Chap 4
Combinational Logic

Hanwool Jeong

hwjeong@kw.ac.kr

Combinational Sequential Logic

- Logic circuit is categorized into combinational/sequential.
- A combinational circuit yields outputs determined from only the present inputs ➔ Specified with Boolean functions.
- In contrast, sequential circuits employ storage elements in addition to logic gates.
  - Their outputs are the ***inputs*** and the ***state*** of the storage elements.
  - Because the ***state*** is a function of previous inputs, the outputs depend not only on present inputs, but also on past inputs
- Let's use what we learned to formulate systematic analysis and design procedures for combinational circuits

Design Procedure

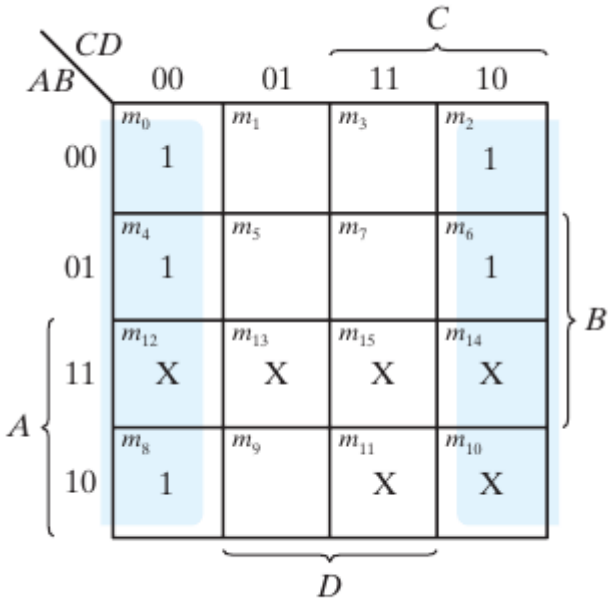
1. From the specifications, determine the required number of inputs and outputs and assign a symbol to each.
2. Derive the truth table that defines the required relationship between inputs and outputs.
3. Obtain the simplified Boolean functions for each output as a function of the input variables.
4. Draw the logic diagram and verify the correctness of the design

Code Conversion Example

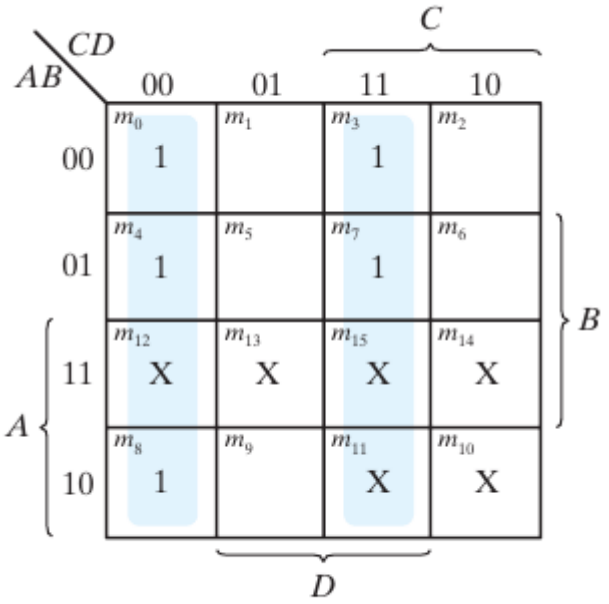
- BCD to Excess-3

Table 4.2
Truth Table for Code Conversion Example

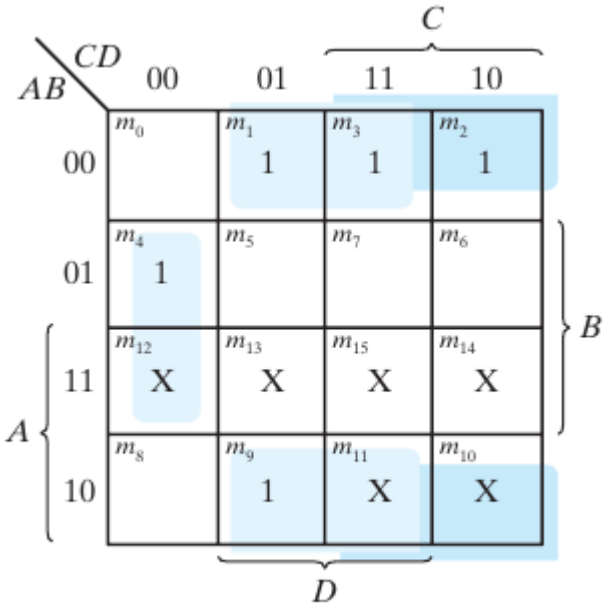
Input BCD				Output Excess-3 Code			
A	*B*	*C*	*D*	*w*	*x*	*y*	*z*
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0



