

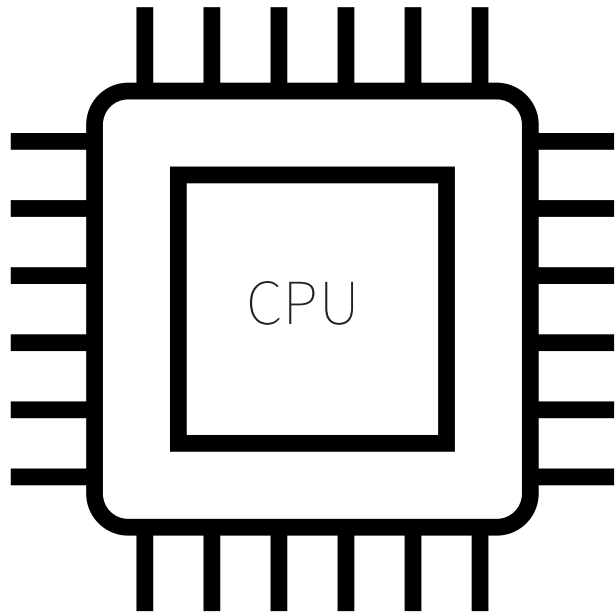
Gates and Logic

CS 3410: Computer System Organization and Programming



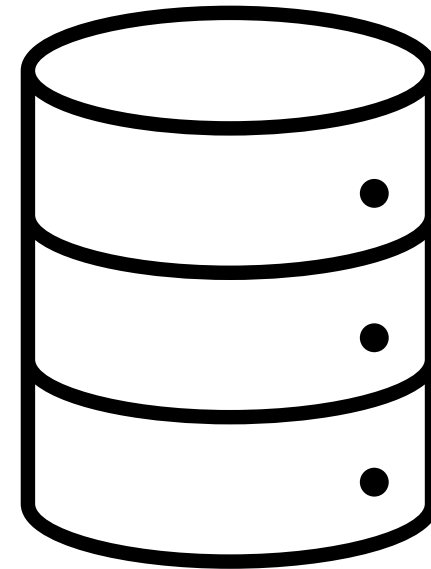
Simplified Computer Architecture

Processor

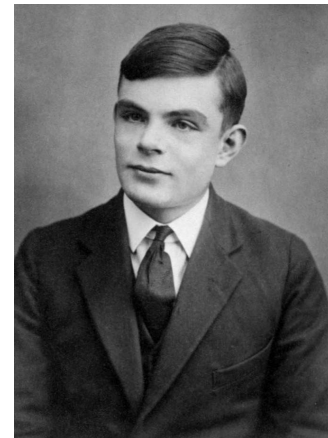


- Runs code; does computations
- Doesn't remember anything

Memory



- Can't compute anything
- Stores data



C

```
int x = 10;
x = 2 * x + 15;
```

compiler

RISC-V
assembly
language

```
addi x5, x0, 10 ← x5 = x0 + 10
mul  x5, x5, 2  ← x5 = x5 * 2
addi x5, x5, 15 ← x5 = x5 + 15
```

x0 = 0

How does add work?

assembler

RISC-V
machine
language

10	x0	x5	op = addi
000000001010	0000000000	0010100100	10010011
0000000000010001010010010010010011111			
000000001111	00101000	0010100100	10010011
15	x5	x5	op = addi

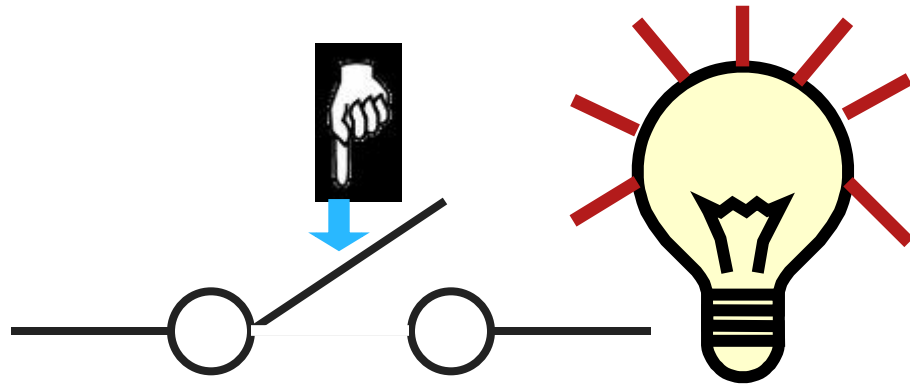
EVERYTHING IS A NUMBER!

Goals for Today: Bottom Up!

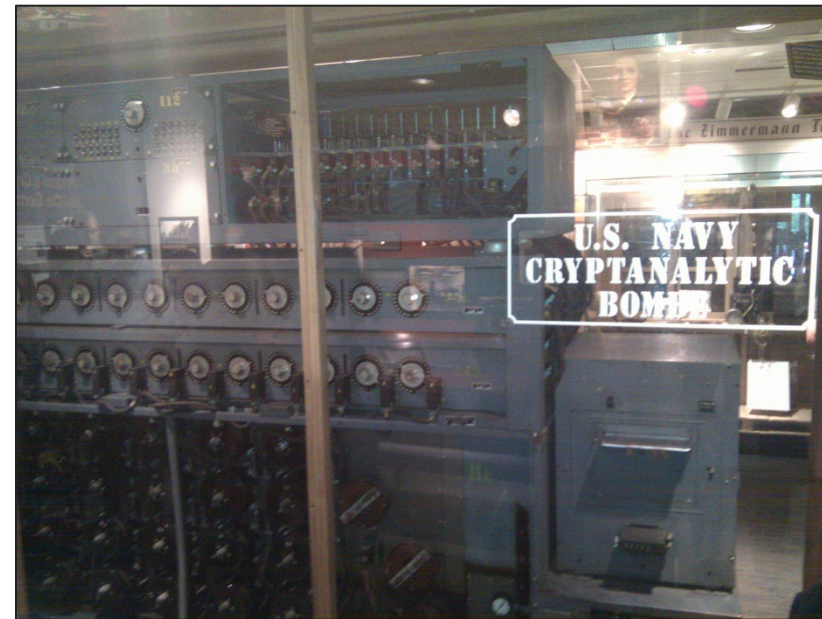
- From Switches to Logic Gates to Logic Circuits
- Logic Gates
 - From switches
 - Truth Tables
- Logic Circuits
 - From Truth Tables to Circuits (Sum of Products)
 - Identity Laws
- Binary Operations
 - One- and four-bit adders
 - Addition (two's complement)
- Transistors (electronic switch)

A switch

Acts as a *conductor* or *insulator*.

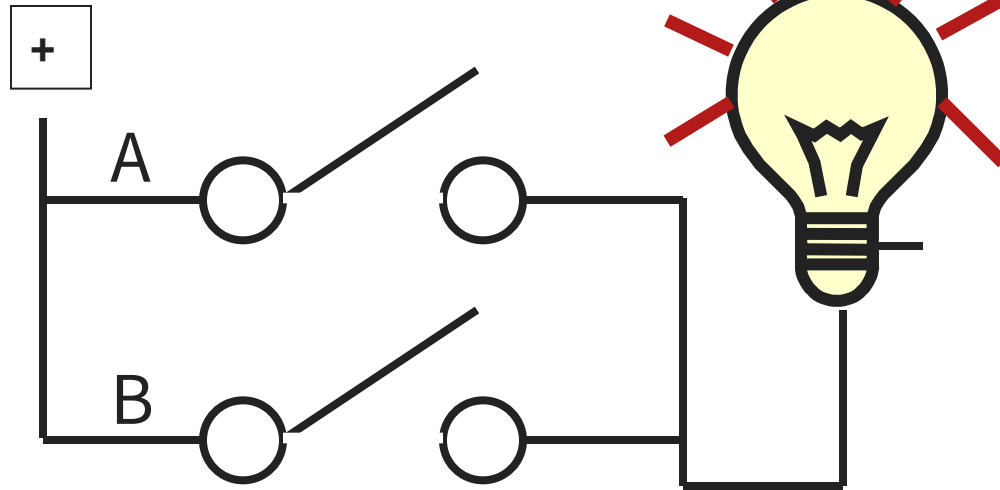


Can be used to build amazing things...



The Bombe used to break the German Enigma machine during World War II

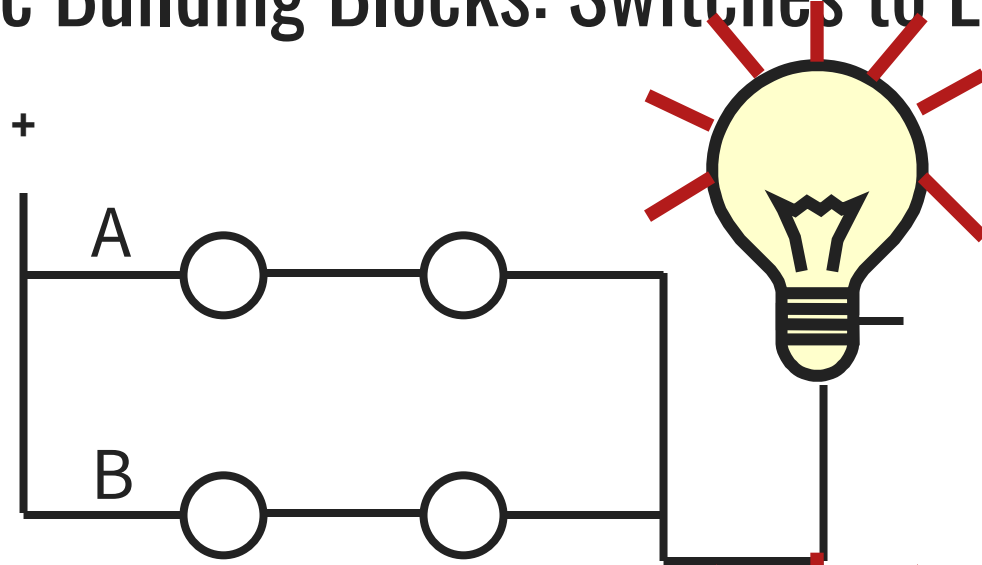
Basic Building Blocks: Switches to Logic Gates



Truth Table

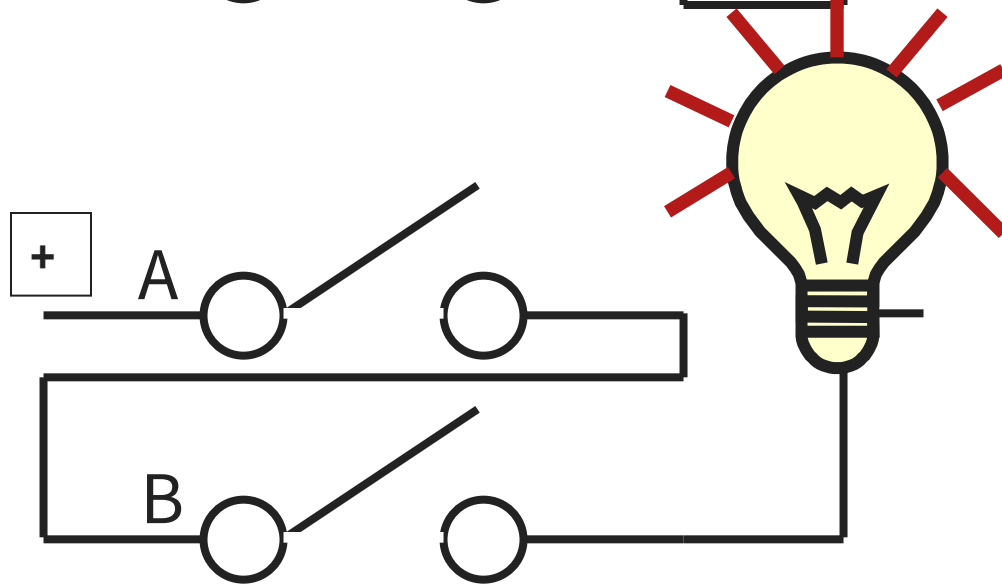
A	B	Light
OFF	OFF	
OFF	ON	
ON	OFF	
ON	ON	

