

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

- **No HW is due next Monday**
- **HW 6 will be due on Monday Oct. 11.**

Administrative Stuff

- Labs next week
- Mini-Project
- This is worth 3% of your grade (x2 labs)
- https://www.ece.iastate.edu/~alexs/classes/2021_Fall_281/labs/Project-Mini/

Quick Review

The problems in which row are easier to calculate?

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

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

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

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

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

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

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

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

Another Way to Do Subtraction

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

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

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

10's complement

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

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

$$= 118 - 100$$

Another Way to Do Subtraction

9's complement

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

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

10's complement

$$= 82 + 36 - 100$$

// Add the first two.

$$= 118 - 100$$

// Just delete the leading 1.
// No need to subtract 100.

$$= 18$$

1's Complement

1' s complement (subtract each digit from 1)

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

Then, in 1' s complement representation K is obtained by subtracting P from $2^n - 1$, namely

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

This means that K can be obtained by inverting all bits of P .

1' s complement (subtract each digit from 1)

Let K be the negative equivalent of an 8-bit positive number P.

Then, in 1' s complement representation K is obtained by subtracting P from $2^8 - 1$, namely

$$K = (2^8 - 1) - P = 255 - P$$

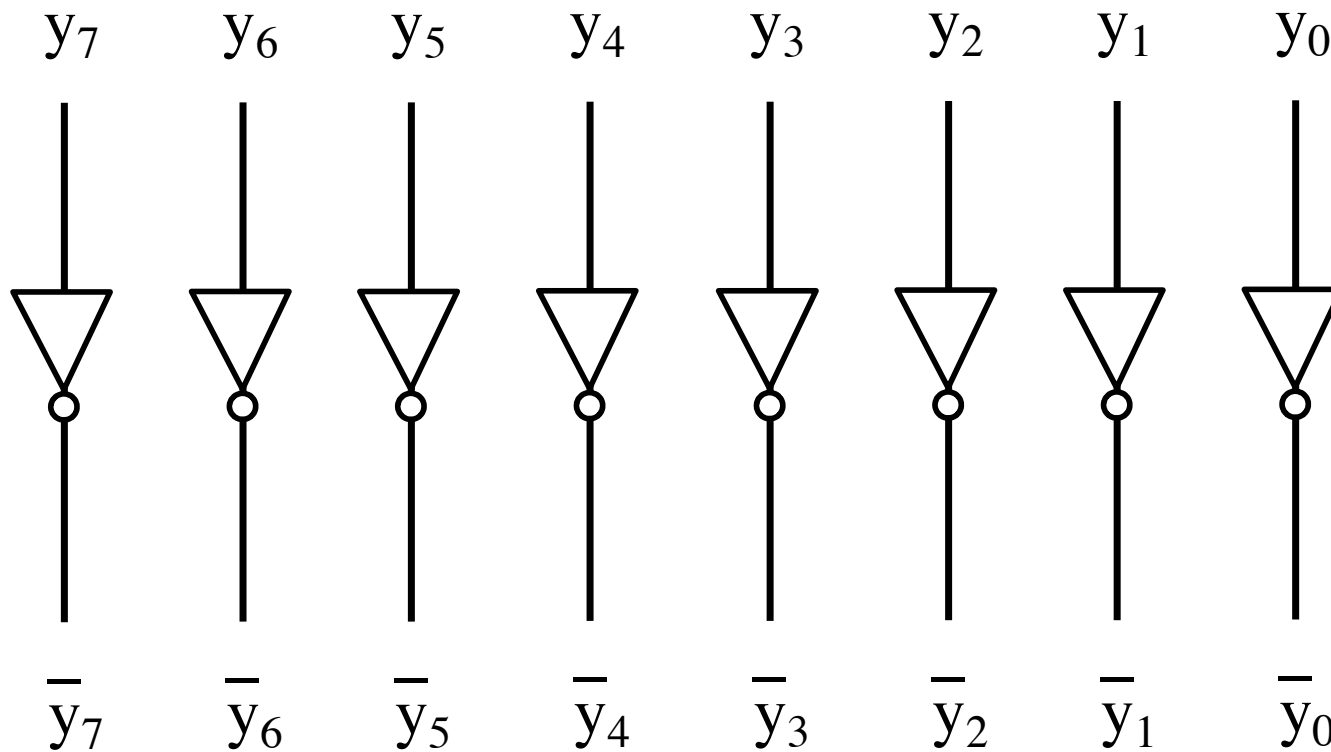
This means that K can be obtained by inverting all bits of P.

Provided that P is between 0 and 127, because the most significant bit must be zero to indicate that it is positive.

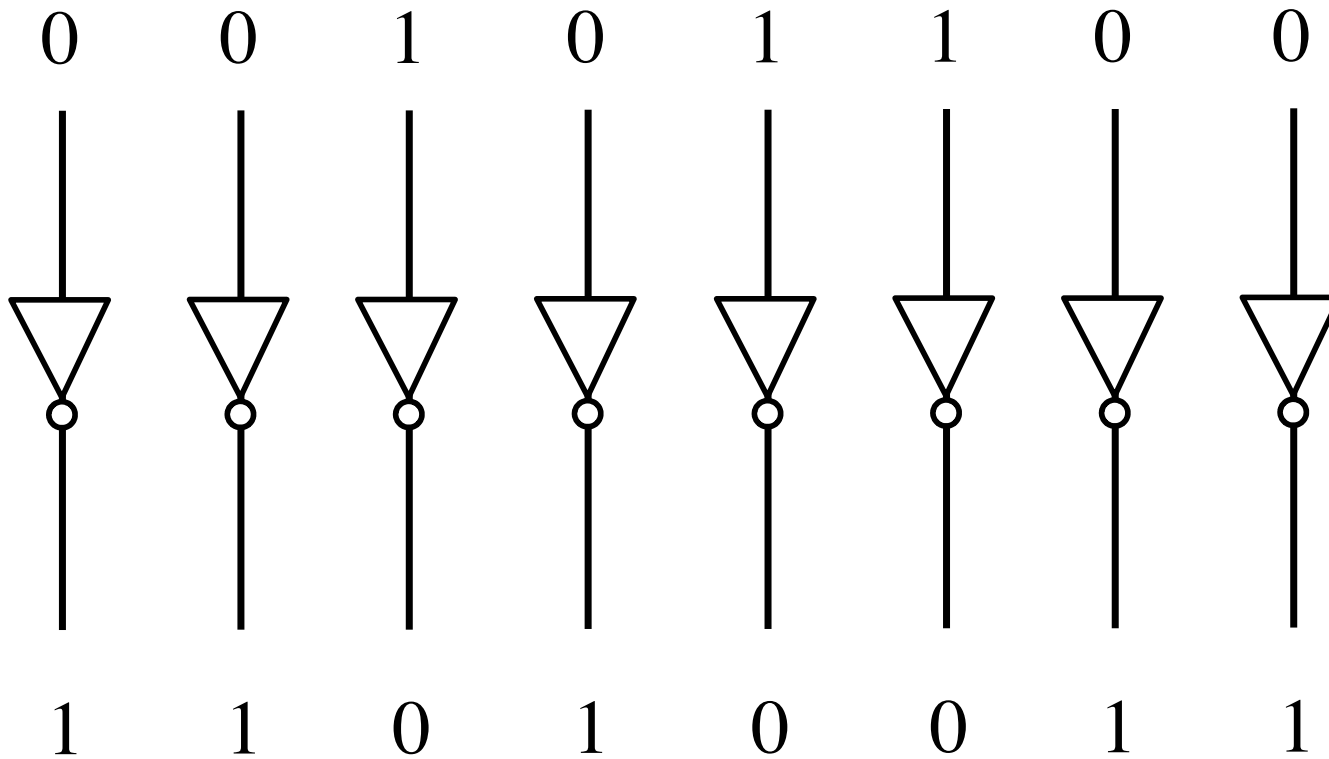
1's complement (subtract each digit from 1)

$$\begin{array}{r} \\ \\ - \\ \hline \end{array}$$

Circuit for negating a number stored in 1's complement representation



Circuit for negating a number stored in 1's complement representation



2's Complement

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.

Deriving 2' s complement

For a positive 8-bit number P, let K_1 and K_2 denote its 1' s and 2' s complements, respectively.

$$K_1 = (2^n - 1) - P = 255 - P$$

$$K_2 = 2^n - P = 256 - P$$

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 ...

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

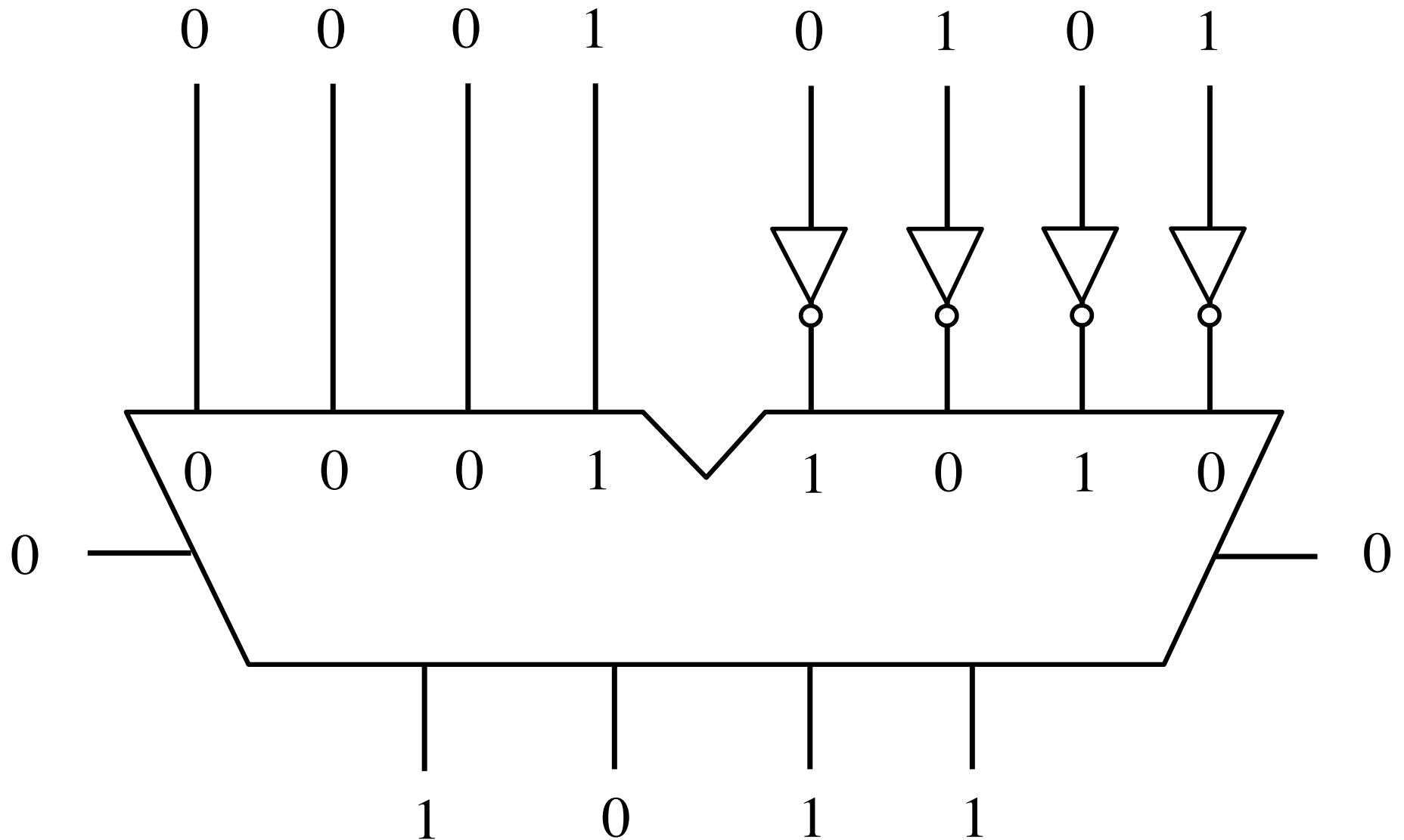
$$\begin{array}{r} 0010 \\ + 1101 \\ \hline 1110 \end{array}$$

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

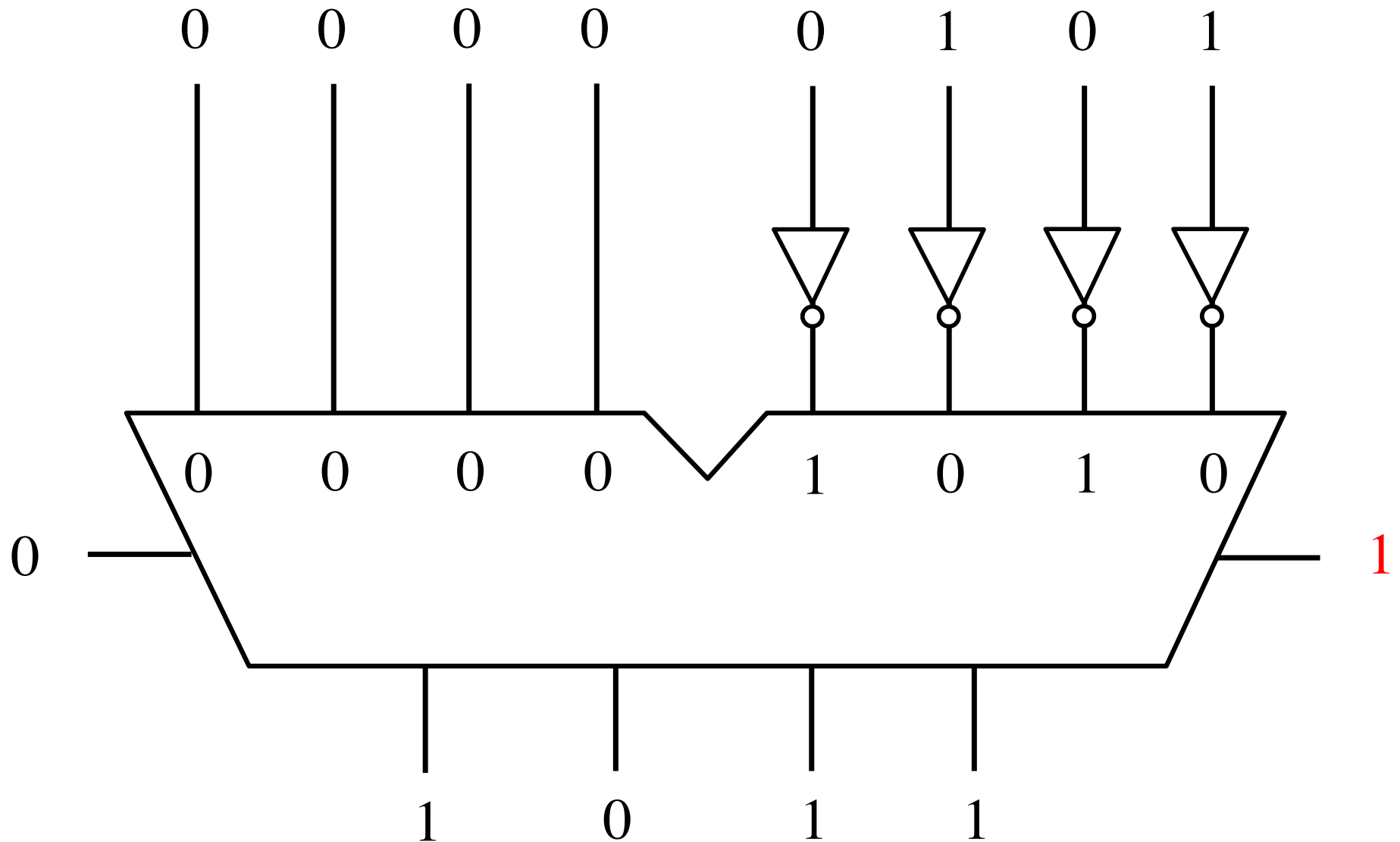
$$\begin{array}{r} 0111 \\ + 1000 \\ \hline 1001 \end{array}$$

Then add 1.

Circuit for negating a number stored in 2's complement representation



Circuit for negating a number stored in 2's complement representation



**Addition of two numbers stored
in 2's complement representation**

There are four cases to consider

- $(+5) + (+2)$
- $(-5) + (+2)$
- $(+5) + (-2)$
- $(-5) + (-2)$

There are four cases to consider

- $(+5) + (+2)$ **positive plus positive**
- $(-5) + (+2)$ **negative plus positive**
- $(+5) + (-2)$ **positive plus negative**
- $(-5) + (-2)$ **negative plus negative**

Positive plus positive

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

$b_3b_2b_1b_0$	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

Negative plus positive


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

$b_3b_2b_1b_0$	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

[Figure 3.9 from the textbook]

Positive plus negative

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




 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

[Figure 3.9 from the textbook]

Negative plus negative

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



 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

[Figure 3.9 from the textbook]

**Subtraction of two numbers stored
in 2's complement representation**

There are four cases to consider

- $(+5) - (+2)$
- $(-5) - (+2)$
- $(+5) - (-2)$
- $(-5) - (-2)$

There are four cases to consider

- $(+5) - (+2)$ **positive minus positive**
- $(-5) - (+2)$ **negative minus positive**
- $(+5) - (-2)$ **positive minus negative**
- $(-5) - (-2)$ **negative minus negative**

There are four cases to consider

- $(+5) - (+2)$
- $(-5) - (+2)$
- $(+5) - (-2)$
- $(-5) - (-2)$

There are four cases to consider

- $(+5) - (+2) = (+5) + (-2)$
- $(-5) - (+2) = (-5) + (-2)$
- $(+5) - (-2) = (+5) + (+2)$
- $(-5) - (-2) = (-5) + (+2)$

There are four cases to consider

- $(+5) - (+2) = (+5) + (-2)$

- $(-5) - (+2) = (-5) + (-2)$

- $(+5) - (-2) = (+5) + (+2)$

- $(-5) - (-2) = (-5) + (+2)$

We can change subtraction into addition ...

There are four cases to consider

- $(+5) - (+2) = (+5) + (-2)$

- $(-5) - (+2) = (-5) + (-2)$

- $(+5) - (-2) = (+5) + (+2)$

- $(-5) - (-2) = (-5) + (+2)$

... if we negate the second number.

There are four cases to consider

- $(+5) - (+2) = (+5) + (-2)$

- $(-5) - (+2) = (-5) + (-2)$

- $(+5) - (-2) = (+5) + (+2)$

- $(-5) - (-2) = (-5) + (+2)$

There are the four addition cases
(arranged in a shuffled order)

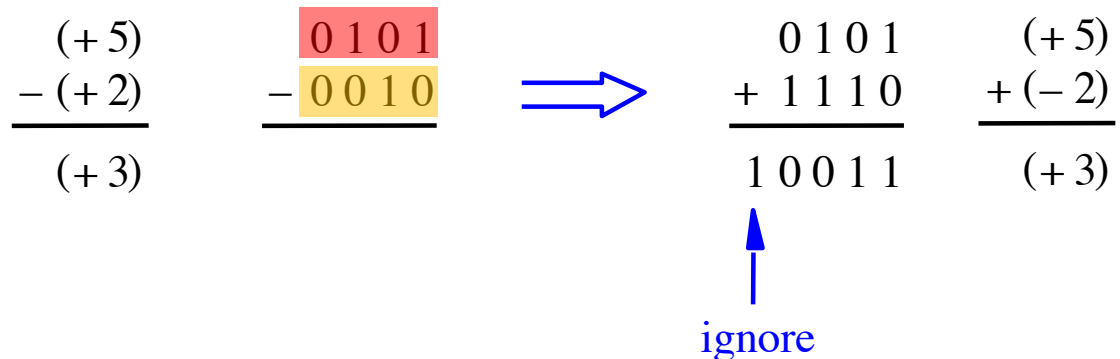
Positive minus positive

$$\begin{array}{r}
 (+5) \\
 - (+2) \\
 \hline
 (+3)
 \end{array}
 \qquad
 \begin{array}{r}
 \color{red}{0101} \\
 - \color{yellow}{0010} \\
 \hline
 \end{array}$$

$b_3b_2b_1b_0$	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]

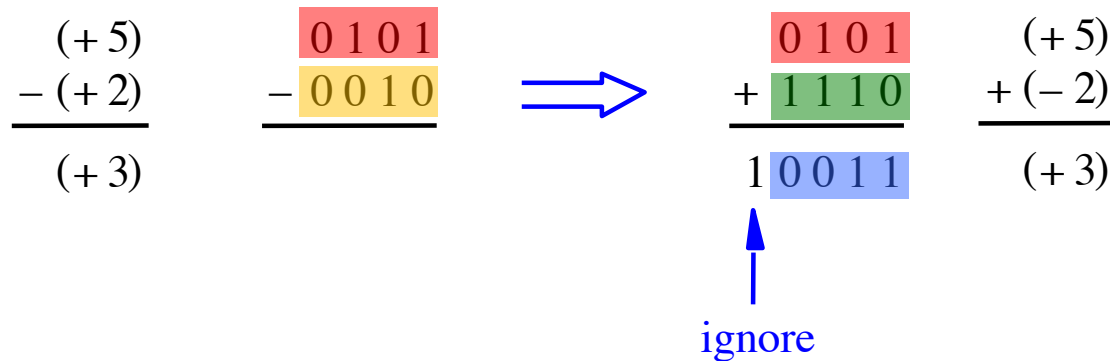
Convert to: Positive plus negative



$b_3b_2b_1b_0$	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]

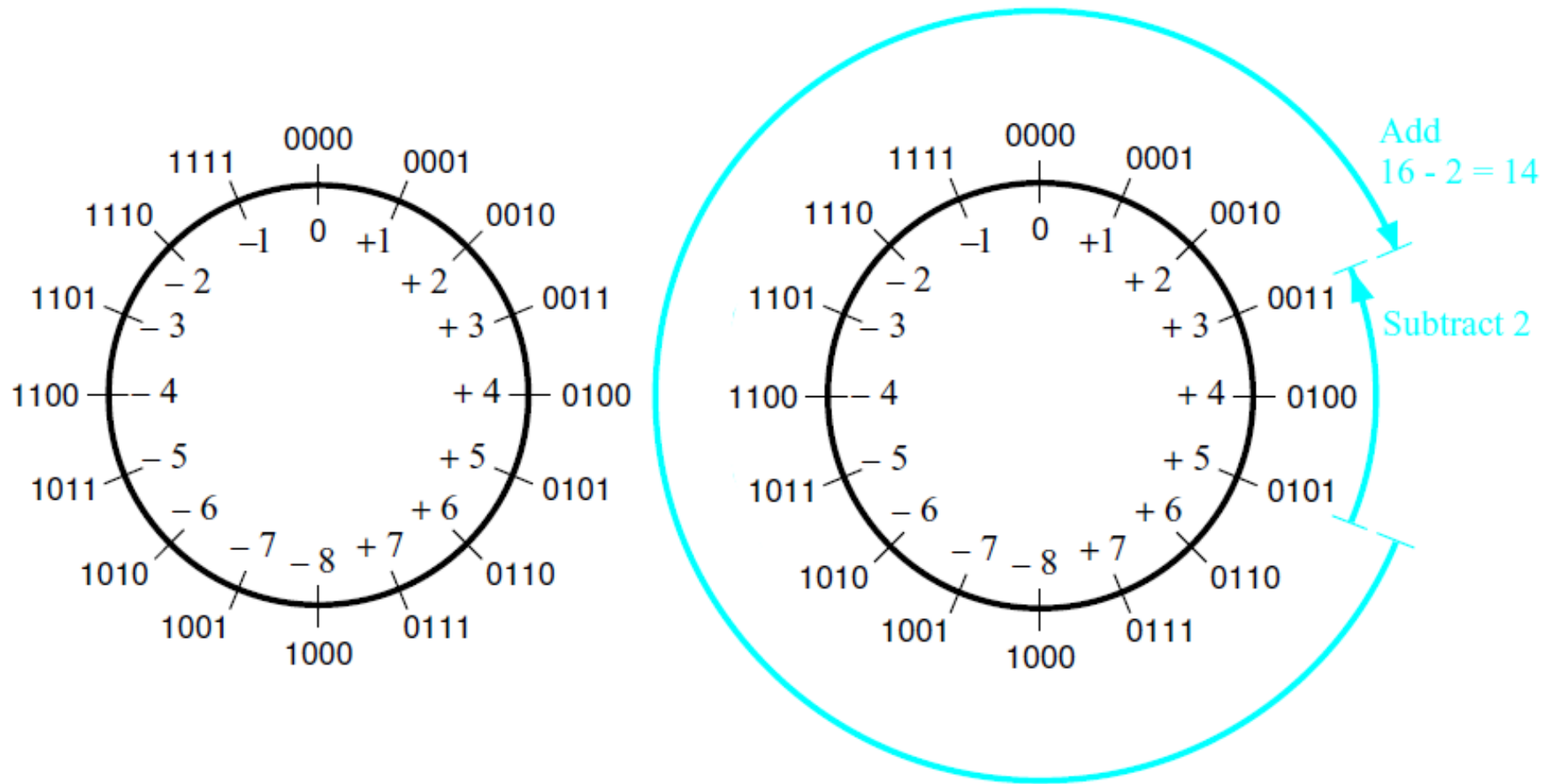
Convert to: Positive plus negative



$b_3b_2b_1b_0$	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

Negative minus positive

$$\begin{array}{r}
 (-5) \\
 - (+2) \\
 \hline
 (-7)
 \end{array}
 \qquad
 \begin{array}{r}
 \color{red}{1011} \\
 - \color{yellow}{0010} \\
 \hline
 \end{array}$$

$b_3b_2b_1b_0$	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]

Convert to: Negative plus negative

$$\begin{array}{r}
 (-5) \\
 - (+2) \\
 \hline
 (-7)
 \end{array}
 \quad
 \begin{array}{r}
 \color{red}{1011} \\
 - \color{yellow}{0010} \\
 \hline
 \end{array}
 \quad
 \Rightarrow
 \quad
 \begin{array}{r}
 \color{red}{1011} \\
 + \color{green}{1110} \\
 \hline
 1 \color{blue}{1001} \\
 \uparrow \\
 \text{ignore}
 \end{array}
 \quad
 \begin{array}{r}
 (-5) \\
 + (-2) \\
 \hline
 (-7)
 \end{array}$$

$b_3b_2b_1b_0$	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]

Positive minus negative

$$\begin{array}{r}
 (+5) \\
 - (-2) \\
 \hline
 (+7)
 \end{array}
 \qquad
 \begin{array}{r}
 \underline{0101} \\
 - \underline{1110} \\
 \hline
 \end{array}$$

$b_3b_2b_1b_0$	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]

Convert to: Positive plus positive

$$\begin{array}{r}
 (+5) \\
 - (-2) \\
 \hline
 (+7)
 \end{array}
 \quad
 \begin{array}{r}
 \color{red}{0101} \\
 - \color{yellow}{1110} \\
 \hline
 \end{array}
 \quad
 \Rightarrow
 \quad
 \begin{array}{r}
 \color{red}{0101} \\
 + \color{green}{0010} \\
 \hline
 \color{blue}{0111}
 \end{array}
 \quad
 \begin{array}{r}
 (+5) \\
 + (+2) \\
 \hline
 (+7)
 \end{array}$$

$b_3b_2b_1b_0$	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]

Negative minus negative

$$\begin{array}{r}
 (-5) \\
 - (-2) \\
 \hline
 (-3)
 \end{array}
 \qquad
 \begin{array}{r}
 \text{1 0 1 1} \\
 - \text{1 1 1 0} \\
 \hline
 \end{array}$$

$b_3b_2b_1b_0$	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]

