CS 61C:
Great Ideas in Computer Architecture
Hardware intro, Digital Logic

Instructors:

Nicholas Weaver & Vladimir Stojanovic

http://inst.eecs.Berkeley.edu/~cs61c/sp16

Levels of Representation/Interpretation

High Level Language Program (e.g., C)

```
temp = v[k];
v[k] = v[k+1];
v[k+1] = temp;
```

Compiler

Assembly Language Program (e.g., MIPS)

```
lw   $t0, 0($2)
lw   $t1, 4($2)
sw   $t1, 0($2)
sw   $t0, 4($2)
```

Anything can be represented as a *number*, i.e., data or instructions

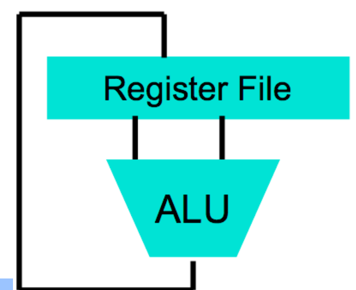
Assembler

Machine Language Program (MIPS)

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

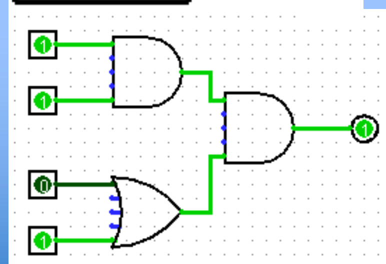
Machine Interpretation

Hardware Architecture Description (e.g., block diagrams)



Architecture Implementation

Logic Circuit Description (Circuit Schematic Diagrams)



Hardware Design

- Next several weeks: how a modern processor is built, starting with basic elements as building blocks
- Why study hardware design?
  - Understand capabilities and limitations of HW in general and processors in particular
  - What processors can do fast and what they can't do fast (avoid slow things if you want your code to run fast!)
  - Background for more in-depth HW courses (EECS 151, CS 152)
  - Hard to know what you'll need for next 30 years
  - There is only so much you can do with standard processors: you may need to design own custom HW for extra performance
    - Even some commercial processors today have customizable hardware!

Synchronous Digital Systems

Hardware of a processor, such as the MIPS, is an example of a Synchronous Digital System

Synchronous:

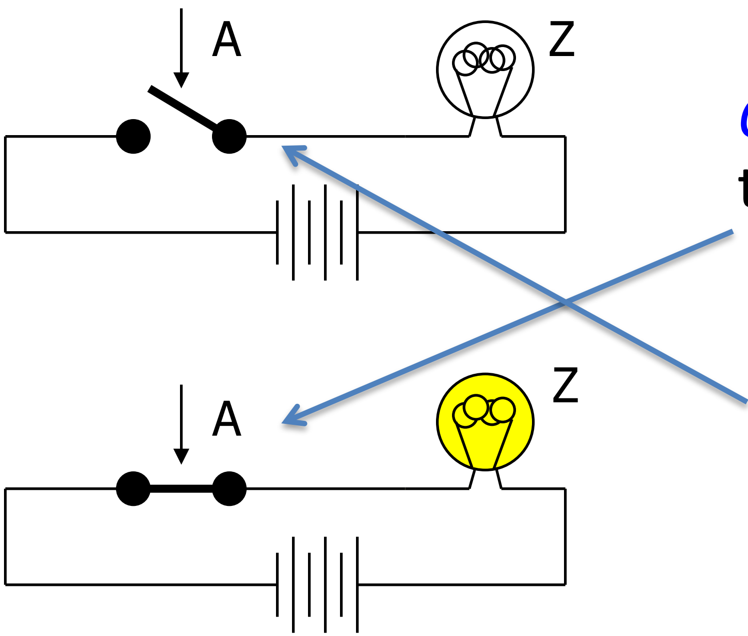
- All operations coordinated by a central clock
  - "Heartbeat" of the system!

Digital:

- Represent all values by discrete values
- Two binary digits: 1 and 0
- Electrical signals are treated as 1's and 0's
  - 1 and 0 are complements of each other
- High /low voltage for true / false, 1 / 0

Switches: Basic Element of Physical Implementations

- Implementing a simple circuit (arrow shows action if wire changes to “1” or is *asserted*):



On-switch (if A is “1” or asserted) turns-on light bulb (Z)

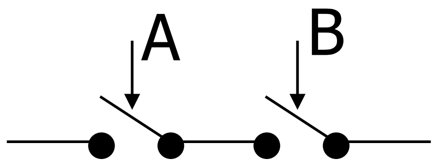
Off-switch (if A is “0” or unasserted) turns-off light bulb (Z)

$$Z \equiv A$$

Switches (cont'd)

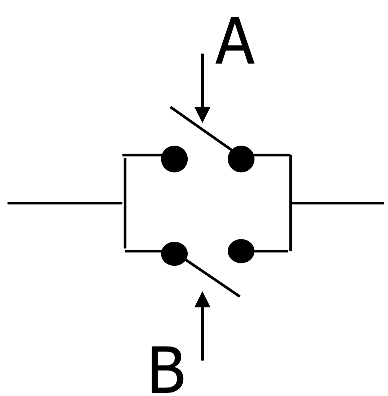
- Compose switches into more complex ones (Boolean functions):

