COMP311: *COMPUTER ORGANIZATION!*

Lecture 7: Gate Minimization, Karnaugh Maps

tinyurl.com/comp311-fa25

Boolean Algebra Laws

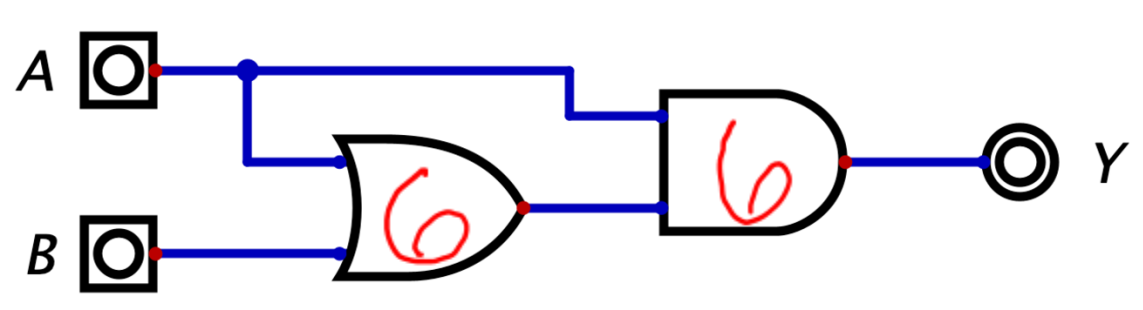
Name	OR Form	AND Form
Identity	$A + 0 = A$	$A \cdot 1 = A$
Null	$A + 1 = 1$	$A \cdot 0 = 0$
Idempotent	$A + A = A$	$A \cdot A = A$
Complement	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
Commutative	$A + B = B + A$	$A \cdot B = B \cdot A$
Associative	$(A + B) + C = A + (B + C)$	$(A \cdot B) \cdot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

Recall: Transistor Count

Gate	Number of Transistors
NOT	2
AND	6
OR	6
NAND	4
NOR	4
XOR	12
XNOR	12

Comparing Number of Transistors Used

Original Circuit



$$Y = A \cdot (A + B)$$

Number of Transistors = 12

Simplified Circuit

$$Y = A$$

Number of Transistors = 0

Your Turn!

Simplify the following Boolean expression: $B + \overline{B}C$

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

Your Turn!

$$B + \bar{B}C$$

$$(B + \bar{B})(B + C)$$
$$(1)(B + C)$$
$$B + C$$

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	$A + \overline{A} = 1$	$A \cdot \overline{A} = 0$
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	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$
De Morgan's	$\overline{A + B} = \overline{A} \cdot \overline{B}$	$\overline{A \cdot B} = \overline{A} + \overline{B}$

Recall: De Morgan's Theorem

$$\overline{A + B} = \overline{A} \cdot \overline{B}$$

$$\overline{AB} = \overline{A} + \overline{B}$$

DeMorgan’s

$$\overline{AB + CD}$$

- How would you apply DeMorgan’s to this expression?

DeMorgan's

$$\overline{AB} = \overline{A} + \overline{C}$$

$$\overline{AB + CD}$$

- How would you apply DeMorgan's to this expression?
- We group our terms based on the lowest operator precedence under the bar
- The order of operations for Boolean Algebra is **NOT, then AND, then OR.** Expressions inside brackets are always evaluated first.
- In this case, the operator with the lowest precedence is OR

$$(\overline{AB})(\overline{CD})$$

$$(\bar{A} + \bar{B})(\bar{C} + \bar{D})$$

DeMorgan's

$$\overline{x+y} = \bar{x} \cdot \bar{y}$$

$$\overline{xy} = \bar{x} + \bar{y}$$

$$\rightarrow \overline{\bar{A}B + \bar{C}}$$

- How would you apply DeMorgan's to this expression?

$$\overline{(\bar{A}\,B) \cdot \bar{C}}$$

$$(A + \bar{B}) \cdot C$$

$$AC + \bar{B}C$$

DeMorgan's

$$\overline{\bar{A}B + \bar{C}}$$

- How would you apply DeMorgan's to this expression?

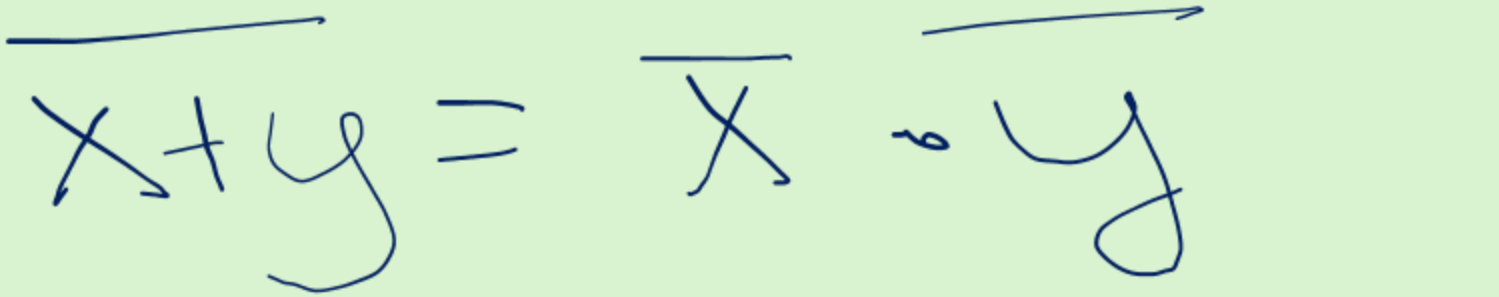
$$\overline{\bar{A}B}C$$

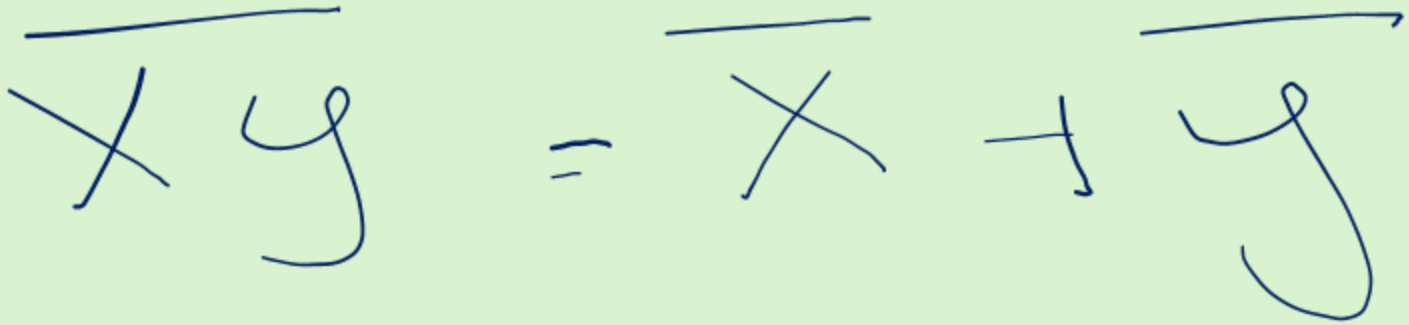
$$(A + \bar{B})C$$

DeMorgan's

- How would you apply DeMorgan's to this expression?

$$\overline{(\bar{A} + B)(C + D)\overline{(E + F)} + G}$$





DeMorgan's

- How would you apply DeMorgan's to this expression?

$$\overline{(\bar{A}+B)(C+D)\overline{(E+F)}+G}$$

$$\overline{(\bar{A}+B)(C+D)\overline{(E+F)}}\cdot\bar{G}$$

$$(\overline{\bar{A}+B}+\overline{C+D}+E+F)\cdot\bar{G}$$

$$(A\bar{B}+\bar{C}\bar{D}+E+F)\cdot\bar{G}$$

Simplify

Your Turn!

Simplify the following Boolean expression:

$$\boldsymbol{ABC + AB\overline{C} + \overline{AC}B + ACB + \overline{\overline{AB}\overline{C}\,\overline{D}} + \overline{\overline{A}B + \overline{C}}}$$

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

Your Turn!

$$ABC + AB\bar{C} + \overline{AC}B + ACB + \overline{\overline{A\bar{B}}\,\overline{C}\,\overline{D}} + \overline{\bar{A}B + \bar{C}}$$

Your Turn!

$$ABC + AB\bar{C} + \overline{AC}B + ACB + \overline{\overline{AB}\overline{C}\,\overline{D}} + \overline{\bar{A}B + \bar{C}}$$

$$AB(C + \bar{C}) + B(\overline{AC} + AC) + AB + C + D + (\overline{\bar{A}B})C$$

