CS 61C: Great Ideas in Computer Architecture

Lecture 9: *Synchronous Digital Systems*

Instructor: Sagar Karandikar

sagark@eecs.berkeley.edu

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

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

You are Here!

Software | *Hardware*

- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions
  All gates @ one time
- Programming Languages

Harness Parallelism & Achieve High Performance

Warehouse Scale Computer

Smart Phone

Levels of Representation/ Interpretation

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

Compiler

Assembly Language Program (e.g., MIPS)

Assembler

Machine Language Program (MIPS)

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

```
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

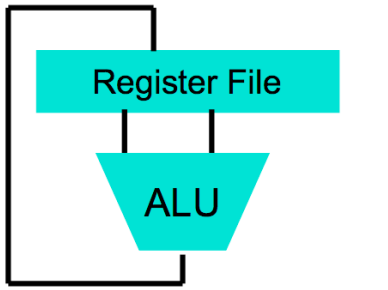
```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```

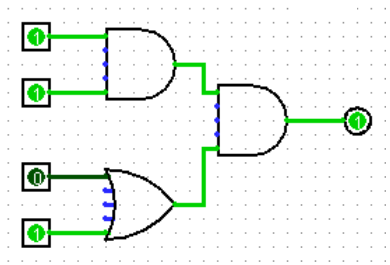
Machine Interpretation

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 (CS 150, 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:

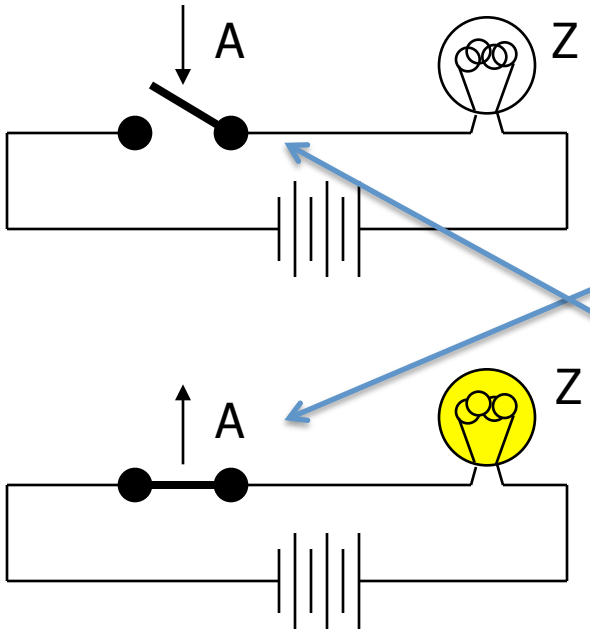
- All operations coordinated by a central clock
  - "Heartbeat" of the system!

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*):



Close switch (if A is "1" or asserted) and turn on light bulb (Z)

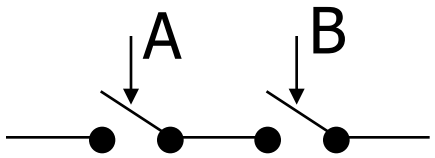
Open switch (if A is "0" or unasserted) and turn off light bulb (Z)

$$Z \equiv A$$

Switches (cont'd)

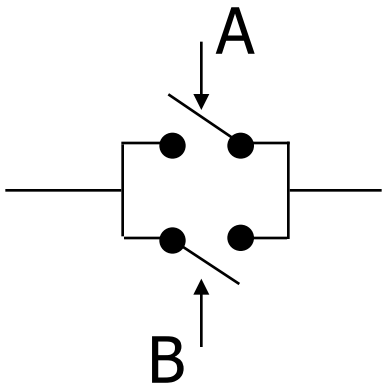
- Compose switches into more complex ones (Boolean functions):

AND



Z ≡ A and B

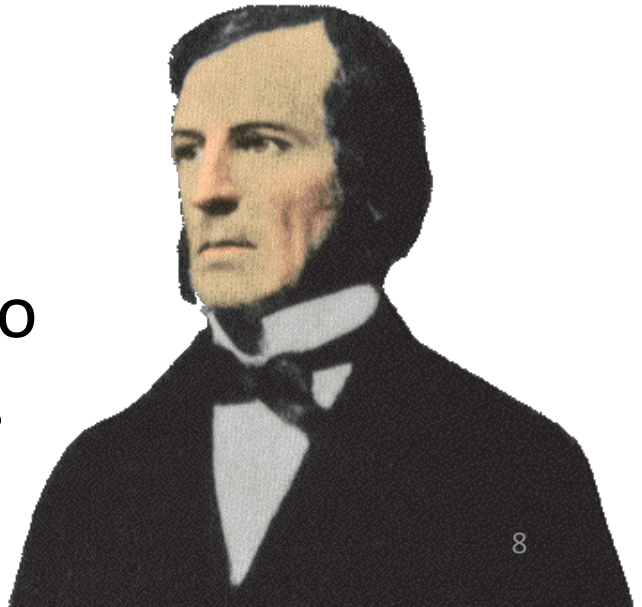
OR



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) 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, …



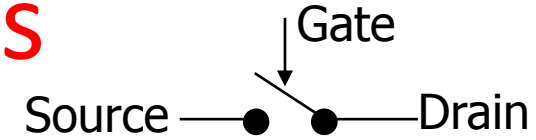
Transistors

- 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

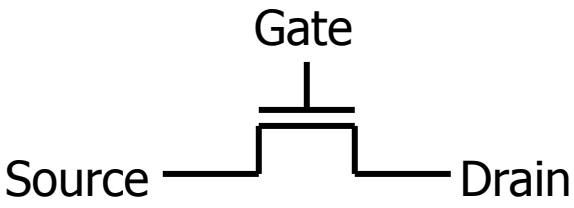
CMOS Transistor Networks

- Modern digital systems designed in CMOS
  - MOS: Metal-Oxide on Semiconductor
  - C for complementary: use *pairs* of normally-open and normally-closed switches
    - Used to be called COS-MOS for complementary-symmetry - MOS
- 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

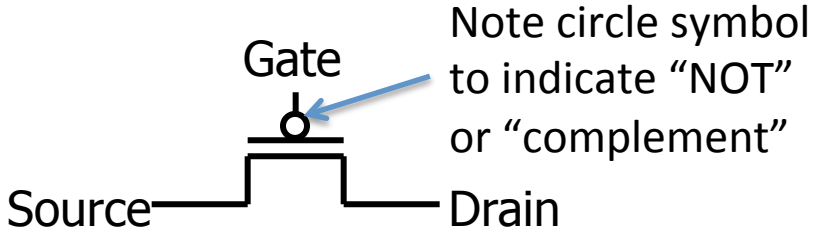
CMOS Transistors



- 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)

