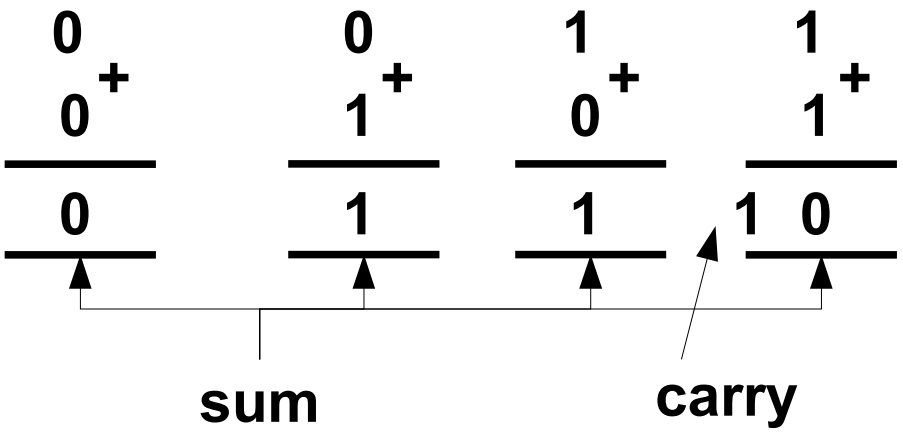
M3 – ALU Design

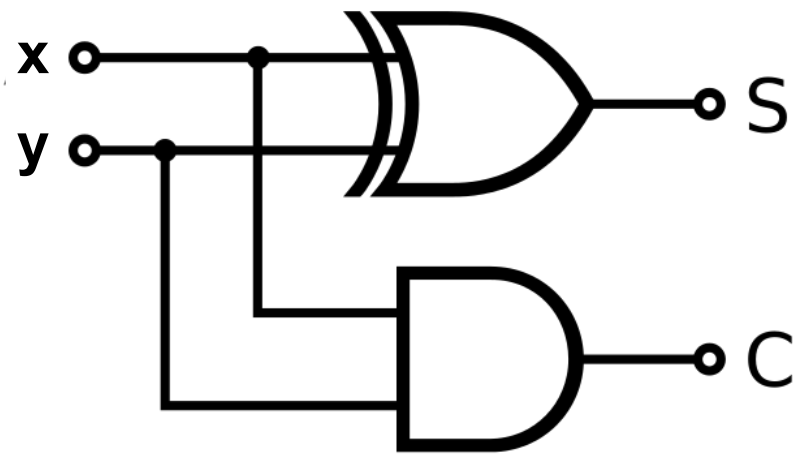
Module Outline

- Integer Arithmetic
  - Adder, Subtractor, Multiplier, Divider
- Arithmetic and Logical Unit Design
  - ALU Design in SystemC

Half Adder

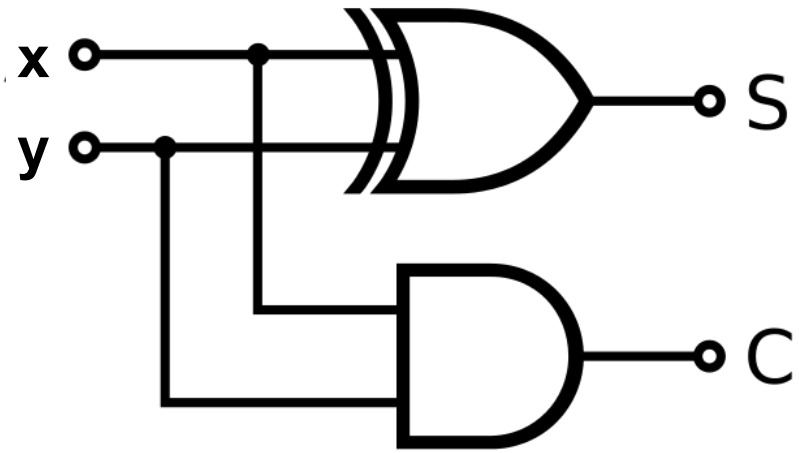


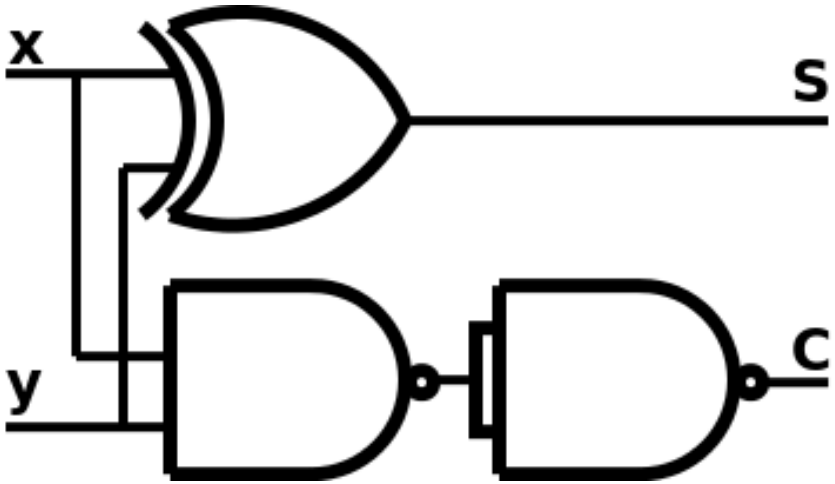
Inputs		Outputs	
x	y	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



$$S = A \oplus B$$
$$C = AB$$

Half Adder





Full Adder

```
  1
  0 +
  1
 ----
1 0
 ----

  0
  0 +
  1
 ----
0 1
 ----

  1
  1 +
  1
 ----
1 1
 ----
```

A	B	C_{in}	$A+B+C_{in}$	S	C_{out}
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	2	0	1
1	0	0	1	1	0
1	0	1	2	0	1
1	1	0	2	0	1
1	1	1	3	1	1

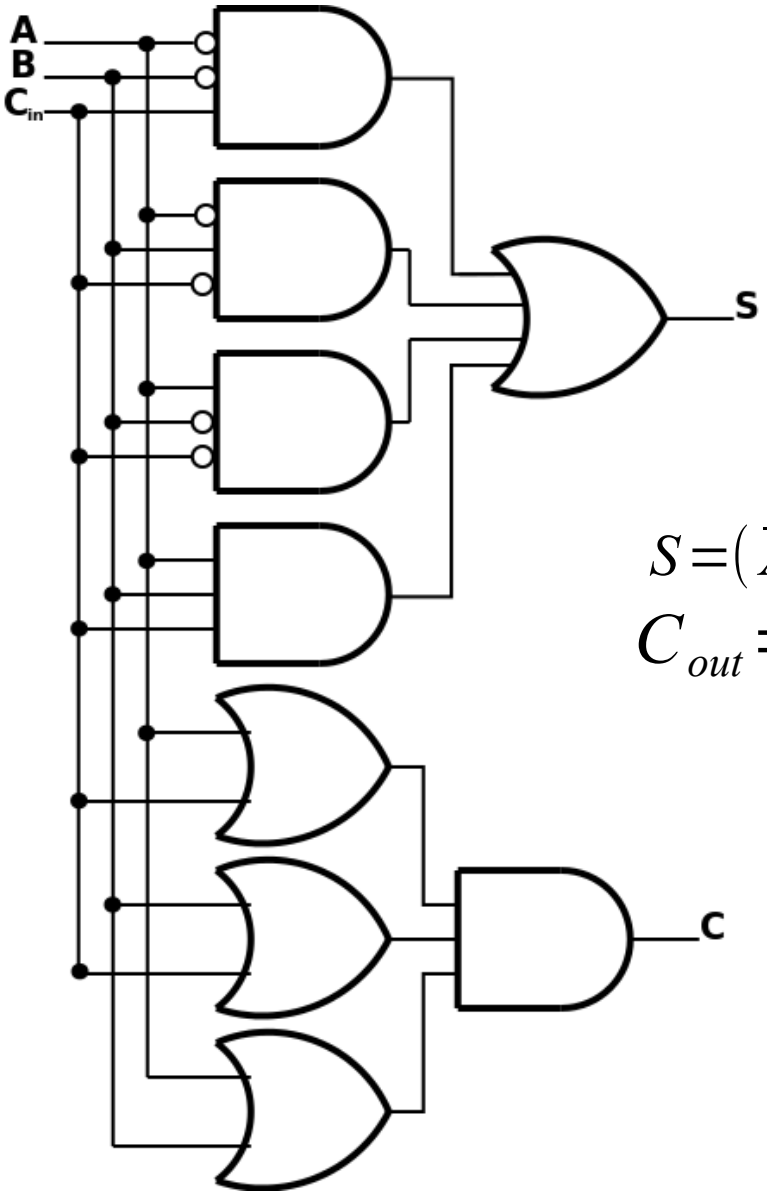
Full Adder

A	B	C_{in}	$A+B+C_{in}$	S	C_{out}
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	1	0
0	1	1	2	0	1
1	0	0	1	1	0
1	0	1	2	0	1
1	1	0	2	0	1
1	1	1	3	1	1