$$AB(1) + B(1) + AB + C + D + (A + \bar{B})C$$

$$AB + B + AB + C + D + AC + \bar{B}C$$

$$AB + B + C + D + AC + \bar{B}C$$

$$B(A + 1) + C(1 + A) + \bar{B}C + D$$

$$B(1) + C(1) + \bar{B}C + D$$

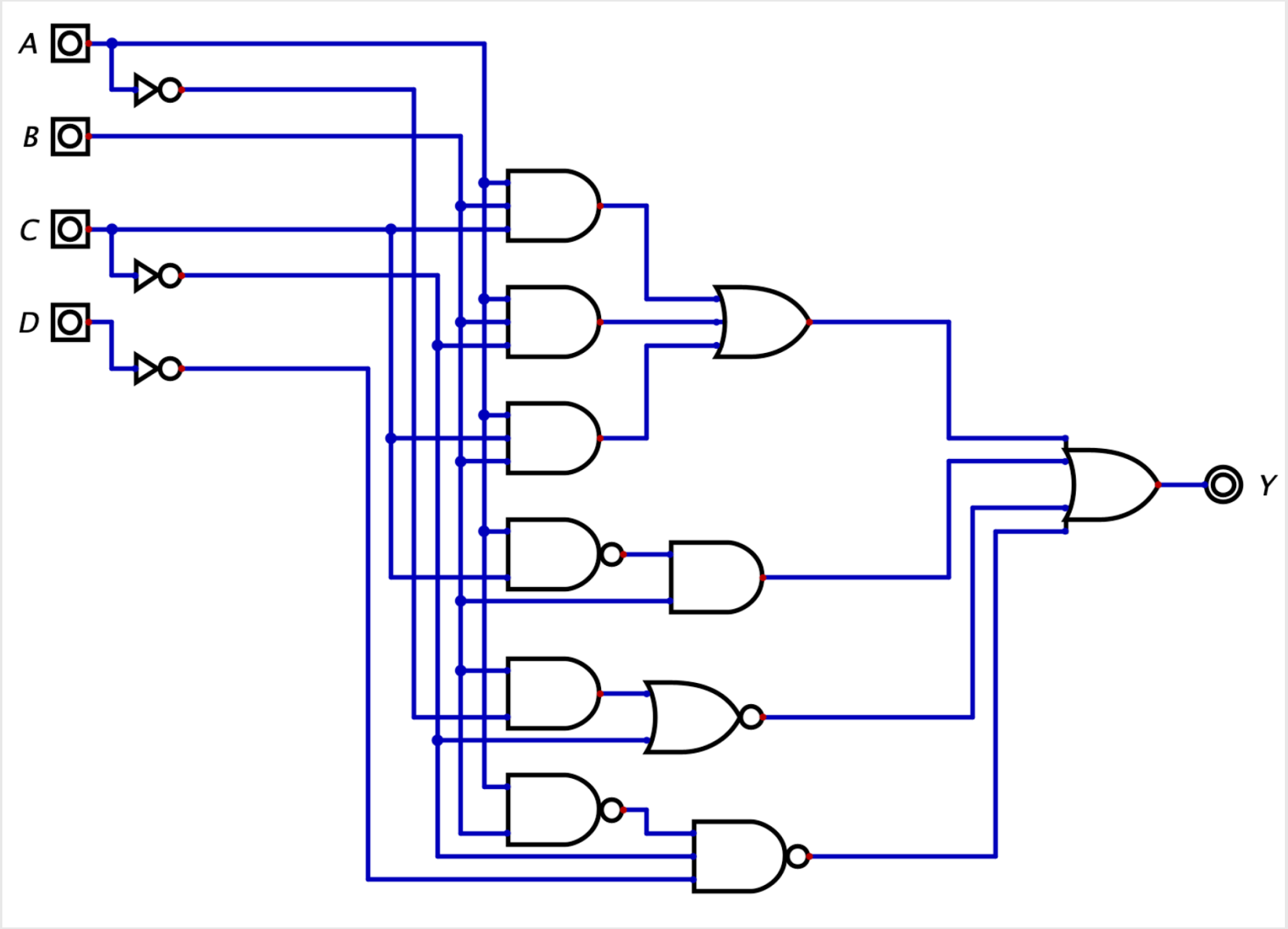
$$B + C + \bar{B}C + D$$

$$B + C(1 + \bar{B}) + D$$

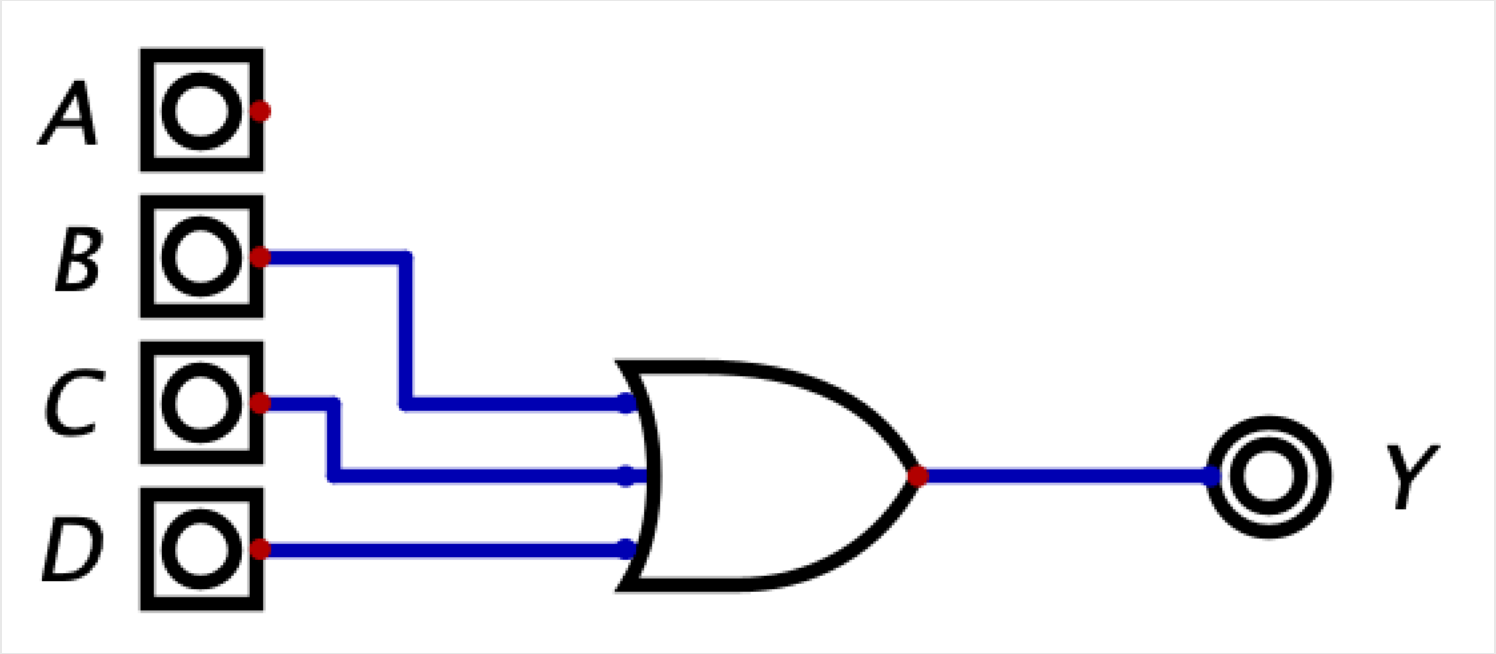
$$B + C(1) + D$$

$$B + C + D$$

Optimizing Our Gates



Optimizing Our Gates

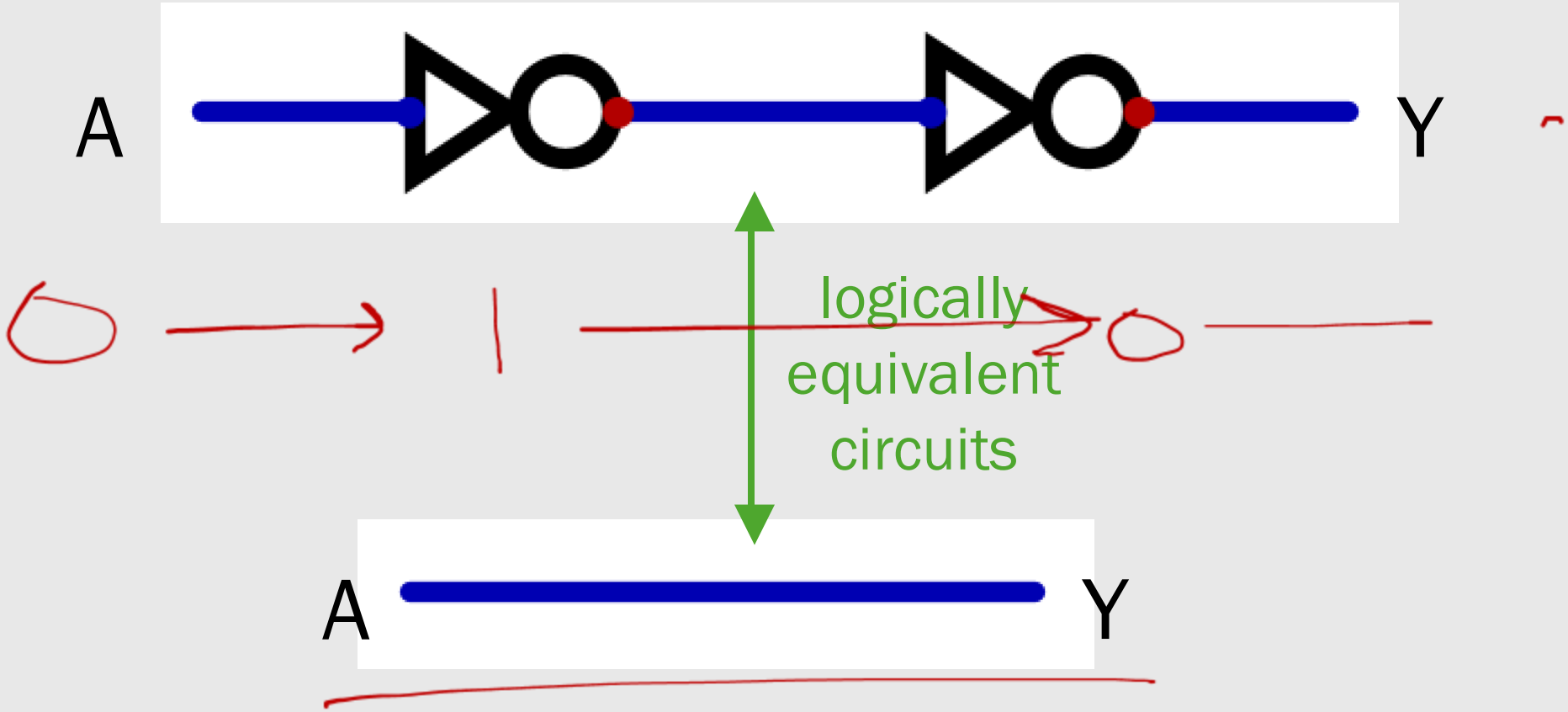


Yay! We did it!

Logically Equivalent Circuits

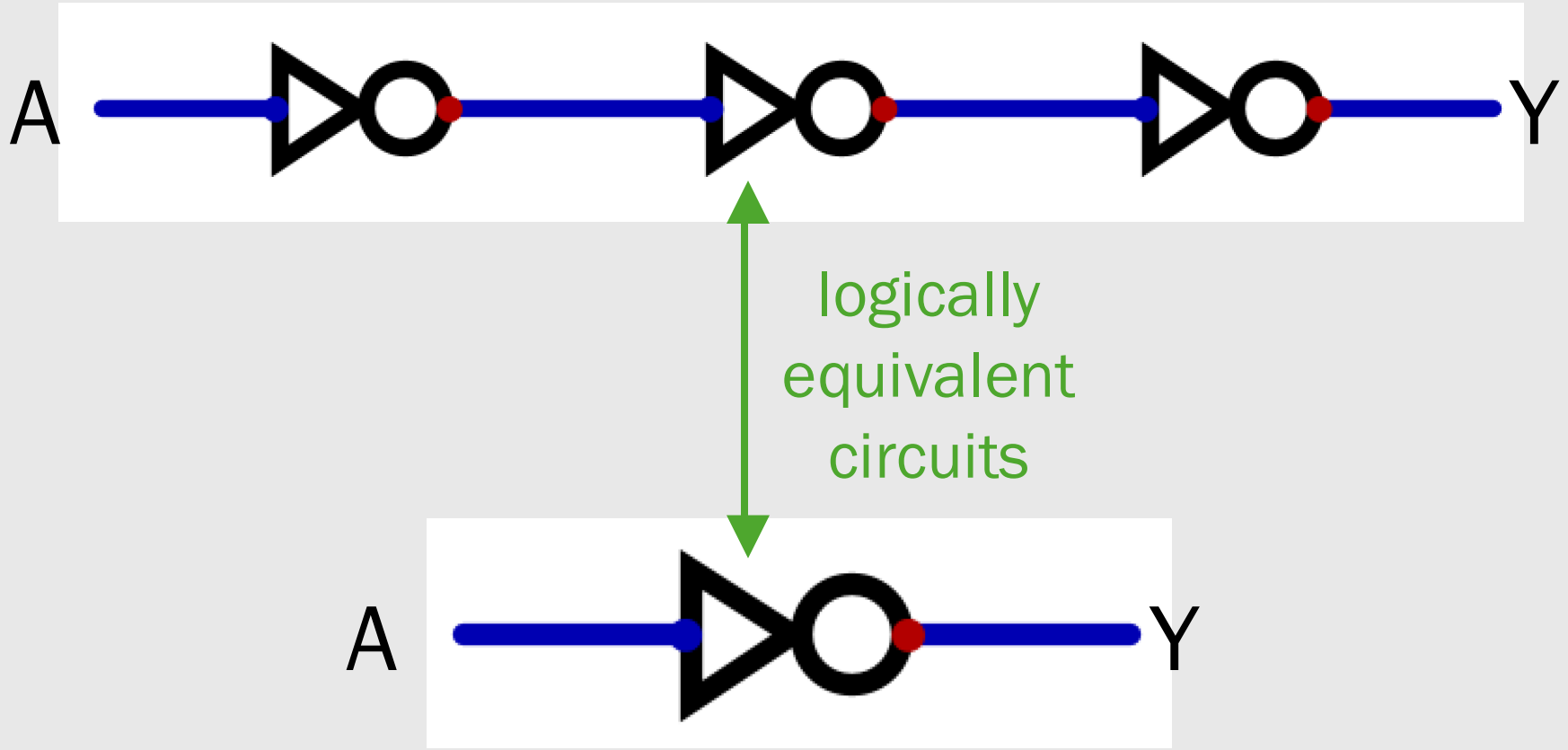
- For any input, they will produce the same output
- In other words, their truth tables are equivalent

Inverters



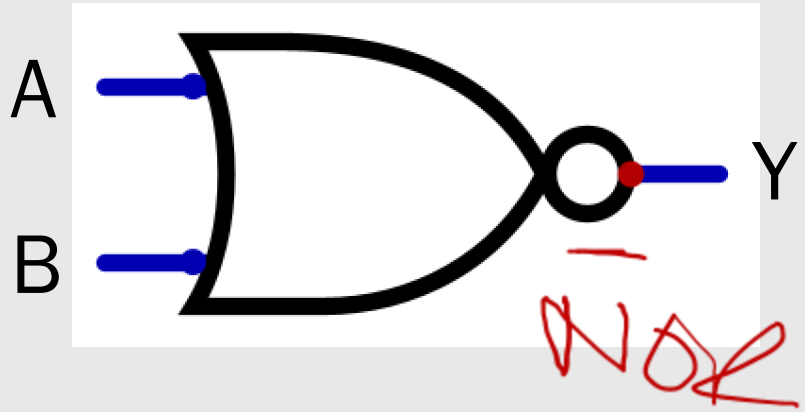
Any even number of inverters chained together is logically equivalent to having no inverters

Inverters

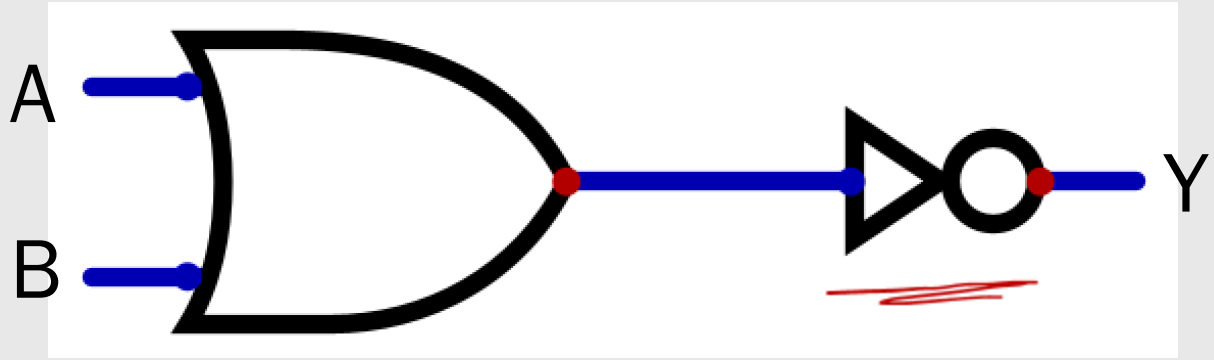


Any odd number of inverters chained together is logically equivalent to having one inverter

Bubble is Logically Equivalent to Inverter

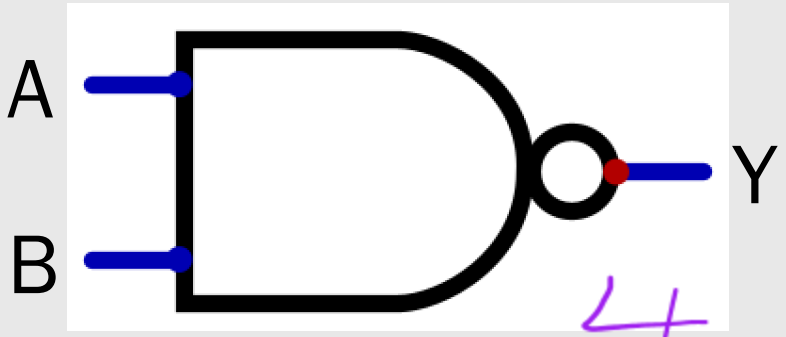


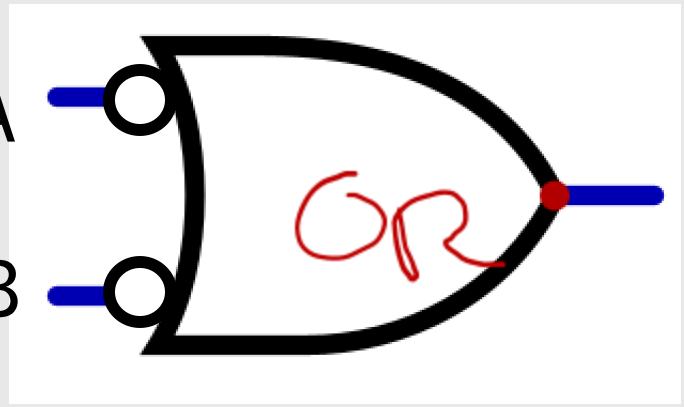
4 transistors



8 transistors

NAND

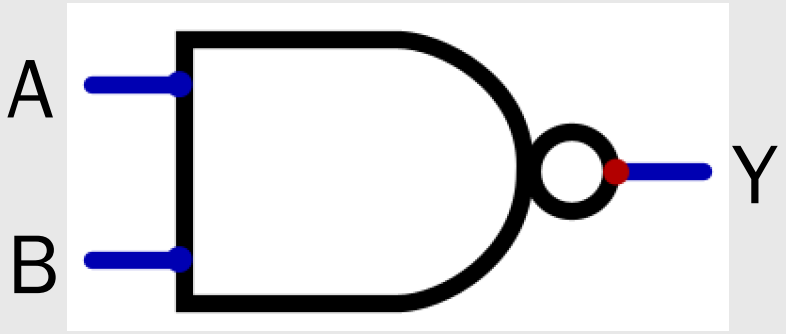




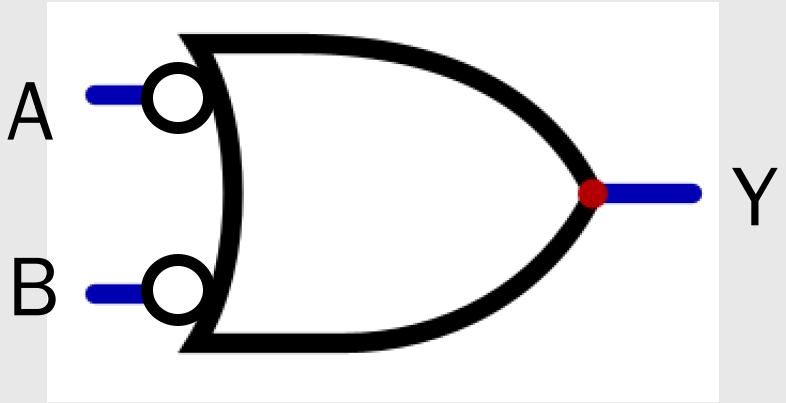
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