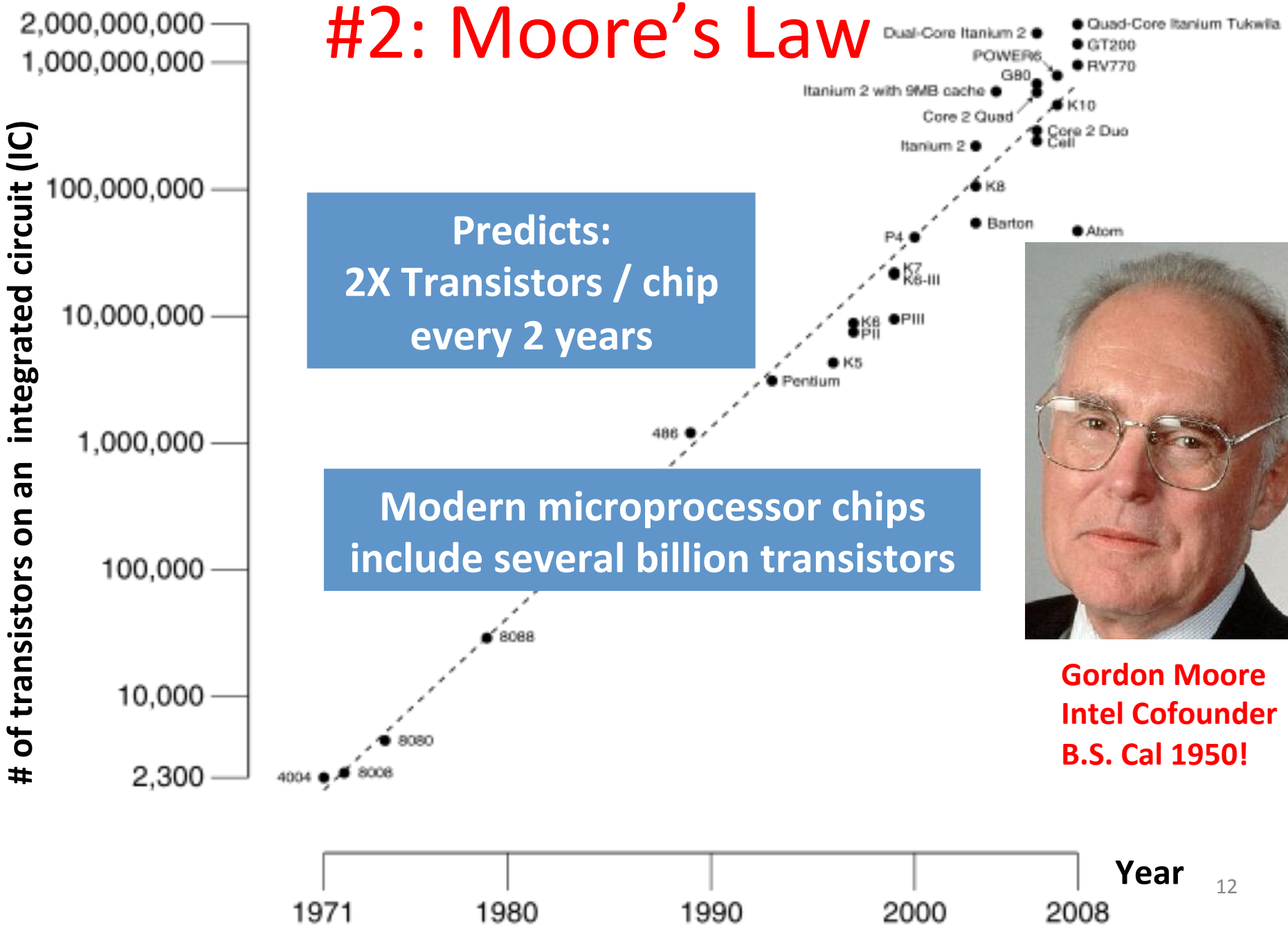


n-channel transitor
open when voltage at Gate is low
closes when:
voltage(Gate) > voltage (Threshold)
(High resistance when gate voltage Low,
Low resistance when gate voltage High)



p-channel transistor
closed when voltage at Gate is low
opens when:
voltage(Gate) > voltage (Threshold)
(Low resistance when gate voltage Low,
High resistance when gate voltage High)

#2: Moore's Law

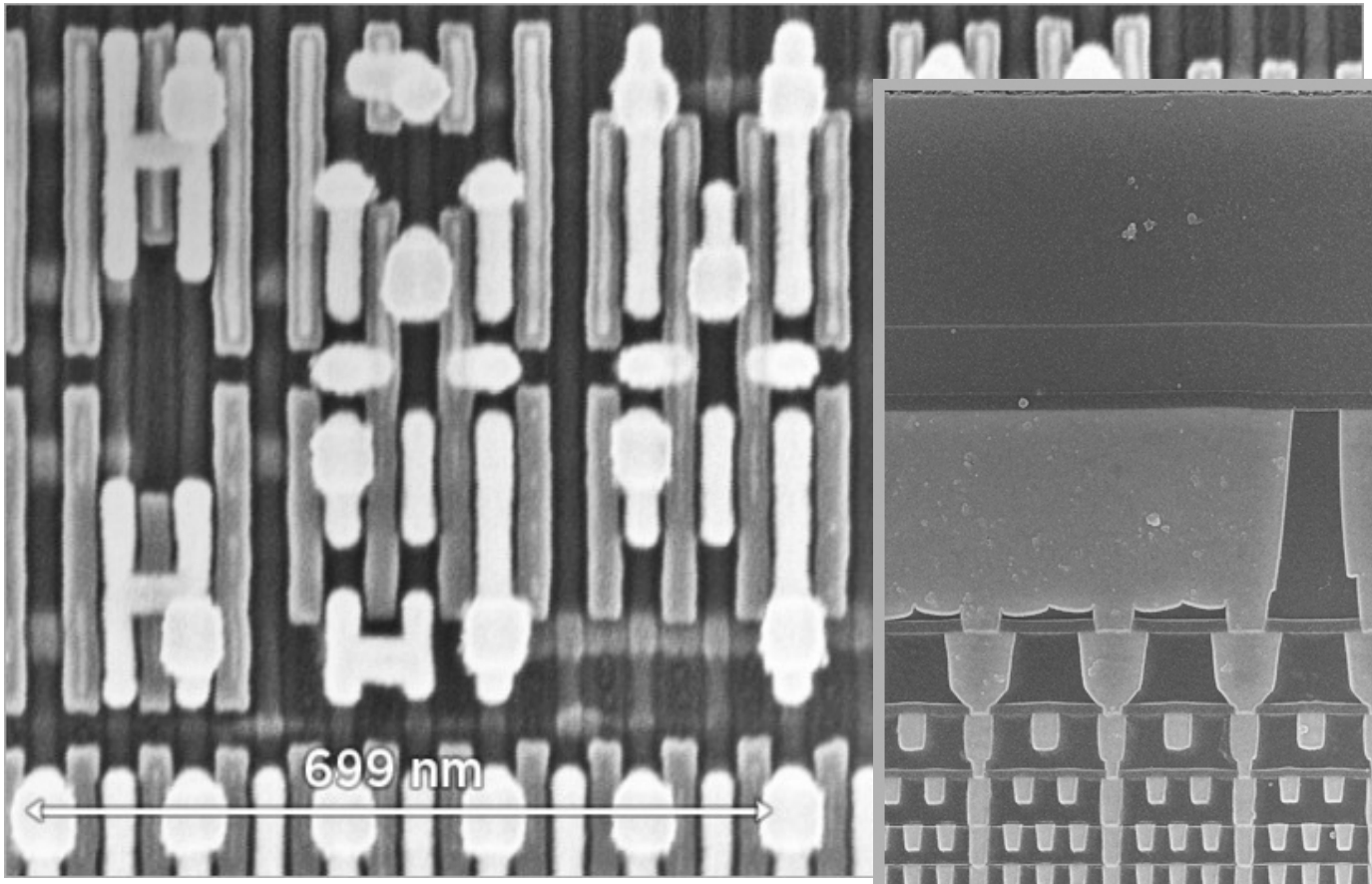


Predicts:
2X Transistors / chip every 2 years

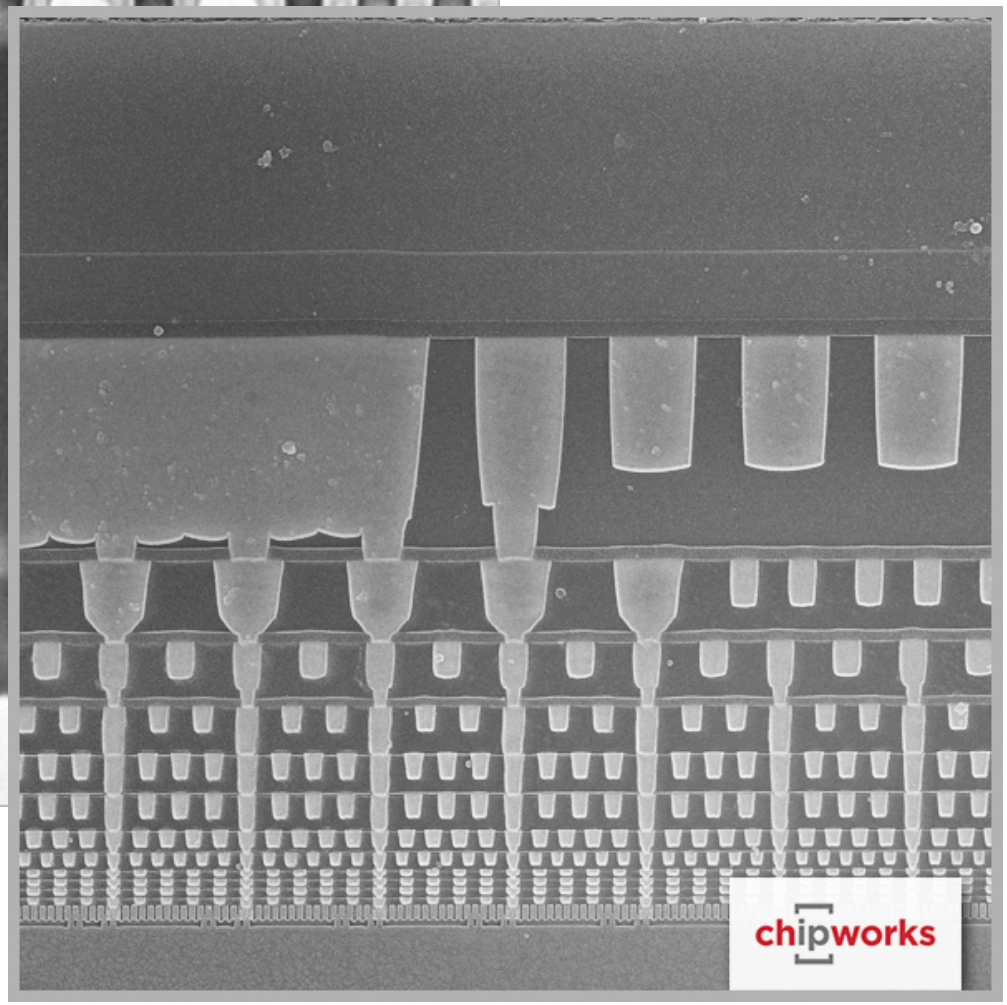
Modern microprocessor chips include several billion transistors

