



CprE 281: Digital Logic

Instructor: Alexander Stoytchev

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

Serial Adder

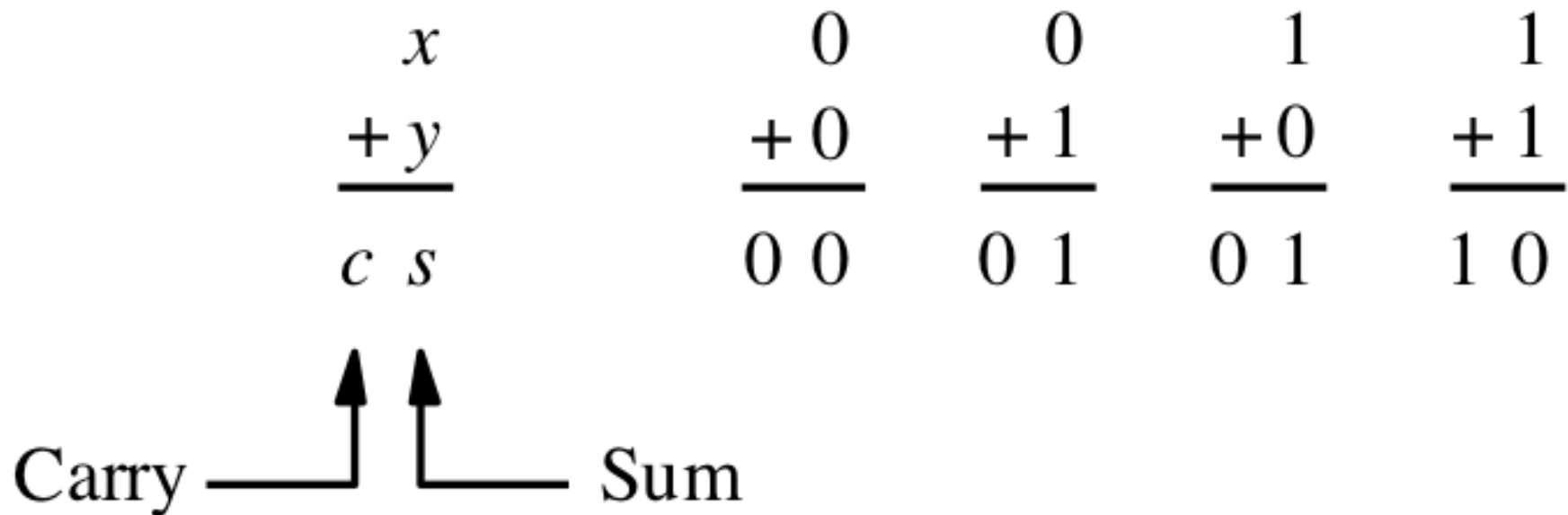
CprE 281: Digital Logic
Iowa State University, Ames, IA
Copyright © Alexander Stoytchev

Administrative Stuff

- **Homework 10 is due on Monday @ 4pm**

Quick Review

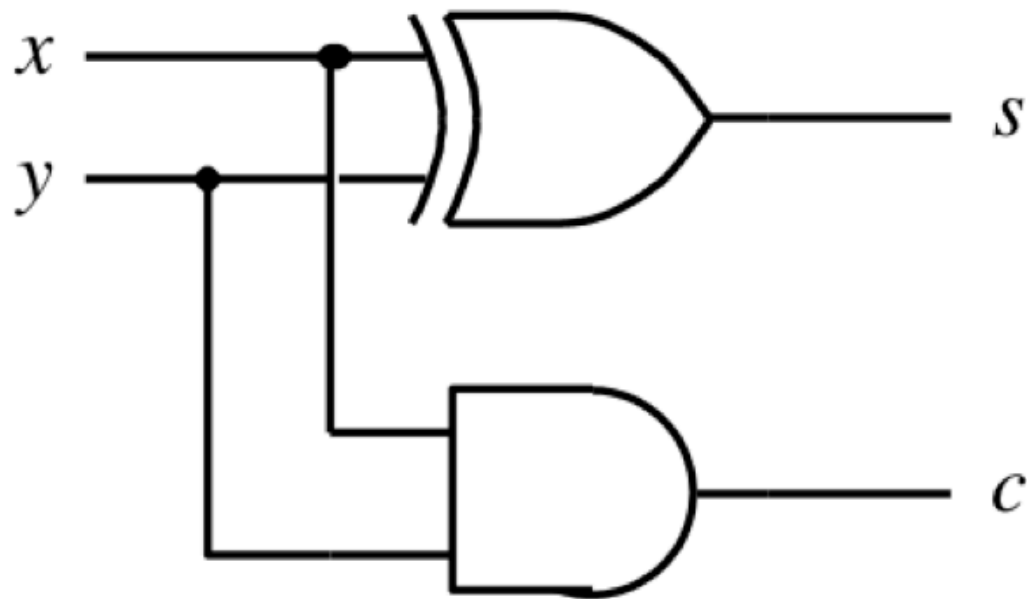
Adding two bits (there are four possible cases)



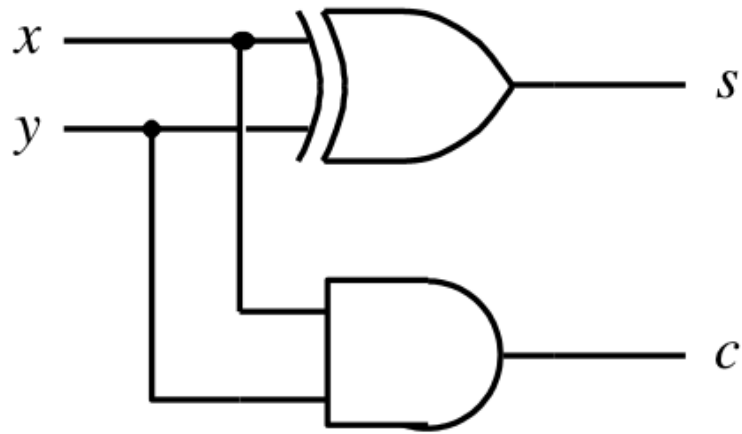
Adding two bits (the truth table)

x	y	Carry c	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

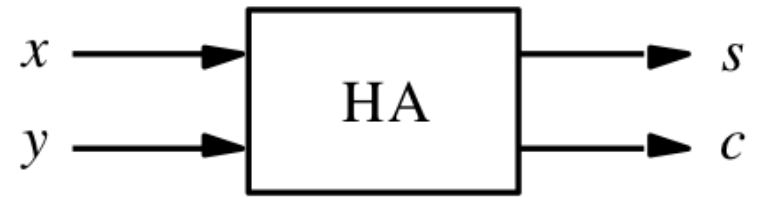
Adding two bits (the logic circuit)



The Half-Adder



(c) Circuit

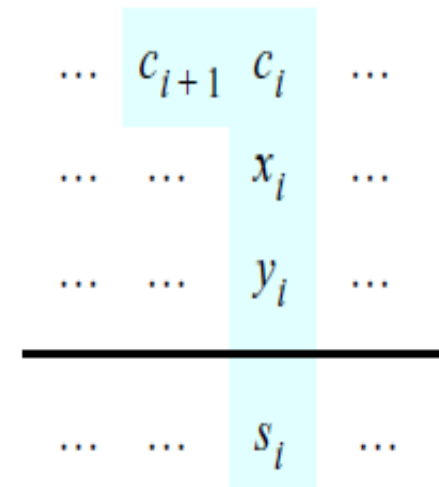


(d) Graphical symbol

Addition of multibit numbers

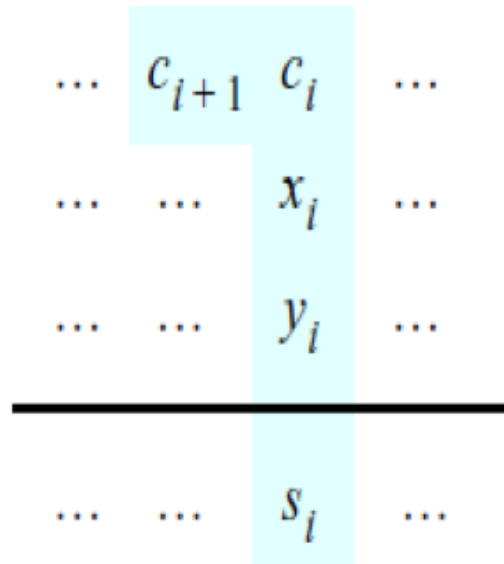
Generated carries \longrightarrow 1 1 1 0

$$\begin{array}{r}
 X = x_4x_3x_2x_1x_0 \quad 01111 \quad (15)_{10} \\
 + Y = y_4y_3y_2y_1y_0 \quad +01010 \quad + (10)_{10} \\
 \hline
 S = s_4s_3s_2s_1s_0 \quad 11001 \quad (25)_{10}
 \end{array}$$



Bit position i

Problem Statement and Truth Table



c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

[Figure 3.2b from the textbook]

[Figure 3.3a from the textbook]

Let's fill-in the two K-maps

c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

		$x_i y_i$			
		00	01	11	10
c_i	0				
	1				

$s_i =$

		$x_i y_i$			
		00	01	11	10
c_i	0				
	1				

$c_{i+1} =$

Let's fill-in the two K-maps

c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

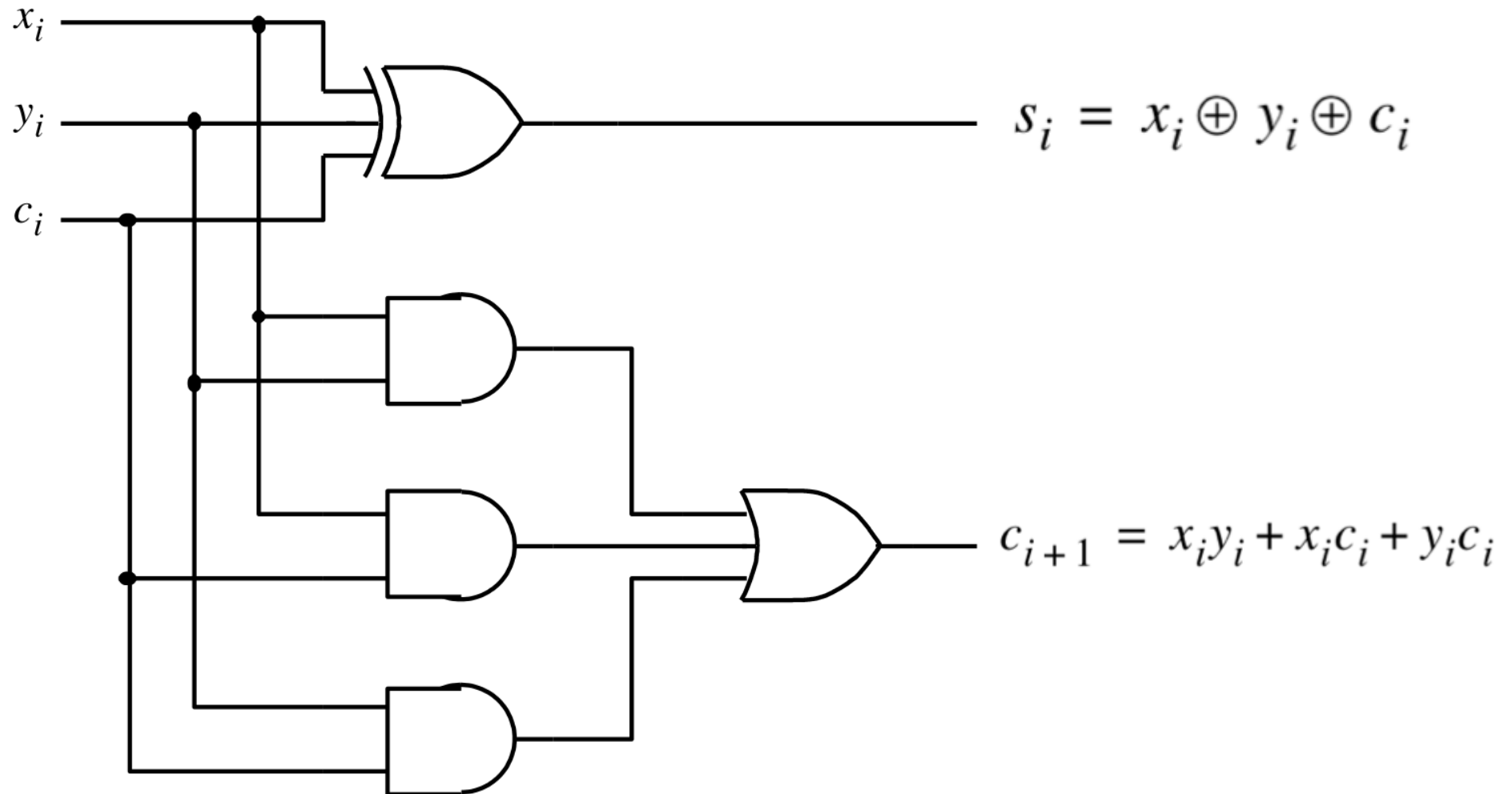
$c_i \backslash x_i y_i$	00	01	11	10
0		1		1
1	1		1	

$$s_i = x_i \oplus y_i \oplus c_i$$

$c_i \backslash x_i y_i$	00	01	11	10
0			1	
1		1	1	1

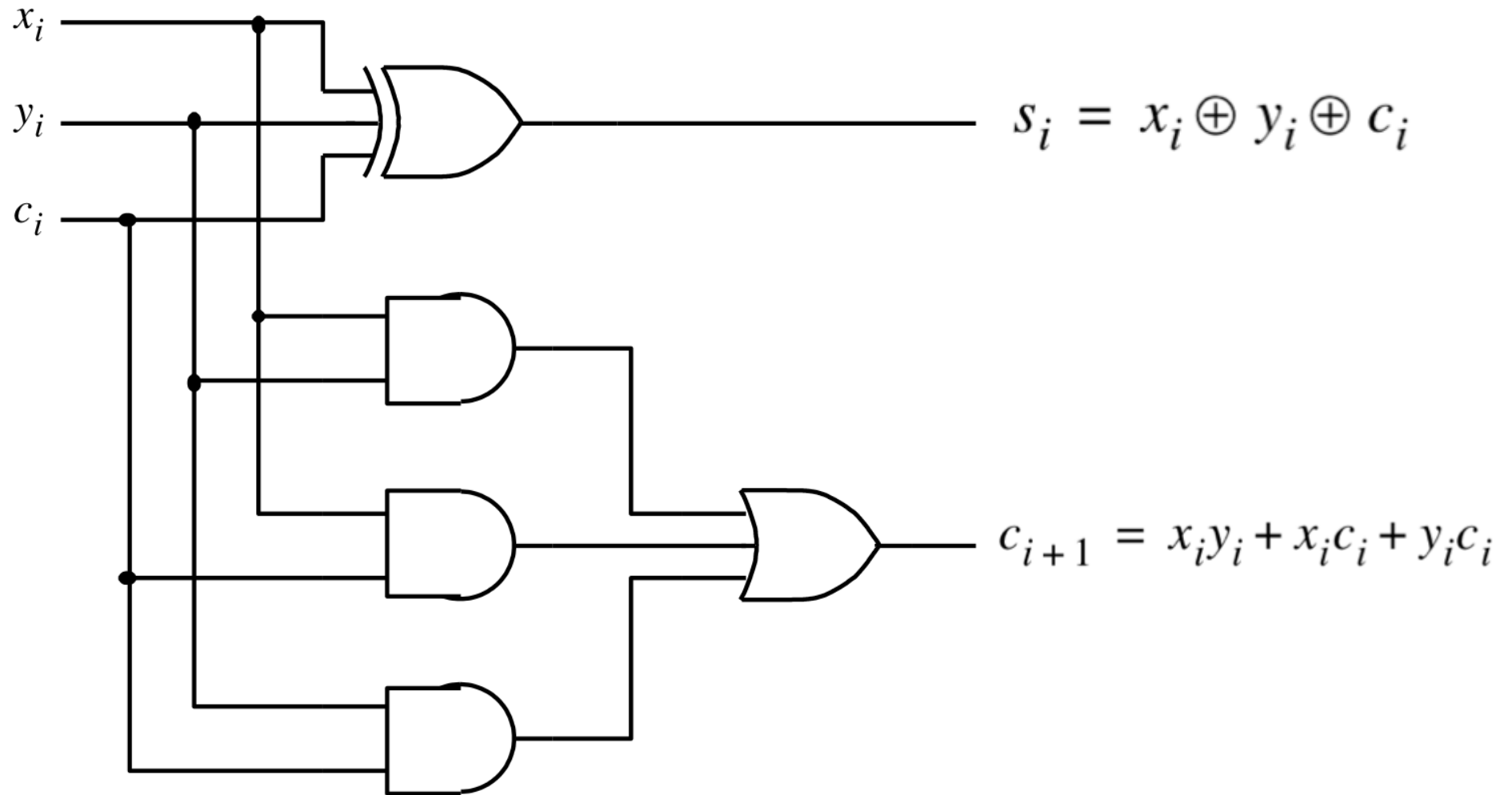
$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

The circuit for the two expressions



[Figure 3.3c from the textbook]

This is called the Full-Adder



[Figure 3.3c from the textbook]

XOR Magic

$$s_i = \bar{x}_i y_i \bar{c}_i + x_i \bar{y}_i \bar{c}_i + \bar{x}_i \bar{y}_i c_i + x_i y_i c_i$$

XOR Magic

$$s_i = \bar{x}_i y_i \bar{c}_i + x_i \bar{y}_i \bar{c}_i + \bar{x}_i \bar{y}_i c_i + x_i y_i c_i$$

$$s_i = (\bar{x}_i y_i + x_i \bar{y}_i) \bar{c}_i + (\bar{x}_i \bar{y}_i + x_i y_i) c_i$$

$$= (x_i \oplus y_i) \bar{c}_i + \overline{(x_i \oplus y_i)} c_i$$

$$= (x_i \oplus y_i) \oplus c_i$$

XOR Magic

$$s_i = \bar{x}_i y_i \bar{c}_i + x_i \bar{y}_i \bar{c}_i + \bar{x}_i \bar{y}_i c_i + x_i y_i c_i$$

Can you prove this?

$$s_i = (\bar{x}_i y_i + x_i \bar{y}_i) \bar{c}_i + (\bar{x}_i \bar{y}_i + x_i y_i) c_i$$

