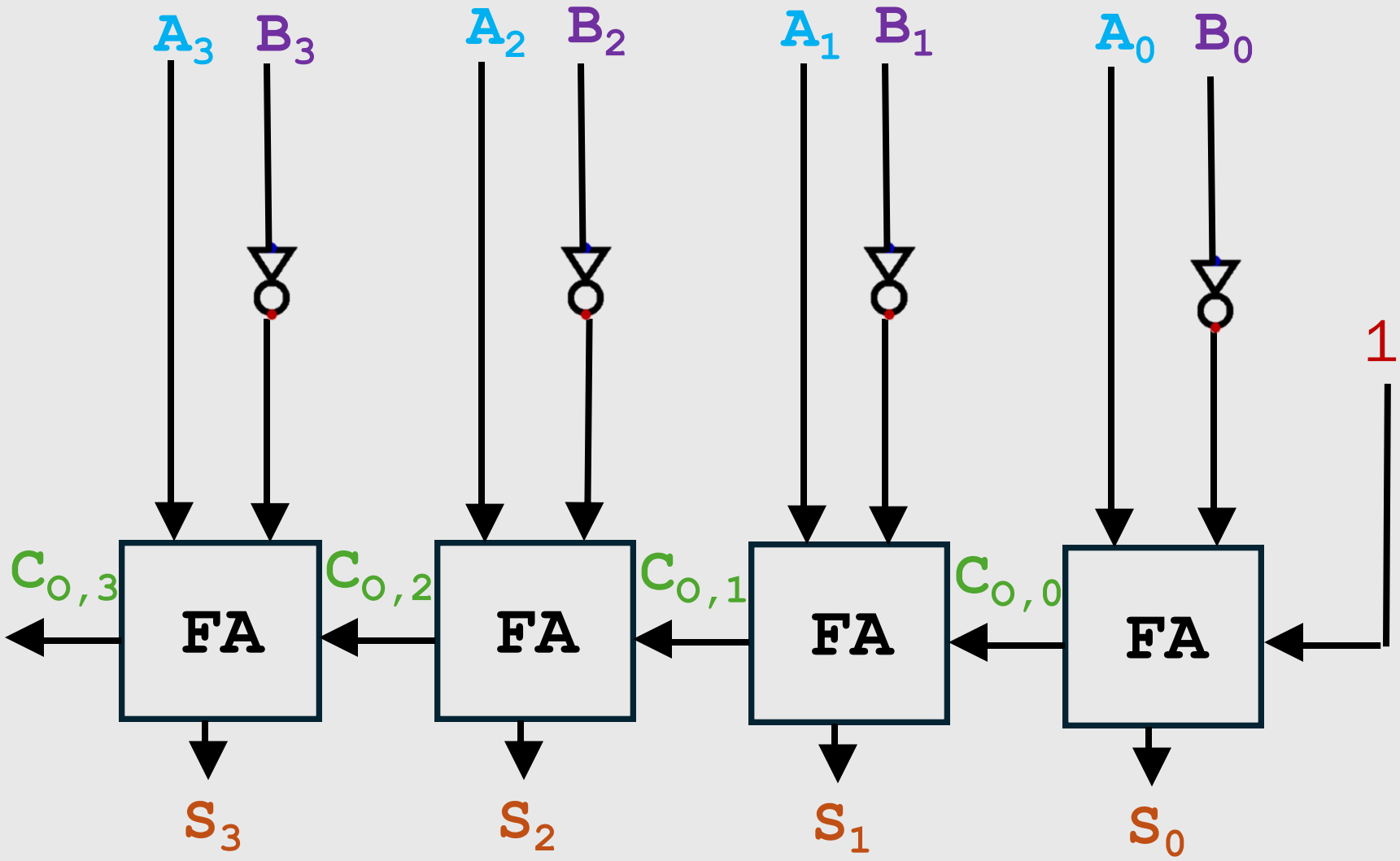
COMP311: *COMPUTER ORGANIZATION!*

Lecture 11: Shifters, Comparators

tinyurl.com/comp311-fa25

