

CprE 281: Digital Logic

Instructor: Alexander Stoytchev

<http://www.ece.iastate.edu/~alexs/classes/>

Fast Adders

CprE 281: Digital Logic
Iowa State University, Ames, IA
Copyright © Alexander Stoytchev

Administrative Stuff

- **HW5 is out**
- **It is due on Monday Oct 1 @ 4pm.**
- **Please write clearly on the first page (in block capital letters) the following three things:**
 - **Your First and Last Name**
 - **Your Student ID Number**
 - **Your Lab Section Letter**
- **Also, please staple all of your pages together.**

Administrative Stuff

- **Labs Next Week**
- **Mini-Project**
- **This one is worth 3% of your grade.**
- **Make sure to get all the points.**
- **http://www.ece.iastate.edu/~alexs/classes/2018_Fall_281/labs/Project-Mini/**

Quick Review

The problems in which row are easier to calculate?

$$\begin{array}{r} \text{—} 82 \\ \text{—} 61 \\ \hline ?? \end{array}$$

$$\begin{array}{r} \text{—} 48 \\ \text{—} 26 \\ \hline ?? \end{array}$$

$$\begin{array}{r} \text{—} 32 \\ \text{—} 11 \\ \hline ?? \end{array}$$

$$\begin{array}{r} \text{—} 82 \\ \text{—} 64 \\ \hline ?? \end{array}$$

$$\begin{array}{r} \text{—} 48 \\ \text{—} 29 \\ \hline ?? \end{array}$$

$$\begin{array}{r} \text{—} 32 \\ \text{—} 13 \\ \hline ?? \end{array}$$

The problems in which row are easier to calculate?

$$\begin{array}{r} 82 \\ - 61 \\ \hline 21 \end{array}$$

$$\begin{array}{r} 48 \\ - 26 \\ \hline 22 \end{array}$$

$$\begin{array}{r} 32 \\ - 11 \\ \hline 21 \end{array}$$

Why?

$$\begin{array}{r} 82 \\ - 64 \\ \hline 18 \end{array}$$

$$\begin{array}{r} 48 \\ - 29 \\ \hline 19 \end{array}$$

$$\begin{array}{r} 32 \\ - 13 \\ \hline 19 \end{array}$$

Another Way to Do Subtraction

$$82 - 64 = 82 + 100 - 100 - 64$$

Another Way to Do Subtraction

$$\begin{aligned}82 - 64 &= 82 + 100 - 100 - 64 \\ &= 82 + (100 - 64) - 100\end{aligned}$$

Another Way to Do Subtraction

$$82 - 64 = 82 + 100 - 100 - 64$$

$$= 82 + (100 - 64) - 100$$

$$= 82 + (99 + 1 - 64) - 100$$

Another Way to Do Subtraction

$$82 - 64 = 82 + 100 - 100 - 64$$

$$= 82 + (100 - 64) - 100$$

$$= 82 + (99 + 1 - 64) - 100$$

$$= 82 + (99 - 64) + 1 - 100$$

Another Way to Do Subtraction

$$82 - 64 = 82 + 100 - 100 - 64$$

$$= 82 + (100 - 64) - 100$$

$$= 82 + (99 + 1 - 64) - 100$$

Does not require borrows

$$= 82 + (99 - 64) + 1 - 100$$

9's Complement (subtract each digit from 9)

$$\begin{array}{r} 99 \\ - 64 \\ \hline 35 \end{array}$$

10's Complement

(subtract each digit from 9 and add 1 to the result)

$$\begin{array}{r} 99 \\ - 64 \\ \hline 35 + 1 = 36 \end{array}$$

Another Way to Do Subtraction

$$82 - 64 = 82 + (99 - 64) + 1 - 100$$

Another Way to Do Subtraction

9's complement

$$82 - 64 = 82 + (99 - 64) + 1 - 100$$

Another Way to Do Subtraction

9's complement

$$\begin{aligned} 82 - 64 &= 82 + (99 - 64) + 1 - 100 \\ &= 82 + 35 + 1 - 100 \end{aligned}$$

Another Way to Do Subtraction

9's complement

$$82 - 64 = 82 + (99 - 64) + 1 - 100$$

$$= 82 + (35 + 1) - 100$$

10's complement

Another Way to Do Subtraction

9's complement

$$82 - 64 = 82 + (99 - 64) + 1 - 100$$

$$= 82 + (35 + 1) - 100$$

10's complement

$$= 82 + 36 - 100$$

Another Way to Do Subtraction

9's complement

$$82 - 64 = 82 + (99 - 64) + 1 - 100$$

$$= 82 + (35 + 1) - 100$$

10's complement

$$= 82 + 36 - 100 \quad // \text{ Add the first two.}$$

$$= 118 - 100$$

Another Way to Do Subtraction

9's complement

$$82 - 64 = 82 + (99 - 64) + 1 - 100$$

10's complement

$$= 82 + (35 + 1) - 100$$

$$= 82 + 36 - 100 \quad // \text{ Add the first two.}$$

$$= 118 - 100 \quad // \text{ Just delete the leading 1.}$$

// No need to subtract 100.

$$= 18$$

2' s complement

Let K be the negative equivalent of an n -bit positive number P .

Then, in 2' s complement representation K is obtained by subtracting P from 2^n , namely

$$K = 2^n - P$$

Deriving 2' s complement

For a positive n-bit number P, let K_1 and K_2 denote its 1' s and 2' s complements, respectively.

$$K_1 = (2^n - 1) - P$$

$$K_2 = 2^n - P$$

Since $K_2 = K_1 + 1$, it is evident that in a logic circuit the 2' s complement can be computed by inverting all bits of P and then adding 1 to the resulting 1' s-complement number.

Find the 2's complement of ...

0 1 0 1

0 0 1 0

0 1 0 0

0 1 1 1

Find the 2' s complement of ...

0 1 0 1

1 0 1 0

0 0 1 0

1 1 0 1

0 1 0 0

1 0 1 1

0 1 1 1

1 0 0 0

Invert all bits.

Find the 2's complement of ...

$$\begin{array}{r} 0101 \\ + 1010 \\ + 1 \\ \hline 1011 \end{array}$$

$$\begin{array}{r} 0010 \\ + 1101 \\ + 1 \\ \hline 1110 \end{array}$$

$$\begin{array}{r} 0100 \\ + 1011 \\ + 1 \\ \hline 1100 \end{array}$$

$$\begin{array}{r} 0111 \\ + 1000 \\ + 1 \\ \hline 1001 \end{array}$$

Then add 1.

Interpretation of four-bit signed integers

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

Interpretation of four-bit signed integers

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

The top half is the same in all three representations.
It corresponds to the positive integers.

Interpretation of four-bit signed integers

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

In all three representations the first bit represents the sign.
If that bit is 1, then the number is negative.

Interpretation of four-bit signed integers

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

Notice that in this representation there are two zeros!

Interpretation of four-bit signed integers

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1


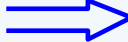
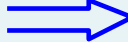
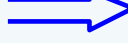

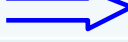
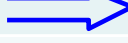
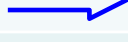



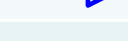
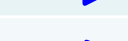
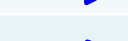


There are two zeros in this representation as well!

Interpretation of four-bit signed integers

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

In this representation there is one more negative number.

Taking the 2's complement negates the number


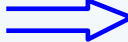
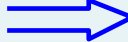
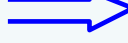

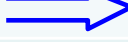
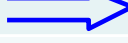
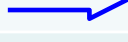



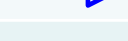
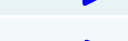
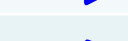


decimal	$b_3 b_2 b_1 b_0$	take the 2's complement	$b_3 b_2 b_1 b_0$	decimal
+7	0111		1001	-7
+6	0110		1010	-6
+5	0101		1011	-5
+4	0100		1100	-4
+3	0011		1101	-3
+2	0010		1110	-2
+1	0001		1111	-1
+0	0000		0000	+0
-8	1000		1000	-8
-7	1001		0111	+7
-6	1010		0110	+6
-5	1011		0101	+5
-4	1100		0100	+4
-3	1101		0011	+3
-2	1110		0010	+2
-1	1111		0001	+1

Taking the 2's complement negates the number

decimal	b ₃ b ₂ b ₁ b ₀	take the 2's complement	b ₃ b ₂ b ₁ b ₀	decimal
+7	0111	⇒	1001	-7
+6	0110	⇒	1010	-6
+5	0101	⇒	1011	-5
+4	0100	⇒	1100	-4
+3	0011	⇒	1101	-3
+2	0010	⇒	1110	-2
+1	0001	⇒	1111	-1
+0	0000	⇒	0000	+0
-8	1000	⇒	1000	-8
-7	1001	⇒	0111	+7
-6	1010	⇒	0110	+6
-5	1011	⇒	0101	+5
-4	1100	⇒	0100	+4
-3	1101	⇒	0011	+3
-2	1110	⇒	0010	+2
-1	1111	⇒	0001	+1

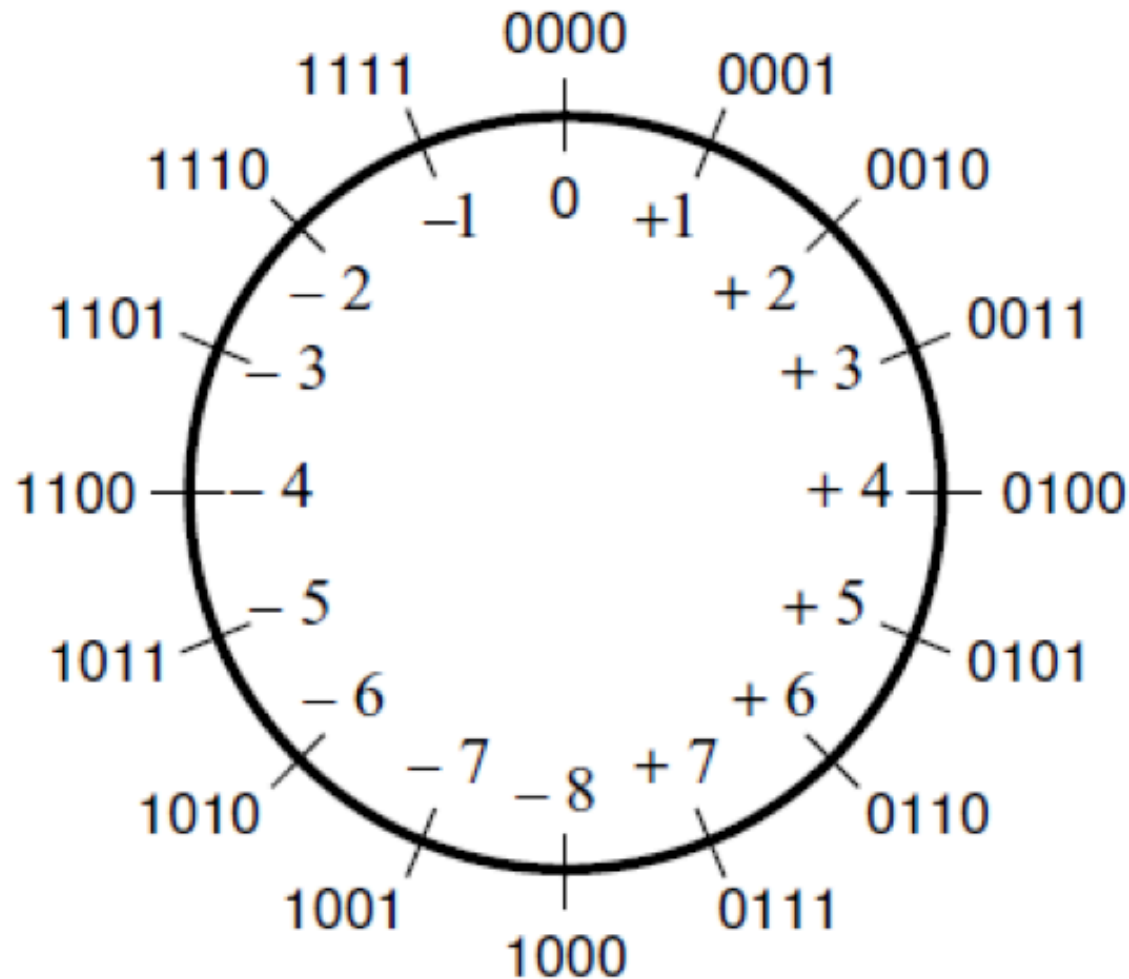
This is the only exception

Taking the 2' s complement negates the number

decimal	$b_3 b_2 b_1 b_0$	take the 2's complement	$b_3 b_2 b_1 b_0$	decimal
+7	0111		1001	-7
+6	0110		1010	-6
+5	0101		1011	-5
+4	0100		1100	-4
+3	0011		1101	-3
+2	0010		1110	-2
+1	0001		1111	-1
+0	0000		0000	+0
-8	1000		1000	-8
-7	1001		0111	+7
-6	1010		0110	+6
-5	1011		0101	+5
-4	1100		0100	+4
-3	1101		0011	+3
-2	1110		0010	+2
-1	1111		0001	+1

And this one too.

The number circle for 2's complement



[Figure 3.11a from the textbook]

A) Example of 2's complement addition

$$\begin{array}{r}
 (+5) \\
 + (+2) \\
 \hline
 (+7)
 \end{array}
 \qquad
 \begin{array}{r}
 0101 \\
 + 0010 \\
 \hline
 0111
 \end{array}$$

$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1


B) Example of 2's complement addition

$$\begin{array}{r}
 (-5) \\
 + (+2) \\
 \hline
 (-3)
 \end{array}
 \qquad
 \begin{array}{r}
 \color{red}{1011} \\
 + \color{green}{0010} \\
 \hline
 \color{blue}{1101}
 \end{array}$$

$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

C) Example of 2's complement addition

$$\begin{array}{r}
 (+5) \quad \quad \quad 0101 \\
 + (-2) \quad \quad \quad 1110 \\
 \hline
 (+3) \quad \quad \quad 10011
 \end{array}$$




 ignore

$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

D) Example of 2's complement addition

$$\begin{array}{r}
 (-5) \\
 + (-2) \\
 \hline
 (-7)
 \end{array}
 \qquad
 \begin{array}{r}
 1011 \\
 + 1110 \\
 \hline
 11001
 \end{array}$$



 ignore

$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Naming Ambiguity: 2's Complement

2's complement has two different meanings:

- **representation for signed integer numbers**
- **algorithm for computing the 2's complement (regardless of the representation of the number)**

Naming Ambiguity: 2's Complement

2's complement has two different meanings:

- representation for signed integer numbers
in 2's complement
- algorithm for computing the 2's complement
(regardless of the representation of the number)
take the 2's complement

Example of 2' s complement subtraction

$$\begin{array}{r} (+5) \\ - (+2) \\ \hline (+3) \end{array} \quad \begin{array}{r} 0101 \\ - 0010 \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{r} 0101 \\ + 1110 \\ \hline 10011 \\ \uparrow \\ \text{ignore} \end{array}$$

\Rightarrow means take the 2's complement

Example of 2's complement subtraction

$$\begin{array}{r} (+5) \\ \textcircled{-} (+2) \\ \hline (+3) \end{array} \quad \begin{array}{r} 0101 \\ - 0010 \\ \hline \end{array} \quad \Rightarrow \quad \begin{array}{r} 0101 \\ \textcircled{+} 1110 \\ \hline 10011 \\ \uparrow \\ \text{ignore} \end{array}$$

Notice that the minus changes to a plus.

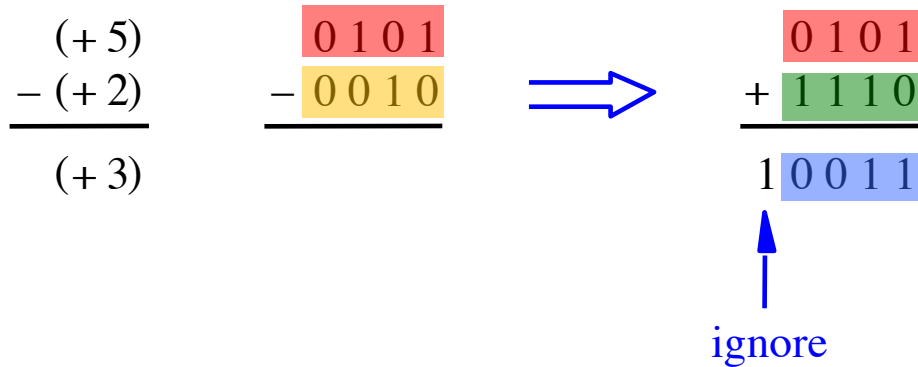
\Rightarrow means take the 2's complement

Example of 2's complement subtraction

$$\begin{array}{r}
 (+5) \\
 - (+2) \\
 \hline
 (+3)
 \end{array}
 \quad
 \begin{array}{r}
 \color{red}{0101} \\
 - \color{yellow}{0010} \\
 \hline
 \end{array}
 \quad
 \Rightarrow
 \quad
 \begin{array}{r}
 0101 \\
 + 1110 \\
 \hline
 10011 \\
 \uparrow \\
 \text{ignore}
 \end{array}$$

$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

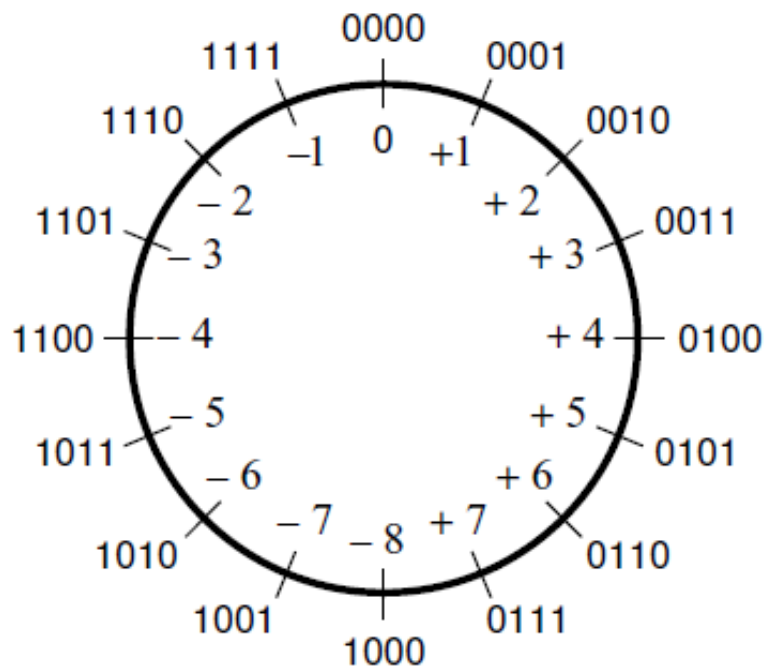
Example of 2's complement subtraction



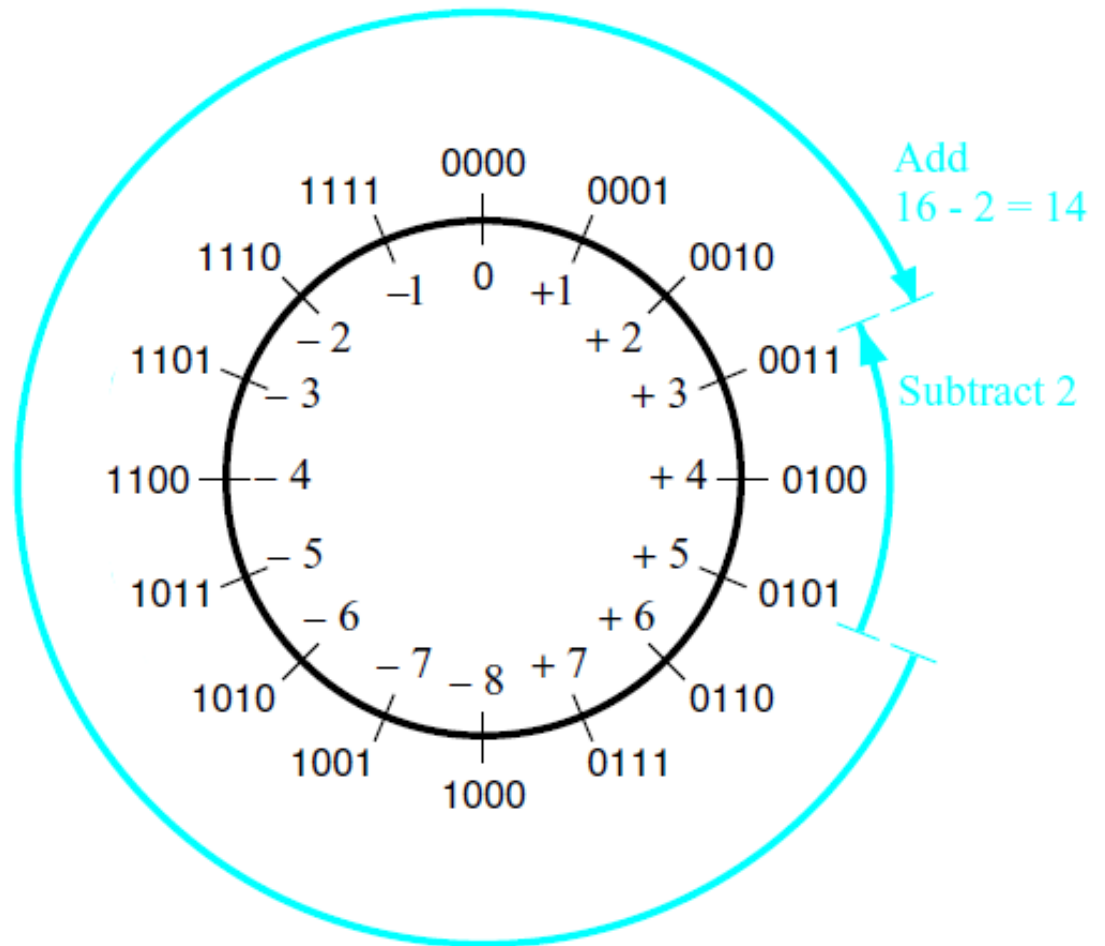
$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

[Figure 3.10 from the textbook]

Graphical interpretation of four-bit 2's complement numbers

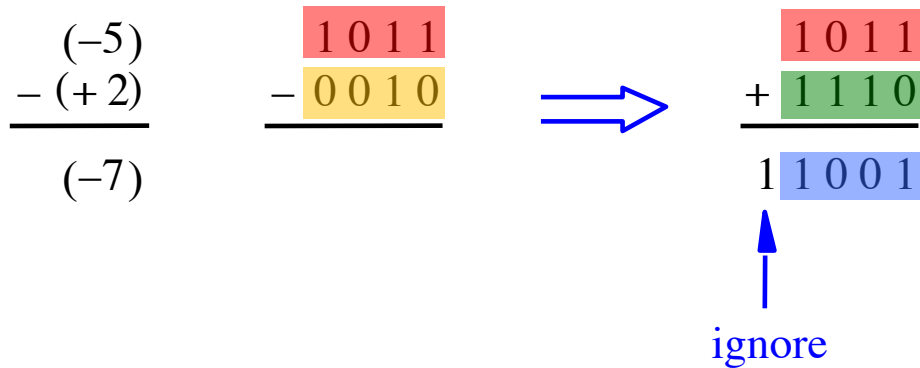


(a) The number circle



(b) Subtracting 2 by adding its 2's complement

Example of 2's complement subtraction



$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

[Figure 3.10 from the textbook]

Example of 2's complement subtraction

$$\begin{array}{r}
 (+5) \quad \quad \quad \boxed{0101} \\
 - (-2) \quad \quad \quad \boxed{1110} \\
 \hline
 (+7)
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{r}
 \quad \quad \quad \boxed{0101} \\
 + \quad \boxed{0010} \\
 \hline
 \quad \quad \boxed{0111}
 \end{array}$$

$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

[Figure 3.10 from the textbook]

Example of 2's complement subtraction

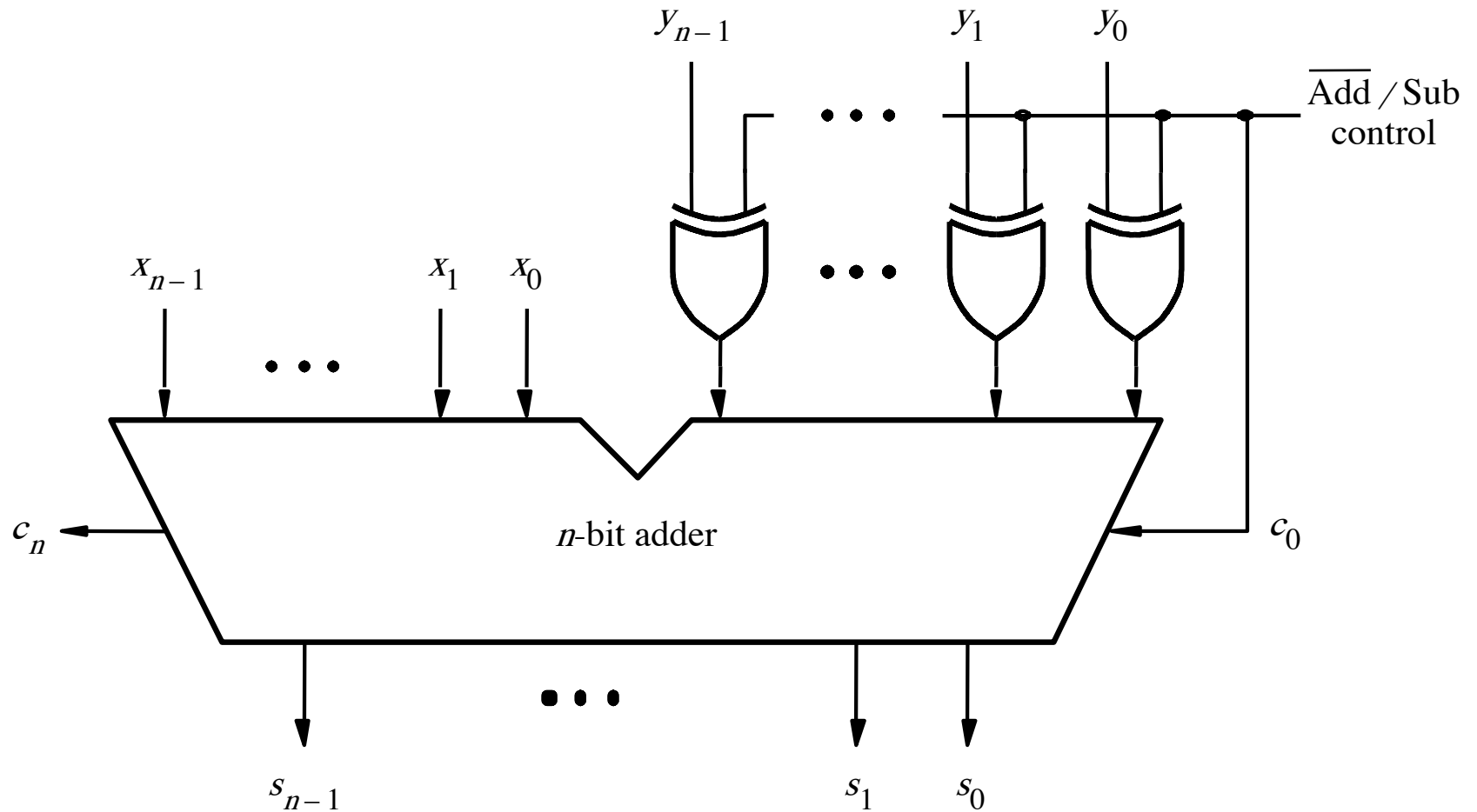
$$\begin{array}{r}
 (-5) \\
 - (-2) \\
 \hline
 (-3)
 \end{array}
 \quad
 \begin{array}{r}
 \text{1011} \\
 - \text{1110} \\
 \hline
 \end{array}
 \quad
 \Rightarrow
 \quad
 \begin{array}{r}
 \text{1011} \\
 + \text{0010} \\
 \hline
 \text{1101}
 \end{array}$$

