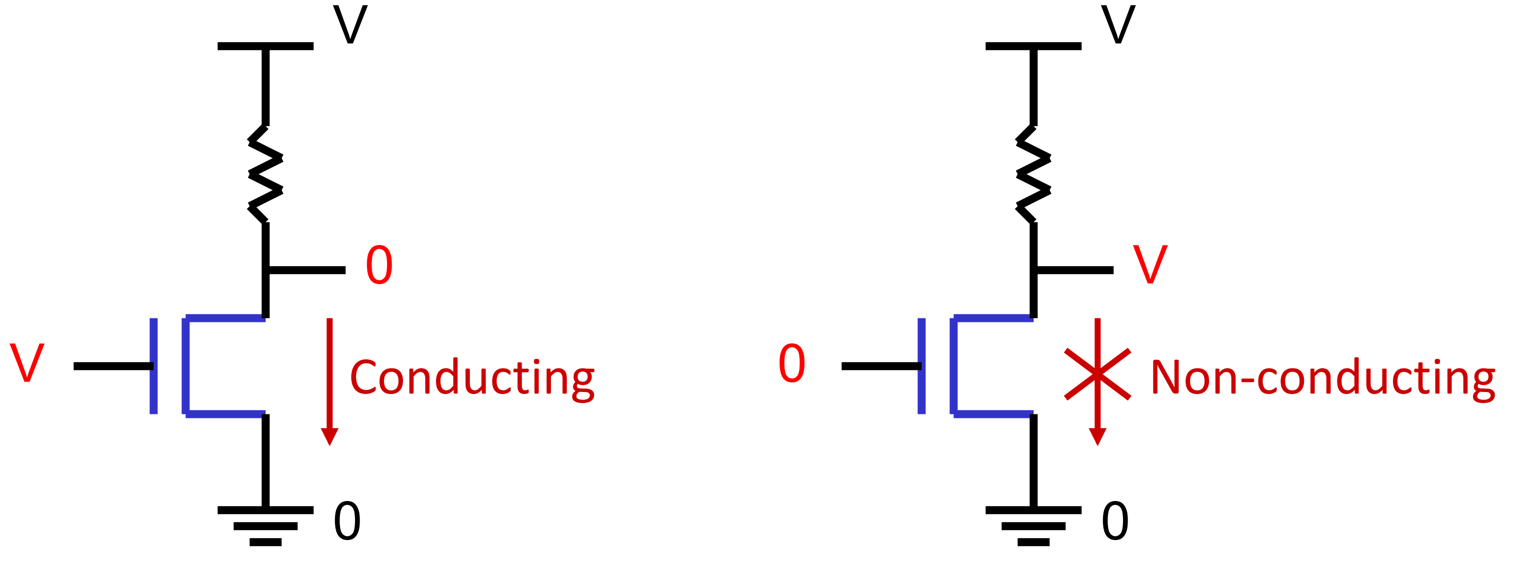
# Lecture 12: Hardware for Arithmetic

- Today's topics:
  - Digital logic intro
  - Logic for common operations
  - Designing an ALU

# Digital Design Basics

- Two voltage levels – high and low (1 and 0, true and false)
  Hence, the use of binary arithmetic/logic in all computers

- A transistor is a 3-terminal device that acts as a switch



# Logic Blocks

- A logic block has a number of binary inputs and produces a number of binary outputs – the simplest logic block is composed of a few transistors

- A logic block is termed *combinational* if the output is only a function of the inputs

- A logic block is termed *sequential* if the block has some internal memory (state) that also influences the output

- A basic logic block is termed a *gate* (AND, OR, NOT, etc.)