$$S = (\bar{A} \cdot \bar{B} \cdot C_i) + (\bar{A} \cdot B \cdot \bar{C}_i) + (A \cdot \bar{B} \cdot \bar{C}_i) + (A \cdot B \cdot C_i)$$

$$C_{out} = (A + C_i) \cdot (B + C_i) \cdot (A + B)$$

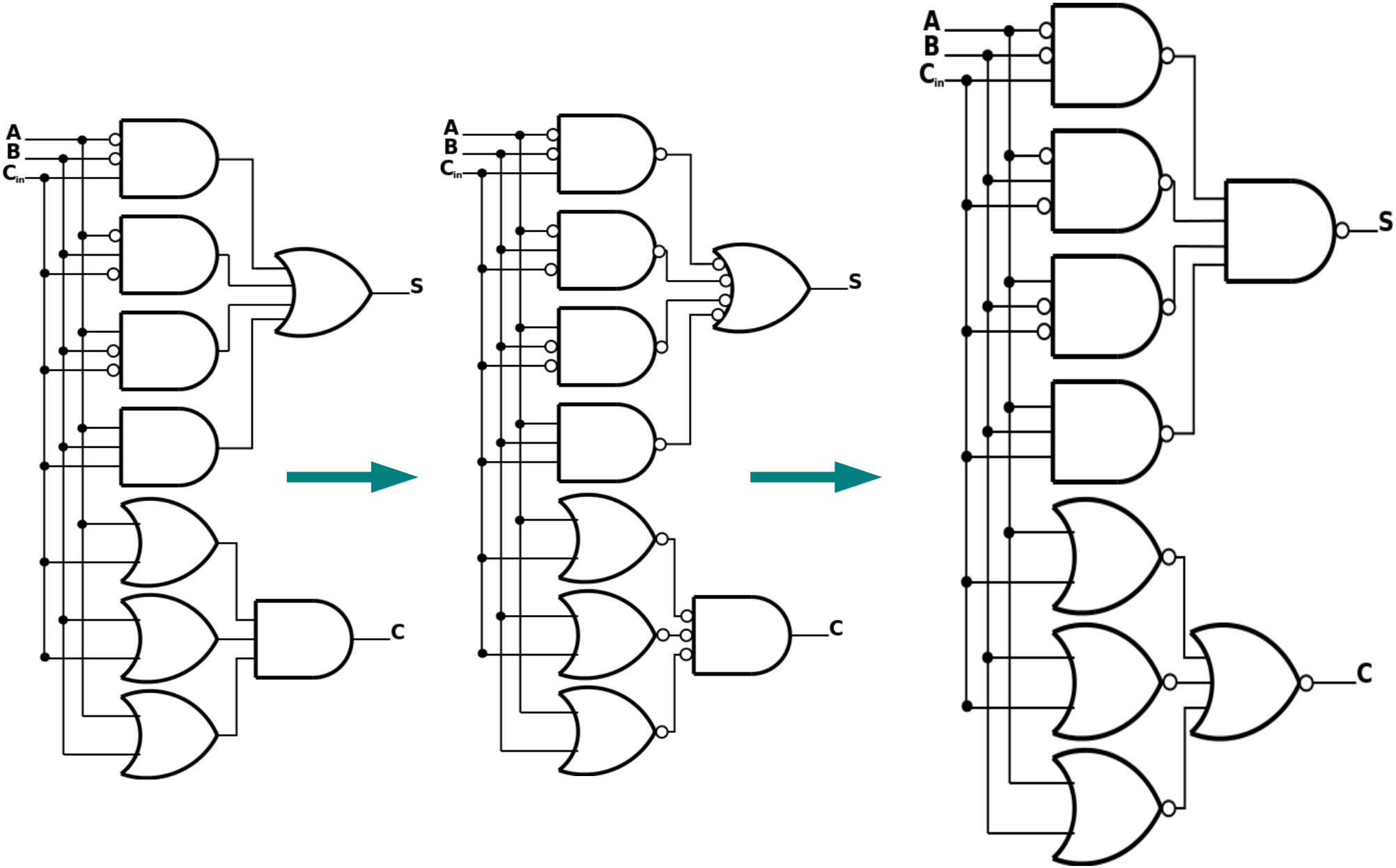
Full Adder – AND, OR, NOT



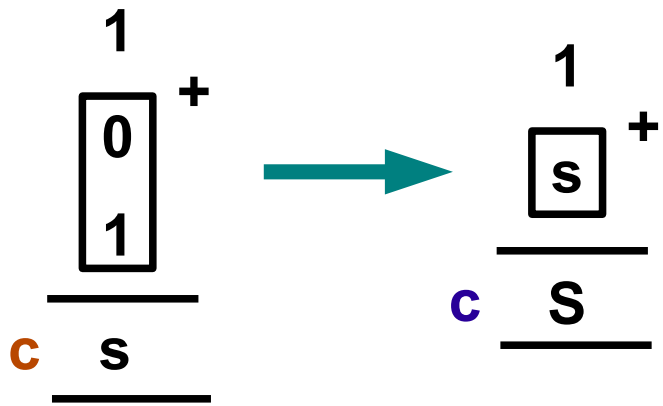
$$S = (\bar{A} \cdot \bar{B} \cdot C_i) + (\bar{A} \cdot B \cdot \bar{C}_i) + (A \cdot \bar{B} \cdot \bar{C}_i) + (A \cdot B \cdot C_i)$$