$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

Take Home Message

- **Subtraction can be performed by simply adding the 2's complement of the second number, regardless of the signs of the two numbers.**
- **Thus, the same adder circuit can be used to perform both addition and subtraction !!!**

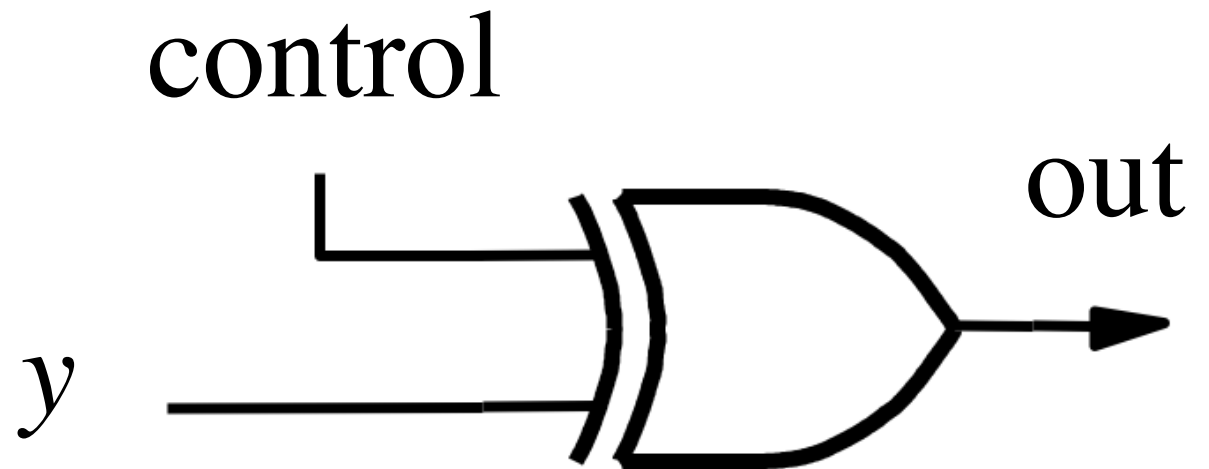
Adder/subtractor unit



[Figure 3.12 from the textbook]

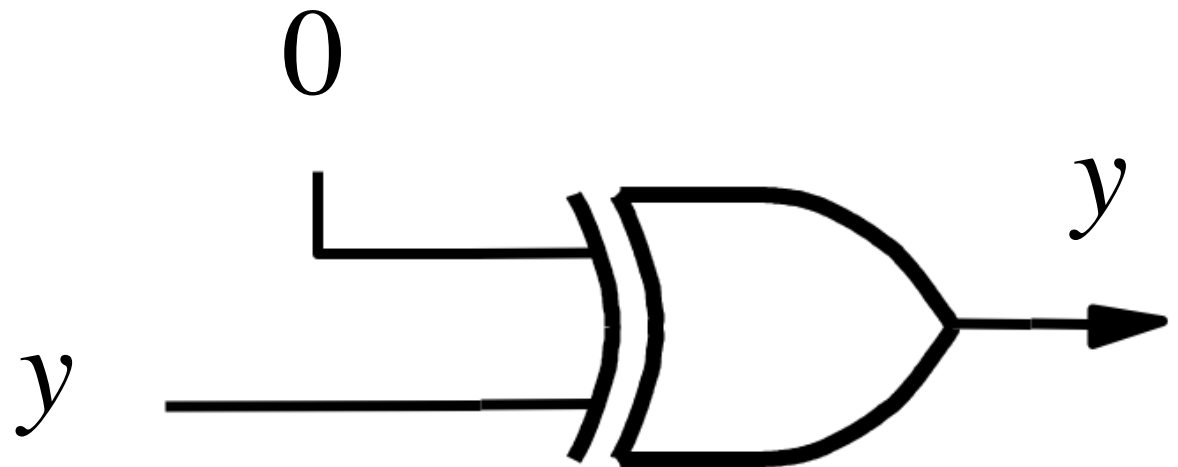

XOR Tricks

control	y	out
0	0	0
0	1	1
1	0	1
1	1	0



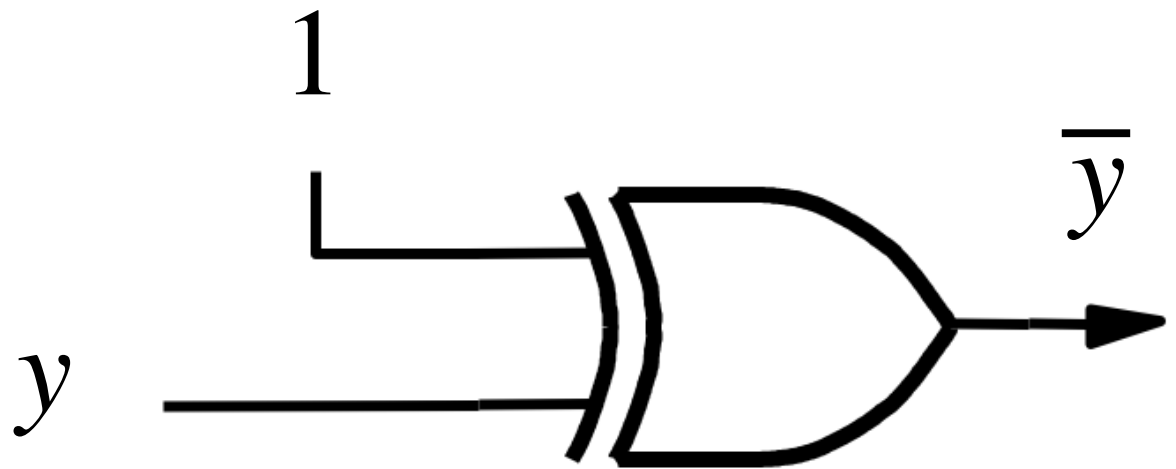
XOR as a repeater

control	y	out
0	0	0
0	1	1

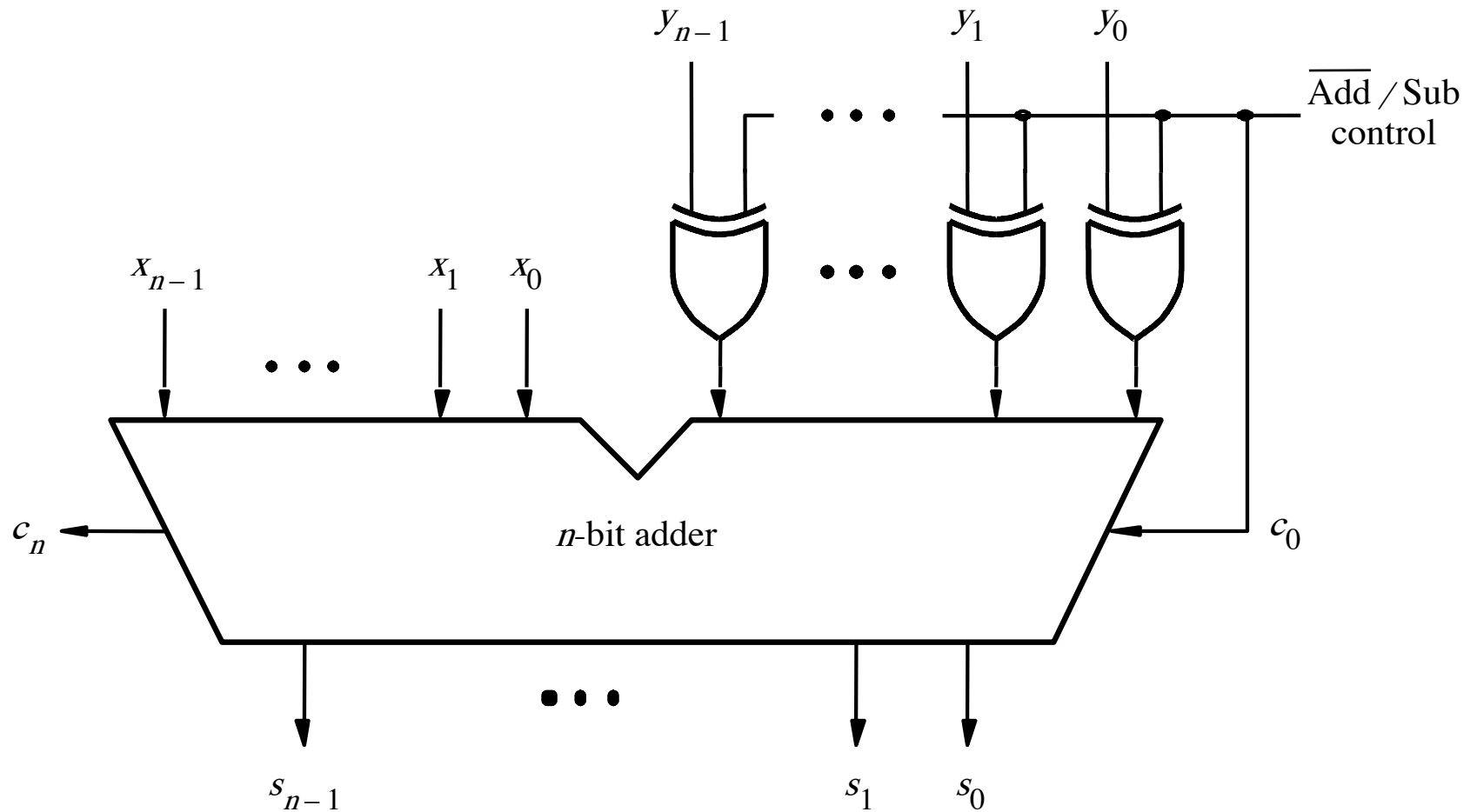


XOR as an inverter

control	y	out
1	0	1
1	1	0

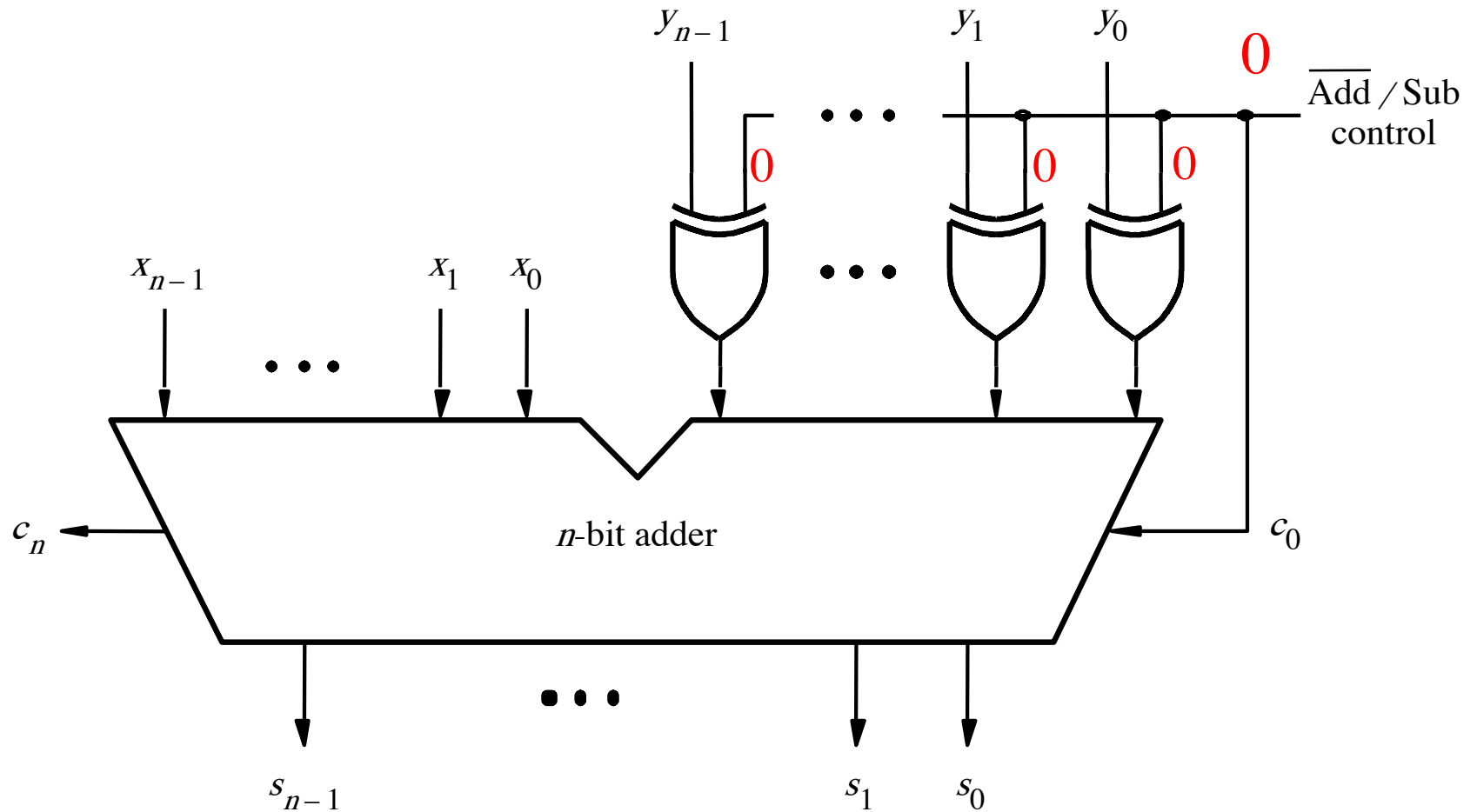


Addition: when control = 0



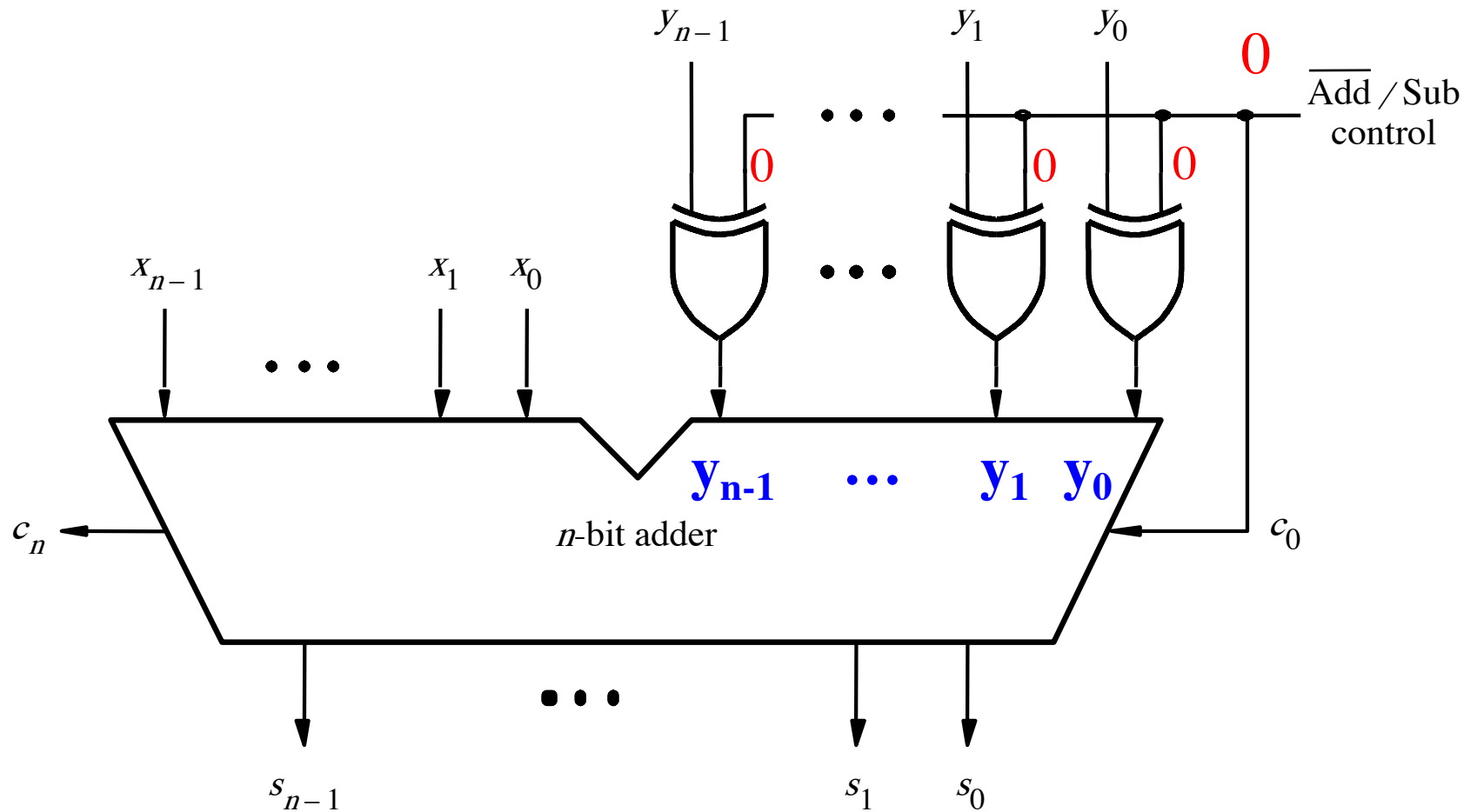
[Figure 3.12 from the textbook]

Addition: when control = 0



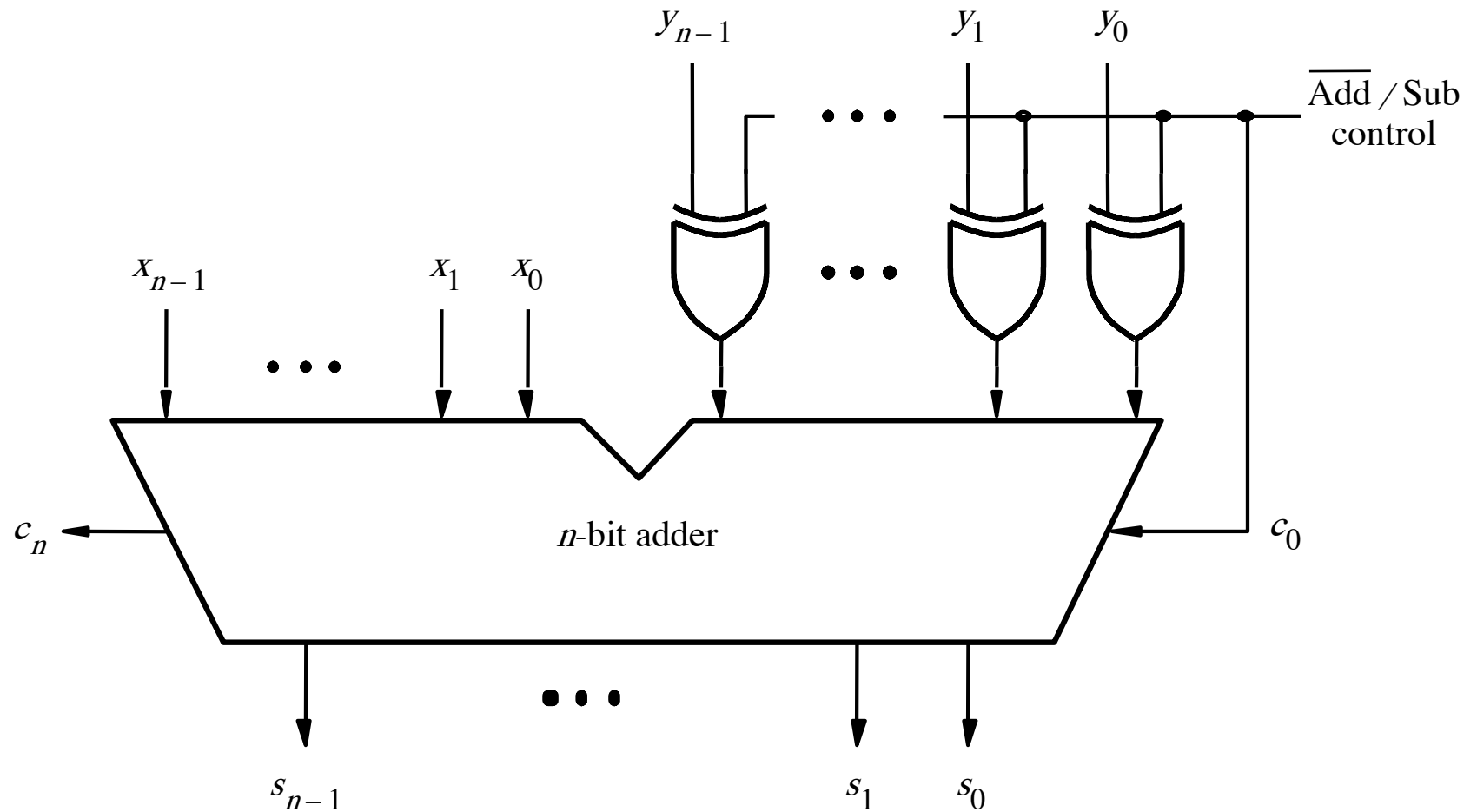
[Figure 3.12 from the textbook]

Addition: when control = 0



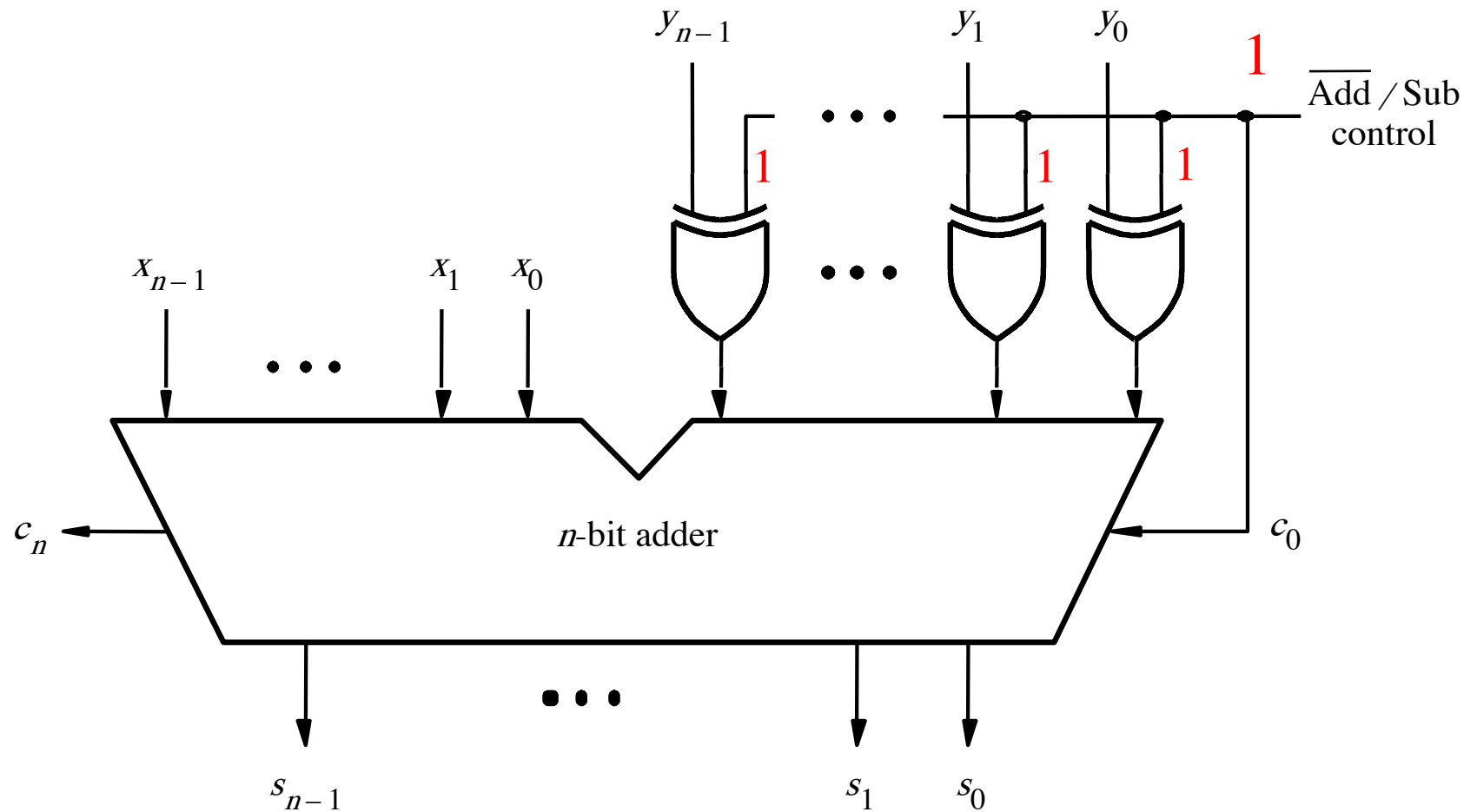
[Figure 3.12 from the textbook]