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

NAND

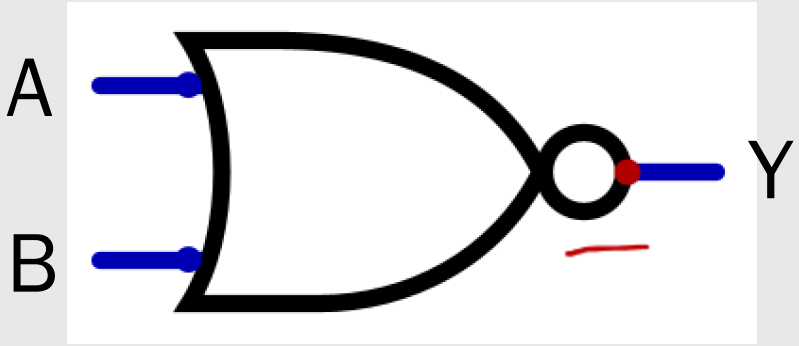


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

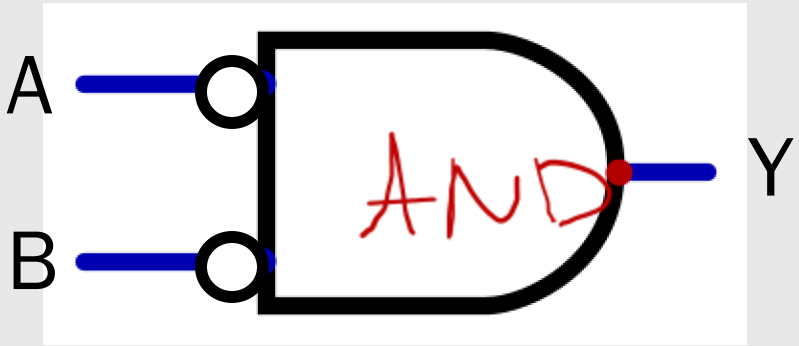


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR



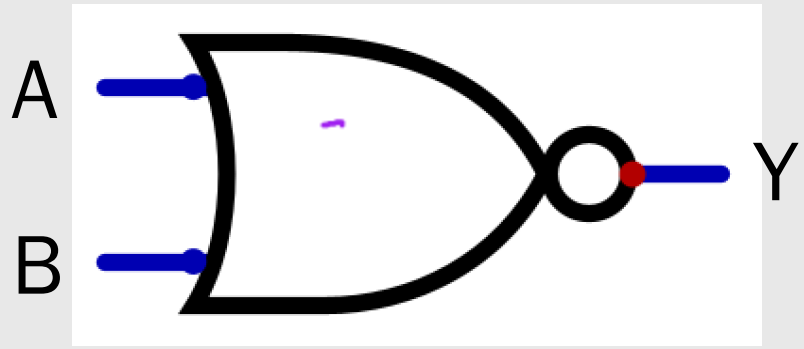
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



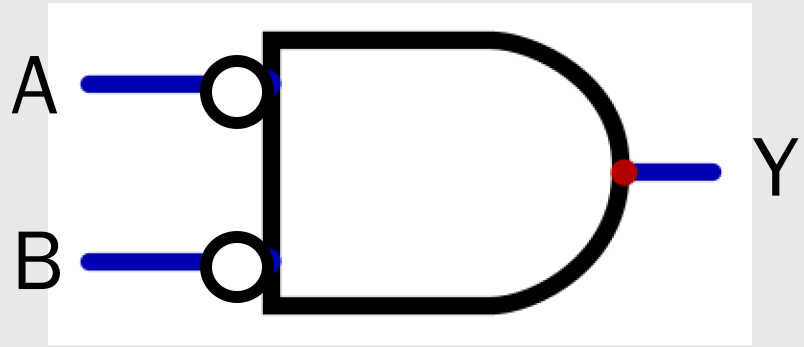
NAND

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

NOR



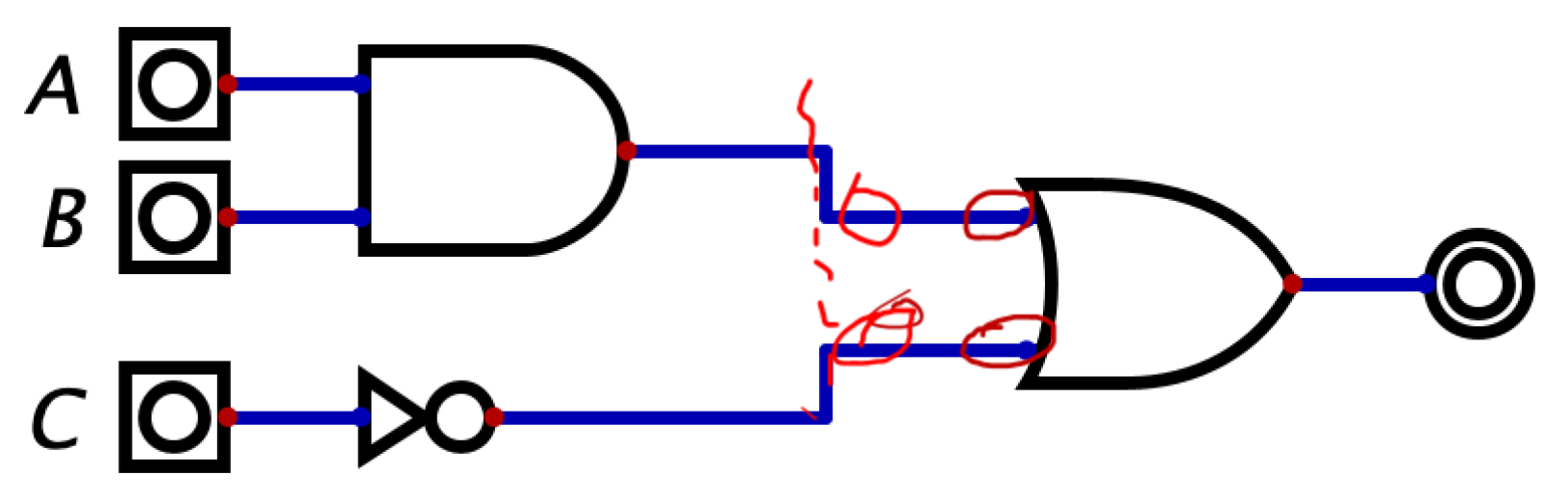
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0



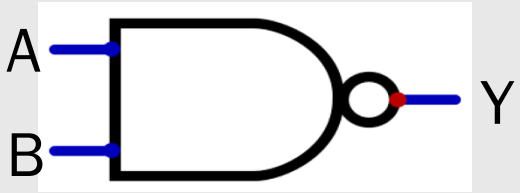
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Ex1: Minimizing Transistor Count

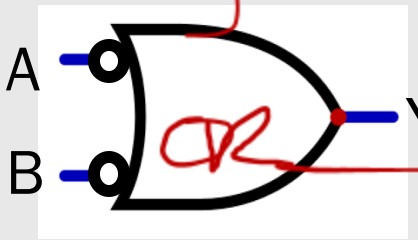
Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters.



NAND gates:



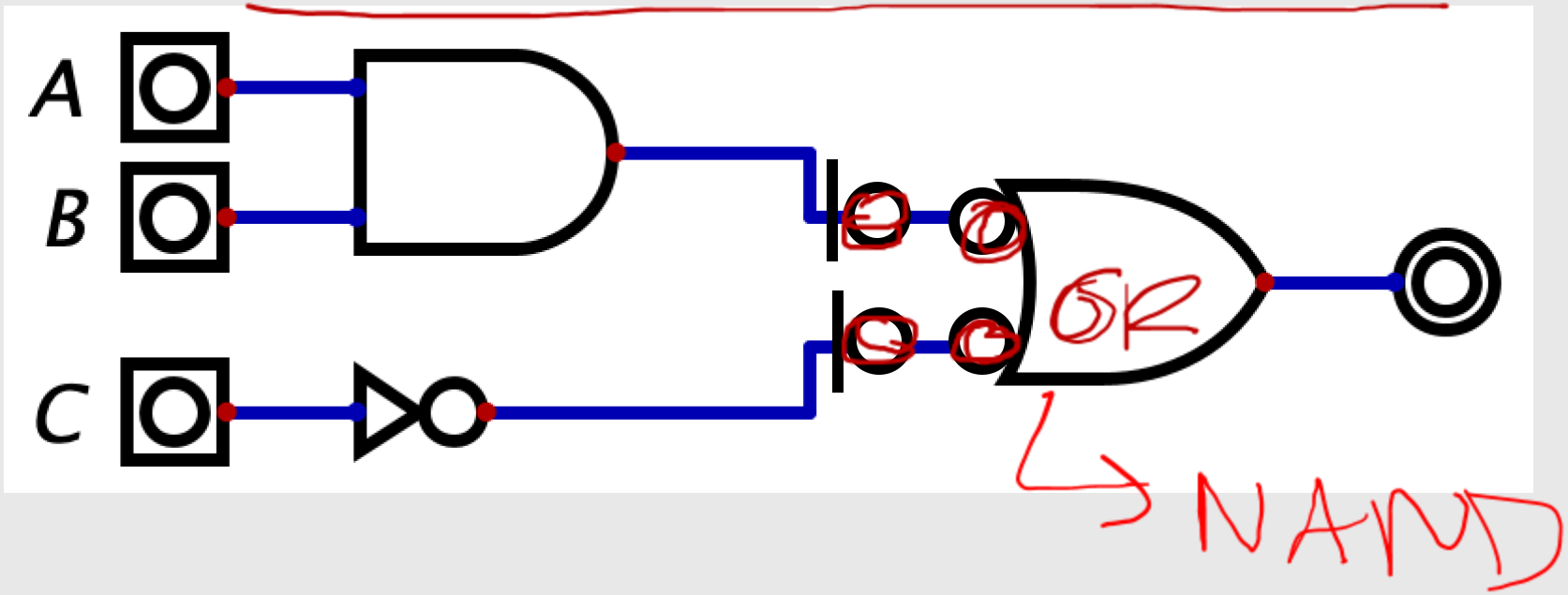
logically equiv.



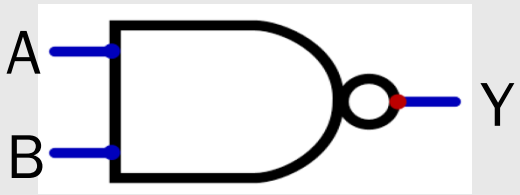
input neg.

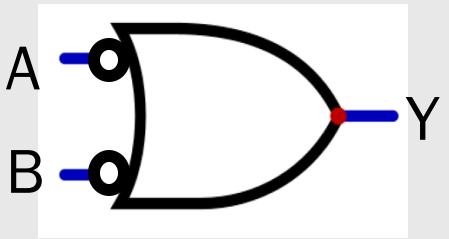
Ex1: Minimizing Transistor Count

Step 1: Start with the right-most gate. Turn it into a NAND gate. Then add bubbles to make the circuit logically equivalent.



