# COMP311: *COMPUTER ORGANIZATION!*

Lecture 12: ALU!!

tinyurl.com/comp311-fa25

# 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

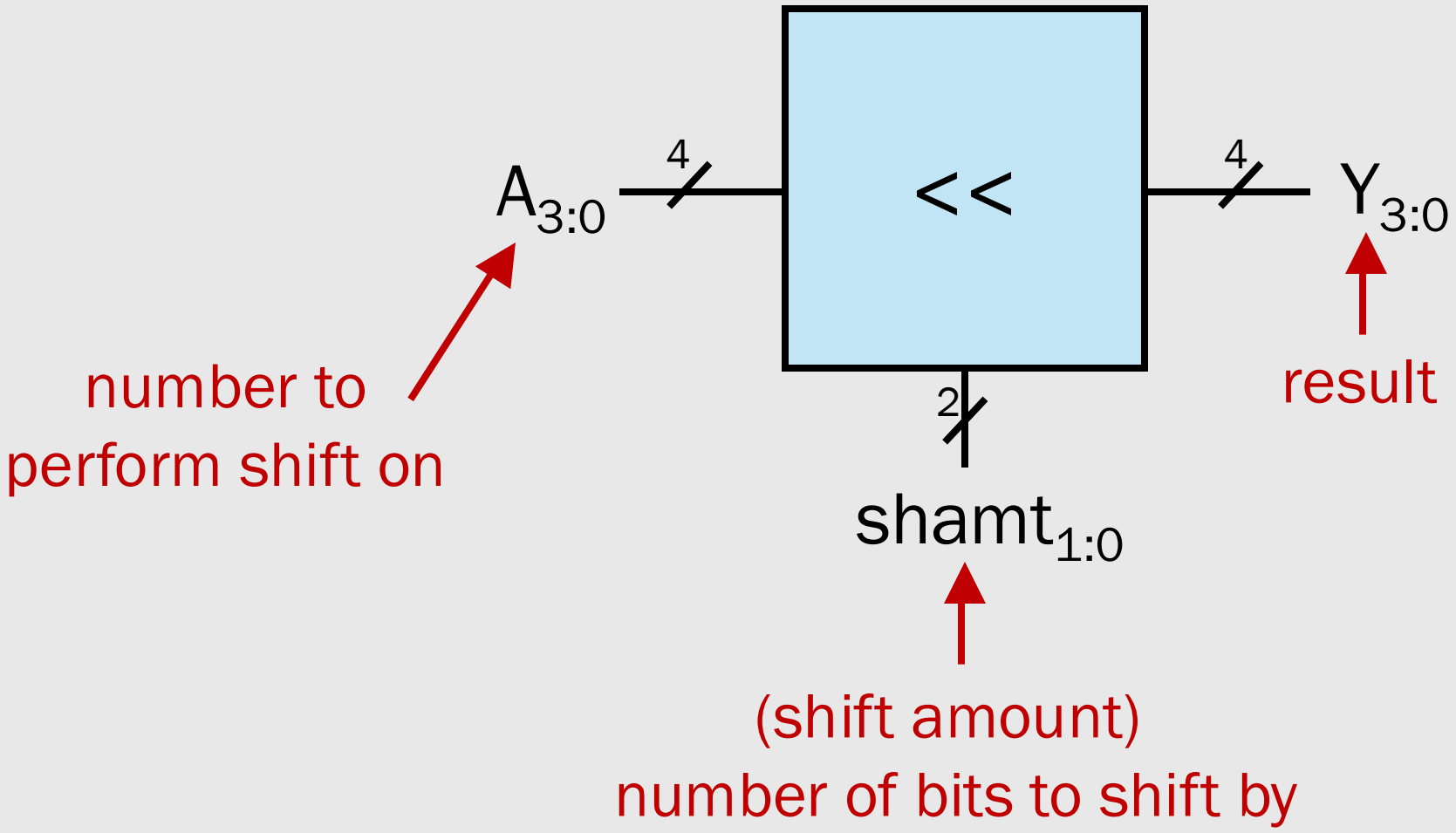
- If I have an n-bit number, how many bits are needed for the shift amount?

