline{B \oplus C}) \quad \text{(read the margin remark)}$$

$$= \bar{A}X + A\bar{X} \quad \text{where } X = B \oplus C$$

$$= A \oplus X$$

$$= A \oplus B \oplus C$$

You can verify that the truth tables of $\overline{B \oplus C}$ and $\bar{B}\bar{C} + BC$ are the same. Also, using de Morgan's theorems, you can see, in general, that

$$\overline{X \oplus Y} = \overline{\bar{X}Y + X\bar{Y}}$$

$$= \overline{\bar{X}Y}\,.\,\overline{X\bar{Y}}$$

$$= (\bar{\bar{X}} + \bar{Y}).(\bar{X} + \bar{\bar{Y}})$$

$$= (X + \bar{Y})(\bar{X} + Y)$$

$$= X\bar{X} + XY + \bar{X}\bar{Y} + Y\bar{Y}$$

$$= \bar{X}\bar{Y} + XY$$

This is the MSP expression for the sum in Table 9.2. This is the output of a 3-input XOR gate. Similarly, we obtain the MSP form of the carry in Table 9.2:

$$\text{Carry} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$= BC(A + \bar{A}) + A\bar{B}C + AB\bar{C}$$

$$= BC + A\bar{B}C + AB\bar{C}$$

$$= C(B + A\bar{B}) + AB\bar{C}$$

$$= C(B + A) + AB\bar{C}$$

$$= BC + A(C + B\bar{C})$$

$$= BC + A(C + B) = BC + AC + AB$$

This is the MSP expression for the carry in Table 9.2. From these two MSP expressions, we obtain the following logic circuit, which is the logic circuit for a full adder (see Fig. 9.3).

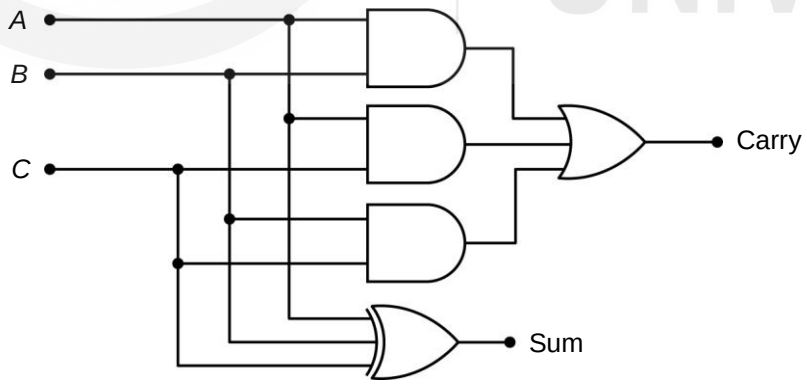

Fig. 9.3: **Circuit for full adder.**

The symbol of the full adder is shown in Fig. 9.4.

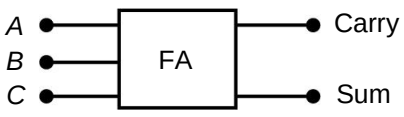

Fig. 9.4: **Symbol of full adder.**

So far, you have learnt about logic circuits that can add two one-bit binary numbers (half adder) and three one-bit binary numbers (full adder). The next question we ask is: How do we add larger binary numbers containing more than one-bit? For this, logic circuits called binary adders are devised. We describe the circuit of a binary adder in the next section.

9.4 BINARY ADDER

Actually, a binary adder is nothing but a cascade of full adder circuits to add larger binary numbers. Suppose we have to add two 4-bit binary numbers:

$A_3 A_2 A_1 A_0$ and $B_3 B_2 B_1 B_0$

We start adding the numbers from the right-most bit. So, we first add A_0 and B_0 using a half adder. Then since there might be a carry leading to addition of three one-bit numbers, we add three full adders.

Thus, for the addition of two 4-bit binary numbers, we require one half adder and three full adders in the circuit. Let us see how the addition is carried out by this cascade of adders. Study the expression given below:

$$\begin{array}{r} A_3\,A_2\,A_1\,A_0 \\ B_3\,B_2\,B_1\,B_0 \\ \hline C_4 S_3\,S_2\,S_1 S_0 \end{array}$$

When we add the first two bits A_0 and B_0, we get the outputs S_0 (sum) and C_1 (carry). The carry C_1 becomes an input of the first full adder.

