CS 110
Computer Architecture
Lecture 10:
Synchronous Digital Systems

Instructors:
Sören Schwertfeger & Chundong Wang

School of Information Science and Technology SIST

ShanghaiTech University

Slides based on UC Berkley's CS61C

Levels of Representation/Interpretation

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

Compiler

Assembly Language Program (e.g., RISC-V)

Machine Language Program (RISC-V)

Machine Interpretation

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

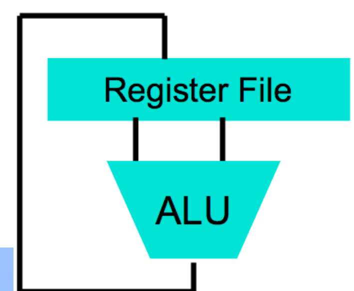
Architecture Implementation

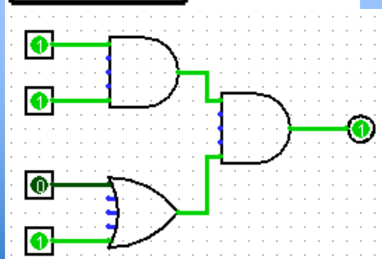
Logic Circuit Description (Circuit Schematic Diagrams)

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

```
lw    xt0, 0(x2)
lw    xt1, 4(x2)
sw    xt1, 0(x2)
sw    xt0, 4(x2)
```

```
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
```





You are Here!

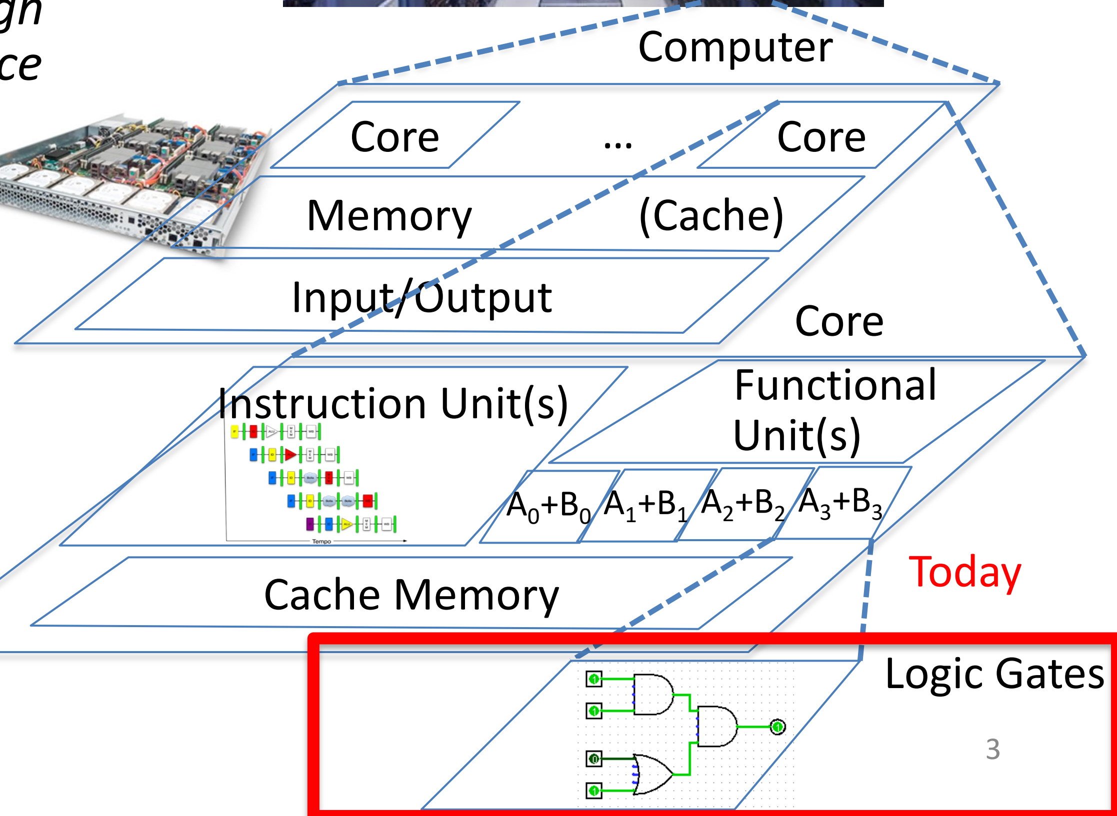
Software | *Hardware*

- 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



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
  - 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!
    - E.g. Google Tensor Processing Unit (TPU)

Synchronous Digital Systems

Hardware of a processor, such as the RISC-V, is an example of a Synchronous Digital System

Synchronous:

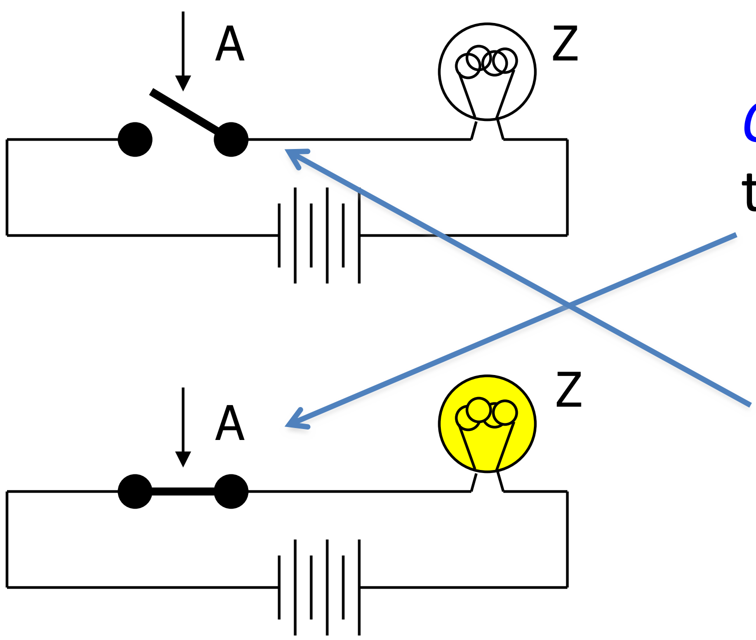
- All operations coordinated by a central clock
  - "Heartbeat" of the system!

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)

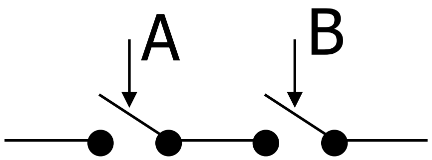
Off-switch (if A is "0" or unasserted) turns-off light bulb (Z)

$$Z \equiv A$$

Switches (cont'd)

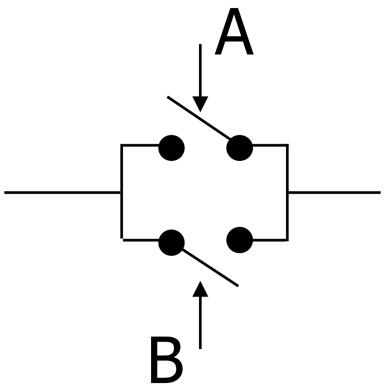
- Compose switches into more complex ones (Boolean functions):

AND



Z ≡ A <u>and</u> B

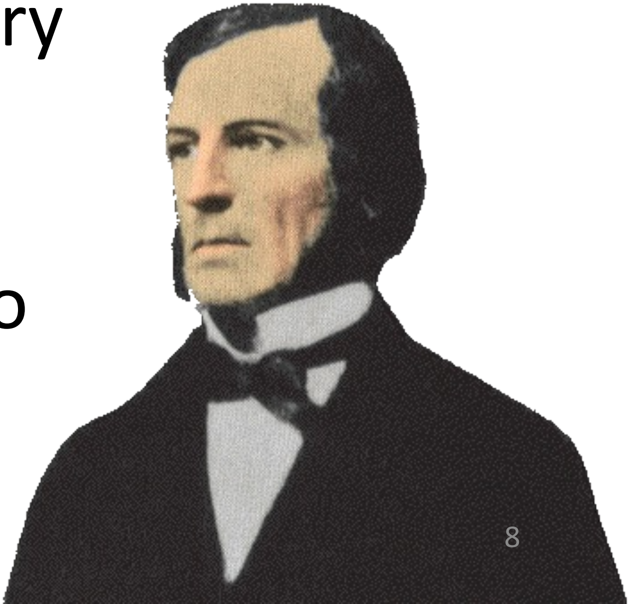
OR



Z ≡ A <u>or</u> 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, …



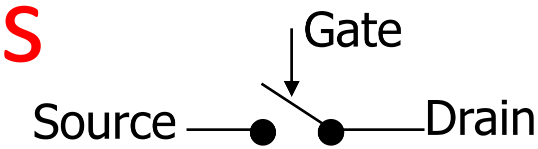
Transistors

- 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

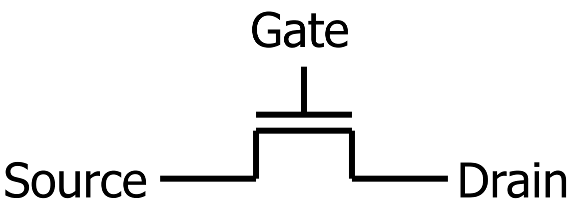
CMOS Transistor Networks

- 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)



