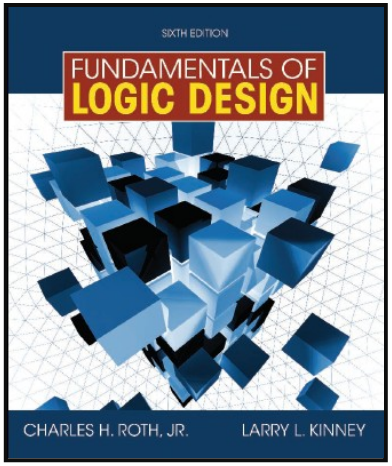
CHAPTER 4

APPLICATIONS OF BOOLEAN ALGEBRA MINTERM AND MAXTERM EXPANSIONS



This chapter in the book includes:

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 behavior of the circuit.

2. Find a simplified algebraic expression for the function.

3. Realize the simplified function using available logic elements.

Section 4.1 (p. 90)

Example 1

$$\underbrace{\text{Mary watches TV}}_{\mathbf{F}}\ \text{if}\ \underbrace{\text{it is Monday night}}_{\mathbf{A}}\ \text{and}\ \underbrace{\text{she has finished her homework.}}_{\mathbf{B}}$$

We will define a two-valued variable to indicate the truth of 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 $F = A \cdot B$

Section 4.1 (p. 90-91)

Example 2

The alarm will ring iff the alarm switch is turned on and the door is not closed, or it is after 6 P.M. and the window is not closed.

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

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

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

Section 4.1 (p. 91)

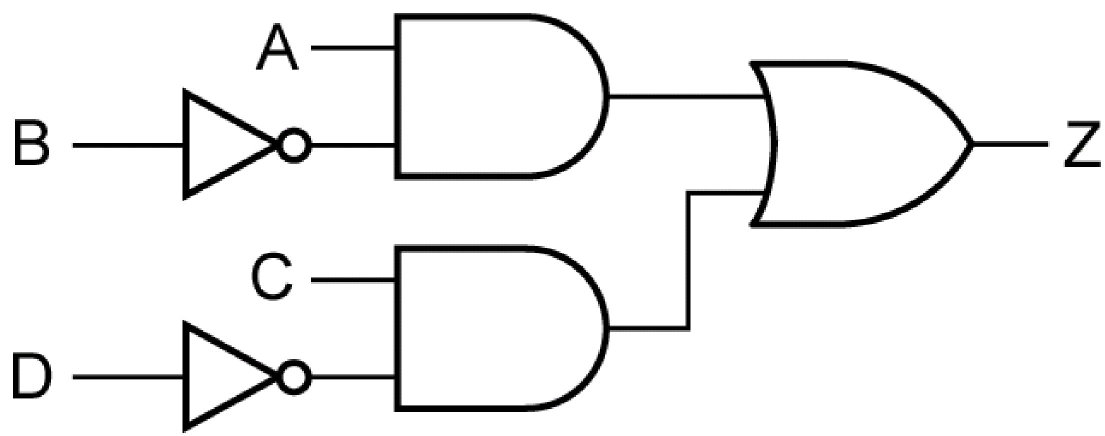
Example 2 (continued)

The alarm will ring iff the alarm switch is turned on and the door is not closed, or it is after 6 P.M. and the window is not closed.

The corresponding equation is:

$$Z = AB' + CD'$$

And the corresponding circuit is:



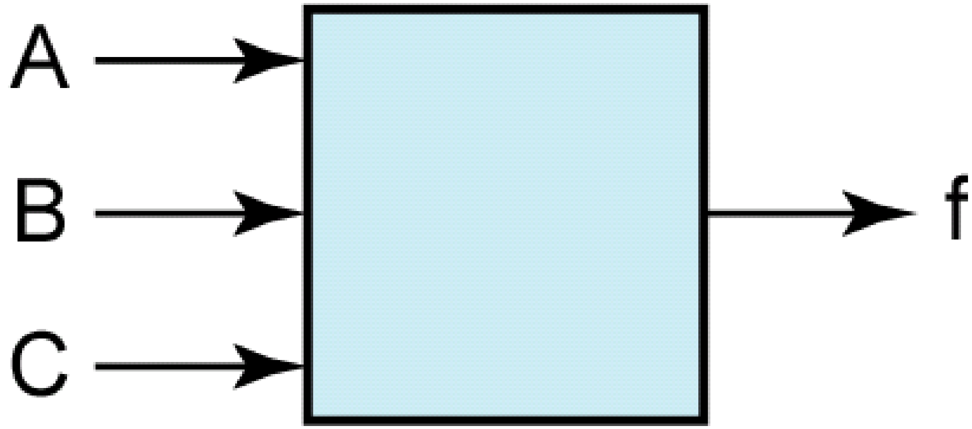
Section 4.1 (p. 91)

Combinational Logic Design using a Truth Table

Suppose we want the output of a circuit to be
$f = 1$ if $N \geq 011_2$ and $f = 0$ if $N < 011_2$.
Then the truth table is:

A B C	f	f'
0 0 0	0	1
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	1	0
1 0 1	1	0
1 1 0	1	0
1 1 1	1	0

(b)



(a)

