COMP311: *COMPUTER ORGANIZATION!*

Lecture 10: Adder Circuits (& More!)

tinyurl.com/comp311-fa25

Logistics!

- Written Assignment due next week!
- Lab 2 will be released today
- Quiz on Tuesday! Review session at 5pm on Monday (Location TBD)
- Quiz Topics
  - *Boolean Algebra + Simplification; SOP form*
  - *K-Maps*
  - *”Don’t Cares” (in relation to to K-Maps, Truth Table Minimization)*
  - *Transistor Minimization (the lecture where we converted things into NAND/NOR gates)*
  - *Multiplexors*

Name	OR Form	AND Form
Identity	$A + 0 = A$	$A \cdot 1 = A$
Null	$A + 1 = 1$	$A \cdot 0 = 0$
Idempotent	$A + A = A$	$A \cdot A = A$
Complement	$\overline{A} + A = 1$	$A \cdot \overline{A} = 0$
Commutative	$A + B = B + A$	$A \cdot B = B \cdot A$
Associative	$(A + B) + C = A + (B + C)$	$(A \cdot B) \cdot C = A \cdot (B \cdot C)$
Distributive	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$
Absorption 1	$A + AB = A$	$A(A + B) = A$
Absorption 2	$A + \overline{A}B = A + B$	$\overline{A}(A + B) = \overline{A}B$
De Morgan's	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Table 1: Boolean Laws

Gate	Number of Transistors
Inverter	2
n-input NAND	2n
n-input NOR	2n
n-input AND	2n + 2
n-input OR	2n + 2

Table 2: Transistor Count

Reminder of where we are going!

A Circuit for If-Statements

```
if (S == 0)
    Y = A
elseif (S == 1)
    Y = B
```

Implementing an if-statement

Truth Table

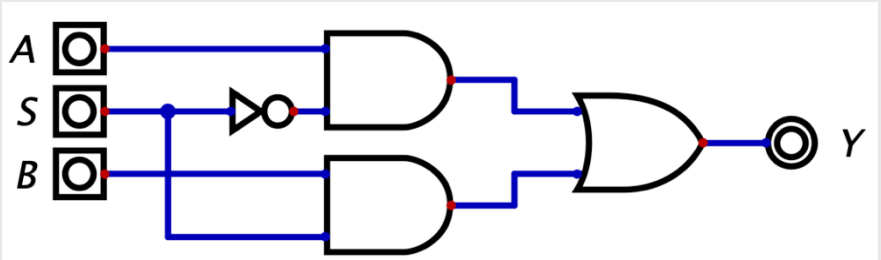
S	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

S	A	B	Y
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

Equation

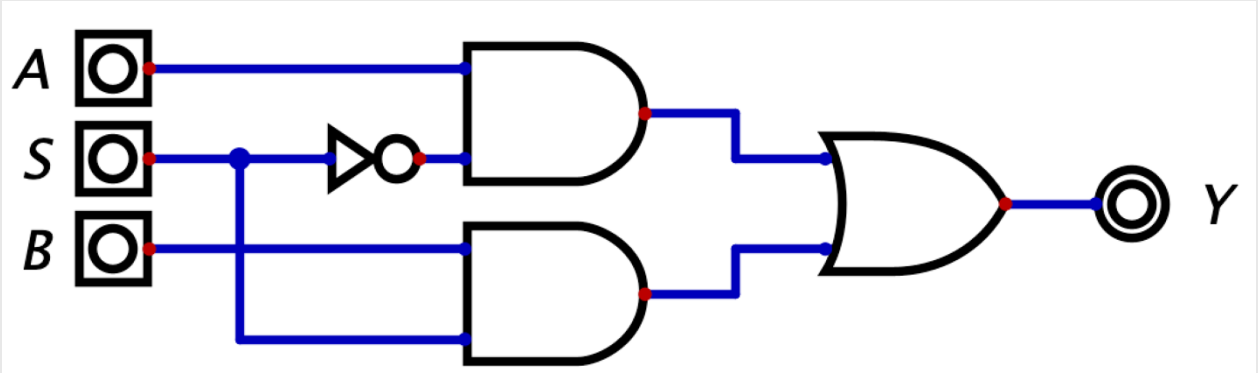
$$Y = A\bar{S} + BS$$

Diagram

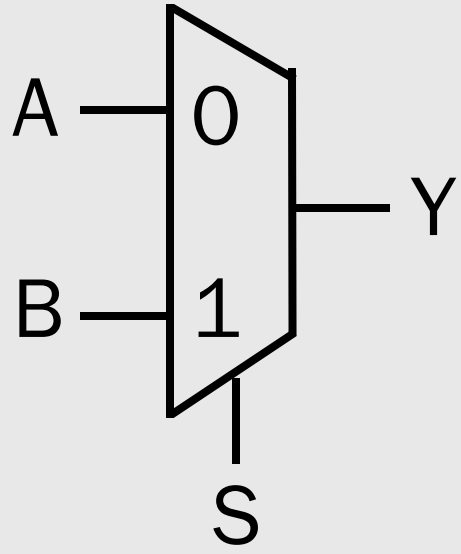


2:1 Multiplexer (2:1 mux)

- A circuit that chooses an output from among two inputs based on the value of a select signal



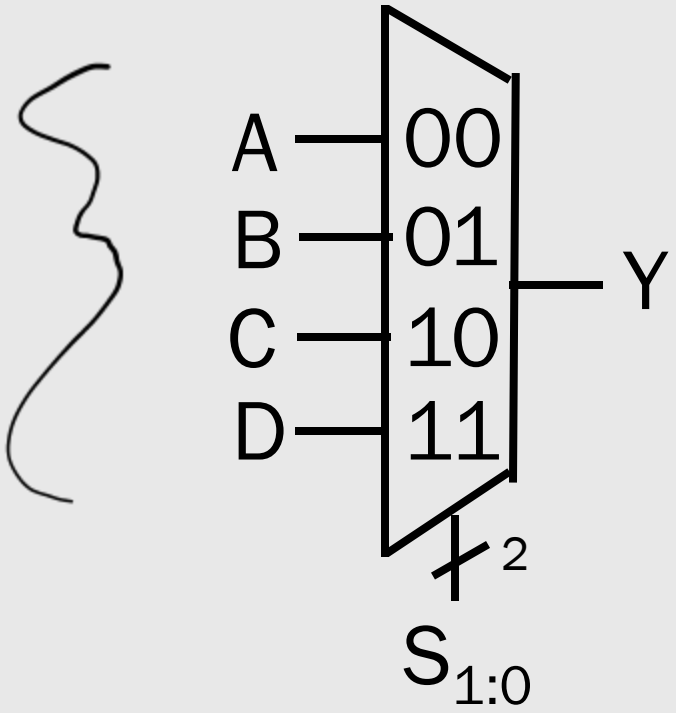
2:1 Multiplexer Schematic
(Gate Diagram)



2:1 Multiplexer Symbol

4:1 Multiplexer (4:1 mux)

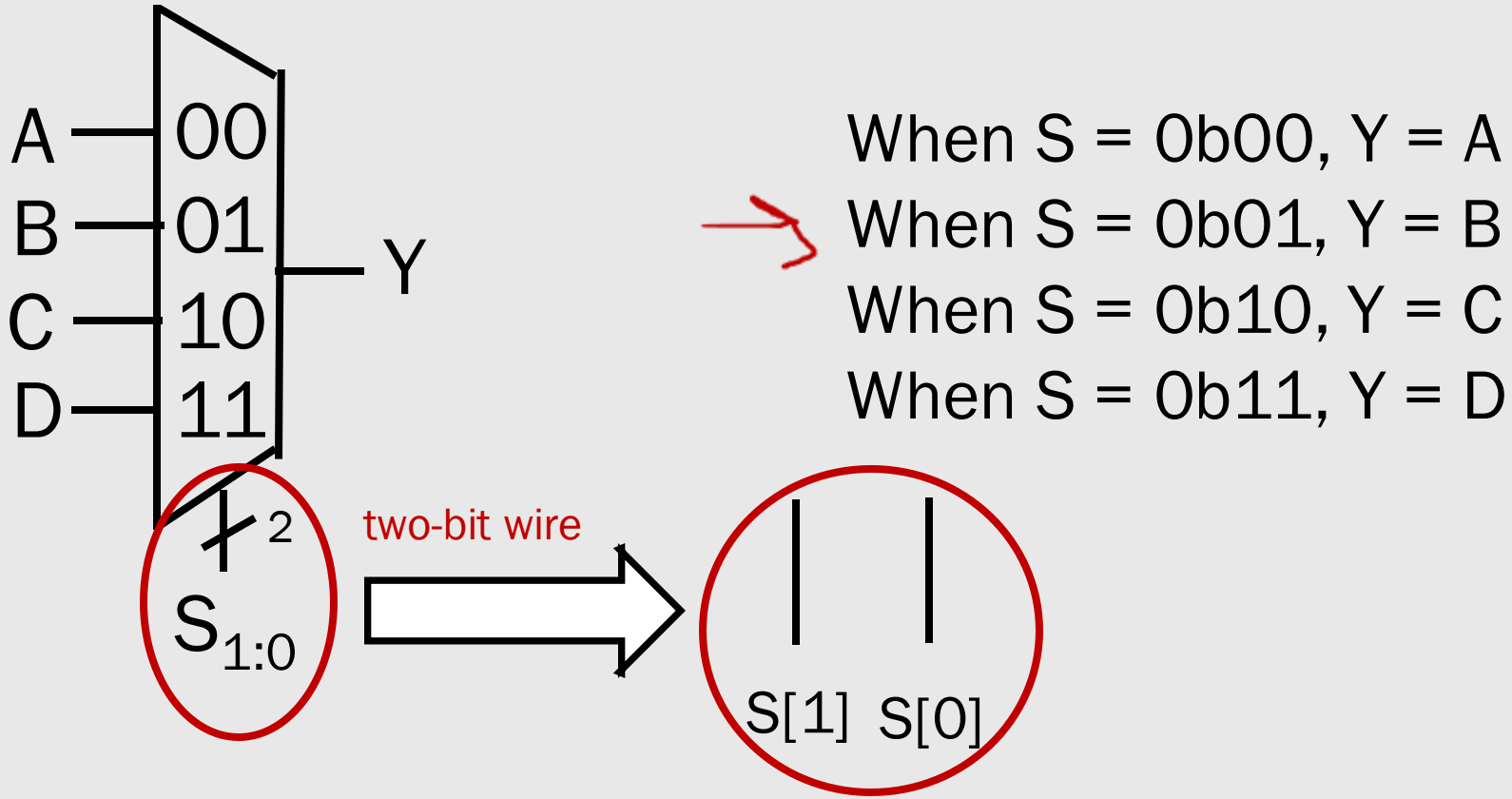
- A circuit that chooses an output from among four inputs based on the value of a select signal



When S = 0b00, Y = A
When S = 0b01, Y = B
When S = 0b10, Y = C
When S = 0b11, Y = D

4:1 Multiplexer (4:1 mux)

- A circuit that chooses an output from among four inputs based on the value of a select signal



Create the gate-level implementation of a 4:1 Multiplexer (4:1 mux)

1. Draw the truth table
2. Find the equation
3. Draw the schematic

Create the gate-level implementation of a 4:1 Multiplexer (4:1 mux)

S1	S0	A	B	C	D	Y
0	0	0	X	X	X	0
0	0	1	X	X	X	1
0	1	X	0	X	X	0
0	1	X	1	X	X	1
1	0	X	X	0	X	0
1	0	X	X	1	X	1
1	1	X	X	X	0	0
1	1	X	X	X	1	1

$$Y = A\bar{S}_1\bar{S}_0 + B\bar{S}_1S_0 + CS_1\bar{S}_0 + D\ S_1S_0$$

4:1 Multiplexer (4:1 mux)

Equation

$$Y = A\overline{S}_1\overline{S}_0 + B\overline{S}_1S_0 + CS_1\overline{S}_0 + D\ S_1S_0$$

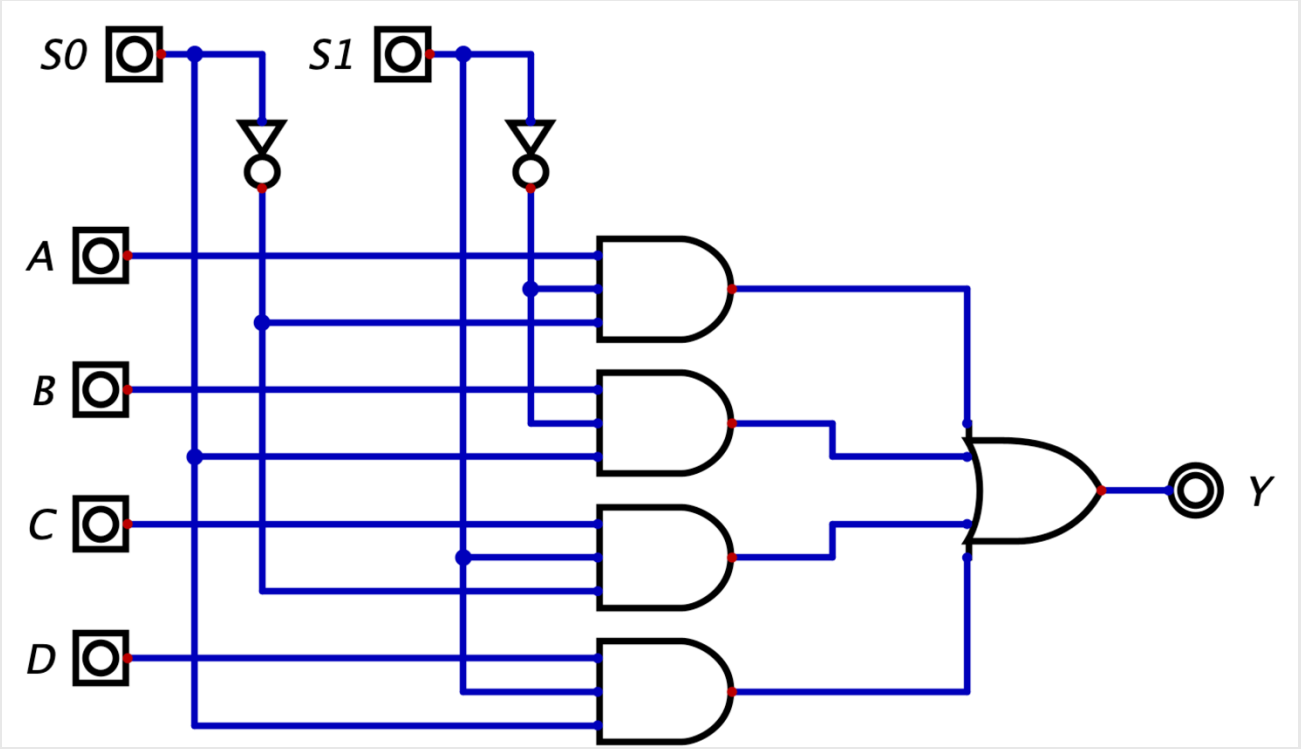
Diagram

4:1 Multiplexer (4:1 mux)

Equation

$$Y = A\overline{S}_1\overline{S}_0 + B\overline{S}_1S_0 + CS_1\overline{S}_0 + D\ S_1S_0$$

Diagram



Multiplexers

- Create a 4:1 multiplexer out of 2:1 multiplexers.