# 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

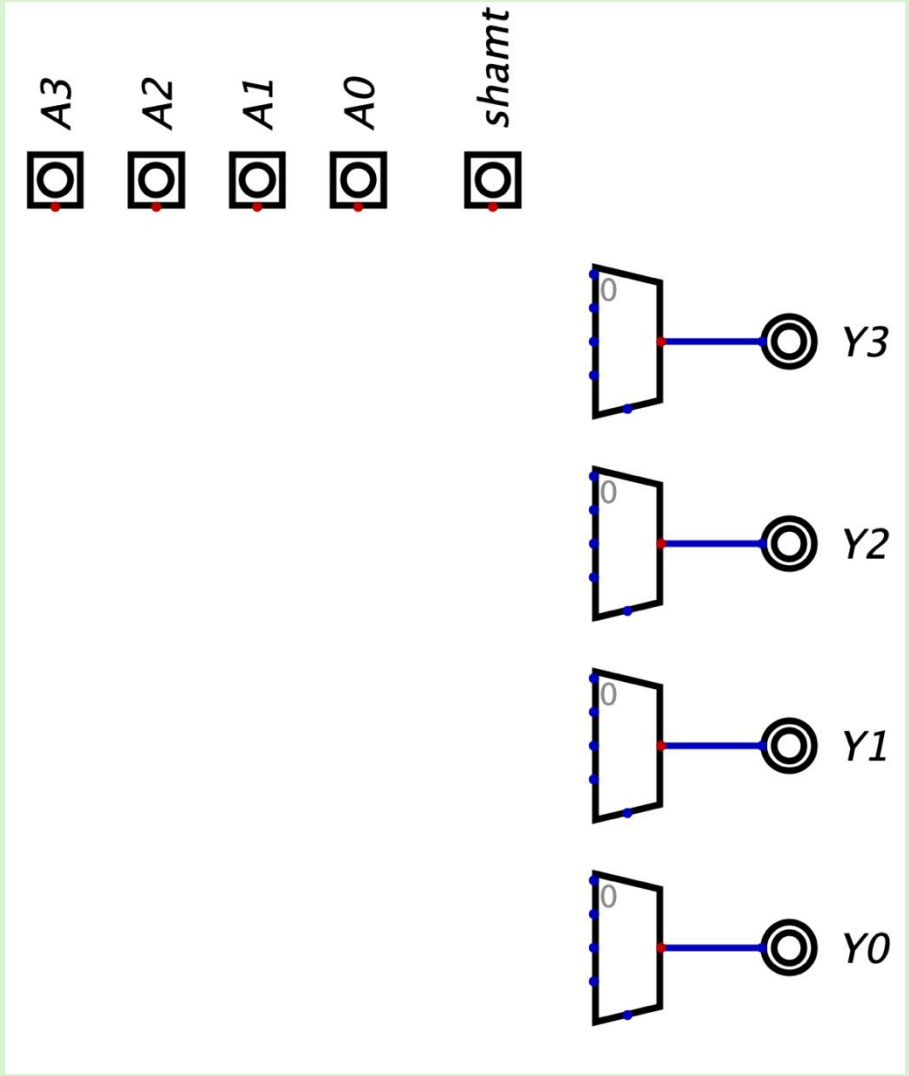
Output Value

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 |

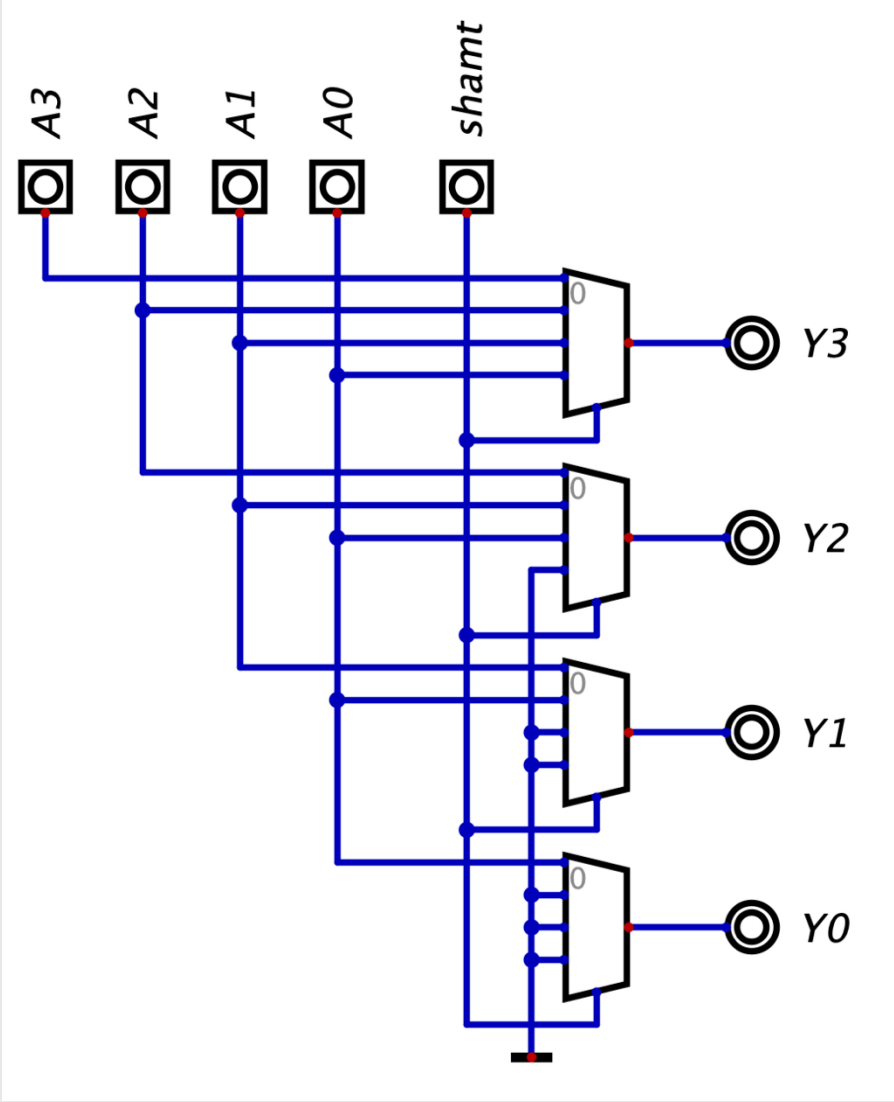
# Left Shift

- 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*

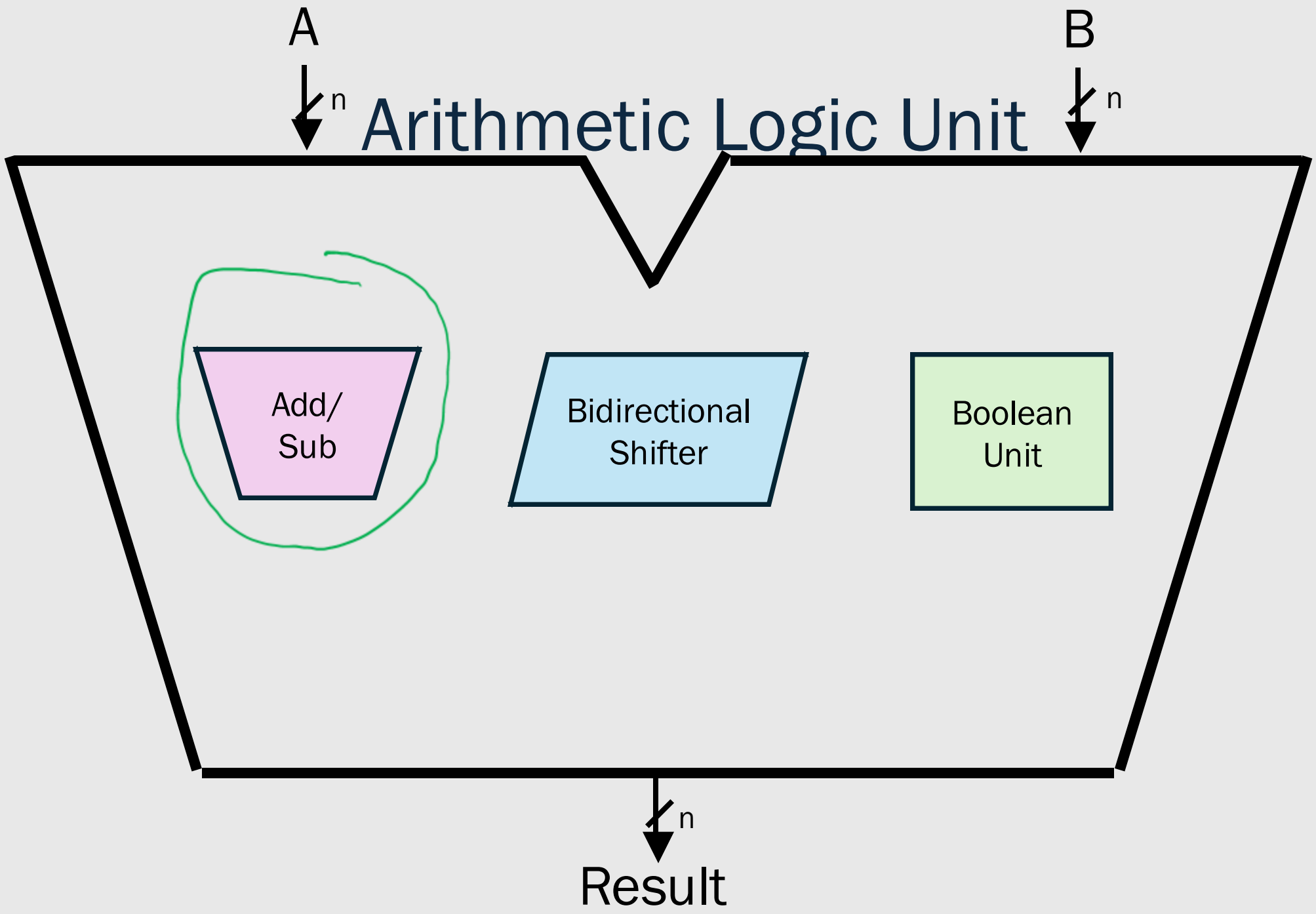


# Left Shift

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

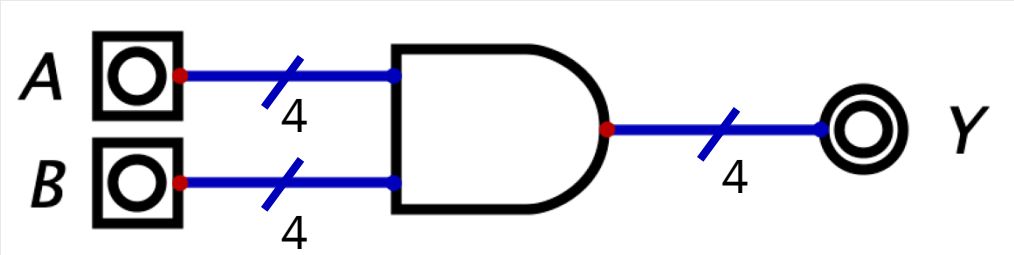


# ARITHMETIC LOGIC UNIT (ALU) ☺

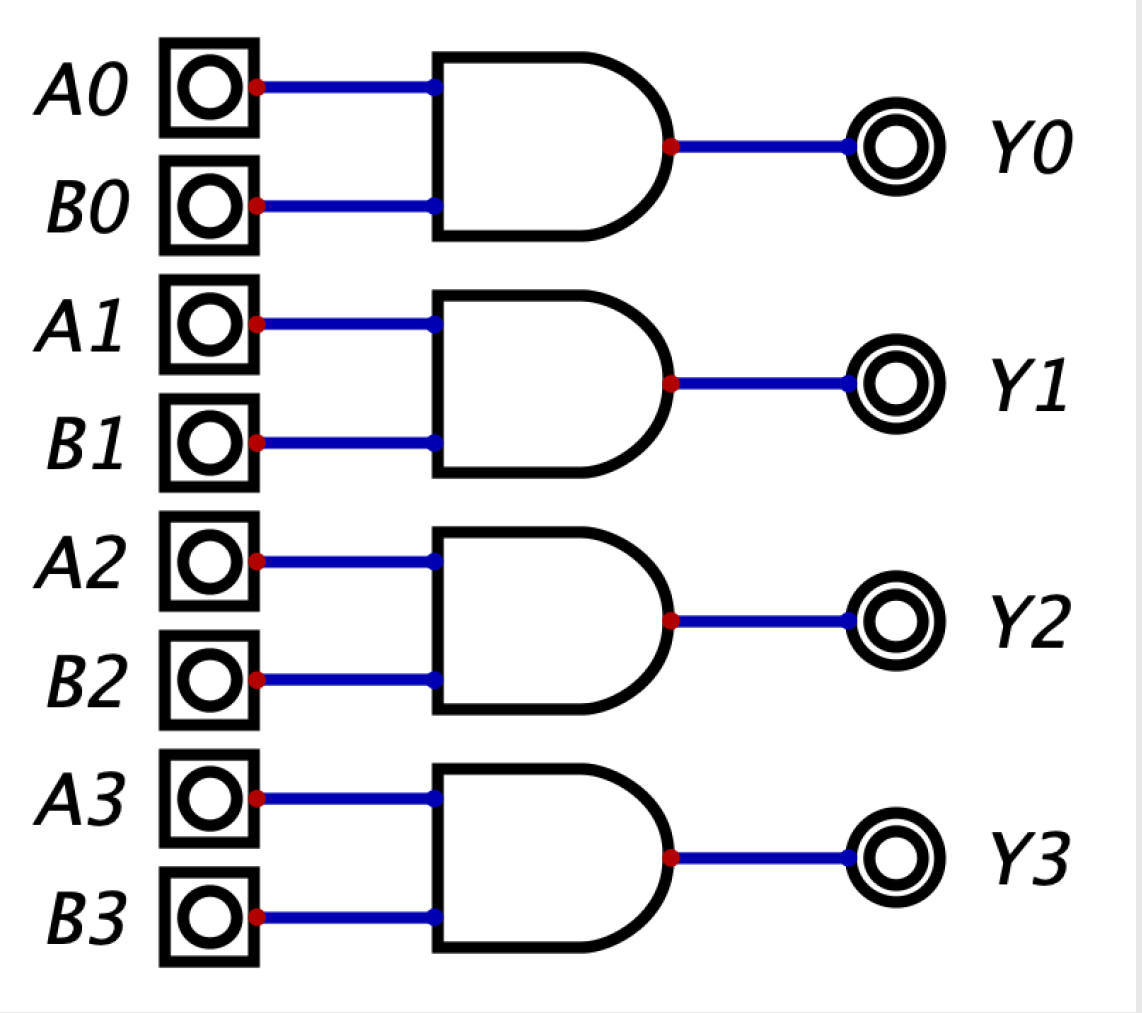

A
n
Arithmetic Logic Unit
B
n
Add/
Sub
Bidirectional
Shifter
Boolean
Unit
n
Result