Basic Building Blocks: Switches to Logic Gates



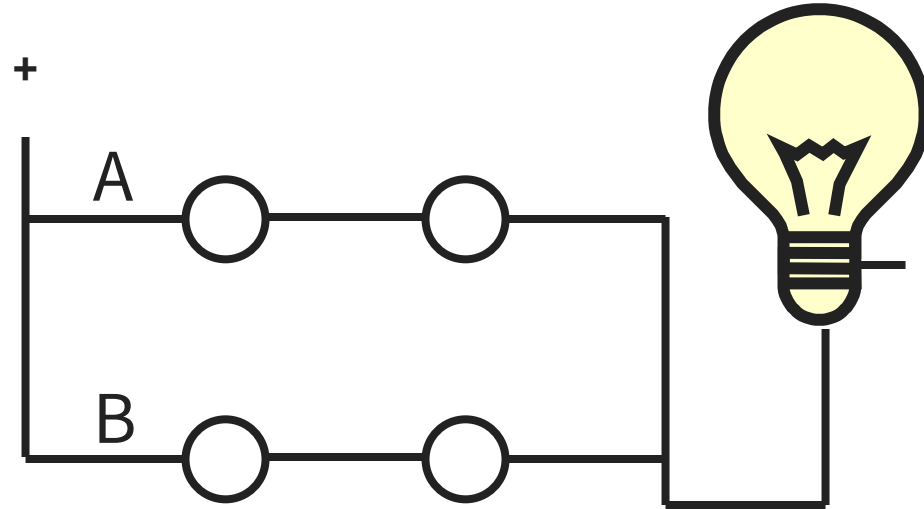
Truth Table

A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON



A	B	Light
OFF	OFF	
OFF	ON	
ON	OFF	
ON	ON	

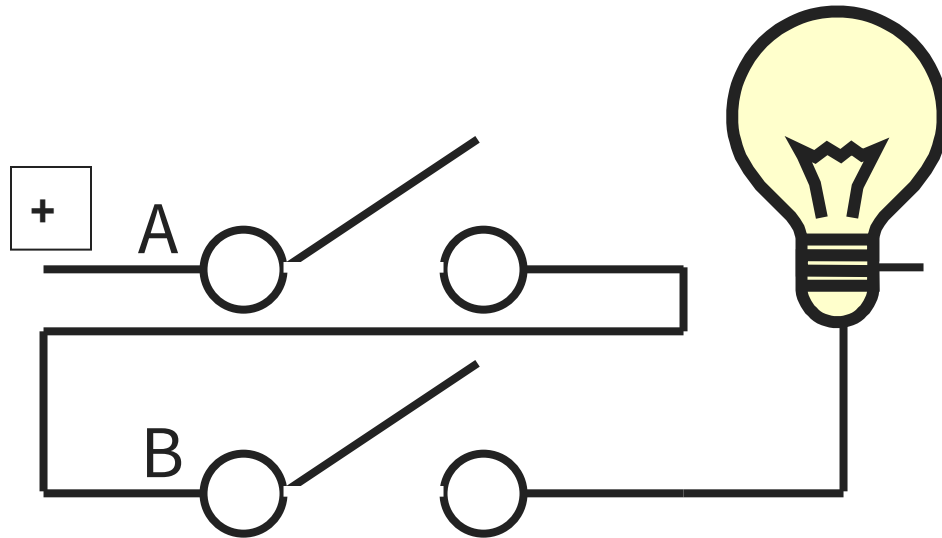
Basic Building Blocks: Switches to Logic Gates



- Either (OR)

Truth Table

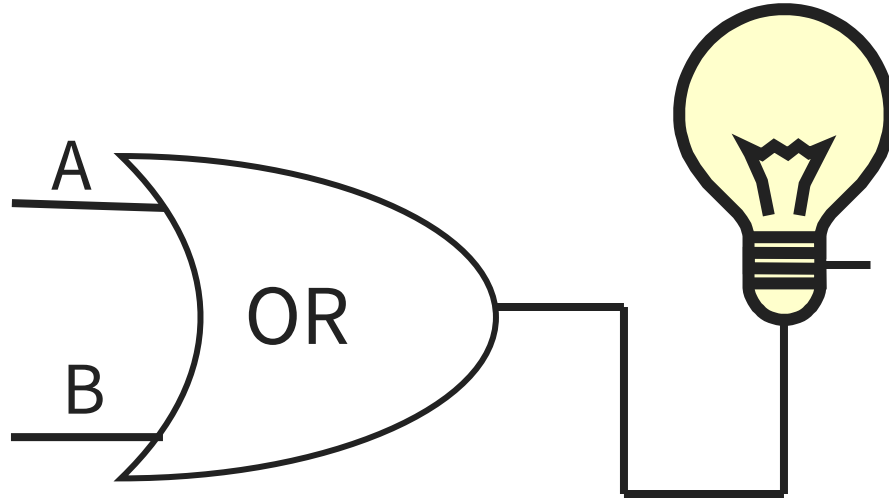
A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON



- Both (AND)

A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

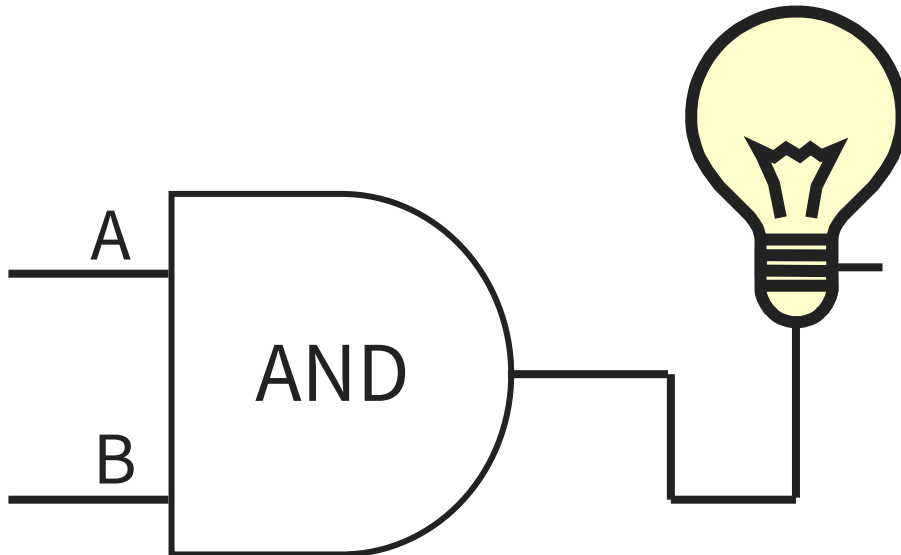
Basic Building Blocks: Switches to Logic Gates



- Either (OR)

Truth Table

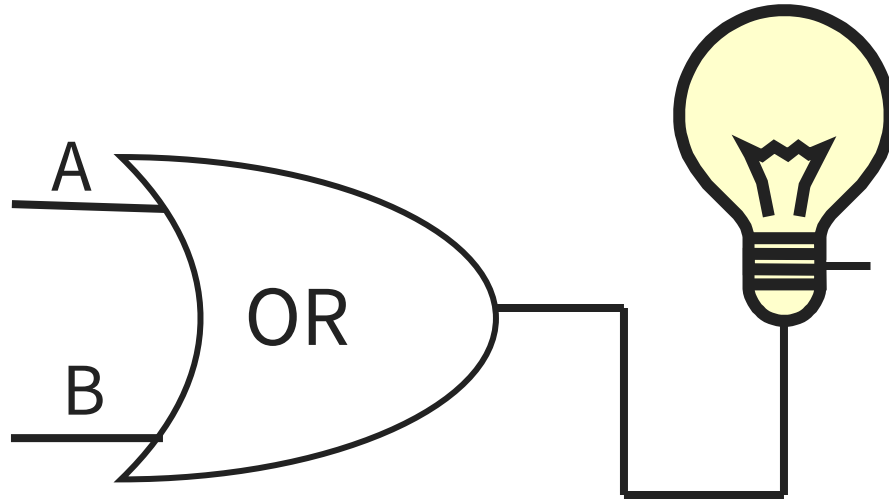
A	B	Light
OFF	OFF	OFF
OFF	ON	ON
ON	OFF	ON
ON	ON	ON



- Both (AND)

A	B	Light
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

Basic Building Blocks: Switches to Logic Gates

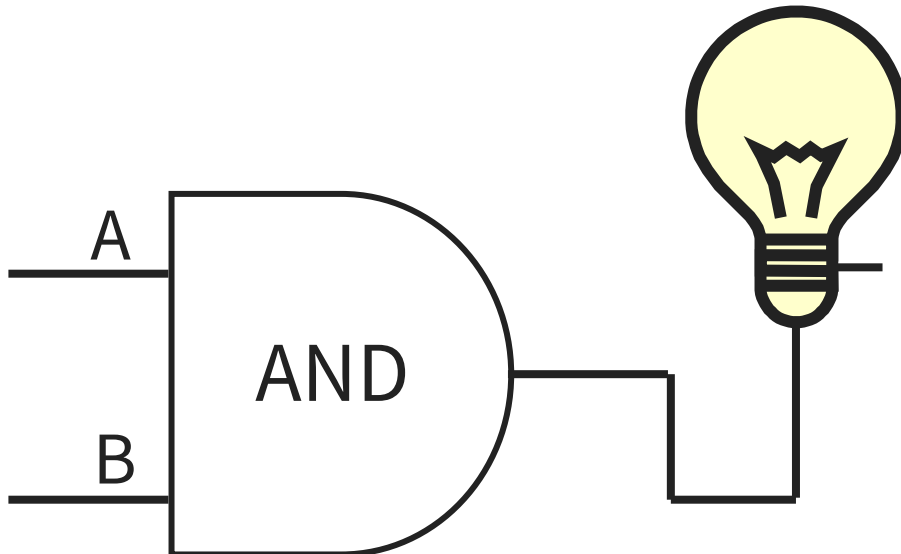


- Either (OR)

Truth Table

A	B	Light
0	0	0
0	1	1
1	0	1
1	1	1

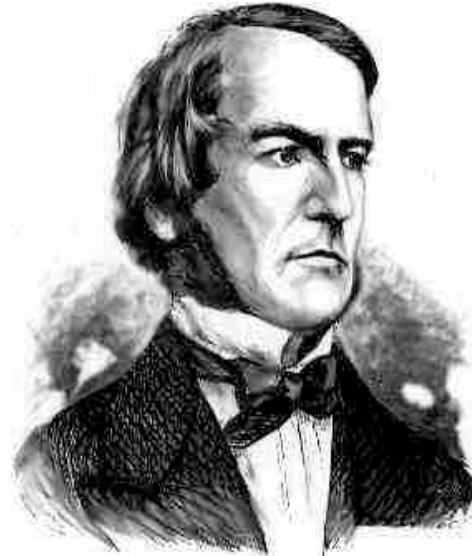
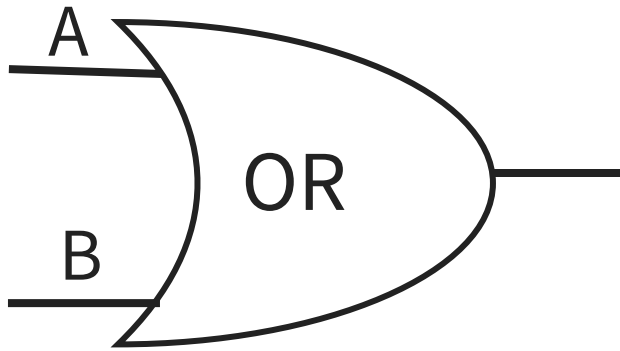
0 = OFF
1 = ON



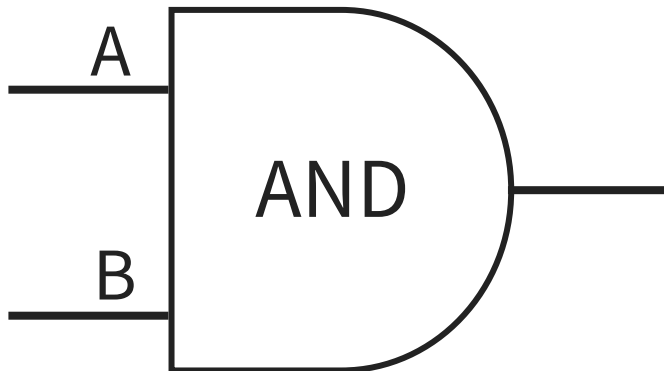
- Both (AND)

A	B	Light
0	0	0
0	1	0
1	0	0
1	1	1

Basic Building Blocks: Switches to Logic Gates



George Boole (1815-1864)



- **Did you know?**
- **George Boole:** Inventor of the idea of logic gates. He was born in Lincoln, England and he was the son of a shoemaker in a low class family.