AND



Z ≡ A and B

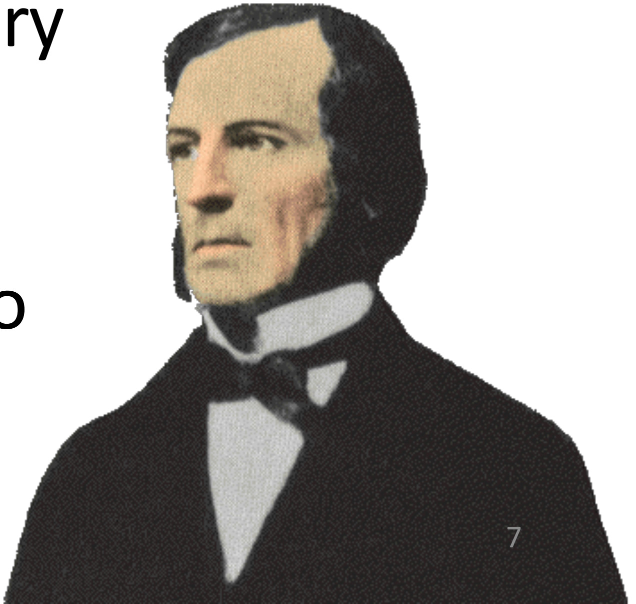
OR



Z ≡ A or B

Historical Note

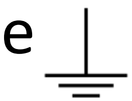
- Early computer designers built ad hoc circuits from switches
- Began to notice common patterns in their work: ANDs, ORs, …
- Master's thesis (by Claude Shannon, 1940) made link between work and 19th Century Mathematician George Boole
  - Called it "Boolean" in his honor
- Could apply math to give theory to hardware design, minimization, …



Transistors

- High voltage (V_{dd}) represents 1, or true

  - In modern microprocessors, Vdd ~ 1.0 Volt
- Low voltage (0 Volt or Ground) represents 0, or false



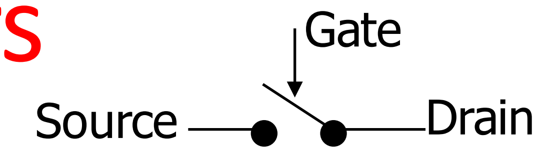
- Pick a midpoint voltage to decide if a 0 or a 1
  - Voltage greater than midpoint = 1
  - Voltage less than midpoint = 0
  - This removes noise as signals propagate – a big advantage of digital systems over analog systems
- If one switch can control another switch, we can build a computer!
- Our switches: CMOS transistors

CMOS Transistor Networks

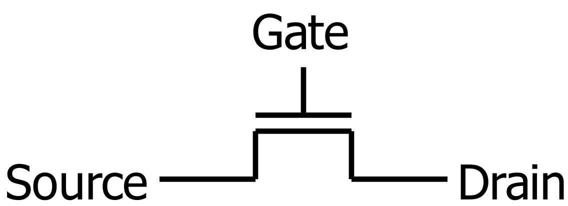
- Modern digital systems designed in CMOS
  - MOS: Metal-Oxide on Semiconductor
  - C for complementary: use *pairs* of normally-*on* and normally-*off* switches

- CMOS transistors act as voltage-controlled switches
  - Similar, though easier to work with, than electro-mechanical relay switches from earlier era
  - Use energy primarily when switching

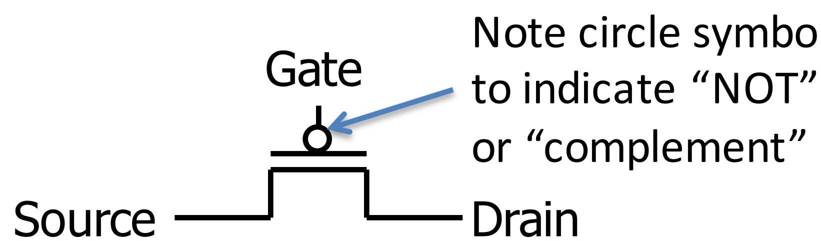
CMOS Transistors



- Three terminals: source, gate, and drain
  - Switch action:
    if voltage on gate terminal is (some amount) higher/lower than source terminal then conducting path established between drain and source terminals (switch is closed)