Figure 4-1

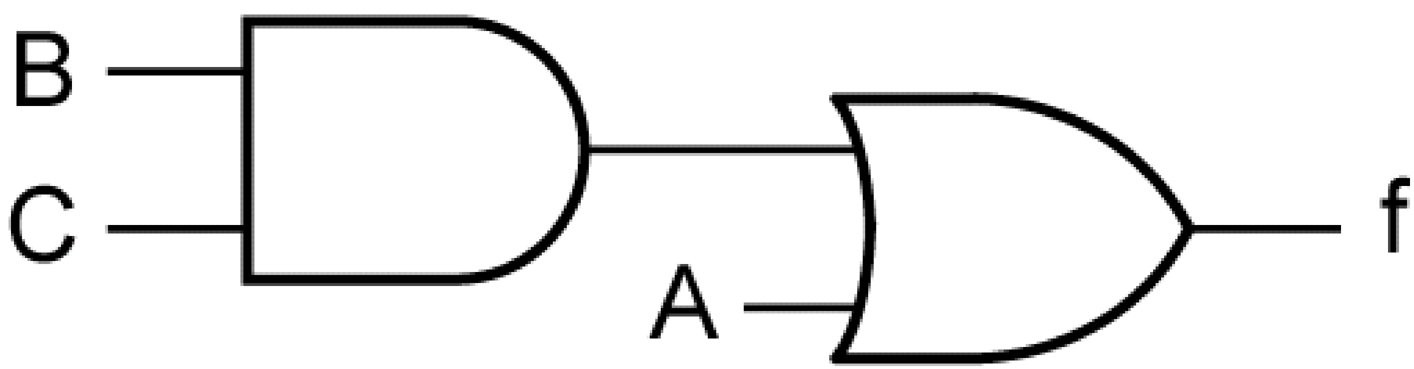
Next, we will derive an algebraic expression for *f* from the truth table by using the combinations of values of A, B, and C for which *f* = 1. For example, the term A′BC is 1 only if A = 0, B = 1, and C = 1. Finding all terms such that *f* = 1 and **OR**ing them together yields:

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

The equation can be simplified by first combining terms and then eliminating A′:

$$f = A'BC + AB' + AB = A'BC + A = A + BC \qquad (4\text{-}2)$$

This equation leads directly to the following circuit:



Instead of writing *f* in terms of the 1's of the function, we may also write *f* in terms of the 0's of the function. Observe that the term A + B + C is 0 only if A = B = C = 0. **AND**ing all of these '0' terms together yields:

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

By combining terms and using the second distributive law, we can simplify the equation:

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

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

Minterm and Maxterm Expansions

$f = A'BC + AB'C' + AB'C + ABC' + ABC$ (4-1)

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)

Section 4.3 (p. 93)

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	$ABC = m_7$	$A' + B' + C' = M_7$

Minterm expansion for a function is unique. Equation (4-1) can be rewritten in terms of m-notation as:

$f = A'BC + AB'C' + AB'C + ABC' + ABC$(4-1)

$f(A, B, C) = m_3 + m_4 + m_5 + m_6 + m_7$ (4-5)

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

$f(A, B, C) = \Sigma\, m(3, 4, 5, 6, 7)$ (4-5)

Minterm Expansion Example

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

$$f = a'b' + a'd + acd'$$

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

$$f = a'b'c'd' + a'b'c'd + a'b'cd' + a'b'cd + a'bc'd + a'bcd + abcd' + ab'cd'$$

0 0 0 0   0 0 0 1   0 0 10   0 0 11   0 10 1   0 111   1110   10 10

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

Section 4.3 (p. 95)

Maxterm Expansion Example

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

$$f = a'(b' + d) + acd'$$
$$= (a' + cd')(a + b' + d) = (a' + c)(a' + d')(a + b' + d)$$
$$= (a' + bb' + c + dd')(a' + bb' + cc' + d')(a + b' + cc' + d)$$
$$= (a' + bb' + c + d)(a' + bb' + c + d')\cancel{(a' + bb' + c + d')}$$
$$(a' + bb' + c' + d')(a + b' + cc' + d)$$
$$= \underset{1000}{(a' + b + c + d)}\underset{1100}{(a' + b' + c + d)}\underset{1001}{(a' + b + c + d')}\underset{1101}{(a' + b' + c + d')}$$
$$\underset{1011}{(a' + b + c' + d')}\underset{1111}{(a' + b' + c' + d')}\underset{0100}{(a + b' + c + d)}\underset{0110}{(a + b' + c' + d)}$$
$$= \Pi\, M(4, 6, 8, 9, 11, 12, 13, 15) \qquad (4\text{-}11)$$

Section 4.3 (p. 96)

Table 4-2. General Truth Table for Three Variables

Table 4-2 represents a truth table for a general function of three variables. Each a_i is a constant with a value of 0 or 1.

A B C	F
0 0 0	a_0
0 0 1	a_1
0 1 0	a_2
0 1 1	a_3
1 0 0	a_4
1 0 1	a_5
1 1 0	a_6
1 1 1	a_7

$$F = a_0m_0 + a_1m_1 + a_2m_2 + \cdots + a_7m_7 = \sum_{i=0}^{7} a_im_i$$

General Minterm and Maxterm Expansions

We can write the minterm expansion for a general function of three variables as follows:

$$F = a_0m_0 + a_1m_1 + a_2m_2 + \cdots + a_7m_7 = \sum_{i=0}^{7} a_im_i \qquad (4\text{-}12)$$

The maxterm expansion for a general function of three variables is:

$$F = (a_0 + M_0)(a_1 + M_1)(a_2 + M_2) \cdots (a_7 + M_7) = \prod_{i=0}^{7} (a_i + M_i) \qquad (4\text{-}13)$$

Section 4.4 (p. 97)

Table 4-3 summarizes the procedures for conversion between minterm and maxterm expansions of F and F′

