UNIT 4

LECTURE 5

APPLICATIONS OF BOOLEAN ALGEBRA
MINTERM AND MAXTERM
EXPANSIONS

OBJECTIVES

1. Given a word description of the desired behavior of a logic circuit, write the output of the circuit as a function of the input variables. Specify this function as an algebraic expression or by means of a truth table, as is appropriate.

2. Given a truth table, write the function (or its complement) as both a minterm expansion (standard sum of products) and a maxterm expansion (standard product of sums). Be able to use both alphabetic and decimal notation.

3. Given an algebraic expression for a function, expand it algebraically to obtain the minterm or maxterm form.

OBJECTIVES

4. Given one of the following: minterm expansion for F, minterm expansion for F, maxterm expansion for F, or maxterm expansion for F , find any of the other three forms.

5. Write the general form of the minterm and maxterm expansion of a function of n variables.

6. Explain why some functions contain don’t-care terms.

7. Explain the operation of a full adder and a full subtracter and derive logic equations for these modules. Draw a block diagram for a parallel adder or subtracter and trace signals on the block diagram.

- In this unit you will learn how to design a combinational logic circuit starting with a word description of the desired circuit behaviour.
- The first step is usually to translate the word description into a truth table or into an algebraic expression.
- Given the truth table for a boolean function, two standard algebraic forms of the function can be derived—**the standard sum of products (minterm expansion) and the standard product of sums (maxterm expansion).**
- Simplification of either of these standard forms leads directly to a realization of the circuit using AND and OR gates.

4.1 CONVERSION OF ENGLISH SENTENCES TO BOOLEAN EQUATIONS

- The three main steps in designing a single-output combinational switching circuit are

  1. Find a switching function that specifies the desired behaviour of the circuit.

  2. Find a simplified algebraic expression for the function.

  3. Realize the simplified function using available logic elements.

4.1 CONVERSION OF ENGLISH SENTENCES TO BOOLEAN EQUATIONS

- For simple problems, it may be possible to go directly from a word description of the desired behaviour of the circuit to an algebraic expression for the output function.
- In other cases, it is better to first specify the function by means of a truth table and then derive an algebraic expression from the truth table.
- Logic design problems are often stated in terms of one or more English sentences.
- The first step in designing a logic circuit is to translate these sentences into Boolean equations.
- **In order to do this**, we must break down each sentence into phrases and associate a Boolean variable with each phrase. If a phrase can have a value of true or false, then we can represent that phrase by a Boolean variable.

4.1 CONVERSION OF ENGLISH SENTENCES TO BOOLEAN EQUATIONS

- **Phrases such as** "she goes to the store" or "today is Monday" can be either true or false, but a command like "go to the store" has no truth value.
- If a sentence has several phrases, we will mark each phrase with a brace.The following sentence has three phrases:

Mary watches TV if it is Monday night and she has finished her homework.

- The "if" and "and" are not included in any phrase; they show the relationships among the phrases.

4.1 CONVERSION OF ENGLISH SENTENCES TO BOOLEAN EQUATIONS

We will define a two-valued variable to indicate the truth or falsity of each phrase:

$F = 1$ if "Mary watches TV" is true; otherwise, $F = 0$.

$A = 1$ if "it is Monday night" is true; otherwise, $A = 0$.

$B = 1$ if "she has finished her homework" is true; otherwise $B = 0$.

Because F is "true" if A and B are both "true", we can represent the sentence by $F = A . B$

4.1 CONVERSION OF ENGLISH SENTENCES TO BOOLEAN EQUATIONS

- We will use the following assignment of variables:

$$\underbrace{\text{The alarm will ring}}_{Z} \quad \text{iff} \quad \underbrace{\text{the alarm switch is on}}_{A} \quad \text{and}$$

$$\underbrace{\text{the door is not closed}}_{B'} \quad \textit{or} \quad \underbrace{\text{it is after 6 P.M.}}_{C} \quad \text{and}$$

$$\underbrace{\text{the window is not closed.}}_{D'}$$

Using this assignment of variables, the above sentence can be translated into the following Boolean equation: Z = AB’ + CD’

4.3 MINTERM AND MAXTERM EXPANSIONS

- Each of the terms in equation (4-1) is referred to as a **minterm**.
- In general, a *minterm* of *n* variables is a product of *n* literals in which each variable appears exactly once in either true or complemented form, but not both. (**A *literal* is a variable or its complement.**)

$$f = A'BC + AB'C' + AB'C + ABC' + ABC \qquad (4\text{-}1)$$

4.3 MINTERM AND MAXTERM EXPANSIONS

- Table 4-1 lists all of the minterms of the three variables A, B, and C.
- Each minterm has a value of 1 for exactly one combination of values of the variables A, B, and C.
- The minterm which corresponds to row i of the truth table is designated mi (i is usually written in decimal).