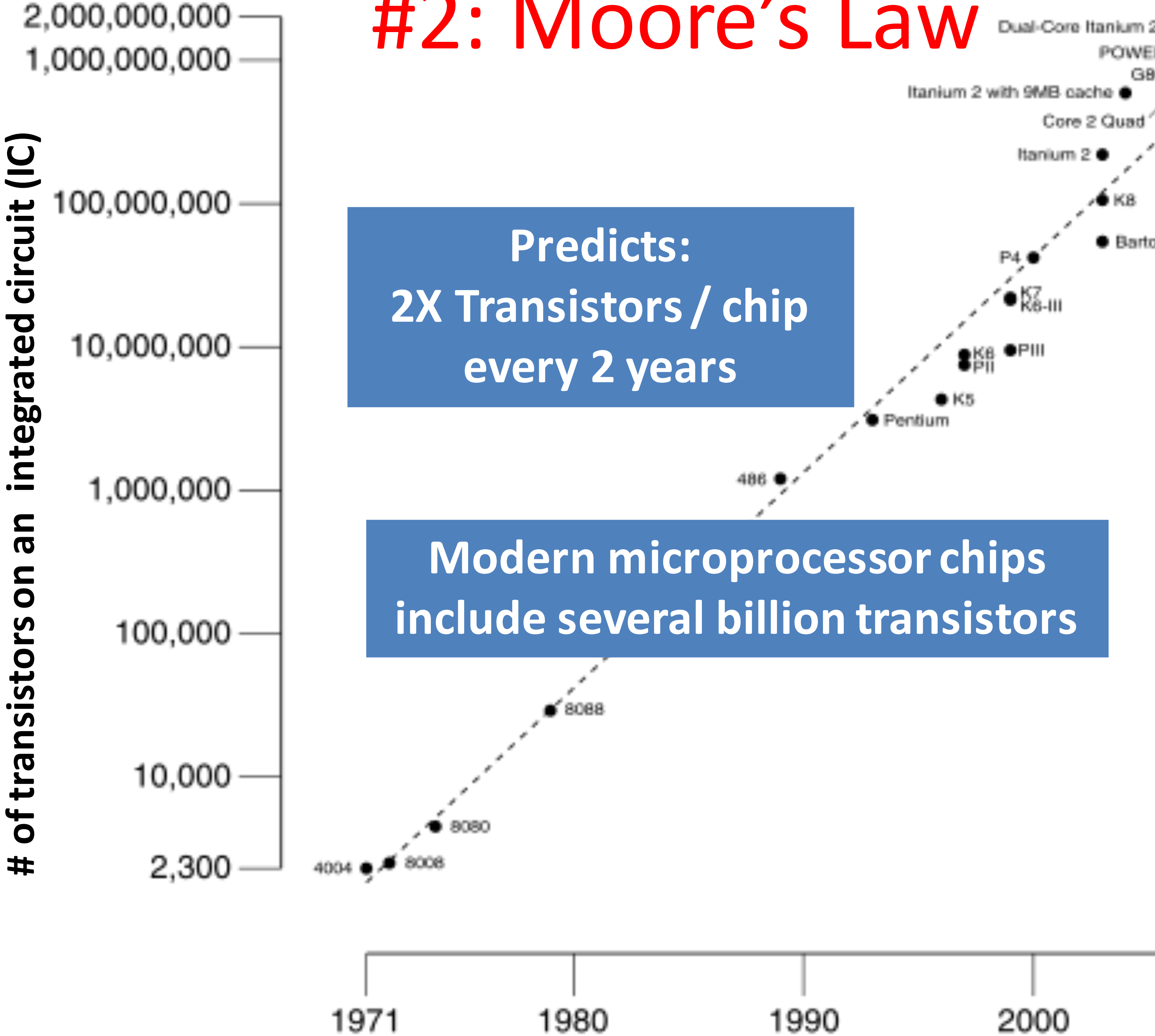


n-channel transitor
off when voltage at Gate is low
on when:
voltage(Gate) > voltage (Threshold)



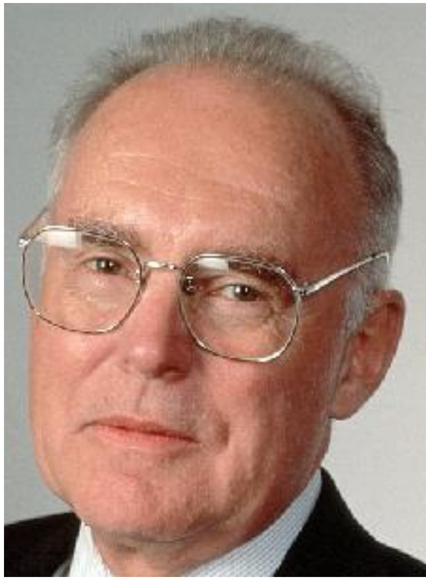
p-channel transistor
on when voltage at Gate is low
off when:
voltage(Gate) > voltage (Threshold)

