8. Foundations of processor design: combinational logic

EECS 370 – Introduction to Computer Organization – Winter 2015

Robert Dick, Andrew Lukefahr, and Satish Narayanasamy

EECS Department
University of Michigan in Ann Arbor, USA

Recap: Floating Point Representation

$$10.625_{10} \Rightarrow 1010.101_2$$

$$1.010101 \times 2^3$$

This must be a 1!
So don't store it.

+/-	Mantissa (1010101)	Exponent (3)
1 bit	23 bits	8 bits

$$10.625_{10} = 0 \quad 10000010 \quad 01010100000000000000000$$

Next 3 Lectures

1. **Combinational Logic:**
   - **Basics of electronics; logic gates, muxes, decoders**

1. Sequential Logic
   - Clocks, latches and flip-flops

2. State Machines
   - Building a simple processor

EECS 370 Introduction to Computer Organization
3

Levels of abstraction

- Quantum level, solid state physics
- Conductors, Insulators, Semiconductors.
- Doping silicon to make diodes and transistors.
- Building simple gates, boolean logic, and truth tables
- Combinational logic: muxes, decoders
- Clocks
- Sequential logic: latches, memory
- State machines
- Processor Control: Machine instructions
- Computer Architecture: Defining a set of instructions

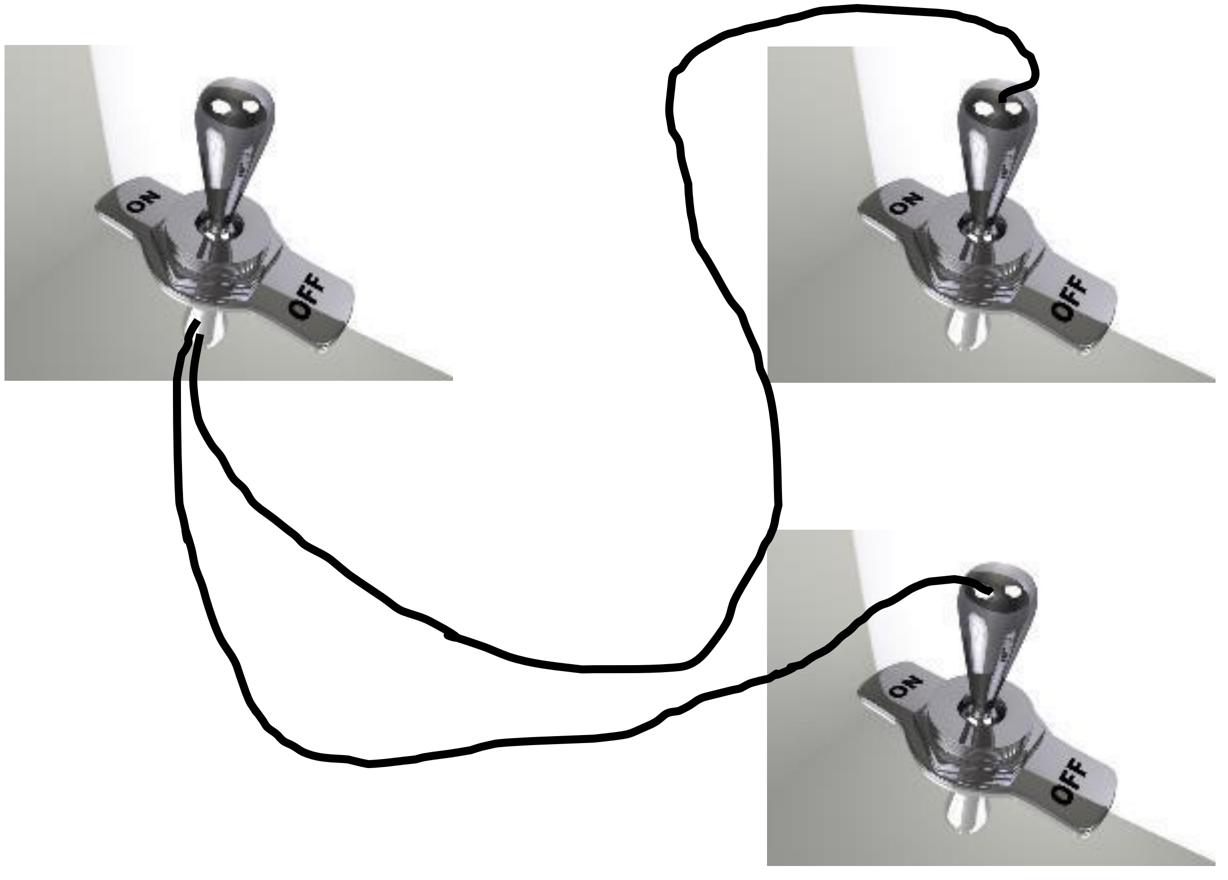
Start with the materials: Conductors and Insulators

- Conductor: a material that permits electrical current to flow easily. (low resistance to current flow)
  - Lattice of atoms with free electrons

- Insulator: a material that is a poor conductor of electrical current (High resistance to current flow)
  - Lattice of atoms with strongly held electrons

- Semi-conductor: a material that can act like a conductor or an insulator depending on conditions. (variable resistance to current flow)

Goal: a switch that controls other switchs



Doped silicon semiconductors

- How do we make tiny electronically controlled switches?
  - Semiconductors: control whether it conducts or not
- Basic semiconductor material of most electronics
- Start with pure crystalline silicon (4 valence electrons)
- Deliberately add a **small** amount of an impurity with a different number of valence electrons
  - Light doping: 1 in 100,000,000 atoms
  - Heavy doping: 1 in 10,000 atoms
- This small amount of impurity can drastically change the electrical properties of the silicon!

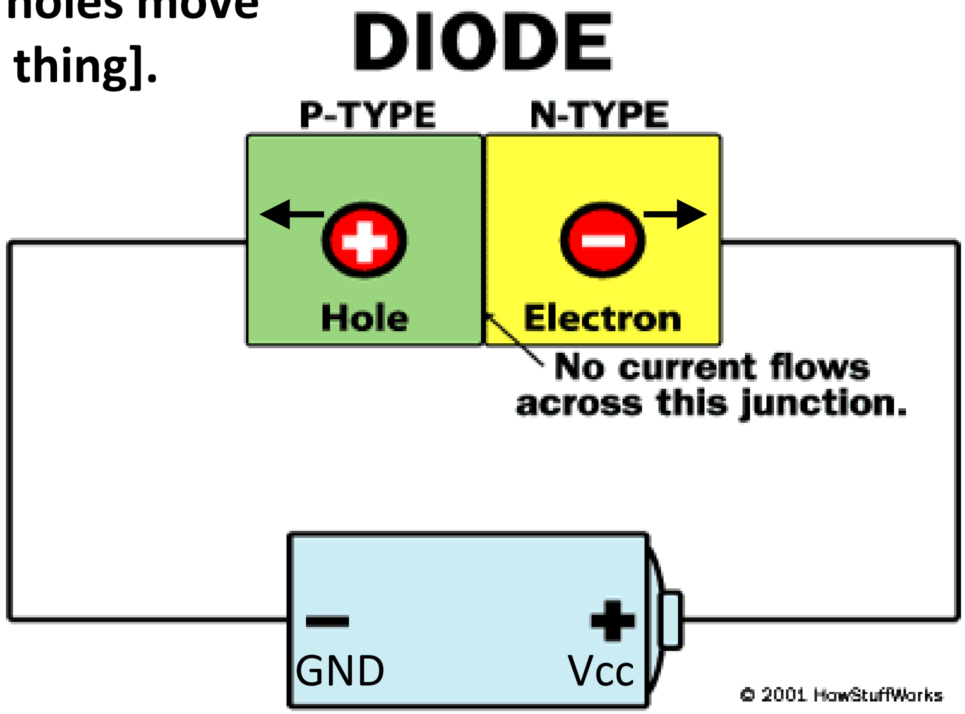
N and P types

- N-type
  - Add atoms of something with 5 valence electrons, such as phosphorus
  - Crystal ends up with some extra relatively free electrons
  - These free electrons can move to carry current

- P-type
  - Add atoms of something with 3 valence electrons, such as boron
  - Crystal ends up with some missing electrons (holes)
  - Moving holes can also carry current

Using doped silicon to make a junction diode

Electrons move from GND to fill holes [or holes move to GND – same thing].

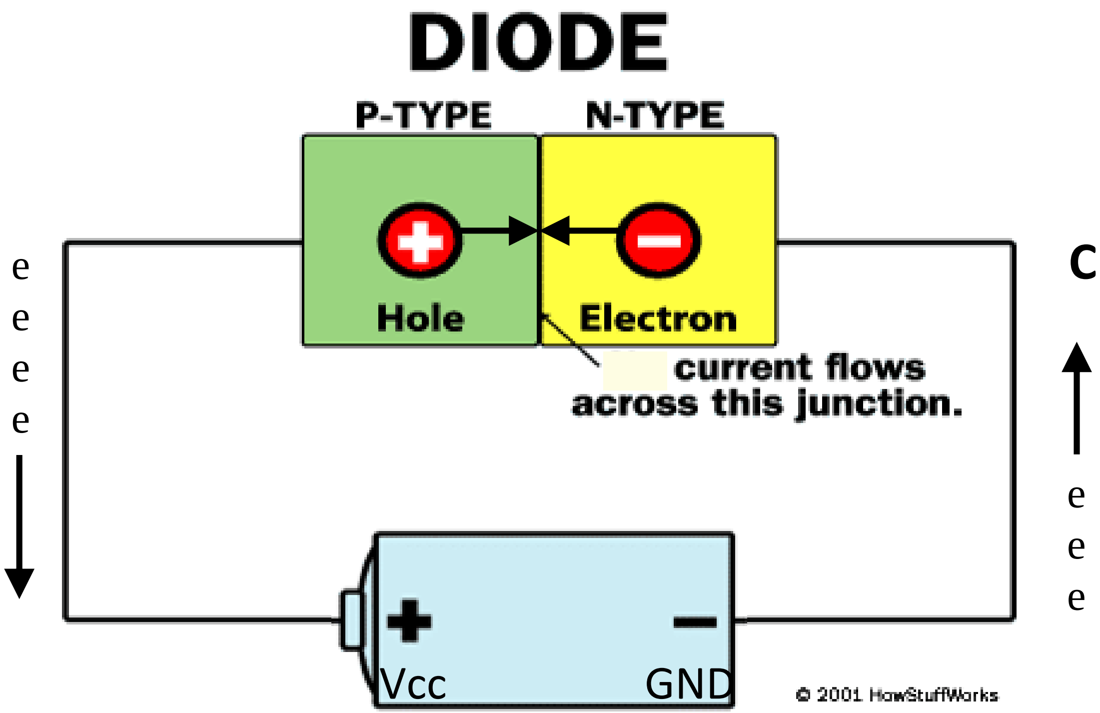


Electrons like to move to Vcc

No current flows across the p-n junction

Using doped silicon to make a junction diode

A junction diode allows current to flow in one direction and blocks it in the other.

