CS 61C:
Great Ideas in Computer Architecture

Lecture 9: *Combinatorial Logic*

Bernhard Boser & Randy Katz

http://inst.eecs.berkeley.edu/~cs61c

Midterm Tuesday

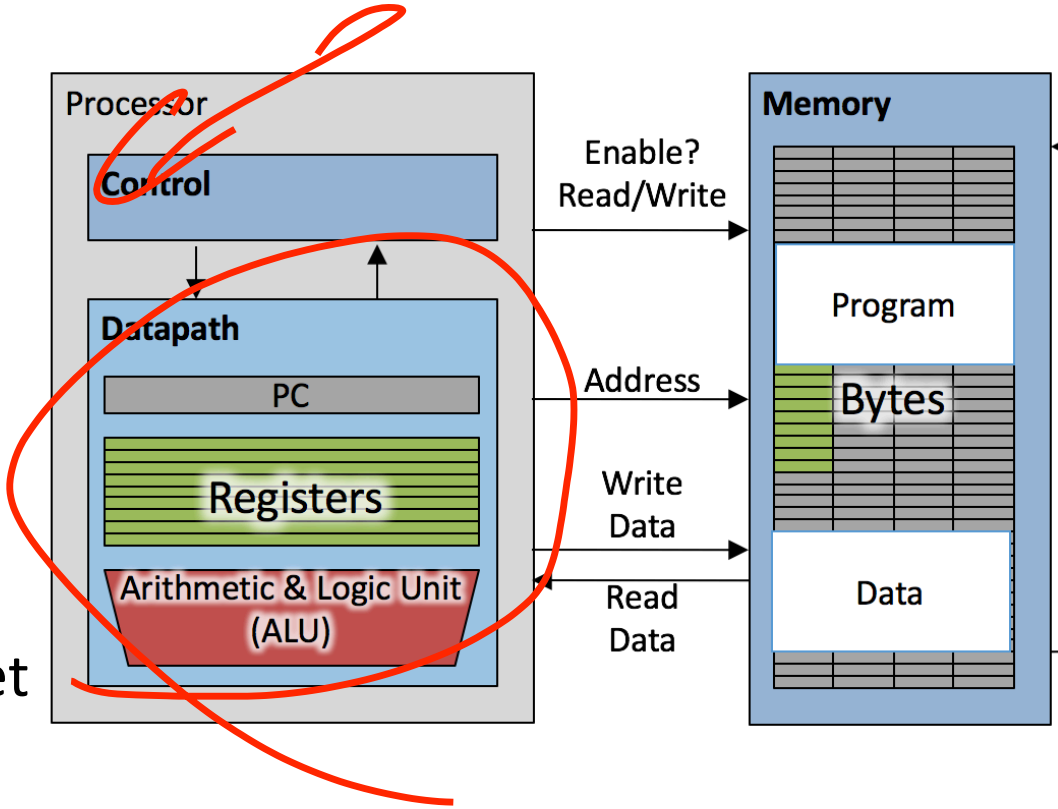
Date: Tuesday 9/27/2016
Time: 3:30 – 5pm
Location:

Login	Room
cs61c-aab to cs61c-aaz	Dwinelle 109
cs61c-aba to cs61c-abt	Dwinelle 179
cs61c-abu to cs61c-acs	Dwinelle 229
cs61c-act to cs61c-adw	Etcheverry 3113
All others	Pauley Ballroom

"WITH THE PROPER TECHNOLOGY, WE COULD MAKE SILICON CHIPS OUT OF THIS SAND."

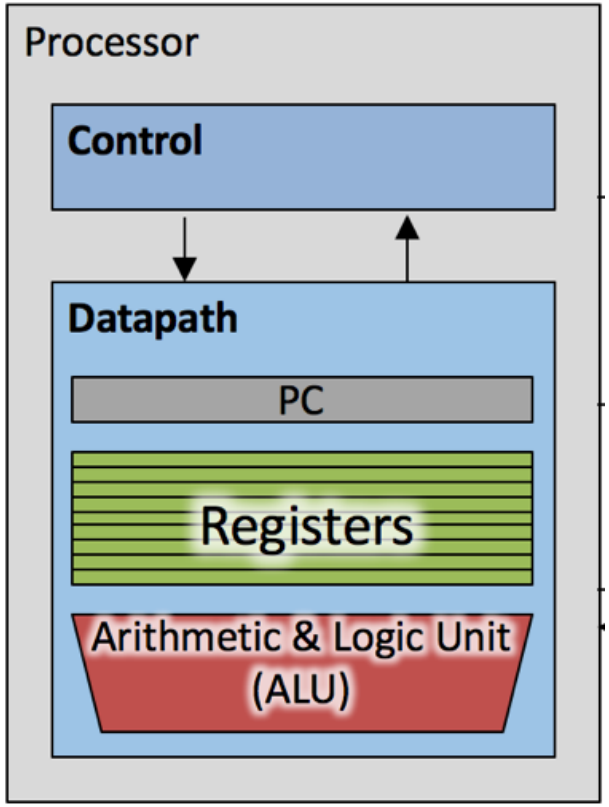
Processor Design in 5 Lectures

1. Combinatorial Logic
   - Logic, arithmetic
   - Hardware!
2. Sequential Logic
   - Memory
   - Time: do this, then that
3. Datapath
   - Implement instruction set
4. Control
   - Telling each piece what to do
5. Pipelining
   - Make it go fast!

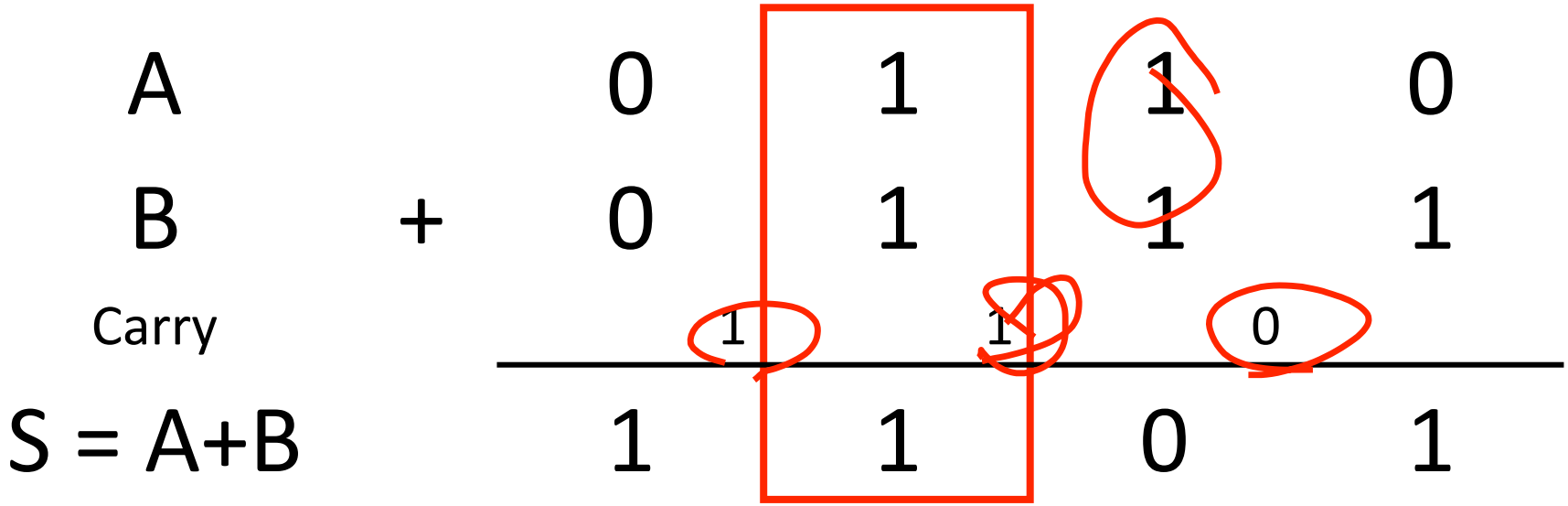


Today's Agenda

- **1-Bit Adder**
- Boolean Algebra
- Hardware: Chips, transistors, volts!
- CMOS Logic Gates
- ALU



Binary Addition



Binary addition is just like decimal addition

Binary Addition

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

1-Bit Adder "Slice" Truth Table:

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

Binary Addition

A		0	1	1	0
B	+	0	1	1	1
S		1	1	0	1

1-Bit Adder "Slice" Truth Table:

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

Binary Addition

1-Bit Adder "Slice" Truth Table:

A	B	C_{in}	S	C_{out}
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

1-Bit Adder Logic Synthesis

Truth Table

A	B	C_{in}	S	C_{out}
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

Boolean Expression

S =

(not A) and (not B) and (C_{in})
or (not A) and (B) and (not C_{in})
or (A) and (not B) and (not C_{in})
or (A) and (B) and (C_{in})

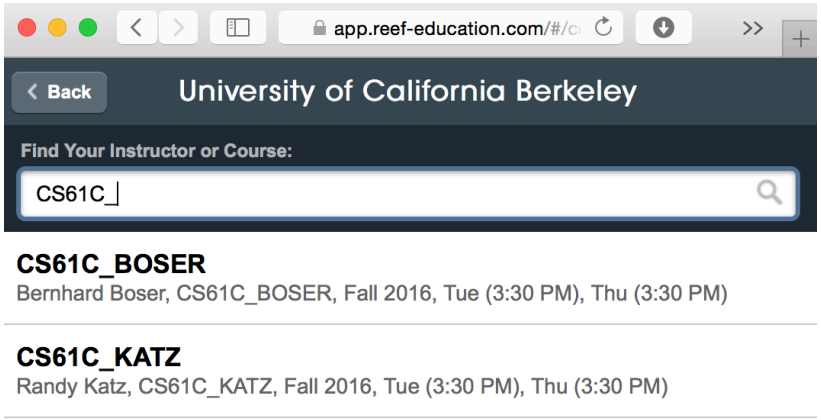
Compact Notation

$$S = A \cdot B \cdot C\downarrow in + A \cdot B \cdot C\downarrow in + A \cdot B \cdot C\downarrow in + A \cdot B \cdot C\downarrow in$$

Same procedure for **C_{out}** …

iClicker Registration

- Physical i>Clicker
  - https://www.ets.berkeley.edu/discover-services/clickers/students-getting-started
  - Register at https://bcourses.berkeley.edu, **not** REEF website!
- REFF
  - Register for *CS61C_BOSER* **and** *CS61C_KATZ* at https://reef-education.com



  - **Important**: use same ***name and SID*** for REEF registration and bcourses
    - Otherwise your scores won't be registered on bcourses

Your Turn

Truth Table

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

Which Boolean expression represents the truth table?

Answer	Boolean Expression
A	$Y=A \cdot B+A \cdot B+A \cdot B$
B	$Y=A \cdot B+A \cdot B$
C	$Y=A \cdot B+A \cdot B$
D	$Y=A \cdot B+A \cdot B$
E	$Y=A \cdot B+A \cdot B+A \cdot B$

Your Turn

Truth Table

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

Which Boolean expression represents the truth table?