We will only deal with combinational circuits today

# Truth Table

- A truth table defines the outputs of a logic block for each set of inputs
- Consider a block with 3 inputs A, B, C and an output E that is true only if *exactly* 2 inputs are true

| A | B | C | E |
|---|---|---|---|

# Truth Table

- A truth table defines the outputs of a logic block for each set of inputs

- Consider a block with 3 inputs A, B, C and an output E that is true only if *exactly* 2 inputs are true

| A | B | C | E |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 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 |

Can be compressed by only representing cases that have an output of 1

# Boolean Algebra

- Equations involving two values and three primary operators:
  - OR : symbol + , X = A + B → X is true if at least one of A or B is true
  - AND : symbol **.** , X = A **.** B → X is true if both A and B are true
  - NOT : symbol $\overline{\ }$ , X = $\overline{A}$ → X is the inverted value of A

# Boolean Algebra Rules

- Identity law : $A + 0 = A$ ; $A . 1 = A$
- Zero and One laws : $A + 1 = 1$ ; $A . 0 = 0$
- Inverse laws : $A . \overline{A} = 0$ ; $A + \overline{A} = 1$
- Commutative laws : $A + B = B + A$ ; $A . B = B . A$
- Associative laws : $A + (B + C) = (A + B) + C$
  $A . (B . C) = (A . B) . C$
- Distributive laws : $A . (B + C) = (A . B) + (A . C)$
  $A + (B . C) = (A + B) . (A + C)$

# DeMorgan's Laws

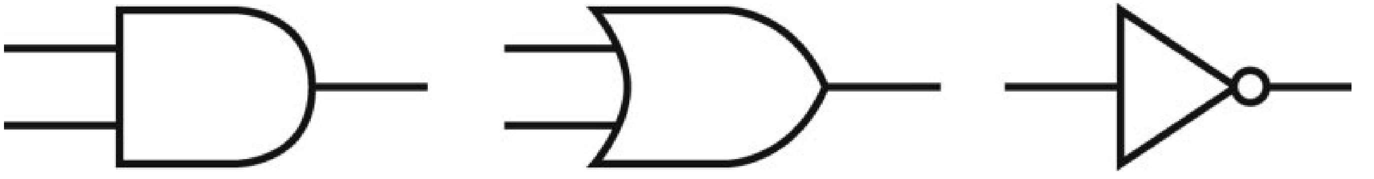
- $\overline{A + B} = \overline{A} \cdot \overline{B}$

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

- Confirm that these are indeed true

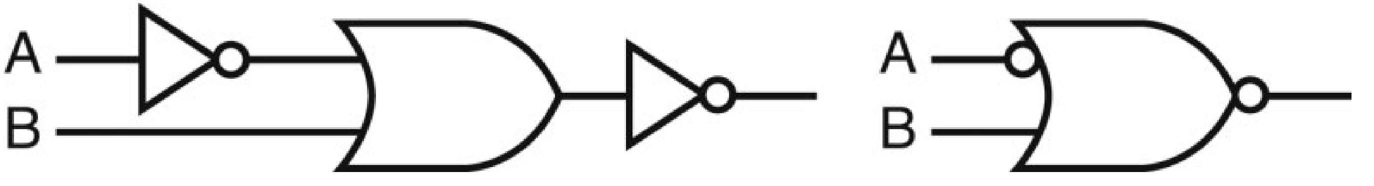
# Pictorial Representations

AND OR NOT



Source: H&P textbook

What logic function is this?




Source: H&P textbook

# Boolean Equation

- Consider the logic block that has an output E that is true only if exactly two of the three inputs A, B, C are true

Multiple correct equations:

Two must be true, but all three cannot be true:

$$E = ((A \,.\, B) + (B \,.\, C) + (A \,.\, C)) \,.\, \overline{(A \,.\, B \,.\, C)}$$

Identify the three cases where it is true:

$$E = (A \,.\, B \,.\, \overline{C}) + (A \,.\, C \,.\, \overline{B}) + (C \,.\, B \,.\, \overline{A})$$

# Sum of Products

- Can represent any logic block with the AND, OR, NOT operators
  - Draw the truth table
  - For each true output, represent the corresponding inputs as a product
  - The final equation is a sum of these products

| A | B | C | E |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 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 |

$(A \,.\, B \,.\, \overline{C}) + (A \,.\, C \,.\, \overline{B}) + (C \,.\, B \,.\, \overline{A})$