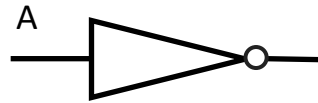
Takeaway

- Binary (two symbols: `true` and `false`) is the basis of Logic Design



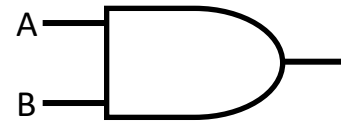
Building Functions: Logic Gates

- NOT:



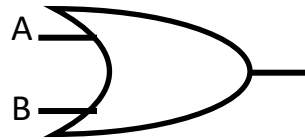
A	Out

- AND:



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

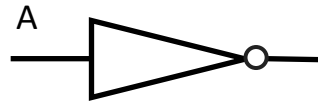
- Logic Gates

- digital circuit that either allows a signal to pass through it or not.
- Used to build logic functions
- There are seven basic logic gates:

AND, OR, NOT,

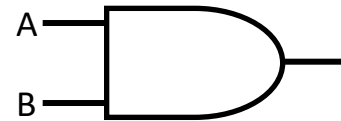
Building Functions: Logic Gates

- NOT:



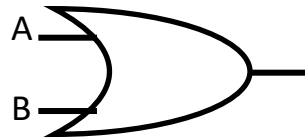
A	Out
0	1
1	0

- AND:



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

- Logic Gates

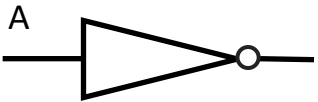
- digital circuit that either allows a signal to pass through it or not.
- Used to build logic functions
- There are seven basic logic gates:

AND, OR, NOT,

NAND (not AND), **NOR** (not OR), **XOR**, and **XNOR** (not XOR) [later]

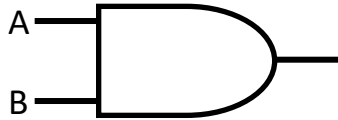
Building Functions: Logic Gates

- NOT:



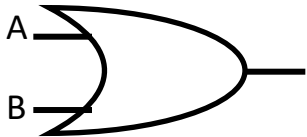
A	Out
0	1
1	0

- AND:



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

- OR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

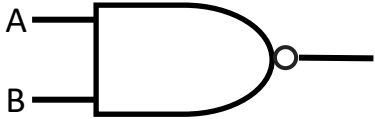
- Logic Gates

- digital circuit that either allows a signal to pass through it or not.
- Used to build logic functions
- There are seven basic logic gates:

AND, OR, NOT,

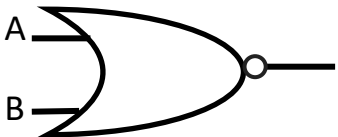
NAND (not AND), **NOR** (not OR), **XOR**, and **XNOR** (not XOR) [later]

NAND:



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Which Gate is this?






PolEV Question #1

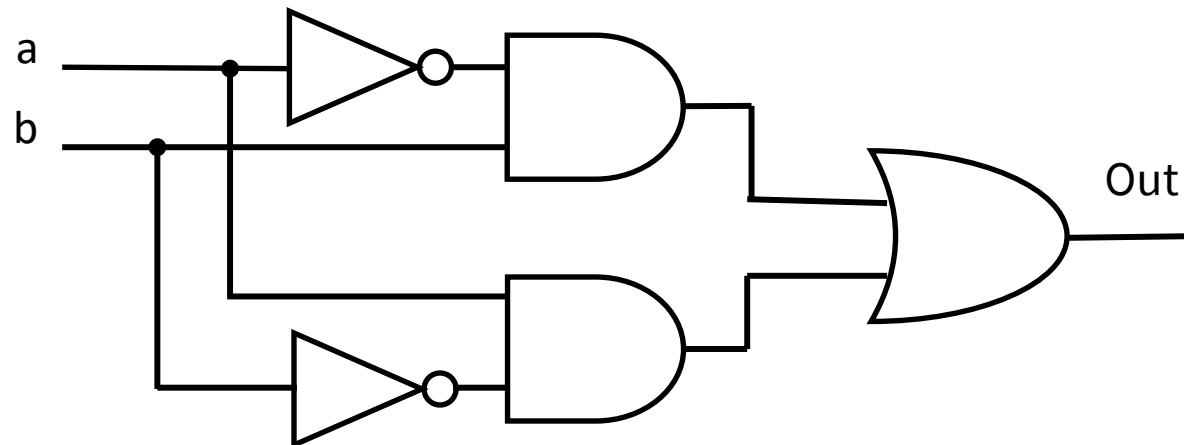
Function:

Symbol:

a	b	Out

Truth Table:

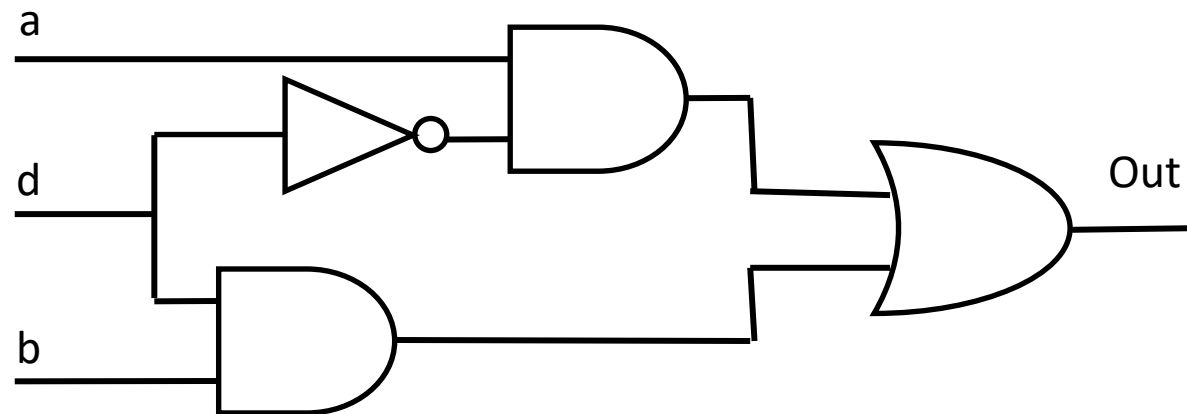
- (A) NOT 
- (B) OR 
- (C) XOR 
- (D) AND 
- (E) NAND 



Activity: Logic Gates

- Fill in the truth table, given the following Logic Circuit made from Logic AND, OR, and NOT gates.
- What does the logic circuit do?

a	b	d	Out
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



Goals for Today: Bottom Up!

- From Switches to Logic Gates to Logic Circuits
- Logic Gates
 - From switches
 - Truth Tables
- Logic Circuits
 - From Truth Tables to Circuits (Sum of Products)
 - Identity Laws
- Binary Operations
 - One- and four-bit adders
 - Addition (two's complement)
- Transistors (electronic switch)



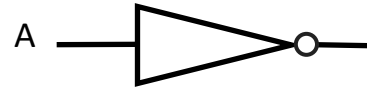
Next Goal

- Given a Logic function, create a Logic Circuit that implements the Logic Function...
- ...and, *with the minimum number of logic gates*
- Fewer gates: A cheaper (\$\$\$) circuit!



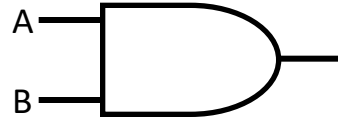
Logic Gates

NOT:



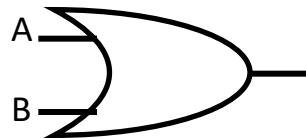
A	Out
0	1
1	0

AND:



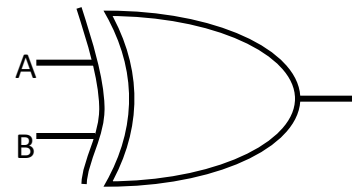
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

XOR:



A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

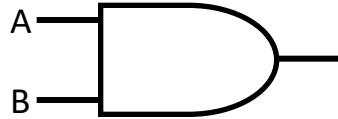
Logic Gates

NOT:



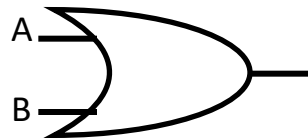
A	Out
0	1
1	0

AND:



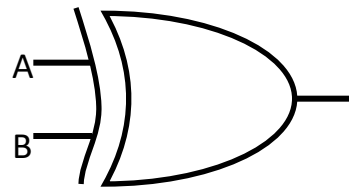
A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

OR:



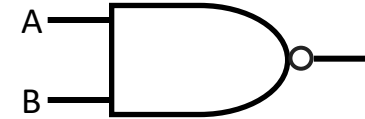
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

XOR:



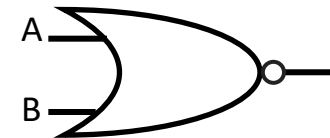
A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

NAND:



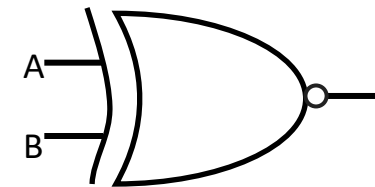
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

XNOR:



A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

Logic Implementation

- How to implement a desired logic function?

a	b	c	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Logic Implementation

- How to implement a desired logic function?

a	b	c	out	minterm
0	0	0	0	$\bar{a} \bar{b} \bar{c}$
0	0	1	1	$\bar{a} \bar{b} c$
0	1	0	0	$\bar{a} b \bar{c}$
0	1	1	1	$\bar{a} b c$
1	0	0	0	$a \bar{b} \bar{c}$
1	0	1	1	$a \bar{b} c$
1	1	0	0	$a b \bar{c}$
1	1	1	0	$a b c$

1) Write **minterms**

2) **sum of products:**

- OR of all minterms where out=1

Logic Implementation

- How to implement a desired logic function?

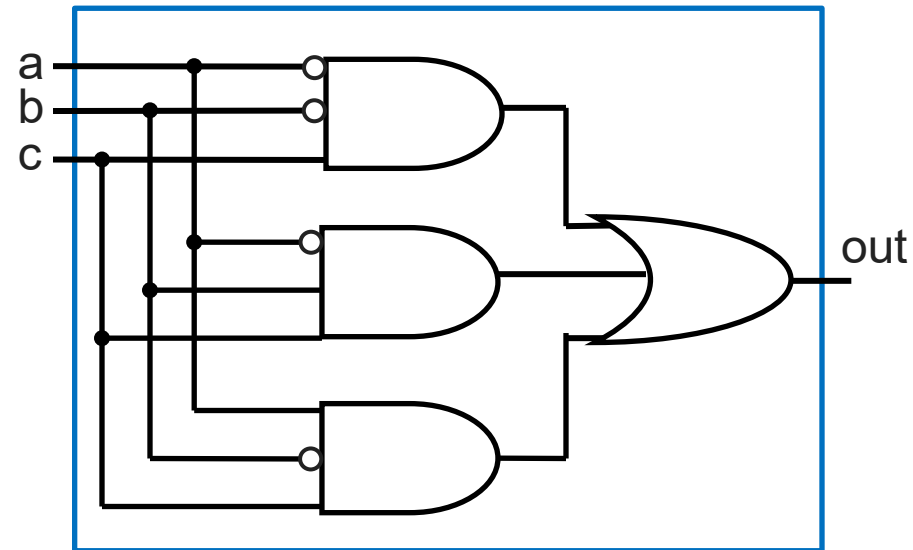
a	b	c	out	minterm
0	0	0	0	$\bar{a} \bar{b} \bar{c}$
0	0	1	1	$\bar{a} \bar{b} c$
0	1	0	0	$\bar{a} b \bar{c}$
0	1	1	1	$\bar{a} b c$
1	0	0	0	$a \bar{b} \bar{c}$
1	0	1	1	$a \bar{b} c$
1	1	0	0	$a b \bar{c}$
1	1	1	0	$a b c$

1) Write **minterms**

2) **sum of products:**

- OR of all minterms where out=1

- E.g. $out = \bar{a} \bar{b} c + \bar{a} b c + a \bar{b} c$



corollary: *any* combinational circuit *can be* implemented in two levels of logic
(ignoring inverters)

Logic Equations

- NOT:
 - $\text{out} = \bar{a} = !a = \neg a$
- AND:
 - $\text{out} = a \cdot b = a \& b = a \wedge b$
- OR:
 - $\text{out} = a + b = a | b = a \vee b$
- XOR:
 - $\text{out} = a \oplus b = a\bar{b} + \bar{a}b$
- Logic Equations
 - Constants: true = 1, false = 0
 - Variables: a, b, out, ...
 - Operators (above): AND, OR, NOT, etc.



Logic Equations

- NOT:
 - $\text{out} = \bar{a} = !a = \neg a$
- AND:
 - $\text{out} = a \cdot b = a \& b = a \wedge b$
- OR:
 - $\text{out} = a + b = a | b = a \vee b$
- XOR:
 - $\text{out} = a \oplus b = a\bar{b} + \bar{a}b$
- Logic Equations
 - Constants: true = 1, false = 0
 - Variables: a, b, out, ...
 - Operators (above): AND, OR, NOT, etc.