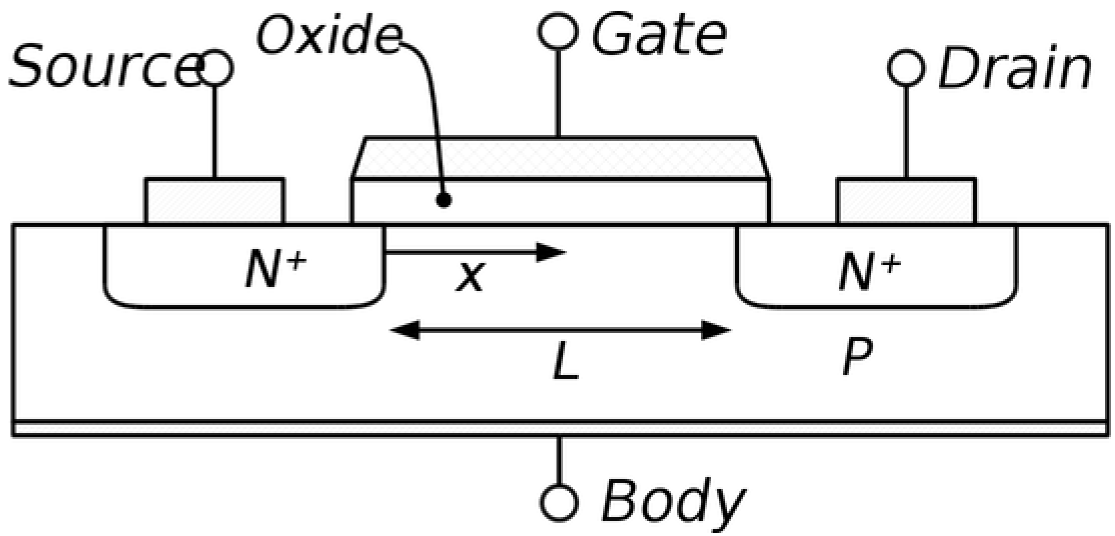


Current flows

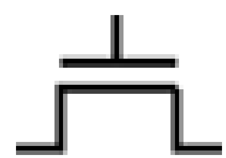
Current flows across the p-n junction

Making a transistor

Our first level of abstraction is the transistor (basically 2 diodes sitting back-to-back)

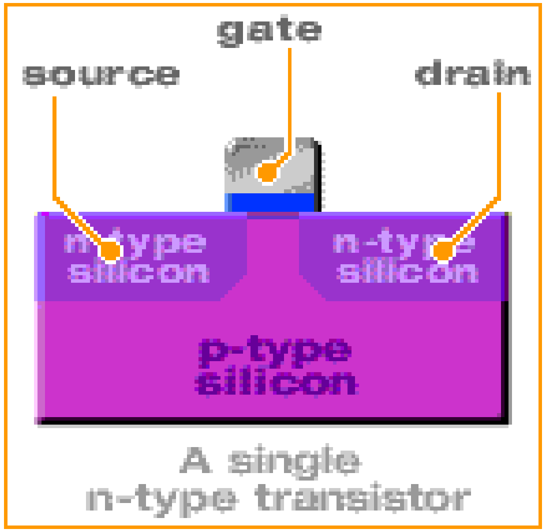


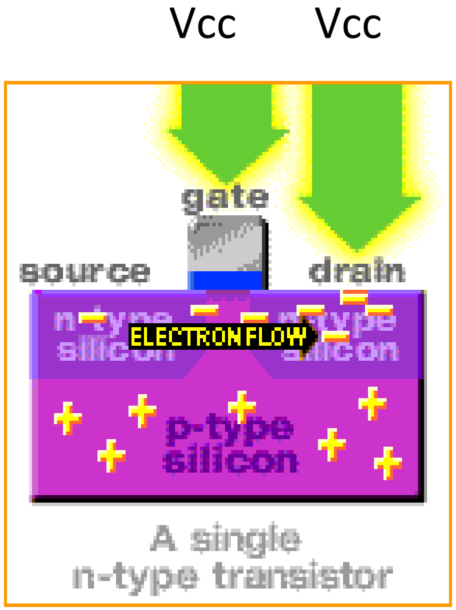
Electrical engineers use a diagram like this:

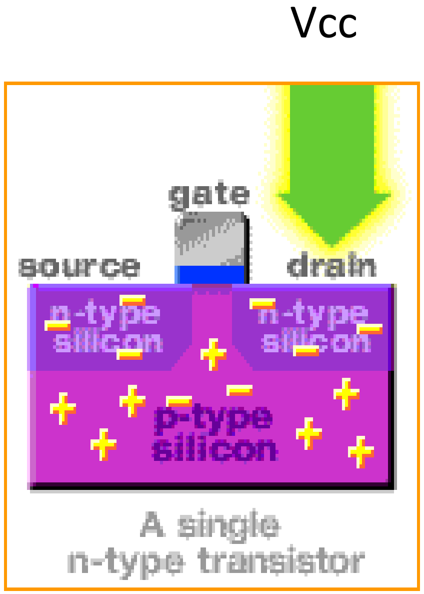


Making a transistor

Transistors are electronic switches connecting the source to the drain if the gate is “on”.







http://www.intel.com/education/transworks/INDEX.HTM

Recent pictures and the near future

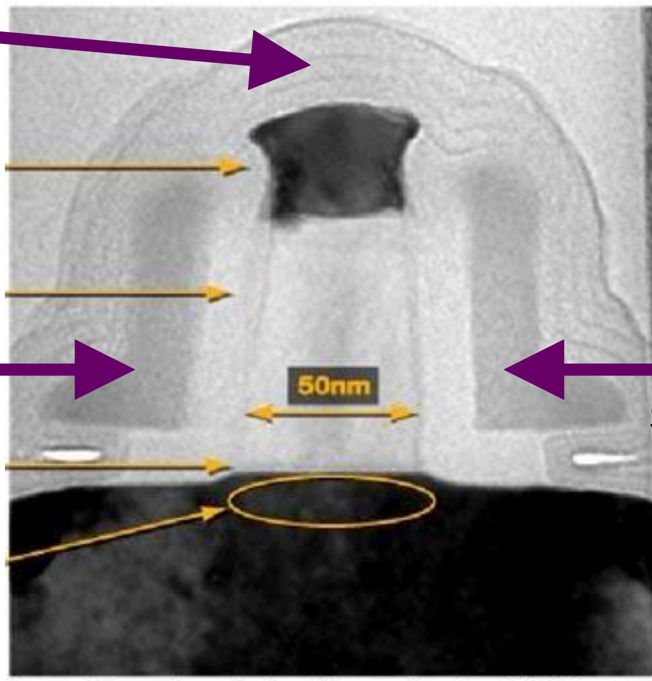
Source: Intel

Today

gat
e

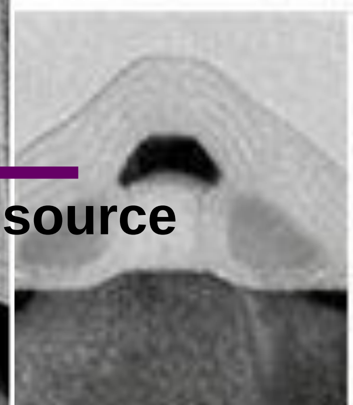
drain

source

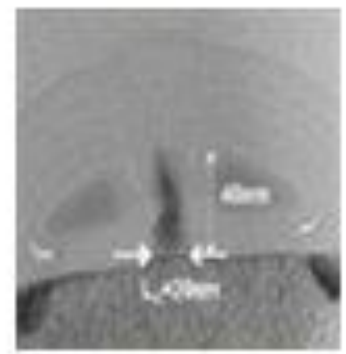


50nm transistor dimension is ~2000x smaller than diameter of human hair

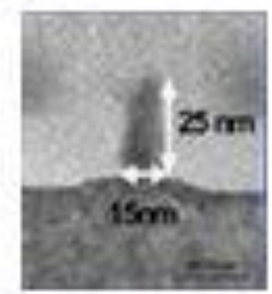
Future



30nm



20nm



15nm

90nm technology 2003

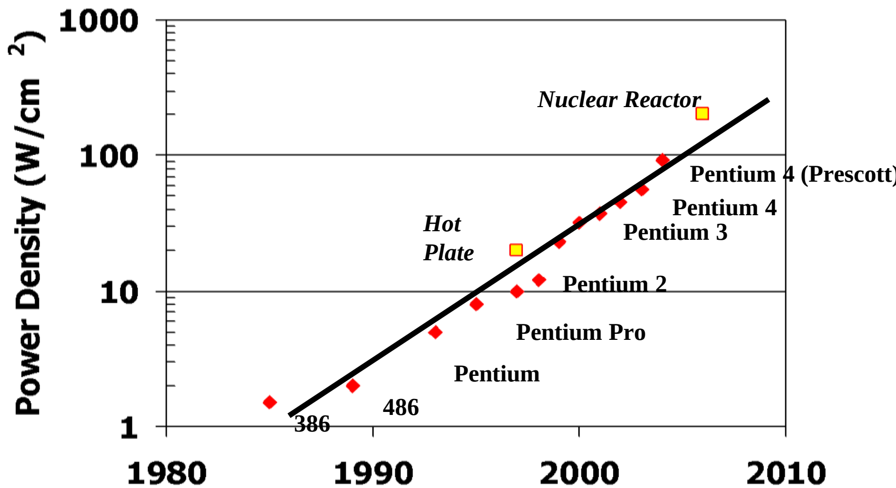
65nm technology 2005

45nm technology 2008

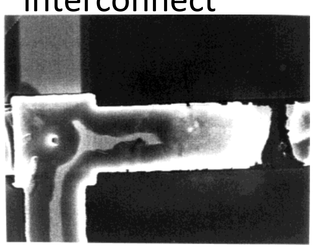
32nm technology 2010

The future carries a lot of problems..

