

10/14/26



# COMP 311: *Computer Organization!*

**Lecture 15: Intro to Arithmetic Circuits**



The University  
of North Carolina  
at Chapel Hill

<https://tinyurl.com/comp311-fa26>

# Logistics Update

---



- WA3 due on 10/20
- ~~Quiz 3 on 10/21~~
- Lab 2 due on 10/22

Quiz 4 on 10/21

---

# Plan for Class Today

---



- 3:35-4:15: K-Map Definitions
- 4:15-4:20: Break
- 4:20-4:50: K-Map Practice

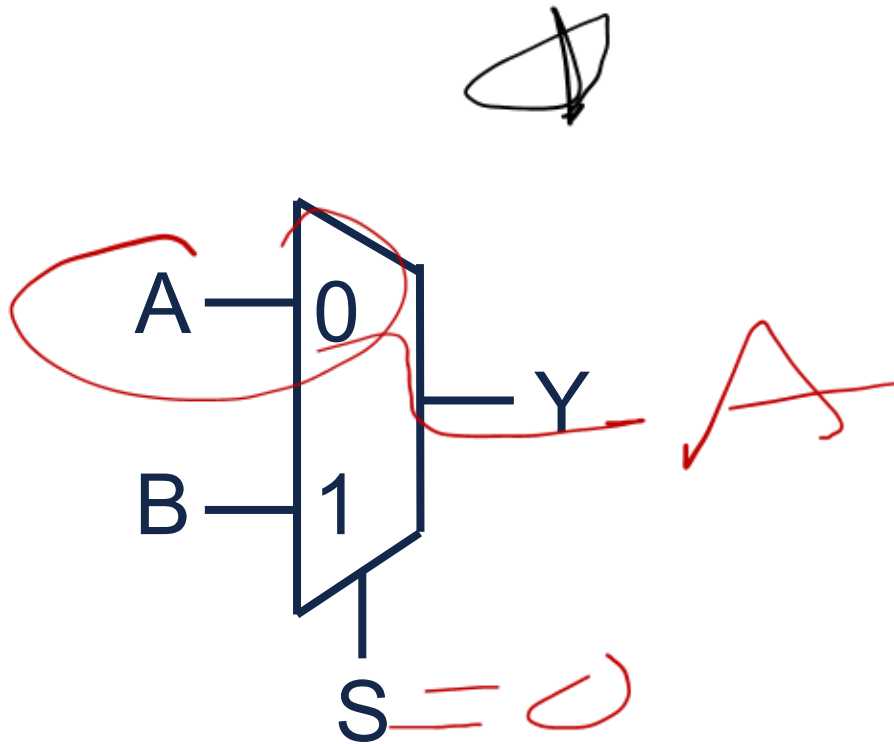
## Guiding Question 😊

How can we simplify Boolean logic without doing a long algebraic derivation?

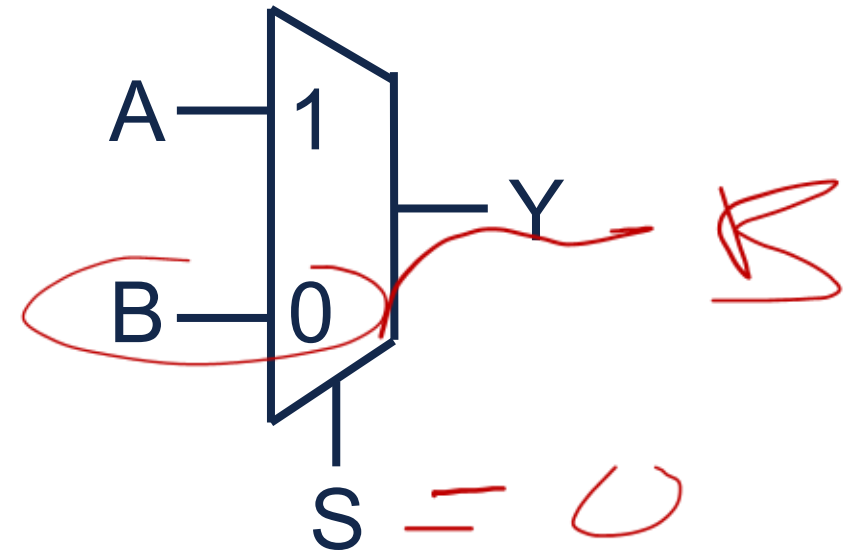


# Multiplexers Review

## 2:1 Multiplexer (2:1 mux)



When  $S = 0$ ,  $Y = A$   
When  $S = 1$ ,  $Y = B$

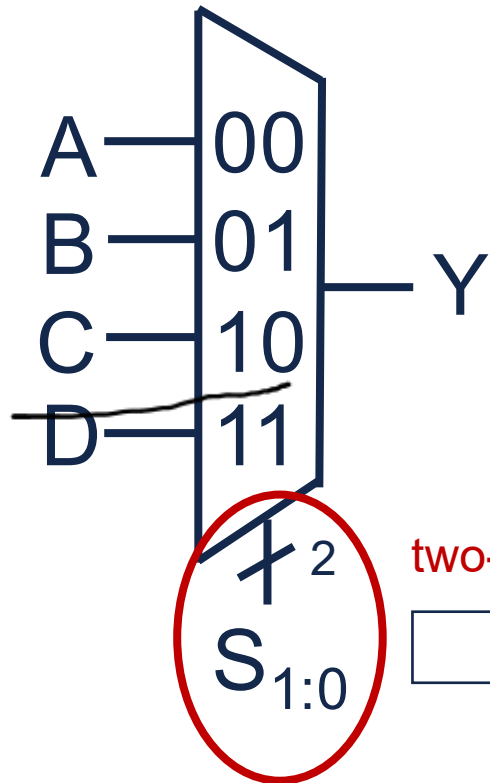


When  $S = \underline{0}$ ,  $Y = B$   
When  $S = 1$ ,  $Y = A$

# 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  $S = \underline{0b00}$ ,  $Y = A$

When  $S = 0b01$ ,  $Y = B$

When  $S = 0b10$ ,  $Y = C$

When  $S = 0b11$ ,  $Y = D$

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



1. Fill in 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 | X | X | X |   |
|    |    | 1 | X | X | X |   |
|    |    | X | 0 | X | X |   |
|    |    | X | 1 | X | X |   |
| 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 |

# 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 |

# 4:1 Multiplexer (4:1 mux)



Equation

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

Diagram

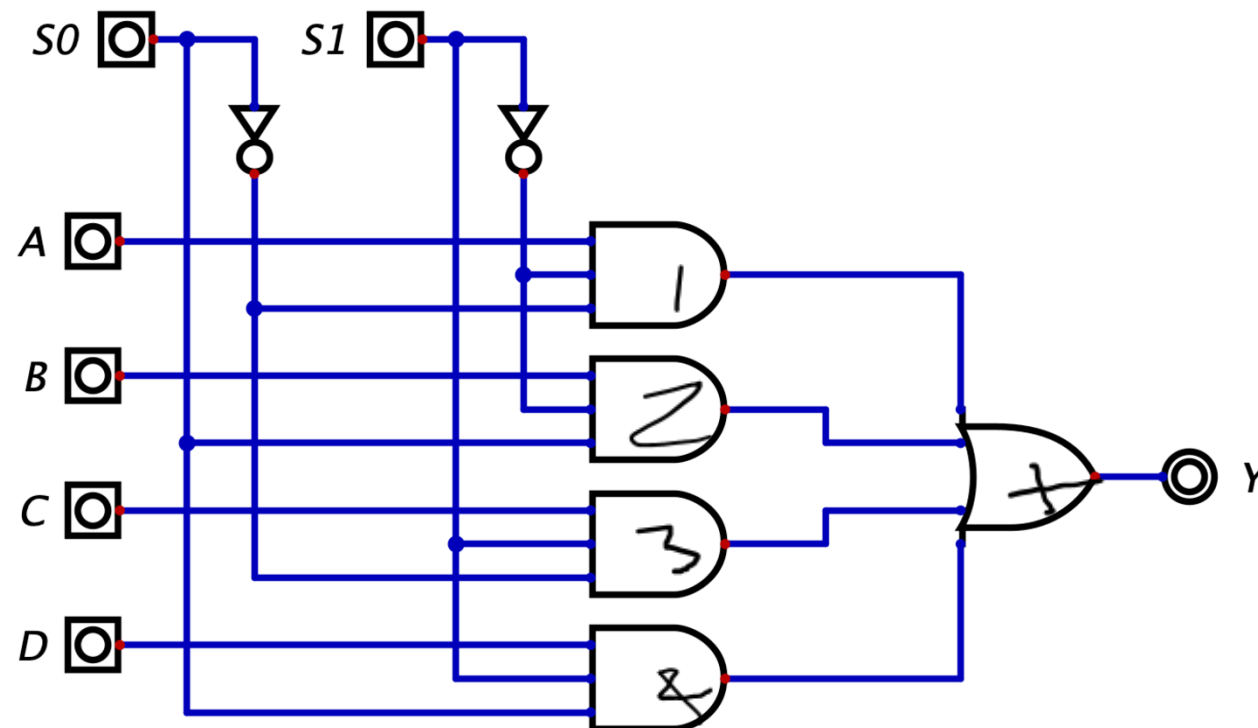
# 4:1 Multiplexer (4:1 mux)



