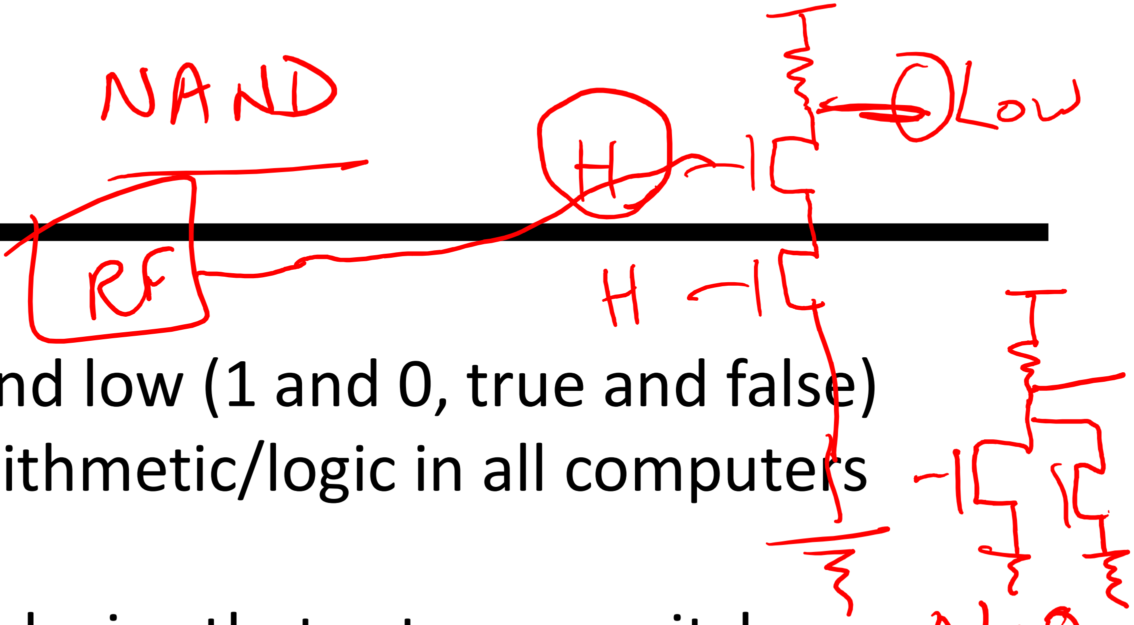
Lecture 12: Hardware for Arithmetic

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

HW 4 due Friday

HW5 posted
later today.
Due in a week

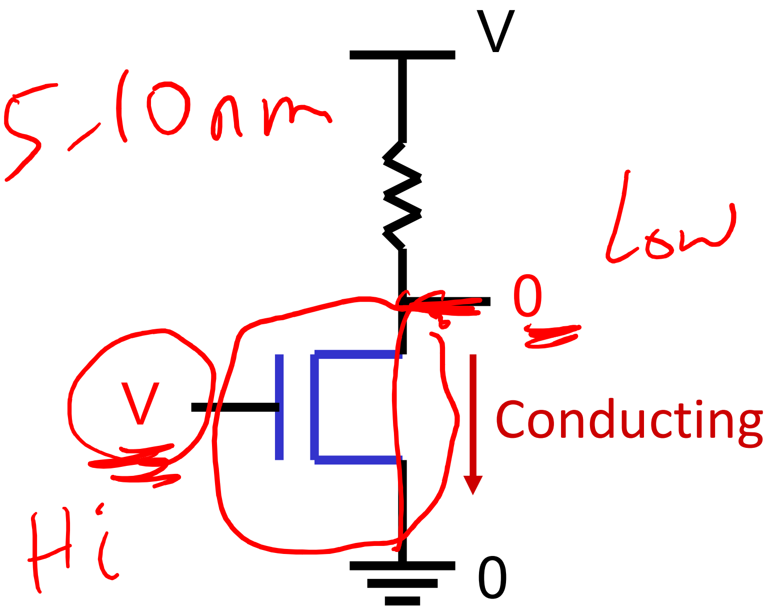
Digital Design Basics

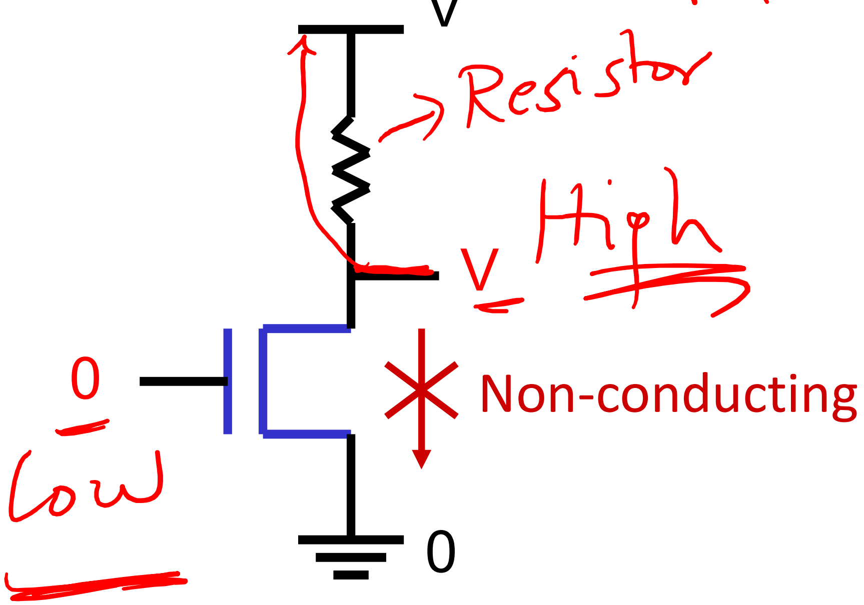


- Two voltage levels – high and low (1 and 0, true and false) Hence, the use of binary arithmetic/logic in all computers