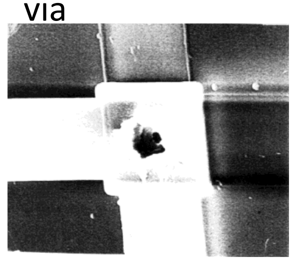
- *Area is NOT one of them*
- Power density – Watts/mm^2
- Reliability (faults)

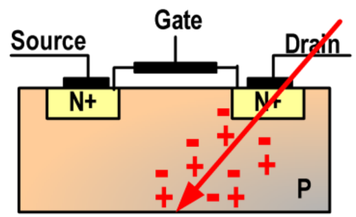


interconnect

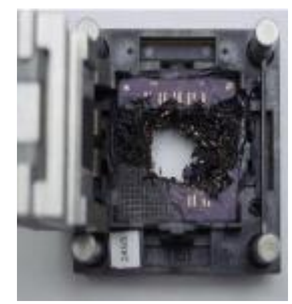


via





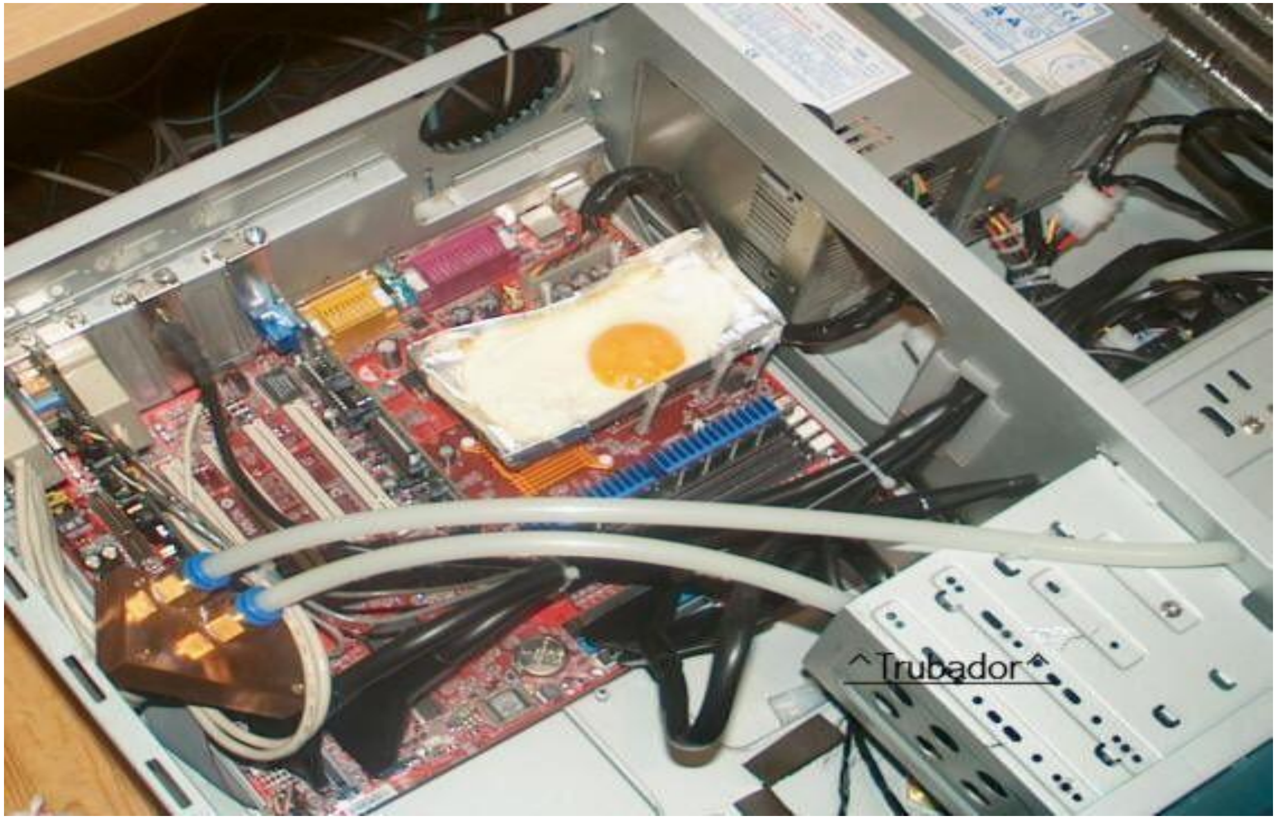
transients



Testing burn-in

- Process variation (not all transistors are equal)
- …

As for power: Cooking-aware Computing



Source: The New York Times, 25 June 2002

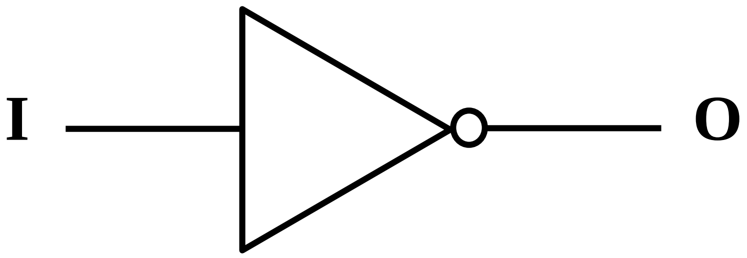
Basic gate: Inverter

CS abstraction
- logic function

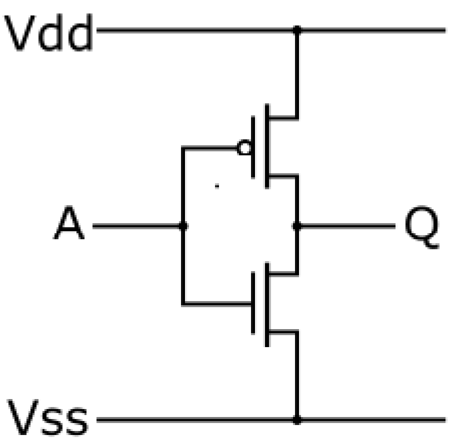
Truth Table

I	O
0	**1**
1	**0**

Schematic symbol (CS/EE)



Transistor-level schematic



CMOS Water Analogy

Electron: water molecule

Charge: weight of water

Voltage: height

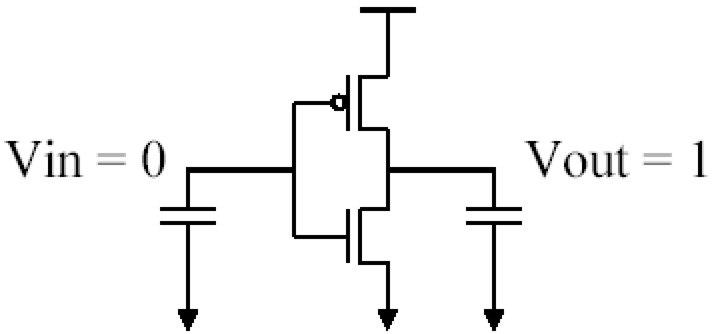
Current: flow rate

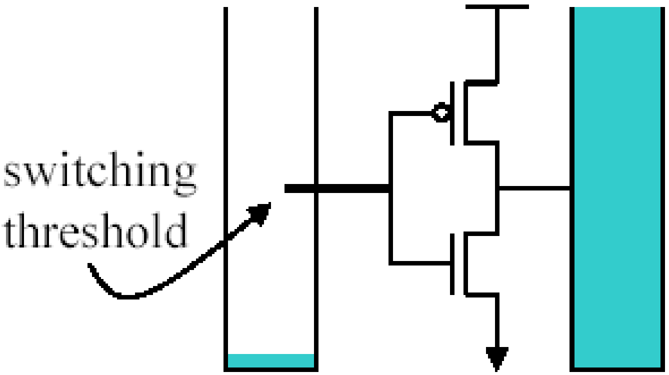
Capacitance: container cross-section

(Think of power-plants that store energy in water towers)

Slide courtesy D. Brooks, Harvard

Liquid Inverter





- Capacitance at input
  - Gates of NMOS, PMOS
  - Metal interconnect
- Capacitance at output
  - Fanout (# connections) to other gates
  - Metal Interconnect