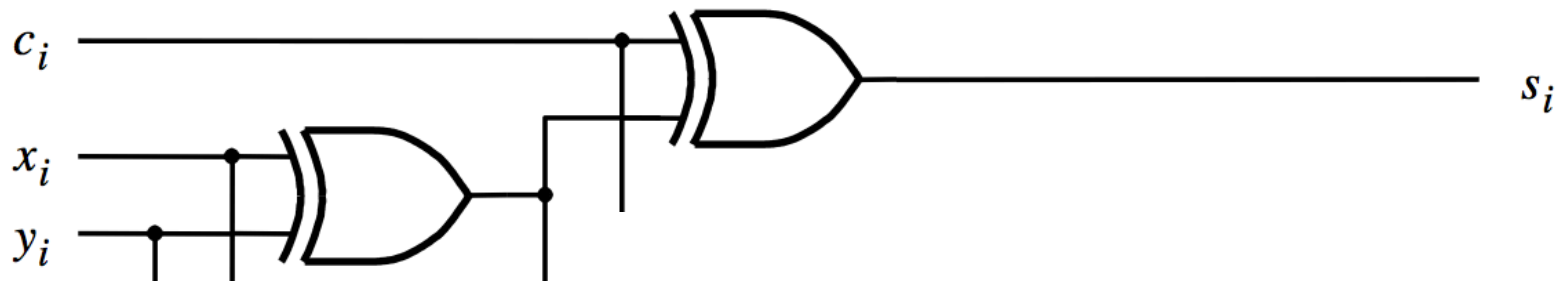
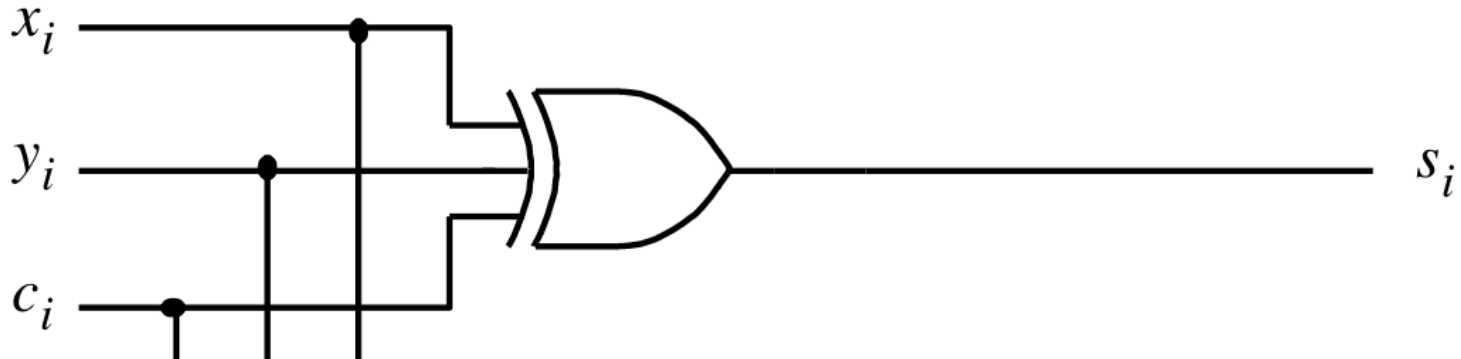
$$= (x_i \oplus y_i) \bar{c}_i + \overline{(x_i \oplus y_i)} c_i$$

$$= (x_i \oplus y_i) \oplus c_i$$

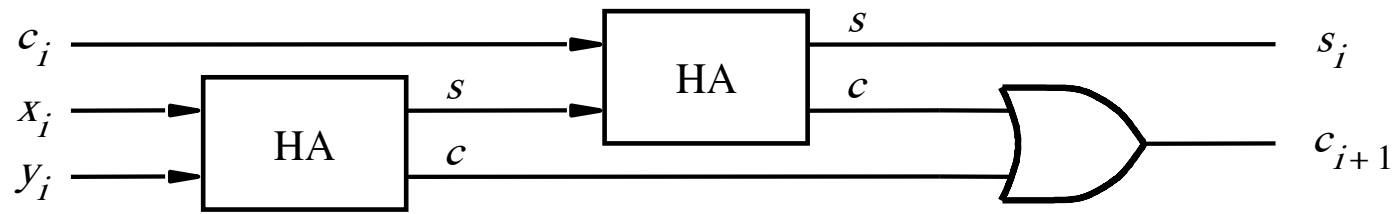
XOR Magic

(s_i can be implemented in two different ways)

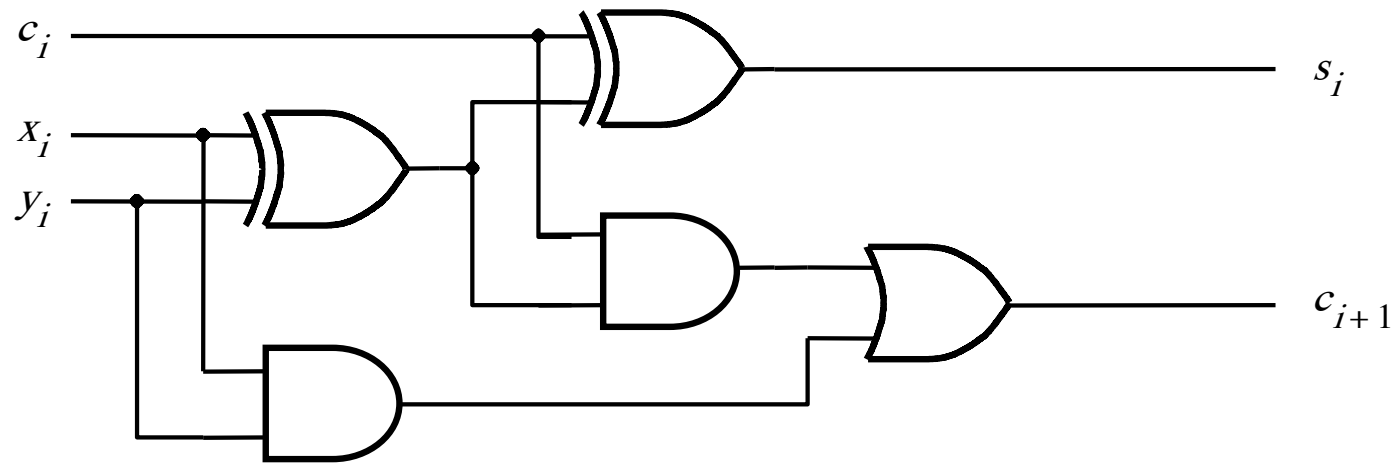
$$s_i = x_i \oplus y_i \oplus c_i$$



A decomposed implementation of the full-adder circuit

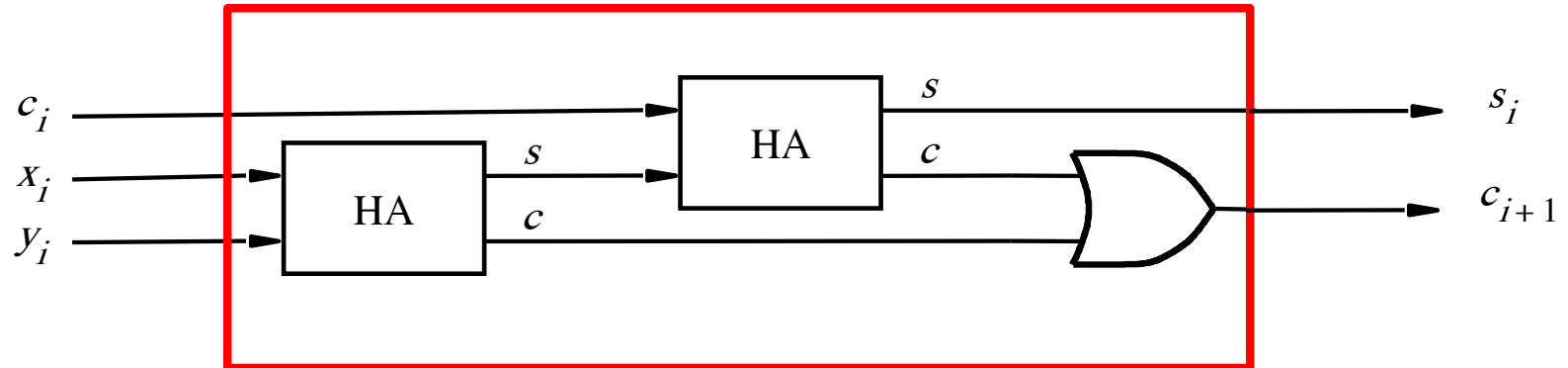


(a) Block diagram

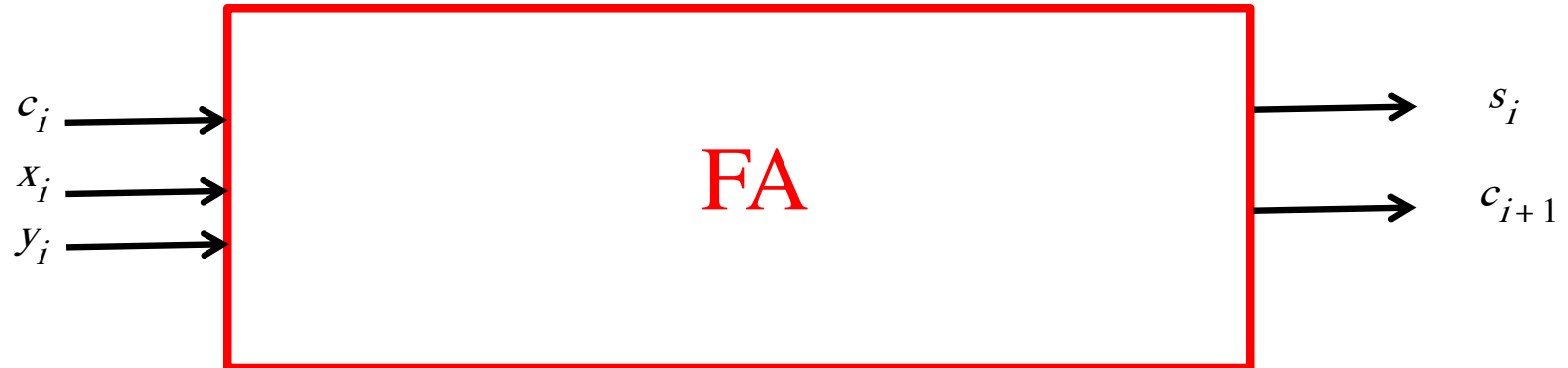


(b) Detailed diagram

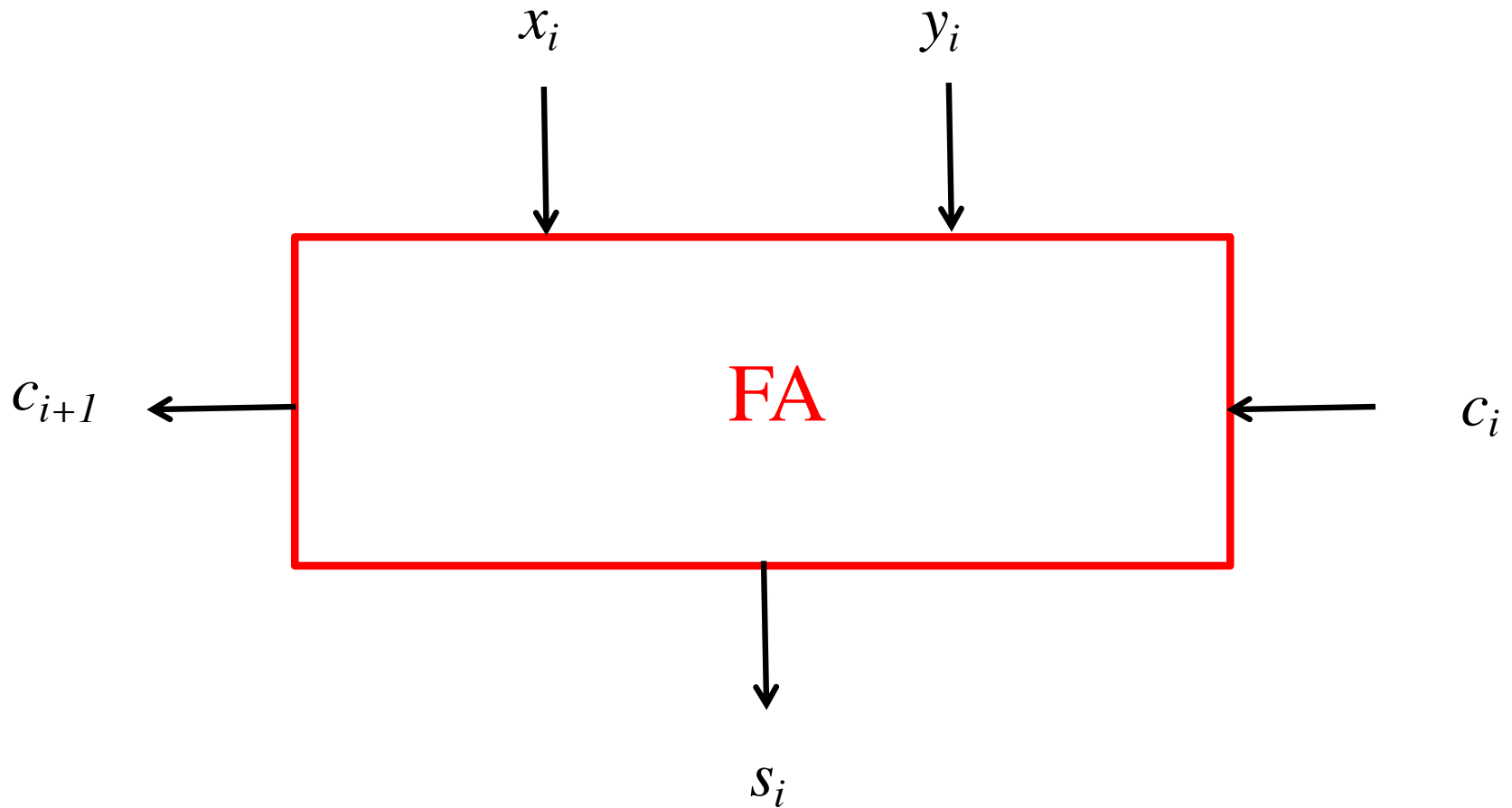
The Full-Adder Abstraction



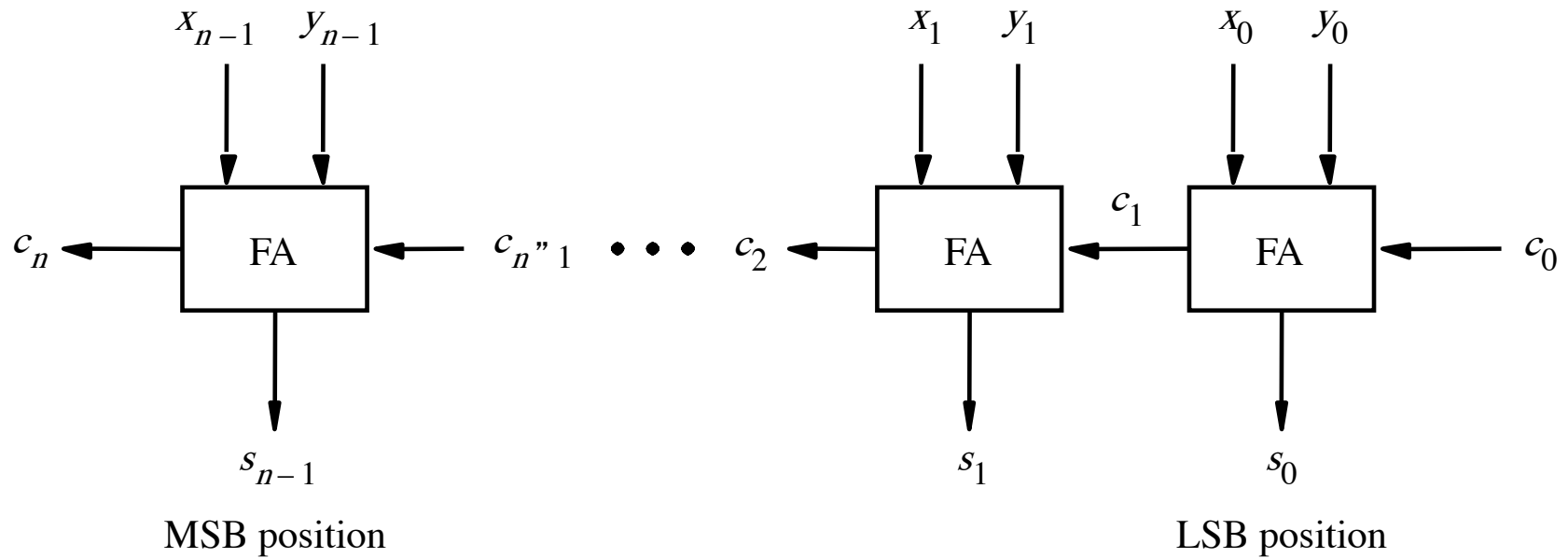
The Full-Adder Abstraction



We can place the arrows anywhere

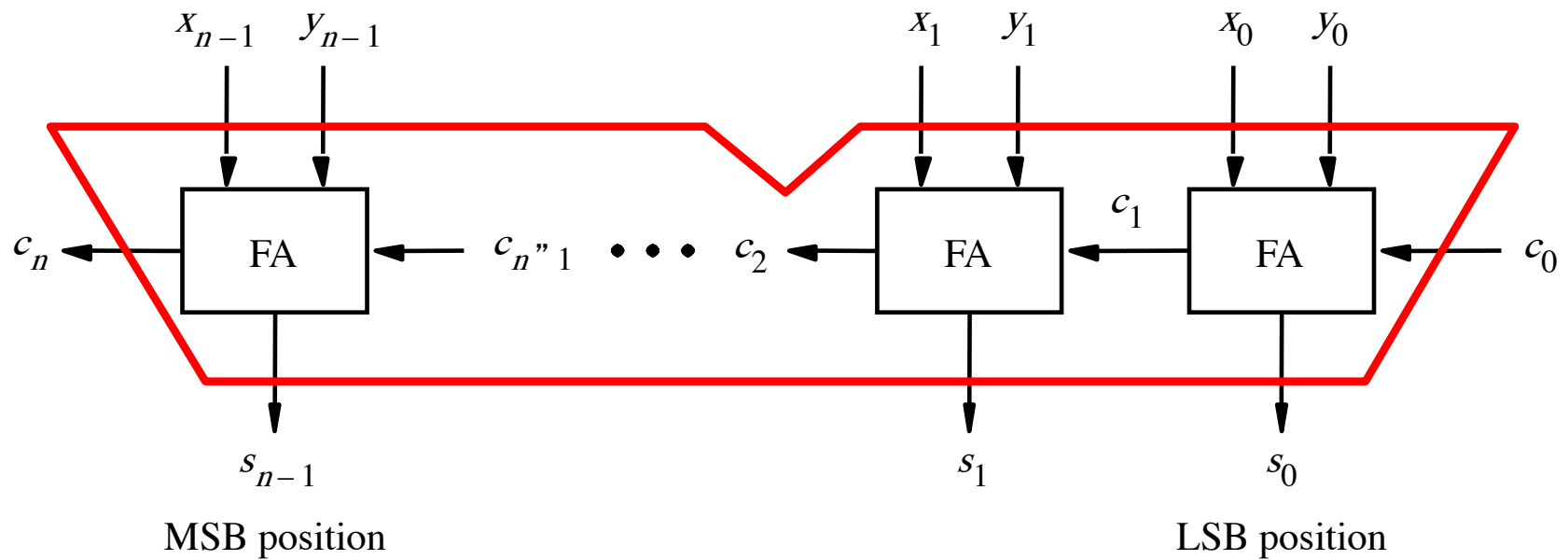


n-bit ripple-carry adder

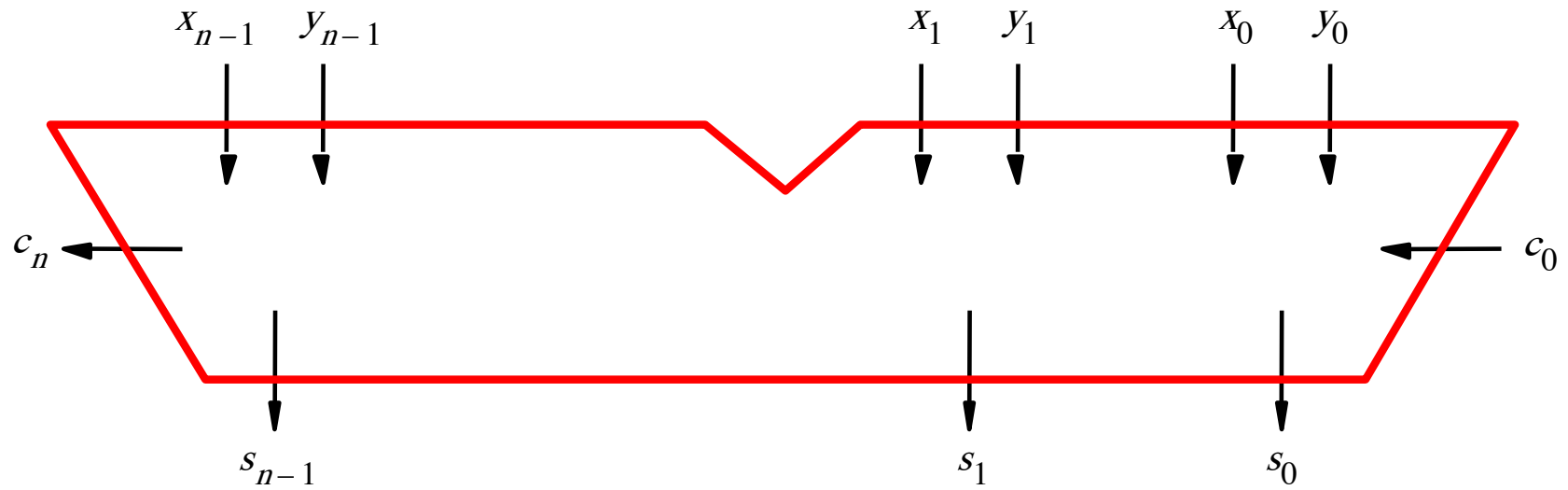


[Figure 3.5 from the textbook]