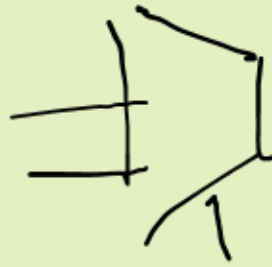
Equation

$$Y = A\bar{S}_1\bar{S}_0 + B\bar{S}_1S_0 + CS_1\bar{S}_0 + DS_1S_0$$

Diagram



# Multiplexers



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

A

B

C

D

$S_1$

$S_0$

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

Y

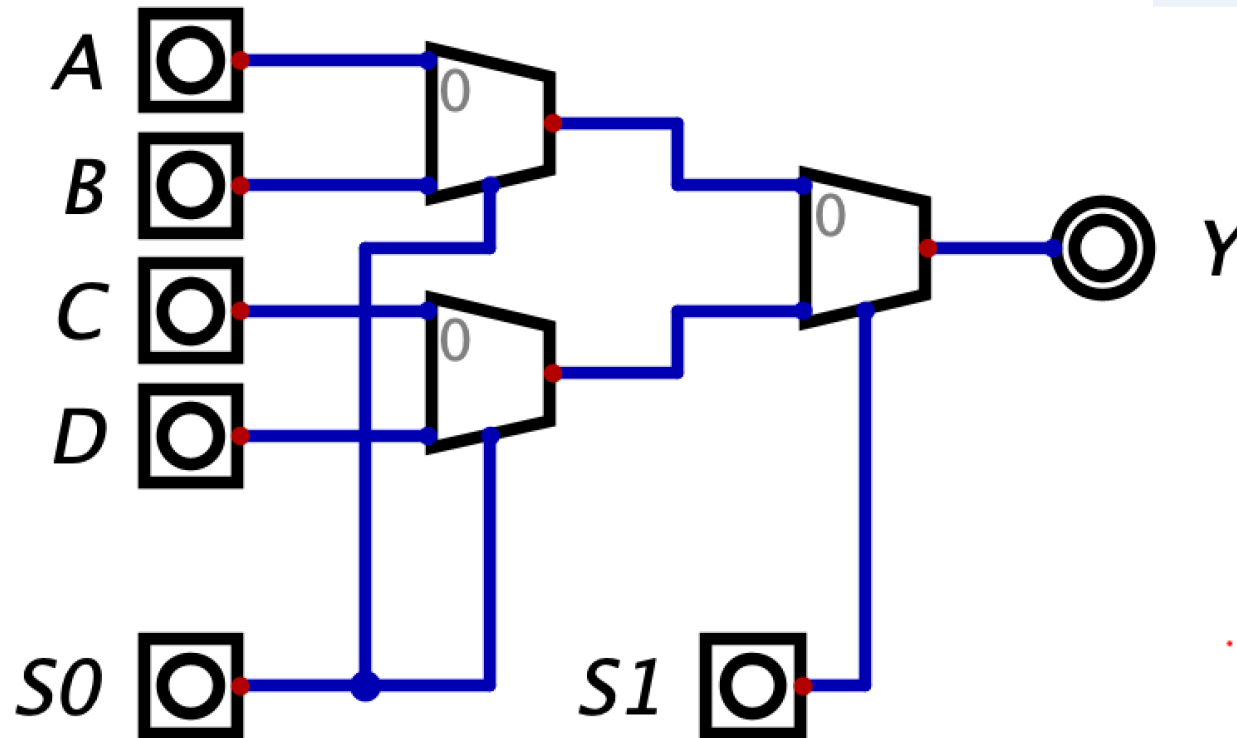
I

# Multiplexers

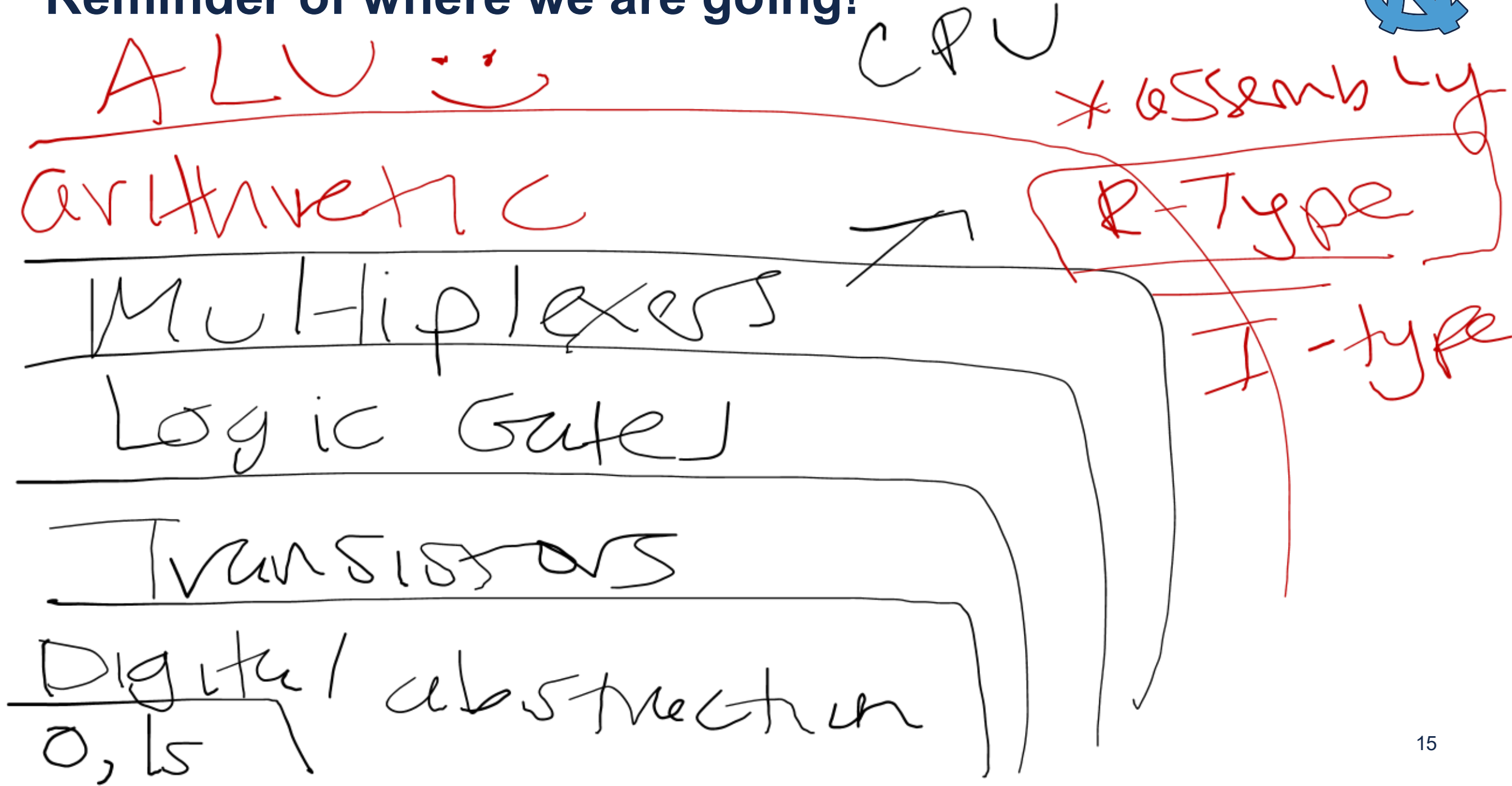
$S = 11$

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

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



# Reminder of where we are going!





# Priority Encoders

# Multiple Output Circuits



| A | B | C | $Y_1$ | $Y_0$ |
|---|---|---|-------|-------|
| 0 | 0 | 0 | 1     | 1     |
| 0 | 0 | 1 | 1     | 0     |
| 0 | 1 | 0 | 1     | 0     |
| 0 | 1 | 1 | 1     | 0     |
| 1 | 0 | 0 | 0     | 1     |
| 1 | 0 | 1 | 0     | 0     |
| 1 | 1 | 0 | 0     | 0     |
| 1 | 1 | 1 | 0     | 1     |

# Multiple Output Circuits



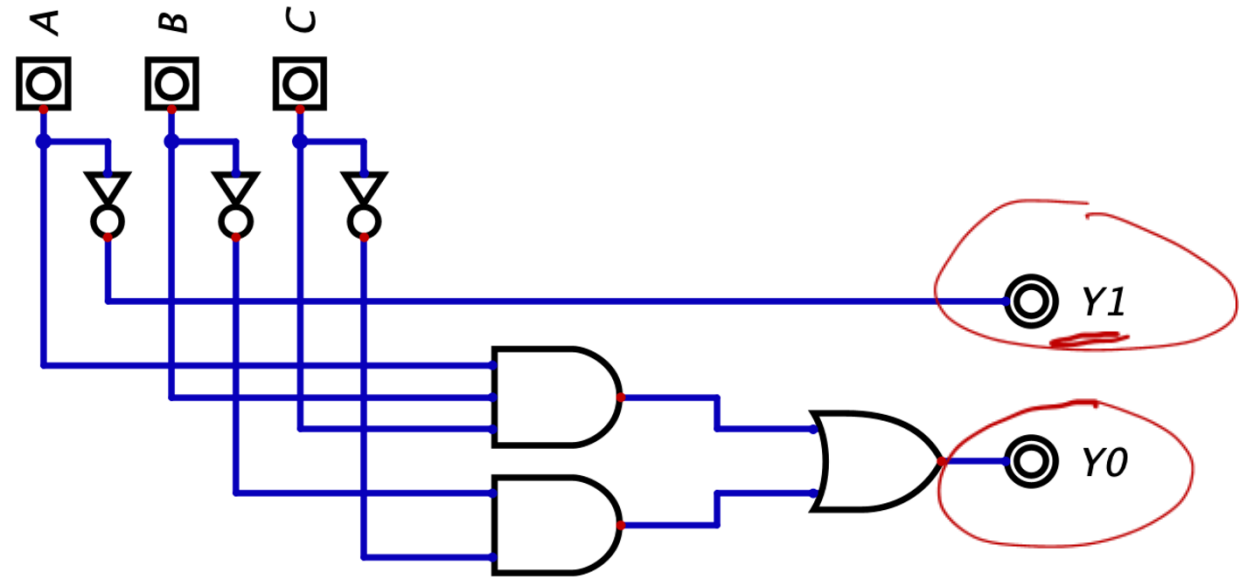
| A | B | C | $Y_1$ | $Y_0$ |
|---|---|---|-------|-------|
| 0 | 0 | 0 | 1     | 1     |
| 0 | 0 | 1 | 1     | 0     |
| 0 | 1 | 0 | 1     | 0     |
| 0 | 1 | 1 | 1     | 0     |
| 1 | 0 | 0 | 0     | 1     |
| 1 | 0 | 1 | 0     | 0     |
| 1 | 1 | 0 | 0     | 0     |
| 1 | 1 | 1 | 0     | 1     |