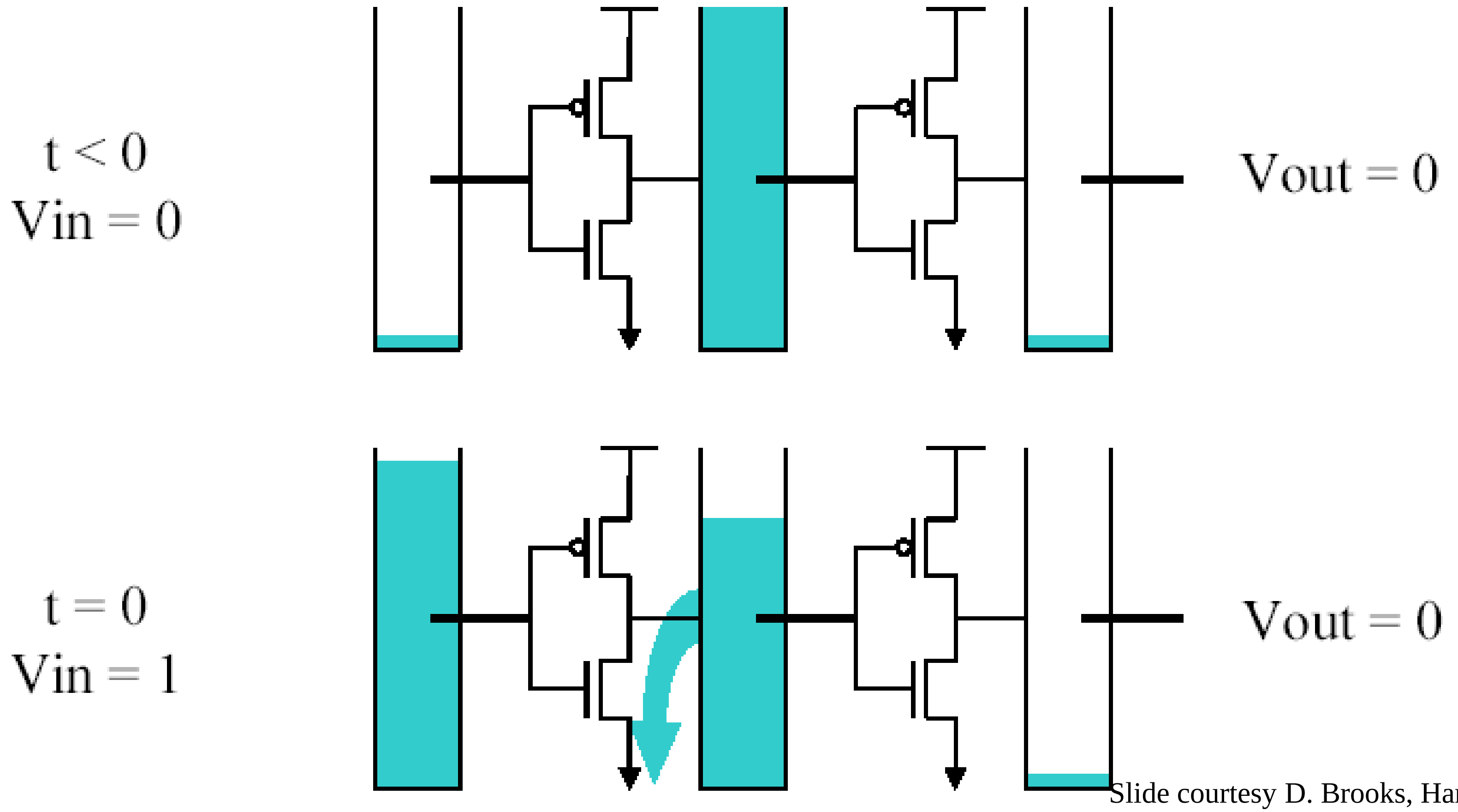
NMOS conducts when water level is above switching threshold

PMOS conducts below

No conduction after container full

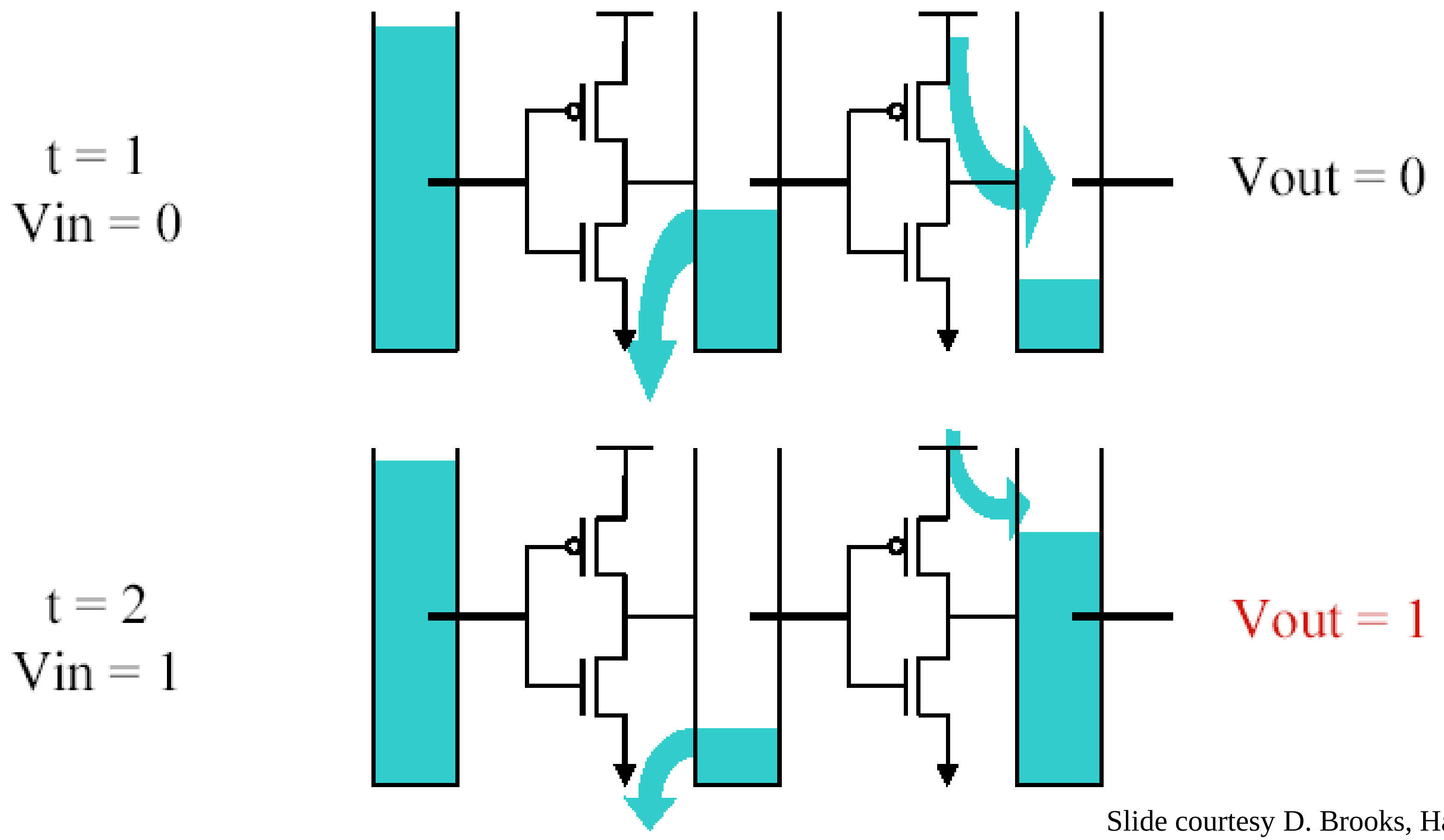
Slide courtesy D. Brooks, Harvard

Inverter Signal Propagation (1)



Slide courtesy D. Brooks, Harvard

Inverter Signal Propagation (2)

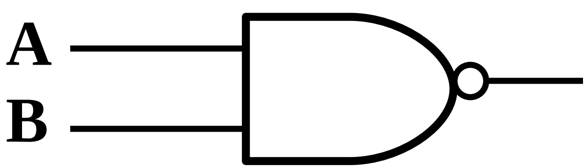


Slide courtesy D. Brooks, Harvard

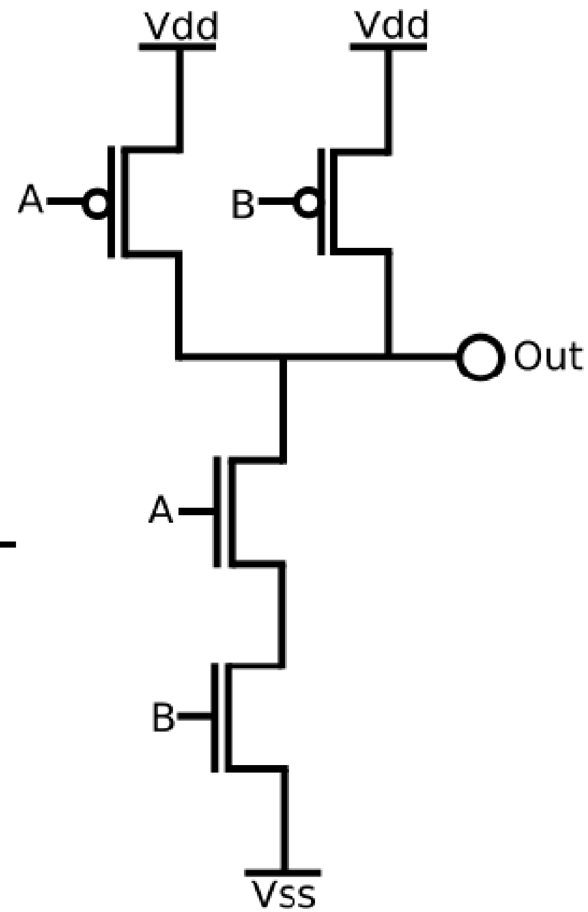
Basic gate: NAND (like the LC-2K NAND)

Truth Table

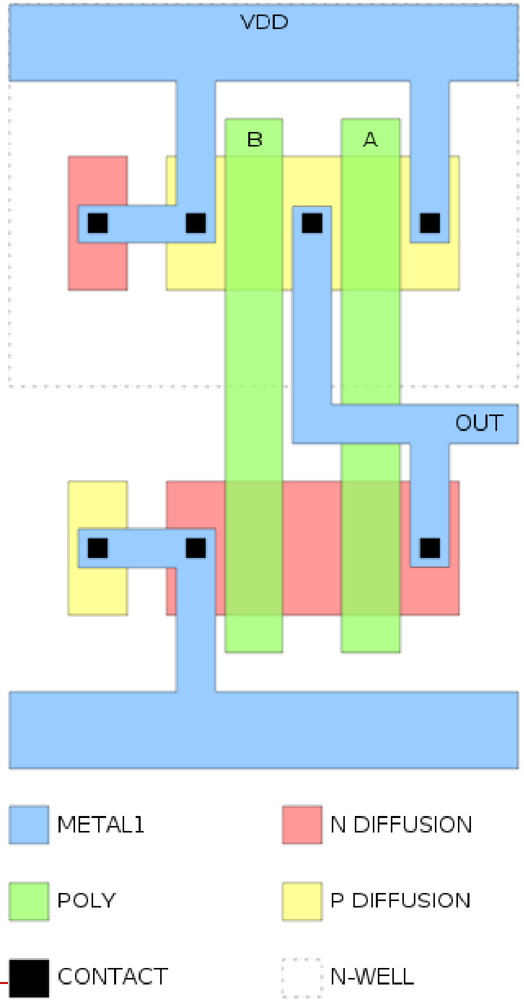
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0