Subtraction: when control = 1



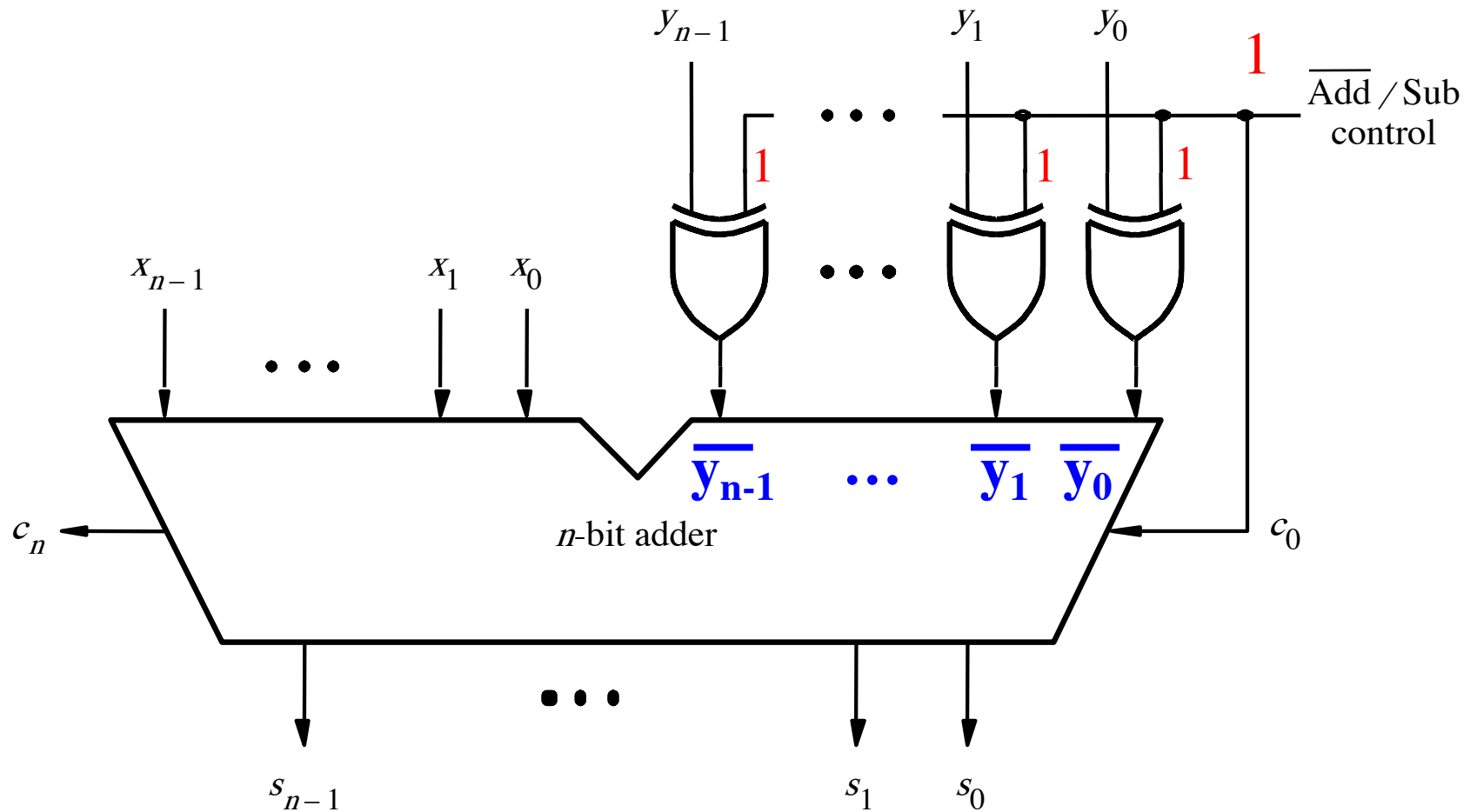
[Figure 3.12 from the textbook]

Subtraction: when control = 1



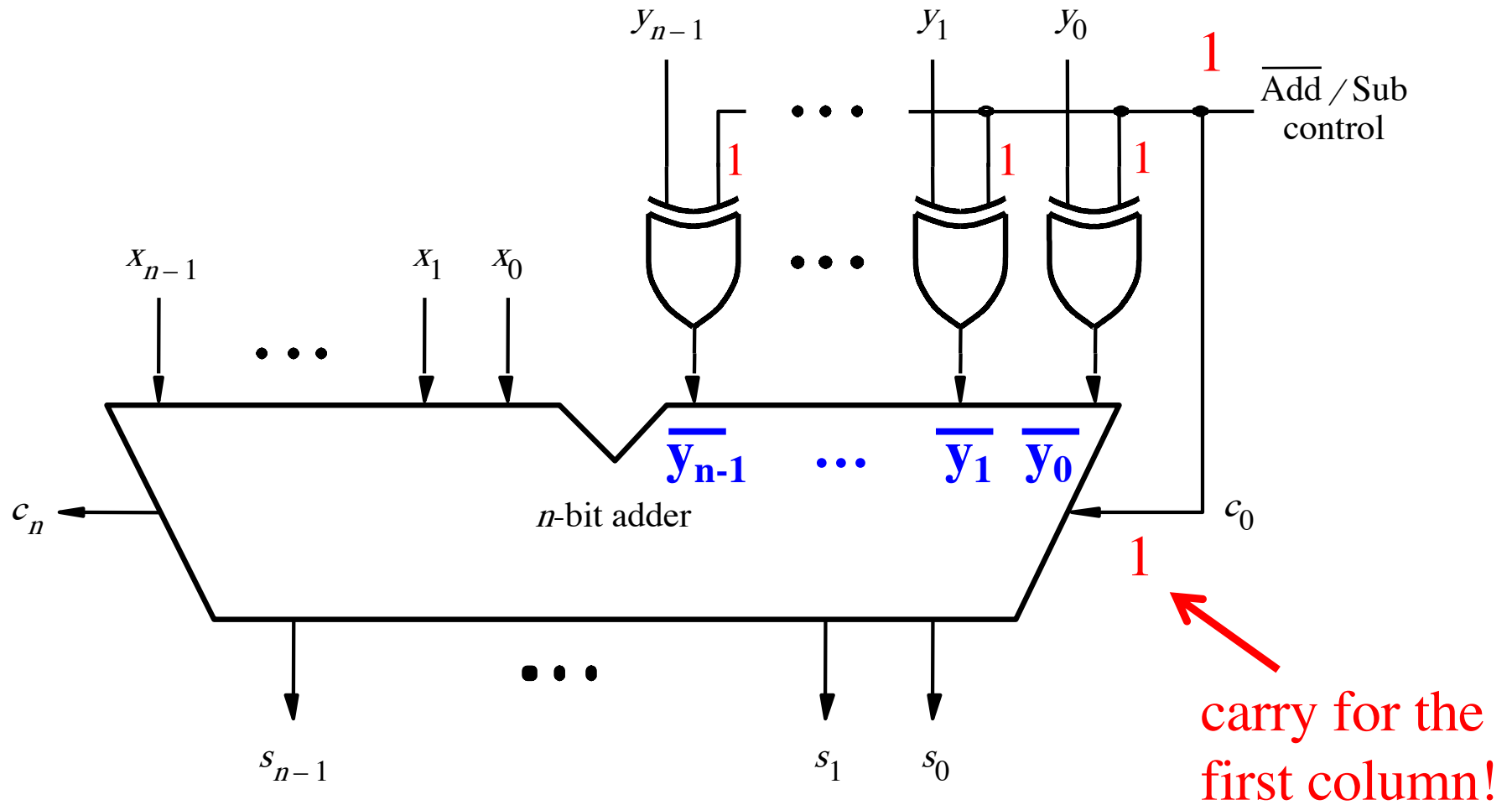
[Figure 3.12 from the textbook]

Subtraction: when control = 1



[Figure 3.12 from the textbook]

Subtraction: when control = 1



[Figure 3.12 from the textbook]

Examples of determination of overflow

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array} \quad + \quad \begin{array}{r} 0111 \\ 0010 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array} \quad + \quad \begin{array}{r} 1001 \\ 0010 \\ \hline 1011 \end{array}$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array} \quad + \quad \begin{array}{r} 0111 \\ 1110 \\ \hline 10101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array} \quad + \quad \begin{array}{r} 1001 \\ 1110 \\ \hline 10111 \end{array}$$

Examples of determination of overflow

		0 1 1 0 0				0 0 0 0 0
(+ 7)		0 1 1 1		(- 7)		1 0 0 1
+ (+ 2)	+	0 0 1 0		+ (+ 2)	+	0 0 1 0
<hr/>		<hr/>		<hr/>		<hr/>
(+ 9)		1 0 0 1		(- 5)		1 0 1 1

		1 1 1 0 0				1 0 0 0 0
(+ 7)		0 1 1 1		(- 7)		1 0 0 1
+ (- 2)	+	1 1 1 0		+ (- 2)	+	1 1 1 0
<hr/>		<hr/>		<hr/>		<hr/>
(+ 5)		1 0 1 0 1		(- 9)		1 0 1 1 1

Include the carry bits: $c_4 c_3 c_2 c_1 c_0$

Examples of determination of overflow

$$\begin{array}{r}
 (+7) \\
 + (+2) \\
 \hline
 (+9)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{01}100 \\
 + \quad 0111 \\
 \quad 0010 \\
 \hline
 1001
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (+2) \\
 \hline
 (-5)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{00}000 \\
 + \quad 1001 \\
 \quad 0010 \\
 \hline
 1011
 \end{array}$$

$$\begin{array}{r}
 (+7) \\
 + (-2) \\
 \hline
 (+5)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{11}100 \\
 + \quad 0111 \\
 \quad 1110 \\
 \hline
 10101
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (-2) \\
 \hline
 (-9)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{10}000 \\
 + \quad 1001 \\
 \quad 1110 \\
 \hline
 10111
 \end{array}$$

Include the carry bits: $\boxed{c_4 c_3} c_2 c_1 c_0$

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

$$\begin{array}{r}
 (+7) \\
 + (+2) \\
 \hline
 (+9)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{01}100 \\
 + \quad 0111 \\
 \quad 0010 \\
 \hline
 1001
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (+2) \\
 \hline
 (-5)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{00}000 \\
 + \quad 1001 \\
 \quad 0010 \\
 \hline
 1011
 \end{array}
 \quad
 \begin{array}{r}
 c_4 = 0 \\
 c_3 = 0
 \end{array}$$

$$c_4 = 1$$

$$c_3 = 1$$

$$\begin{array}{r}
 (+7) \\
 + (-2) \\
 \hline
 (+5)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{11}100 \\
 + \quad 0111 \\
 \quad 1110 \\
 \hline
 10101
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (-2) \\
 \hline
 (-9)
 \end{array}
 \quad
 \begin{array}{r}
 \boxed{10}000 \\
 + \quad 1001 \\
 \quad 1110 \\
 \hline
 10111
 \end{array}
 \quad
 \begin{array}{r}
 c_4 = 1 \\
 c_3 = 0
 \end{array}$$

Include the carry bits: $\boxed{c_4 c_3} c_2 c_1 c_0$

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

$$\begin{array}{r}
 (+7) \\
 + (+2) \\
 \hline
 (+9)
 \end{array}
 \quad
 \begin{array}{r}
 + \\
 \boxed{01}100 \\
 0111 \\
 0010 \\
 \hline
 1001
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (+2) \\
 \hline
 (-5)
 \end{array}
 \quad
 \begin{array}{r}
 + \\
 \boxed{00}000 \\
 1001 \\
 0010 \\
 \hline
 1011
 \end{array}$$

$$c_4 = 0$$

$$c_3 = 0$$

$$c_4 = 1$$

$$c_3 = 1$$

$$\begin{array}{r}
 (+7) \\
 + (-2) \\
 \hline
 (+5)
 \end{array}
 \quad
 \begin{array}{r}
 + \\
 \boxed{11}100 \\
 0111 \\
 1110 \\
 \hline
 10101
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (-2) \\
 \hline
 (-9)
 \end{array}
 \quad
 \begin{array}{r}
 + \\
 \boxed{10}000 \\
 1001 \\
 1110 \\
 \hline
 10111
 \end{array}$$

$$c_4 = 1$$

$$c_3 = 0$$

Overflow occurs only in these two cases.

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array}$$

$$\begin{array}{r} \boxed{01}100 \\ + 0111 \\ \hline 0010 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array}$$

$$\begin{array}{r} \boxed{00}000 \\ + 1001 \\ \hline 0010 \\ \hline 1011 \end{array}$$

$$c_4 = 0$$

$$c_3 = 0$$

$$c_4 = 1$$

$$c_3 = 1$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array}$$

$$\begin{array}{r} \boxed{11}100 \\ + 0111 \\ \hline 1110 \\ \hline 10101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array}$$

$$\begin{array}{r} \boxed{10}000 \\ + 1001 \\ \hline 1110 \\ \hline 10111 \end{array}$$

$$c_4 = 1$$

$$c_3 = 0$$

$$\text{Overflow} = c_3 \bar{c}_4 + \bar{c}_3 c_4$$

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array}$$

$$\begin{array}{r} \boxed{01}100 \\ + \quad 0111 \\ \hline 0010 \\ + \quad 0010 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array}$$

$$\begin{array}{r} \boxed{00}000 \\ + \quad 1001 \\ \hline 0010 \\ + \quad 0010 \\ \hline 1011 \end{array}$$

$$c_4 = 0$$

$$c_3 = 0$$

$$c_4 = 1$$

$$c_3 = 1$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array}$$

$$\begin{array}{r} \boxed{11}100 \\ + \quad 0111 \\ \hline 1110 \\ + \quad 1110 \\ \hline 10101 \end{array}$$


$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array}$$

$$\begin{array}{r} \boxed{10}000 \\ + \quad 1001 \\ \hline 1110 \\ + \quad 1110 \\ \hline 10111 \end{array}$$

$$c_4 = 1$$

$$c_3 = 0$$

$$\text{Overflow} = c_3 \bar{c}_4 + \bar{c}_3 c_4$$



XOR

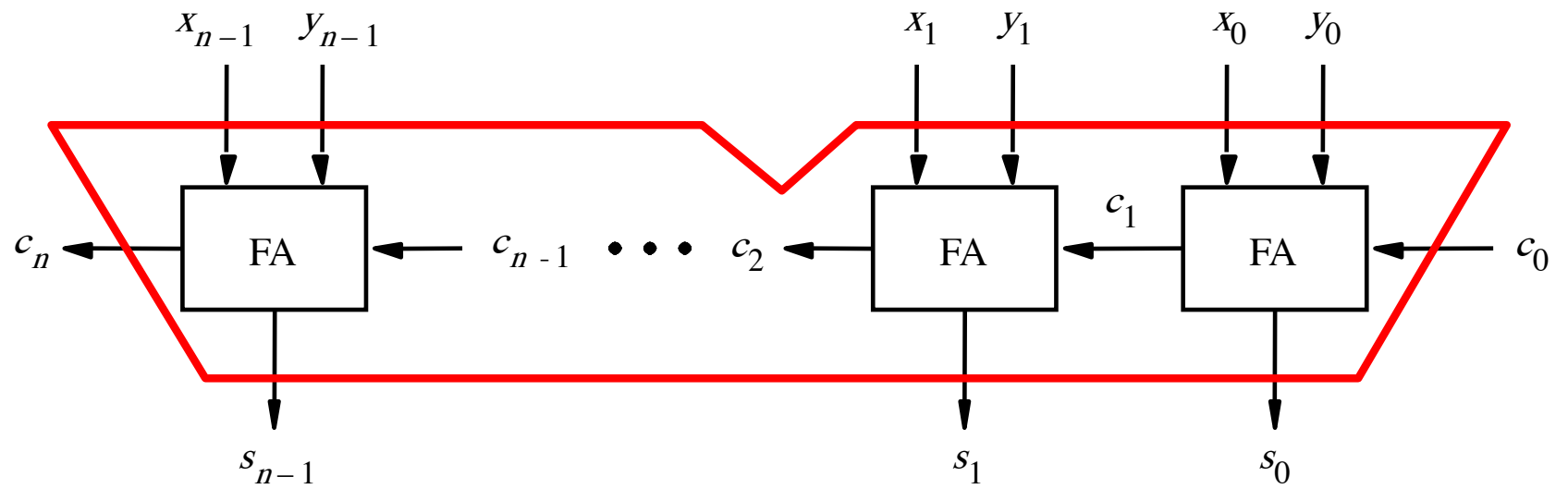
Calculating overflow for 4-bit numbers with only three significant bits

$$\begin{aligned}\text{Overflow} &= c_3 \bar{c}_4 + \bar{c}_3 c_4 \\ &= c_3 \oplus c_4\end{aligned}$$

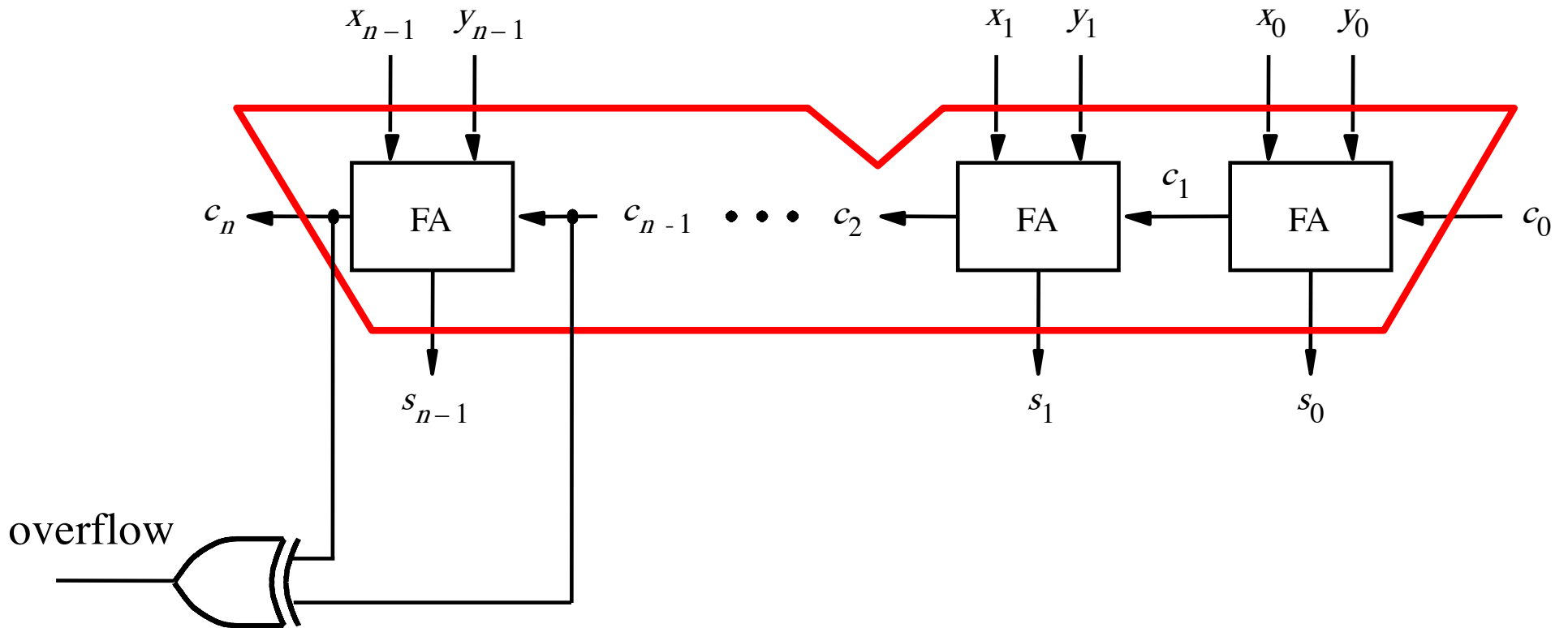
Calculating overflow for n-bit numbers with only n-1 significant bits

$$\text{Overflow} = c_{n-1} \oplus c_n$$

Detecting Overflow



Detecting Overflow (with one extra XOR)



Another way to look at the overflow issue

$$\begin{array}{r} + \\ X = x_3 \ x_2 \ x_1 \ x_0 \\ Y = y_3 \ y_2 \ y_1 \ y_0 \end{array}$$

$$S = s_3 \ s_2 \ s_1 \ s_0$$

Another way to look at the overflow issue

$$\begin{array}{r} + \quad X = \quad \boxed{x_3} \quad x_2 \quad x_1 \quad x_0 \\ \quad Y = \quad \boxed{y_3} \quad y_2 \quad y_1 \quad y_0 \\ \hline \quad S = \quad \boxed{s_3} \quad s_2 \quad s_1 \quad s_0 \end{array}$$

If both numbers that we are adding have the same sign but the sum does not, then we have an overflow.

Examples of determination of overflow

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array} \quad + \quad \begin{array}{r} 0111 \\ 0010 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array} \quad + \quad \begin{array}{r} 1001 \\ 0010 \\ \hline 1011 \end{array}$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array} \quad + \quad \begin{array}{r} 0111 \\ 1110 \\ \hline 10101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array} \quad + \quad \begin{array}{r} 1001 \\ 1110 \\ \hline 10111 \end{array}$$

Examples of determination of overflow

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array} \quad + \quad \begin{array}{r} \boxed{0}111 \\ 0010 \\ \hline \boxed{1}001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array} \quad + \quad \begin{array}{r} \boxed{1}001 \\ 0010 \\ \hline \boxed{1}011 \end{array}$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array} \quad + \quad \begin{array}{r} \boxed{0}111 \\ 1110 \\ \hline 1\boxed{0}101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array} \quad + \quad \begin{array}{r} \boxed{1}001 \\ 1110 \\ \hline 1\boxed{0}111 \end{array}$$

Examples of determination of overflow

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array} \quad + \quad \begin{array}{r} \boxed{0}111 \\ 0010 \\ \hline \boxed{1}001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array}$$

$$\begin{array}{r} \boxed{1}001 \\ 0010 \\ \hline \boxed{1}011 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array} \quad + \quad \begin{array}{r} \boxed{0}111 \\ 1110 \\ \hline \boxed{1}0101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array}$$

$$\begin{array}{r} \boxed{1}001 \\ 1110 \\ \hline \boxed{1}0111 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

Examples of determination of overflow

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array} \quad + \quad \begin{array}{r} \boxed{0}111 \\ \boxed{0}010 \\ \hline \boxed{1}001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array}$$

$$\begin{array}{r} \boxed{1}001 \\ \boxed{0}010 \\ \hline \boxed{1}011 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array} \quad + \quad \begin{array}{r} \boxed{0}111 \\ \boxed{1}110 \\ \hline \boxed{1}0101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array}$$

$$\begin{array}{r} \boxed{1}001 \\ \boxed{1}110 \\ \hline \boxed{1}0111 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

In 2's complement, both +9 and -9 are not representable with 4 bits.

Examples of determination of overflow

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array} \quad + \quad \begin{array}{r} 0111 \\ 0010 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array} \quad + \quad \begin{array}{r} 1001 \\ 0010 \\ \hline 1011 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array} \quad + \quad \begin{array}{r} 0111 \\ 1110 \\ \hline 10101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array} \quad + \quad \begin{array}{r} 1001 \\ 1110 \\ \hline 10111 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

Overflow occurs only in these two cases.

Examples of determination of overflow

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (+2) \\ \hline (+9) \end{array}$$

$$+ \begin{array}{r} 0111 \\ 0010 \\ \hline 1001 \end{array}$$

$$\begin{array}{r} (-7) \\ + (+2) \\ \hline (-5) \end{array}$$

$$+ \begin{array}{r} 1001 \\ 0010 \\ \hline 1011 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 0 \\ s_3 &= 1 \end{aligned}$$

$$\begin{aligned} x_3 &= 0 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

$$\begin{array}{r} (+7) \\ + (-2) \\ \hline (+5) \end{array}$$

$$+ \begin{array}{r} 0111 \\ 1110 \\ \hline 10101 \end{array}$$

$$\begin{array}{r} (-7) \\ + (-2) \\ \hline (-9) \end{array}$$

$$+ \begin{array}{r} 1001 \\ 1110 \\ \hline 10111 \end{array}$$

$$\begin{aligned} x_3 &= 1 \\ y_3 &= 1 \\ s_3 &= 0 \end{aligned}$$

$$\text{Overflow} = \bar{x}_3 \bar{y}_3 s_3 + x_3 y_3 \bar{s}_3$$

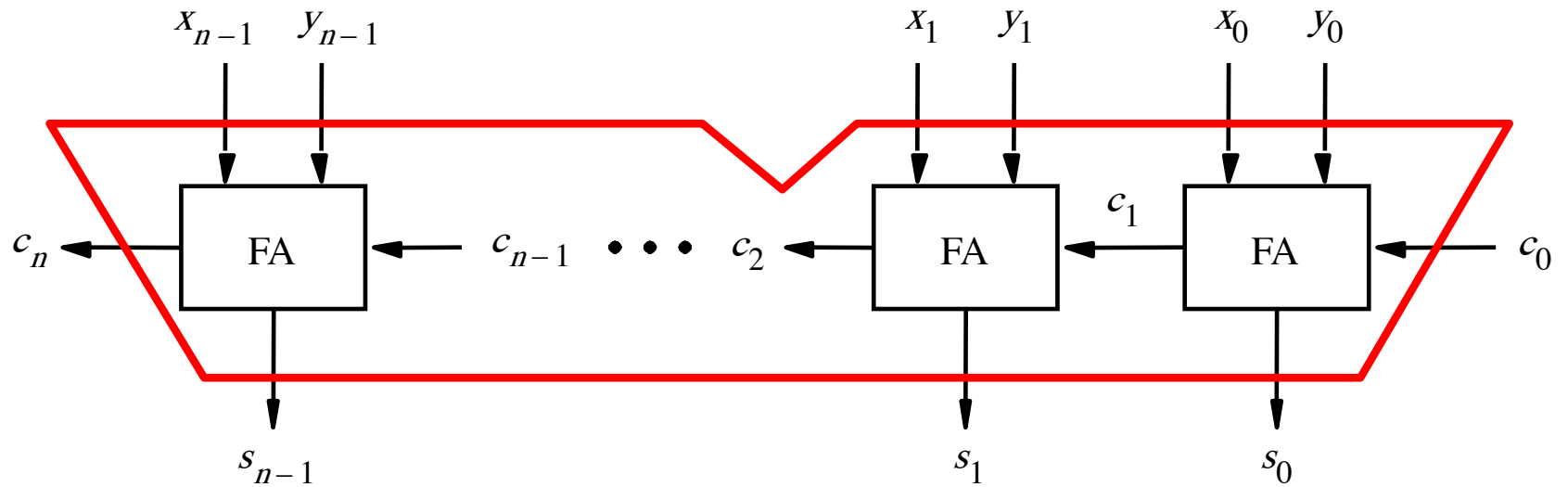
Another way to look at the overflow issue

$$\begin{array}{rcccc} + & X = & x_3 & x_2 & x_1 & x_0 \\ & Y = & y_3 & y_2 & y_1 & y_0 \\ \hline & S = & s_3 & s_2 & s_1 & s_0 \end{array}$$

If both numbers that we are adding have the same sign but the sum does not, then we have an overflow.

$$\text{Overflow} = \bar{x}_3 \bar{y}_3 s_3 + x_3 y_3 \bar{s}_3$$

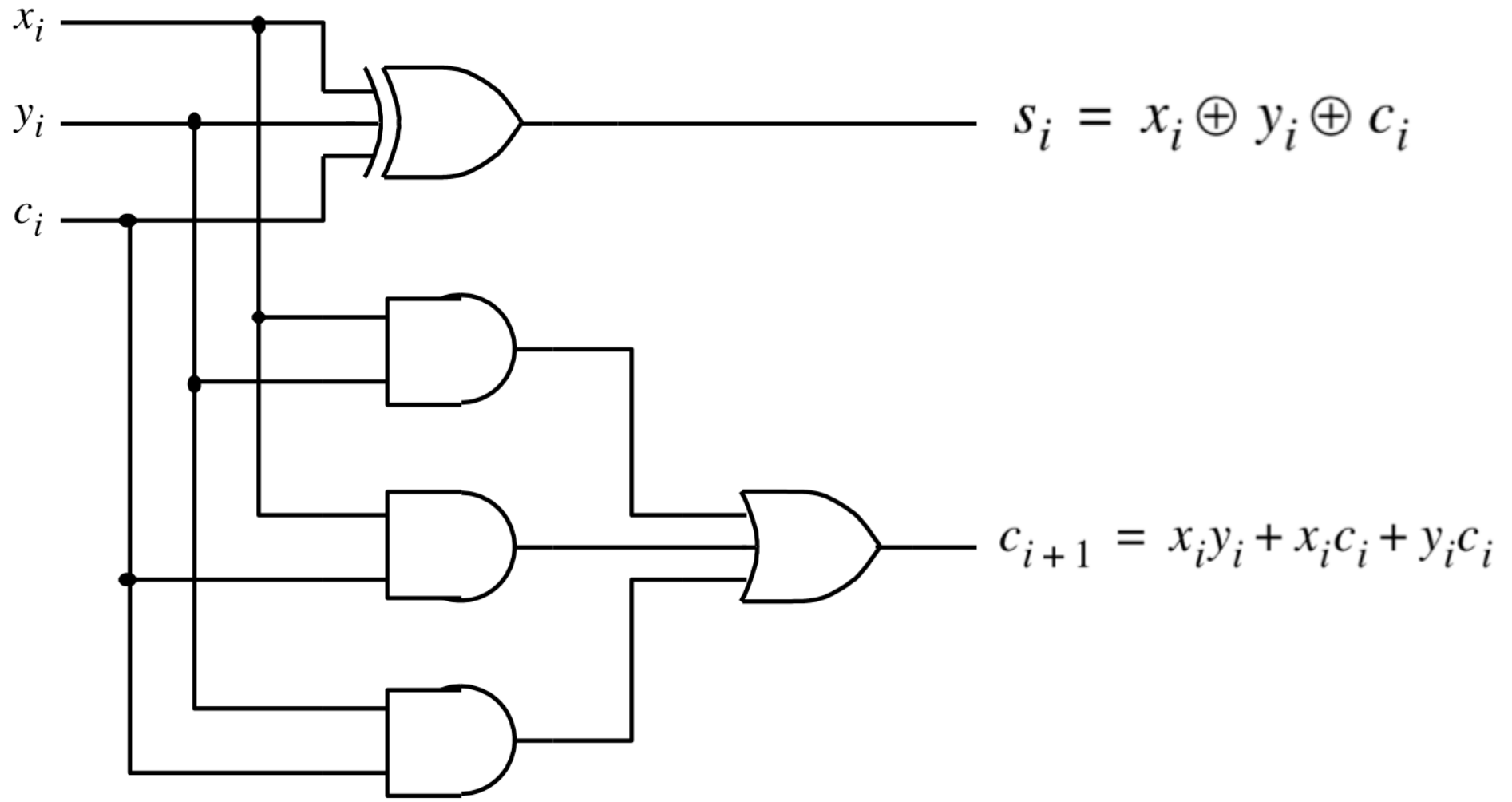
How long does it take to compute all sum bits and all carry bits?