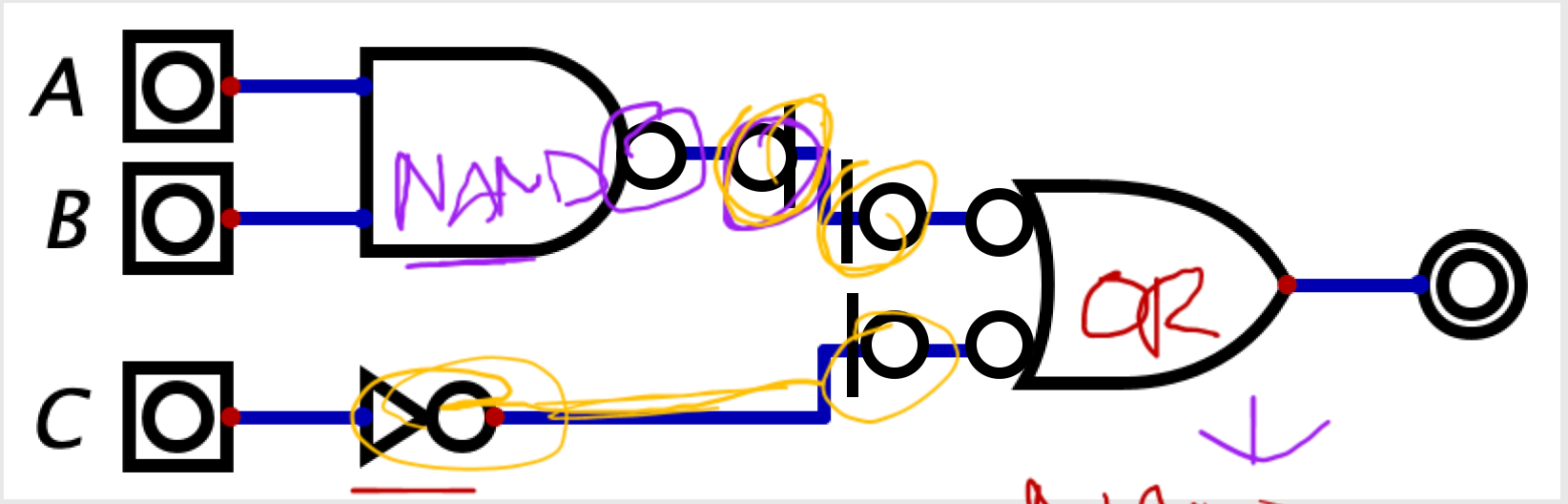
NAND gates:



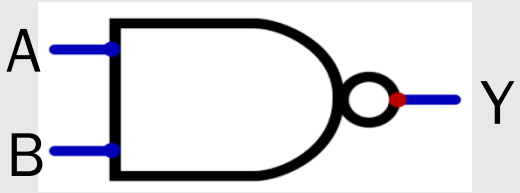


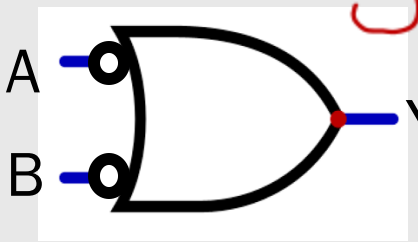
Ex1: Minimizing Transistor Count

Step 2: Now make the top left gate a NAND gate. Then add a bubble to make the circuit logically equivalent.



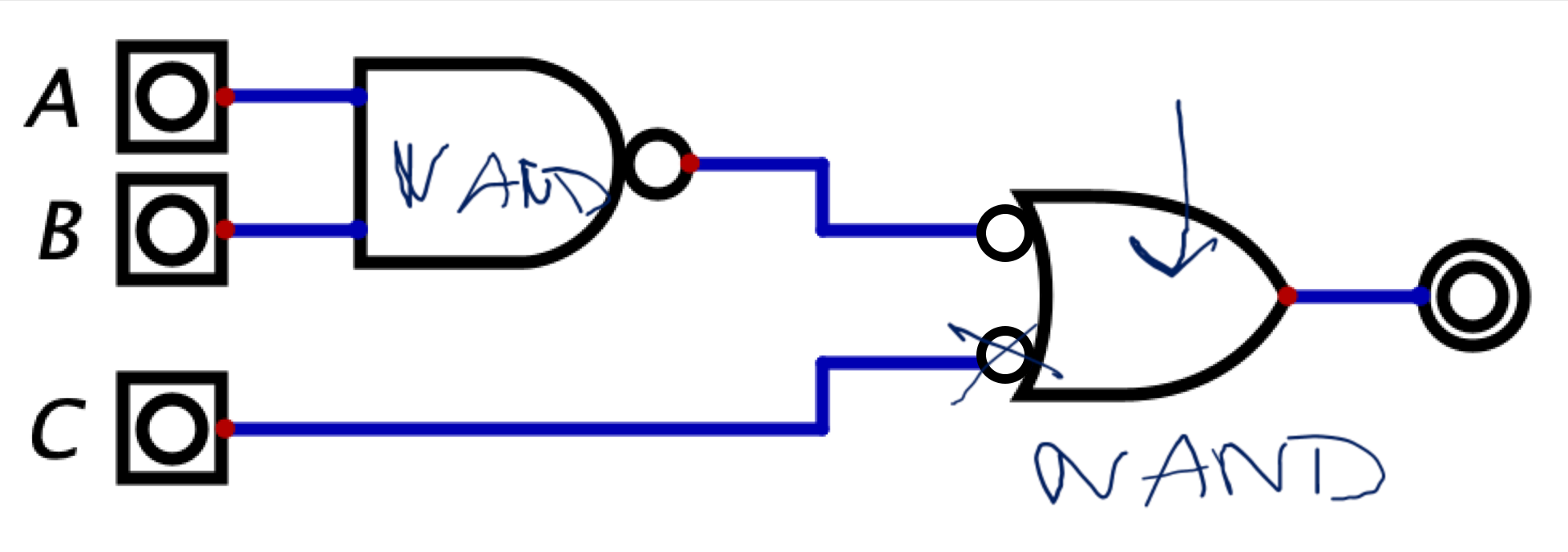
NAND gates:





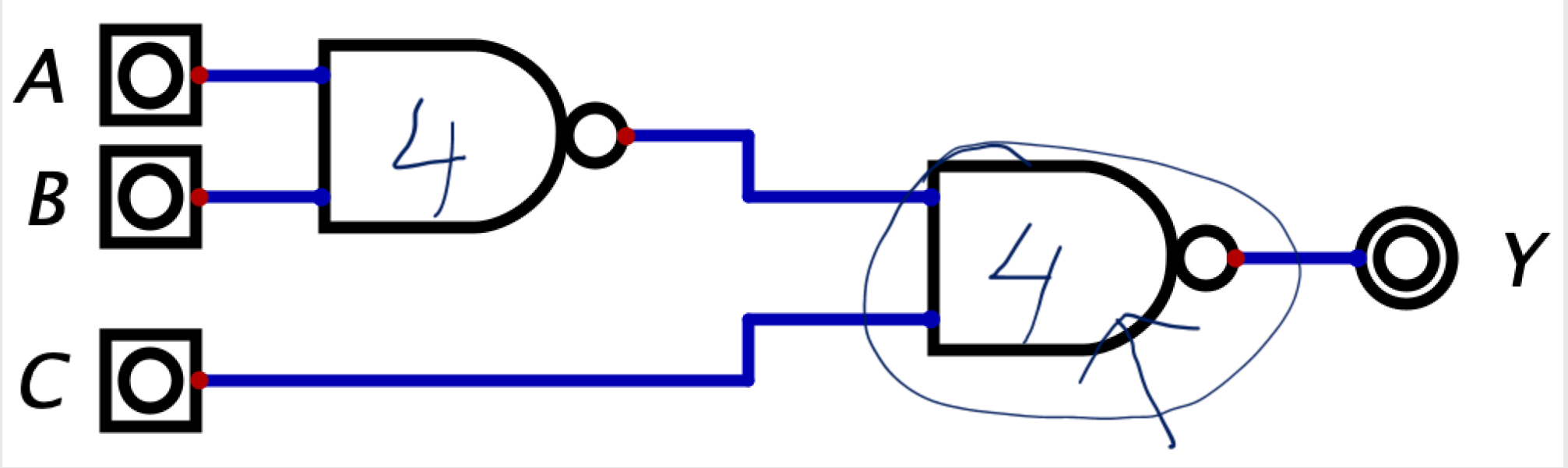
Ex1: Minimizing Transistor Count

Step 3: Cancel out the free bubbles.



Ex1: Minimizing Transistor Count

Step 4: Draw the gates in their conventional form.

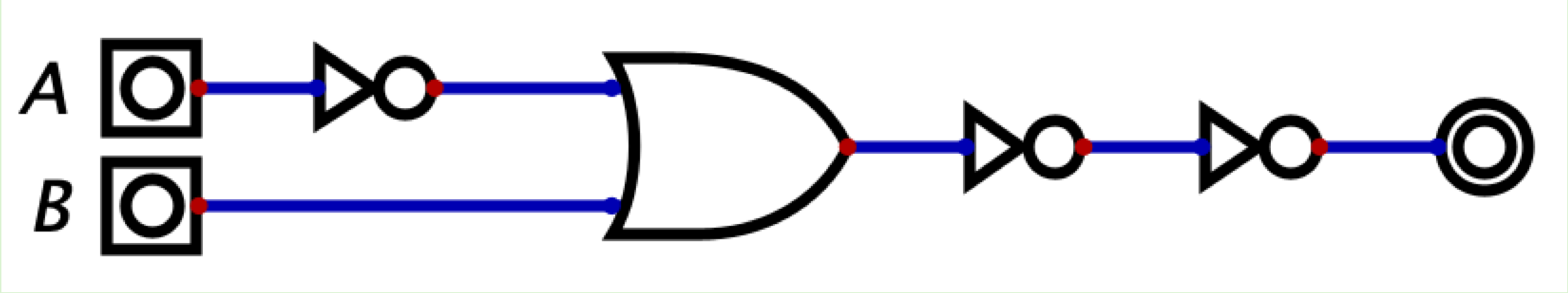


Original Transistor Count: 14

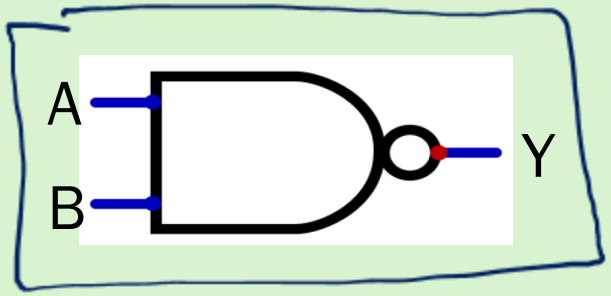
Final Transistor Count: 8

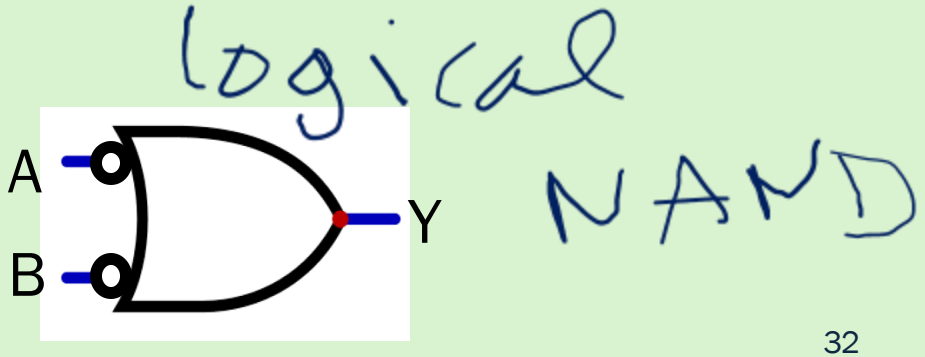
Minimizing Transistor Count

Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters. Compare the number of transistors in the original circuit and the new circuit.



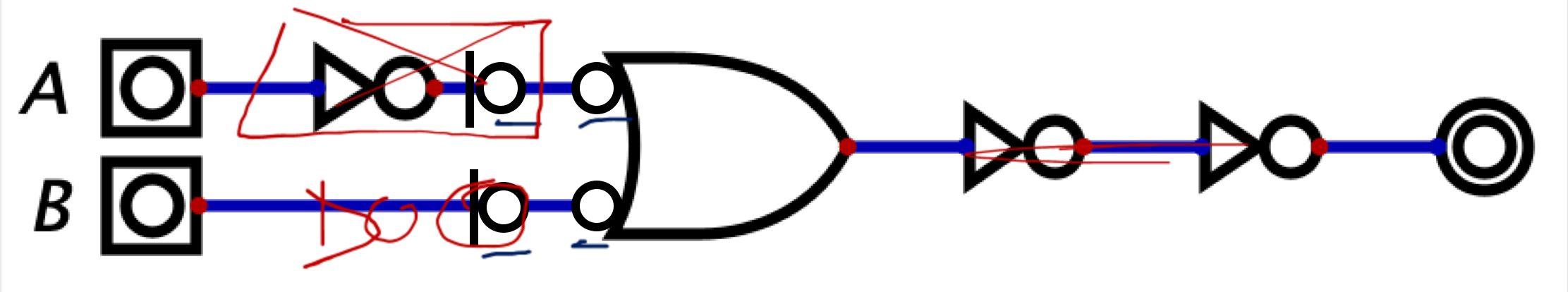
NAND gates:



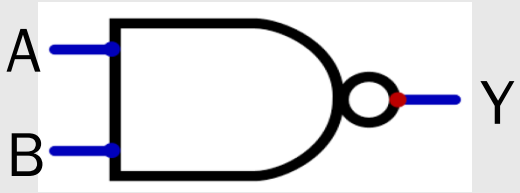


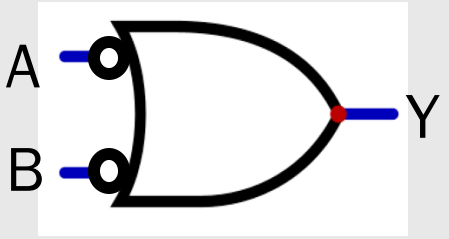
Minimizing Transistor Count

Step 1: Transform the OR gate into a NAND gate. Add bubbles to make the circuit logically equivalent.



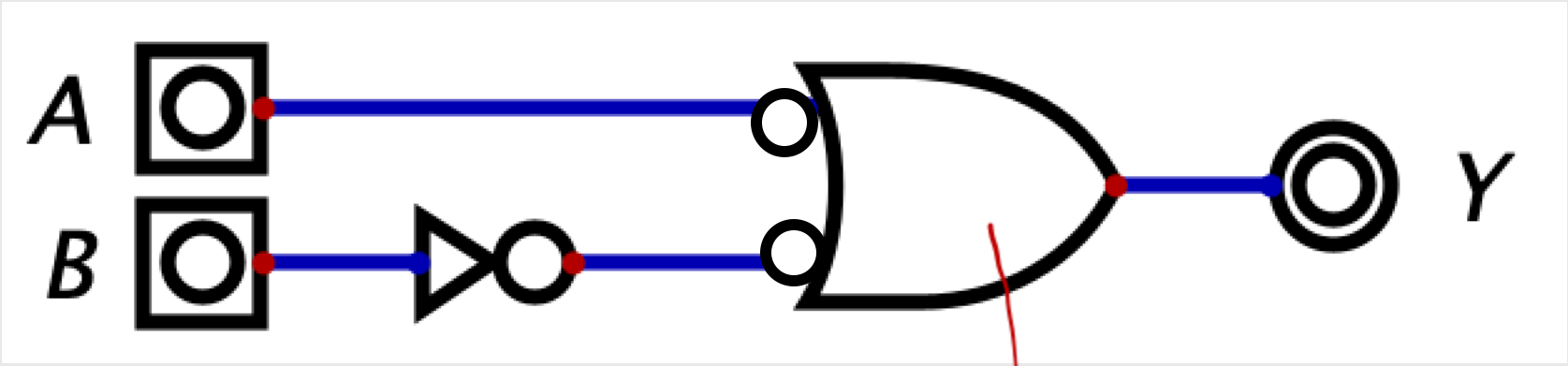
NAND gates:





Minimizing Transistor Count

Step 2: Cancel out extra bubbles and inverters.