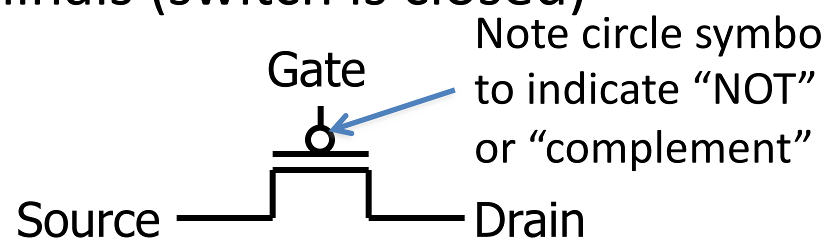
n-channel transitor

off when voltage at Gate is low

on when:

voltage (Gate) > voltage (Threshold)

(**High** resistance when gate voltage **Low**,
Low resistance when gate voltage **High**)



p-channel transistor

on when voltage at Gate is low

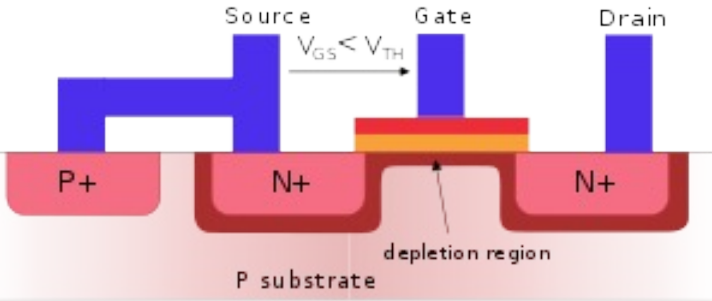
off when:

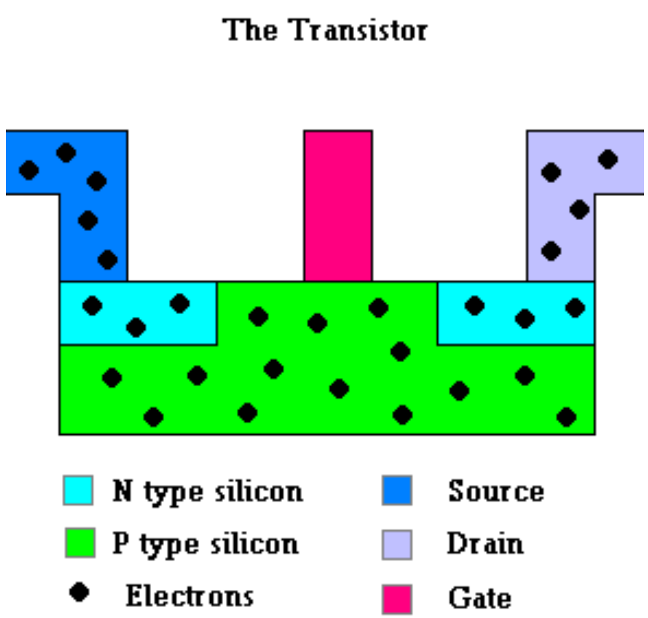
voltage (Gate) > voltage (Threshold)

(**Low** resistance when gate voltage **Low**,
High resistance when gate voltage **High**)

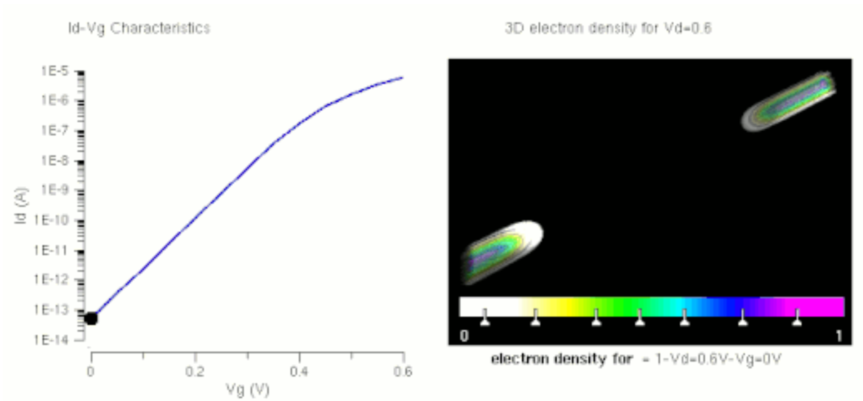
Field-Effect Transistor (FET) => CMOS circuits use a combination of p-type and n-type metal–oxide–semiconductor field-effect transistors => MOSFET

MOSFET Operation





http://ecampus.matc.edu/mihalj/scitech/unit4/transistors/transistors.htm



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

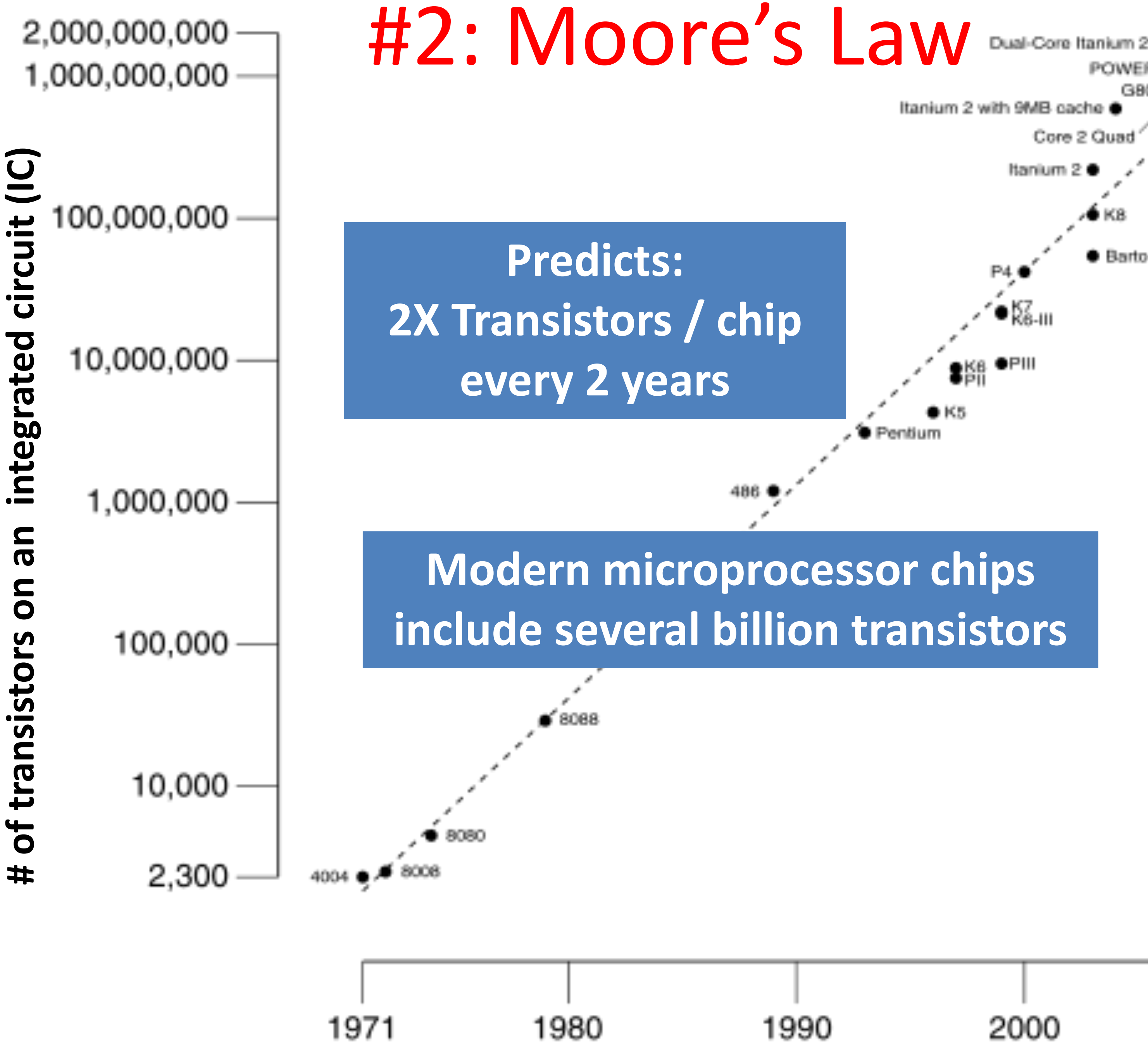
MOSFET scaling
(process nodes)

- 10 μm – 1971
- 6 μm – 1974
- 3 μm – 1977
- 1.5 μm – 1981
- 1 μm – 1984
- 800 nm – 1987
- 600 nm – 1990
- 350 nm – 1993
- 250 nm – 1996
- 180 nm – 1999
- 130 nm – 2001
- 90 nm – 2003
- 65 nm – 2005
- 45 nm – 2007
- 32 nm – 2009
- 22 nm – 2012
- 14 nm – 2014
- 10 nm – 2016
- 7 nm – 2018
- 5 nm – *~2020*