# 4-bit AND

Symbol

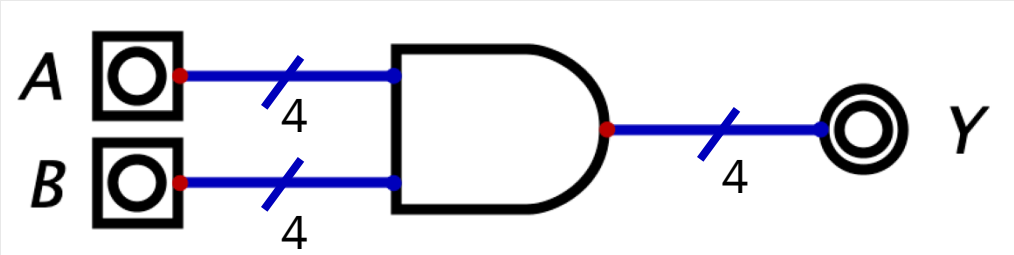



Implementation

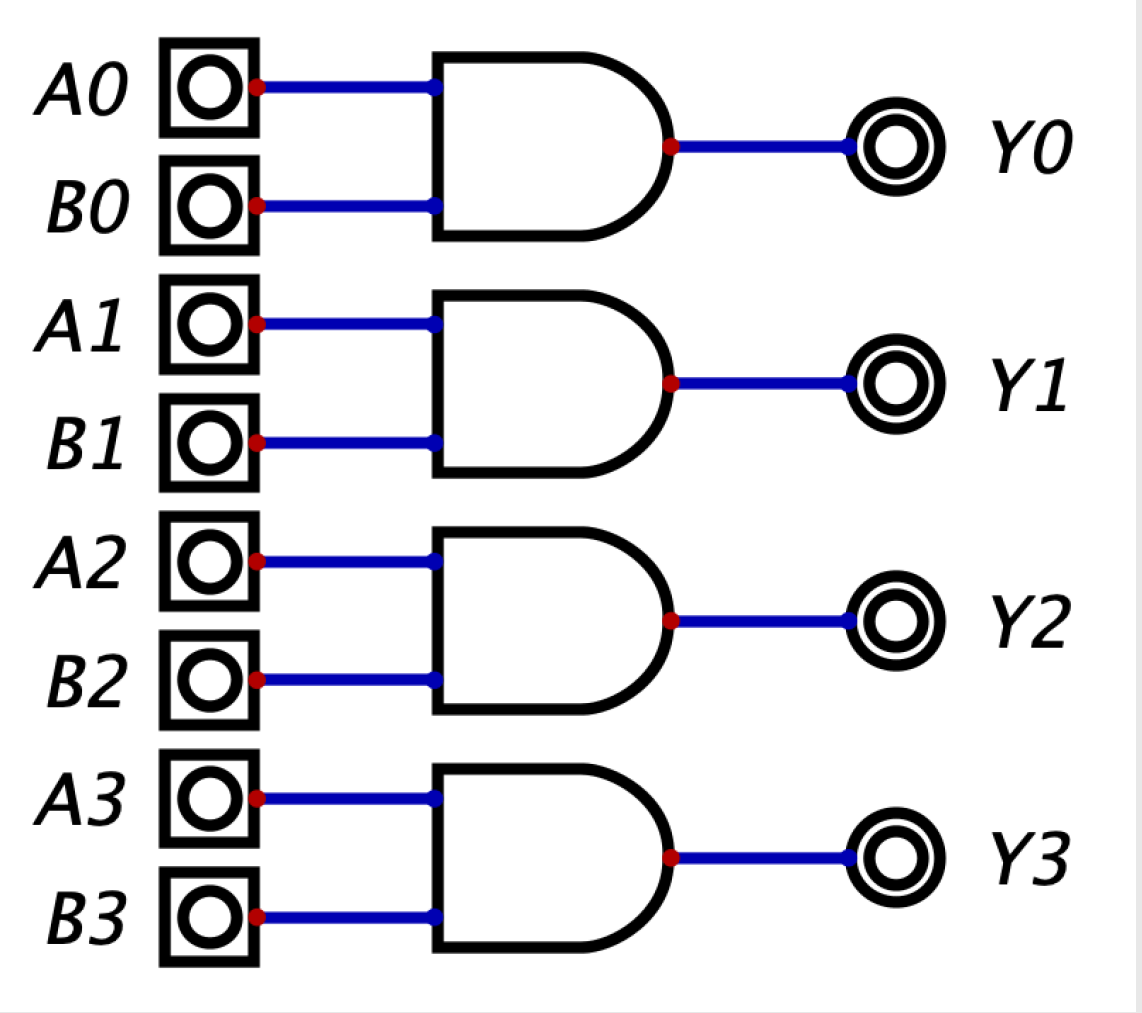


# Ex: 0b1111 AND 0b1010

Symbol

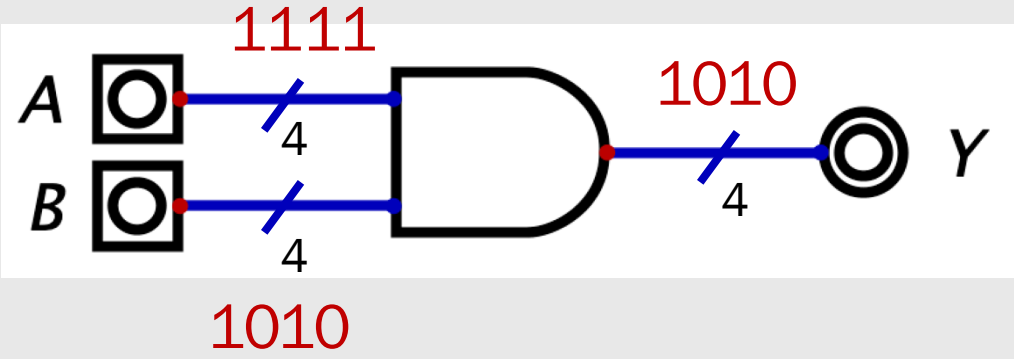



Implementation

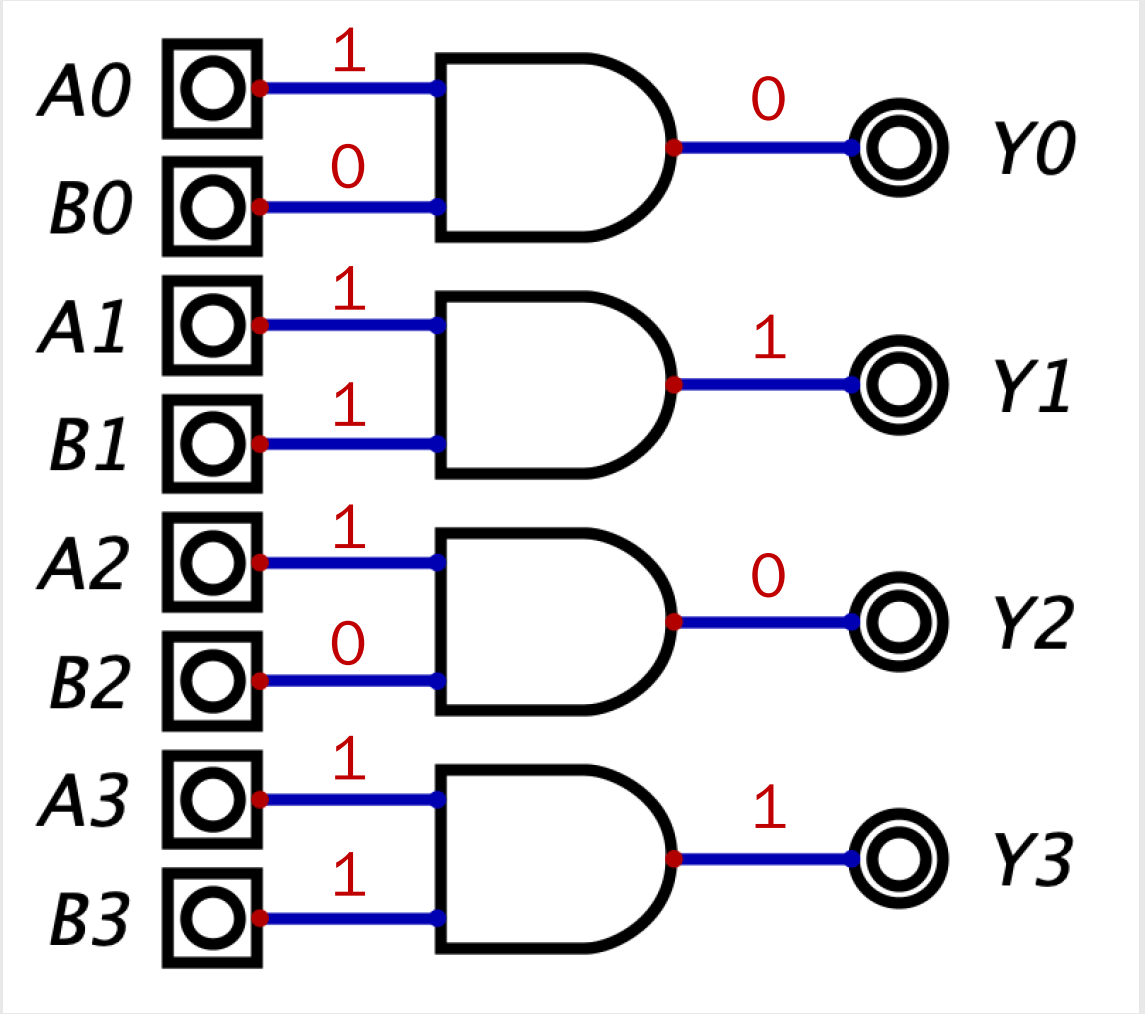


# Ex: 0b1111 AND 0b1010

Symbol

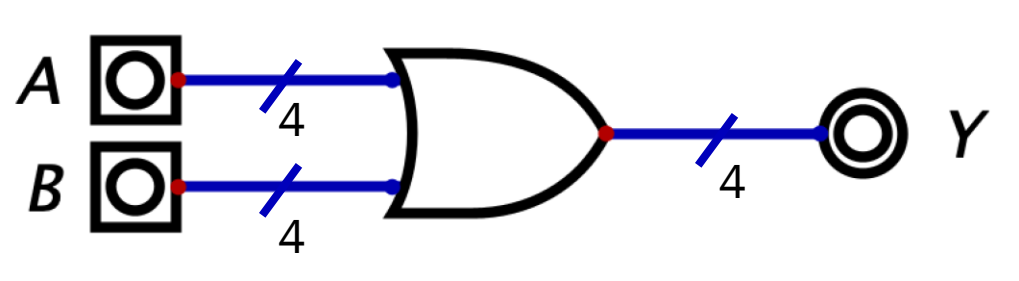



Implementation

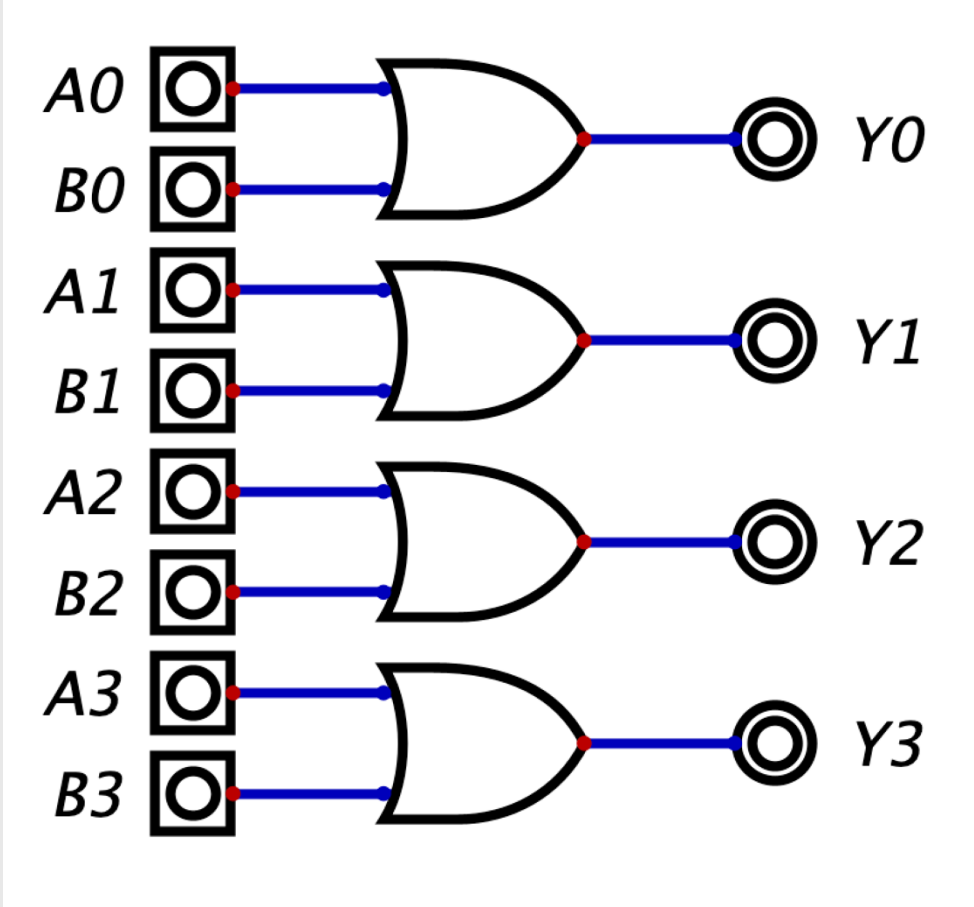


# 4-bit OR

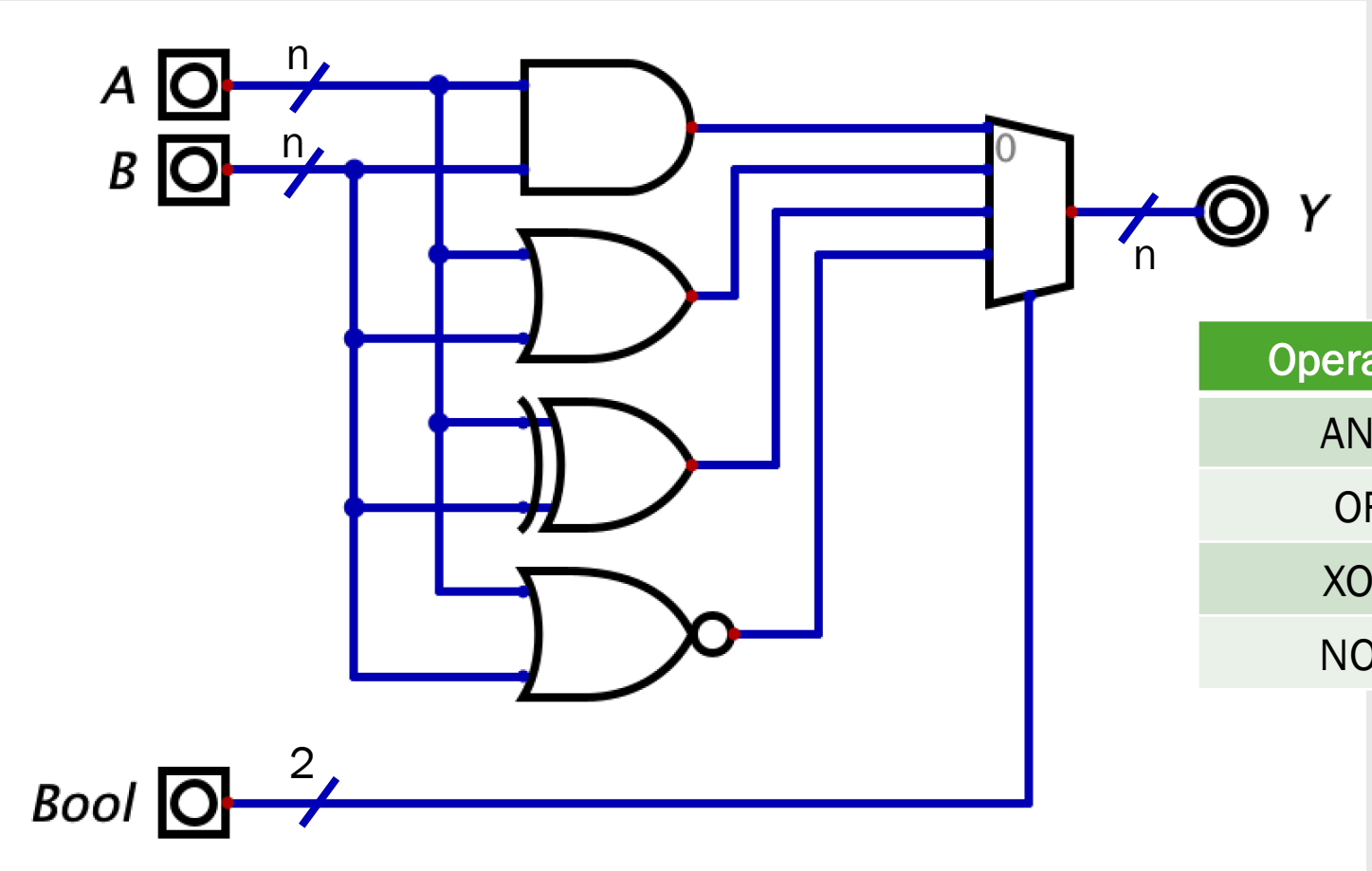
Symbol




Implementation

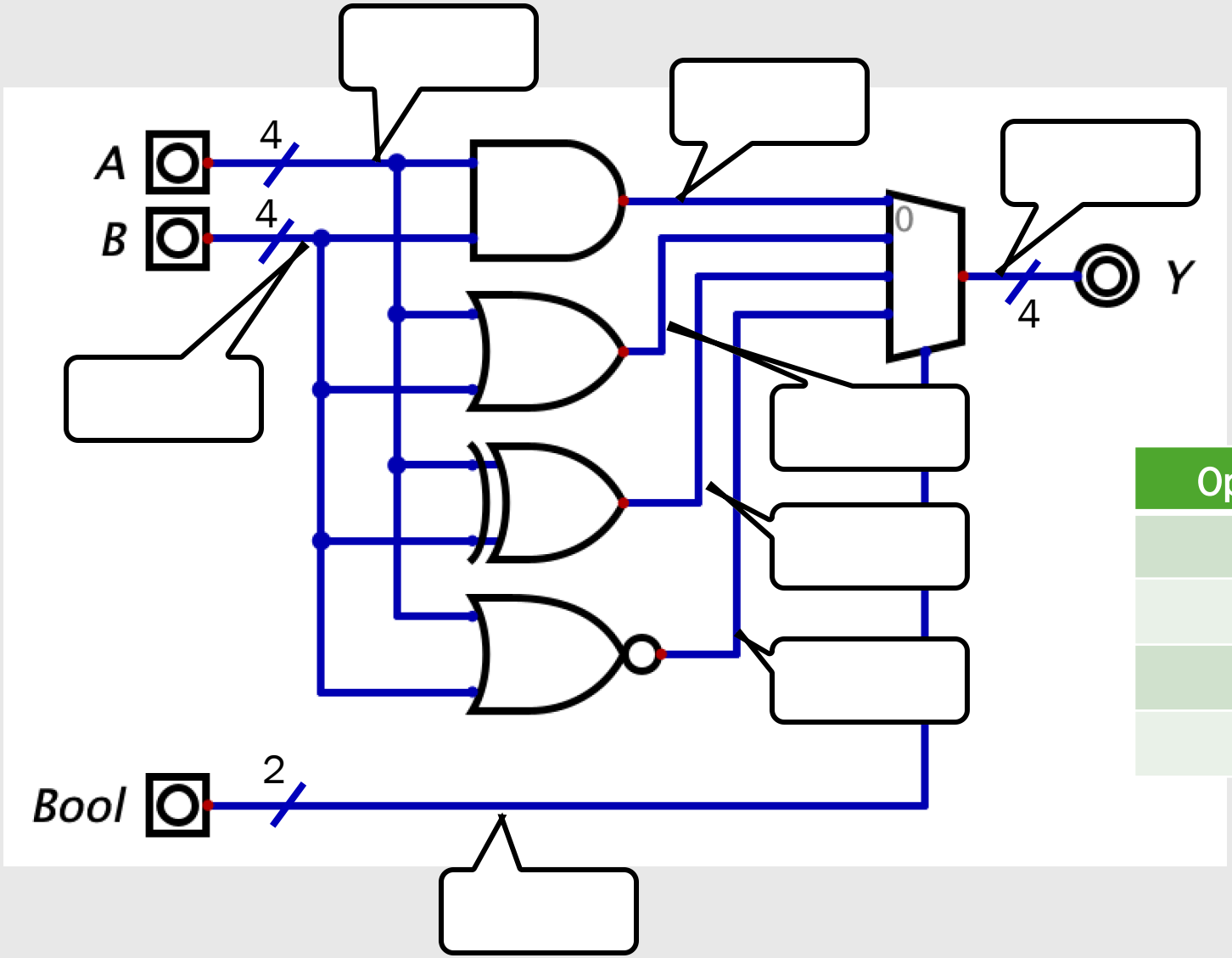


# Boolean Unit



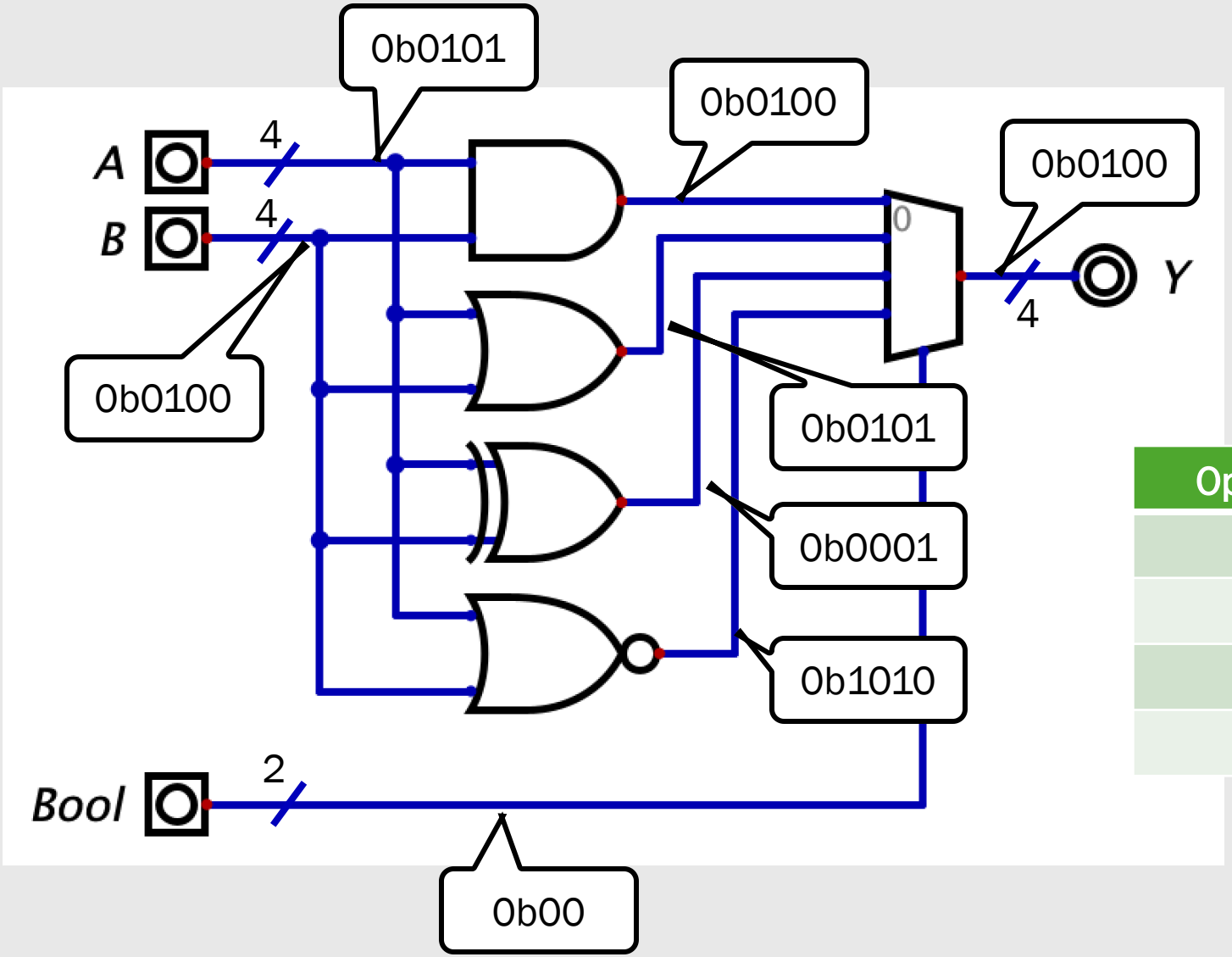

| Operation | Expression | Bool |
|---|---|---|
| AND | A x B | 0b00 |
| OR | A + B | 0b01 |
| XOR | $A \oplus B$ | 0b10 |
| NOR | $\overline{A + B}$ | 0b11 |

Let's say n = 4. I want to perform Y = A AND B where A is 5 and B is 4.



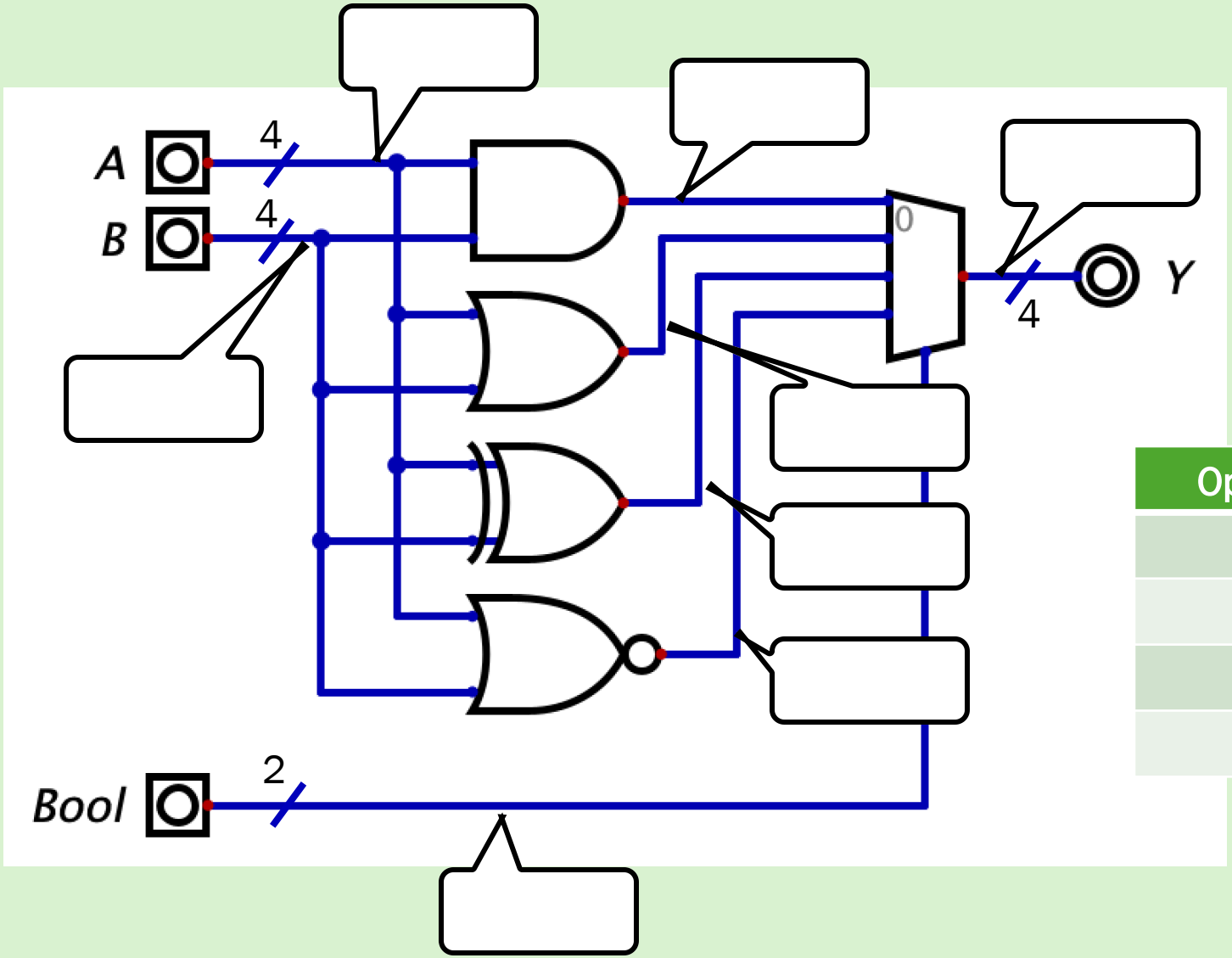

| Operation | Expression | Bool |
|---|---|---|
| AND | A x B | 0b00 |