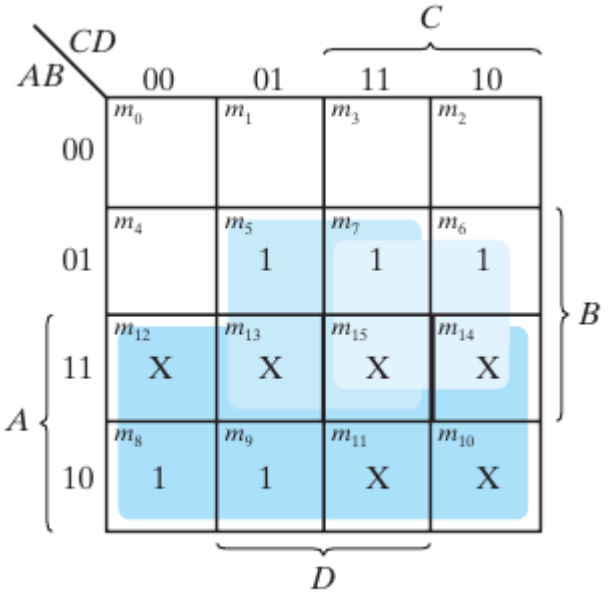
$z = D'$



$y = CD + C'D'$



$x = B'C + B'D + BC'D'$



$w = A + BC + BD$

Binary Addition

- Can you perform 1011 + 0011 =?
- What is the maximum bit number of the output?
- How can we design a circuit perform the above?

Components for Binary Adder

- 0+ 0 = 0, 0 + 1 = 1, 1 + 0 = 1, and 1 + 1 = 10
- 1+1 = 10 : Output consists of two digit
  ➔ Higher significant bit is called Carry to be added to the next higher order bit
- Thus, two circuits are needed for binary addition
  - Circuit for the addition of two bits: **half adder for** LSB
  - Circuit for the addition of three bits (two significant bits and a previous carry: is a **full adder** .