Transistor-level schematic



Layout schematic

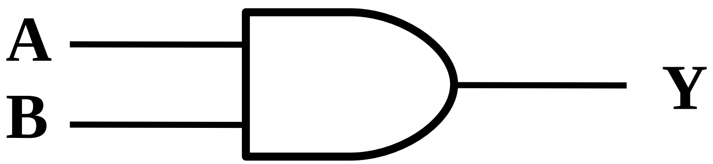


Basic gates: AND and OR

AND

Truth Table

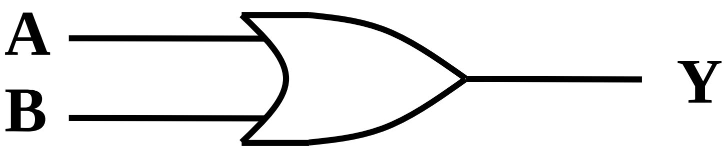
A	B	Y
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**



OR

Truth Table

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



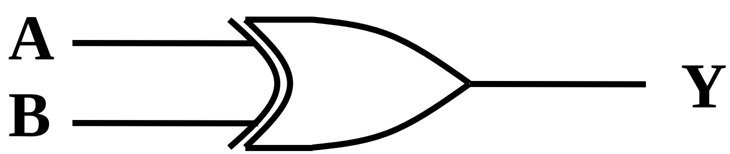
Exercise

- NAND can be used to implement *all* other logic functions!
- Exercise:
- Implement INV using only NAND gates
- Implement AND using only NAND gates
- Implement OR using only NAND gates
- Implement XOR using only NAND gates

Basic gate: XOR (eXclusive OR)

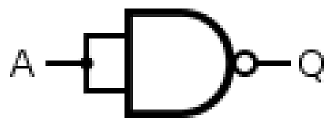
Truth Table

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

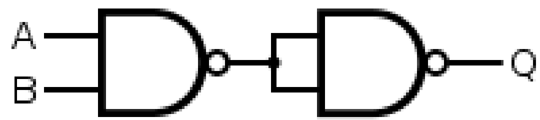


Exercise

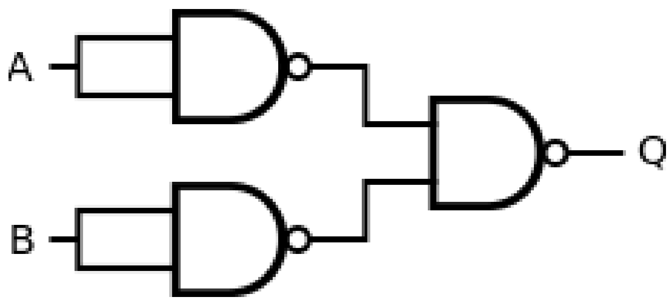
- INV



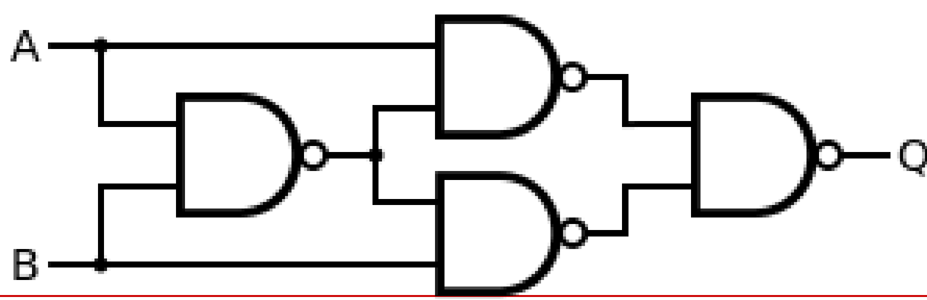
- AND



- OR



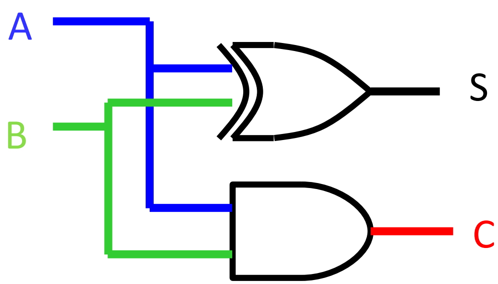
- XOR



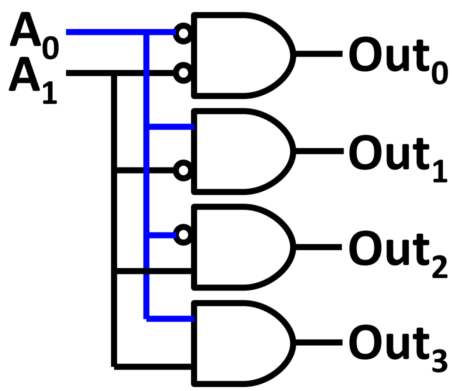
Combinational Circuits implement Boolean expressions

- Output is determined exclusively by the input
- **No memory**: Output is valid only as long as input is
  - Adder is the basic gate of the ALU (Lecture 9)
  - Decoder is the basic gate of indexing
  - MUX is the basic gate controlling data movement

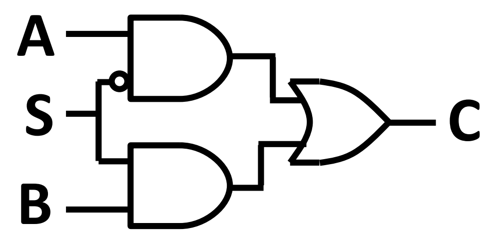
Half Adder



Decoder



MUX



- Exercise: write the truth tables for each of these

Half Adder

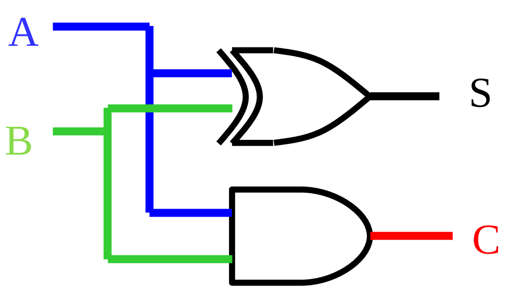
- Carry bit (C) can use an AND gate
- Sum bit (S) can use an XOR gate

Truth Table

Add 2 1-bit numbers

A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Circuit



Decoder

A_1	A_0	Out_3
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

Out_3 is just an AND gate

A_1	A_0	Out_2
0	**0**	**0**
0	**1**	**0**
1	**0**	**1**
1	**1**	**0**

Out_2 would be an AND gate if A_0 was inverted

A_1	A_0	Out_1
0	**0**	**0**
0	**1**	**1**
1	**0**	**0**
1	**1**	**0**

Invert A_1

A_1	A_0	Out_0
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**0**

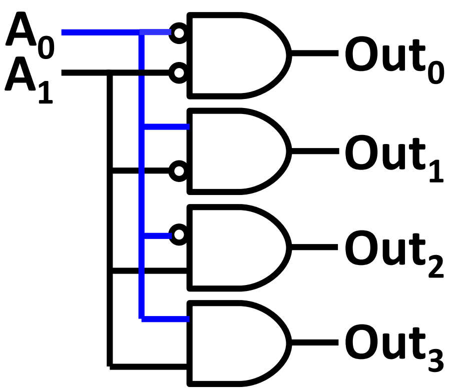
Invert A_1 and A_2

Truth Table

Select a single line given an index

A_1	A_0	$Out_{3\text{-}0}$
0	**0**	**0001**
0	**1**	**0010**
1	**0**	**0100**
1	**1**	**1000**

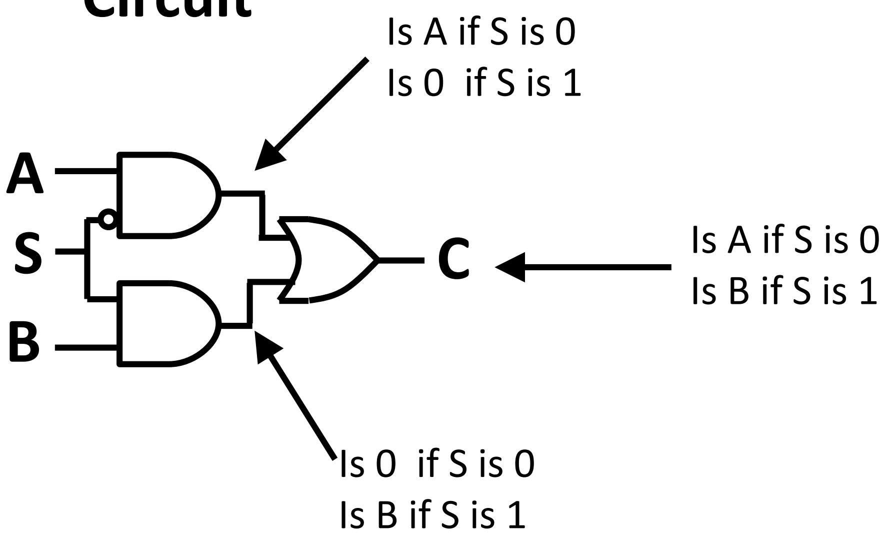
Circuit



Multiplexor (MUX)

- Input S selects either input A or input B
- This is called a 2x1 MUX, since it has 2 inputs and 1 output

Circuit



Truth Table

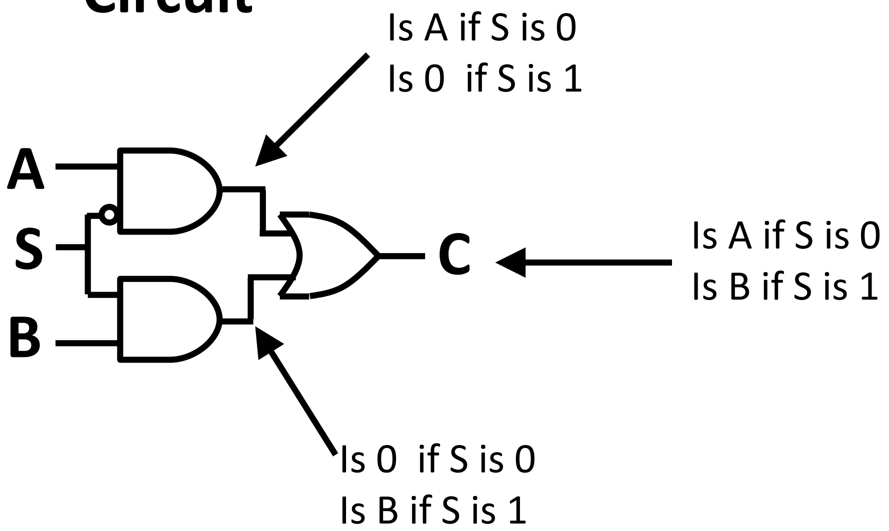
Select one of multiple
input lines to pass to the output