Can we perform addition even faster?

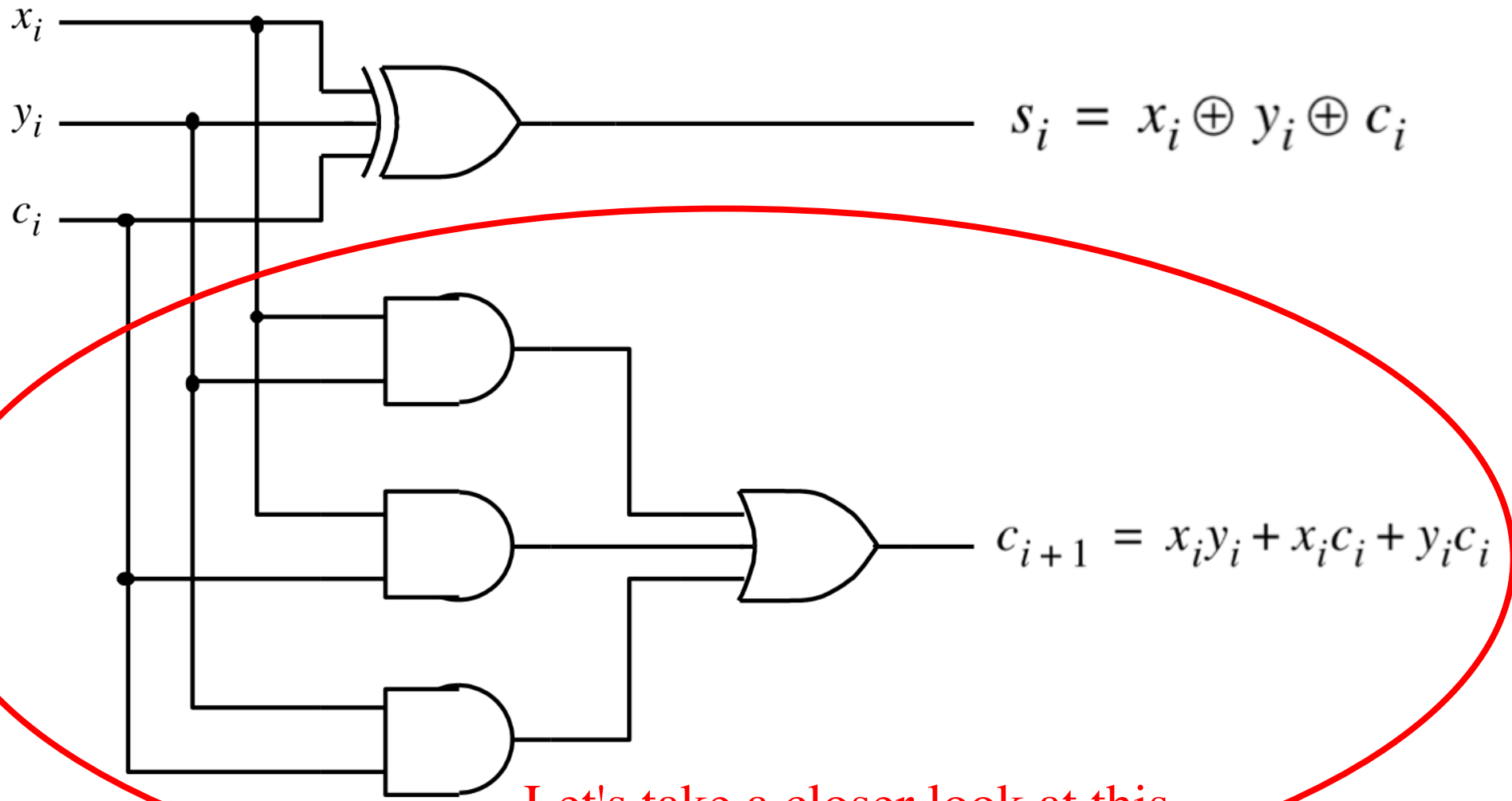
The goal is to evaluate very fast if the carry from the previous stage will be equal to 0 or 1.

The Full-Adder Circuit



[Figure 3.3c from the textbook]

The Full-Adder Circuit



Let's take a closer look at this.

Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

Decomposing the Carry Expression

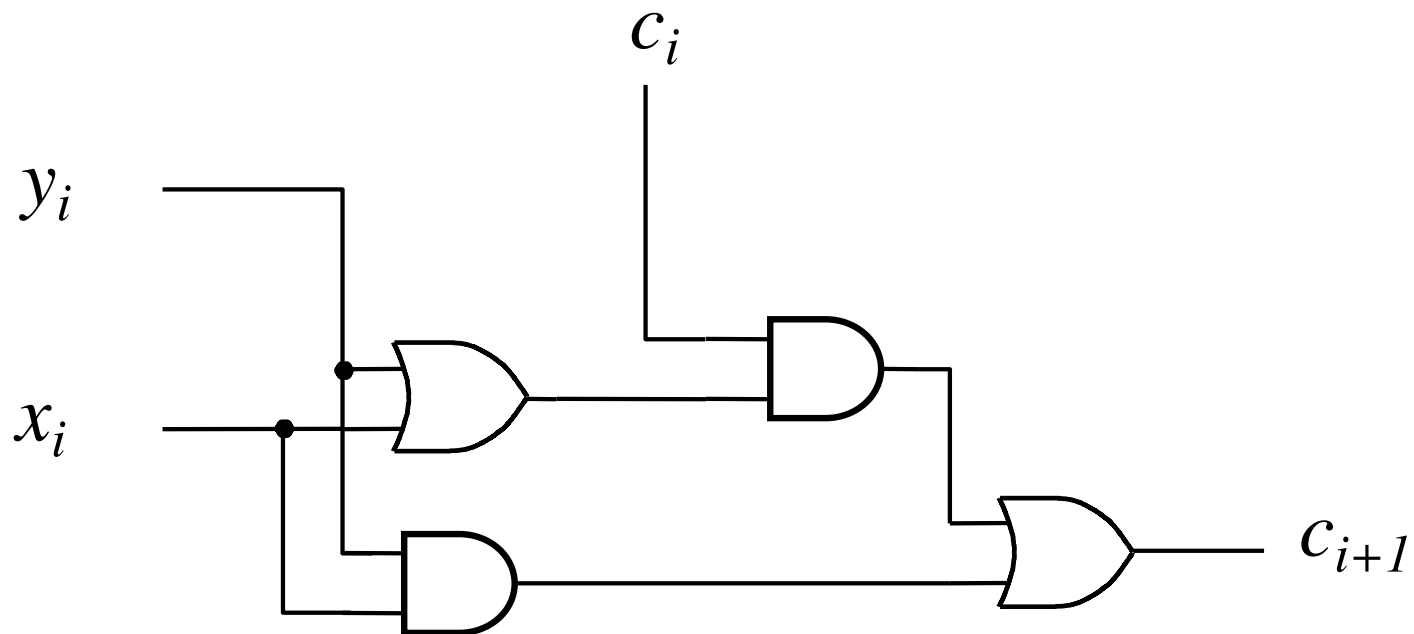
$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$c_{i+1} = x_i y_i + (x_i + y_i) c_i$$

Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

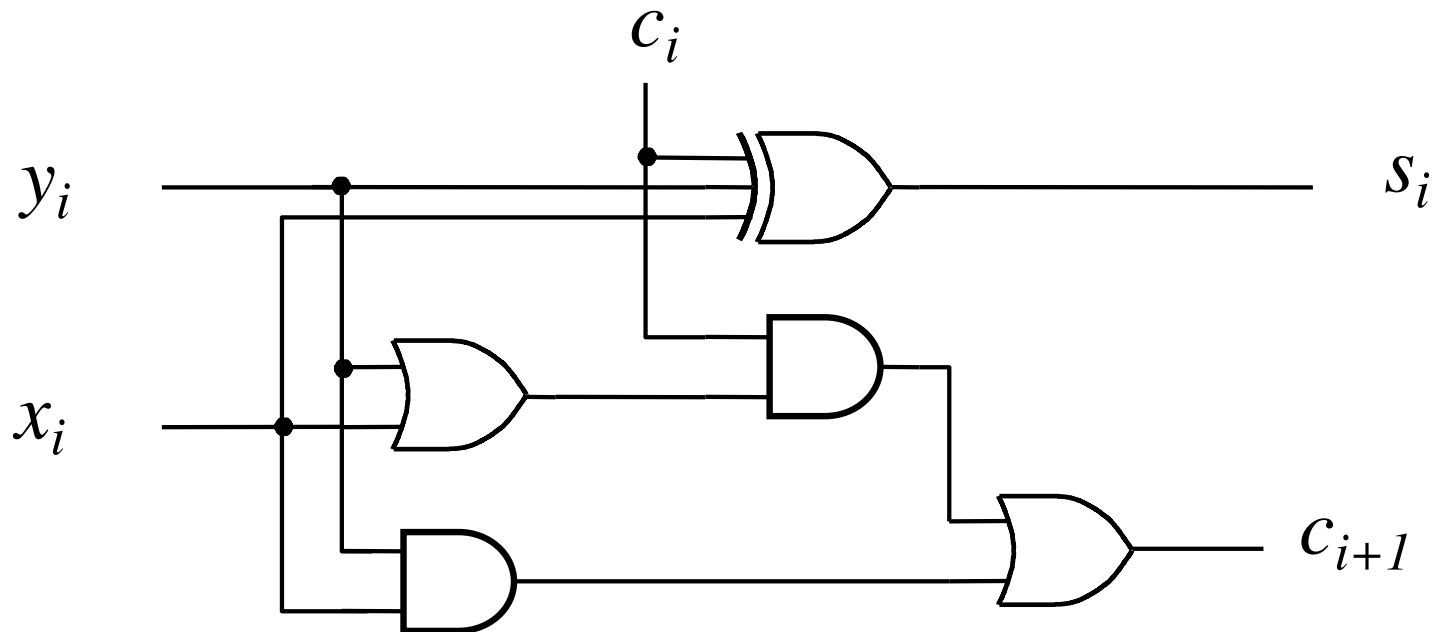
$$c_{i+1} = x_i y_i + (x_i + y_i) c_i$$



Another Way to Draw the Full-Adder Circuit

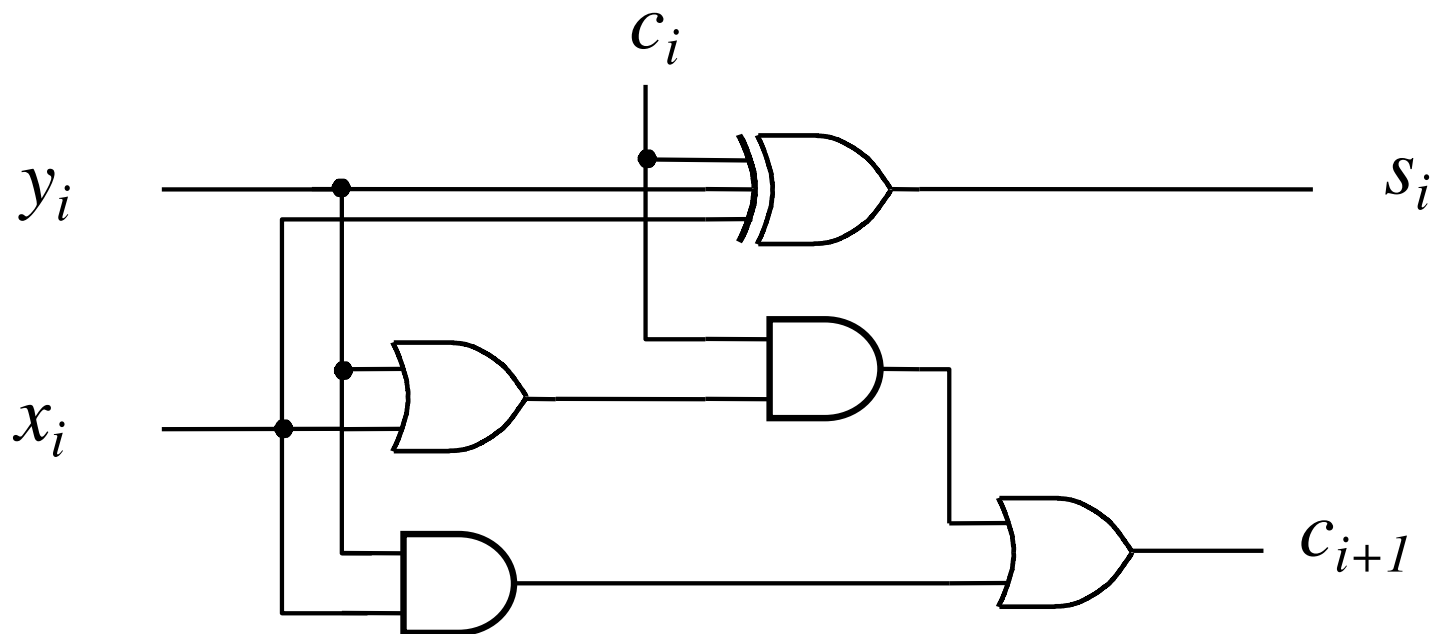
$$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$C_{i+1} = x_i y_i + (x_i + y_i) c_i$$



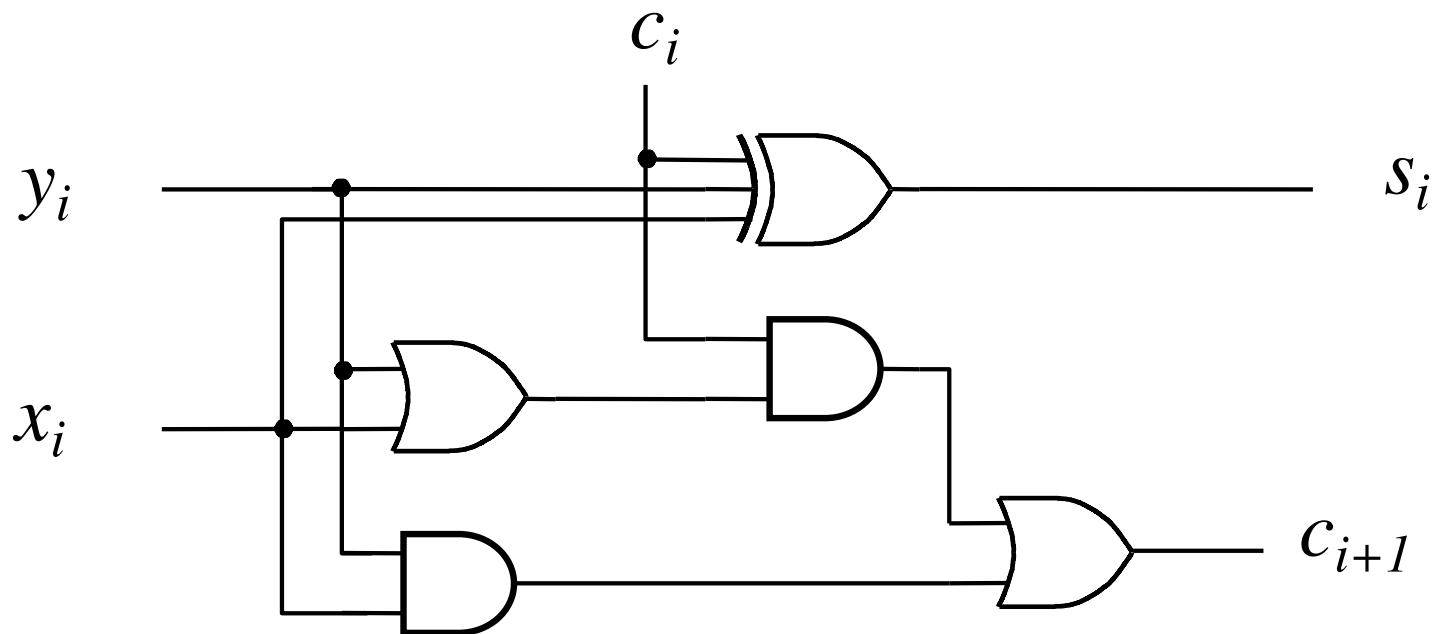
Another Way to Draw the Full-Adder Circuit

$$C_{i+1} = x_i y_i + (x_i + y_i)C_i$$



Another Way to Draw the Full-Adder Circuit

$$C_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} c_i$$

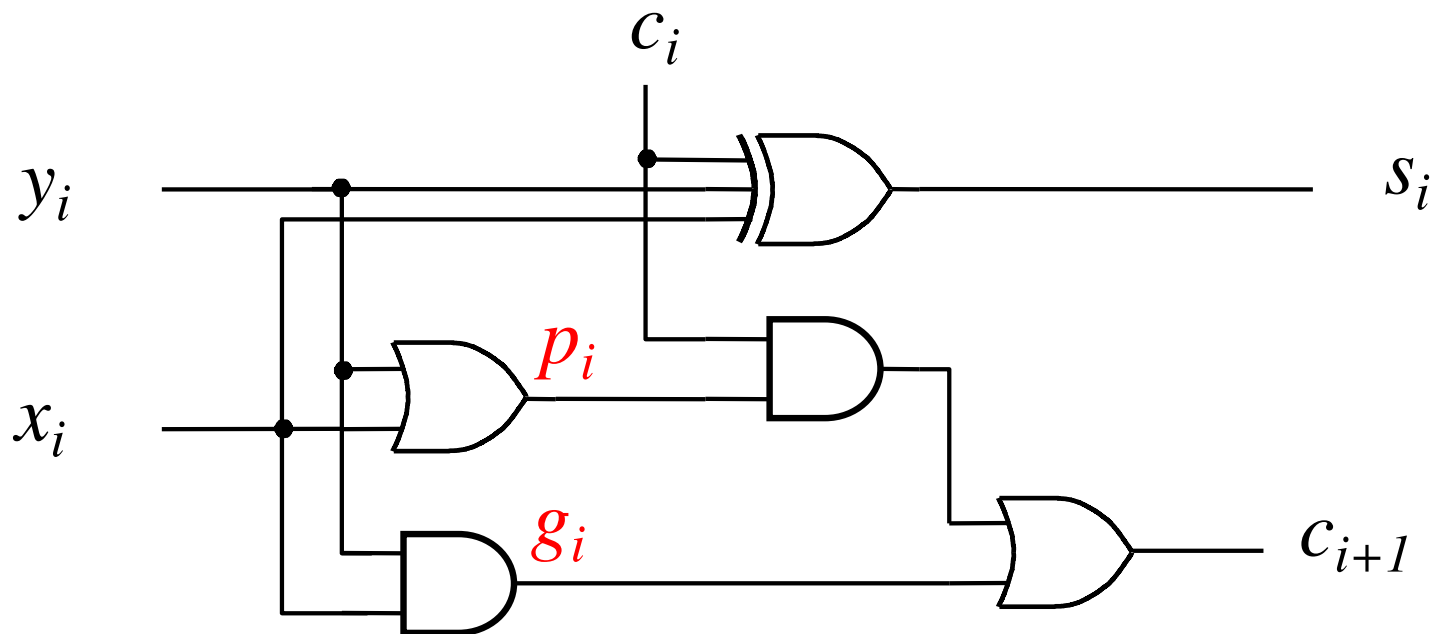


Another Way to Draw the Full-Adder Circuit

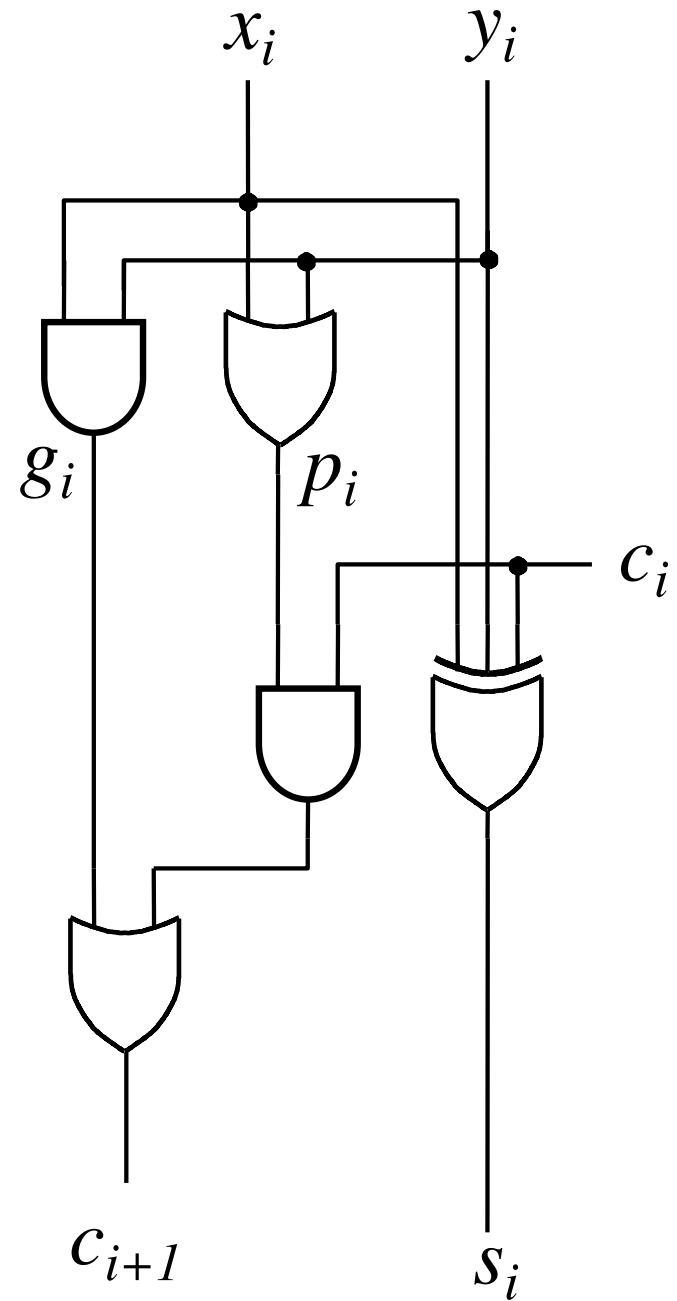
g - generate

p - propagate

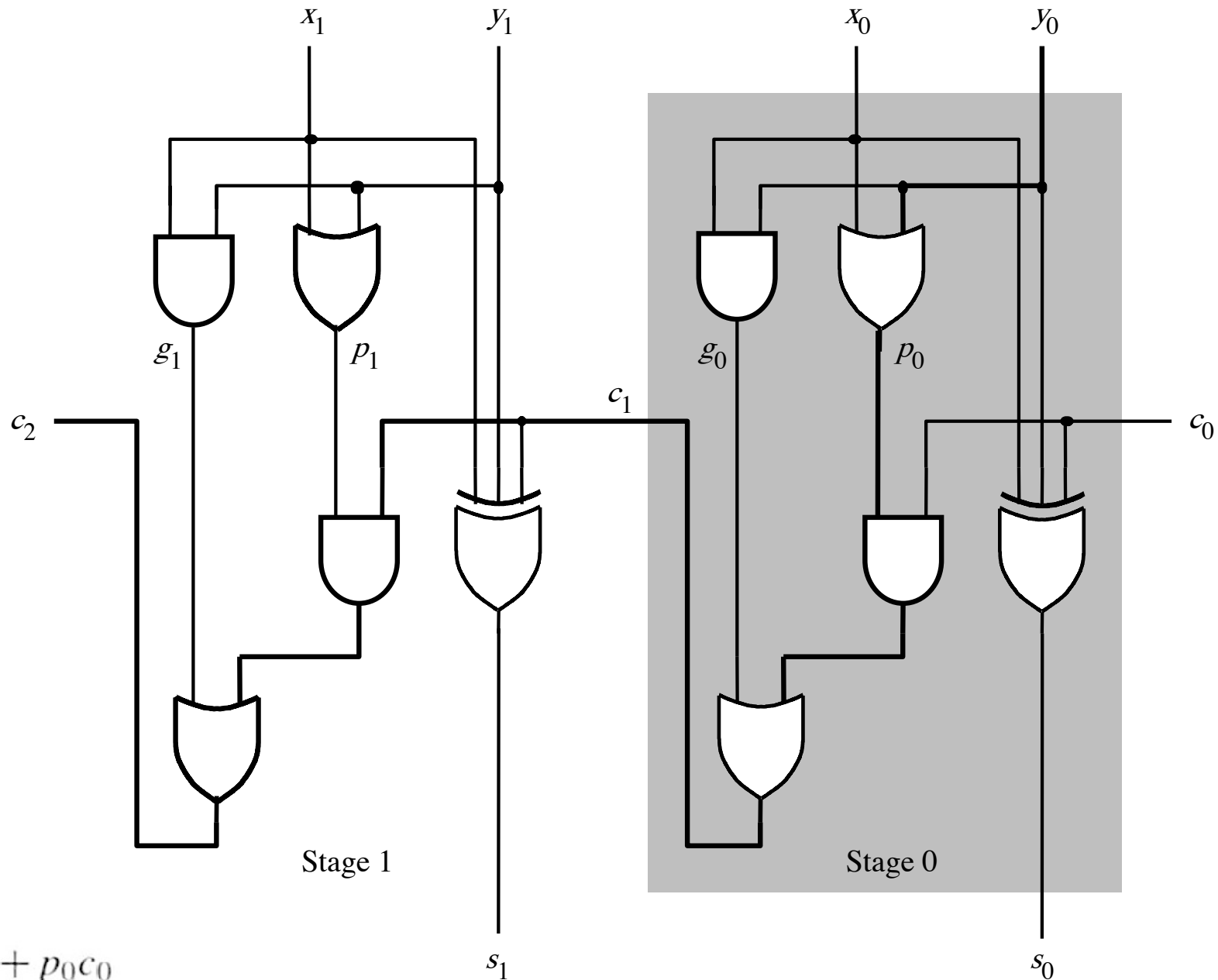
$$C_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} C_i$$