Answer	Boolean Expression
A	$Y=A\cdot B+A\cdot B+A\cdot B$
B	$Y=A\cdot B+A\cdot B$
C	$Y=A\cdot B+A\cdot B$
D	$Y=\overline{A}\cdot B+A\cdot\overline{B}$
E	$Y=A\cdot B+A\cdot B+A\cdot B$

Logic Representations

Example: Exclusive OR "XOR"

Truth Table

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

Boolean Expression

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

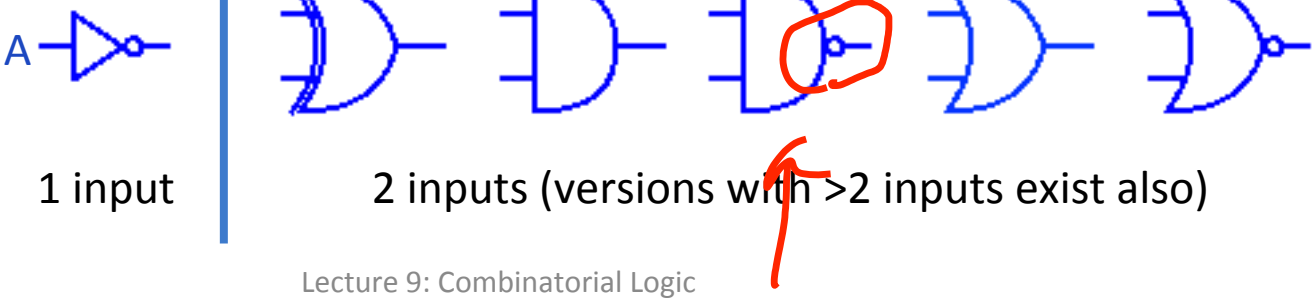
Symbol

Three different representations of the same thing.

1 & 2 Input Logic Gates

Inputs		Gate					
A	B	NOT A	XOR	AND	NAND	OR	NOR
0	0	1	0	0	1	0	1
0	1	1	1	0	1	1	0
1	0	0	1	0	1	1	0
1	1	0	0	1	0	1	0
Boolean expression:		$\overline{A}$	A⊕B	A·B	$\overline{A \cdot B}$	A+B	$\overline{A+B}$

Symbol:



1 input

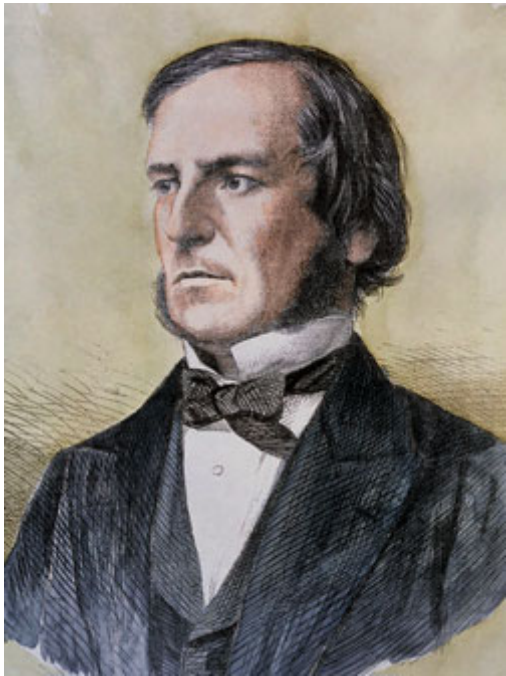
2 inputs (versions with >2 inputs exist also)

Today's Agenda

- 1-Bit Adder
- **Boolean Algebra**
- Hardware: Chips, transistors, volts!
- CMOS Logic Gates
- ALU

Boolean Algebra

- Early computer designers built ad hoc circuits
- Began to notice common patterns in their work: NOTs, ANDs, ORs, XORs, …
- Master's thesis (by Claude Shannon, 1940) made connection to work of 19th Century Mathematician George Boole
  - Called it "Boolean Algebra" in his honor
- Apply formal math to hardware design, logic minimization, …
- Many techniques, we only look at the most straightforward approach



Laws of Boolean Algebra

$X \overline{X} = 0$	$X + \overline{X} = 1$	Complementarity
$X\ 0 = 0$	$X + 1 = 1$	Laws of 0's and 1's
$X\ 1 = X$	$X + 0 = X$	Identities
$X\ X = X$	$X + X = X$	Idempotent Laws
$X\ Y = Y\ X$	$X + Y = Y + X$	Commutativity
$(X\ Y)\ Z = X\ (Y\ Z)$	$(X + Y) + Z = Z + (Y + Z)$	Associativity
$X\ (Y + Z) = X\ Y + X\ Z$	$X + Y\ Z = (X + Y)\ (X + Z)$	Distribution
$X\ Y + X = X$	$(X + Y)\ X = X$	Uniting Theorem
$\overline{X}\ Y + X = X + Y$	$(\overline{X} + Y)\ X = X\ Y$	Uniting Theorem v. 2
$\overline{X\ Y} = \overline{X} + \overline{Y}$	$\overline{X + Y} = \overline{X}\ \overline{Y}$	DeMorgan's Law

Boolean Algebraic Simplification Example

$$y = ab + a + c$$

Fun Facts

- DeMorgan's Law:

$A \cdot B = A + B$

- AND and OR are interchangeable:
  - E.g. realize AND with OR (and a few inverters, NOT's)

- Cray 1 Super computer
  - Built entirely from NOR gates (no ANDs) and memory chips
  - More facts:
    - 160 Mflops, 8 Mbytes memory
    - 115 kW power, 5.5 tons
    - year: 1975
    - $8 Mio bucks

Compare to Bernhard's laptop:

- 8 Gbytes of memory (1000x)
- 16 Gflops (100x)
- 2015 (+ 40 years)
- $1600 (x 1/5000)

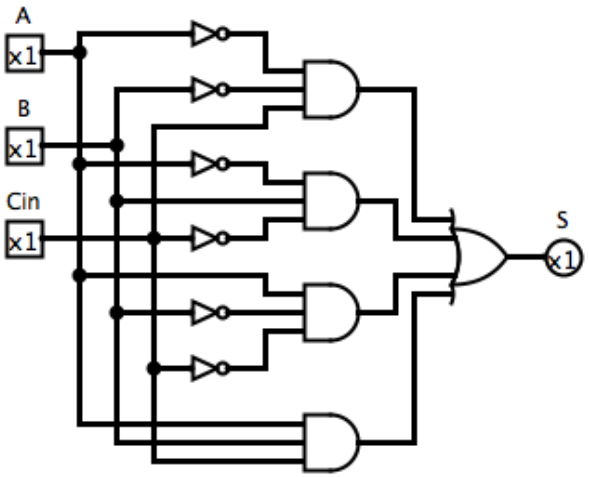
Cray-1 Super Computer

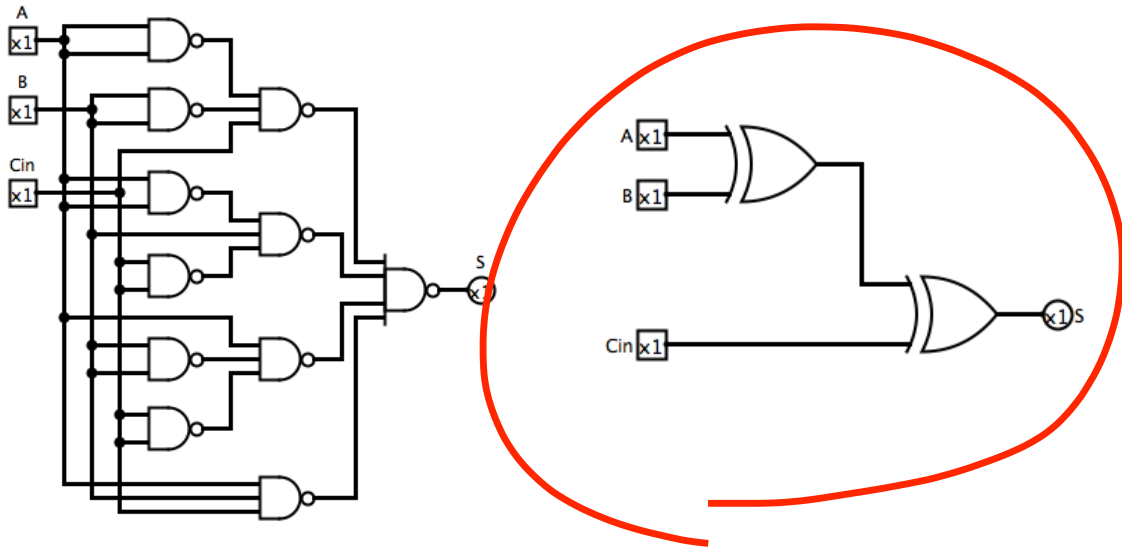
Why this "designer's" shape?

Logic Minimization (adder example)

$$S = A \cdot B \cdot C_{in} + A \cdot B \cdot C_{in} + A \cdot B \cdot C_{in} + A \cdot B \cdot C_{in}$$

Valid realizations (same truth table):





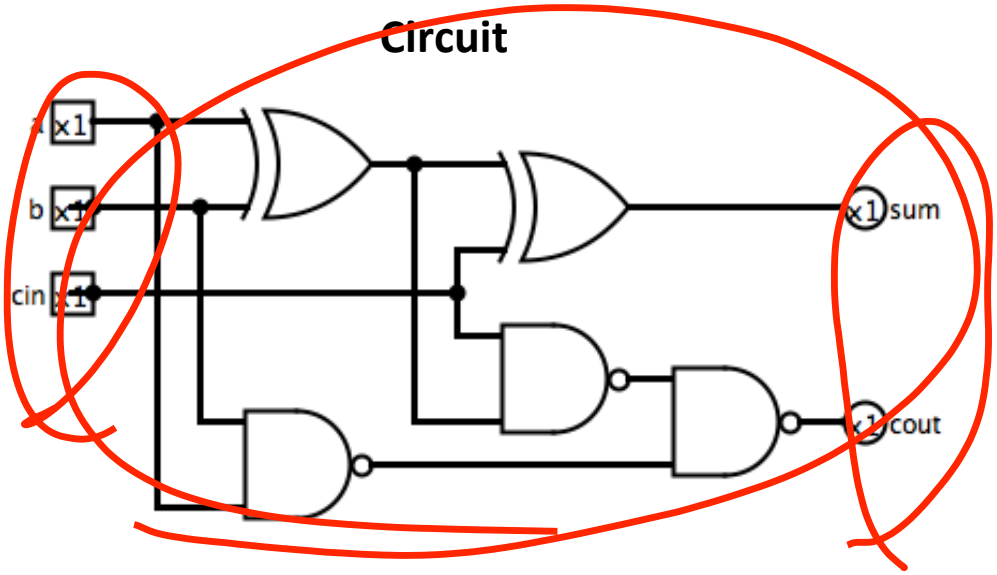
Which one is "best"?