So, the first full adder adds three 1-bit binary numbers: C_1, A_1 and B_1. Its outputs are S_1 (sum) and C_2 (carry), which becomes an input of the second full adder, and so on.

Thus, each full adder has three inputs (C_n, A_n and B_n). The outputs of the third full adder are: S_3 (sum) and C_4 (carry) Thus, the circuit of the binary adder is as shown in Fig. 9.5.

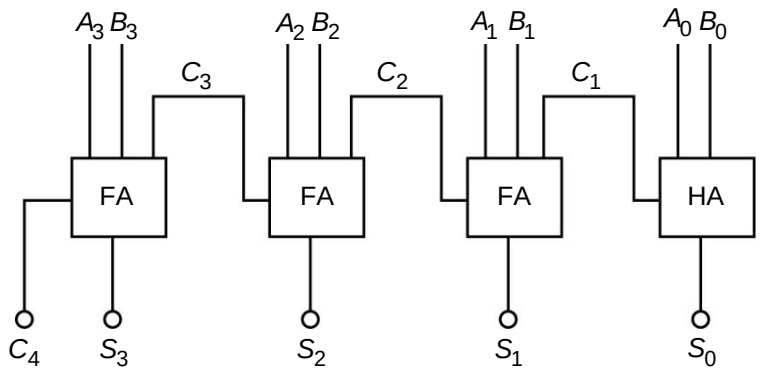

Fig. 9.5: **A 4-bit binary adder.**

Let us consider an example to illustrate addition of 4-bit binary numbers using a binary adder.

EXAMPLE 9.1 : ADDITION OF 4-BIT BINARY NUMBERS

Add the following 4-bit binary numbers:

1011

and 1101

SOLUTION ■ We will use the binary adder along with Table 9.2 as follows:

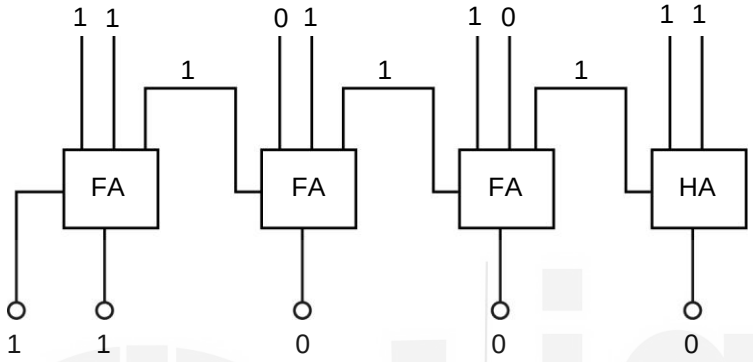

Fig. 9.6: **Adding 11 and 13 using a 4-bit binary adder.**

So, we get the sum as: 11000

This is 24 in decimal system, the sum of 11 (1011) and 13 (1101) obtained using the binary adder.

You may like to check whether you have understood the discussion so far. Try SAQ 1.

SAQ 1 – Logic circuit for addition of binary numbers

Draw the digital circuit for a 2-bit binary adder. How is it different from a full adder?

Next we consider the question: How do we subtract one binary number from another? For this we use the binary adder-subtractor. Let us explain what it is.

9.5 BINARY ADDER-SUBTRACTOR

To use the circuit of a binary adder-subtractor, you first need to learn how to represent a negative number. Let us explain how this is done.

9.5.1 Signed Binary Numbers

In the binary number system, the plus (+) sign is represented by the digit 0 and the minus (–) sign by the digit 1. These digits are prefixed to the binary number. So, the positive decimal number is written as a binary number with the prefix 0. The numbers – 1, – 2, – 3 are written as 1001, 1010, 1101, where the first digit from the left (1) is for the minus sign. For larger signed decimal

numbers, we require more bits. So, we can express the decimal numbers as 16-bit binary numbers with their signs prefixed as shown in Table 9.3 below:

Table 9.3: Writing signed decimal numbers as 16-bit binary numbers

Positive Decimal number	Positive Binary number	Negative Decimal number	Negative Binary number
+ 1	0000 0000 0000 0001	– 1	1000 0000 0000 0001
+ 2	0000 0000 0000 0010	– 2	1000 0000 0000 0010
+ 3	0000 0000 0000 0011	– 3	1000 0000 0000 0011
+ 4	0000 0000 0000 0100	– 4	1000 0000 0000 0100
+ 5	0000 0000 0000 0101	– 5	1000 0000 0000 0101
+ 6	0000 0000 0000 0110	– 6	1000 0000 0000 0110
+ 7	0000 0000 0000 0111	– 7	1000 0000 0000 0111
+ 8	0000 0000 0000 1000	– 8	1000 0000 0000 1000
+ 9	0000 0000 0000 1001	– 9	1000 0000 0000 1001