Yet Another Way to Draw It (Just Rotate It)



Now we can Build a Ripple-Carry Adder

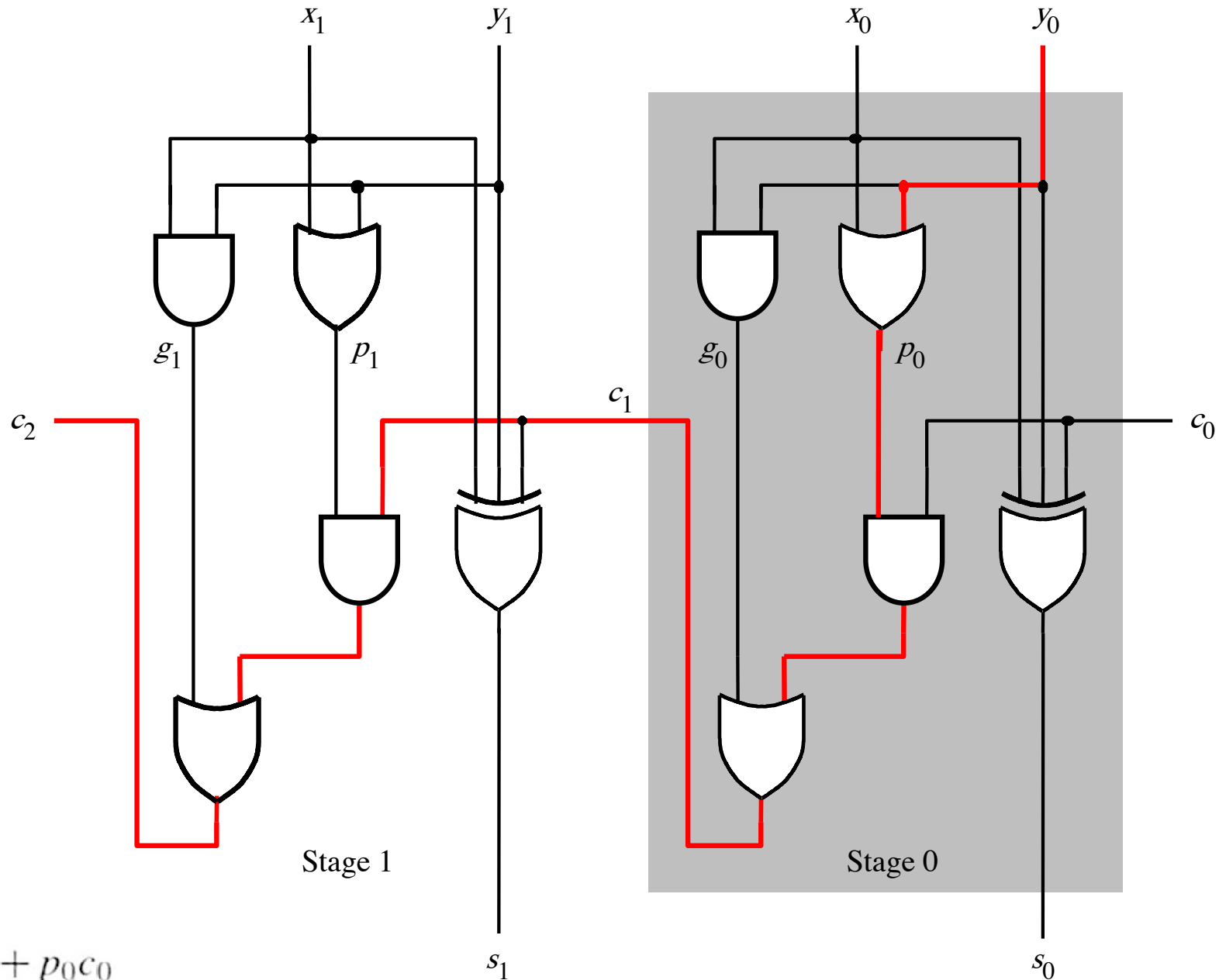


$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

[Figure 3.14 from the textbook]

Now we can Build a Ripple-Carry Adder

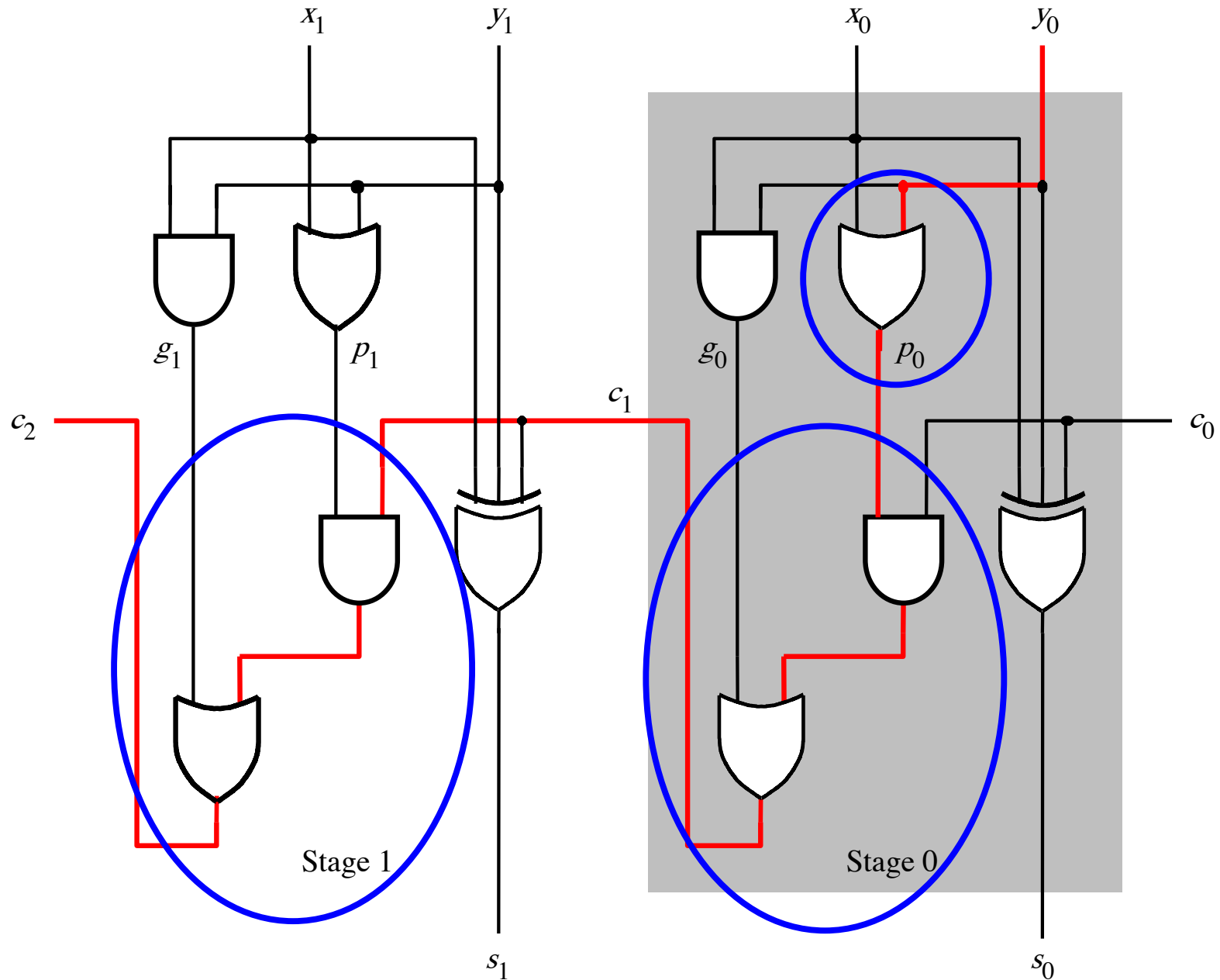


$$c_1 = g_0 + p_0 c_0$$

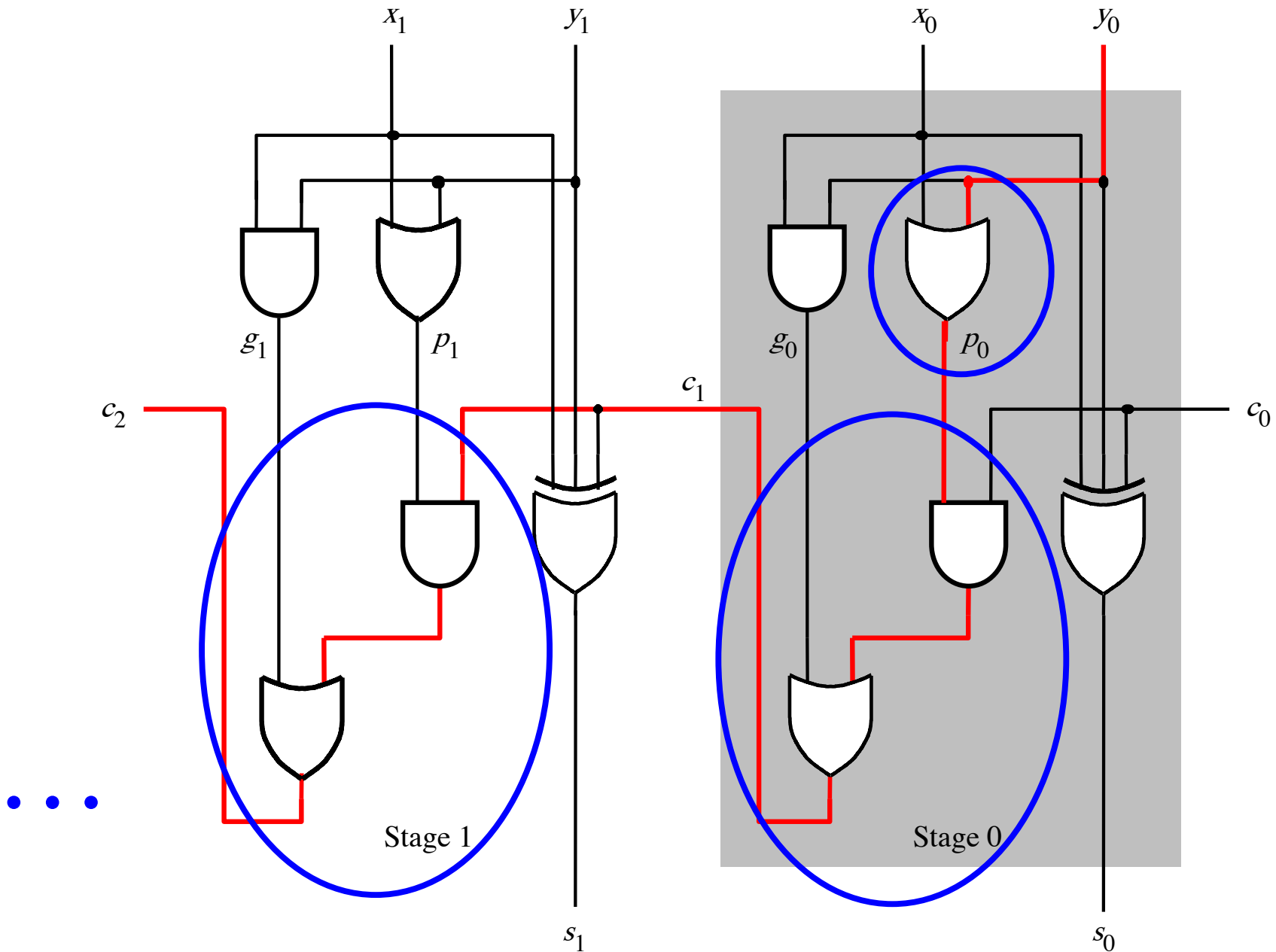
$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

[Figure 3.14 from the textbook]

The delay is 5 gates (1+2+2)



n-bit ripple-carry adder: $2n+1$ gate delays



Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} c_i$$

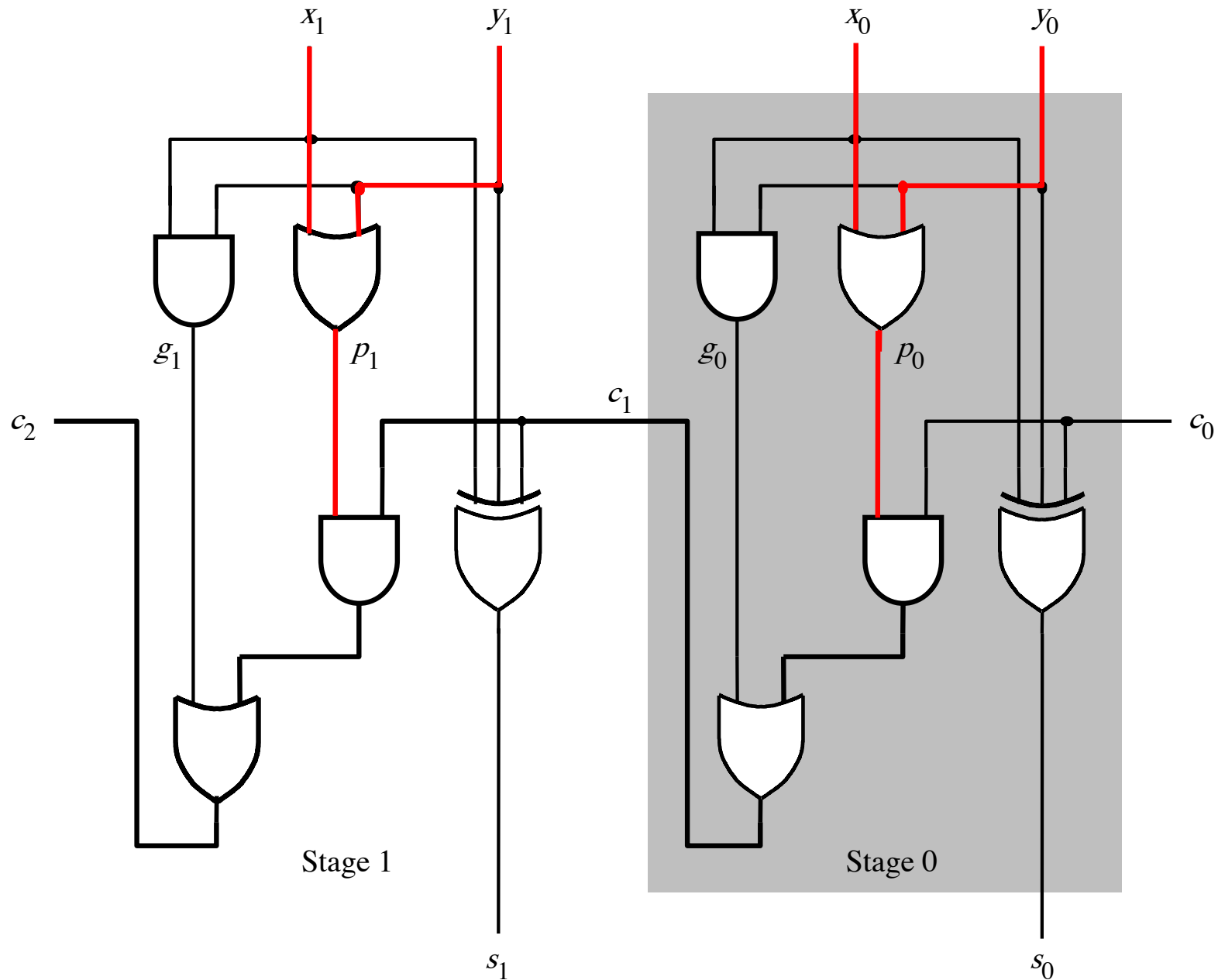
Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{P_i} c_i$$

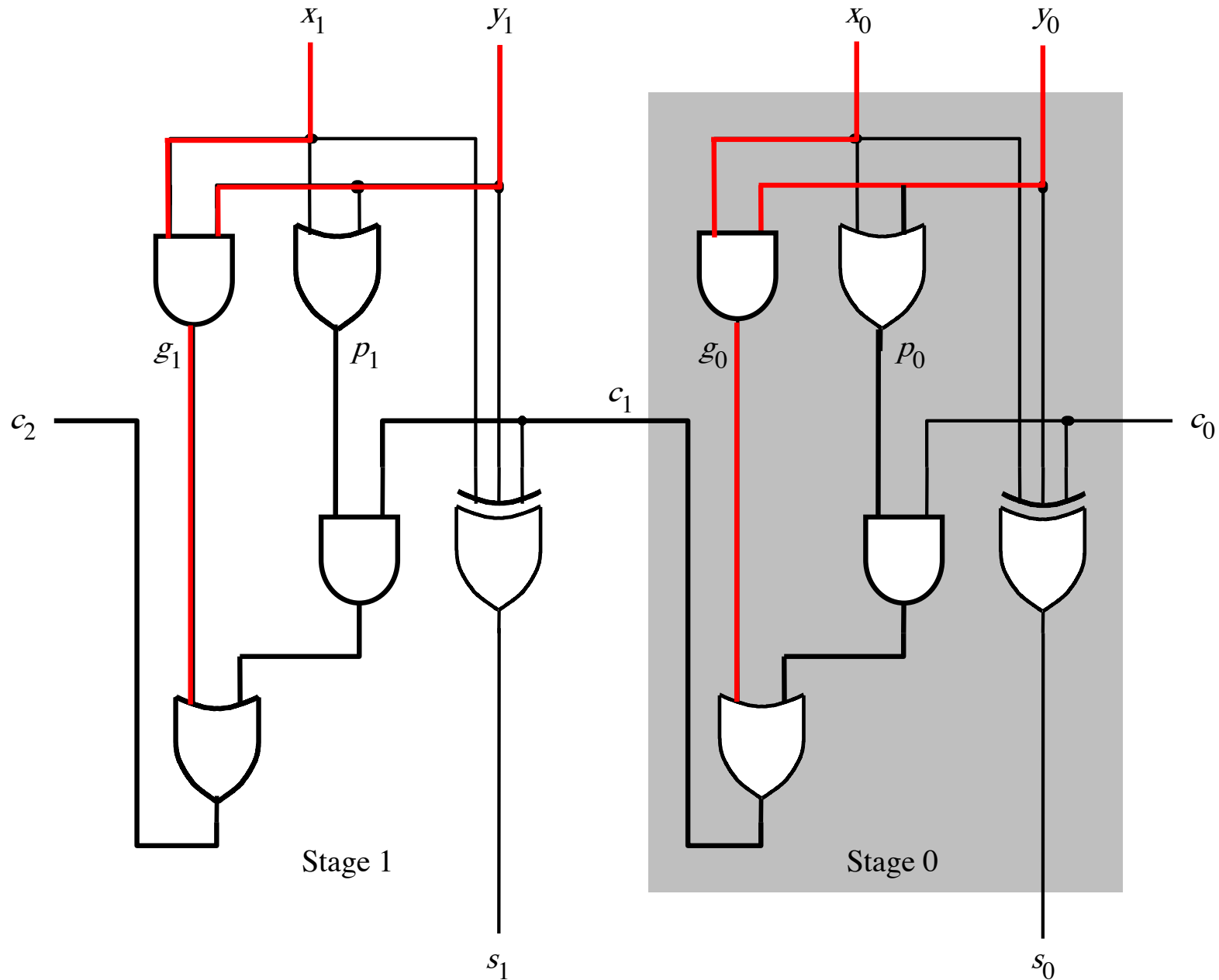
(1 gate delay) (1 gate delay)

It takes 1 gate delay to compute all p_i signals



[Figure 3.14 from the textbook]

It takes 1 gate delay to compute all g_i signals



[Figure 3.14 from the textbook]

Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} c_i$$

Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} c_i$$

$$c_{i+1} = g_i + p_i c_i$$

Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)c_i}_{p_i}$$

$$c_{i+1} = g_i + p_i c_i$$

recursive
expansion of
 c_i

$$c_{i+1} = g_i + p_i (g_{i-1} + p_{i-1} c_{i-1})$$

Decomposing the Carry Expression

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

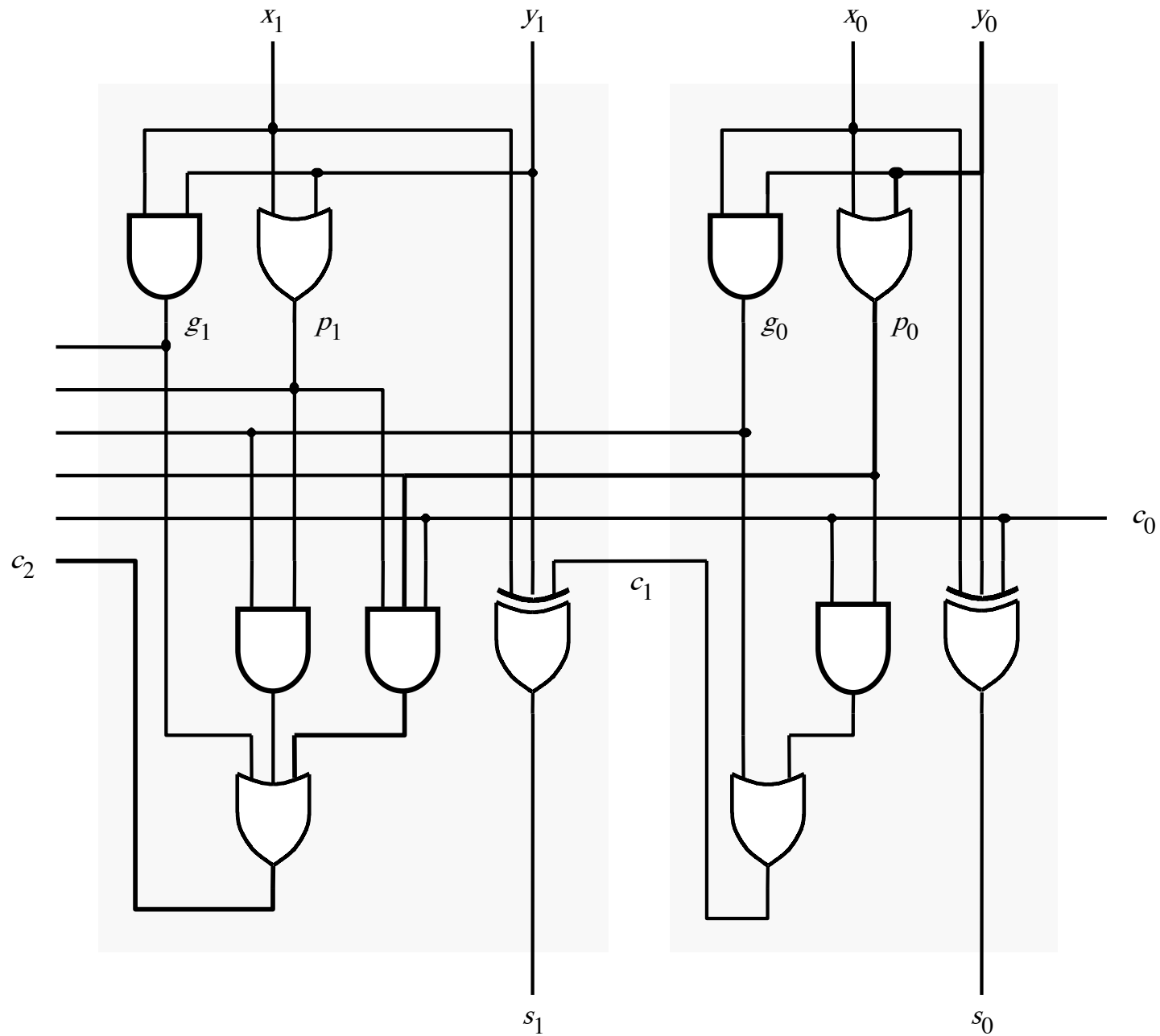
$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} c_i$$

$$c_{i+1} = g_i + p_i c_i$$

$$c_{i+1} = g_i + p_i (g_{i-1} + p_{i-1} c_{i-1})$$

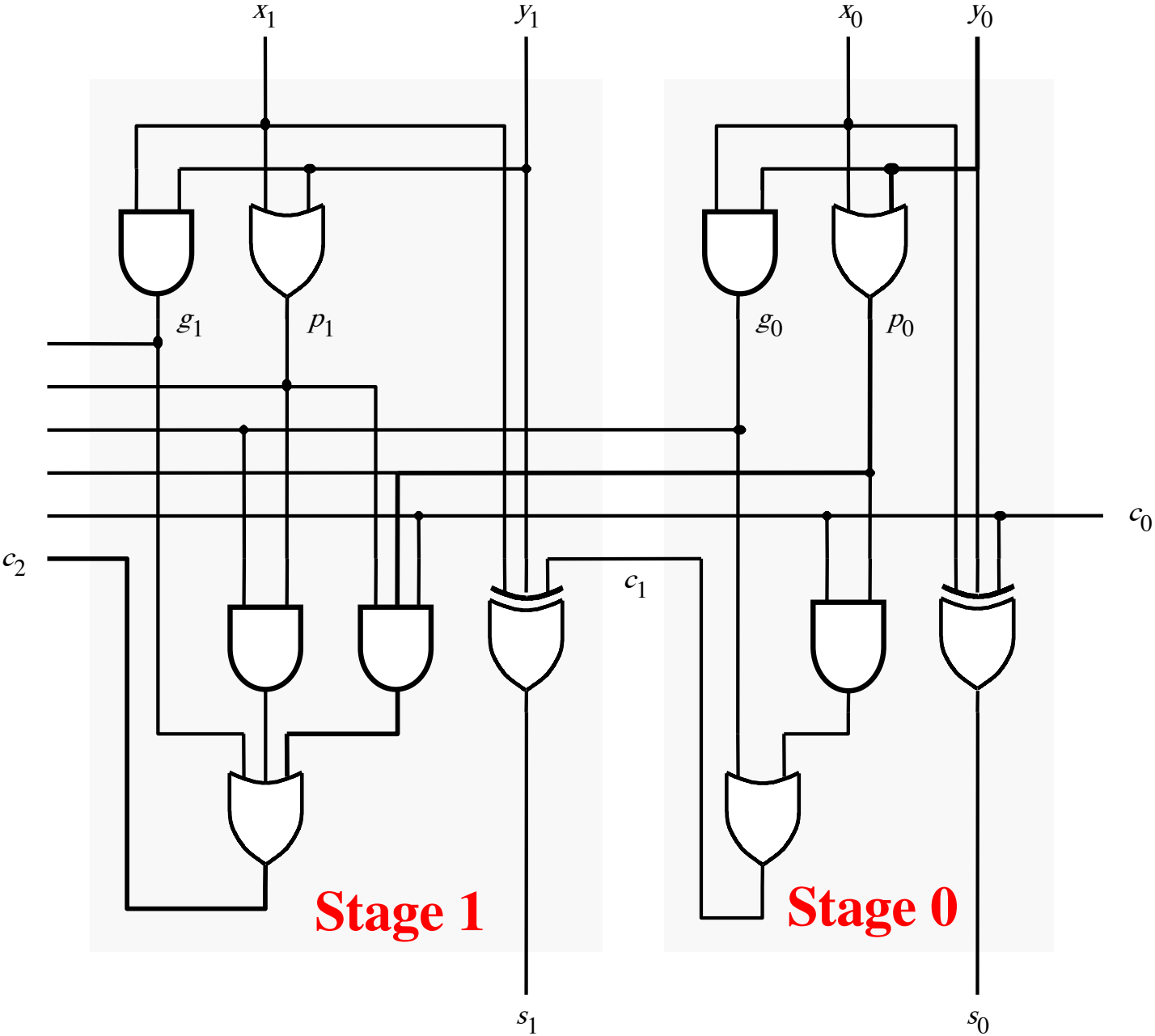
$$c_{i+1} = g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-1}$$