Truth Table of Half Adder/Full Adder

Table 4.3
Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

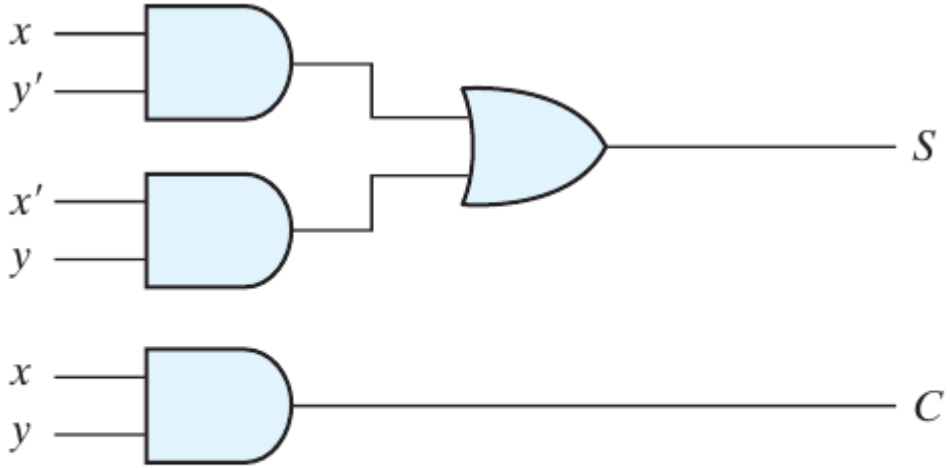
Table 4.4
Full Adder

x	y	z	C	S
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

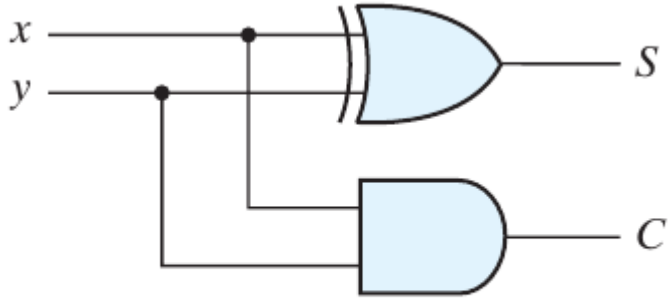
Circuit Design for Half Adder

Table 4.3
Half Adder

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



(a) $S = xy' + x'y$
$C = xy$