Table 4-3. Conversion of Forms

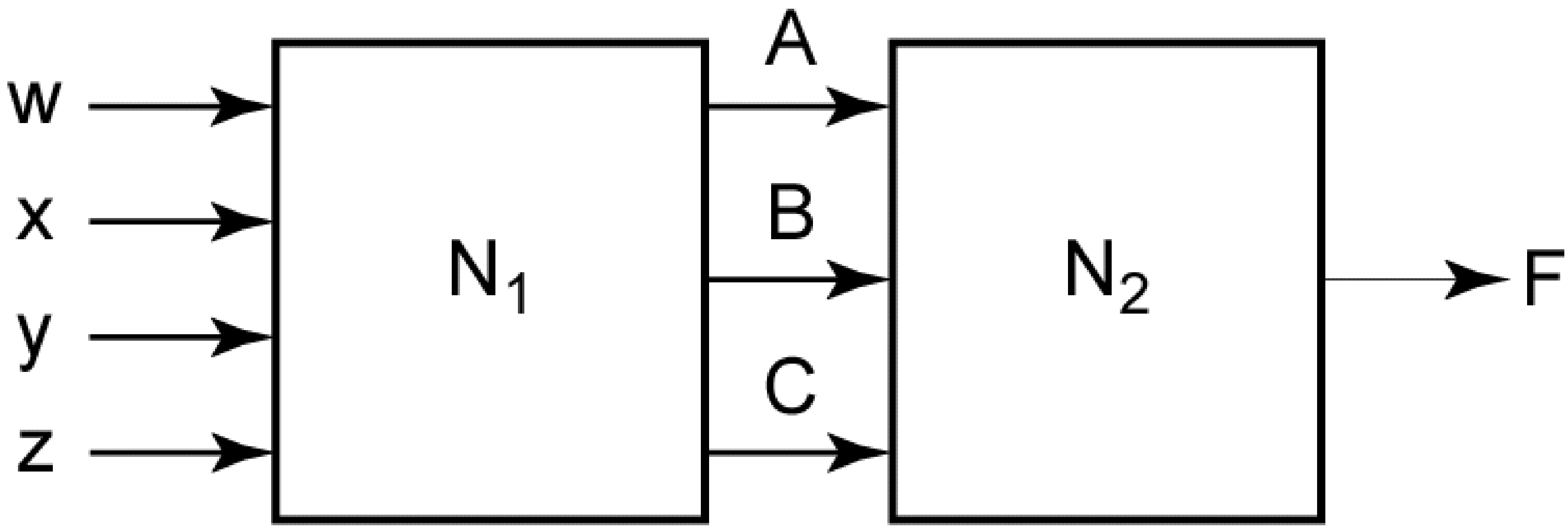
GIVEN FORM	DESIRED FORM: Minterm Expansion of *F*	DESIRED FORM: Maxterm Expansion of *F*	DESIRED FORM: Minterm Expansion of *F′*	DESIRED FORM: Maxterm Expansion of *F′*
Minterm Expansion of *F*	————	maxterm nos. are those nos. not on the minterm list for *F*	list minterms not present in *F*	maxterm nos. are the same as minterm nos. of *F*
Maxterm Expansion of *F*	minterm nos. are those nos. not on the maxterm list for *F*	————	minterm nos. are the same as maxterm nos. of *F*	list maxterms not present in *F*

Table 4-4. Application of Table 4-3

GIVEN FORM	DESIRED FORM: Minterm Expansion of f	Maxterm Expansion of f	Minterm Expansion of f'	Maxterm Expansion of f'
$f = \Sigma m(3, 4, 5, 6, 7)$	______________	$\Pi M(0, 1, 2)$	$\Sigma m(0, 1, 2)$	$\Pi M(3, 4, 5, 6, 7)$
$f = \Pi M(0, 1, 2)$	$\Sigma m(3, 4, 5, 6, 7)$	______________	$\Sigma m(0, 1, 2)$	$\Pi M(3, 4, 5, 6, 7)$

Incompletely Specified Functions

A large digital system is usually divided into many subcircuits. Consider the following example in which the output of circuit N_1 drives the input of circuit N_2:



Section 4.5 (p. 99)

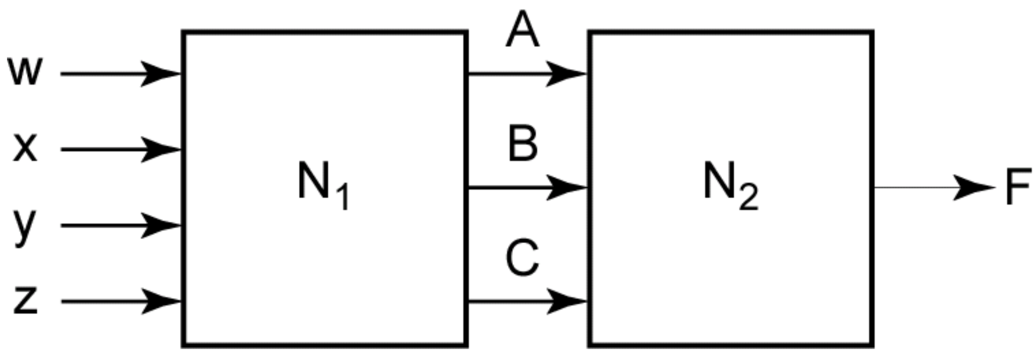


Table 4-5: Truth Table with Don't Cares

A	*B*	*C*	*F*
0	0	0	1
0	0	1	X
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	1

Let us assume the output of N_1 does not generate all possible combinations of values for *A*, *B*, and *C*. In particular, we will assume there are no combinations of values for *w, x, y,* and *z* which cause *A, B,* and *C* to assume values of 001 or 110.