- Can also use “product of sums”
- Any equation can be implemented with an array of ANDs, followed by an array of ORs

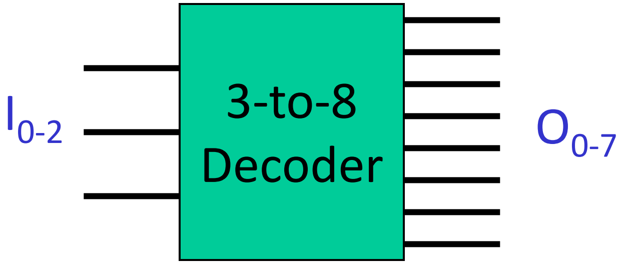
# NAND and NOR

- NAND : NOT of AND : A nand B = $\overline{A . B}$
- NOR : NOT of OR : A nor B = $\overline{A + B}$
- NAND and NOR are *universal gates*, i.e., they can be used to construct any complex logical function

# Common Logic Blocks – Decoder

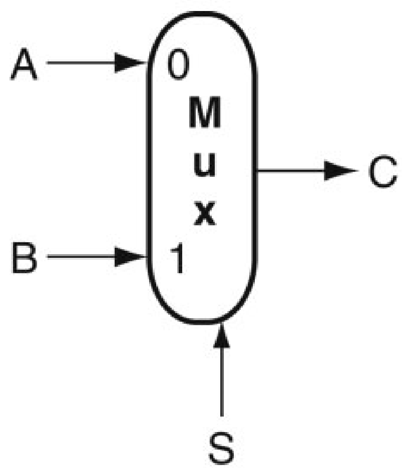
Takes in N inputs and activates one of $2^N$ outputs

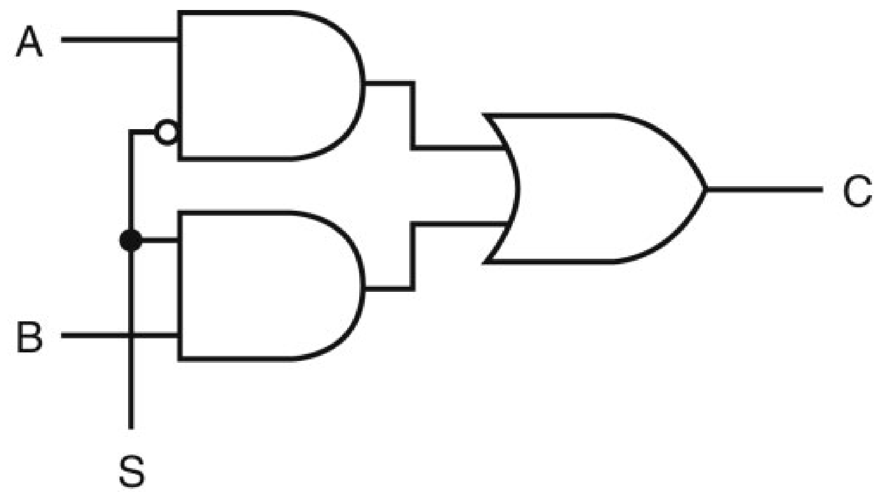
| $I_0$ | $I_1$ | $I_2$ | $O_0$ | $O_1$ | $O_2$ | $O_3$ | $O_4$ | $O_5$ | $O_6$ | $O_7$ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |



# Common Logic Blocks – Multiplexor

- Multiplexor or selector: one of N inputs is reflected on the output depending on the value of the $\log_2 N$ selector bits




Source: H&P textbook

2-input mux

# Adder Algorithm

| | | | | |
|---|---|---|---|---|
| | 1 | 0 | 0 | 1 |
| | 0 | 1 | 0 | 1 |
| Sum | 1 | 1 | 1 | 0 |
| Carry | 0 | 0 | 0 | 1 |