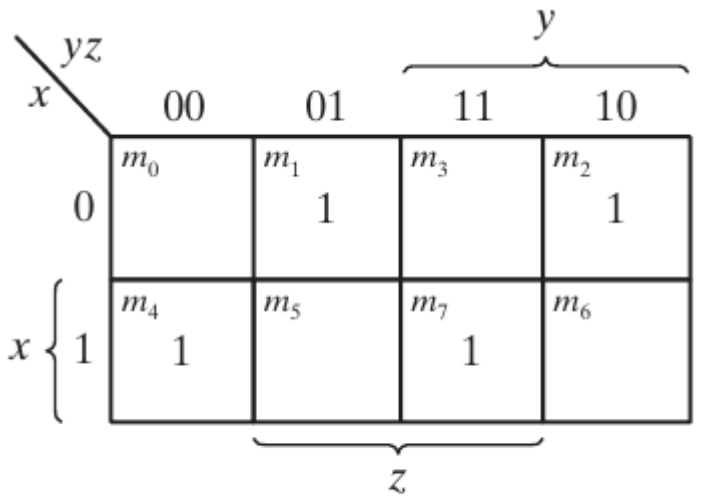


(b) $S = x \oplus y$
$C = xy$

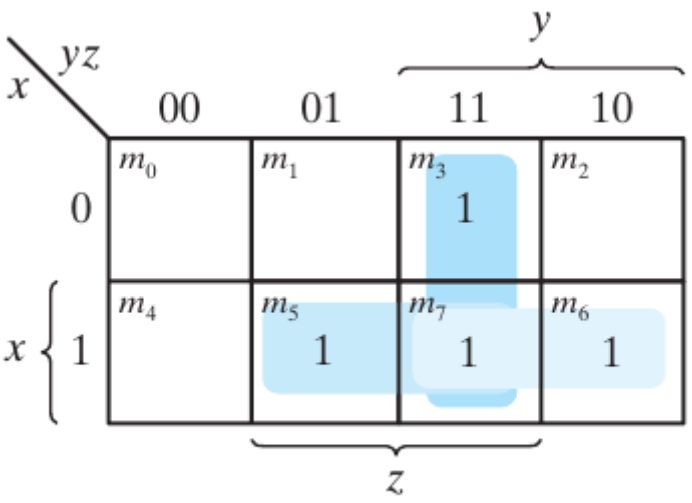
Circuit Design for Full Adder

Table 4.4
Full Adder

x	y	z	C	S
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

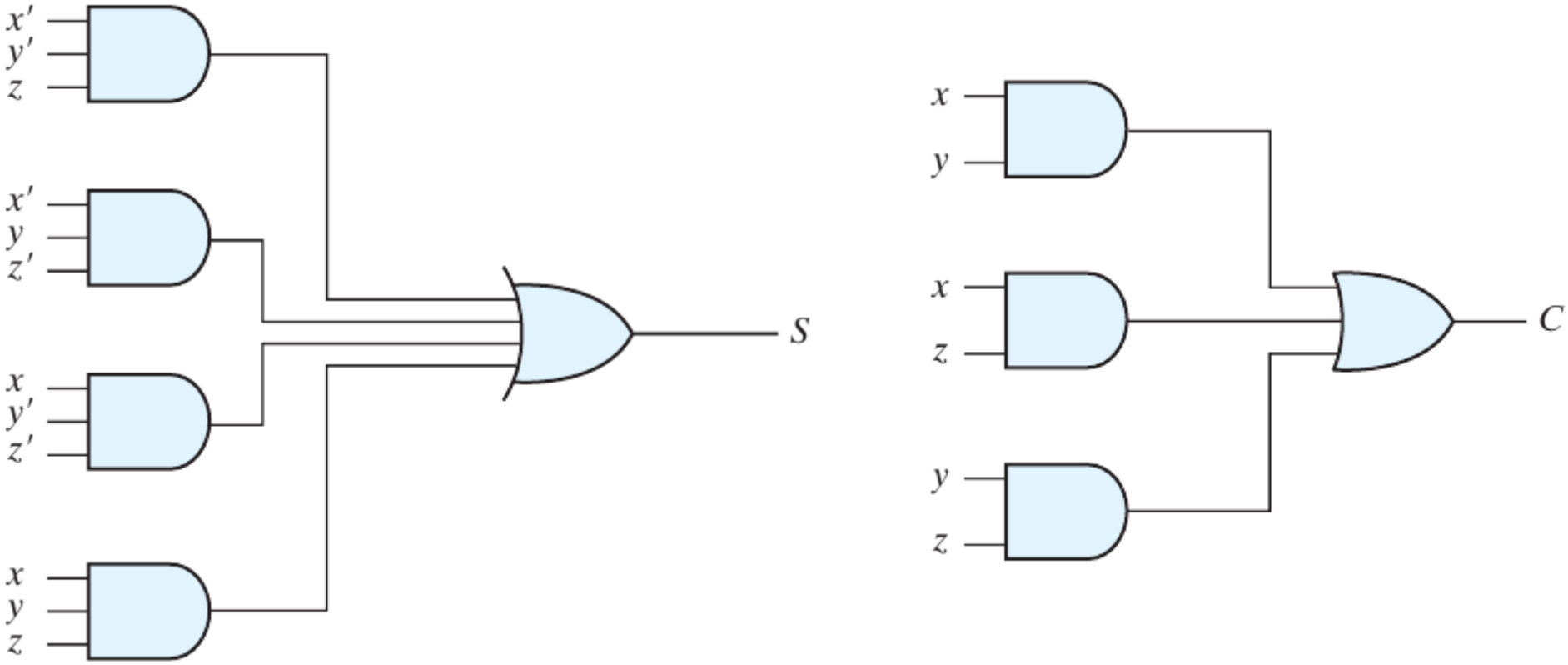


(a) $S = x'y'z + x'yz' + xy'z' + xyz$



(b) $C = xy + xz + yz$

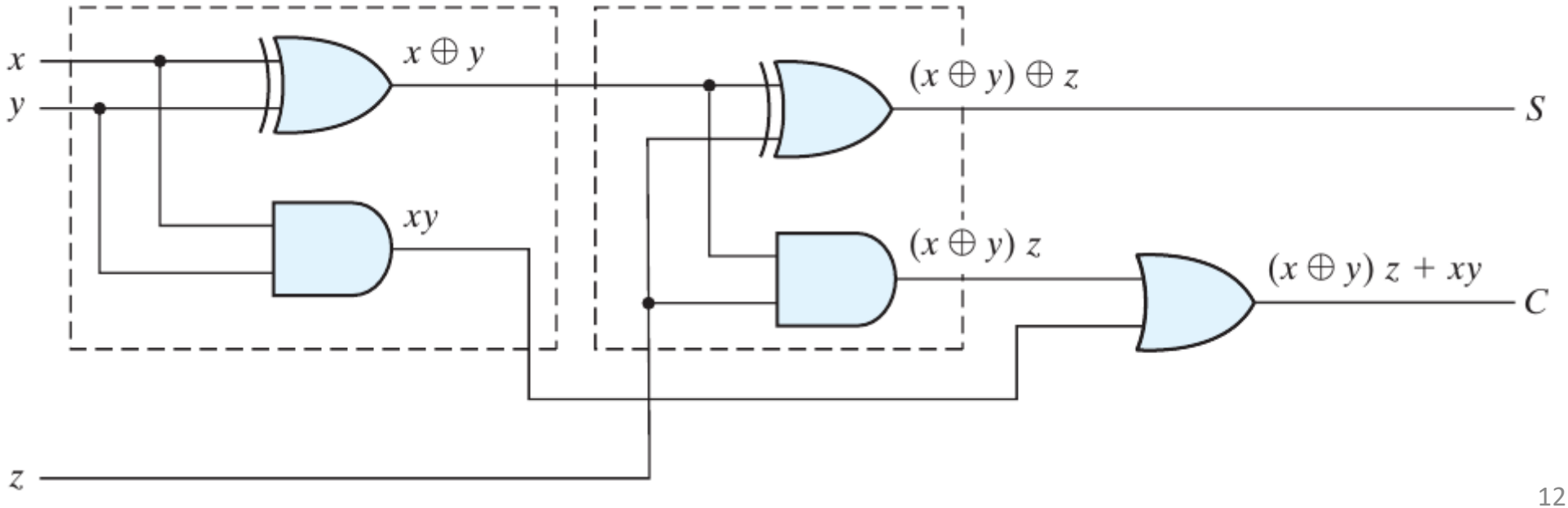
Circuit Design for Full Adder (cont'd)