Future

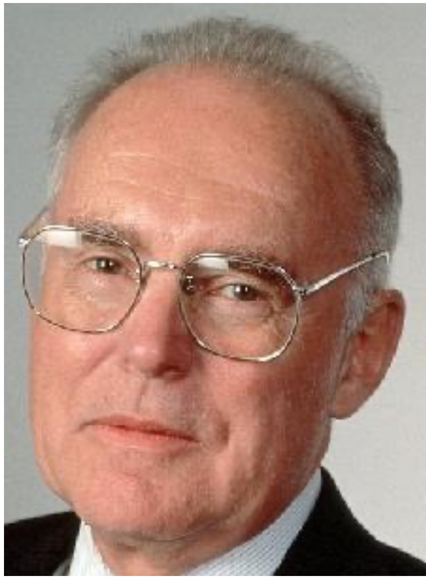
- 3 nm – *~2021*
- 2 nm – *~2024*

#2: Moore's Law



Predicts:
2X Transistors / chip every 2 years

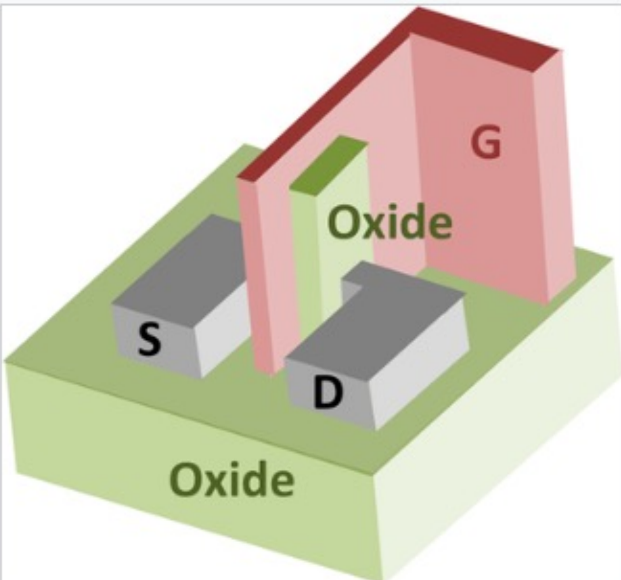
Modern microprocessor chips include several billion transistors



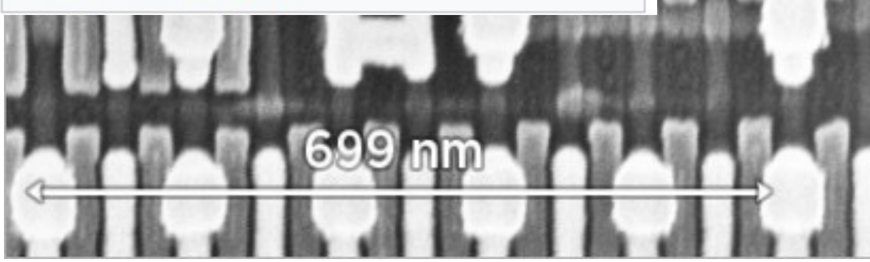
Gordon Moore
Intel Cofounder

Intel 14nm Technology

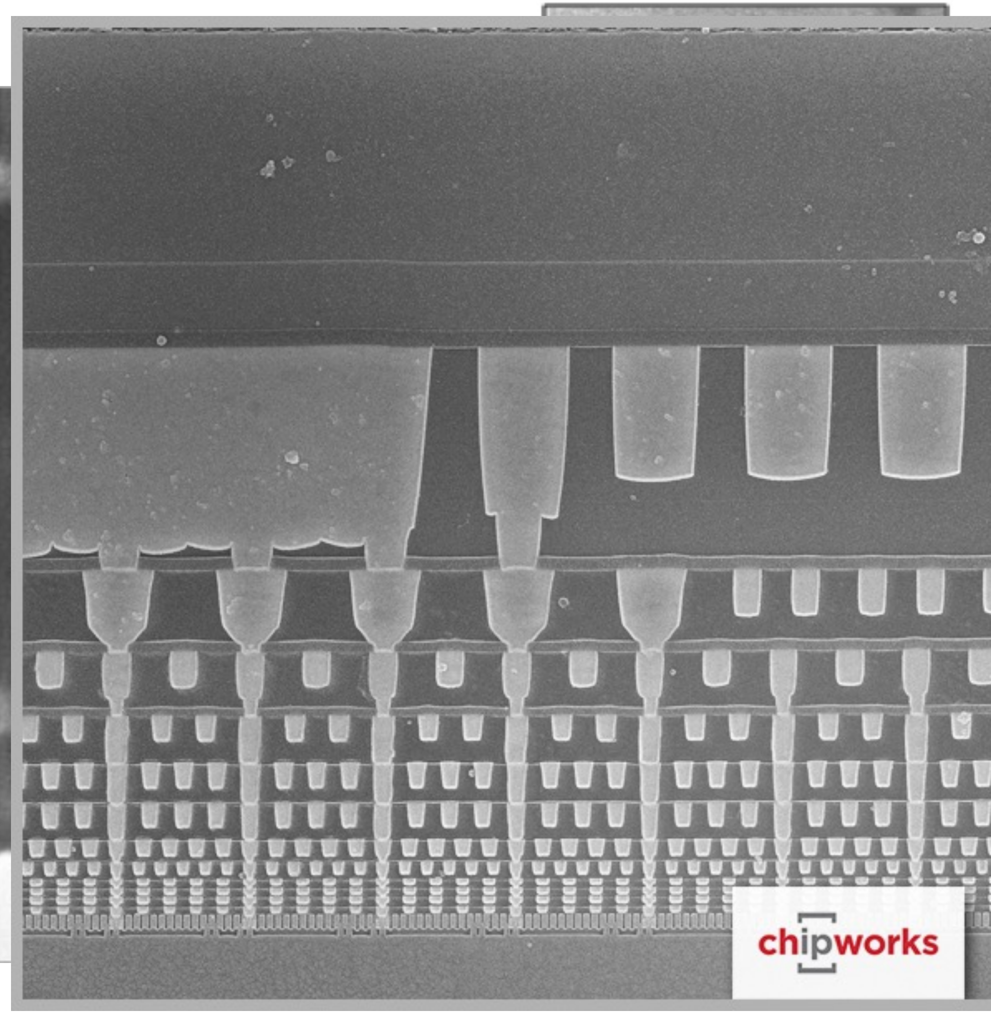
1 nm = 1 / 1,000,000,000 m; wavelength visible light: 400 – 700 nm



A FinFET (fin field-effect transistor), a type of multi-gate MOSFET.