S1	S0	Y
0	0	A
0	1	B
1	0	C
1	1	D

A

B

C

D

S_1

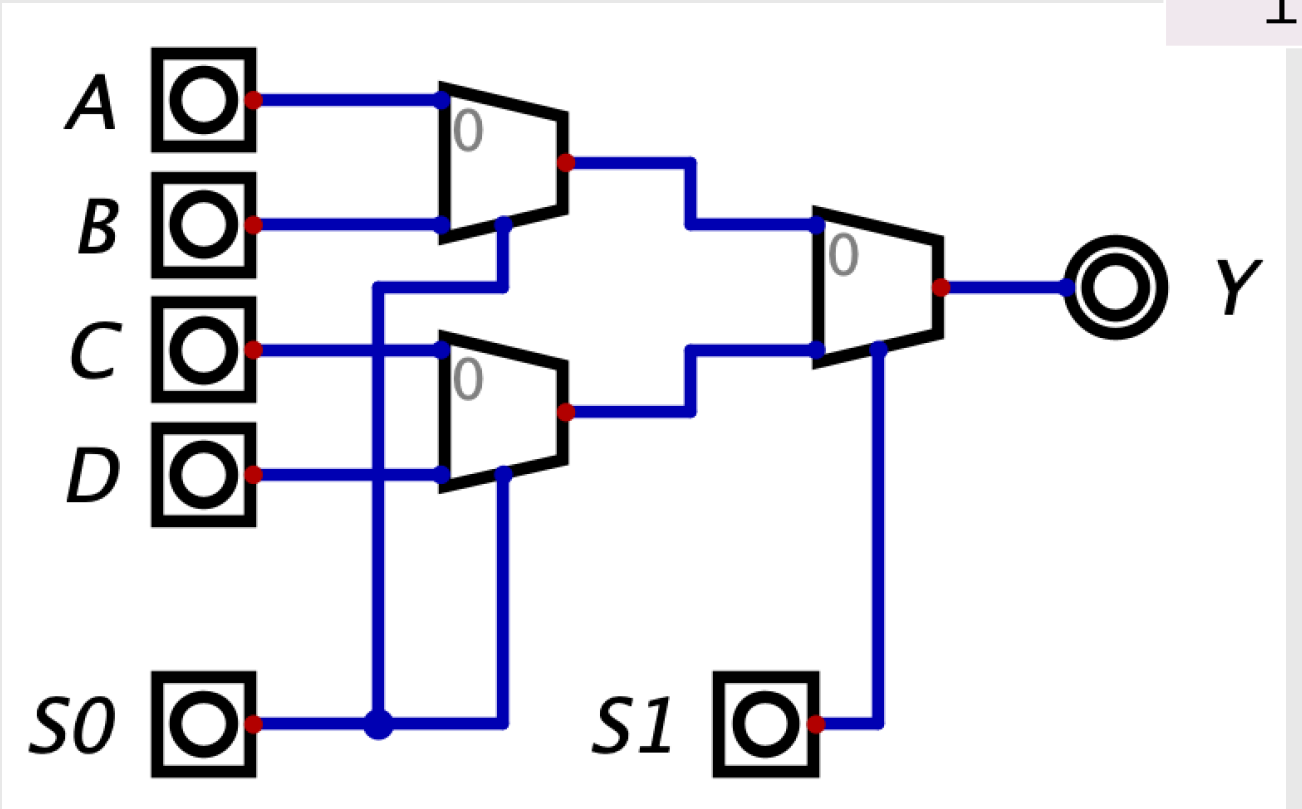
S_0

Y

Multiplexers

- Create a 4:1 multiplexer out of 2:1 multiplexers.

S1	S0	Y
0	0	A
0	1	B
1	0	C
1	1	D

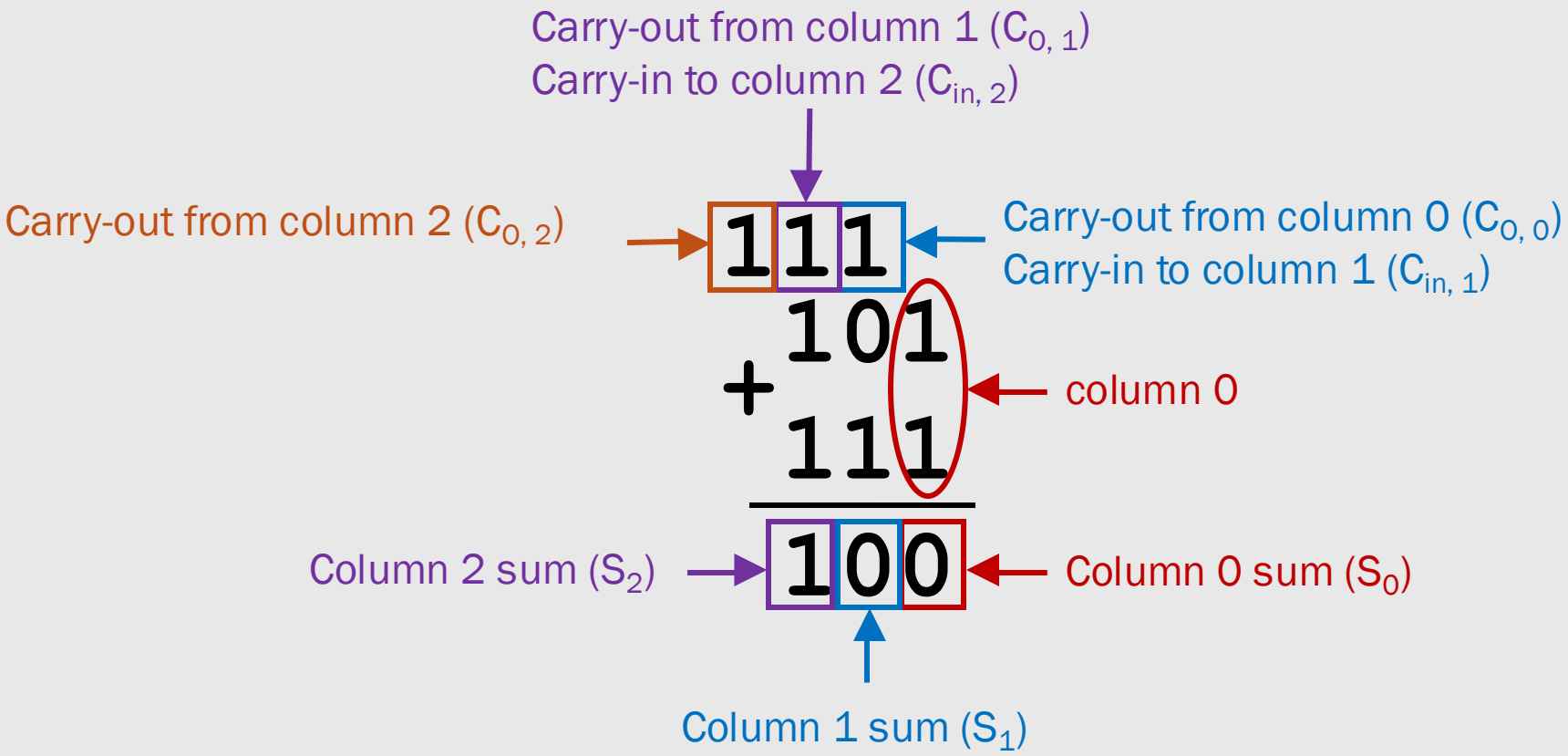


ADDER CIRCUIT!

Recall: Binary Addition

$$\begin{array}{r} 101 \\ +\ 111 \\ \hline \end{array}$$

Recall: Binary Addition



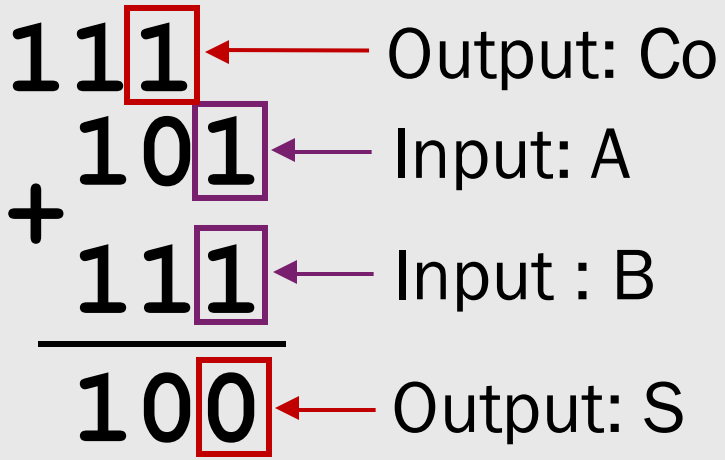
Half Adder

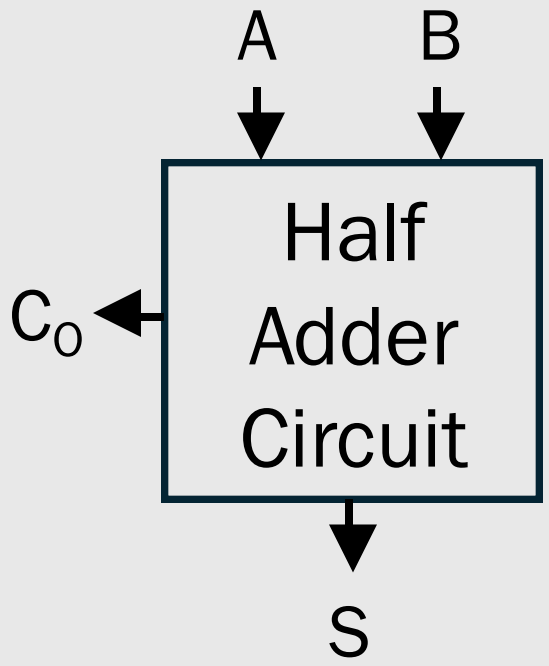
- We'll start off by building a circuit that can add together two one-bit values
- This circuit will be able to add together the two bits in column 0

```
 111
  101
+ 111
-----
 1100
```

Half Adder

- We'll start off by building a circuit that can add together two one-bit values
- This circuit will be able to add together the two bits in column 0





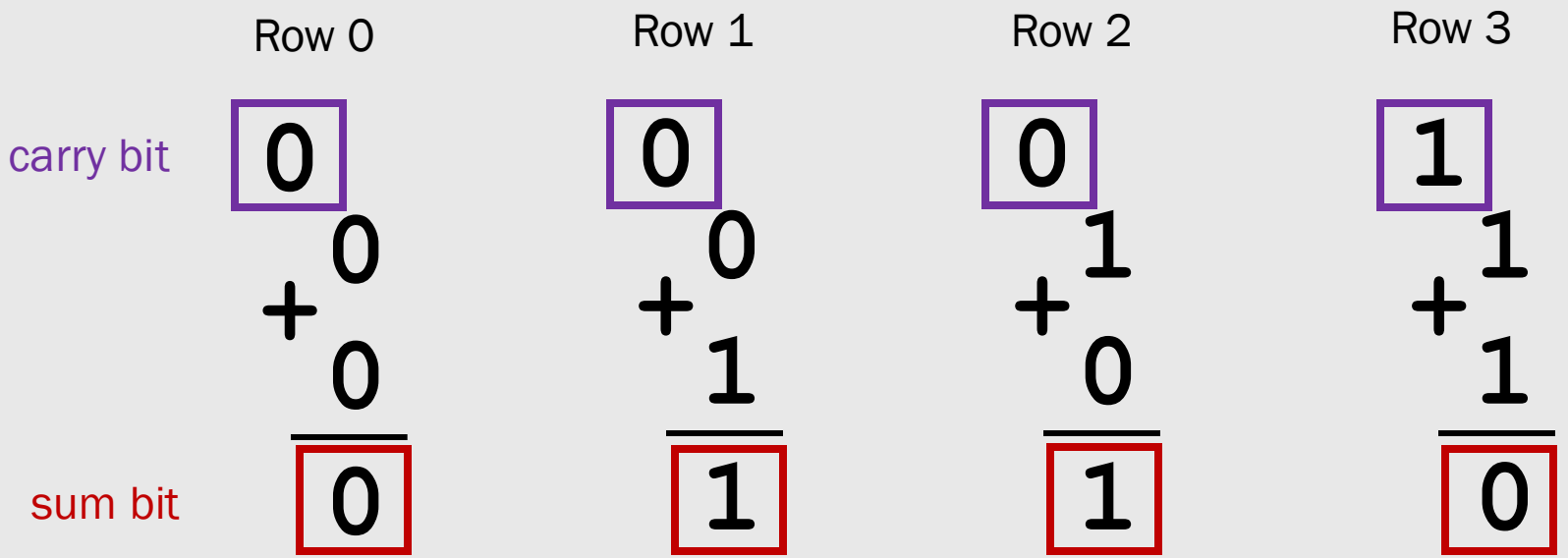
Half Adder

A	B	C_0	S
0	0		
0	1		
1	0		
1	1		

$$\begin{array}{r} 0 \\ +\ 0 \\ \hline \end{array} \qquad \begin{array}{r} 0 \\ +\ 1 \\ \hline \end{array} \qquad \begin{array}{r} 1 \\ +\ 0 \\ \hline \end{array} \qquad \begin{array}{r} 1 \\ +\ 1 \\ \hline \end{array}$$