$$C_{out} = (A + C_i) \cdot (B + C_i) \cdot (A + B)$$

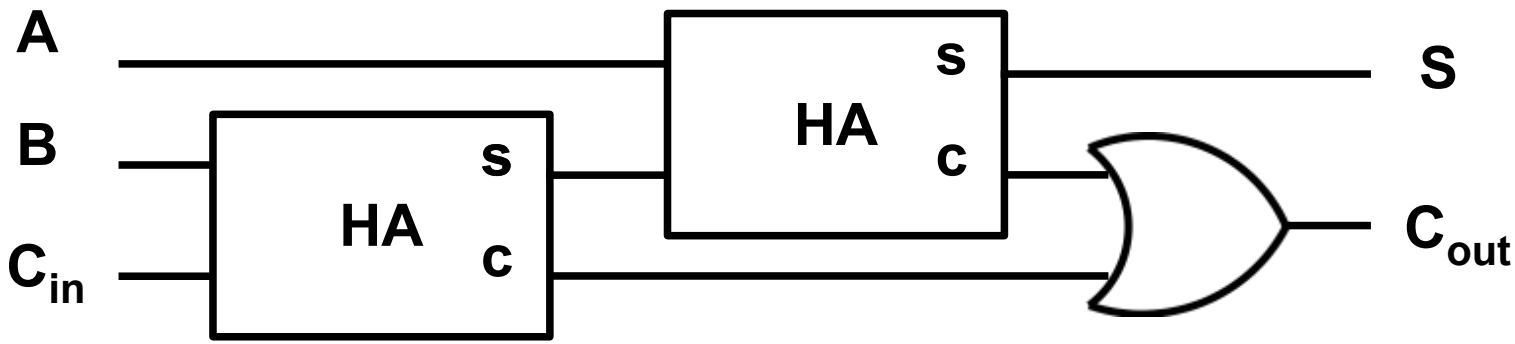
Full Adder – NAND, NOR, NOT

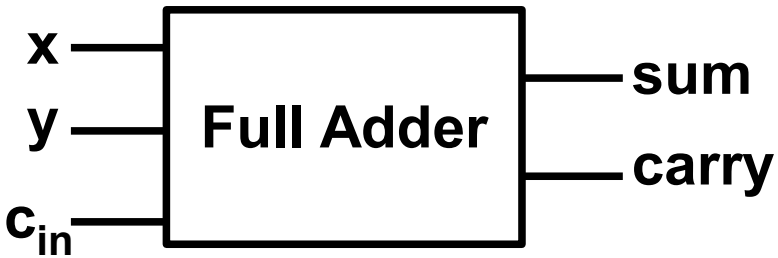


Full Adder

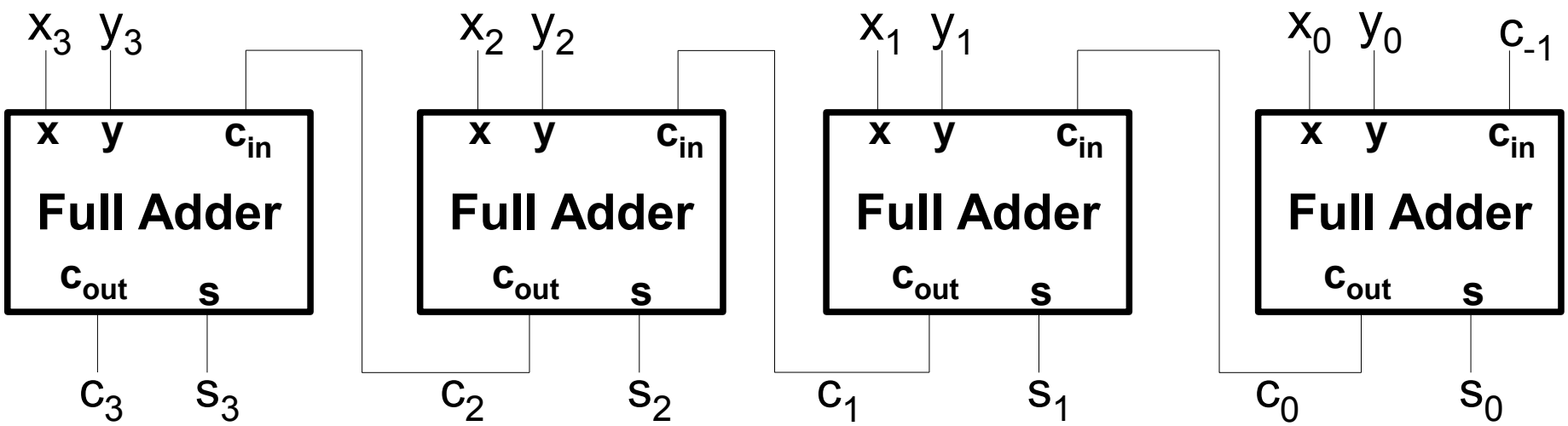


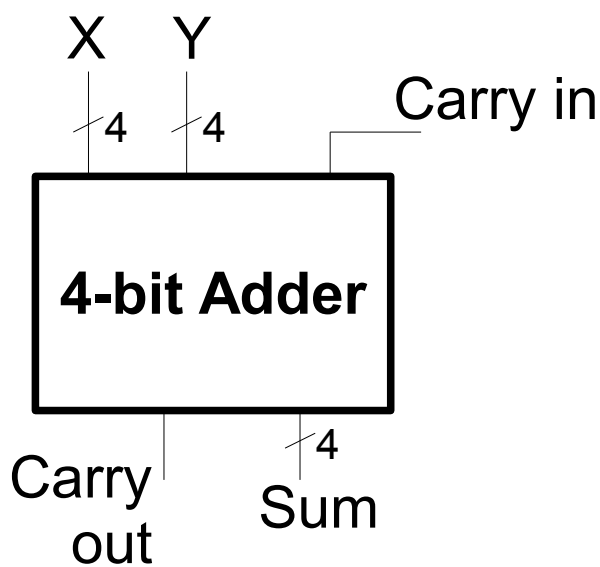
C ← c OR c



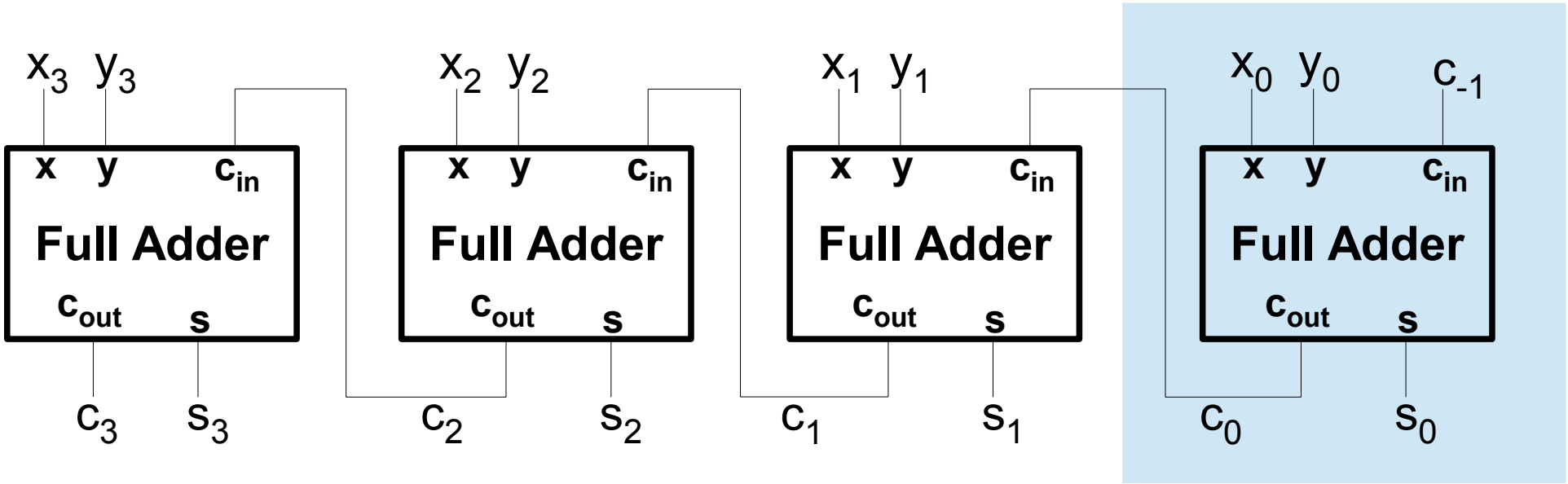


4-bit Ripple Carry Adder