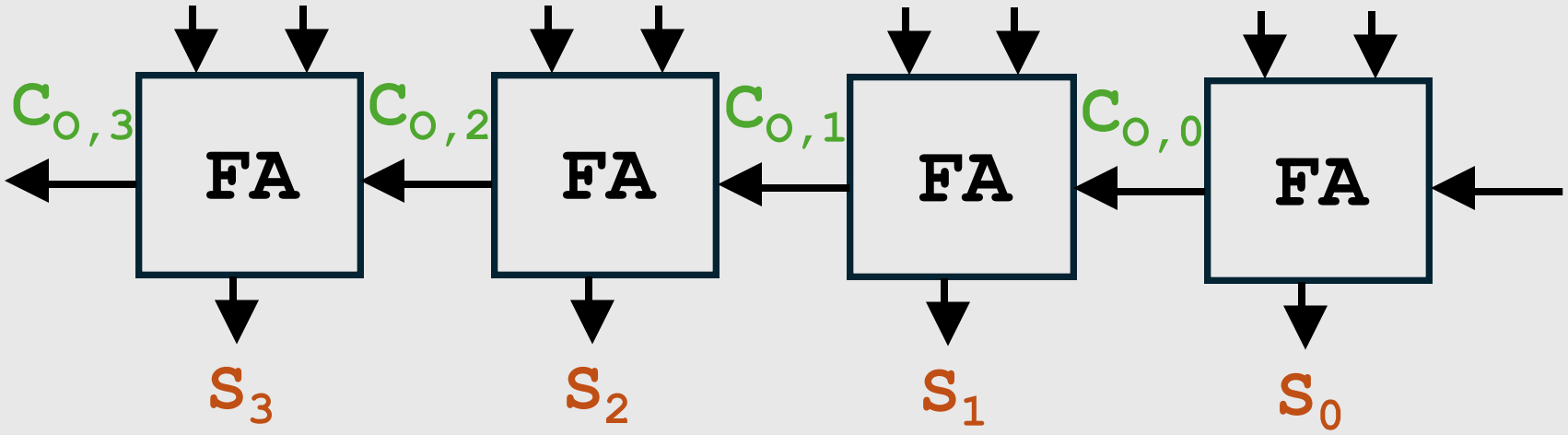
Subtractor



Adder/subtractor

A_3 B_3 A_2 B_2 A_1 B_1 A_0 B_0

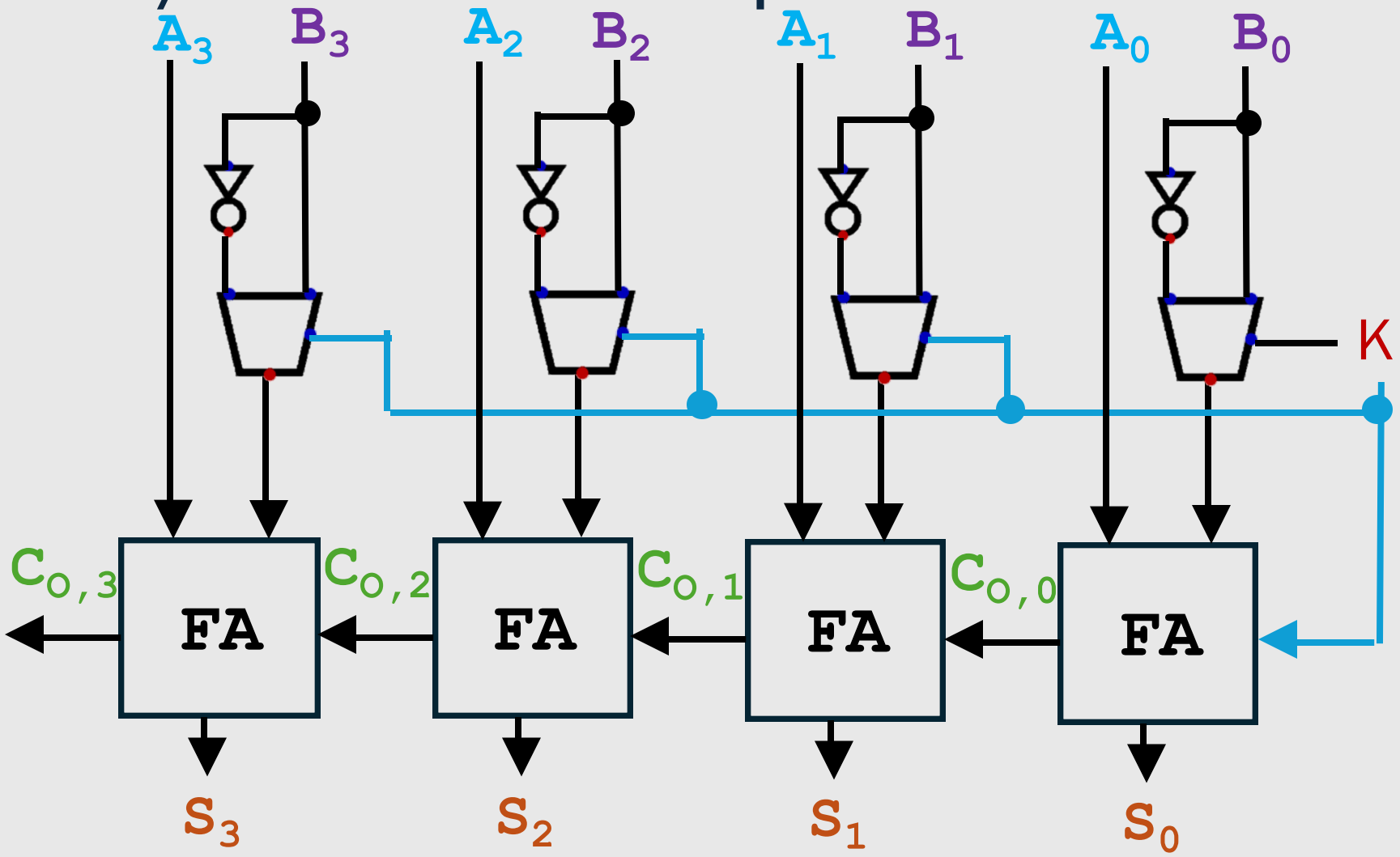
K If K == 0, A + B