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

inverter





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 → ↳ memory-less ccts
- 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.)

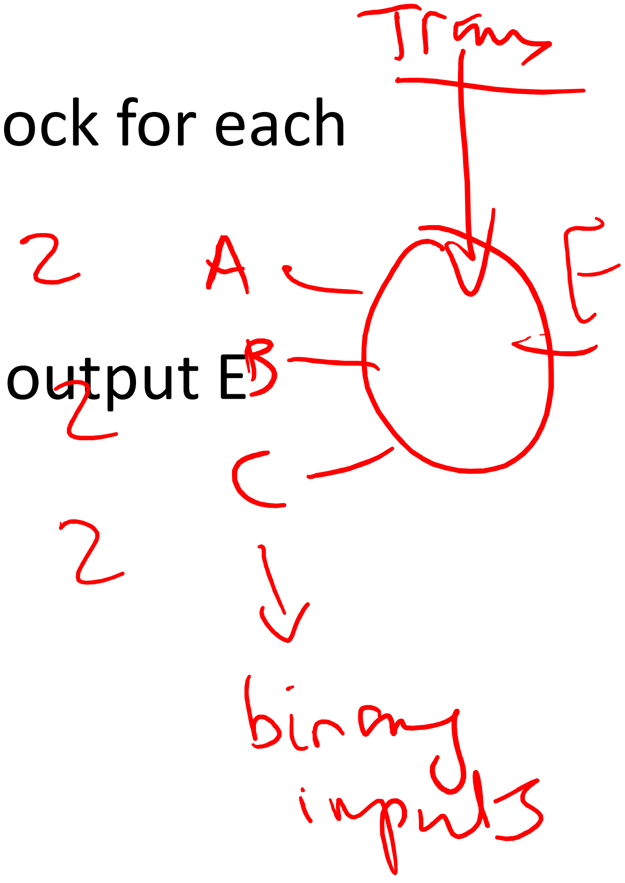
We will only deal with combinational circuits today

Truth Table

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

English

- 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

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

$+ \qquad \bullet \qquad \overline{}$

$\vee \qquad \wedge \qquad \neg A$

- 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 : $A . (B + C) = (A . B) + (A . C)$