#2: Moore's Law



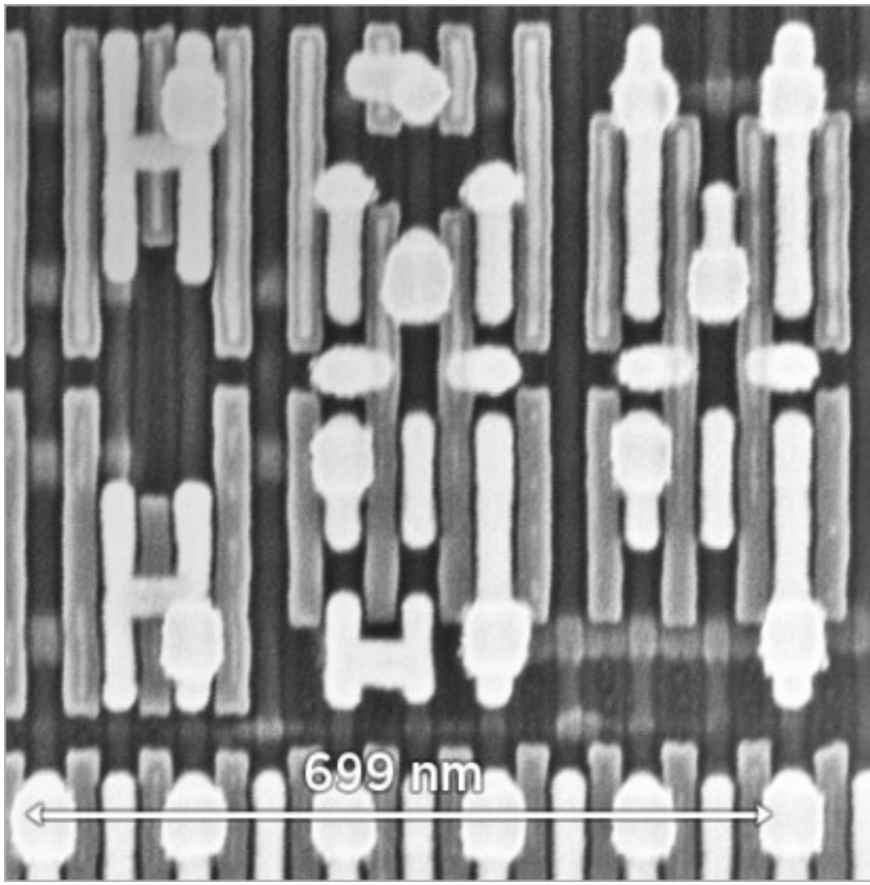
Predicts:
2X Transistors / chip every 2 years

