6. Basic processor design – Combinational logic

EECS 370 – Introduction to Computer Organization – Winter 2007

Prof. Valeria Bertacco & Prof. Scott Mahlke

EECS Department
University of Michigan in Ann Arbor, USA

Next 4 Lectures

1. Combinational Logic:
   - Basics of electronics; logic gates, muxes, decoders
2. Sequential Logic:
   - clocks and data storage
3. ALU design
   - Building an adding circuit
4. State Machines
   - Building a simple processor

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

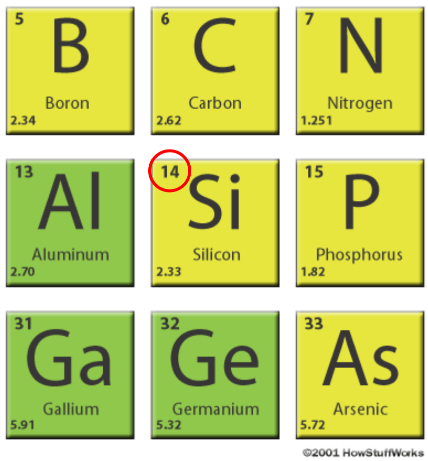
Start with the materials: Conductors and Insulators

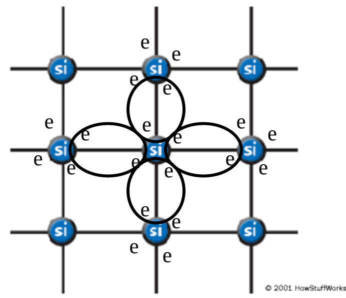
- Conductor: a material that permits electrical current to flow easily. (low resistance to current flow)
  - Lattice of atoms with free electrons

- 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)

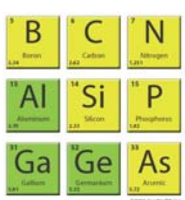
Making a semiconductor using silicon



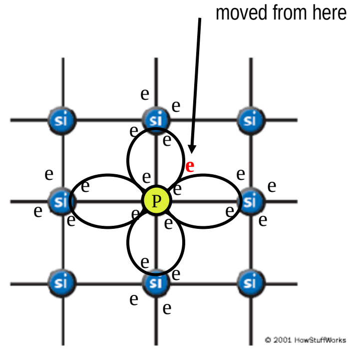


N-type Doping

- We can increase the conductivity by adding atoms of phosphorus or arsenic to the silicon lattice.
  - They have more electrons (1 more) which is free to wander…
  - This is called n-type doping since we have some free (negatively charged) electrons

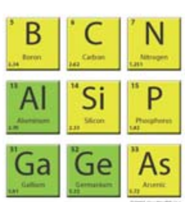


Making a semiconductor using silicon



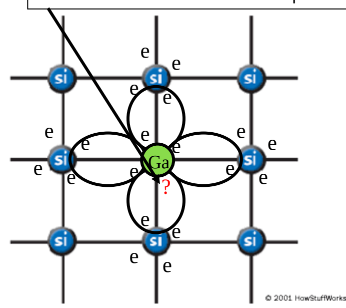
P-type Doping

- Interestingly, we can also improve the conductivity by adding atoms of gallium or boron to the silicon lattice.
  - They have fewer electrons (1 fewer) which creates a hole. Holes also conduct current by stealing electrons from their neighbor (thus moving the hole).
  - This is called p-type doping since we have fewer (negatively charged) electrons in the bond holding the atoms together.



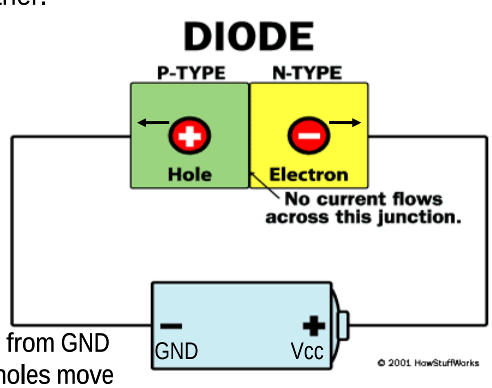
Making a semiconductor using silicon

This atom will accept an electron even though it is one too many since it fills the eight electron positions in this shell. Again this lets current flow since the electron must come from somewhere to fill this position.