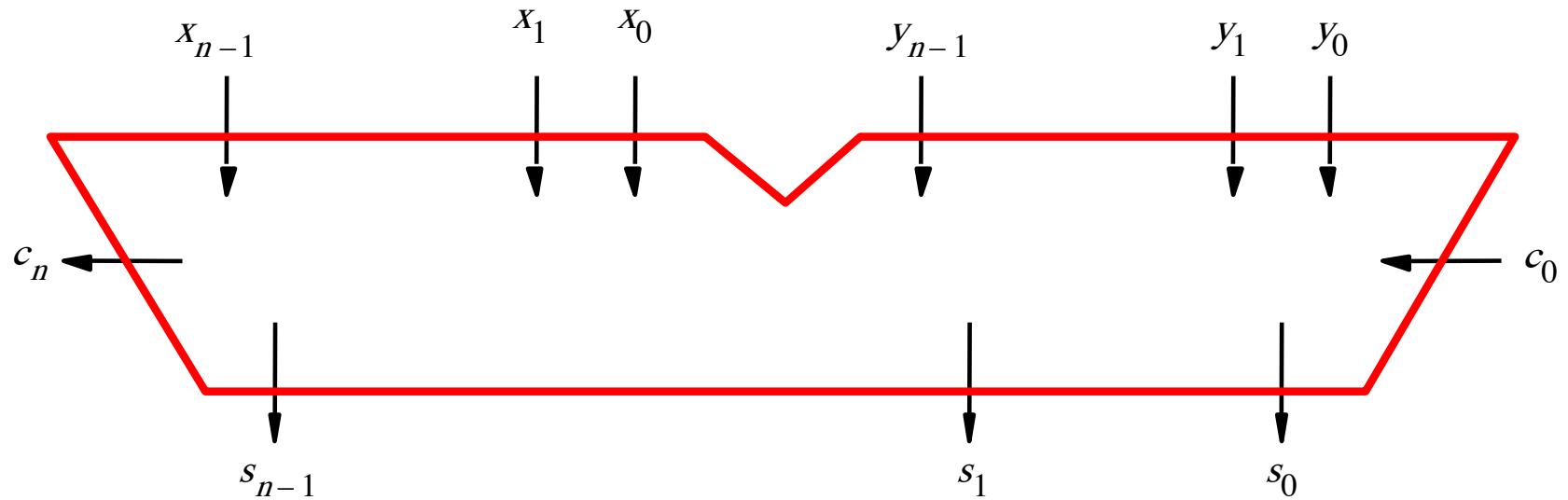
n-bit ripple-carry adder abstraction



n-bit ripple-carry adder abstraction



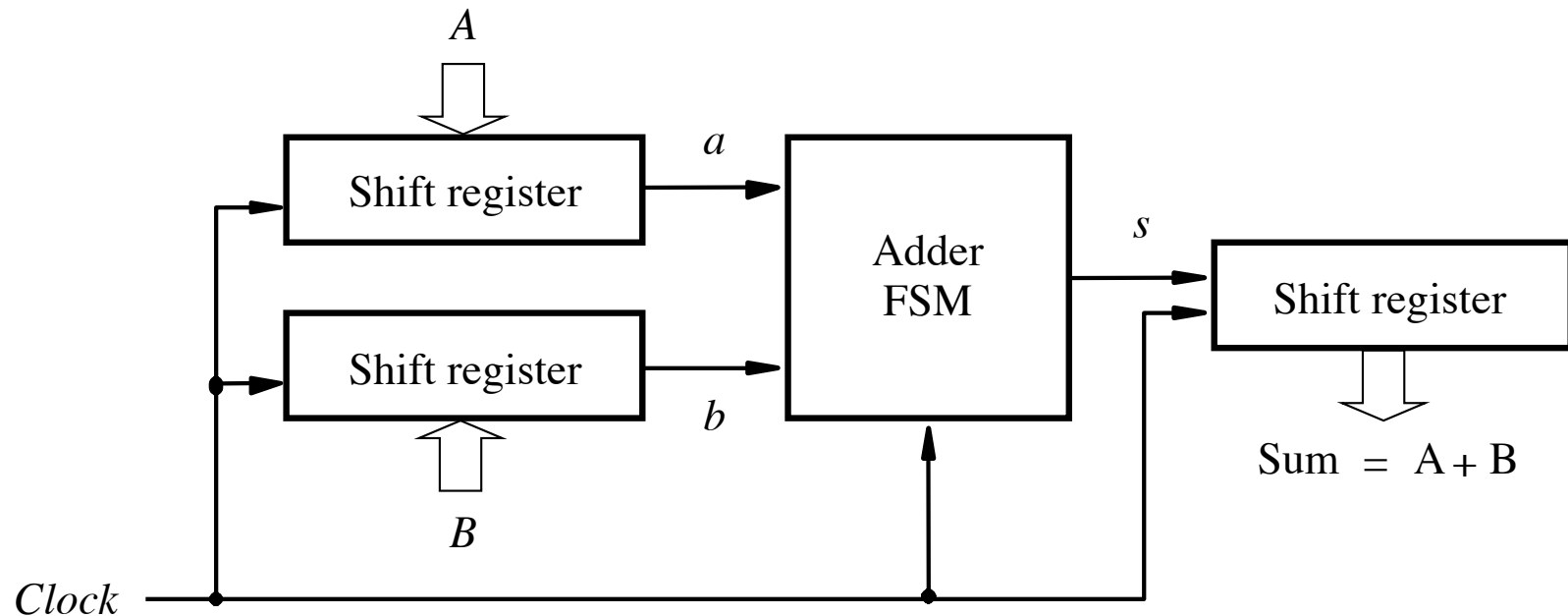
The x and y lines are typically grouped together for better visualization, but the underlying logic remains the same



Serial Adder

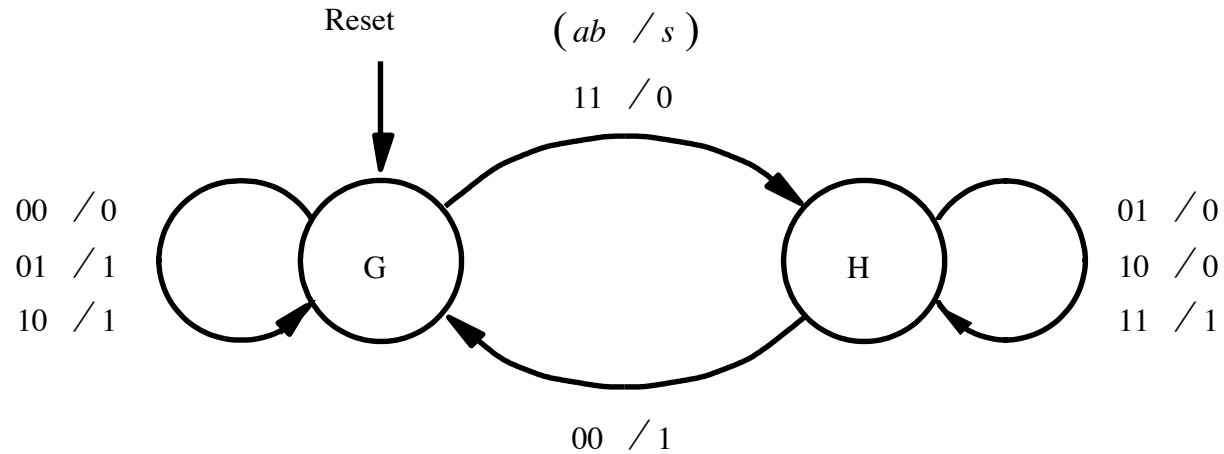
- **The n-bit adder requires all bits to be provided at the same time.**
- **In some cases we may want to add the numbers as the bits come in.**
- **Also, with an n-bit adder we are limited to n-bits. Circuits for larger n are more complex.**
- **Can we add arbitrarily long numbers.**

Block diagram for the serial adder



Mealy Machine Implementation

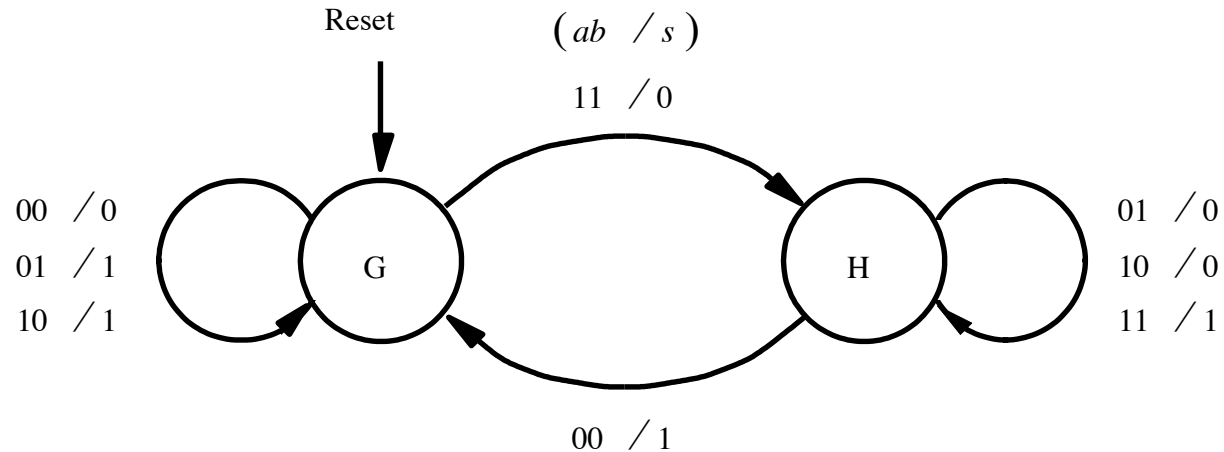
State diagram for the serial adder FSM



G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

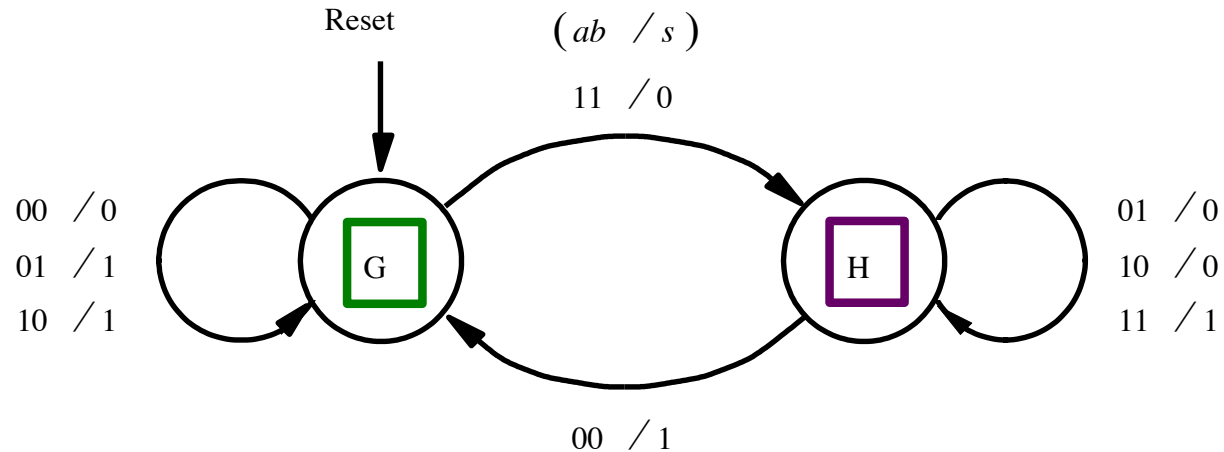


c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

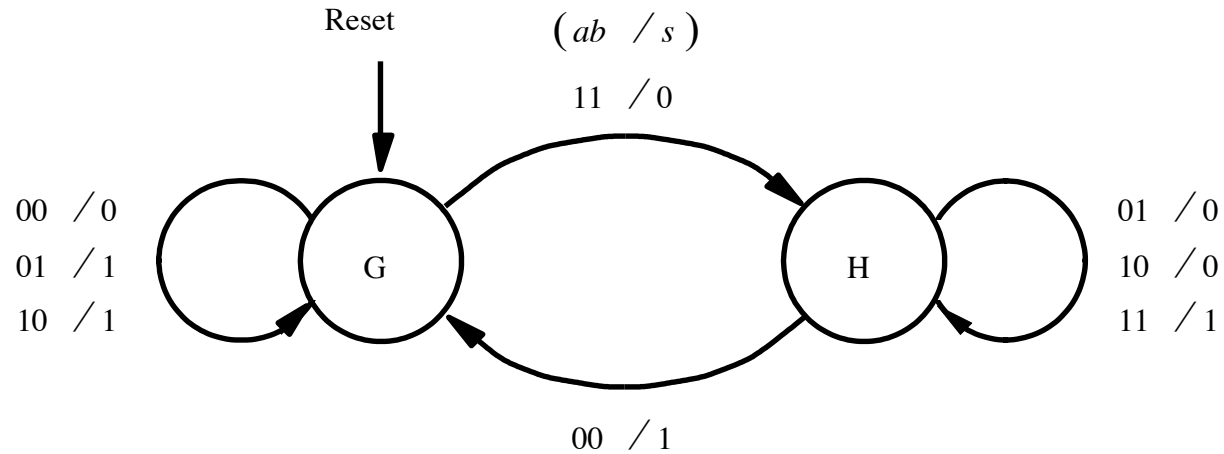


c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

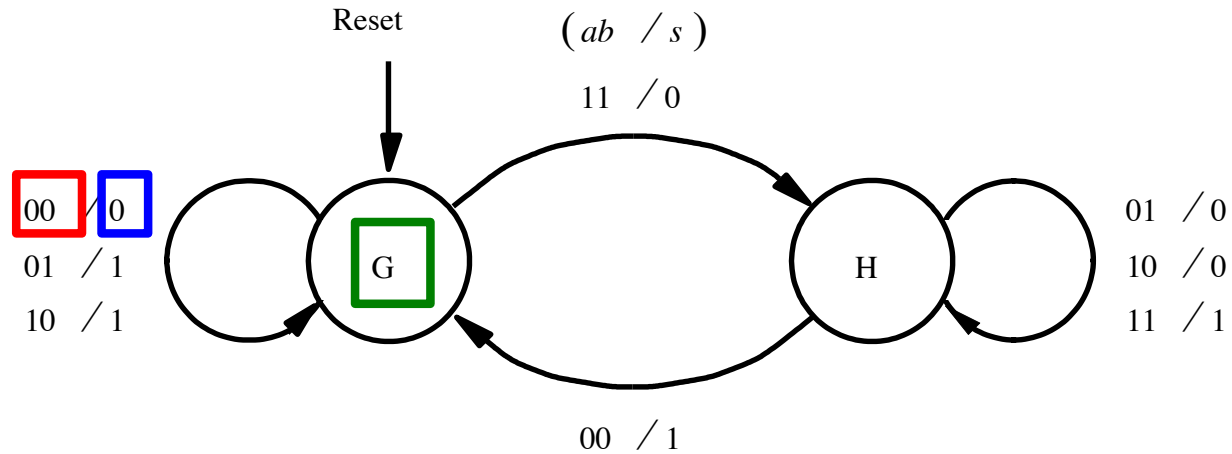


c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

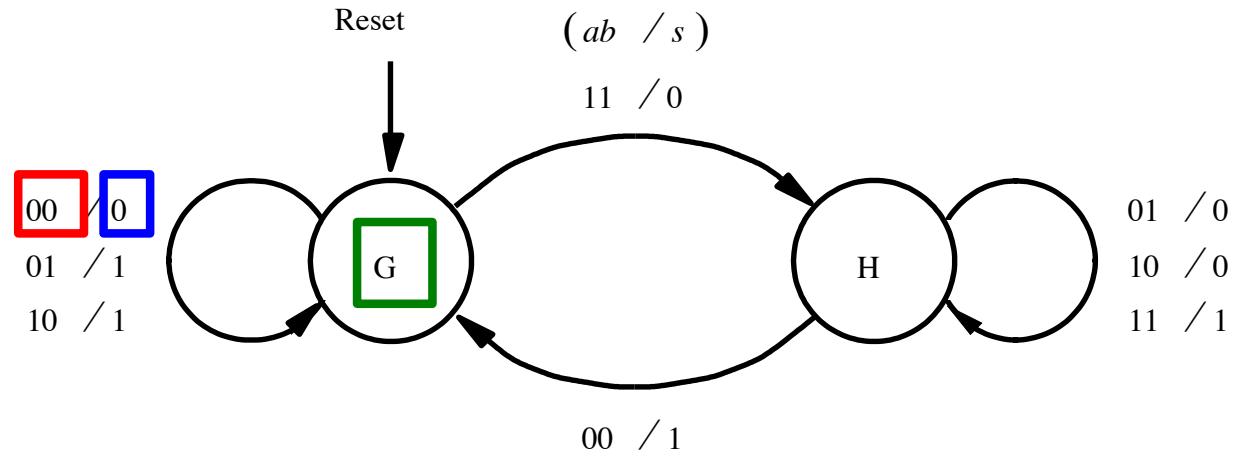


c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

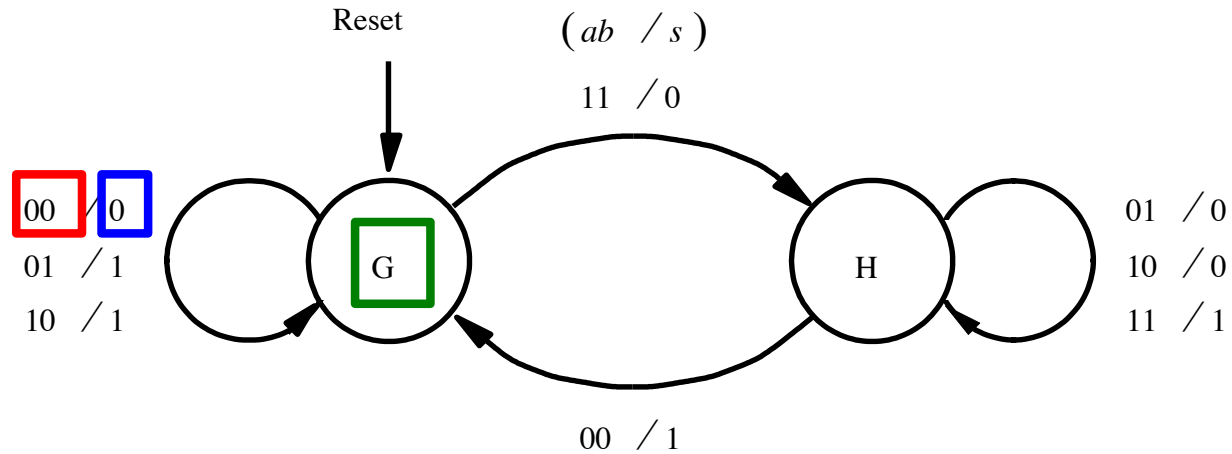


c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

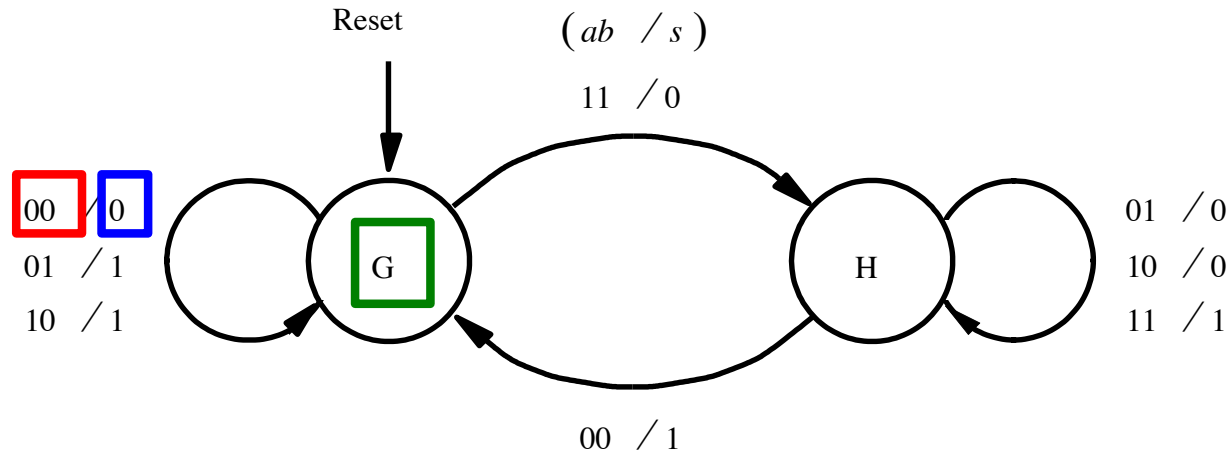


c_i	x_i	y_i	c_{i+1}	s_i
G	0	0	G	0
G	0	1	G	1
G	1	0	G	1
G	1	1	H	0
H	0	0	G	1
H	0	1	H	0
H	1	0	H	0
H	1	1	H	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