Gordon Moore
Intel Cofounder
B.S. Cal 1950!

Intel 14nm Technology



Plan view of transistors



Side view of wiring layers

Scale of the Universe

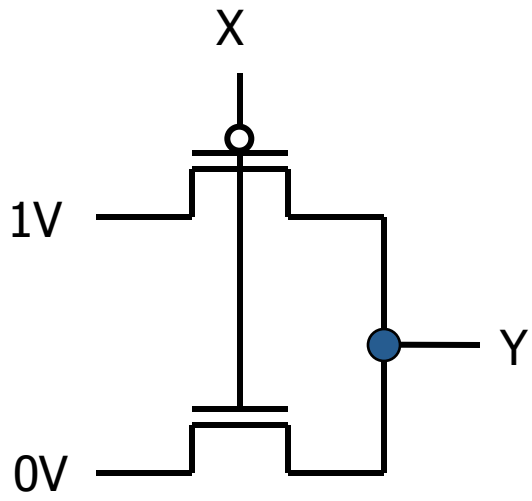
http://htwins.net/scale2/

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
closed when voltage at Gate is low
opens when:
voltage(Gate) > voltage (Threshold)



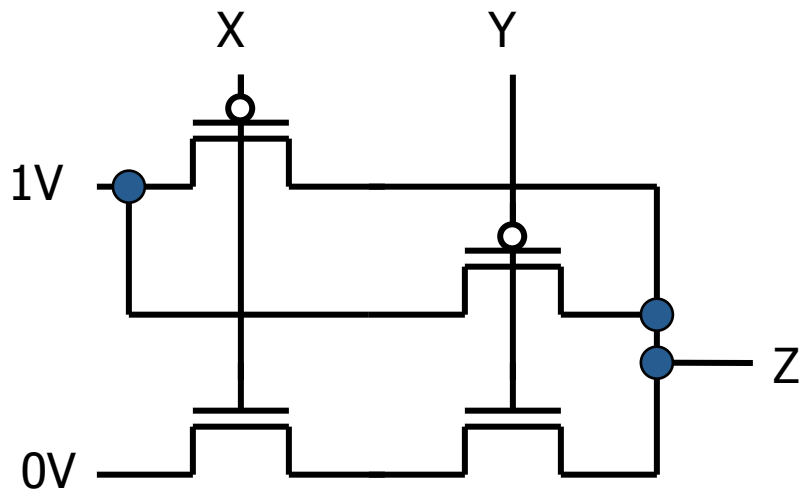
n-channel transitor
open when voltage at Gate is low
closes when:
voltage(Gate) > voltage (Threshold)

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

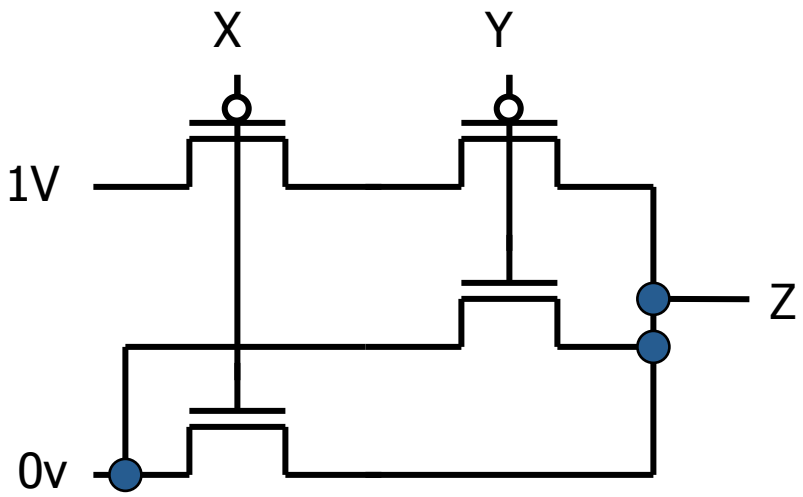


what is the relationship between x, y and z?

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

- HW2 out
  - We recommend doing this before the midterm
- Proj 2-1 out
  - Make sure you test your code on hive machines, that's where we'll grade them
  - Team registration problems? Email Jay

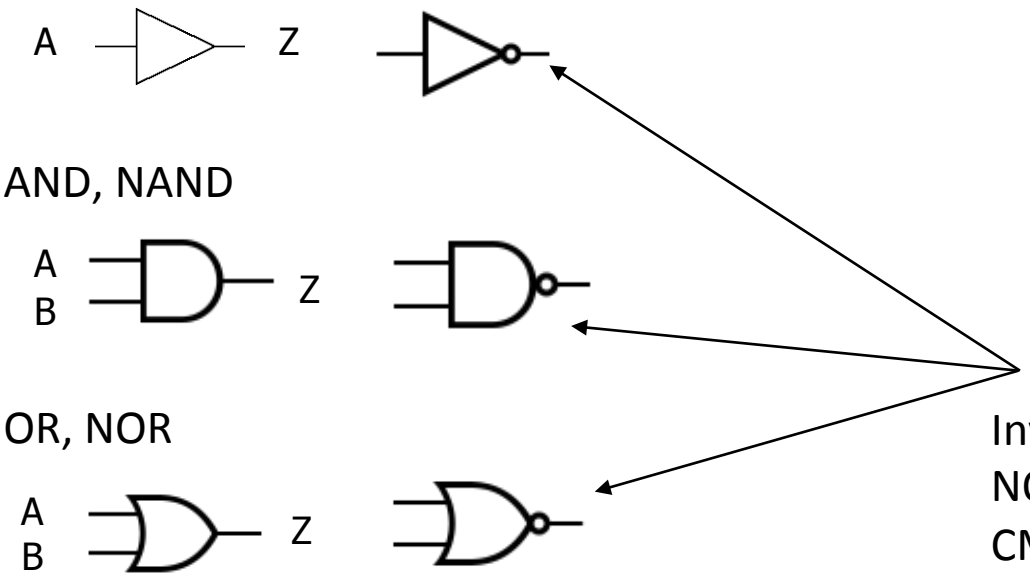
Administrivia

- Midterm this Thursday
  - In this room, at this time
  - One 8.5”x11” handwritten cheatsheet
  - We’ll provide a MIPS green sheet
  - No electronics
  - Covers up to and including last lecture (07/02)
  - **TA-led review session tonight (07/06) from 5-8pm in HP Auditorium**