| OR | A + B | 0b01 |
| XOR | A⊕B | 0b10 |
| NOR | $\overline{A + B}$ | 0b11 |

Let's say n = 4. I want to perform Y = A AND B where A is 5 and B is 4.



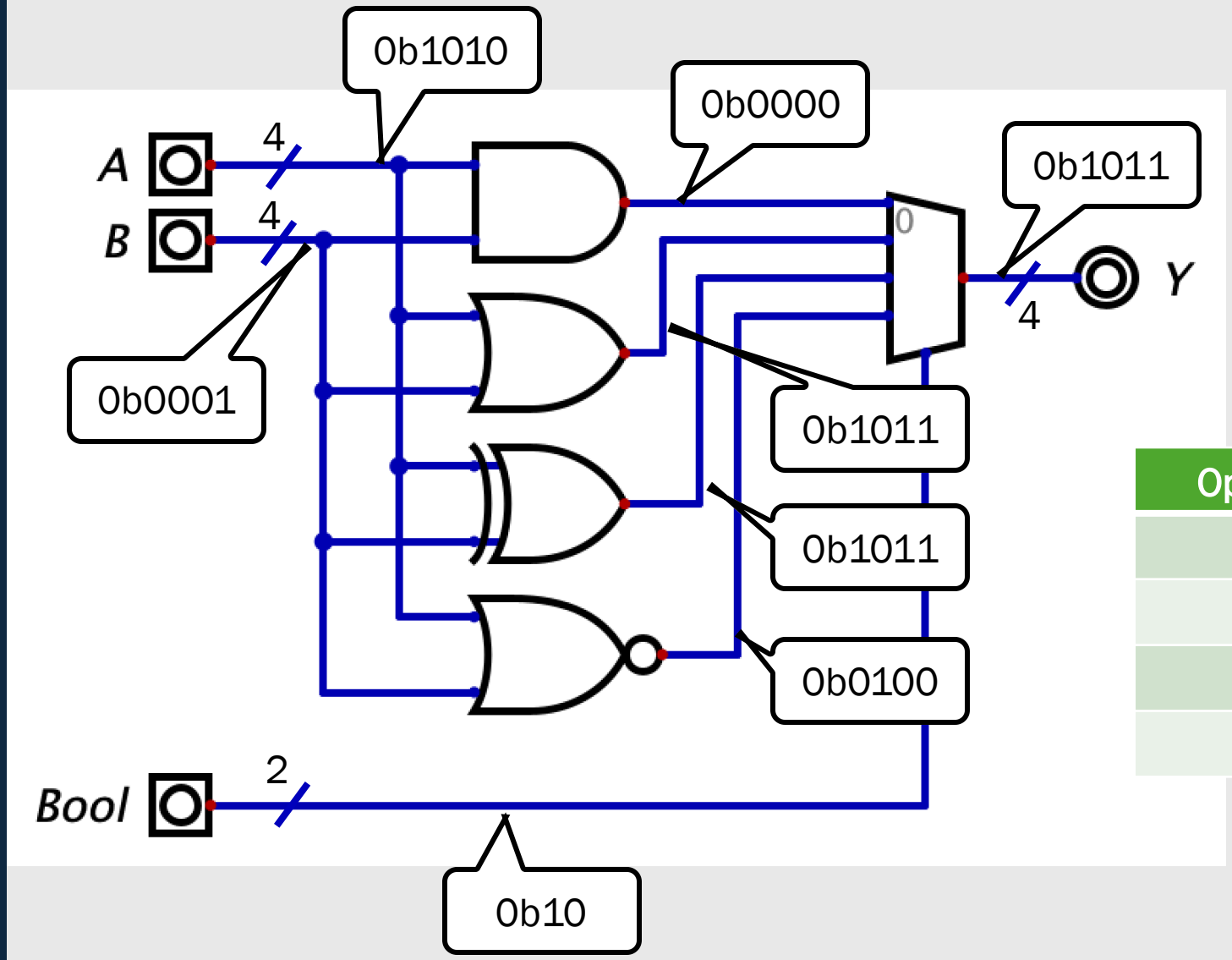

| Operation | Expression | Bool |
|---|---|---|
| AND | A x B | 0b00 |
| OR | A + B | 0b01 |
| XOR | $A \oplus B$ | 0b10 |
| NOR | $\overline{A + B}$ | 0b11 |

What are the values on each signal when we want to perform Y = A XOR B where A is 0b1010 and B is 0b0001?




| Operation | Expression | Bool |
|---|---|---|
| AND | A x B | 0b00 |
| OR | A + B | 0b01 |
| XOR | $A \oplus B$ | 0b10 |
| NOR | $\overline{A + B}$ | 0b11 |

# What are the values on each signal when we want to perform Y = A XOR B where A is 0b1010 and B is 0b0001?



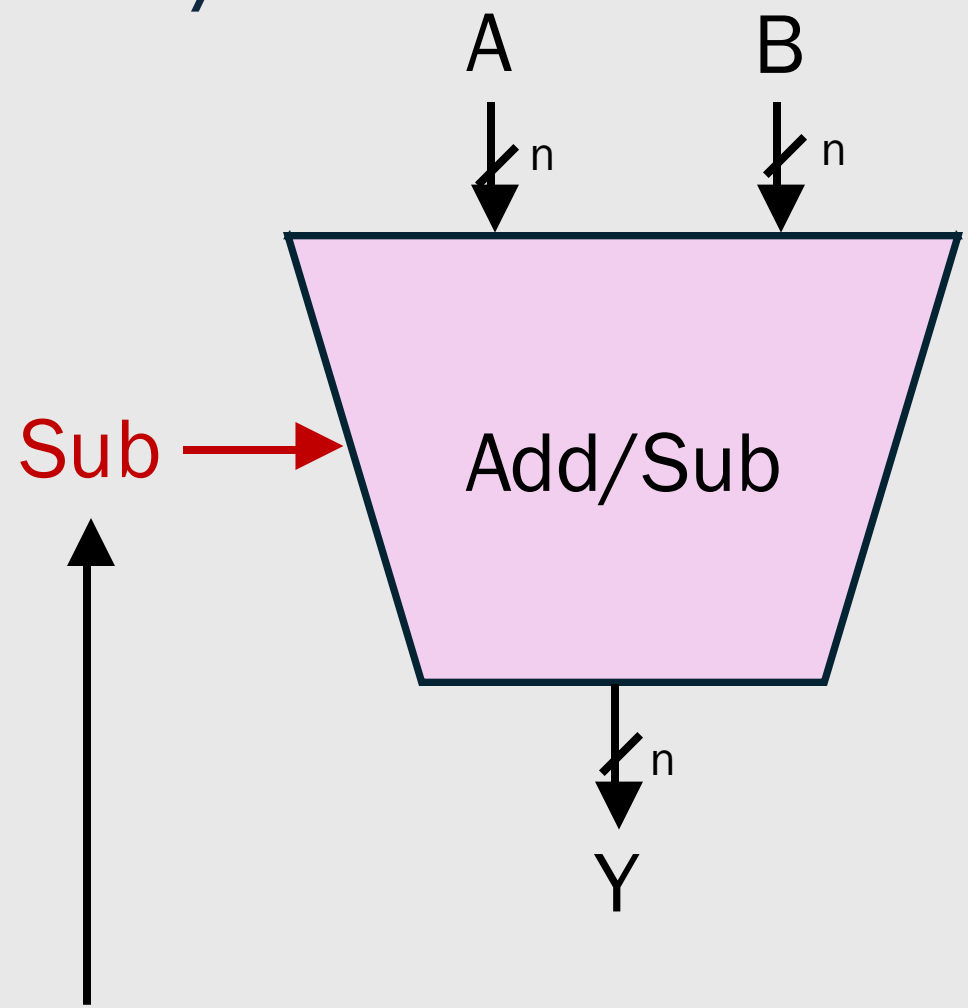

| Operation | Expression | Bool |
|---|---|---|
| AND | A x B | 0b00 |
| OR | A + B | 0b01 |
| XOR | A⊕B | 0b10 |
| NOR | $\overline{A + B}$ | 0b11 |