If K == 1, A - B



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

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

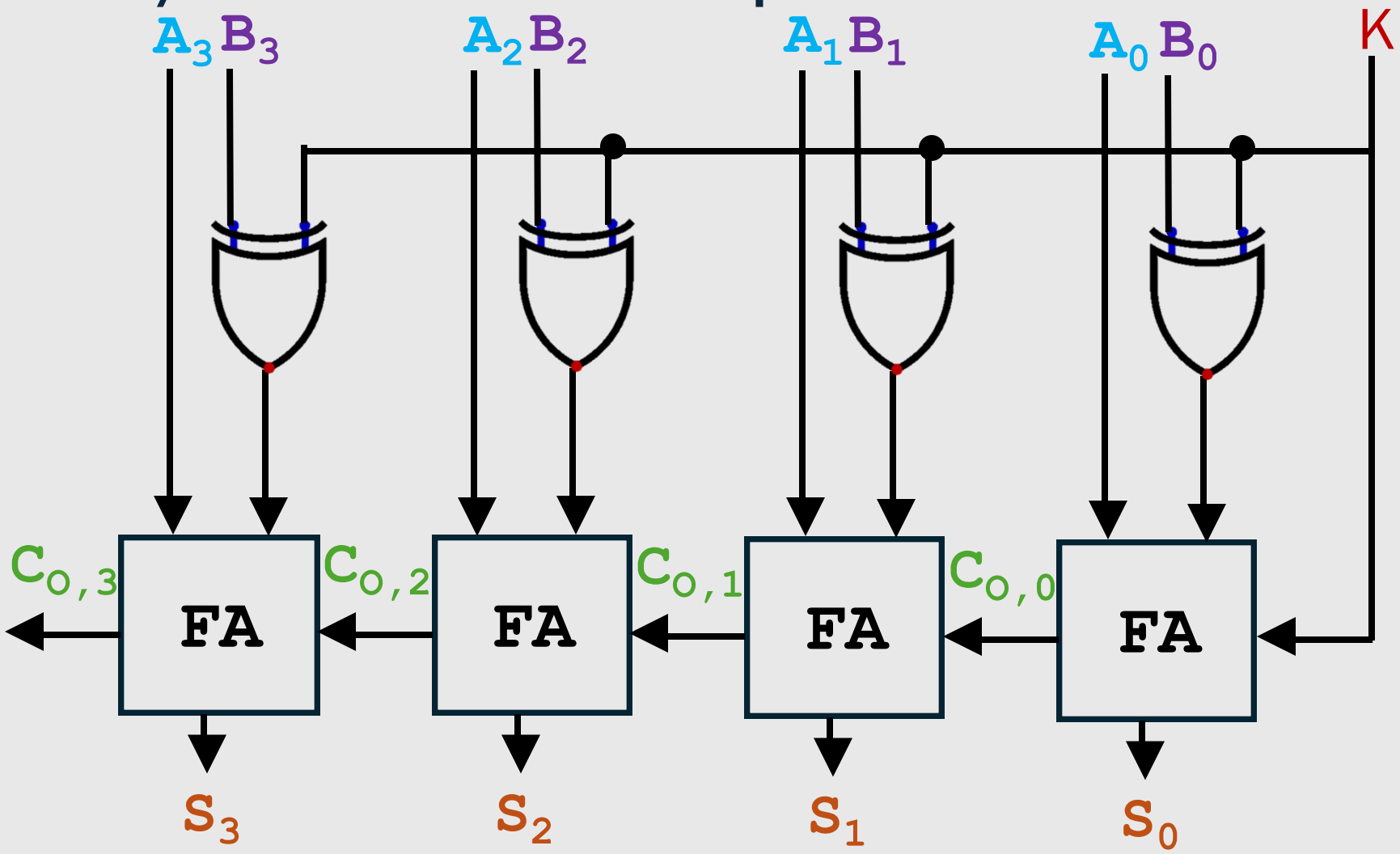