Break

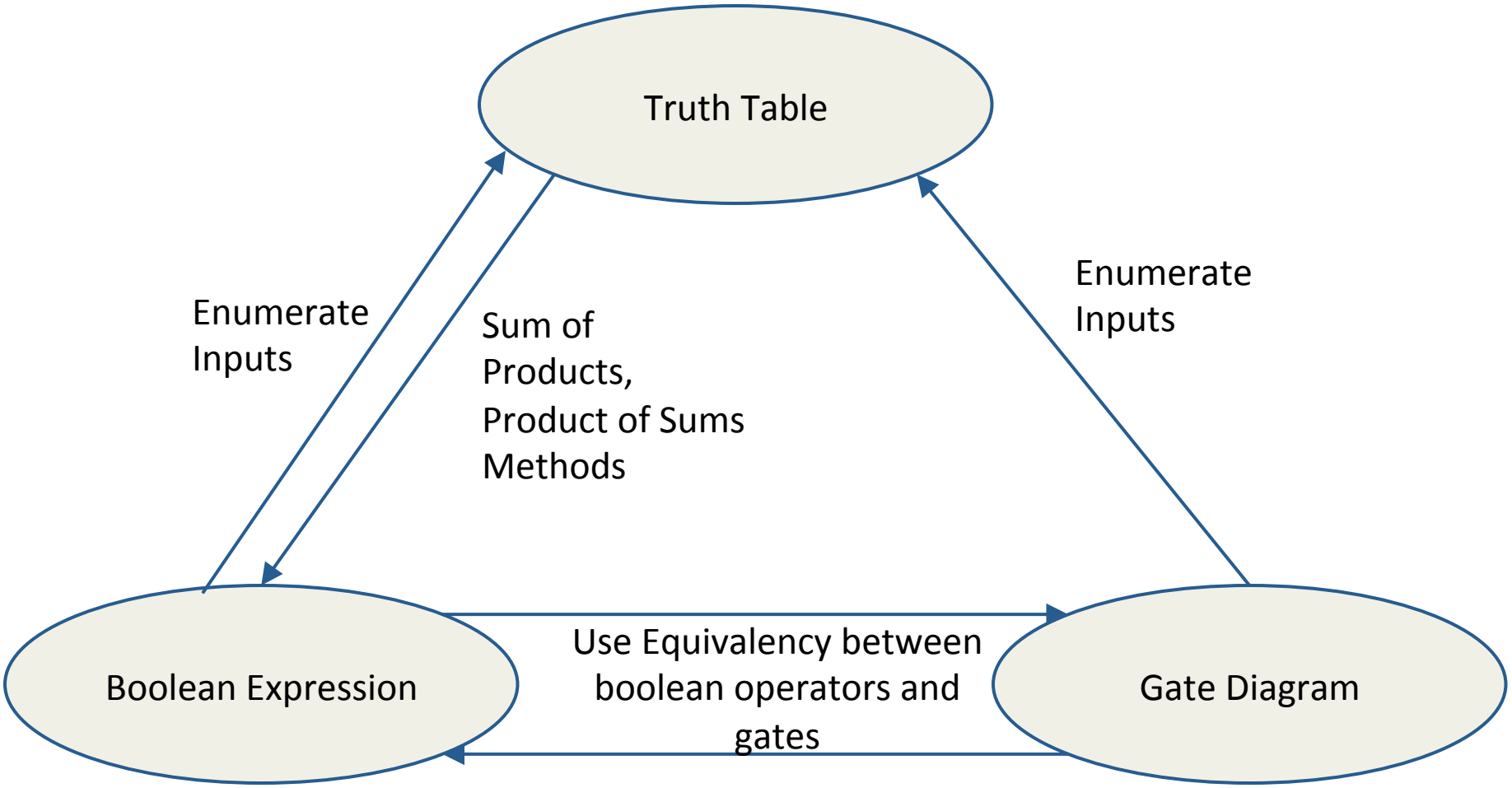
Combinational Logic Symbols

- Common combinational logic systems have standard symbols called logic gates
  - Buffer, NOT
  - AND, NAND
  - OR, NOR

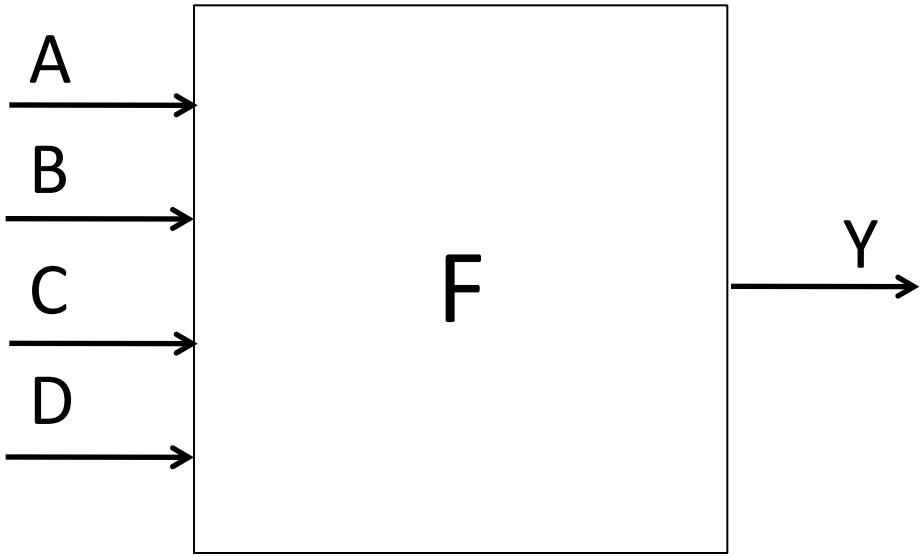


Inverting versions (NOT, NAND, NOR) easiest to implement with CMOS transistors (the switches we have available and use most)

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

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)

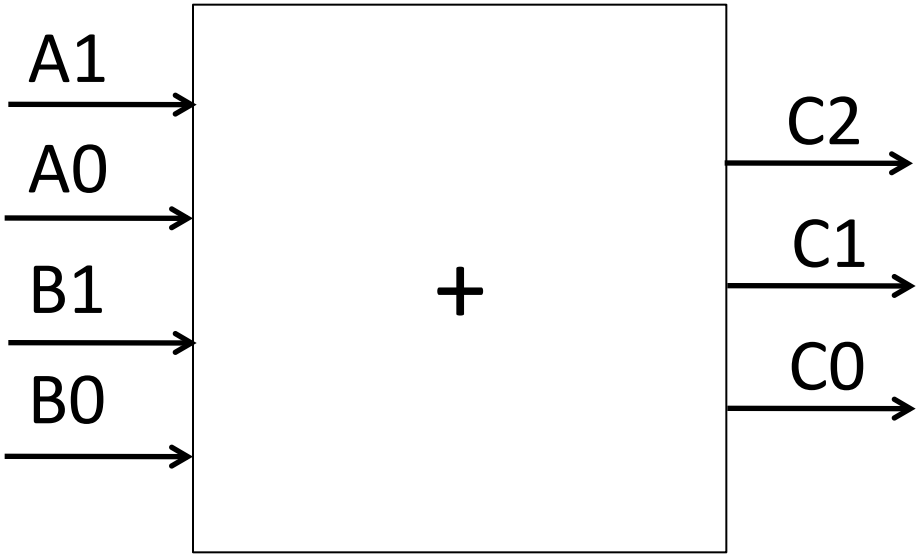
Truth Table Example #1:
y= F(a,b): 1 iff a ≠ b

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_1a_0	b_1b_0	$c_2c_1c_0$

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

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

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)

Boolean Algebra

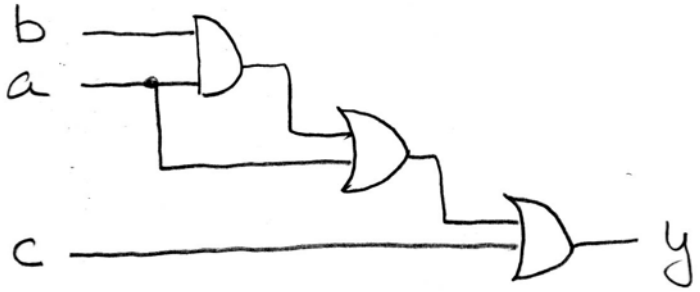
- Use plus "+" for OR
  - "logical sum"
- Use product for AND (a•b or implied via ab)
  - "logical product"
- "Hat" to mean complement (NOT)
- Thus

  $ab + a + \overline{c}$

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

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

Boolean Algebra: Circuit & Algebraic Simplification



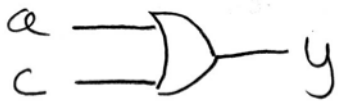
original circuit

$$y = ((ab) + a) + c$$

equation derived from original circuit

$$= ab + a + c$$
$$= a(b + 1) + c$$
$$= a(1) + c$$
$$= a + c$$

algebraic simplification



simplified circuit

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
$(X\ Y)\ Z = Z\ (Y\ Z)$	$(X + Y) + Z = Z + (Y + Z)$	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
$\overline{X}\ Y + X = X + Y$	$(\overline{X} + Y)\ X = X\ Y$	United Theorem v. 2
$\overline{X\ Y} = \overline{X} + \overline{Y}$	$\overline{X + Y} = \overline{X}\ \overline{Y}$	DeMorgan's Law