$$Y_1 = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC$$

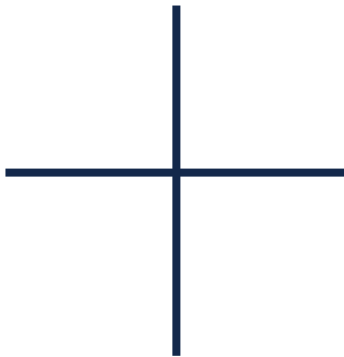
$$Y_1 = \bar{A}$$

$$Y_0 = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + ABC$$

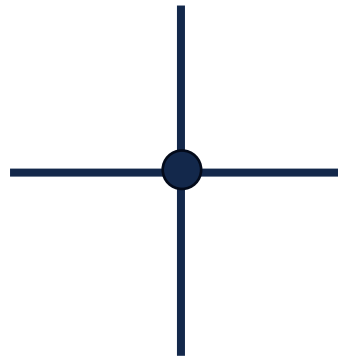
$$Y_0 = \bar{B}\bar{C} + ABC$$



# Wire Crossing Notation



Not connected



Connected



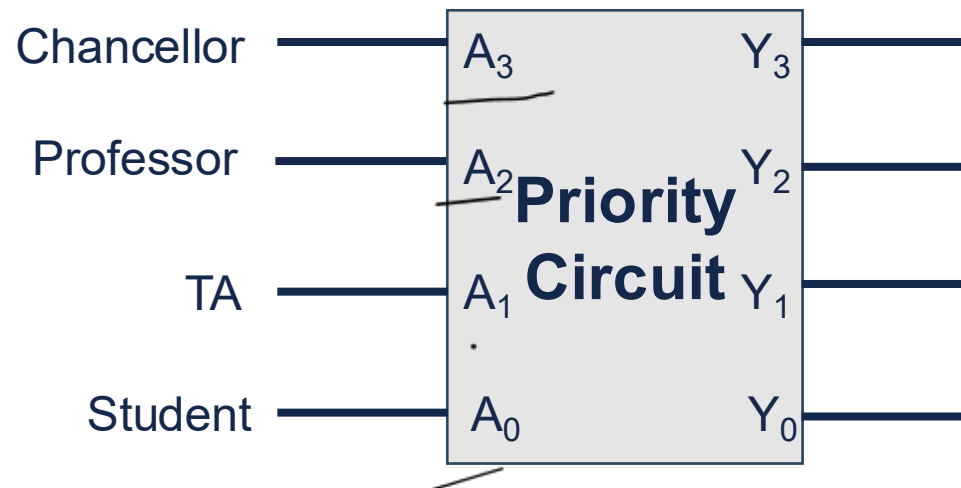
Connected



# Priority Circuit



I want to design a circuit that will determine who can reserve a classroom space. Anyone who wants to reserve the room will assert their input (set their input to 1). The order of priority for room reservations is the chancellor, professors, TAs, and students. Assume that only one person from each category will submit a request at a given time. The requestor with the highest priority will get the room reservation.



The output that is 1 will tell us who gets the room reservation (only one output will be 1 at any time)

# Circuit Design Activity



| A <sub>3</sub> | A <sub>2</sub> | A <sub>1</sub> | A <sub>0</sub> | Y <sub>3</sub> | Y <sub>2</sub> | Y <sub>1</sub> | Y <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 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              |                |                |                |                |

1. Fill in the truth table
2. Write down the simplified SOP equation
3. Draw the circuit diagram



# Don't Cares in the Priority Circuit

Whenever  $A_3 = 1$ ...

- $Y_3 = 1$
- $Y_2 = 0$
- $Y_1 = 0$
- $Y_0 = 0$

It does not matter  
and  $A_0$  are!

We can say  
“**don't care**” at  
values of  $A_2$ ,  $A_1$ , and  $A_0$

***Don't-cares can be used to simplify logic!***

For input combinations that never occur, the designer may assume either 1 or 0, whichever is more useful to achieving a minimum-cost implementation.

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 0     | 0     | 1     | 0     | 0     | 0     | 1     |
| 0     | 0     | 1     | 0     | 0     | 0     | 1     | 0     |
| 0     | 0     | 1     | 1     | 0     | 0     | 1     | 1     |
| 0     | 1     | 0     | 0     | 0     | 1     | 0     | 0     |
| 0     | 1     | 0     | 1     | 0     | 1     | 0     | 1     |
| 0     | 1     | 1     | 0     | 0     | 1     | 1     | 0     |
| 0     | 1     | 1     | 1     | 0     | 1     | 1     | 1     |
| 1     | 0     | 0     | 0     | 1     | 0     | 0     | 0     |
| 1     | 0     | 0     | 1     | 1     | 0     | 0     | 0     |
| 1     | 0     | 1     | 0     | 1     | 0     | 0     | 0     |
| 1     | 0     | 1     | 1     | 1     | 0     | 0     | 0     |
| 1     | 1     | 0     | 0     | 1     | 1     | 0     | 0     |
| 1     | 1     | 0     | 1     | 1     | 1     | 0     | 0     |
| 1     | 1     | 1     | 0     | 1     | 1     | 1     | 0     |
| 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     |

✓  
X  
11



# Don't Cares in the Priority Circuit

Whenever  $A_3 = 1$ ...

- $Y_3 = 1$
- $Y_2 = 0$
- $Y_1 = 0$
- $Y_0 = 0$

It does not matter what  $A_2$ ,  $A_1$ , and  $A_0$  are!

We can say that we  
“**don't care**” about the  
values of  $A_2$ ,  $A_1$ , and  $A_0$

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 0     | 0     | 1     | 0     | 0     | 0     | 1     |
| 0     | 0     | 1     | 0     | 0     | 0     | 1     | 0     |
| 0     | 0     | 1     | 1     | 0     | 0     | 1     | 0     |
| 0     | 1     | 0     | 0     | 0     | 1     | 0     | 0     |
| 0     | 1     | 0     | 1     | 0     | 1     | 0     | 0     |
| 0     | 1     | 1     | 0     | 0     | 1     | 0     | 0     |
| 0     | 1     | 1     | 1     | 0     | 1     | 0     | 0     |
| 1     | X     | X     | X     | 1     | 0     | 0     | 0     |



# Don't Cares in the Priority Circuit

Whenever  $A_3 = 0$  and  $A_2 = 1$

- $Y_3 = 0$
- $Y_2 = 1$
- $Y_1 = 0$
- $Y_0 = 0$

It does not matter what  $A_1$ , and  $A_0$  are

We can say that we  
“**don't care**” about the  
values of  $A_1$  and  $A_0$

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 0     | 0     | 1     | 0     | 0     | 0     | 1     |
| 0     | 0     | 1     | 0     | 0     | 0     | 1     | 0     |
| 0     | 0     | 1     | 1     | 0     | 0     | 1     | 0     |
| 0     | 1     | 0     | 0     | 0     | 1     | 0     | 0     |
| 0     | 1     | 0     | 1     | 0     | 1     | 0     | 0     |
| 0     | 1     | 1     | 0     | 0     | 1     | 0     | 0     |
| 0     | 1     | 1     | 1     | 0     | 1     | 0     | 0     |
| 1     | X     | X     | X     | 1     | 0     | 0     | 0     |



# Don't Cares in the Priority Circuit

Whenever  $A_3 = 0$  and  $A_2 = 1$

- $Y_3 = 0$
- $Y_2 = 1$
- $Y_1 = 0$
- $Y_0 = 0$

It does not matter what  $A_1$ , and  $A_0$  are

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 0     | 0     | 1     | 0     | 0     | 0     | 1     |
| 0     | 0     | 1     | 0     | 0     | 0     | 1     | 0     |
| 0     | 0     | 1     | 1     | 0     | 0     | 1     | 0     |
| 0     | 1     | X     | X     | 0     | 1     | 0     | 0     |
| 1     | X     | X     | X     | 1     | 0     | 0     | 0     |

We can say that we  
“**don't care**” about the  
values of  $A_1$  and  $A_0$



# Don't Cares in the Priority Circuit

Whenever  $A_3 = 0$ ,  $A_2 = 0$ , and  $A_1 = 1$

- $Y_3 = 0$
- $Y_2 = 0$
- $Y_1 = 1$
- $Y_0 = 0$

It does not matter what  $A_0$  is

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 0     | 0     | 1     | 0     | 0     | 0     | 1     |
| 0     | 0     | 1     | 0     | 0     | 0     | 1     | 0     |
| 0     | 0     | 1     | 1     | 0     | 0     | 1     | 0     |
| 0     | 1     | X     | X     | 0     | 1     | 0     | 0     |
| 1     | X     | X     | X     | 1     | 0     | 0     | 0     |

We can say that we  
“**don't care**” about the  
value  $A_0$



# Don't Cares in the Priority Circuit

Whenever  $A_3 = 0$ ,  $A_2 = 0$ , and  $A_1 = 1$

- $Y_3 = 0$
- $Y_2 = 0$
- $Y_1 = 1$
- $Y_0 = 0$

It does not matter what  $A_0$  is

| $A_3$ | $A_2$ | $A_1$ | $A_0$ | $Y_3$ | $Y_2$ | $Y_1$ | $Y_0$ |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| 0     | 0     | 0     | 1     | 0     | 0     | 0     | 1     |
| 0     | 0     | 1     | X     | 0     | 0     | 1     | 0     |
| 0     | 1     | X     | X     | 0     | 1     | 0     | 0     |
| 1     | X     | X     | X     | 1     | 0     | 0     | 0     |

We can say that we  
“don’t care” about the  
value  $A_0$

# Boolean Equations from Don't Cares



| A <sub>3</sub> | A <sub>2</sub> | A <sub>1</sub> | A <sub>0</sub> | Y <sub>3</sub> | Y <sub>2</sub> | Y <sub>1</sub> | Y <sub>0</sub> |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| 0              | 0              | 0              | 0              | 0              | 0              | 0              | 0              |
| 0              | 0              | 0              | 1              | 0              | 0              | 0              | 1              |
| 0              | 0              | 1              | X              | 0              | 0              | 1              | 0              |
| 0              | 1              | X              | X              | 0              | 1              | 0              | 0              |
| 1              | X              | X              | X              | 1              | 0              | 0              | 0              |

$$\begin{aligned}Y_3 &= \overline{A_3} \\Y_2 &= \overline{A_3} \overline{A_2} \\Y_1 &= \overline{A_3} \overline{A_2} \overline{A_1} \\Y_0 &= \overline{A_3} \overline{A_2} \overline{A_1} A_0\end{aligned}$$

Transforming the inputs to don't cares makes finding the simplified SOP equations much easier!