Using doped silicon to make a junction diode

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

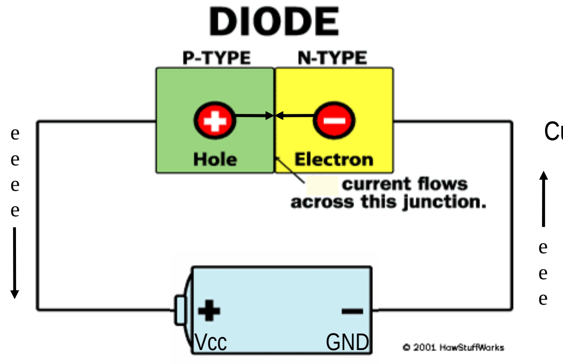


Electrons like to move to Vcc

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

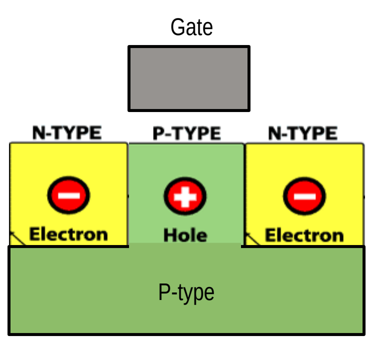
Using doped silicon to make a junction diode

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



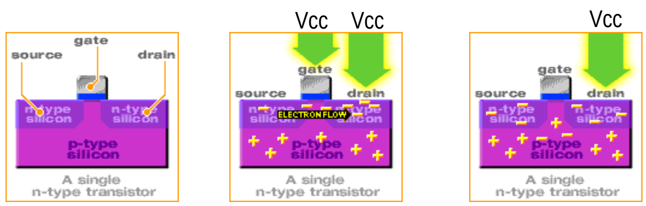
Making a transistor

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



Making a transistor

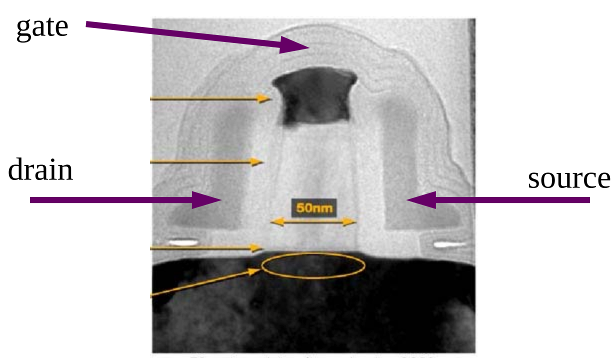
Transistors are electronic switches connecting the source to the drain if the gate is “on”.



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

A recent picture...

- 90nm silicon technology from Intel



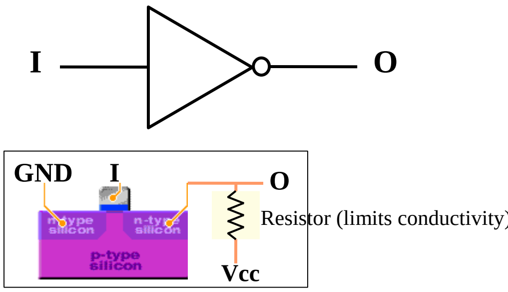
50nm transistor dimension is ~2000x smaller than diameter of human hair

Abstractions in CS (gates)

Basic Gate: Inverter

Truth Table

I	O
0	1
1	0



http://www.howstuffworks.com/boolean.htm

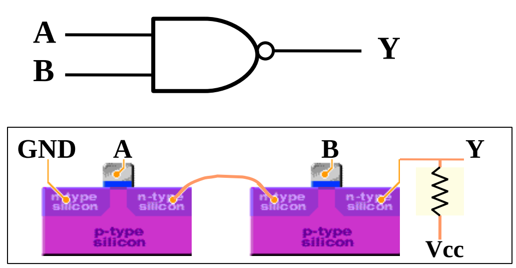
Textbook: See Appendix B

Abstractions in CS (gates)

Basic Gate: NAND (like the LC2Kx Nand)

Truth Table

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

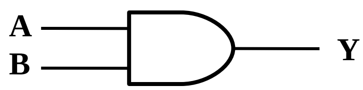


Abstractions in CS (gates)

Basic Gate: AND

Truth Table

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

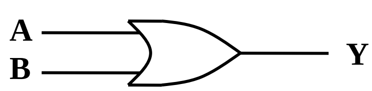


Abstractions in CS (gates)

Other Basic Gates: OR gate

Truth Table

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

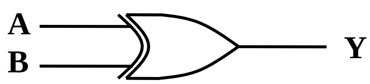


Abstractions in CS (gates)

Other Basic Gates: XOR gate

Truth Table

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



Combinational Circuits

- Implement Boolean expressions
  - Output is determined exclusively by the input
  - No memory: Output is valid only as long as input is

Half Adder

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

Decoder

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

MUX

Select one of multiple input lines to pass to the output

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

Half Adder

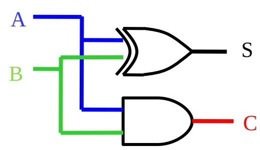
- Carry bit (C) can use an AND gate
- Sum bit (S) can use an XOR gate

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

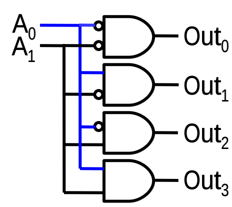
Invert A_1 and A_2

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

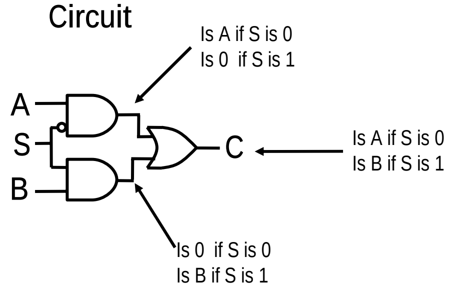
Circuit



Multiplexor (MUX)

- Input S selects either input A or input B

Circuit



Truth Table

Select one of multiple input lines to pass to the output

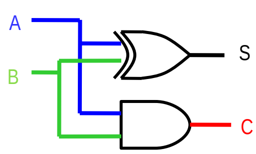
A	B	S	C
a	b	0	a
a	b	1	b

This is called a 2x1 MUX since it has 2 inputs and 1 output.
How would you build a 4x1 MUX?