# Bidirectional Shifter

- The left bit tells you the direction of the shift (left or right)
- The right bit tells you the type of shift (arithmetic or logical)
- Logical and arithmetic left shift are the same

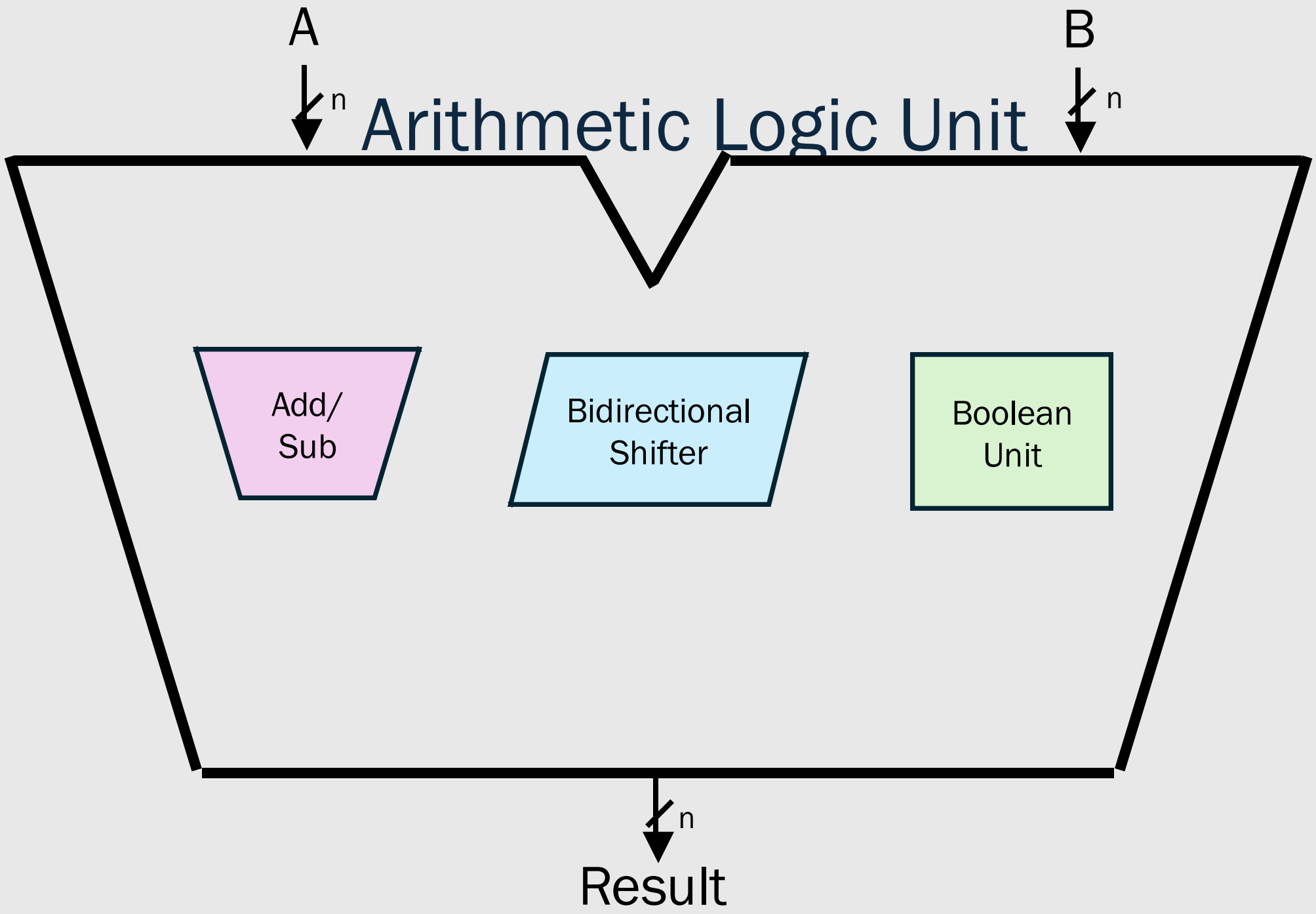
| Operation | Expression | Bool |
|---|---|---|
| Left Shift | B << A | 0 0 |
| None | N/A | 0 1 |
| Logical Right Shift | B >> A | 1 0 |
| Arithmetic Right Shift | B >>> A | 1 1 |

# Add/Sub

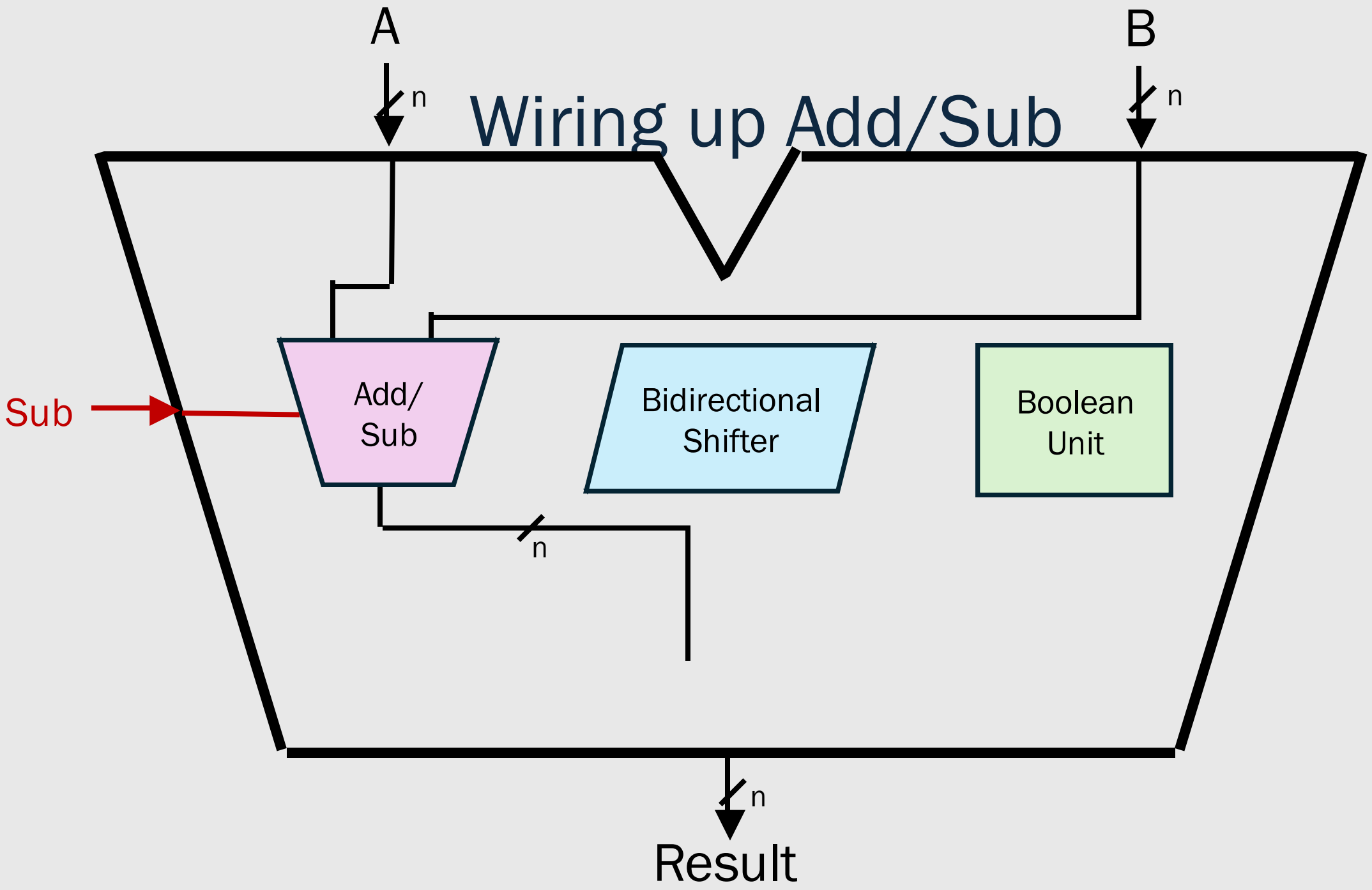



| Operation | Expression | Sub |
|---|---|---|
| Add | A + B | 0 |
| Sub | A - B | 1 |

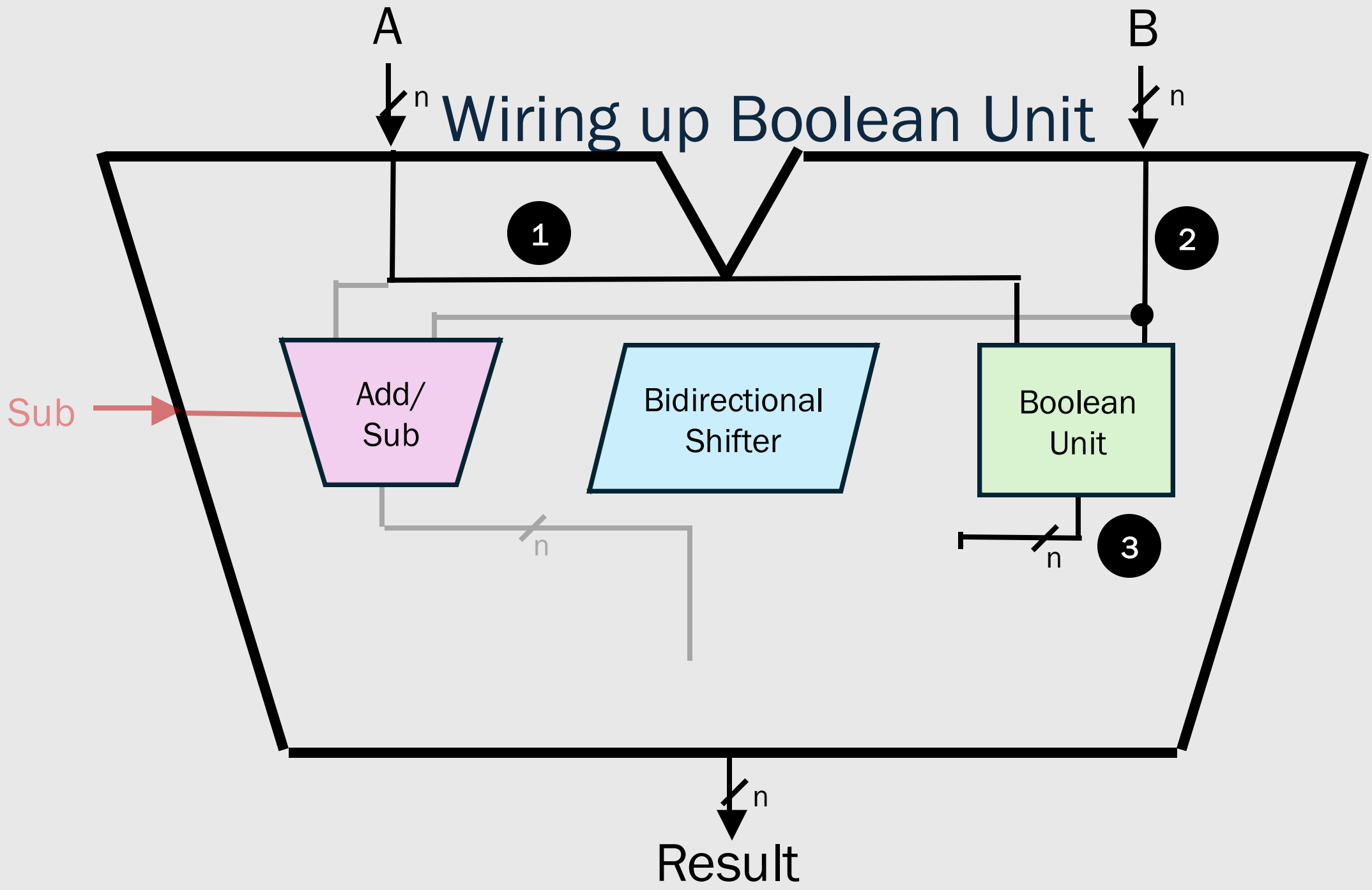
Control signal: Tells the circuit which operation to perform


A
n
Arithmetic Logic Unit
B
n
Add/
Sub
Bidirectional
Shifter
Boolean
Unit
n
Result