Convert to: Negative plus positive

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

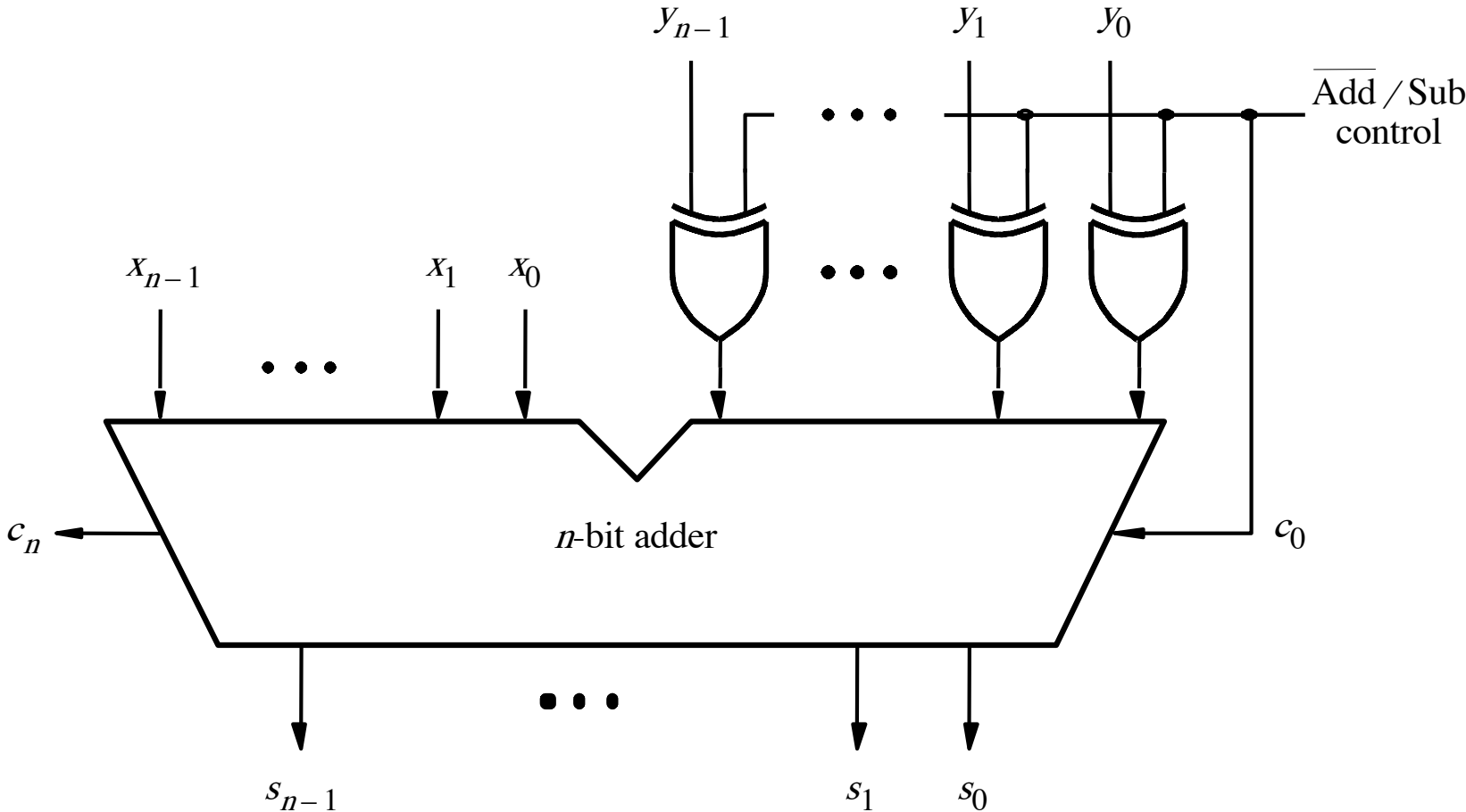
$b_3b_2b_1b_0$	2's complement
0111	+7
0110	+6
0101	+5
0100	+4
0011	+3
0010	+2
0001	+1
0000	+0
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

[Figure 3.10 from the textbook]

Take Home Message

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

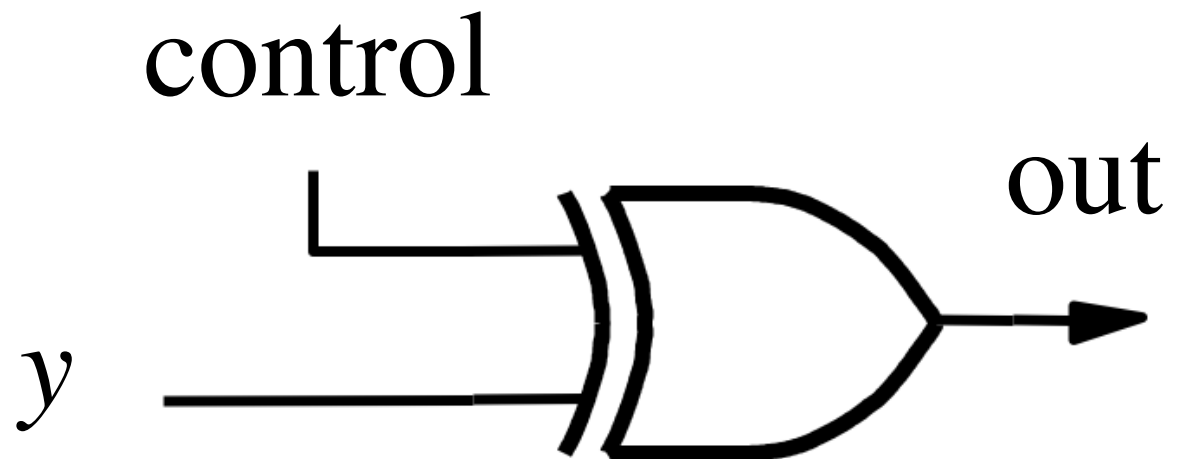
Adder/subtractor unit



[Figure 3.12 from the textbook]

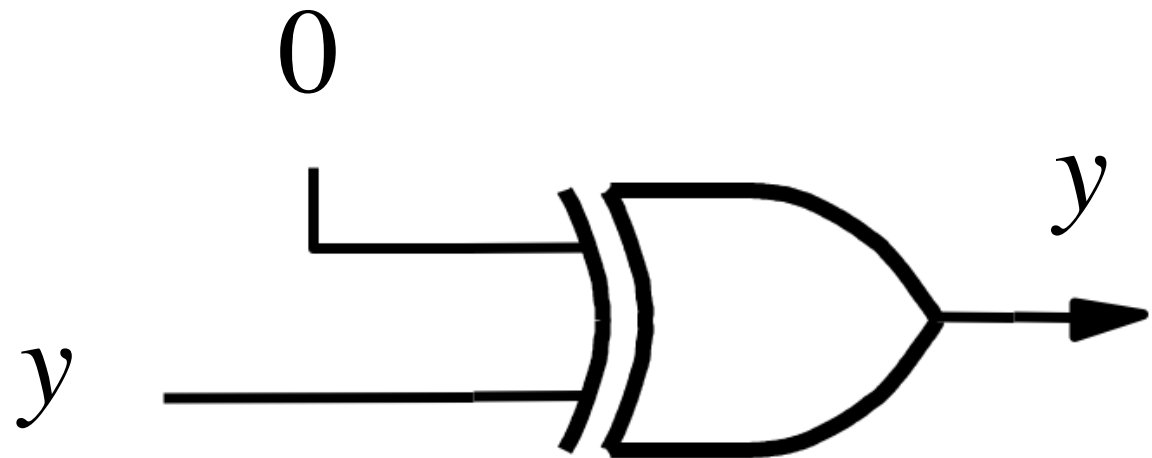

XOR Tricks

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



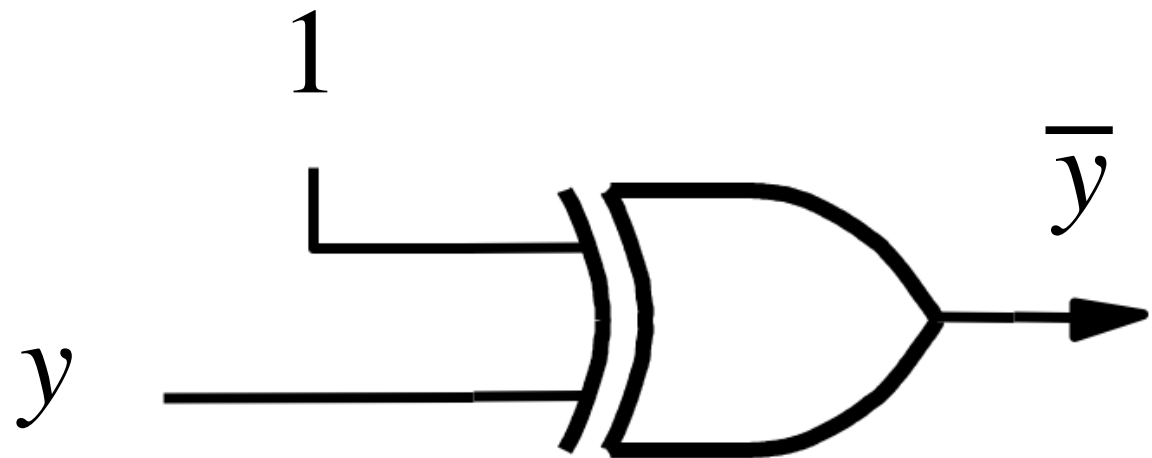
XOR as a repeater

control	y	out
0	0	0
0	1	1



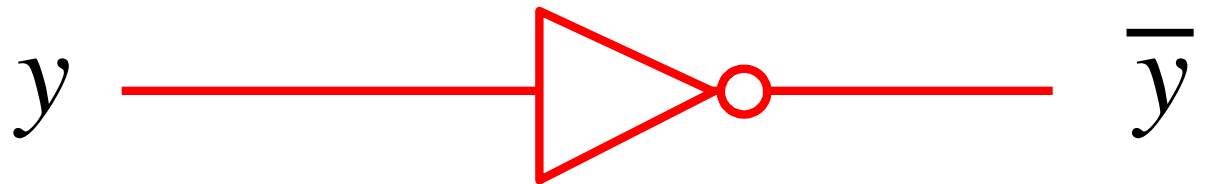
XOR as an inverter

control	y	out
1	0	1
1	1	0

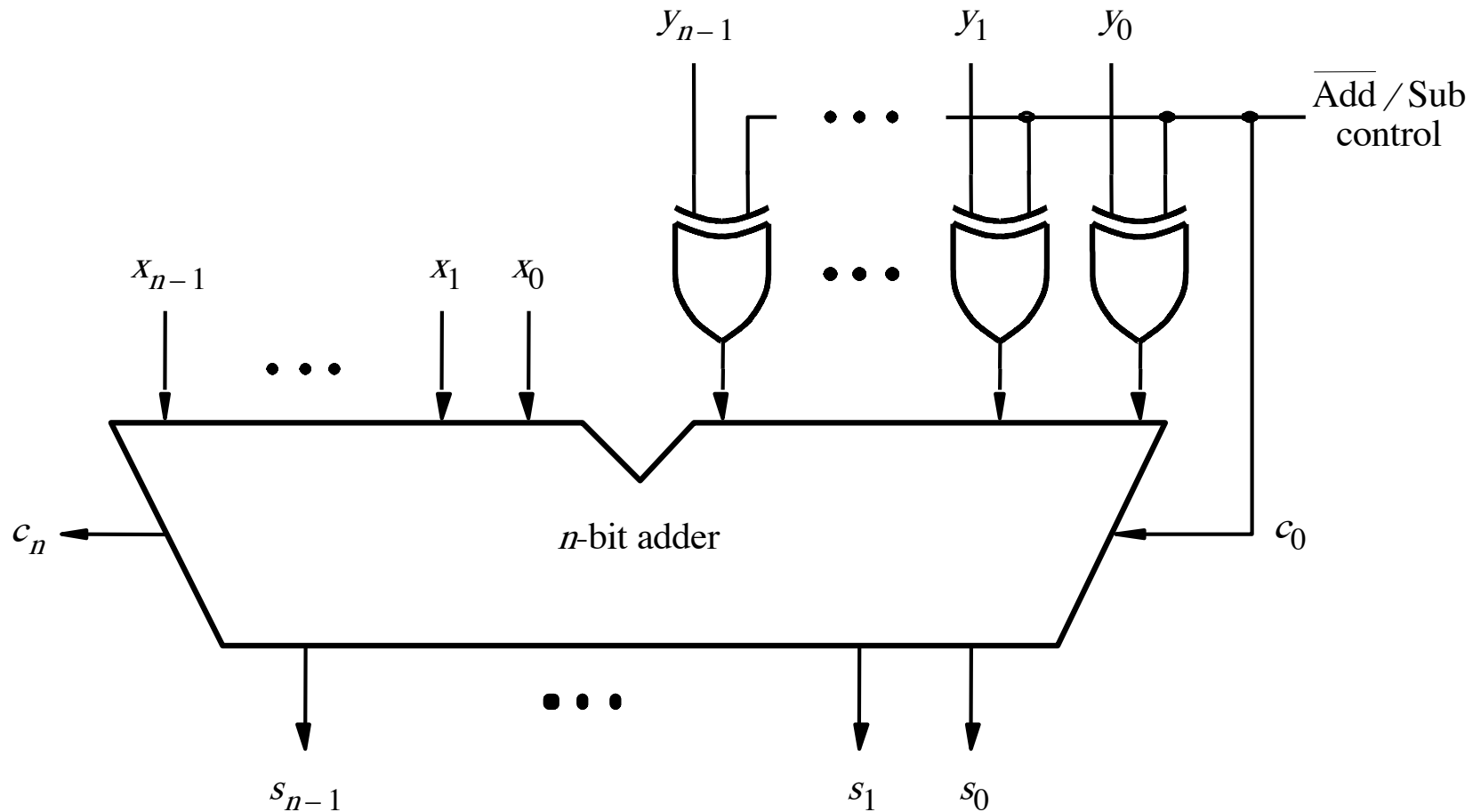


XOR as an inverter

control	y	out
1	0	1
1	1	0

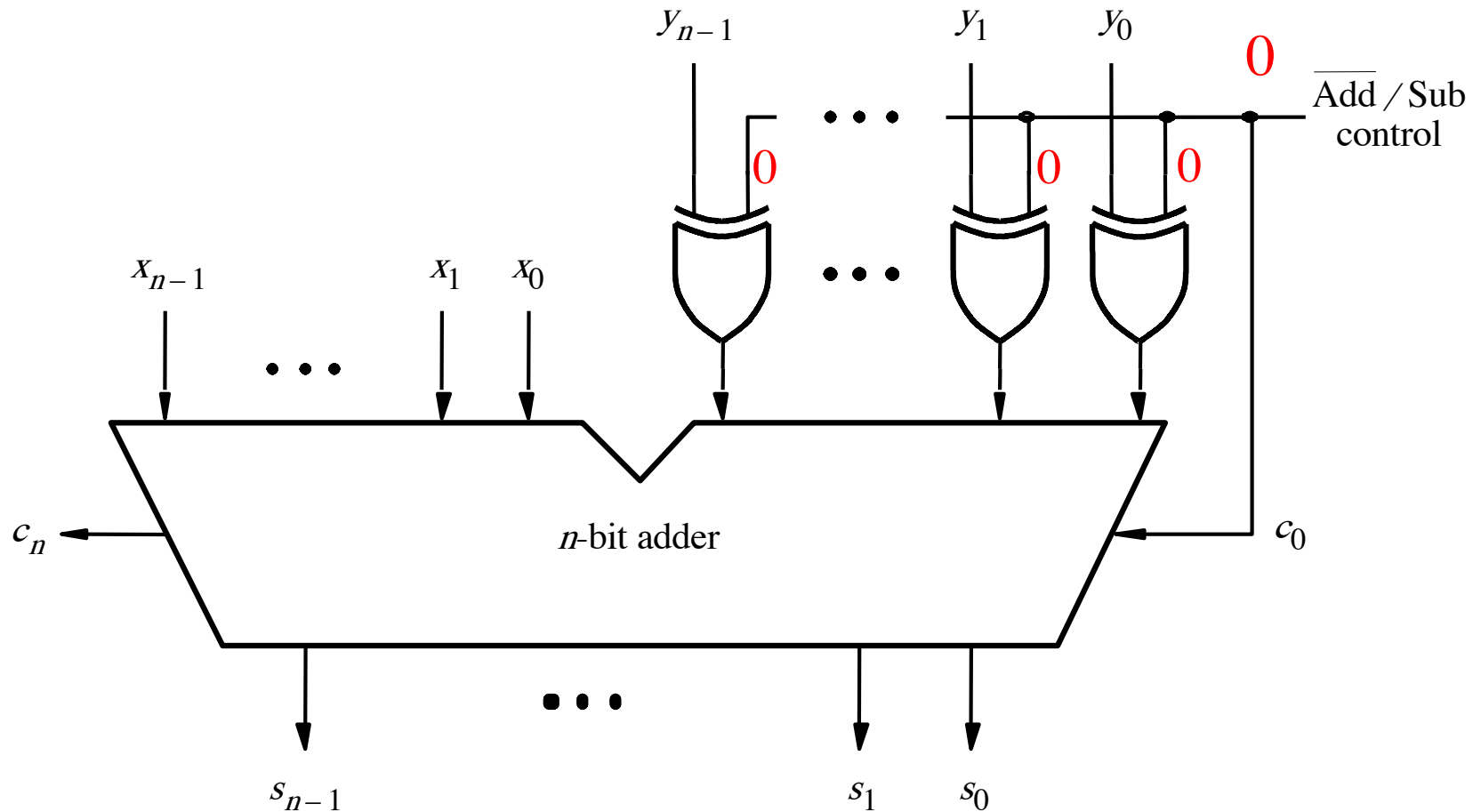


Addition: when control = 0



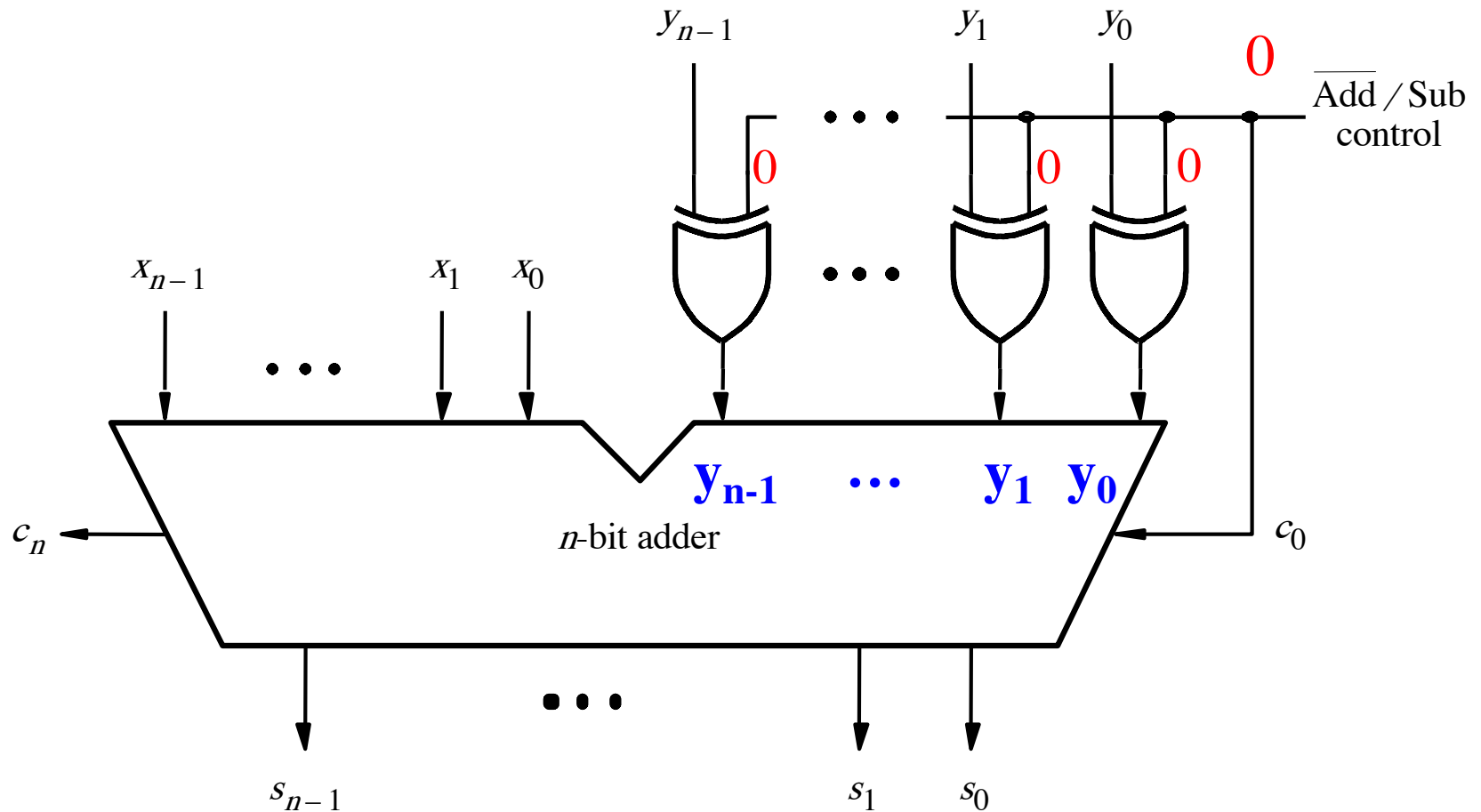
[Figure 3.12 from the textbook]

Addition: when control = 0



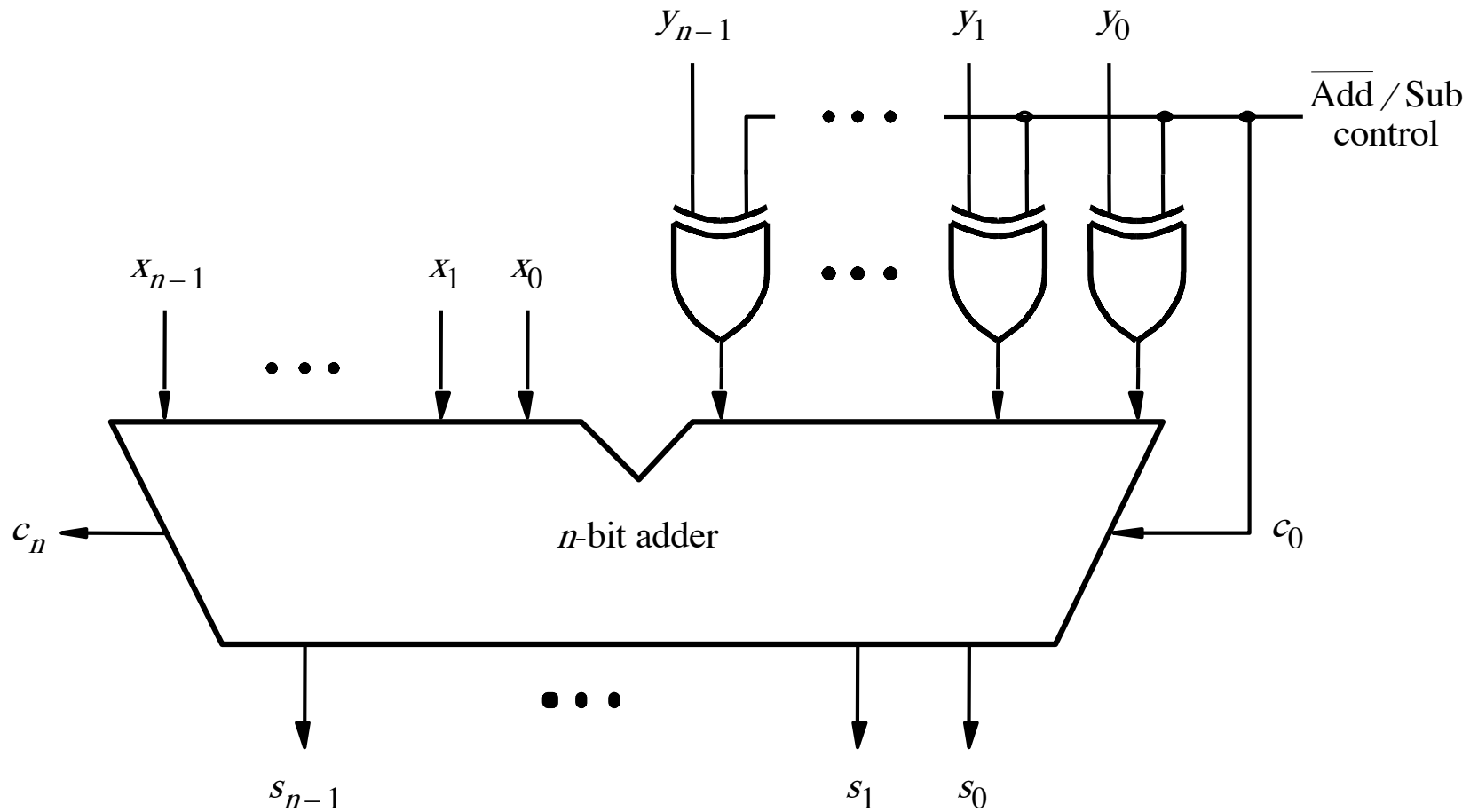
[Figure 3.12 from the textbook]

Addition: when control = 0



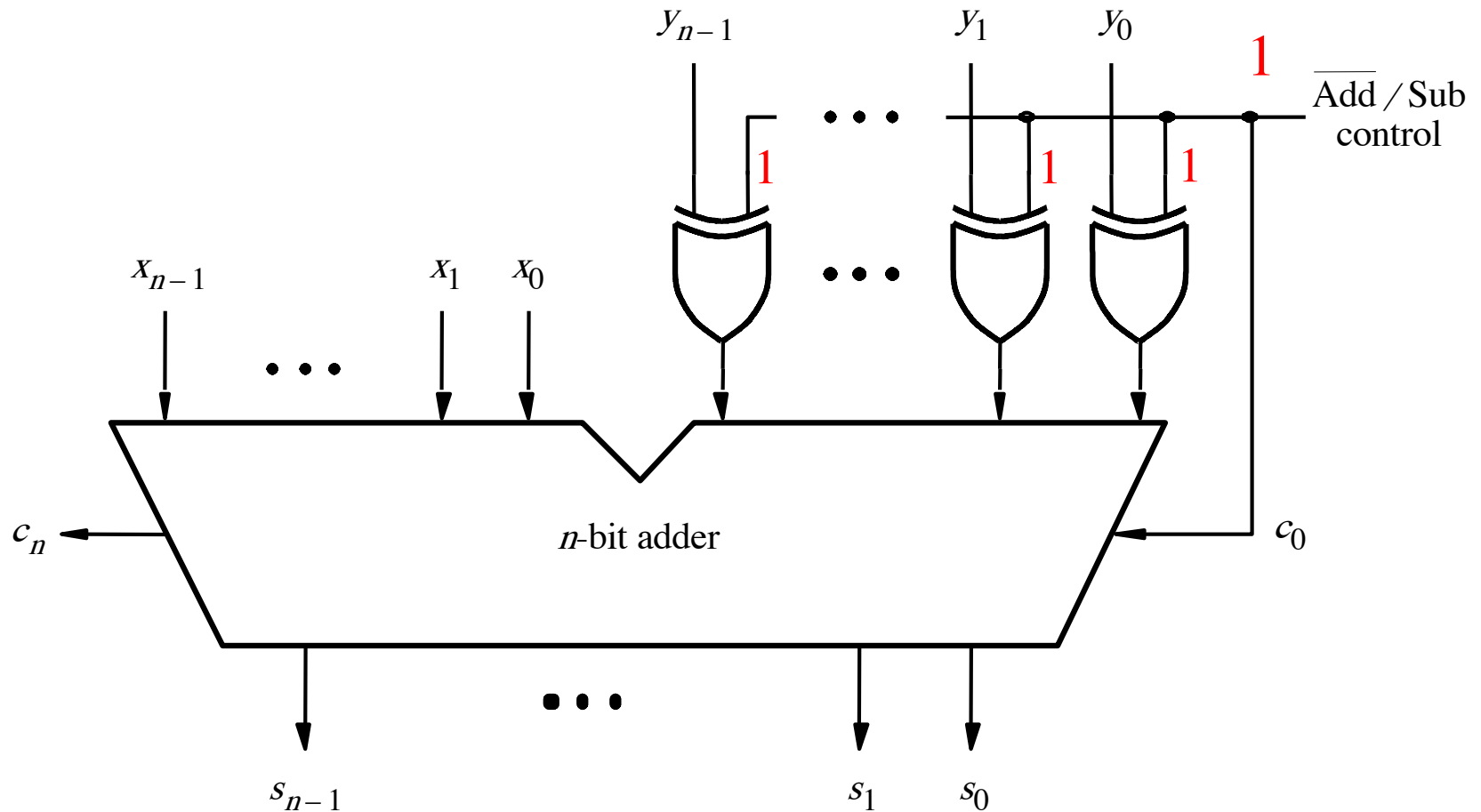
[Figure 3.12 from the textbook]