  $A + (B . C) = (A + B) . (A + C)$

$7 \times (4+3)$

$= (7 \times 4) + (7 \times 3$

DeMorgan's Laws

- $\underline{\overline{A + B}} = \overline{A} . \overline{B}$

Inverse
Negation of OR = AND of inverses

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

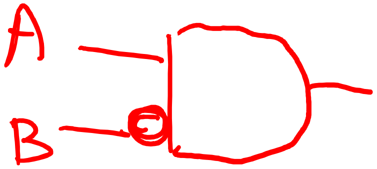
↑

- Confirm that these are indeed true

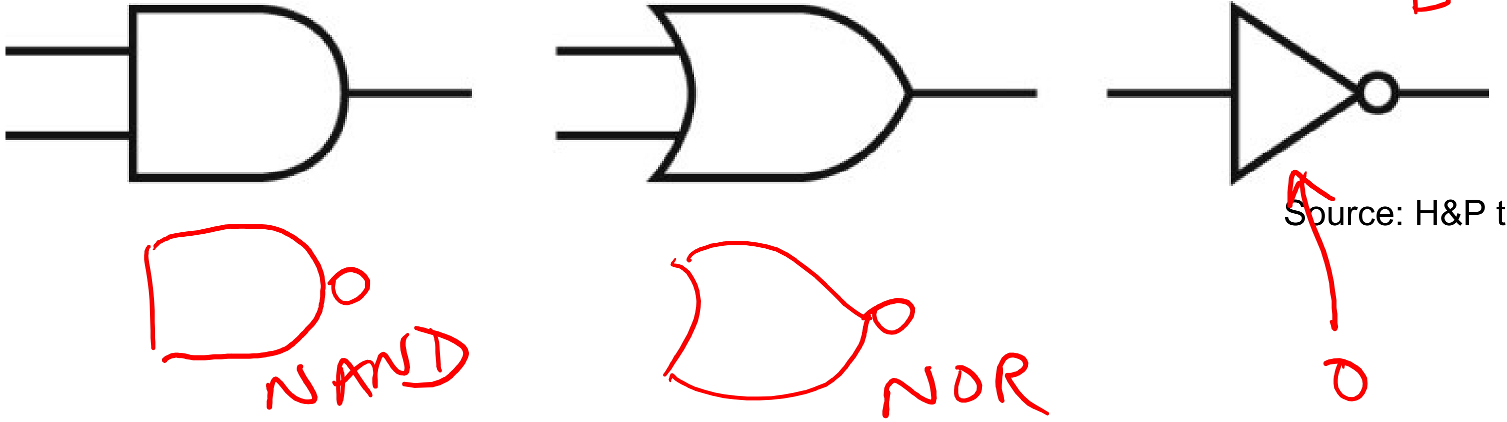
Pictorial Representations

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

$= A \cdot \overline{B}$

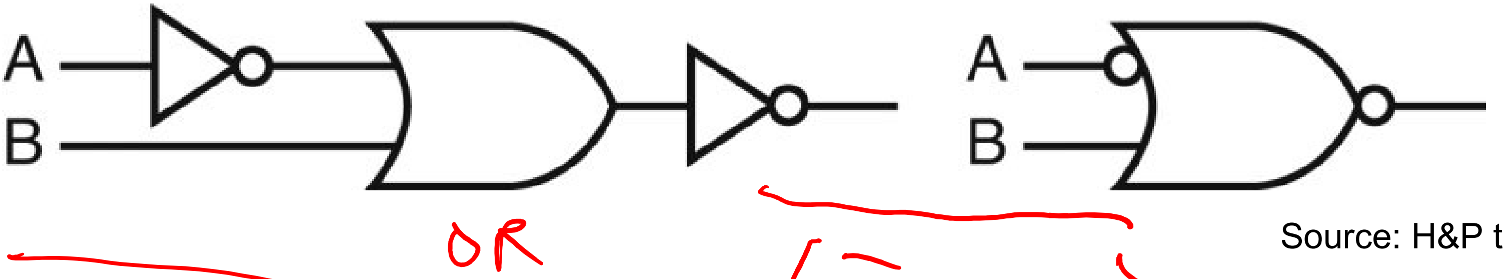


AND OR NOT



Source: H&P textbook

What logic function is this?



$\overline{(\overline{A}+B)}$

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:

0 if A=B=C=1

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

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

Identify the three cases where it is true:

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

Sum of Products

product of sums

where the cases output = 0

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

- 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

describes the cases where output = 1

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

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

1 if

A = 0
B = 1
C = 1

- 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 = $\overline{A + B}$

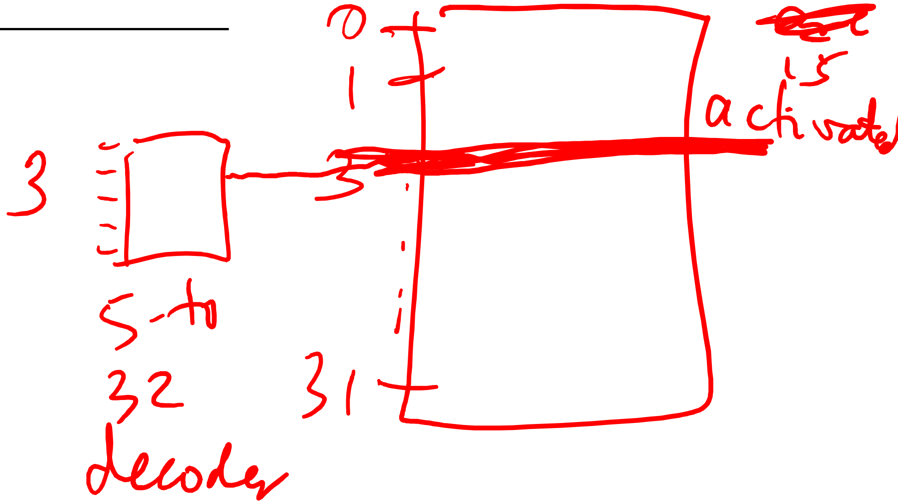
- NAND and NOR are *universal gates*, i.e., they can be used to construct any complex logical function