Half Adder

A	B	C_0	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

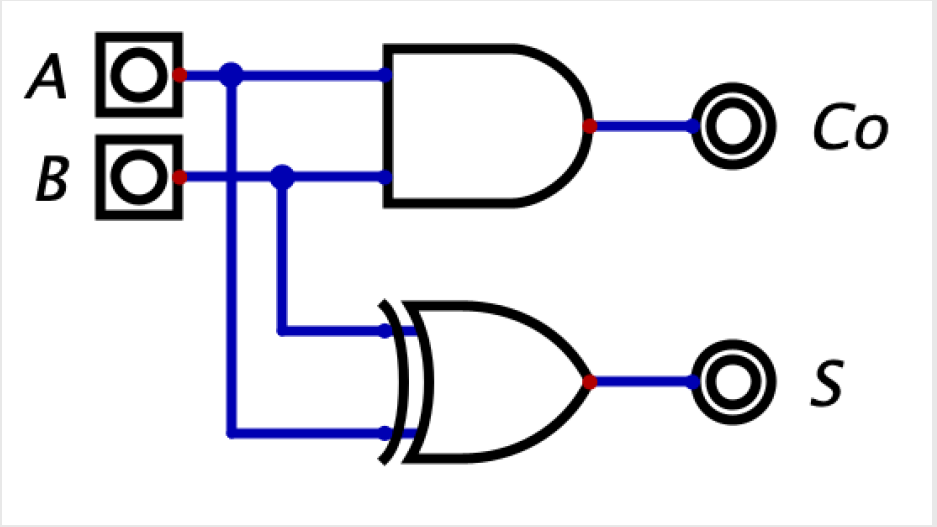


Half Adder

A	B	C_0	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

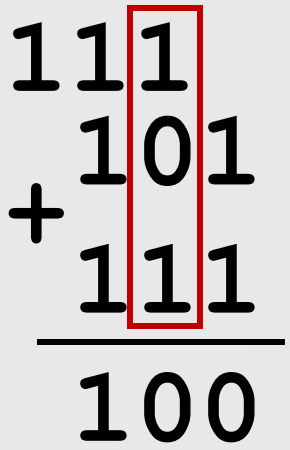
$$S = A \oplus B$$

$$C_O = AB$$



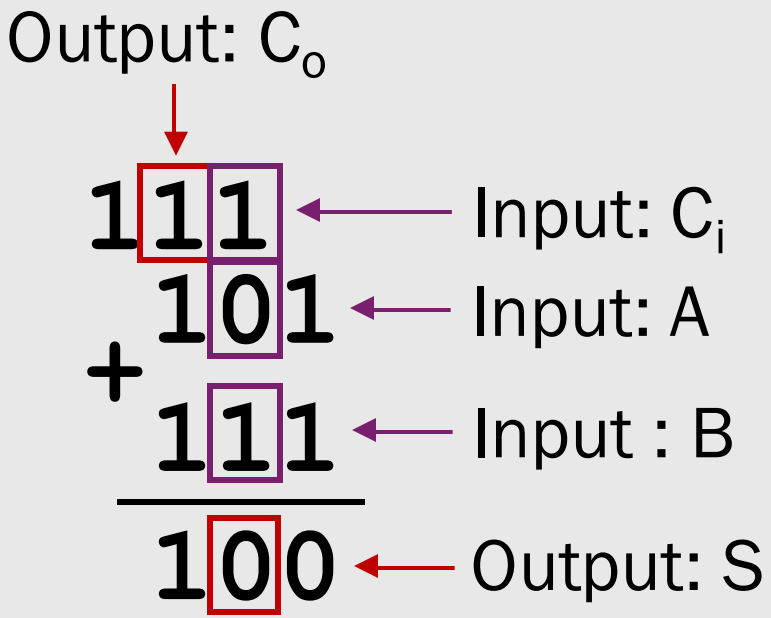
Full Adder

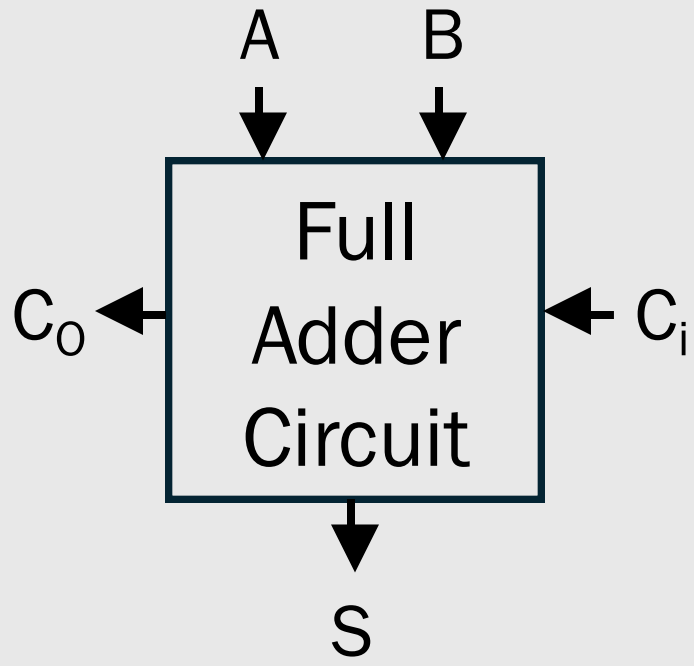
- We need to build a circuit that can add together three one-bit values



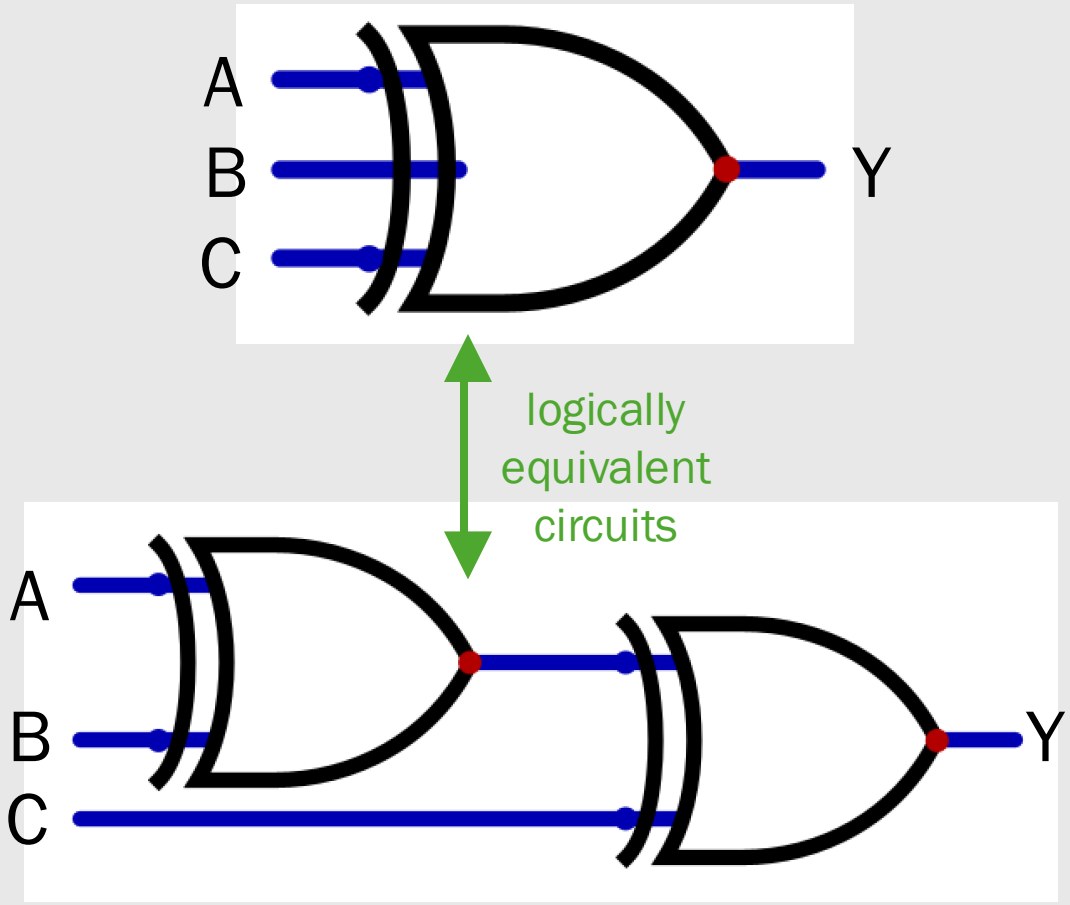
Full Adder

- This circuit will be able to add together the three bits in column 1





Recall: 3-input XOR



A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

For any number of inputs, the output is 1 when an odd number of inputs is 1

Transistor Count

Gate	# of Transistors
NOT	2
n-input AND	2n + 2
n-input OR	2n + 2
n-input NAND	2n
n-input NOR	2n
2-input XOR	12
3-input XOR	28
2-input XNOR	12
3-input XNOR	28

Create the gate-level implementation of a full adder

1. Draw the truth table
2. Find the equation
3. Draw the schematic. Try to minimize the circuit.
4. How many transistors does your solution use?

Full Adder

A	B	C_i	C_o	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		

Equations

Full Adder

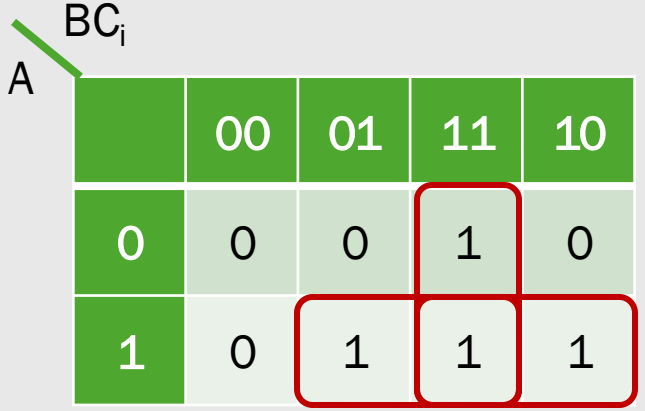
Diagram

How many transistors are in this design?