1-Bit Adder

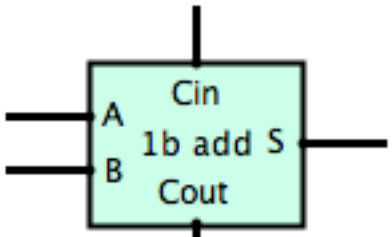
Truth table

A	B	C_{in}	Y	C_{out}
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

Circuit



Symbol



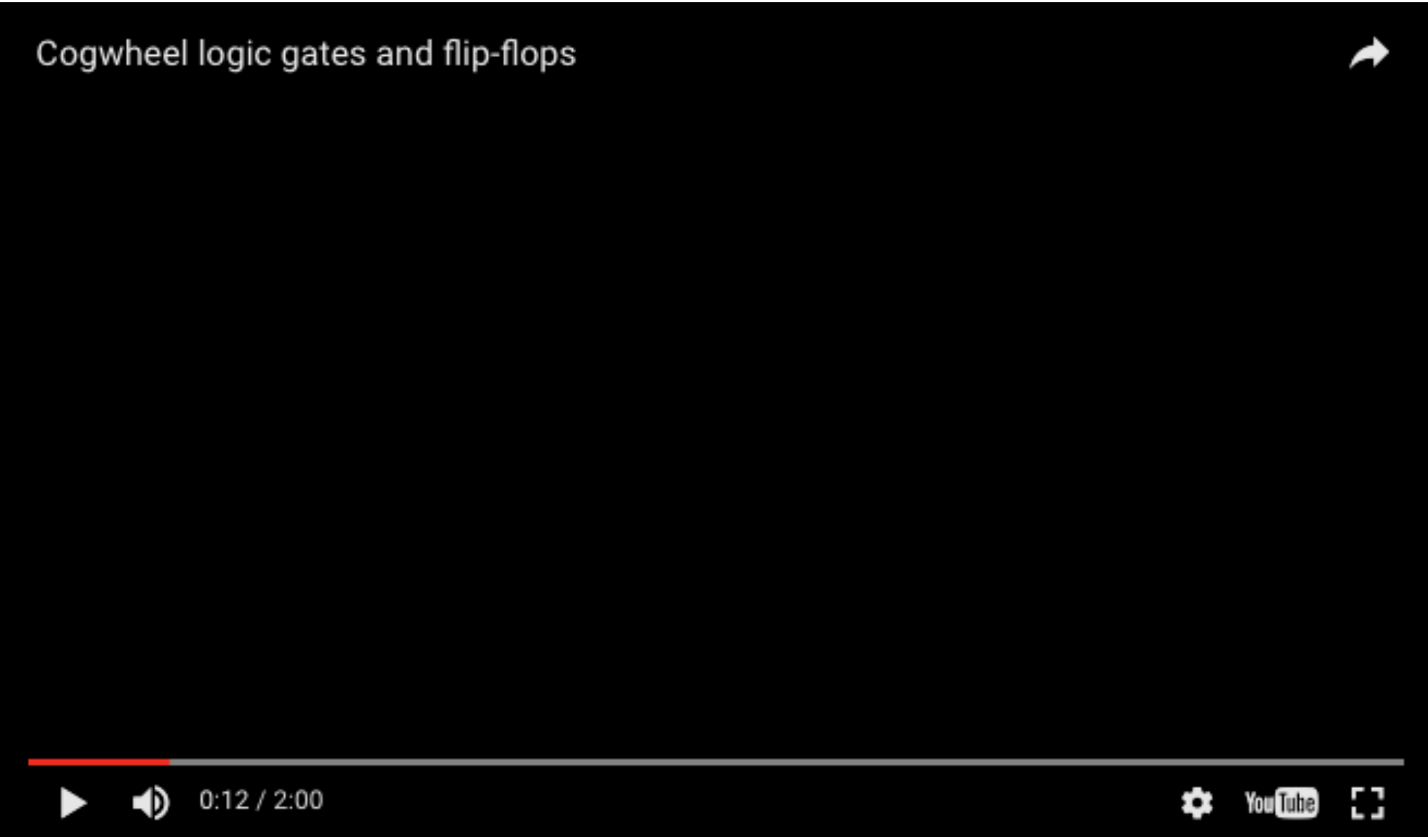
How do we make this (from SAND)?

Break!

Today's Agenda

- 1-Bit Adder
- Boolean Algebra
- **Hardware: Chips, transistors, Volts!**
- CMOS Logic Gates
- ALU

Software → Hardware

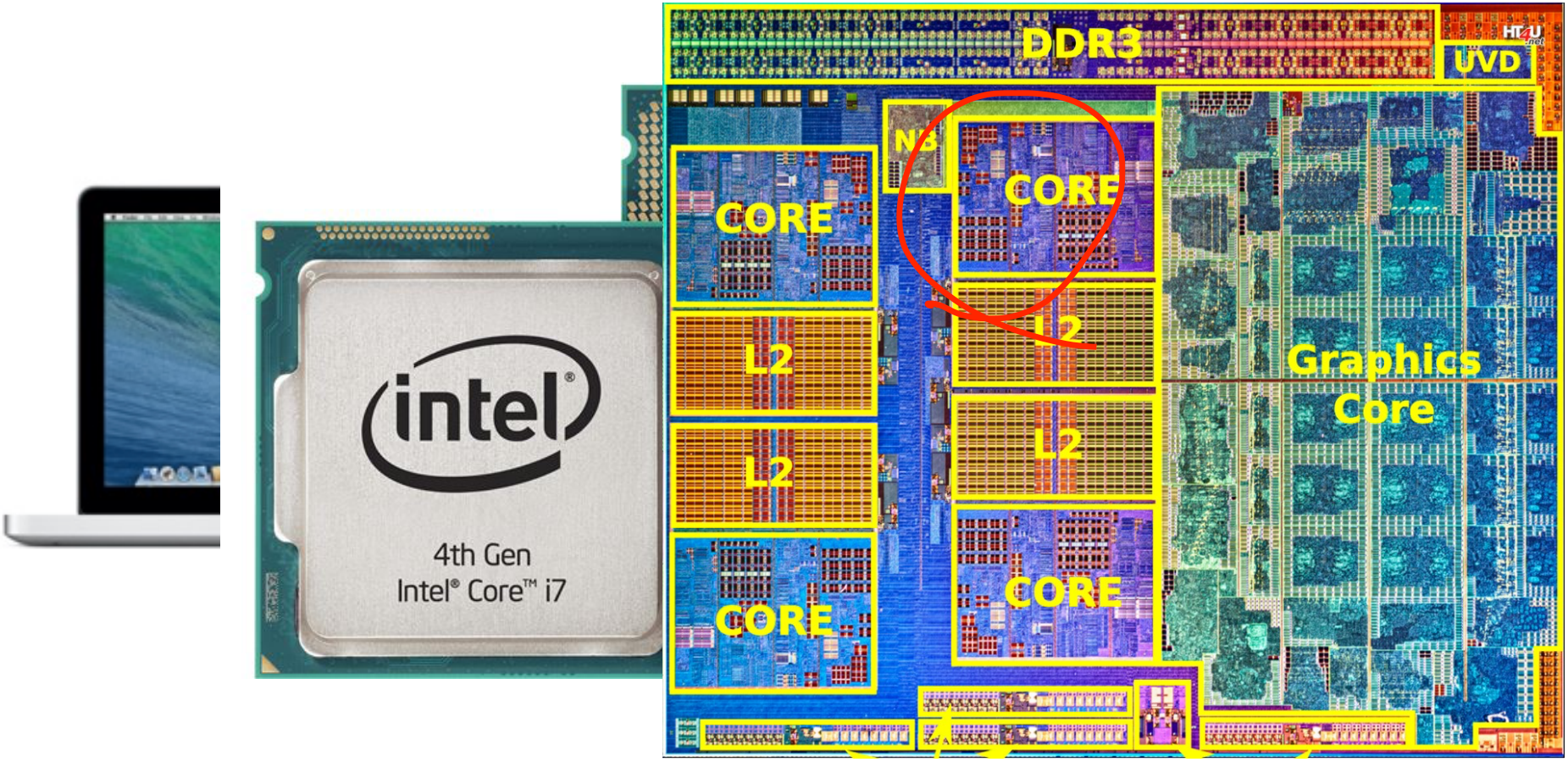


Logic Gates: Wish list

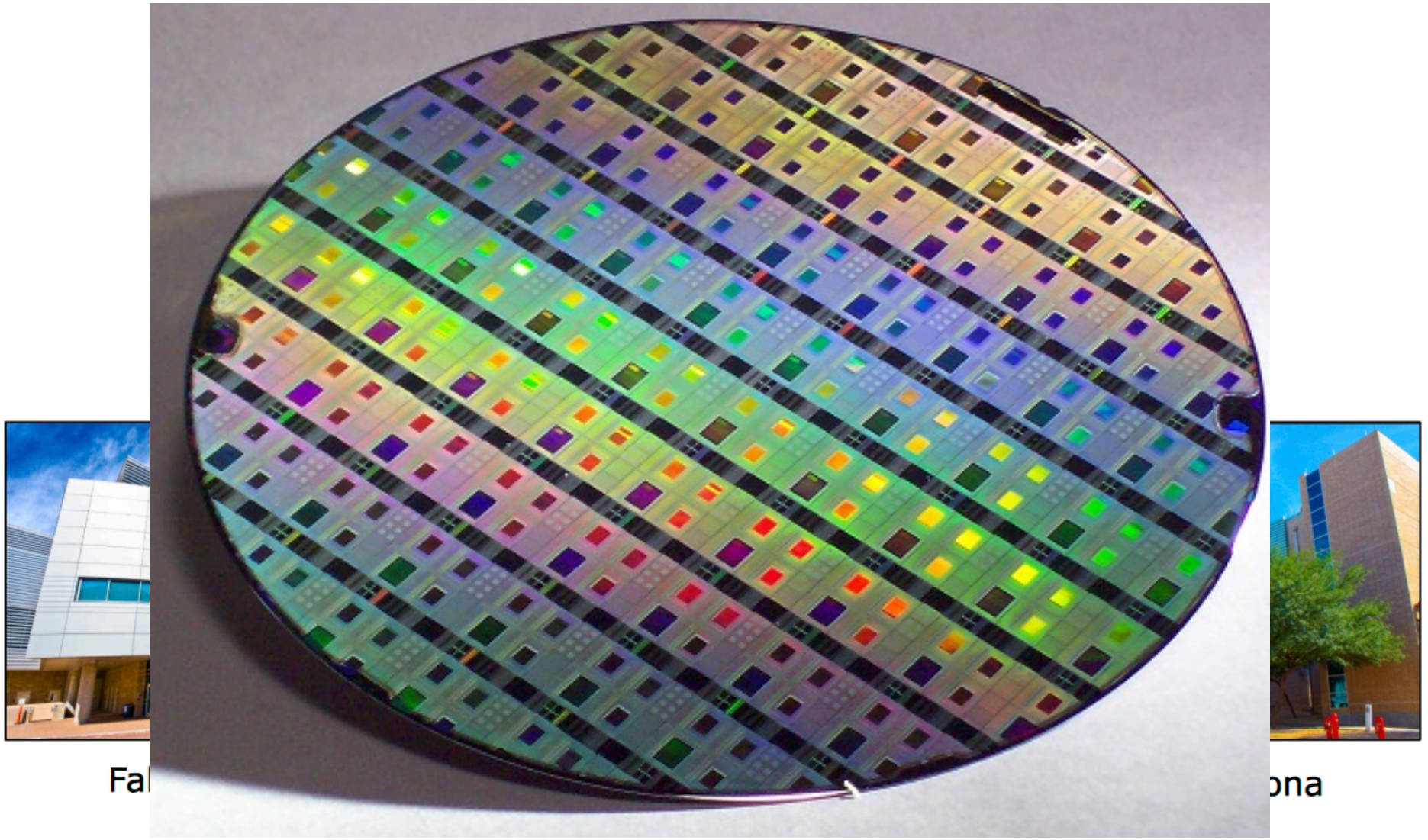
- Compute Boolean functions
- Input to output power gain
  - Gears do not: Cranking a chain of 100 cogs takes a lot of force (and may even break a gear)
- Restore logic value
  - What if a gear turns slowly or skips?
  - Is it a 1 or a 0?
  - Restore a clear 1 or 0 before ambiguity arises!
- Small, fast, inexpensive, reliable, low power, …