Boolean Algebraic Simplification Example

$$y = ab + a + c$$

Boolean Algebraic Simplification Example

$$
\begin{aligned}
y &= ab + a + c \\
&= a(b+1) + c && \textit{distribution, identity} \\
&= a(1) + c && \textit{law of 1's} \\
&= a + c && \textit{identity}
\end{aligned}
$$

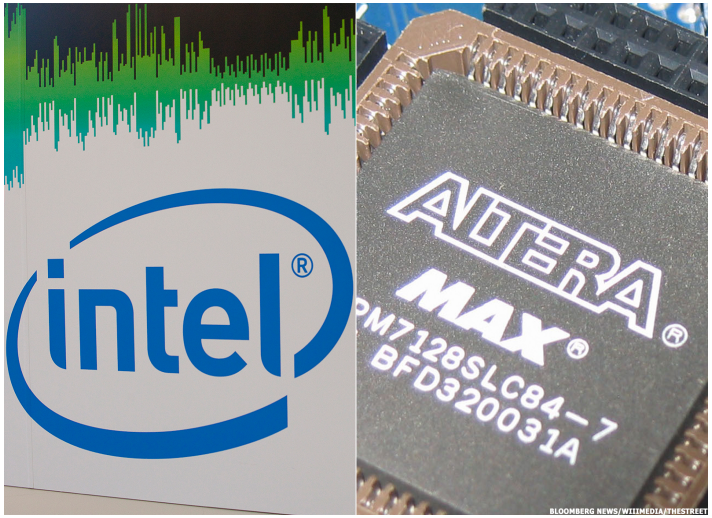
a	b	c	y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Clickers/Peer Instruction

- Simplify $Z = A+BC + \overline{A} \bullet \overline{(BC)}$
- A: $Z = 0$
- B: $Z = A(1+ BC)$
- C: $Z = \overline{(A + BC)}$
- D: $Z = BC$
- E: $Z = 1$

In the News: Intel buys Altera

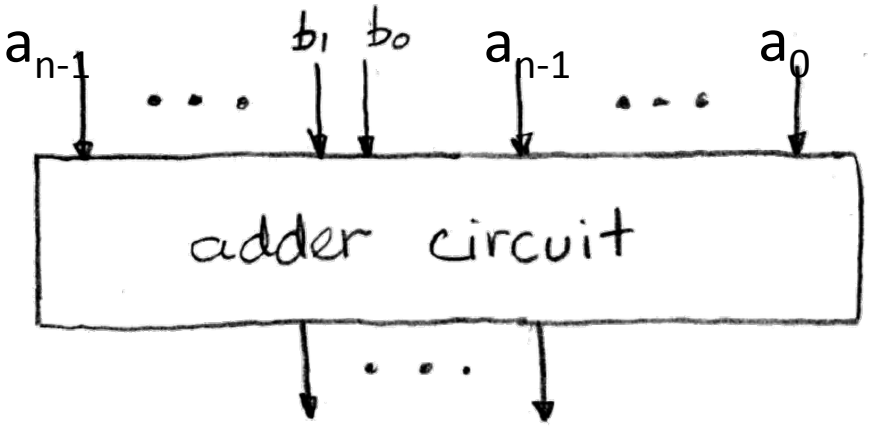
- Intel purchased Altera, an FPGA (Field Programmable Gate Array) company for 16.7 billion
- Goal is to place reconfigurable hardware on Intel server chips
- Take CS150 to learn how to program FPGAs

