EECS 151/251A Discussion 2

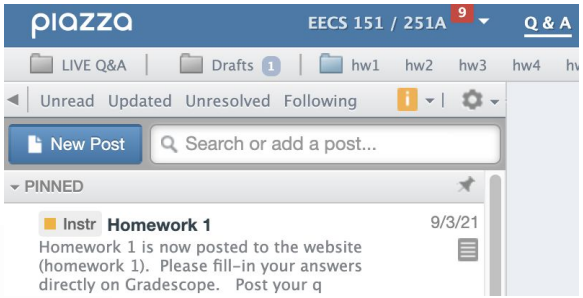
Daniel Grubb
9/7, 9/8

About Me (Daniel Grubb)

- EECS graduate student
  - Advised by Bora
  - EECS undergrad at Cal too
- I took EECS151 4.5 years ago… from Vighnesh and Bob!
  - Lots of course staff experience, so learn and ask questions
- I’ve been in Berkeley (and northern California) for a while now, so ask me about suggestions for things to do/see/eat!

EECS151 General Course Recap

- Stay on top of your work for this class
  - You can get a job straight from this course!
- I recommend looking over both ASIC and FPGA labs - useful info in both
  - FPGA - focus on Verilog, practical circuits, physical demos
  - ASIC - focus on design, tool flow
- Piazza and office hours - use liberally!
- The suggested texts are extremely helpful!
  - Lots of history lessons in courses like this



Textbooks

- **Recommended** Digital Design and Computer Architecture, 2nd ed, David Money Harris & Sarah L. Harris (H & H)
- **Recommended** Digital Integrated Circuits: A Design Perspective, Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolić (RCN)
- **Useful** Computer Organization and Design RISC-V Edition, David Patterson and John Hennessy (P&H)
- **Useful** CMOS VLSI Design, Neil Weste, David Harris (W&H)

Agenda

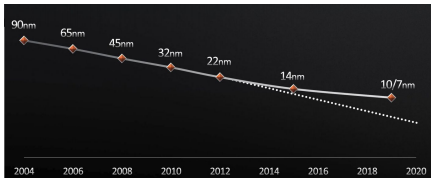
- Administrivia
- Lecture recap
- Digital? CMOS Inverter
- Verilog primer

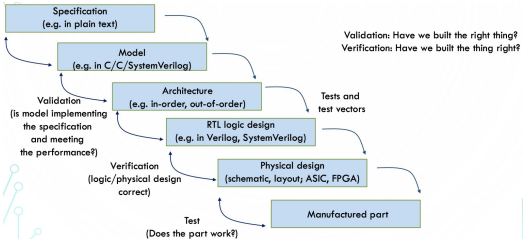
Administrivia

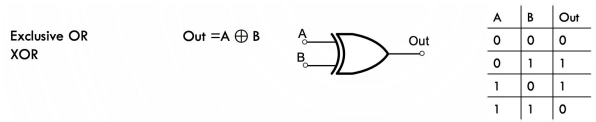
- Reminder that Monday & Wednesday discussions are in-person only
- Homework 1 due on Friday, 9/10
- Monday labs will roll-over to the next week due to the holiday
- Lab 1 due on Friday, 9/10
- Lectures
  - Ironing out the overflow room, wifi
  - Watch the recorded ones - power, performance, cost, robustness
    - The course will explore quantitative optimization of these metrics
- Reminder that labs are individual and projects will be in pairs

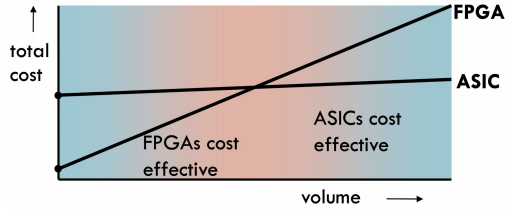
What has the course discussed so far?