Now we can Build a **Carry-Lookahead Adder**



[Figure 3.15 from the textbook]

The first two stages of a carry-lookahead adder



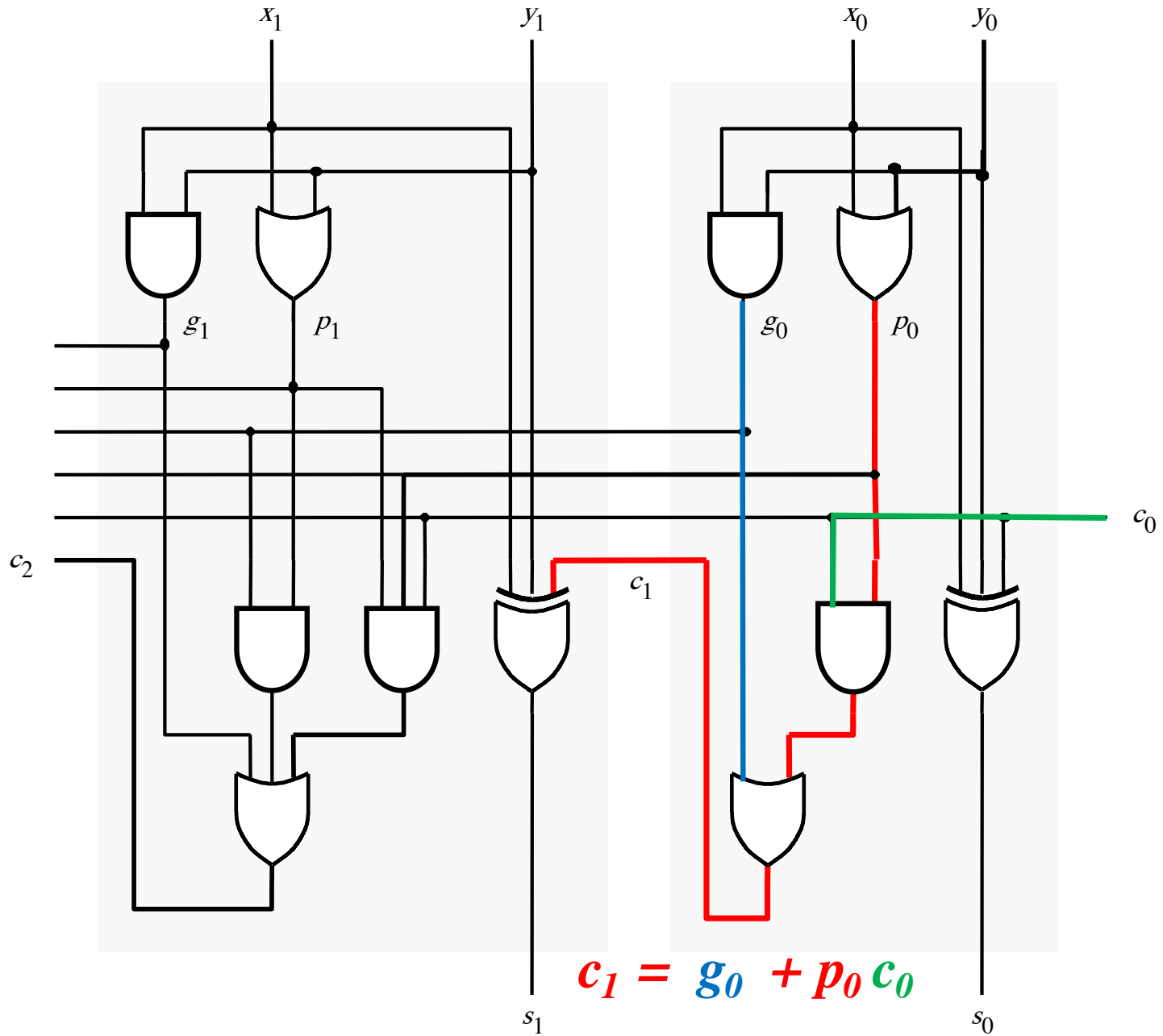
...

[Figure 3.15 from the textbook]

Carry for the first stage

$$c_1 = g_0 + p_0 c_0$$

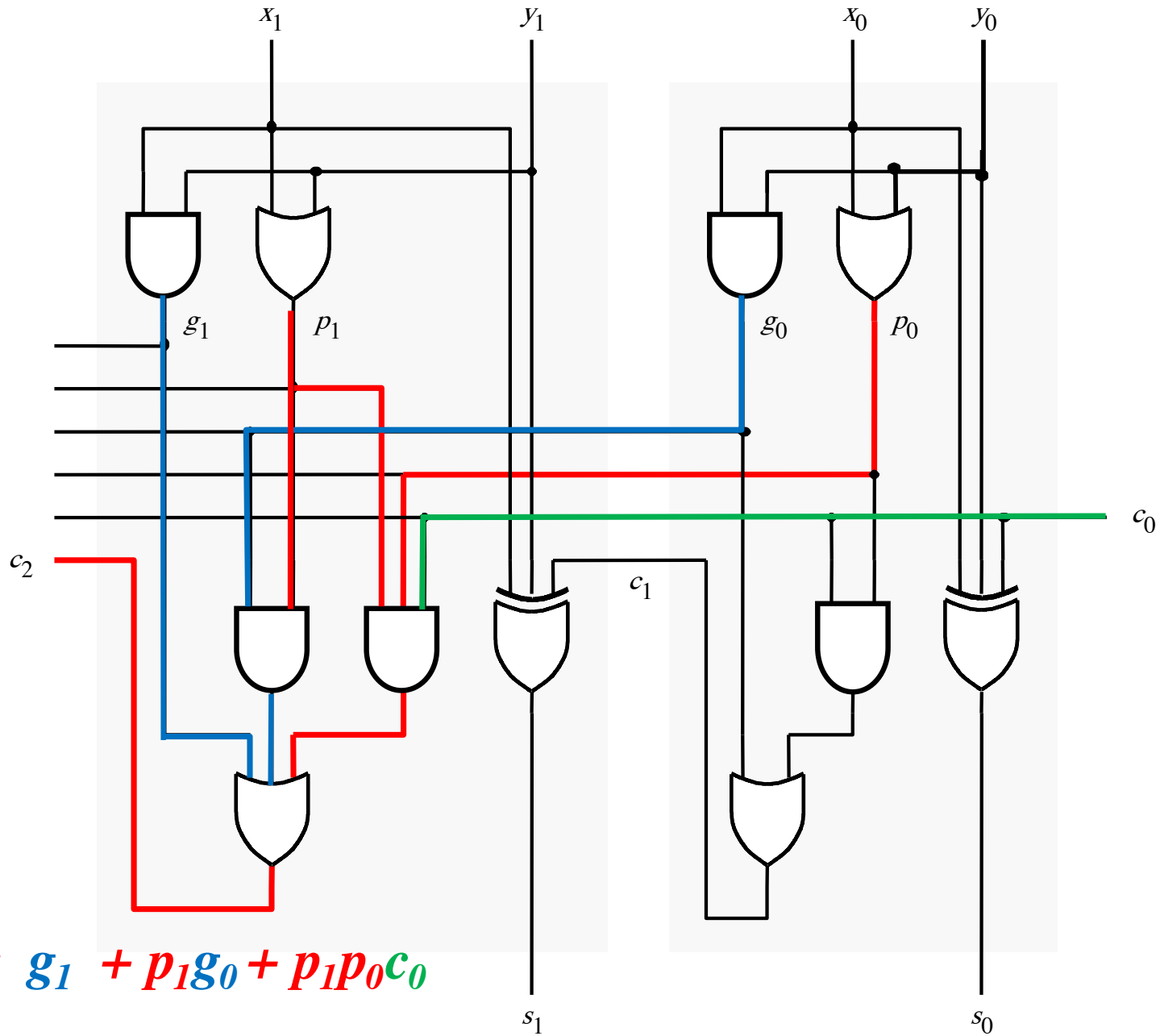
Carry for the first stage



Carry for the second stage

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

Carry for the second stage



Carry for the first two stages

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

Carry for the first two stages

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + \underline{p_1} g_0 + \underline{p_1 p_0} c_0$$

Carry for the first two stages

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + \underline{p_1 g_0} + \underline{p_1 p_0 c_0}$$

$$= g_1 + p_1 \underbrace{(g_0 + p_0 c_0)}_{c_1}$$

Carry for the first two stages

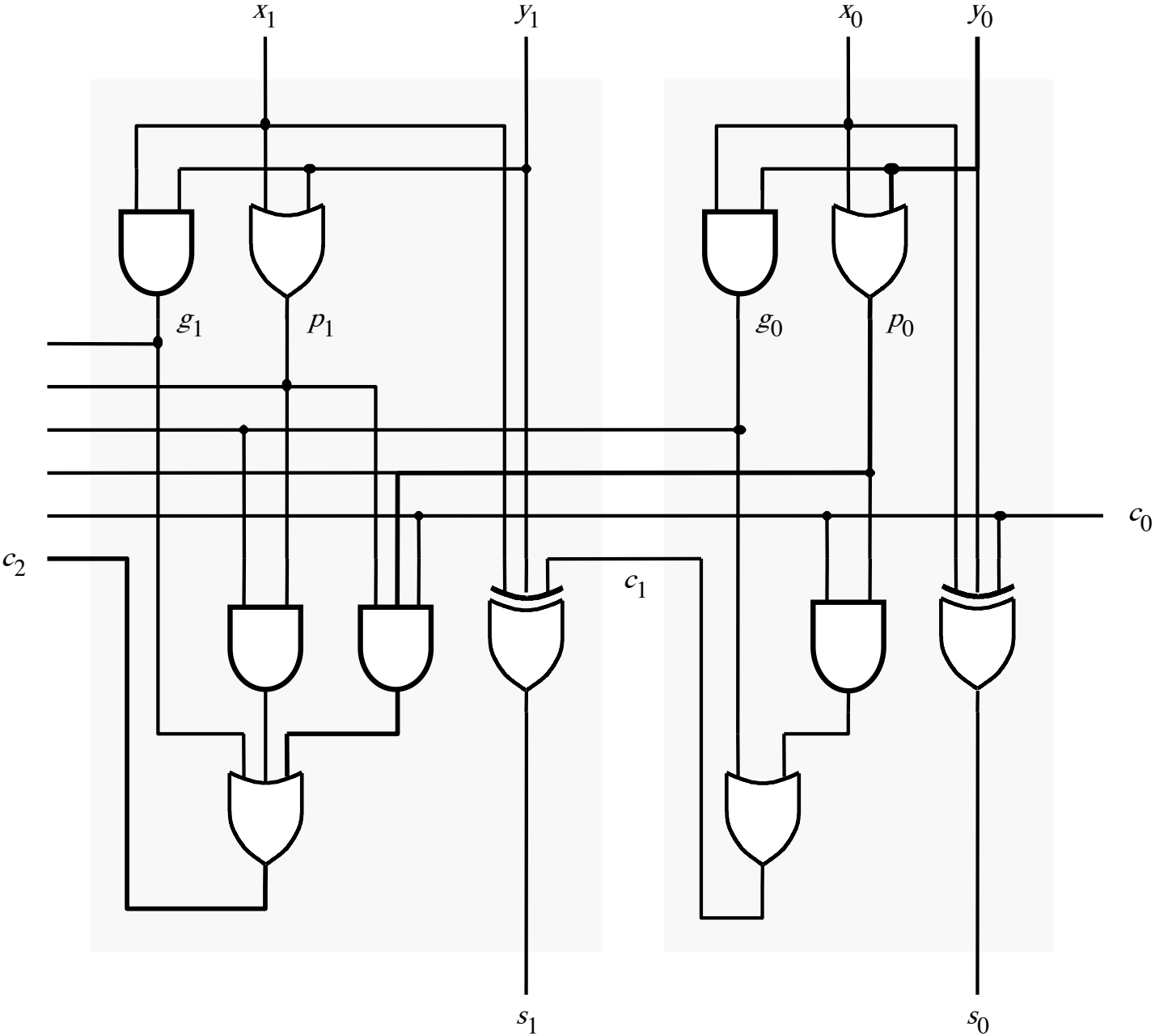
$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

$$= g_1 + p_1 \underbrace{(g_0 + p_0 c_0)}_{c_1}$$

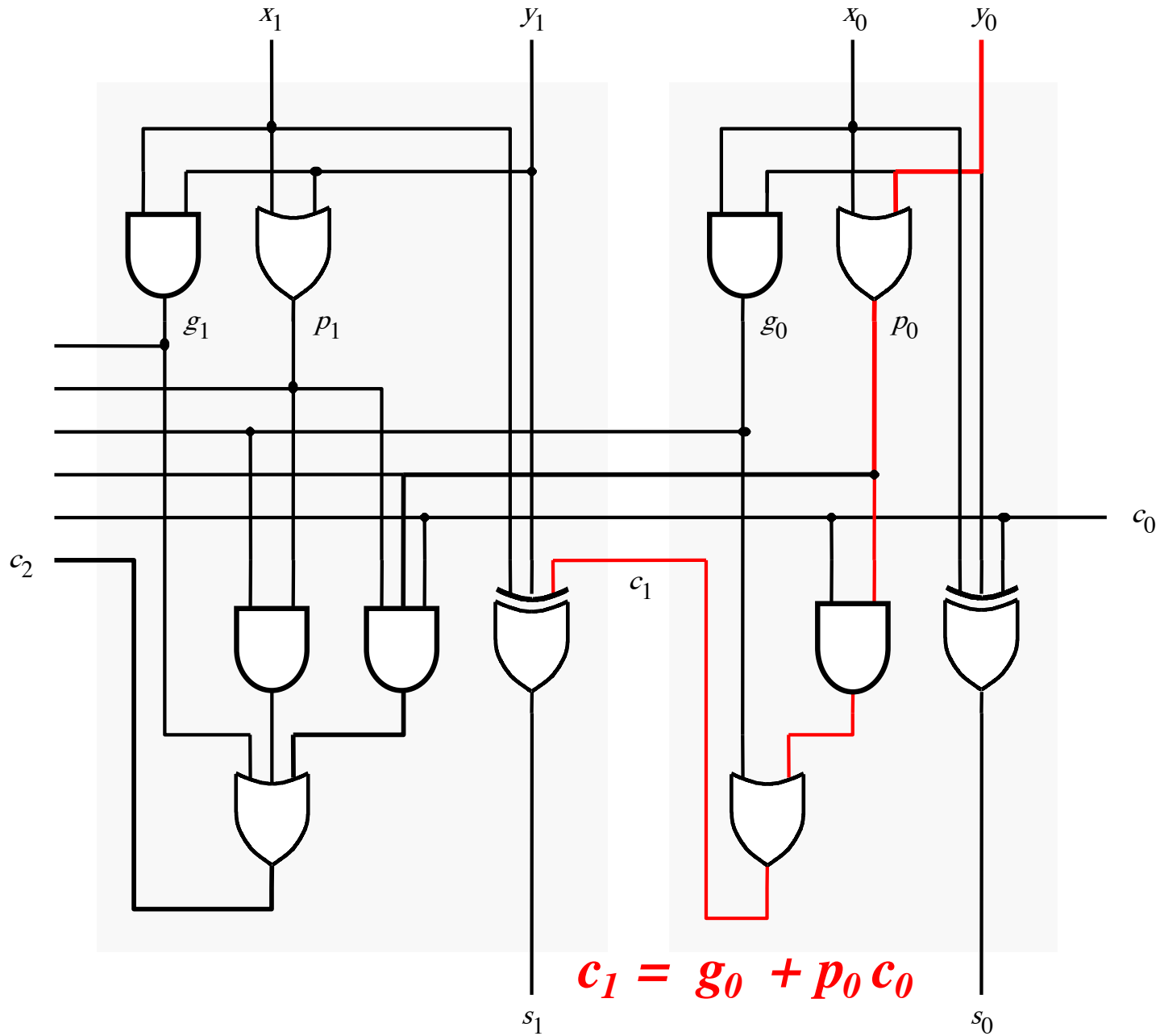
$$= g_1 + p_1 c_1$$

The first two stages of a carry-lookahead adder

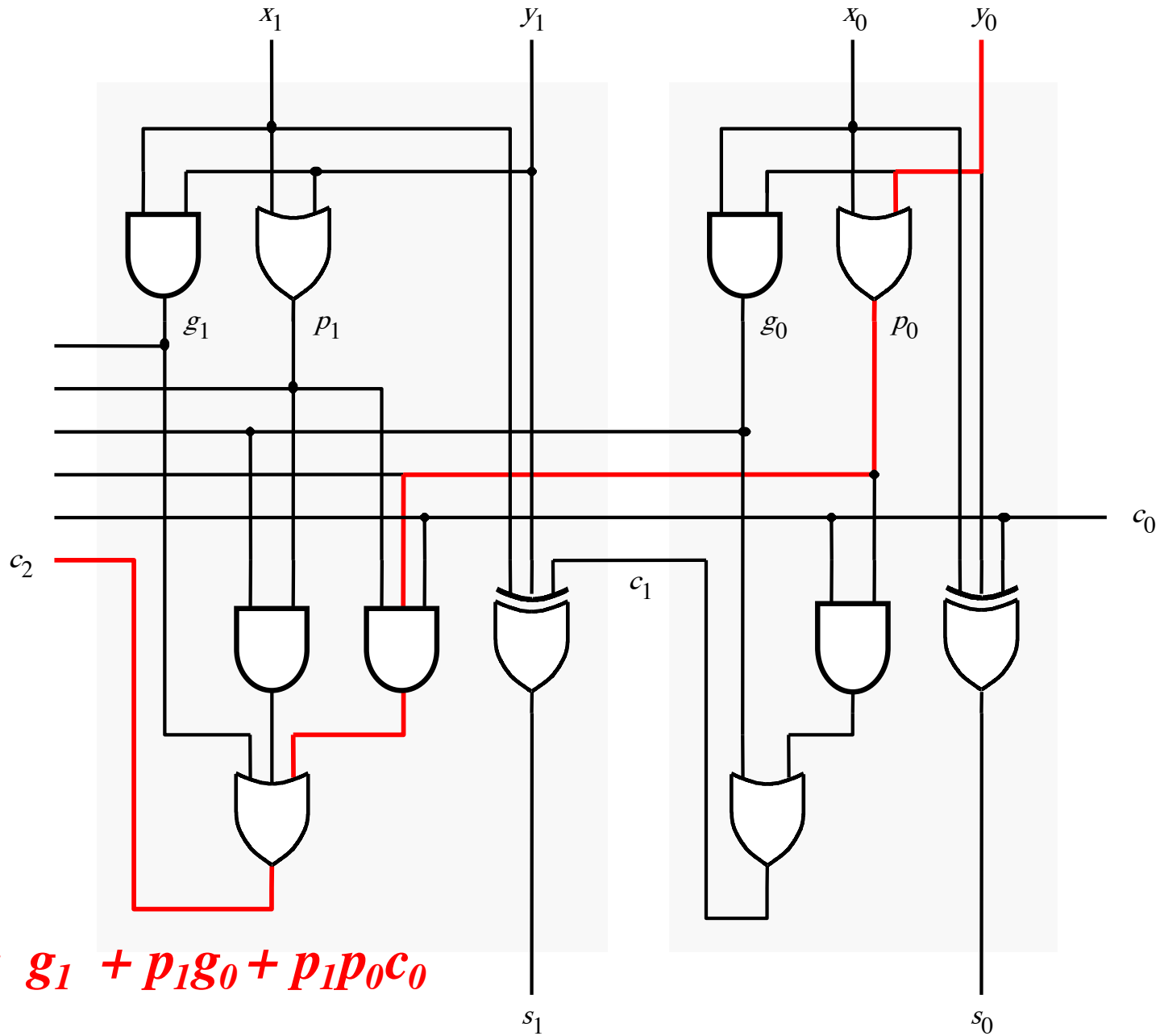


[Figure 3.15 from the textbook]