TABLE 4-1 Minterms and Maxterms for Three Variables

Row No.	A B C	Minterms	Maxterms
0	0 0 0	$A'B'C' = m_0$	$A + B + C = M_0$
1	0 0 1	$A'B'C = m_1$	$A + B + C' = M_1$
2	0 1 0	$A'BC' = m_2$	$A + B' + C = M_2$
3	0 1 1	$A'BC = m_3$	$A + B' + C' = M_3$
4	1 0 0	$AB'C' = m_4$	$A' + B + C = M_4$
5	1 0 1	$AB'C = m_5$	$A' + B + C' = M_5$
6	1 1 0	$ABC' = m_6$	$A' + B' + C = M_6$
7	1 1 1	$ABC = m_7$	$A' + B' + C' = M_7$

4.3 MINTERM AND MAXTERM EXPANSIONS

- When a function f is written as a sum of minterms as in Equation (4-1), this is referred to as a *minterm expansion or a standard sum of products*.
- Equation (4-1) can be rewritten in terms of m-notation as

$$f(A, B, C) = m_3 + m_4 + m_5 + m_6 + m_7 \quad (4\text{-}5)$$

This can be further abbreviated by listing only the decimal subscripts in the form

$$f(A, B, C) = \sum m(3, 4, 5, 6, 7) \quad (4\text{-}5a)$$

4.3 MINTERM AND MAXTERM EXPANSIONS

- Each of the sum terms (or factors) in Equation (4-3) is referred to as a **maxterm**.
- In general, a **maxterm** of n variables is a sum of n literals in which each variable appears exactly once in either true or complemented form, but not both.

$$f = (A + B + C)(A + B + C')(A + B' + C) \qquad (4\text{-}3)$$

4.3 MINTERM AND MAXTERM EXPANSIONS

- **Maxterms** are often written in abbreviated form using **M-notation**.
- The maxterm which corresponds to row i of the truth table is designated M_i.

- **Note that each maxterm is the complement of the corresponding minterm, that is, $M_i = m'_i$.**

4.3 MINTERM AND MAXTERM EXPANSIONS

- **Equation (4-3) can be rewritten in M-notation as**

$$f(A, B, C) = M_0 M_1 M_2 \quad (4\text{-}6)$$

This can be further abbreviated by listing only the decimal subscripts in the form

$$f(A, B, C) = \Pi\, M(0, 1, 2) \quad (4\text{-}6a)$$

where Π means a product.

4.3 MINTERM AND MAXTERM EXPANSIONS

Given the minterm or maxterm expansions for f, the minterm or maxterm expansions for the complement of f are easy to obtain. Because f' is 1 when f is 0, the minterm expansion for f' contains those minterms not present in f. Thus, from Equation (4-5),

$$f' = m_0 + m_1 + m_2 = \sum m(0, 1, 2) \qquad (4\text{-}7)$$

4.3 MINTERM AND MAXTERM EXPANSIONS

Similarly, the maxterm expansion for f' contains those maxterms not present in f. From Equation (4-6),

$$f' = \Pi M(3,4,5,6,7) = M_3M_4M_5M_6M_7 \qquad (4\text{-}8)$$

Because the complement of a minterm is the corresponding maxterm, Equation (4-8) can be obtained by complementing Equation (4-5):

$$f' = (m_3 + m_4 + m_5 + m_6 + m_7)' = m_3'm_4'm_5'm_6'm_7' = M_3M_4M_5M_6M_7$$

Similarly, Equation (4-7) can be obtained by complementing Equation (4-6):

$$f' = (M_0M_1M_2)' = M_0' + M_1' + M_2' = m_0 + m_1 + m_2$$

Example

Find the minterm expansion of $f(a,b,c,d) = a'(b' + d) + acd'$.

$$f = a'b' + a'd + acd'$$
$$= a'b'(c + c')(d + d') + a'd(b + b')(c + c') + acd'(b + b')$$
$$= a'b'c'd' + a'b'c'd + a'b'cd' + a'b'cd + \cancel{a'b'c'd} + \cancel{a'b'cd}$$
$$+ a'bc'd + a'bcd + abcd' + ab'cd' \qquad (4\text{-}9)$$