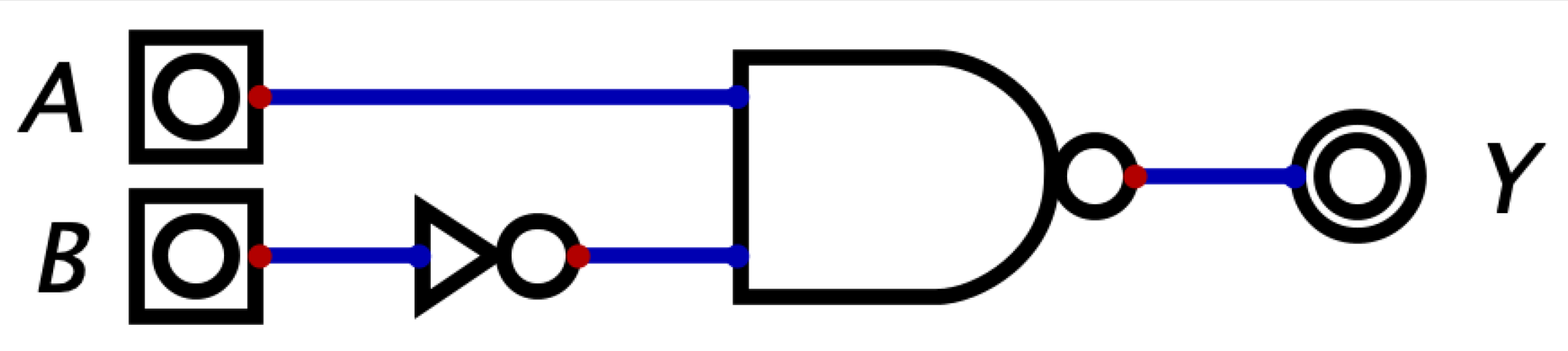


logical

NAND

Minimizing Transistor Count

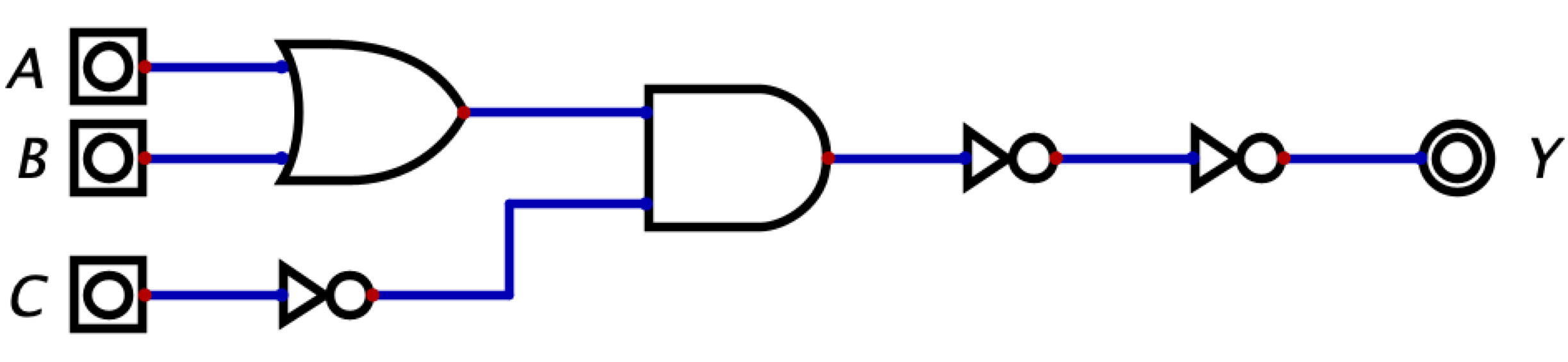
Step 3: Draw the gates in their conventional form.



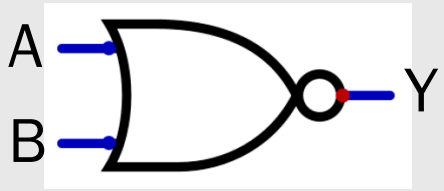
Original Transistor Count: 12
Final Transistor Count: 6

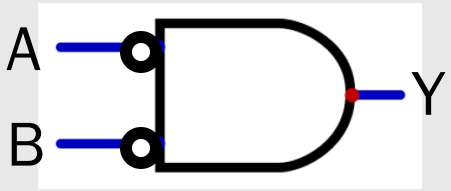
Ex2: Minimizing Transistor Count

Minimize the number of transistors in this circuit by redesigning it to use only NOR gates and inverters.



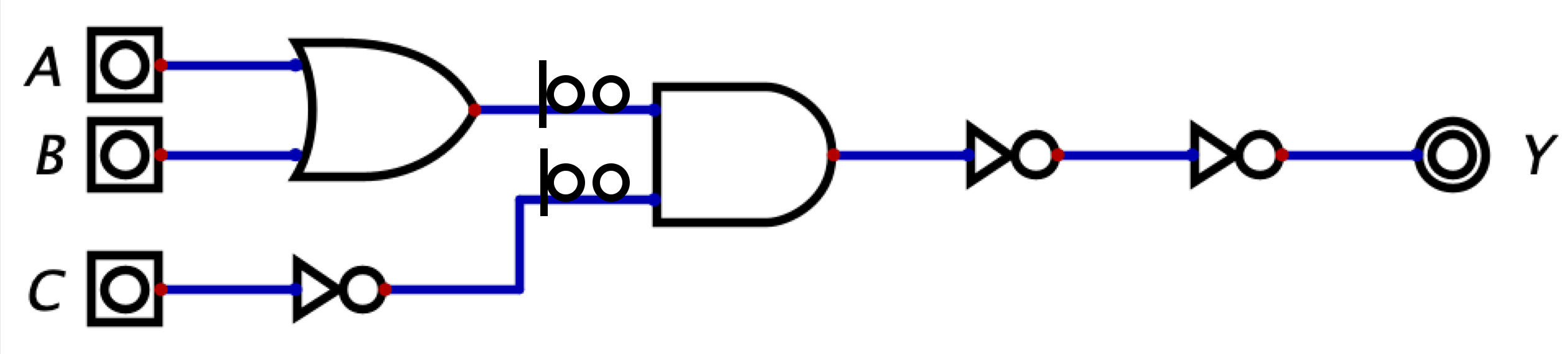
NOR gates:



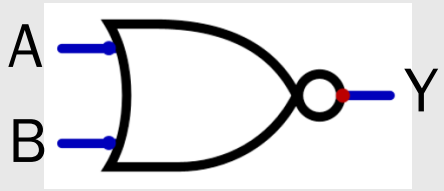


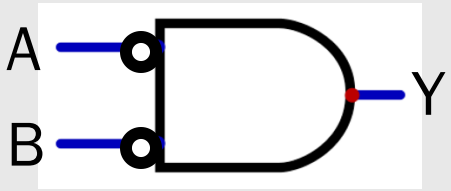
Ex2: Minimizing Transistor Count

Step 1: Start with the right-most gate that is not an inverter. Turn it into a NOR gate. Then add bubbles to make the circuit logically equivalent.



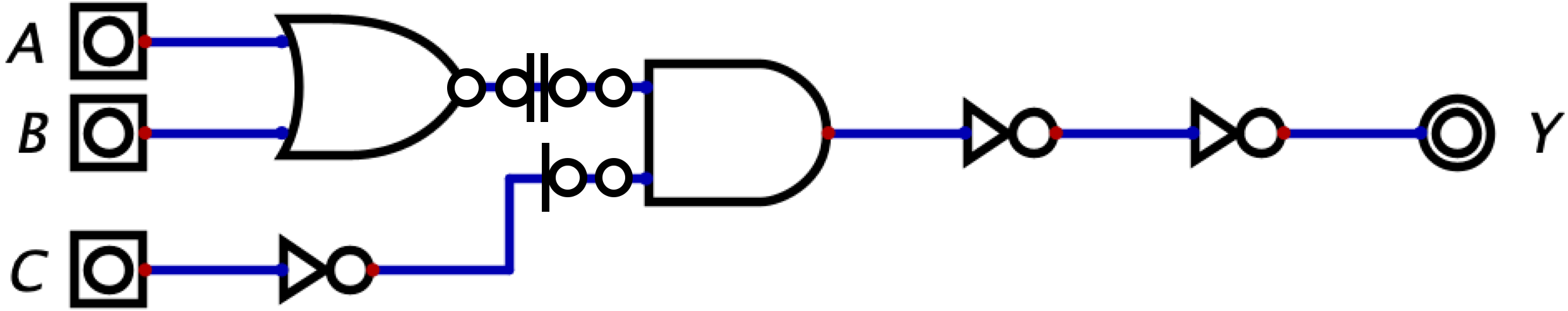
NOR gates:



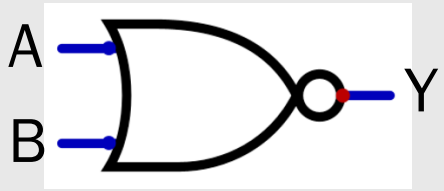


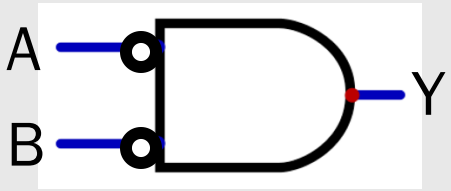
Ex2: Minimizing Transistor Count

Step 2: Move to the next gate on the left. Transform this into a NOR gate. Add a bubble to make the circuit logically equivalent.



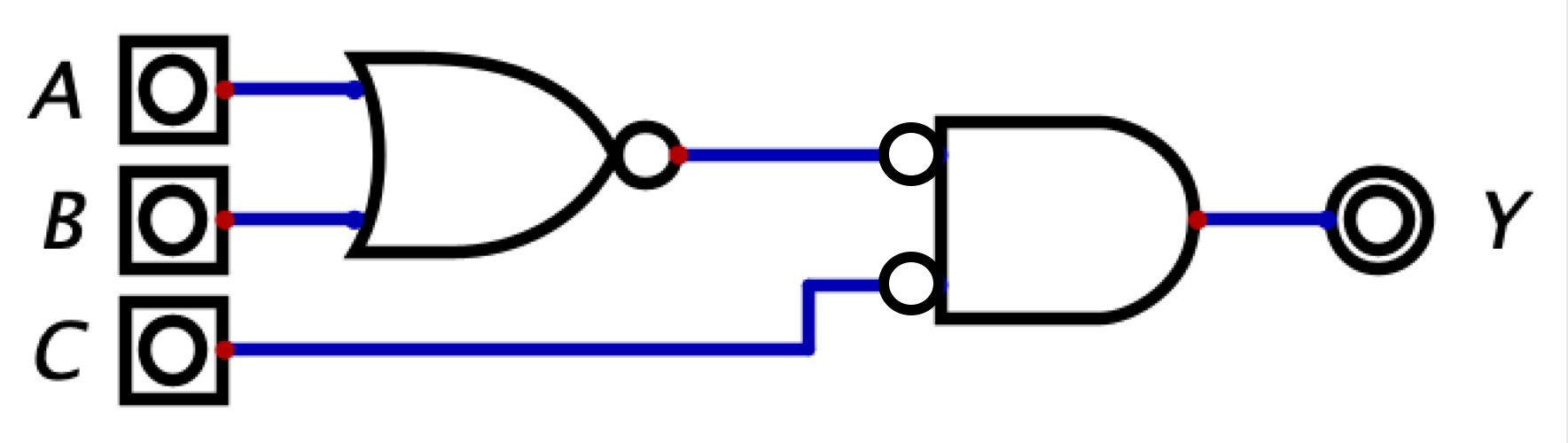
NOR gates:





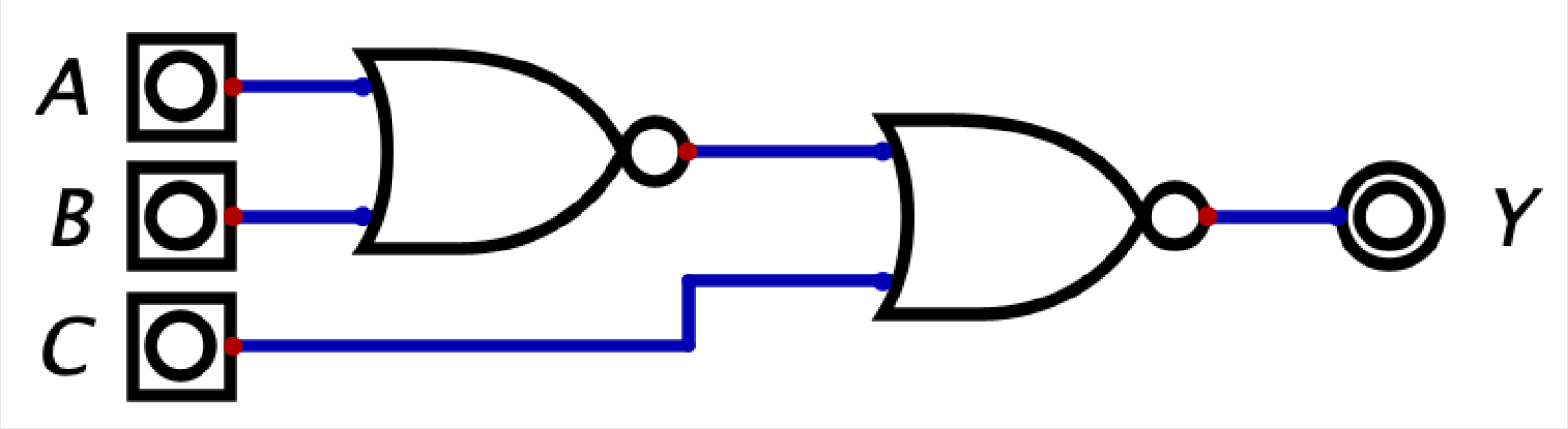
Ex2: Minimizing Transistor Count

Step 3: Cancel out the extra bubbles and inverters.