Modern microprocessor chips include several billion transistors

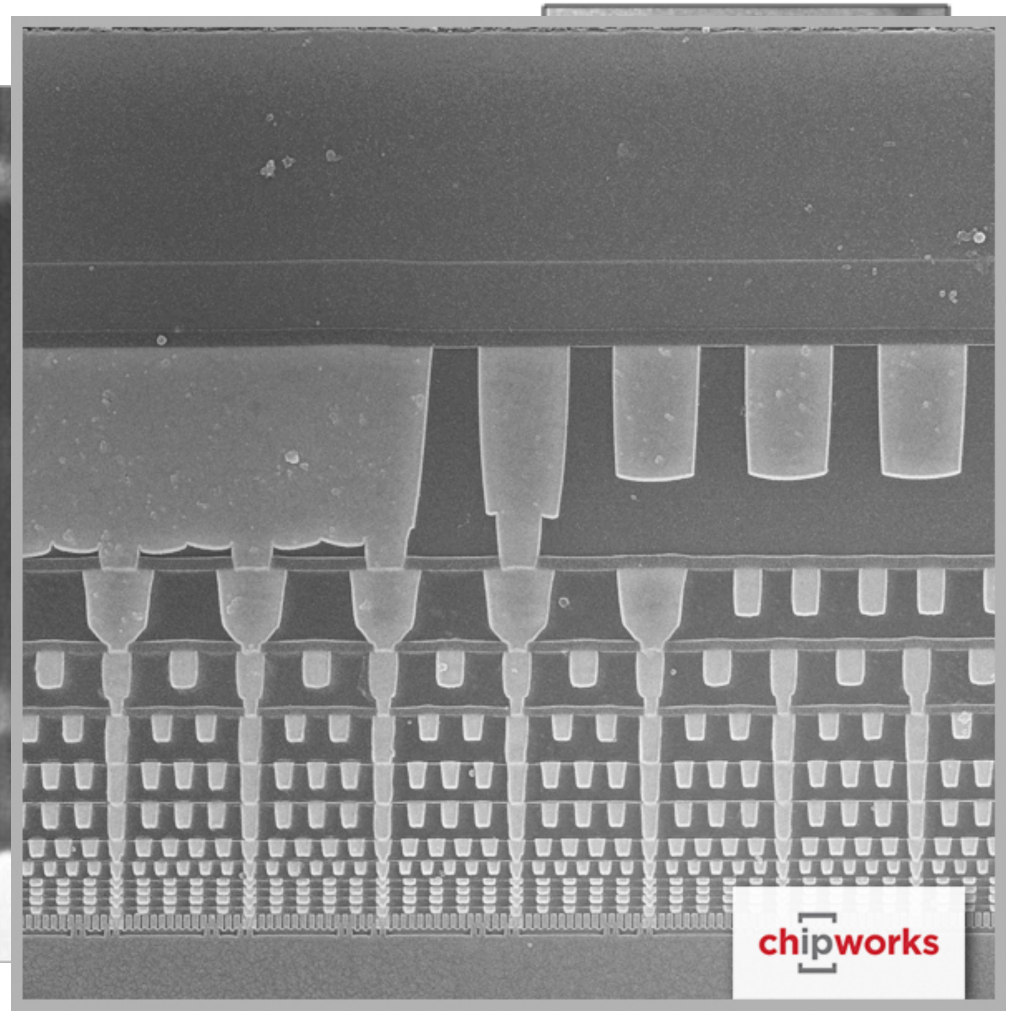


Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Intel 14nm Technology

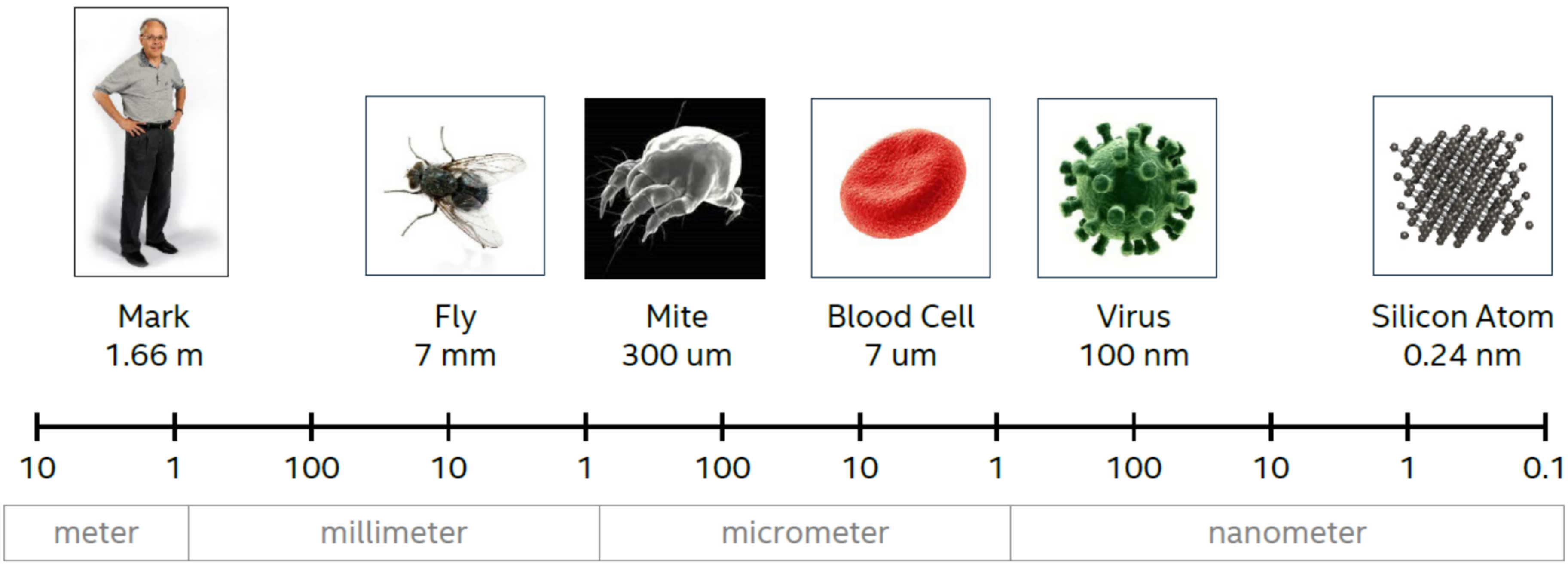


Plan view of transistors



Side view of wiring layers

Sense of Scale



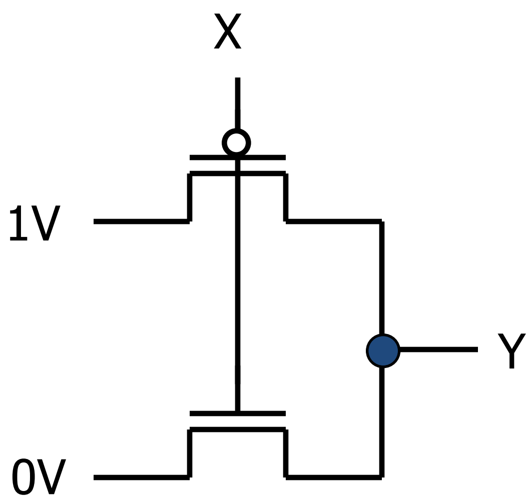
Source: Mark Bohr, IDF14

CMOS Circuit Rules

- Don't pass weak values => Use Complementary Pairs
  - N-type transistors pass weak 1's (V_{dd} - V_{th})
  - N-type transistors pass strong 0's (ground)
  - Use N-type transistors only to pass 0's (N for negative)
  - Converse for P-type transistors: Pass weak 0s, strong 1s
    - Pass weak 0's (V_{th}), strong 1's (V_{dd})
    - Use P-type transistors only to pass 1's (P for positive)
  - Use pairs of N-type and P-type to get strong values
- Never leave a wire undriven
  - Make sure there's always a path to V_{dd} or GND
- Never create a path from V_{dd} to GND (ground)
  - This would short-circuit the power supply!