A	B	S	C
a	b	0	a
a	b	1	b

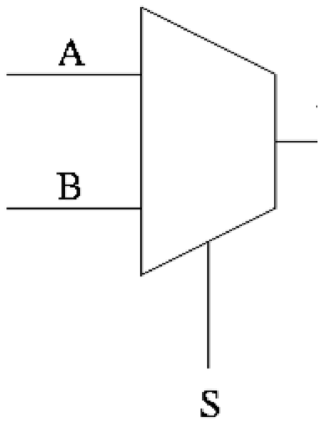
Exercise

- Build (draw) a 4x1 mux
- Hint: use 2x1 muxes

Circuit

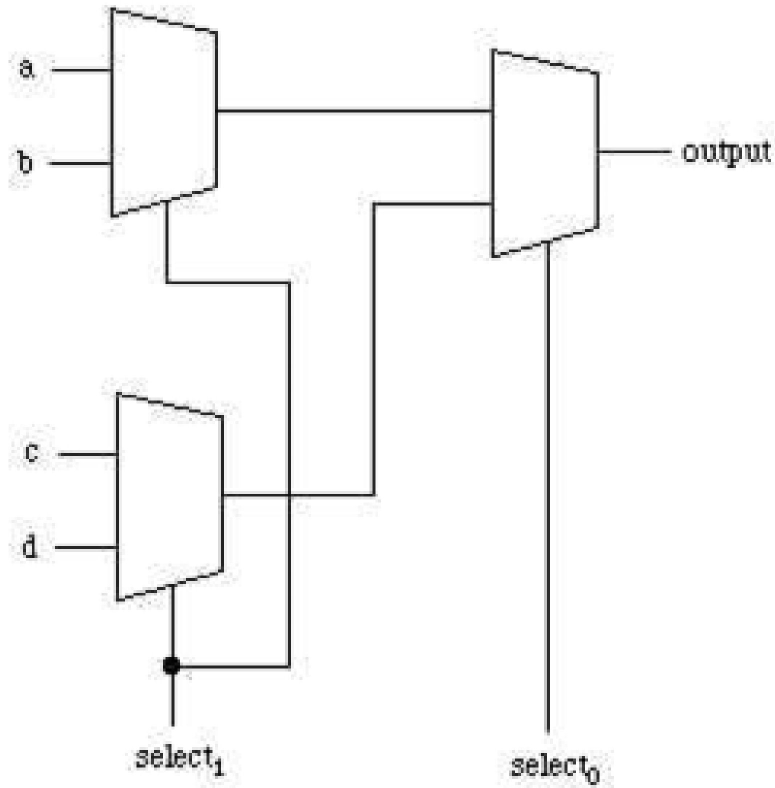


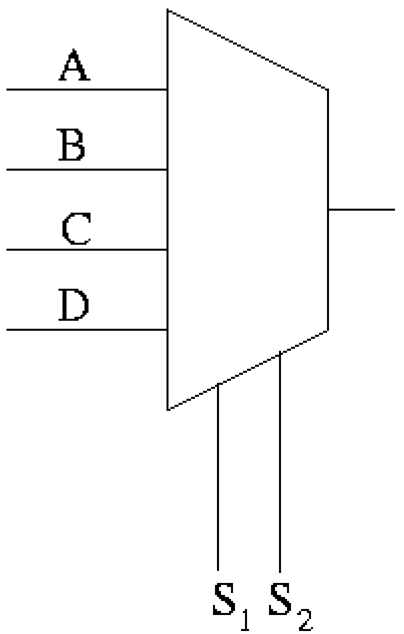
Symbol



Exercise

- 4x1 mux made from 2x1 muxes

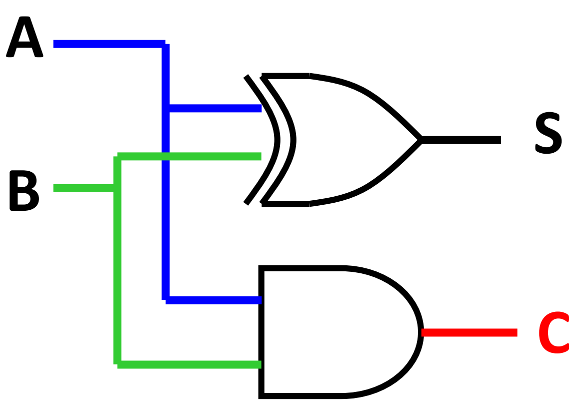




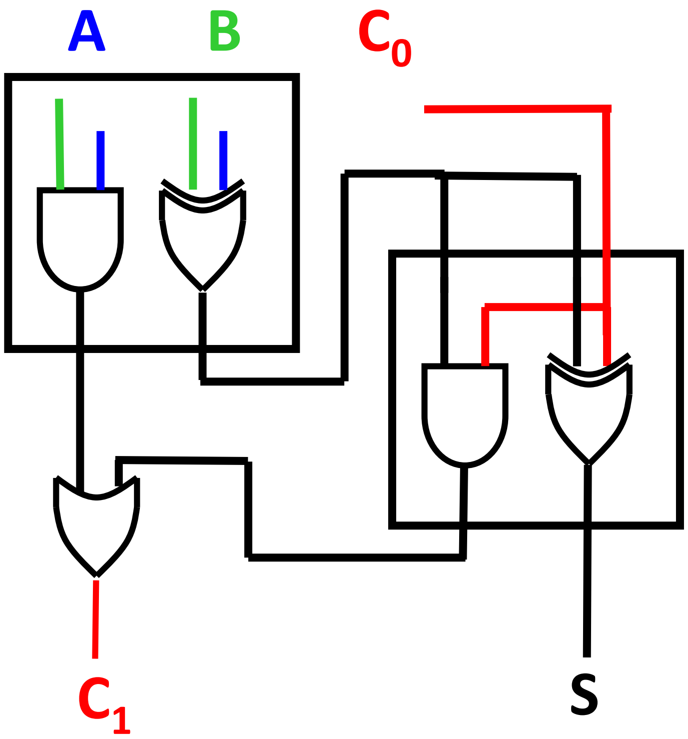
Building combinational circuits: Half and Full adder

Half Adder

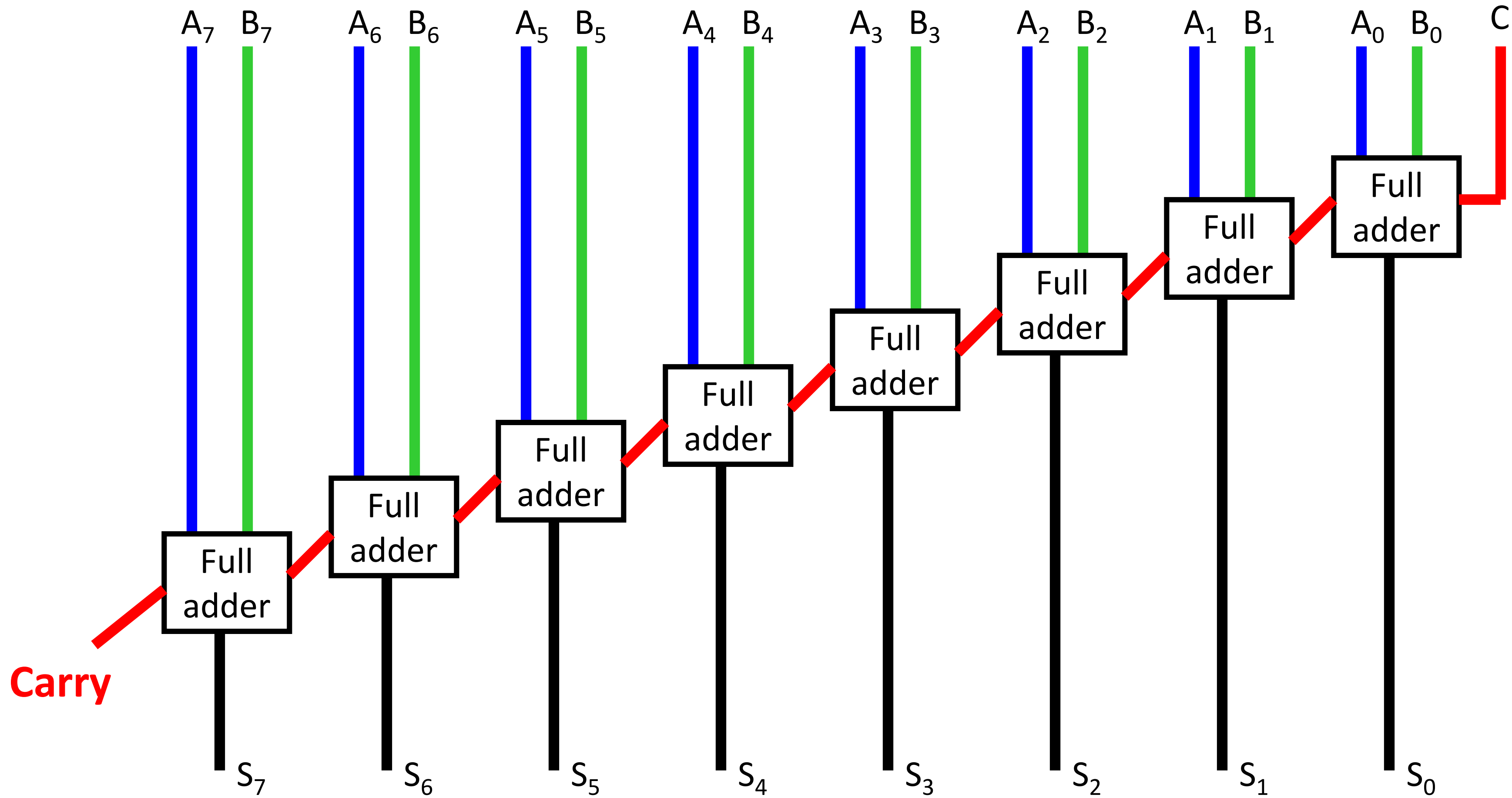
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Full Adder