NAND:

- $\text{out} = \overline{a \cdot b} = !(a \& b) = \neg (a \wedge b)$

NOR:

- $\text{out} = \overline{a + b} = !(a | b) = \neg (a \vee b)$

XNOR:

- $\text{out} = \overline{a \oplus b} = ab + \bar{a}\bar{b}$



Identities

Identities useful for manipulating logic equations

- For optimization & ease of implementation

$$a + 0 =$$

$$a + 1 =$$

$$a + \bar{a} =$$

$$a \cdot 0 =$$

$$a \cdot 1 =$$

$$a \cdot \bar{a} =$$



Identities

Identities useful for manipulating logic equations

- For optimization & ease of implementation

$$\overline{(a + b)} =$$

$$\overline{(a \cdot b)} =$$

$$a + a b =$$

$$a(b+c) =$$

$$\overline{a(b + c)} =$$



Minimization Example

$$a + 0 = a$$

$$a + 1 = 1$$

$$a + \bar{a} = 1$$

$$a \cdot 0 = 0$$

$$a \cdot 1 = a$$

$$a \cdot \bar{a} = 0$$

$$a + a b = a$$

$$a(b+c) = ab + ac$$

$$\overline{(a + b)} = \bar{a} \bullet \bar{b}$$

$$\overline{(ab)} = \bar{a} + \bar{b}$$

$$\overline{a(b + c)} = \bar{a} + \bar{b} \bullet \bar{c}$$

Minimize this logic equation:

$$(a+b)(a+c) =$$



Minimization Example

PolIEV Question #2

$$a + 0 = a$$

$$a + 1 = 1$$

$$a + \bar{a} = 1$$

$$a \cdot 0 = 0$$

$$a \cdot 1 = a$$

$$a \cdot \bar{a} = 0$$

$$a + a b = a$$

$$a (b+c) = ab + ac$$

$$\overline{(a + b)} = \bar{a} \cdot \bar{b}$$

$$\overline{(ab)} = \bar{a} + \bar{b}$$

$$\overline{a(b + c)} = \bar{a} + \bar{b} \cdot \bar{c}$$

$$(a+b)(a+c) \rightarrow a + bc$$

How many gates were required before and after?

BEFORE

AFTER

(A) 2 OR

1 OR

(B) 2 OR, 1 AND

2 OR

(C) 2 OR, 1 AND

1 OR, 1 AND

(D) 2 OR, 2 AND

2 OR

(E) 2 OR, 2 AND

2 OR, 1 AND



Checking Equality w/Truth Tables

circuits \leftrightarrow truth tables \leftrightarrow equations

Example: $(a+b)(a+c) = a + bc$

a	b	c					
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					



Checking Equality w/Truth Tables

circuits \leftrightarrow truth tables \leftrightarrow equations

Example: $(a+b)(a+c) = a + bc$

a	b	c					
0	0	0					
0	0	1					
0	1	0					
0	1	1					
1	0	0					
1	0	1					
1	1	0					
1	1	1					



Checking Equality w/Truth Tables

circuits \leftrightarrow truth tables \leftrightarrow equations

Example: $(a+b)(a+c) = a + bc$

a	b	c	a+b	a+c	LHS	bc	RHS
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	1	1	1	1
1	0	0	1	1	1	0	1
1	0	1	1	1	1	0	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Goals for Today: Bottom Up!

- From Switches to Logic Gates to Logic Circuits
- Logic Gates
 - From switches
 - Truth Tables
- Logic Circuits
 - From Truth Tables to Circuits (Sum of Products)
 - Identity Laws
- Binary Operations
 - One- and four-bit adders
 - Addition (two's complement)
- Transistors (electronic switch)

Next Goal

Binary Arithmetic: Add and Subtract two binary numbers

Binary Addition (Revisited)

Addition works the same for all bases

- Add the digits in each position
- Propagate the carry

Binary addition is pretty easy

- Combine two bits at a time
- Along with a carry

$$\begin{array}{r} 1 \\ 183 \\ + 254 \\ \hline 437 \end{array}$$

Carry-in

$$\begin{array}{r} 111 \text{ Carry-out} \\ 001110 \\ + 011100 \\ \hline 101010 \end{array}$$

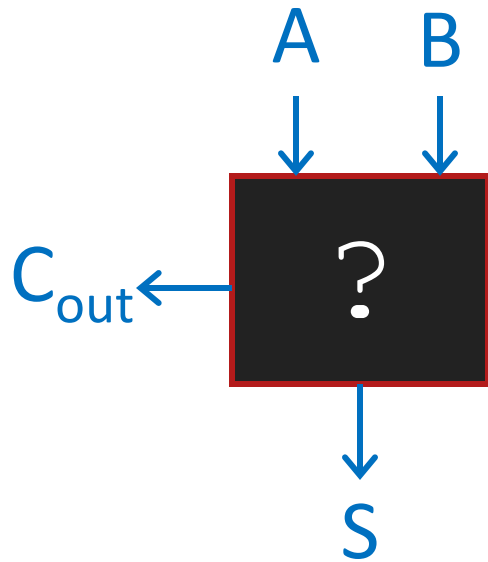


Binary Addition

- Binary addition requires
 - Add of **two bits** PLUS **carry-in**
 - Also, **carry-out** if necessary



1-bit Half Adder



A	B	C _{out}	S
0	0		
0	1		
1	0		
1	1		

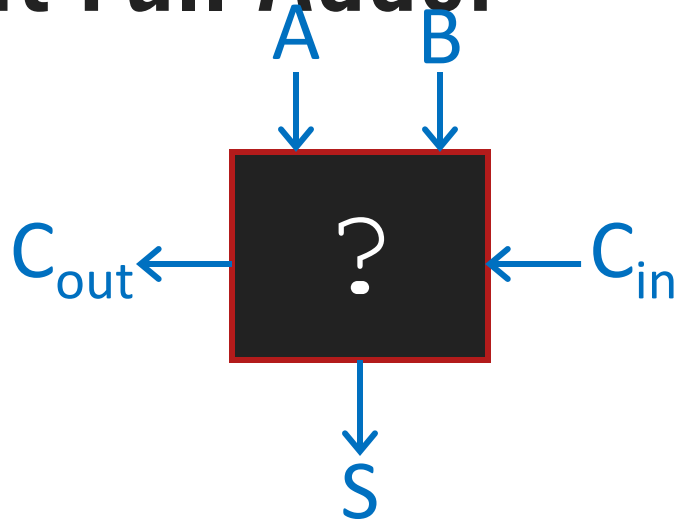
- Adds two 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- No carry-in

PolEV Question #3

What is the equation for C_{out}?

- a) $A + B$
- b) AB
- c) $A \oplus B$
- d) $A + !B$
- e) $!A!B$

1-bit Full Adder

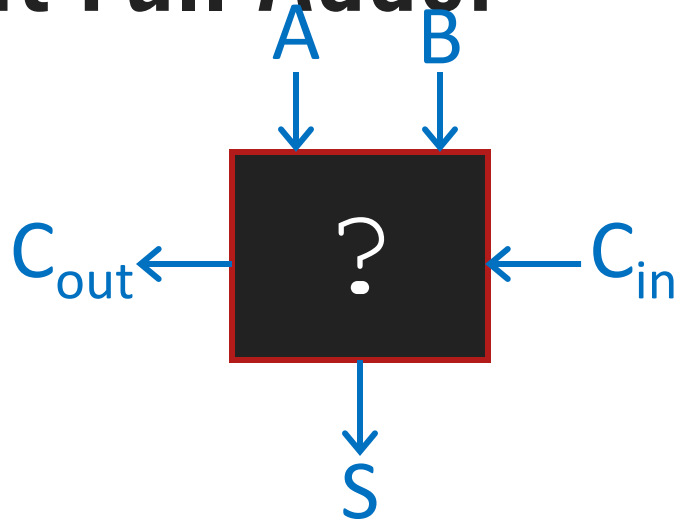


A	B	C_{in}	C_{out}	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- Can be cascaded

- Fill in Truth Table
- Create Sum-of-Product Form
- Draw the Circuits

1-bit Full Adder



- Adds three 1-bit numbers
- Computes 1-bit result and 1-bit carry-out
- Can be cascaded

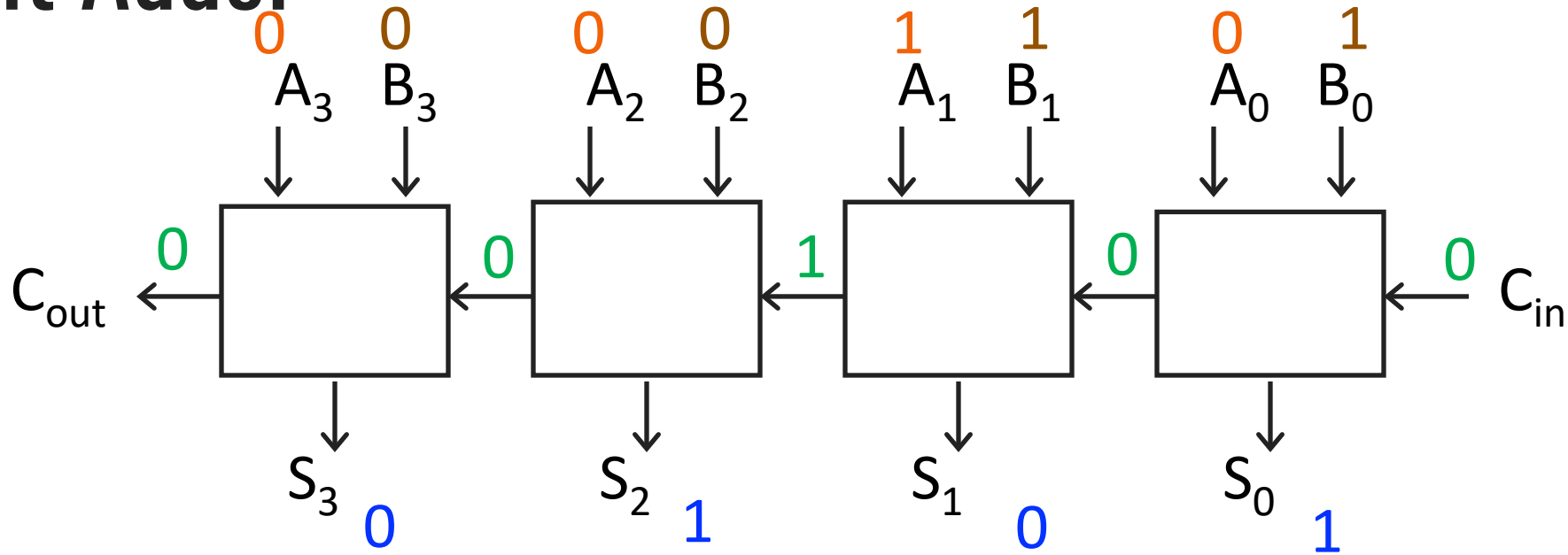
A	B	C _{in}	C _{out}	S
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

PolEV Question #4

What is the equation for C_{out}?

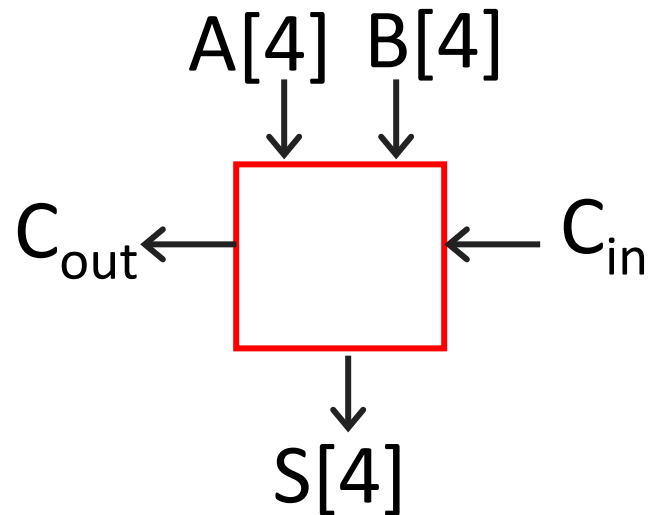
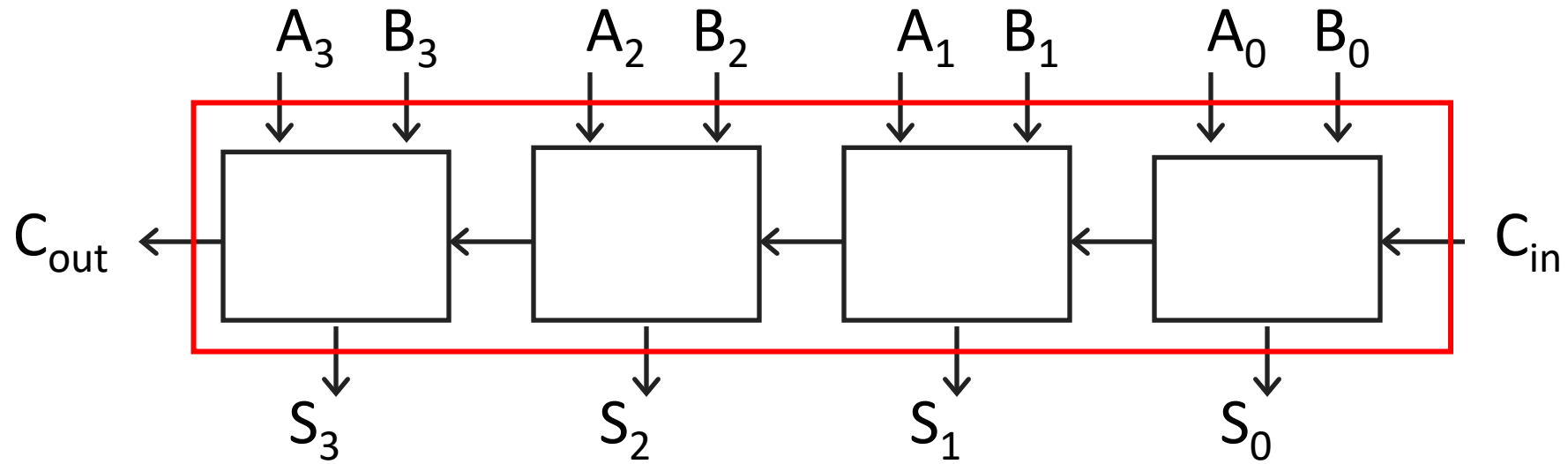
- a) $A + B + C_{in}$
- b) $\neg A + \neg B + \neg C_{in}$
- c) $A \oplus B \oplus C_{in}$
- d) $\overline{A}BC + \overline{A}B\overline{C} + A\overline{B}C + ABC$
- e) $\overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$