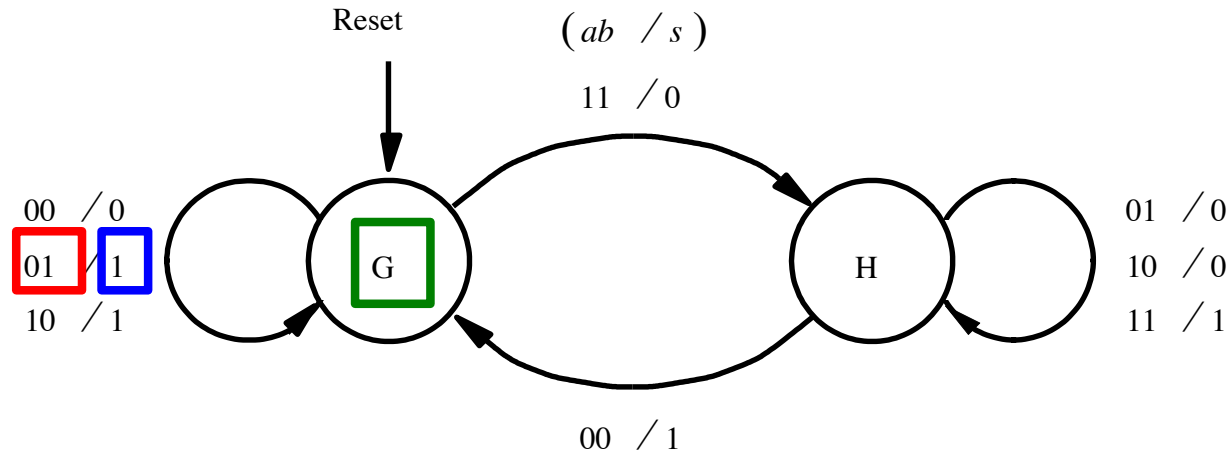


c_i	x_i	y_i	c_{i+1}	s_i
G	0	0	G	0
G	0	1	G	1
G	1	0	G	1
G	1	1	H	0
H	0	0	G	1
H	0	1	H	0
H	1	0	H	0
H	1	1	H	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

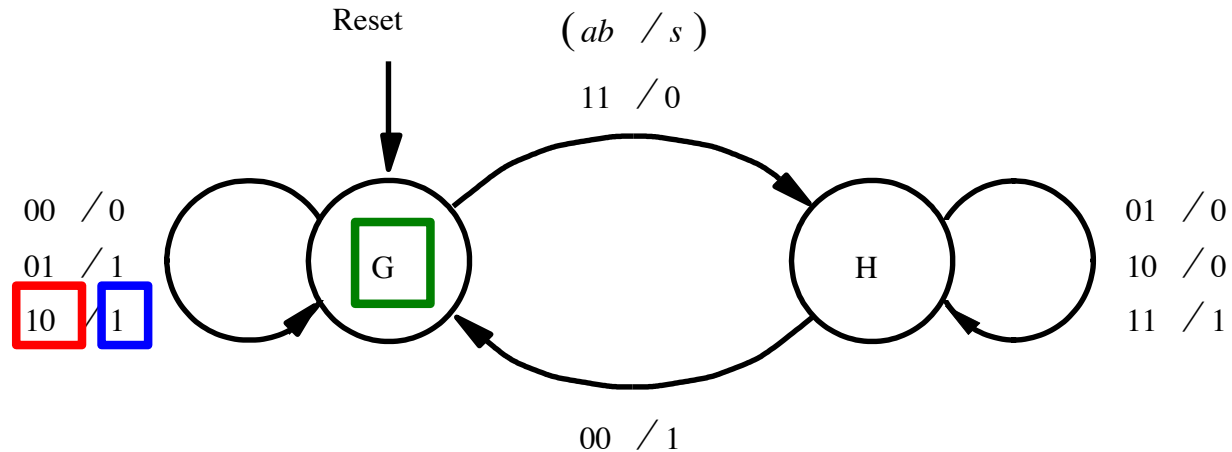


c_i	x_i	y_i	c_{i+1}	s_i
G	0	0	G	0
G	0	1	G	1
G	1	0	G	1
G	1	1	H	0
H	0	0	G	1
H	0	1	H	0
H	1	0	H	0
H	1	1	H	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

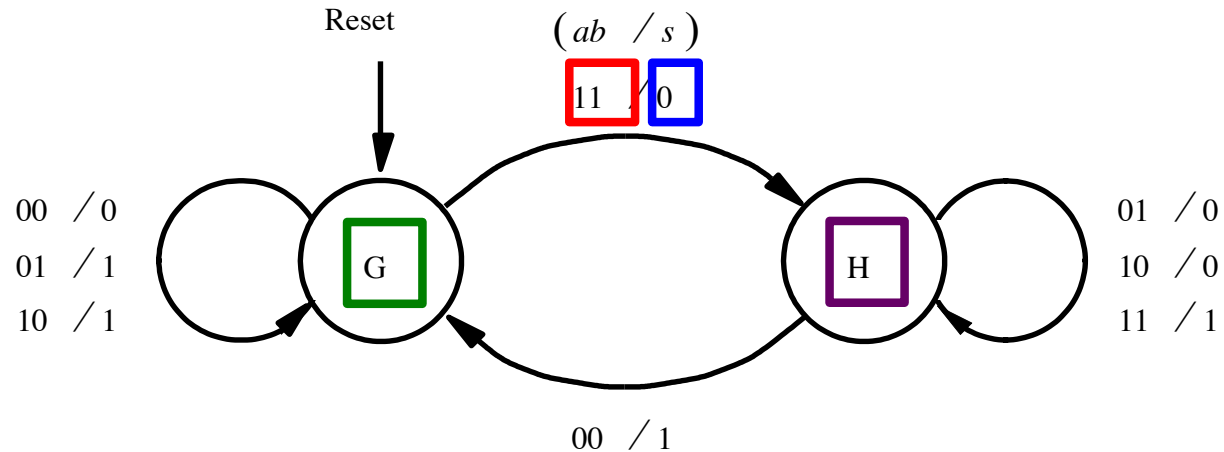


c_i	x_i	y_i	c_{i+1}	s_i
G	0	0	G	0
G	0	1	G	1
G	1	0	G	1
G	1	1	H	0
H	0	0	G	1
H	0	1	H	0
H	1	0	H	0
H	1	1	H	1

G: carry-in = 0

H: carry-in = 1

State diagram for the serial adder FSM

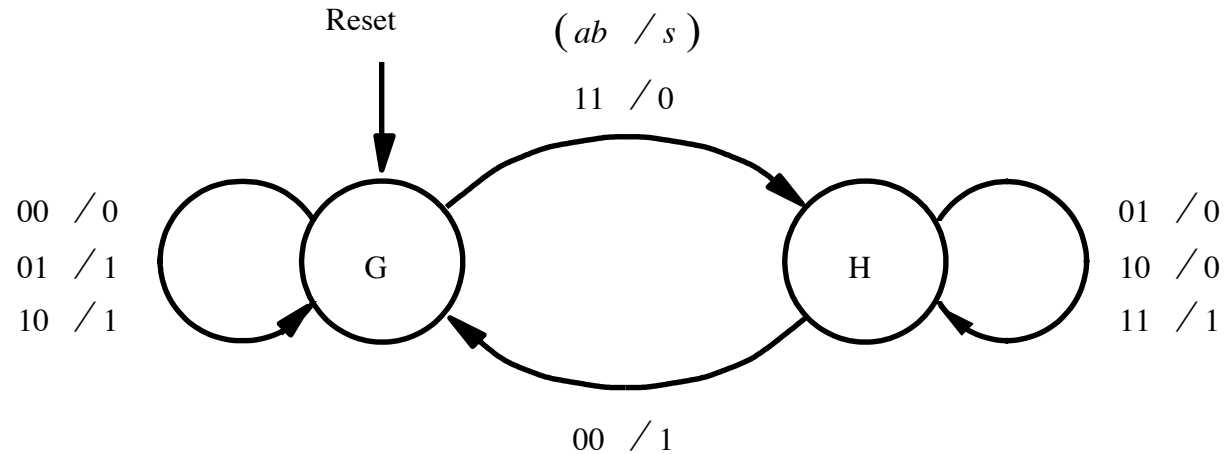


c_i	x_i	y_i	c_{i+1}	s_i
G	0	0	G	0
G	0	1	G	1
G	1	0	G	1
G	1	1	H	0
H	0	0	G	1
H	0	1	H	0
H	1	0	H	0
H	1	1	H	1

G: carry-in = 0

H: carry-in = 1

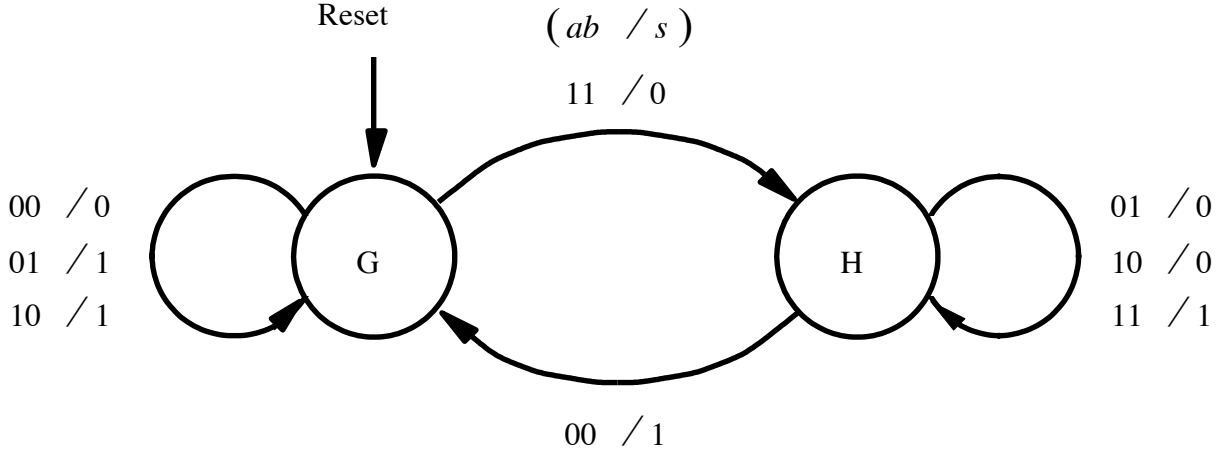
State diagram for the serial adder FSM



State table for the serial adder FSM

Present state	Next state				Output s			
	$ab = 00$	01	10	11	00	01	10	11
G								
H								

State diagram for the serial adder FSM

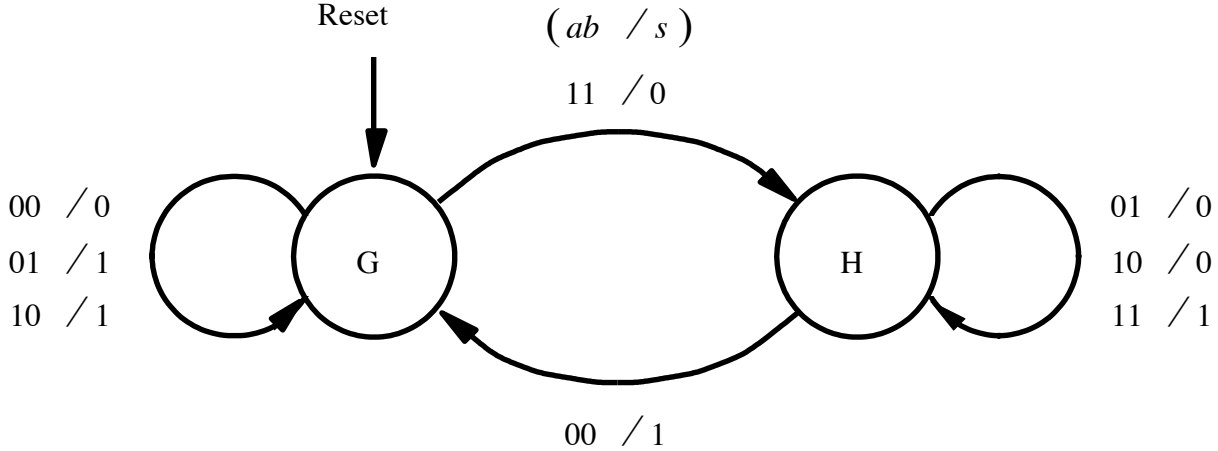


State table for the serial adder FSM

Present state	Next state				Output <i>s</i>			
	<i>ab</i> = 00	01	10	11	00	01	10	11
G	G	G	G	H				
H	G	H	H	H				

[Figure 6.40 & 6.41 from the textbook]

State diagram for the serial adder FSM



State table for the serial adder FSM

Present state	Next state				Output <i>s</i>			
	<i>ab</i> = 00	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

[Figure 6.40 & 6.41 from the textbook]

State table for the serial adder FSM

Present state	Next state				Output s			
	$ab = 00$	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

State table for the serial adder FSM

Present state	Next state				Output s			
	$ab = 00$	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

State-assigned table for the serial adder

Present state	Next state				Output			
	$ab = 00$	01	10	11	00	01	10	11
y	Y				s			
0								
1								

State table for the serial adder FSM

Present state	Next state				Output s			
	$ab = 00$	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

State-assigned table for the serial adder

Present state	Next state				Output			
	$ab = 00$	01	10	11	00	01	10	11
y	Y				s			
0	0	0	0	1				
1	0	1	1	1				

State table for the serial adder FSM

Present state	Next state				Output s			
	$ab = 00$	01	10	11	00	01	10	11
G	G	G	G	H	0	1	1	0
H	G	H	H	H	1	0	0	1

State-assigned table for the serial adder

Present state	Next state				Output			
	$ab = 00$	01	10	11	00	01	10	11
y	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Derivation of Y and s

Present state y	Next state				Output			
	$ab = 00$	01	10	11	00	01	10	11
	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

Derivation of Y and s

Present state y	Next state				Output			
	$ab=00$	01	10	11	00	01	10	11
	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

y	a	b	Y	s
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

Derivation of Y and s

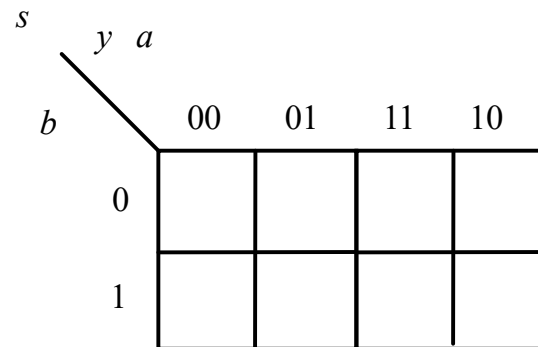
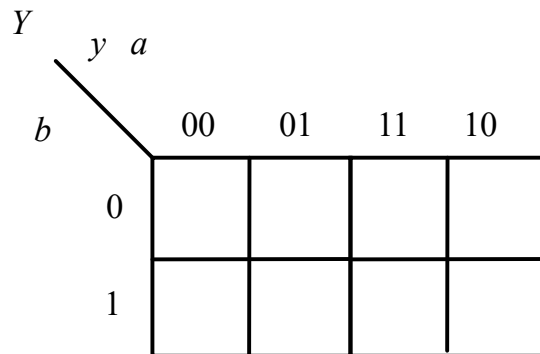
Present state y	Next state				Output			
	$ab=00$	01	10	11	00	01	10	11
	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

y	a	b	Y	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Derivation of Y and s

Present state y	Next state				Output			
	$ab=00$	01	10	11	00	01	10	11
	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

y	a	b	Y	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Derivation of Y and s

Present state y	Next state				Output			
	$ab=00$	01	10	11	00	01	10	11
	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

y	a	b	Y	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Y

y	a				
		00	01	11	10
b	0	0	0	1	0
1	0	1	1	1	

s

y	a				
		00	01	11	10
b	0	0	1	0	1
1	1	0	1	0	

Derivation of Y and s

Present state y	Next state				Output			
	$ab=00$	01	10	11	00	01	10	11
	Y				s			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

y	a	b	Y	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Y

y	a	b			
		00	01	11	10
0		0	0	1	0
1		0	1	1	1

s

y	a	b			
		00	01	11	10
0		0	1	0	1
1		1	0	1	0

Derivation of Y and s

Present state <i>y</i>	Next state				Output			
	<i>ab</i> = 00	01	10	11	00	01	10	11
	<i>Y</i>				<i>s</i>			
0	0	0	0	1	0	1	1	0
1	0	1	1	1	1	0	0	1

<i>y</i>	<i>a</i>	<i>b</i>	<i>Y</i>	<i>s</i>
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Y