Full Adder

A	B	C_i	C_O	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

$$S = A \oplus B \oplus Ci$$

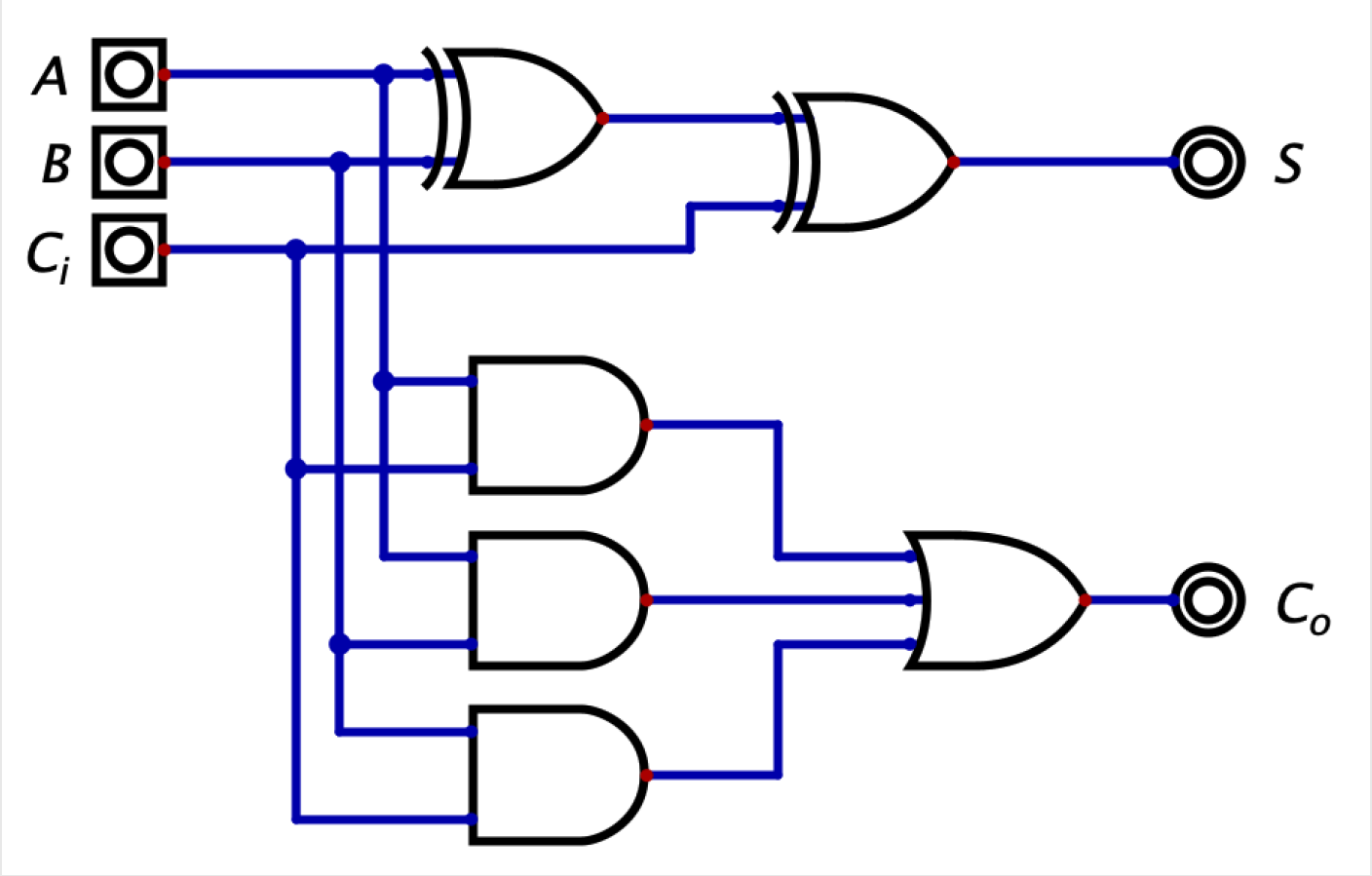


$$C_O = ACi + AB + BCi$$

Full Adder

$$S = A \oplus B \oplus Ci$$

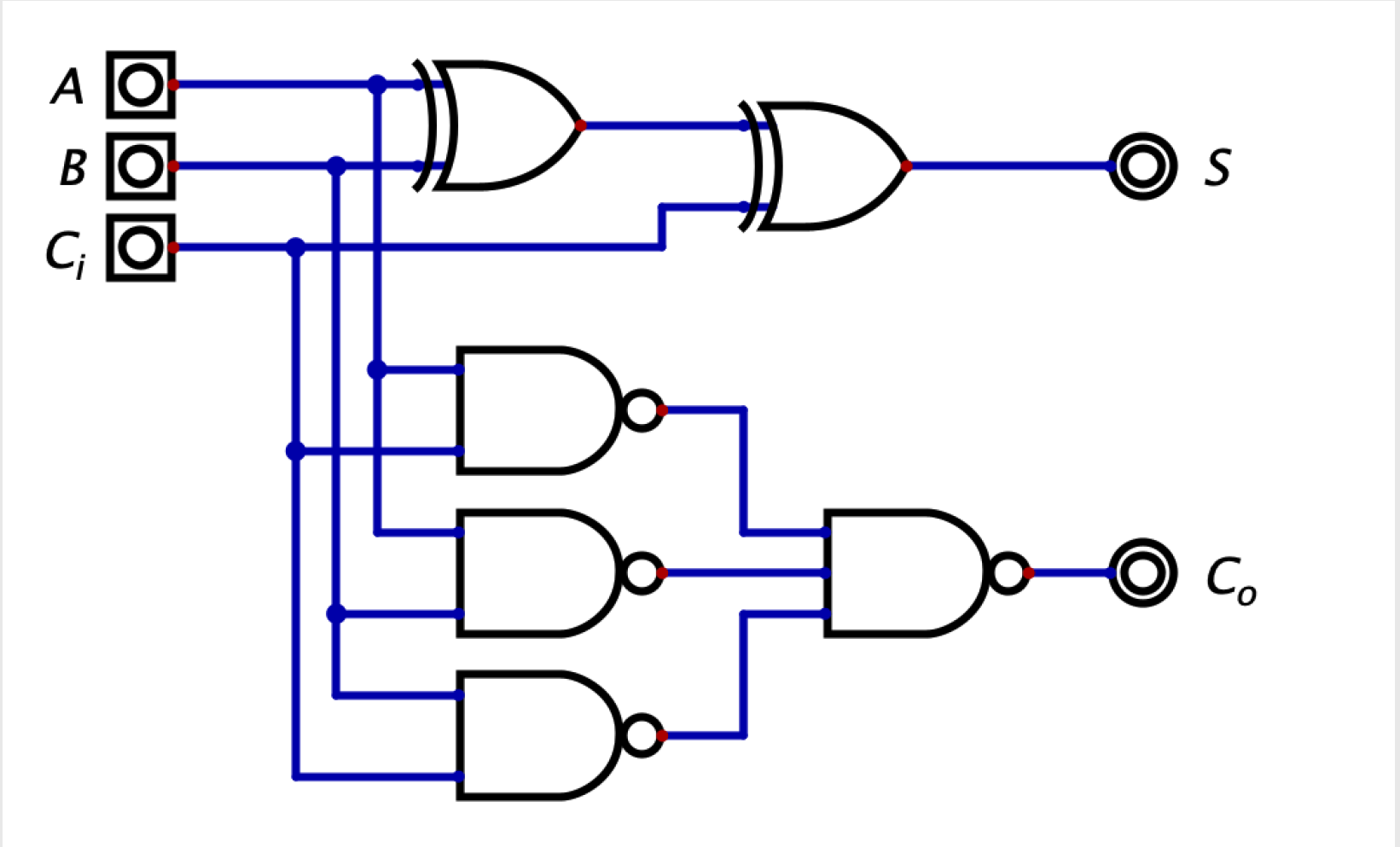
$$C_O = ACi + AB + BCi$$



Full Adder: transistor minimization

$$S = A \oplus B \oplus Ci$$

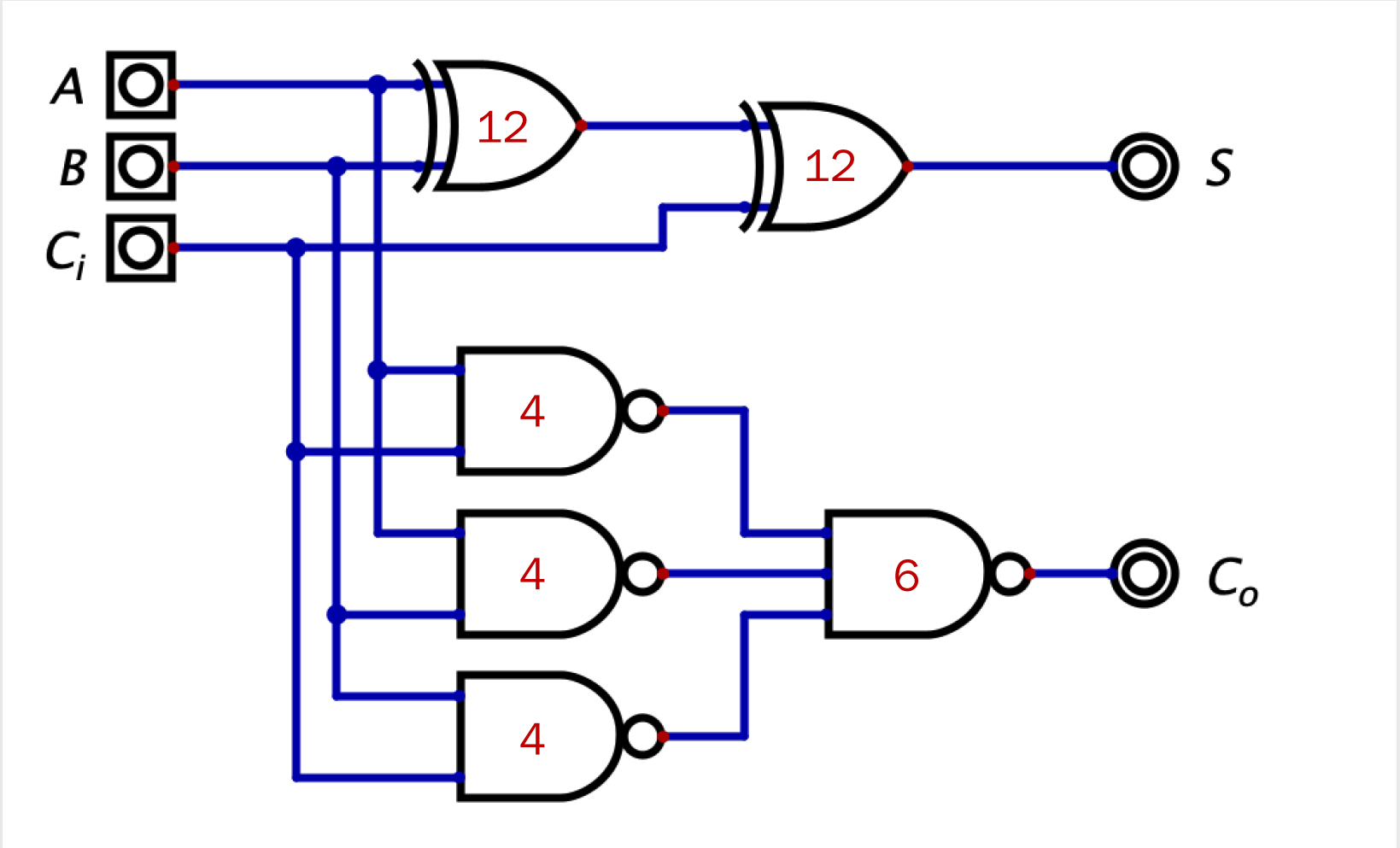
$$C_O = ACi + AB + BCi$$



Full Adder: transistor minimization

$S = A \oplus B \oplus Ci$

$C_O = ACi + AB + BCi$



Transistor count = 42

Full Adder: Alternate implementation

A	B	C_i	C_o	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

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

$$C_o =$$

Full Adder: Alternate Implementation

A	B	C_i	C_0	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

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

$$C_O = \bar{A}BC_i + A\bar{B}C_i + AB\overline{C}_i + ABCi$$