Section 4.5 (p. 99)

When we realize the function, we must specify values for the don't-cares. It is desirable to choose values which will help simplify the function. If we assign the value 0 to both X's, then

$$F = A'B'C' + A'BC + ABC = A'B'C' + BC$$

If we assign 1 to the first X and 0 to the second, then

$$F = A'B'C' + A'B'C + A'BC + ABC = A'B' + BC$$

If we assign 1 to both X's, then

$$F = A'B'C' + A'B'C + A'BC + ABC' + ABC$$

$$= A'B' + BC + AB$$

The second choice of values leads to the simplest solution.

The minterm expansion for Table 4-5 is:

$$F = \Sigma\, m(0, 3, 7) + \Sigma d(1, 6)$$

The maxterm expansion for Table 4-5 is:

$$F = \Pi\, M(2, 4, 5) \bullet \Pi\, D\,(1, 6)$$

Table 4-5

A B C	F
0 0 0	1
0 0 1	X
0 1 0	0
0 1 1	1
1 0 0	0
1 0 1	0
1 1 0	X
1 1 1	1

Examples of Truth Table Construction

We will design a simple binary adder that adds two 1-bit binary numbers, *a* and *b*, to give a 2-bit sum. The numeric values for the adder inputs and outputs are as follows:

a	b	Sum	
0	0	00	$(0 + 0 = 0)$
0	1	01	$(0 + 1 = 1)$
1	0	01	$(1 + 0 = 1)$
1	1	10	$(1 + 1 = 2)$

Section 4.6 (p. 100)

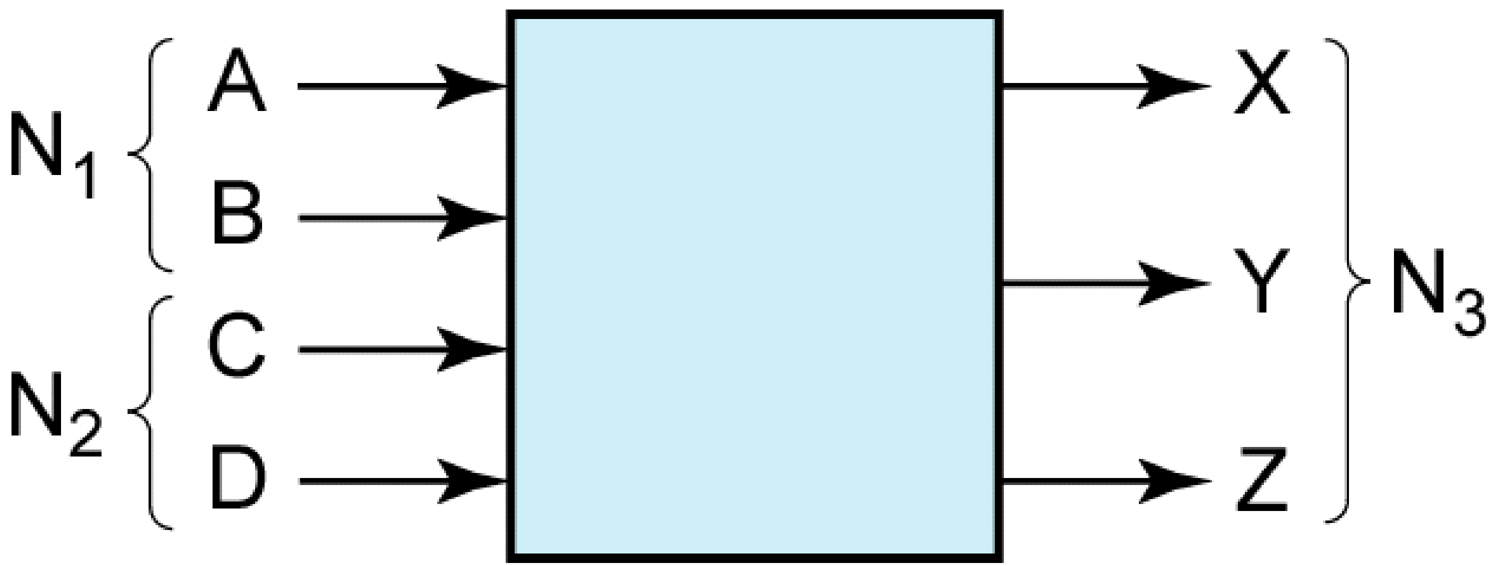
We will represent inputs to the adder by the logic variables *A* and *B* and the 2-bit sum by the logic variables *X* and *Y*, and we construct a truth table:

A	*B*	*X*	*Y*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Because a numeric value of 0 is represented by a logic 0 and a numeric value of 1 by a logic 1, the 0's and 1's in the truth table are exactly the same as in the previous table. From the truth table,

$$X = AB \text{ and } Y = A'B + AB' = A \oplus B$$

Ex: Design an adder which adds two 2-bit binary numbers to give a 3-bit binary sum. Find the truth table for the circuit. The circuit has four inputs and three outputs as shown:



TRUTH TABLE:

N_1		N_2		N_3		
A	*B*	*C*	*D*	*X*	*Y*	*Z*
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	1	0
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0

TRUTH TABLE:

N_1		N_2		N_3		
A	*B*	*C*	*D*	*X*	*Y*	*Z*
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Section 4.6 (p. 101)

Design of Binary Adders and Subtractors

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.