<i>y</i>	<i>a</i>			
<i>b</i>	00	01	11	10
0	0	0	1	0
1	0	1	1	1

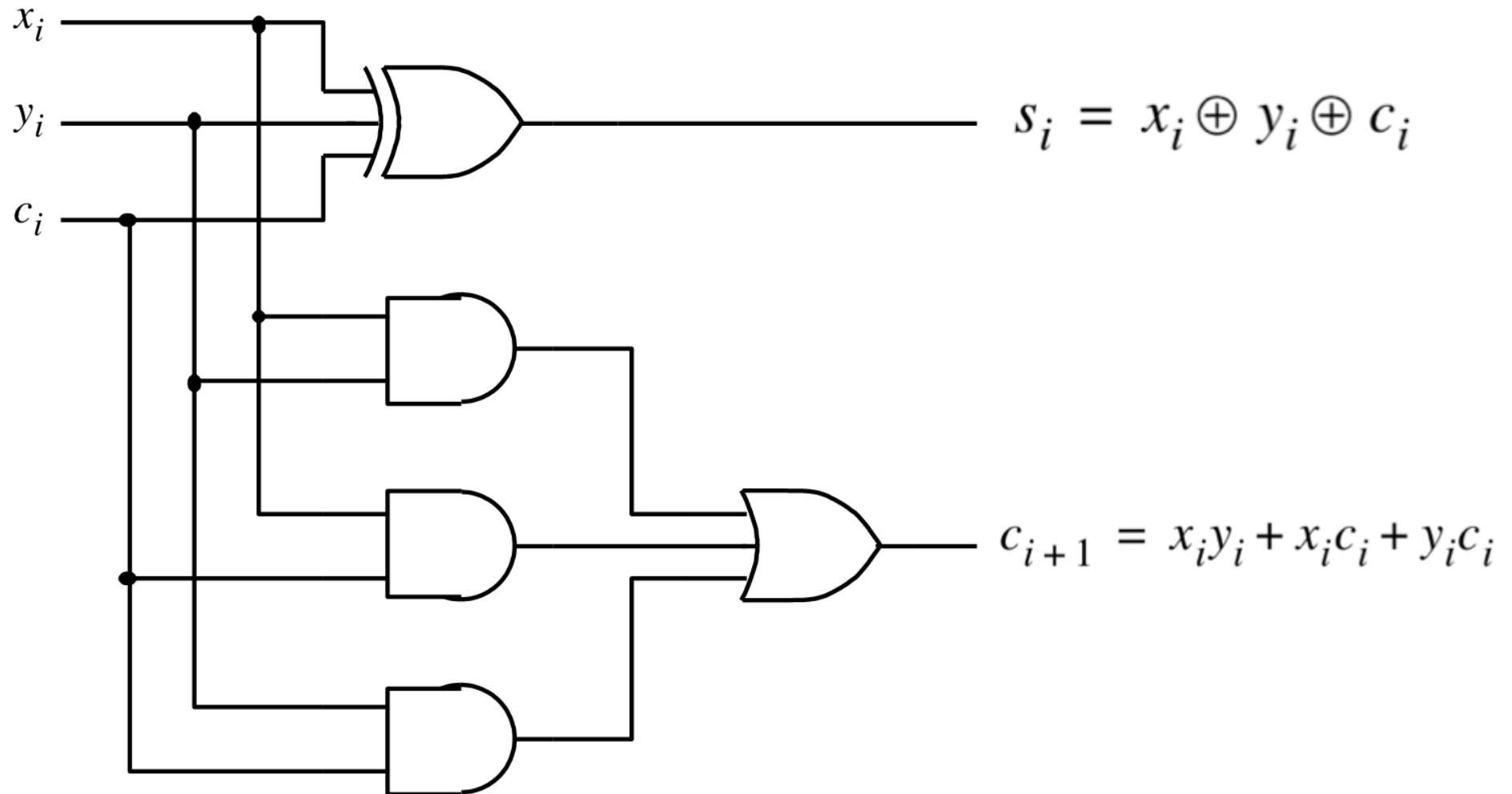
s

<i>y</i>	<i>a</i>			
<i>b</i>	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$Y = ab + ay + by$$

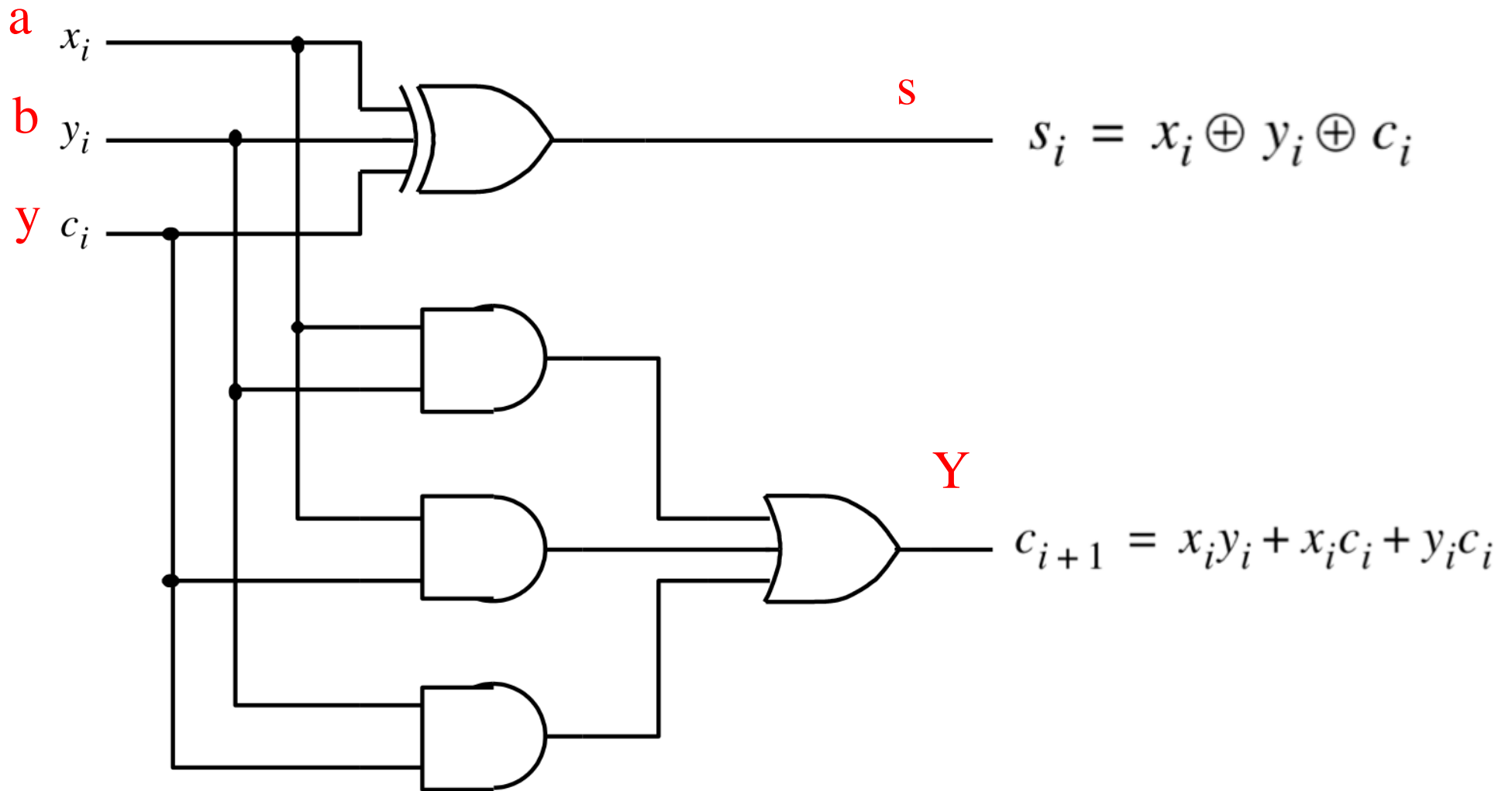
$$s = \text{XOR}(\text{XOR}(a, b), y)$$

The circuit for the two expressions



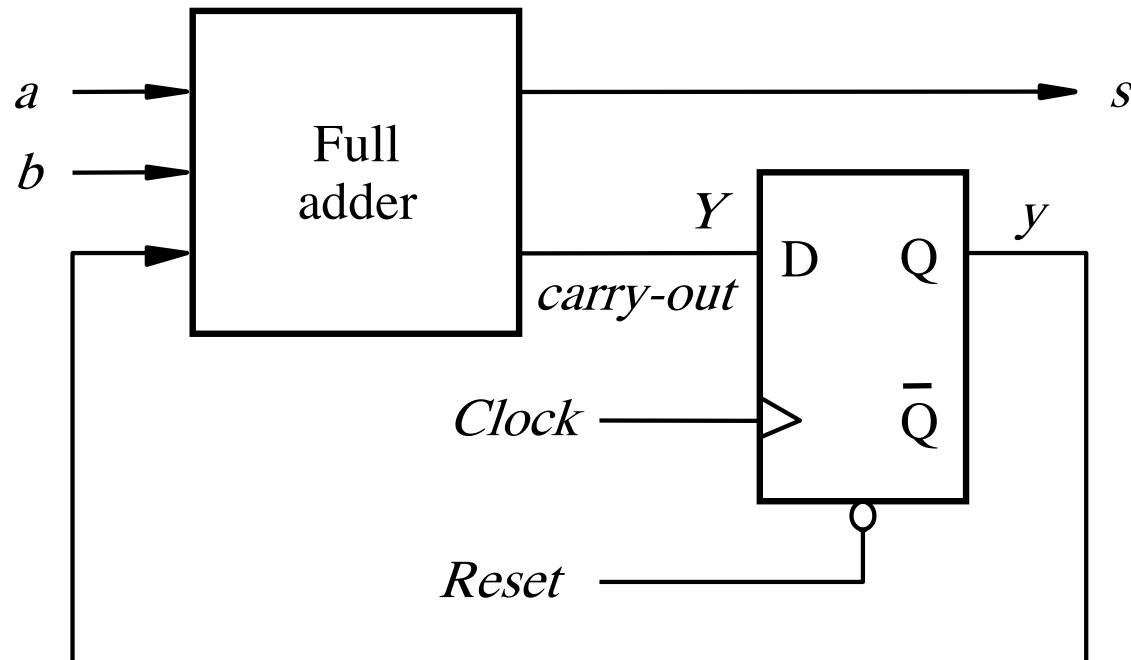
[Figure 3.3c from the textbook]

The circuit for the two expressions



[Figure 3.3c from the textbook]

Circuit for the serial adder FSM



$$Y = ab + ay + by$$

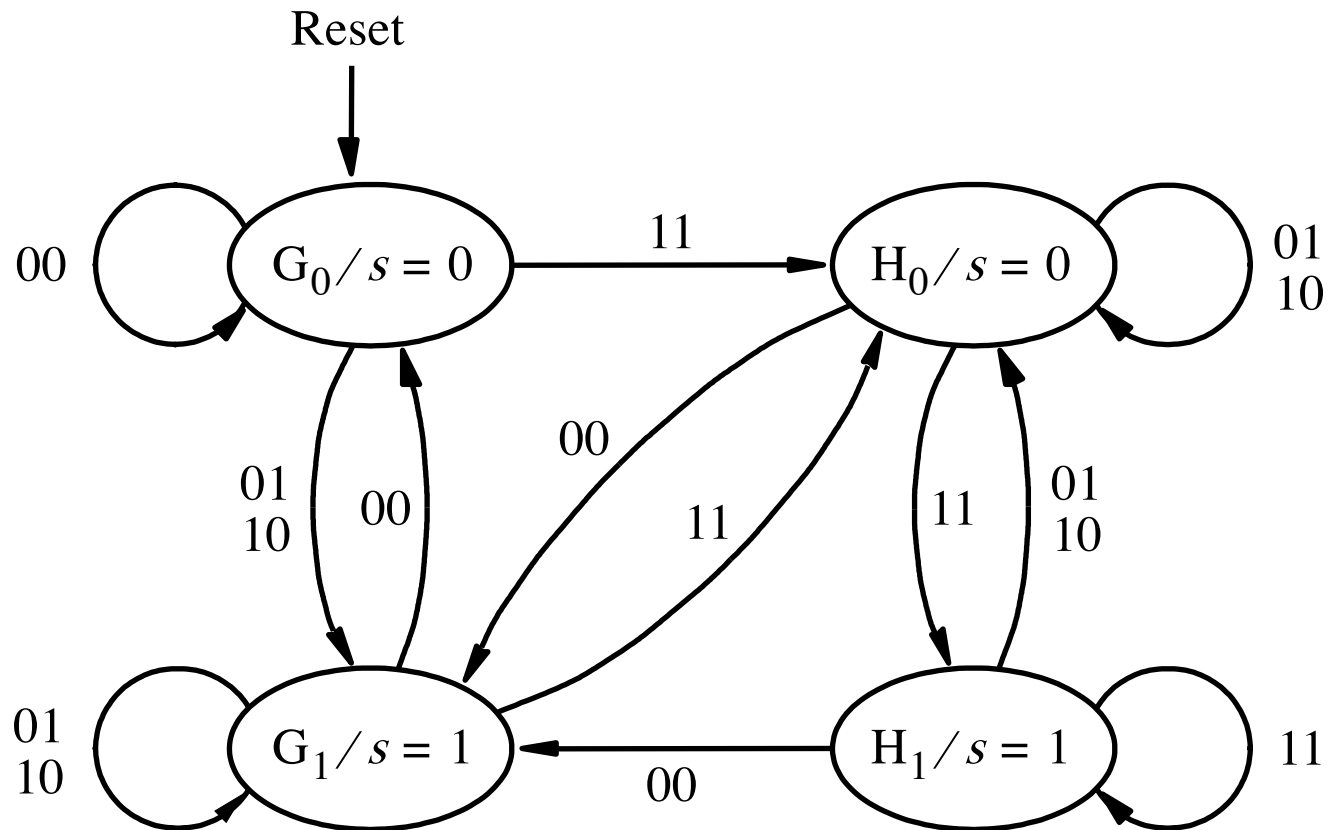
(carry bit from FA)

$$s = \text{XOR}(\text{XOR}(a, b), y)$$

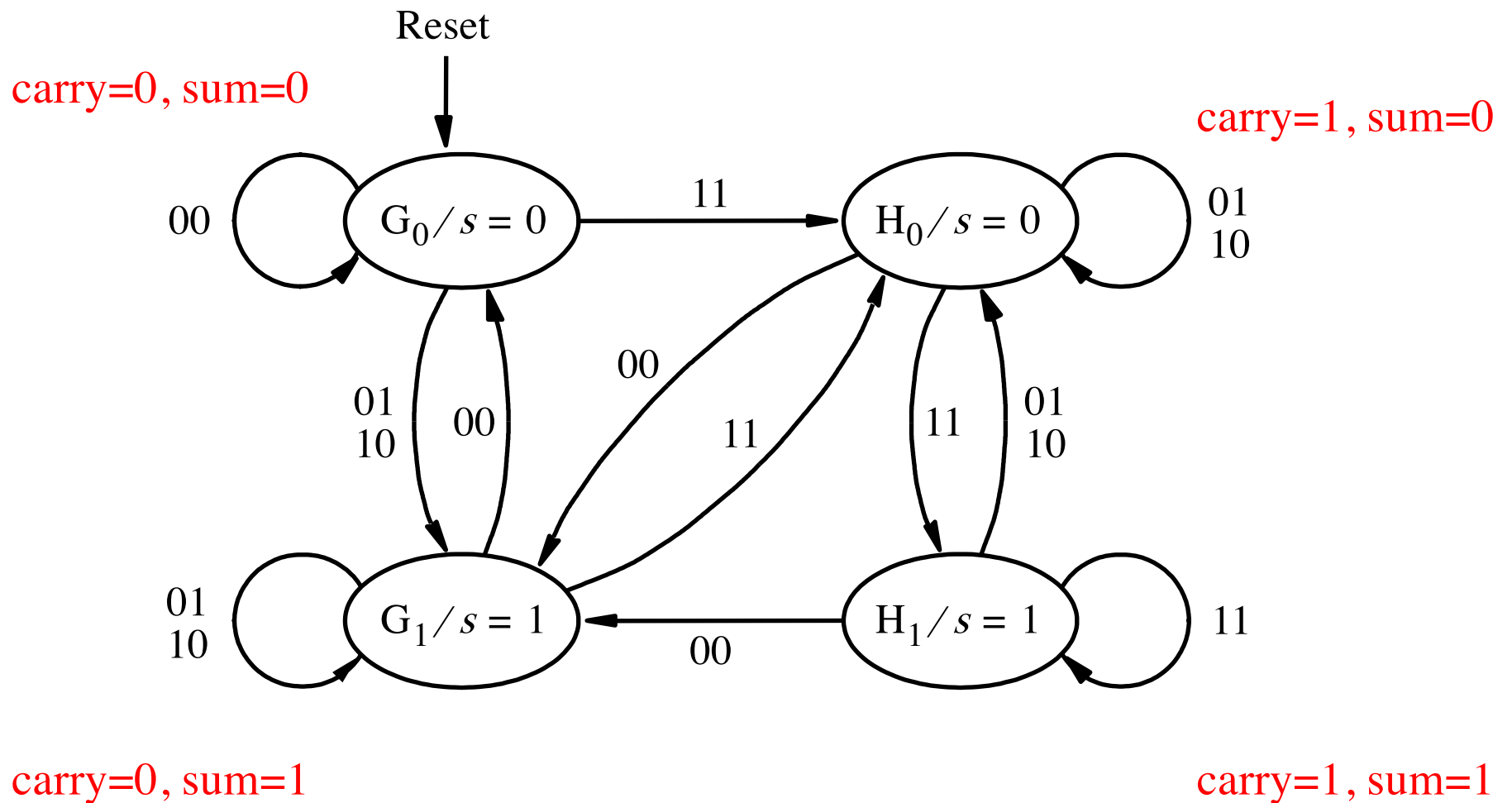
(sum bit from FA)

Moore Machine Implementation

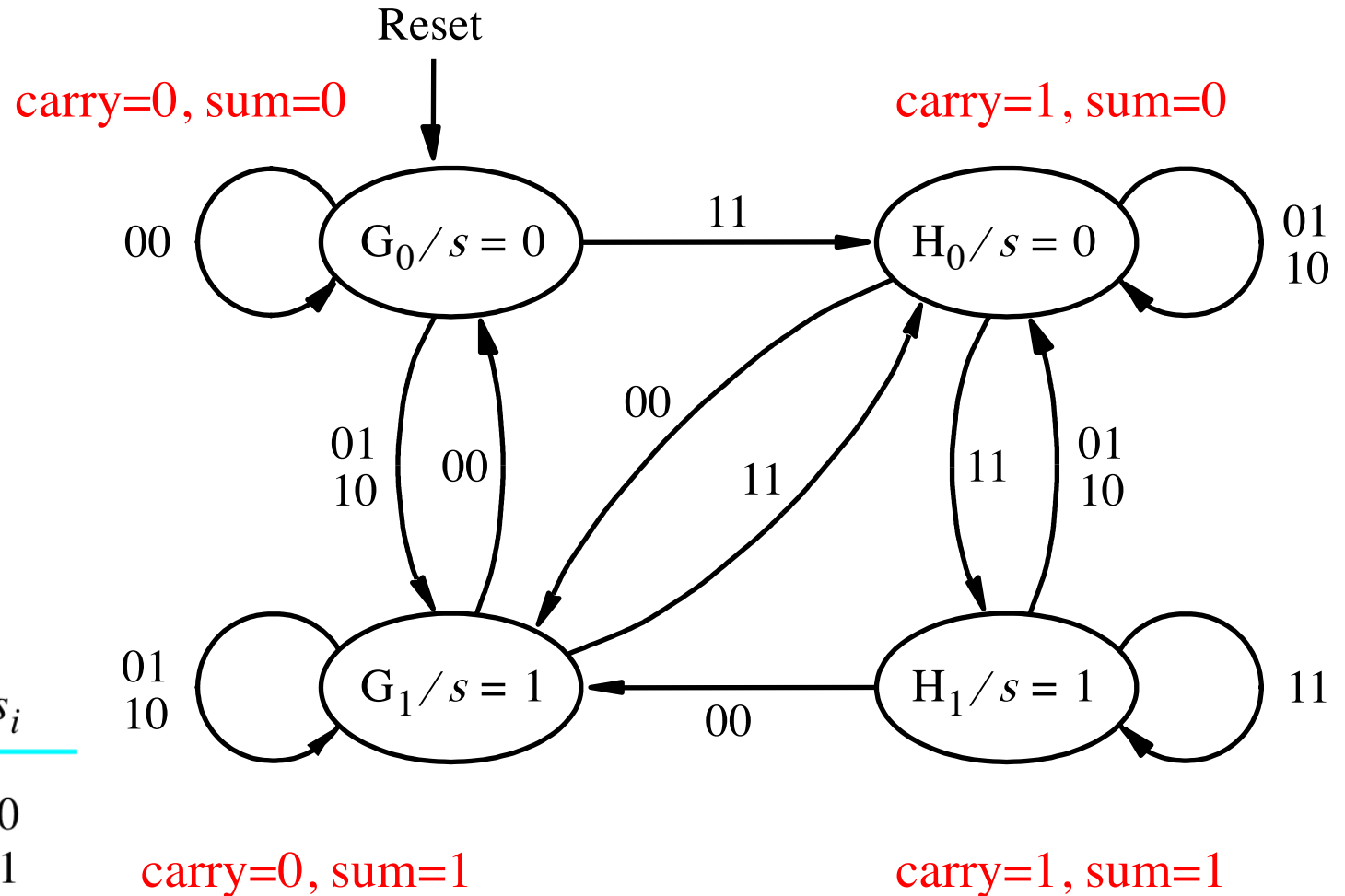
State diagram for the Moore-type serial adder FSM