$$C_O = \bar{A}BC_i + A\bar{B}C_i + AB(\overline{C}_i + Ci)$$

$$C_O = \bar{A}BC_i + A\bar{B}C_i + AB$$

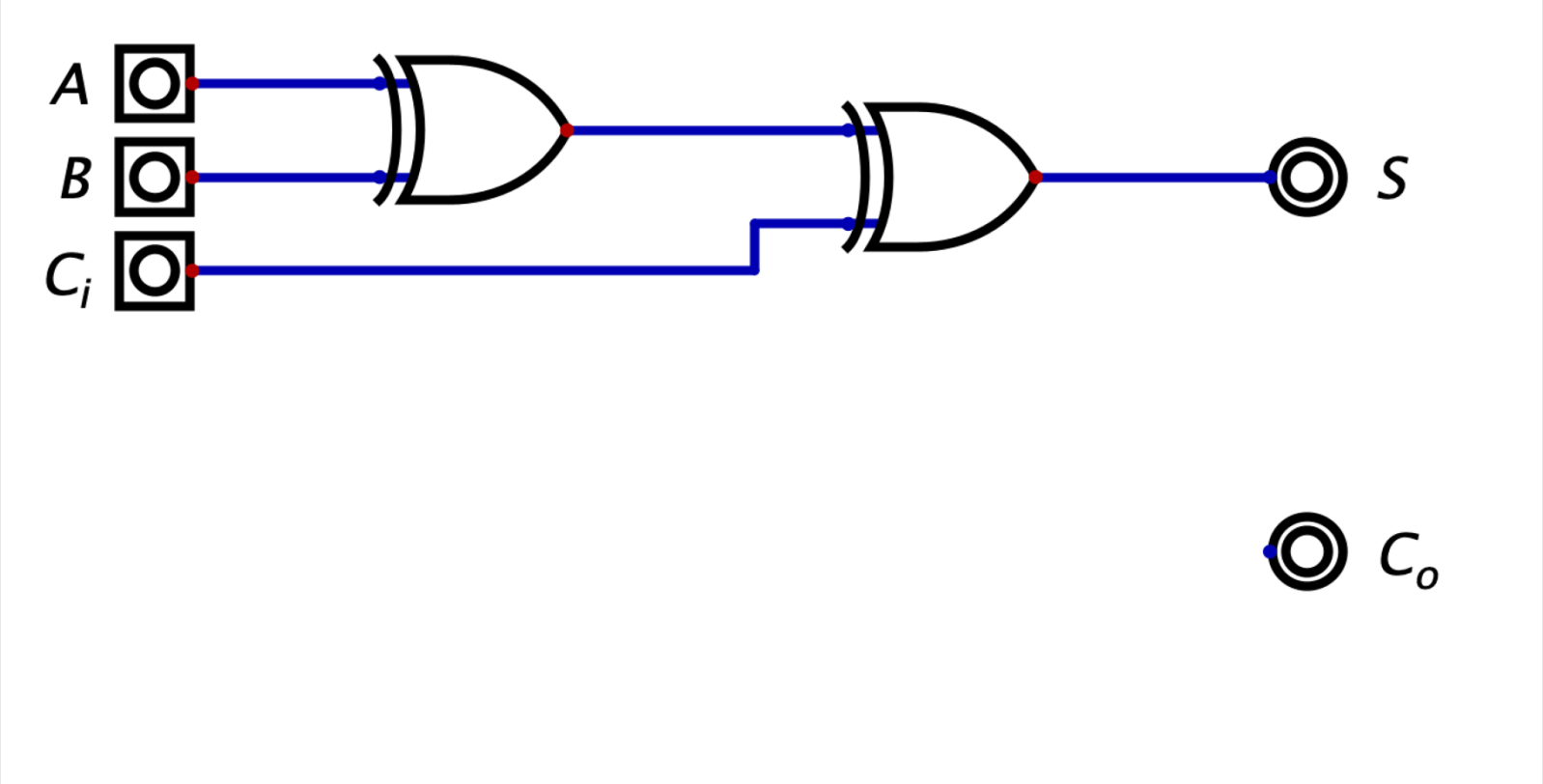
$$C_O = C_i(\bar{A}B + A\bar{B}) + AB$$

$$C_O = C_i(A \oplus B) + AB$$

Full Adder: Alternate Implementation

$S = A \oplus B \oplus Ci$

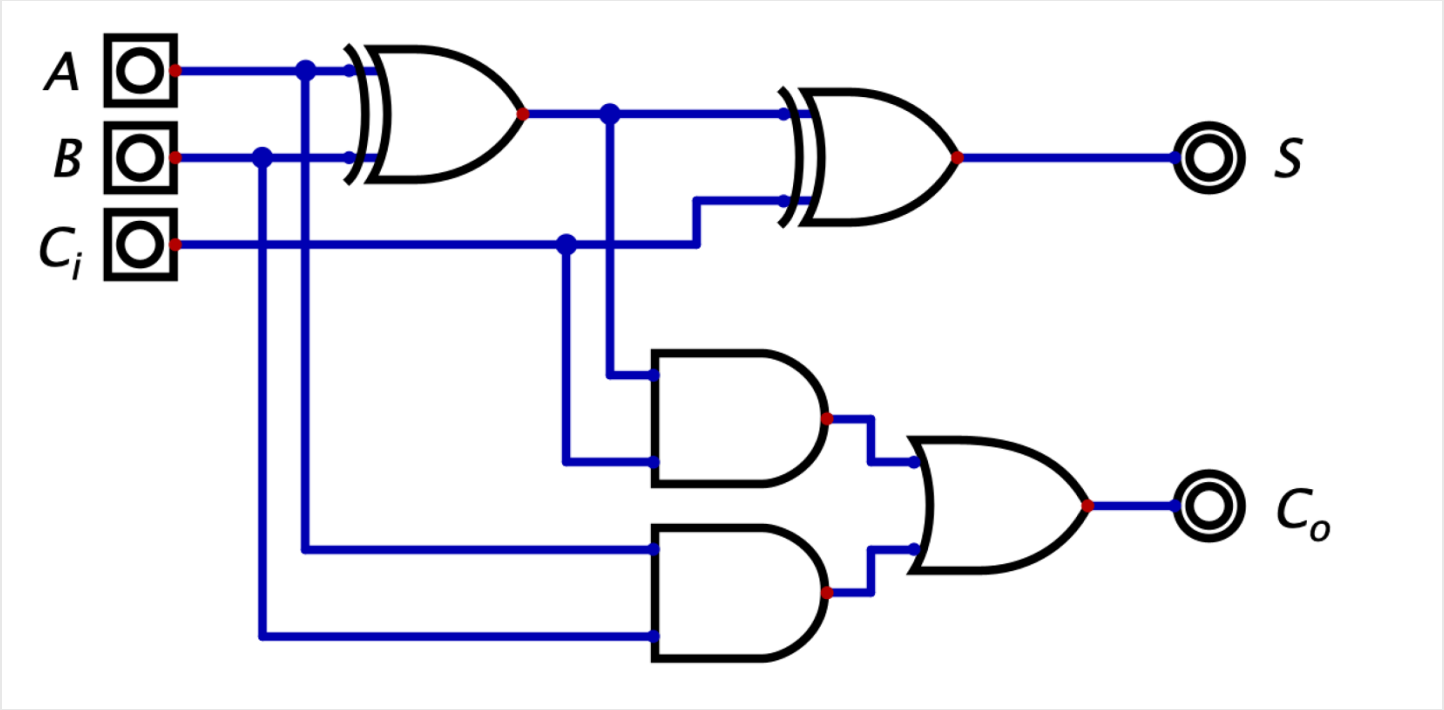
$C_O = C_i(A \oplus B) + AB$



Full Adder: Alternate Implementation

$$S = A \oplus B \oplus Ci$$

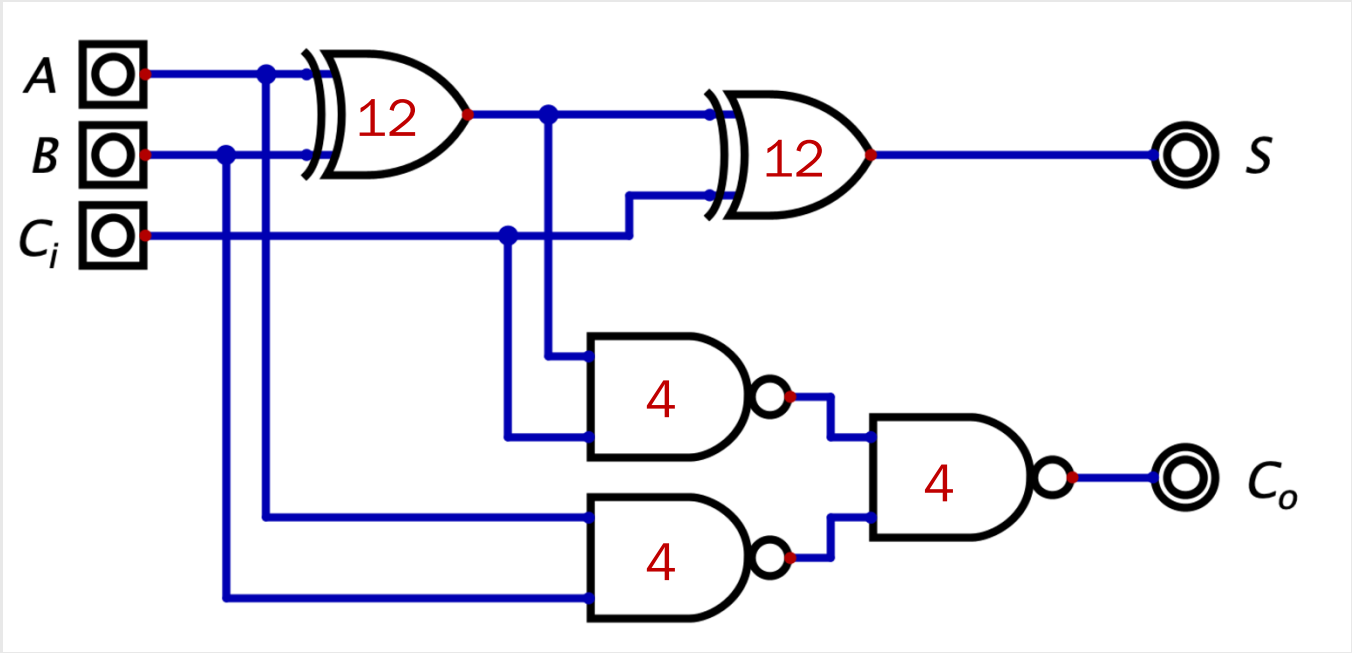
$$C_O = C_i(A \oplus B) + AB$$



Full Adder: Reducing Transistor Count

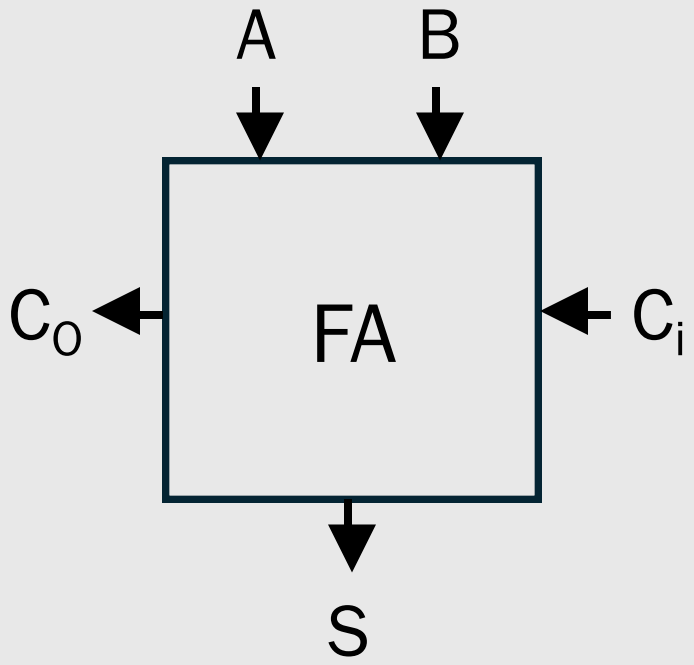
$S = A \oplus B \oplus Ci$

$C_O = C_i(A \oplus B) + AB$

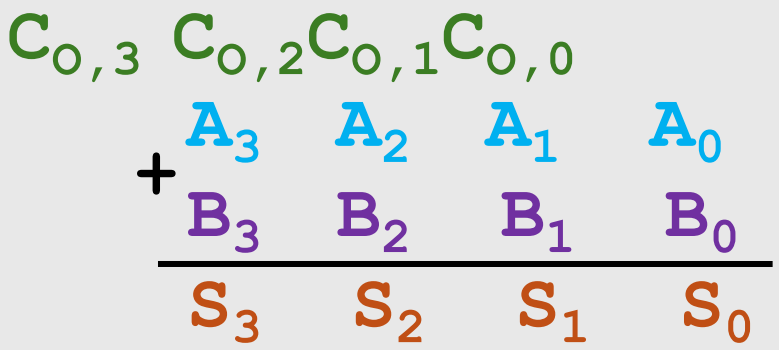


Transistor Count: 36

Full Adder Abstraction



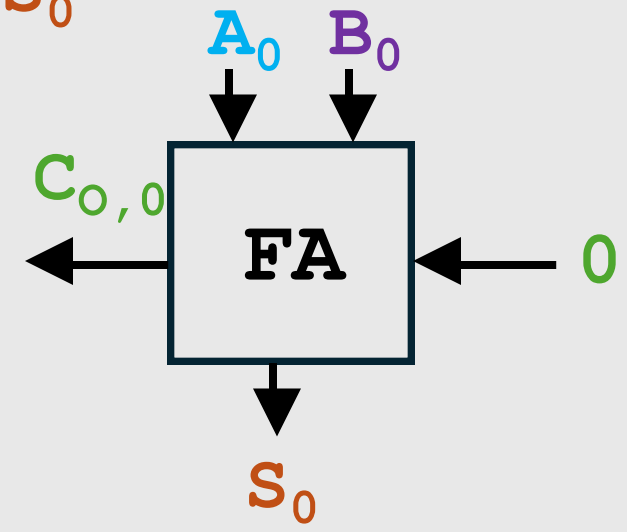
4-bit Adder



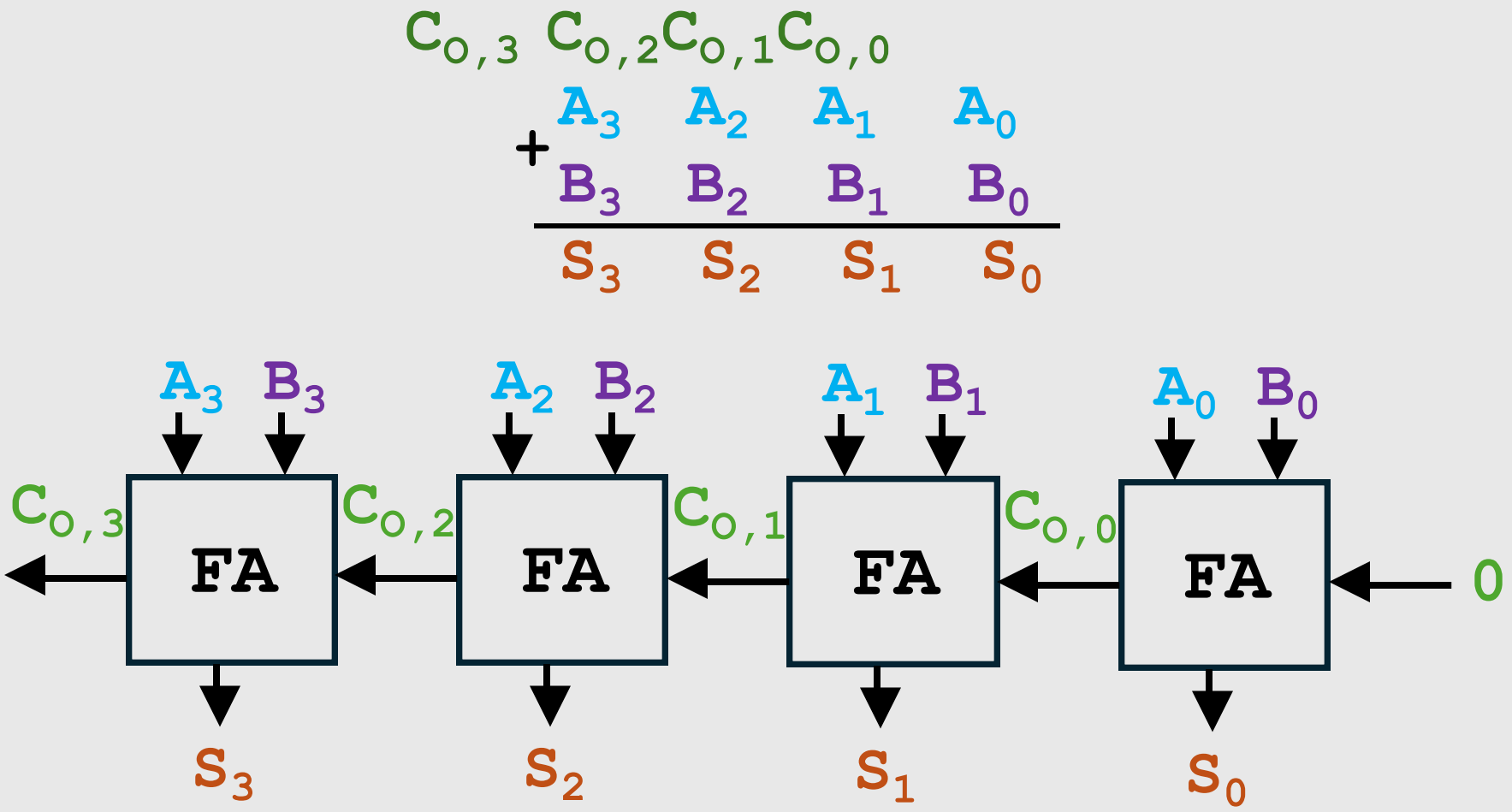
4-bit Adder

- How would you use 4 full-adders to add together A and B?