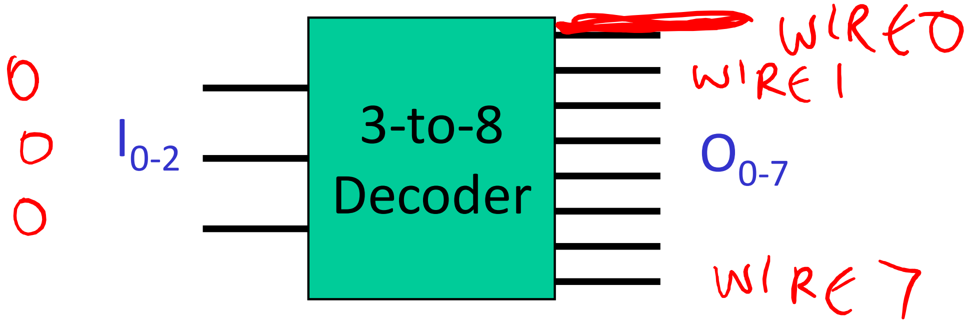
Common Logic Blocks – Decoder

storage ccts need a decoder
input = addr
out = 1 of many addr

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





$$O_0 = \bar{I}_0 \cdot \bar{I}_1 \cdot \bar{I}_2$$

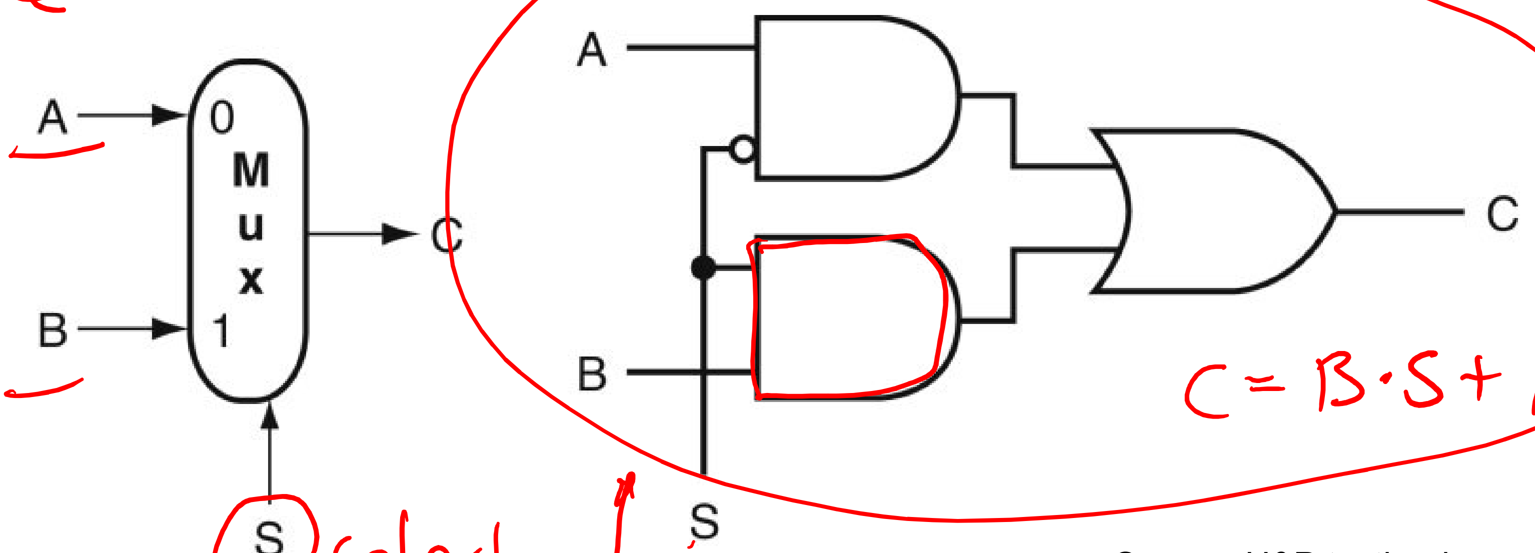
(sum of prod)

A	B	S	C

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

Design the cct for a 4 to 1 MUX



selector

$C = B \cdot S + A \cdot \bar{S}$

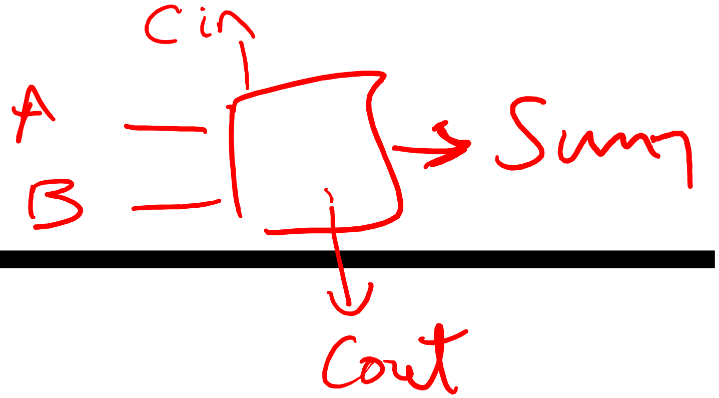
Source: H&P textbook

2-input mux

if (s==00) then E=A
if(S==01) then E=B
if(S==10) E=C
if(S==11) E=D

if S==0, then C=A
if S==1, then C=B

Adder Algorithm



		0	1	
	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	Cout	Sum
0	0	0	0	0		
0	0	1	1	0		
0	1	0				
0	1	1	0	1	1	0
1	0	0				
1	0	1				
1	1	0				
1	1	1	1	1		

Adder Algorithm

$$A\cdot B\cdot Cin + A\cdot B\cdot \overline{Cin}$$
$$= A\cdot B\cdot (Cin + \overline{Cin})$$
$$= A\cdot B$$

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

	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:

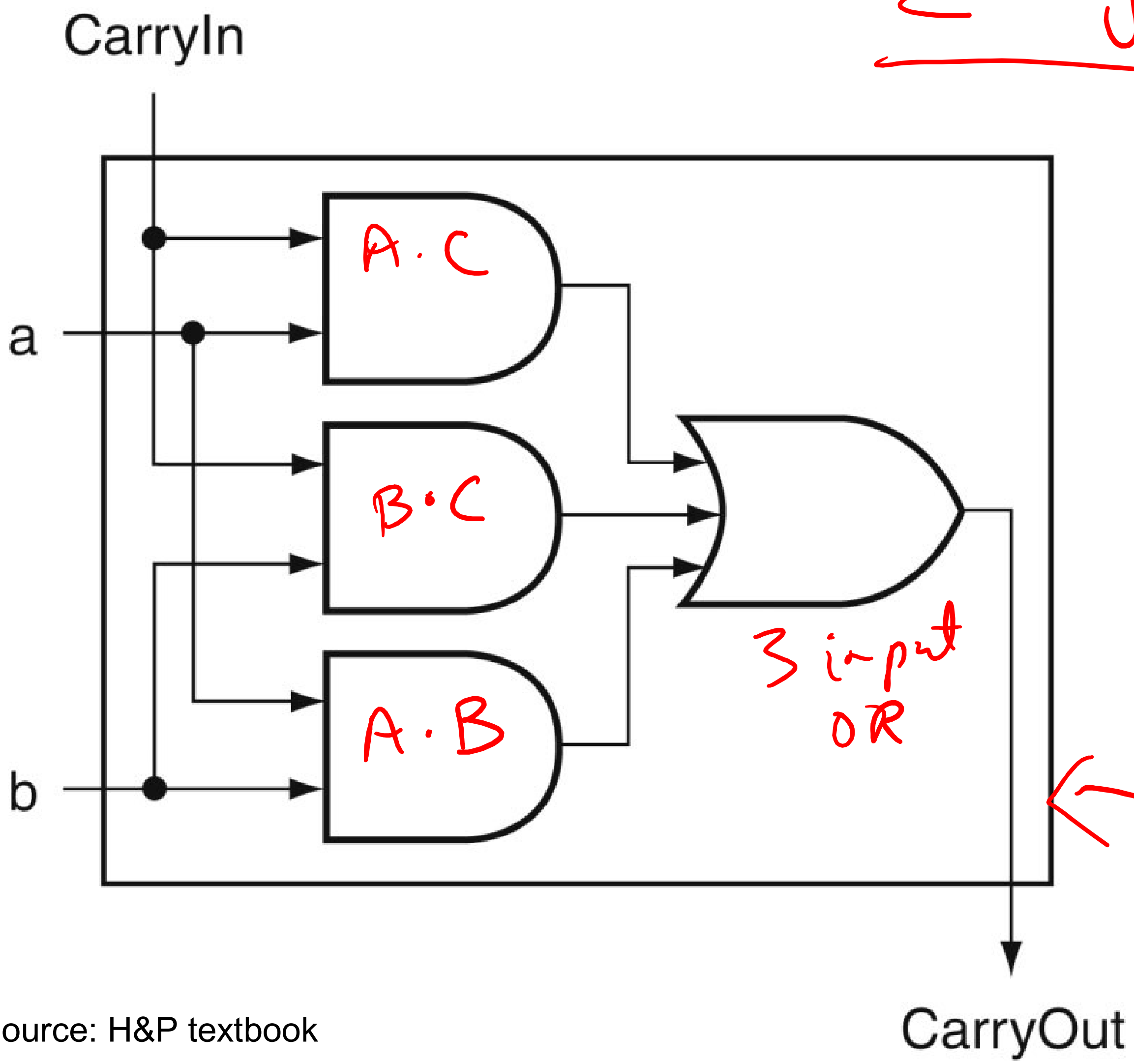
$$Sum = Cin\ .\ \overline{A}\ .\ \overline{B}\ +$$
$$B\ .\ \overline{Cin}\ .\ \overline{A}\ +$$
$$A\ .\ \overline{Cin}\ .\ \overline{B}\ +$$
$$A\ .\ B\ .\ Cin$$

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

Carry Out Logic

Adding 2 input bits (and carry-in)

= 2 gate delays



Source: H&P textbook

Equations:

$$\text{Sum} = \text{Cin} . \overline{A} . \overline{B} + B . \overline{\text{Cin}} . \overline{A} + A . \overline{\text{Cin}} . \overline{B} + A . B . \text{Cin}$$

$$\text{Cout} = A . B . \text{Cin} + A . B . \overline{\text{Cin}} + A . \text{Cin} . \overline{B} + B . \text{Cin} . \overline{A}$$