Subtraction: when control = 1



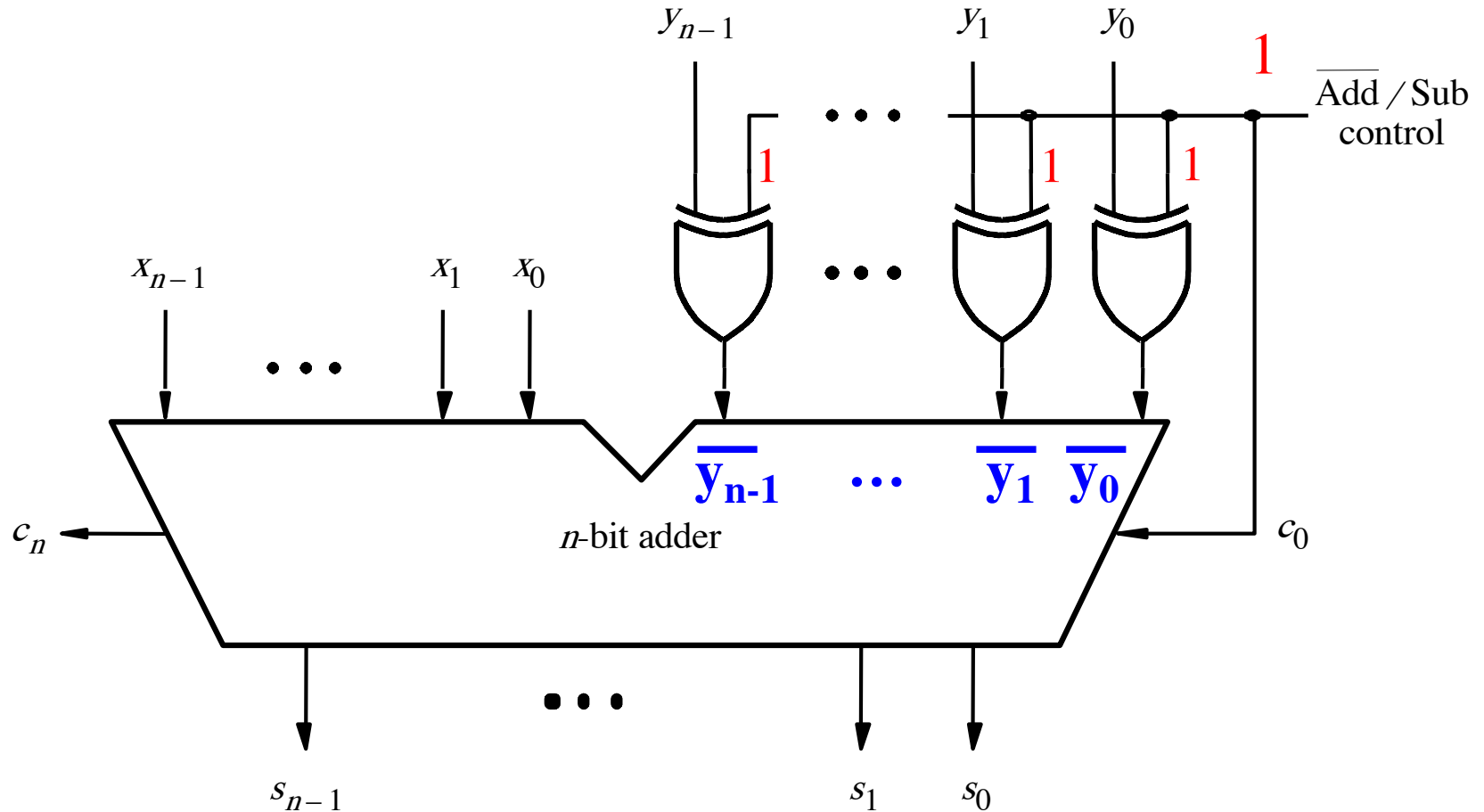
[Figure 3.12 from the textbook]

Subtraction: when control = 1



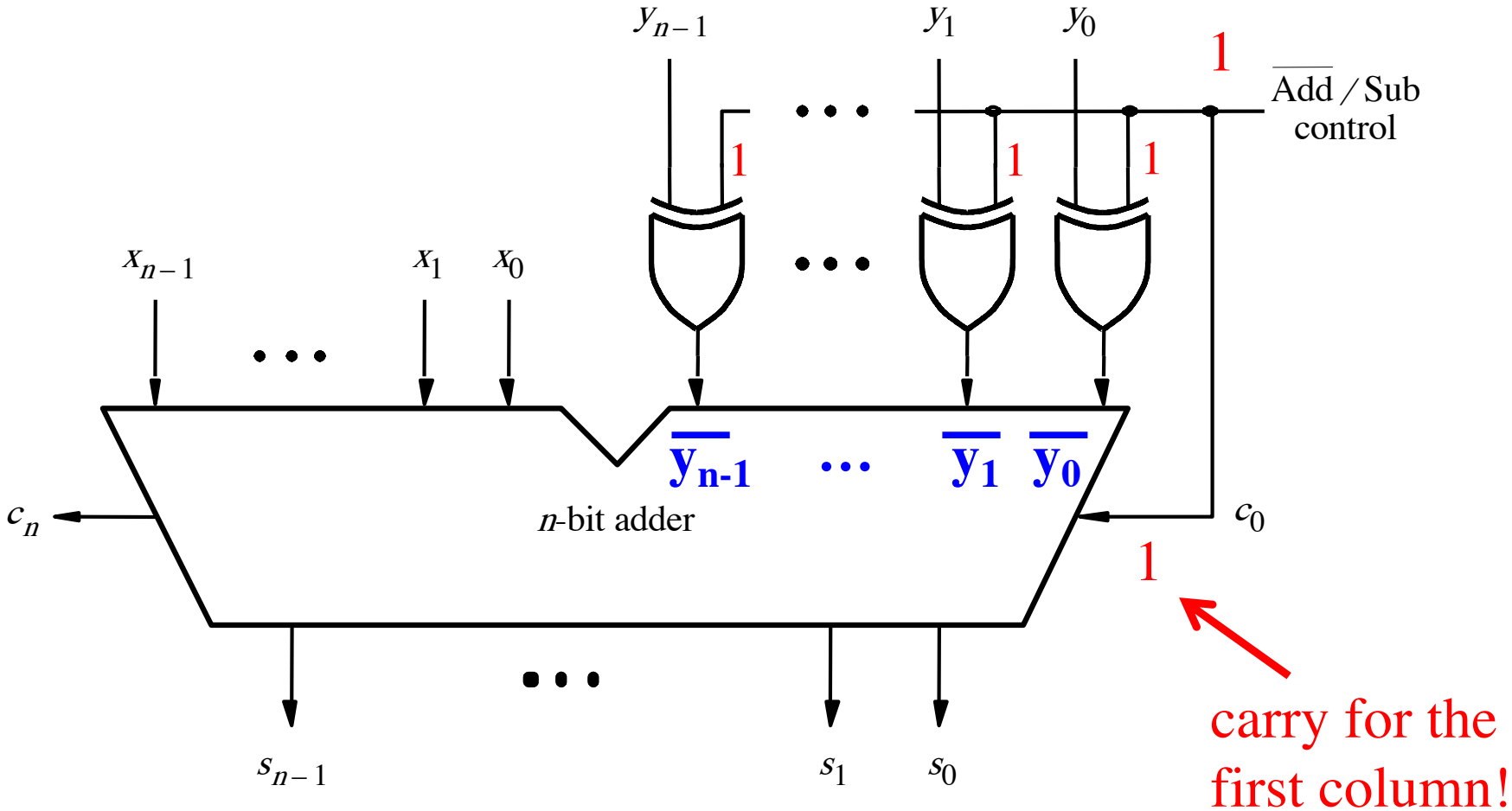
[Figure 3.12 from the textbook]

Subtraction: when control = 1



[Figure 3.12 from the textbook]

Subtraction: when control = 1



[Figure 3.12 from the textbook]

Overflow Detection

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

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

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

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

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

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

Examples of determination of overflow

$$\begin{array}{r}
 (+7) \\
 + (+2) \\
 \hline
 (+9)
 \end{array}
 \quad
 \begin{array}{r}
 01100 \\
 + 0111 \\
 + 0010 \\
 \hline
 1001
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (+2) \\
 \hline
 (-5)
 \end{array}
 \quad
 \begin{array}{r}
 00000 \\
 + 1001 \\
 + 0010 \\
 \hline
 1011
 \end{array}$$

$$\begin{array}{r}
 (+7) \\
 + (-2) \\
 \hline
 (+5)
 \end{array}
 \quad
 \begin{array}{r}
 11100 \\
 + 0111 \\
 + 1110 \\
 \hline
 10101
 \end{array}$$

$$\begin{array}{r}
 (-7) \\
 + (-2) \\
 \hline
 (-9)
 \end{array}
 \quad
 \begin{array}{r}
 10000 \\
 + 1001 \\
 + 1110 \\
 \hline
 10111
 \end{array}$$

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

Examples of determination of overflow

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

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

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

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

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

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

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

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

$$c_4 = 1$$

$$c_3 = 1$$

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

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

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

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

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

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

$$c_4 = 0$$

$$c_3 = 0$$

$$c_4 = 1$$

$$c_3 = 1$$

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

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

$$c_4 = 1$$

$$c_3 = 0$$

Overflow occurs only in these two cases.

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

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

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

$$c_4 = 0$$

$$c_3 = 0$$

$$c_4 = 1$$

$$c_3 = 1$$

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

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

$$c_4 = 1$$

$$c_3 = 0$$

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

Examples of determination of overflow

$$c_4 = 0$$

$$c_3 = 1$$

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

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

$$c_4 = 0$$

$$c_3 = 0$$

$$c_4 = 1$$

$$c_3 = 1$$


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

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

$$c_4 = 1$$

$$c_3 = 0$$

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



XOR

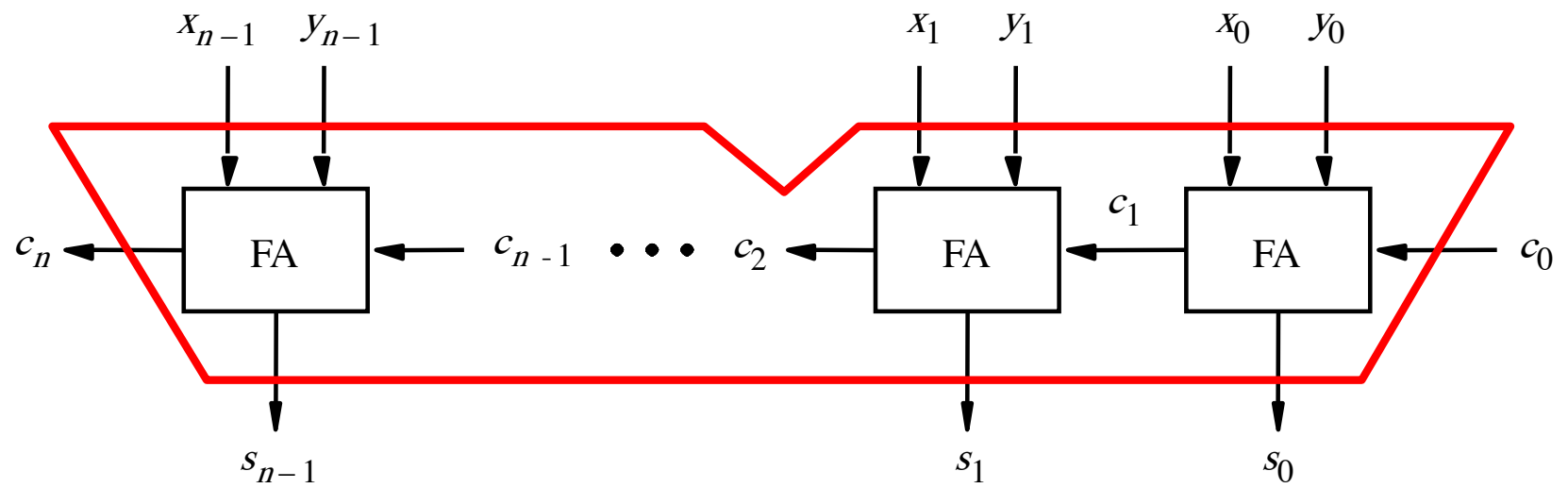
Calculating overflow for 4-bit numbers with only three significant bits

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

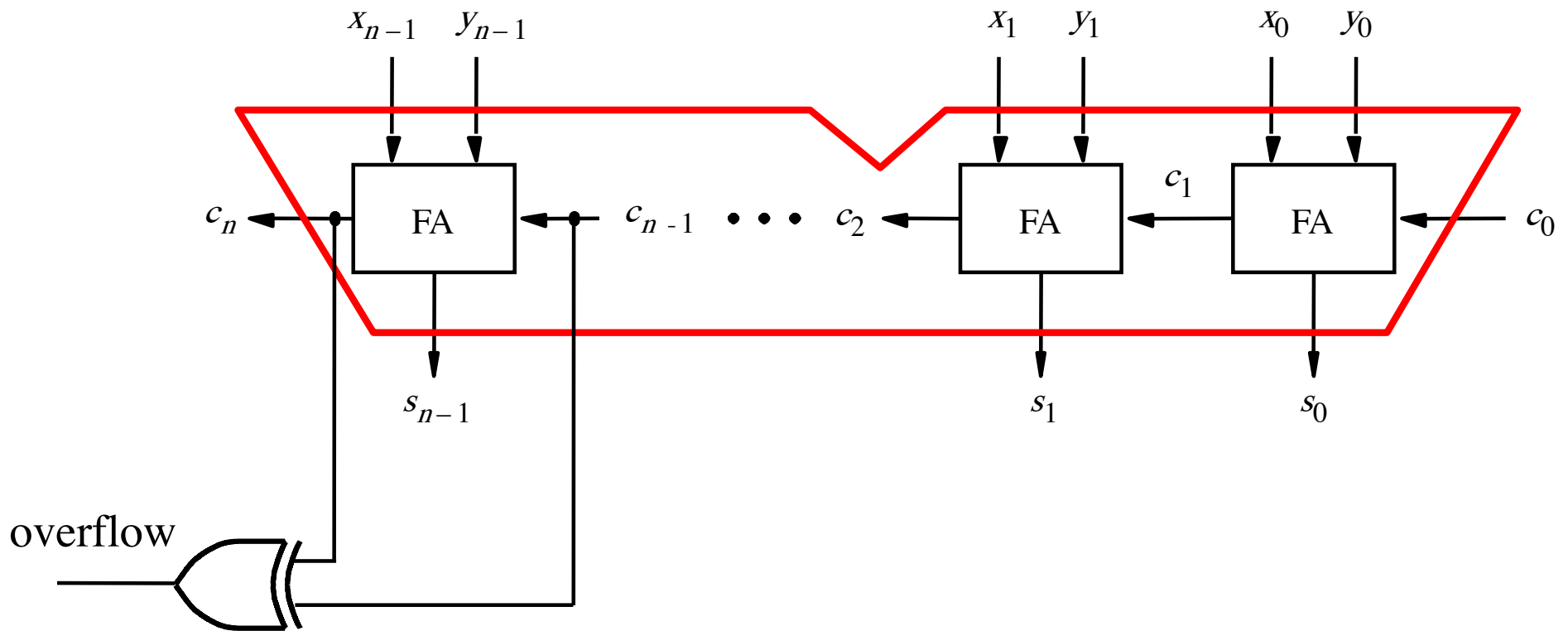
Calculating overflow for n-bit numbers with only n-1 significant bits

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

Detecting Overflow

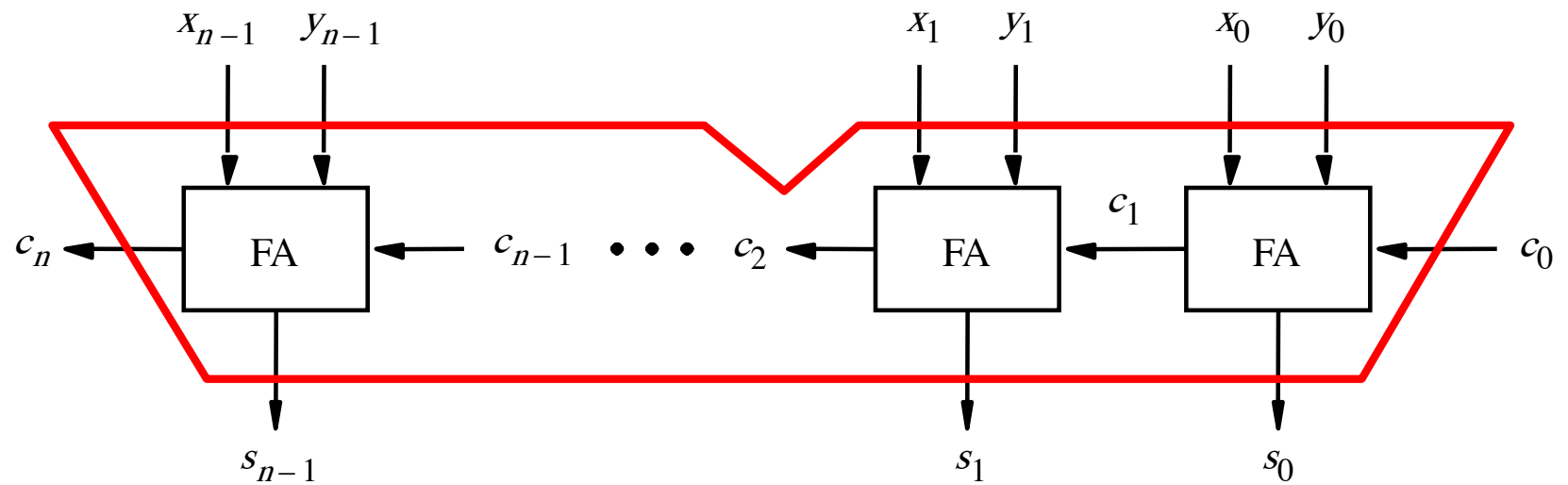


Detecting Overflow (with one extra XOR)

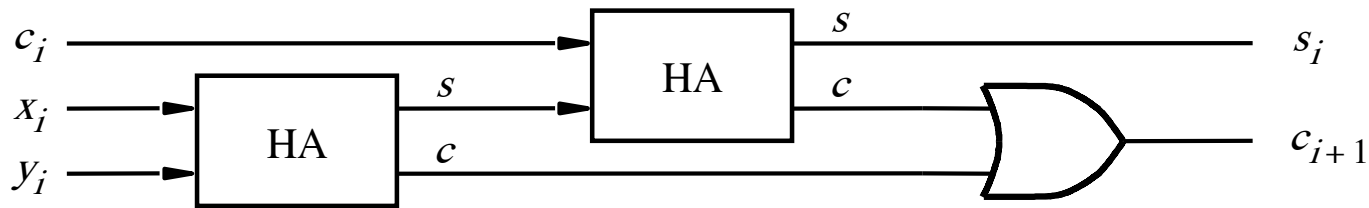


A ripple-carry adder

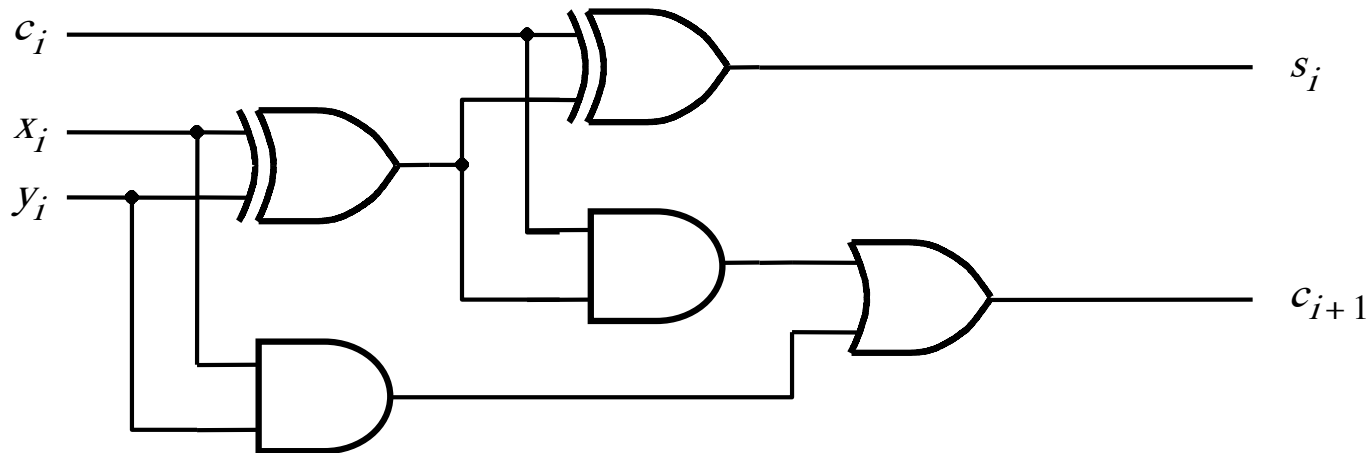
How long does it take to compute all sum bits and all carry bits?