4-bit Adder



- Adds two 4-bit numbers, along with carry-in
- Computes 4-bit result and carry out
- $3 + 2 = 5$
- Carry-out \rightarrow result $>$ 4 bits

4-bit Adder

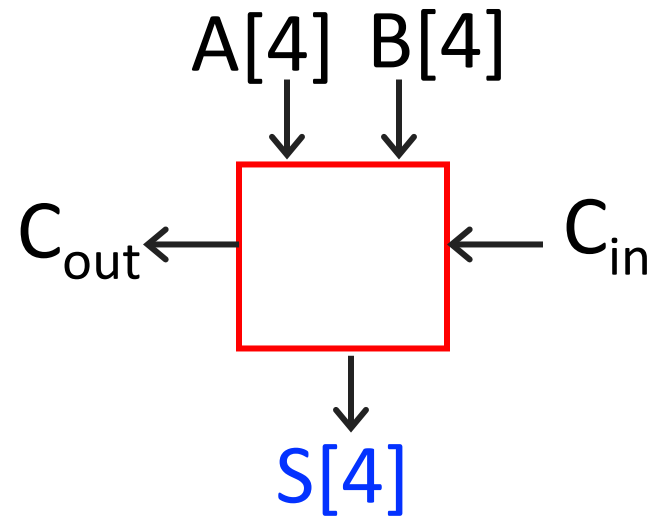


Build it
and
box it!
(Theme of 3410)

PolIEV Question #5

What's the largest **sum** you can calculate with a 4 bit adder?
(Give your answer in base 10. Assume unsigned numbers)

- a) 4
- b) 1,111
- c) 15
- d) 16
- e) 4000



Binary Subtraction

Why create a new circuit?

Just use addition using two's complement math

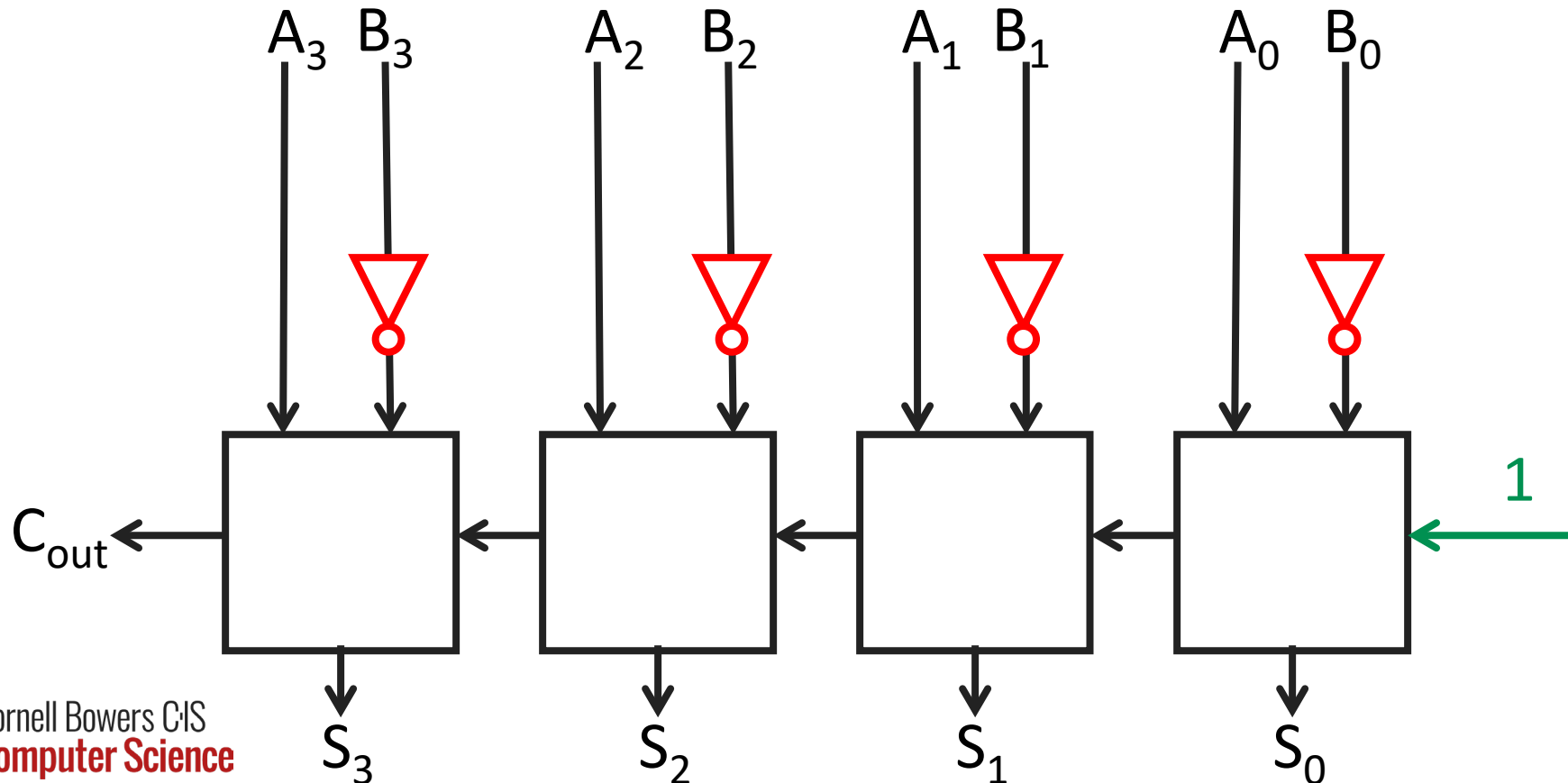
How?

Binary Subtraction

Two's Complement Subtraction

- Subtraction is addition with a negated operand
 - Negation is done by **inverting all bits** and **adding one**

$$A - B = A + (-B) = A + (\bar{B} + 1)$$

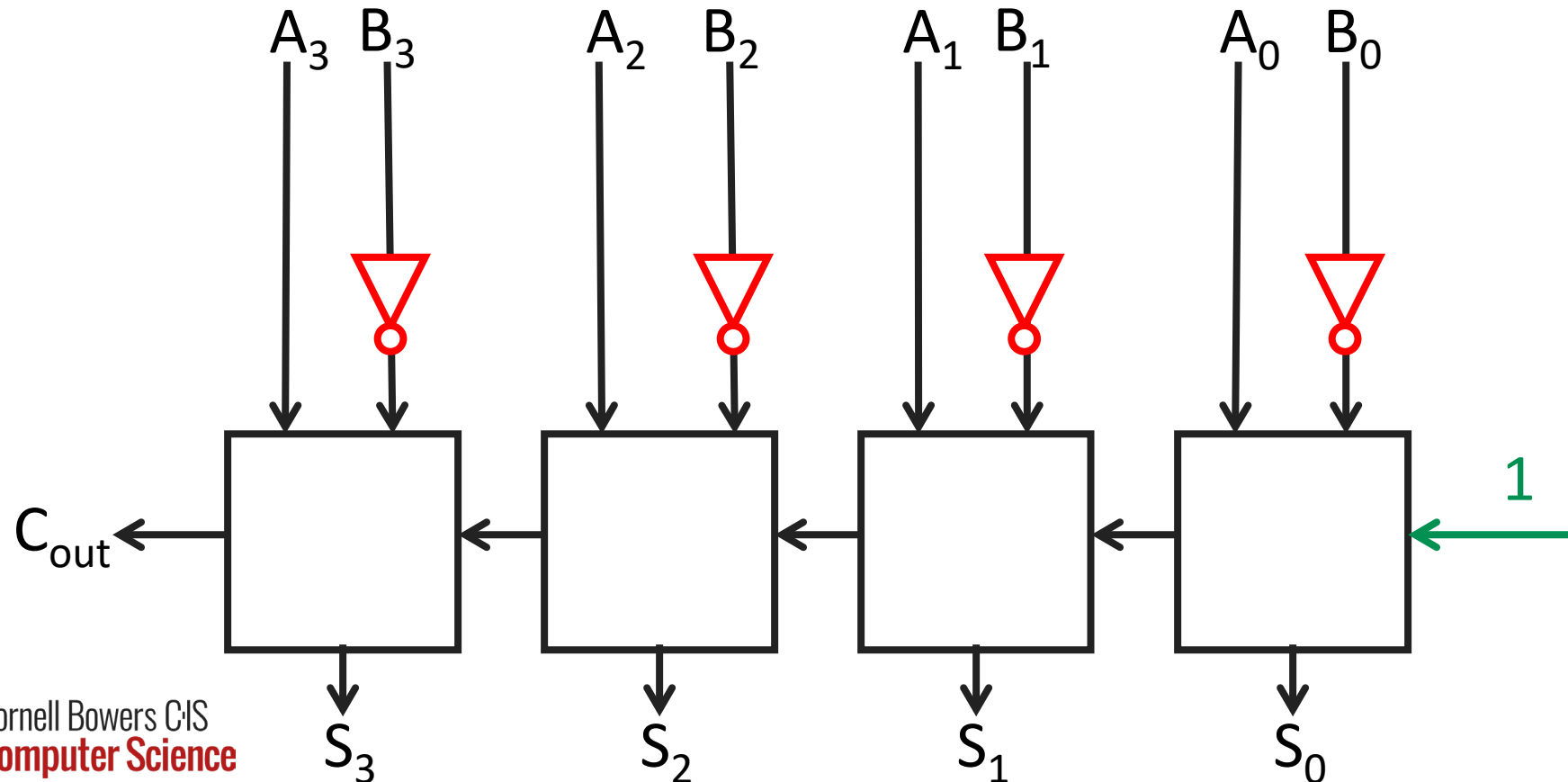


Binary Subtraction

Two's Complement Subtraction

- Subtraction is addition with a negated operand
 - Negation is done by **inverting all bits** and **adding one**