# Output Values Can Also Be Don't Care!



| A | B | Y        |
|---|---|----------|
| 0 | 0 | 1        |
| 0 | 1 | <u>X</u> |
| 1 | 0 | 0        |
| 1 | 1 | 0        |

When A is 0 and and  
B is 1, the output can  
be 0 or 1!

# Output Values Can Also Be Don't Care!



There are two possible implementations of this truth table

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | X |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

# Output Values Can Also Be Don't Care!



There are two possible implementations of this truth table

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Since the output of row1 can be either a 0 or 1, let's see what happens if we treat the output as 0.

# Output Values Can Also Be Don't Care!

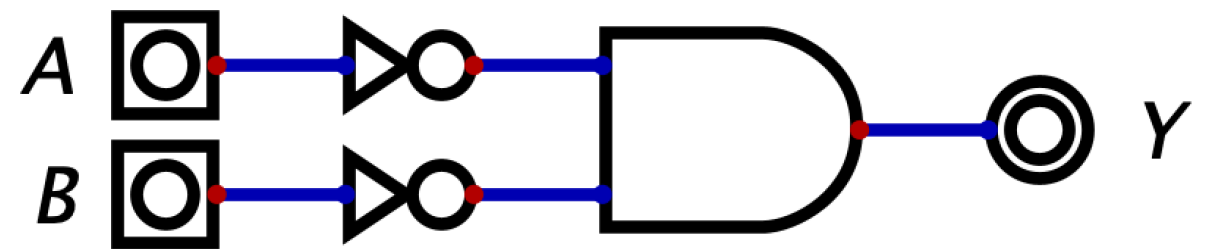


There are two possible implementations of this truth table

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

If we treat the X as a 0, then we get the following circuit:

$$Y = \bar{A}\bar{B}$$



# Output Values Can Also Be Don't Care!



There are two possible implementations of this truth table

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Since the output of row1 can be either a 0 or 1, let's see what happens if we treat the output as **1**.

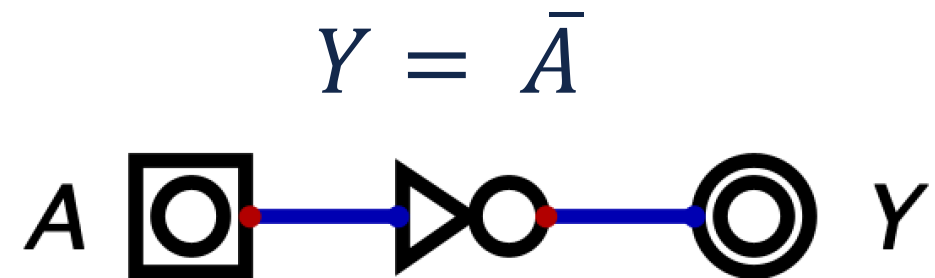
# Output Values Can Also Be Don't Care!



There are two possible implementations of this truth table

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

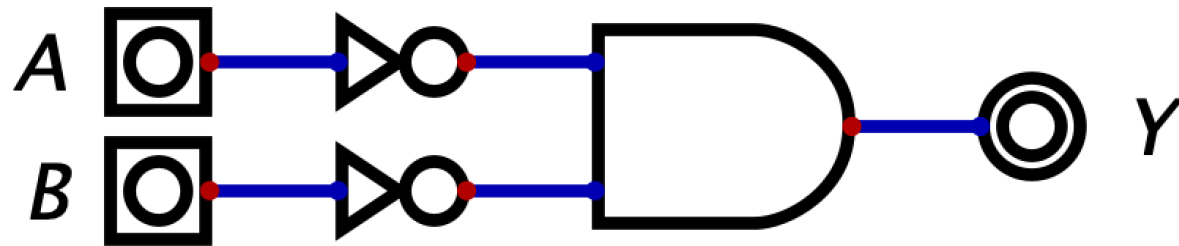
If we treat the X as a 1, then we get the following circuit:



# Which circuit should I choose for my design?



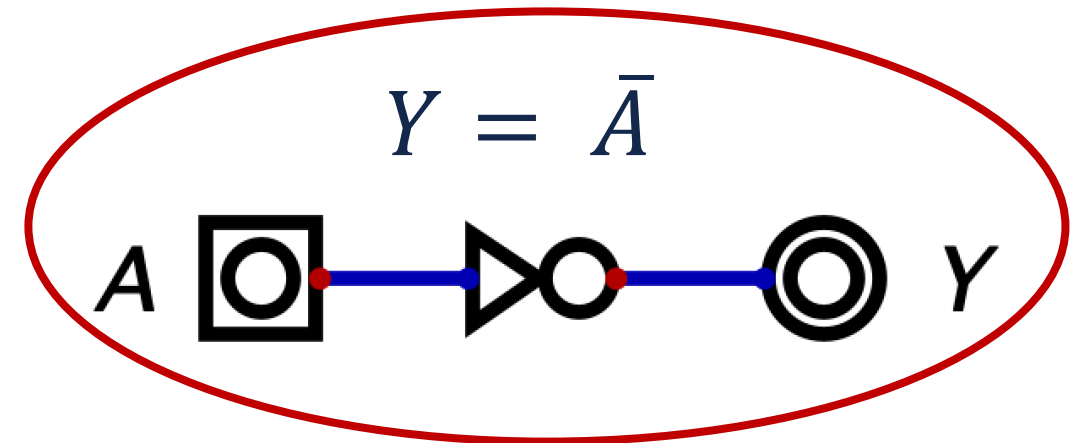
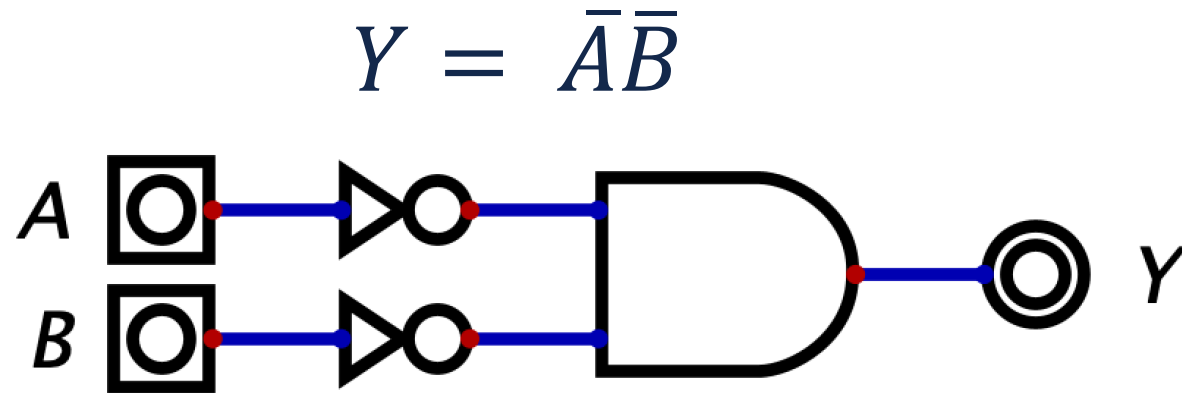
$$Y = \bar{A}\bar{B}$$



$$Y = \bar{A}$$



# Which circuit should I choose for my design?



This circuit is better because it contains fewer transistors

# Don't Care Outputs



- When designing our circuits, we will treat X as a 0 or a 1 depending on which decision yields the fewest number of transistors.
- How do we know which decision is better?
  - In a K-map, if treating the X as a 1 yields fewer or larger circles, then treat it as a 1.
  - If not, treat the X as a 0.
  - If there are multiple X's in a truth table, handle them individually
    - i.e. you can treat some X's as 1s and some X's as 0s

# Don't Cares in K-Maps



| AB \ CD | CD |    |    |    |    |
|---------|----|----|----|----|----|
|         |    | 00 | 01 | 11 | 10 |
| 00      | 0  | 0  | 0  | 0  | 0  |
| 01      | 0  | 1  | X  | 0  |    |
| 11      | 0  | 1  | X  | 0  |    |
| 10      | 0  | 0  | 0  | 0  | 0  |

4  
choices

# Don't Cares in K-Maps



- There are four ways we can interpret these don't cares. Which way yields the design with the fewest transistors?