$$N \oplus 0 = N$$

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



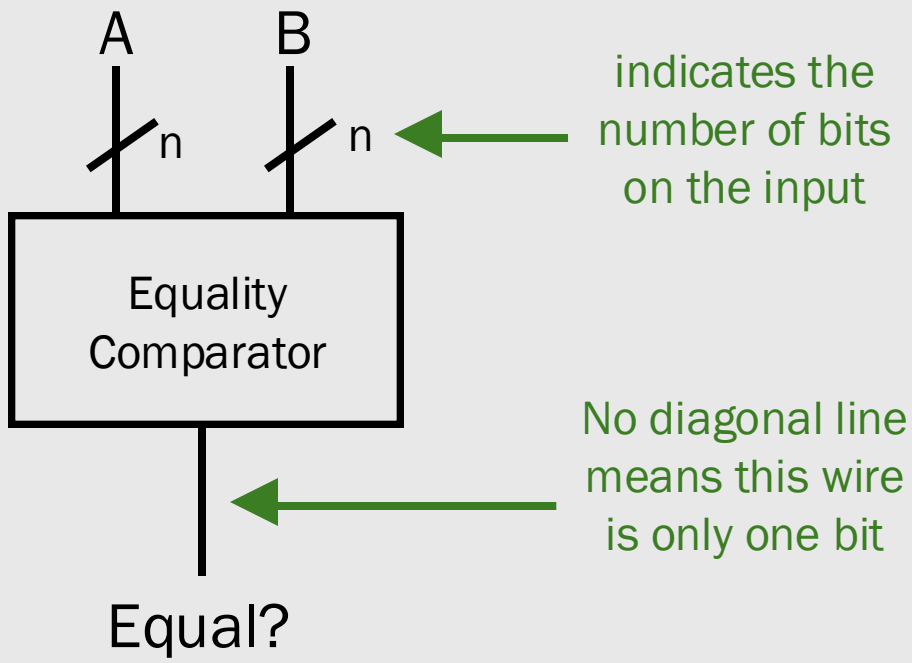
COMPARATOR CIRCUITS