Section 4.7 (p. 104)

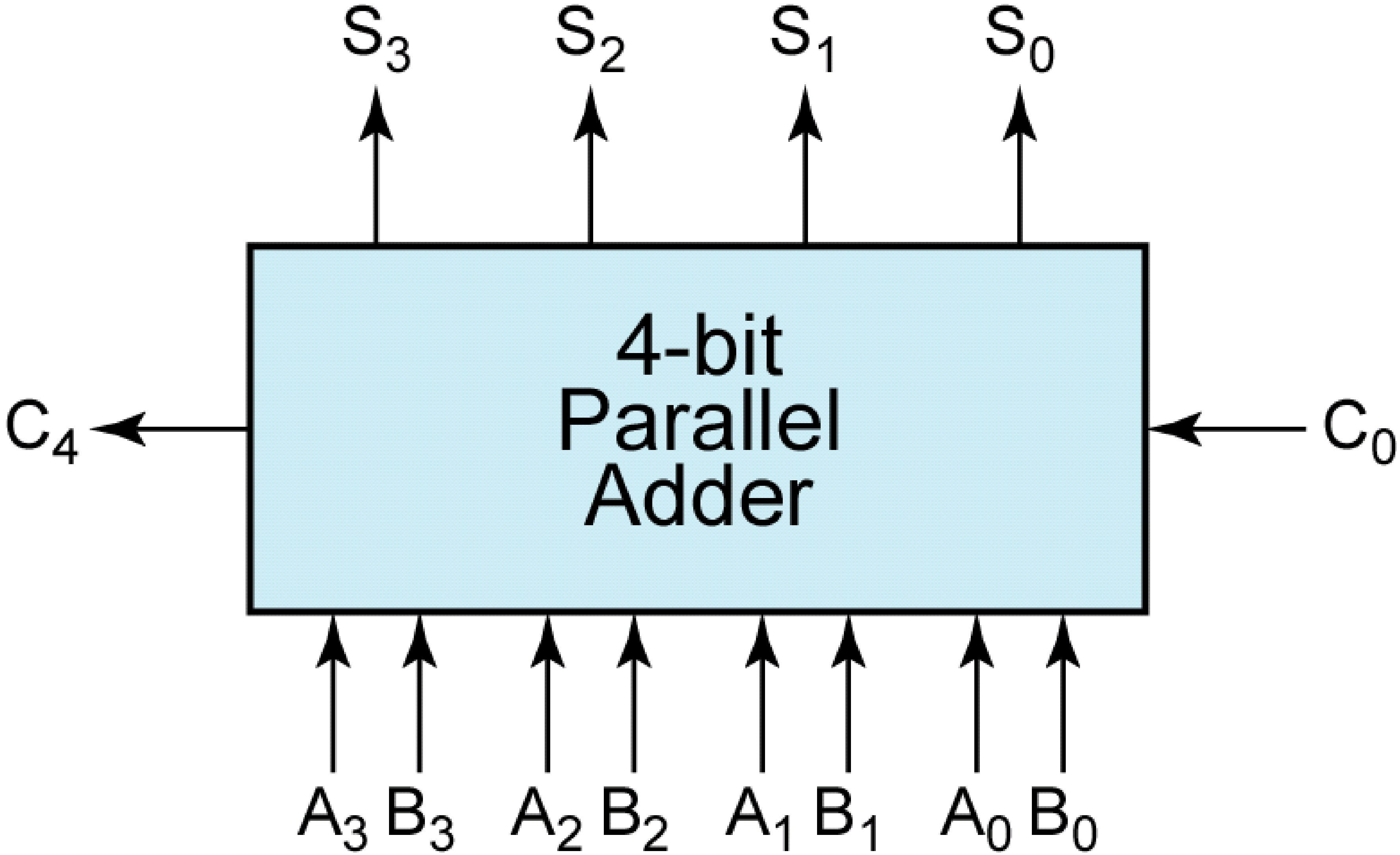


Figure 4-2: **Parallel Adder for 4-Bit Binary Numbers**

One approach would be to construct a truth table with nine inputs and five outputs and then derive and simplify the five output equations.

A better method is to design a logic module that adds two bits and a carry, and then connect four of these modules together to form a 4-bit adder.