|    |    | CD |    |    |    |  |
|----|----|----|----|----|----|--|
| AB |    | 00 | 01 | 11 | 10 |  |
|    | 00 | 0  | 0  | 0  | 0  |  |
|    | 01 | 0  | 1  | 0  | 0  |  |
|    | 11 | 0  | 1  | 1  | 0  |  |
|    | 10 | 0  | 0  | 0  | 0  |  |

|    |    | CD |    |    |    |
|----|----|----|----|----|----|
| AB |    | 00 | 01 | 11 | 10 |
|    | 00 | 0  | 0  | 0  | 0  |
|    | 01 | 0  | 1  | 1  | 0  |
|    | 11 | 0  | 1  | 1  | 0  |
|    | 10 | 0  | 0  | 0  | 0  |

|    |    | CD |    |    | 10 |
|----|----|----|----|----|----|
| AB |    | 00 | 01 | 11 | 10 |
|    | 00 | 0  | 0  | 0  | 0  |
|    | 01 | 0  | 1  | 0  | 0  |
|    | 11 | 0  | 1  | 0  | 0  |
|    | 10 | 0  | 0  | 0  | 0  |

|    |    | CD |    |    |    |
|----|----|----|----|----|----|
| AB |    | 00 | 01 | 11 | 10 |
|    | 00 | 0  | 0  | 0  | 0  |
|    | 01 | 0  | 1  | 1  | 0  |
|    | 11 | 0  | 1  | 0  | 0  |
|    | 10 | 0  | 0  | 0  | 0  |

# Don't Cares in K-Maps



- There are four ways we can interpret these don't cares. Which way yields the design with the fewest transistors?

|    |    | CD |    |    |    |  |
|----|----|----|----|----|----|--|
| AB |    | 00 | 01 | 11 | 10 |  |
|    | 00 | 0  | 0  | 0  | 0  |  |
|    | 01 | 0  | 1  | 0  | 0  |  |
|    | 11 | 0  | 1  | 1  | 0  |  |
|    | 10 | 0  | 0  | 0  | 0  |  |

|    |    | CD |    |    |    |
|----|----|----|----|----|----|
| AB |    | 00 | 01 | 11 | 10 |
|    | 00 | 0  | 0  | 0  | 0  |
|    | 01 | 0  | 1  | 1  | 0  |
|    | 11 | 0  | 1  | 1  | 0  |
|    | 10 | 0  | 0  | 0  | 0  |

|    |    | CD |    |    | 10 |
|----|----|----|----|----|----|
| AB |    | 00 | 01 | 11 | 10 |
|    | 00 | 0  | 0  | 0  | 0  |
|    | 01 | 0  | 1  | 0  | 0  |
|    | 11 | 0  | 1  | 0  | 0  |
|    | 10 | 0  | 0  | 0  | 0  |

|    |    | CD |    |    |    |
|----|----|----|----|----|----|
| AB |    | 00 | 01 | 11 | 10 |
|    | 00 | 0  | 0  | 0  | 0  |
|    | 01 | 0  | 1  | 1  | 0  |
|    | 11 | 0  | 1  | 0  | 0  |
|    | 10 | 0  | 0  | 0  | 0  |

$$Y = B\bar{C}D$$

# Don't Cares in K-Maps



- There are four ways we can interpret these don't cares. Which way yields the design with the fewest transistors?

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 0  | 0  |  |
| 11      | 0  | 1  | 1  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D + ABD$$

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 1  | 0  |  |
| 11      | 0  | 1  | 1  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 0  | 0  |  |
| 11      | 0  | 1  | 0  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D$$

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 1  | 0  |  |
| 11      | 0  | 1  | 0  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |



# Don't Cares in K-Maps

- There are four ways we can interpret these don't cares. Which way yields the design with the fewest transistors?

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 0  | 0  |  |
| 11      | 0  | 1  | 1  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D + ABD$$

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 1  | 0  |  |
| 11      | 0  | 1  | 1  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 0  | 0  |  |
| 11      | 0  | 1  | 0  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D$$

| AB \ CD | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 1  | 0  |  |
| 11      | 0  | 1  | 0  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D + \bar{A}BD$$

# Don't Cares in K-Maps



- There are four ways we can interpret these don't cares. Which way yields the design with the fewest transistors?

| CD \ AB | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 0  | 0  |  |
| 11      | 0  | 1  | 1  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D + ABD$$

| CD \ AB | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 1  | 0  |  |
| 11      | 0  | 1  | 1  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = BD$$

This implementation contains the fewest number of transistors

| CD \ AB | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 0  | 0  |  |
| 11      | 0  | 1  | 0  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D$$

| CD \ AB | CD |    |    |    |  |
|---------|----|----|----|----|--|
|         | 00 | 01 | 11 | 10 |  |
| 00      | 0  | 0  | 0  | 0  |  |
| 01      | 0  | 1  | 1  | 0  |  |
| 11      | 0  | 1  | 0  | 0  |  |
| 10      | 0  | 0  | 0  | 0  |  |

$$Y = B\bar{C}D + \bar{A}BD$$



# Adder Circuits!

# Recall: Binary Addition



$$\begin{array}{r} \phantom{1}1\phantom{1} \\ + 101 \\ \underline{111} \\ 1000 \end{array}$$



# 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

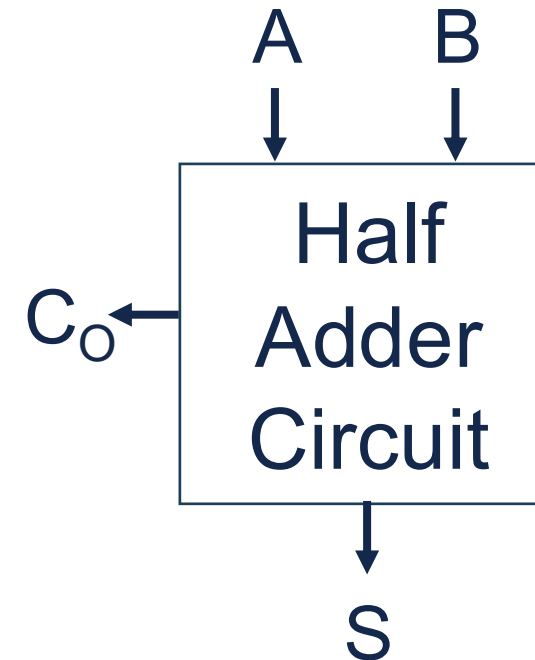
$$\begin{array}{r} 111 \\ + 101 \\ \hline 100 \end{array}$$

# 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

$$\begin{array}{r} 11\boxed{1} \leftarrow \text{Output: } C_o \\ + 10\boxed{1} \leftarrow \text{Input: } A \\ + 11\boxed{1} \leftarrow \text{Input: } B \\ \hline 10\boxed{0} \leftarrow \text{Output: } S \end{array}$$



# Half Adder



| A | B | C <sub>o</sub> | S |
|---|---|----------------|---|
| 0 | 0 | 0              | 0 |
| 0 | 1 | 0              | 1 |
| 1 | 0 | 0              | 1 |
| 1 | 1 | 1              | 0 |

$$\begin{array}{r} \times \\ 0 \\ + 0 \\ \hline 0 \end{array}$$

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

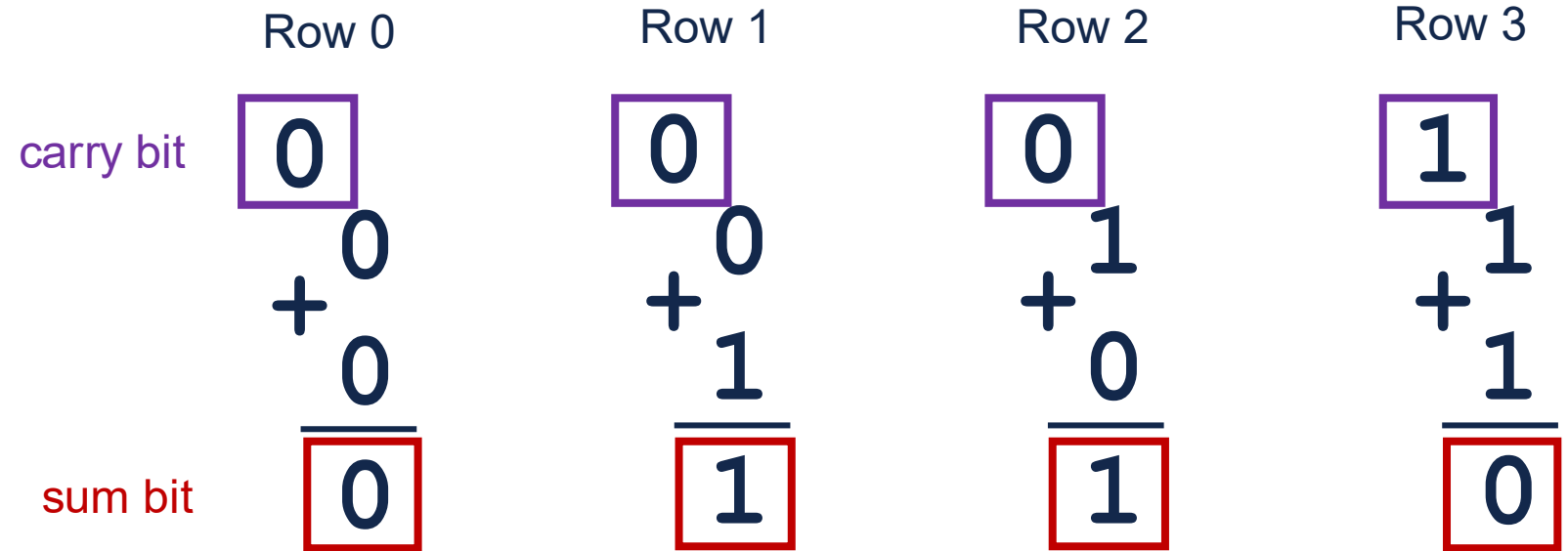
$$\begin{array}{r} \checkmark \\ 1 \\ + 0 \\ \hline 1 \end{array}$$