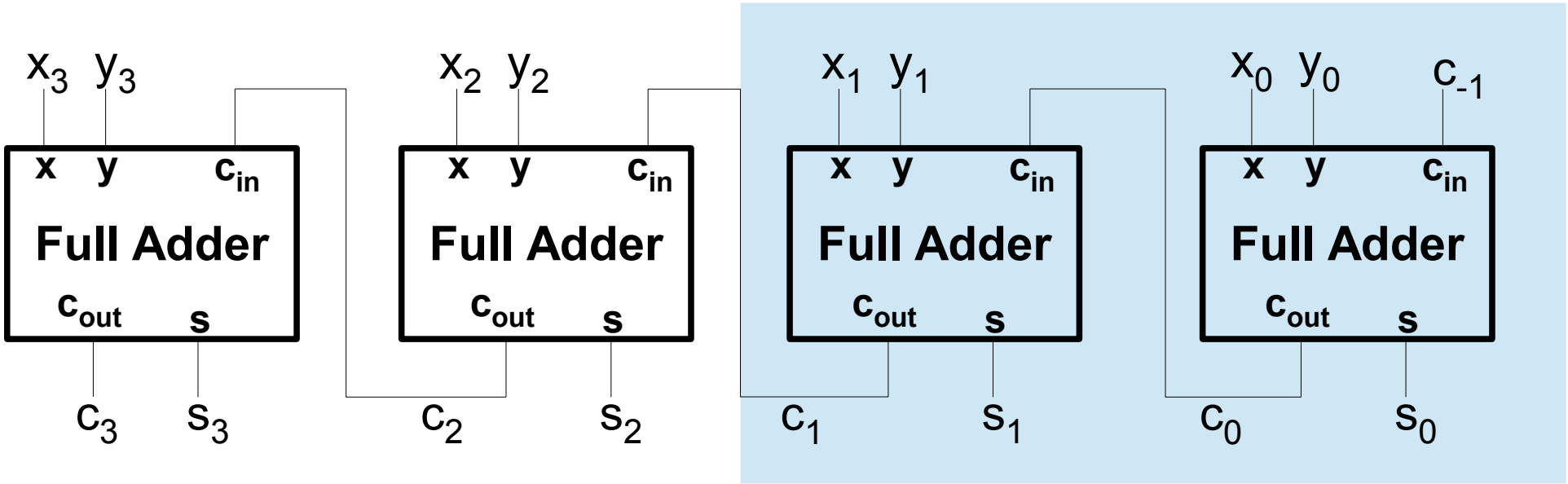


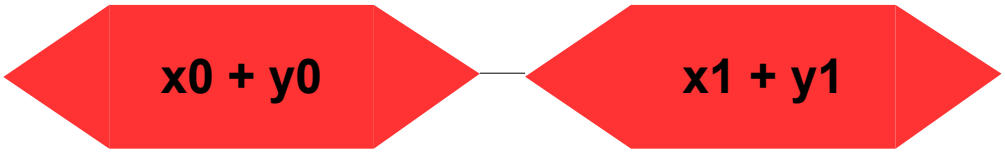
4-bit Ripple Carry Adder



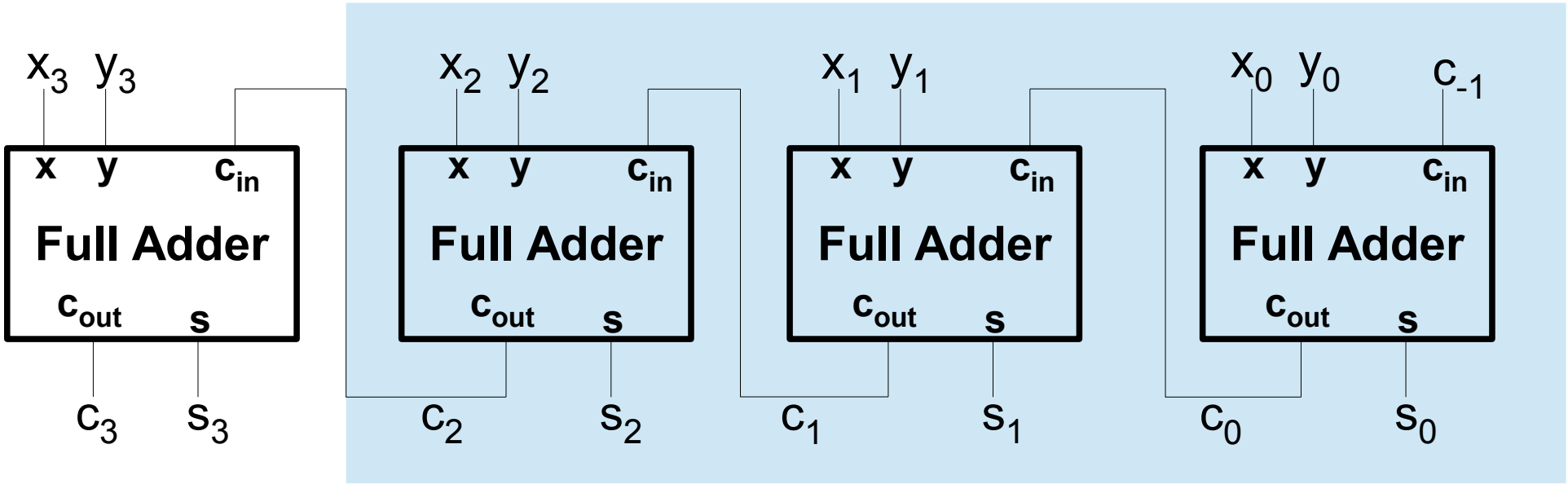
x0 + y0

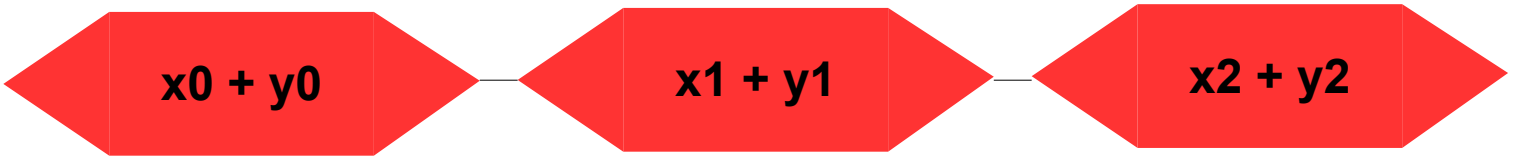
4-bit Ripple Carry Adder



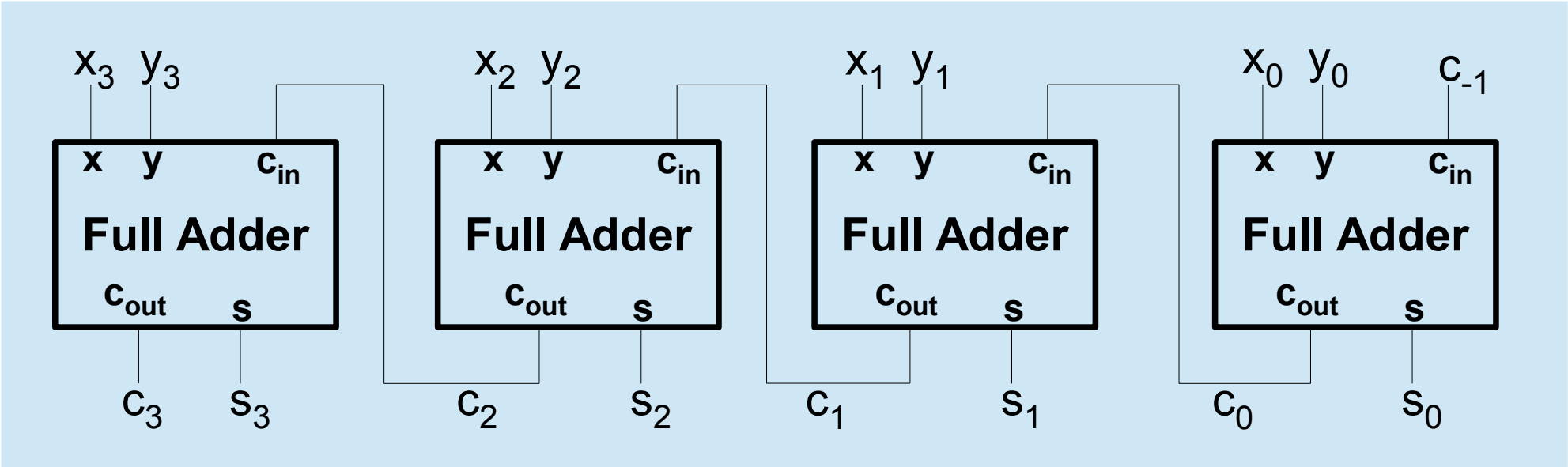


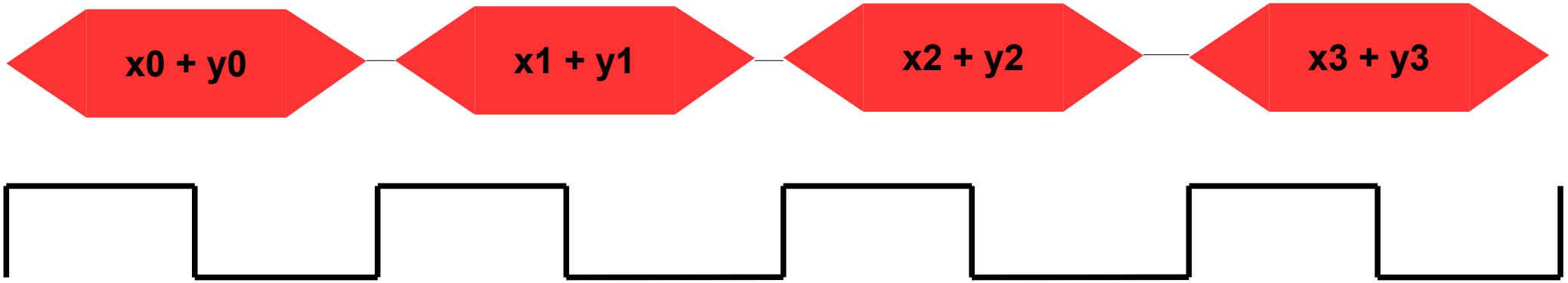
4-bit Ripple Carry Adder