Electronics is currently the only technology that meets all these requirements

Logic in "Hardware"!



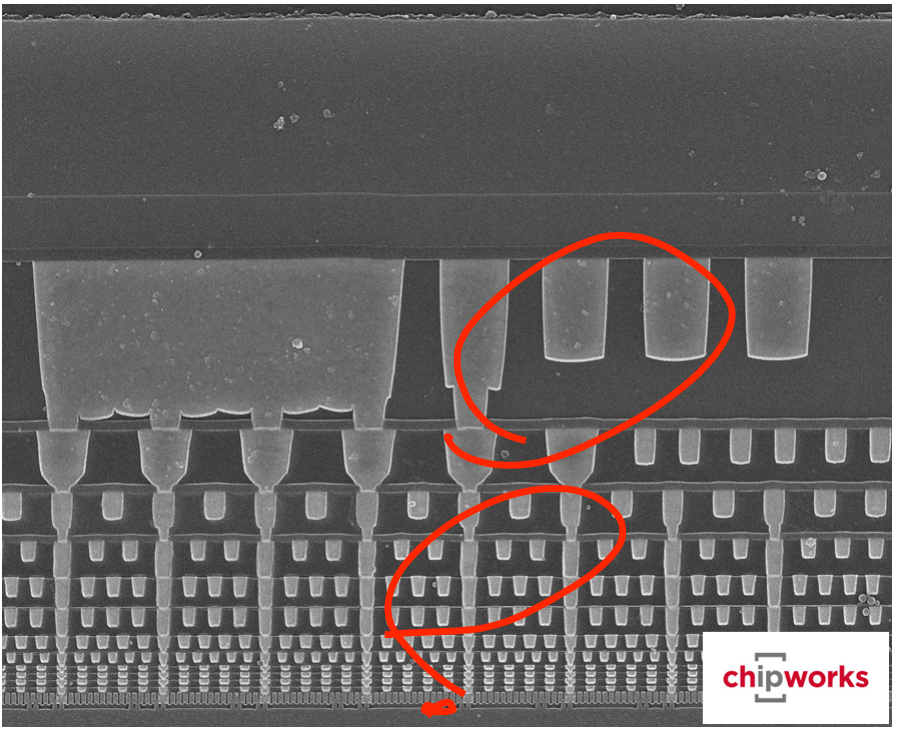
22 nm Manufacturing Fabs



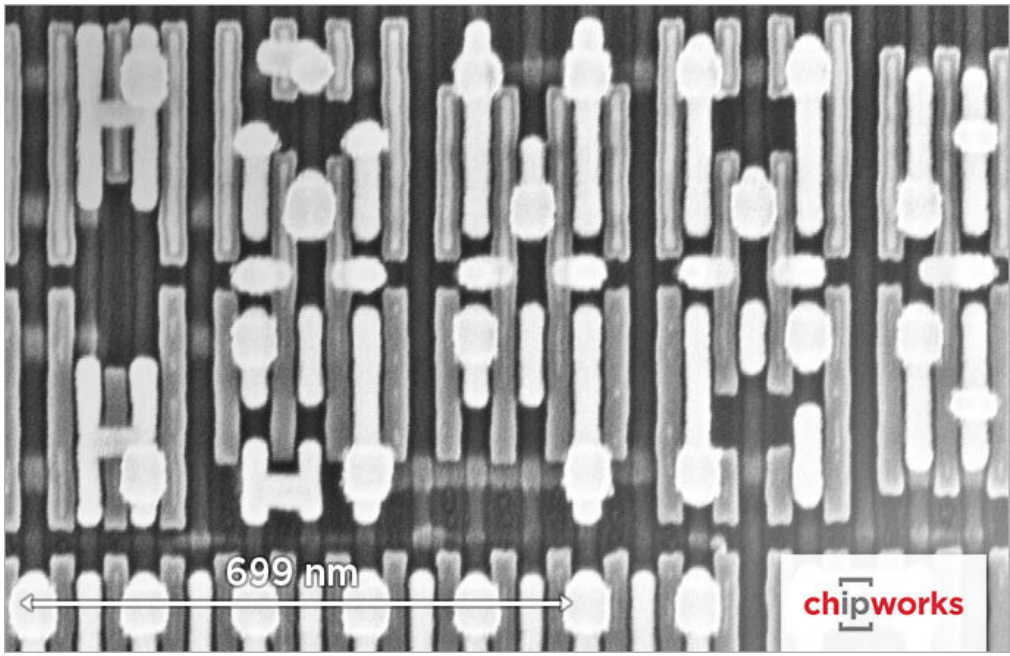
http://www.intel.com/content/dam/www/public/us/en/documents/presentation/silicon-technology-leadership-presentation.pdf

CMOS Technology

- CMOS "Complementary Metal Oxide Semiconductor"
  - Presently prevalent chip technology