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

# Half Adder



| A | B | C <sub>o</sub> | S |
|---|---|----------------|---|
| 0 | 0 | 0              | 0 |
| 0 | 1 | 0              | 1 |
| 1 | 0 | 0              | 1 |
| 1 | 1 | 1              | 0 |



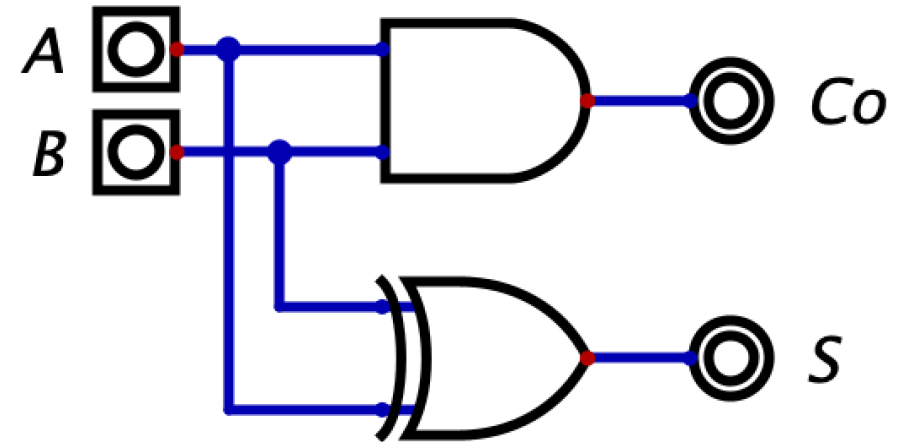
# Half Adder



| A | B | $C_o$ | 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

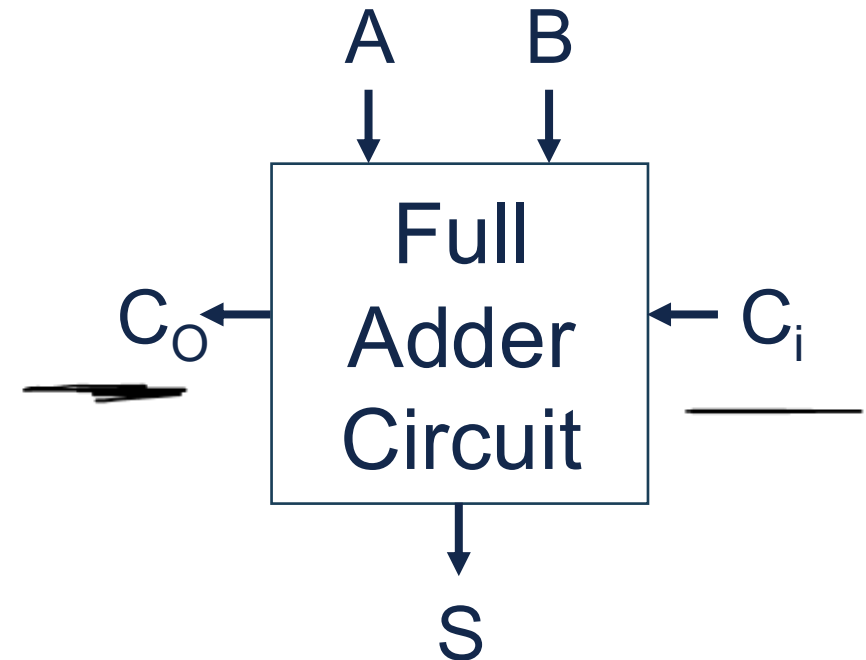
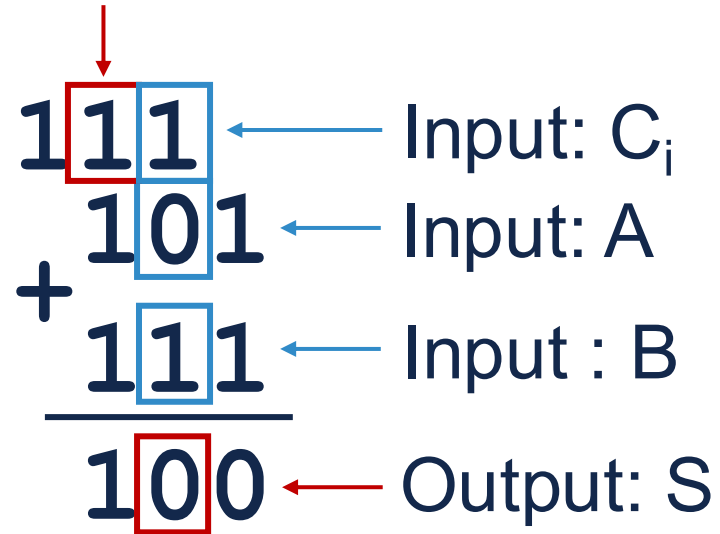
$$\begin{array}{r} 111 \\ + 101 \\ + 111 \\ \hline 100 \end{array}$$

# Full Adder



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

Output:  $C_o$



# Full Adder

| A | B | C <sub>i</sub> | C <sub>o</sub> | 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              |                |   |

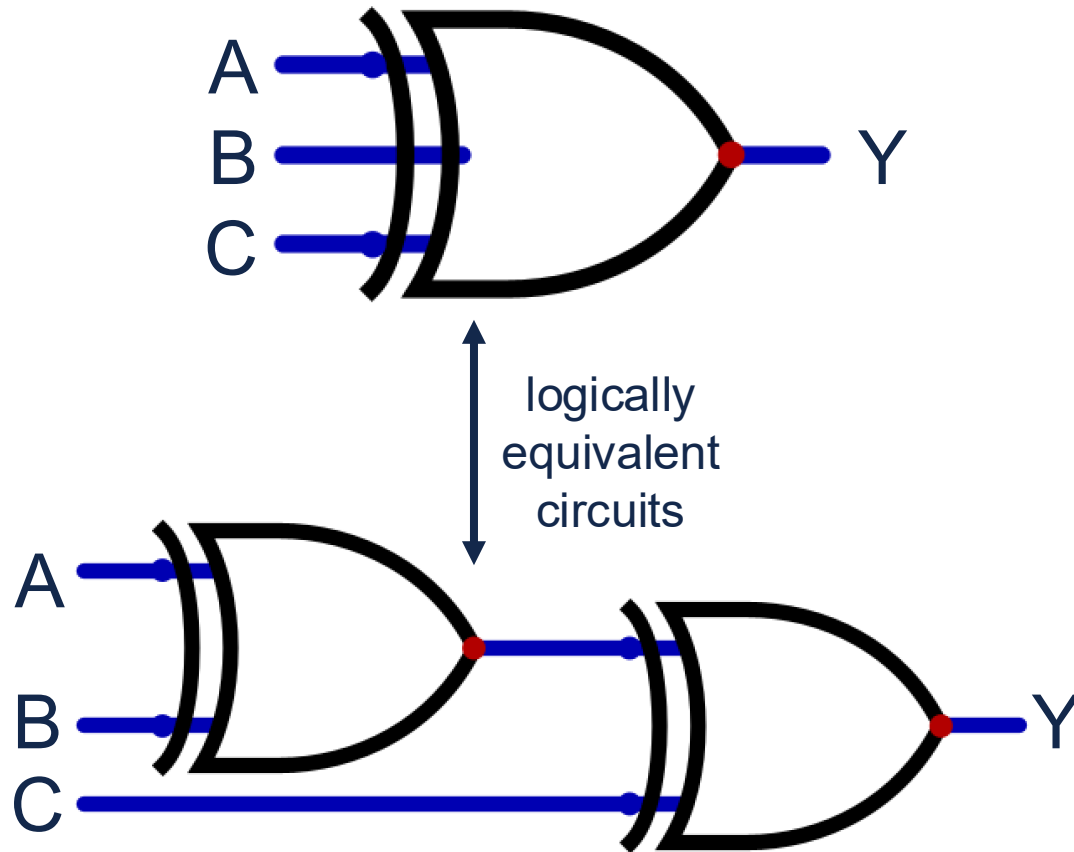
1. Fill in truth table
2. Find the equations
3. Draw the schematic and count the transistors in your design



| BC <sub>i</sub> |  | 00 | 01 | 11 | 10 |
|-----------------|--|----|----|----|----|
| A               |  |    |    |    |    |
| 0               |  |    |    |    |    |
| 1               |  |    |    |    |    |