Note that in Table 9.3, the first bit from the left represents the sign of the number: 0 for + and 1 for –. In the same way, we can express larger signed numbers. For example,

+ 15 as 0000 0000 0000 1111

– 15 as 1000 0000 0000 1111

+ 16 as 0000 0000 0001 0000

– 16 as 1000 0000 0001 0000

+ 24 as 0000 0000 0001 1000

– 24 as 1000 0000 0001 1000

+ 49 as 0000 0000 0011 0001

– 49 as 1000 0000 0011 0001

and so on. REMEMBER that the leading bit (the first bit from the left) represents the sign and the remaining bits, the magnitude of the number. But the logic circuits for adding and subtracting signed binary numbers became too complex in terms of both hardware and software (programming).

So, 2's complements were used to simplify binary addition and subtraction. You will learn about it now.

9.5.2 2's Complement

Recall that the NOT gate or the inverter logic circuit produces 0 output for input 1 and vice-versa. When every bit in a binary string is inverted, it is called the **1's complement** of that string.

So, consider the string:

$A = 0101$

The 1's complement of A is:

$\overline{A} = 1010$

Similarly, we **define the 2's complement** as follows:

The 2's complement of a string, say, A is the binary string obtained by adding 1 to the 1's complement of A. It is denoted by A' and given by:

$$A' = \overline{A} + 1 \qquad (9.1)$$

For example, the 2's complement of the string $A = 0101$ is given as:

$A' = 1010 + 1 = 1011$

If we take the 2's complement twice, we get the original string back. For example, the 2's complement of $A' = 1011$ above is:

$A'' = \overline{A'} + 1 = 0100 + 1 = 0101$

which is the original string A = 0101. Therefore, we get:

$$A'' = A \qquad (9.2)$$

Eq. (9.2) is read as: "The double complement of A is equal to A." You can verify this statement for 1's complement as well.

You may be wondering: What is the advantage of using 2's complement of a string?

The advantage is that due to Eq. (9.2), we can use 2's complements to represent negative numbers for use in computers.

But before we explain how to do so, one word about the terminology. In Sec. 9.5.1, you have learnt how to represent negative decimal numbers in an equivalent binary form. You know that in binary numbers, we use the leading bit or the most significant bit (the leftmost bit) as the sign bit: 0 for positive number and 1 for negative number. This representation of a signed binary number is commonly referred to as the **sign-magnitude notation**. However, when we deal with arithmetic operations in computers, the 2's complements of numbers are more convenient to use for negative numbers.

So, **2's complement gives us an alternative way of representing negative numbers** that is easy to use in computers.

You must remember the concept of the complement of a negative number in any number system. Consider the decimal system. When we subtract, say, 2 from 10, we get 8. Hence, 8 is the complement of -2, [i.e., $8-(-2)=10$]. Similarly, when you consider the octal system, (3-bit string), then complements can be obtained by subtracting from 8, e.g. $8_{10}-5_{10}=3_{10}$ Hence, 011 (3_{10}) is the complement of (5_{10}). A '1' bit is added in the left of MSB to indicate that it is a negative number.

The advantage in using 2's complement is that it lets us subtract numbers by using addition. Since positive numbers always start with 0, their 2's complement will start with 1 (read the margin remark too).

Table 9.4 gives a few negative numbers in sign-magnitude notation as well as in 2's complement notation. Note that in this table, we have used the 3-bit equivalent of a binary number as we are using the leading bit for the sign of the number. So, you can see that the 2's complement representation gives us a different string of bits for the negative numbers than the sign-magnitude representation. You need to remember the following about the 2's complement representation of numbers:

1. The leading bit or the MSB represents the sign: 0 for plus and 1 for minus.
2. The positive decimal numbers are in the sign-magnitude representation.
3. The negative decimal numbers are in the 2's complement representation.