8-bit Ripple Carry Adder

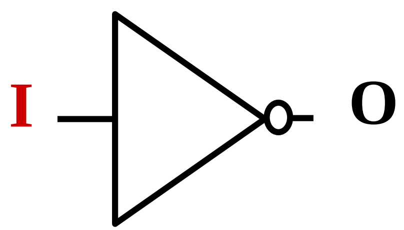


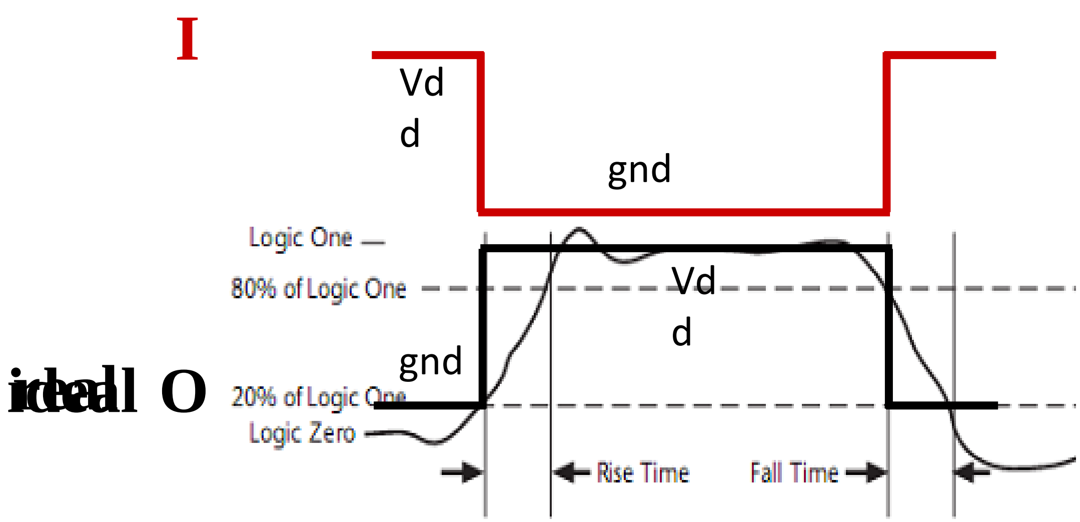
Unfortunately this has a very large propagation time for 32 or 64 bit adds

One more problem: Propagation delay in combinational gates

- Gate outputs do not change exactly when inputs do.
  - Transmission time over wires (~speed of light)
  - Saturation time to make transistor gate switch

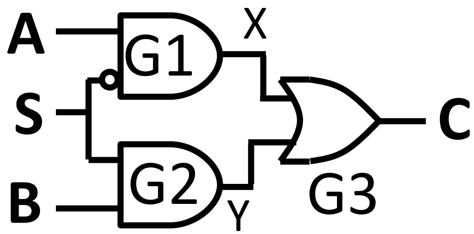
⇒ Every combinatorial circuit has a propagation delay (time between input and output stabilization)

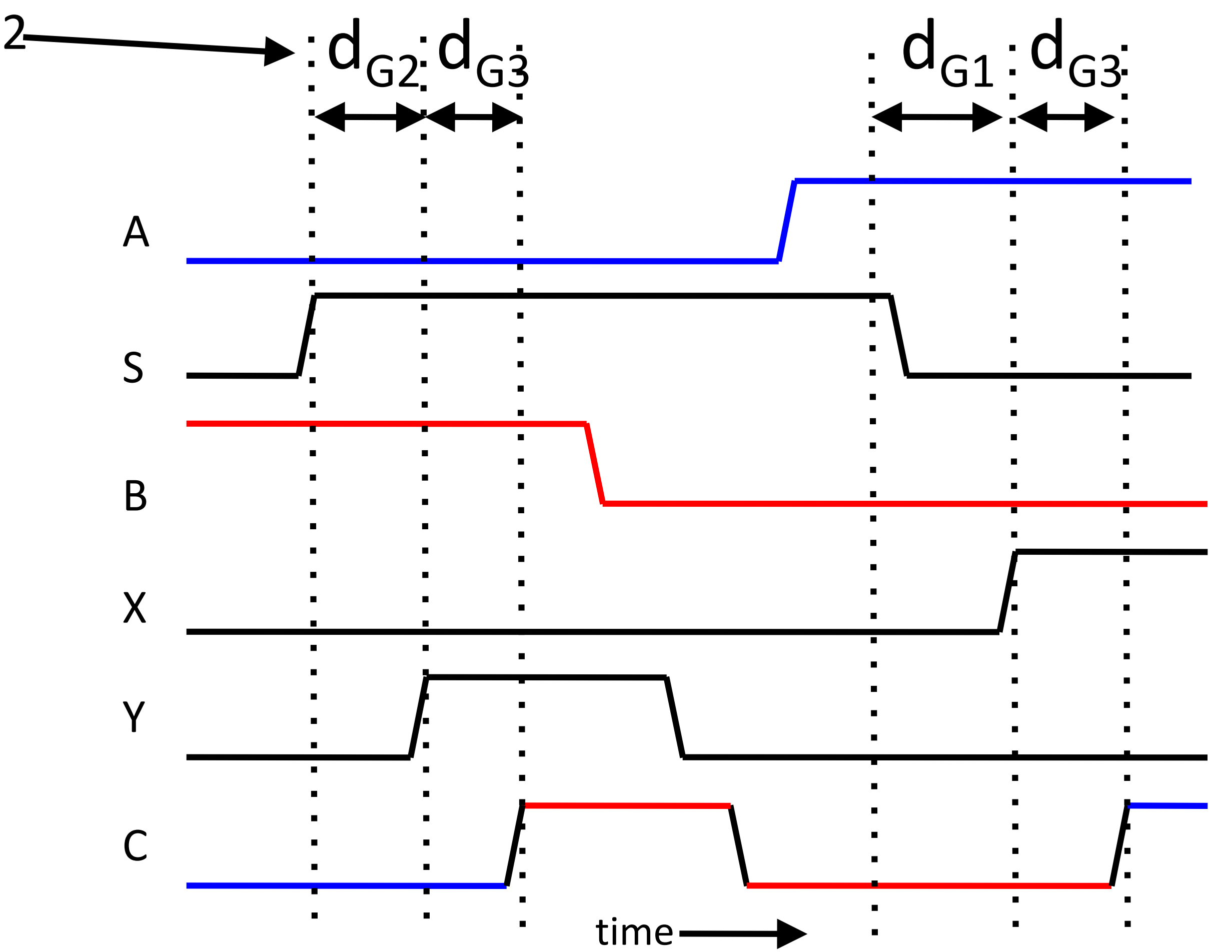




Timing in Combinational Circuits

MUX



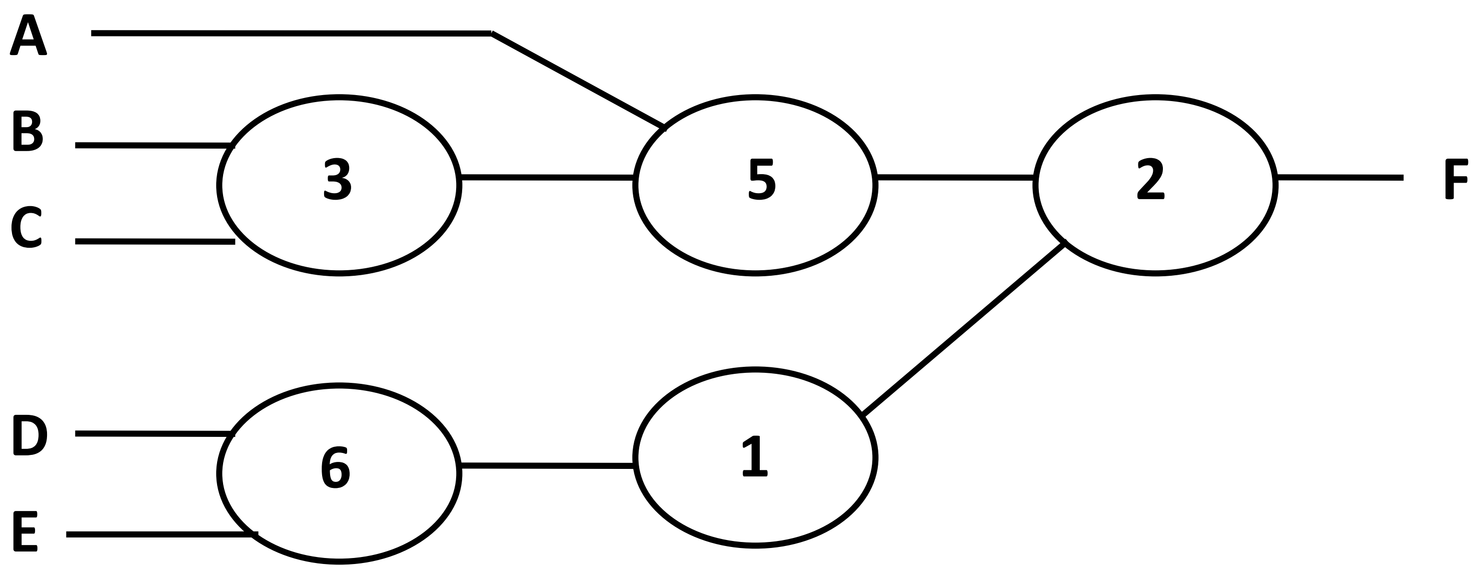


What is the input/output delay (or simply, delay) of the MUX?

Waveform viewers are part of designers' daily life

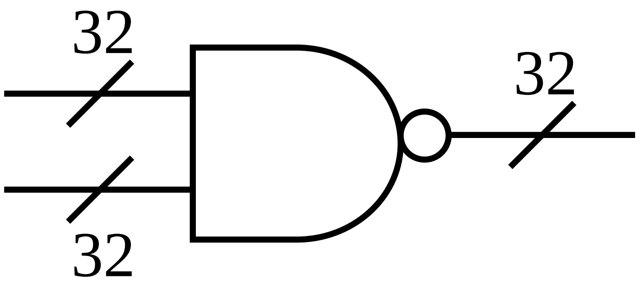
What is the delay of this Circuit?