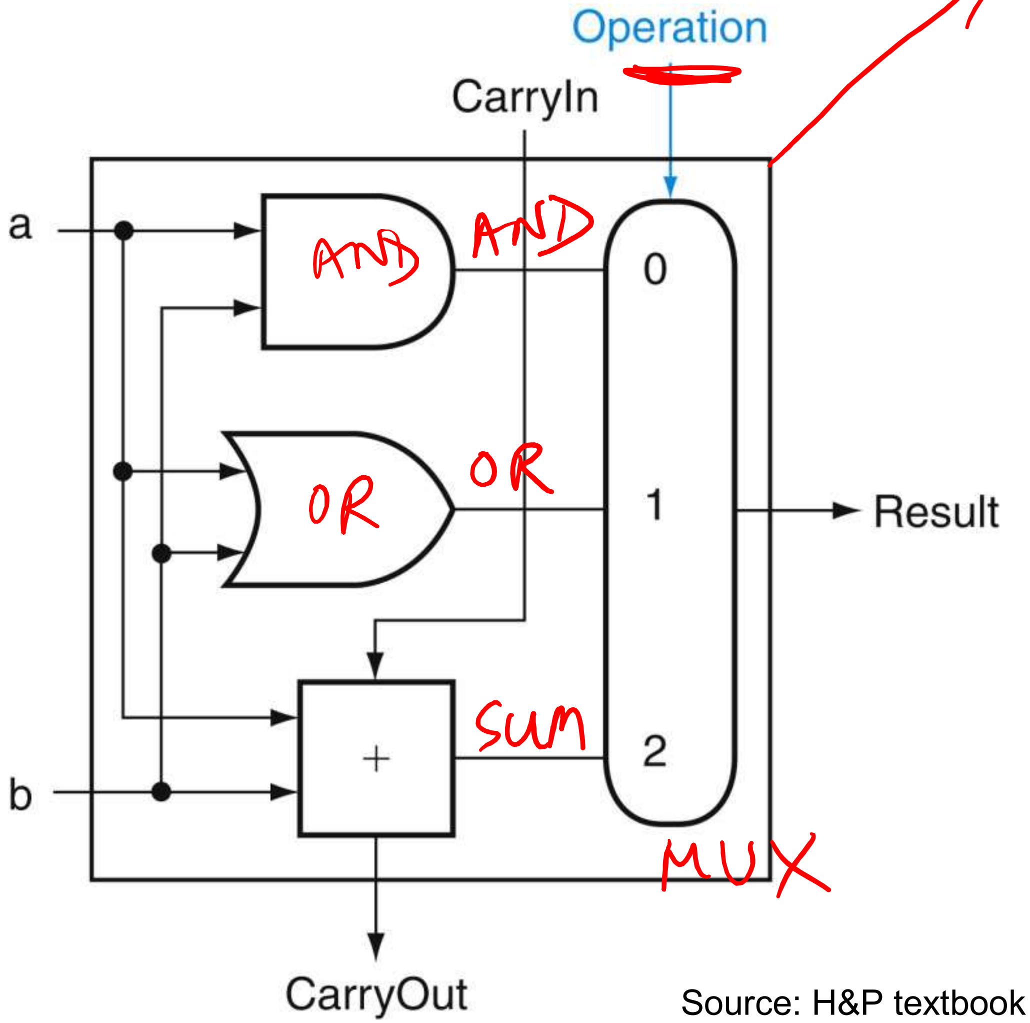
$$= \underline{A . B} + \underline{A . \text{Cin}} + \underline{B . \text{Cin}}$$