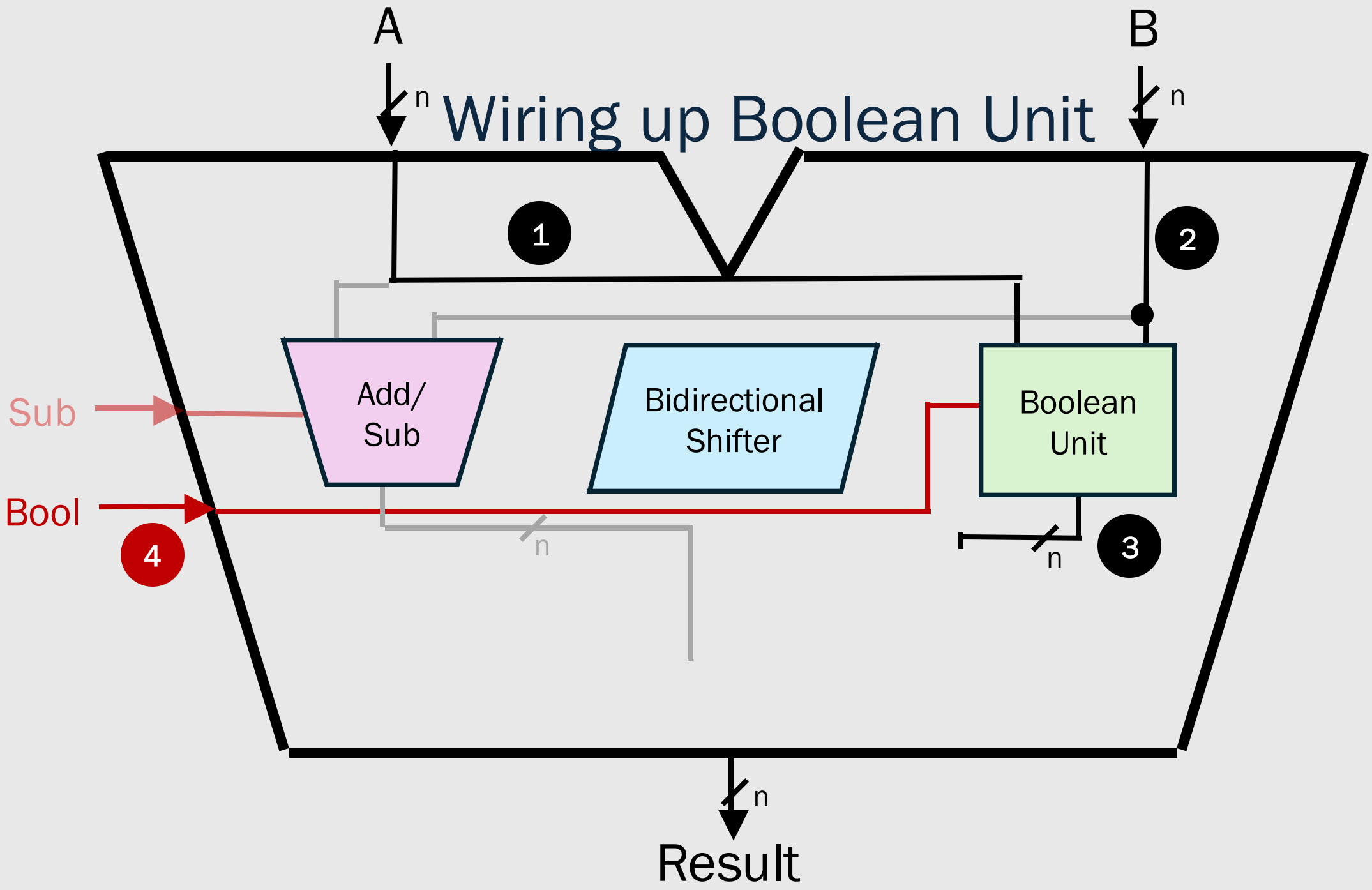
# Wiring up Add/Sub



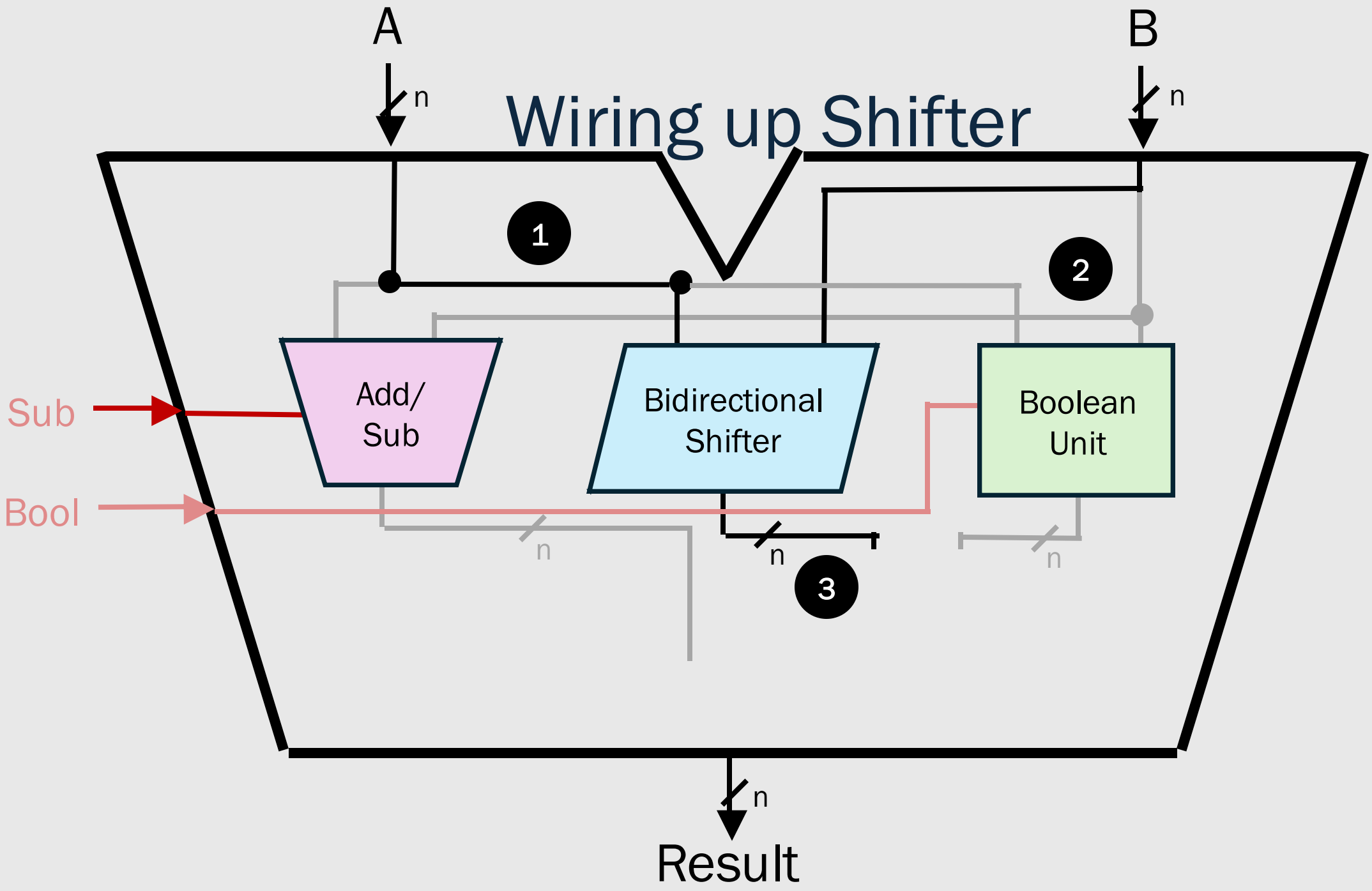
# Wiring up Boolean Unit



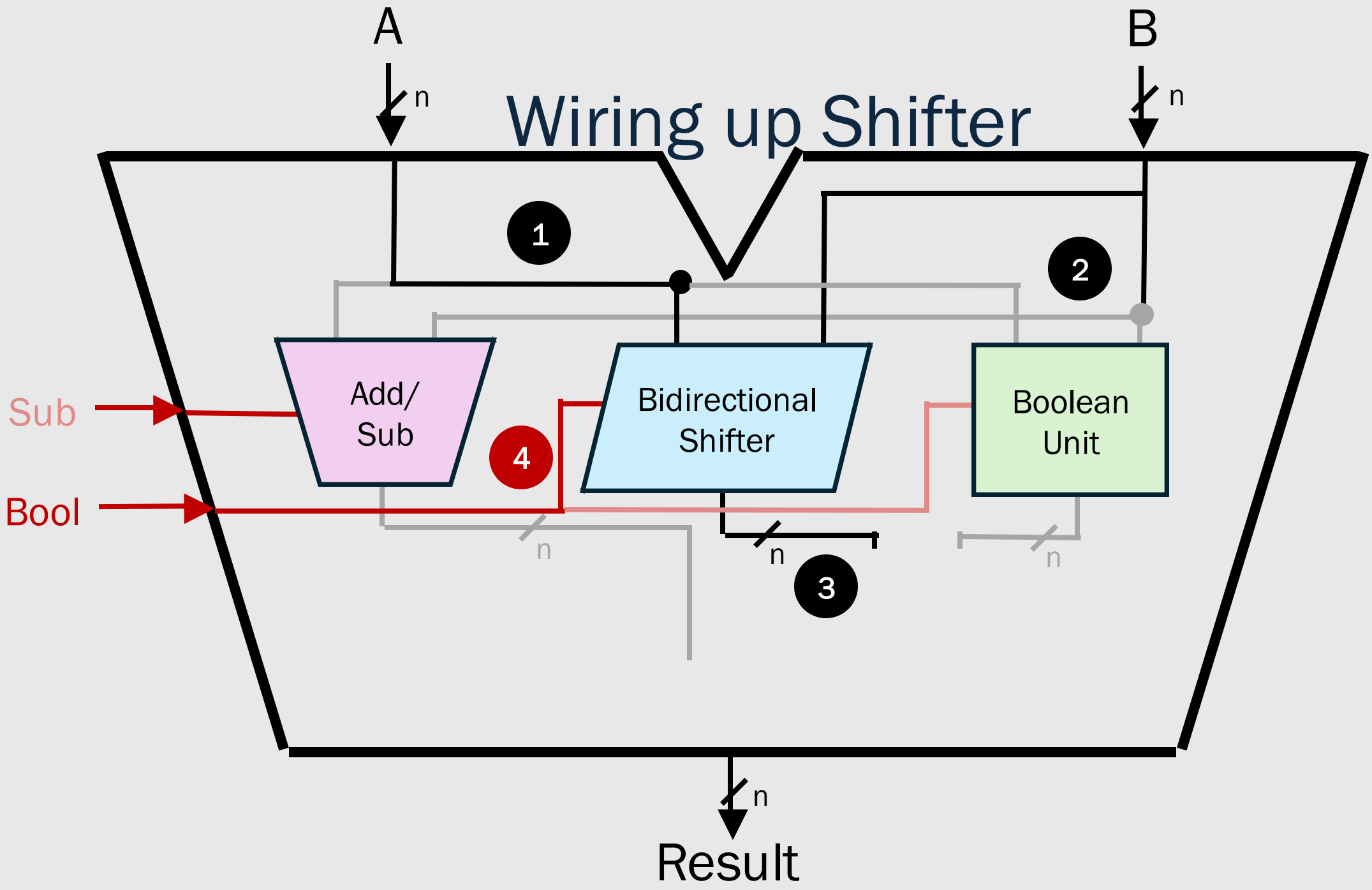
# Wiring up Boolean Unit



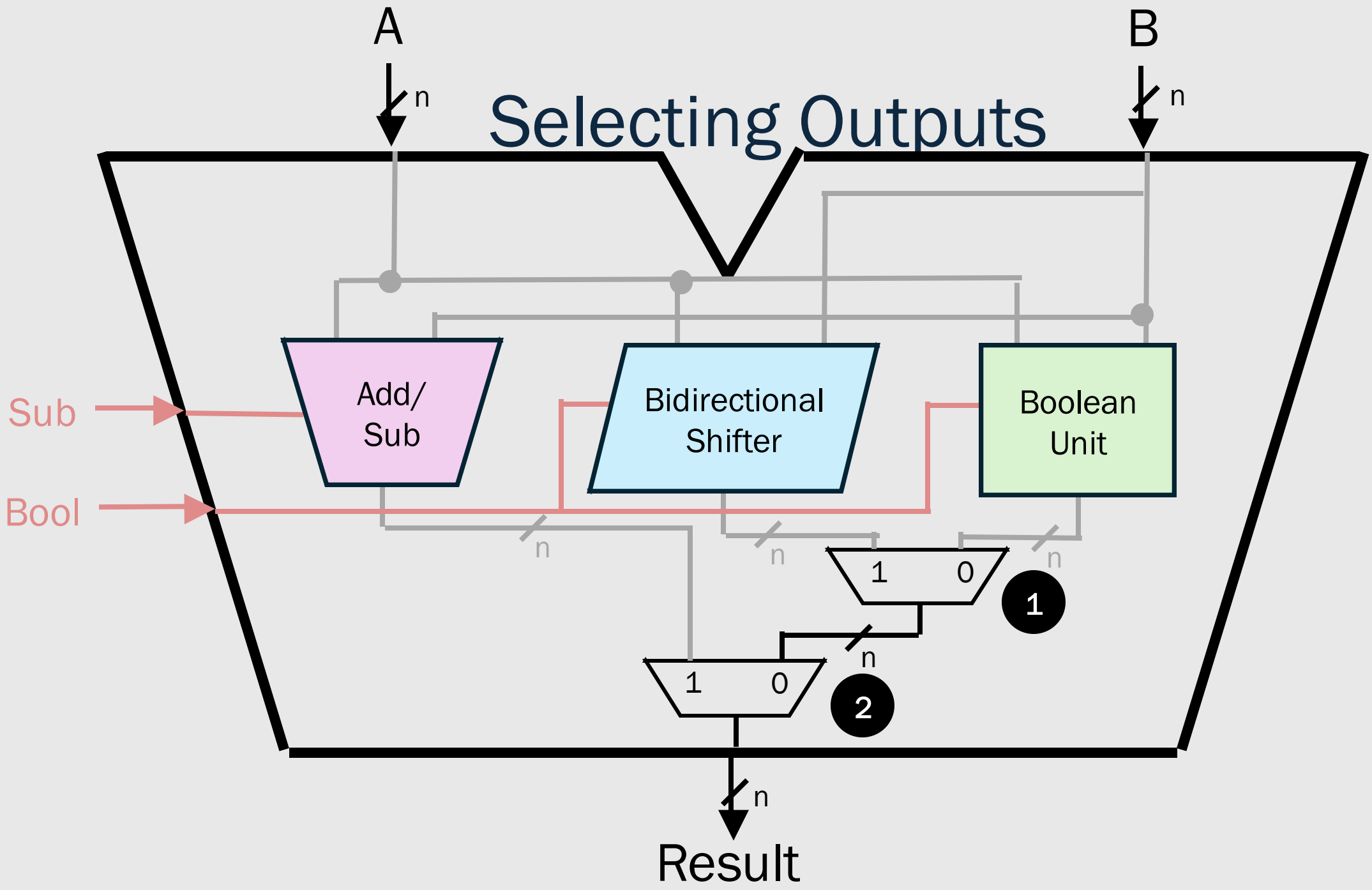
# Wiring up Shifter



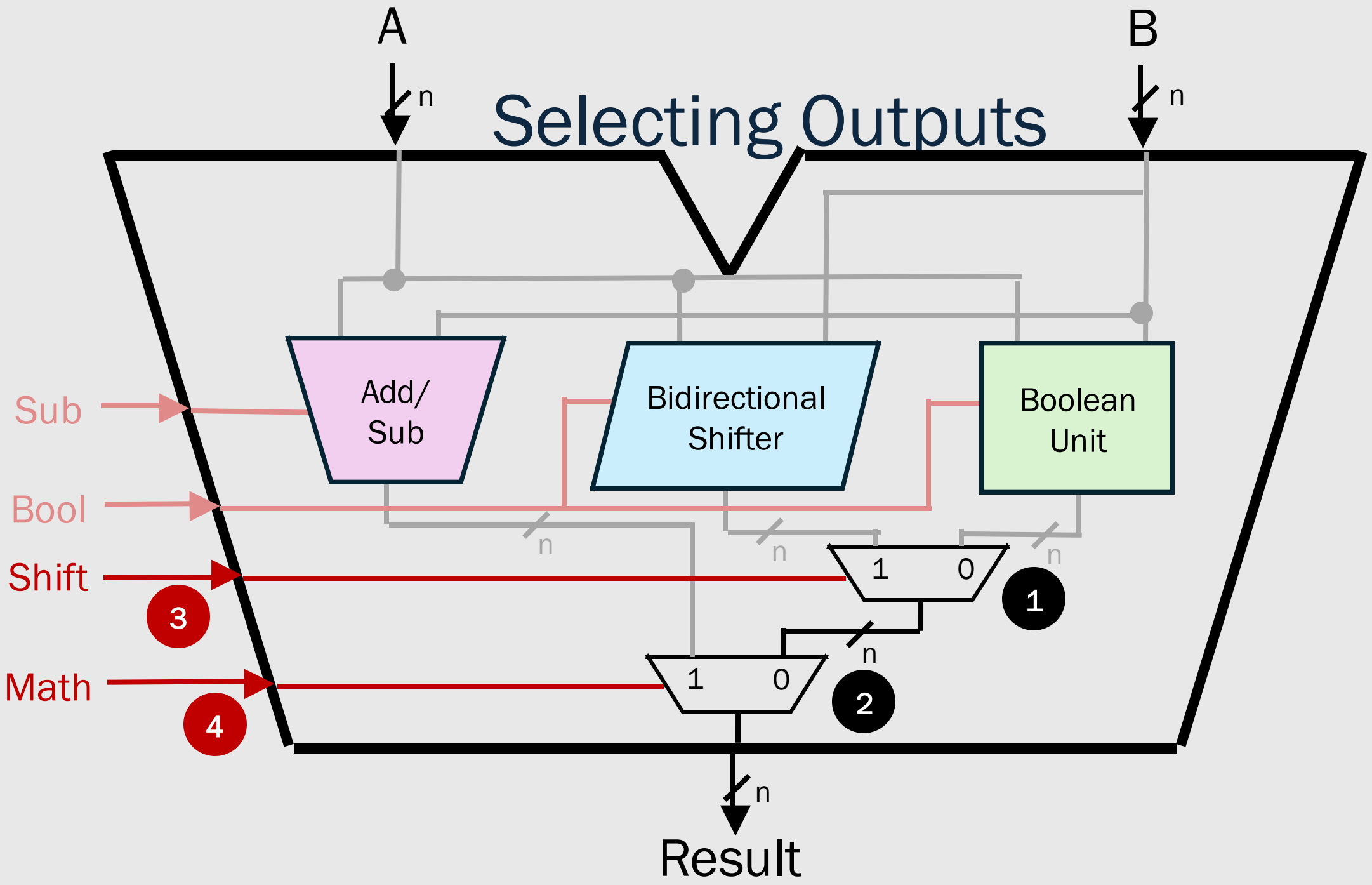
# Wiring up Shifter

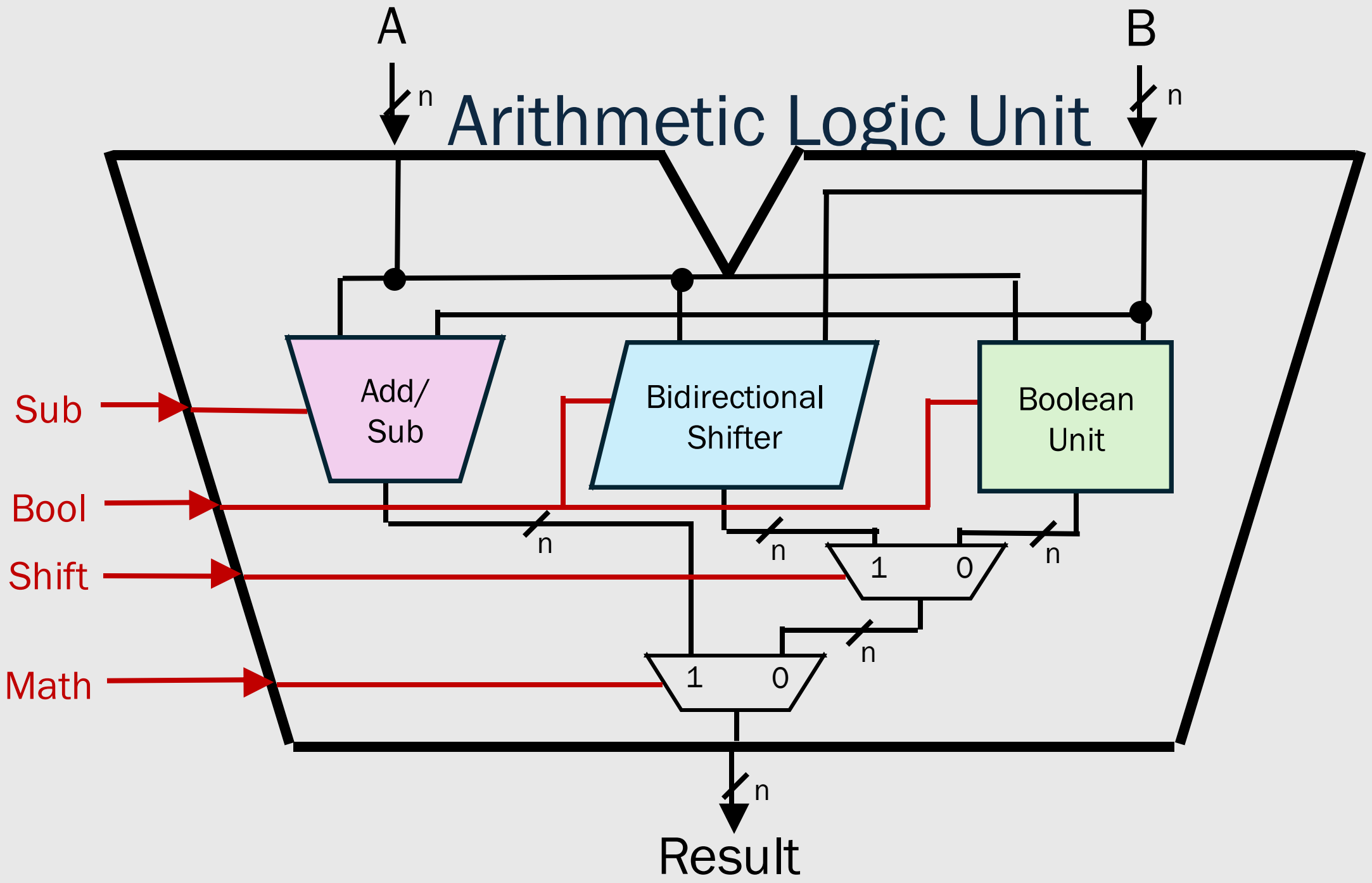


# Selecting Outputs



# Selecting Outputs




A
n
B
n
Arithmetic Logic Unit
Sub
Bool
Shift
Math
Add/
Sub
Bidirectional
Shifter
Boolean
Unit
n
n
n
1
0
n
1
0
n
Result

# ALU Control Bits

| Operation | Expression | Sub |
|---|---|---|
| Add | A + B | 0 |
| Sub | A - B | 1 |

| Operation | Expression | Bool |