- Semiconductor trends
  - Moore's Law? Dennard scaling?
- Design process
  - Abstraction, hierarchy
- Boolean logic
- ASICs and FPGAs
  - Full custom -> standard cells; CAD
  - Gate arrays -> FPGAs
- Lots of background
  - Now starting engineering material



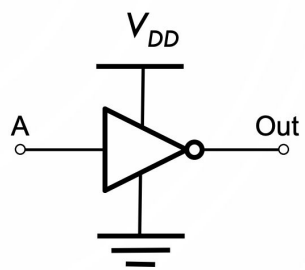


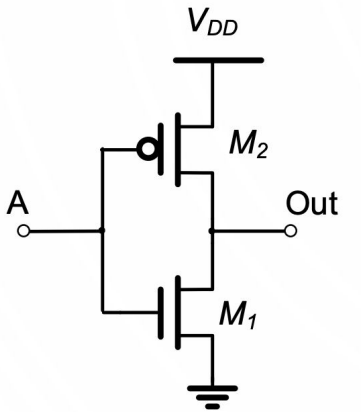


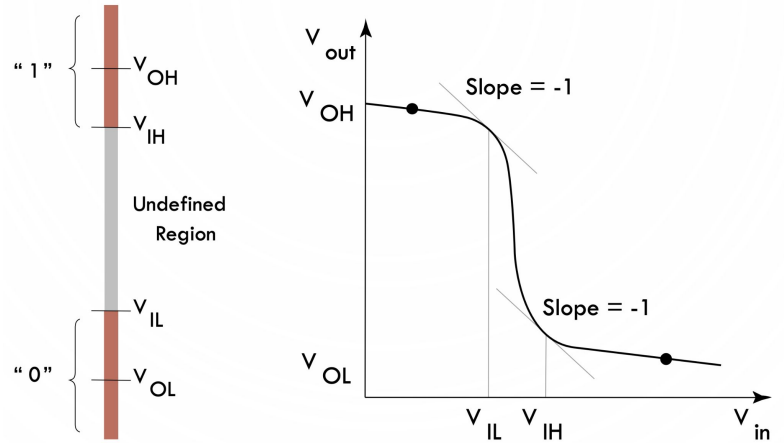


Digital?: CMOS Inverter

- Complementary Metal Oxide Semiconductor
  - Inverter has 1 NMOS and 1 PMOS
- in: low -> out: high; in: high -> out: low
  - Low (0): Vin < VL or Vin < Vth,n
  - High (1): Vin > VH or Vin > VDD - Vth,p
- NMOS and PMOS each have an “on” and “off” resistance
- Voltage Transfer Curve (VTC)
  - Mid/switching-point is a function of PMOS vs NMOS strength