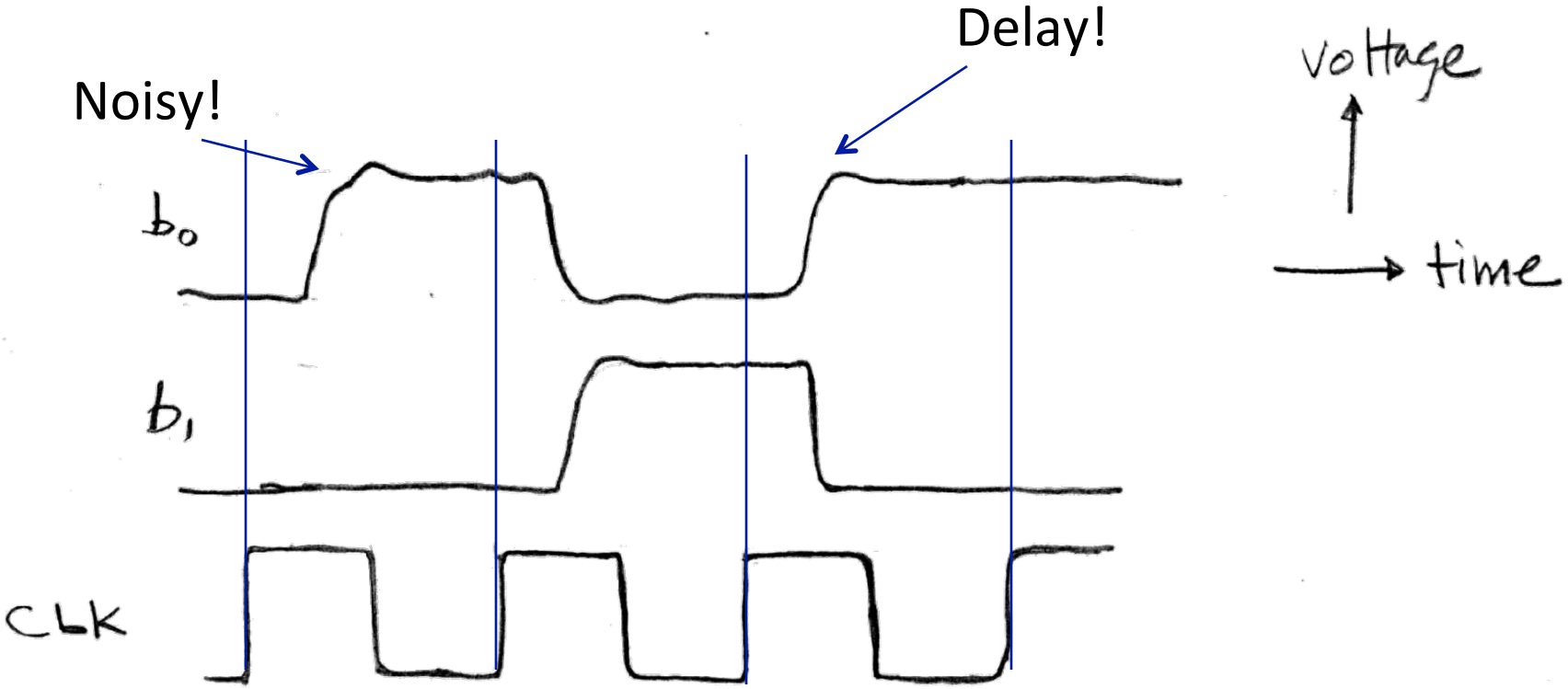


Break

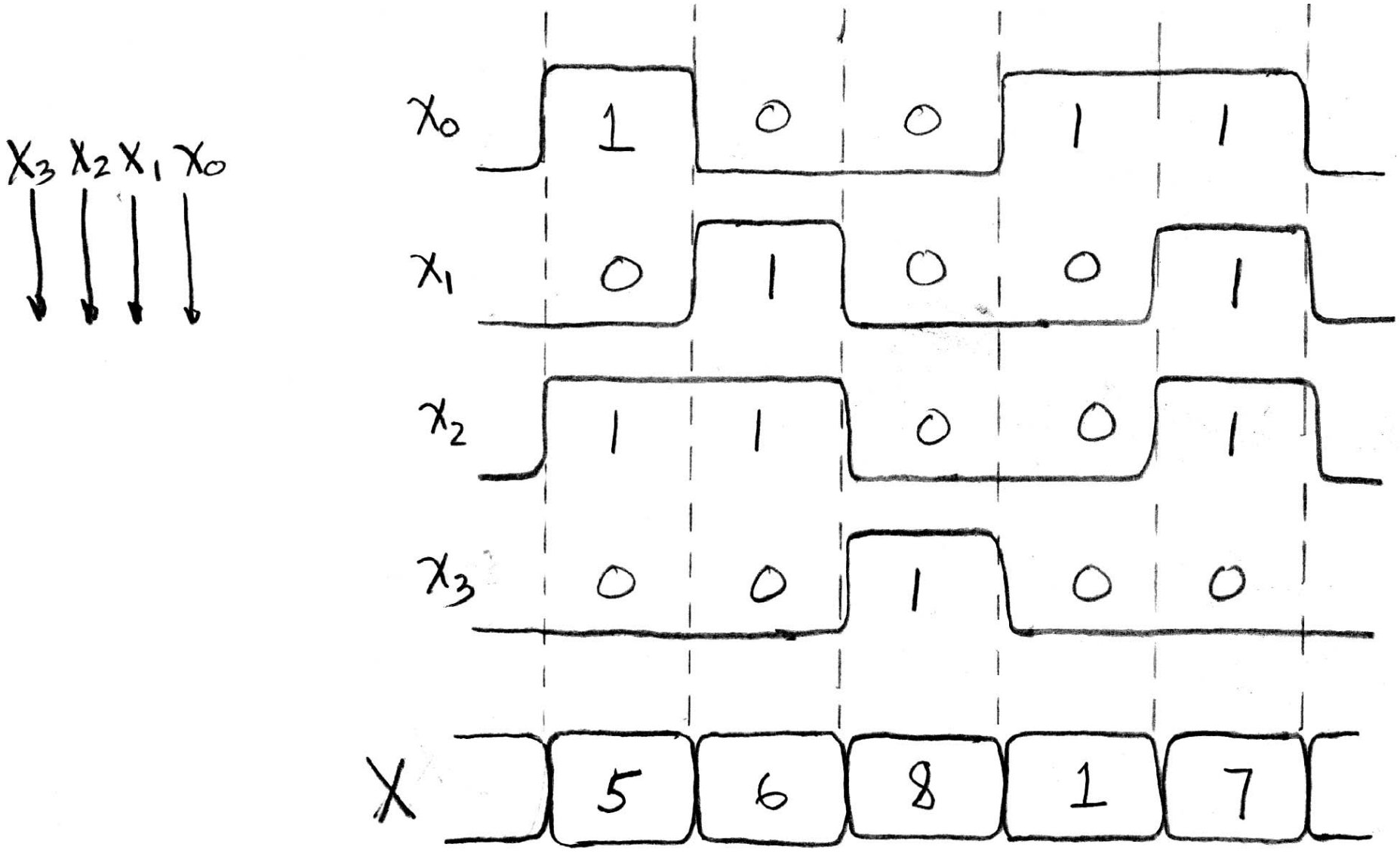
- (Maybe)