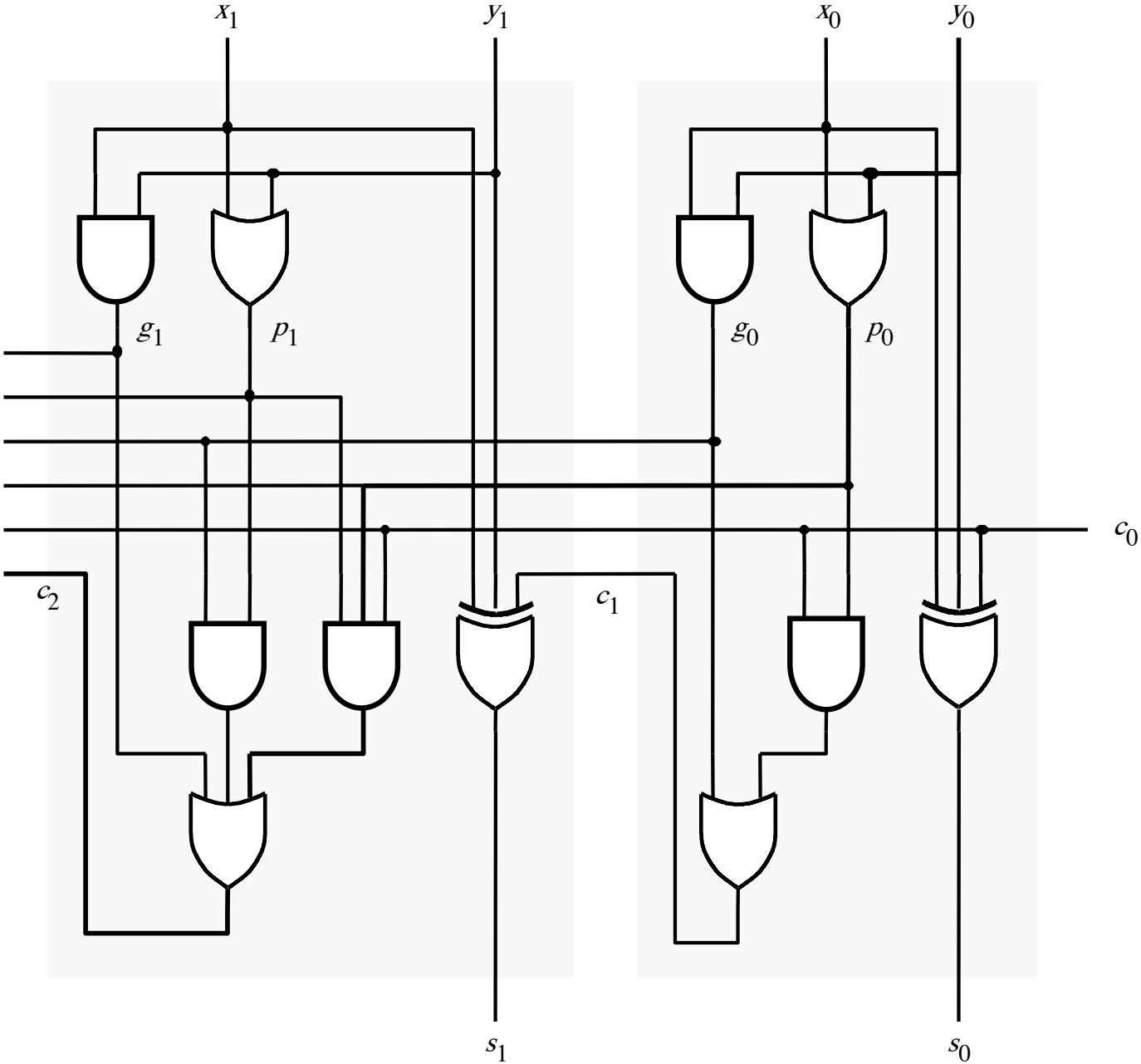
It takes 3 gate delays to generate c_1



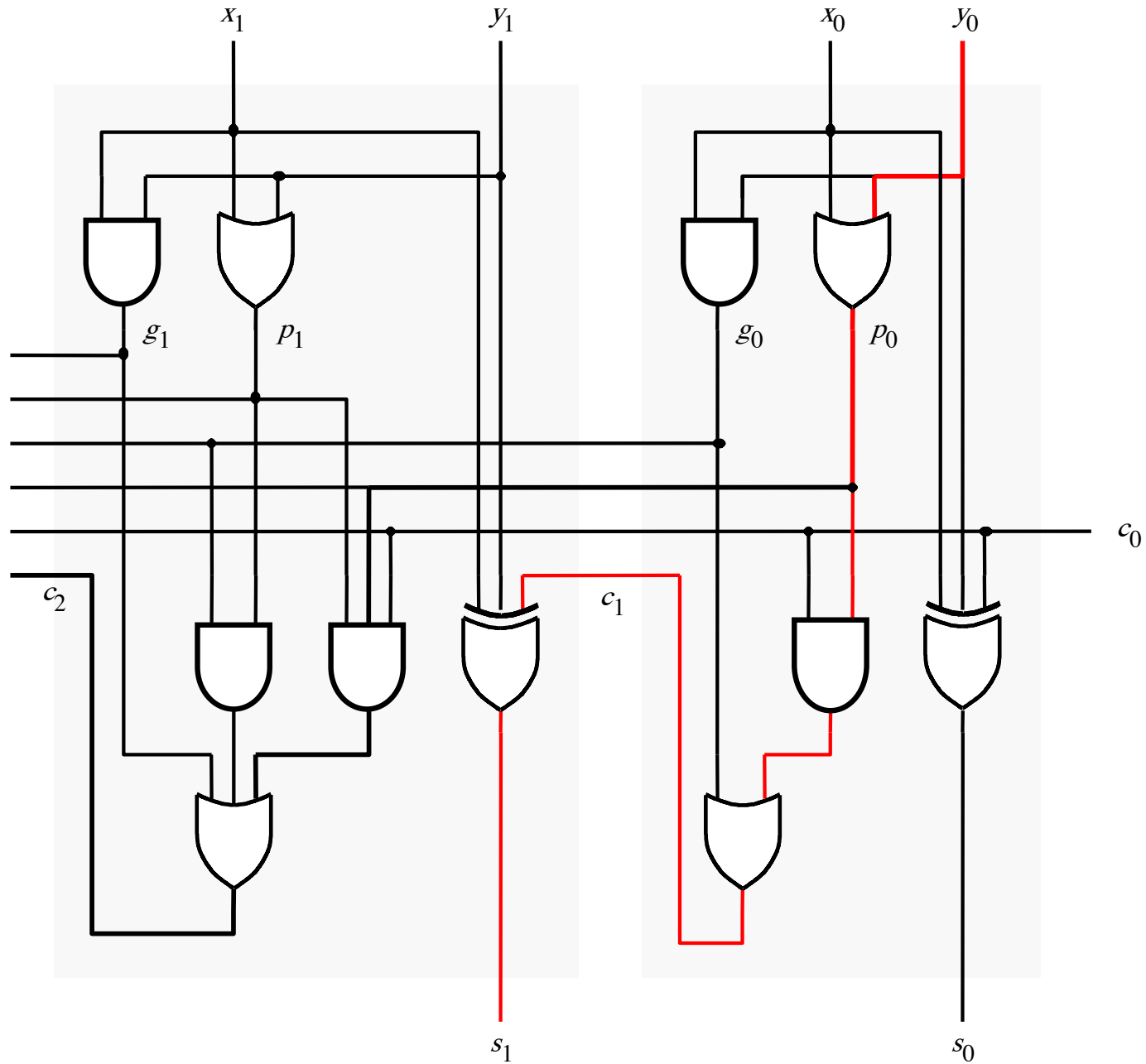
It takes 3 gate delays to generate c_2



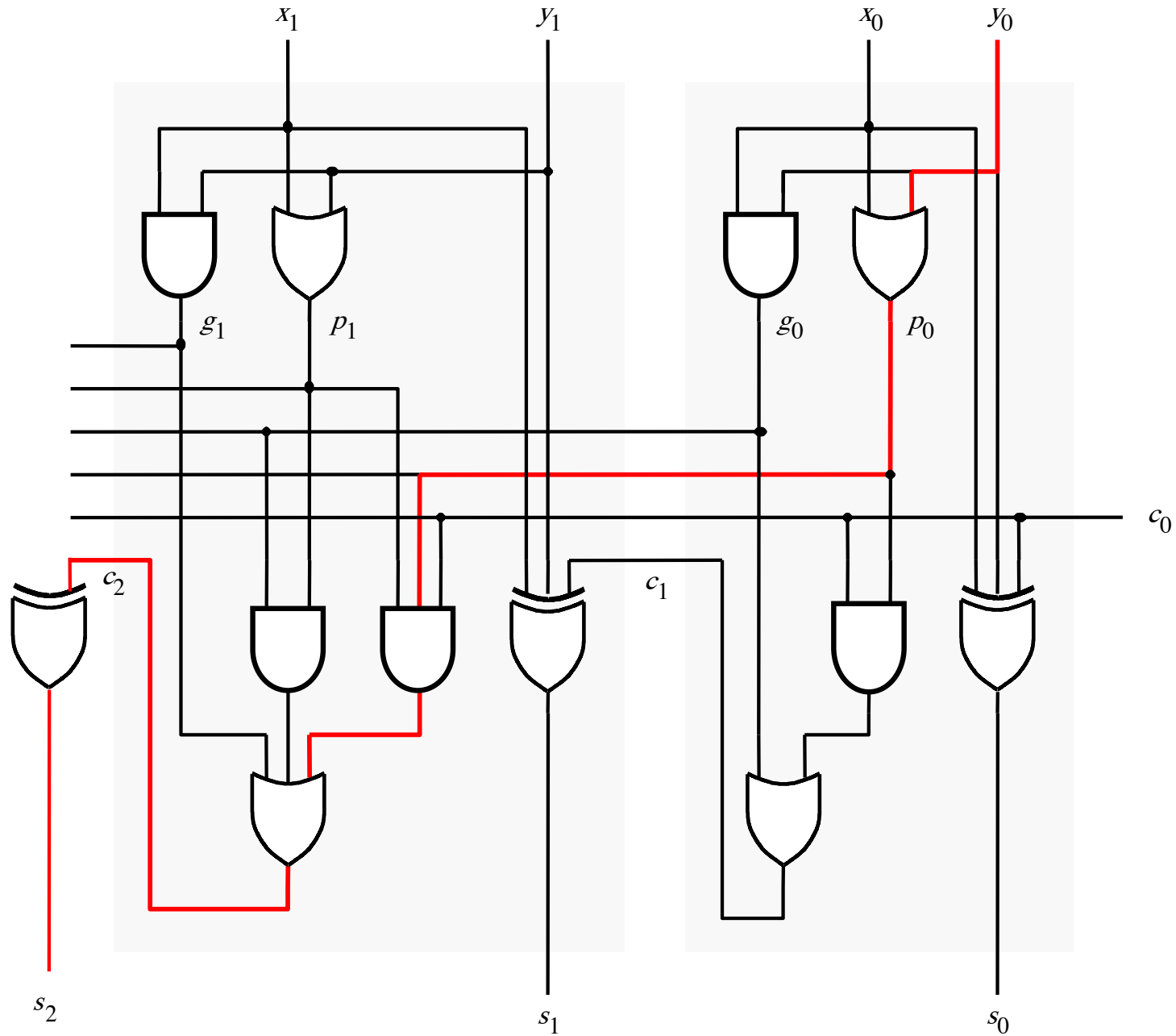
The first two stages of a carry-lookahead adder



It takes 4 gate delays to generate s_1



It takes 4 gate delays to generate s_2



N-bit Carry-Lookahead Adder

- **It takes 3 gate delays to generate all carry signals**
- **It takes 1 more gate delay to generate all sum bits**
- **Thus, the total delay through an n-bit carry-lookahead adder is only 4 gate delays!**

Expanding the Carry Expression

$$c_{i+1} = g_i + p_i c_i$$

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

$$c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$$

...

$$\begin{aligned} c_8 = & g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4 \\ & + p_7 p_6 p_5 p_4 g_3 + p_7 p_6 p_5 p_4 p_3 g_2 \\ & + p_7 p_6 p_5 p_4 p_3 p_2 g_1 + p_7 p_6 p_5 p_4 p_3 p_2 p_1 g_0 \\ & + p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0 c_0 \end{aligned}$$

Expanding the Carry Expression

$$c_{i+1} = g_i + p_i c_i$$

$$c_1 = g_0 + p_0 c_0$$

$$c_2 = g_1 + p_1 g_0 + p_1 p_0 c_0$$

$$c_3 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$$

...

$$c_8 = g_7 + p_7 g_6 + p_7 p_6 g_5 + p_7 p_6 p_5 g_4$$

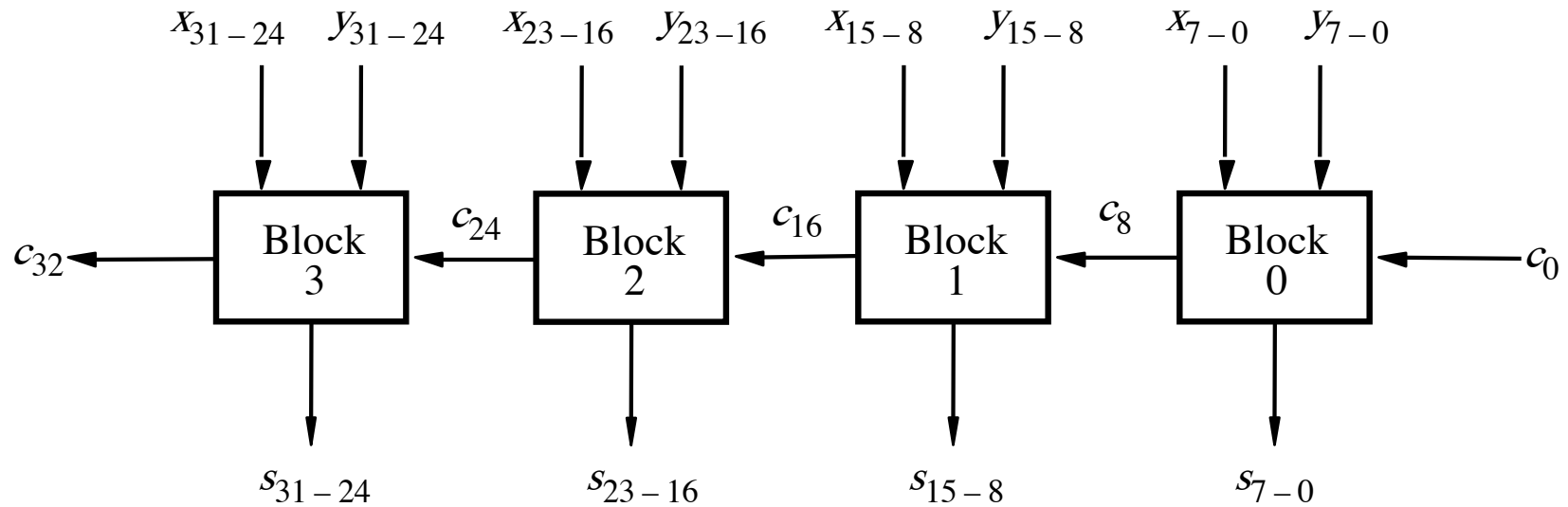
Even this takes
only 3 gate delays

$$+ p_7 p_6 p_5 p_4 g_3 + p_7 p_6 p_5 p_4 p_3 g_2$$

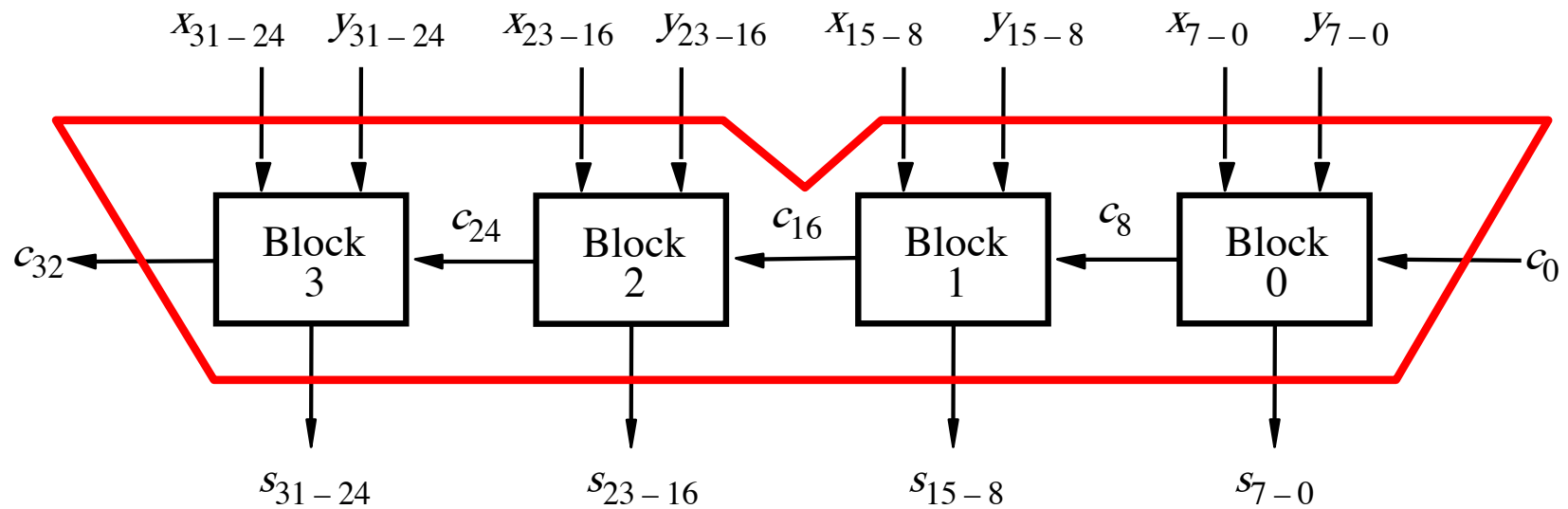
$$+ p_7 p_6 p_5 p_4 p_3 p_2 g_1 + p_7 p_6 p_5 p_4 p_3 p_2 p_1 g_0$$

$$+ p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0 c_0$$

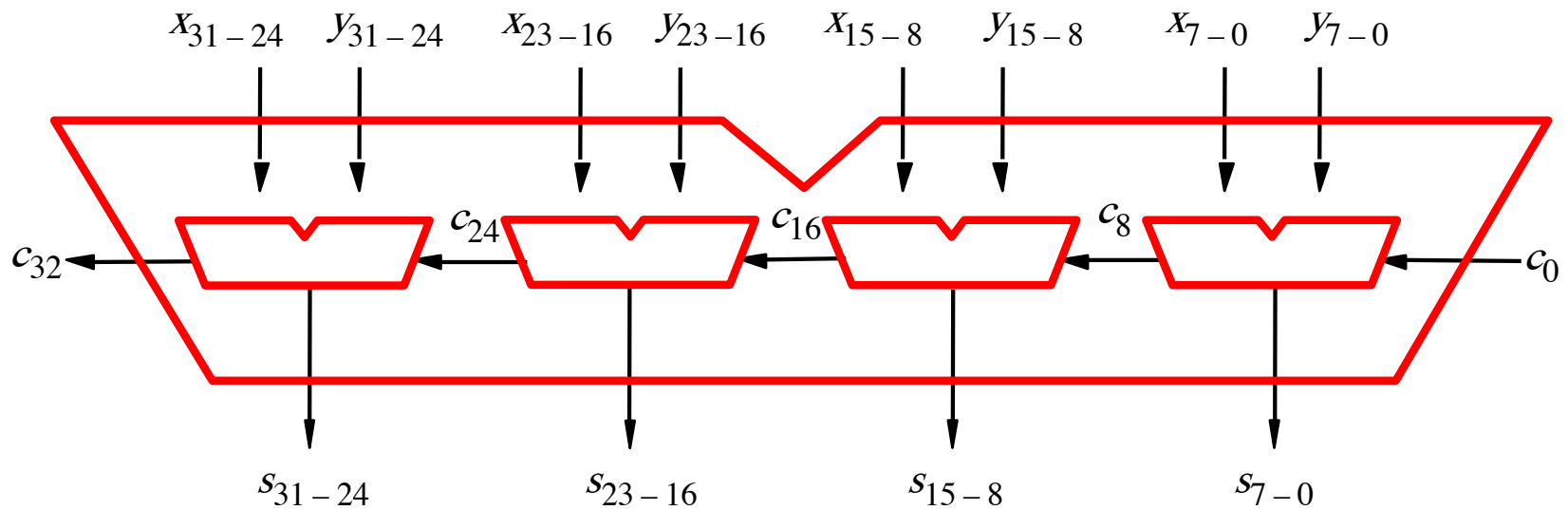
A hierarchical carry-lookahead adder with ripple-carry between blocks



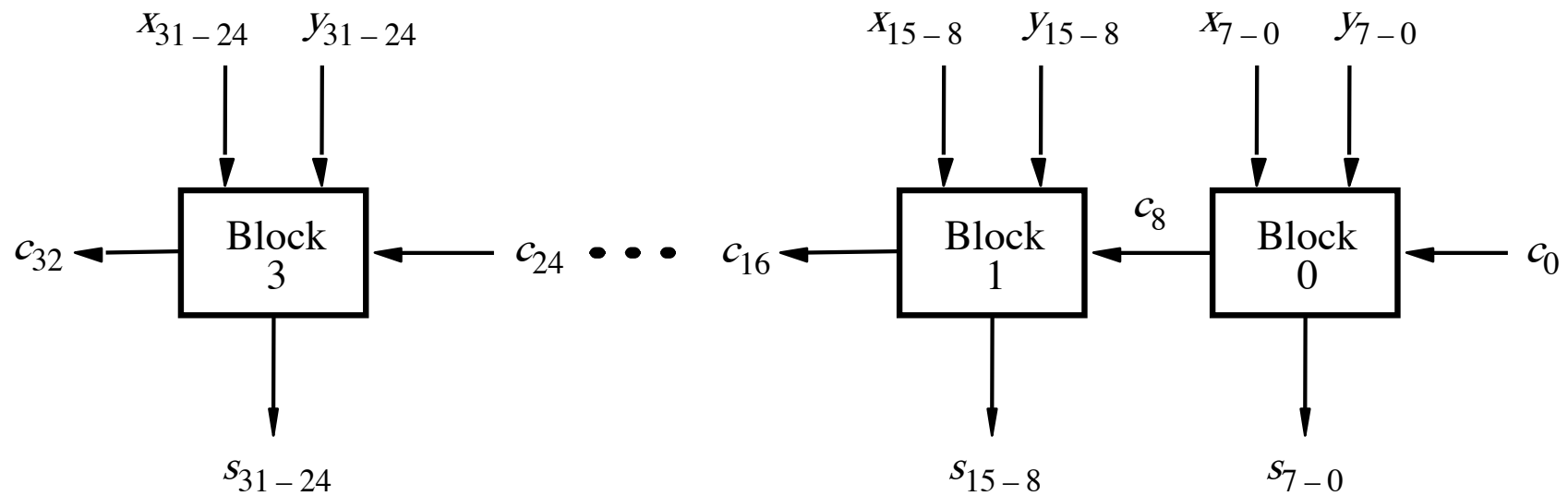
A hierarchical carry-lookahead adder with ripple-carry between blocks



A hierarchical carry-lookahead adder with ripple-carry between blocks

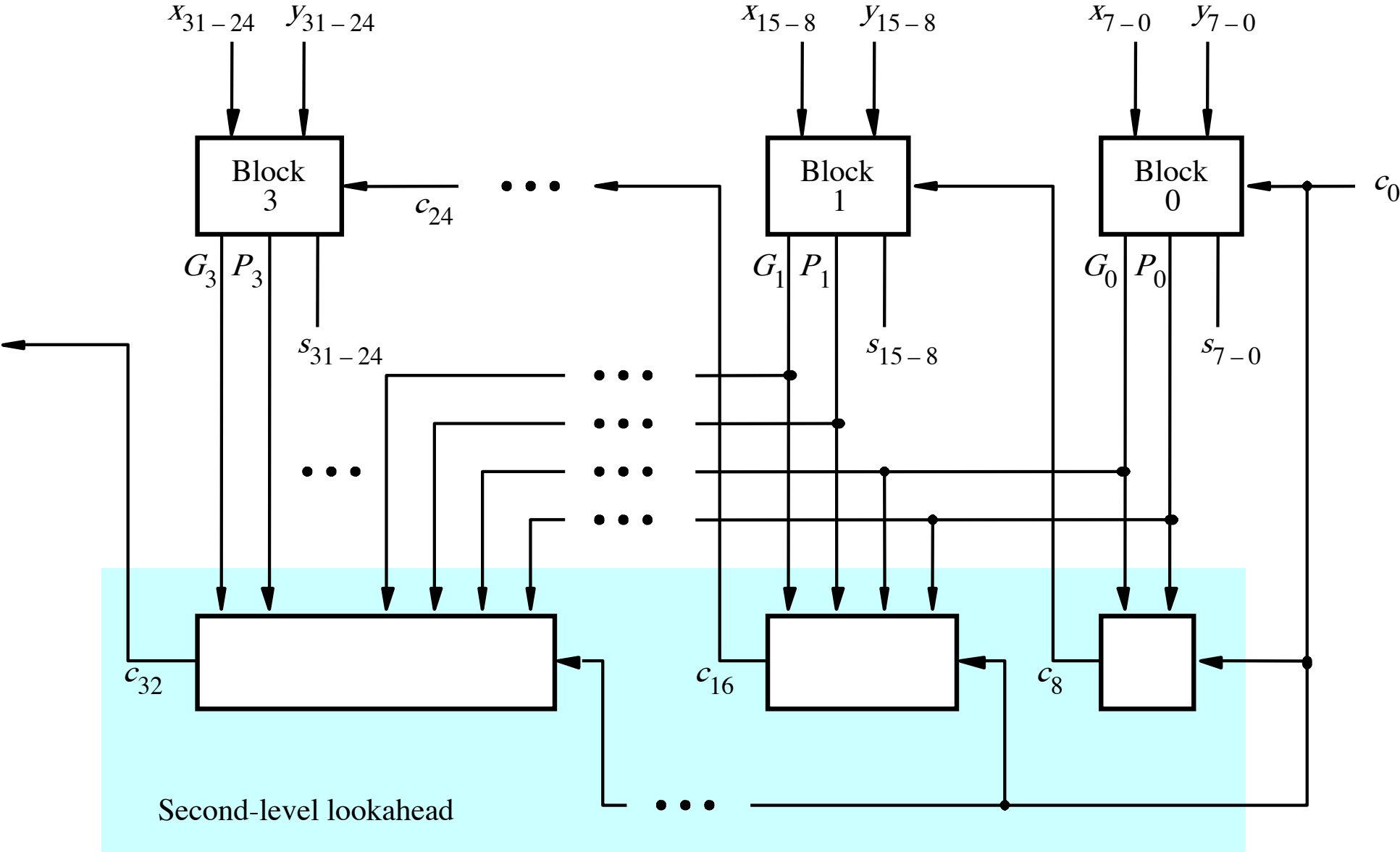


A hierarchical carry-lookahead adder with ripple-carry between blocks



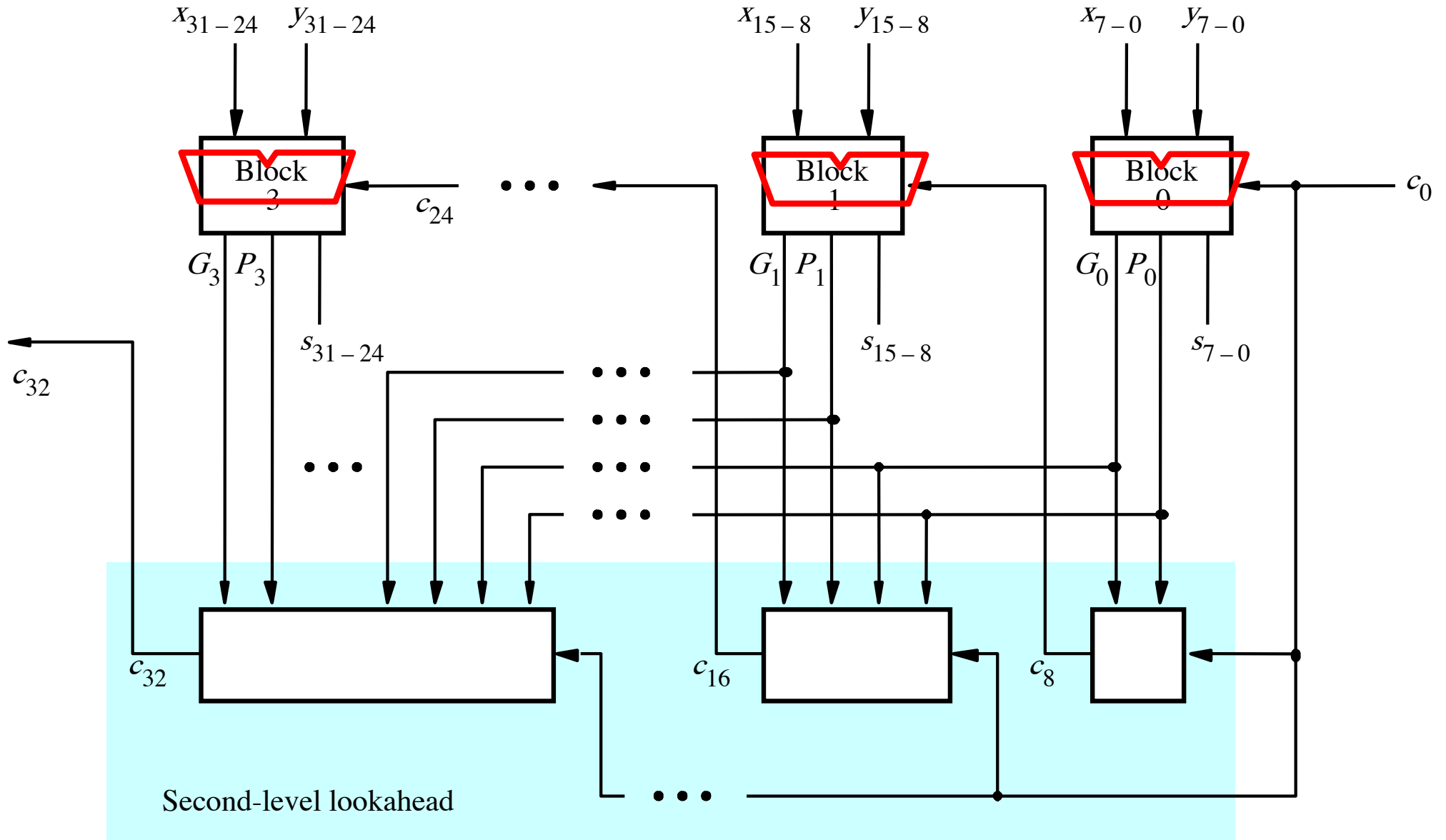
[Figure 3.16 from the textbook]