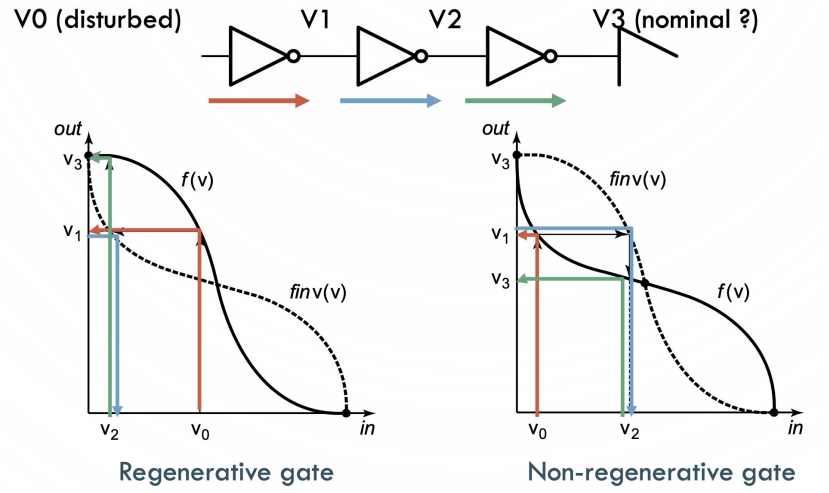


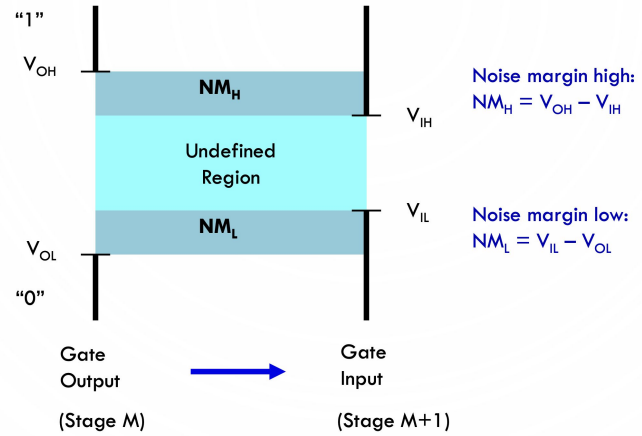




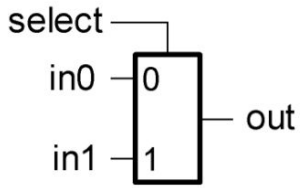
Noise Margins

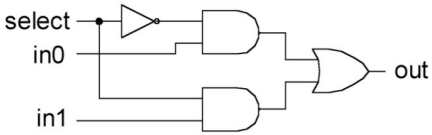
- Noisy, but still want 1s and 0s (digital)
  - “Restoration” / “regeneration” property
- Noise margins
  - The amount of noise that can be tolerated such that the signal can be correctly interpreted by the next gate
  - Eg. in a chain of inverters
- VTC is low gain/stable near supplies, high gain regime between





Hardware Description Languages





- Languages that describe hardware
  - Writing hardware, not software! (despite some common syntax)
  - Hardware is running all in parallel at all times
- Examples
  - Verilog, VHDL, SystemVerilog, Chisel, …
  - Verilog is language of choice for this course and most companies
- What is RTL? -> Register Transfer Level
  - Describe synchronous digital systems with operations on signals flowing between registers
- Not everything you can write is “synthesizable”
- Build up systems in lab from very simple -> complex!

```
module mux2 (
  input [1:0] in,
  input select,
  output out);
assign out = select ? in[1] : in[0];
endmodule
```

Verilog

- Let’s go over some of the Verilog Primer Slides and some previous discussion Verilog slides
- The primer slides and some other resources can be found on the course website

Blocking vs Non-Blocking

Blocking

```
reg c, out;
wire a, b, d;
always @(*) begin
    c = a | b;
    out = c & d;
end
```

Non-Blocking

```
reg c, out;
wire a, b, clk;
always @(posedge clk) begin
    //use val of ‘out’ before clk edge
    c <= out | a;
    //use val of ‘c’ before clk edge
    out <= c & b;
end
```

- Don’t mix blocking and non-blocking

Revisiting Last Discussion Examples

```
module mux2 (
  input [1:0] in,
  input select,
  output out);
assign out = (select = 1) ? in[1] : in[0]
endmodule
```

```
module mux2_behav (in,out,out_b)
  input [0:1] in;
  input select;
  output out;
  output out_b;
always @ (in)
  if (select == 1)
    out = in[1]
    out_b = ~in[1]
  else
   out = in[0]
   out_b = ~in[0]
endmodule
```

Corrected Example

```
module mux2 (
  input [1:0] in,
  input select,
  output out);
assign out = select ? in[1] : in[0];
endmodule
```

```
module mux2_behav (in,select,out,out_b);
  input [1:0] in;
  input select;
  output reg out;
  output reg out_b;
always @ (*) begin
  if (select == 1) begin
    out = in[1];
    out_b = ~in[1];
  end else begin
   out = in[0];
   out_b = ~in[0];
  end
end
endmodule
```

What you'll do in lab 2:

- Verilog basics
- Simulation
- Very helpful to draw out diagrams
  - For labs and the project, try to keep a set of digital diagrams that you can edit and add to
  - They will help you design more efficiently and can go straight into your project report!
    - Draw.io, omnigraffle, etc.
- Learn by doing!
  - Writing Verilog
  - Simulating and writing testbenches
  - Debugging
  - Check your logs for warnings -> the tools can sometimes obscure errors