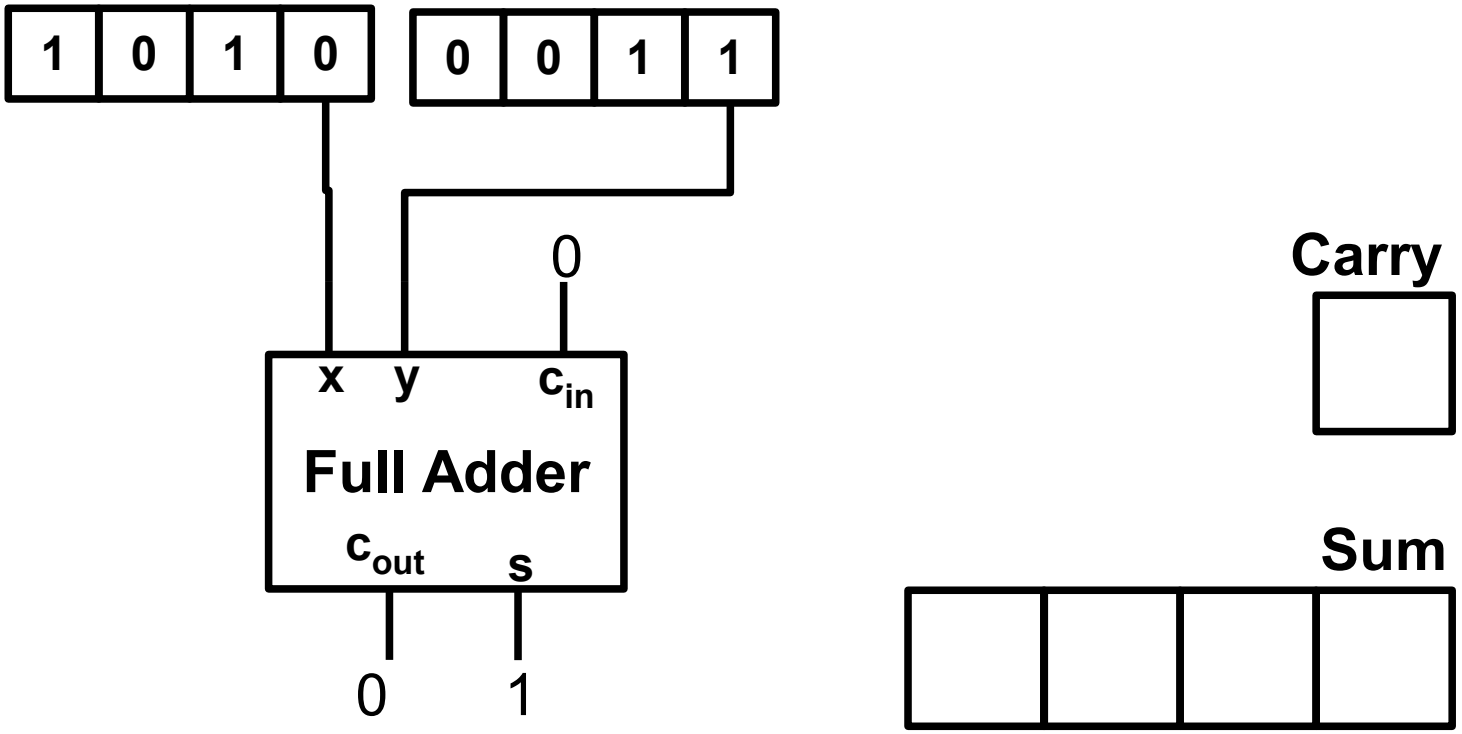


4-bit Ripple Carry Adder



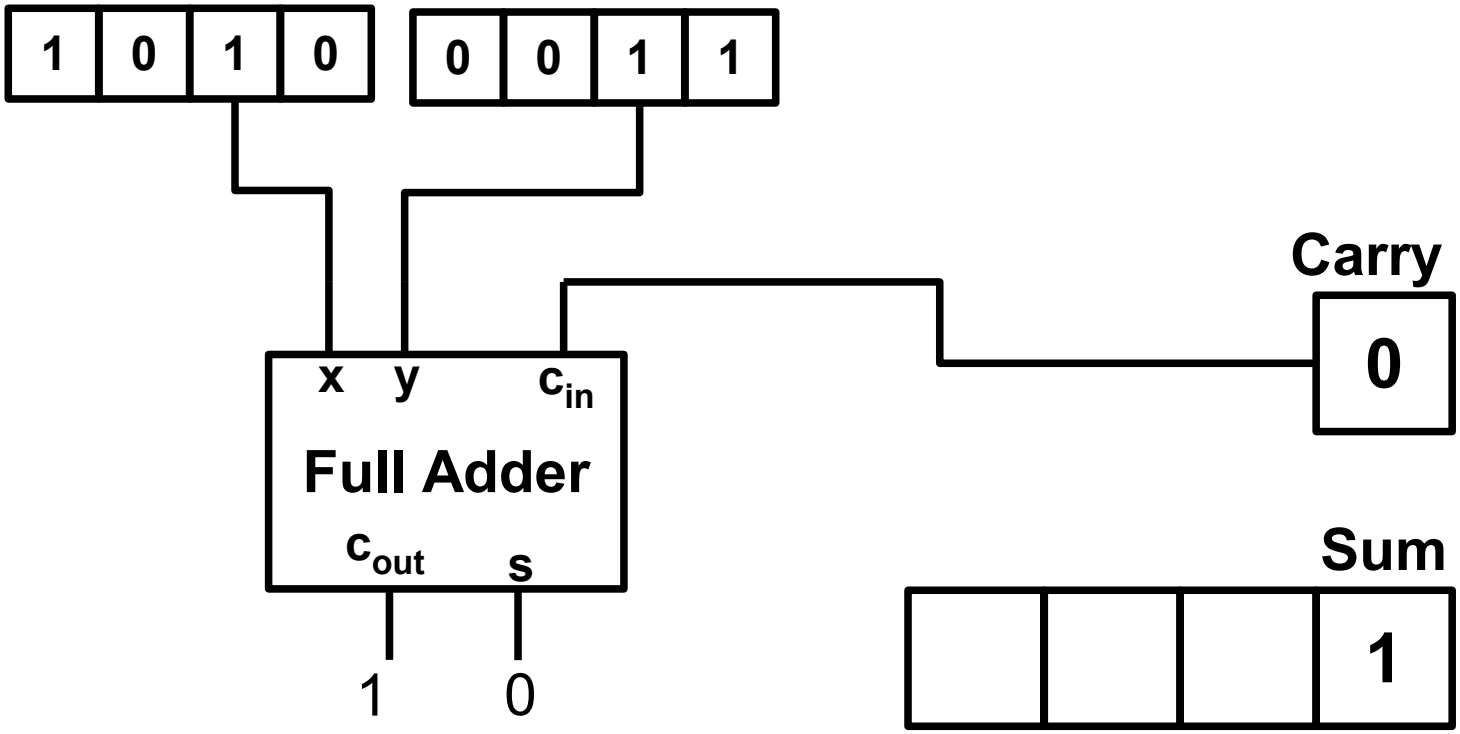


4-bit Ripple Carry Adder



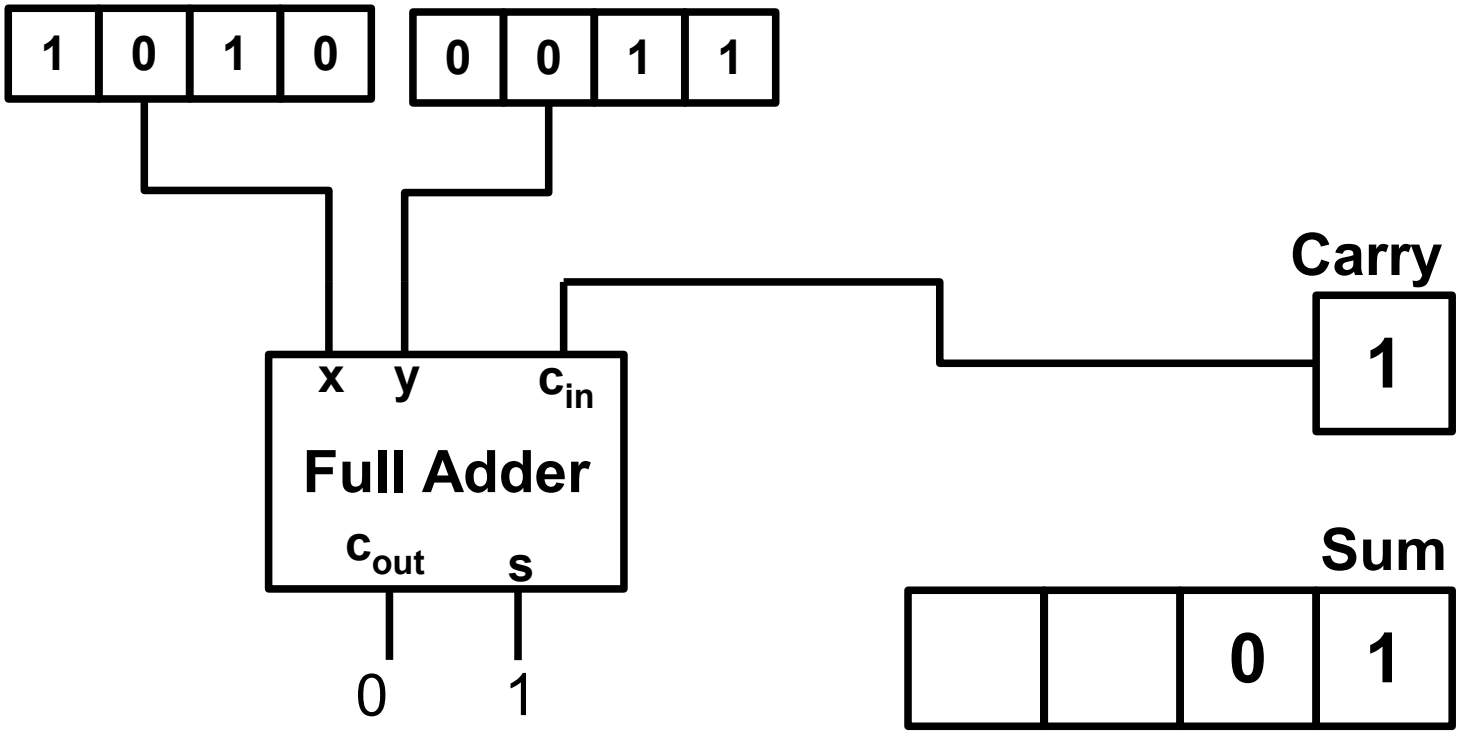
Clock Cycle: 1

4-bit Ripple Carry Adder



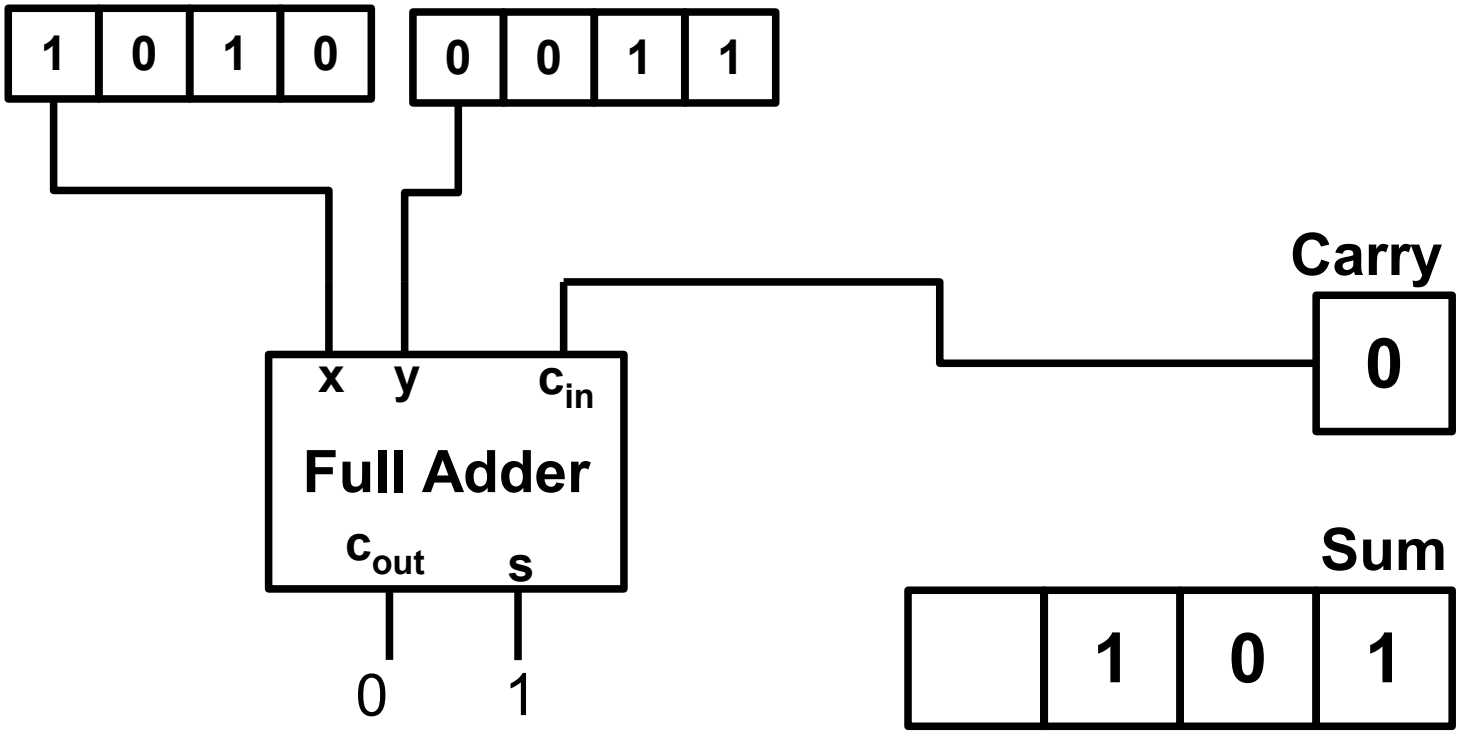
Clock Cycle: 2

4-bit Ripple Carry Adder



Clock Cycle: 3

4-bit Ripple Carry Adder