Delays through the modular implementation of the full-adder circuit

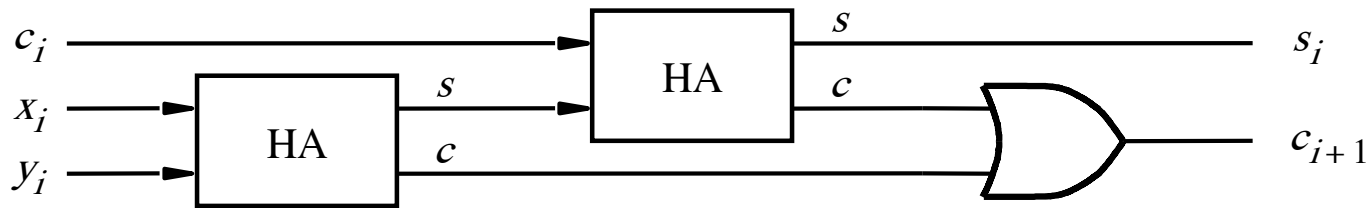


(a) Block diagram

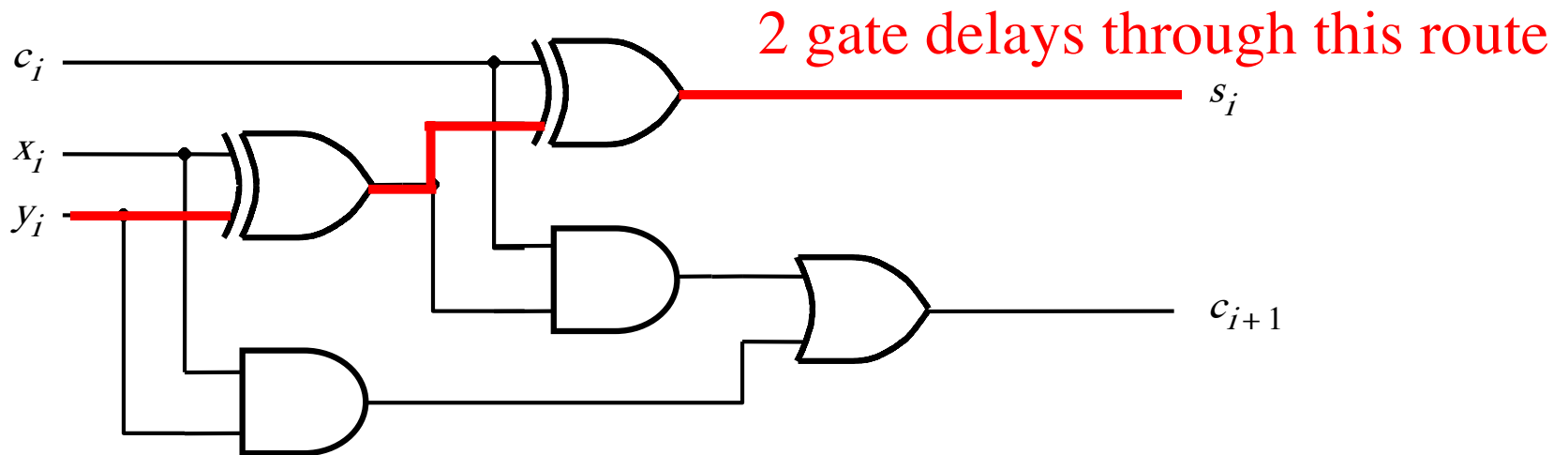


(b) Detailed diagram

Delays through the modular implementation of the full-adder circuit

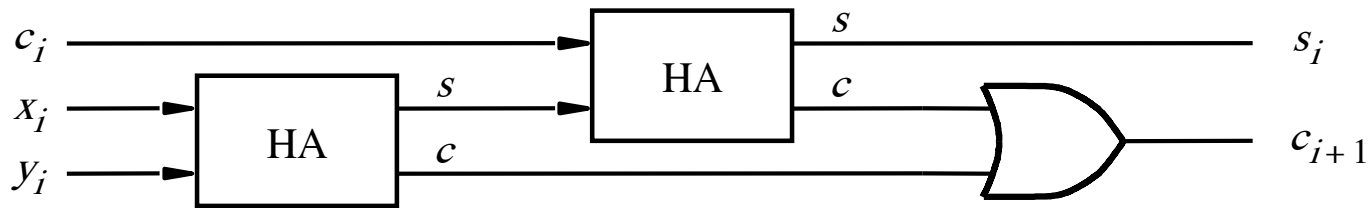


(a) Block diagram

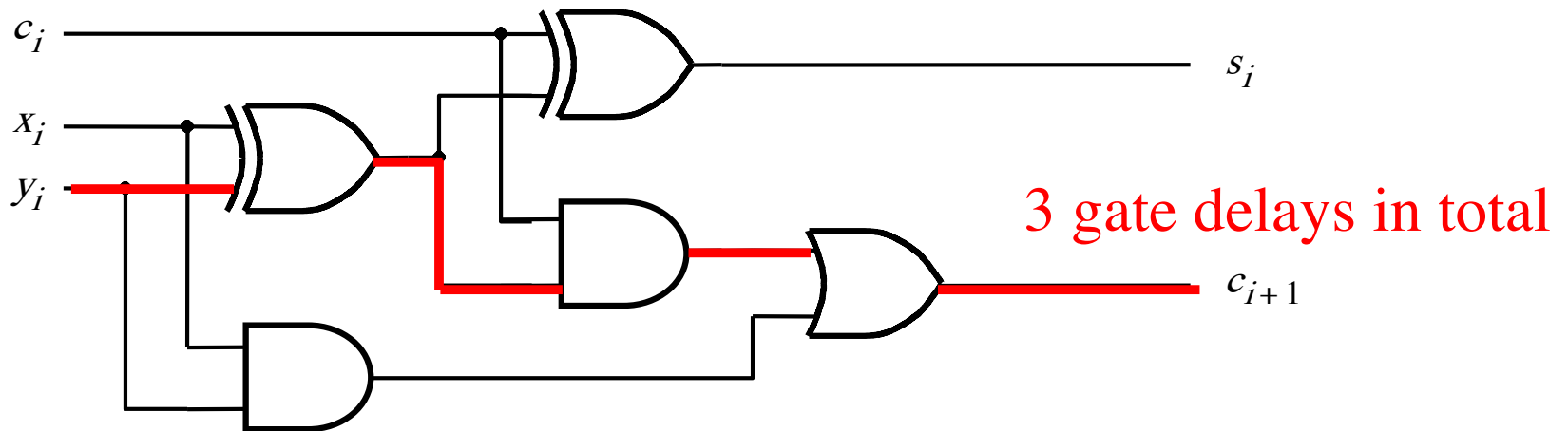


(b) Detailed diagram

Delays through the modular implementation of the full-adder circuit

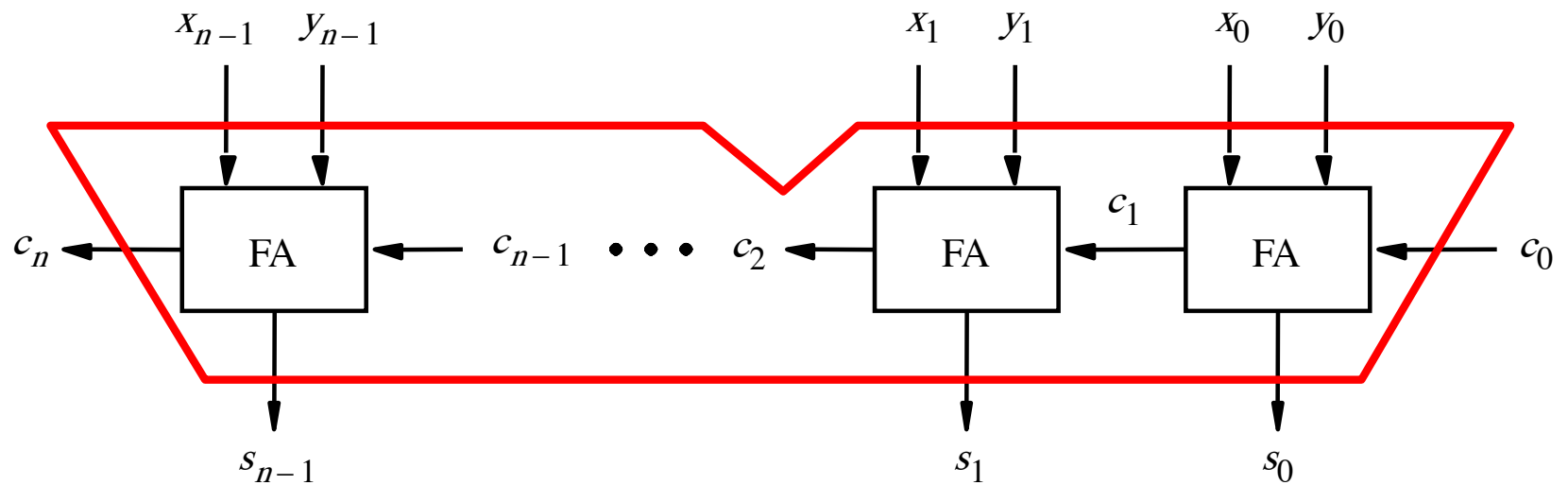


(a) Block diagram



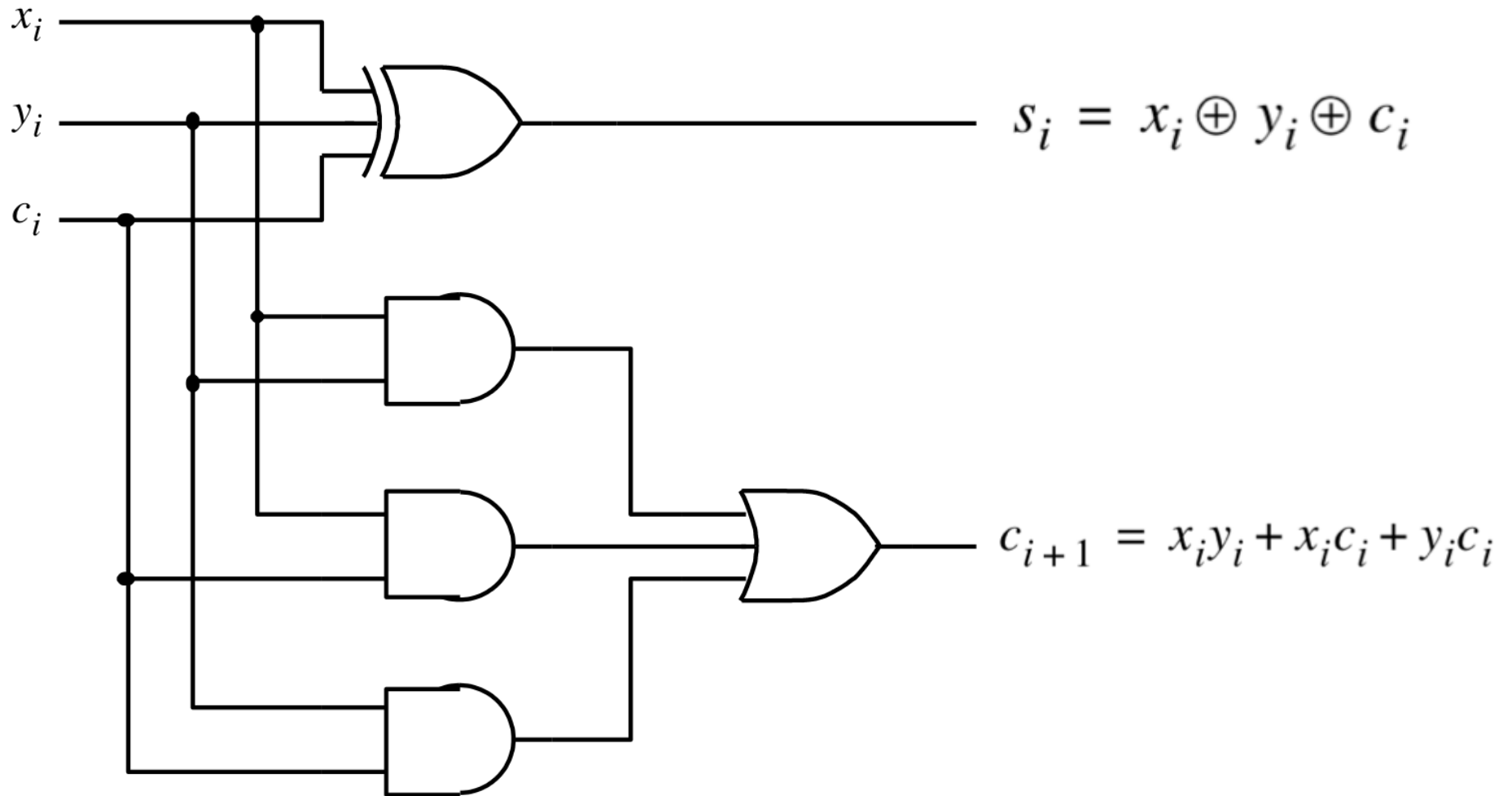
(b) Detailed diagram

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



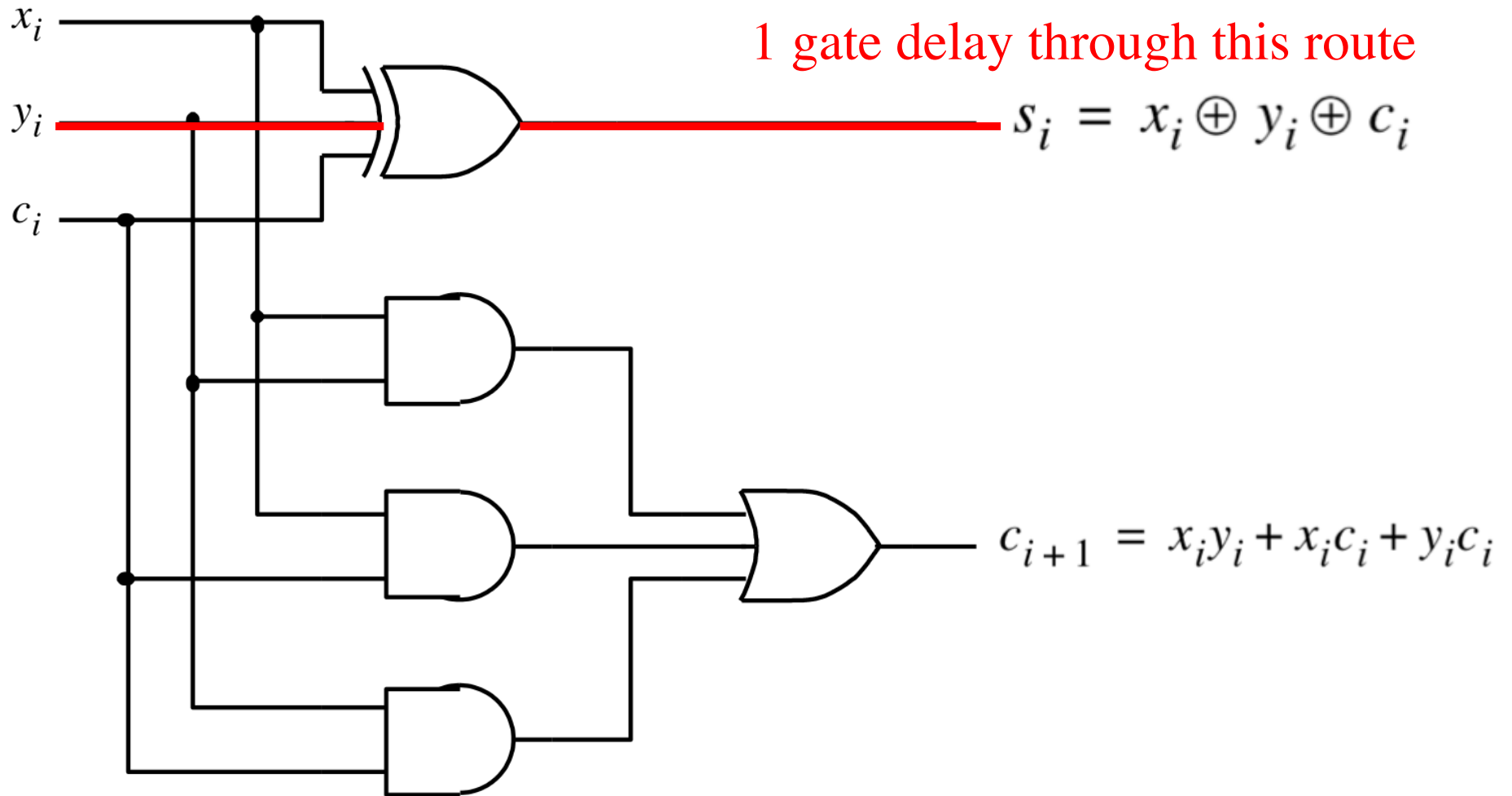
It takes $3n$ gate delays?

Delays through the Full-Adder circuit



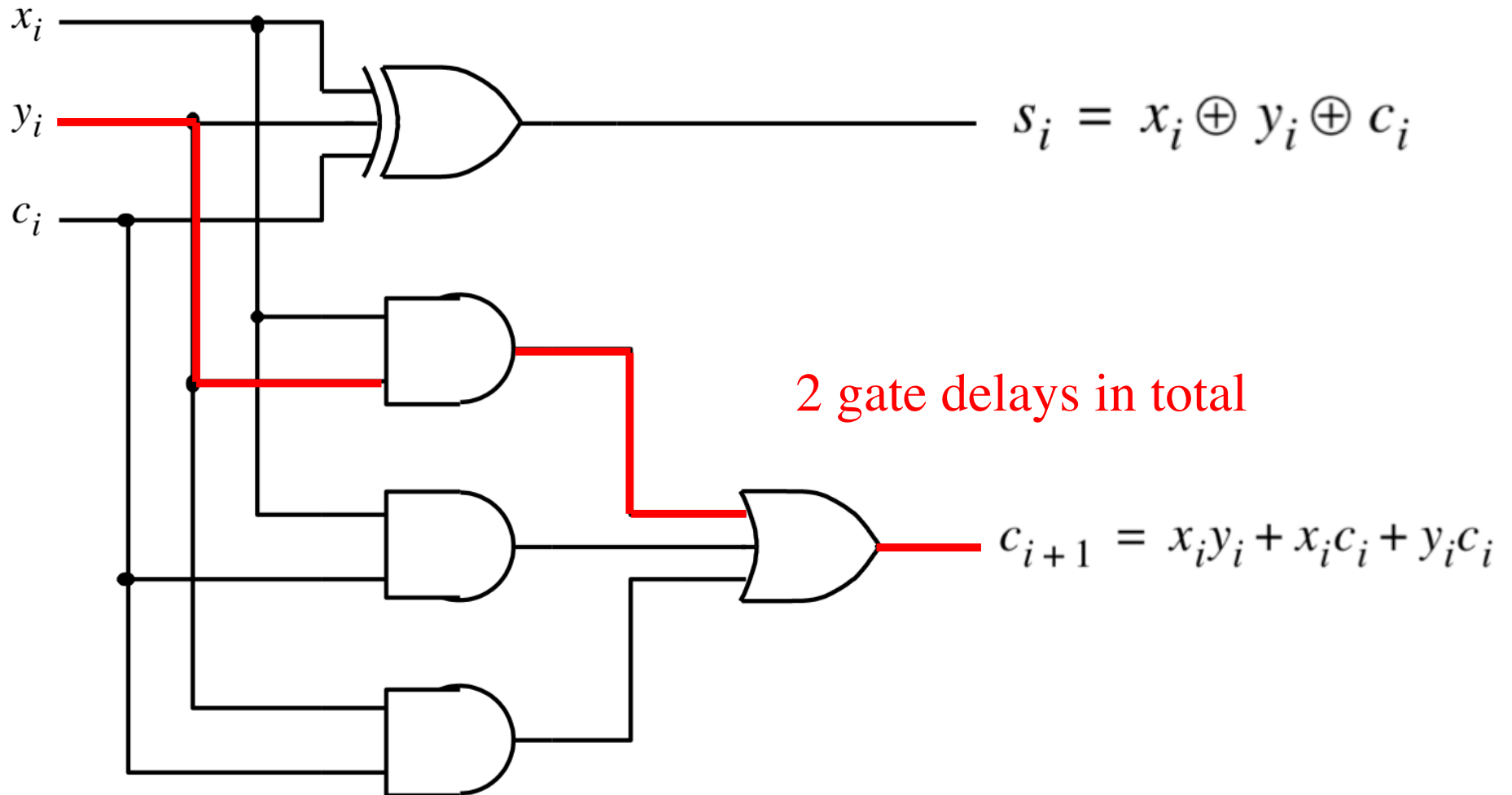
[Figure 3.3c from the textbook]

Delays through the Full-Adder circuit



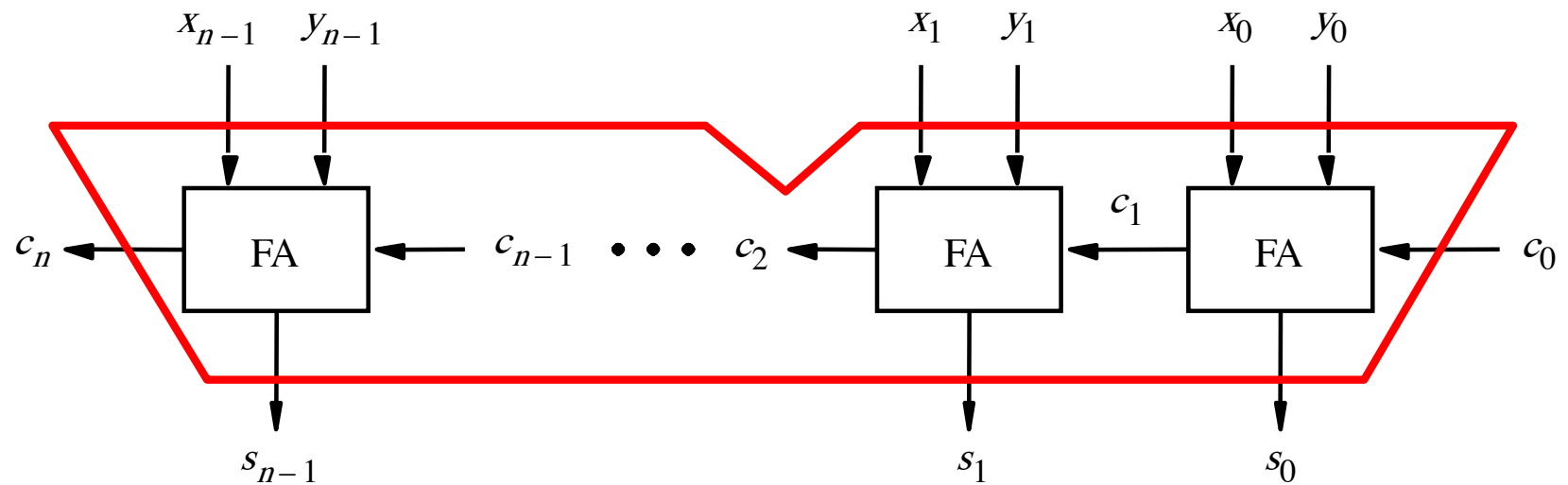
[Figure 3.3c from the textbook]

Delays through the Full-Adder circuit



[Figure 3.3c from the textbook]

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



It takes $2n$ gate delays?

Can we perform addition even faster?

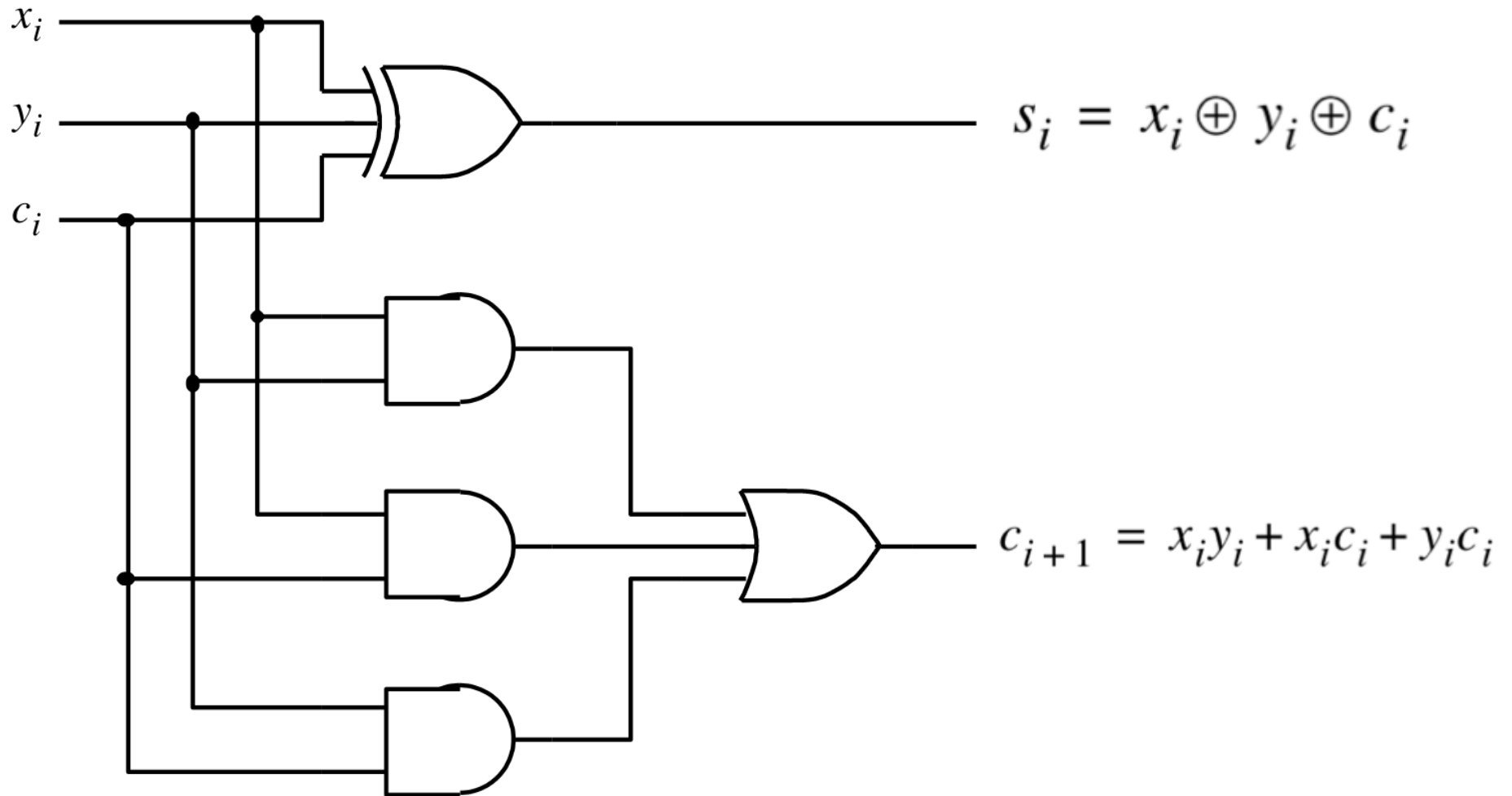
The goal is to evaluate very fast if the carry from the previous stage will be equal to 0 or 1.

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.

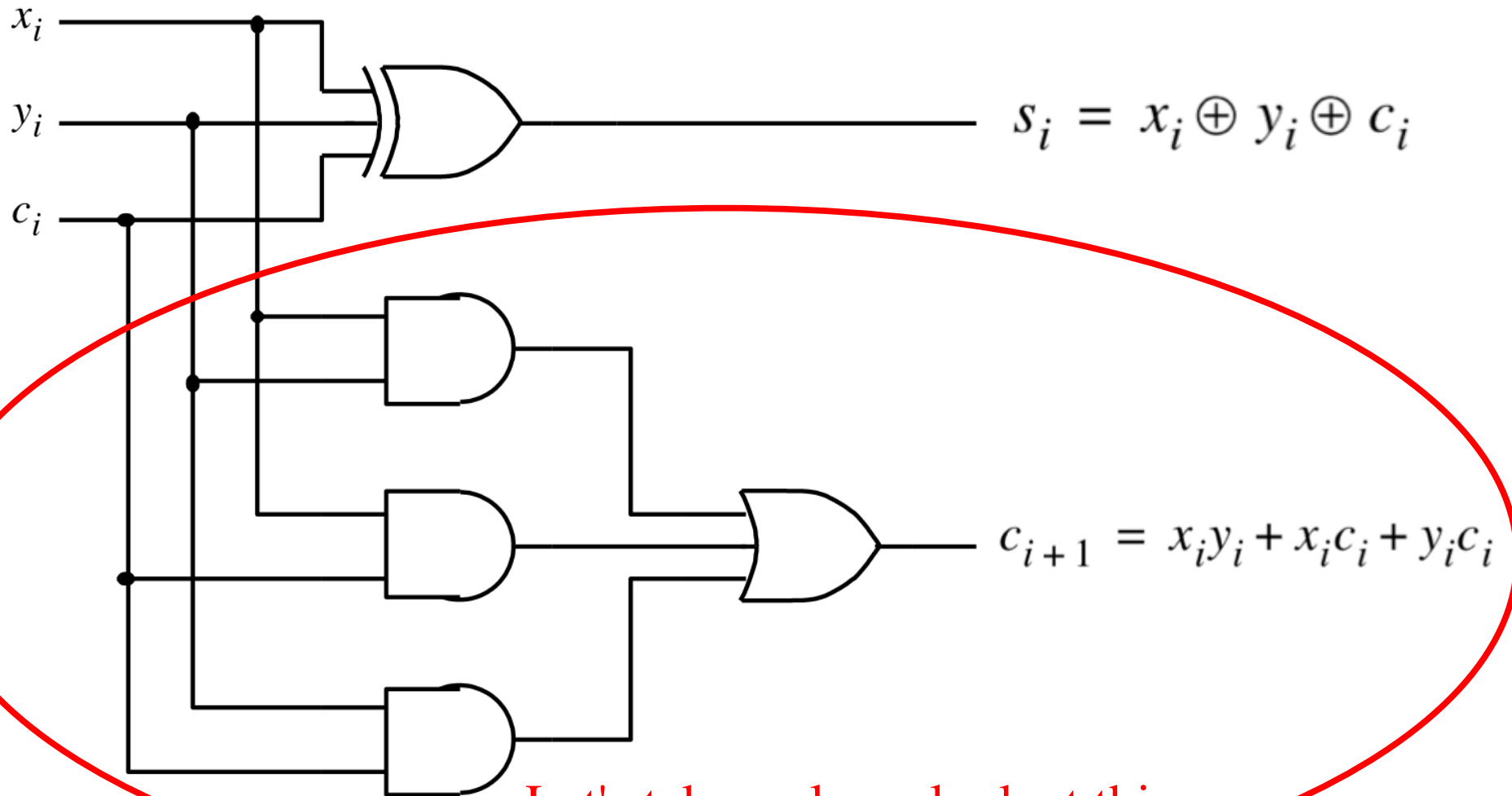
To accomplish this goal we will have to redesign the full-adder circuit yet again.

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

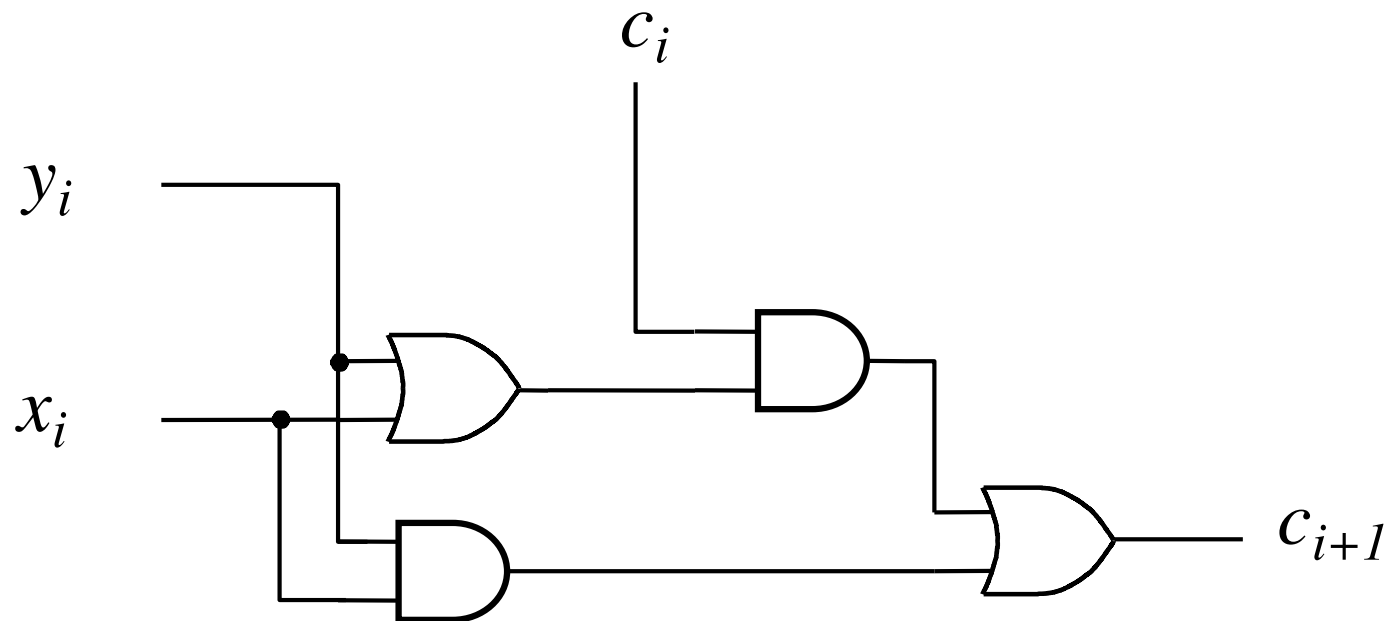
$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

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

Decomposing the Carry Expression

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

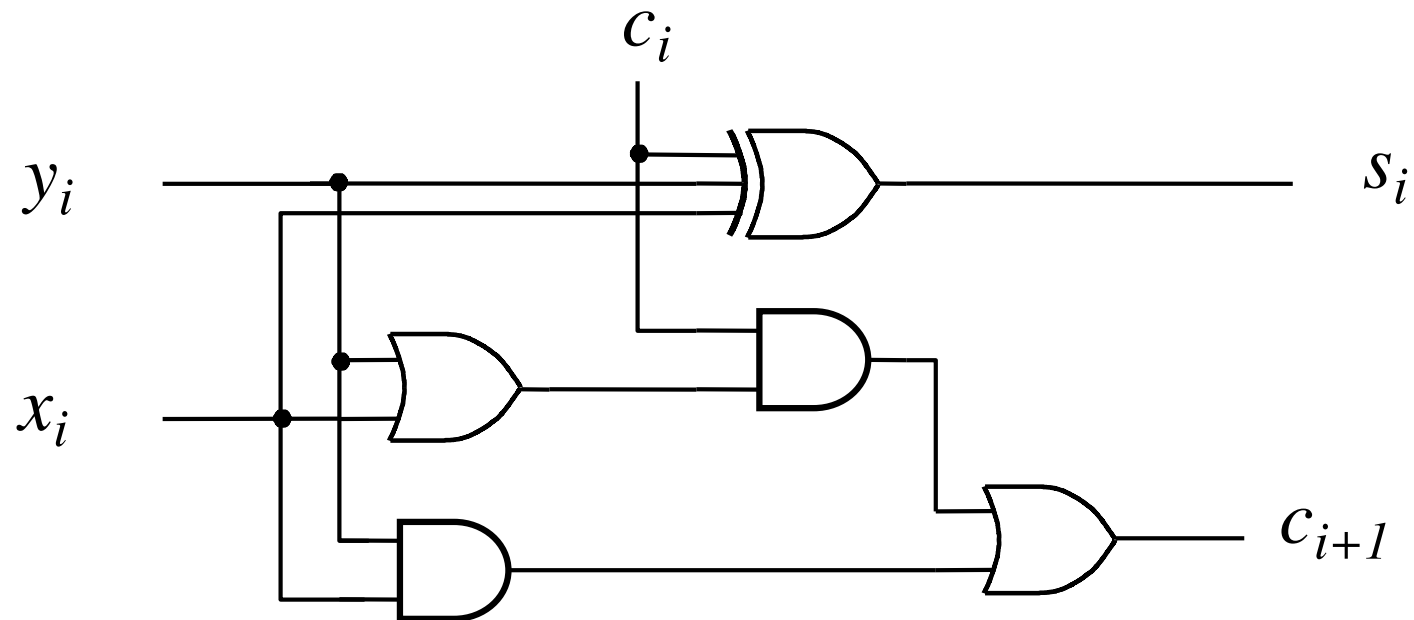
$$C_{i+1} = x_i y_i + (x_i + y_i) c_i$$



Another Way to Draw the Full-Adder Circuit

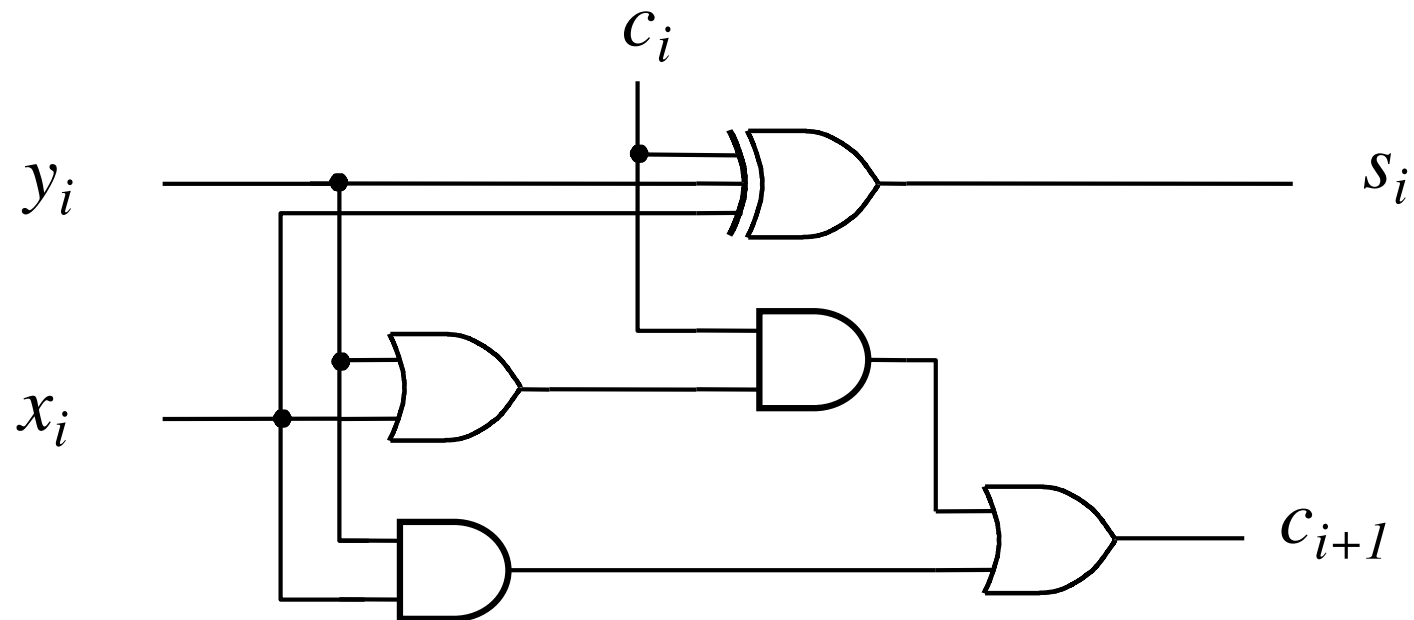
$$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