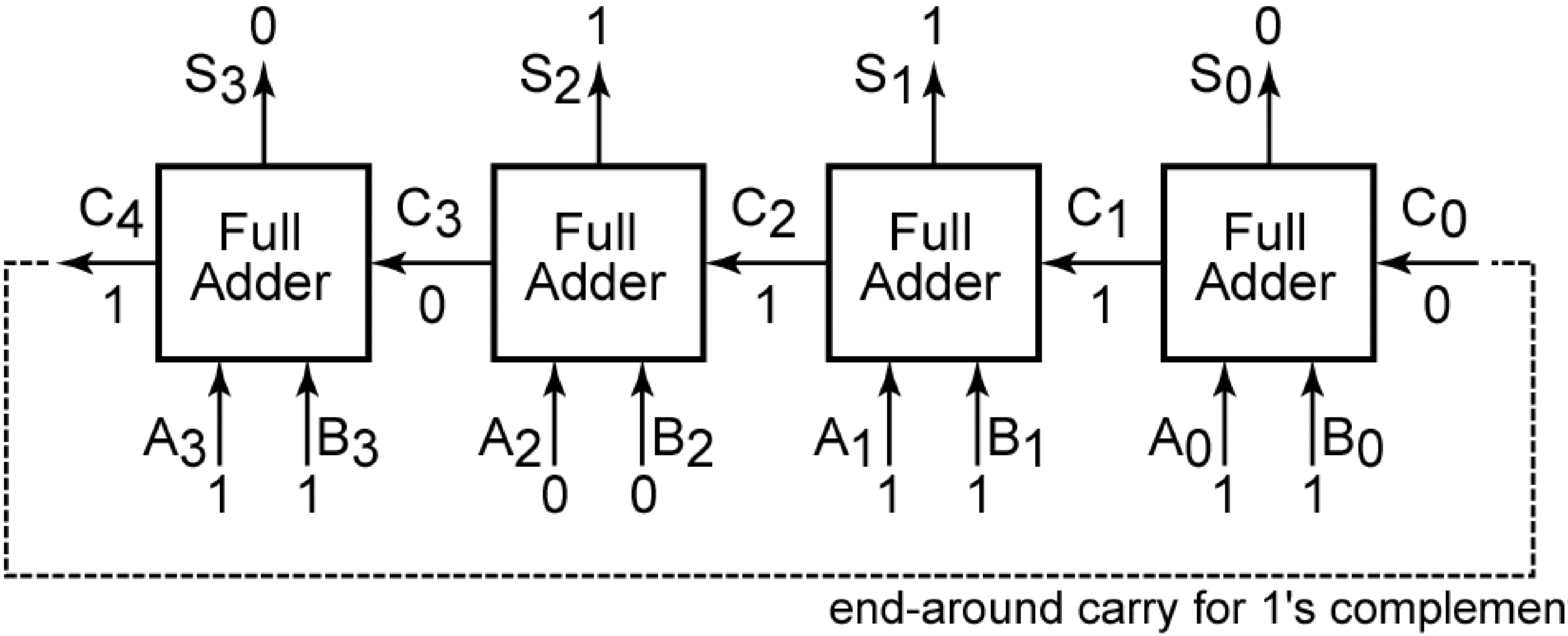
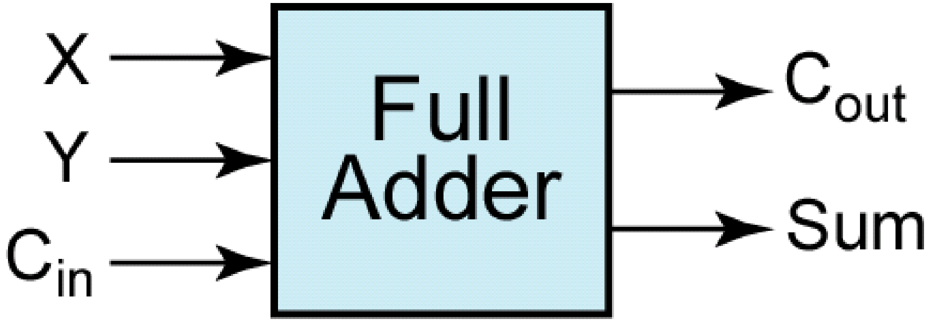


Figure 4-3: **Parallel Adder Composed of Four Full Adders**



X	Y	C_{in}	C_{out}	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

Figure 4-4: **Truth Table for a Full Adder**

Full Adder Logic Equations

The logic equations for the full adder derived from the truth table are:

$$\begin{aligned} Sum &= X'Y'C_{in} + X'YC'_{in} + XY'C'_{in} + XYC_{in} \\ &= X'(Y'C_{in} + YC'_{in}) + X(Y'C'_{in} + YC_{in}) \\ &= X'(Y \oplus C_{in}) + X(Y \oplus C_{in})' = X \oplus Y \oplus C_{in} \end{aligned} \quad (4\text{-}20)$$

$$\begin{aligned} C_{out} &= X'YC_{in} + XY'C_{in} + XYC'_{in} + XYC_{in} \\ &= (X'YC_{in} + XYC_{in}) + (XY'C_{in} + XYC_{in}) + (XYC'_{in} + XYC_{in}) \\ &= YC_{in} + XC_{in} + XY \end{aligned} \quad (4\text{-}21)$$

Section 4.7 (p. 105)