Ex2: Minimizing Transistor Count

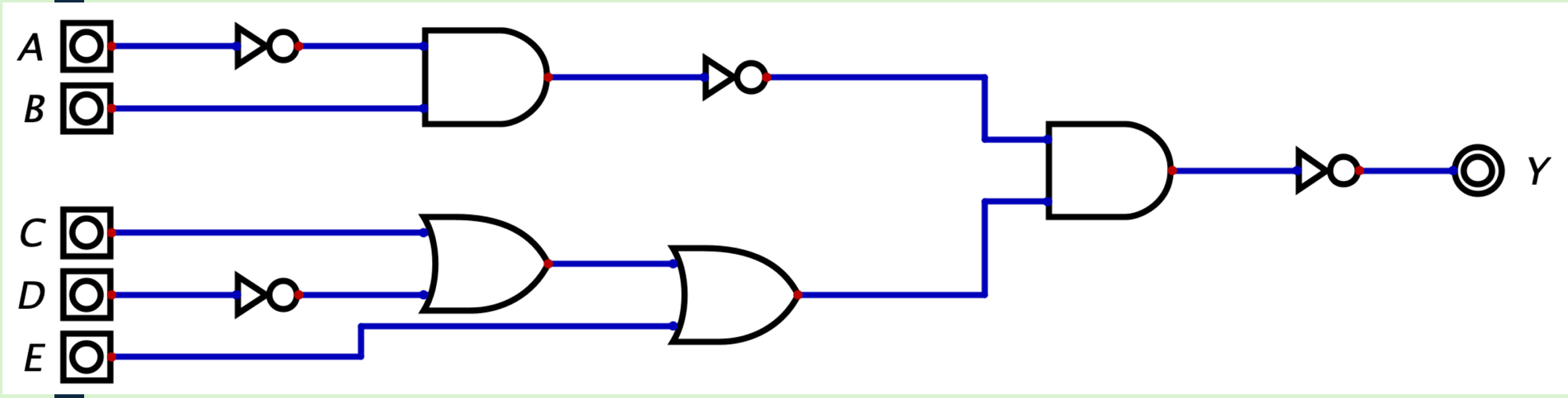
Step 4: Draw the gates in their conventional form.



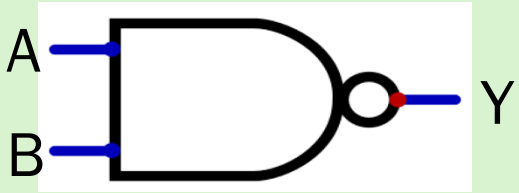
Original Transistor Count: 18
Final Transistor Count: 8

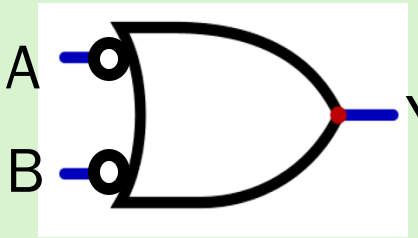
Minimizing Transistor Count

Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters. Compare the number of transistors in the original circuit and the new circuit.



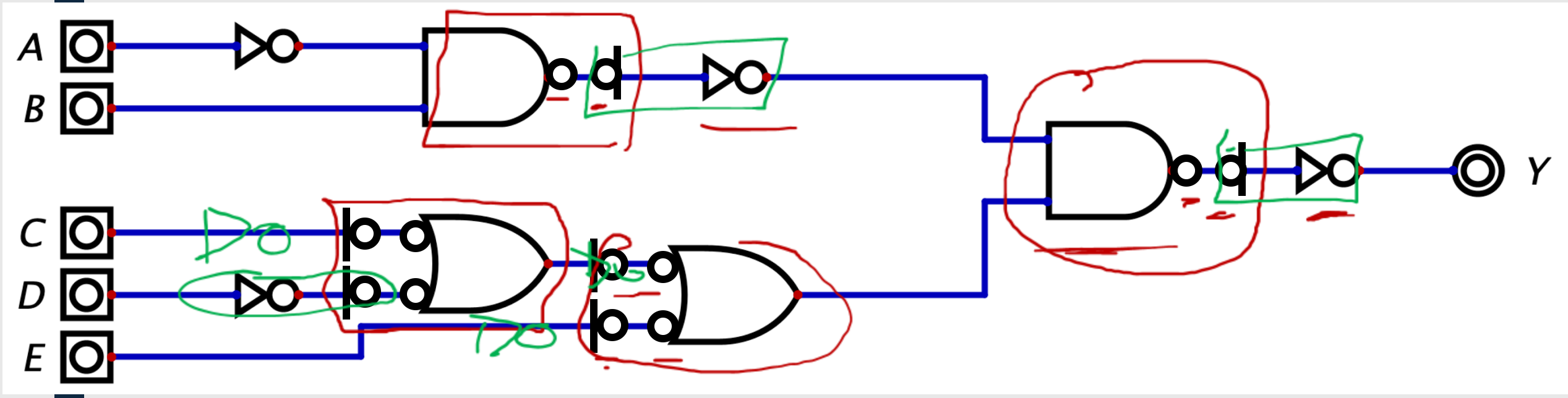
NAND gates:



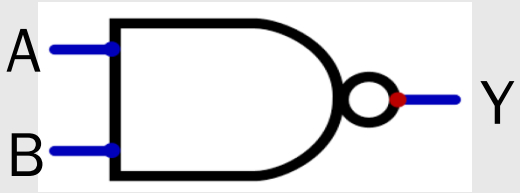


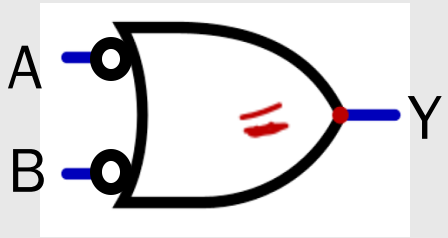
Minimizing Transistor Count

Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters. Compare the number of transistors in the original circuit and the new circuit.



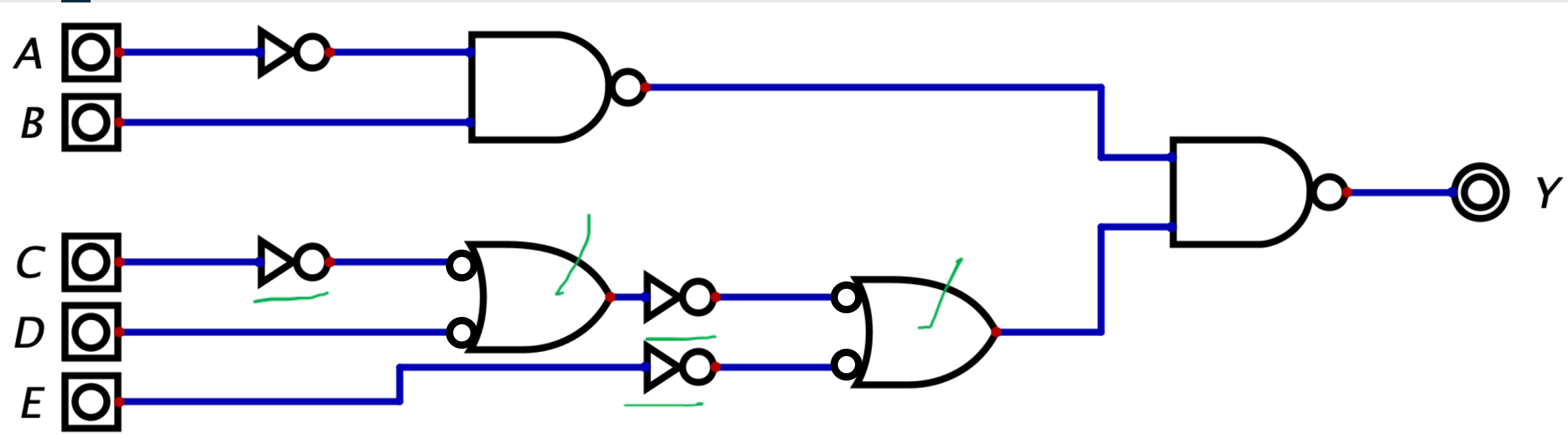
NAND gates:



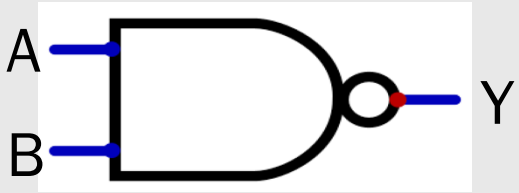


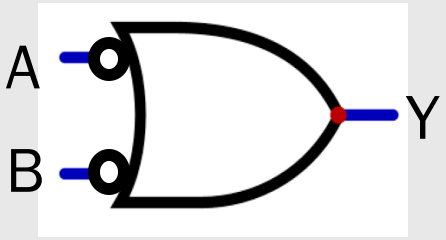
Minimizing Transistor Count

Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters. Compare the number of transistors in the original circuit and the new circuit.



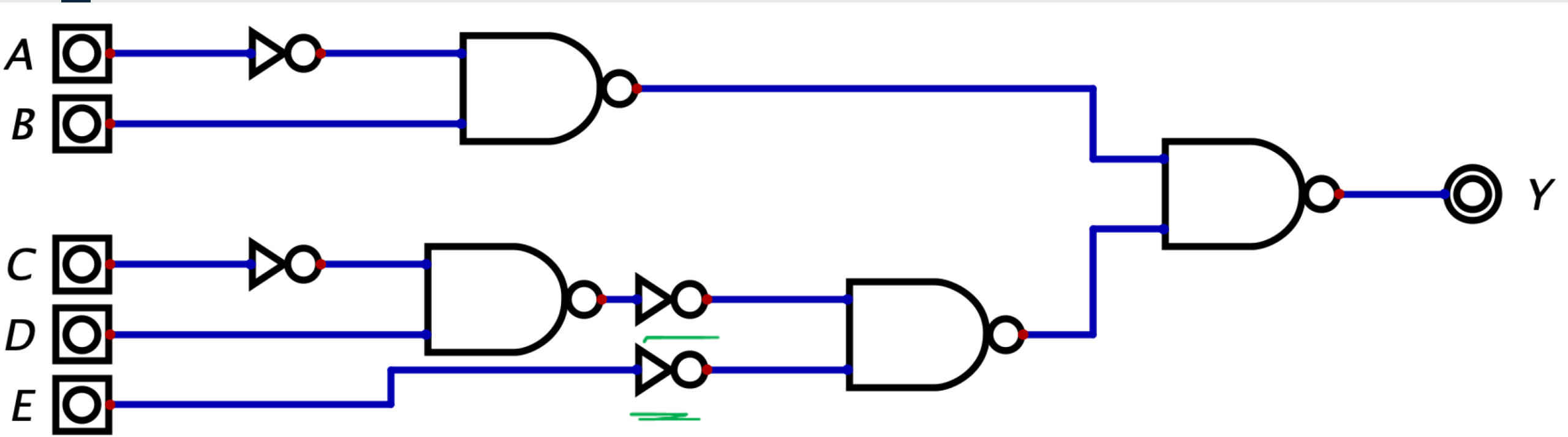
NAND gates:





Minimizing Transistor Count

Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters. Compare the number of transistors in the original circuit and the new circuit.