|---|---|---|
| Left Shift | B << A | 00 |
| Logical Right Shift | B >> A | 10 |
| Arithmetic Right Shift | B >>> A | 11 |

| Operation | Expression | Bool |
|---|---|---|
| AND | A x B | 00 |
| OR | A + B | 01 |
| XOR | A⊕B | 10 |
| NOR | $\overline{A + B}$ | 11 |

| Operation | Sub | Bool | Shift | Math |
|---|---|---|---|---|
| A + B | | | | |
| A - B | | | | |
| B << A | | | | |
| B >> A | | | | |
| B >>> A | | | | |
| A AND B | | | | |
| A OR B | | | | |
| A XOR B | | | | |
| A NOR B | | | | |

# ALU Control Bits

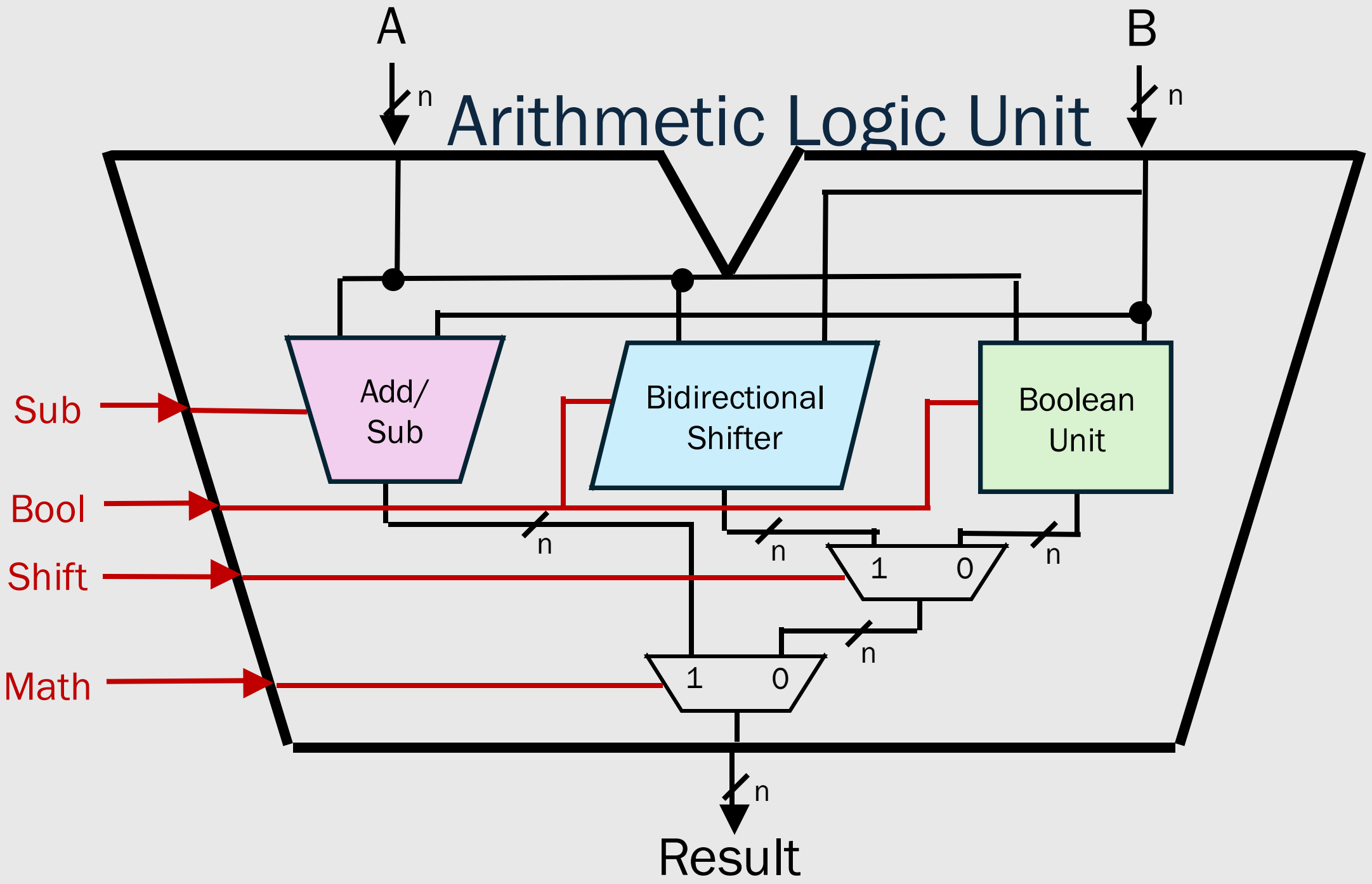
| Operation | Expression | Sub |
|---|---|---|
| Add | A + B | 0 |
| Sub | A - B | 1 |