Table 9.4: Sign-magnitude and 2's complement representation of decimal numbers

Decimal numbers	Sign-magnitude representation	2's complement representation
+ 7	0111	0111
+ 6	0110	0110
+ 5	0101	0101
+ 4	0100	0100
+ 3	0011	0011
+ 2	0010	0010
+ 1	0001	0001
– 1	1001	1111
– 2	1010	1110
– 3	1011	1101
– 4	1100	1100
– 5	1101	1011
– 6	1110	1010
– 7	1111	1001

For example, 2's complement of + 2 is:

1101 + 1 = 1110

Note that it is – 2 in Table 9.4. The same is true for all negative numbers given in Table 9.4. You should verify this yourself.

So, **ALWAYS REMEMBER: The 2's complement of a binary number corresponding to a positive decimal number is equivalent to the negative of that decimal number**. **Taking the 2's complement of a binary number is the same as changing the sign of the equivalent decimal number.** You should check this out for Table 9.4 (read the margin remark).

You may like to practice obtaining the 2's complements of binary numbers. Solve SAQ 2.

SAQ 2 – 2's complement of binary numbers

Obtain the 2's complements of the following binary numbers:

a) 0000 1111, b) 0011 1100, c) 0000 1110 and d) 0001 0001.

With this preliminary introduction to the 2's complement representation, you are now ready to learn about the binary adder-subtractor.

9.5.3 2's Complement Binary Adder-Subtractor

The arithmetic operations in computers are carried out by using the binary adder-subtractor based on 2's complement representation of negative decimal numbers. This is because the logic circuit becomes much simpler with this representation. Fig. 9.7 shows the circuit that can add and subtract 4-bit binary numbers in the 2's complement representation. Let us explain its working first.

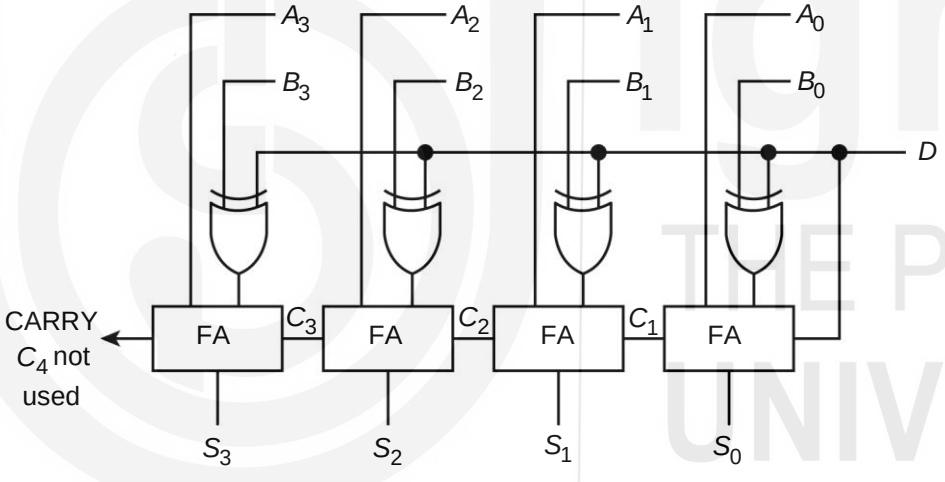

Fig. 9.7: **A 2's complement binary adder-subtractor circuit.**

Refer to Fig. 9.7. When the entity marked *D* in the figure is low or 0, the circuit works as an adder and when it is high or 1, it works as a subtractor. Let us explain how it works.

Truth table for XOR gate

D	*B*	*Y*
0	0	0
1	0	1
0	1	1
1	1	0

Binary Addition

Note from Fig.9.7 that *D* and B_0 are the inputs of a XOR gate. Recall the truth table of the XOR gate (Table 7.7 of Unit 7, given in the margin here). When the input *D* is low, you can see from Table 7.7 and Fig. 9.7 that the input B_0 is the output of the XOR gate. This is true for all other XOR gates. So, when *D* is low or 0, the *B* bits are transmitted by the gates without inversion. Hence, the output of each full adder is a sum:

$$S_0 = A_0 + B_0,\ S_1 = A_1 + B_1,\ S_2 = A_2 + B_2\ \ S_3 = A_3 + B_3 \qquad (9.3)$$

which we write as the equation:

$$\mathbf{S} = \mathbf{A} + \mathbf{B} \qquad (9.4)$$

Eq. (9.4) is the same as Eq. (9.3). So, when *D* is low or 0, the output of the binary adder-subtractor shown in Fig. 9.7 is the string: $S_3 S_2 S_1 S_0$

Note that the carry of each full adder is an input to the next full adder, and each sum is the output of a full adder. The final CARRY is not used in the answer here as it represents the signed bit whereas $S_3 S_2 S_1 S_0$ are the numeral bits.