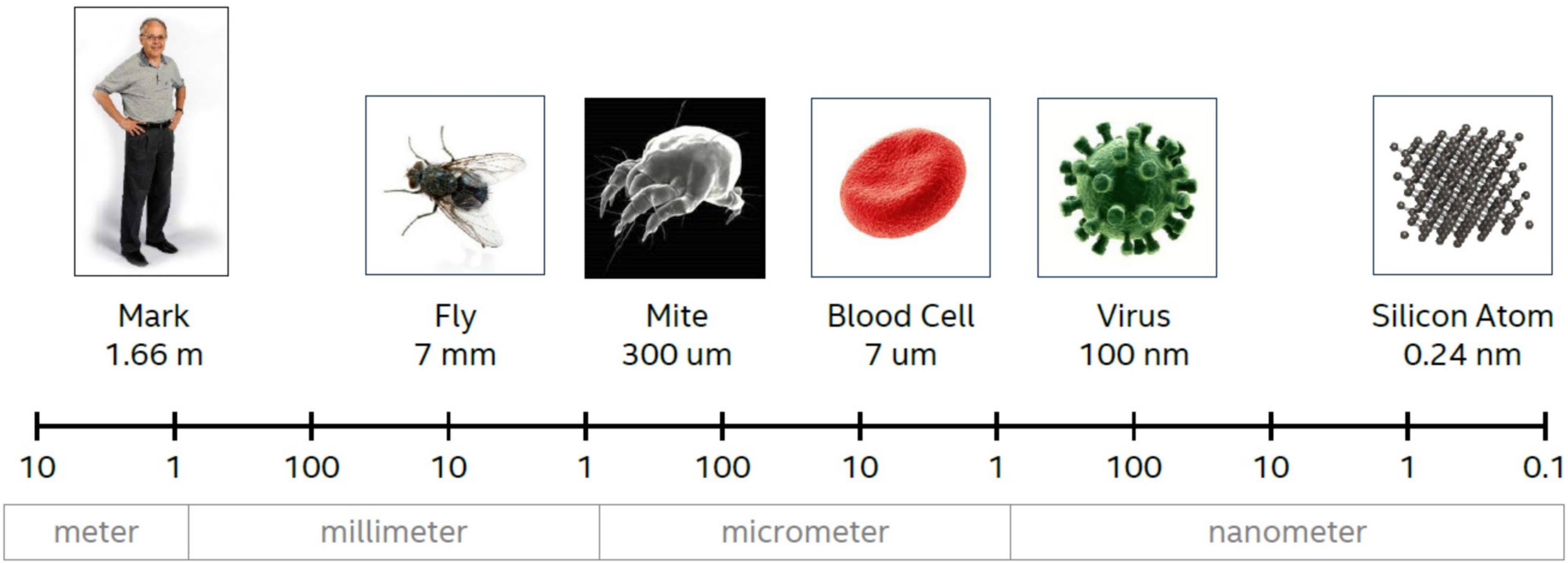


Plan view of transistors



Side view of wiring layers

Sense of Scale



1 nm = 1 / 1,000,000,000 m; wavelength visible light: 400 – 700 nm

14nm about 58 Silicon Atoms …

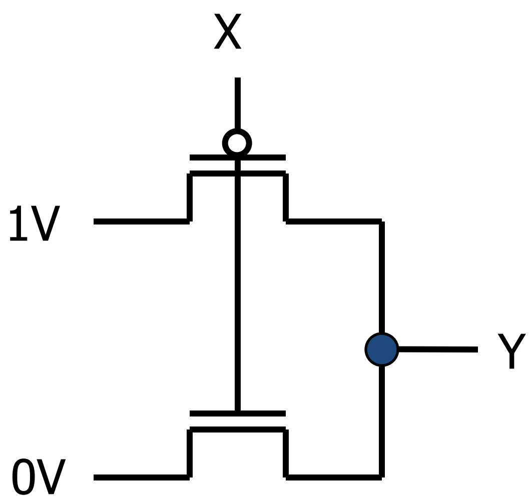
Source: Mark Bohr, IDF14

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)



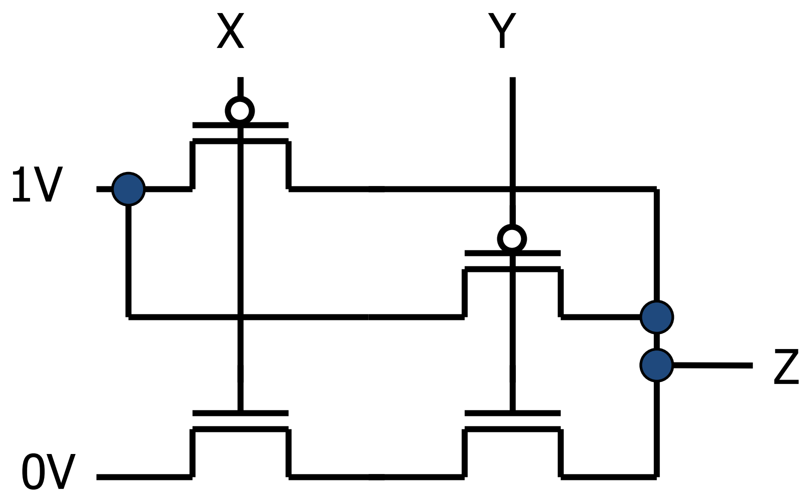
n-channel transitor
off when voltage at Gate is low
on when:
voltage(Gate) > voltage (Threshold)

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

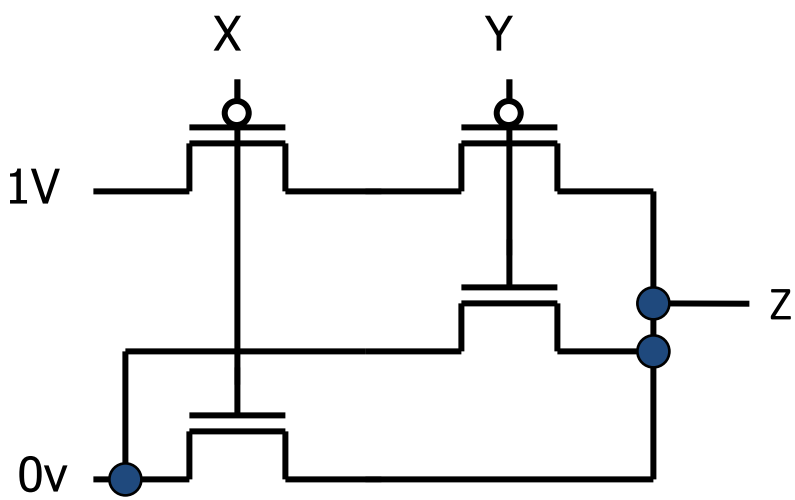


what is the
relationship between x, y and z?

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)

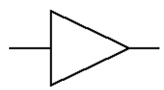
Question



X	Y	Z: A	Z: B	Z: C	Z: 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

Combinational Logic Symbols

- Common combinational logic systems have standard symbols called logic gates
  - Buffer, NOT

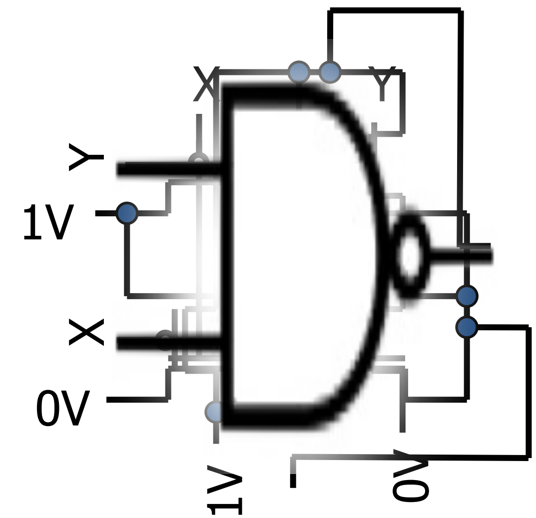
A  Z 

  - AND, NAND

A
B  Z 

  - OR, NOR

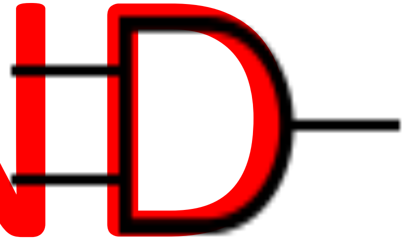
A
B  Z 



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

Remember…

- AND



- OR

Boolean Algebra

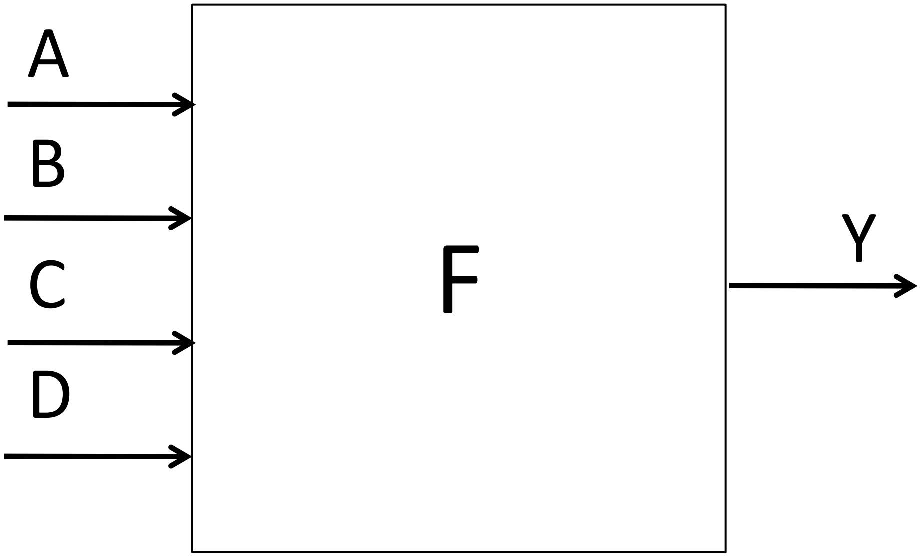
- Use plus “+” for OR
  - “logical sum” 1+0 = 0+1 = 1 (True); 1+1=2 (True); 0+0 = 0 (False)
- Use product for AND (a•b or implied via ab)
  - “logical product” 0*0 = 0*1 = 1*0 = 0 (False); 1*1 = 1 (True)
- “Hat” to mean complement (NOT)
- Thus

$ab + a + \overline{c}$

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

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

Truth Tables for Combinational Logic



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

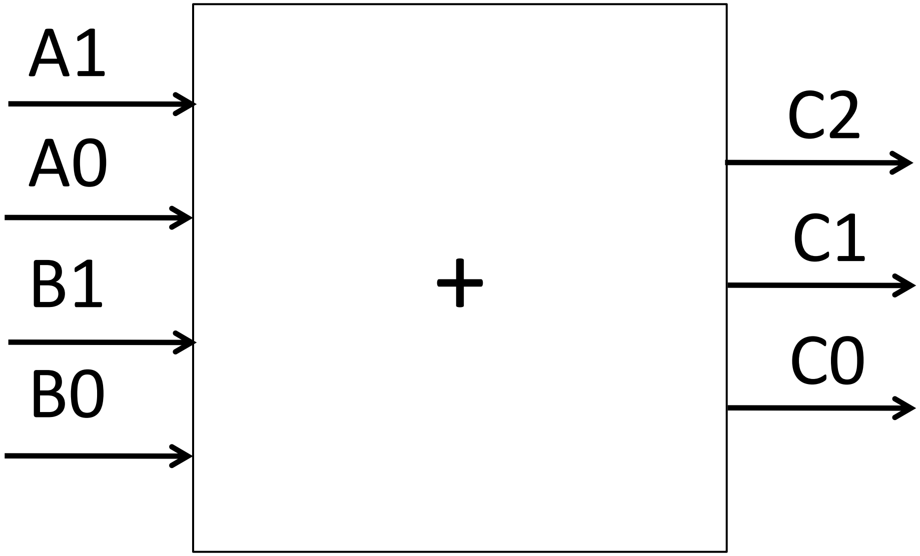
How many logic functions can be defined with N inputs?