Full Adder With Two Half Adders

$$
\begin{aligned}
S &= z \oplus (x \oplus y) \\
&= z'(xy' + x'y) + z(xy' + x'y)' \\
&= z'(xy' + x'y) + z(xy + x'y') \\
&= xy'z' + x'yz' + xyz + x'y'z
\end{aligned}
$$

$$
C = z(xy' + x'y) + xy = xy'z + x'yz + xy
$$

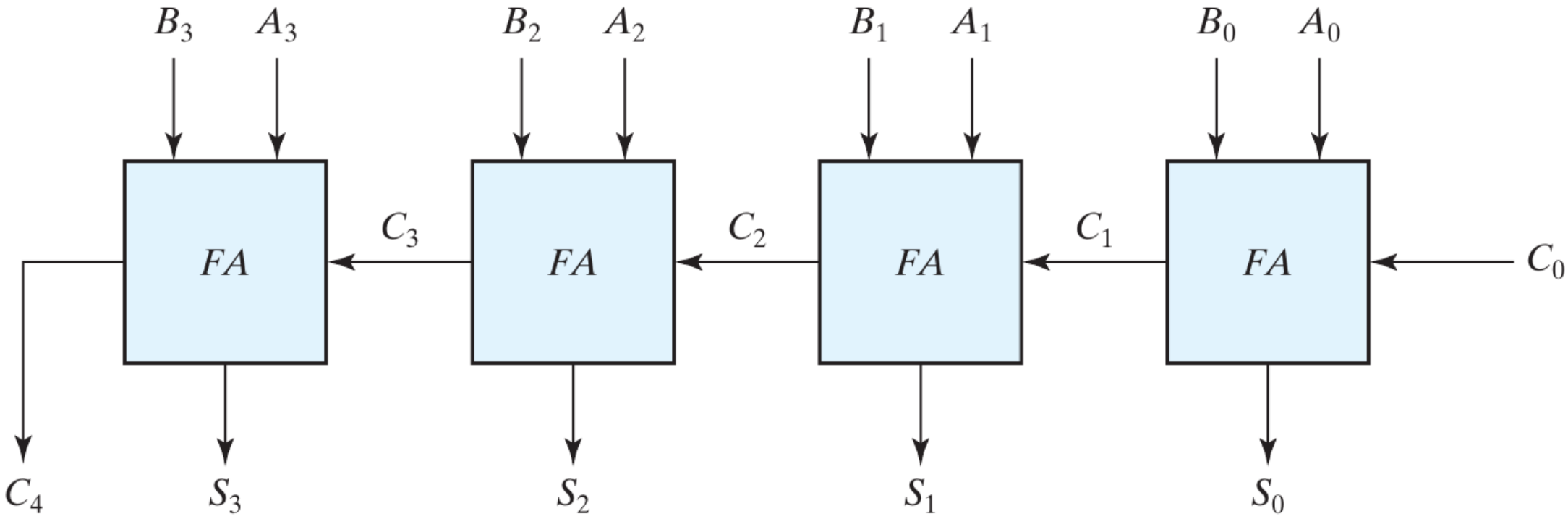


Full Adder With Two Half Adders

Table 4.4
Full Adder

x	*y*	*z*	*C*	*S*
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

Four-Bit Adder Design
Carry Propagation Adder



- Carry propagation time is critical. We should “wait” the less significant bit results to derive LSB and final Carry out.
- Is there any technique we can reduce the carry propagation time?