As an example, consider the inputs to the binary adder-subtractor:

A = 0001 1001

B = 0000 1111

When *D* is zero, these numbers are added by the binary adder-subtractor as explained above. So, the result is:

$$\begin{array}{rr} & 0001\ 1001 \\ \underline{+} & \underline{0000\ 1111} \\ A + B = & 0010\ 1000 \end{array}$$

Note that this is the sum of decimal numbers 25 and 15, equal to 40.

Binary Subtraction

The 2's complement binary adder-subtractor works as a subtractor when *D* is high or 1 in Fig. 9.7. Let us explain how.

Truth table for XOR gate

D	*B*	*Y*
0	0	0
1	0	1
0	1	1
1	1	0

Note that *D* and B_0 are the inputs of a XOR gate. So, from the truth table of the XOR gate (given in the margin here), you can see that when *D* is 1, the output of the XOR gate is the 1's complement of B_0, that is $\overline{B_0}$. This is true for all other XOR gates. So, when *D* is high or 1, the *B* bits are transmitted by the gates after inversion as their 1's complements.

Now *D* is also an input to the full adder along with the *A* bits. This means that 1 is added to the 1's complement of $\overline{B_0}$, which gives us the 2's complement of B_0, that is, B_0'. So, the XOR gate output is $\overline{B_0}$, and adding 1 results in B_0'. So, the output of each full adder is:

$$S_0 = A_0 + B_0',\ S_1 = A_1 + B_1',\ S_2 = A_2 + B_2',\ S_3 = A_3 + B_3' \qquad (9.5)$$

which is equivalent to:

$$S_0 = A_0 - B_0,\ S_1 = A_1 - B_1,\ S_2 = A_2 - B_2,\ S_3 = A_3 - B_3 \qquad (9.6)$$

because the 2's complement is equivalent to a change in sign as explained in Sec. 9.5.2.

We write Eq. (9.6) as the compact equation:

$$\mathbf{S} = \mathbf{A} - \mathbf{B} \qquad (9.7)$$

So, when *D* is high or 1, the output of the binary adder-subtractor shown in Fig. 9.7 yields the subtracted number as the string: $S_3 S_2 S_1 S_0$

Note again that the carry of each full adder is an input to the next full adder, and each sum of Eq. (9.5) is the output of a full adder. Also the final CARRY is

not used in the answer here as it represents the signed bit whereas $S_3 S_2 S_1 S_0$ are the numeral bits.

As an example, let us consider the same inputs to the binary adder-subtractor:

A = 0001 1001

B = 0000 1111

when D is 1. Now, the binary adder-subtractor subtracts B from A by adding the 2's complements of B as follows:

1's complement of B is 1111 0000 and adding 1 to it, we get the 2's complement of B: 1111 0001.

Therefore, the output of the binary adder-subtractor is:

$$\begin{array}{rr} & 0001\ 1001 \\ + & 1111\ 0001 \\ \hline A - B = & 0000\ 1010 \end{array}$$

Note that this is the subtraction of decimal numbers 15 from 25, which yields 10:

$$\begin{array}{rr} & 25 \\ - & 15 \\ \hline & 10 \end{array}$$

We now end the discussion in this unit in which you have learnt the simple applications of logic circuits in adding and subtracting binary numbers. Such circuits are used in the arithmetic operations in a computer. Let us now summarise the contents of the unit.

9.6 SUMMARY

Concept	Description
Half adder	■ A **half adder** is a logic circuit comprising an AND gate and a XOR gate, which is used for the binary addition of two 1-bit binary numbers.
Full adder	■ A **full adder** is a logic circuit used for the binary addition of three 1-bit binary numbers. It comprises three AND gates, one XOR gate and one OR gate.
Binary adder	■ A **binary adder** is a cascade of full adder circuits used to add larger binary numbers. The circuit used for binary addition of 4-bit binary numbers comprises one half adder and three full adders.
2's Complement binary adder-subtractor	■ The **binary adder-subtractor** uses sign-magnitude representation of numbers for binary addition and the 2's complement representation of numbers for binary subtraction. • In the **sign-magnitude representation** of binary numbers, the leading bit or the most significant bit (MSB) (the first bit from the left) represents the sign and the remaining bits, the magnitude of the number: The plus (+) sign is represented by the digit 0 and the minus (–) sign by the digit 1.