Signals and Waveforms

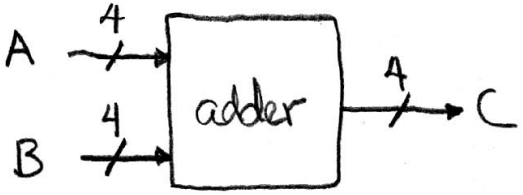




Signals and Waveforms: Grouping

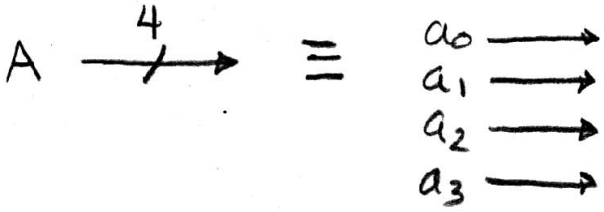


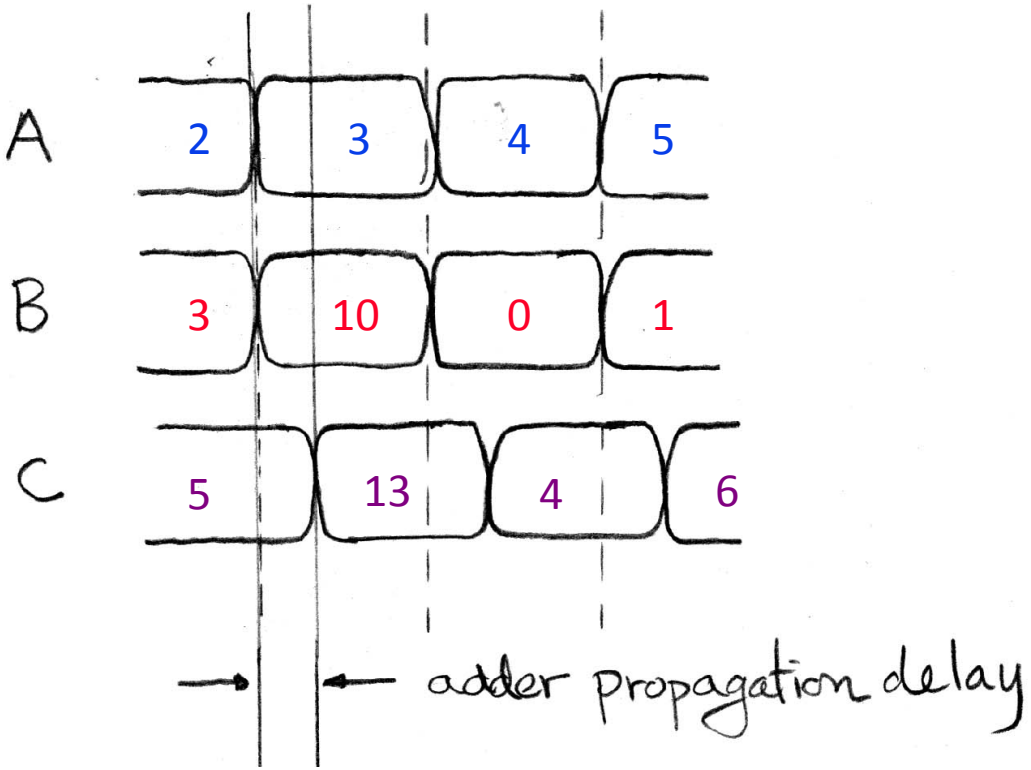
Signals and Waveforms: Circuit Delay



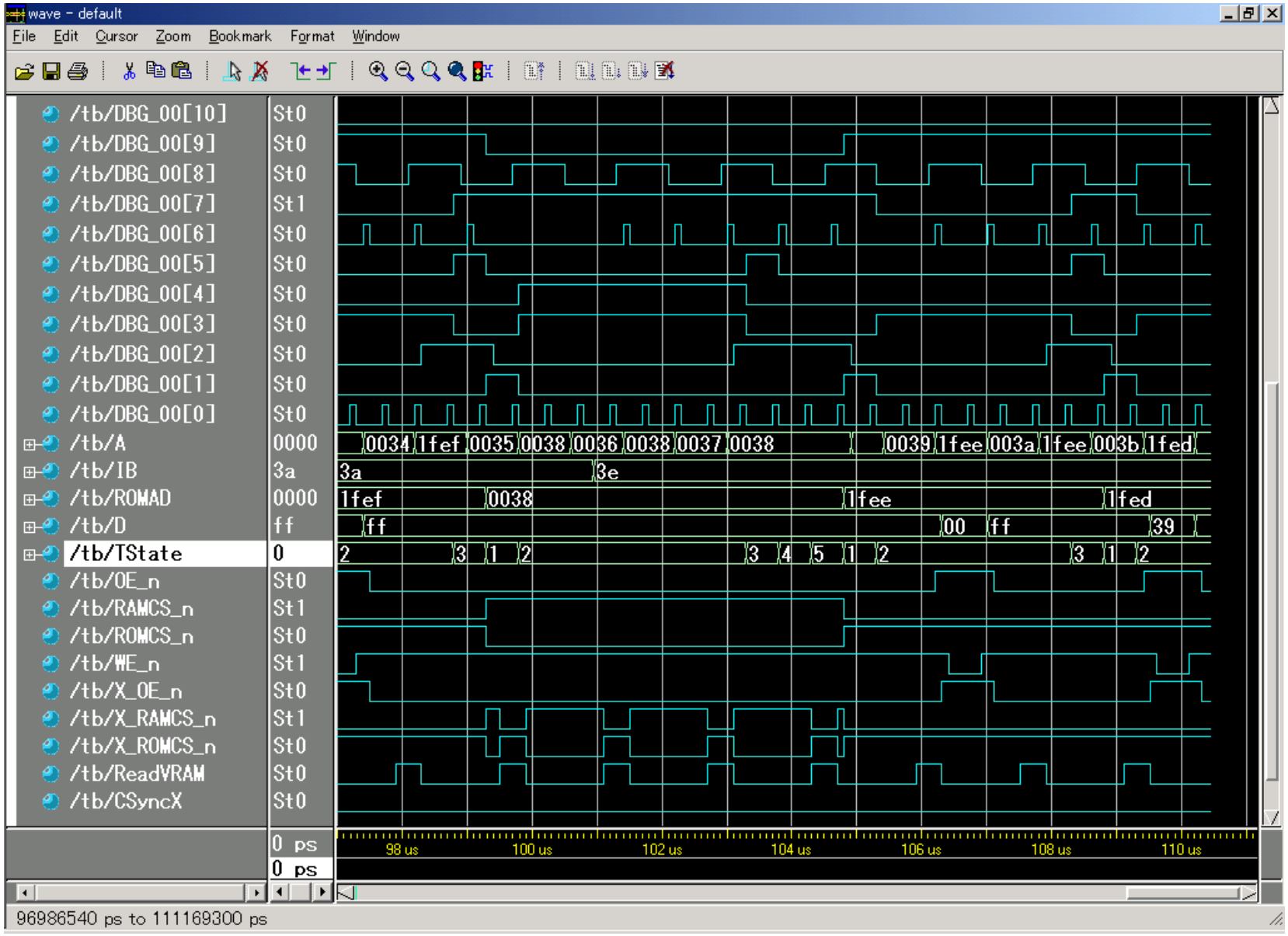
$A = [a_3, a_2, a_1, a_0]$

$B = [b_3, b_2, b_1, b_0]$





Sample Debugging Waveform



Type of Circuits

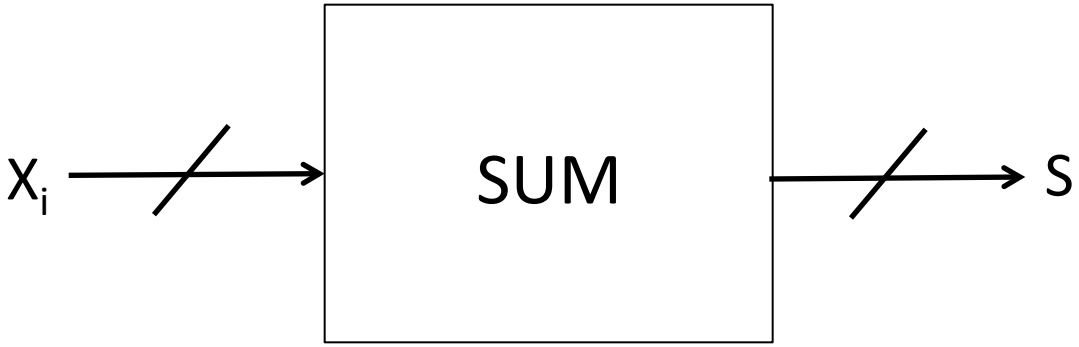
- *Synchronous Digital Systems* consist of two basic types of circuits:
  - Combinational Logic (CL) circuits
    - Output is a function of the inputs only, not the history of its execution
    - E.g., circuits to add A, B (ALUs)
  - Sequential Logic (SL)
    - Circuits that "remember" or store information
    - aka "State Elements"
    - E.g., memories and registers (Registers)

Uses for State Elements

- Place to store values for later re-use:
  - Register files (like $1-$31 in MIPS)
  - Memory (caches and main memory)
- *Help control flow of information between combinational logic blocks*
  - State elements hold up the movement of information at input to combinational logic blocks to allow for orderly passage

Accumulator Example

Why do we need to control the flow of information?



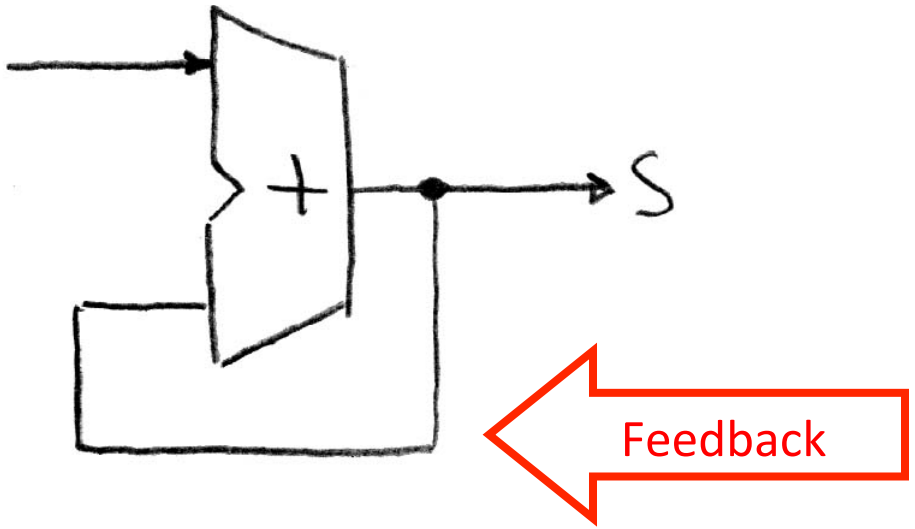
Want:

```
S=0;
for (i=0;i<n;i++)
    S = S + Xi
```

Assume:

- Each X value is applied in succession, one per cycle
- After n cycles the sum is present on S

First Try: Does this work?

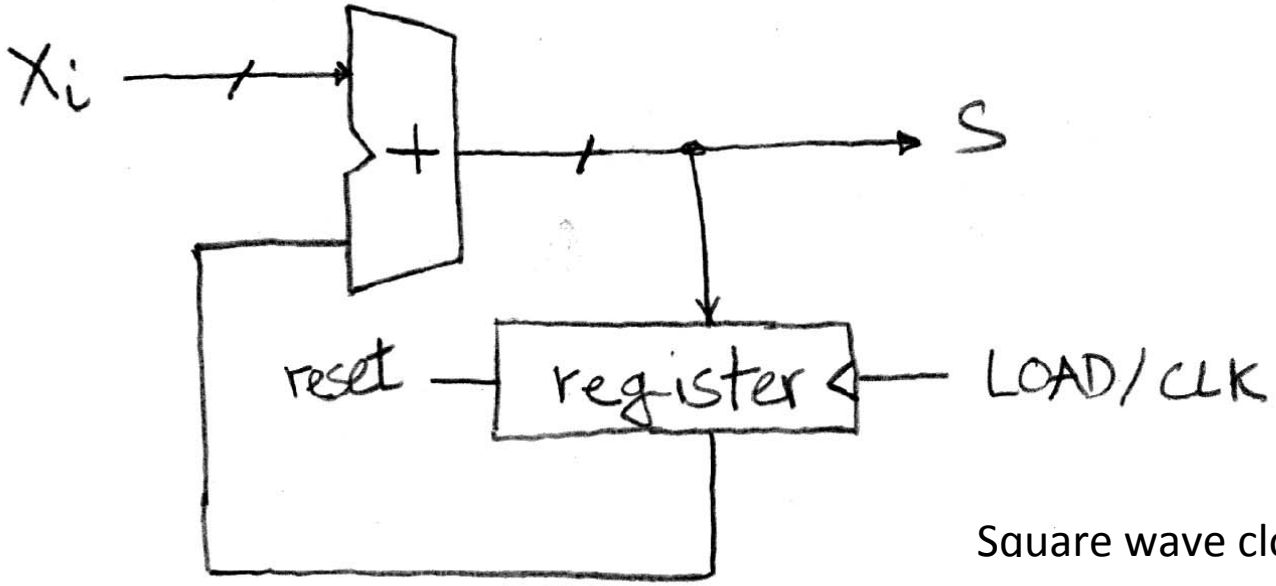


No!

Reason #1: How to control the next iteration of the ‘for’ loop?

Reason #2: How do we say: ‘S=0’?

Second Try: How About This?