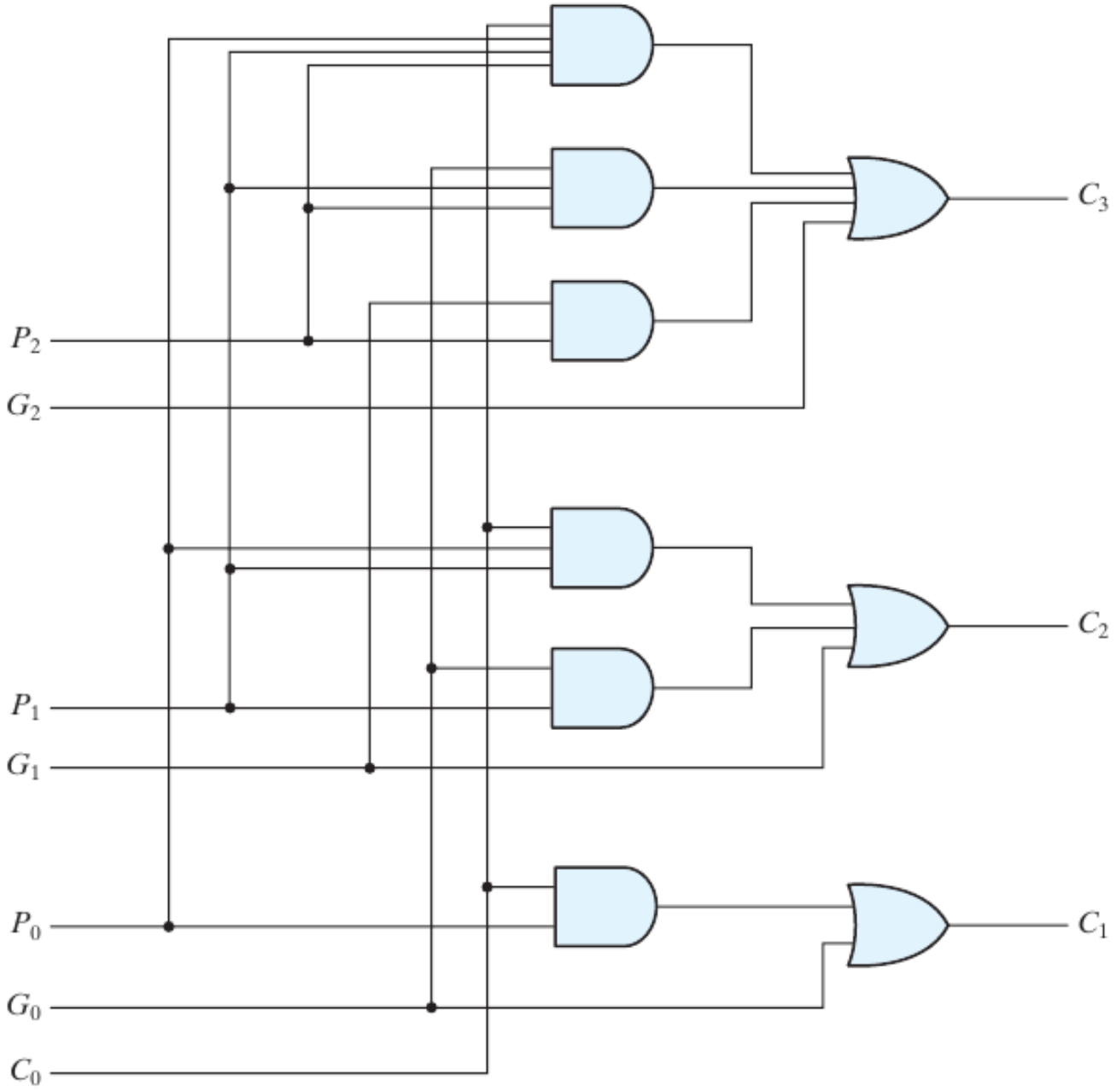
Consider How Carry is Generated for Each Bit