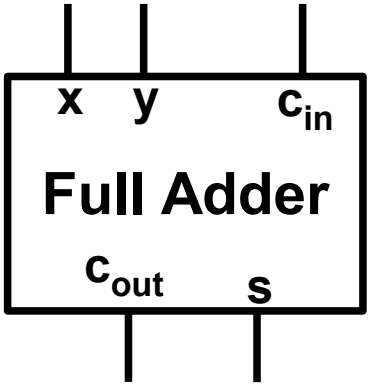


Clock Cycle: 4

4-bit Ripple Carry Adder

1	0	1	0

0	0	1	1



Carry

1

Sum

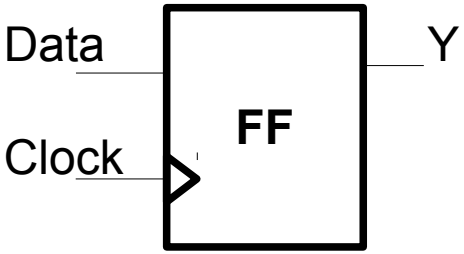
0	1	0	1

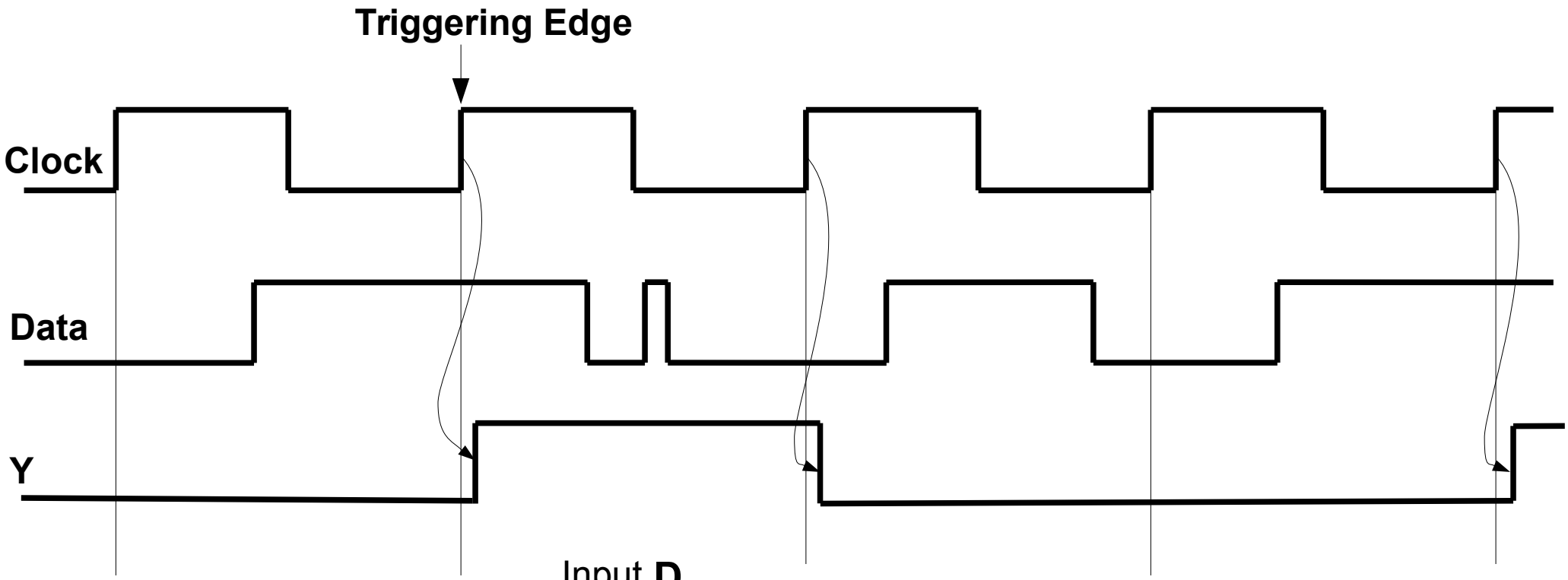
Clock Cycle: 5

4-bit Ripple Carry Adder

- Use the same Full Adder
- Insert a circuit element that can remember state
  - Flip Flop