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



How Many Rows?

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

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

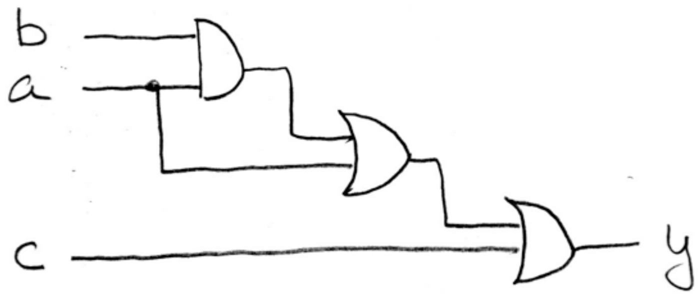
Y =

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

Boolean Algebra: Circuit & Algebraic Simplification



original circuit

$$\downarrow$$

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

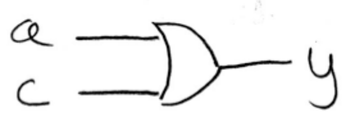
equation derived from original circuit

$$\downarrow$$

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

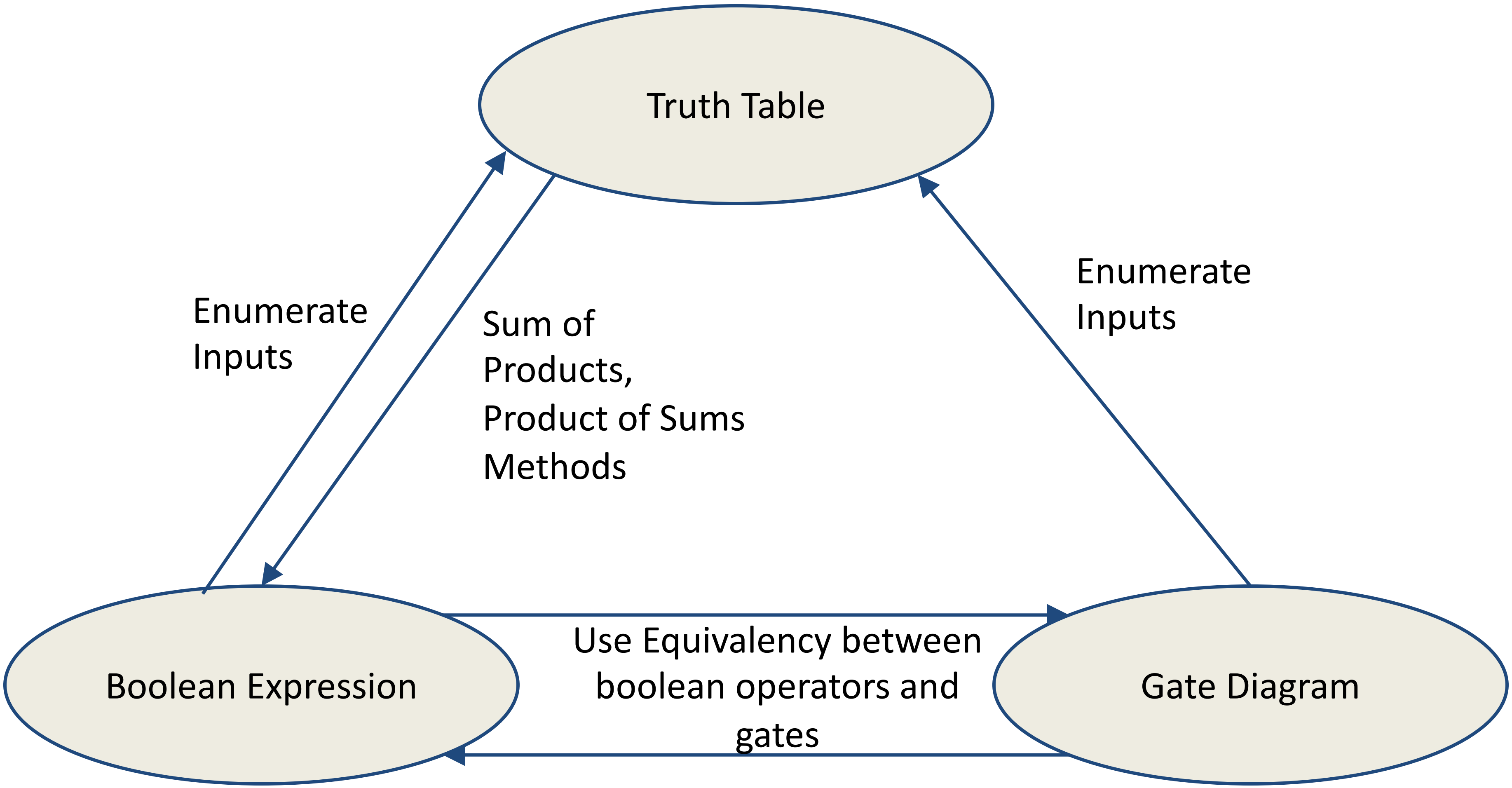
algebraic simplification

$$\downarrow$$



simplified circuit

Representations of Combinational Logic (groups of logic gates)



Laws of Boolean Algebra

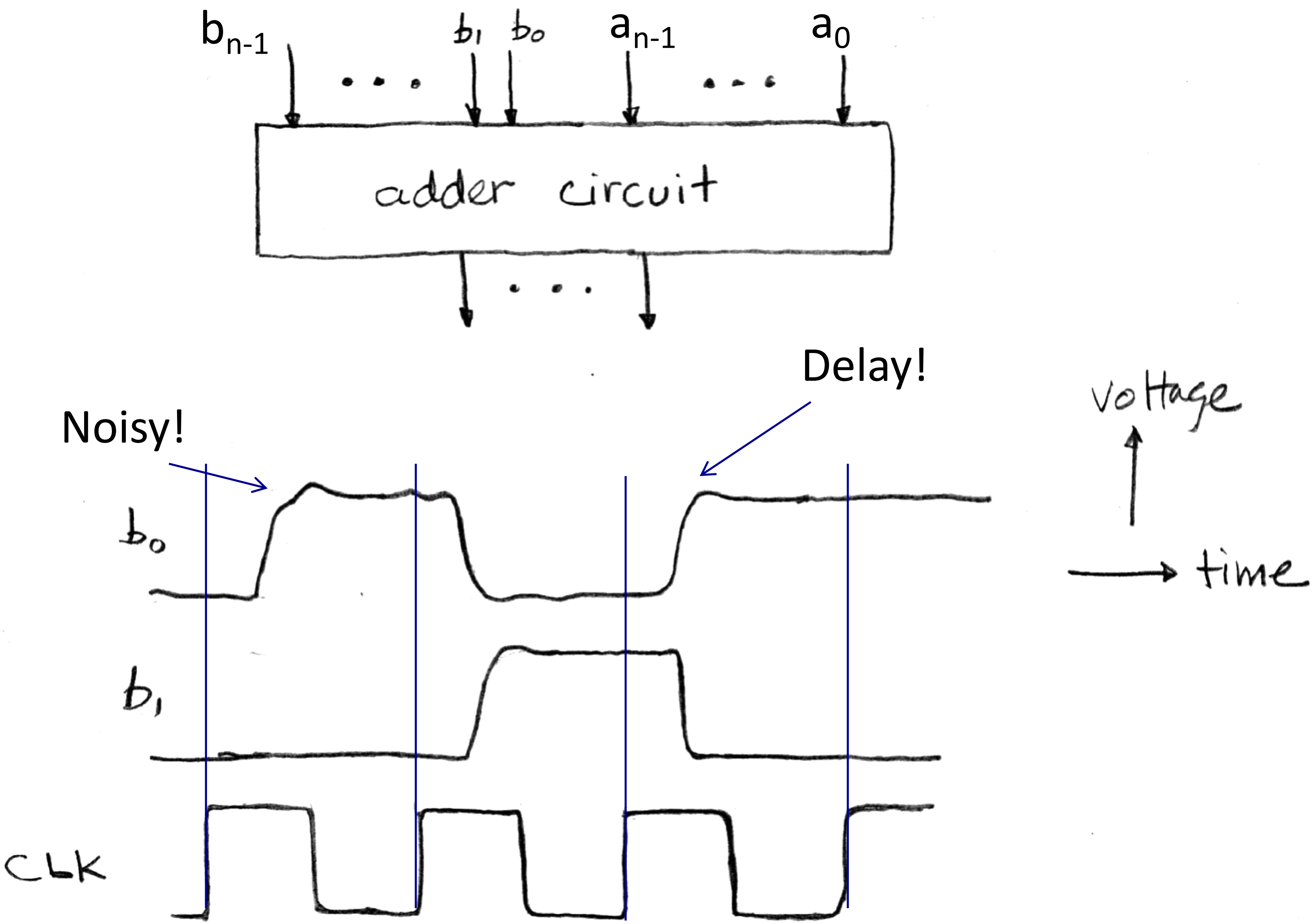
$X \overline{X} = 0$	$X + \overline{X} = 1$	Complementarity
$X\,0 = 0$	$X + 1 = 1$	Laws of 0's and 1's
$X\,1 = X$	$X + 0 = X$	Identities
$X\,X = X$	$X + X = X$	Idempotent Laws
$X\,Y = Y\,X$	$X + Y = Y + X$	Commutativity
$(X\,Y)\,Z = X\,(Y\,Z)$	$(X + Y) + Z = X + (Y + Z)$	Associativity
$X\,(Y + Z) = X\,Y + X\,Z$	$X + Y\,Z = (X + Y)\,(X + Z)$	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