Each oval represents one gate, the type does not matter



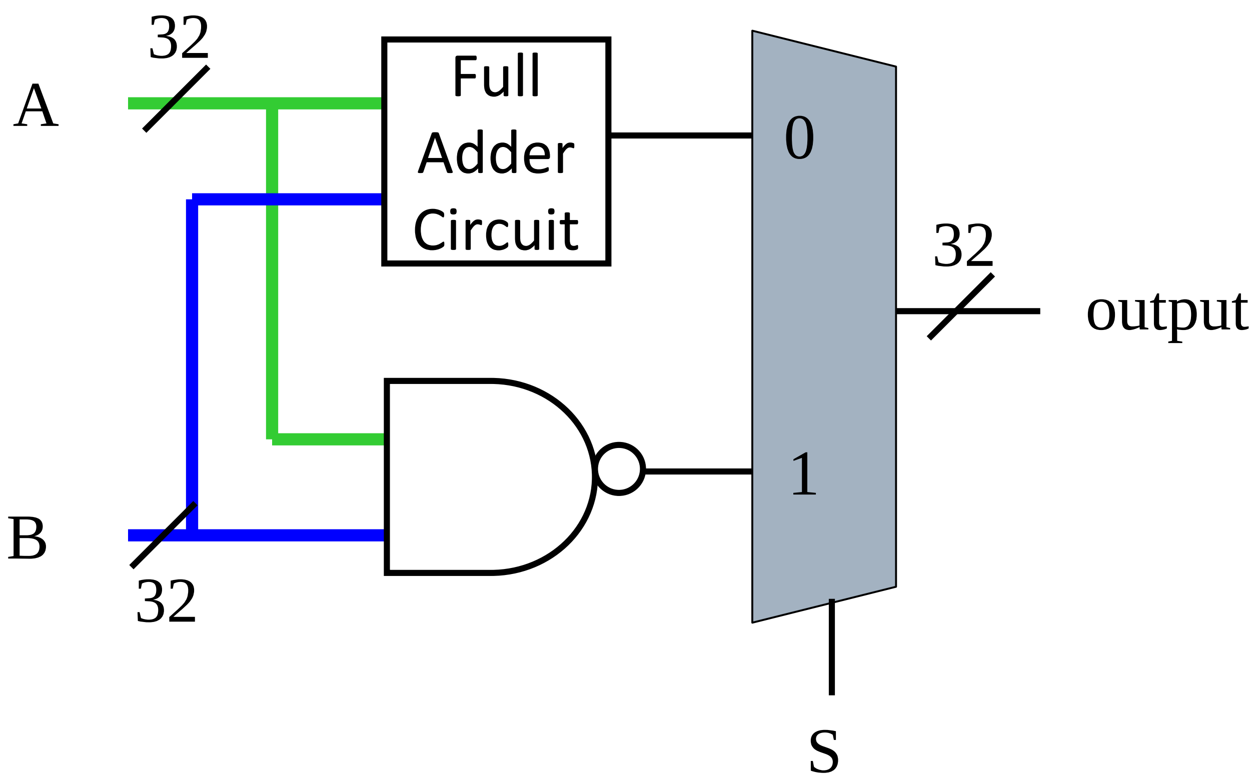
Exercise

- Use the blocks we have learned about so far (full adder, NAND, mux) to build this circuit
  - Input A, 32 bits
  - Input B, 32 bits
  - Input S, 1 bit
  - Output, 32 bits
  - When S is low, the output is A+B, when S is high, the output is NAND(a,b)
- Hint: you can express multi-bit gates like this:

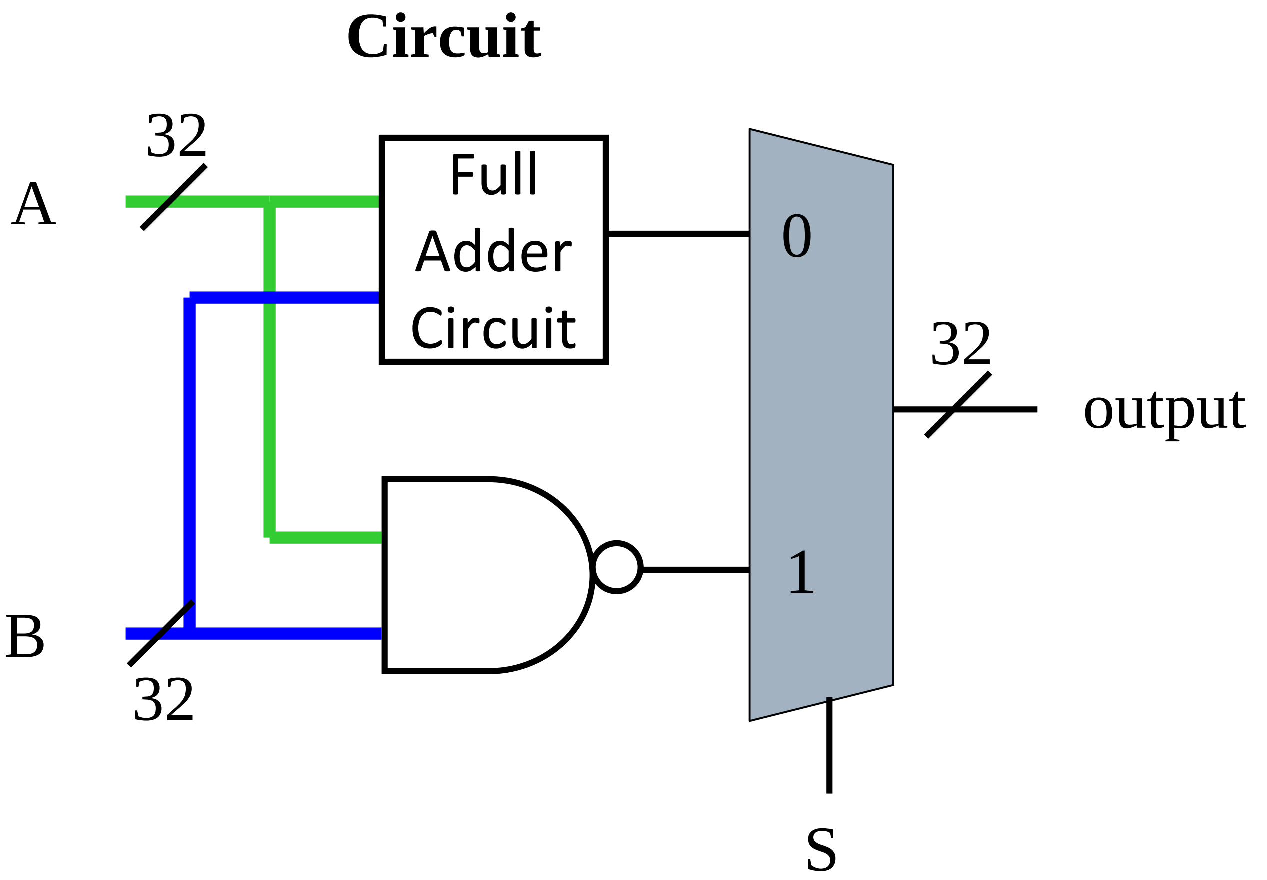


Exercise

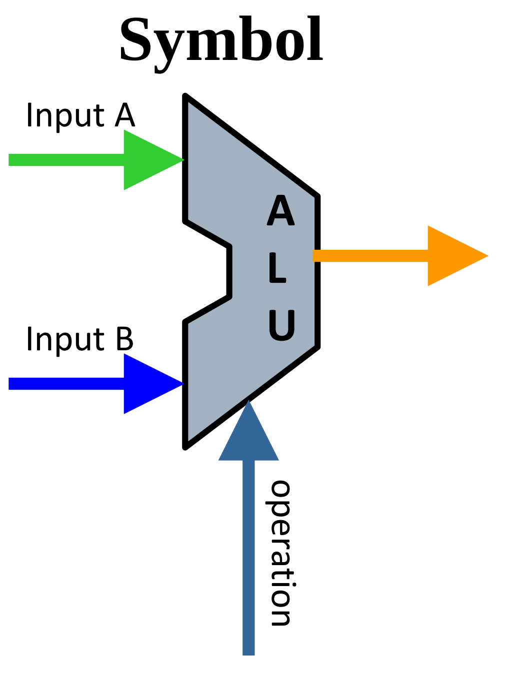
- This is a basic ALU (Arithmetic Logic Unit)
- It is the heart of a computer processor!



LC-2K ALU



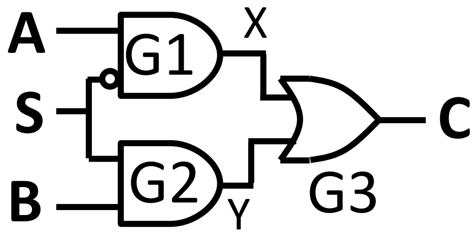
Circuit
32
A
Full
Adder
Circuit
0
32
output
1
B
32
S



Symbol
Input A
A
L
U
Input B
operation

Verifying a Circuit

- How many possible inputs are there for a 2-1 mux?

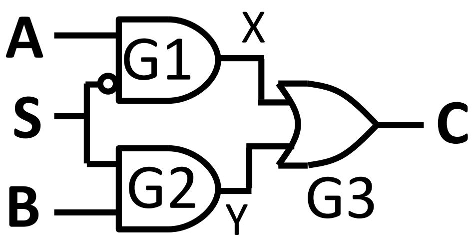


- How many possible inputs are there for a Core i7 with a 1,366 pins? For simplicity, assume all the pins are inputs.

- How long would it take to try them all?

Verifying a Circuit

- How many possible inputs are there for a 2-1 mux?
  - A, B, or C could be 0 or 1, so 2^3 = 8
  - Easy to test all possible inputs



- How many possible inputs are there for a Core i7 with a 1,366 pins? For simplicity, assume all the pins are inputs.
  - 2^1366 = REALLY BIG NUMBER
  - Comparison: ~10^80 = 2^266 atoms in the universe

- How long would it take to try them all?
  - We don’t have enough time!

Verifying a Circuit

- It’s hard to verify combinational circuits
- *It gets worse when we add memory to the circuit*
- Next time: sequential circuits