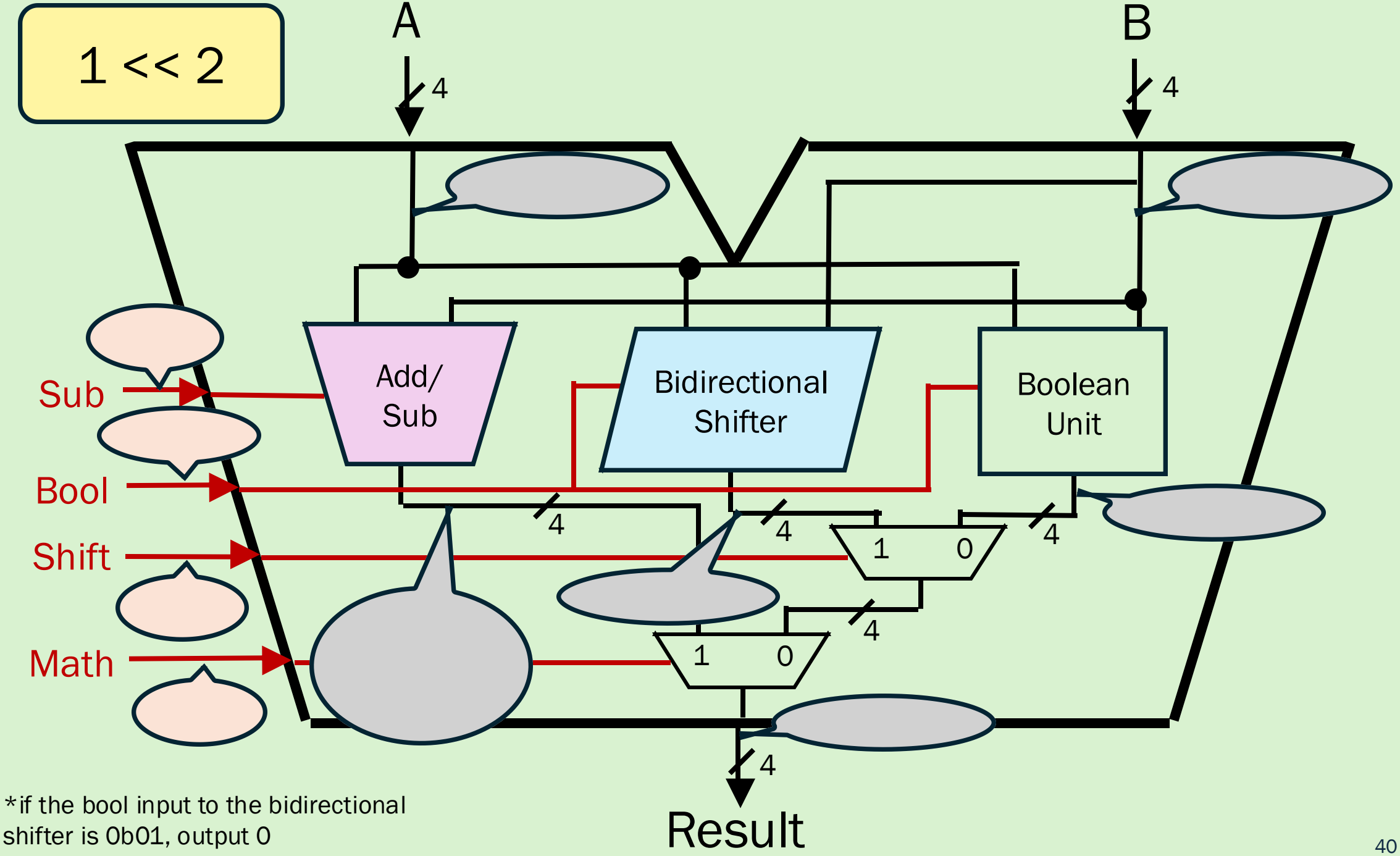
| Operation | Expression | Bool |
|---|---|---|
| Left Shift | B << A | 00 |
| Logical Right Shift | B >> A | 10 |
| Arithmetic Right Shift | B >>> A | 11 |

| Operation | Expression | Bool |
|---|---|---|
| AND | A x B | 00 |
| OR | A + B | 01 |
| XOR | A⊕B | 10 |
| NOR | $\overline{A + B}$ | 11 |

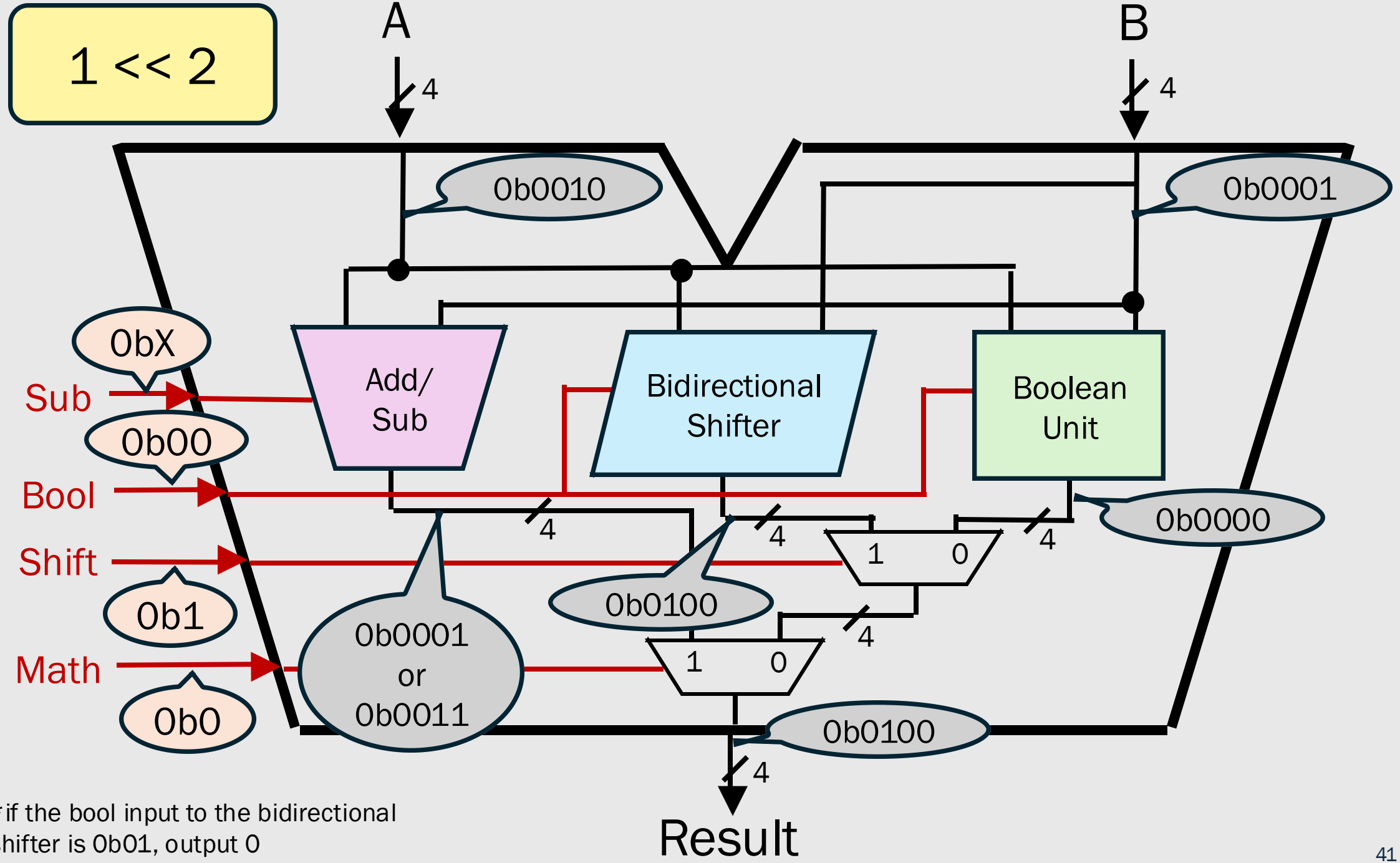
| Operation | Sub | Bool | Shift | Math |
|---|---|---|---|---|
| A + B | 0b0 | 0bXX | 0bX | 0b1 |
| A - B | 0b1 | 0bXX | 0bX | 0b1 |
| B << A | 0bX | 0b00 | 0b1 | 0b0 |
| B >> A | 0bX | 0b10 | 0b1 | 0b0 |
| B >>> A | 0bX | 0b11 | 0b1 | 0b0 |
| A AND B | 0bX | 0b00 | 0b0 | 0b0 |
| A OR B | 0bX | 0b01 | 0b0 | 0b0 |
| A XOR B | 0bX | 0b10 | 0b0 | 0b0 |
| A NOR B | 0bX | 0b11 | 0b0 | 0b0 |

# Arithmetic Logic Unit



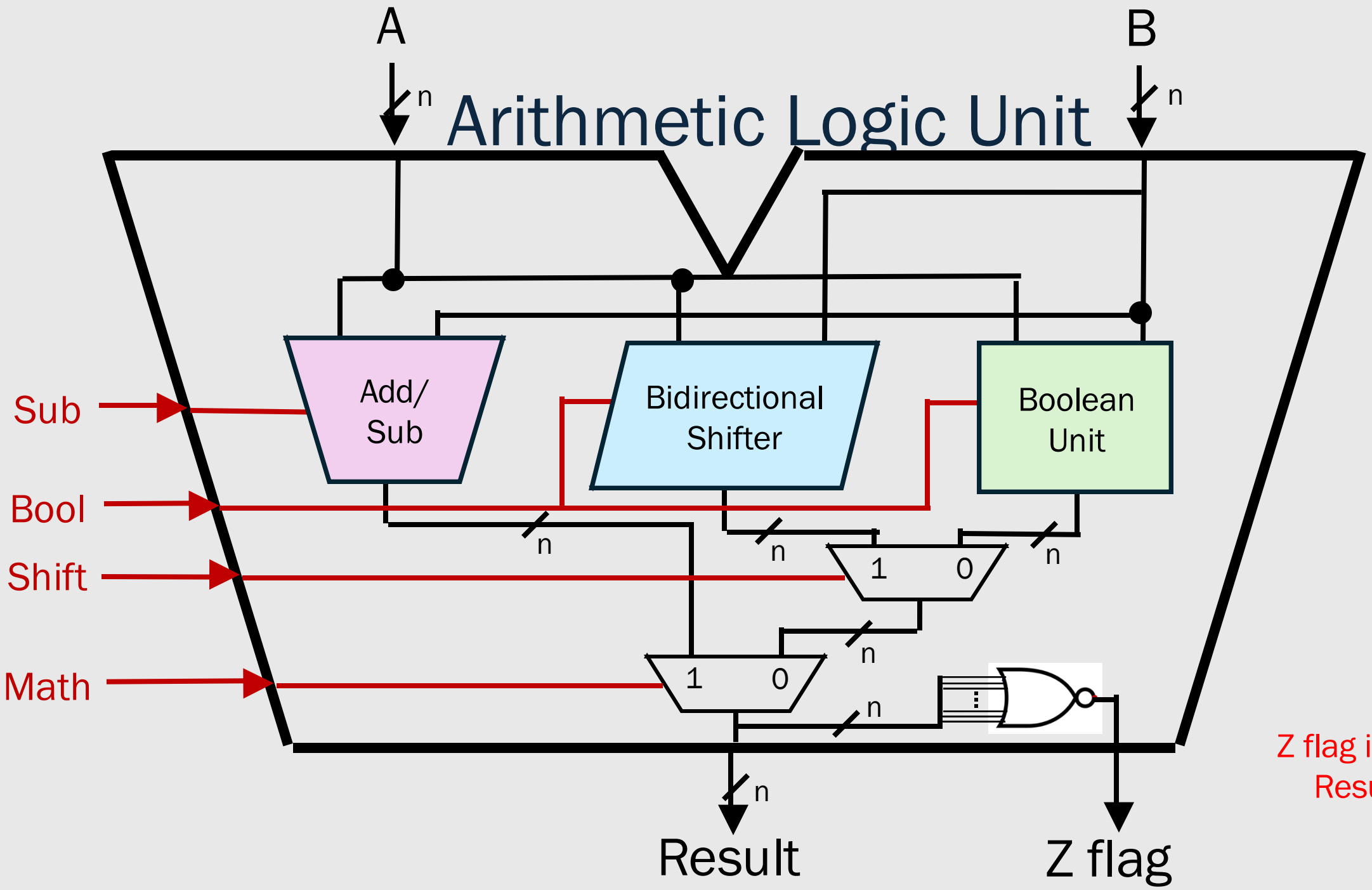



*if the bool input to the bidirectional shifter is 0b01, output 0




*if the bool input to the bidirectional shifter is 0b01, output 0

# Arithmetic Logic Unit




Z flag is 1 when Result is 0