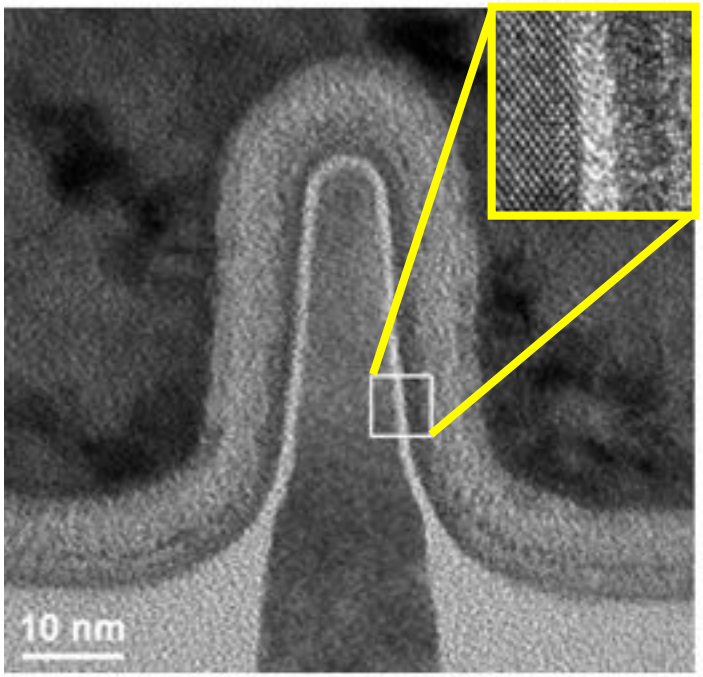


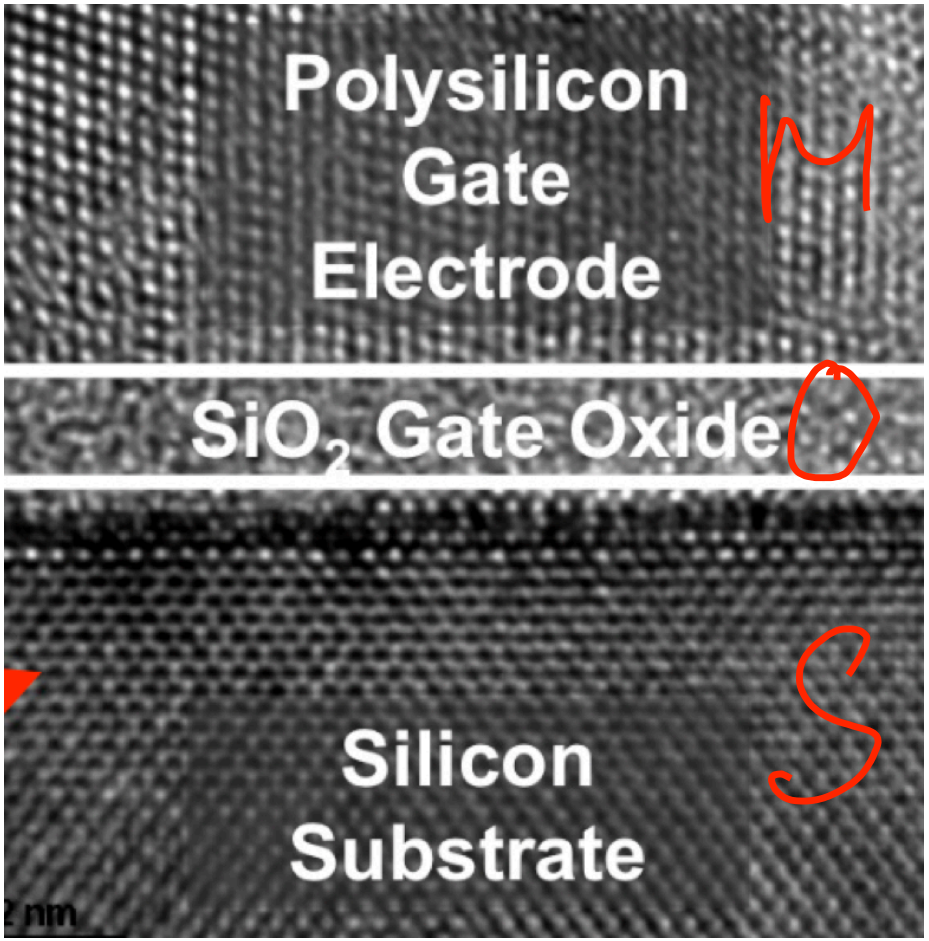
14nm CMOS cross section



14nm CMOS top view

14nm Transistor





Source: Intel press foils

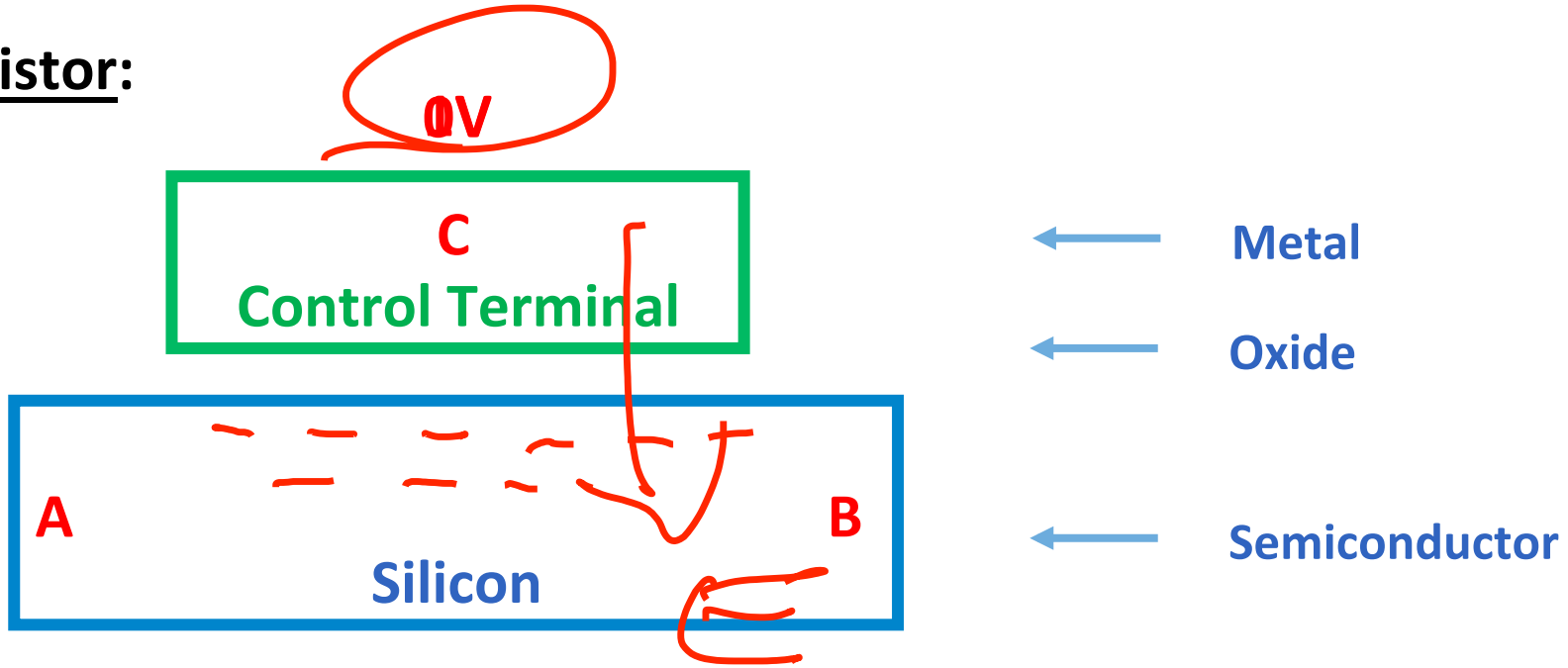
Transistor is < 100 atoms across!

Transistors for building Logic

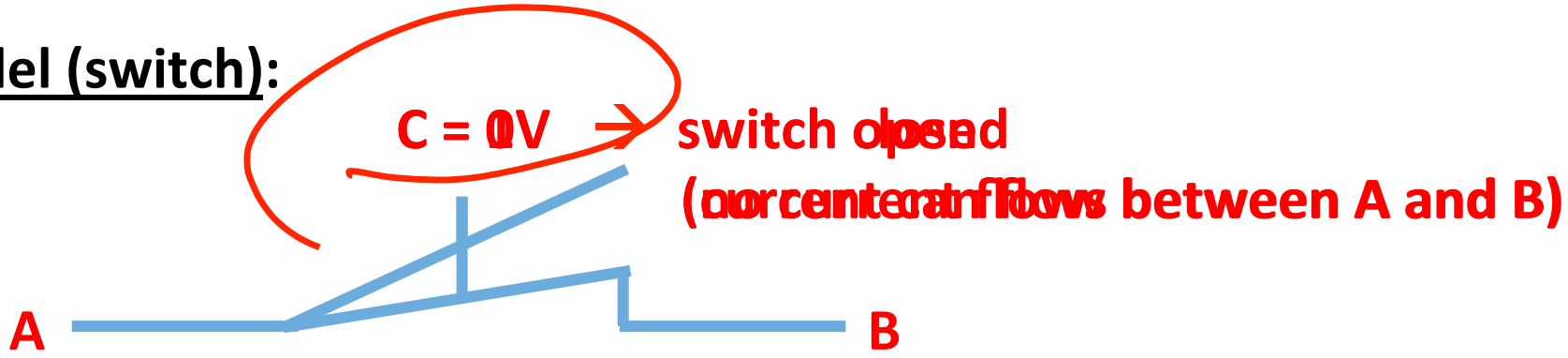
- To a computer, everything is 1's and 0's
- In electronic circuits we represent 1's and 0's with voltages. E.g.
  - 1 → 1V (5V works too, but consumes more power)
  - 0 → 0V
- Fault tolerance. E.g.
  - What is 0.7V? Closer to 1V than 0V → interpret as 1
  - Similar with 0.3V → 0V → 0
  - A great benefit of representing only two values, 0 and 1

MOS Transistor Operation

Transistor:



Model (switch):



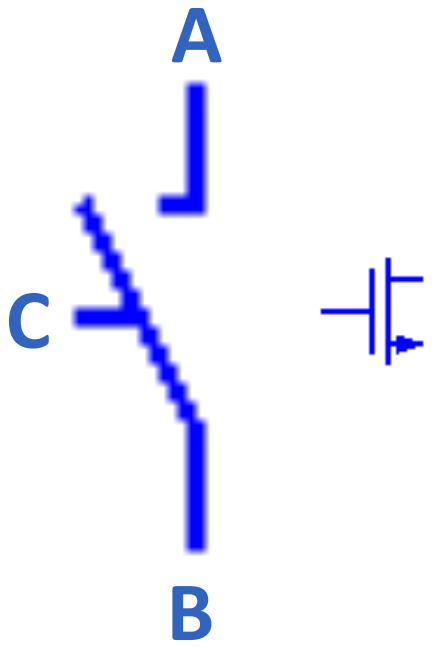
Yes, this is simplistic. If you cringe, you do not need this explanation. Stick with what you know already.

CMOS
Complementary Metal Oxide Semiconductor

NMOS Transistor (= Switch)

Switch closed for C = 1V (“1”)

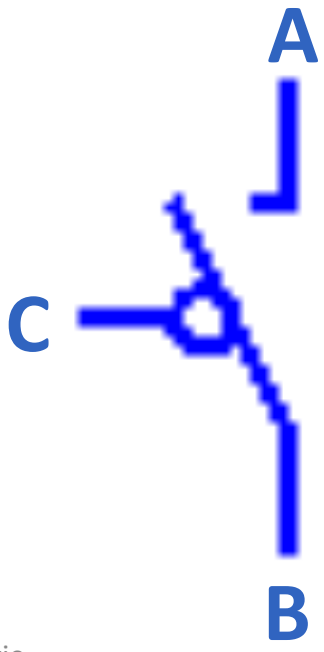
open for C = 0V



PMOS Transistor (= Switch)

Switch closed for C = 0V (“0”)

open for C = 1V



Today's Agenda

- 1-Bit Adder
- Boolean Algebra
- Hardware: Chips, transistors, volts!
- **CMOS Logic Gates**
- ALU

Logic NOT (Inverter)

Truth Table

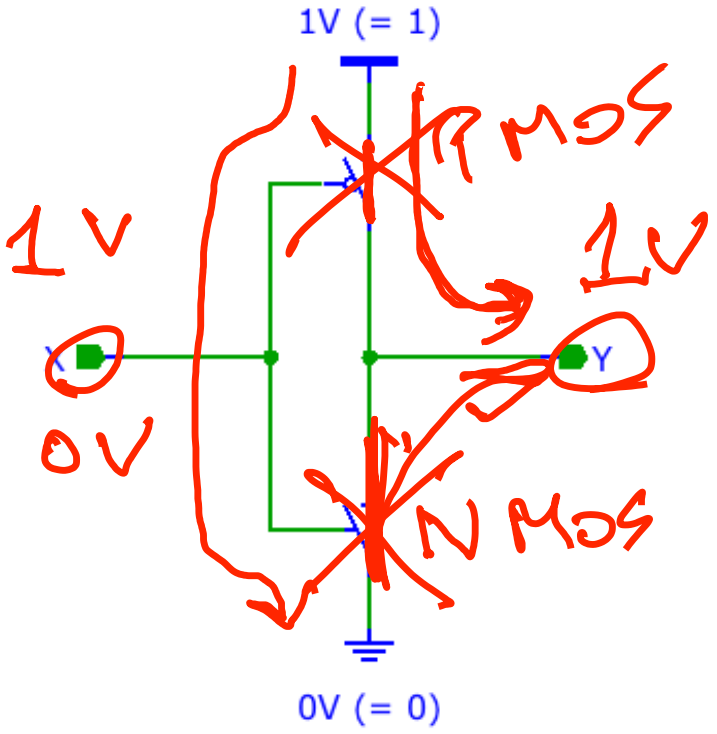
X (input)	Y (output)
0	1
1	0

Equation

$Y = X$

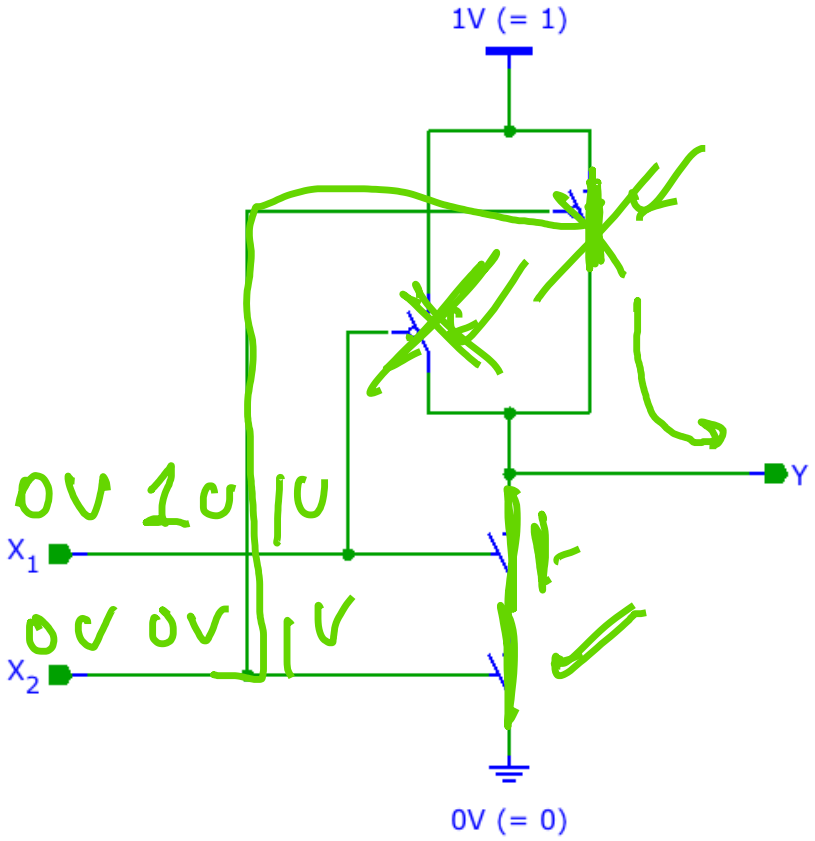
Symbol

Transistor Circuit



Logic NAND ("NOT – AND")