Hint: Maybe a K-Map would help for C<sub>o</sub>

# 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 |

$\equiv S$  



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

# Transistor Count

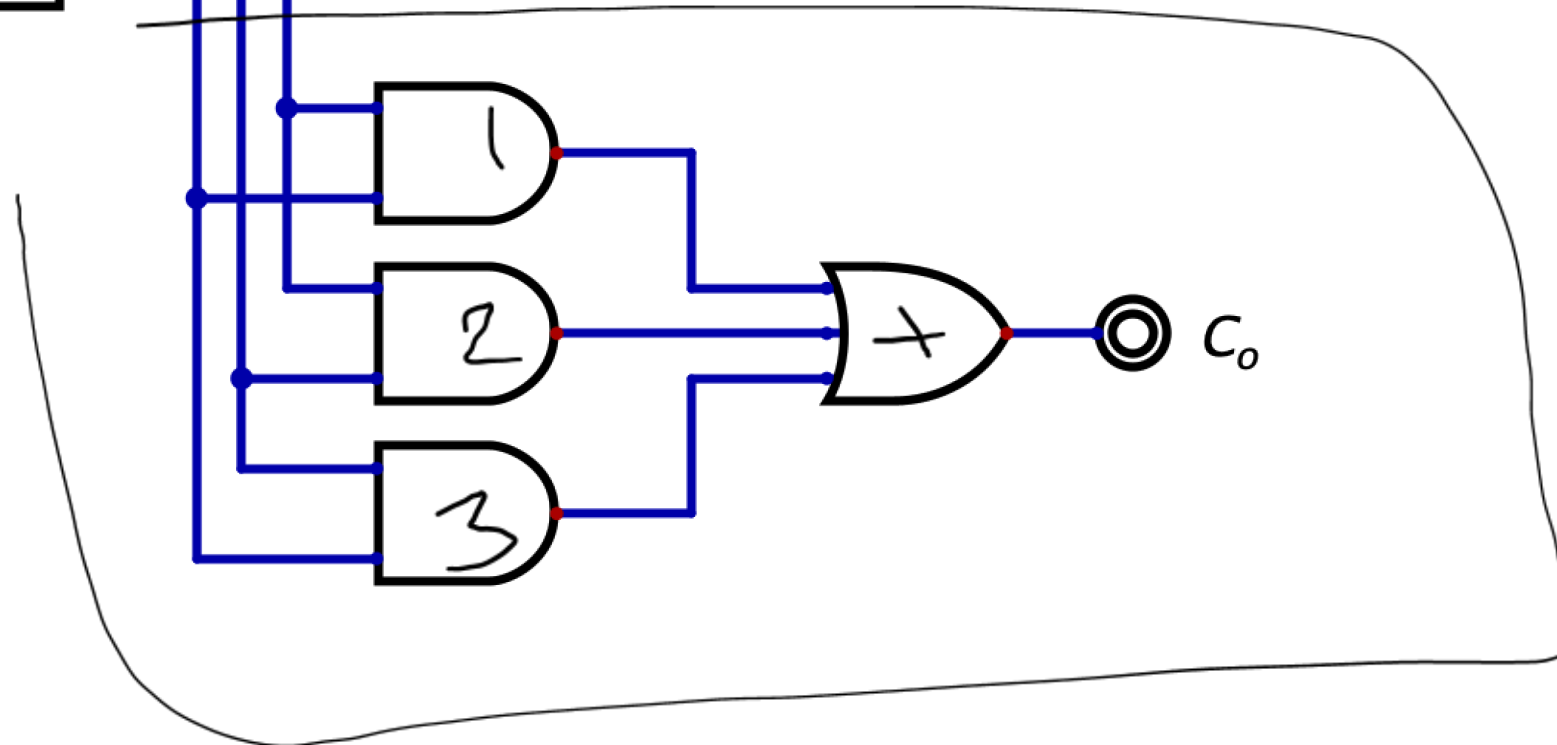
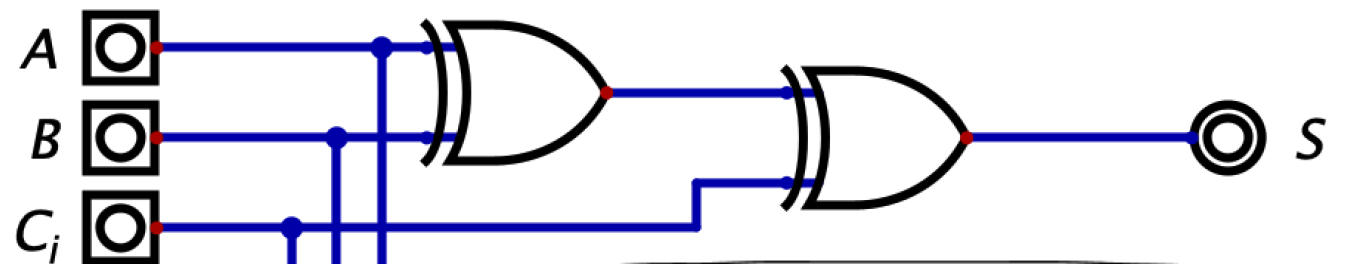


| Gate         | # of Transistors |
|--------------|------------------|
| NOT          | 2                |
| n-input AND  | $2n + 2$         |
| n-input OR   | $2n + 2$         |
| n-input NAND | $2n$             |
| n-input NOR  | $2n$             |
| 2-input XOR  | 12               |
| 3-input XOR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