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



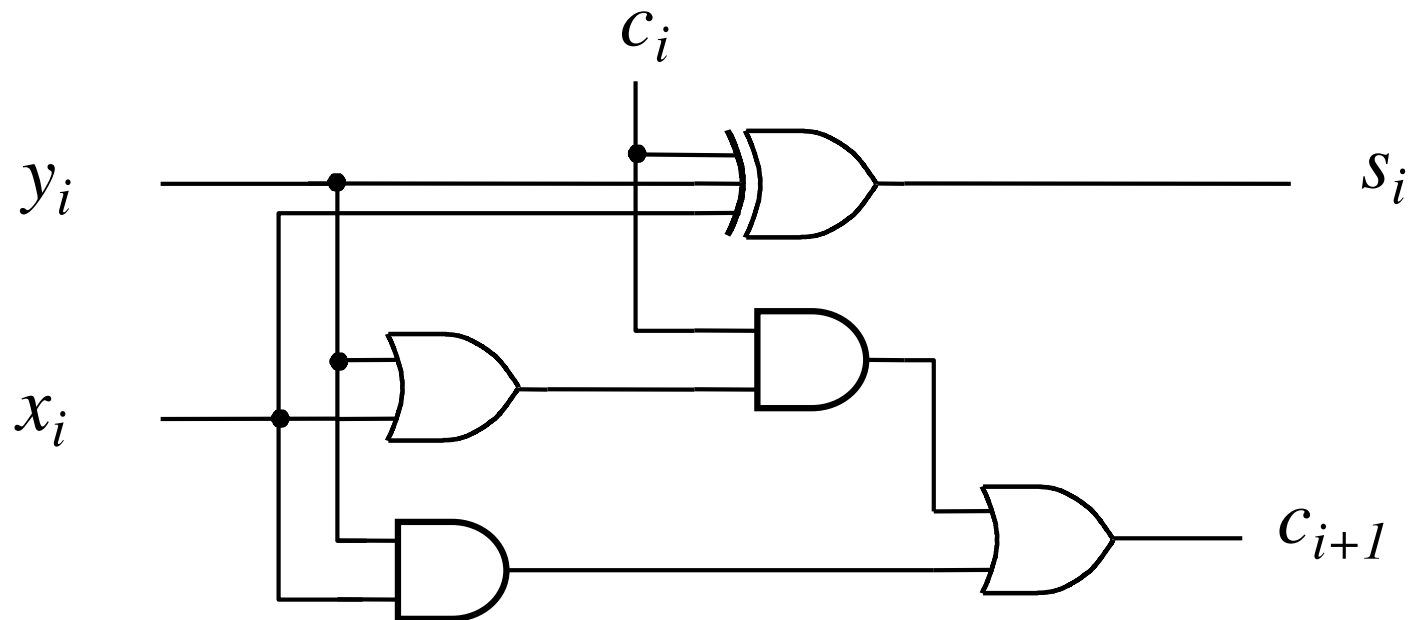
Another Way to Draw the Full-Adder Circuit

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



Another Way to Draw the Full-Adder Circuit

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

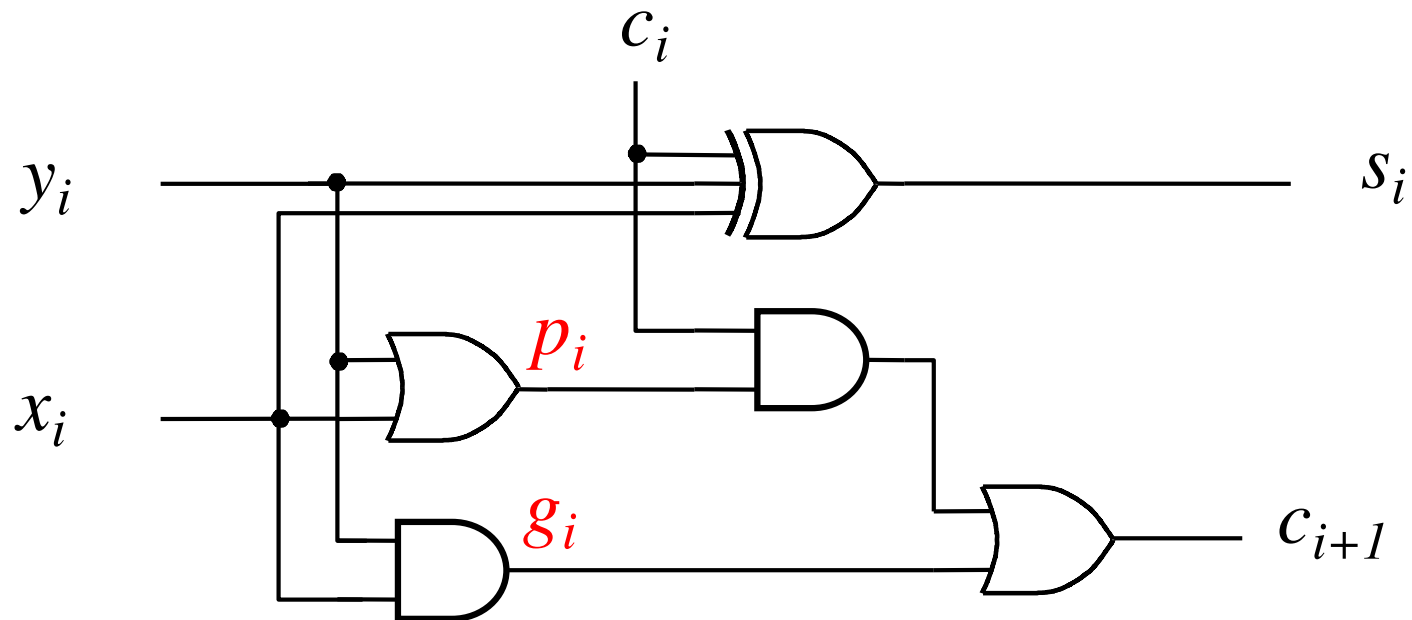


Another Way to Draw the Full-Adder Circuit

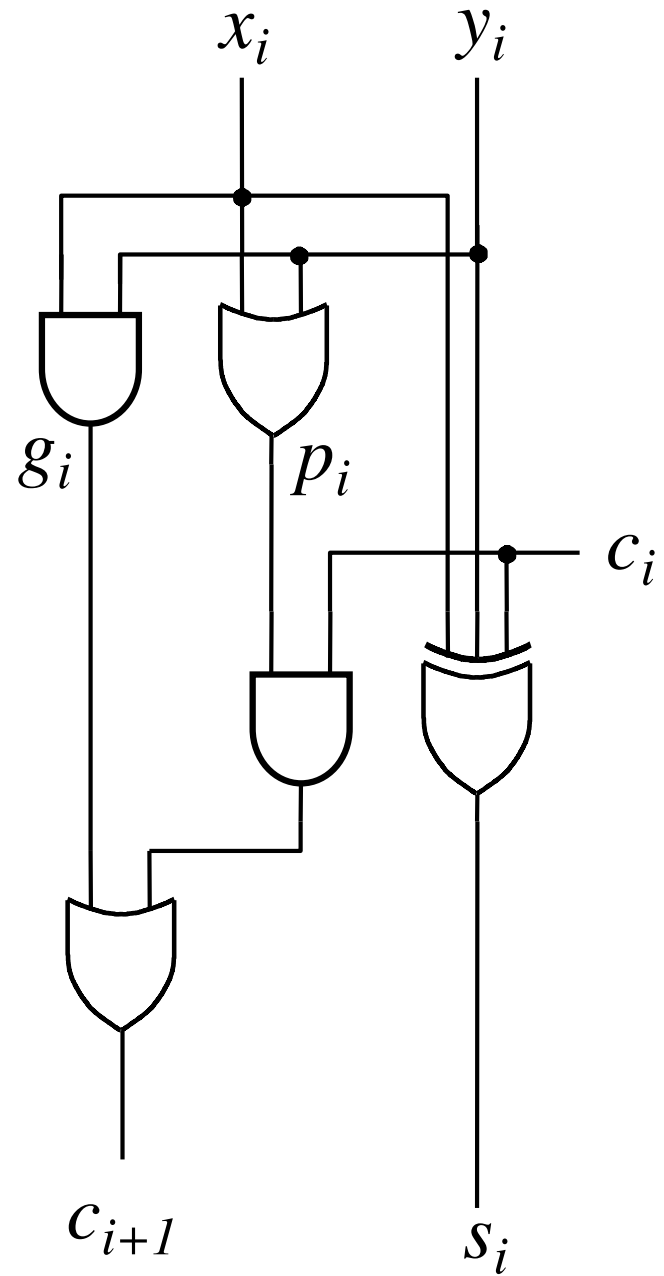
g - generate

p - propagate

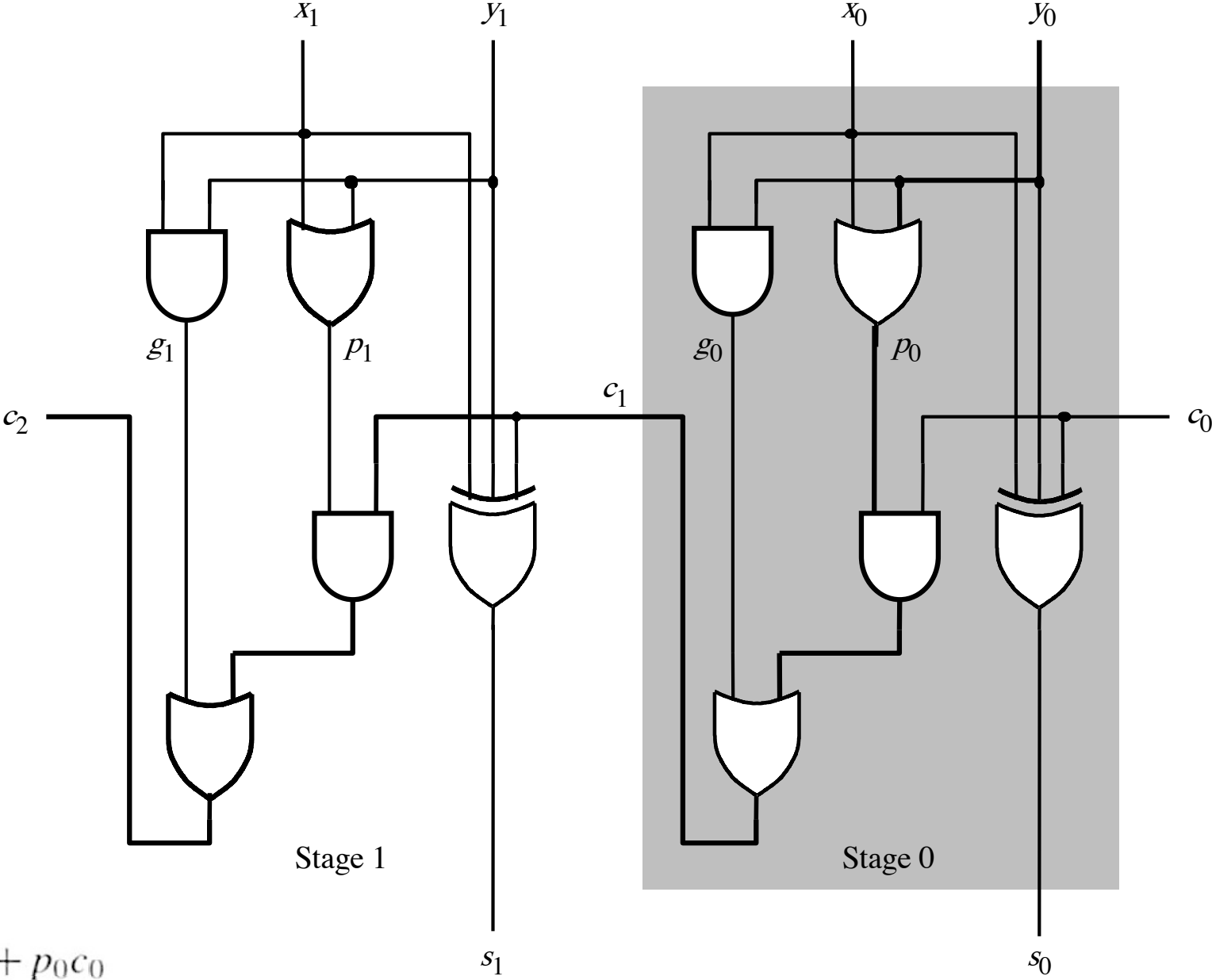
$$C_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} c_i$$



Yet Another Way to Draw It (Just Rotate It)



Now we can Build a Ripple-Carry Adder

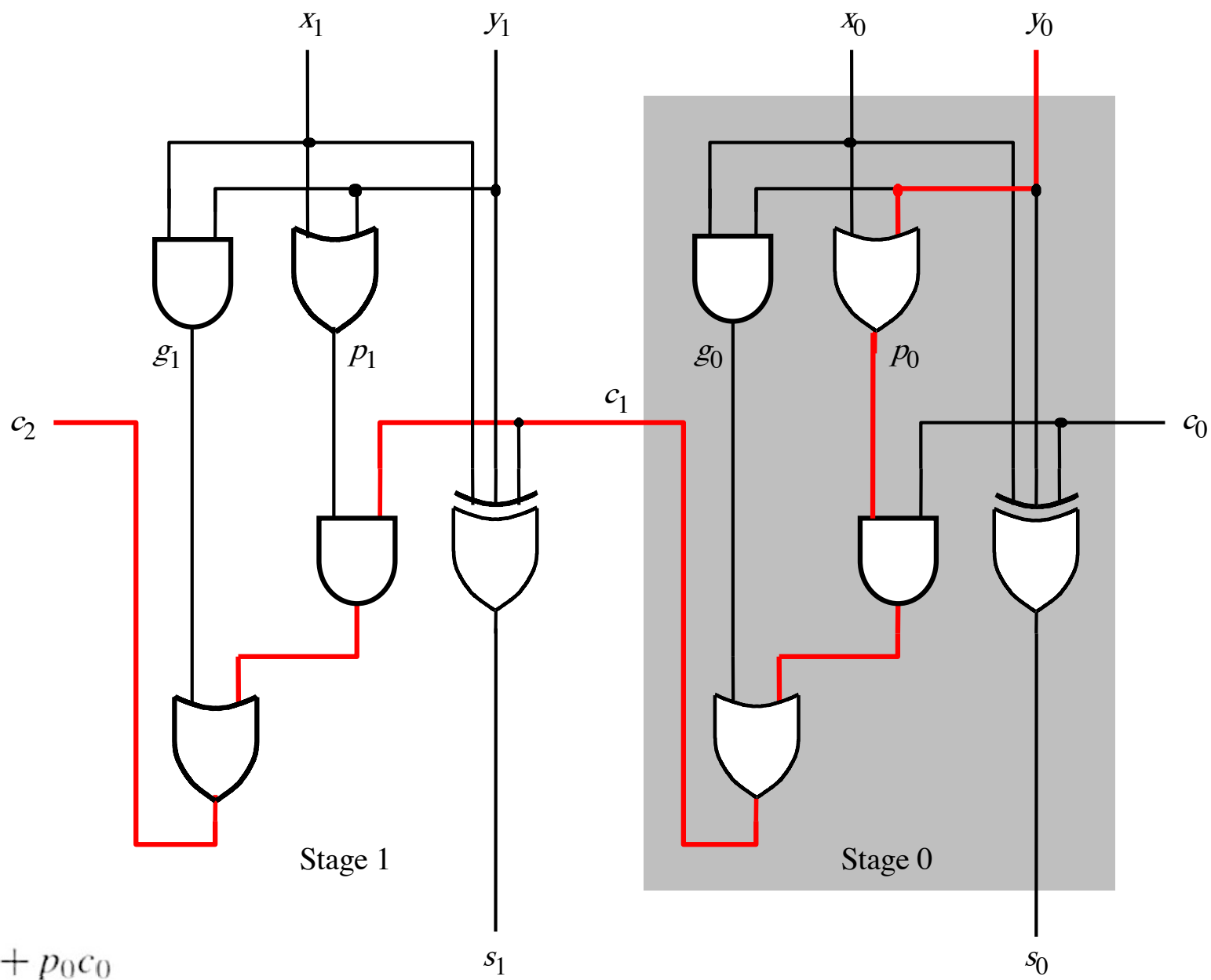


$$c_1 = g_0 + p_0 c_0$$

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

[Figure 3.14 from the textbook]

Now we can Build a Ripple-Carry Adder

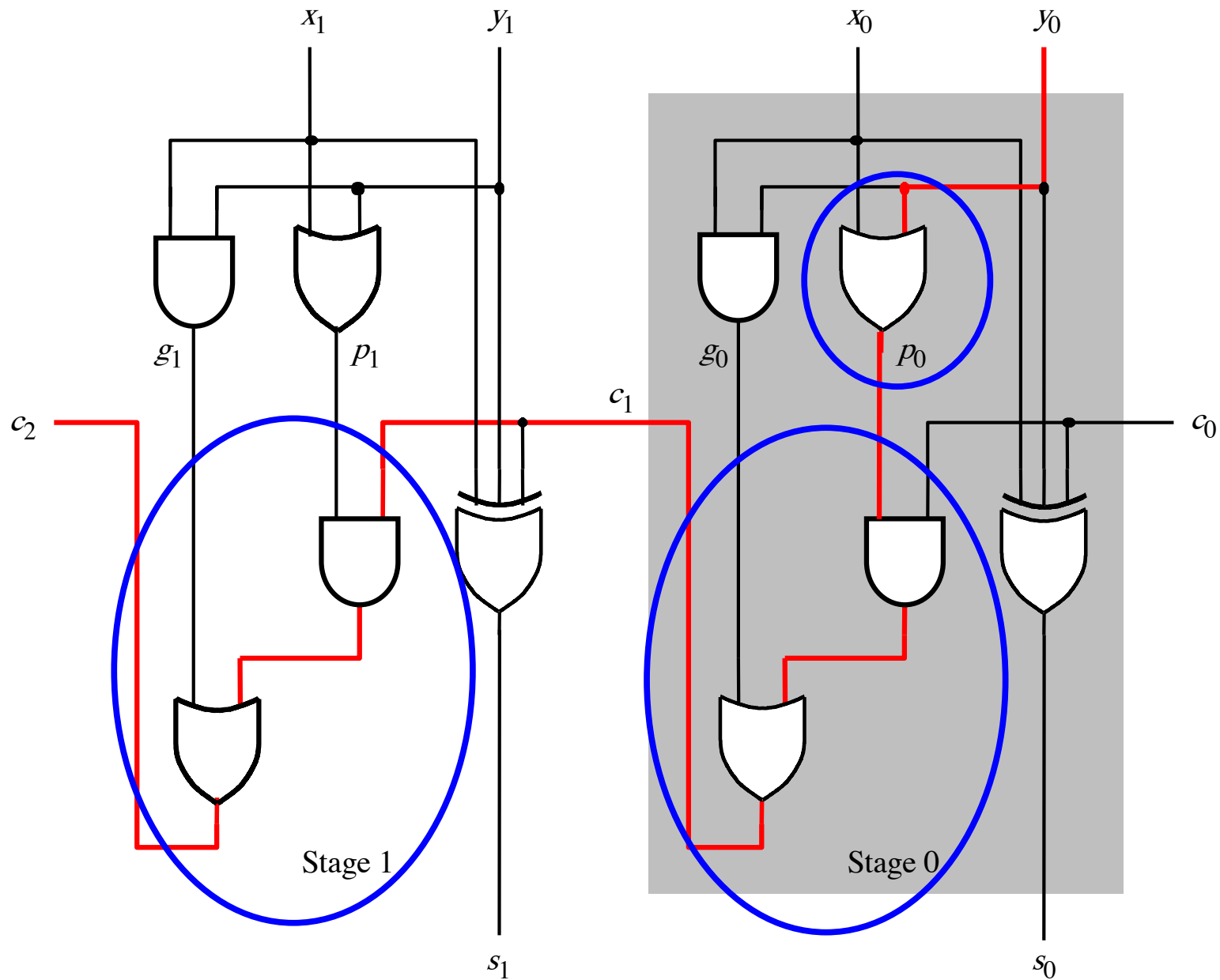


$$c_1 = g_0 + p_0c_0$$

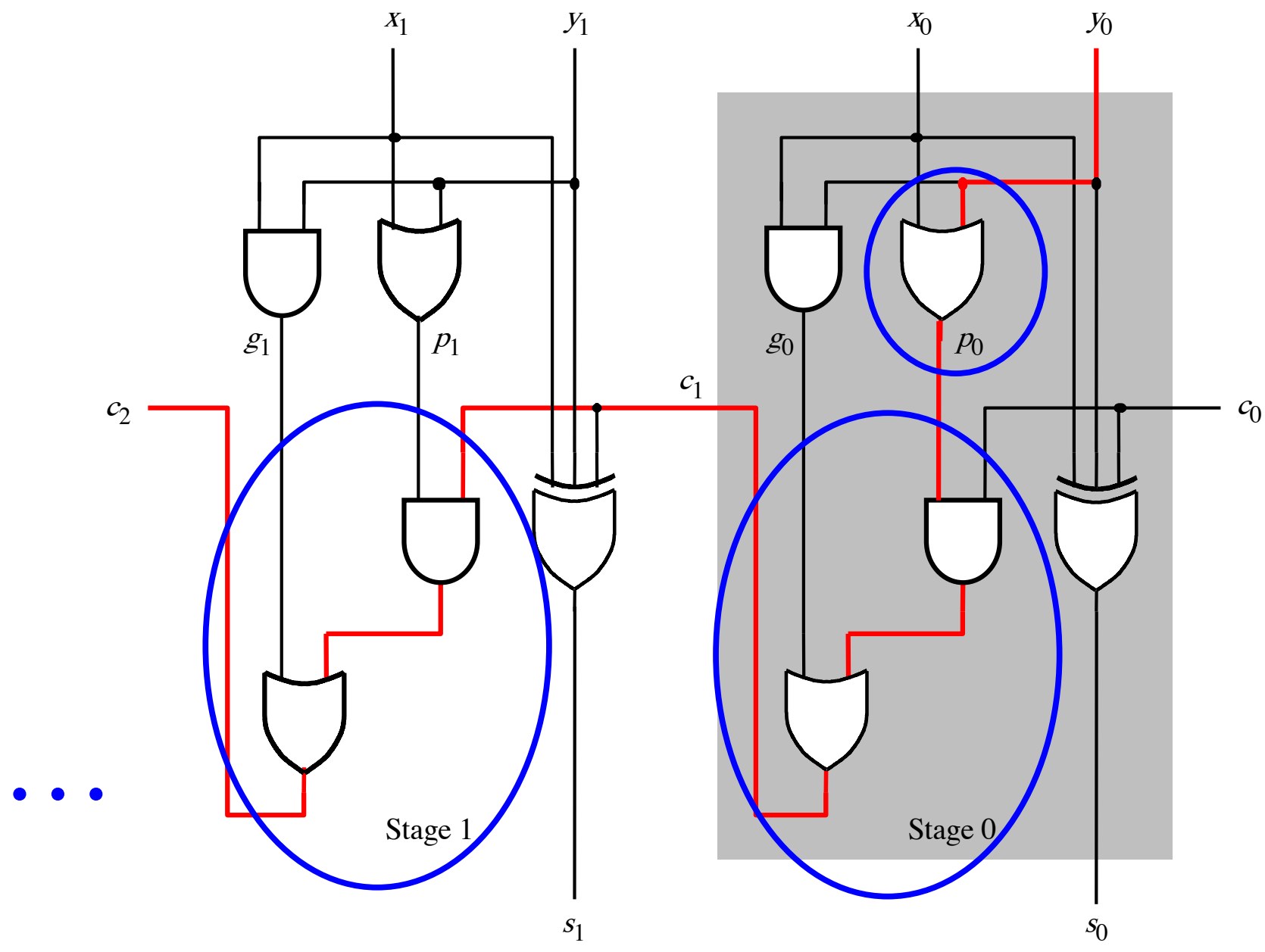
$$c_2 = g_1 + p_1g_0 + p_1p_0c_0$$

[Figure 3.14 from the textbook]

2-bit ripple-carry adder: 5 gate delays (1+2+2)



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



n-bit Ripple-Carry Adder

- **It takes 1 gate delay to generate all g_i and p_i signals**
- **+2 more gate delays to generate carry 1**
- **+2 more gate delay to generate carry 2**
- **...**
- **+2 more gate delay to generate carry n**
- **Thus, the total delay through an n-bit ripple-carry adder is $2n+1$ gate delays!**

n-bit Ripple-Carry Adder

- It takes 1 gate delay to generate all g_i and p_i signals
- +2 more gate delays to generate carry 1
- +2 more gate delay to generate carry 2
- ...
- +2 more gate delay to generate carry n
- Thus, the total delay through an n-bit ripple-carry adder is $2n+1$ gate delays!

This is slower by 1 than the original design?!