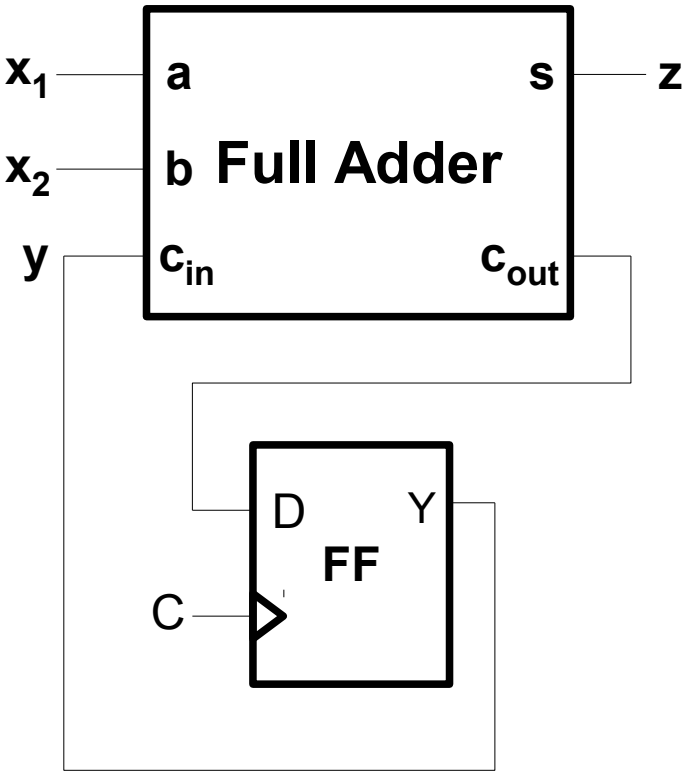
Flip Flop





		Input D		
		0	1	
State Y(i)	0	0	1	State Y(i+1)
	1	0	1	

Serial Adder



	Input: x_1x_2			
	00	**01**	**10**	**11**
S_0 (y=0)	**S_0,0**	**S_0,1**	**S_0,1**	**S_1,0**
S_1 (y=1)	**S_0,1**	**S_1,0**	**S_1,0**	**S_1,1**

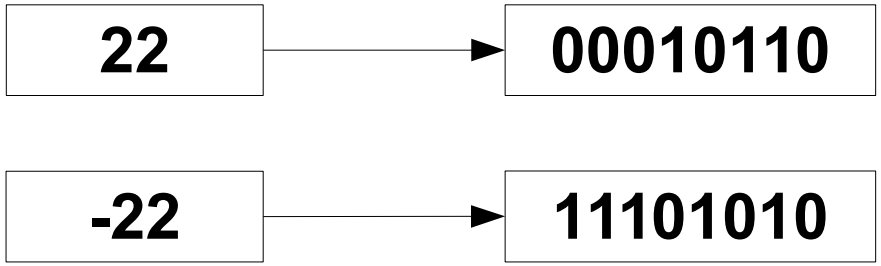
Overflow

$$v = \bar{x}_{n-1} \cdot \bar{y}_{n-1} \cdot c_{n-2} + x_{n-1} \cdot y_{n-1} \cdot \bar{c}_{n-2}$$

$$c_{n-1} = x_{n-1} \cdot y_{n-1} + x_{n-1} \cdot c_{n-2} + y_{n-1} \cdot c_{n-2}$$

$$v = c_{n-1} \, xor \, c_{n-2}$$

Subtractor



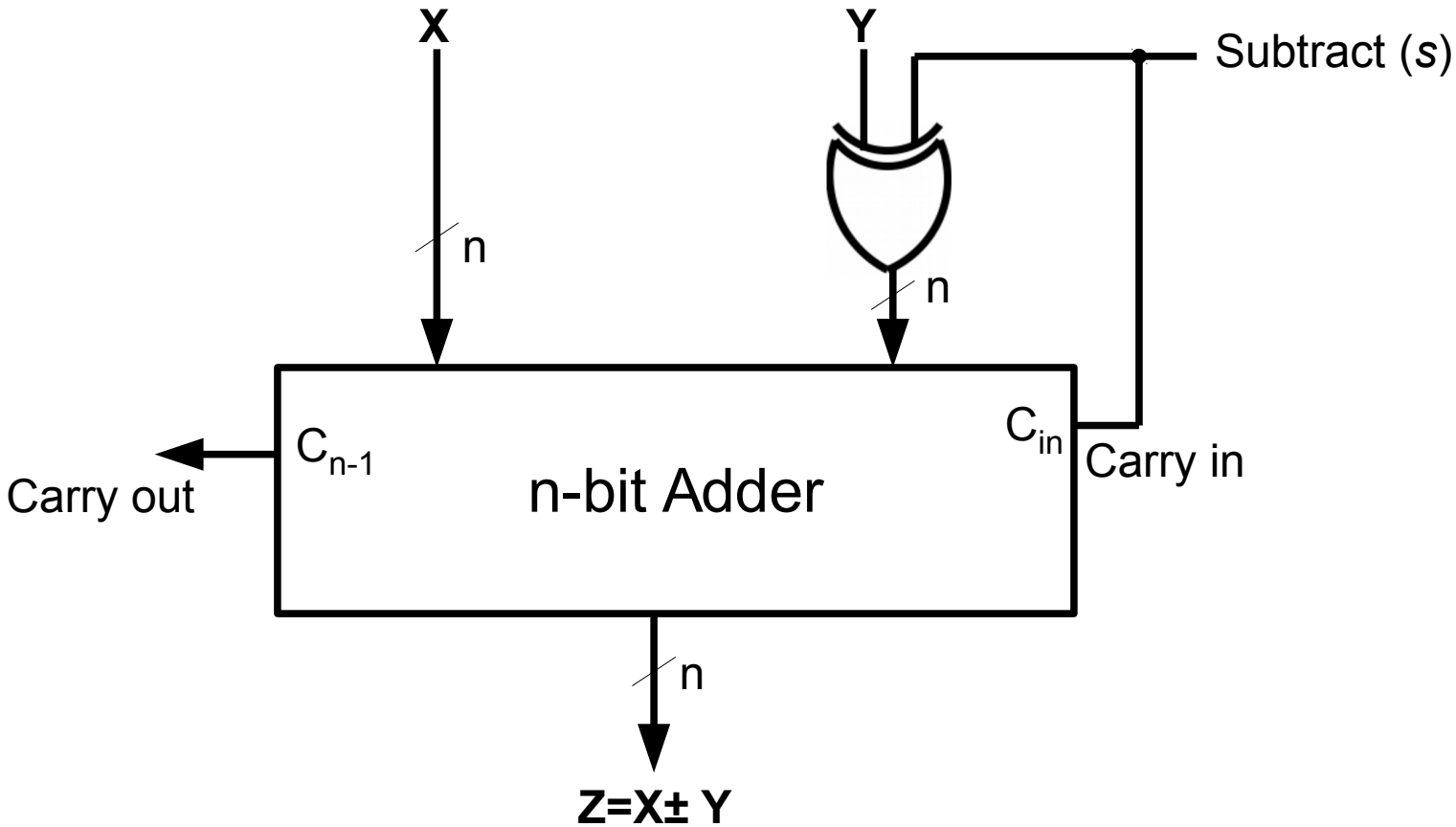
Negative of a 2's complement number is obtained by