Example Program

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

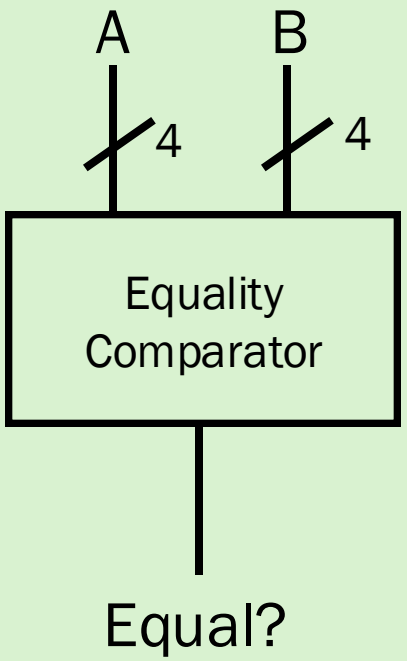
Equality Comparator

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

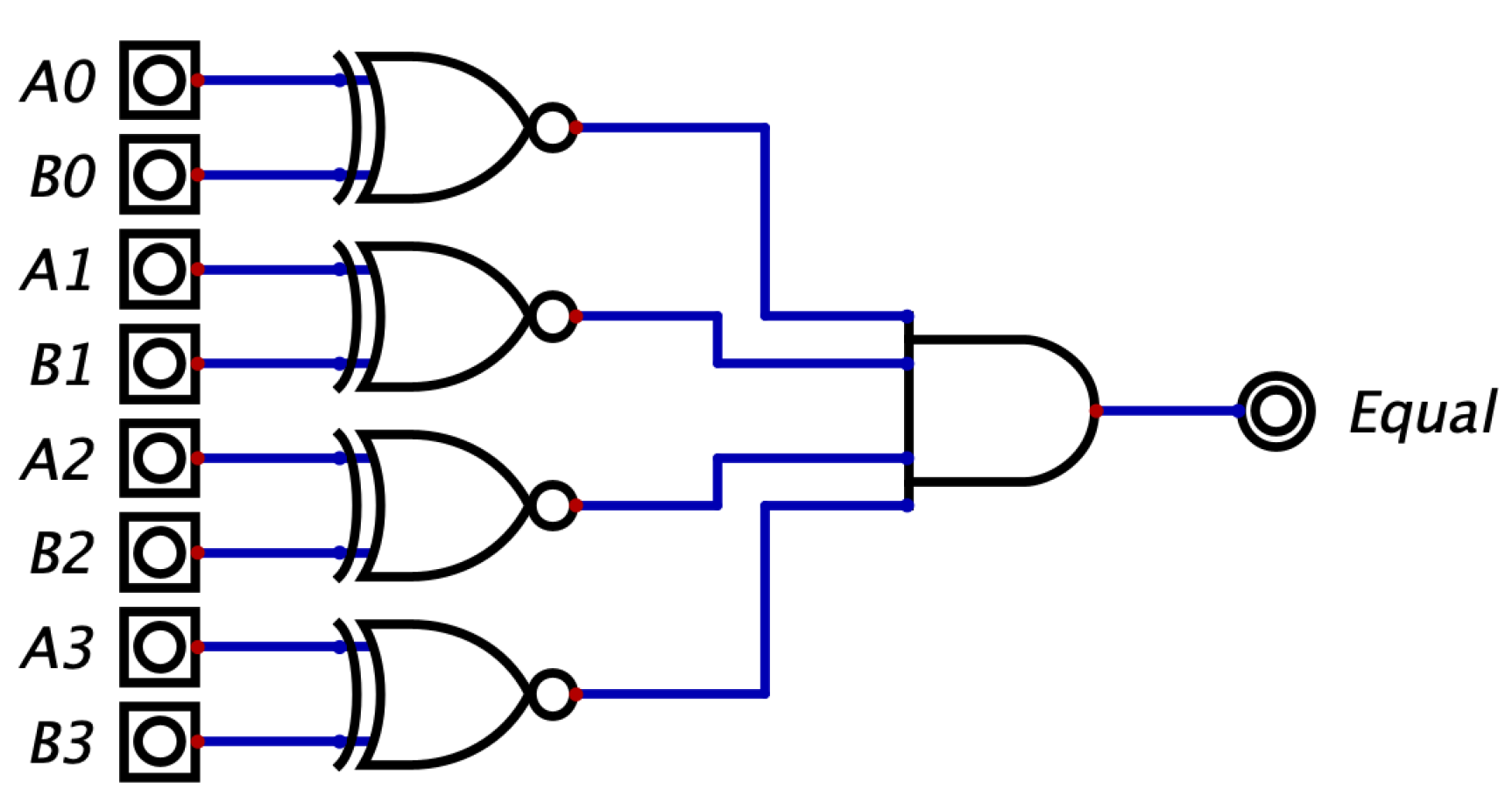
B_2

A_3

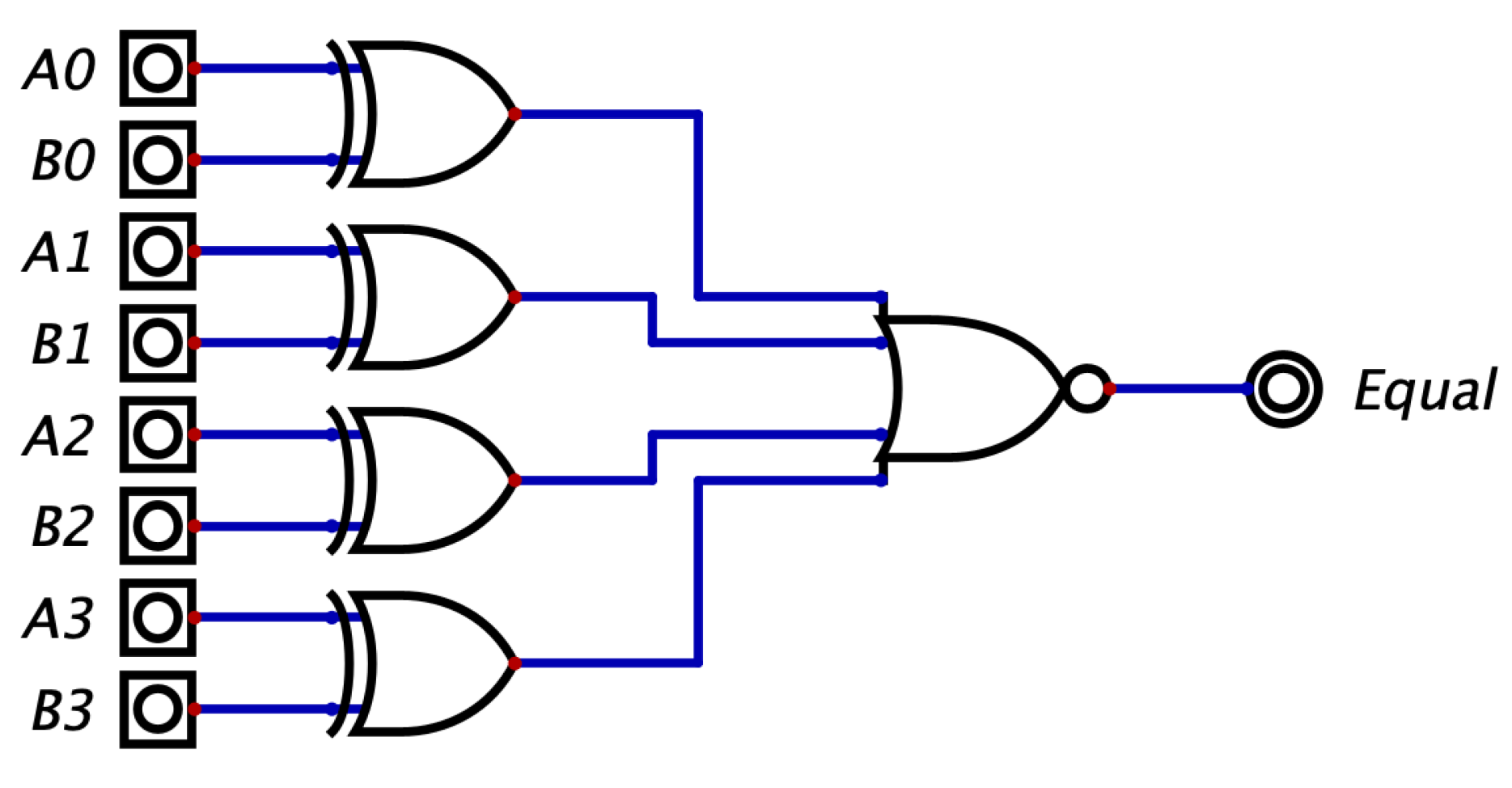
B_3