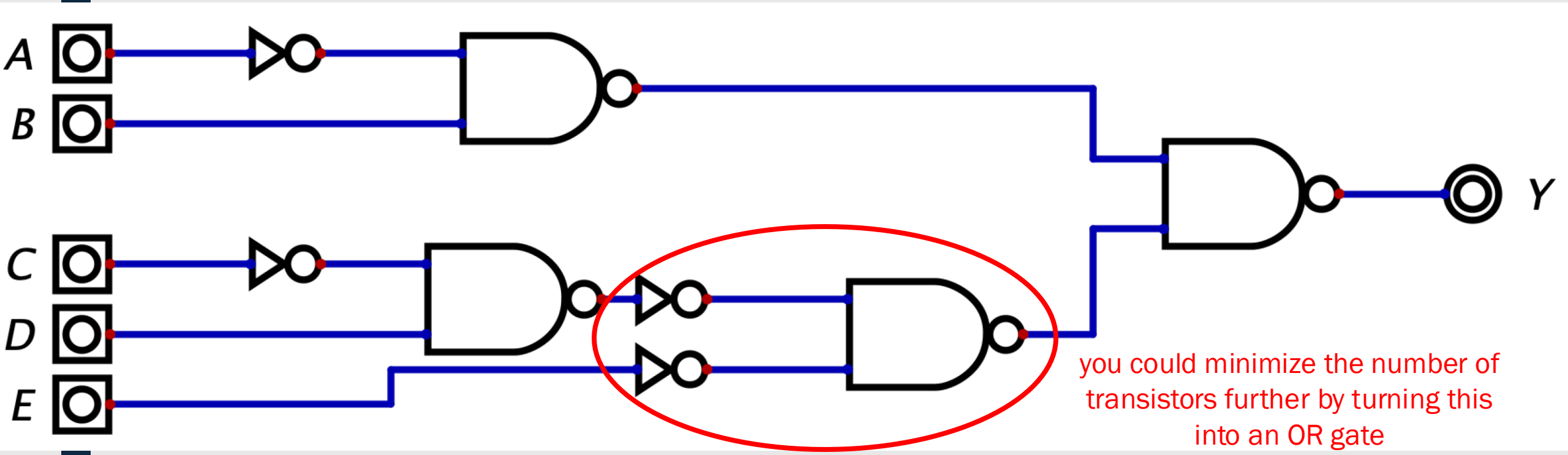


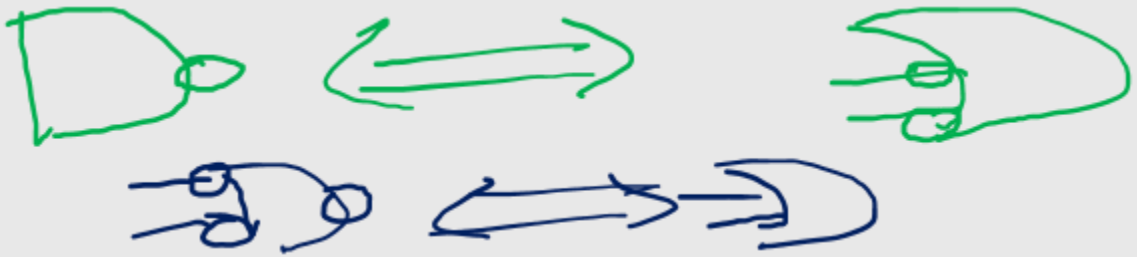
Original Transistor Count: 32

Final Transistor Count: 24

Minimizing Transistor Count

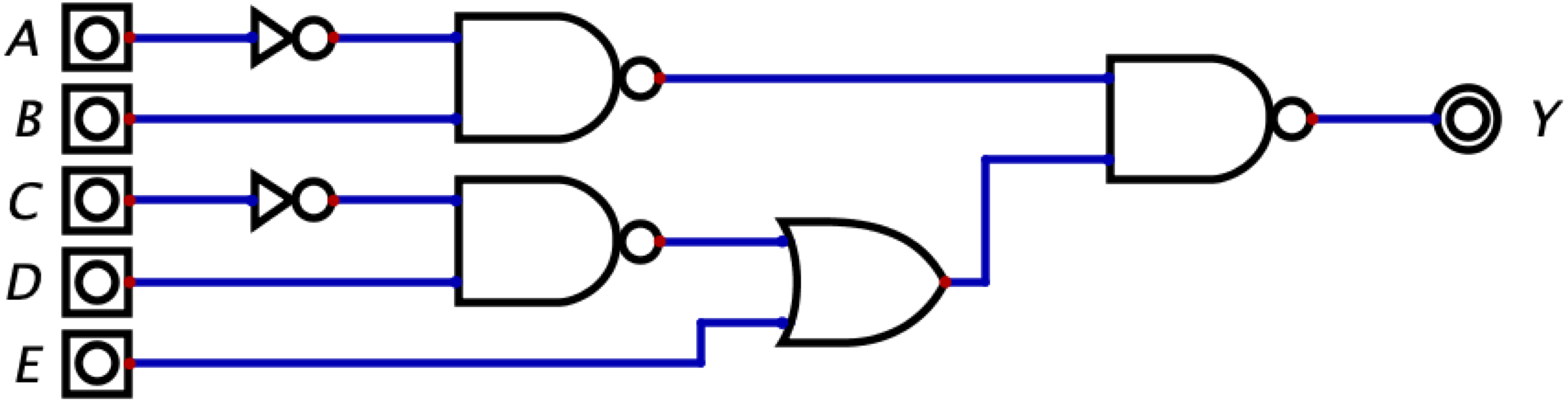
Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters. Compare the number of transistors in the original circuit and the new circuit.





Minimizing Transistor Count

Minimize the number of transistors in this circuit by redesigning it to use only NAND gates and inverters. Compare the number of transistors in the original circuit and the new circuit.

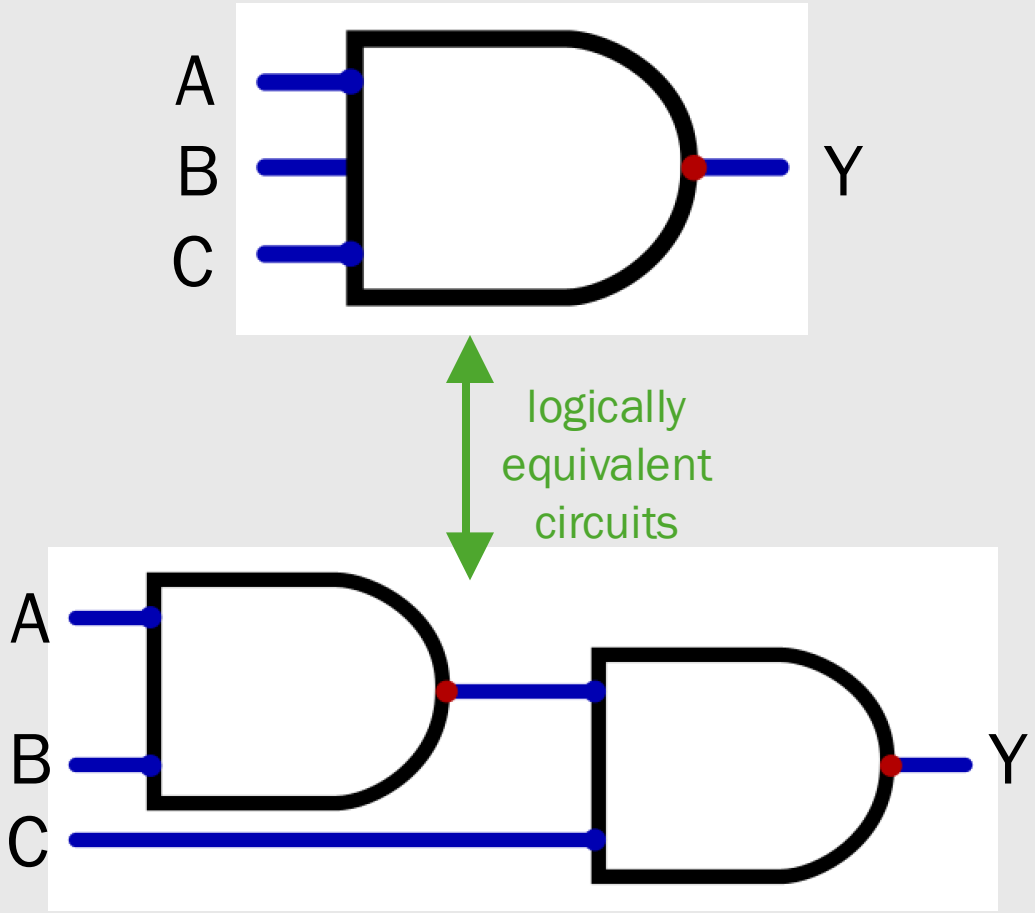


Original Transistor Count: 32

Final Transistor Count: 22

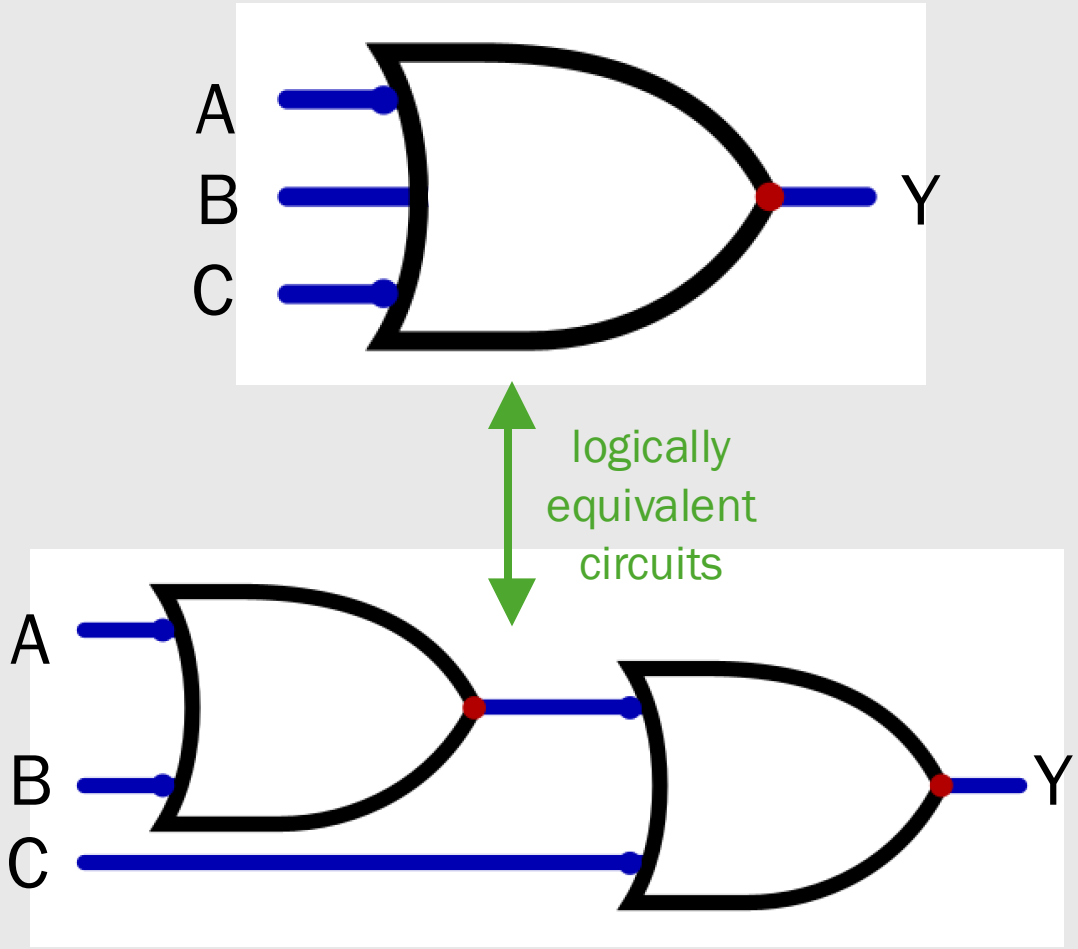
GATES WITH 2+ INPUTS

3-input AND



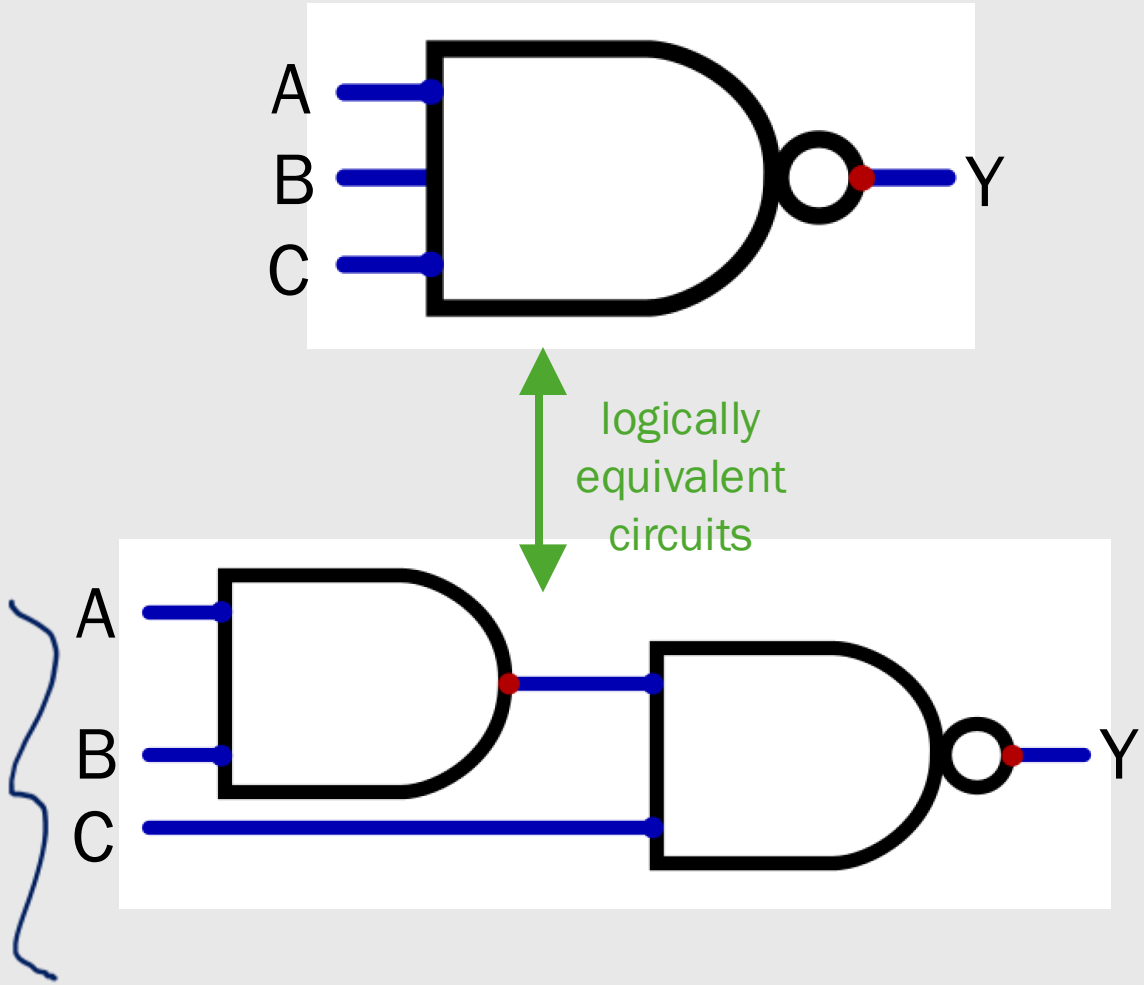
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3-input OR