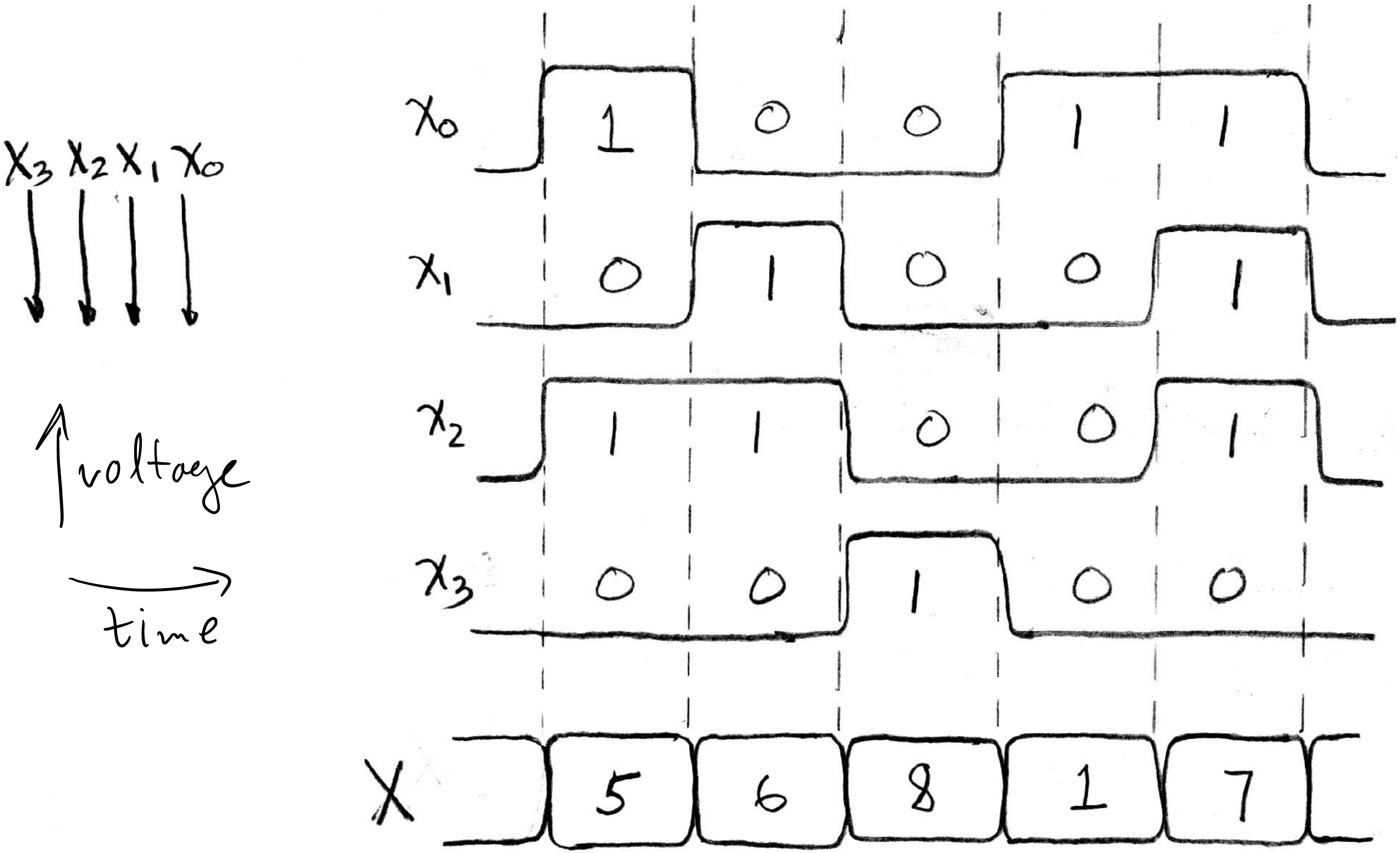
$$y = ab + a + c$$

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

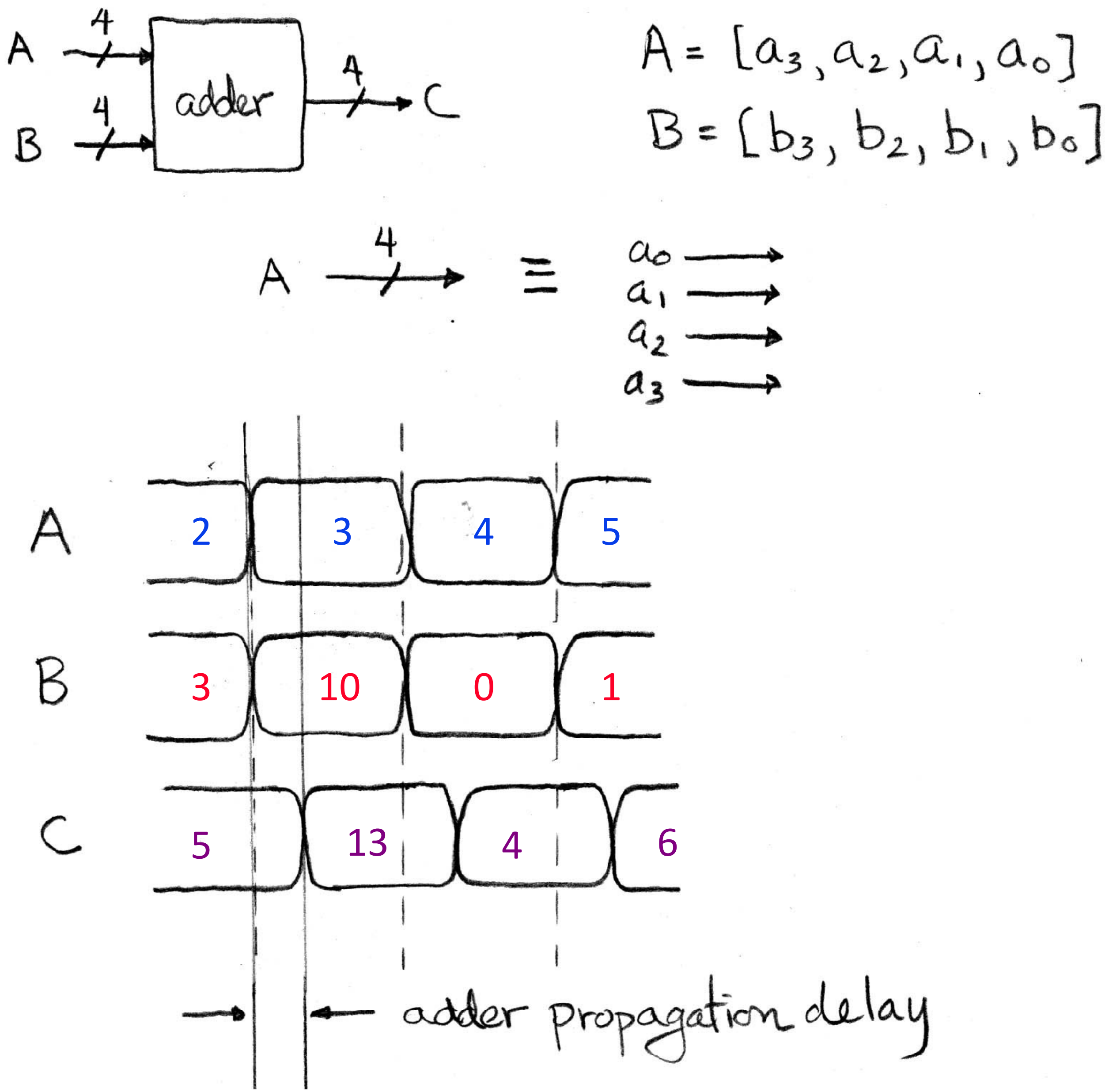
Signals and Waveforms



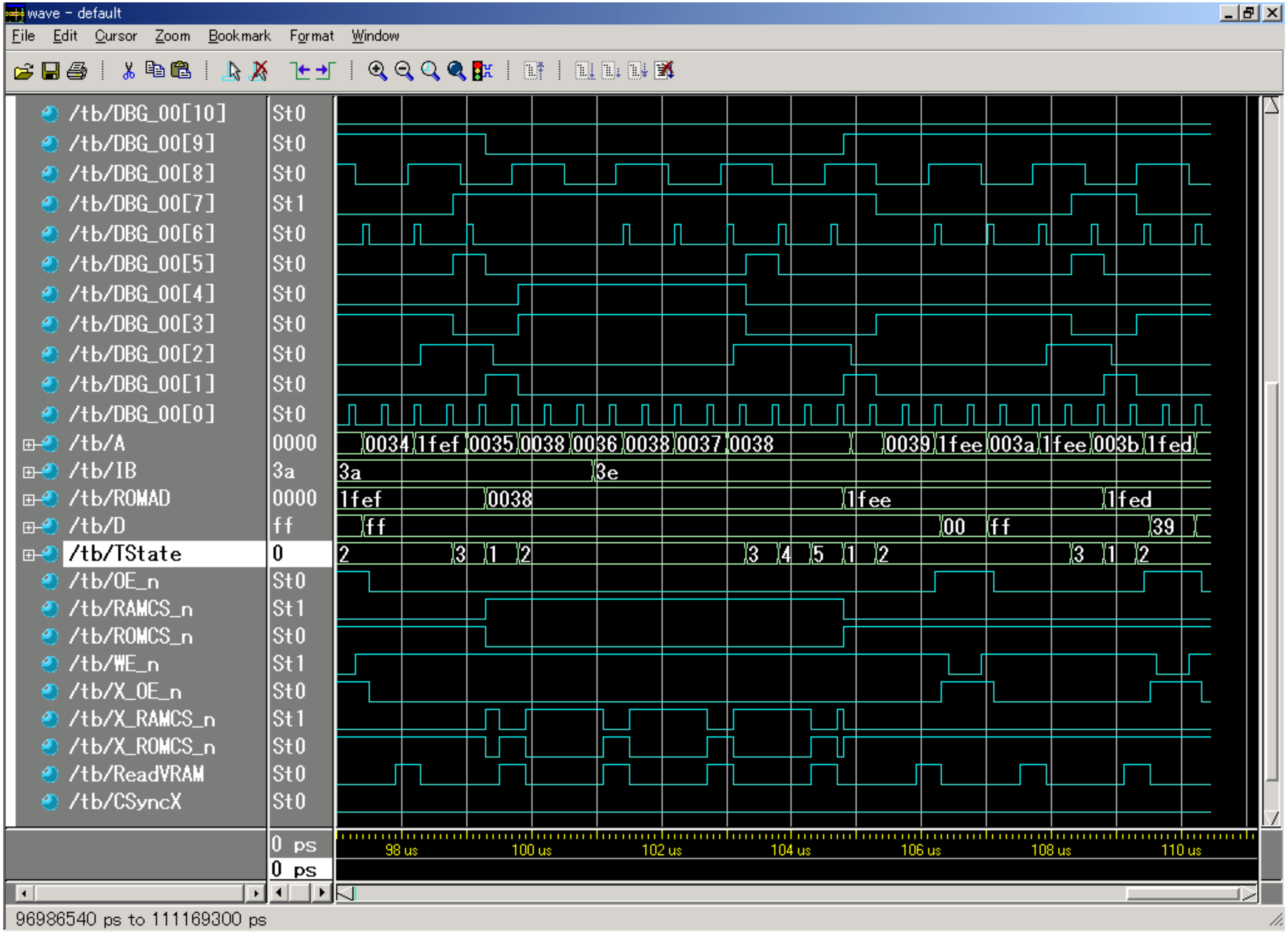
Signals and Waveforms: Grouping



Signals and Waveforms: Circuit Delay



Sample Debugging Waveform



Type of Circuits

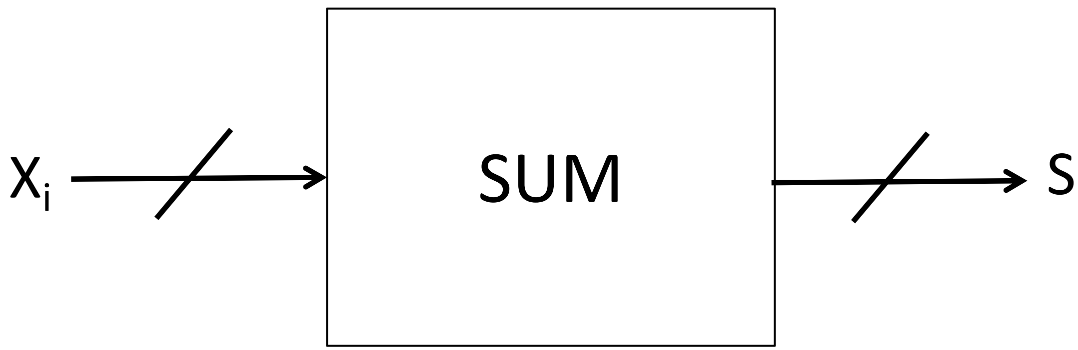
- *Synchronous Digital Systems* consist of two basic types of circuits:
  - Combinational Logic (CL) circuits
    - Output is a function of the inputs only, not the history of its execution
    - E.g., circuits to add A, B (ALUs)
  - Sequential Logic (SL)
    - Circuits that “remember” or store information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

Uses for State Elements

- Place to store values for later re-use:
  - Register files (like x1-x31 in RISC-V)
  - 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?