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

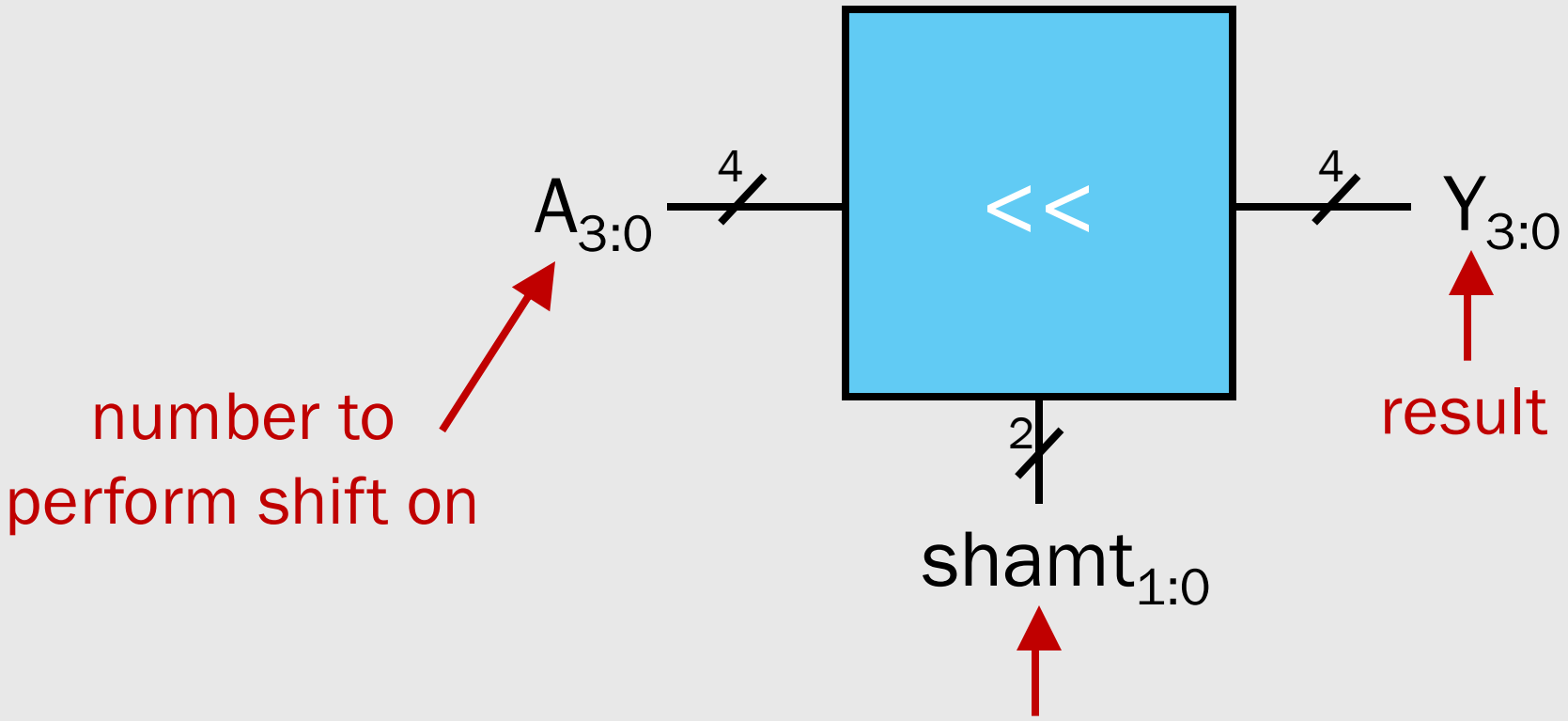
- If I have an n-bit number, how many bits are needed for the shift amount?