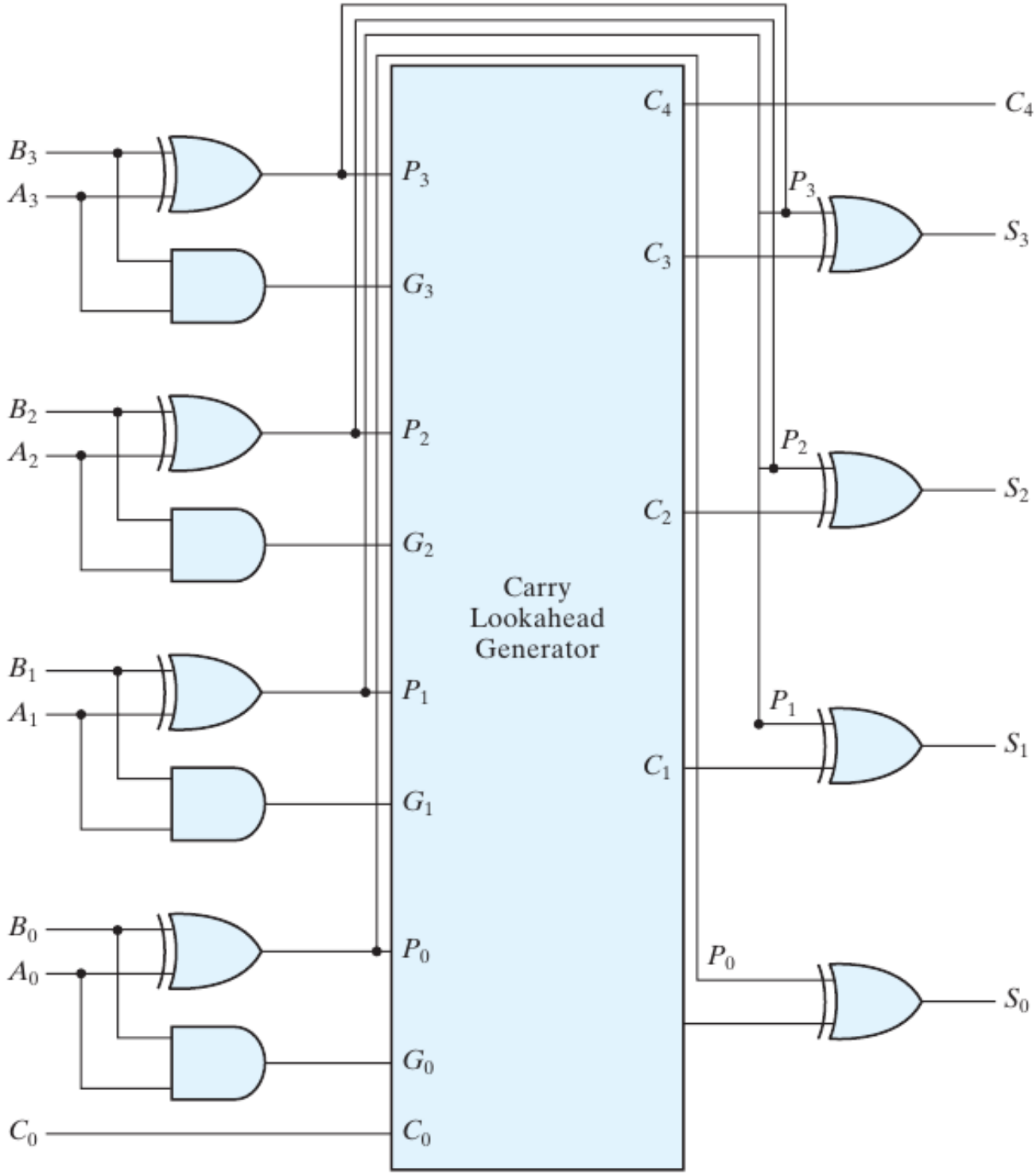
- For a given bit *n*, how does the carry-out C_{n+1} generate?

$$C_{n+1} = C_n (A_n \oplus B_n) + AB$$

- With defining P_n and G_n as

- Then C_3 can be calculated step by step as:


P_2
G_2
P_1
G_1
P_0
G_0
C_0
C_3
C_2
C_1



$$S_i = P_i \oplus C_i$$

Binary Subtraction

- Sign representation of binary number: two's complement
- MSB is used for representing "sign"

Table 1.3
Signed Binary Numbers

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

+ 6	00000110	− 6	11111010
+13	00001101	+13	00001101
+19	00010011	+ 7	00000111
+ 6	00000110	− 6	11111010
−13	11110011	−13	11110011
− 7	11111001	−19	11101101

Overflow

carries:	0	1	carries:	1	0
+70	0	1000110	−70	1	0111010
+80	0	1010000	−80	1	0110000
+150	1	0010110	−150	0	1101010

- An overflow can be detected by observing the carry-in and carry-out of the sign bit operation.

Binary Subtractor