1-Bit ALU with Add, Or, And

- Multiplexor selects between Add, Or, And operations

which inst

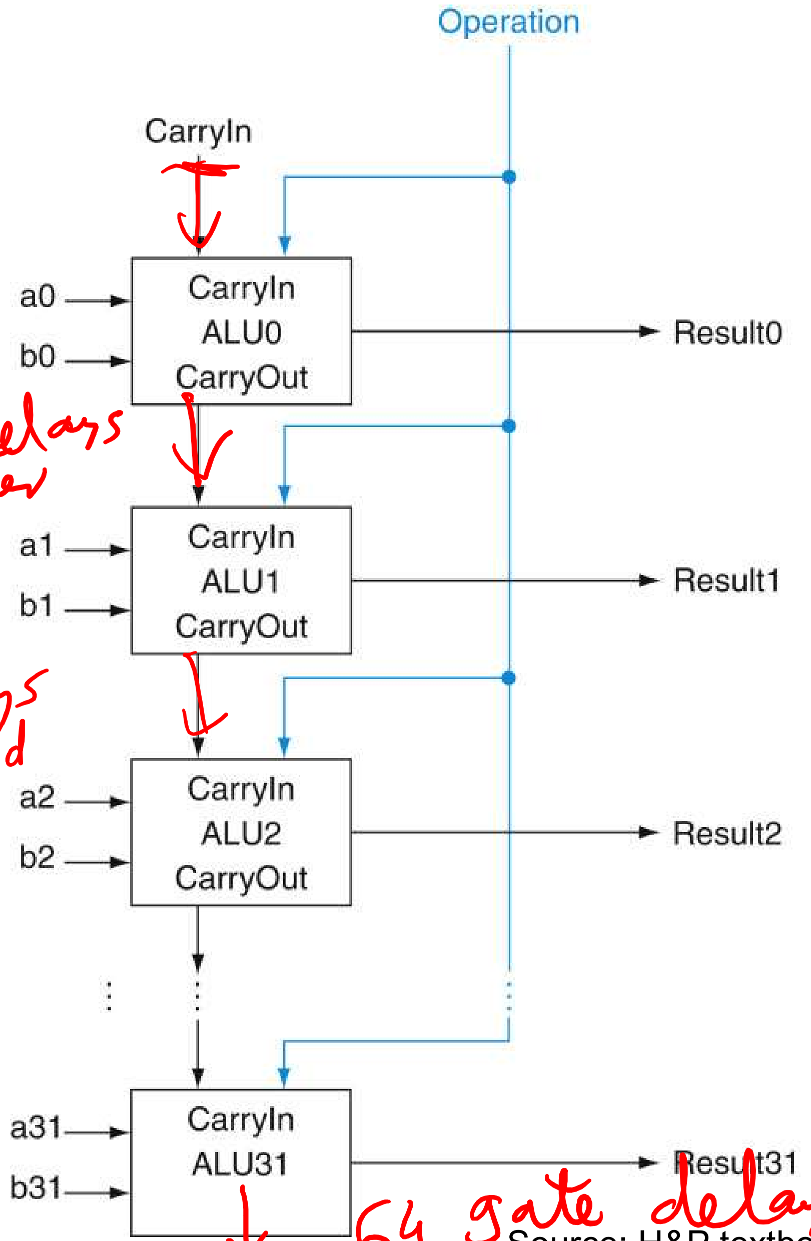
1 bit Add also AND OR



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



2 delays later

4 delays have elapsed

gate delays for 32b addition = 64

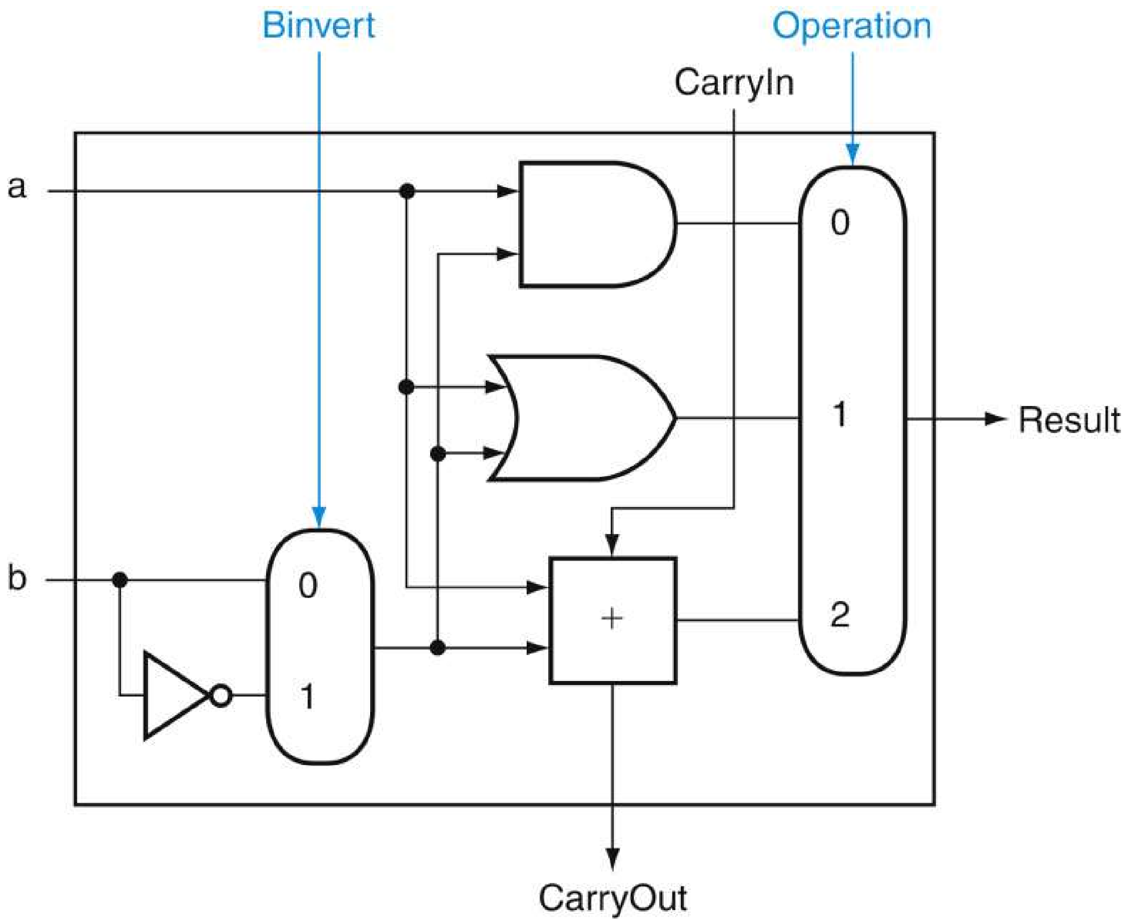