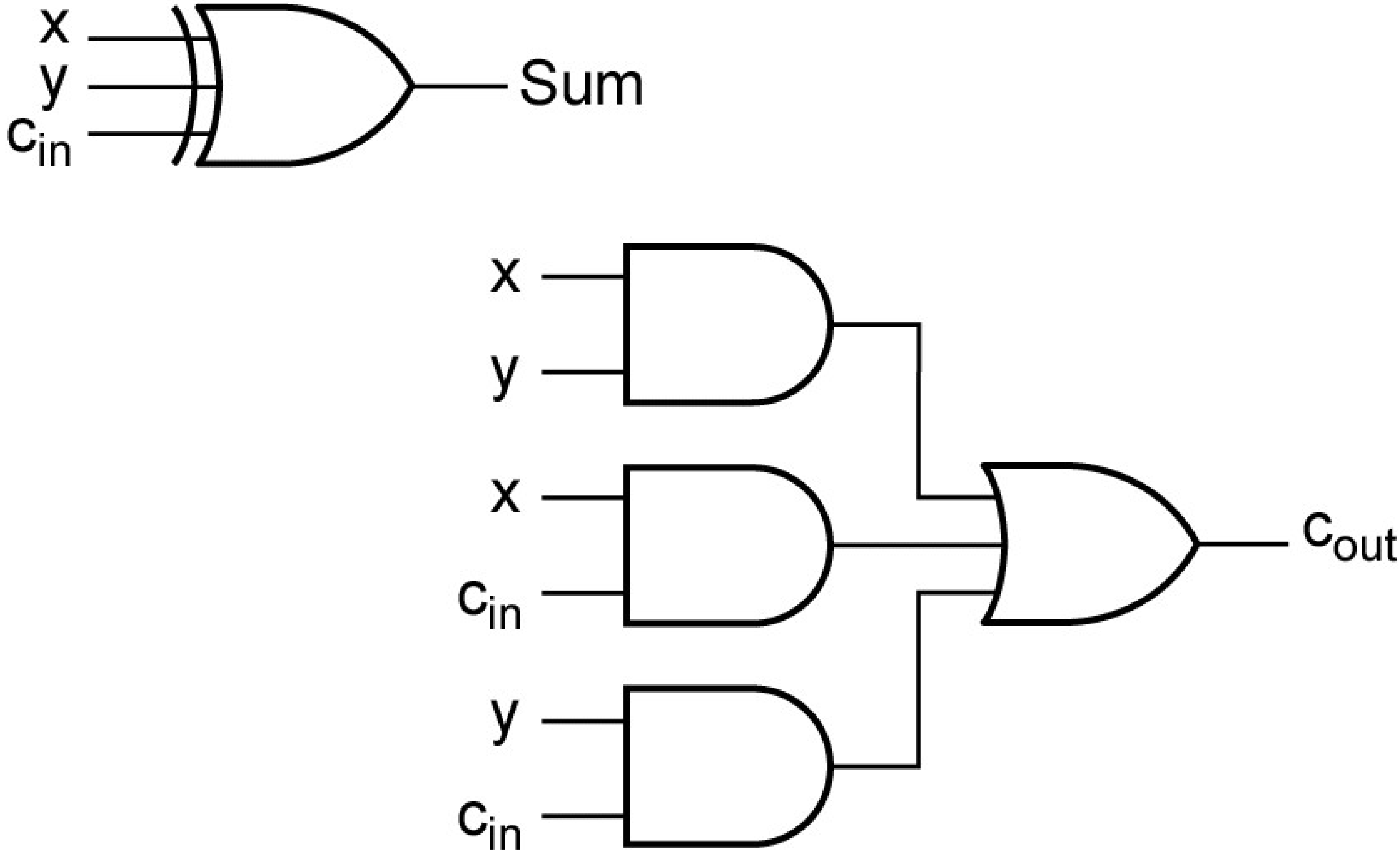


Figure 4-5: **Implementation of Full Adder**

Overflow for Signed Binary Numbers

An overflow has occurred if adding two positive numbers gives a negative result or adding two negative numbers gives a positive result.

We define an overflow signal, V = 1 if an overflow occurs. For Figure 4-3, V = $A_3'B_3'S_3 + A_3B_3S_3'$

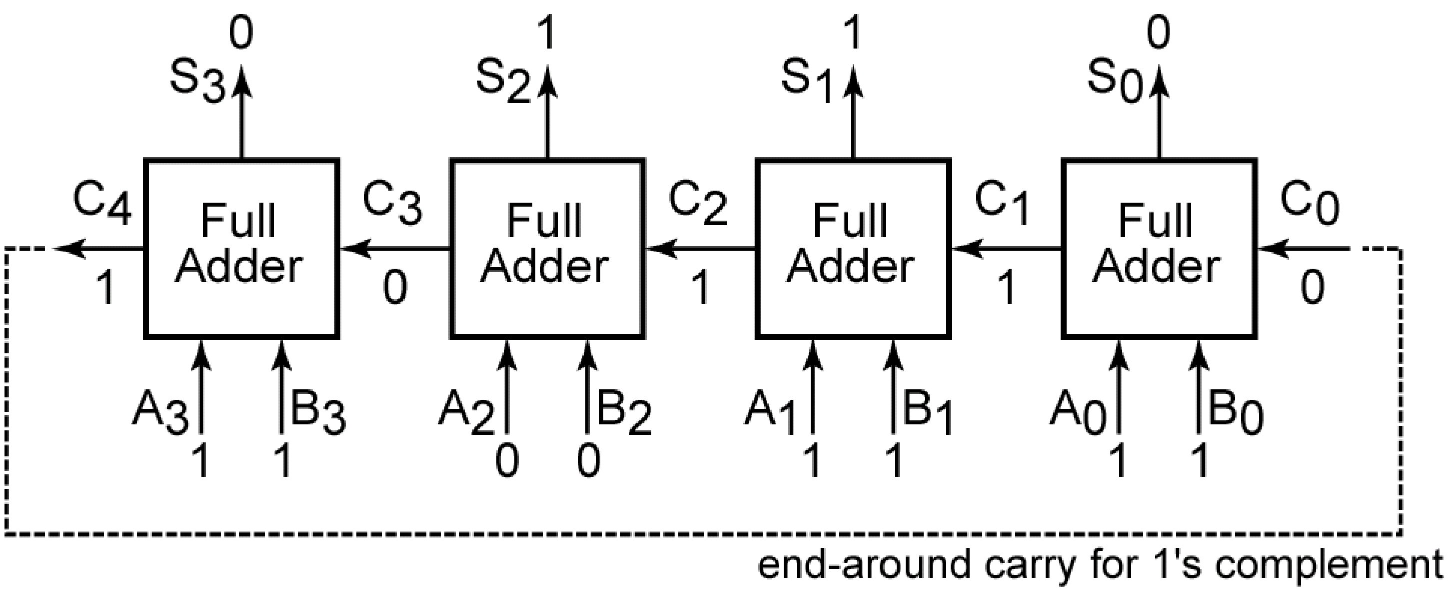


Figure 4-3

Full Adders 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.