- The **2's complement** of a binary number *A* is obtained by adding 1 to the 1's complement of *A*. It is given by:

$$A' = \overline{A} + 1$$

where the 1's complement of *A* is obtained by inversion of each bit of *A*.

The 2's complement of a binary number is equivalent to the negative of the corresponding decimal number. **Taking the 2's complement of a binary number is the same as changing the sign of the equivalent decimal number**.

- The **2's complement binary adder-subtractor** circuit for 4-bit binary numbers comprises four XOR gates and four full adders. When the input to it is 0, it operates as an adder. When the input to it is 1, it operates as a subtractor.

9.7 TERMINAL QUESTIONS

1. State the differences between a half adder and a full adder.
2. What is the difference in the logic circuits of a 4-bit binary adder and a 2's complement 4-bit binary adder-subtractor?
3. Obtain the 1's complement and 2's complement of the following binary numbers:

   0010, 00100010, 00101011, 00011011, 0000110111001011
4. Draw the binary adder circuit for adding the 4-bit binary numbers corresponding to the decimal numbers 8 and 10.
5. Draw the 2's complement binary adder-subtractor circuit for the subtraction of the binary equivalent of 8 from the binary equivalent of 15.

9.8 SOLUTIONS AND ANSWERS

Self-Assessment Questions

1. See Fig. 9.8.

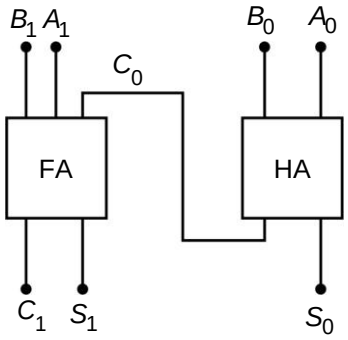

Fig. 9.8: **A 2-bit binary adder.**

A 2-bit binary adder can add 2-bit binary numbers whereas a full adder can add only 1-bit binary numbers.

2. The 2's complements of the given binary numbers are obtained by adding 1 to their 1's complements:

   a) 1111 0000 + 1 = 1111 0001

   b) 1100 0011 + 1 = 1100 0100

   c) 1111 0001 + 1 = 1111 0010

   d) 1110 1110 + 1 = 1110 1111

Terminal Questions

1. A half adder adds two 1-bit binary numbers whereas a full adder adds three 1-bit binary numbers. The logic circuit of a half adder comprises one AND and one XOR gate whereas the logic circuit of a full adder comprises three AND gates, one XOR gate and one OR gate.

2. The logic circuit of a 4-bit binary adder comprises four full adders whereas the logic circuit of a 2's complement 4-bit binary adder-subtractor comprises four XOR gates and four full adders.

3. The 1's complement and 2's complement of the binary numbers are as follows:

   0010: 1's complement : 1101; 2's complement : 1101 + 1 = 1110

   0010 0010: 1's complement : 1101 1101;

   2's complement : 1101 1101 + 1= 1101 1110

   0010 1011: 1's complement : 1101 0100;

   2's complement : 1101 0100 + 1= 1101 0101

   00011011: 1's complement : 1110 0100;

   2's complement : 1110 0100 + 1= 1110 0101

   0000 1101 1100 1011: 1's complement : 1111 0010 0011 0100;

   2's complement : 1111 0010 0011 0100 + 1= 1111 0010 0011 0101

4. We follow Example 9.1. See Fig. 9.9. The 4-bit binary equivalents of the decimal numbers 8 and 10 are: 1000 and 1010. So, we provide the inputs to the half adder and full adders accordingly as shown in Fig. 9.9. The sum is: 10010, which is the decimal number 18.