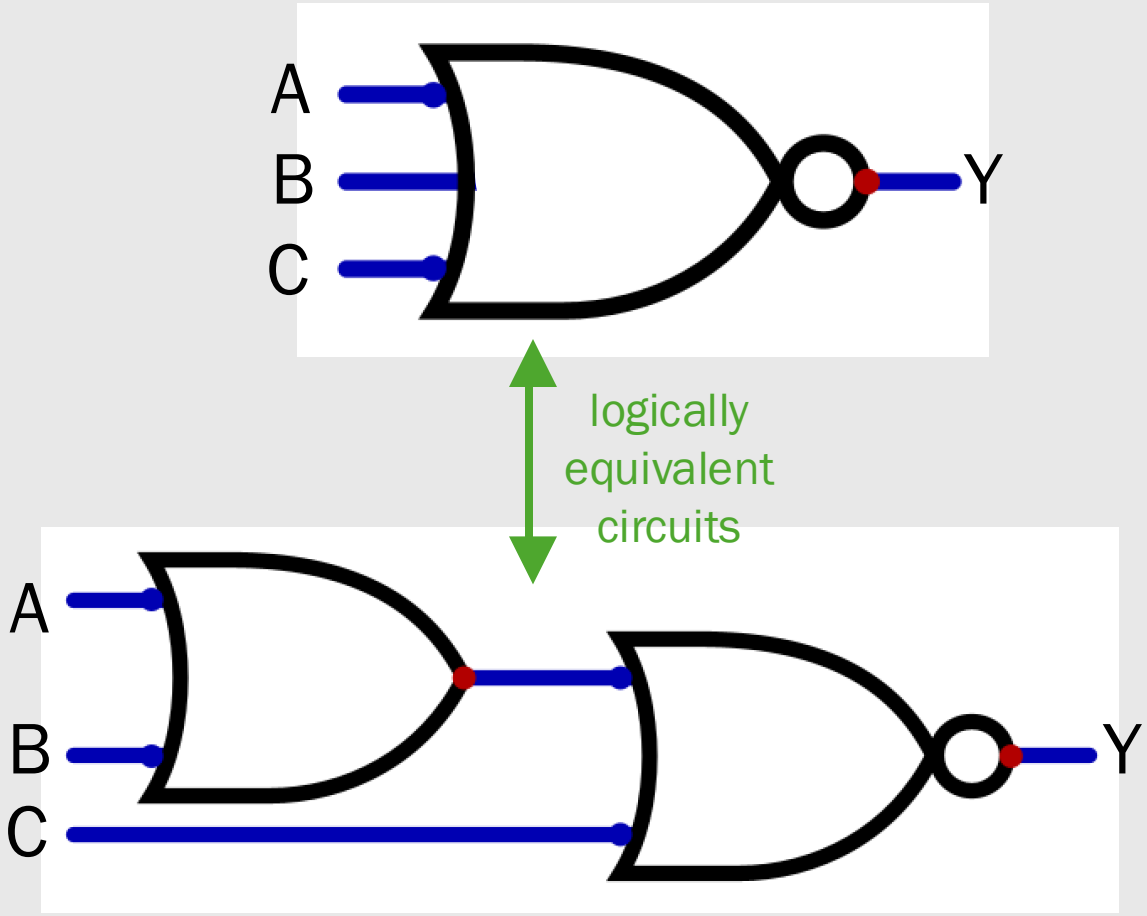
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

3-input NAND



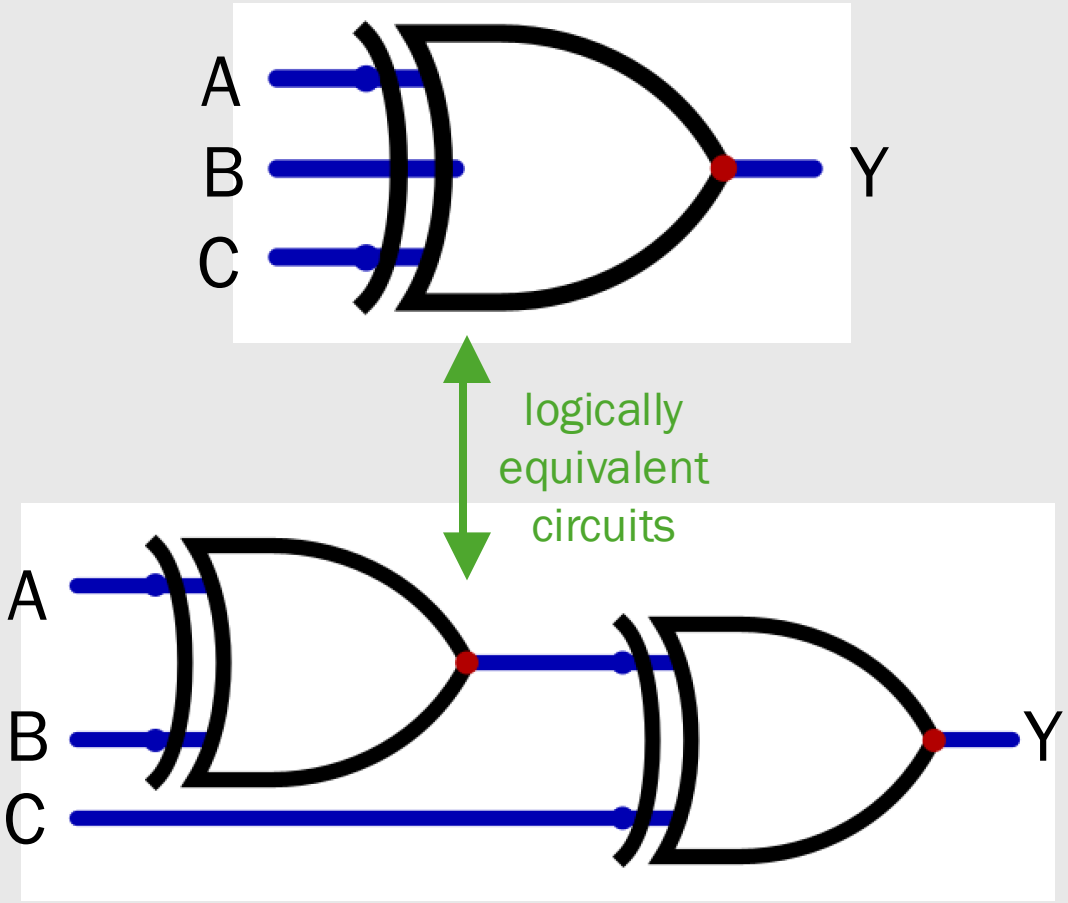
A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

3-input NOR



A	B	C	Y
0	0	0	1
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	0

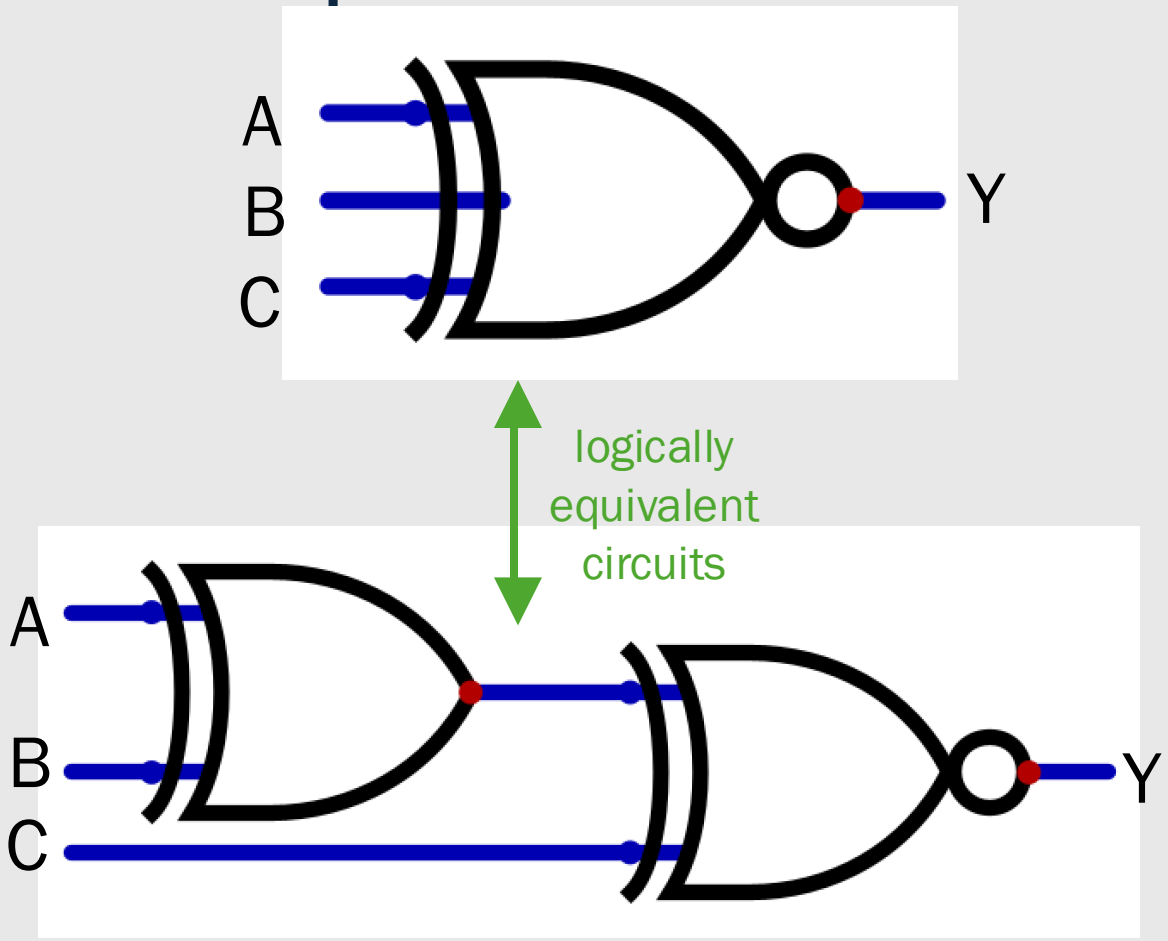
3-input XOR



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

3-input XNOR



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

A Systematic Design Approach

Truth Table

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

-it's systematic!
-it works!

1) Write the functional spec as a truth table
2) Write down a Boolean expression for every '1' in the output

$$Y = (\overline{C} * \overline{B} * A) + (\overline{C} * B * A) + (C * B * \overline{A}) + (C * B * A)$$

3) Wire up the ideal gates, replace them with equivalent realizable gates, and we're done!

This approach will always give us logic expressions in a particular form:

SUM-OF-PRODUCTS!

Straightforward Synthesis

Truth Table

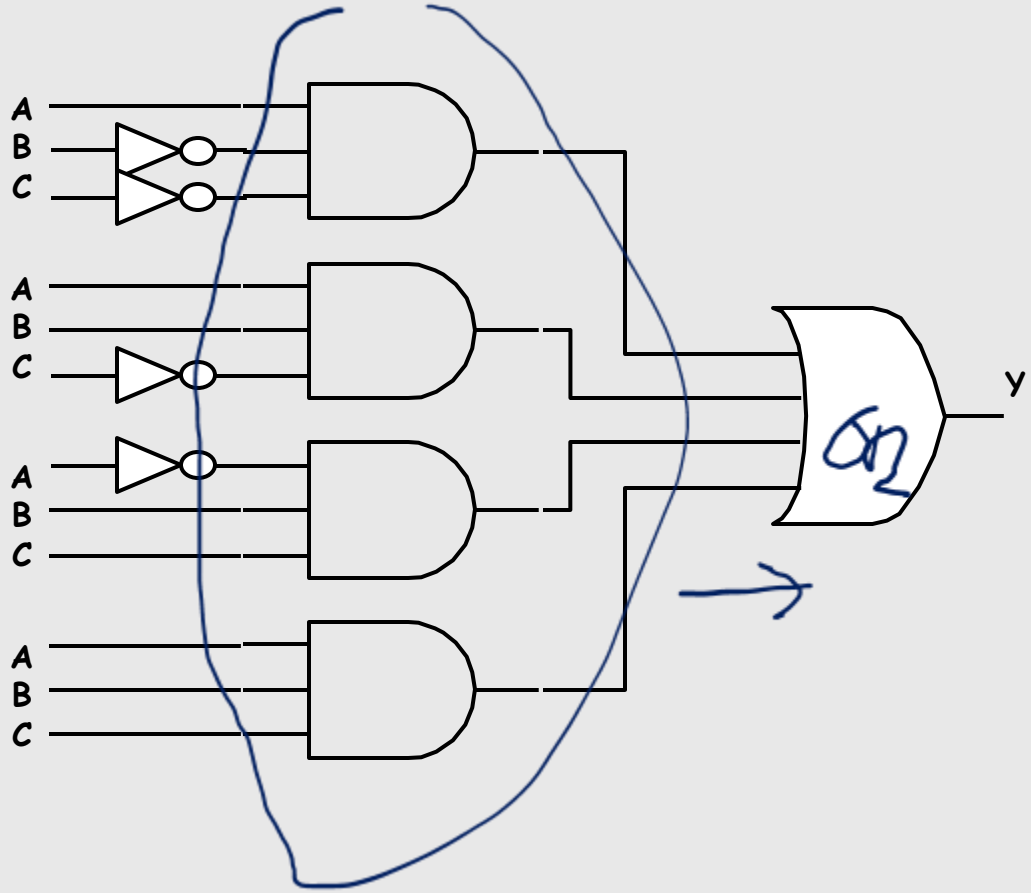
C	B	A	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

We can implement

SUM-OF-PRODUCTS

with just 3 levels of logic.

INVERTERS/AND/OR



$$Y = (\overline{C} * \overline{B} * A) + (\overline{C} * B * A) + (C * B * \overline{A}) + (C * B * A)$$

A_1	A_0	B_1	B_0	Y
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

I want to build a circuit that will output 1 if A > B, and 0 if A <= B. A and B are both two-bit unsigned numbers.

Additionally, define A_1 and B_1 as the left bits of A and B, and A_0 and B_0 as the right bits.

$$y = \bar{A_1} A_0 \bar{B_1} \bar{B_0}$$

Let's Try it Out

A_0	A_1	B_0	B_1	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

I want to build a circuit that will output 1 if A > B, and 0 if A <= B. A and B are both two-bit unsigned numbers.

Additionally, define A_1 and B_1 as the left bits of A and B, and A_0 and B_0 as the right bits.

$$\bar{A}_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 B_0 + A_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 \bar{B}_0$$

Simplify?

$$\bar{A}_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 \bar{B}_0 + A_1 \bar{A}_0 \bar{B}_1 B_0 + A_1 A_0 \bar{B}_1 \bar{B}_0 + A_1 A_0 \bar{B}_1 B_0 + A_1 A_0 B_1 \bar{B}_0$$

Let's Try it Out

A_0	A_1	B_0	B_1	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

I want to build a circuit that will output 1 if A > B, and 0 if A <= B. A and B are both two-bit unsigned numbers.

Additionally, define A_1 and B_1 as the left bits of A and B, and A_0 and B_0 as the right bits.

$$A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$$