Duplicate terms have been crossed out, because $X + X = X$. This expression can then be converted to decimal notation:

$$\begin{array}{lcccccccc} f = & a'b'c'd' + & a'b'c'd + & a'b'cd' + & a'b'cd + & a'bc'd + & a'bcd + & abcd' + & ab'cd' \\ & 0000 & 0001 & 0010 & 0011 & 0101 & 0111 & 1110 & 1010 \end{array}$$

$$f = \Sigma\, m(0, 1, 2, 3, 5, 7, 10, 14) \qquad (4\text{-}10)$$

4.3 MINTERM AND MAXTERM EXPANSIONS

The maxterm expansion for f can then be obtained by listing the decimal integers (in the range 0 to 15) which do not correspond to minterms of f:

$$f = \Pi\, M(4, 6, 8, 9, 11, 12, 13, 15)$$

4.7 DESIGN OF BINARY ADDERS AND SUBTRACTERS

- We will design a **parallel adder that adds two 4-bit unsigned binary numbers and a carry input to give a 4-bit sum and a carry output (see figure 4-2).**
- One approach would be to construct a truth table with nine inputs and five outputs and then derive and simplify the five output equations.
- Because each equation would be a function of nine variables before simplification, this approach would very difficult, and the resulting logic circuit would be very complex.

4.7 DESIGN OF BINARY ADDERS AND SUBTRACTERS

- A better method is to design a logic module that adds two bits and a carry, and then connect four of modules together to form a 4-bit adder as shown in figure 4-3.
- Each of the modules is called a full adder.
- The carry output from the first full adder serves as the carry input to the second full adder, etc.