Truth Table for the above operations:

| A | B | Cin | Sum | Cout |
|---|---|---|---|---|
| 0 | 0 | 0 | | |
| 0 | 0 | 1 | | |
| 0 | 1 | 0 | | |
| 0 | 1 | 1 | | |
| 1 | 0 | 0 | | |
| 1 | 0 | 1 | | |
| 1 | 1 | 0 | | |
| 1 | 1 | 1 | | |

# Adder Algorithm

| | | | | |
|---|---|---|---|---|
| | 1 | 0 | 0 | 1 |
| | 0 | 1 | 0 | 1 |
| Sum | 1 | 1 | 1 | 0 |
| Carry | 0 | 0 | 0 | 1 |

Truth Table for the above operations:

| A | B | Cin | Sum | Cout |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

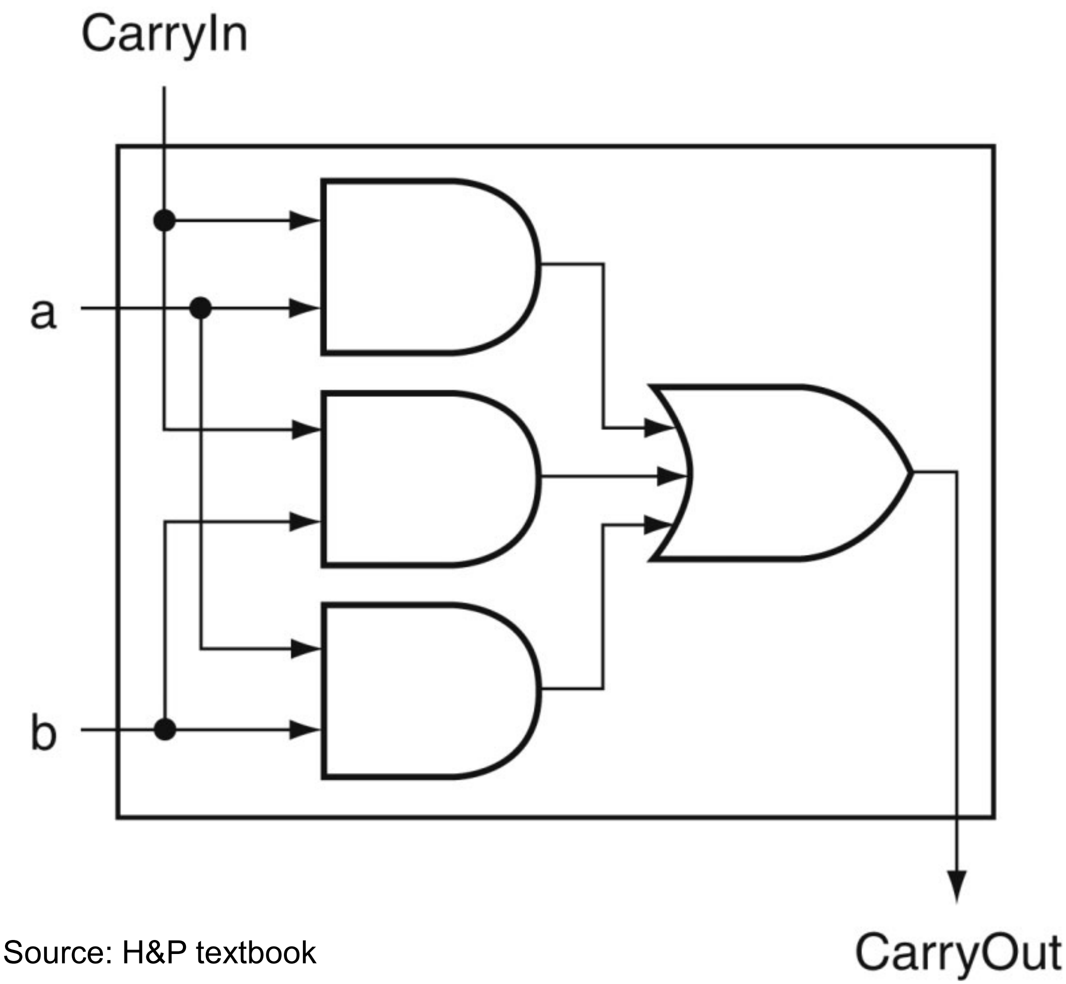
Equations:

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

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

$$= A \cdot B + A \cdot Cin + B \cdot Cin$$

# Carry Out Logic




Source: H&P textbook

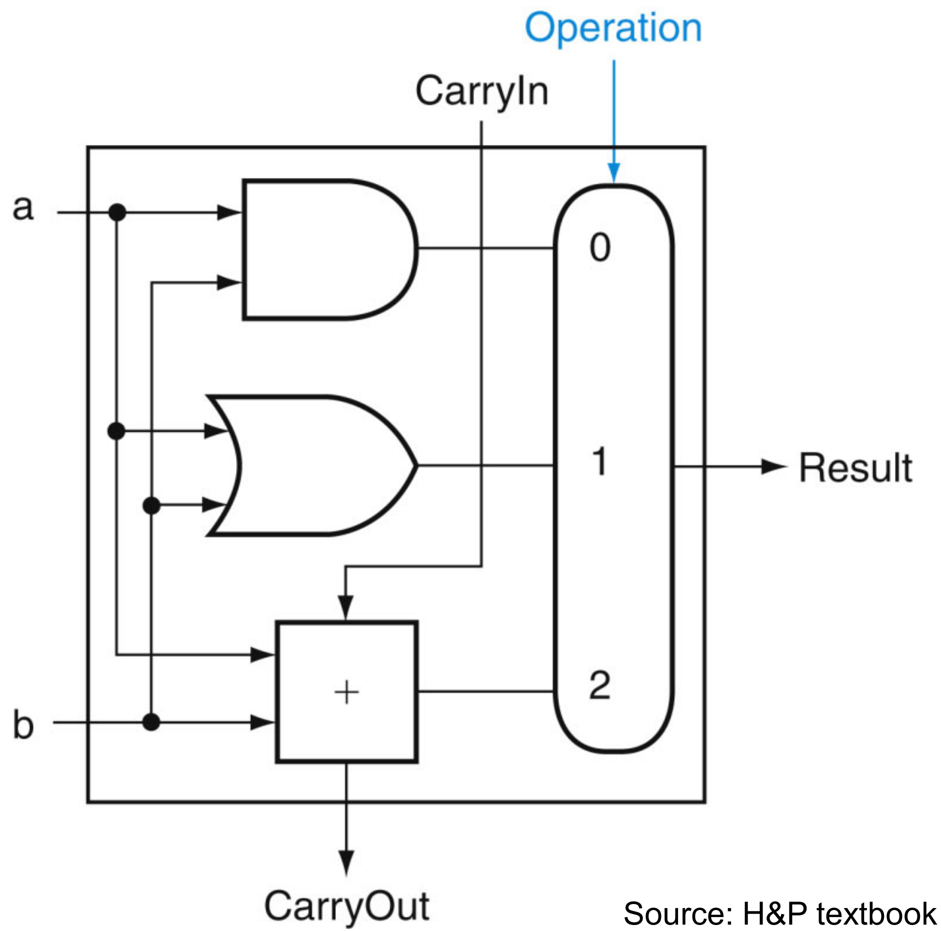
Equations:

$$
\begin{aligned}
Sum = {} & Cin \,.\, \overline{A} \,.\, \overline{B} \;+ \\
& B \,.\, \overline{Cin} \,.\, \overline{A} \;+ \\
& A \,.\, \overline{Cin} \,.\, \overline{B} \;+ \\
& A \,.\, B \,.\, Cin
\end{aligned}
$$

$$
\begin{aligned}
Cout = {} & A \,.\, B \,.\, Cin \;+ \\
& A \,.\, B \,.\, \overline{Cin} \;+ \\
& A \,.\, Cin \,.\, \overline{B} \;+ \\
& B \,.\, Cin \,.\, \overline{A} \\
= {} & A \,.\, B \;+ \\
& A \,.\, Cin \;+ \\
& B \,.\, Cin
\end{aligned}
$$