CMOS Networks

p-channel transistor
on when voltage at Gate is low
off when:
voltage(Gate) > voltage (Threshold)



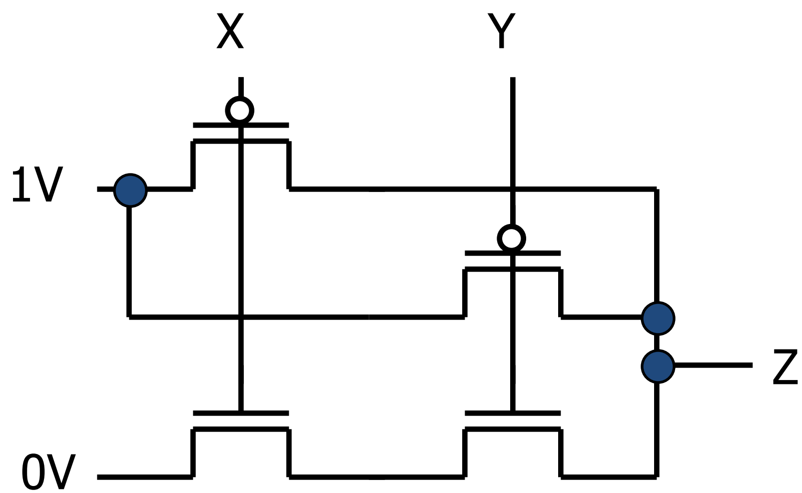
n-channel transitor
off when voltage at Gate is low
on when:
voltage(Gate) > voltage (Threshold)

what is the relationship between x and y?

x	y
0 Volt (GND)	1 Volt (Vdd)
1 Volt (Vdd)	0 Volt (GND)

Called an *inverter* or *not gate*

Two-Input Networks



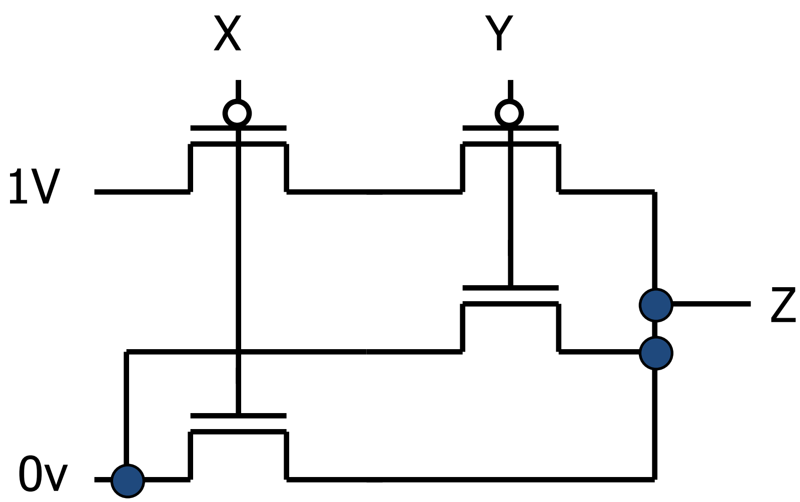
what is the relationship between x, y and z?

x	y	z
0 Volt	0 Volt	1 Volt
0 Volt	1 Volt	1 Volt
1 Volt	0 Volt	1 Volt
1 Volt	1 Volt	0 Volt

Called a *NAND gate*

(NOT AND)

Clickers/Peer Instruction



x	y	z: A	B	C	D	
0 Volt	0 Volt	0	0	1	1	Volts
0 Volt	1 Volt	0	1	0	1	Volts
1 Volt	0 Volt	0	1	0	1	Volts
1 Volt	1 Volt	1	1	0	0	Volts

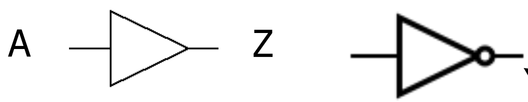
Administrivia

- Proj2-1 due Sunday 2/21!
- MT1 next **Thursday** night
  - If you are in DSP or have an exam conflict, you should have received an email from Fred. If not, email Fred and William.
  - EE16B students will take it at the normal time
  - We will post room assignments on Piazza over the weekend
- Proj2-2 will be released late next week.