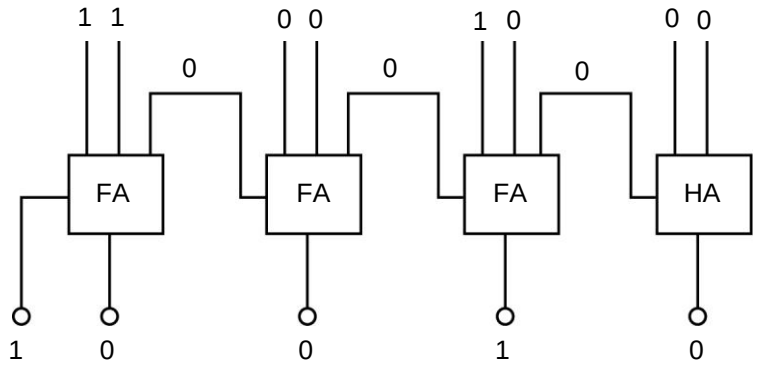

Fig. 9.9: **Adding decimal numbers 8 and 10 using a 4-bit binary adder.**

5. See Fig. 9.10. The 4-bit binary equivalents of the decimal numbers 8 and 15 are: 1000 and 1111. So, we provide the inputs to the XOR gate and full adders accordingly as shown in Fig. 9.10. For subtraction *D* is to be taken as 1.So, we get

$$\begin{array}{r} 1111 \\ -1000 \\ \hline 0111 \end{array}$$

The output is: 0111, which is the decimal number 7.

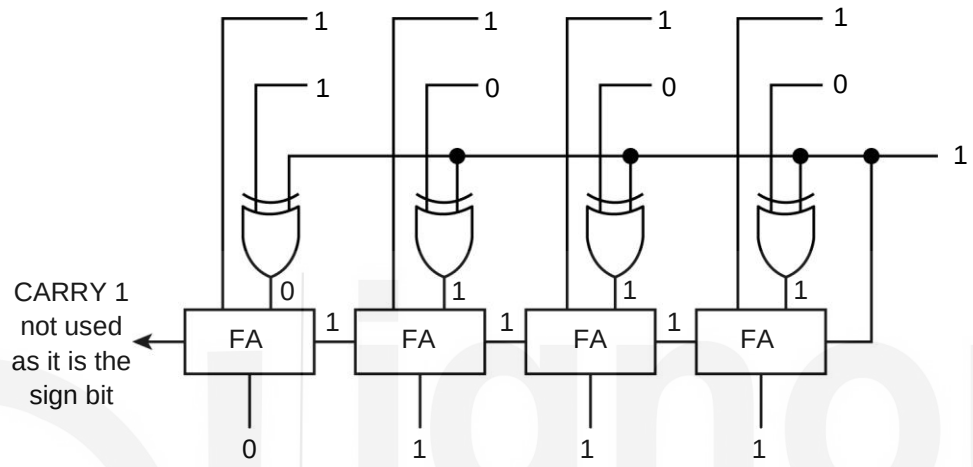

Fig. 9.10: **Subtracting 8 from 15 using a 2's complement binary adder-subtractor.**

FURTHER READINGS

1. **Electronic Principles** by Malvino A.P., Tata McGraw Hill Publishing Co. Ltd., 3rd Edition, Fifteenth Reprint (1992).
2. **Integrated Electronics: Analog and Digital Circuits and Systems** by Millman, J. and Halkias,C.C., McGraw Hill Book Co., International Edition, 29th Printing (1986).
3. **Digital Principles and Applications** by Leach D.P., Malvino A.P., Saha G., Tata McGraw Hill Education Pvt. Ltd., 7th Edition (Special Indian Edition, 2011)

TABLE OF PHYSICAL CONSTANTS

Symbol	Quantity	Value
c	Speed of light in vacuum	3.00×10^{8} ms^{-1}
μ_0	Permeability of free space	1.26×10^{-6} NA^{-2}
ε_0	Permittivity of free space	8.85×10^{-12} C^2 N^{-1}m^{-2}
$1/4\pi\varepsilon_0$		8.99×10^{9} Nm^2 C^{-2}
e	Charge of the proton	1.60×10^{-19} C
$-e$	Charge of the electron	-1.60×10^{-19} C
h	Planck's constant	6.63×10^{-34} Js
$\hbar$	$h / 2\pi$	1.05×10^{-34} Js
m_e	Electron rest mass	9.11×10^{-31} kg
$-e/m_e$	Electron charge to mass ratio	-1.76×10^{11} Ckg^{-1}
m_p	Proton rest mass	1.67×10^{-27} kg (1 amu)
m_n	Neutron rest mass	1.68×10^{-27} kg
a_0	Bohr radius	5.29×10^{-11} m
N_A	Avogadro constant	6.02×10^{23} mol^{-1}
R	Universal gas constant	8.31 Jmol^{-1}K^{-1}
k_B	Boltzmann constant	1.38×10^{-23} J K^{-1}
G	Universal gravitational constant	6.67×10^{-11} Nm^2 kg^{-2}