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

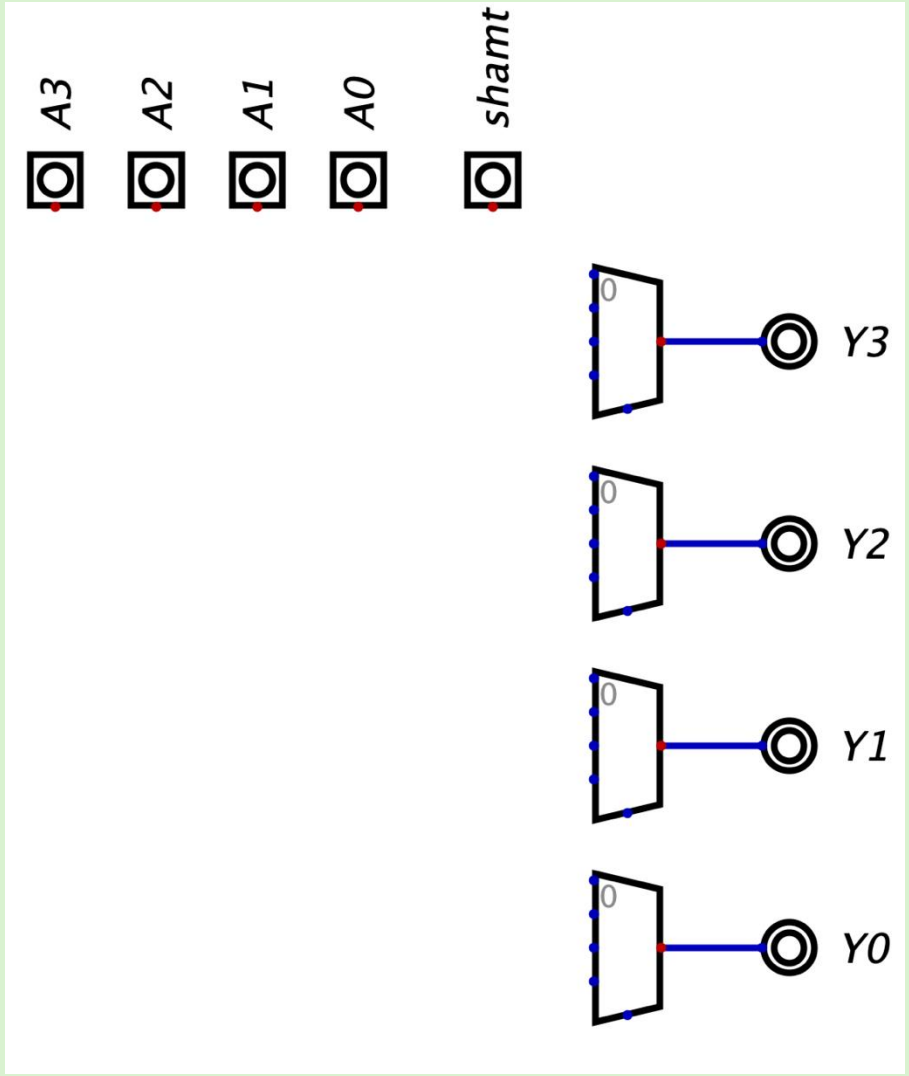
Output Value

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.