FIGURE 4-2
Parallel Adder for 4-Bit Binary Numbers

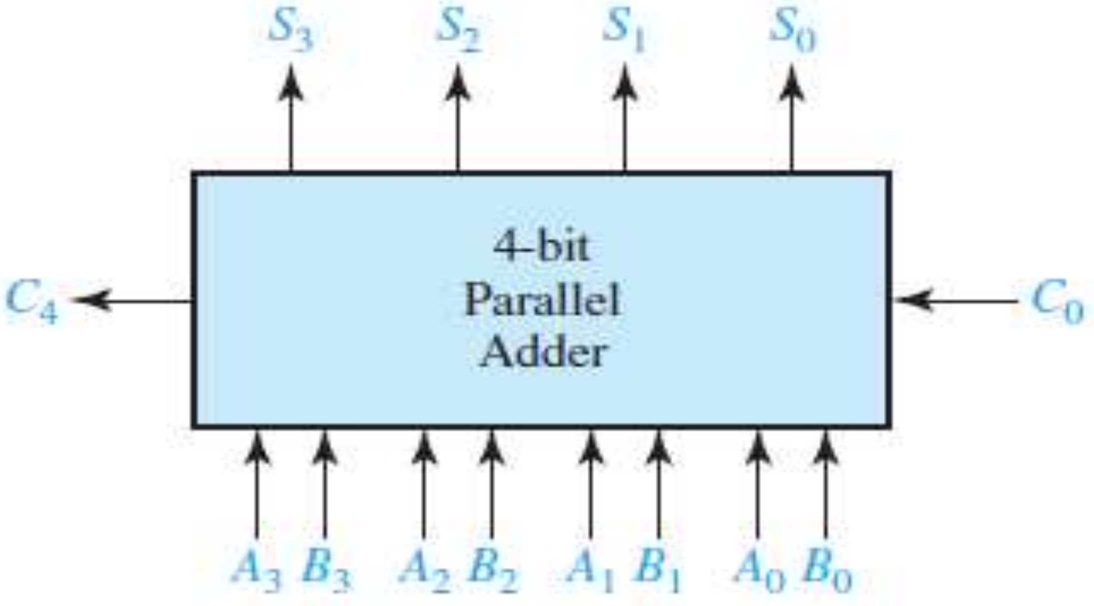
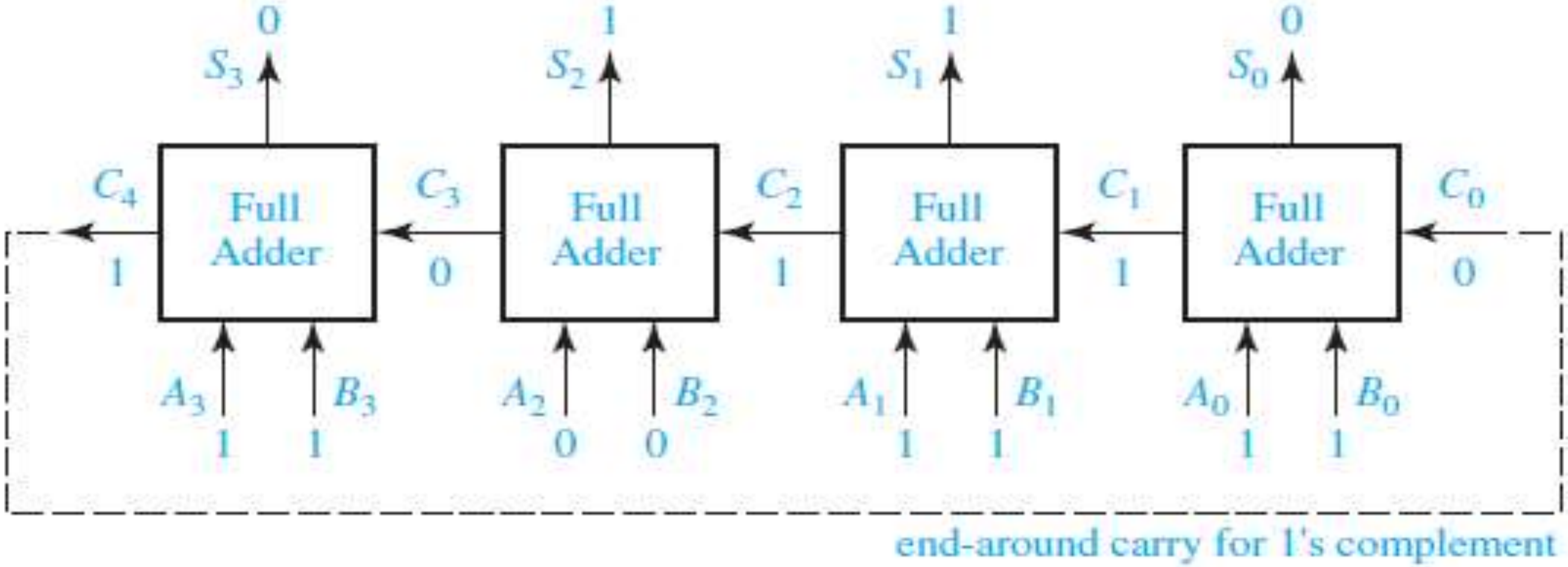


FIGURE 4-3
Parallel Adder Composed of Four Full Adders



In the example of Figure 4-3, we perform the following addition:

$$\begin{array}{rl} 10110 & \text{(carries)} \\ 1011 & \\ +\ 1011 & \\ \hline 10110 & \end{array}$$

The full adder to the far right adds $A_0 + B_0 + C_0 = 1 + 1 + 0$ to give a sum of 10_2, which gives a sum $S_0 = 0$ and a carry out of $C_1 = 1$. The next full adder adds $A_1 + B_1 + C_1 = 1 + 1 + 1 = 11_2$, which gives a sum $S_1 = 1$ and a carry $C_2 = 1$. The carry continues to propagate from right to left until the left cell produces a final carry of $C_4 = 1$.

4.7 DESIGN OF BINARY ADDERS AND SUBTRACTERS

- **NOTE**

- **If the number of bits is large, a parallel binary adder of the type shown in Figure 4-4 may be rather slow because the carry generated in the first cell might have to propagate all of the way to the last cell.**
- **Other types of adders, such as a carry-look ahead adder, 2 may be used to speed up the carry propagation.**