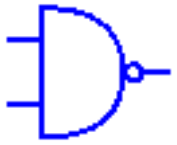
Transistor Circuit



Truth Table

X_1	X_2	Y
0	0	1
0	1	1
1	0	1
1	1	0

$$Y = X_1 \cdot X_2$$

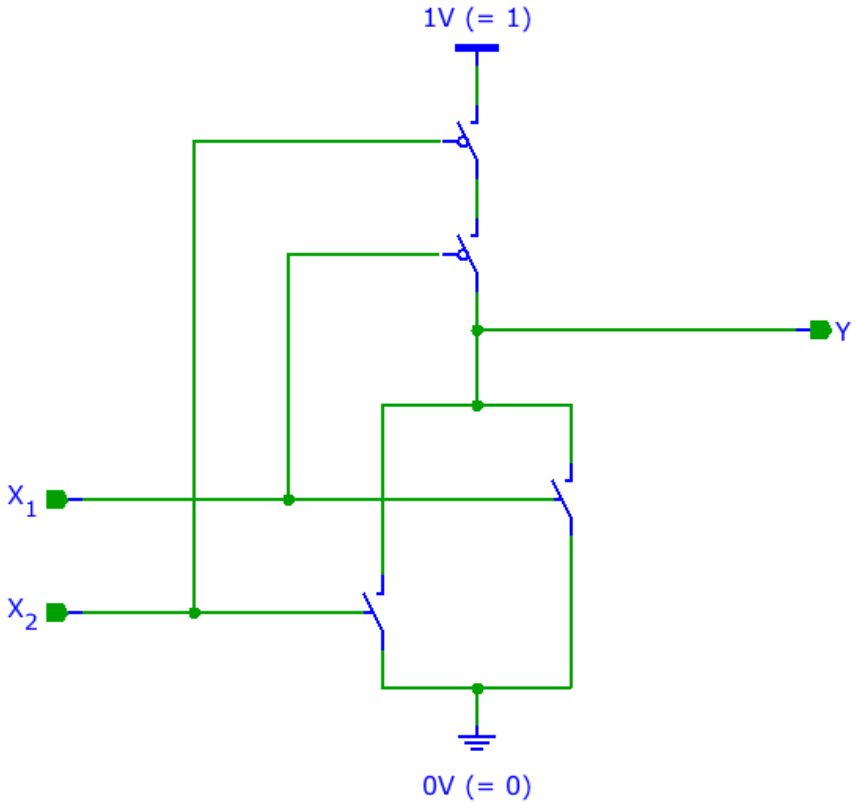


AND

- How do you realize logic “AND” in CMOS?

Your Turn

Transistor Circuit



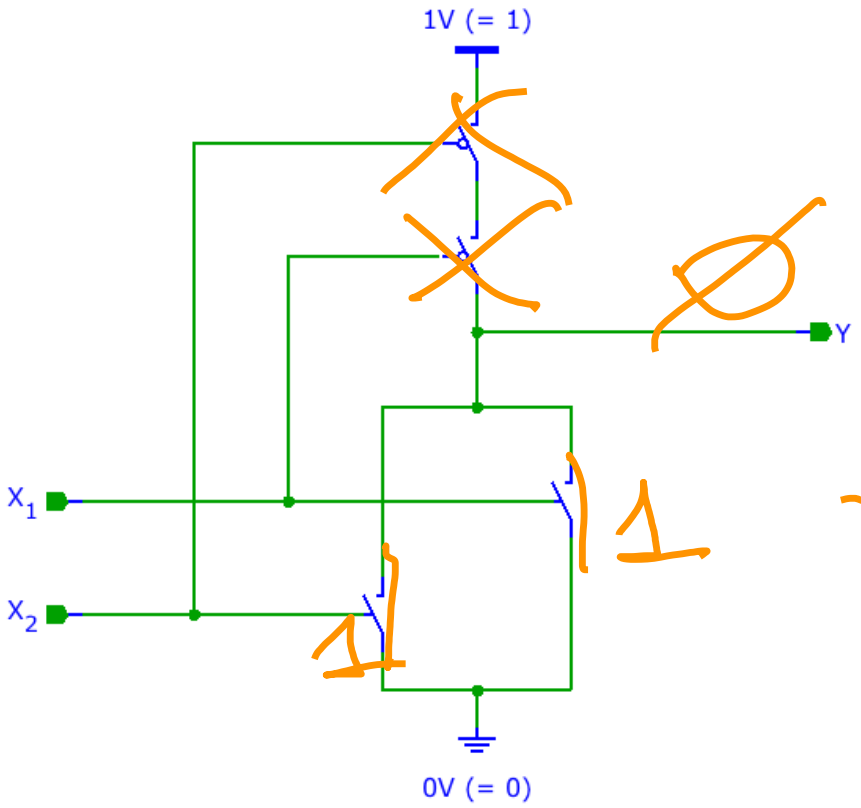
Truth Table

Which column (A, B, C, D, E) has the correct values for Y?

X_1	X_2	Y				
		A	B	C	D	E
0	0	0	1	1	0	1
0	1	1	0	1	1	0
1	0	1	0	0	0	1
1	1	1	0	1	1	1

Your Turn

Transistor Circuit



Truth Table

X_1	X_2	Y				
		A	B	C	D	E
0	0	0	1	1	0	1
0	1	1	0	1	1	0
1	0	1	0	0	0	1
1	1	1	0	1	1	1

NOR ("NOT OR")

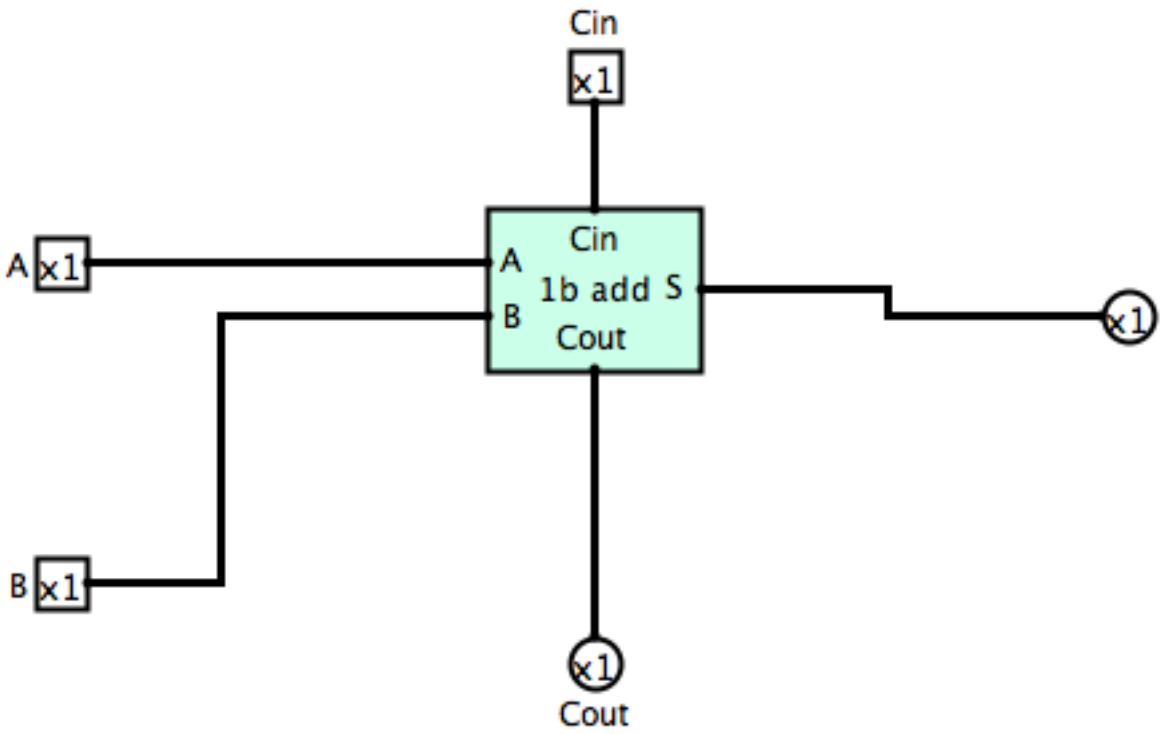
Today's Agenda

- 1-Bit Adder
- Boolean Algebra
- Hardware: Chips, transistors, volts!
- CMOS Logic Gates
- **ALU**