A carry-lookahead adder

Decomposing the Carry Expression

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

Decomposing the Carry Expression

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

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

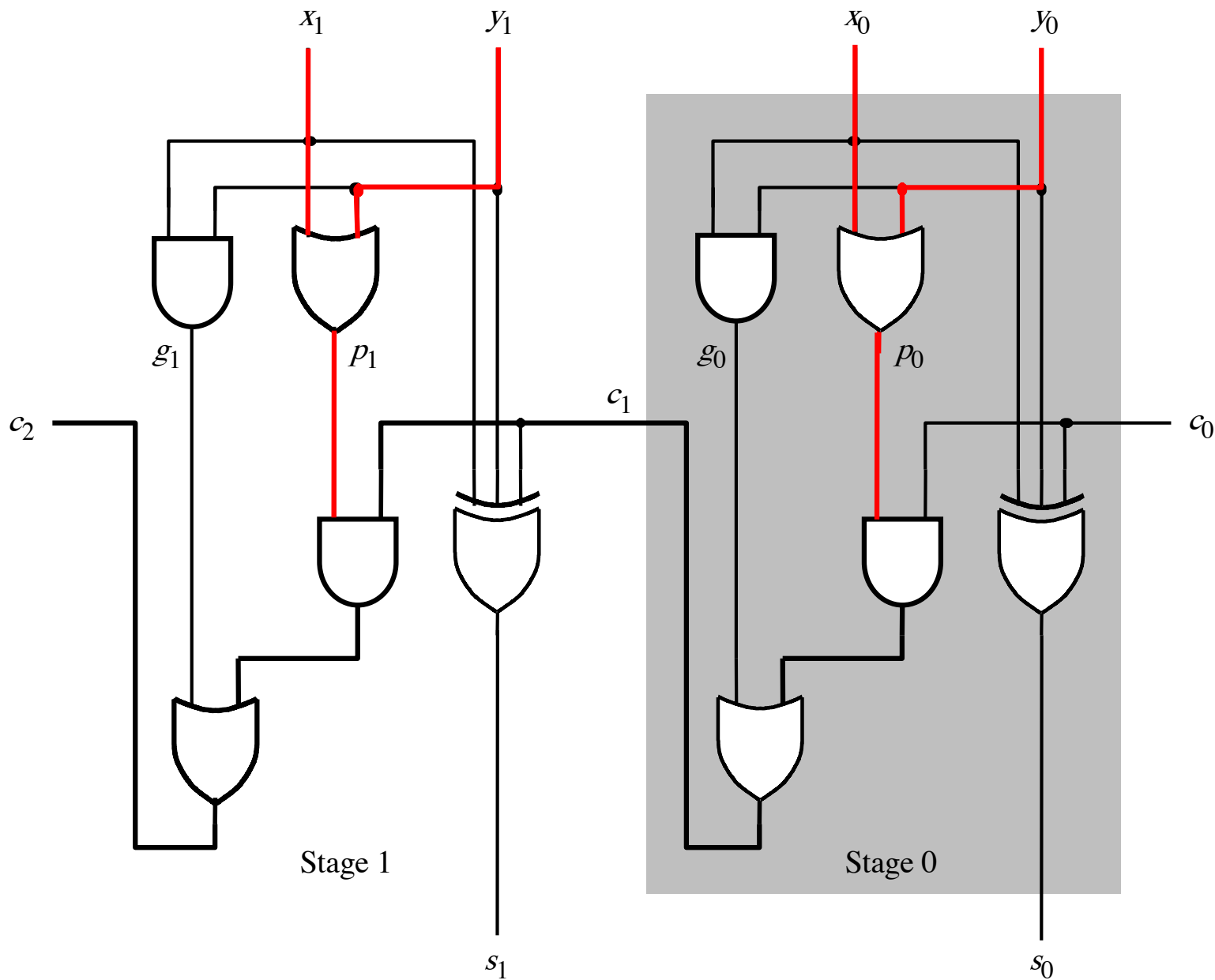
Decomposing the Carry Expression

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

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

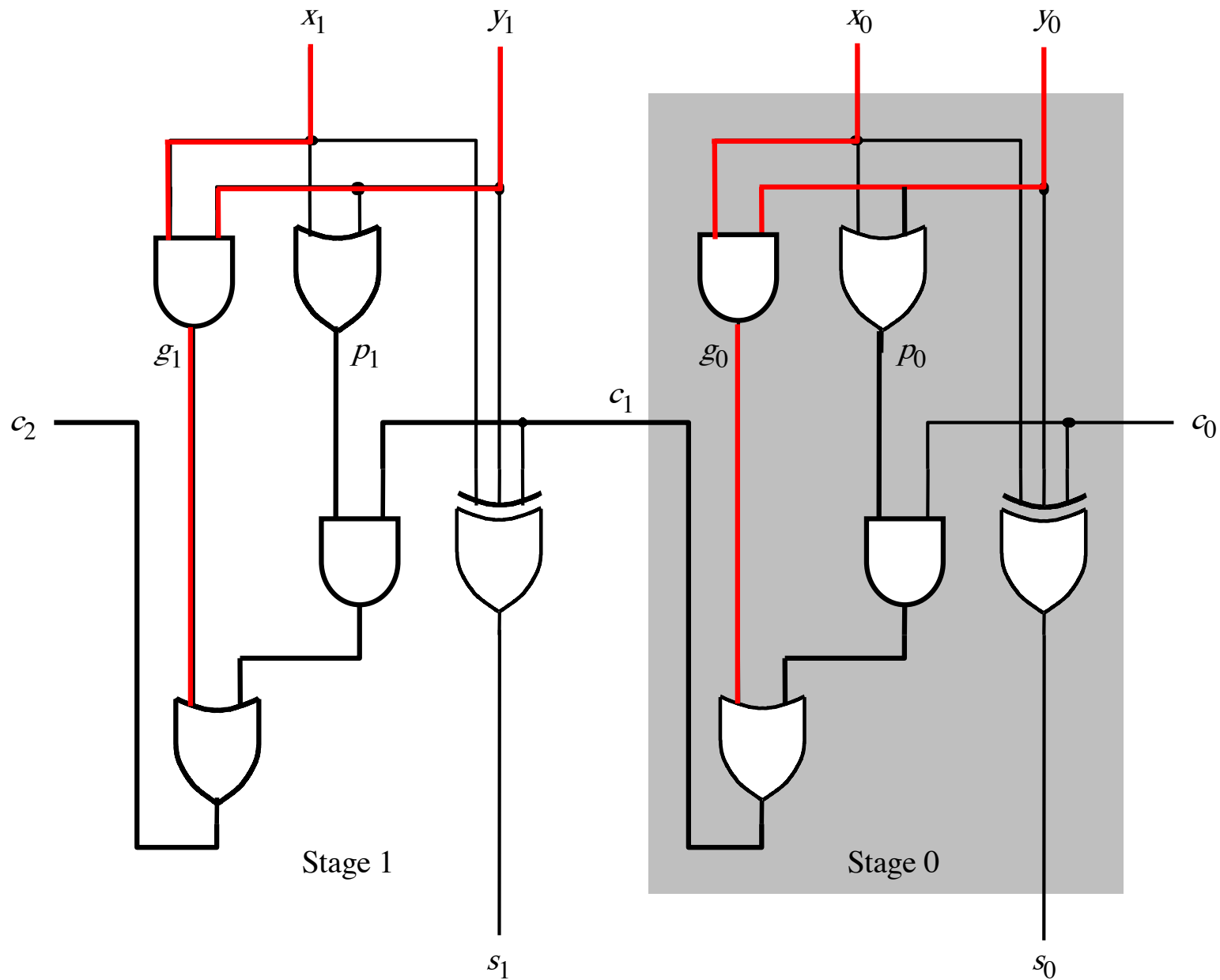
(1 gate delay) (1 gate delay)

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

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

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

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

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

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

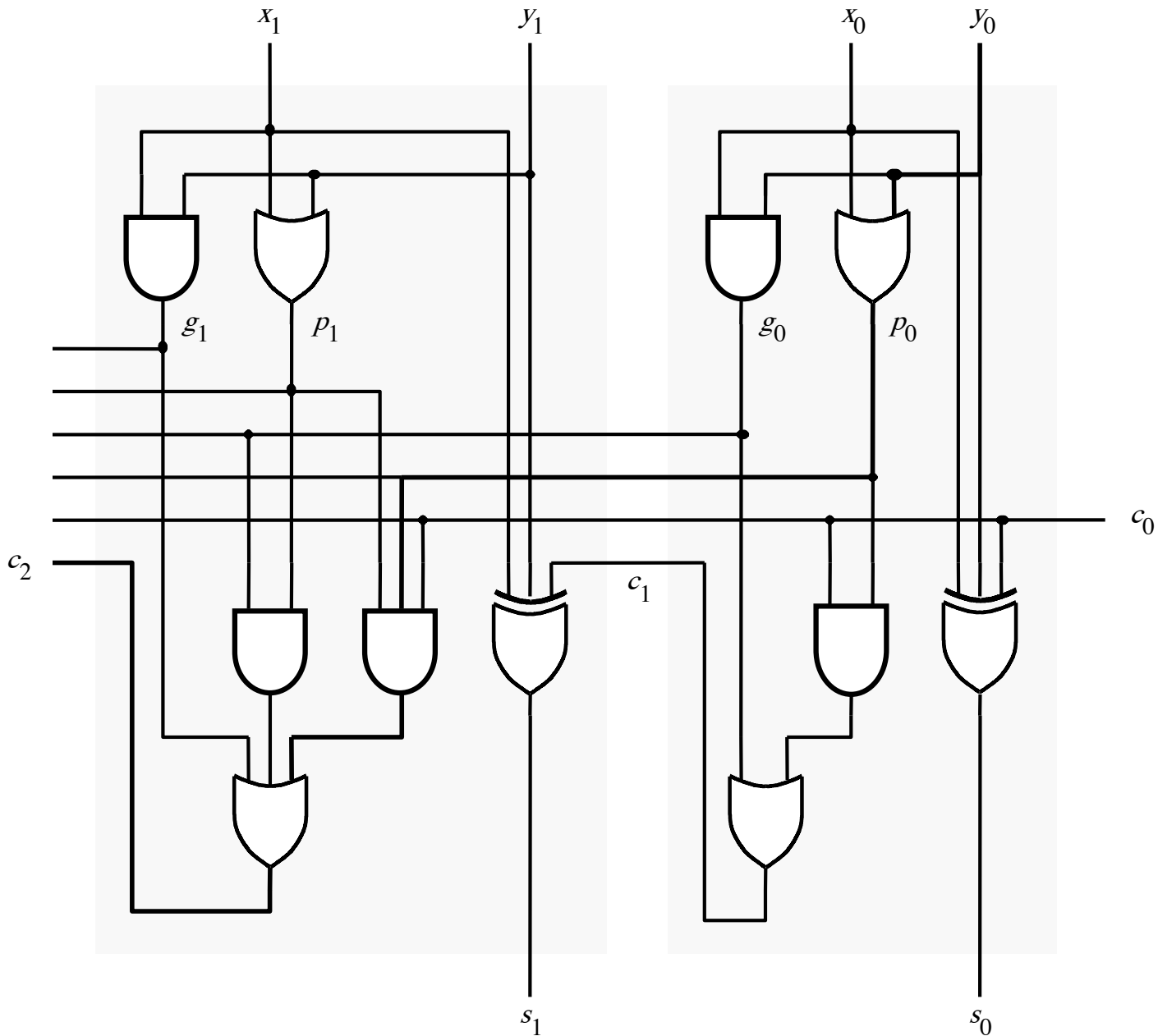
$$c_{i+1} = \underbrace{x_i y_i}_{g_i} + \underbrace{(x_i + y_i)}_{p_i} c_i$$

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

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

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

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

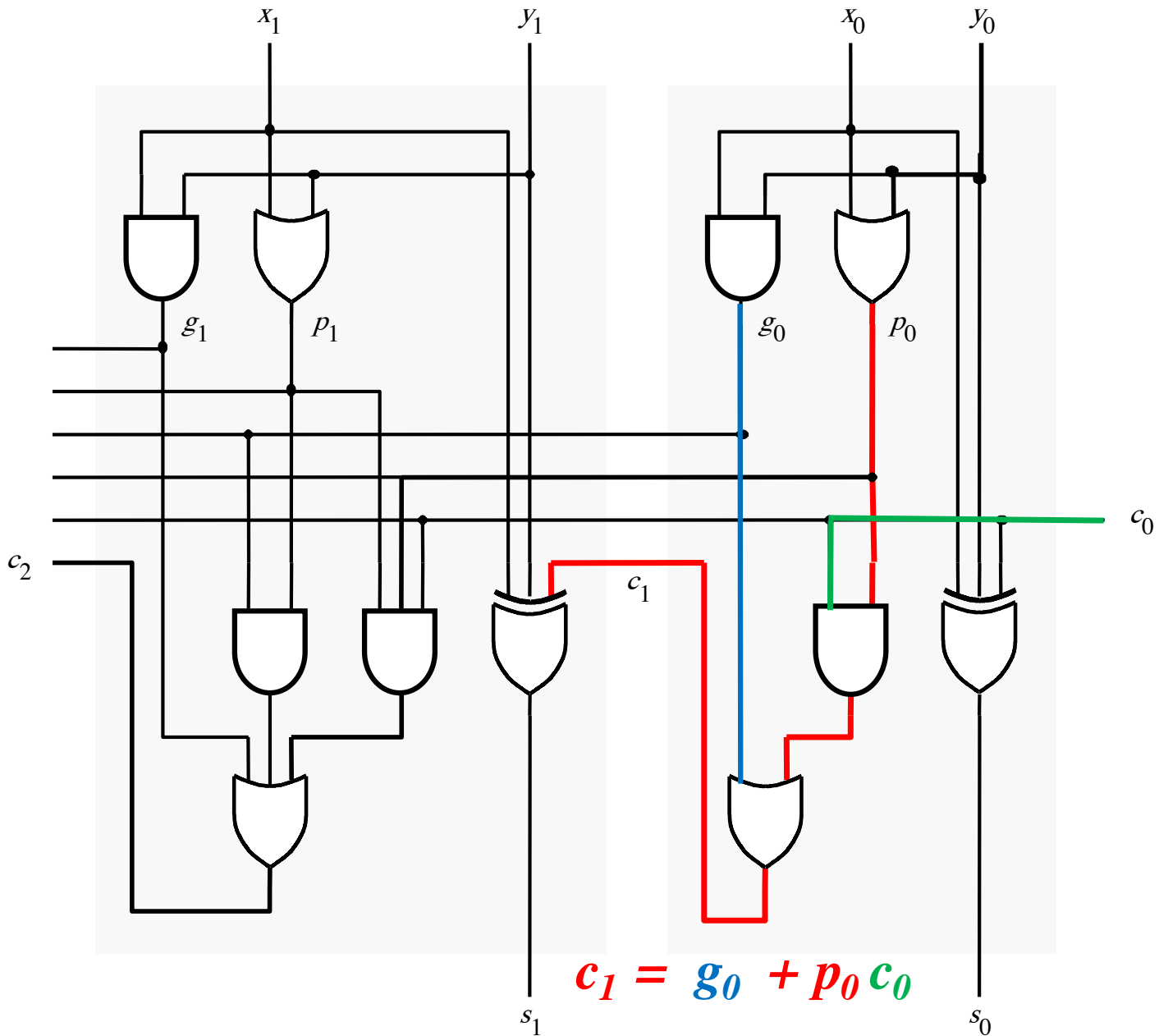


[Figure 3.15 from the textbook]

Carry for the first stage

$$c_1 = g_0 + p_0 c_0$$

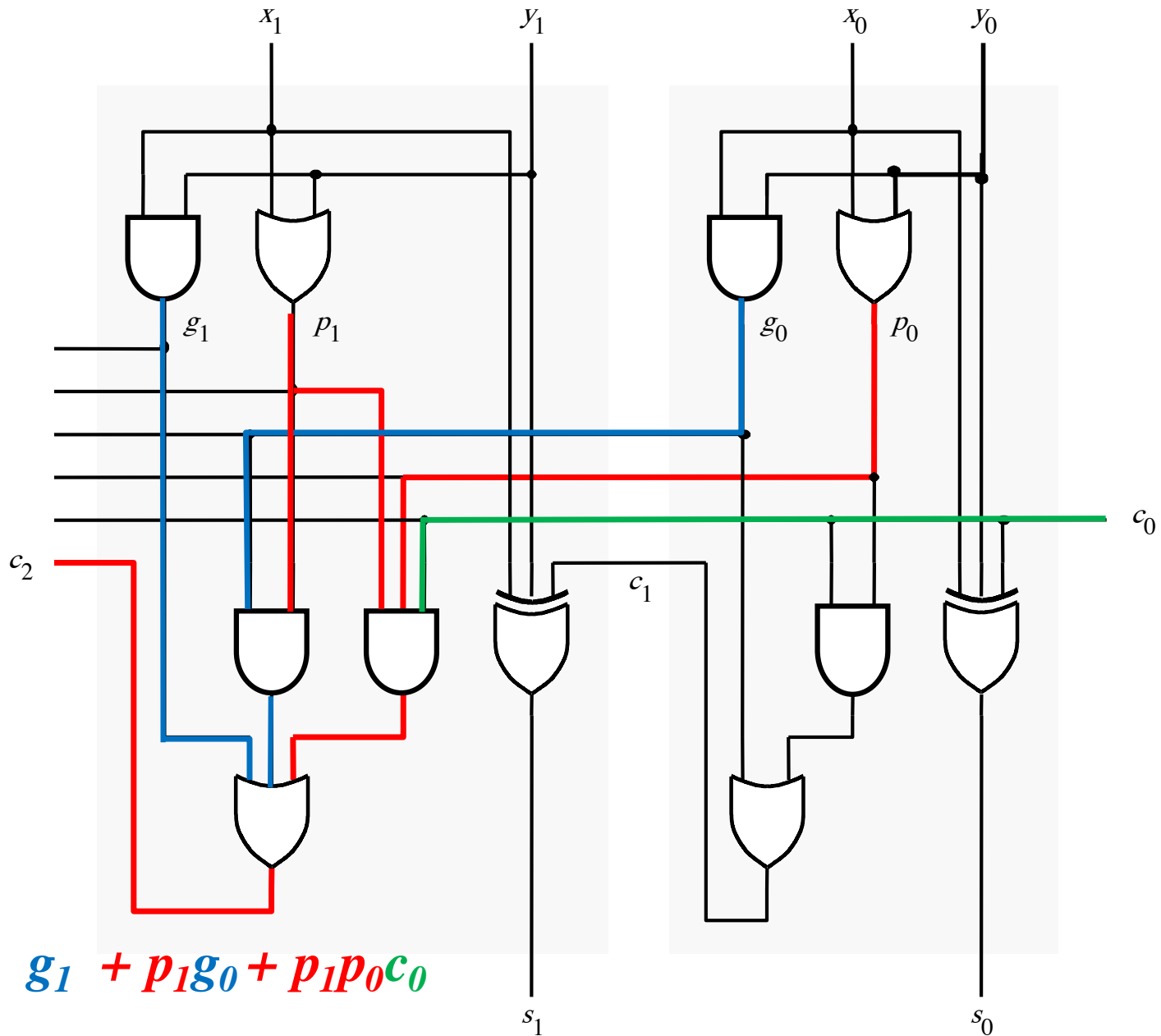
Carry for the first stage



Carry for the second stage

$$c_2 = g_1 + p_1g_0 + p_1p_0c_0$$

Carry for the second stage



$$c_2 = g_1 + p_1g_0 + p_1p_0c_0$$

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

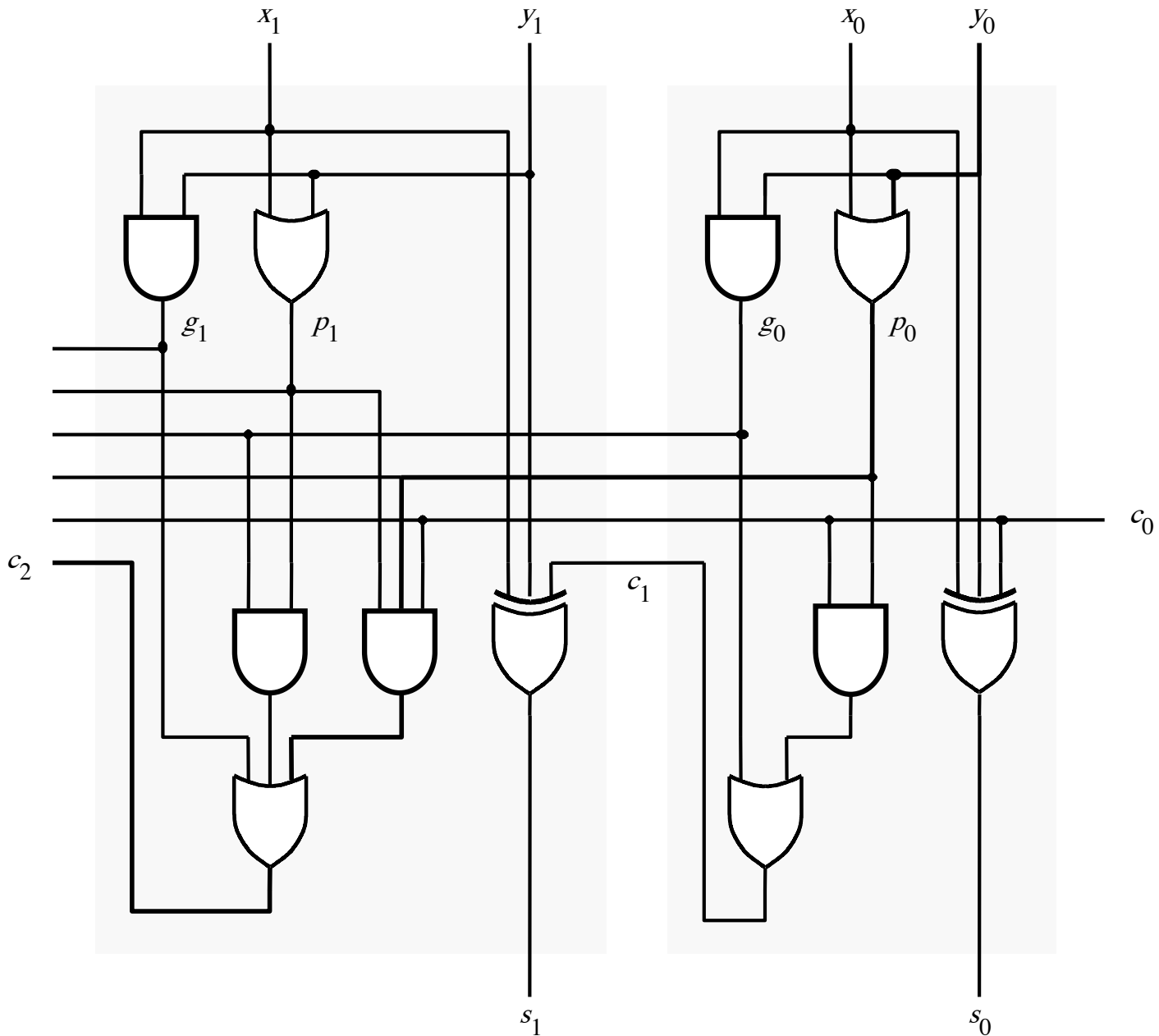
$$c_1 = g_0 + p_0 c_0$$

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

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

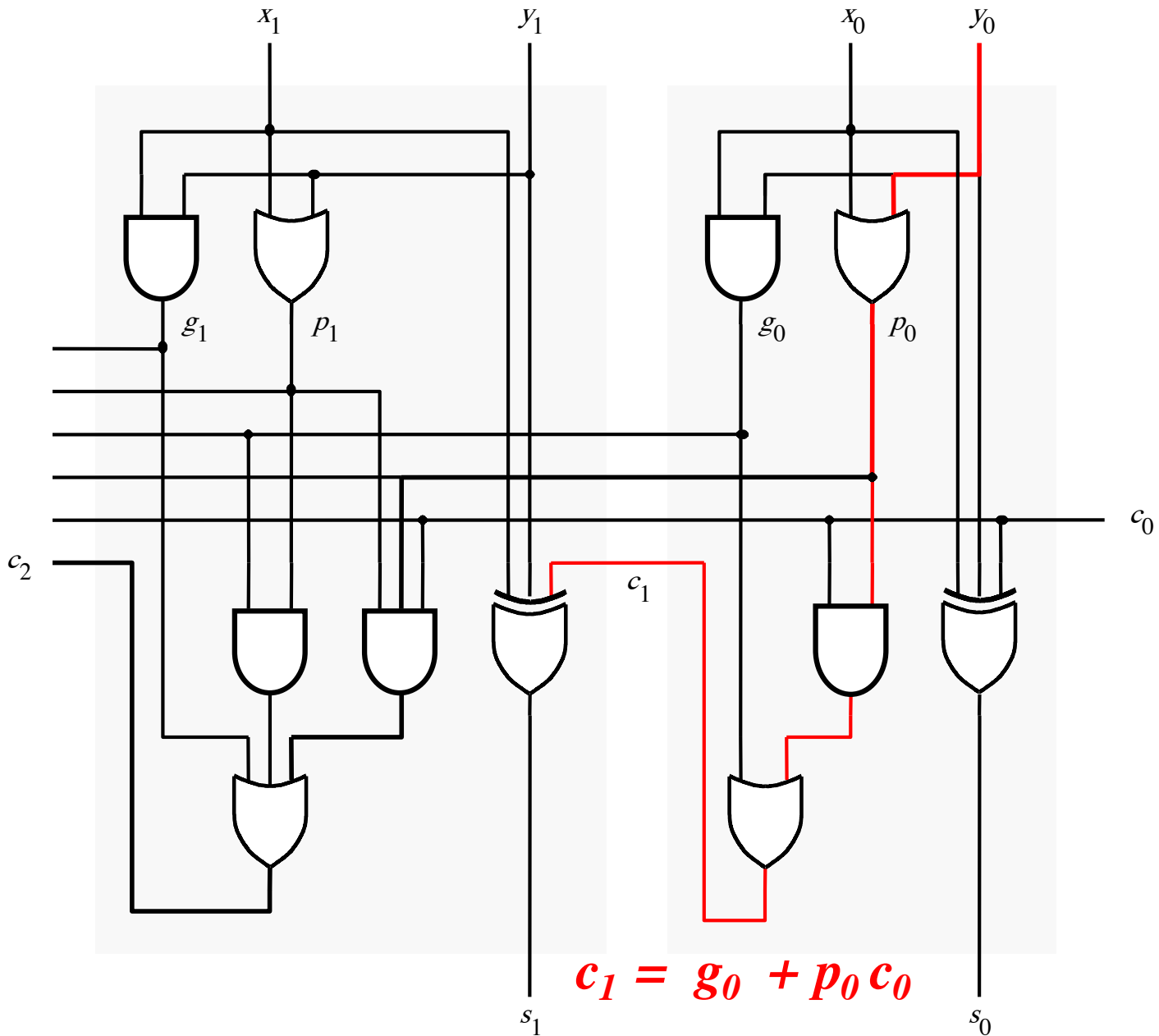
$$= g_1 + p_1 c_1$$

The first two stages of a carry-lookahead adder

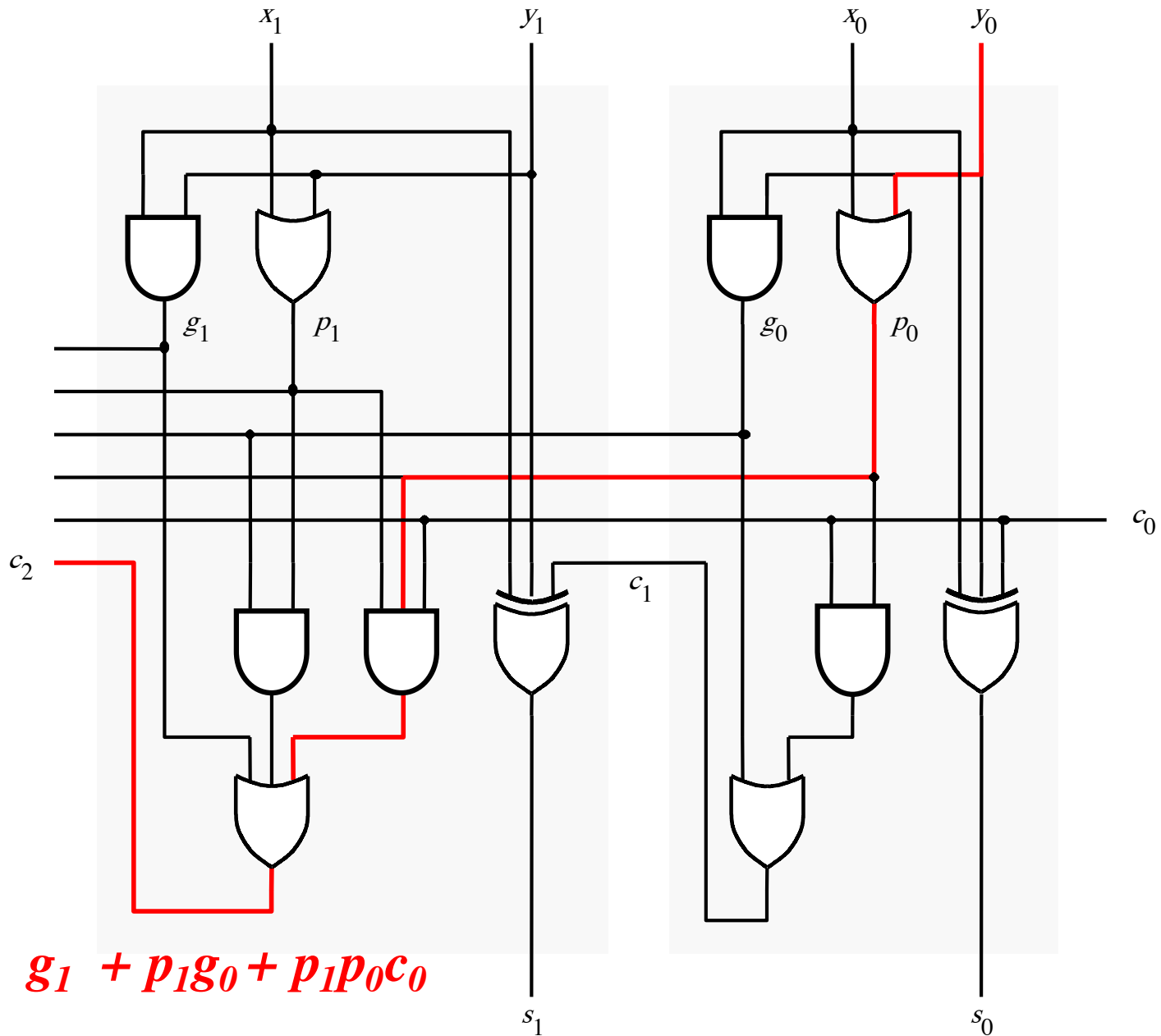


[Figure 3.15 from the textbook]