64 gate delays have elapsed

Source: H&P textbook

Incorporating Subtraction

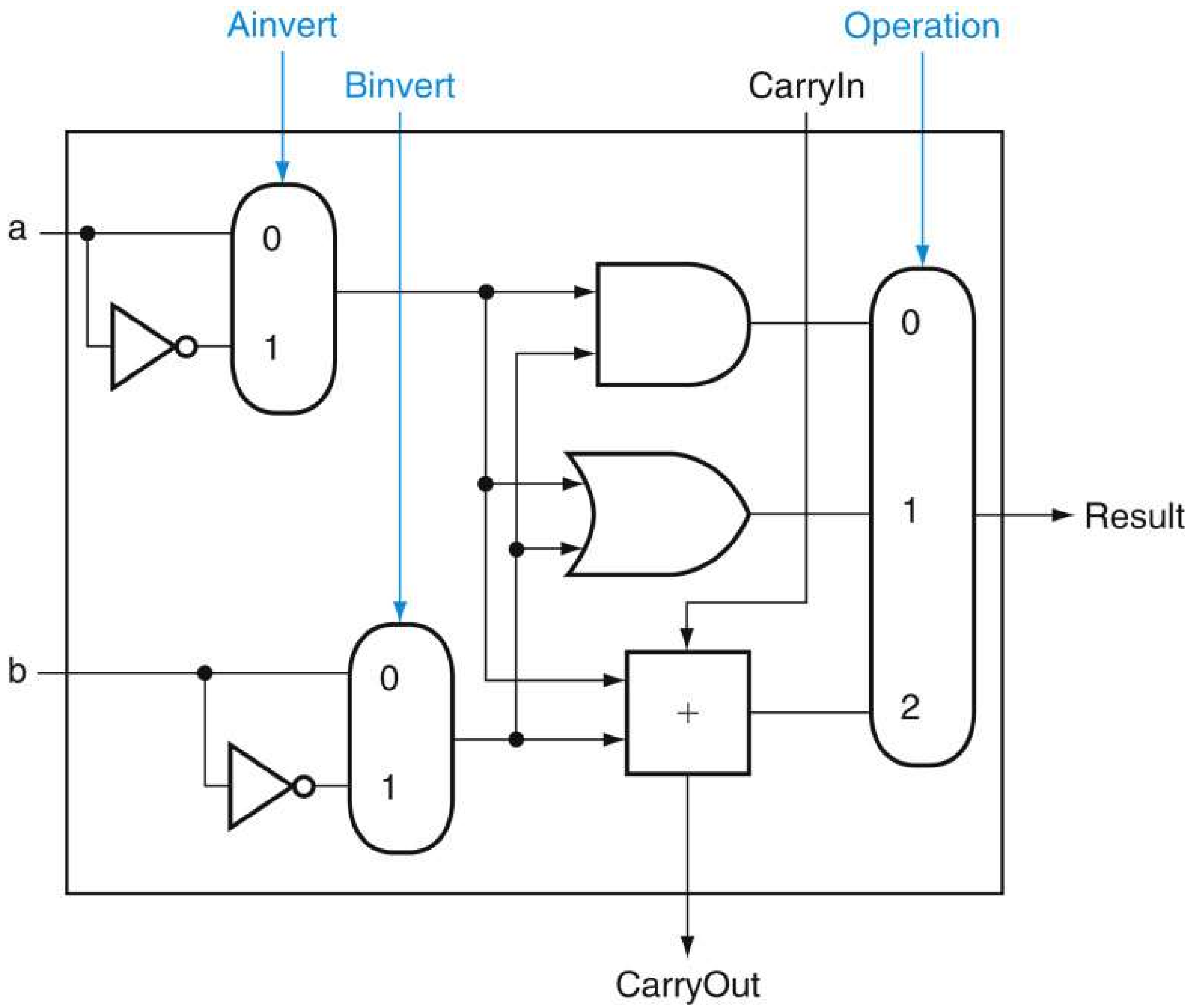
Must invert bits of B and add a 1

- 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

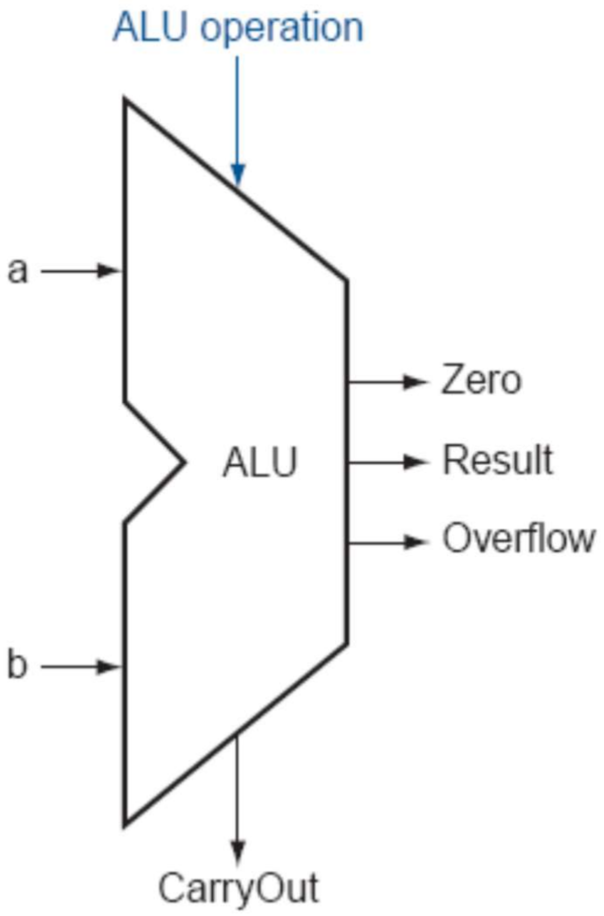


Source: H&P textbook

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