State diagram for the Moore-type serial adder FSM



State diagram for the Moore-type serial adder FSM

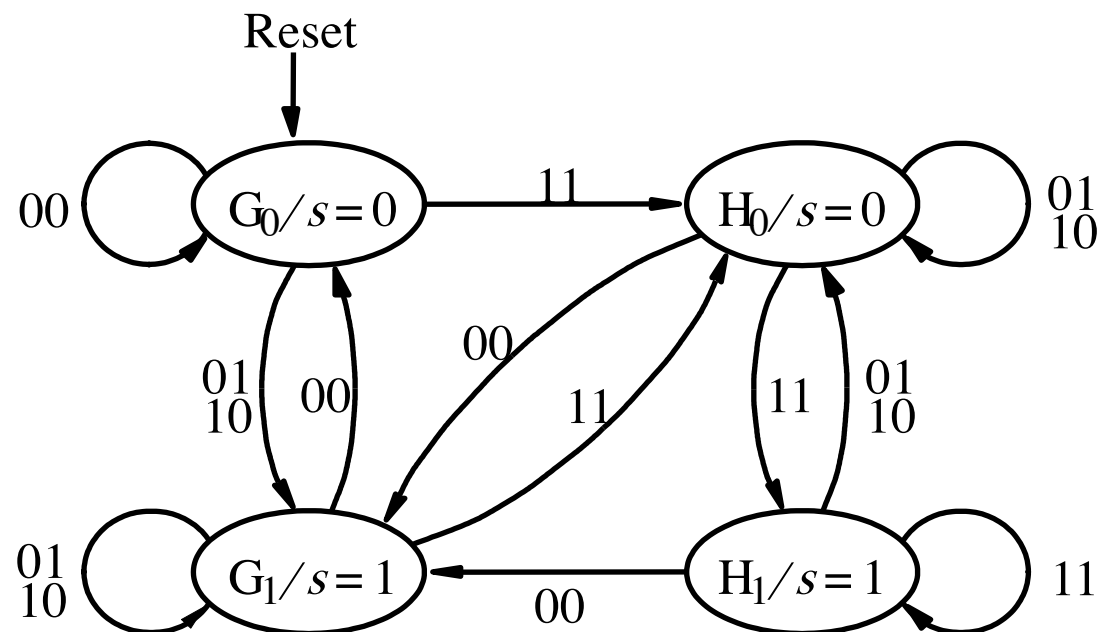


c_i	x_i	y_i	c_{i+1}	s_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

[Figure 6.44 from the textbook]

State table for the Moore-type serial adder FSM

Present state	Next state				Output s
	$ab = 00$	01	10	11	
G_0	G_0	G_1	G_1	H_0	0
G_1	G_0	G_1	G_1	H_0	1
H_0	G_1	H_0	H_0	H_1	0
H_1	G_1	H_0	H_0	H_1	1



[Figure 6.45 from the textbook]

State table for the Moore-type serial adder FSM

Present state	Next state				Output s
	$ab = 00$	01	10	11	
G_0	G_0	G_1	G_1	H_0	0
G_1	G_0	G_1	G_1	H_0	1
H_0	G_1	H_0	H_0	H_1	0
H_1	G_1	H_0	H_0	H_1	1

State table for the Moore-type serial adder FSM

Present state	Next state				Output s
	$ab = 00$	01	10	11	
G_0	G_0	G_1	G_1	H_0	0
G_1	G_0	G_1	G_1	H_0	1
H_0	G_1	H_0	H_0	H_1	0
H_1	G_1	H_0	H_0	H_1	1

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00					
01					
10					
11					

State table for the Moore-type serial adder FSM

Present state	Next state				Output s
	$ab = 00$	01	10	11	
G_0	G_0	G_1	G_1	H_0	0
G_1	G_0	G_1	G_1	H_0	1
H_0	G_1	H_0	H_0	H_1	0
H_1	G_1	H_0	H_0	H_1	1

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

[Figure 6.45 & 6.46 from the textbook]

State-assigned table for the Moore-type serial adder FSM

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

Deriving Y1, Y2, and s

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Deriving Y_1

Present state y_2y_1	Next state				Output s				
	$ab = 00$	01	10	11					
	<div style="display: flex; justify-content: space-around; align-items: center;"> <div style="border: 1px solid black; width: 15px; height: 15px; background-color: #ADD8E6; margin-right: 5px;"></div> Y_1 </div>								
00		0		1		1		0	0
01		0		1		1		1	0
10		1		0		0		0	1
11		1		0		0		1	1

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0		
0	0	0	1		
0	0	1	0		
0	0	1	1		
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		
1	0	0	0		
1	0	0	1		
1	0	1	0		
1	0	1	1		
1	1	0	0		
1	1	0	1		
1	1	1	0		
1	1	1	1		

Deriving Y_1

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	□ Y_1				
00	□ 0	□ 1	□ 1	□ 0	0
01	□ 0	□ 1	□ 1	□ 0	1
10	□ 1	□ 0	□ 0	□ 1	0
11	□ 1	□ 0	□ 0	□ 1	1

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	

Deriving Y_1

Present state y_2y_1	Next state				Output s
	$ab=00$	01	10	11	
	Y_1				
00	0	1	1	0	0
01	0	1	1	0	1
10	1	0	0	1	0
11	1	0	0	1	1

	y_2y_1			
ab	00	01	11	10
00				
01				
11				
10				

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	

Deriving Y_1

Present state y_2y_1	Next state				Output s
	$ab=00$	01	10	11	
	Y_1				
00	0	1	1	0	0
01	0	1	1	0	1
10	1	0	0	1	0
11	1	0	0	1	1

y_2y_1	ab			
	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	

Deriving Y_1

Present state y_2y_1	Next state				Output s
	$ab=00$	01	10	11	
	Y_1				
00	0	1	1	0	0
01	0	1	1	0	1
10	1	0	0	1	0
11	1	0	0	1	1

$a b$		y_2y_1			
		00	01	11	10
00	00	0	0	1	1
	01	1	1	0	0
	11	0	0	1	1
	10	1	1	0	0

$$Y_1 = a \oplus b \oplus y_2$$

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	

Deriving Y_2

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2 <input type="checkbox"/>				
00	0 <input type="checkbox"/>	0 <input type="checkbox"/>	0 <input type="checkbox"/>	1 <input type="checkbox"/>	0
01	0 <input type="checkbox"/>	0 <input type="checkbox"/>	0 <input type="checkbox"/>	1 <input type="checkbox"/>	1
10	0 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	0
11	0 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	1

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	1	
0	0	1	1	0	
0	1	0	0	0	
0	1	0	1	1	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	1	0	0	
1	0	1	1	1	
1	1	0	0	1	
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	

Deriving Y_2

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2 <input type="checkbox"/>				
00	0 <input type="checkbox"/>	0 <input type="checkbox"/>	0 <input type="checkbox"/>	1 <input type="checkbox"/>	0
01	0 <input type="checkbox"/>	0 <input type="checkbox"/>	0 <input type="checkbox"/>	1 <input type="checkbox"/>	1
10	0 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	0
11	0 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	1 <input type="checkbox"/>	1

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Deriving Y_2

Present state y_2y_1	Next state				Output s
	$ab=00$	01	10	11	
	Y_2				
00	0	0	0	1	0
01	0	0	0	1	1
10	0	1	1	1	0
11	0	1	1	1	1

		y_2y_1			
	ab	00	01	11	10
00					
01					
11					
10					

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Deriving Y_2

Present state y_2y_1	Next state				Output s
	$ab=00$	01	10	11	
	Y_2				
00	0	0	0	1	0
01	0	0	0	1	1
10	0	1	1	1	0
11	0	1	1	1	1

		y_2y_1			
		00	01	11	10
ab	00	0	0	0	0
	01	0	0	1	1
	11	1	1	1	1
	10	0	0	1	1

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Deriving Y_2

Present state y_2y_1	Next state				Output s
	$ab=00$	01	10	11	
	Y_2 				
00	0 	0 	0 	1 	0
01	0	0	0	1	1
10	0	1	1	1	0
11	0	1	1	1	1

		y_2y_1			
		00	01	11	10
ab	00	0	0	0	0
	01	0	0	1	1
	11	1	1	1	1
	10	0	0	1	1

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Deriving Y_2

Present state y_2y_1	Next state				Output s
	$ab=00$	01	10	11	
	Y_2				
00	0	0	0	1	0
01	0	0	0	1	1
10	0	1	1	1	0
11	0	1	1	1	1

		y_2y_1			
		00	01	11	10
ab	00	0	0	0	0
	01	0	0	1	1
	11	1	1	1	1
	10	0	0	1	1

$$Y_2 = ab + ay_2 + by_2$$

y_2	y_1	a	b	Y_1	Y_2
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Deriving s

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	0 0	01	0 1	10	0
01	0 0	01	0 1	10	1
10	0 1	10	1 0	11	0
11	0 1	10	1 0	11	1

y_2	y_1	s
0	0	
0	1	
1	0	
1	1	

Deriving s

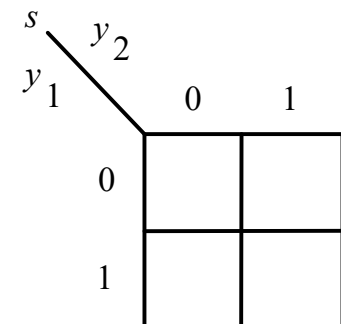
Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	0 0	01	0 1	10	0
01	0 0	01	0 1	10	1
10	0 1	10	1 0	11	0
11	0 1	10	1 0	11	1

y_2	y_1	s
0	0	0
0	1	1
1	0	0
1	1	1

Deriving s

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

y_2	y_1	s
0	0	0
0	1	1
1	0	0
1	1	1



Deriving s

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

y_2	y_1	s
0	0	0
0	1	1
1	0	0
1	1	1

		y_2	
		0	1
y_1	0	0	0
	1	1	1

The bottom-right cell (1,1) containing '1' is circled in red.

Deriving s

Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

y_2	y_1	s
0	0	0
0	1	1
1	0	0
1	1	1

		y_2	
		0	1
y_1	0	0	0
	1	1	1

$$s = y_1$$

State-assigned table for the Moore-type serial adder FSM

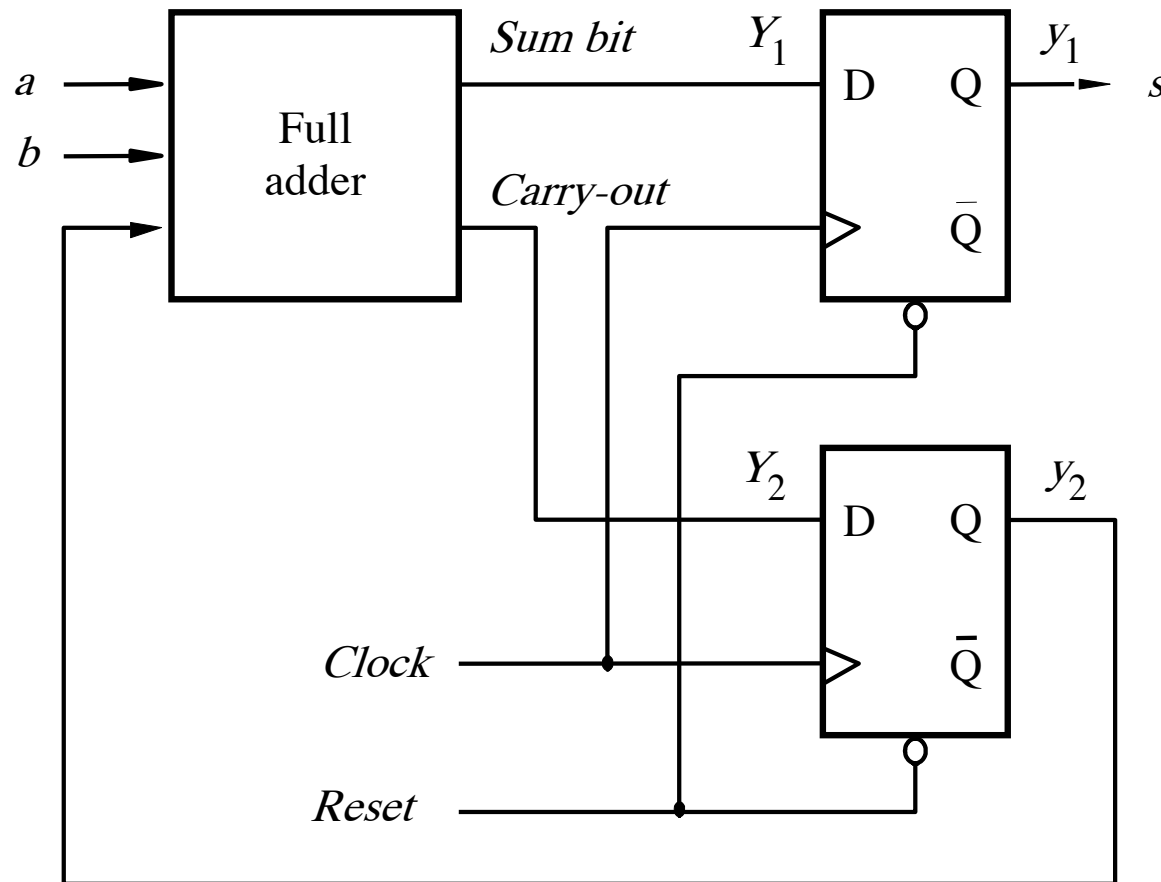
Present state y_2y_1	Next state				Output s
	$ab = 00$	01	10	11	
	Y_2Y_1				
00	00	01	01	10	0
01	00	01	01	10	1
10	01	10	10	11	0
11	01	10	10	11	1

$$Y_1 = a \oplus b \oplus y_2$$

$$Y_2 = ab + ay_2 + by_2$$

$$s = y_1$$

Circuit for the Moore-type serial adder FSM

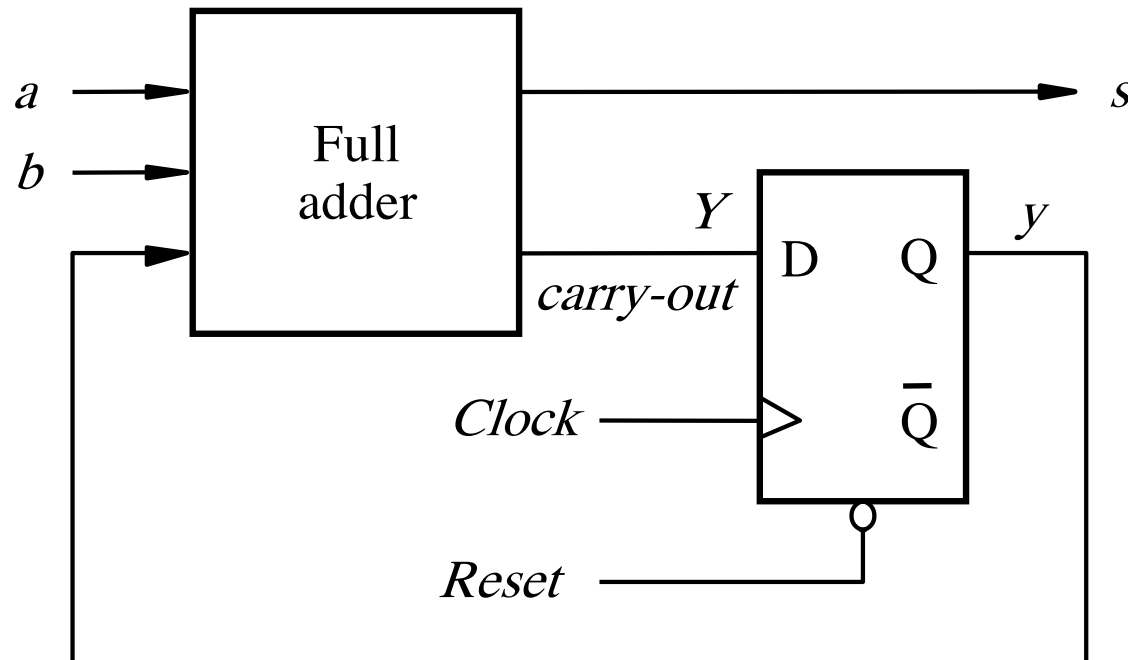


$$Y_1 = a \oplus b \oplus y_2 \quad (\text{sum bit from FA})$$

$$Y_2 = ab + ay_2 + by_2 \quad (\text{carry bit from FA})$$

$$s = y_1$$

Circuit for the Mealy-type serial adder FSM



$$s = \text{XOR}(\text{XOR}(a, b), y) \quad (\text{sum bit from FA})$$

$$Y = ab + ay + by \quad (\text{carry bit from FA})$$

Arbiter Circuit

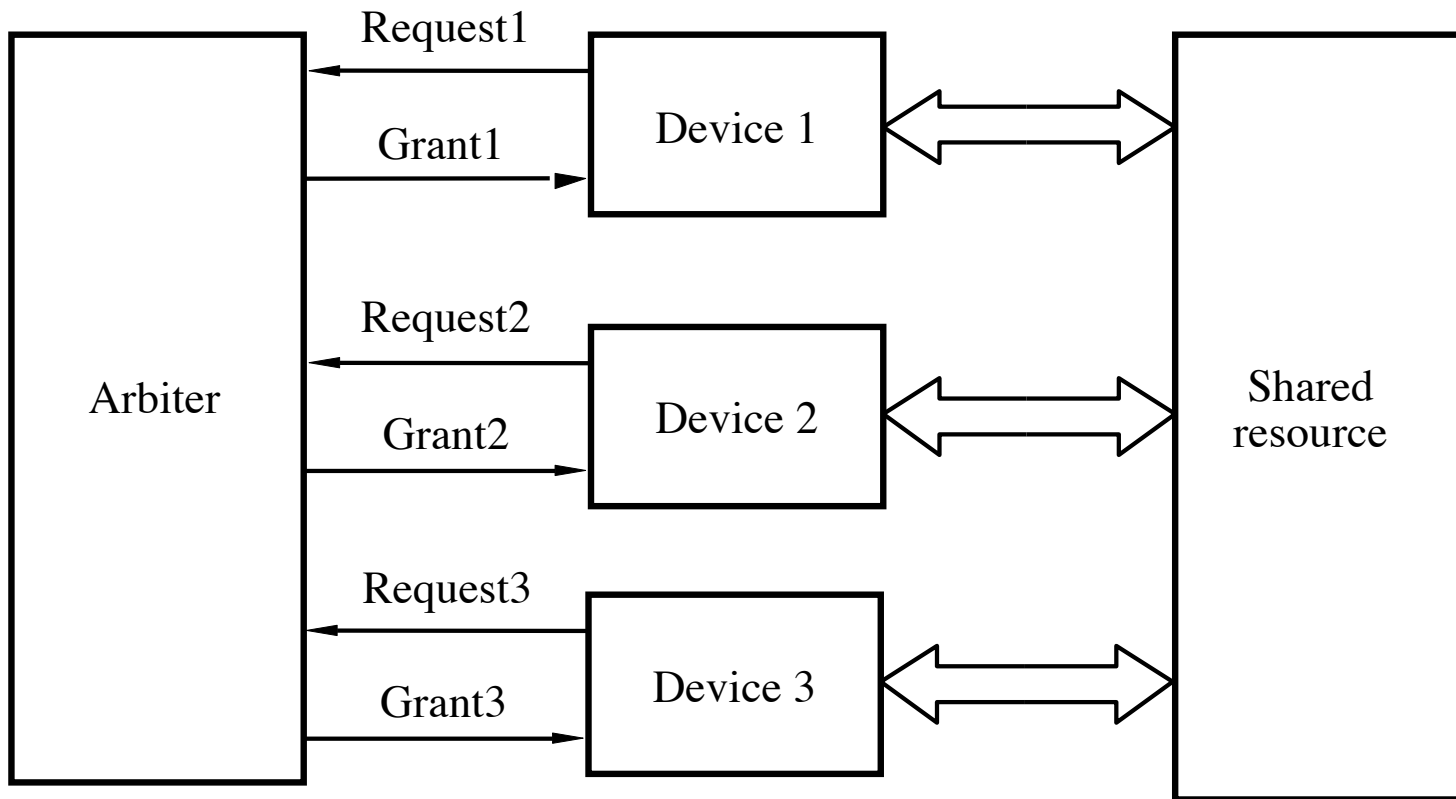
Goal

- **Design a machine that controls access by several devices to a shared resource.**
- **The resource can be used by only one device at a time.**
- **Any changes can occur only on the positive edge of the clock signal.**
- **Each device provides one input to the FSM, which is called a request.**
- **The FSM produces one output for each device, which is called a grant.**