$$A - B = A + (-B) = A + (\bar{B} + 1)$$



Add or subtract with
XOR gate

sub?	B_0	new B_0
0	0	0
0	1	1
1	0	1
1	1	0

if subtracting, invert B_0

Binary Subtraction

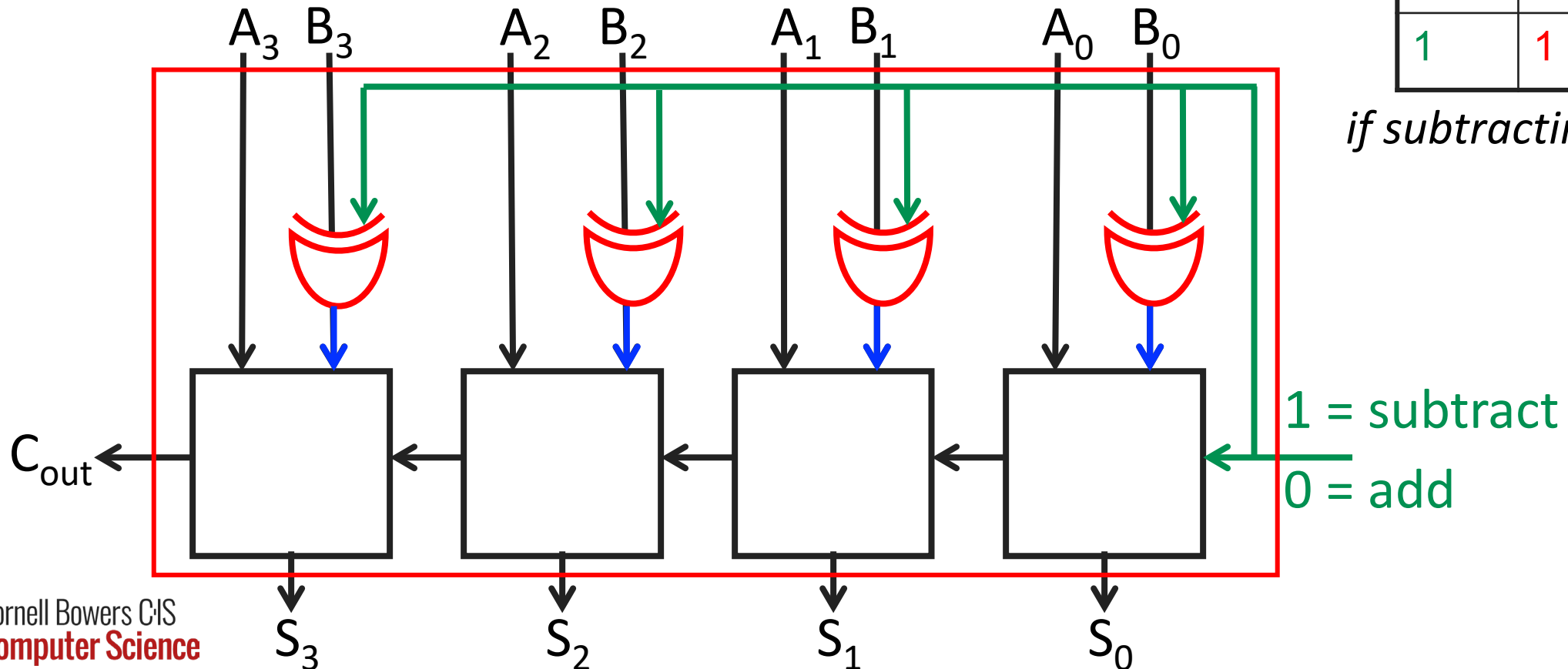
Two's Complement Subtraction

- Subtraction is addition with a negated operand
 - Negation is done by **inverting all bits** and **adding one**
- $$A - B = A + (-B) = A + (\bar{B} + 1)$$

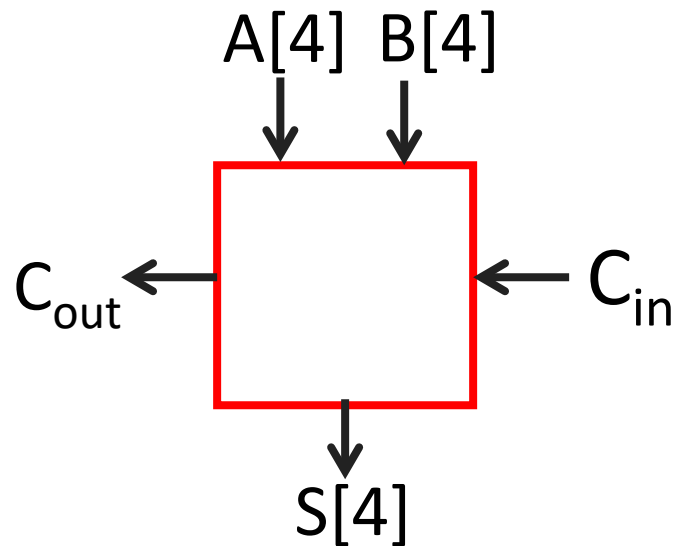
Add or subtract with
XOR gate

sub?	B_0	new B_0
0	0	0
0	1	1
1	0	1
1	1	0

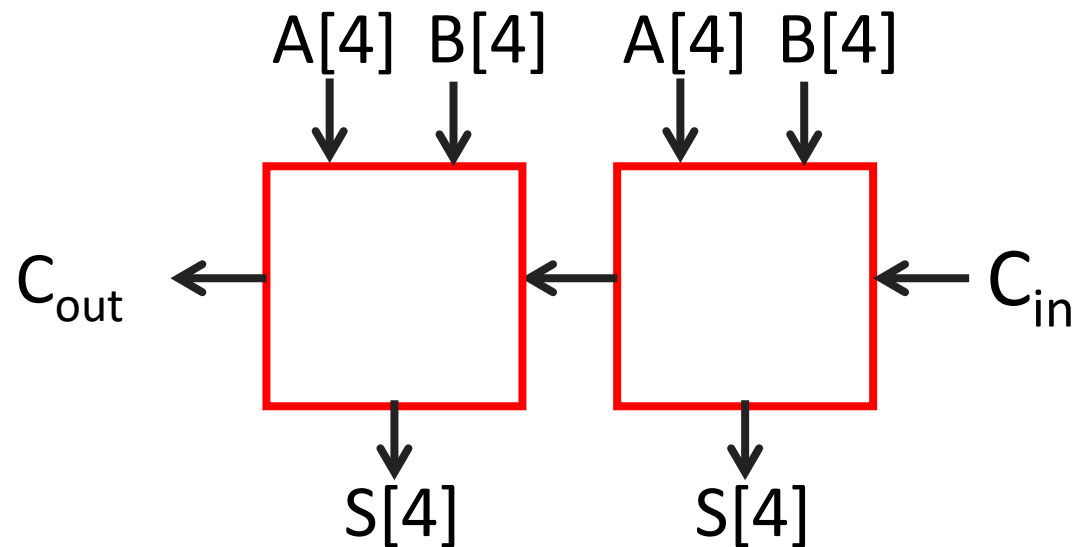
if subtracting, invert B_0



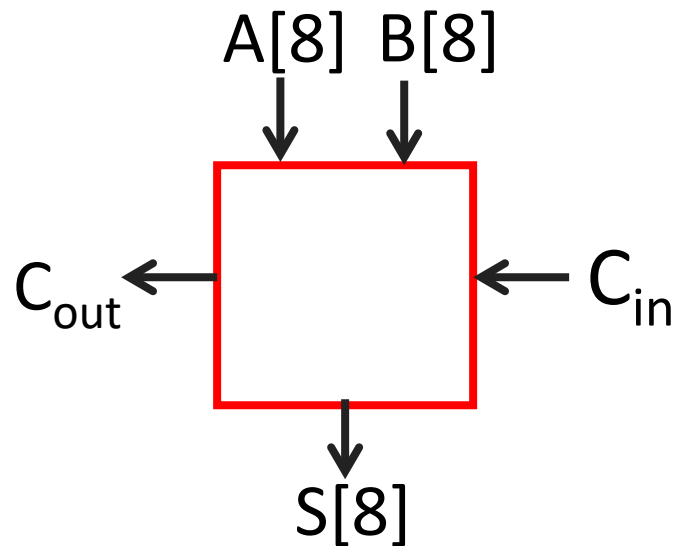
4-bit Adder with Two's Complement



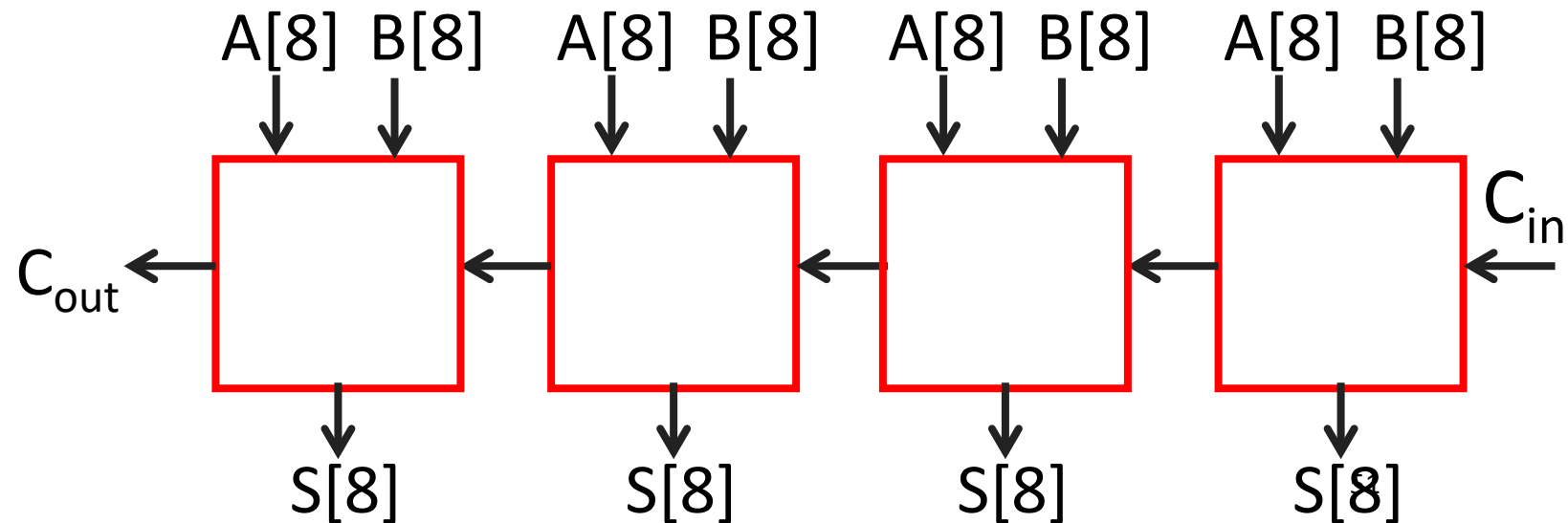
8-bit Adder with Two's Complement



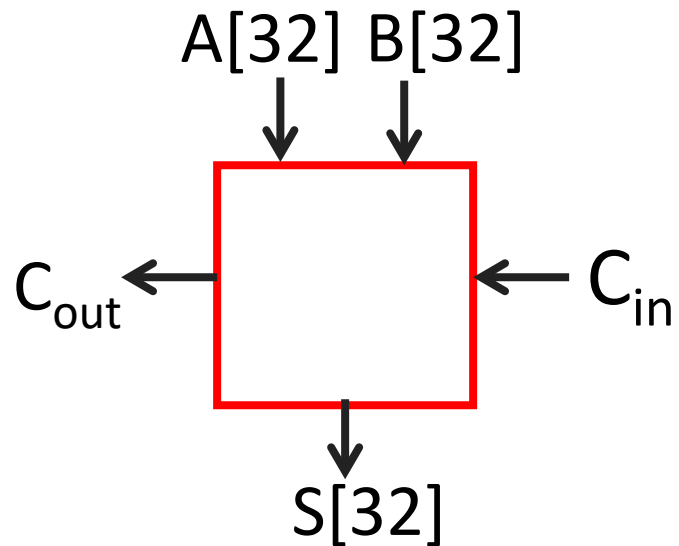
8-bit Adder with Two's Complement



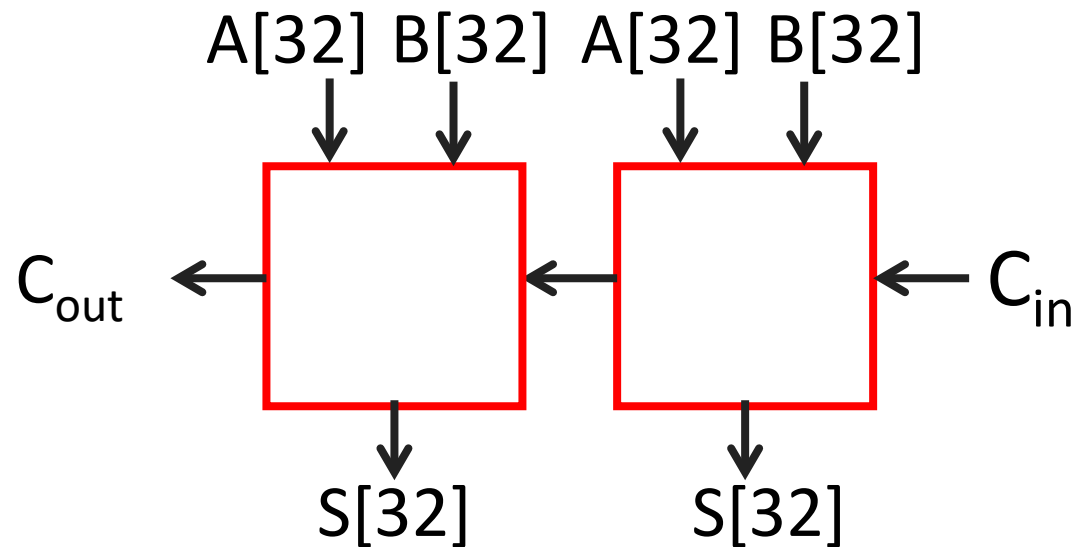
32-bit Adder with Two's Complement



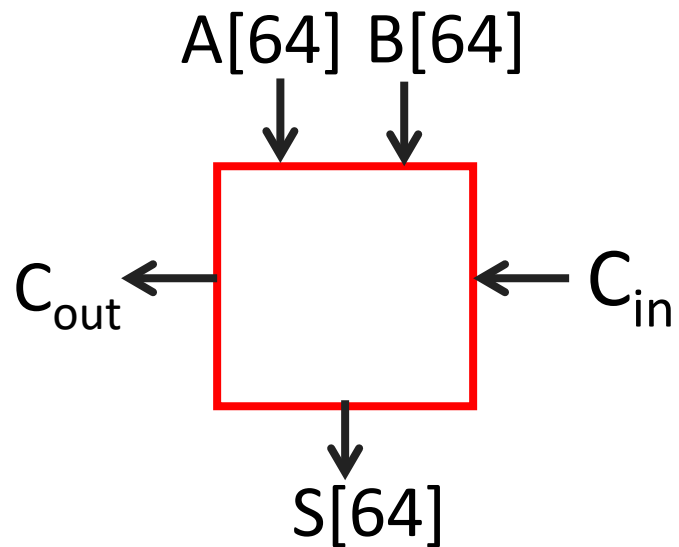
32-bit Adder with Two's Complement



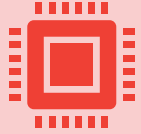
64-bit Adder with Two's Complement



64-bit Adder with Two's Complement



Takeaway



Digital computers are implemented via logic circuits and thus represent *all* numbers in binary (base 2).