$$
\begin{array}{cccccc}
 & C_{o,3} & C_{o,2} & C_{o,1} & C_{o,0} & \\
 & A_3 & A_2 & A_1 & A_0 & \\
+ & B_3 & B_2 & B_1 & B_0 & \\
\hline
 & S_3 & S_2 & S_1 & S_0 &
\end{array}
$$



4-bit Adder


$C_{O,3}$ $C_{O,2}$ $C_{O,1}$ $C_{O,0}$
A_3 A_2 A_1 A_0
+
B_3 B_2 B_1 B_0
S_3 S_2 S_1 S_0
A_3 B_3
A_2 B_2
A_1 B_1
A_0 B_0
$C_{O,3}$
$C_{O,2}$
$C_{O,1}$
$C_{O,0}$
FA
FA
FA
FA
0
S_3
S_2
S_1
S_0

ADDER-SUBTRACTOR CIRCUIT

Recall: Two's Complement

Ex: Represent -15 in two's complement with 7 bits

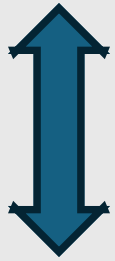
- To represent a negative number, take the positive representation of the number, flip the bits, and add 1

Unsigned: `0b 000 1111`
Flip bits: `0b 111 0000`
Add one: `0b 111 0001`

Recall Two's Complement Subtraction

$$\begin{array}{r} -10 \\ -\quad 4 \\ \hline -14 \end{array} \qquad \begin{array}{r} 10110_2 \\ -\ 00100_2 \\ \hline \end{array}$$

A – B == A + (-B)



Equivalent Operations

$$\begin{array}{r} -10 \\ +\quad -4 \\ \hline -14 \end{array} \qquad \begin{array}{r} 10110_2 \\ +\ 11100_2 \\ \hline 10010_2 \end{array}$$

Subtraction

- Instead of building a separate circuit for subtraction, we will modify the ripple carry adder such that it can perform addition and subtraction
- When we want to perform A – B
  - *We will actually perform A + (-B)*
  - *We need to modify our circuit such that it can negate B*

Subtraction

- Let's say that we want to perform the operation A – B where A = 5 and B = 1.
- We will modify the circuit such that it can flip each bit of B and add 1.

$$\begin{array}{r} 0101_2 \\ - \; 0001_2 \\ \hline \end{array}$$

$$\begin{array}{r} 0101_2 \\ + \; 1110_2 \\ + \qquad 1_2 \\ \hline 0100_2 \end{array}$$

Keep A the same

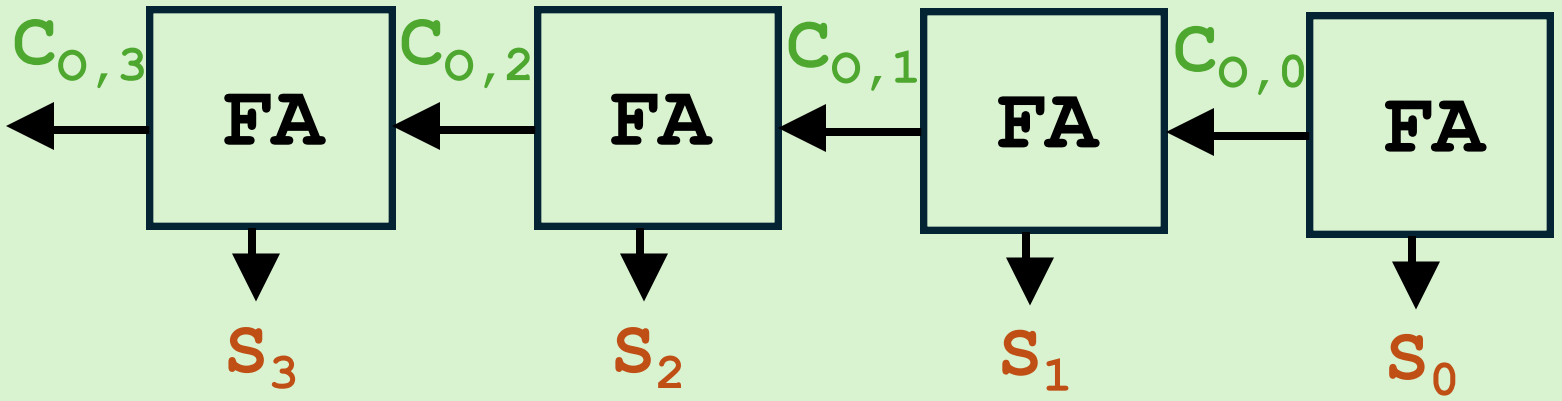
Flip the bits of B

Add 1

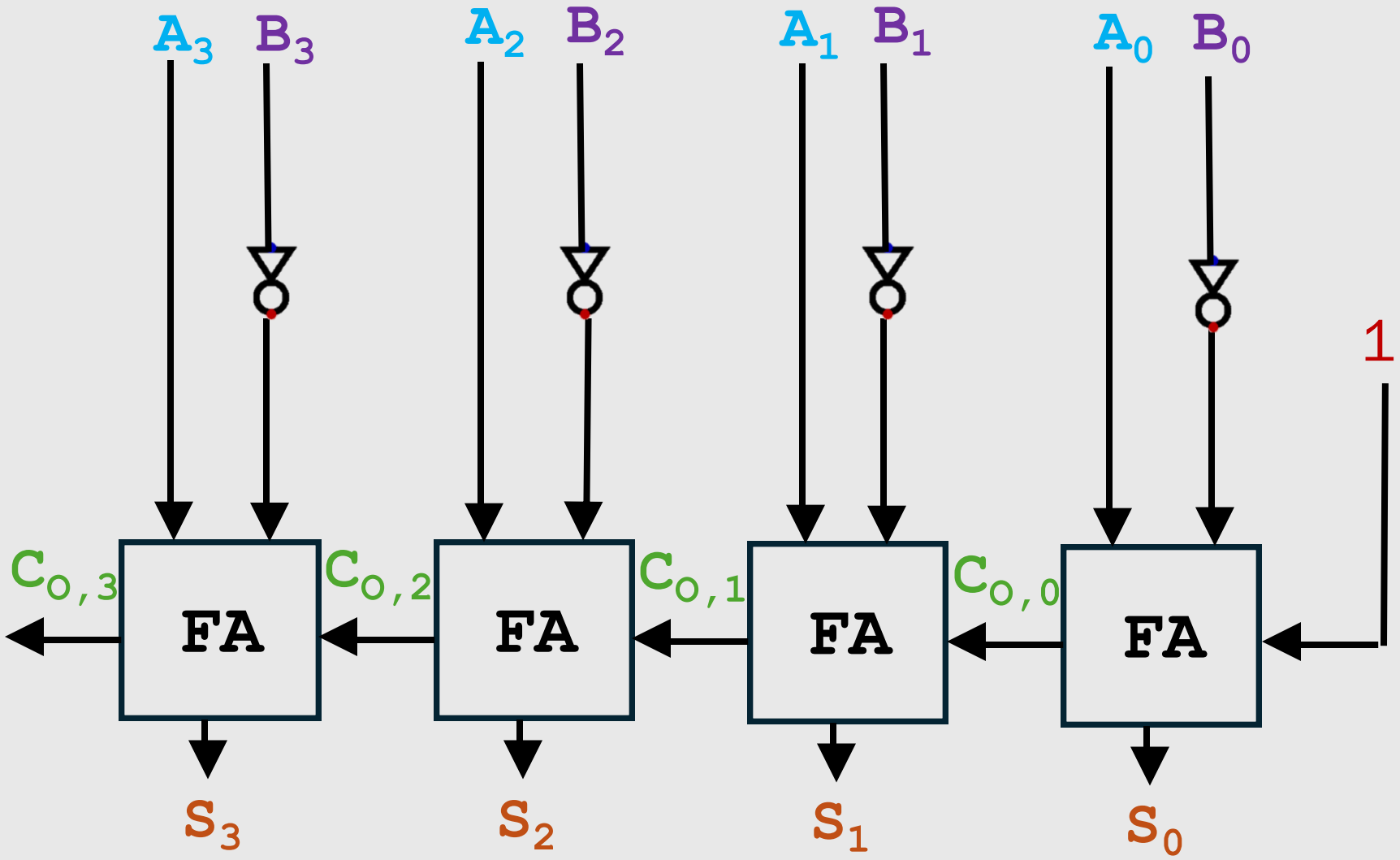
Subtractor

A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0

How would you wire up this circuit to have it correctly perform subtraction?