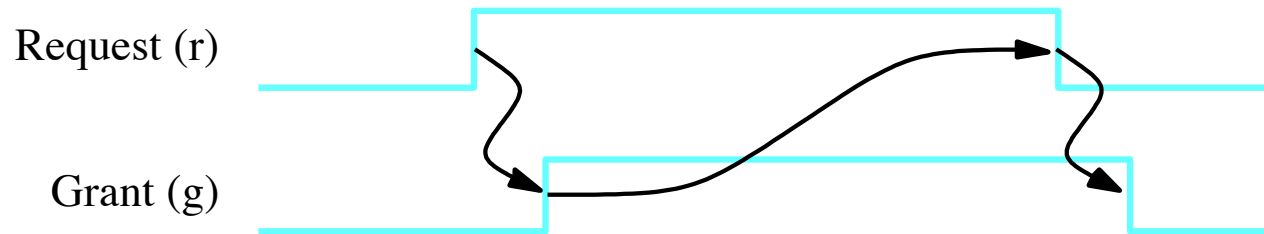
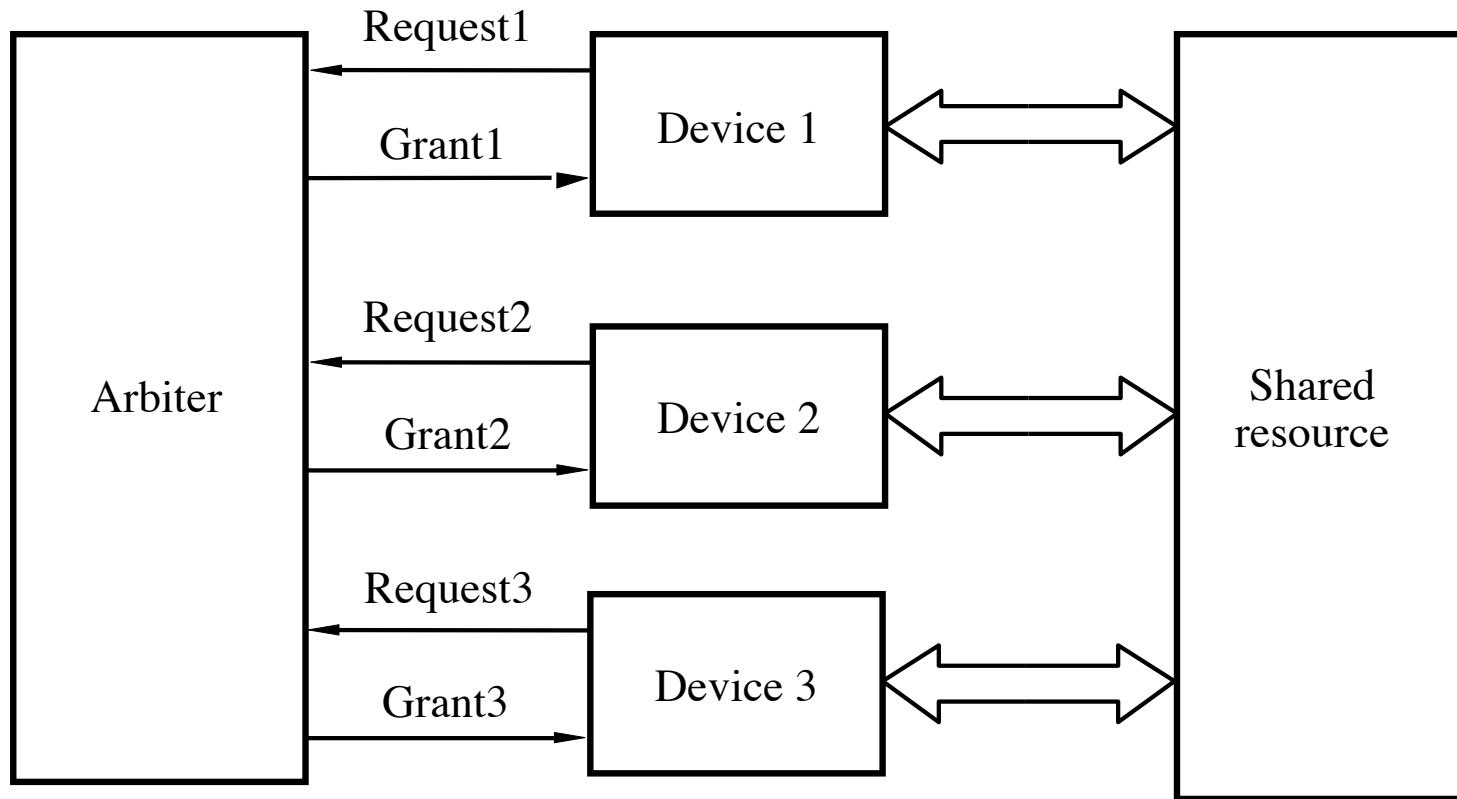
Goal

- **The requests from the devices are prioritized.**
- **If two requests are active at the same time, then only the device with the highest priority will be given access to the shared resource.**
- **After a device is done with the shared resource, it must make its request signal equal to 0.**
- **If there are no outstanding requests, then the FSM stays in an Idle state.**

Conceptual Diagram

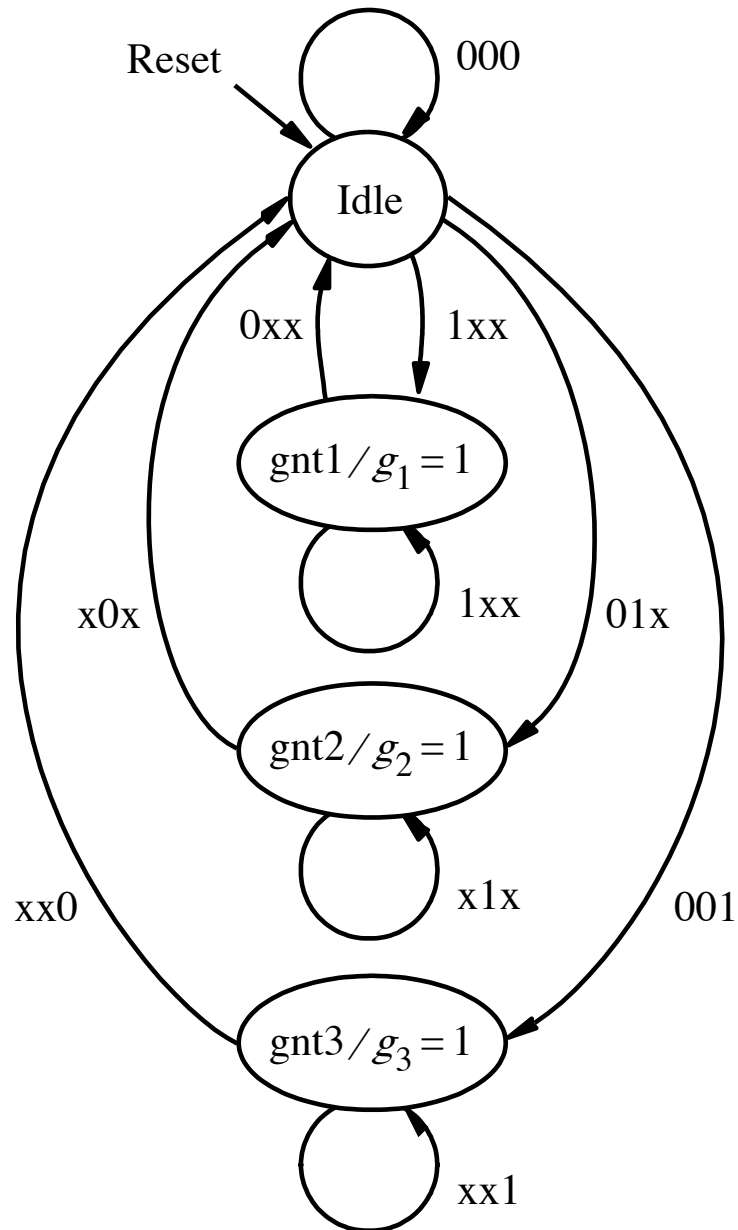


Conceptual Diagram



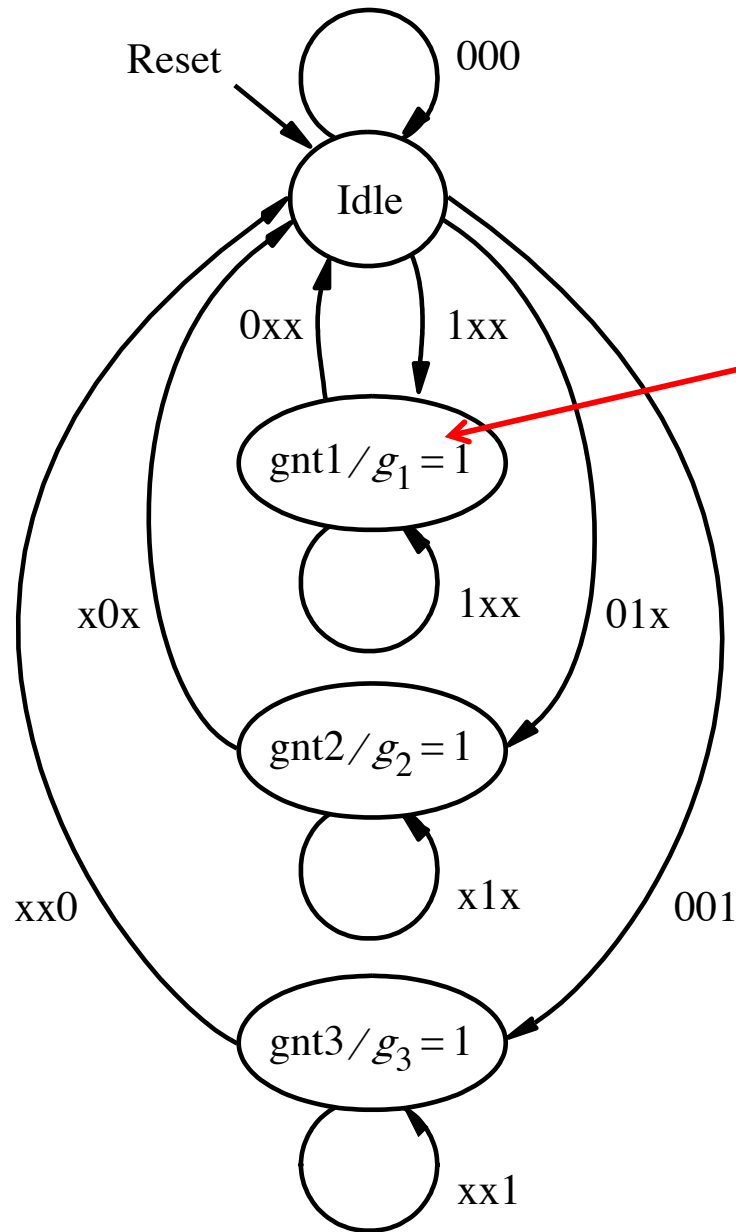
[Figure 9.20 from the textbook]

State diagram for the arbiter



[Figure 6.72 from the textbook]