# 1-Bit ALU with Add, Or, And

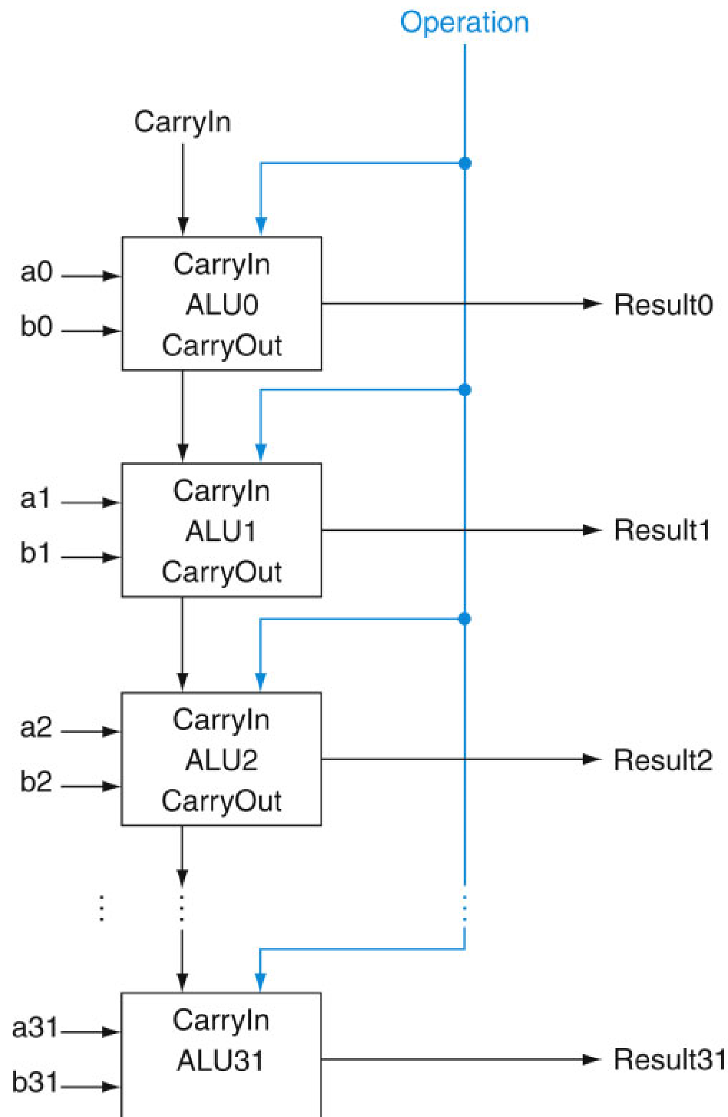
- Multiplexor selects between Add, Or, And operations




Source: H&P textbook

# 32-bit Ripple Carry Adder

1-bit ALUs are connected "in series" with the carry-out of 1 box going into the carry-in of the next box