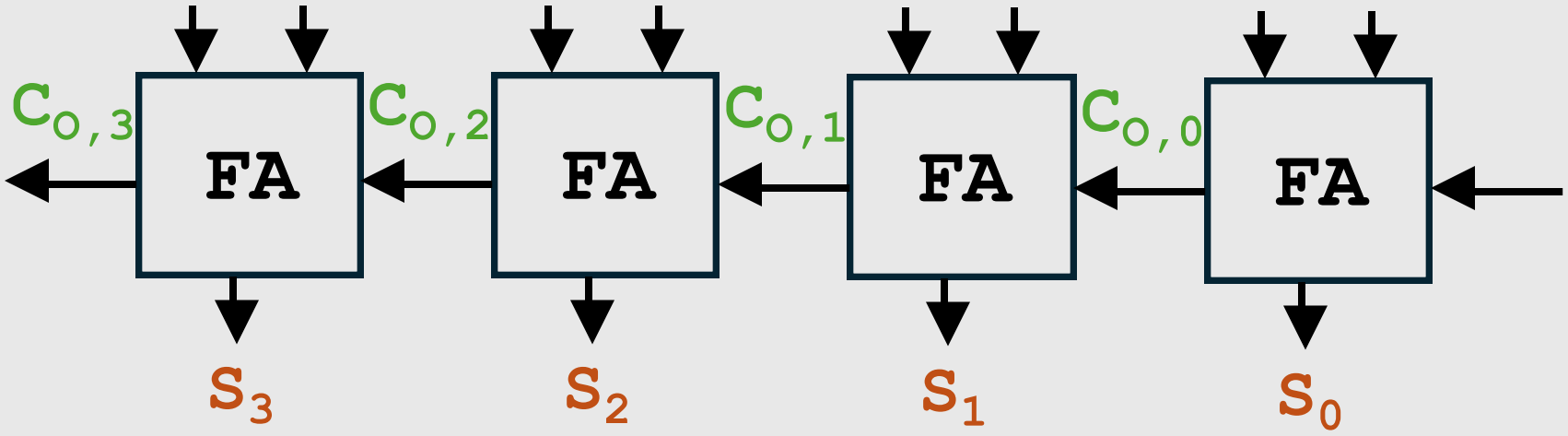
Subtractor



Adder/subtractor

A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0

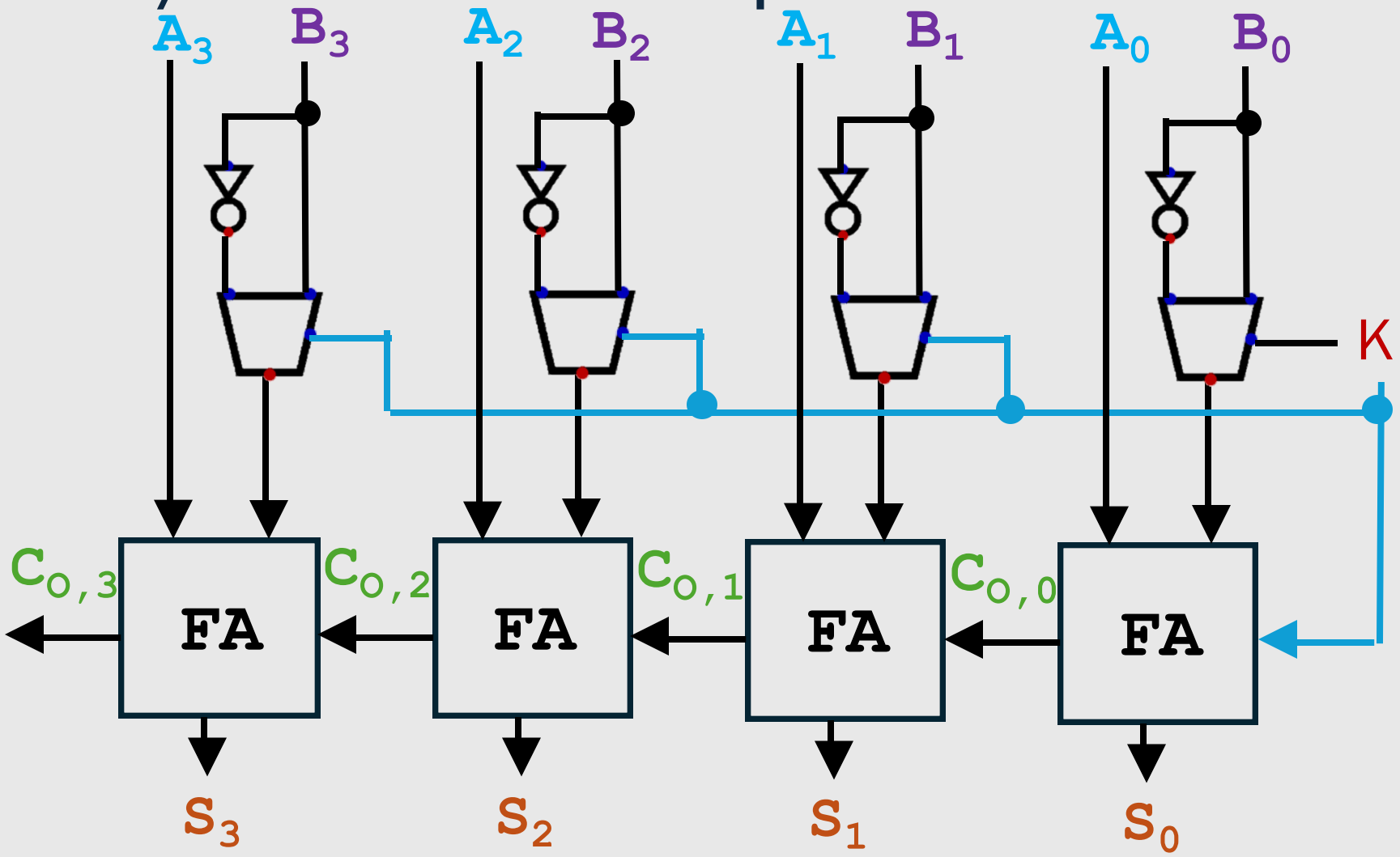
K If K == 0, A + B
If K == 1, A - B



Hints

- To perform subtraction, we must take the 2's complement of B. To take 2's complement, we must flip the bits and add one.
  - *What component can we use to flip bits?*
  - *Where in the circuit can we add 1?*

Adder/Subtractor Implementation 1



XOR

$$N \oplus 1 = \overline{N}$$

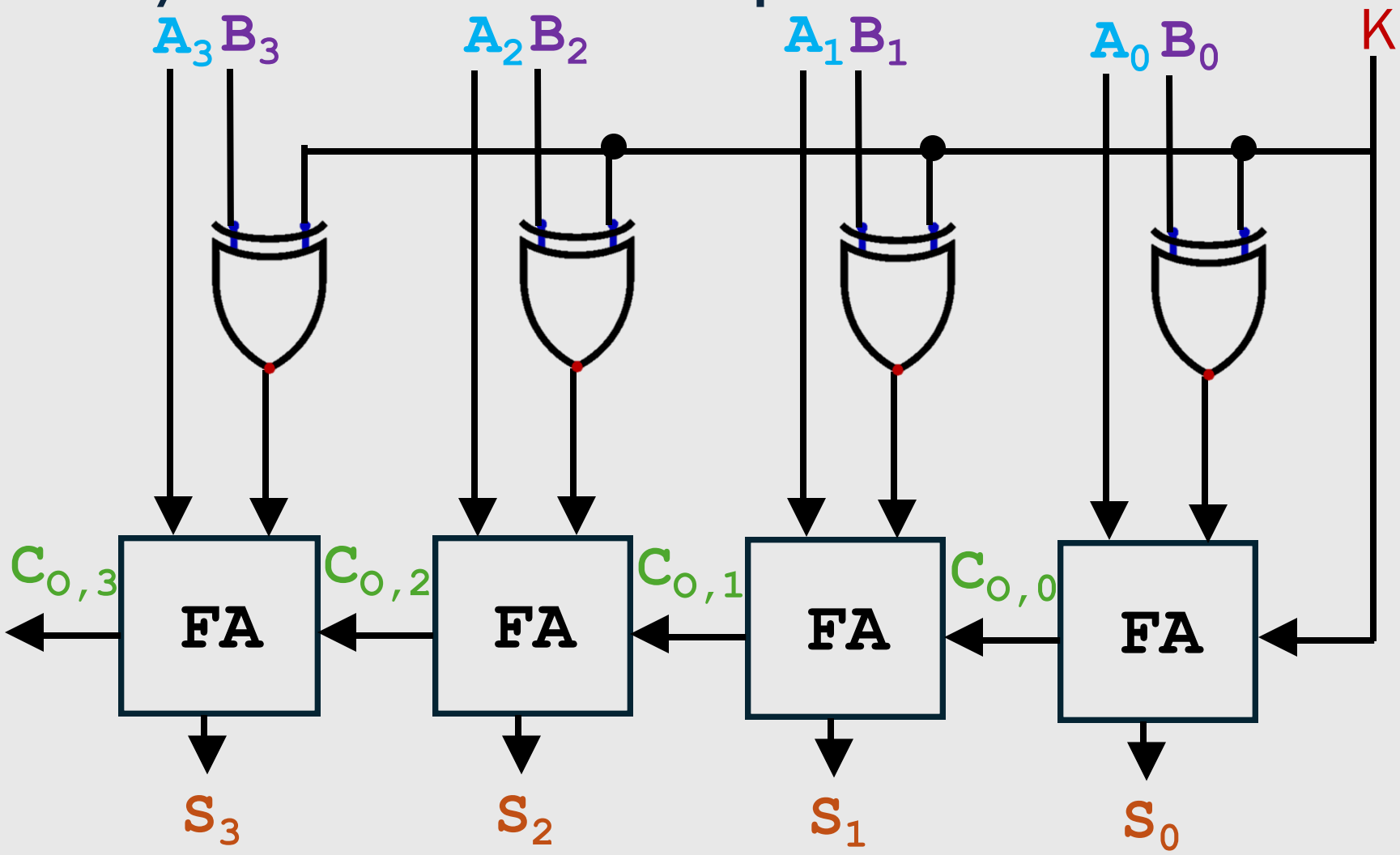
$$N \oplus 0 = N$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

If we XOR B_i with K

- When K = 0, $B_i \oplus K = B_i$
- When K = 1, $B_i \oplus K = \overline{B_i}$

Adder/Subtractor Implementation 2



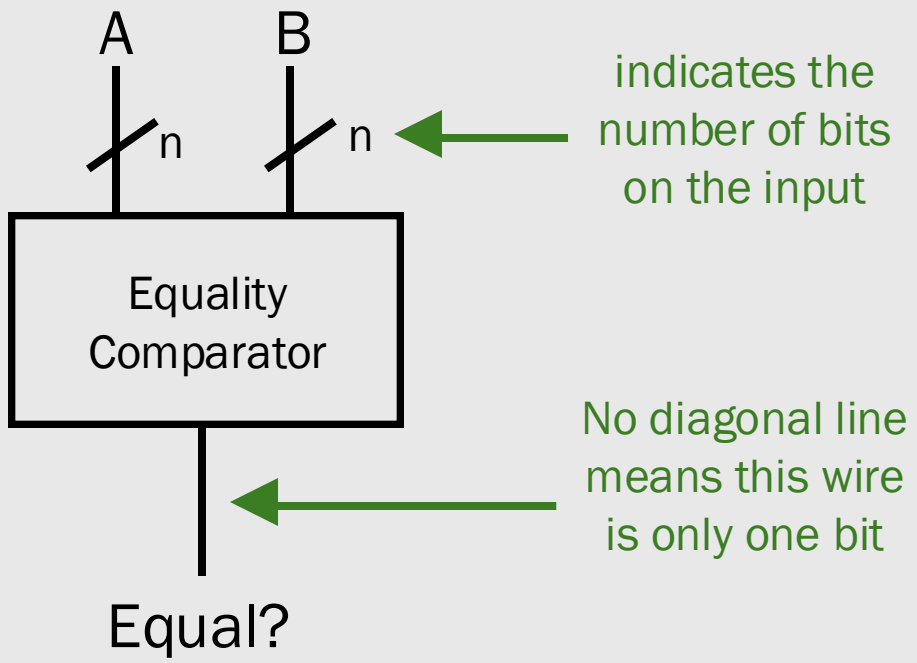
COMPARATOR CIRCUITS

Example Program

```
if (A == B) {
    C = D * 4;
} else {
    C = D – (E + F);
}
```

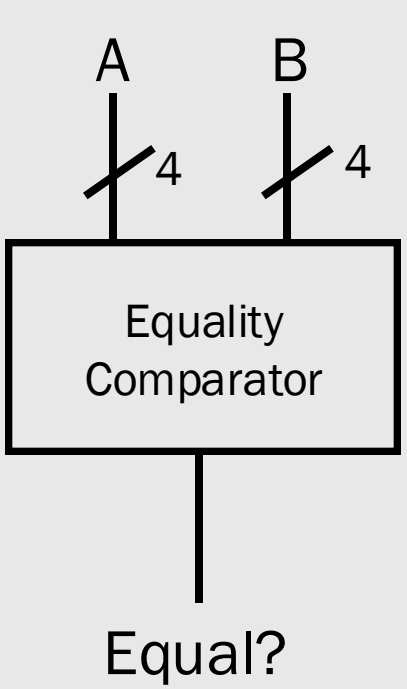
Equality Comparator

Determines whether the two inputs are equal



Equality Comparator

Design a 4-bit equality comparator that will output 1 if the inputs are equal and 0 if the inputs are not equal