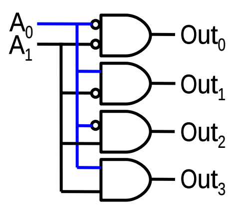
Combinational Circuits

- 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

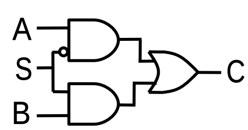
Half Adder



Decoder



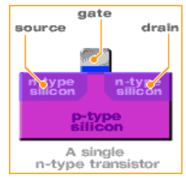
MUX



Propagation through Combinational Gates

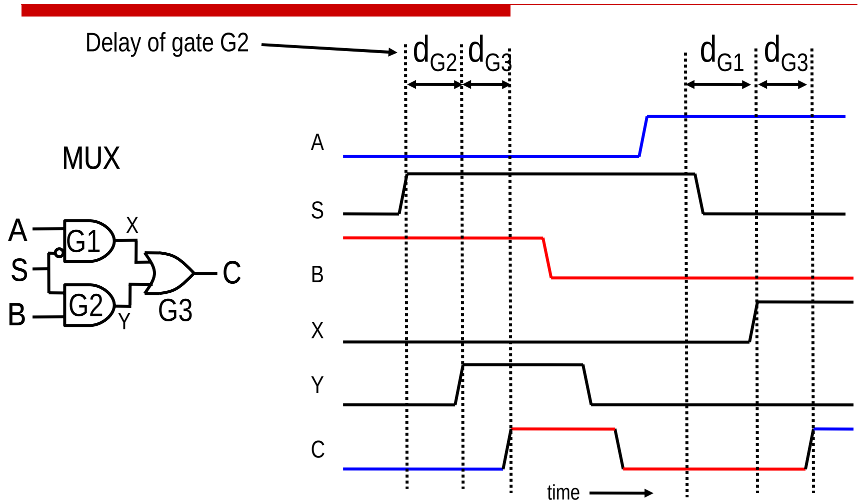
- Gate outputs do not change exactly when inputs do.
  - Transmission time over wires (~speed of light)
  - Saturation time to make transistor gate switch

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



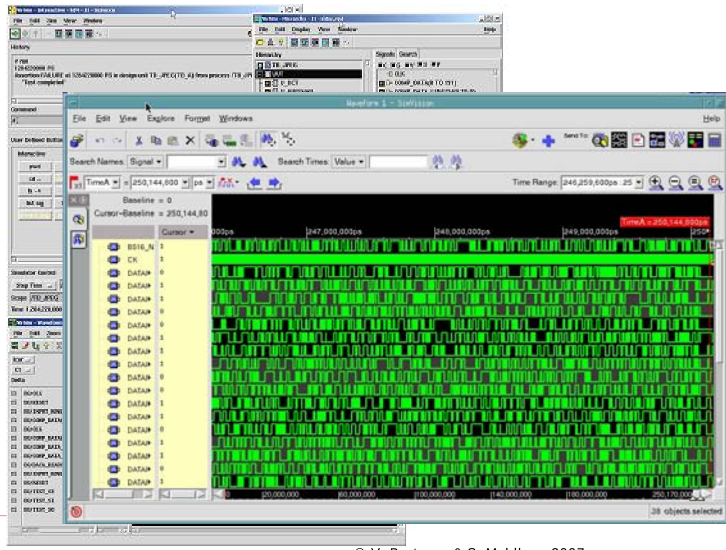
Electron saturation of p-type silicon at gate

Timing in Combinational Circuits

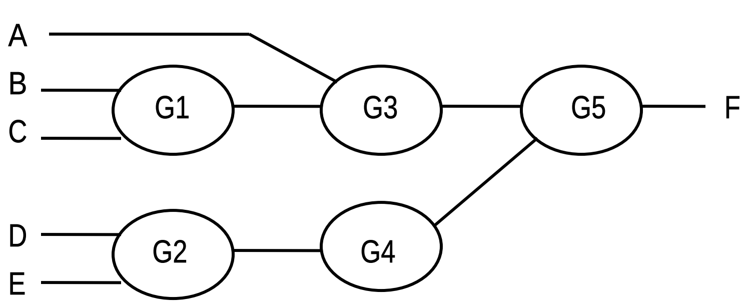


What is the input/output delay (or simply, delay) of the MUX?

Waveform viewers are part of designers' daily life



How about this Circuit?



Next Time: State and Sequencing