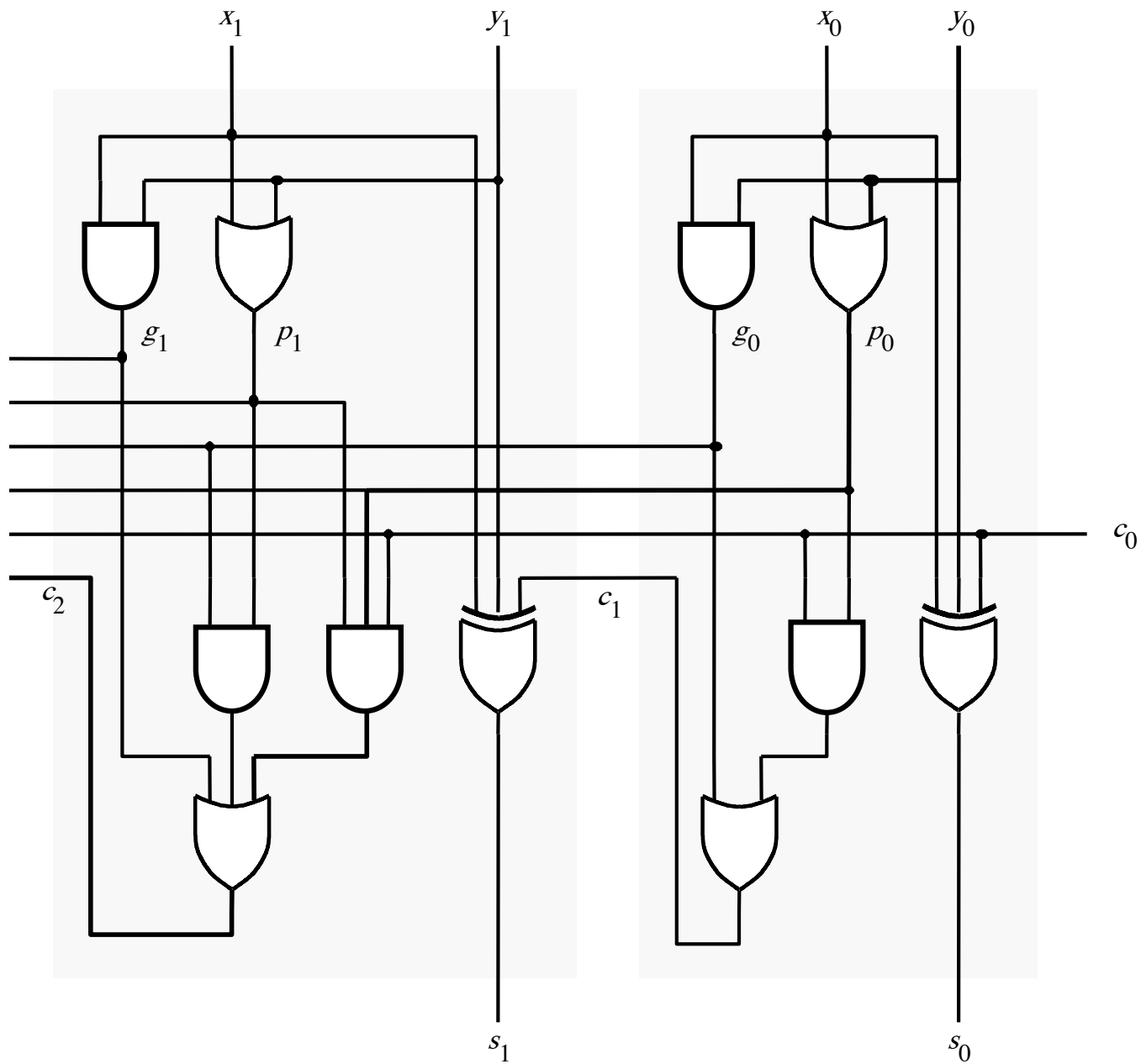
It takes 3 gate delays to generate c_1



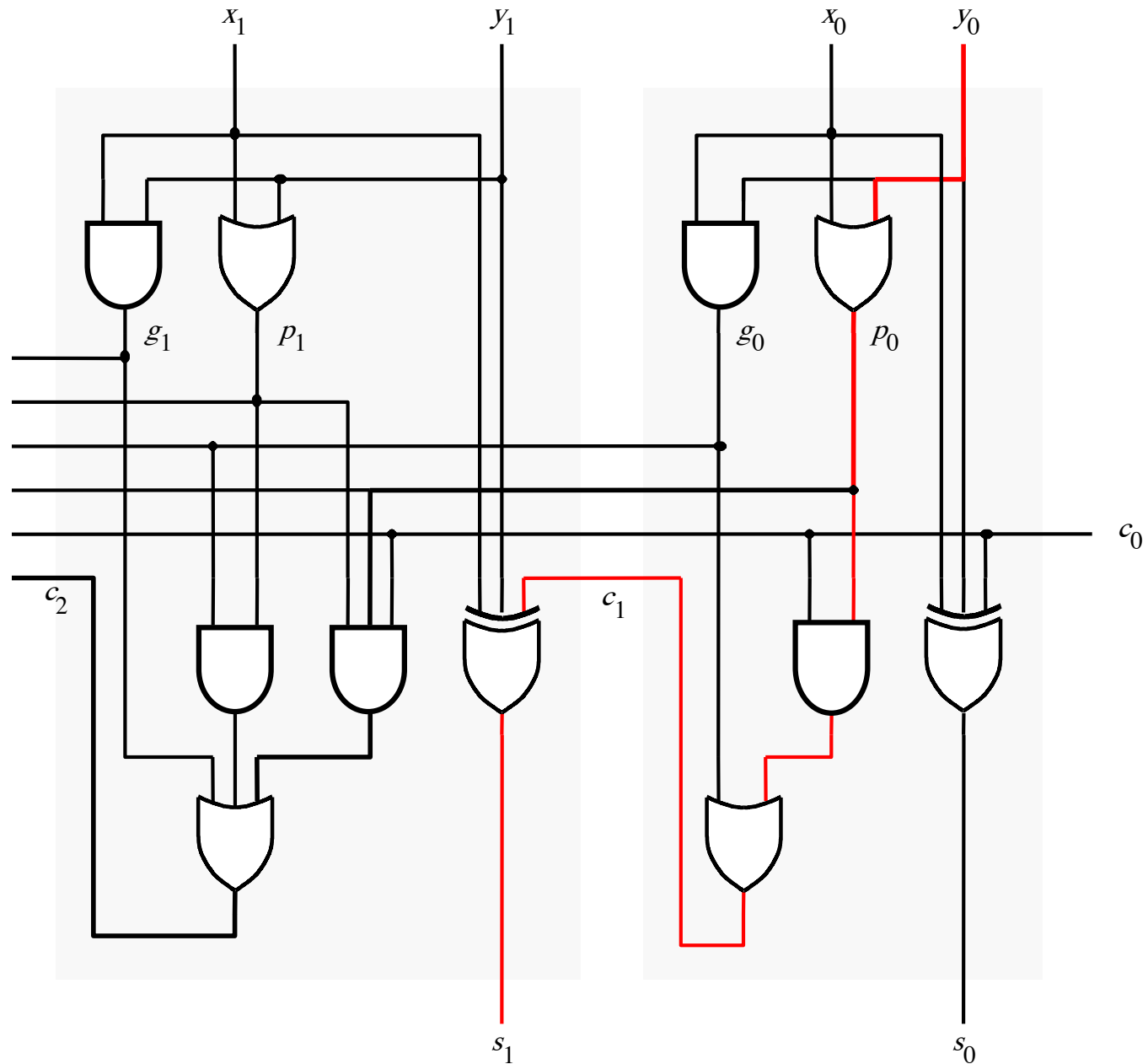
It takes 3 gate delays to generate c_2



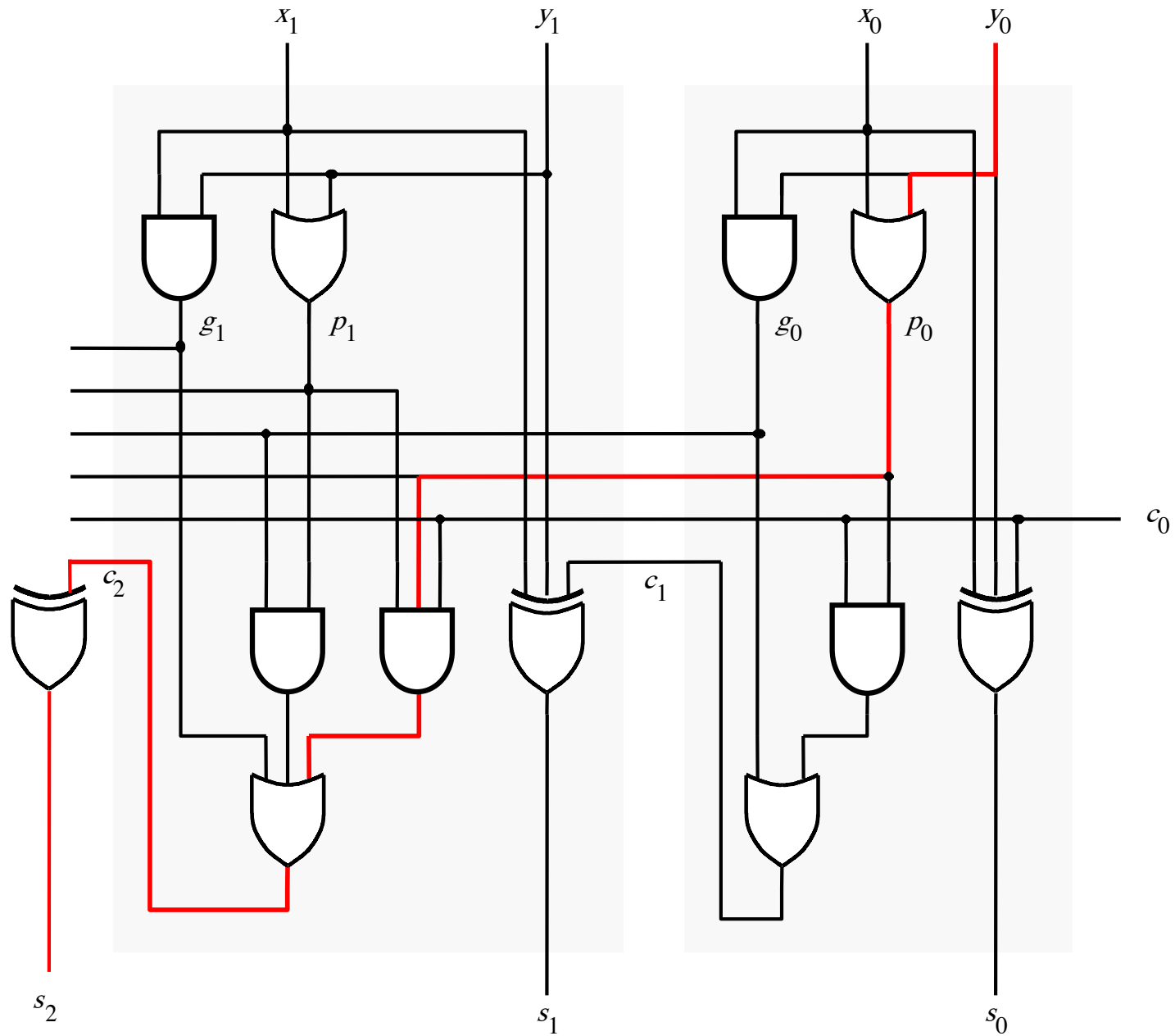
The first two stages of a carry-lookahead adder



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 1 gate delay to generate all g_i and p_i signals**
- **It takes 2 more 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

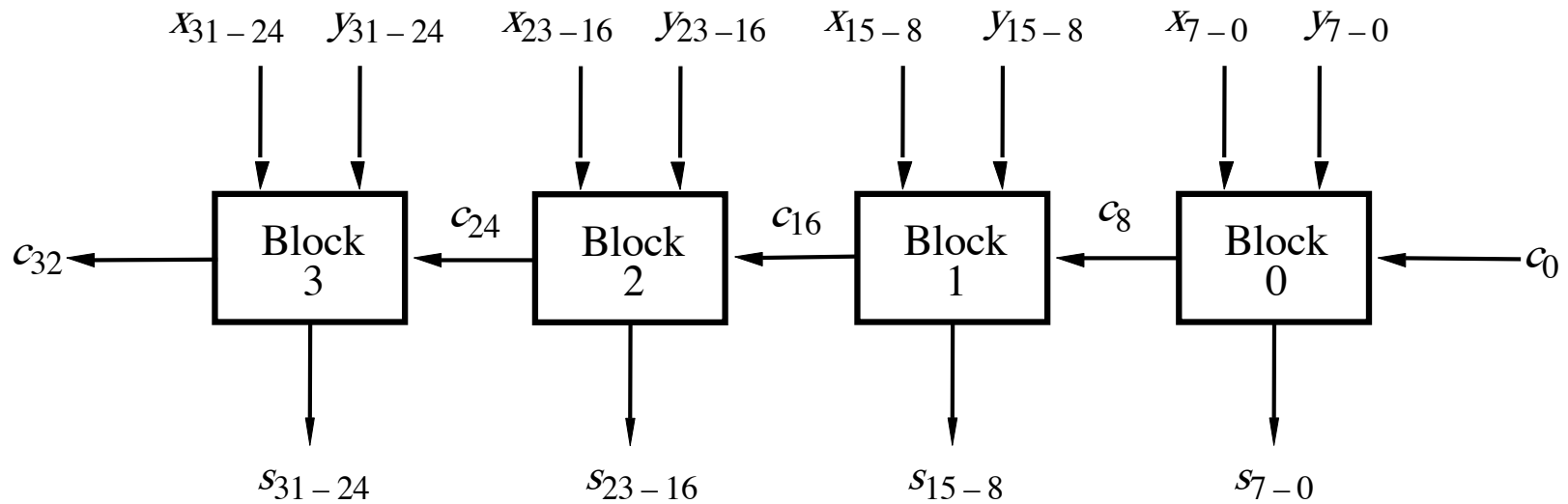
$$+ p_7 p_6 p_5 p_4 g_3 + p_7 p_6 p_5 p_4 p_3 g_2$$

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

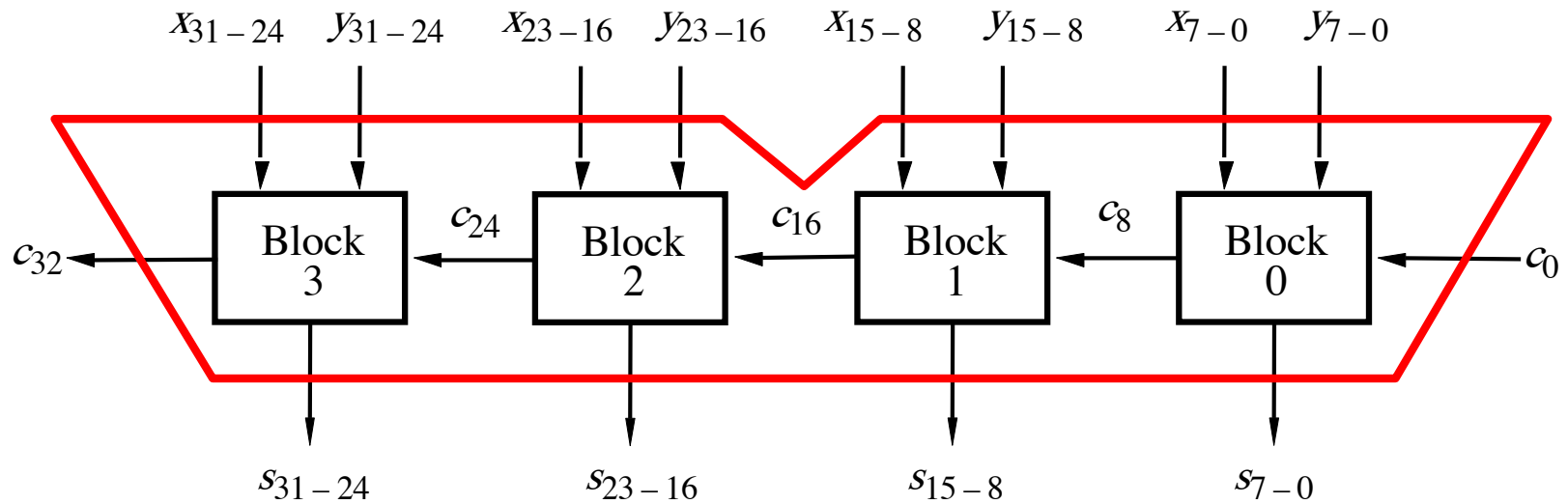
$$+ p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0 c_0$$

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

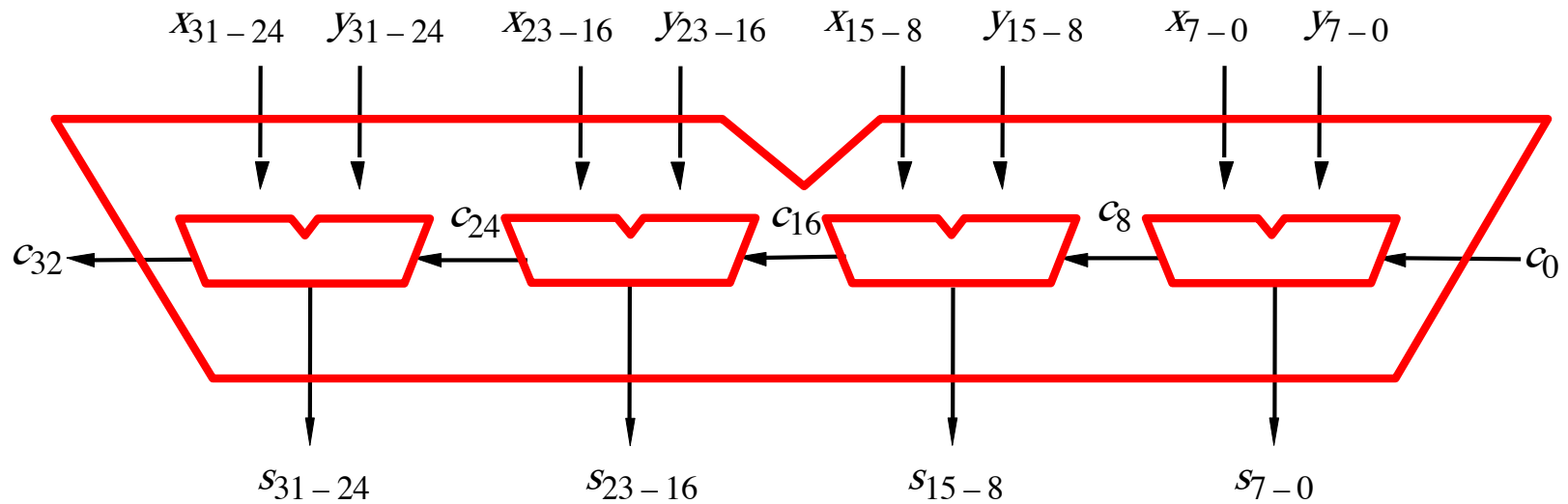
A hierarchical carry-lookahead adder with ripple-carry between blocks



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

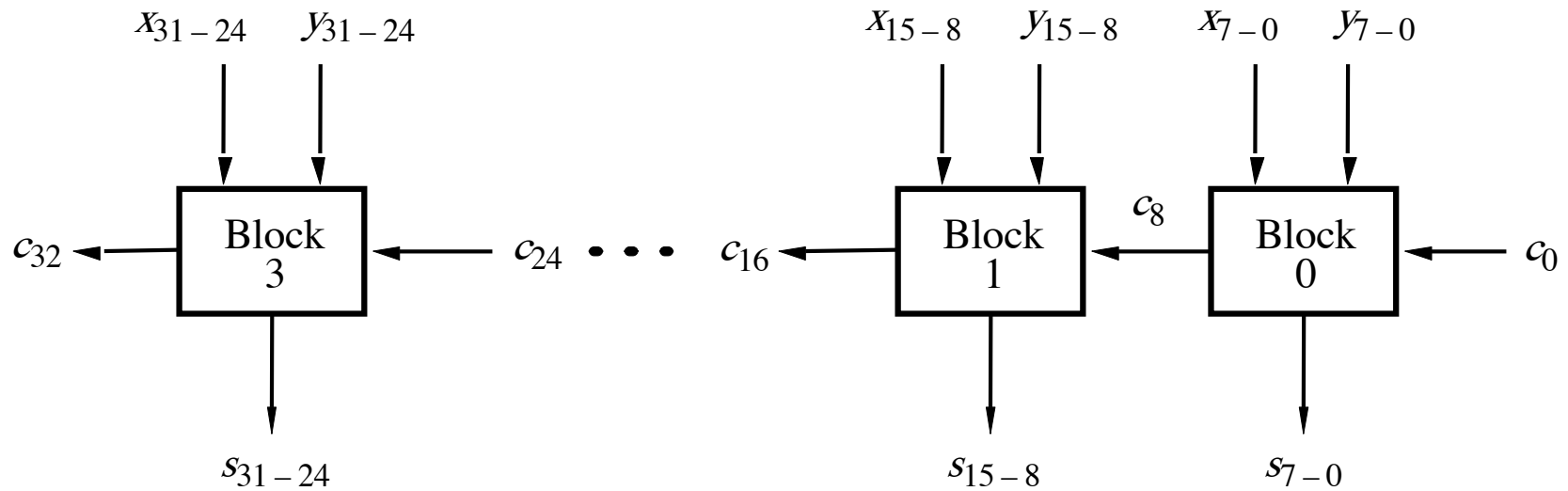


A hierarchical carry-lookahead adder with ripple-carry between blocks



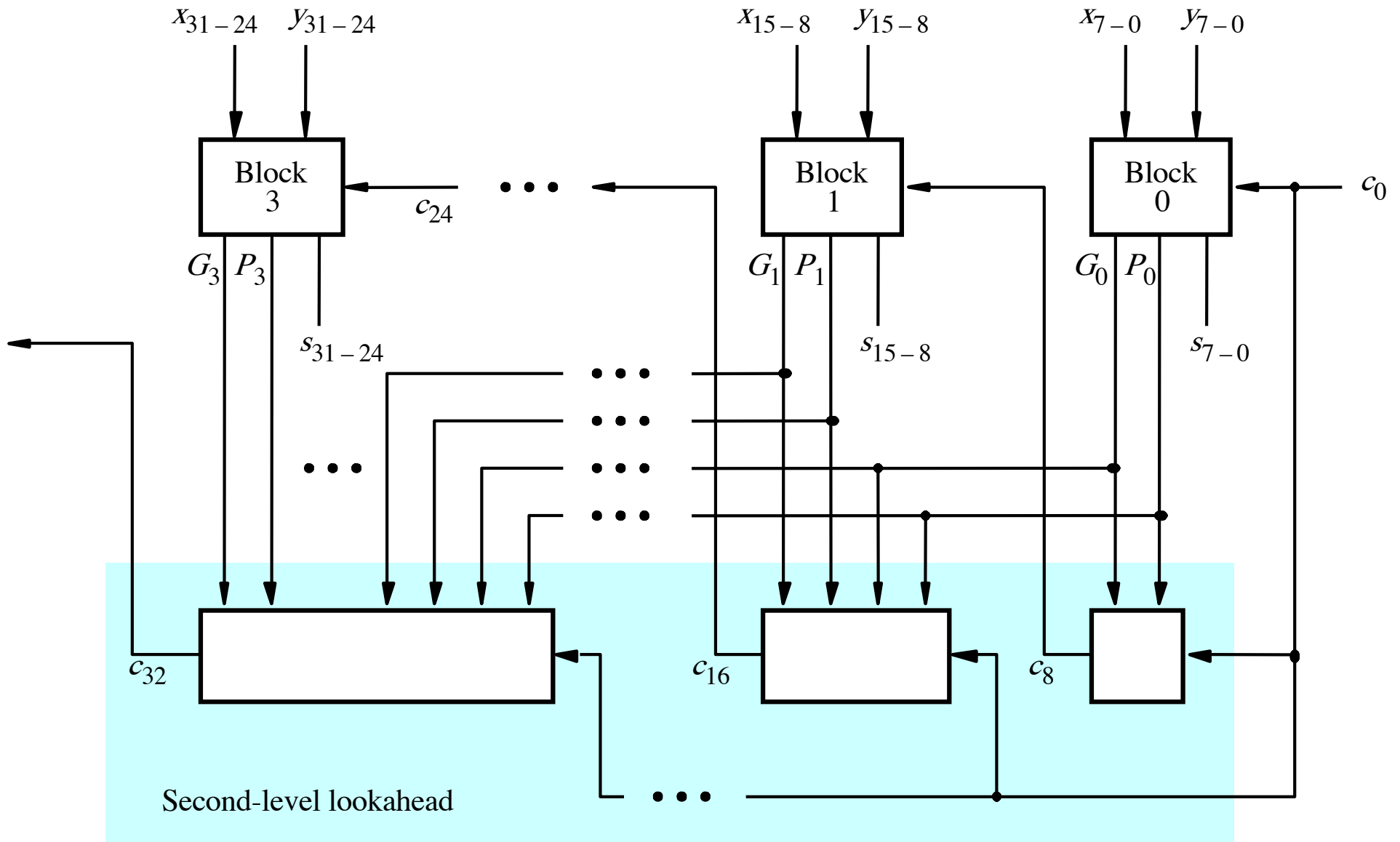
A hierarchical carry-lookahead adder

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]

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

3-gate delays

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

G_0 →

P_0 →

2-gate delays

$$c_8 = G_0 + P_0c_0$$

The Hierarchical Carry Expression

3-gate delays

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

G_0 → (points to the first four terms of the expression)
 P_0 → (points to the last term of the expression)
 2-gate delays (under the last term)

$$c_8 = \underbrace{G_0}_{\substack{\text{3-gate} \\ \text{delays}}} + \underbrace{P_0}_{\substack{\text{2-gate} \\ \text{delays}}} c_0$$

The Hierarchical Carry Expression

3-gate delays

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

G_0 points to the first four terms of the expression.
 P_0 points to the last term of the expression, which is enclosed in a red box labeled "2-gate delays".

$$c_8 = \underbrace{G_0}_{\substack{\text{3-gate} \\ \text{delays}}} + \underbrace{P_0 c_0}_{\substack{\text{3-gate} \\ \text{delays}}}$$

The Hierarchical Carry Expression

3-gate delays

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

G_0 →

P_0 →

2-gate delays

$$c_8 = G_0 + P_0c_0$$

4-gate
delays

The Hierarchical Carry Expression

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

$$\begin{aligned}c_{16} = & g_{15} + p_{15}g_{14} + p_{15}p_{14}g_{13} + p_{15}p_{14}p_{13}g_{12} \\ & + p_{15}p_{14}p_{13}p_{12}g_{11} + p_{15}p_{14}p_{13}p_{12}p_{11}g_{10} \\ & + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}g_9 + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9g_8 \\ & + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9p_8c_8\end{aligned}$$

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 same expression, just add 8 to all subscripts

$$\begin{aligned}c_{16} = & g_{15} + p_{15}g_{14} + p_{15}p_{14}g_{13} + p_{15}p_{14}p_{13}g_{12} \\ & + p_{15}p_{14}p_{13}p_{12}g_{11} + p_{15}p_{14}p_{13}p_{12}p_{11}g_{10} \\ & + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}g_9 + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9g_8 \\ & + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9p_8c_8\end{aligned}$$

The Hierarchical Carry Expression

3-gate delays

$$C_8 = \begin{aligned} &g_7 + p_7g_6 + p_7p_6g_5 + p_7p_6p_5g_4 \\ &+ p_7p_6p_5p_4g_3 + p_7p_6p_5p_4p_3g_2 \\ &+ p_7p_6p_5p_4p_3p_2g_1 + p_7p_6p_5p_4p_3p_2p_1g_0 \\ &+ p_7p_6p_5p_4p_3p_2p_1p_0c_0 \end{aligned}$$

G_0 points to the first three lines of the expression.

P_0 points to the last term of the expression, which is enclosed in a red box.