Register is used to hold up the transfer of data to adder

Square wave clock sets when things change

Rough timing …

High (1)
LOAD/CLK
Low (0)

Rounded Rectangle per clock means could be 1 or 0

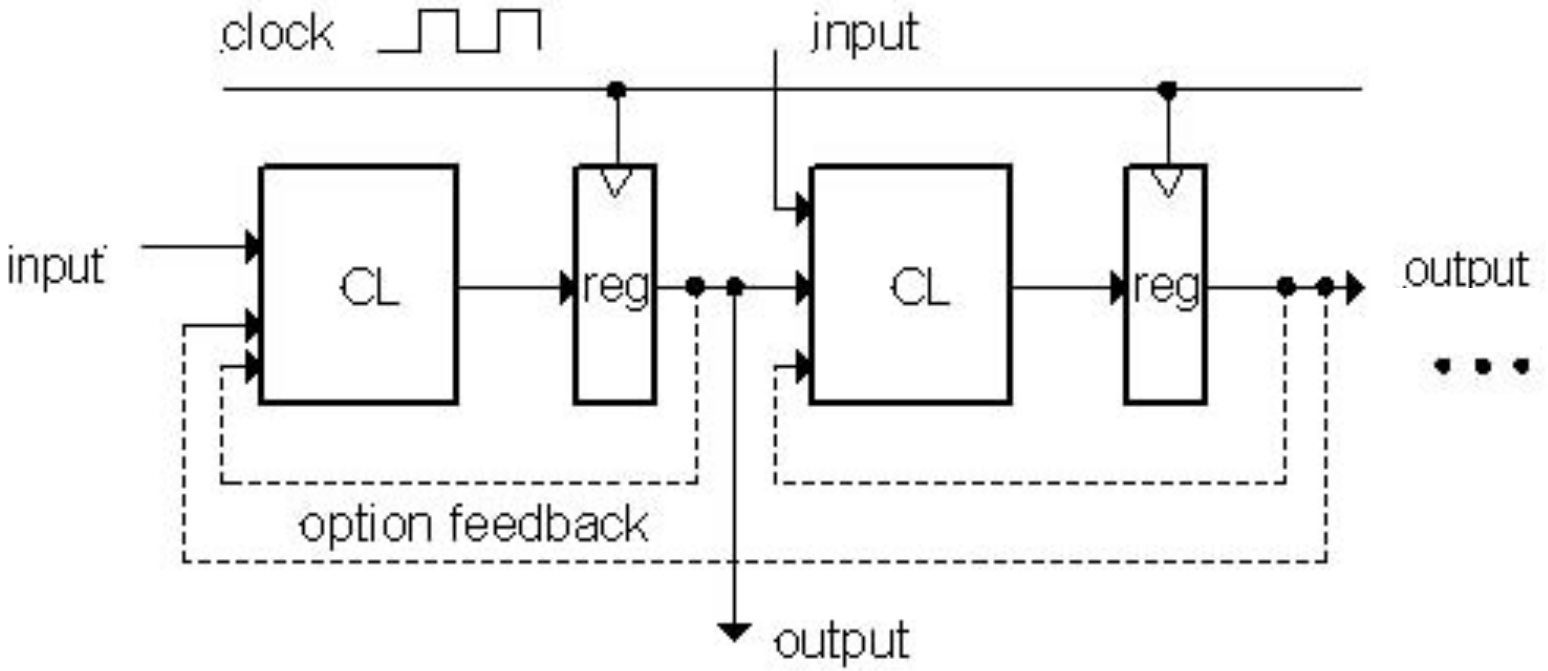
High (1)
S
Low (0)

Xi must be ready **before** clock edge due to adder delay

High (1)
X
Low (0)

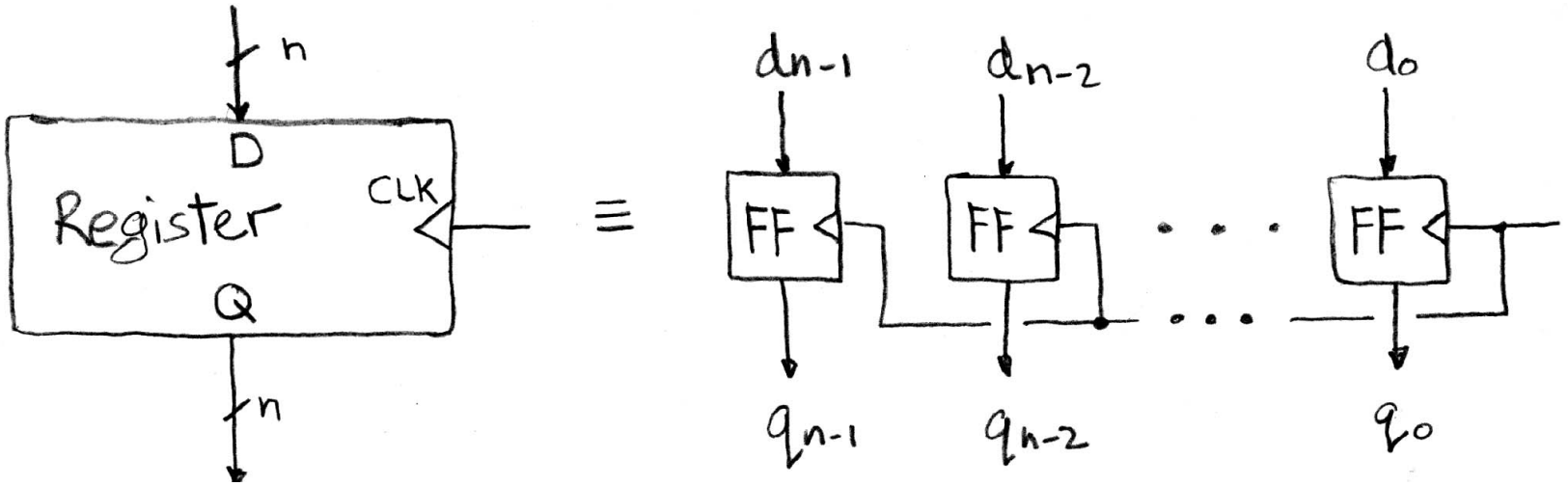
Time

Model for Synchronous Systems



- Collection of Combinational Logic blocks separated by registers
- Feedback is optional
- Clock signal(s) connects only to clock input of registers
- Clock (CLK): steady square wave that synchronizes the system
- Register: several bits of state that samples on rising edge of CLK (positive edge-triggered) or falling edge (negative edge-triggered)

Register Internals



- n instances of a "Flip-Flop"
- Flip-flop name because the output flips and flops between 0 and 1
- D is "data input", Q is "data output"
- Also called "D-type Flip-Flop"

Camera Analogy Timing Terms

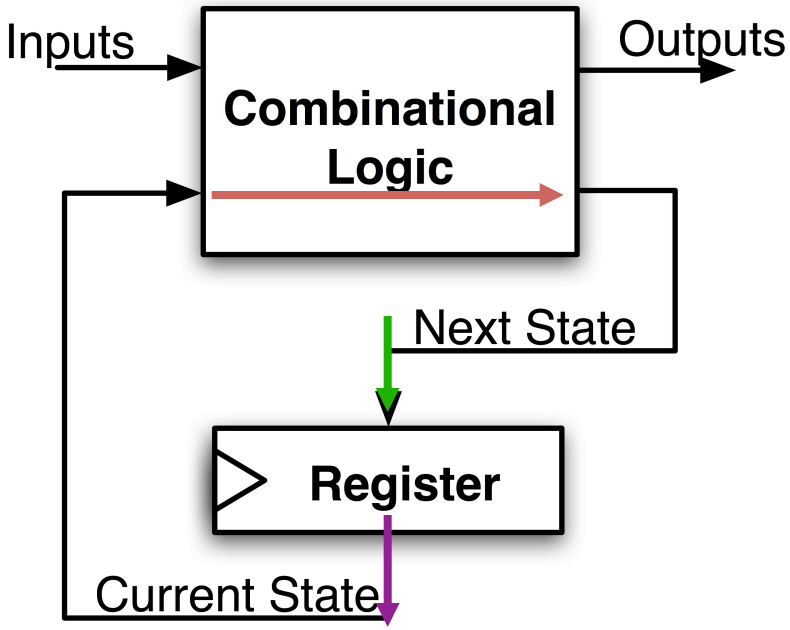
- Want to take a portrait – timing right before and after taking picture
- *Set up time* – don't move since about to take picture (open camera shutter)
- *Hold time* – need to hold still after shutter opens until camera shutter closes
- *Time click to data* – time from open shutter until can see image on output (viewscreen)

Hardware Timing Terms

- Setup Time: when the input must be stable *before* the edge of the CLK
- Hold Time: when the input must be stable *after* the edge of the CLK
- "CLK-to-Q" Delay: how long it takes the output to change, measured from the edge of the CLK

Maximum Clock Frequency

- What is the maximum frequency of this circuit?



Hint:
Frequency = 1/Period

Max Delay = Setup Time + CLK-to-Q Delay + CL Delay

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
- State Machines
  - Finite State Machines: made from *Stateless* combinational logic and *Stateful* "Memory" Logic (aka Registers)
  - Clocks synchronize D-FF change (Setup and Hold times important!)
  - More about these next time