# Full Adder

$$S = A \oplus B \oplus C_i$$

$$C_o = AC_i + AB + BC_i$$

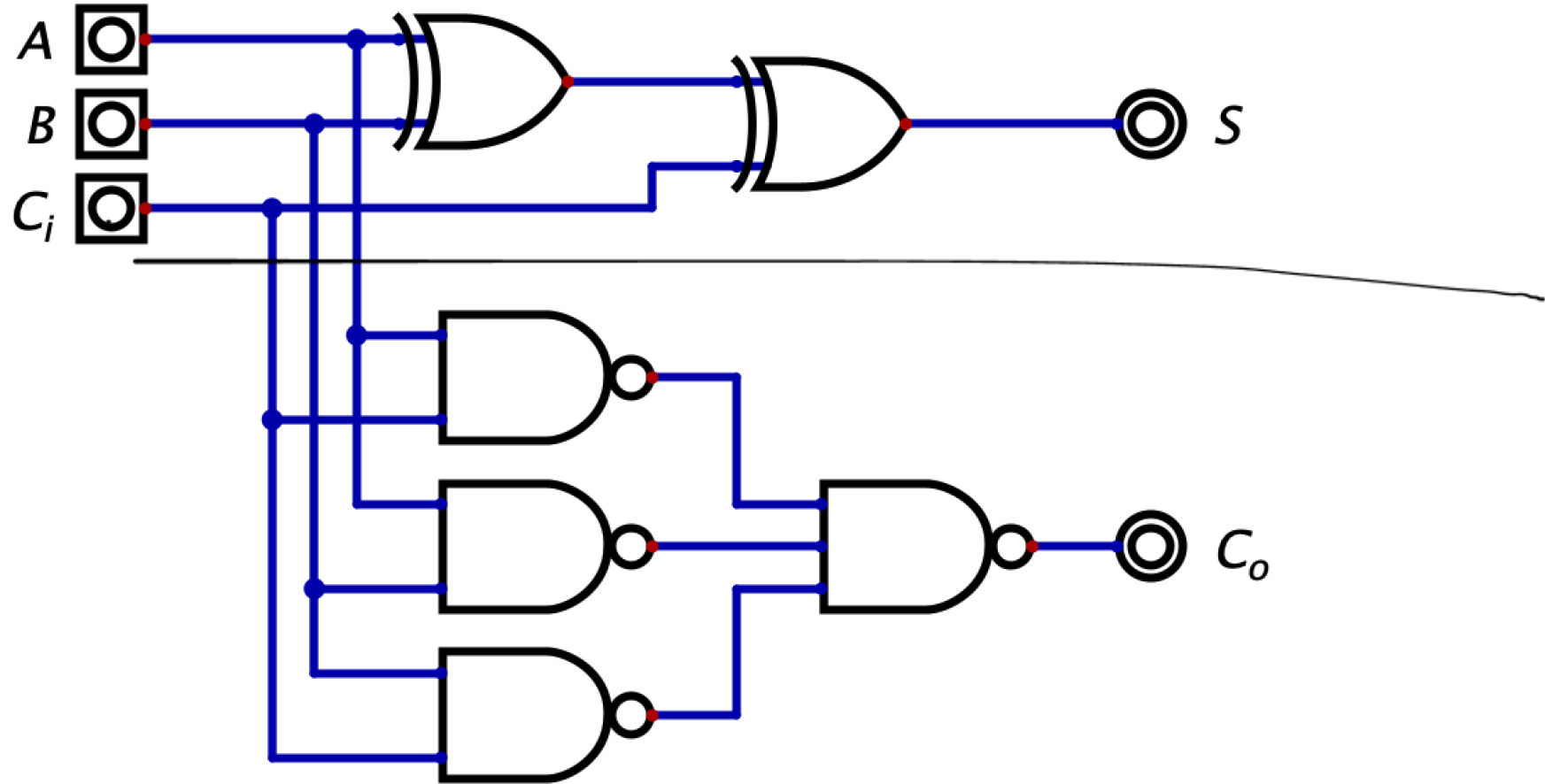


# Full Adder: transistor minimization



$$S = A \oplus B \oplus C_i$$

$$C_o = AC_i + AB + BC_i$$

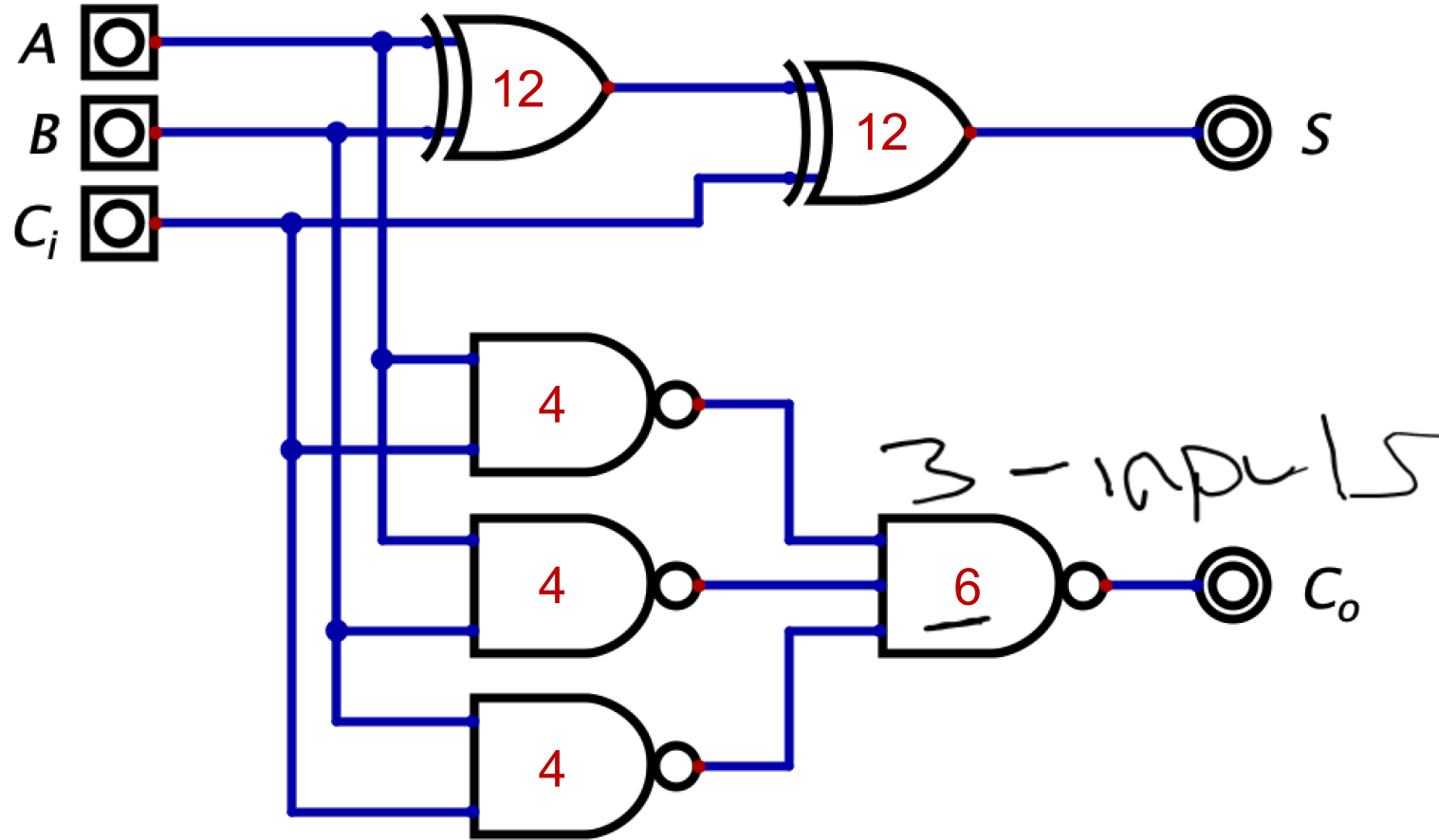


# Full Adder: transistor minimization



$$S = A \oplus B \oplus C_i$$

$$C_o = AC_i + AB + BC_i$$



Transistor count = 42

# Full Adder: Alternate implementation



| A | B | C <sub>i</sub> | C <sub>o</sub> | 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 <sub>i</sub> | C <sub>o</sub> | 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 |

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

1 2 3 4

$$C_o = \bar{A}BC_i + A\bar{B}C_i + AB\bar{C}_i + ABC_i$$

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

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

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

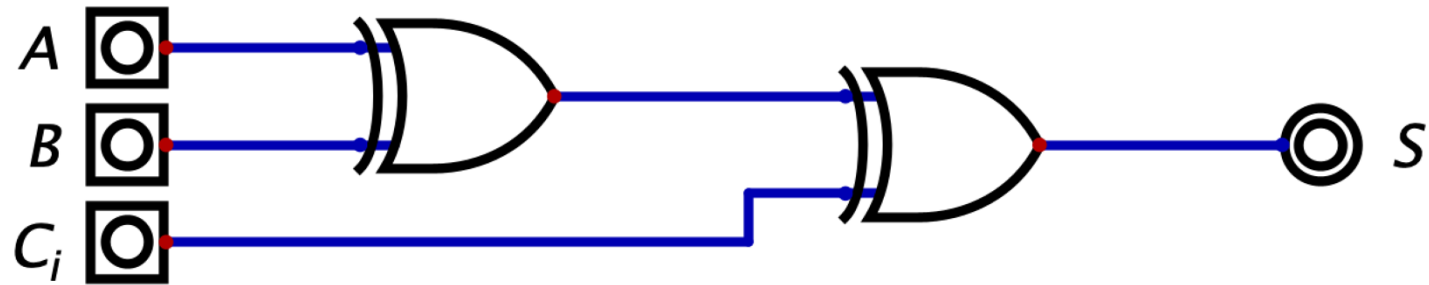
$$C_o = C_i(A \oplus B) + AB$$

# Full Adder: Alternate Implementation



$$S = A \oplus B \oplus C_i$$

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



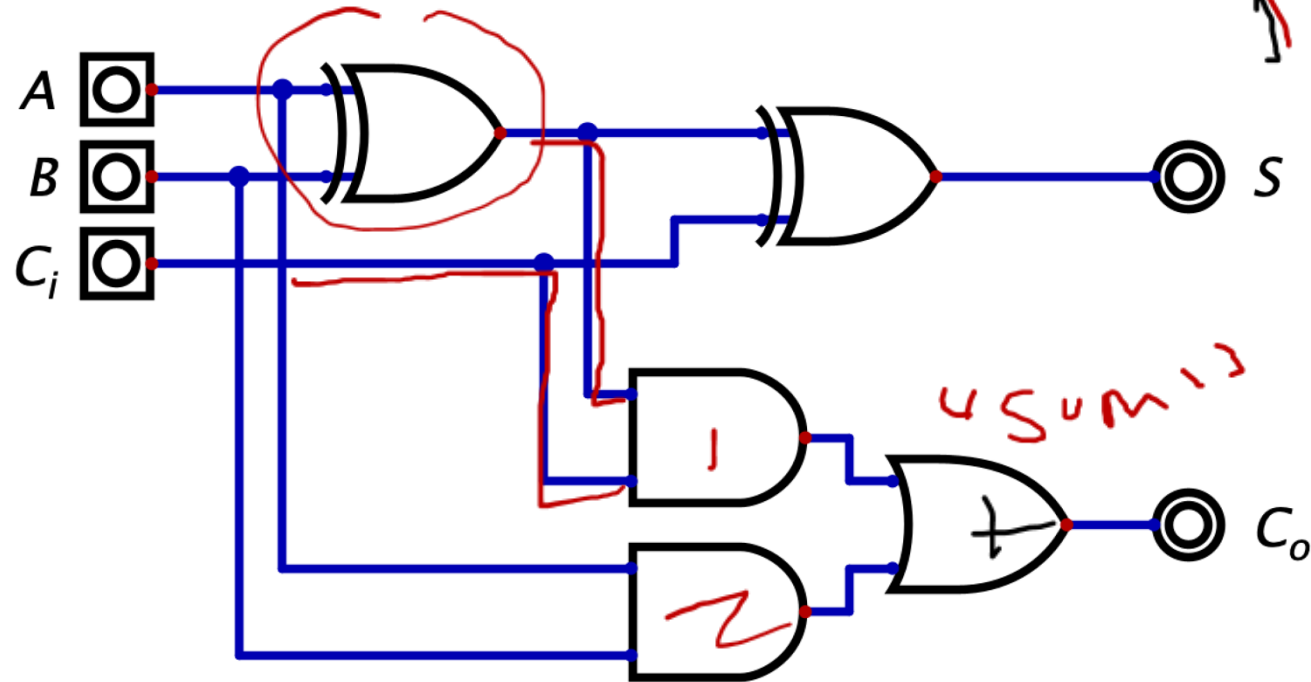
  $C_o$

# Full Adder: Alternate Implementation



$$S = A \oplus B \oplus C_i$$

$$C_o = C_i \underbrace{(A \oplus B)}_{\uparrow} + \underbrace{AB}_Z$$

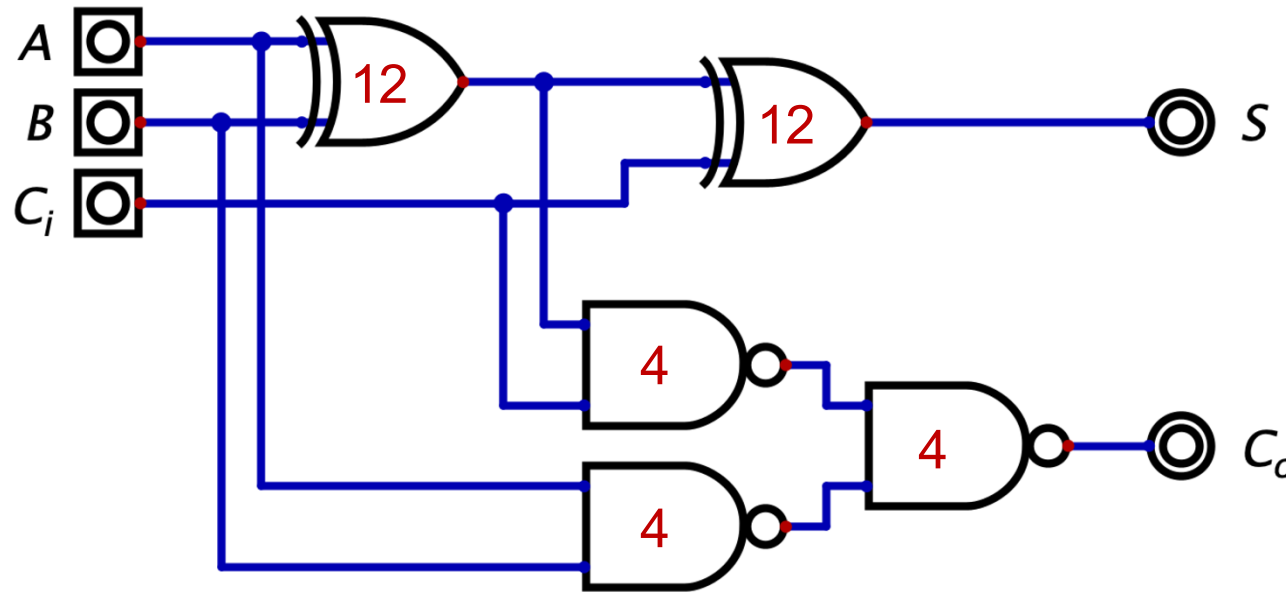


# Full Adder: Reducing Transistor Count



$$S = A \oplus B \oplus C_i$$

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



Transistor Count: 36

# Full Adder Abstraction