A_0

B_0

A_1

B_1

A_2

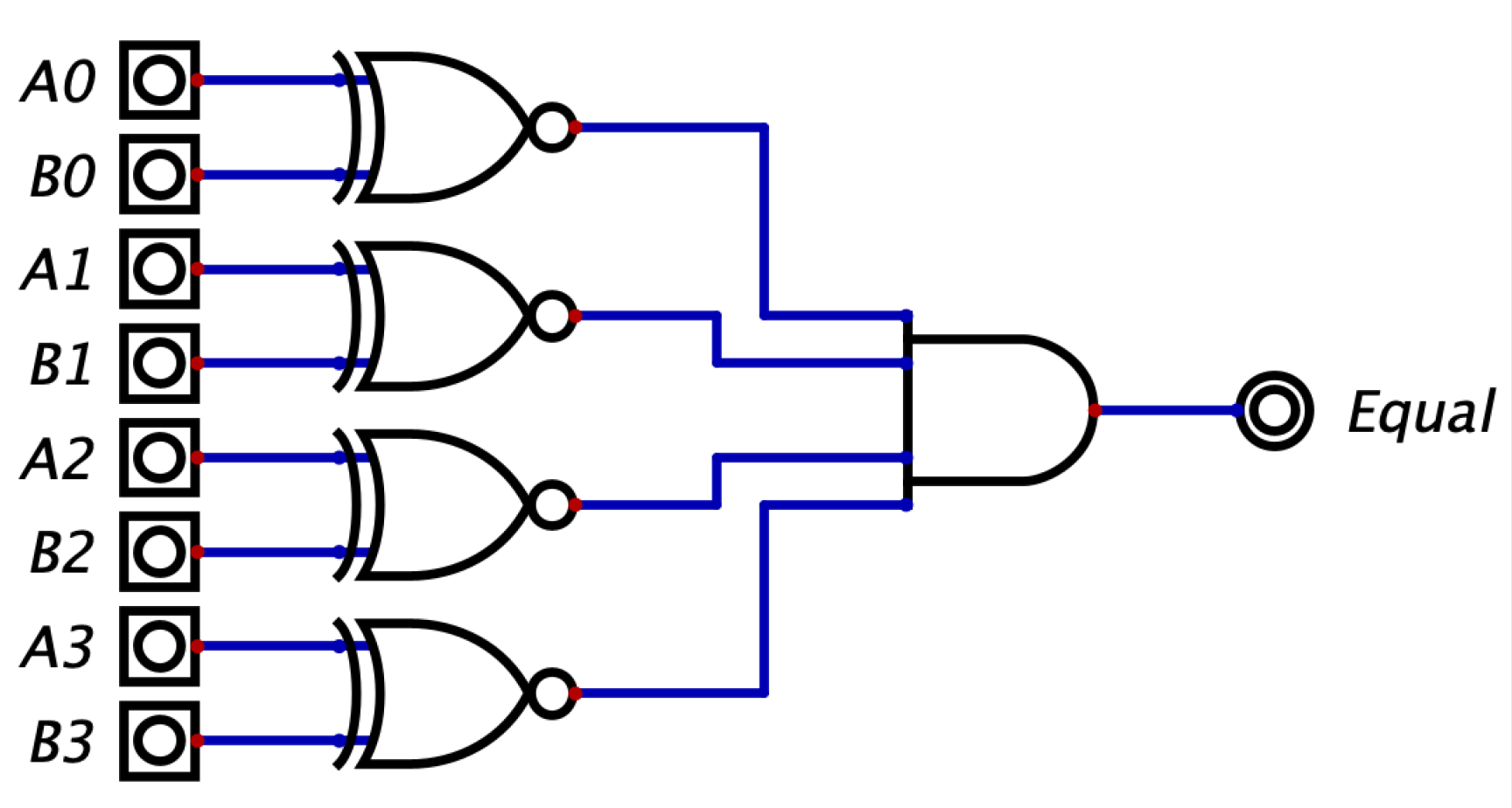
B_2

A_3

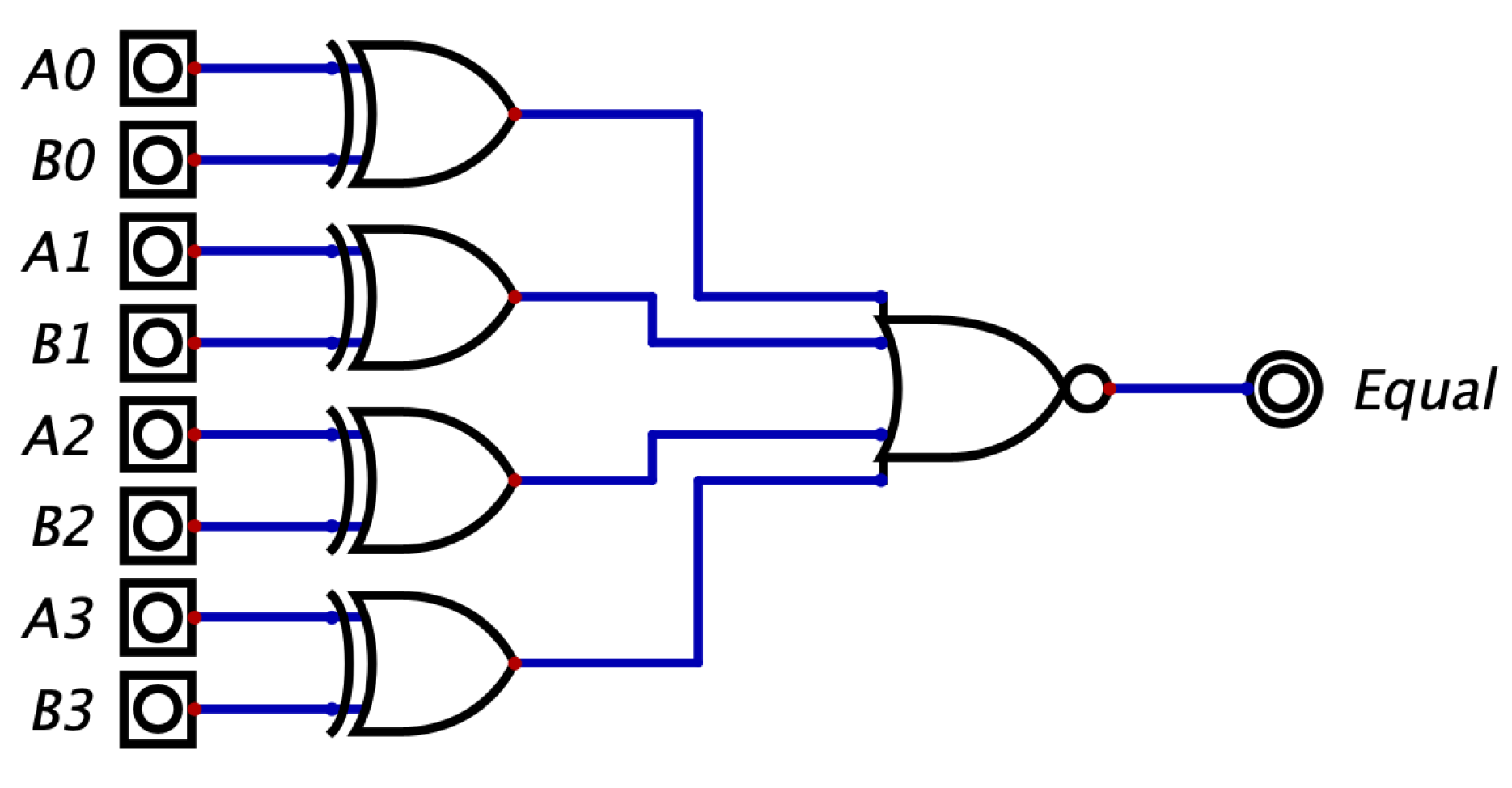
B_3

Equal

Equality Comparator Solution 1



Equality Comparator Solution 2



LEFT SHIFT

Left Shifting

- Let's say my input, A, is a 4-bit number: 0b1101

Shift Amount	Result
0	
1	
2	
3	
4	
5	
6	
7	
8	

Left Shifting

- Let's say my input, A, is a 4-bit number: 0b1101

Shift Amount	Result
0	0b1101
1	0b1010
2	0b0100
3	0b1000
4	0b0000
5	0b0000
6	0b0000
7	0b0000
8	0b0000

Shift Amount

- Let's say my input, A, is a 4-bit number: 0b1101

Shift Amount	Result
0	0b1101
1	0b1010
2	0b0100
3	0b1000
4	0b0000
5	0b0000
6	0b0000
7	0b0000
8	0b0000

If we shift A by more than 3 bits, the result will always be 0. We want to make our shifter circuit as simple as possible, so we will only support a shift amount range of 0-3.

Shift Amount

Shift Amount	Result
0	0b1101
1	0b1010
2	0b0100
3	0b1000
4	0b0000
5	0b0000
6	0b0000
7	0b0000
8	0b0000

We need to be able to shift by up to 3 bits.

To represent the numbers 0-3, we need 2 bits.

Shift Amount

Shift Amount	Result
0	0b10101
1	0b01010
2	0b10100
3	0b01000
4	0b10000
5	0b00000
6	0b00000
7	0b00000
8	0b00000

We need to be able to shift by up to 4 bits.

To represent the numbers 0-4, we need 3 bits.

Shift Amount

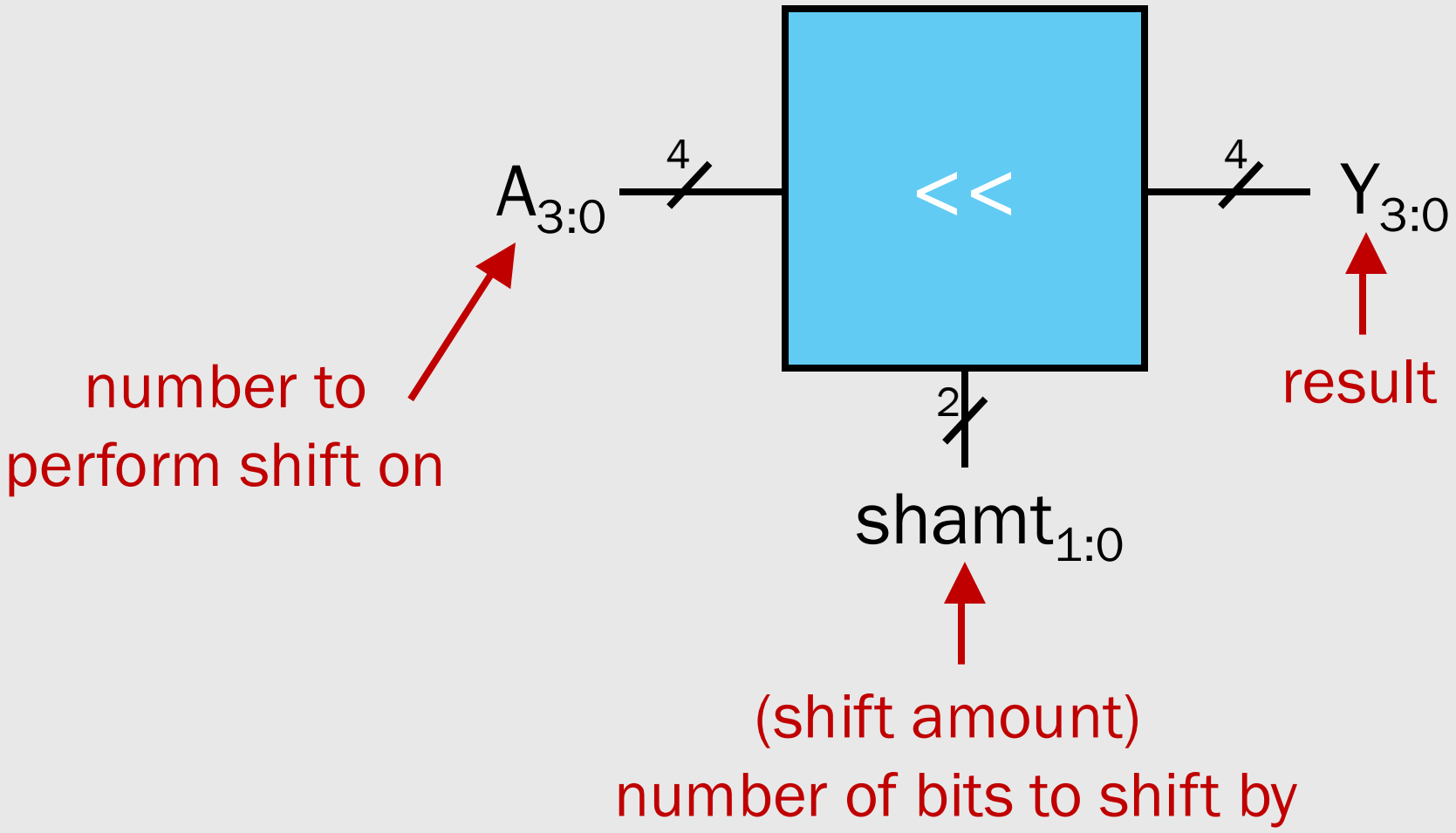
- If I have an n-bit number, how many bits are needed for the shift amount?

Shift Amount

- If I have an n-bit number, how many bits are needed for the shift amount?
  - *ceil(log$_2$n)*

Left Shift

- Design a circuit that can perform a left shift operation on a 4-bit number.



Left Shift

Output Value

Shift Amount	Y_3	Y_2	Y_1	Y_0
0				
1				
2				
3				

Input: $A_3A_2A_1A_0$

Left Shift

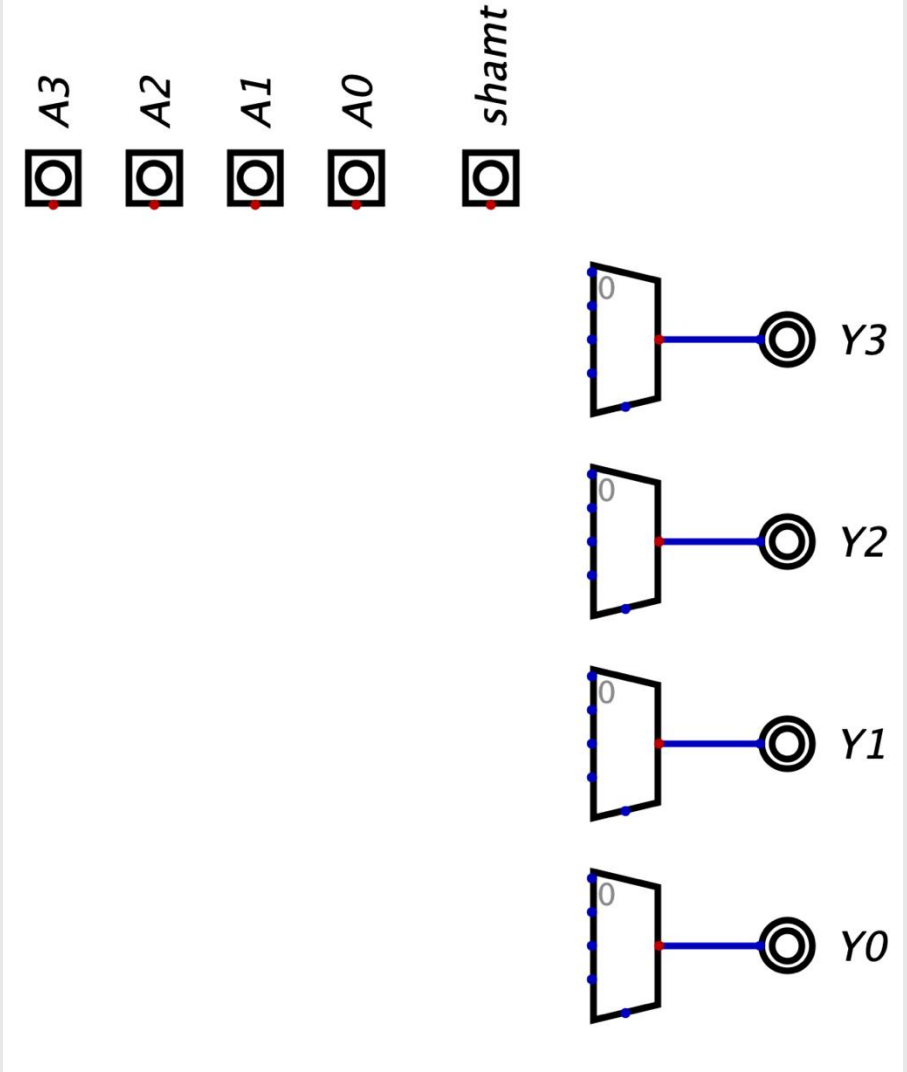
Output Value

Shift Amount

	Y_3	Y_2	Y_1	Y_0
0	A_3	A_2	A_1	A_0
1	A_2	A_1	A_0	0
2	A_1	A_0	0	0
3	A_0	0	0	0

Left Shift

- Design a circuit that can perform a variable left shift on a 4-bit number.
  - *shamt is a 2-bit input*
  - *you may use ground for 0*



Left Shift

- Design a circuit that can perform a variable left shift on a 4-bit number.