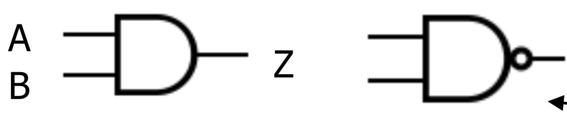
Combinational Logic Symbols

- Common combinational logic systems have standard symbols called logic gates

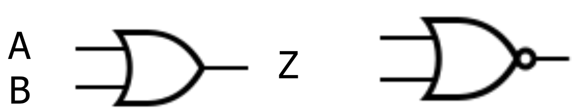
  - Buffer, NOT



  - AND, NAND



  - OR, NOR



Inverting versions (NOT, NAND, NOR) easiest to implement with CMOS transistors (the switches we have available and use most)

Boolean Algebra

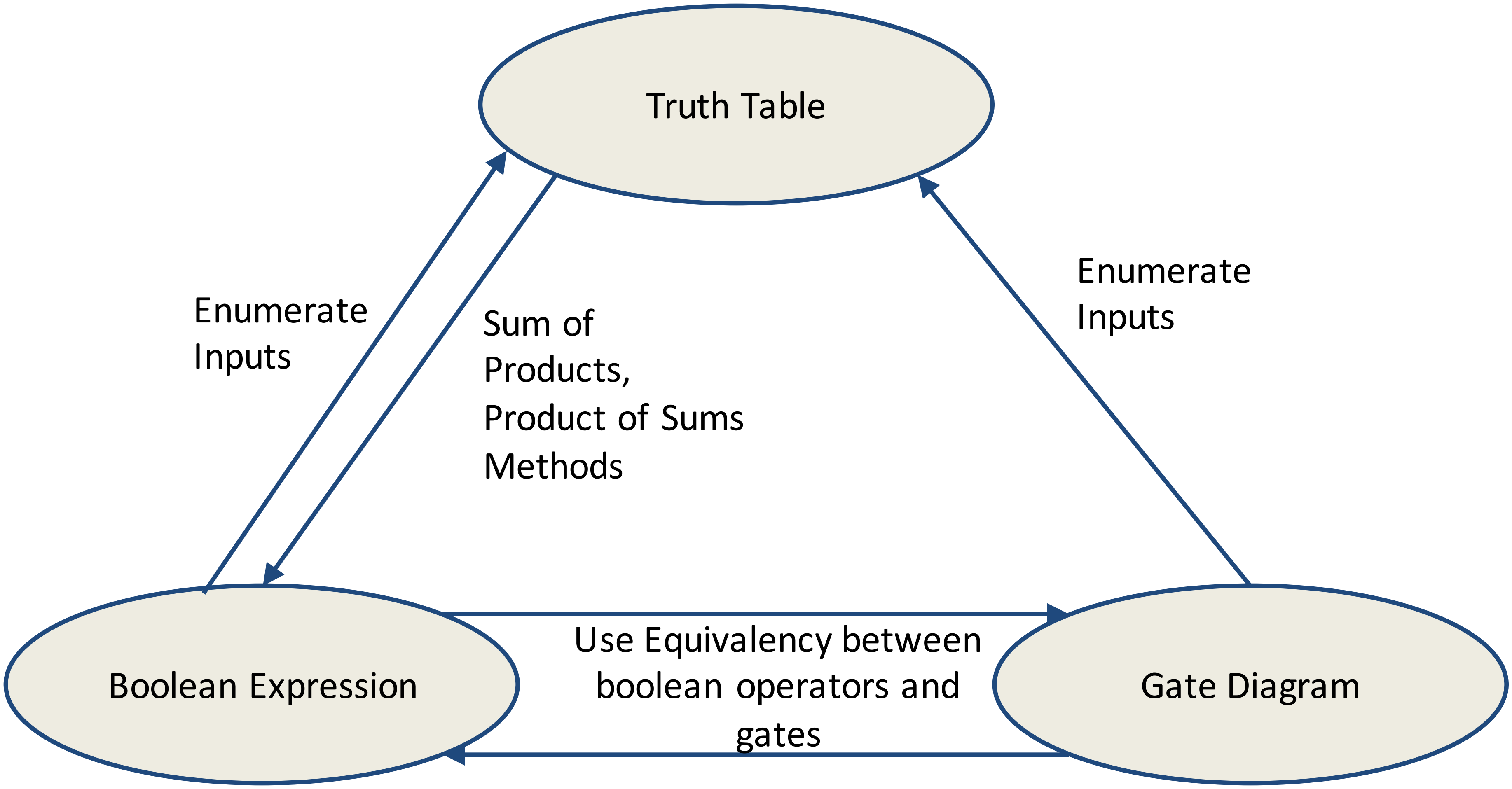
- Use plus “+” for OR
  - “logical sum”
- Use product for AND (a•b or implied via ab)
  - “logical product”
- “Hat” to mean complement (NOT)
- Thus