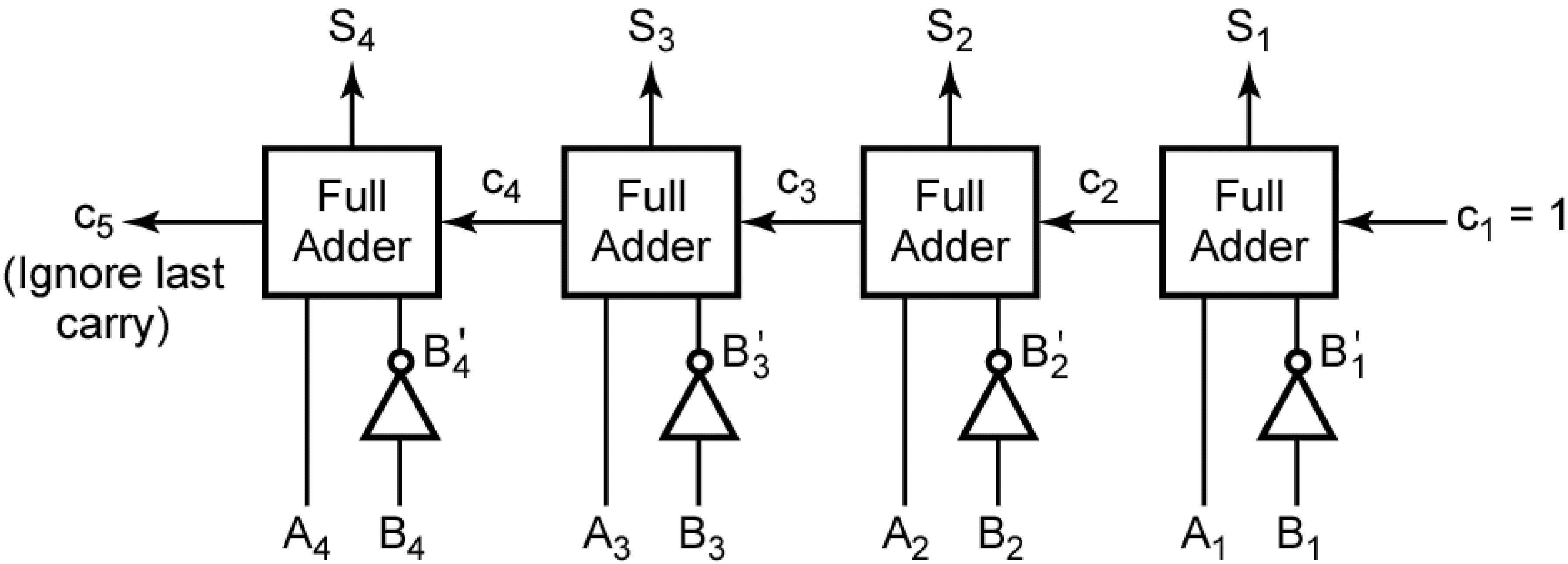


Figure 4-6: **Binary Subtracter Using Full Adders**

Alternatively, direct subtraction can be accomplished by employing a full subtracter in a manner analogous to a full adder.

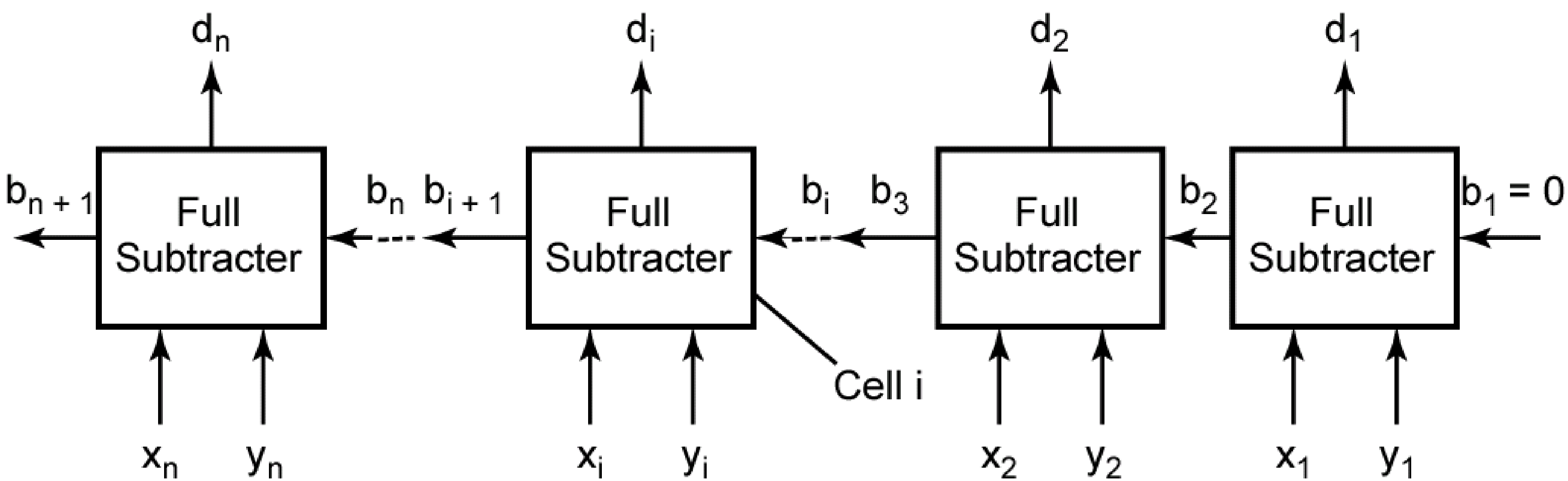


Figure 4-7: **Parallel Subtracter**

Table 4.6. Truth Table for Binary Full Subtracter

x_i	y_i	b_i	b_{i+1}	d_i
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Consider $x_i = 0$, $y_i = 1$, and $b_i = 1$:

	Column i Before Borrow	Column i After Borrow	
x_i	0	10	
$-b_i$	-1	-1	
$-y_i$	-1	-1	
d_i		0	$(b_{i+1} = 1)$

Section 4.7 (p. 107)