LIST OF BLOCKS AND UNITS: BPHET-143

BLOCK 1: PHYSICS OF SEMICONDUCTOR DEVICES

Unit 1 Essentials of Semiconductor Physics
Unit 2 Junction Diodes
Unit 3 Transistors
Unit 4 Bipolar Junction Transistor Biasing
Unit 5 Transistor Circuit Analysis

BLOCK 2: DIGITAL CIRCUITS

Unit 6 Number Systems
Unit 7 Logic Gates
Unit 8 Logic Circuits
Unit 9 Applications of Logic Circuits

BLOCK 3: ANALOG CIRCUITS

Unit 10 Amplifiers
Unit 11 Oscillators
Unit 12 Regulated Power Supply

BLOCK 4: OPERATIONAL AMPLIFIER AND INSTRUMENTATION

Unit 13 Operational Amplifier
Unit 14 Applications of Operational Amplifier
Unit 15 Cathode Ray Oscilloscope
Unit 16 Timer Circuits

SYLLABUS: DIGITAL AND ANALOG CIRCUITS AND INSTRUMENTATION (BPHET-143) 4 Credits

Physics of Semiconductor Devices: Bonding in semiconductors, intrinsic and extrinsic semiconductors, concept of band gap and its consequences, concept of hole – drift and diffusion current, *I-V* characteristics of a semiconductor, gradient driven flow, parameters governing carrier mobility (drift velocity, saturation of drift velocity and breakdown). Formation of *p-n* junction, barrier potential, *I-V* characteristics of *p-n* junction diode, current flow mechanism under forward and reverse bias conditions (drift, diffusion and recombination), *p-n* junction diode as a rectifier, other types of diodes – zener, LED, solar cell, photodetectors (construction, working and nature of *I-V* characteristics). Double junction devices (bipolar junction transistor and field effect transistor) – construction, working mechanisms, biasing conditions (for *n-p-n* and *p-n-p* transistors), CE, CB, CC configurations of BJT, transistor biasing methods, input-output characteristics of common emitter (CE) configuration – cut-off, active and saturation regions, current gains α and β and their relationship, load line and quiescent operating point (linear and switching operations of transistor). Equivalent circuit of transistor, *h*-parameter equivalent circuit, analysis of CE amplifier using hybrid model – input and output impedances, current, voltage and power gain.

Digital Circuits: Binary, octal, hex, decimal number systems and their interconversions, codes (BCD and ASCII), Boolean arithmetic, Boolean theorems, De Morgan's theorem, logic gate (AND, OR, NOT, NAND, NOR, XOR, XNOR), symbols, truth tables, gate circuit realization using diodes and transistors. Simplification of logic circuits using Boolean algebra, fundamental products, minterms and maxterms, conversion of a truth table into an equivalent logic circuit by (1) sum of product (SOP) method and (2) Karnaugh map. Binary addition, binary subtraction (using 2's complement method), half adder, full adder and subtractor, 4-bit binary adder-subtractor.

Analog Circuits: Amplifiers: Classification of amplifiers (*A, B, AB, C*) with applications, small signal-low frequency amplifiers, power amplifiers – push-pull amplifier, negative feedback and its effect on amplifier performance, cascade amplifiers – coupling networks and gain calculation.

Oscillators: Positive feedback and oscillators, Barkhausen criterion for self-sustained oscillations, classification of oscillators RF – *L-C* oscillator (Colpitt's and Hartley oscillators, circuits, working and design), AF – *R-C* oscillator (Wein bridge and phase shift oscillators, frequency determination, circuits, working and design)

Power supplies: Rectifiers (HW, FW – bridge, centre tapped), calculation of ripple factor, rectification efficiency, filters (*R-C, L-C*), voltage regulation by potential divider, zener diode and series pass voltage regulator.

Operational Amplifier and Instrumentation: Op-Amp: Virtual ground, ideal op-amp and IC-741 characteristics (transfer, offset, bandwidth, CMRR, slew rate), common and differential mode gains, op-amp as a comparator, zero crossing detector, feedback in op-amp, inverting and non-inverting amplifier, op-amp as an adder, subtractor, differentiator, integrator.

Instrumentation: Introduction to CRO, block diagram, functions and knobs on CRO panel, applications of CRO to study waveforms, measurement of voltage, frequency and phase, (mention of digital storage oscilloscope), timer IC 555, block diagram, pin-out diagram and its application as astable and monostable multivibrator.