4.7 DESIGN OF BINARY ADDERS AND SUBTRACTERS

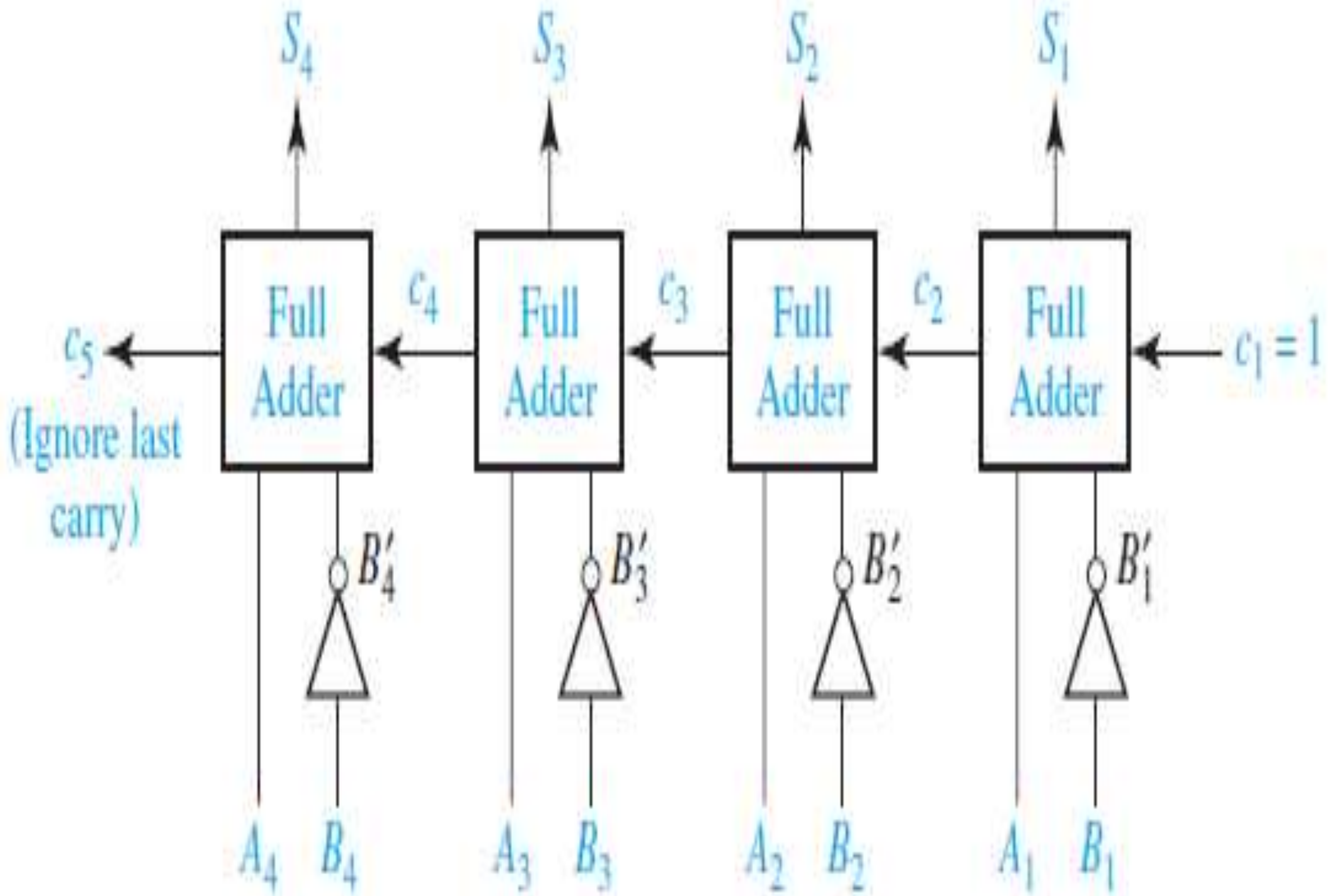
- **Subtraction of binary** numbers is most easily accomplished by adding the complement of the number to be subtracted. To compute A - B, add the complement of B to A.
- This gives the correct answer because A + (-B) = A - B.
- Either 1's or 2's complement is used depending on the type of adder employed.

4.7 DESIGN OF BINARY ADDERS AND SUBTRACTERS

- The circuit of Figure 4-6 may be used to form A - B **using the 2's complement** representation for negative numbers.
- The 2's complement of B can be formed by first finding the 1's complement and then adding 1.
- The 1's complement is formed by inverting each bit of B, and the addition of 1 is effectively accomplished by putting a 1 into the carry input of the first full adder.

using the 2's complement

FIGURE 4-6
Binary Subtracter Using Full Adders



Example

$A = 0110 \quad (+6)$

$B = 0011 \quad (+3)$

The adder output is

$$
\begin{array}{lrl}
 & 0110 & (+6) \\
 & +1100 & (1\text{'s complement of } 3) \\
 & +\quad 1 & (\text{first carry input}) \\
\hline
(1) & 0011 = 3 = 6 - 3 &
\end{array}
$$