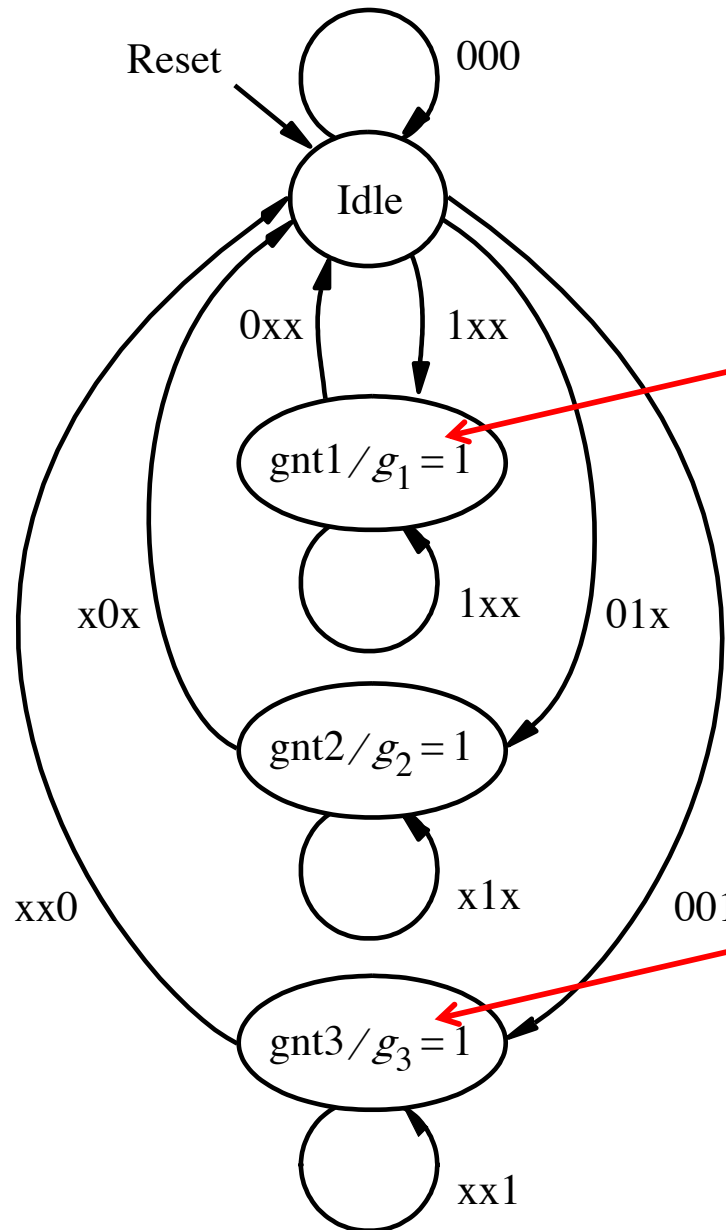
State diagram for the arbiter



Highest Priority Device

xx1

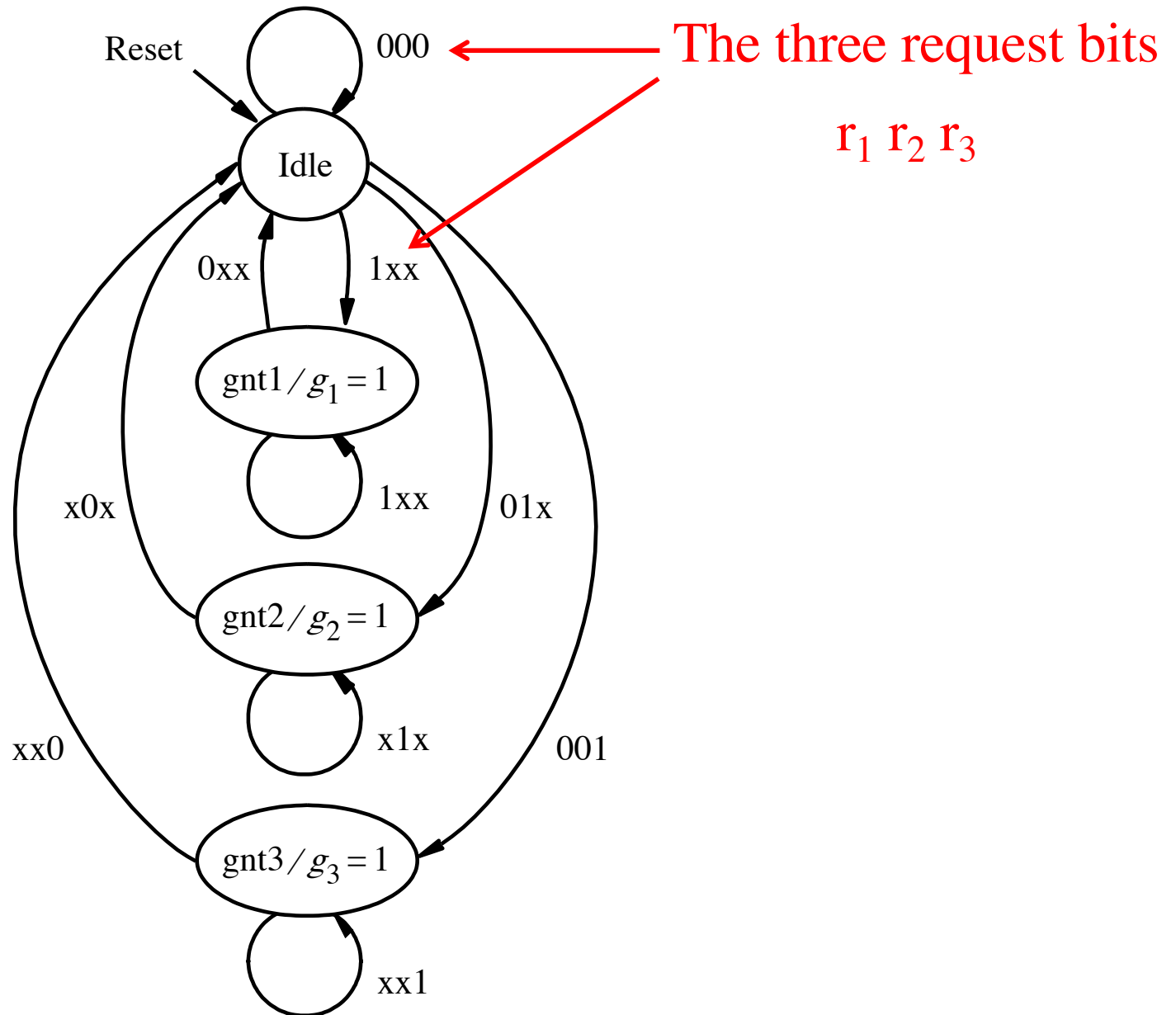
State diagram for the arbiter



Highest Priority Device

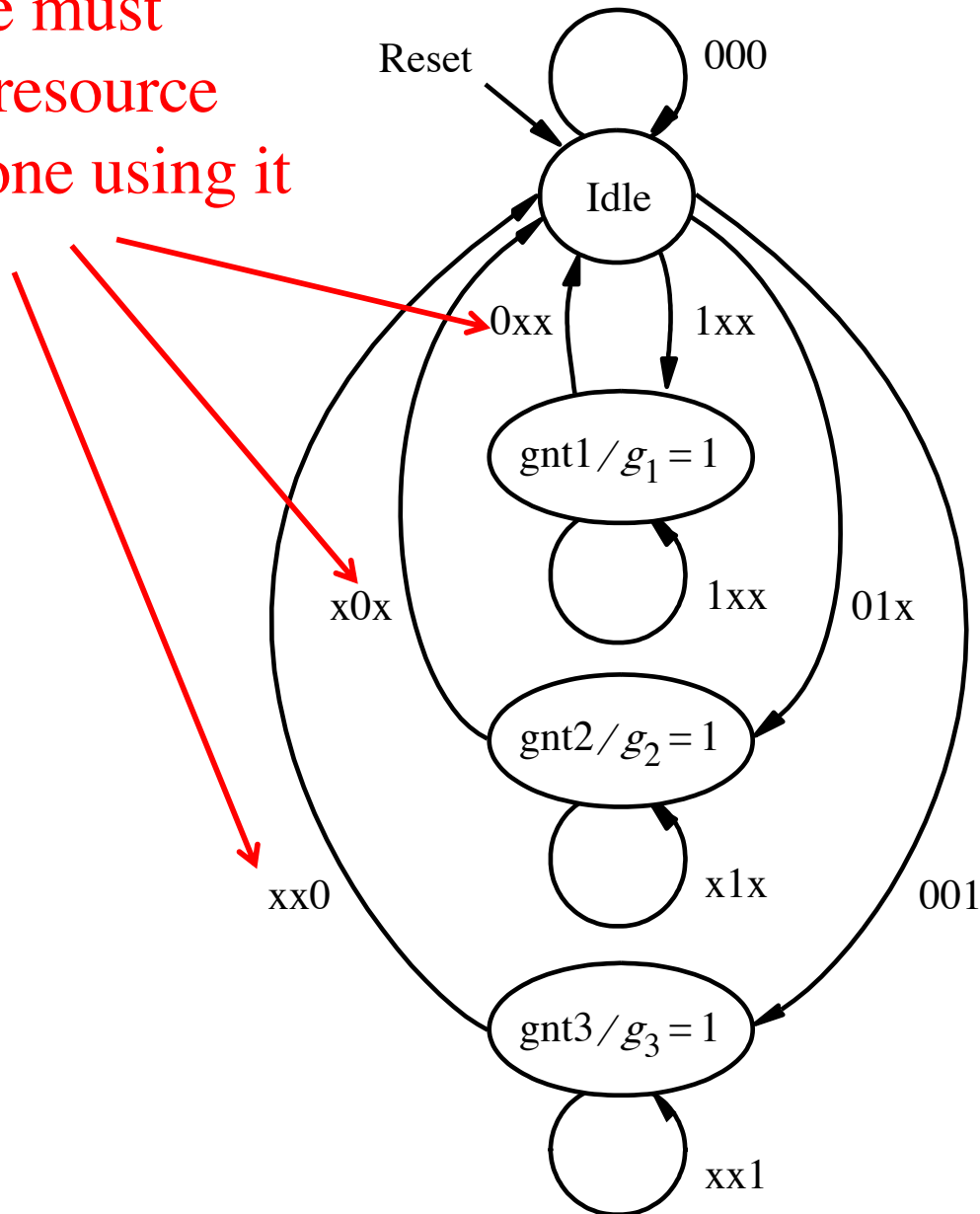
Lowest Priority Device

State diagram for the arbiter

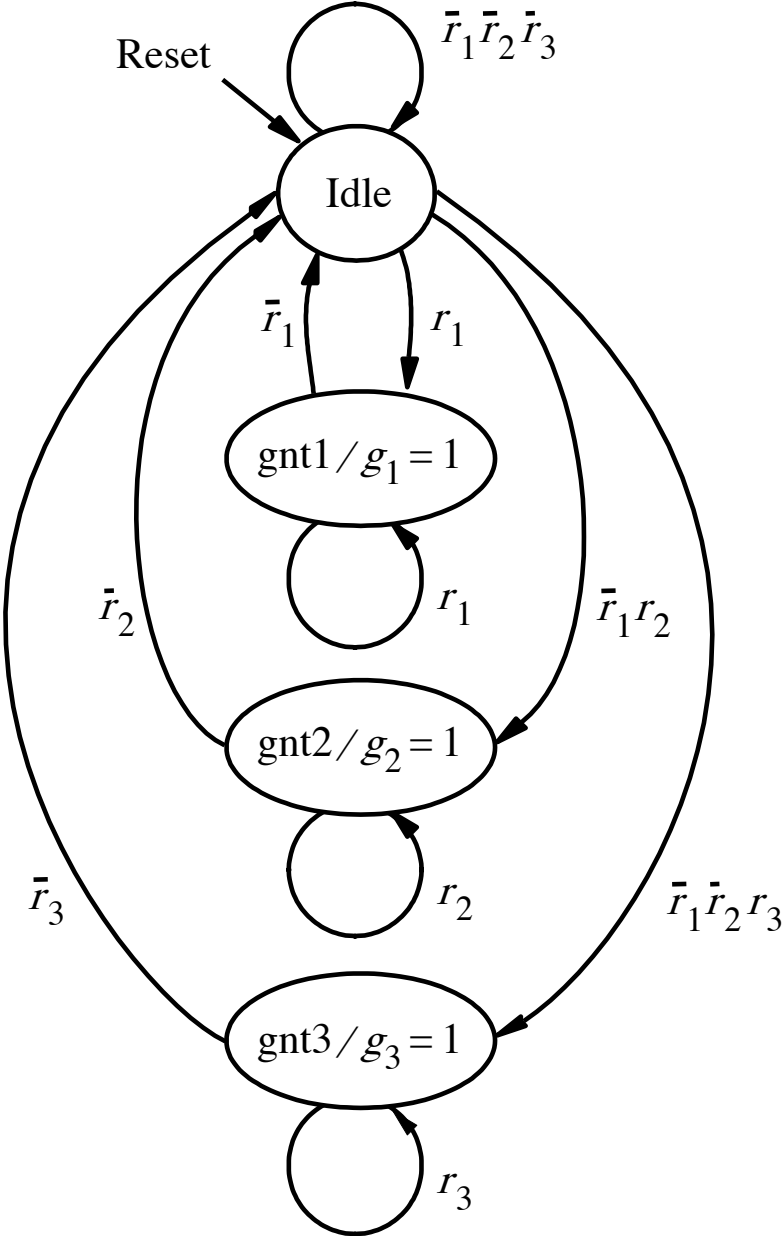


State diagram for the arbiter

Each device must release the resource after it is done using it



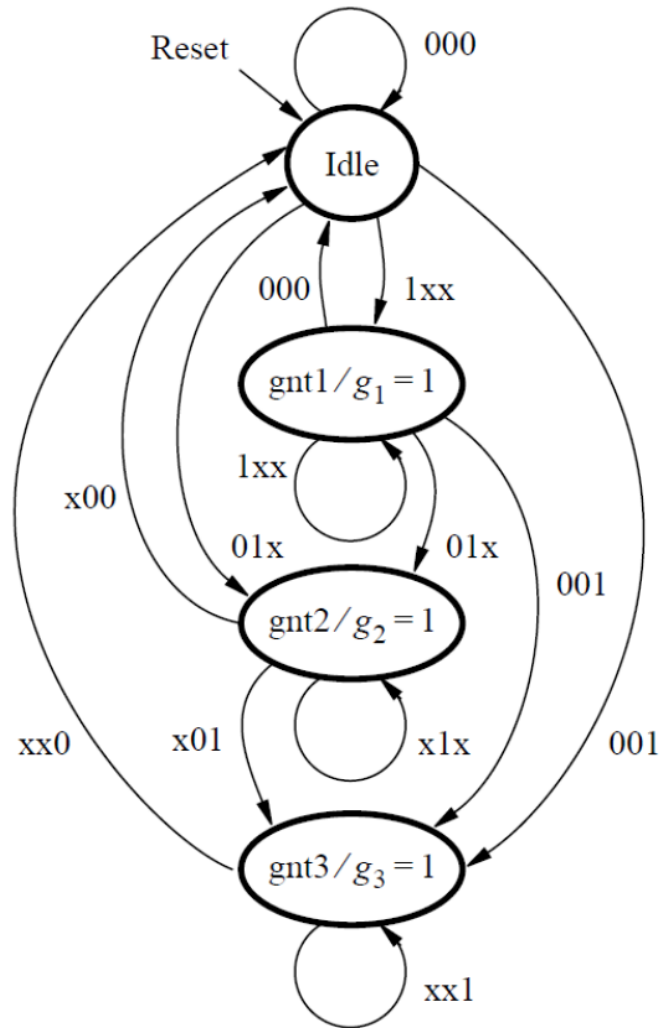
Alternative style of state diagram for the arbiter



[Figure 6.73 from the textbook]

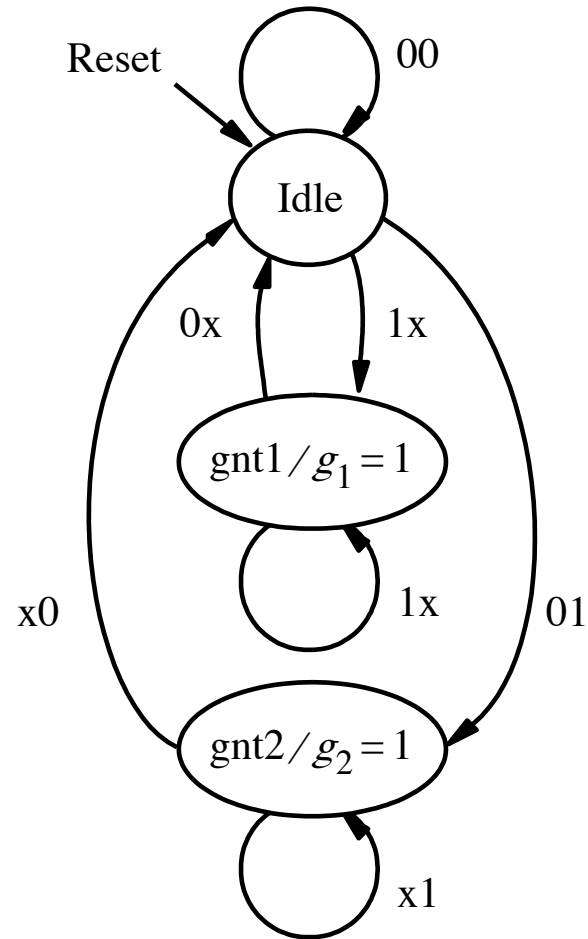
**This design has one flaw:
If device1 and device2 raise requests all the
time, then device3 will never get serviced.**

This state diagram solves this problem

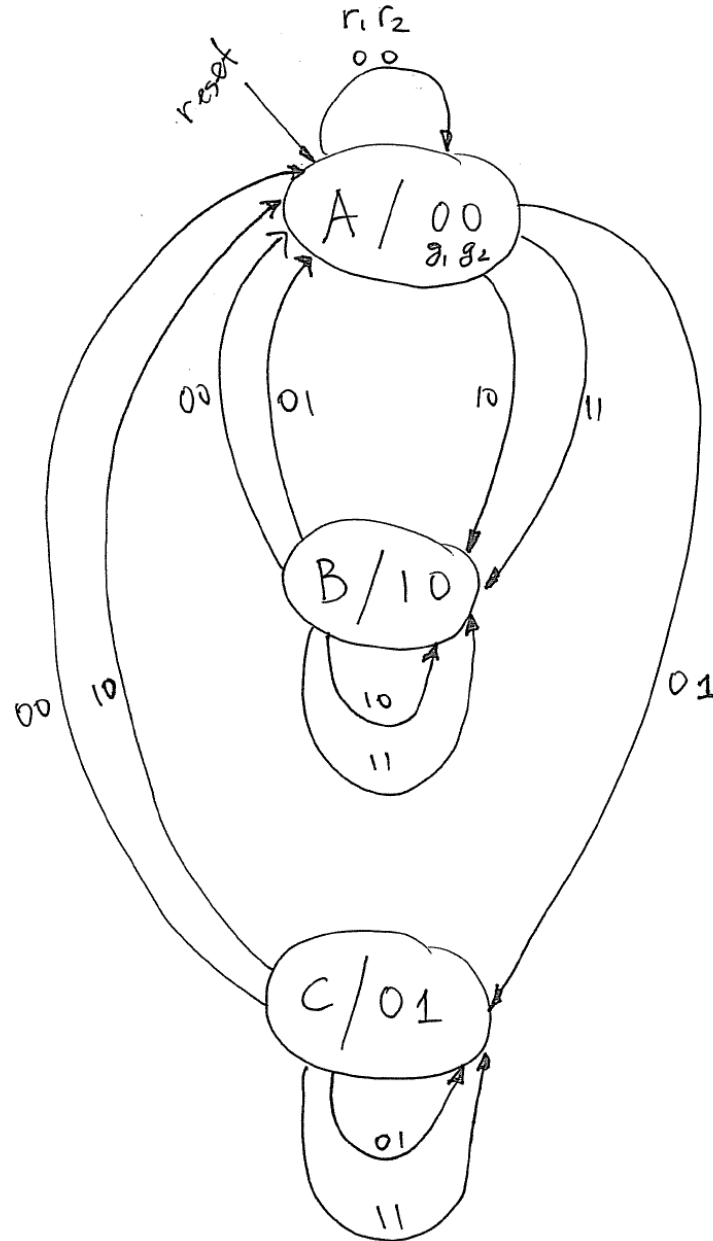


**Let's look at a simpler example with
only two devices that need to use
the shared resource**

State diagram for the simpler arbiter



State diagram for the arbiter circuit



State Table

	$r_1 r_2 = 00$	01	10	11	Output
A	A	C	B	B	00
B	A	A	B	B	10
C	A	C	A	C	01

State-Assigned Table

	Present state $y_2 y_1$	Next state				Output $g_1 g_2$
		$r_1 r_2 = 00$	01	10	11	
		$Y_2 Y_1$				
A	00	00	10	01	01	00
B	01	00	00	01	01	10
C	10	00	10	00	10	01
	11	dd	dd	dd	dd	dd

Output Expressions

Output expressions

$$g_1 = \gamma_1$$

$$g_2 = \gamma_2$$

Next State Expressions

Y_2

$r_1 r_2$		$Y_2 Y_1$			
		00	01	11	10
.	00	0	0	d	0
	01	1	0	d	1
	11	0	0	d	1
	10	0	0	d	0

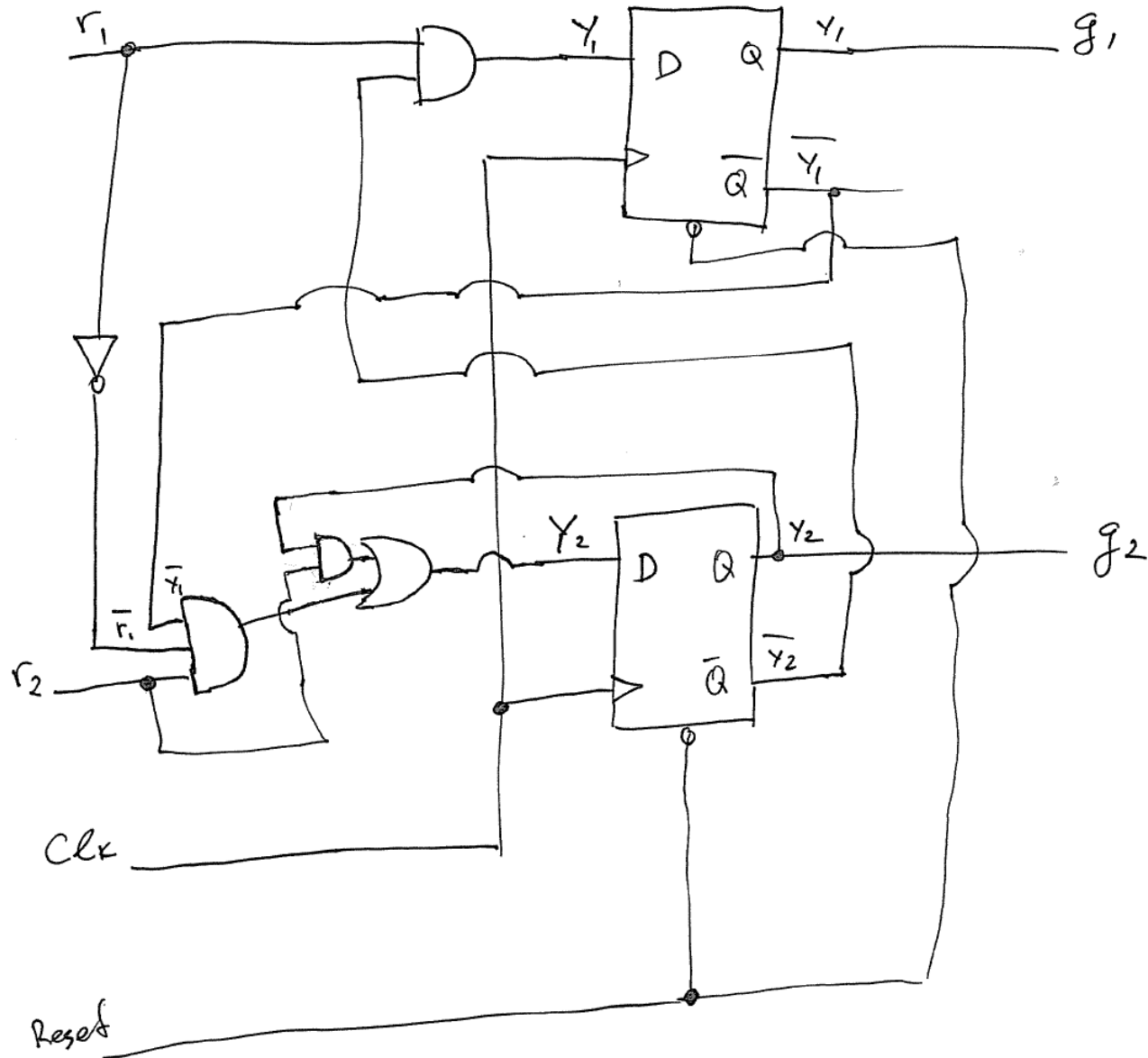
$$Y_2 = r_2 Y_2 + \bar{r}_1 r_2 \bar{Y}_1$$

Y_1

$r_1 r_2$		$Y_2 Y_1$			
		00	01	11	10
.	00	0	0	d	0
	01	0	0	d	0
	11	1	1	d	0
	10	1	1	d	0

$$Y_1 = r_1 \bar{Y}_2$$

Circuit Diagram



Questions?

THE END