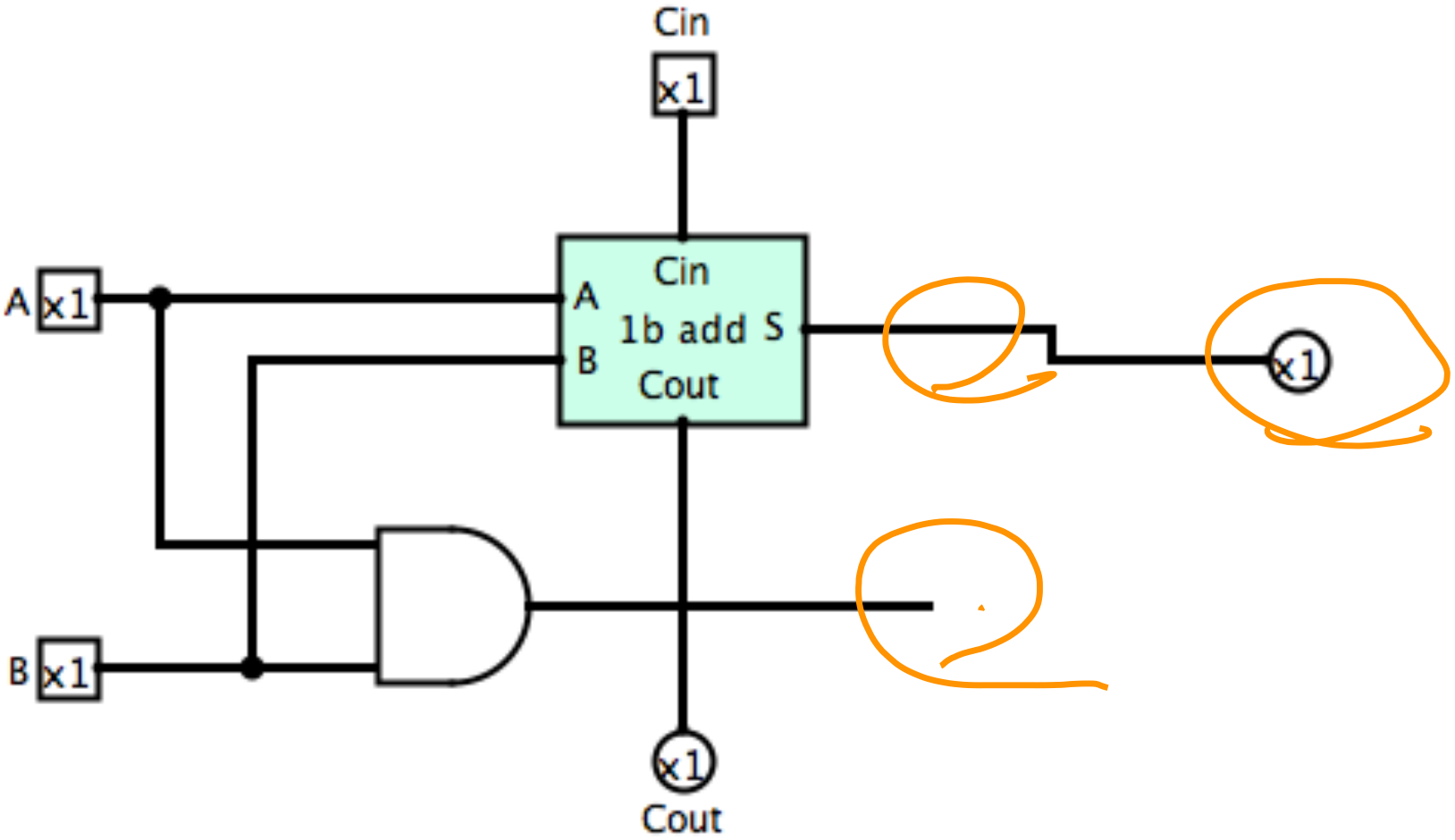
1-Bit "Mini" ALU Slice, Step 1

`add, and`

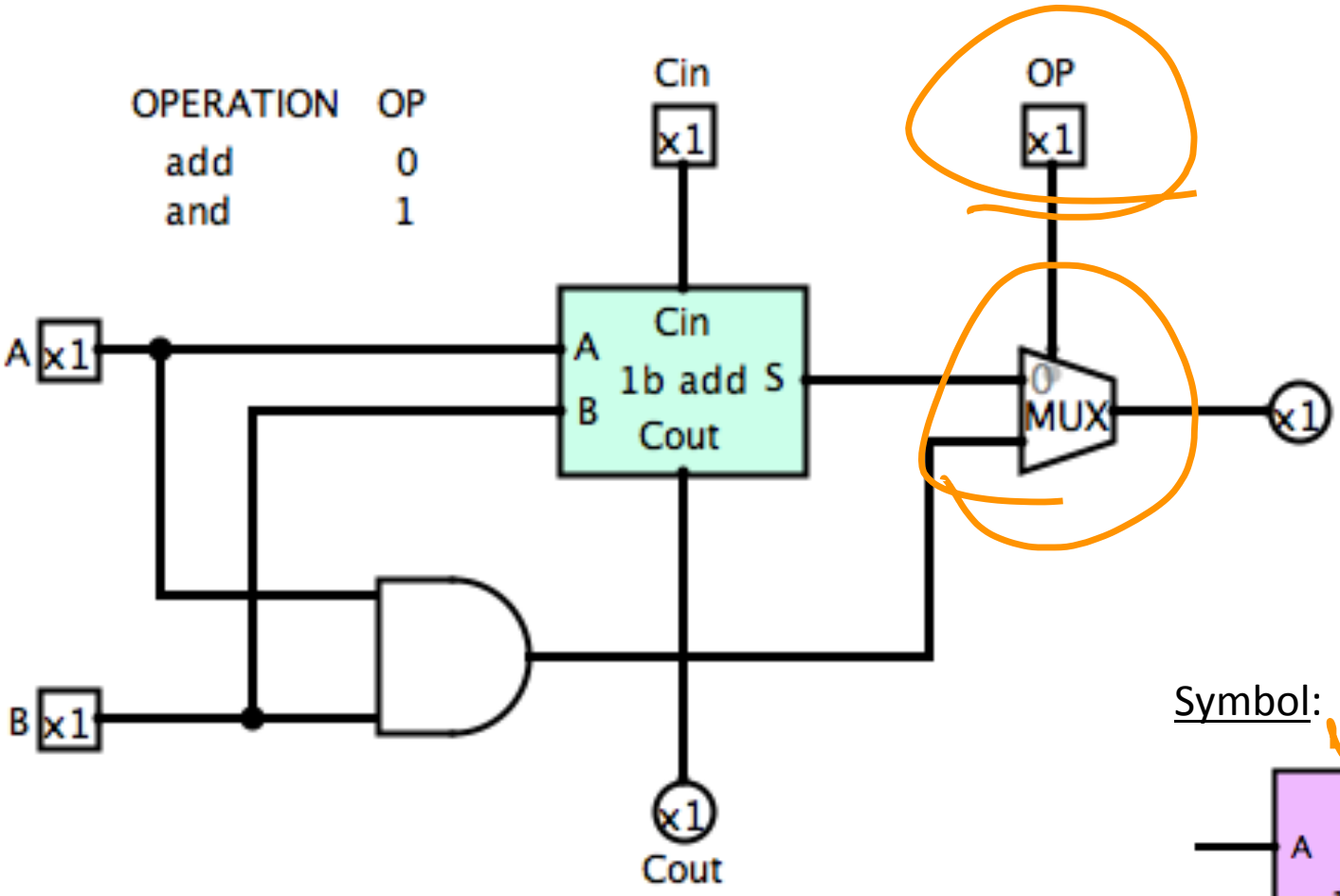
1-Bit “Mini” ALU Slice, Step 2



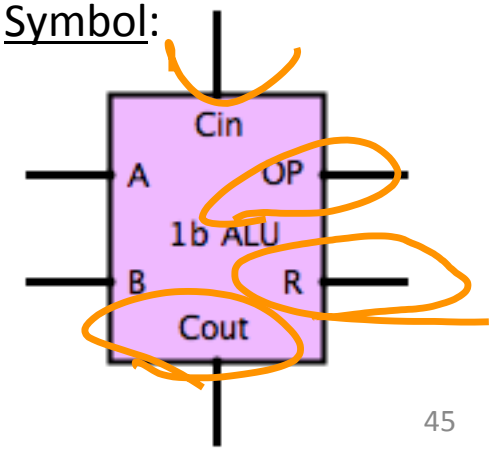
1-Bit "Mini" ALU Slice, Step 3



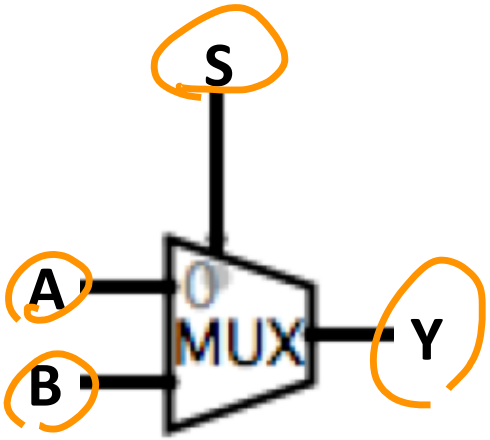
1-Bit "Mini" ALU Slice, Complete



Symbol:



1-Bit MUX (Multiplexor)



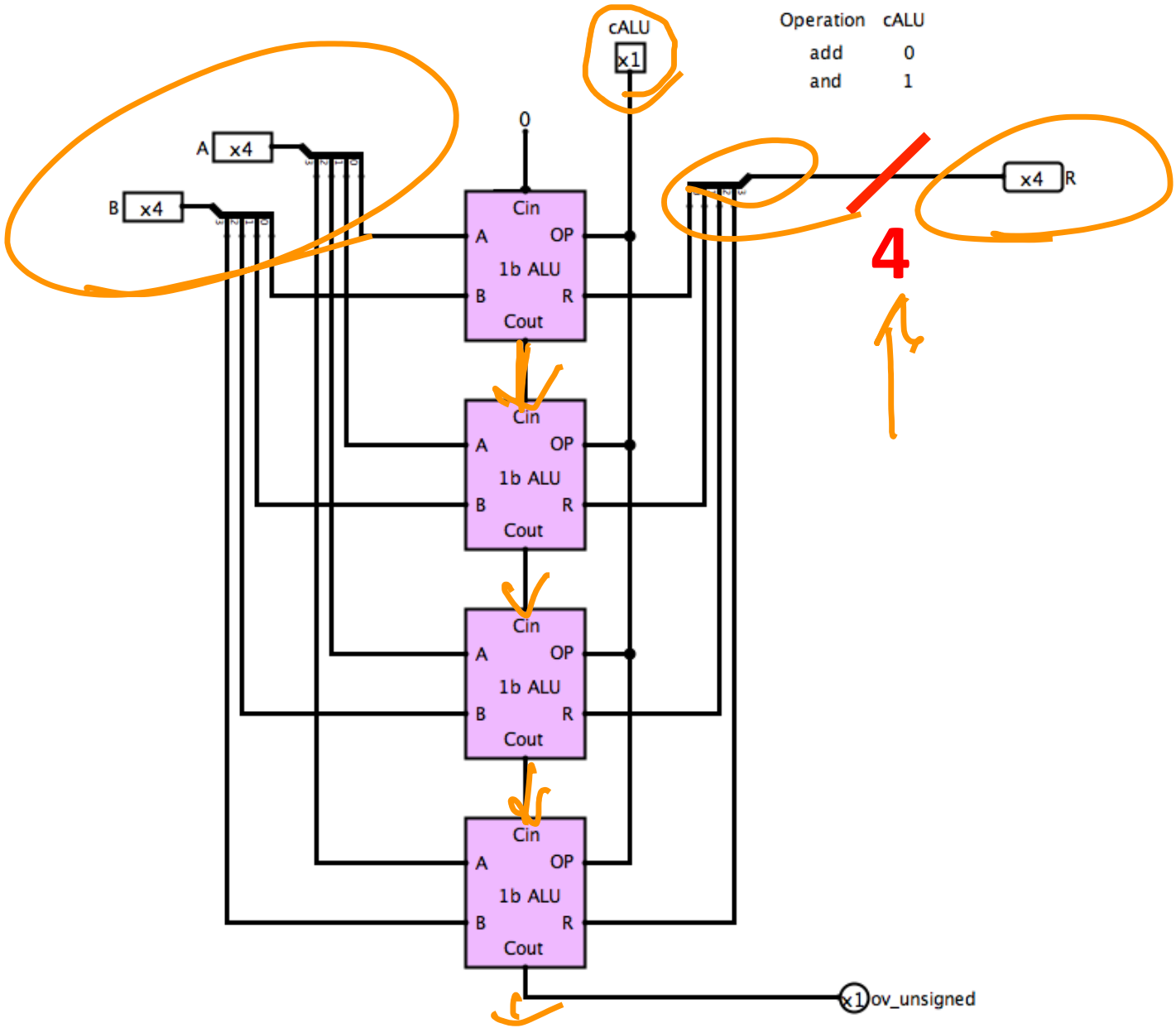
A	B	S(elect)	Y
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1