A hierarchical carry-lookahead adder

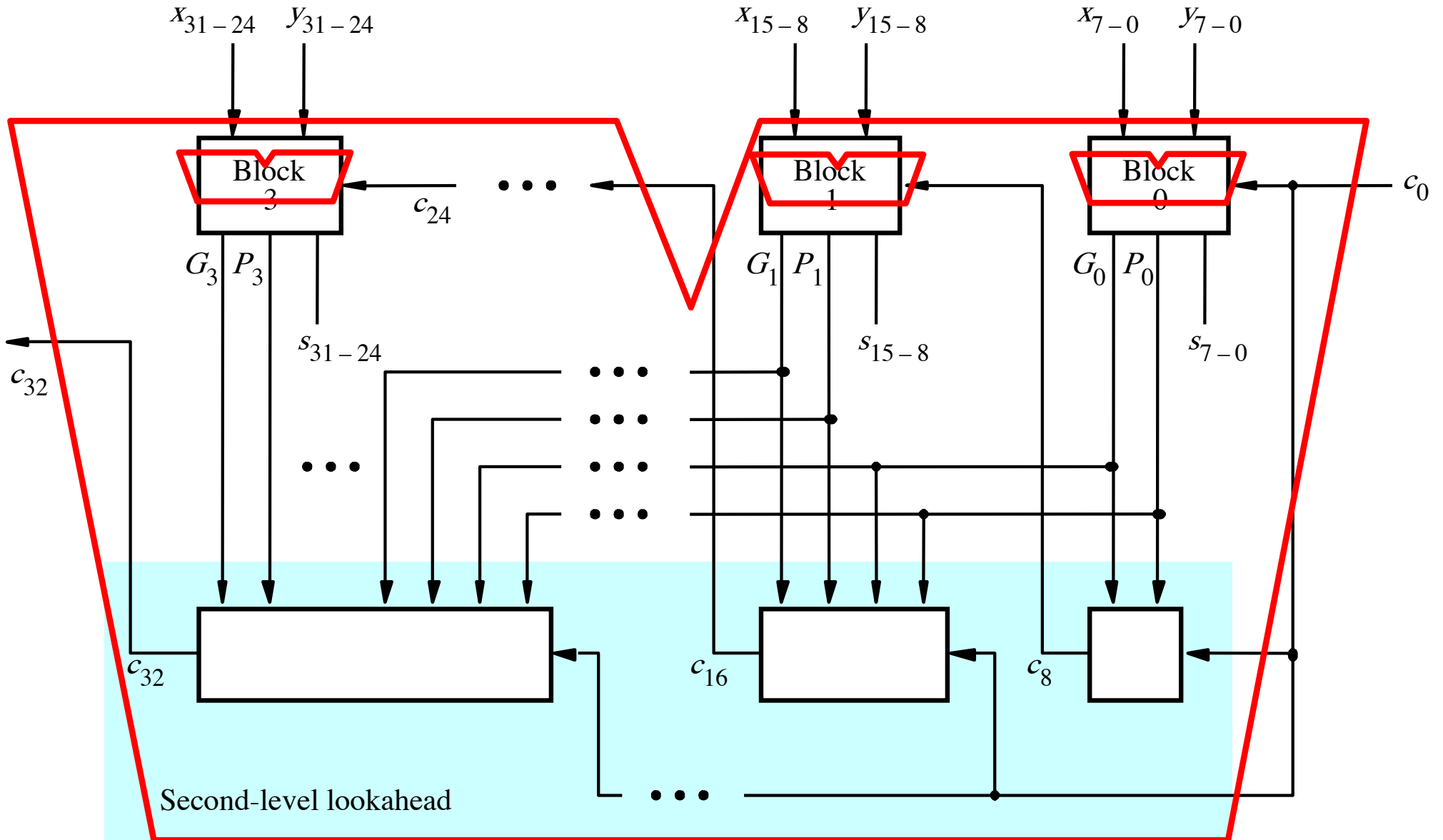


[Figure 3.17 from the textbook]

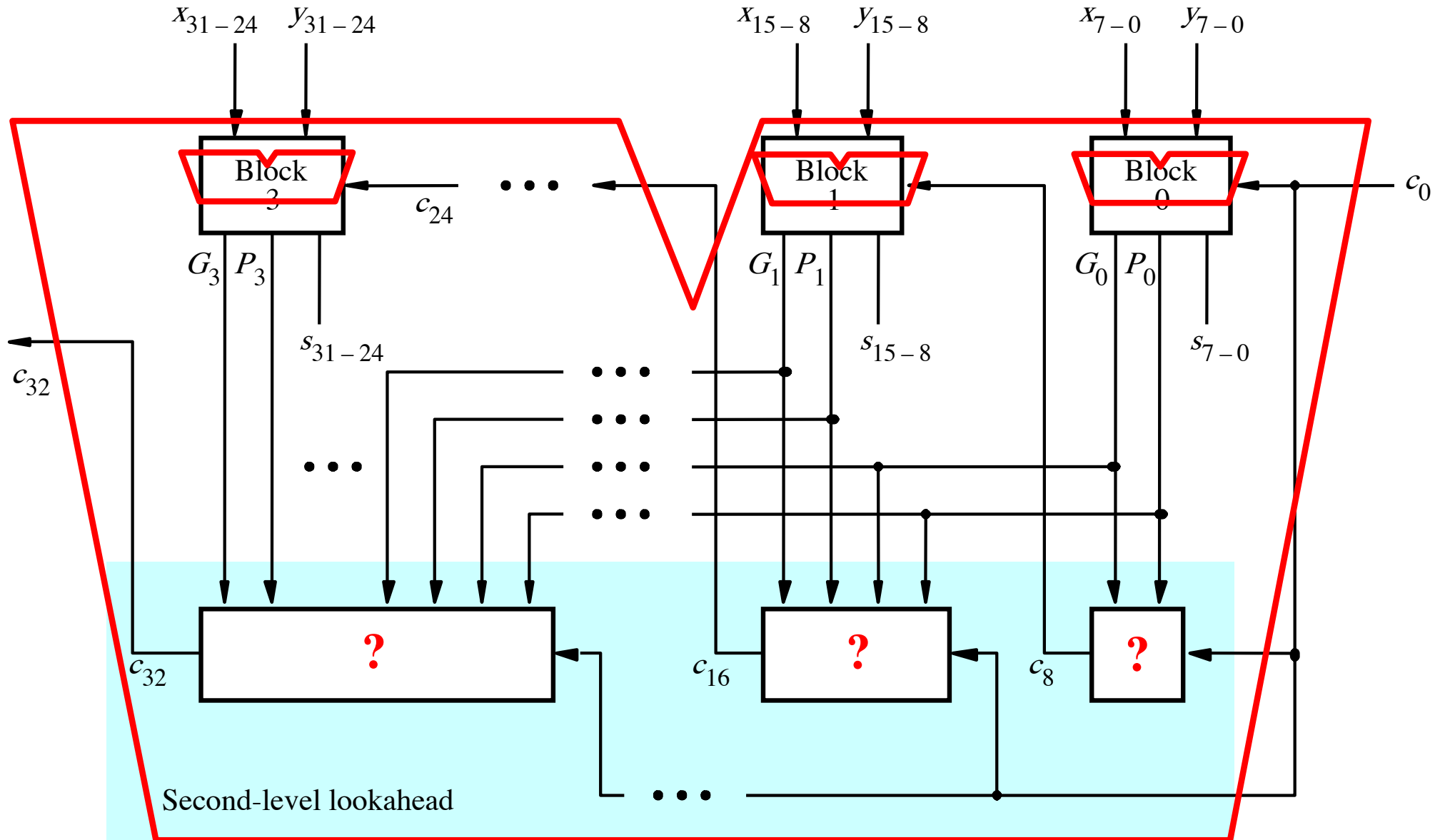
A hierarchical carry-lookahead adder



A hierarchical carry-lookahead adder



A hierarchical carry-lookahead adder



The Hierarchical Carry Expression

$$\begin{aligned}c_8 = & g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 \\ & + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 \\ & + p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1g_0 \\ & + p_7p_6p_5p_4p_3p_2p_1p_0c_0\end{aligned}$$

The Hierarchical Carry Expression

$$\begin{aligned}c_8 = & g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 \\ & + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 \\ & + p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1g_0 \\ & + p_7p_6p_5p_4p_3p_2p_1p_0c_0\end{aligned}$$

The Hierarchical Carry Expression

$$c_8 = g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 \\ + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 \\ + p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1g_0 \\ + p_7p_6p_5p_4p_3p_2p_1p_0c_0$$

G_0 →

P_0 →

The Hierarchical Carry Expression

$$c_8 = g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 \\ + p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 \\ + p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1g_0 \\ + p_7p_6p_5p_4p_3p_2p_1p_0c_0$$

G_0 →

P_0 →

$$c_8 = G_0 + P_0c_0$$

The Hierarchical Carry Expression

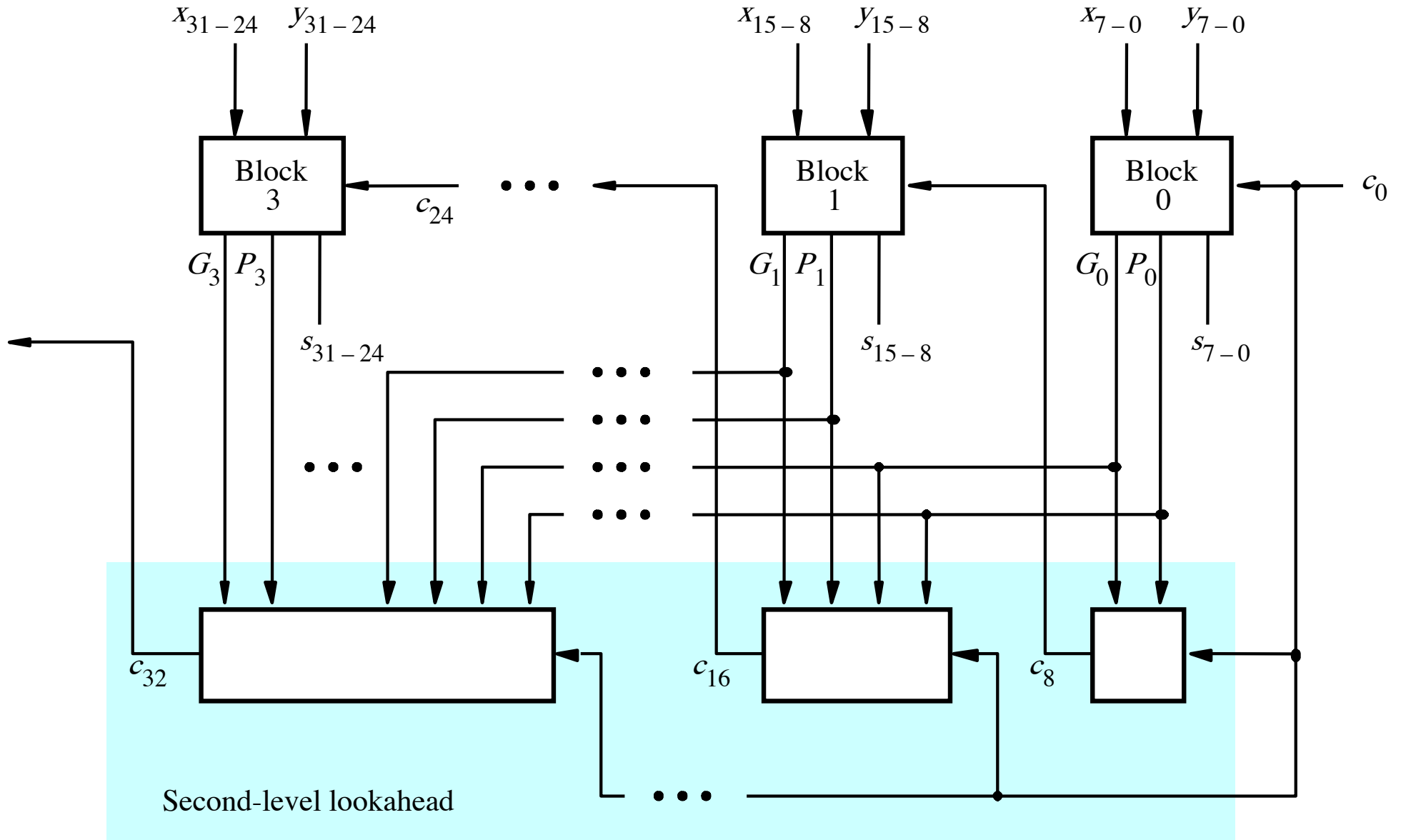
$$c_8 = G_0 + P_0 c_0$$

$$\begin{aligned} c_{16} &= G_1 + P_1 c_8 \\ &= G_1 + P_1 G_0 + P_1 P_0 c_0 \end{aligned}$$

$$c_{24} = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_0$$

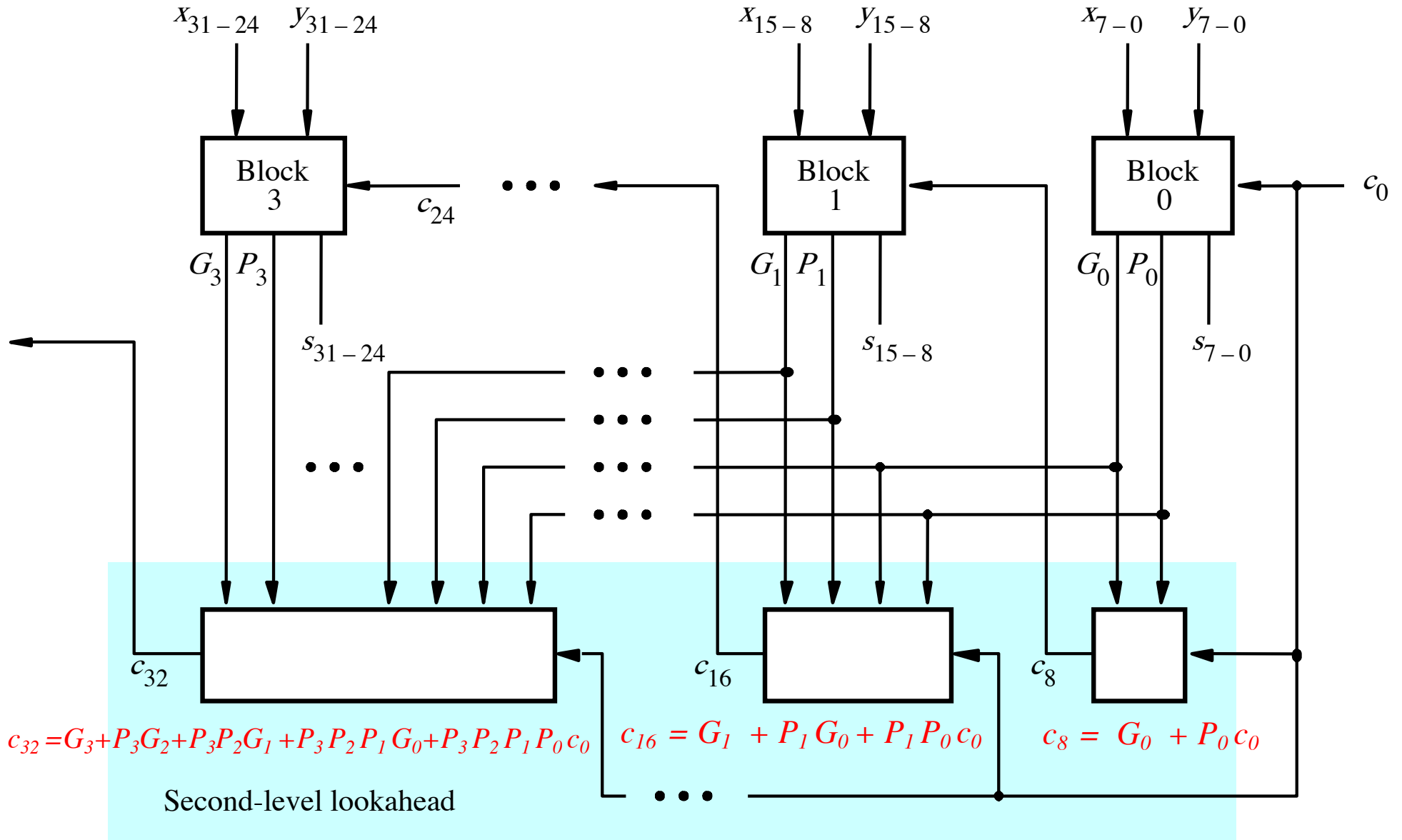
$$c_{32} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0$$

A hierarchical carry-lookahead adder



[Figure 3.17 from the textbook]

A hierarchical carry-lookahead adder

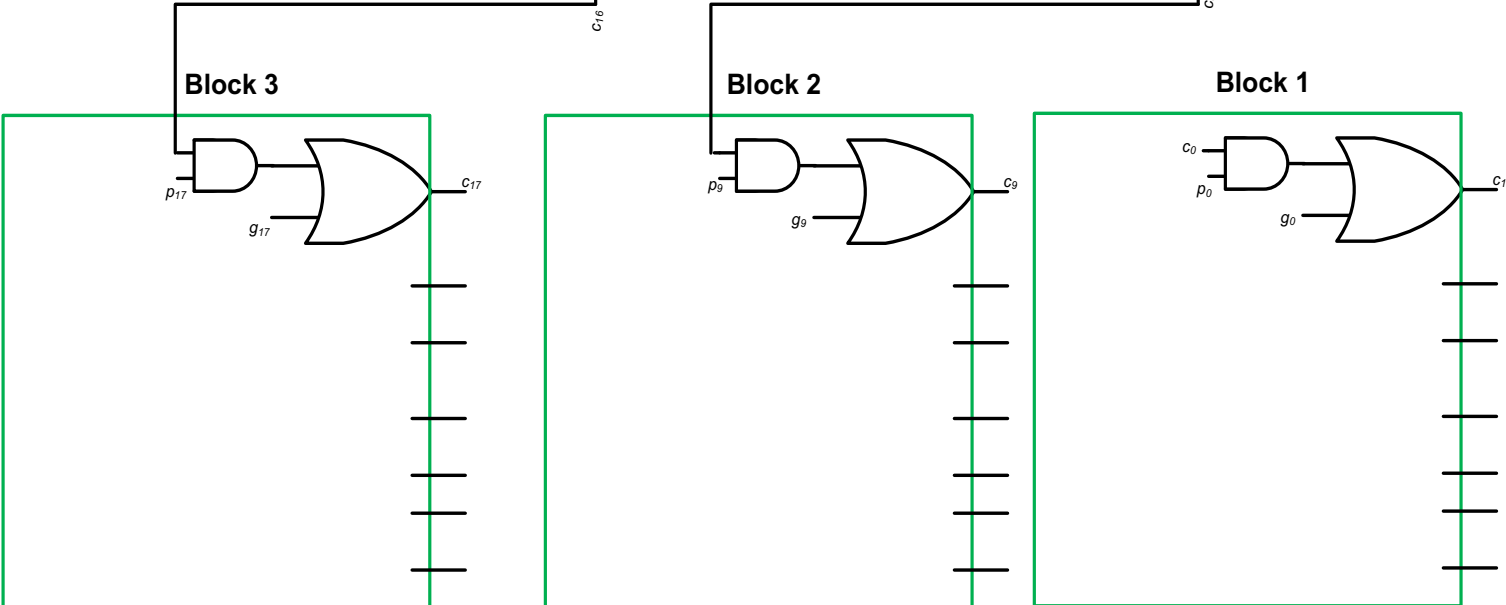
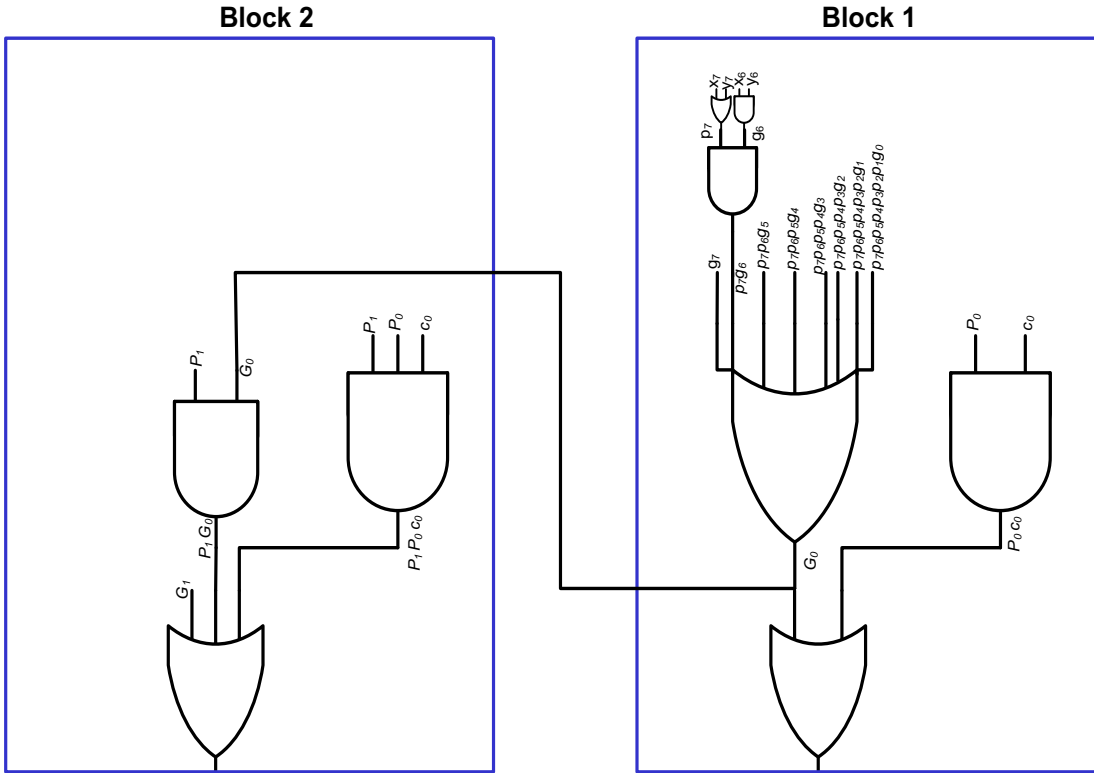


[Figure 3.17 from the textbook]

Hierarchical CLA Adder Carry Logic

SECOND
LEVEL
HIERARCHY

- C8 – 5 gate delays
- C16 – 5 gate delays
- C24 – 5 Gate delays
- C32 – 5 Gate delays



FIRST LEVEL HIERARCHY

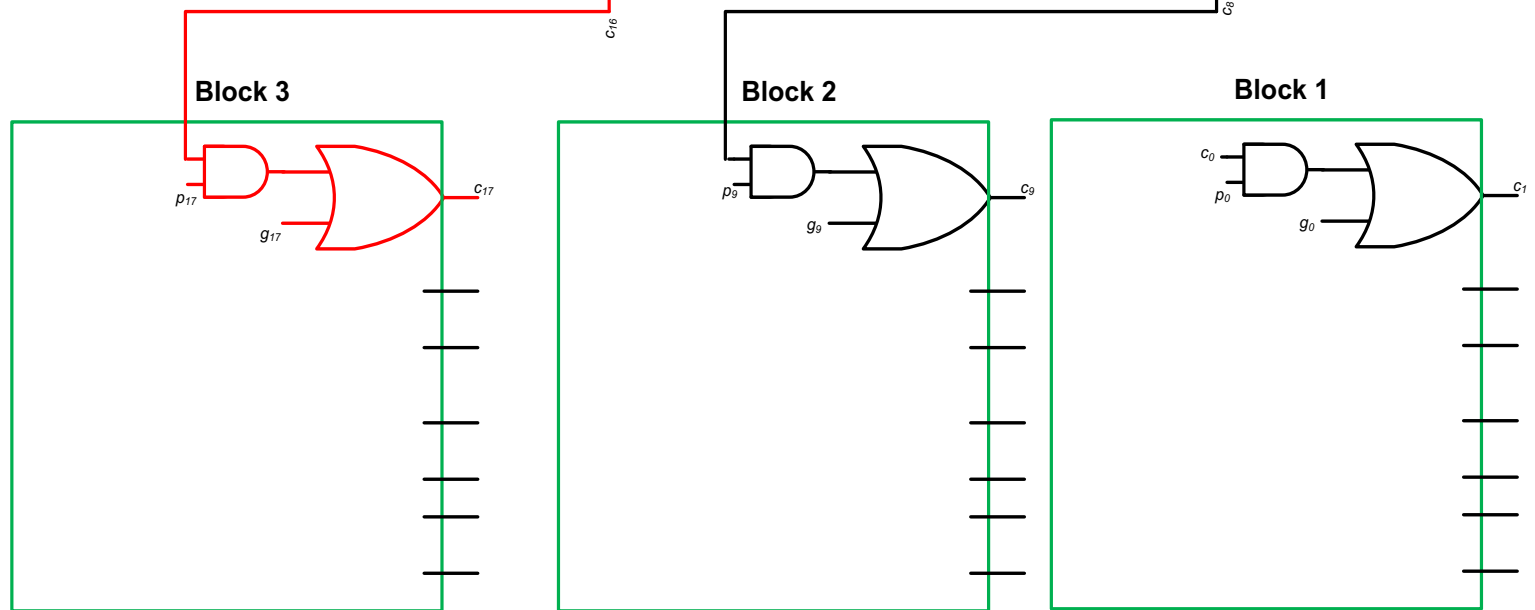
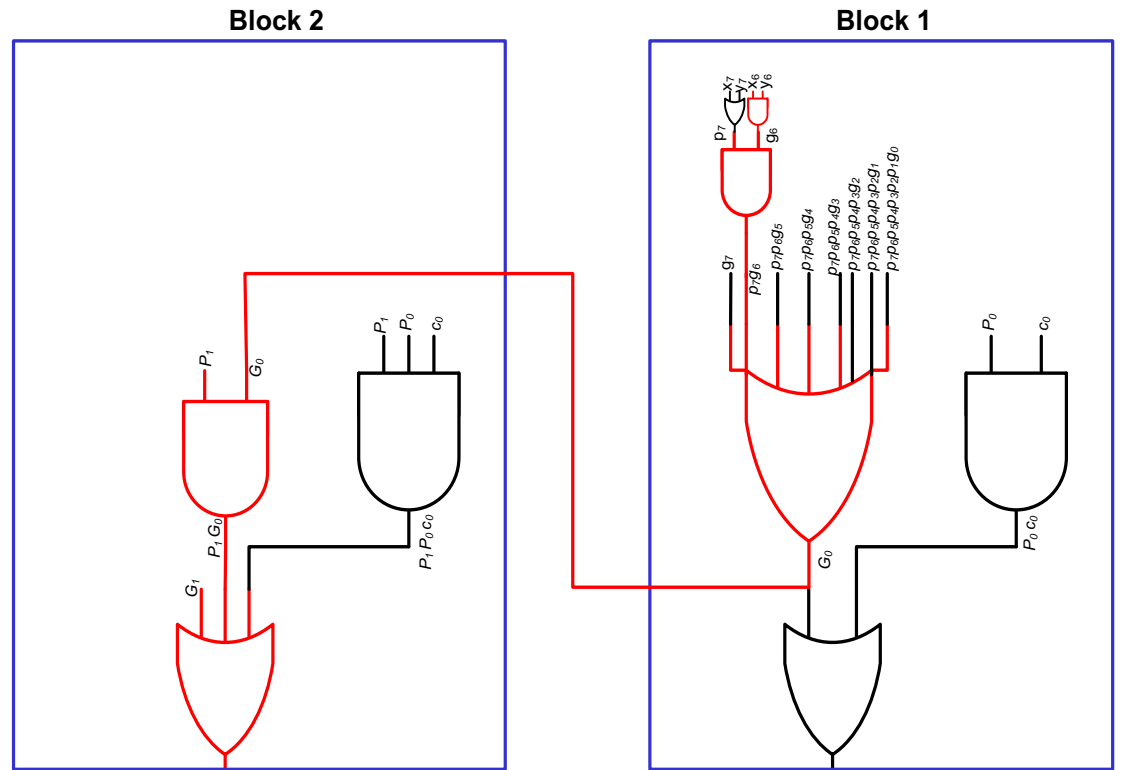
Hierarchical CLA Critical Path

SECOND
LEVEL
HIERARCHY

C9 – 7 gate delays

C17 – 7 gate delays

C25 – 7 Gate delays



FIRST LEVEL HIERARCHY

Total Gate Delay Through a Hierarchical Carry-Lookahead Adder

- Is 8 gates
 - 3 to generate all G_j and P_j
 - +2 to generate c_8 , c_{16} , c_{24} , and c_{32}
 - +2 to generate internal carries in the blocks
 - +1 to generate the sum bits (one extra XOR)

Questions?

THE END