2-gate delays

$$C_{16} = \begin{aligned} &g_{15} + p_{15}g_{14} + p_{15}p_{14}g_{13} + p_{15}p_{14}p_{13}g_{12} \\ &+ p_{15}p_{14}p_{13}p_{12}g_{11} + p_{15}p_{14}p_{13}p_{12}p_{11}g_{10} \\ &+ p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}g_9 + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9g_8 \\ &+ p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9p_8c_8 \end{aligned}$$

The Hierarchical Carry Expression

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

3-gate delays

$$\begin{aligned}
 c_{16} = & g_{15} + p_{15}g_{14} + p_{15}p_{14}g_{13} + p_{15}p_{14}p_{13}g_{12} \\
 & + p_{15}p_{14}p_{13}p_{12}g_{11} + p_{15}p_{14}p_{13}p_{12}p_{11}g_{10} \\
 & + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}g_9 + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9g_8 \\
 & + p_{15}p_{14}p_{13}p_{12}p_{11}p_{10}p_9p_8c_8
 \end{aligned}$$

G_1 →

P_1 →

2-gate delays

The Hierarchical Carry Expression

$$c_8 = G_0 + P_0 c_0$$

The Hierarchical Carry Expression

$$c_8 = \textcircled{G_0} + P_0 c_0$$

3-gate delays

The Hierarchical Carry Expression

$$c_8 = G_0 + P_0 c_0$$

4-gate delays

The Hierarchical Carry Expression

$$c_8 = G_0 + P_0 c_0$$

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

The Hierarchical Carry Expression

$$c_8 = \textcircled{G_0} + P_0 c_0$$

3-gate delays

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

3-gate delays

The Hierarchical Carry Expression

$$c_8 = G_0 + P_0 c_0$$

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

3-gate delays

The Hierarchical Carry Expression

$$c_8 = G_0 + P_0 c_0$$

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

4-gate delays

The Hierarchical Carry Expression

$$c_8 = G_0 + P_0 c_0$$

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

5-gate delays

The Hierarchical Carry Expression

$$c_8 = G_0 + P_0 c_0$$

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

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

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

The Hierarchical Carry Expression

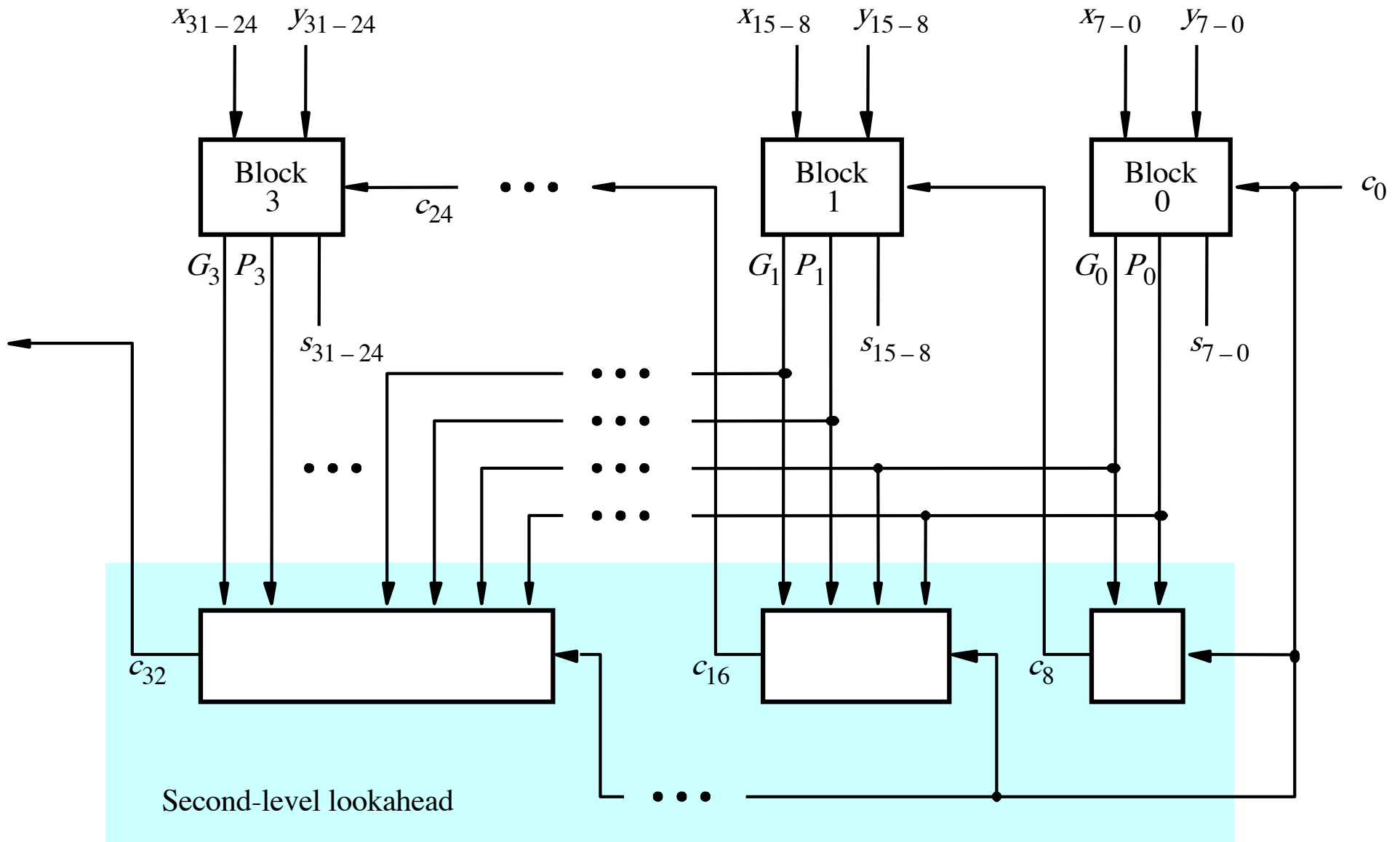
$$c_8 = G_0 + P_0 c_0 \quad \text{4-gate delays}$$

$$\begin{aligned} c_{16} &= G_1 + P_1 c_8 && \text{5-gate delays} \\ &= G_1 + P_1 G_0 + P_1 P_0 c_0 \end{aligned}$$

$$c_{24} = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 c_0 \quad \text{5-gate delays}$$

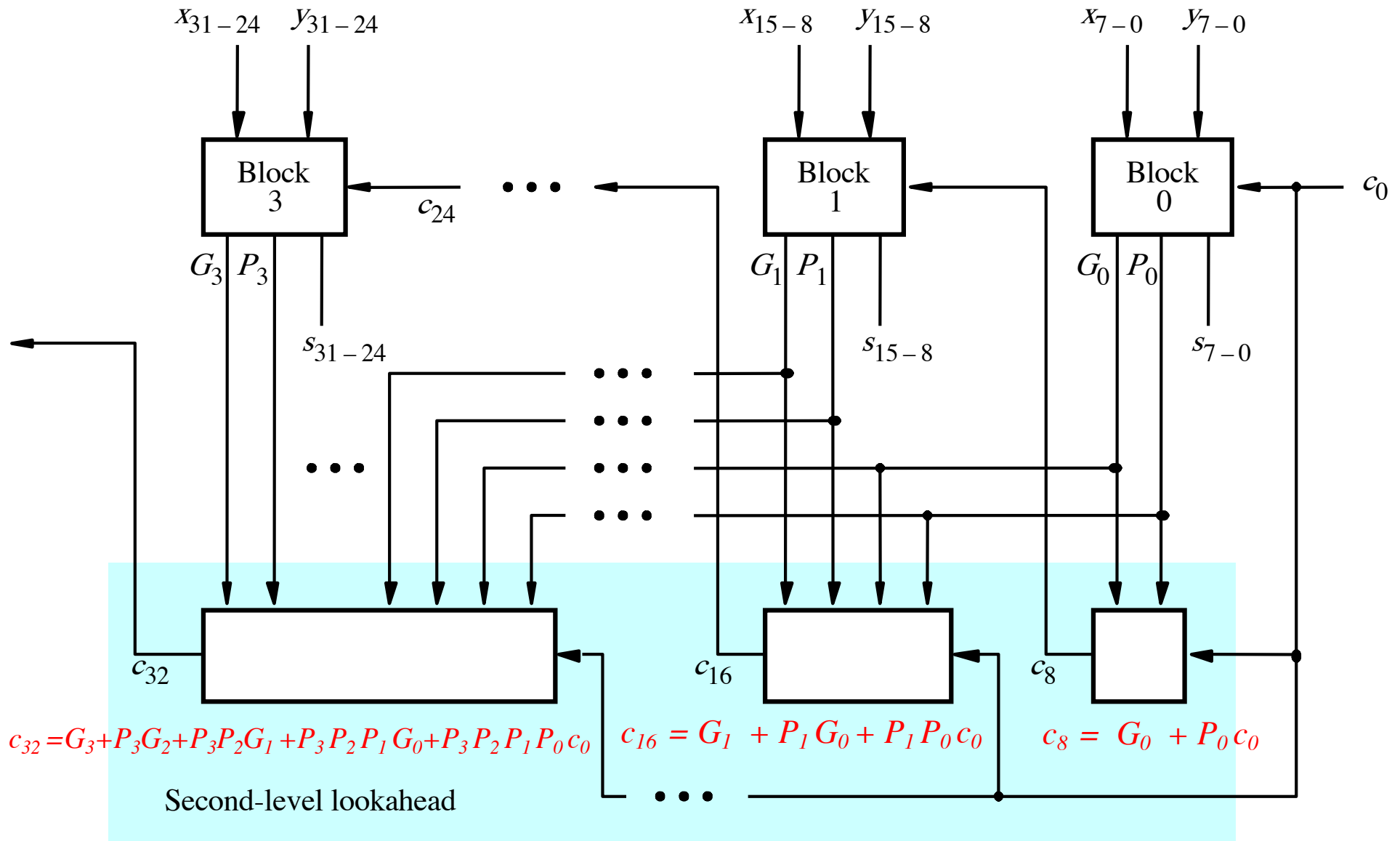
$$c_{32} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 c_0 \quad \text{5-gate delays}$$

A hierarchical carry-lookahead adder



[Figure 3.17 from the textbook]

A hierarchical carry-lookahead adder



[Figure 3.17 from the textbook]

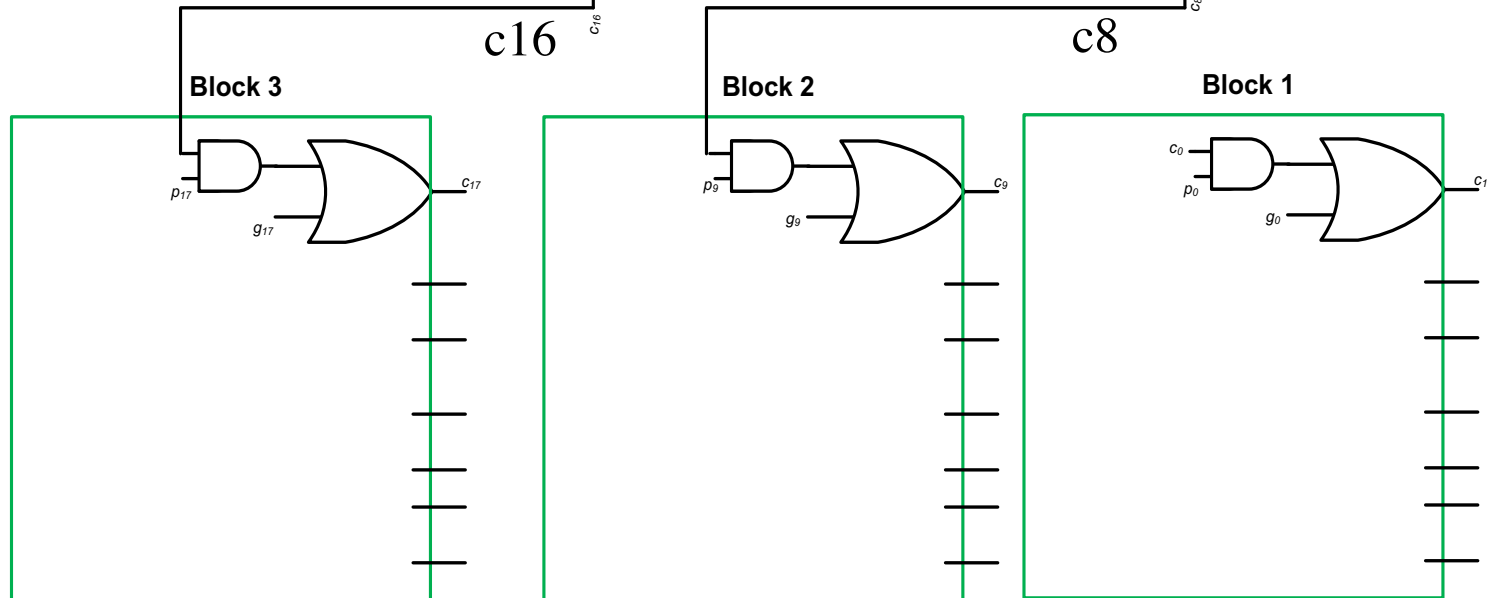
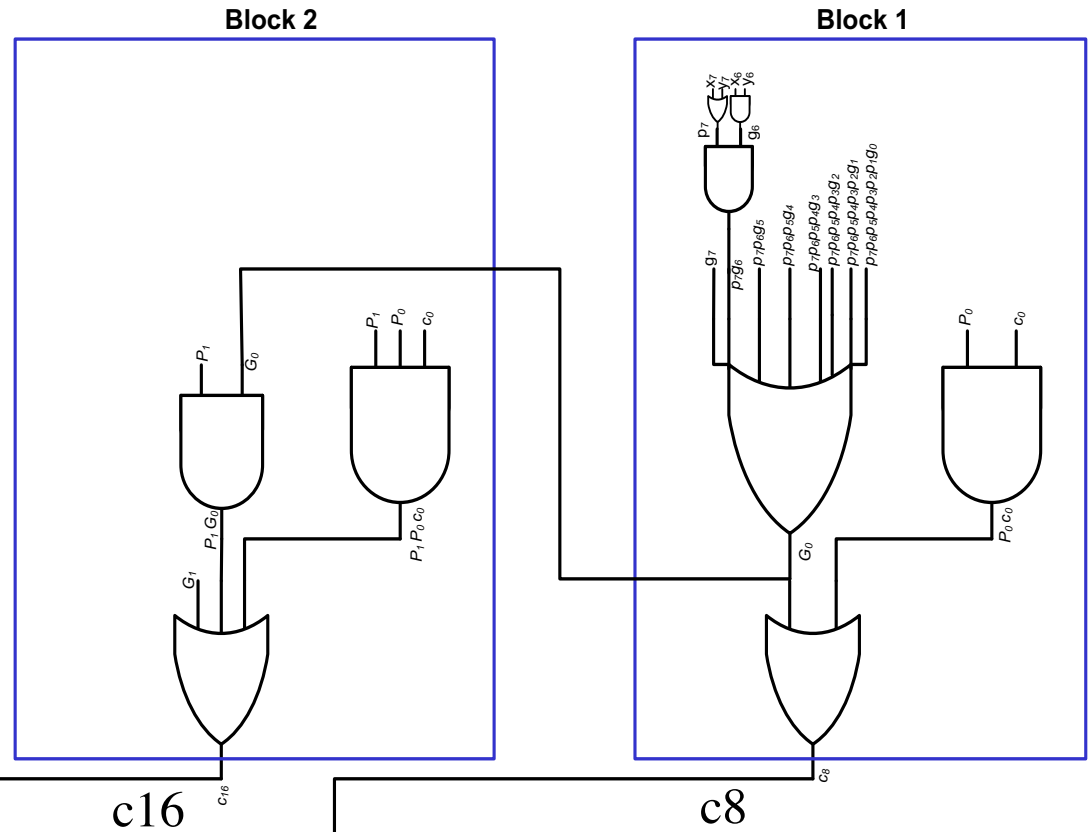
Total Gate Delay Through a Hierarchical Carry-Lookahead Adder

- The total delay is 8 gates:
 - 3 to generate all G_i and P_i signals
 - +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)

Hierarchical CLA Adder Carry Logic

SECOND
LEVEL
HIERARCHY

- C8 - 4 gate delays
- C16 - 5 gate delays
- C24 - 5 Gate delays
- C32 - 5 Gate delays

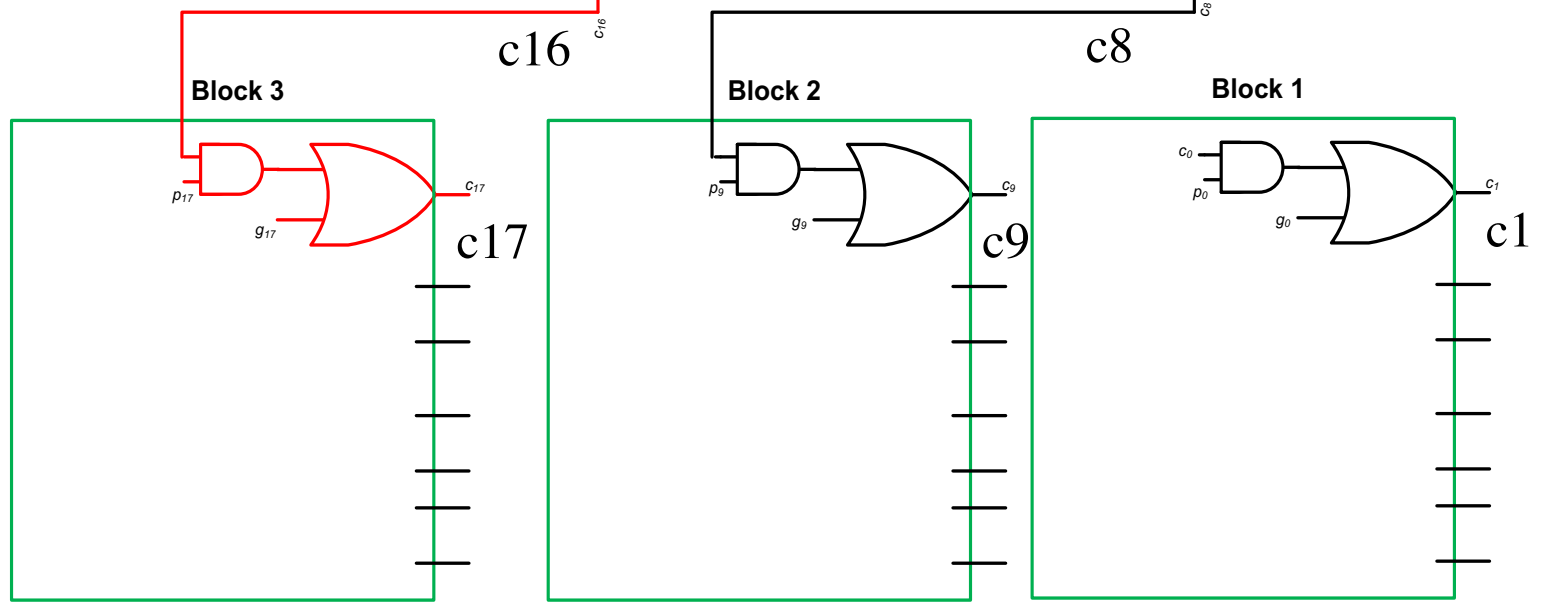
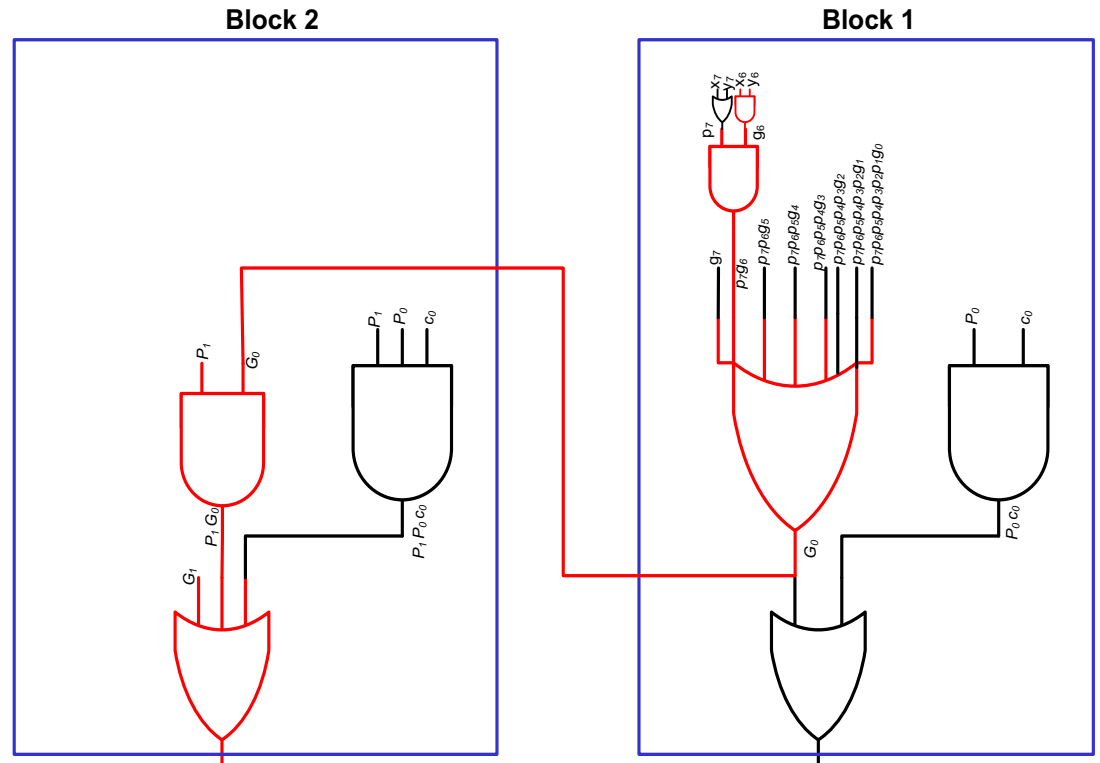


FIRST LEVEL HIERARCHY

Hierarchical CLA Critical Path

- C1** - 2 gate delays
- C9** - 6 gate delays
- C17** - 7 gate delays
- C25** - 7 Gate delays

SECOND
LEVEL
HIERARCHY

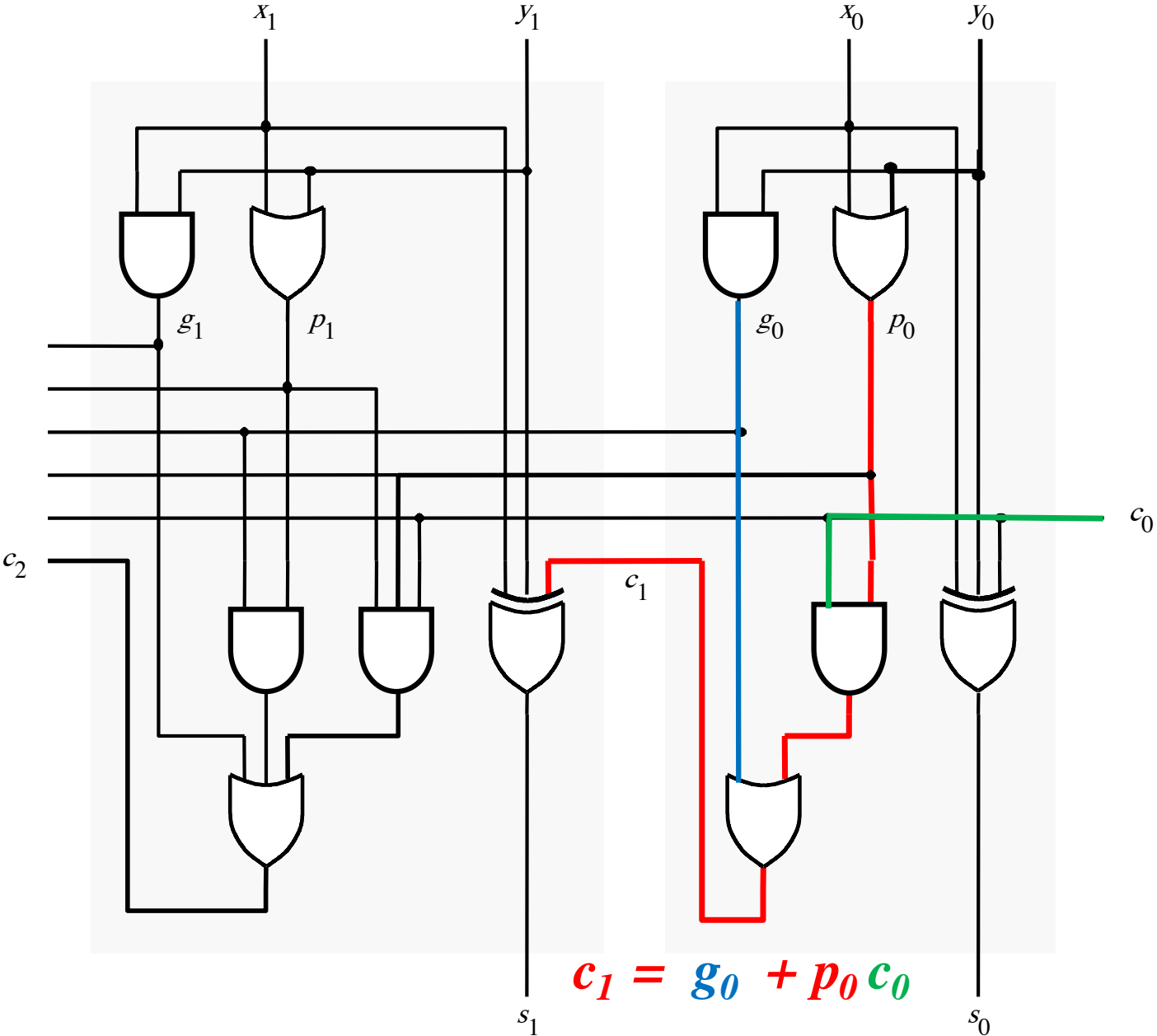


FIRST LEVEL HIERARCHY

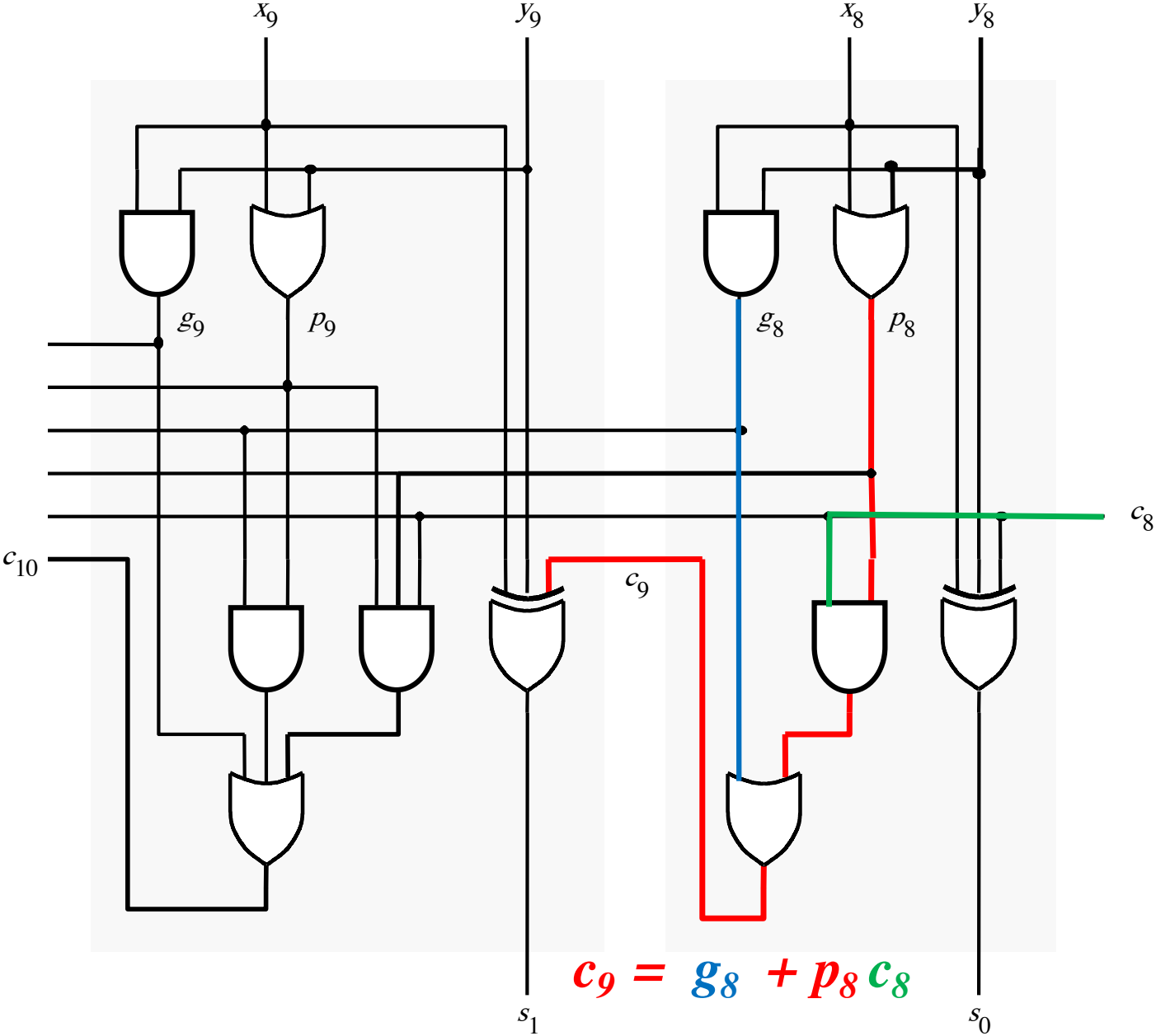
Total Gate Delay Through a Hierarchical Carry-Lookahead Adder

- The total delay is 8 gates:
 - 3 to generate all G_i and P_i signals
 - +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)

2 more gate delays for the internal carries within a block



2 more gate delays for the internal carries within a block



Total Gate Delay Through a Hierarchical Carry-Lookahead Adder

- The total delay is **8 gates**:
 - 3 to generate all G_i and P_i signals
 - +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