We (humans) often write numbers as decimal and hexadecimal for convenience, so need to be able to convert to binary and back (to understand what computer is doing!).



Adding two 1-bit numbers generalizes to adding two numbers of any size since 1-bit full adders can be cascaded.

Goals for Today: Bottom Up!

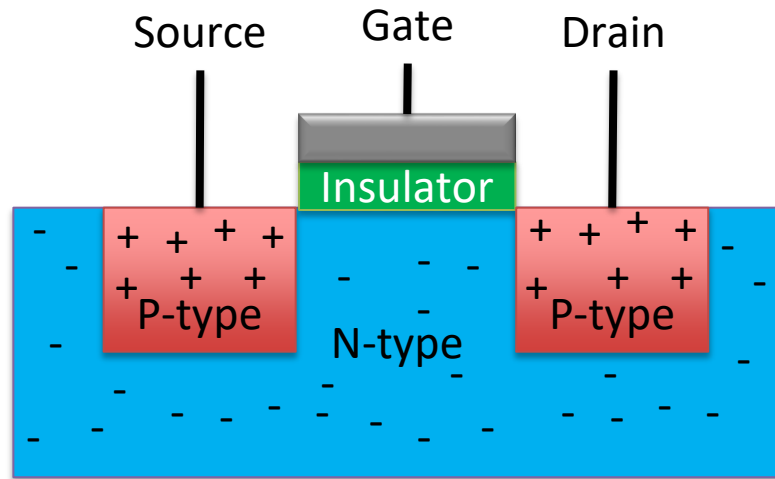
- From Switches to Logic Gates to Logic Circuits
- Logic Gates
 - From switches
 - Truth Tables
- Logic Circuits
 - From Truth Tables to Circuits (Sum of Products)
 - Identity Laws
- Binary Operations
 - One- and four-bit adders
 - Addition (two's complement)
- Transistors (electronic switch)

Silicon Valley & the Semiconductor Industry

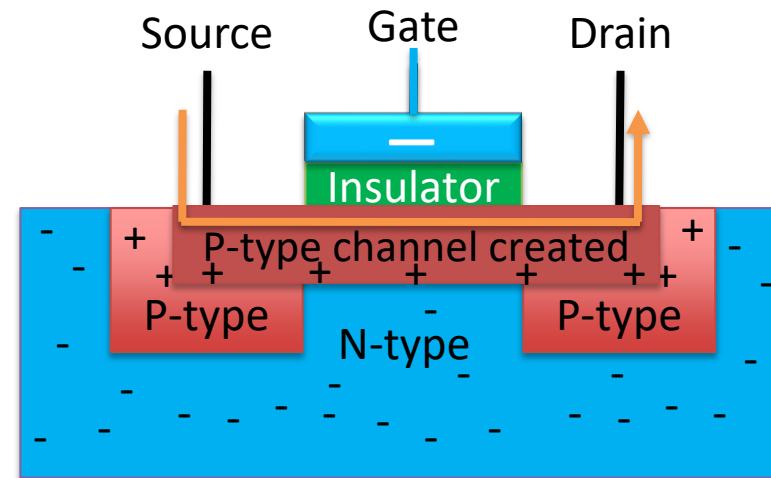
- Transistors:
- Youtube video “How does a transistor work”
<https://www.youtube.com/watch?v=IcrBqCFLHIY>
- Break: show some Transistor, Fab, Wafer photos



Transistors 101



P-Transistor Off



P-Transistor On

N-Type Silicon: negative free-carriers (electrons)

P-Type Silicon: positive free-carriers (holes)

P-Transistor: negative charge on gate generates electric field that creates a (+ charged) p-channel connecting source & drain

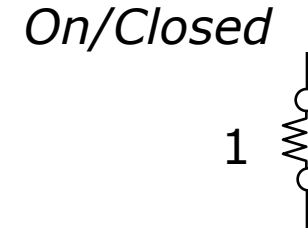
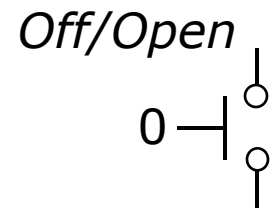
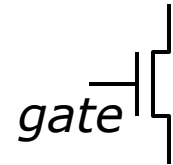
N-Transistor: works the opposite way

Metal-Oxide Semiconductor (Gate-Insulator-Silicon)

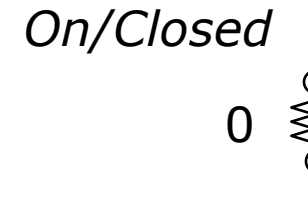
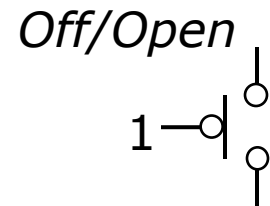
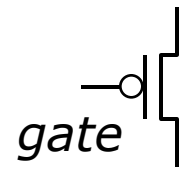
- Complementary MOS = **CMOS** technology uses both p- & n-type transistors

CMOS Notation

N-type



P-type



Gate input controls whether current can flow between the other two terminals or not.

Hint: the “o” bubble of the p-type tells you that this gate wants a 0 to be turned on

PolEV Question #6

Which of the following statements is false?

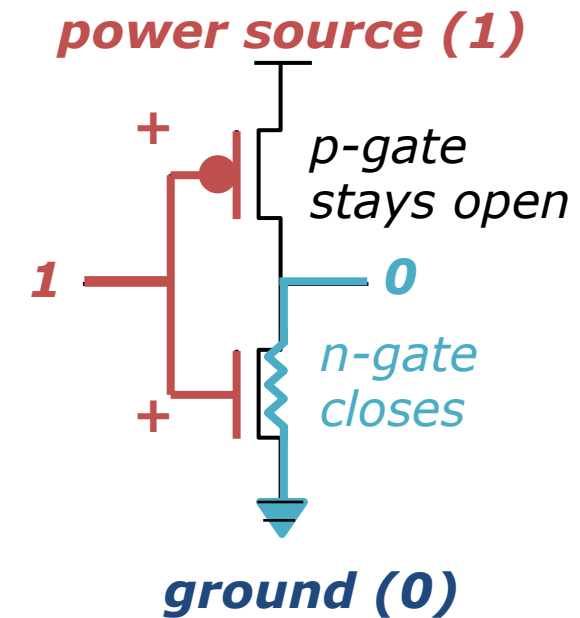
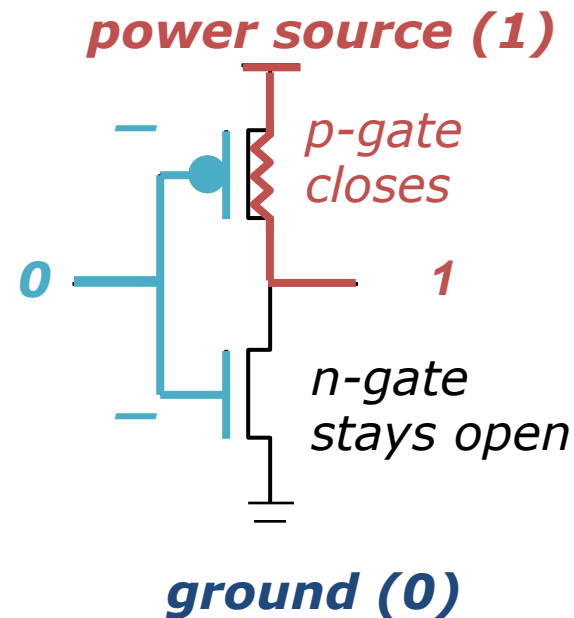
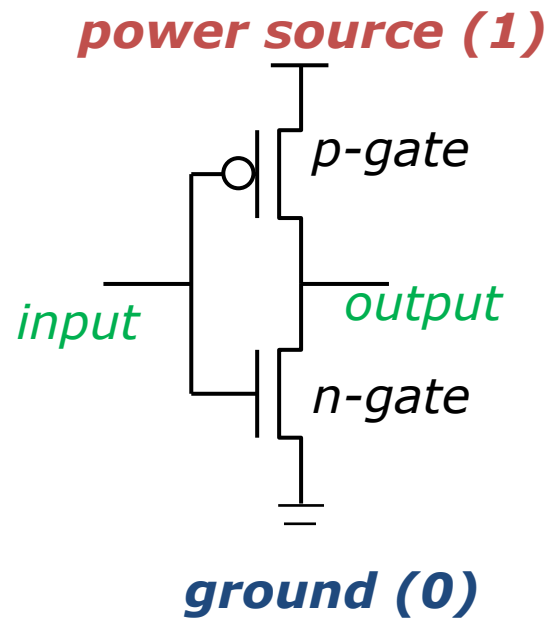
- (A) P- and N-type transistors are both used in CMOS designs
- (B) As transistors get smaller, the frequency of your processor will keep getting faster
- (C) As transistors get smaller, you can fit more and more of them on a single chip
- (D) Pure silicon is a semi conductor
- (E) Experts believe that Moore's Law will soon end



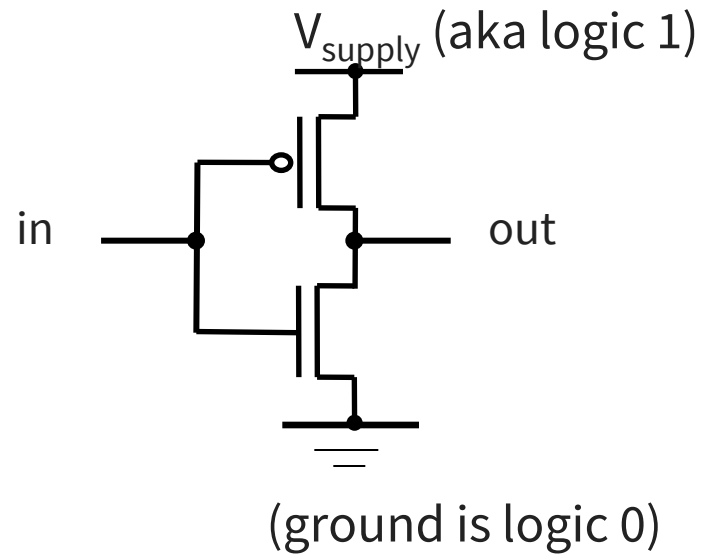
2-Transistor Combination: NOT

- Logic gates are constructed by combining transistors in complementary arrangements
- Combine p&n transistors to make a NOT gate:

CMOS Inverter :