  $ab + a + \overline{c}$

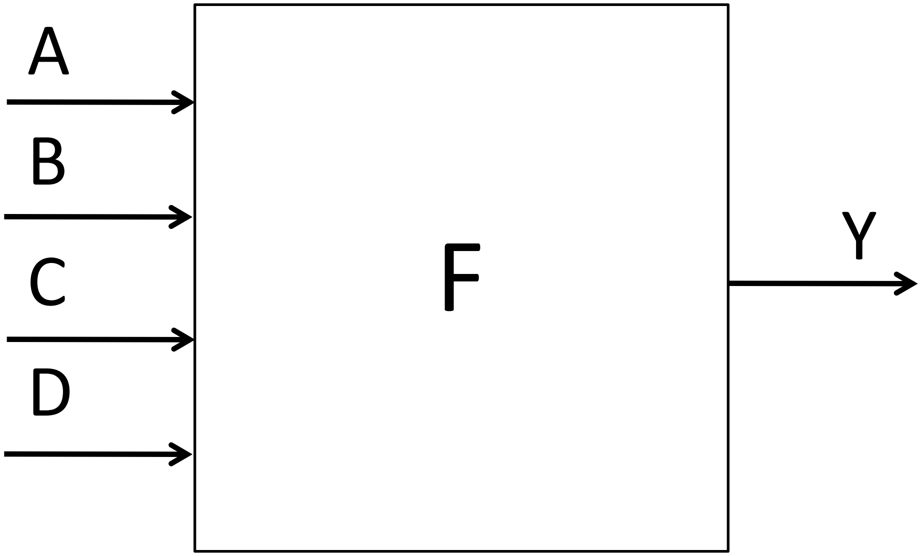
  $= a \bullet b + a + \overline{c}$

  = (a AND b) OR a OR (NOT c)

Representations of Combinational Logic (groups of logic gates)



Truth Tables for Combinational Logic



Exhaustive list of the output value generated for each combination of inputs

How many logic functions can be defined with N inputs?

a	b	c	d	y
0	0	0	0	F(0,0,0,0)
0	0	0	1	F(0,0,0,1)
0	0	1	0	F(0,0,1,0)
0	0	1	1	F(0,0,1,1)
0	1	0	0	F(0,1,0,0)
0	1	0	1	F(0,1,0,1)
0	1	1	0	F(0,1,1,0)
0	1	1	1	F(0,1,1,1)
1	0	0	0	F(1,0,0,0)
1	0	0	1	F(1,0,0,1)
1	0	1	0	F(1,0,1,0)
1	0	1	1	F(1,0,1,1)
1	1	0	0	F(1,1,0,0)
1	1	0	1	F(1,1,0,1)
1	1	1	0	F(1,1,1,0)
1	1	1	1	F(1,1,1,1)

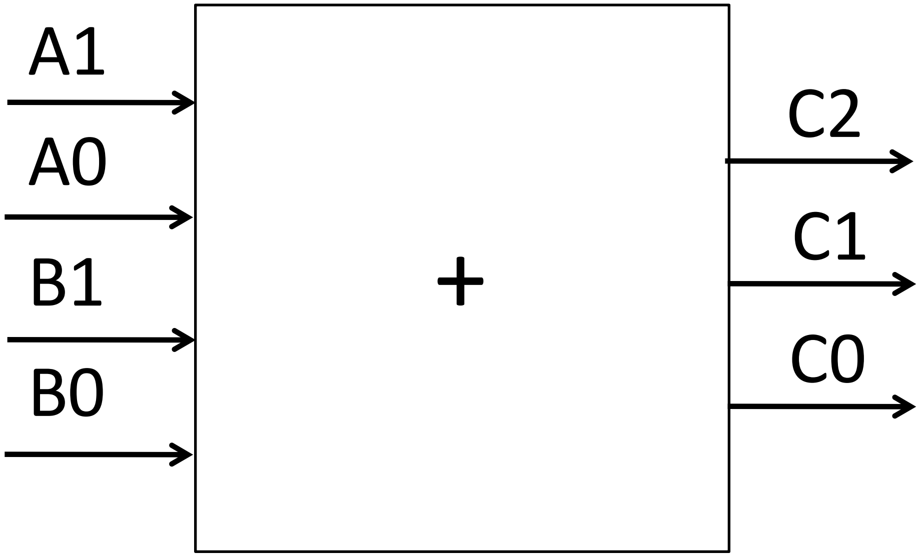
Truth Table Example #1:
y= F(a,b): 1 iff a ≠ b

a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table Example #2: 2-bit Adder

A	B	C
$a_1 a_0$	$b_1 b_0$	$c_2 c_1 c_0$

How Many Rows?



Truth Table Example #3: 32-bit Unsigned Adder

A	B	C
000 ... 0	000 ... 0	000 ... 00
000 ... 0	000 ... 1	000 ... 01
.	.	.
.	.	.
.	.	.
111 ... 1	111 ... 1	111 ... 10

How Many Rows?

Truth Table Example #4: 3-input Majority Circuit

Y =

This is called *Sum of Products* form; Just another way to represent the TT as a logical expression

More simplified forms (fewer gates and wires)

a	b	c	y
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	1

And in Conclusion, …

- Multiple Hardware Representations
  - Analog voltages quantized to represent logic 0 and logic 1
  - Transistor switches form gates: AND, OR, NOT, NAND, NOR
  - Truth table mapped to gates for combinational logic design
  - Boolean algebra for gate minimization