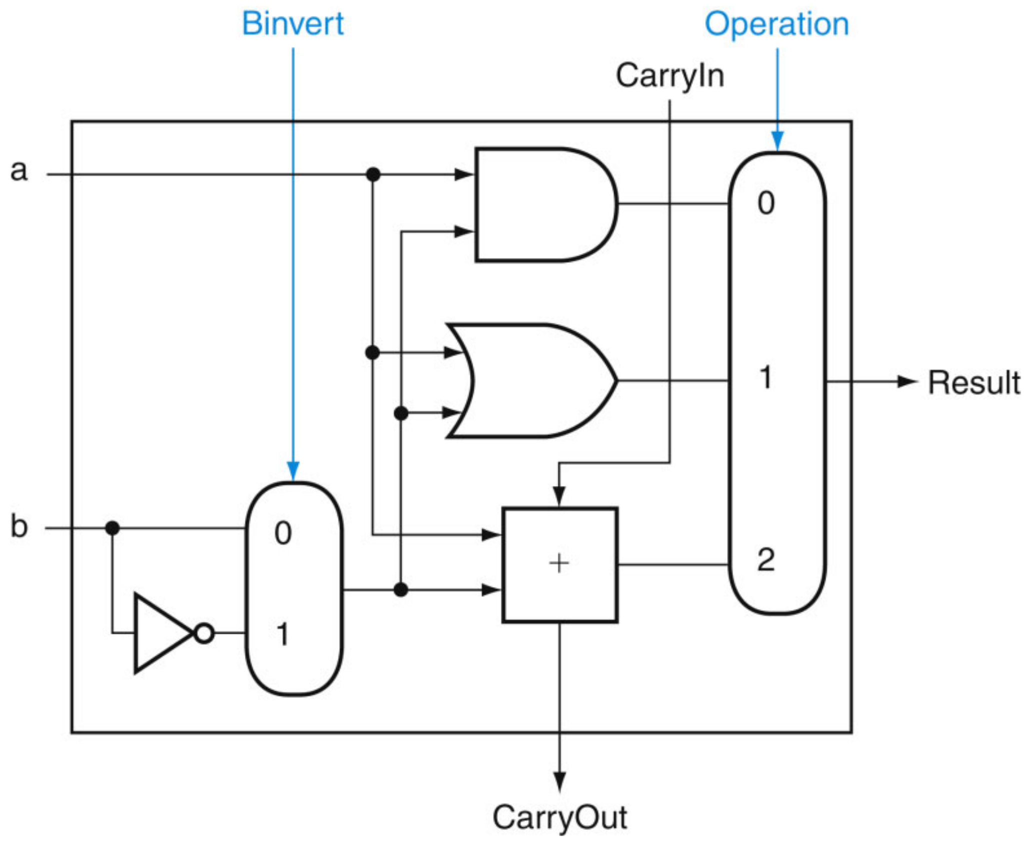



Source: H&P textbook

# Incorporating Subtraction

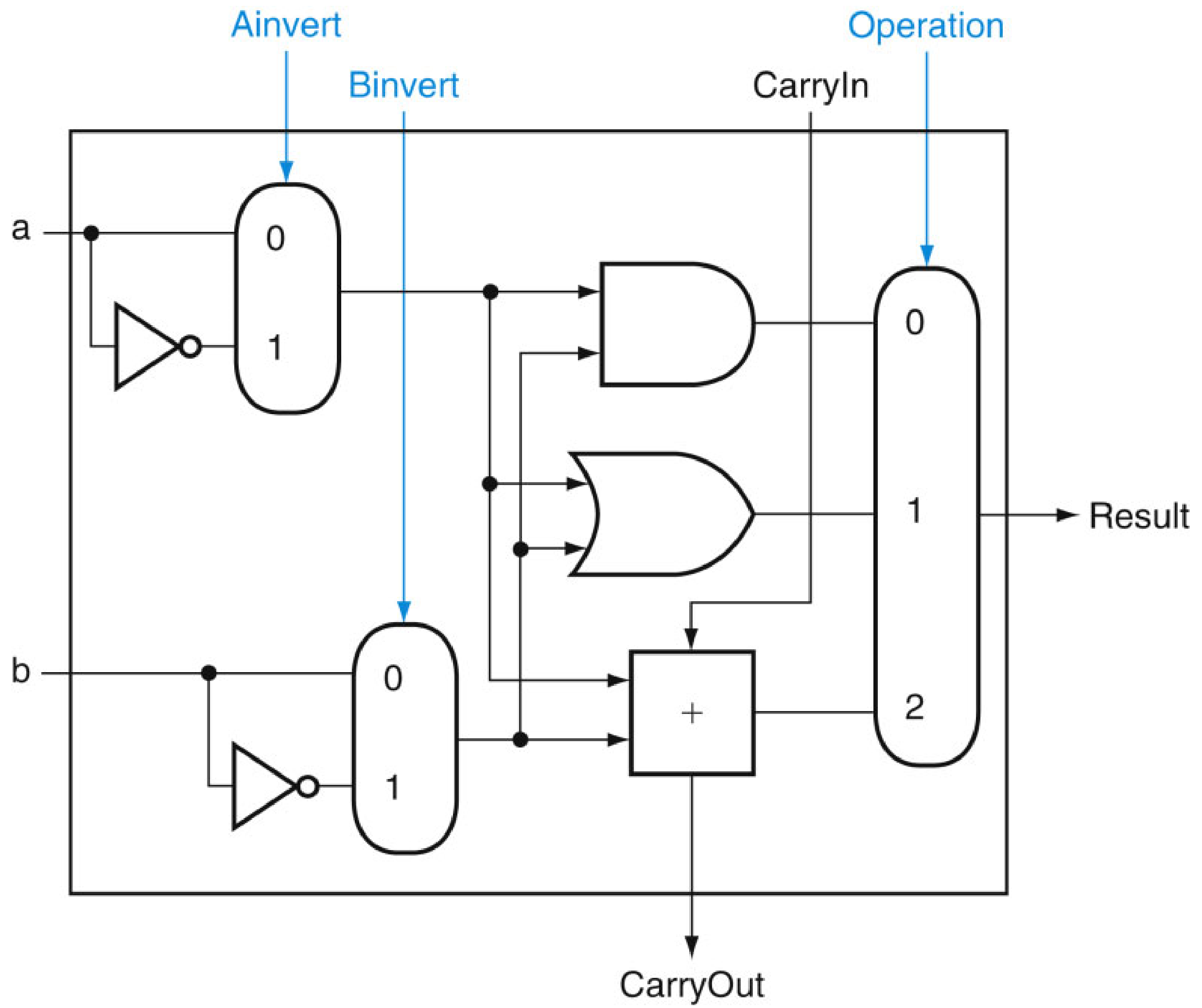
Must invert bits of B and add a 1

- Include an inverter
- CarryIn for the first bit is 1
- The CarryIn signal (for the first bit) can be the same as the Binvert signal




Source: H&P textbook