- Inverting the bits.
- Adding 1

- Easy method to invert bits: XOR with 1

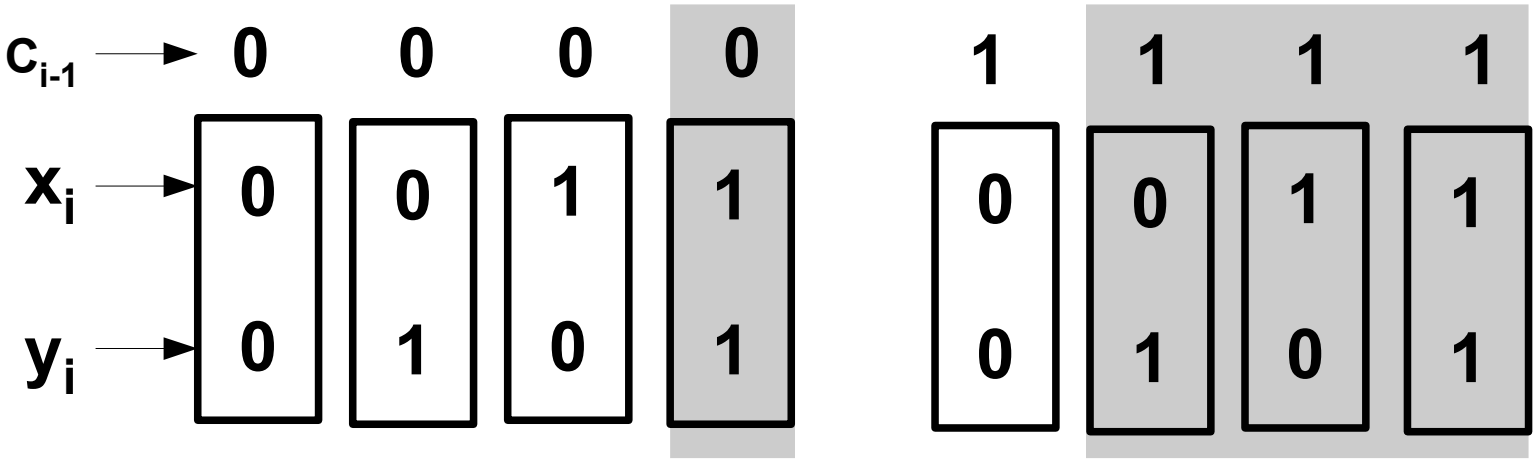
X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Subtractor



High Speed Adders

- Reduce the time required to form carry signals
- When is carry = 1?

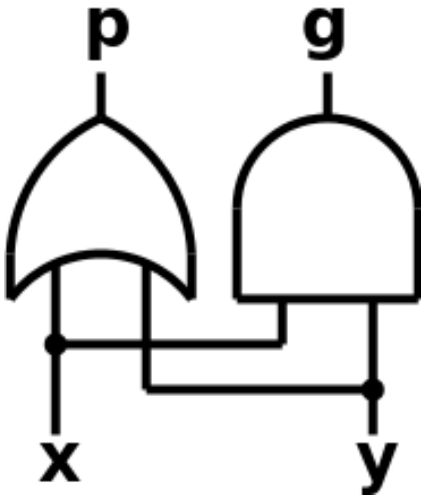


Generate and Propagate

- Generate
  - Stage i generates a carry bit without C_{i-1}
- Propagate
  - Stage i propogates C_{i-1} if x_i or y_i is 1.

$$g_i = x_i y_i$$

$$p_i = x_i + y_i$$



Carry-Lookahead Adder

The Carry Equation

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

$$g_i = x_i y_i \qquad p_i = x_i + y_i$$

$$\boxed{c_i = g_i + p_i g_{i-1} + p_i p_{i-1} c_{i-2}}$$

$$\begin{cases} c_i = g_i + p_i c_{i-1} \\ c_{i-1} = g_{i-1} + p_{i-1} c_{i-2} \\ c_{i-2} = g_{i-2} + p_{i-2} c_{i-3} \end{cases}$$

$$\boxed{c_i = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + p_i p_{i-1} p_{i-2} c_{i-3}}$$

4-bit Generate and Propagate

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

$$c_0 = g_0 + p_0 c_{-1}$$

$$c_1 = g_1 + p_1 c_0$$

$$c_1 = g_1 + p_1 g_0 + p_1 p_0 c_{-1}$$

$$c_2 = g_2 + p_2 c_1$$

$$c_3 = g_3 + p_3 c_2$$

$$c_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_{-1}$$

$$c_3 = g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 + p_3 p_2 p_1 p_0 c_{-1}$$

Carry is generated without waiting for the addition at previous level to complete.

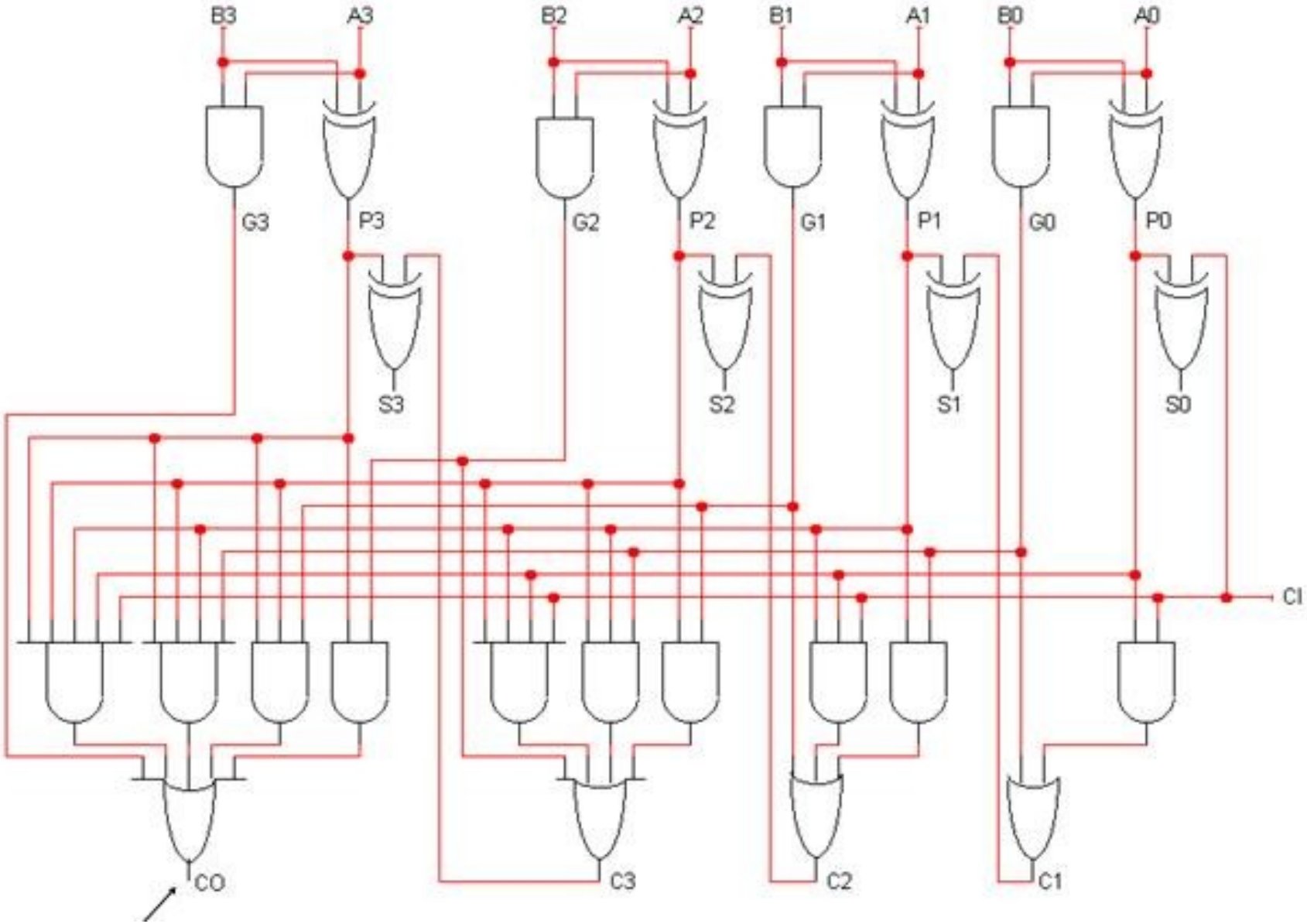
Carry Lookahead Adder

- Compare $\longrightarrow$ $z_i = x_i \, xor \, y_i \, xor \, c_{i-1}$

  $\longrightarrow$ $z_i = p_i \, xor \, g_i \, xor \, c_{i-1}$

- CLA has 4 levels of gates – max delay 4d
- No. of gates grows as n^2.
- High fan-in, high fan-out

CLA


B3
A3
B2
A2
B1
A1
B0
A0
G3
P3
G2
P2
G1
P1
G0
P0
S3
S2
S1
S0
CI
CO
C3
C2
C1