X stand's for don't care:

- the output is same for X=0 and X=1
- X'es reduce the size of the truth table and simplify logic synthesis
- Use them when you can!

$$Y = A \cdot S + B \cdot S$$

4-Bit ALU: **and, add**



2's Complement Subtraction

- Idea:

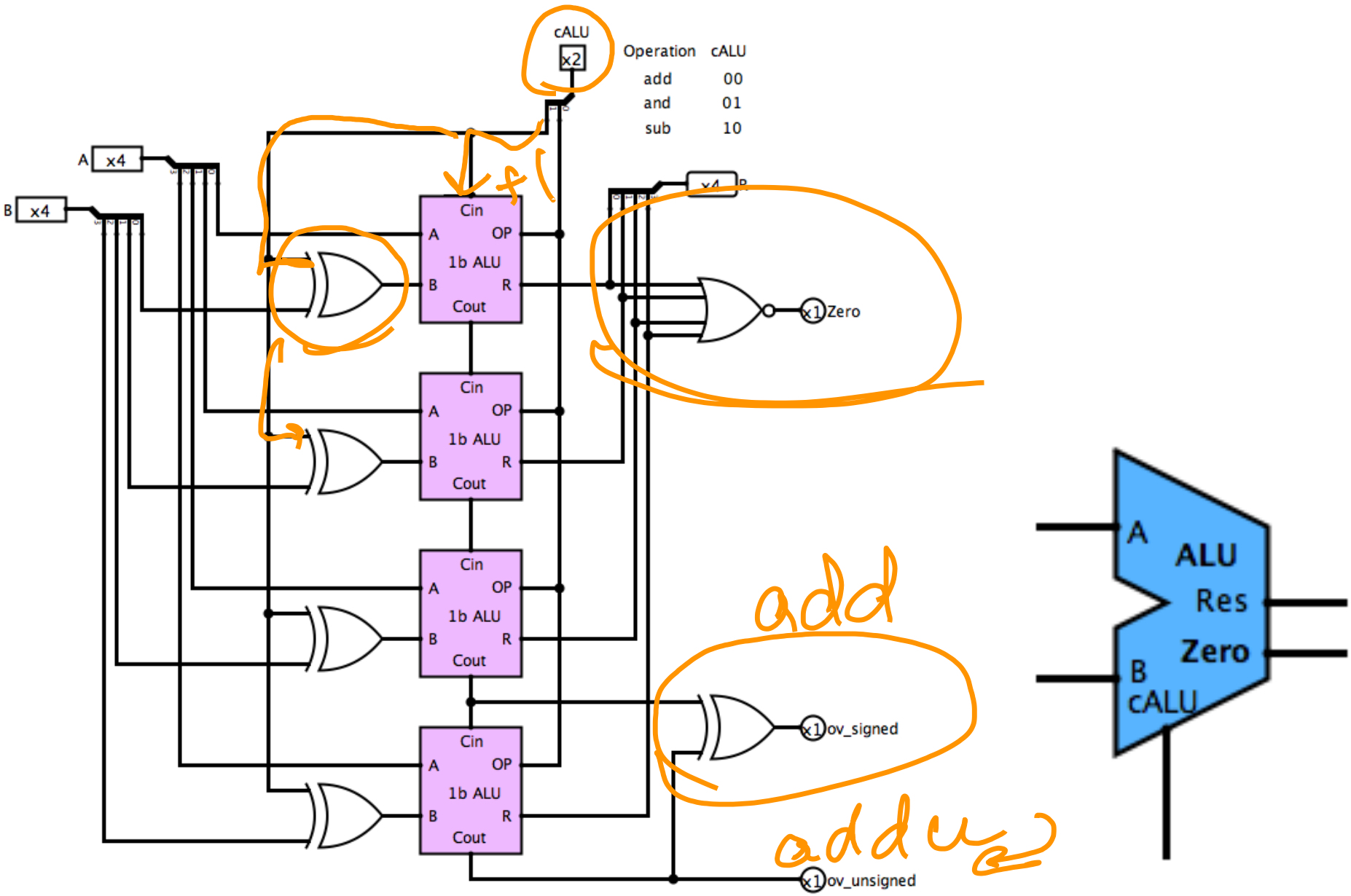
  `a - b = a + (-b)`

- How compute **`-b`**?

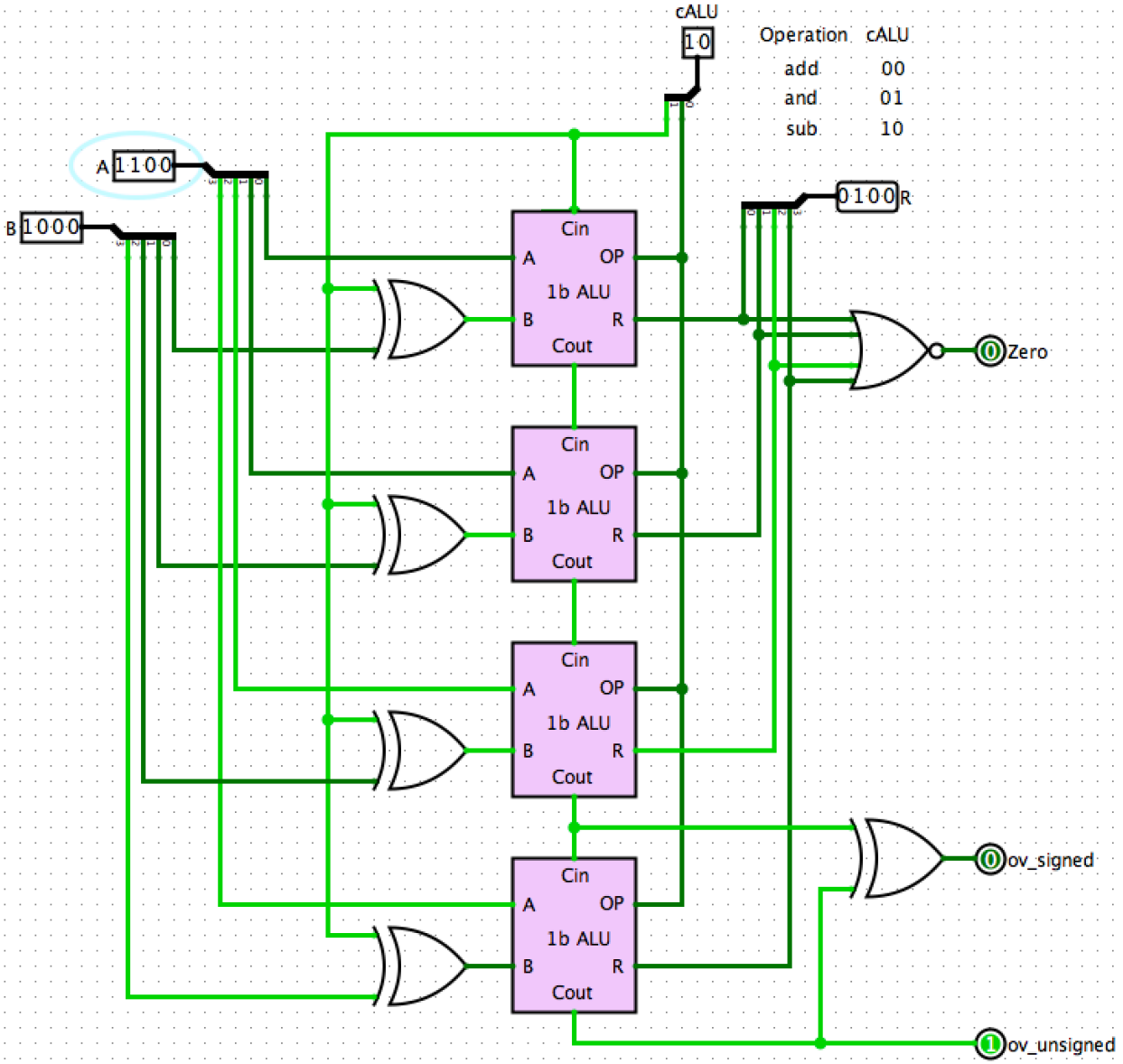
  Invert all bits and add 1!

- **`add/sub/and`** ALU on next slide …

4-Bit ALU: **add, and, sub, beq**



ALU Demo …


cALU
10
Operation cALU
add 00
and 01
sub 10
A 1100
B 1000
0100 R
Cin
A OP
1b ALU
B R
Cout
Zero
ov_signed
ov_unsigned

And where is my chip?

- Lab 1:
  - Synthesize (part of) MIPS
  - Tool: Multisim
- To go all the way to a chip …
  - More powerful tool
  - And learn yet another language, Verilog
  - Covered in EE/CS 151

Today's Agenda

- 1-Bit Adder
- Boolean Algebra
- Hardware: Chips, transistors, volts!
- CMOS Logic Gates
- ALU
- **And in Conclusion, …**

And in Conclusion, …

- Arithmetic operations (**`add, sub, and, or, …`**) can be manipulated with **Boolean algebra**
- Boolean operations (**`and, or, ...`**) can be realized efficiently with **electronic circuits**
  - Basis for modern computer hardware
- Complex functions can be composed by combining simpler ones
  - E.g. **`and, nor, …`** → **`ALU`** → **`Computer`**

Extras

Your Turn

- Odd parity: 1 for odd number of inputs
- Truth table: which column for Y (A … E) is correct?

			Y				
X_1	X_2	X_2	**A**	**B**	**C**	**D**	**E**
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0
0	1	0	0	1	1	1	1
0	1	1	1	0	0	0	0
1	0	0	0	1	1	1	0
1	0	1	1	0	0	0	0
1	1	0	1	0	1	0	1
1	1	1	0	1	1	1	0

Your Turn

- Odd parity: 1 for odd number of inputs
- Truth table: which column for Y (A … E) is correct?

			Y				
X_1	X_2	X_2	**A**	**B**	**C**	**D**	**E**
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	0
0	1	0	0	1	1	1	1
0	1	1	1	0	0	0	0
1	0	0	0	1	1	1	0
1	0	1	1	0	0	0	0
1	1	0	1	0	1	0	1
1	1	1	0	1	1	1	0