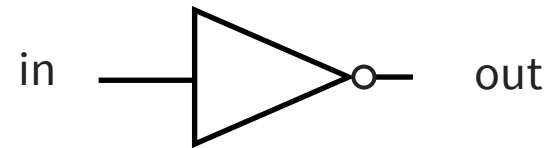


Inverter



Function: NOT

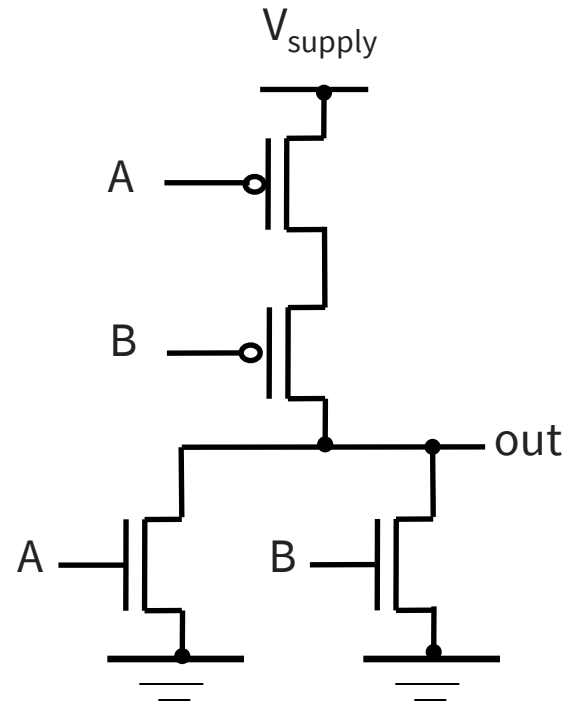
Symbol:



Truth Table:

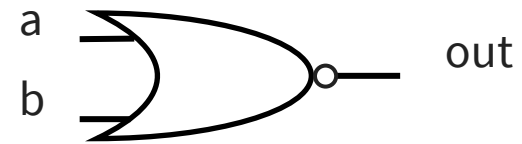
In	Out
0	1
1	0

NOR Gate



Function: NOR

Symbol:

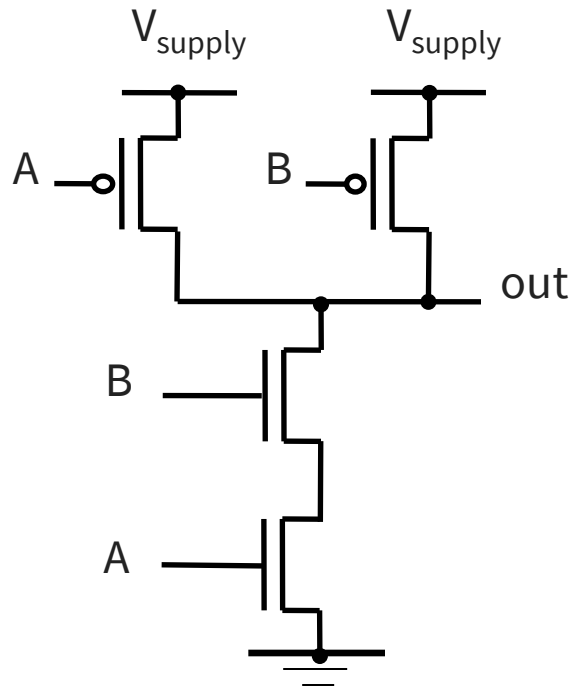


Truth Table:

A	B	out
0	0	1
0	1	0
1	0	0
1	1	0

Which Gate is this?

PolEV Question #7








Function:

Symbol:

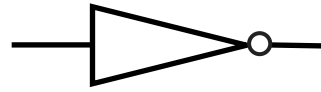
Truth Table:

A	B	out
0	0	
0	1	
1	0	
1	1	

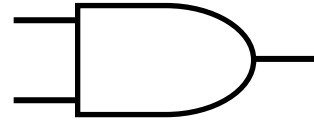
- (A) NOT 
- (B) OR 
- (C) XOR 
- (D) AND 
- (E) NAND 

Building Functions (Revisited)

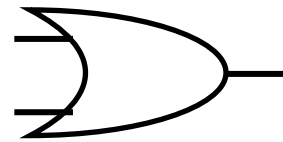
- NOT:



- AND:



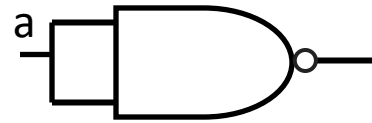
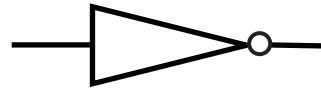
- OR:



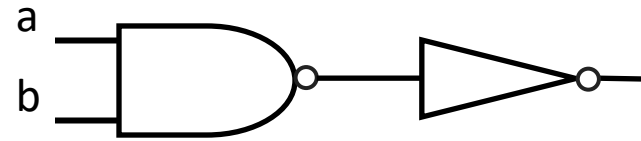
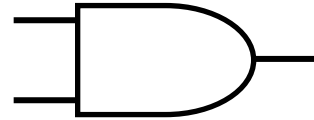
- NAND and NOR are universal
 - Can implement *any* function with NAND or just NOR gates
 - useful for manufacturing

Building Functions (Revisited)

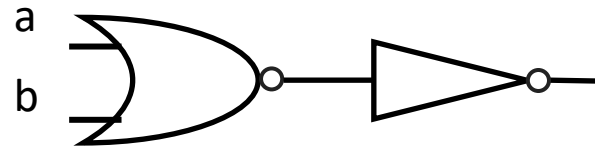
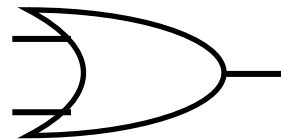
- NOT:



- AND:

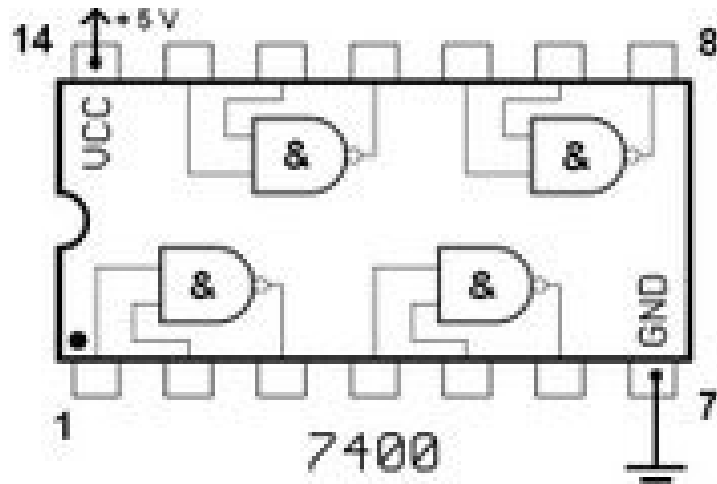


- OR:



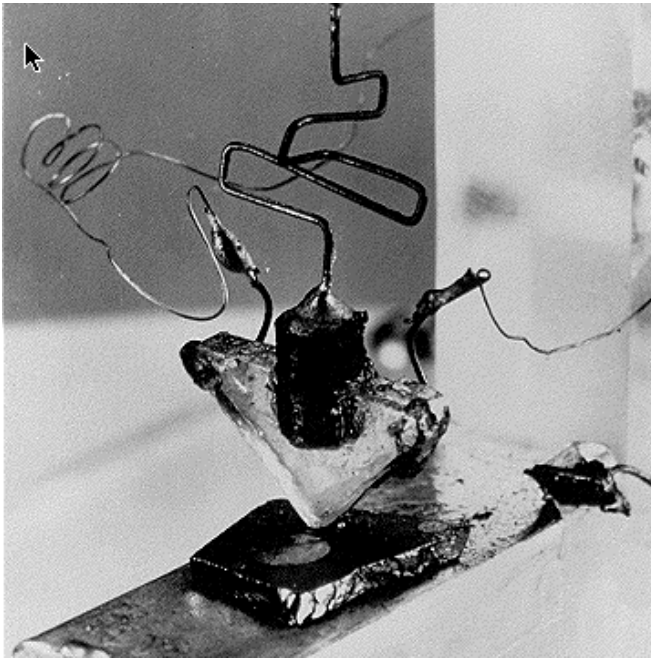
- NAND and NOR are universal
 - Can implement *any* function with NAND or just NOR gates
 - useful for manufacturing

Logic Gates



- One can buy gates separately
 - ex. 74xxx series of integrated circuits
 - cost ~\$1 per chip, mostly for packaging and testing
- Cumbersome, but possible to build devices using gates put together manually

Then and Now



4 performance cores
Improved branch prediction
Wider decode and execution engines
Next-generation ML accelerators

6 efficiency cores

Neural Engine
16-core design
Faster and more efficient

10-core GPU
Next-generation architecture
Dynamic Caching
Mesh shading
Ray tracing

Display engine
Tandem OLED support
Brightness and color compensation
10Hz-120Hz ProMotion support

https://en.wikipedia.org/wiki/Apple_M4

https://en.wikipedia.org/wiki/Transistor_count

The first transistor

- One workbench at AT&T Bell Labs
- 1947
- Bardeen, Brattain, and Shockley

Apple M4

- 28 billion transistors, 3nm
- 177 square millimeters
- 4x-10x performance, 4x-6x efficiency, 8x-40x GPU, 16x Neural processing cores



Big Picture: Abstraction

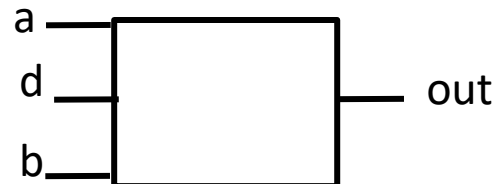
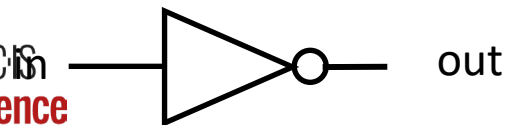
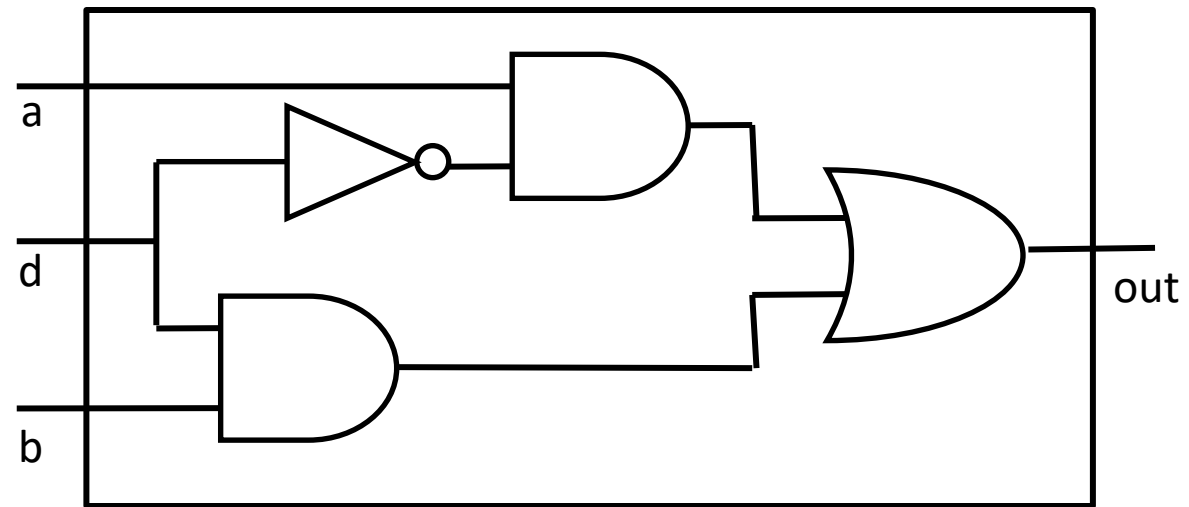
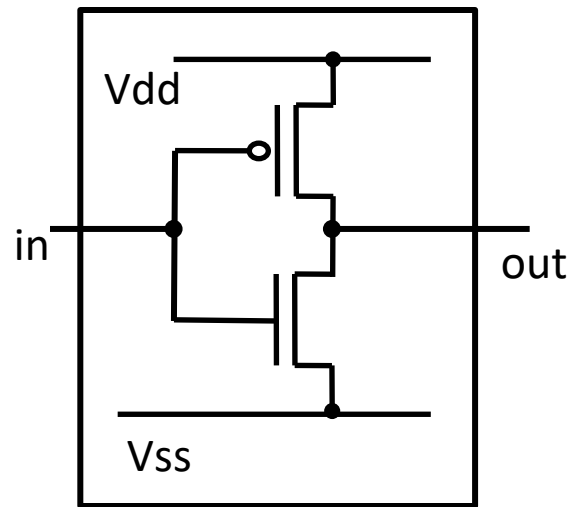
- Hide complexity through simple abstractions

- **Simplicity**

- Box diagram represents inputs and outputs

- **Complexity**

- Hides underlying NMOS- and PMOS-transistors and atomic interactions



Summary

- Most modern devices made of billions of transistors
 - You will build a processor in this course!
 - Modern transistors made from semiconductor materials
 - Transistors used to make logic gates and logic circuits
- We can now implement any logic circuit
 - Use P- & N-transistors to implement NAND/NOR gates
 - Use NAND or NOR gates to implement the logic circuit
 - *Efficiently*: use K-maps to find required minimal terms