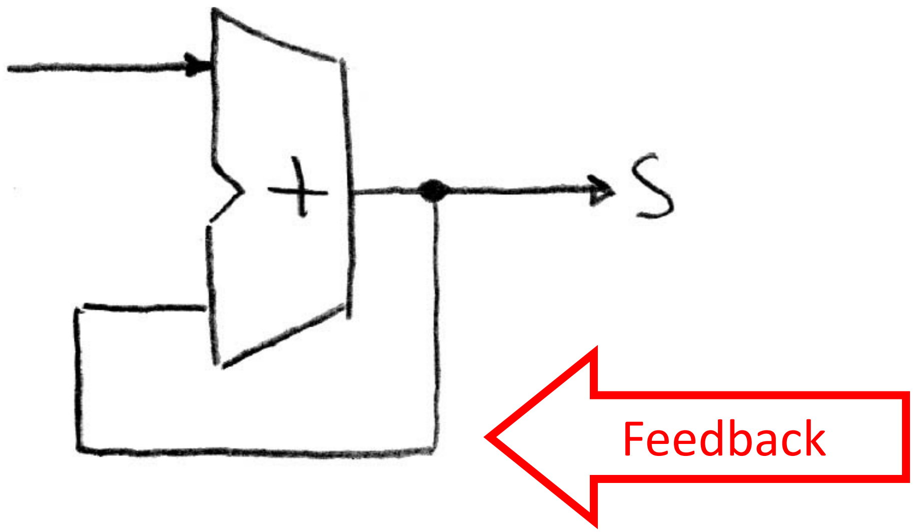
Want:

```
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?

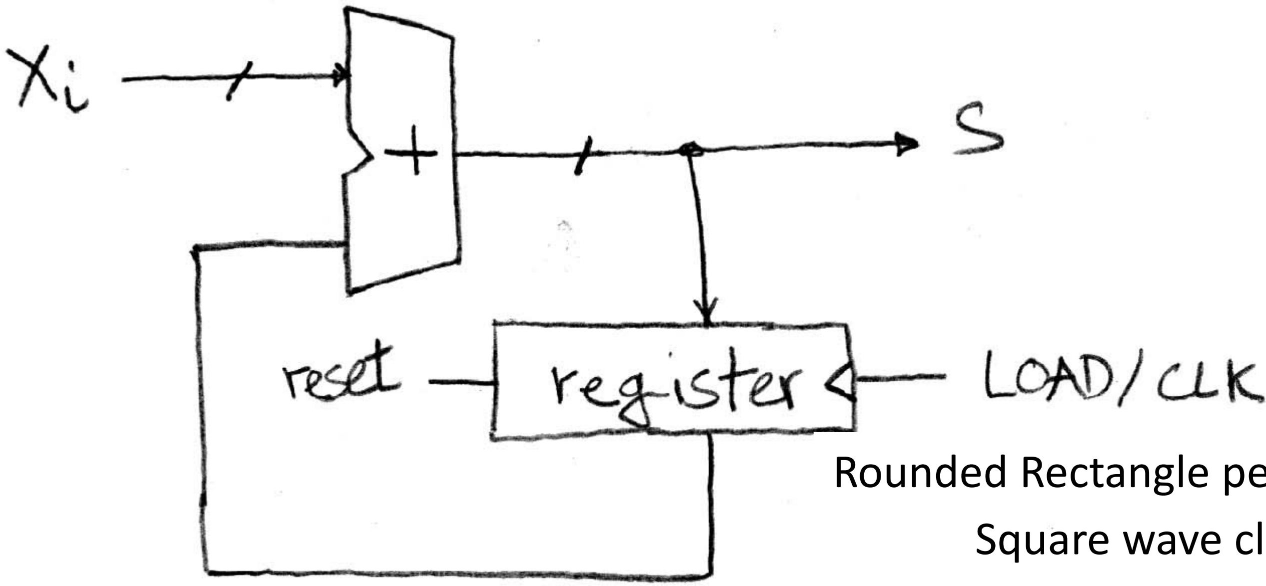


No!

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

Rounded Rectangle per clock means could be 1 or 0

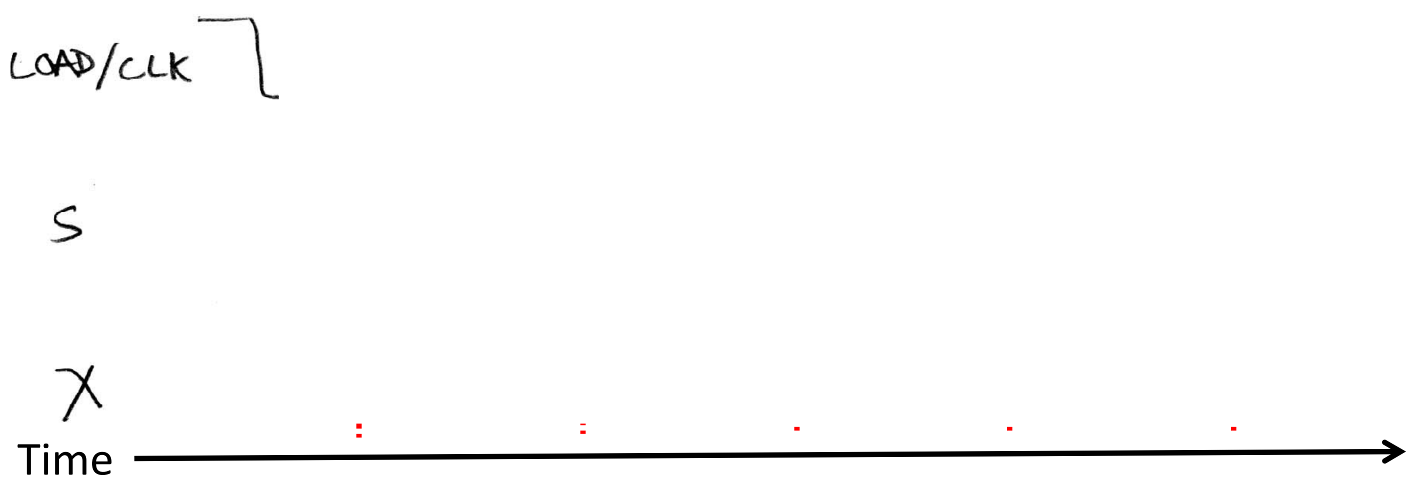
Square wave clock sets when things change

Rough timing …

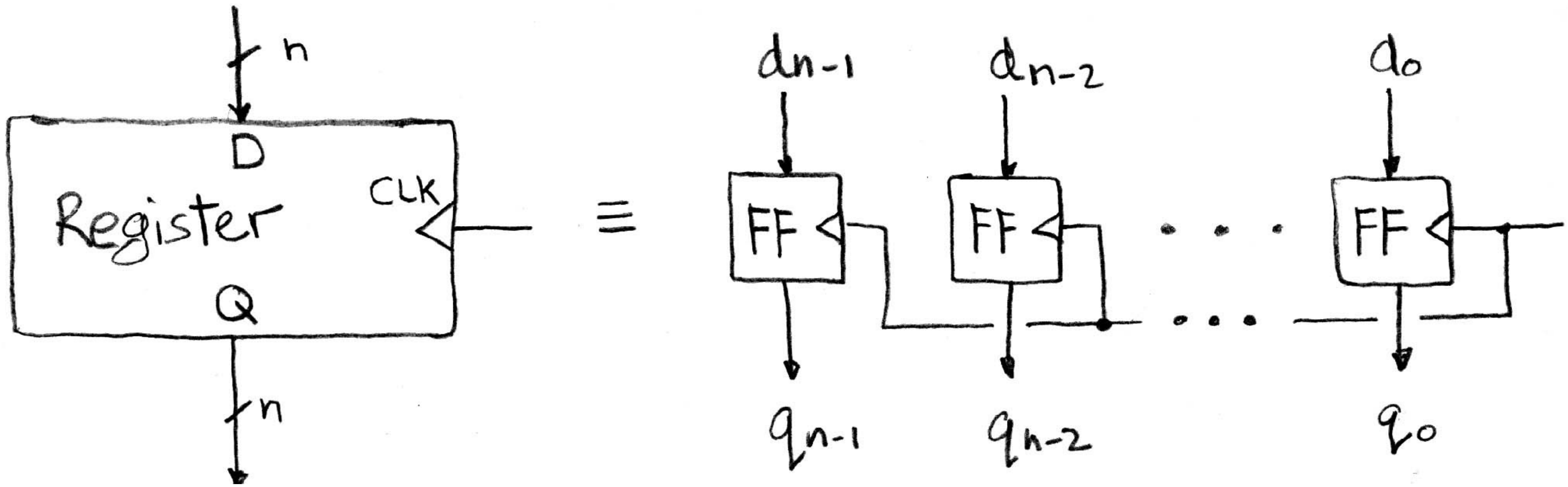
High (1)
Low (0)

High (1)
Low (0)

High (1)
Low (0)



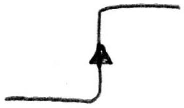
Register Internals



- 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"

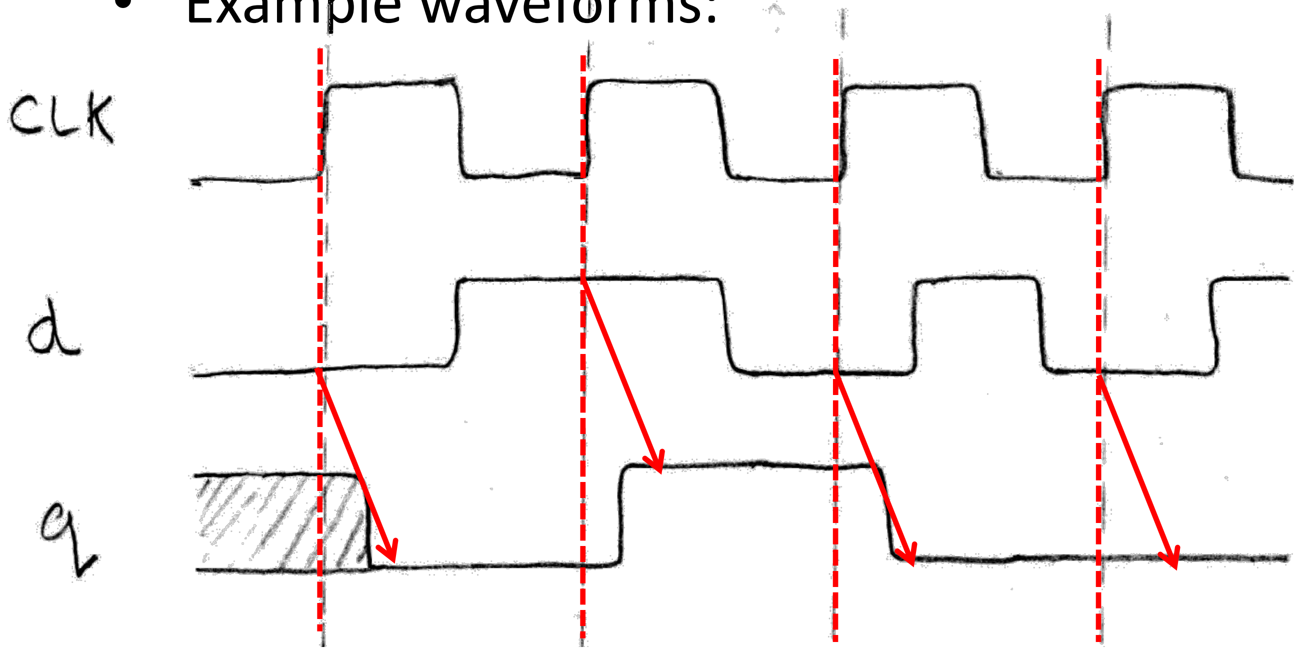
Flip-Flop Operation

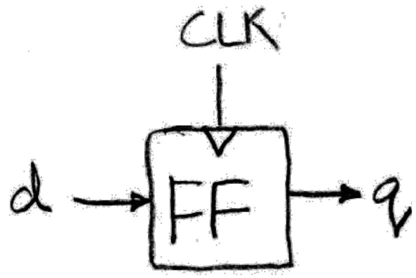
- Edge-triggered d-type flip-flop
  - This one is "positive edge-triggered"



- "On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored."

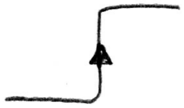
- Example waveforms:





Flip-Flop Timing

- Edge-triggered d-type flip-flop
  - This one is "positive edge-triggered"



- "On the rising edge of the clock, the input d is sampled and transferred to the output. At all other times, the input d is ignored."

- Example waveforms (more detail):