# Incorporating NOR and NAND

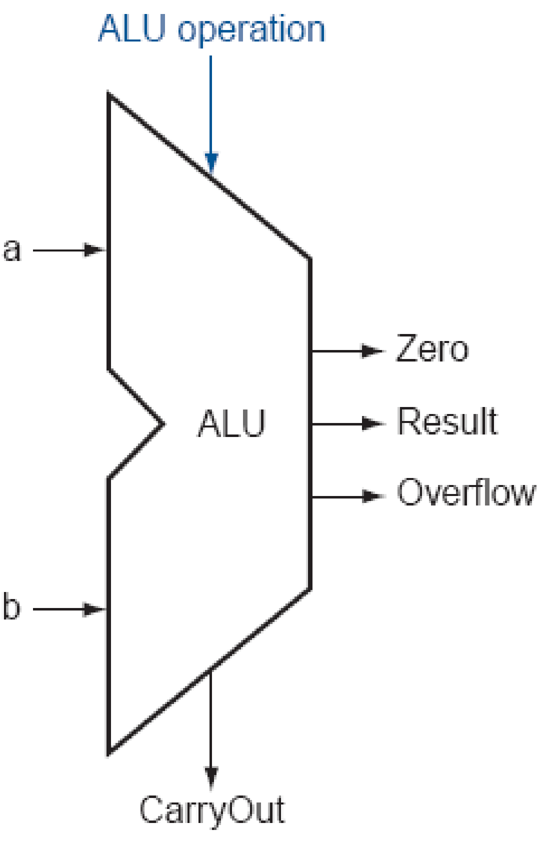



Source: H&P textbook

# Control Lines

What are the values
of the control lines
and what operations
do they correspond to?

| | Ai | Bn | Op |
|---|---|---|---|
| AND | 0 | 0 | 00 |
| OR | 0 | 0 | 01 |
| Add | 0 | 0 | 10 |
| Sub | 0 | 1 | 10 |
| NAND | 1 | 1 | 01 |
| NOR | 1 | 1 | 00 |




Source: H&P textbook