

# EECS 151/251A

## Discussion 1

Alisha Menon

8/30/21, 8/31/21, and 9/1/21

# About me (Alisha Menon)

- 4<sup>th</sup> yeah PhD student advised by Jan Rabaey
- Research in hardware-efficient biomedical classification, devices and systems
- Hobbies: dance, biking, tennis



# My job:

- To help **you** get the most out of this class!
- Discussion sections
  - Monday 3-4pm, social sciences building 170
  - Tuesday 8-9am, virtual
  - Wednesday 3-4pm, genetics and plant bio 107
  - Review the past week
  - Answer questions
  - Example problems
- Office hours
  - Alisha's: Monday 2-3pm
  - A GSI will have one everyday, times listed on course website:
    - <https://inst.eecs.berkeley.edu/~eecs151/fa21/>
- Piazza
  - Please contact GSIs here instead of email!
  - GSIs will try to respond within 24 hours

# Recommendations for success

- Put the most effort into labs/project!
  - They make you a great **engineer**, not just a good IC student
    - Understand **abstraction** leverage it for productive design
    - Stay in circuit design: Apple shows you how desperate they are!!!
  - Choose partners wisely
    - Balance your strengths/weaknesses
- Homework stresses **understanding**, not correctness
  - Open-ended questions have higher weighting
- Stay up-to-date on industry & research trends!
  - IEEE [Computing & Semiconductors](#), [EE Times](#), [Semiconductor Engineering](#), [TechInsights](#)
  - IEEE [CAS](#) & [Computer Societies](#), [ACM](#), etc.

# Agenda

- Administrativia
- ASIC vs FPGA
- Simple logic gates
- Boolean logic problem
- Verilog intro

# Administrivia

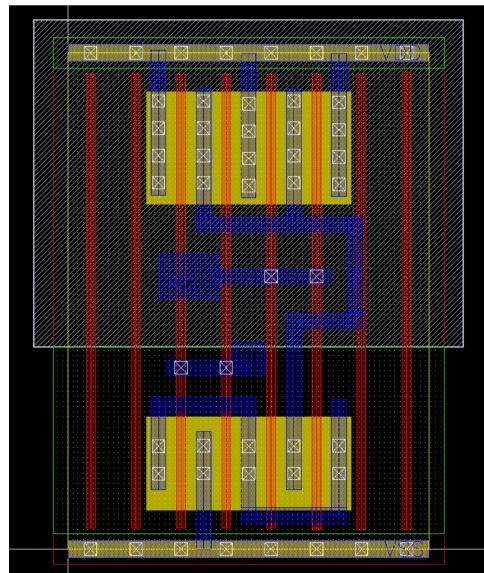
- Homework 1 will be posted this week
- Discussions are 3 times a week — please pick one
  - We have to account for capacity limitations this semester
- Labs are in Cory 111/117

# ASIC vs FPGA

# ASIC

- Application-Specific Integrated Circuit
- Use standard cells, SRAM, custom analog circuits

2-input NAND gate



Metal: Blue  
Transistors: Red + Yellow

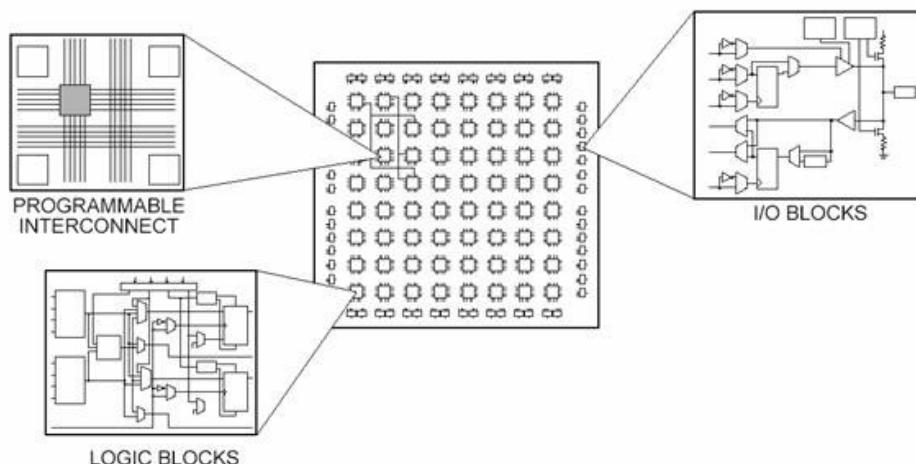
Apple's A11 Bionic SoC



<https://www.macworld.com/article/3275567/iphone-ipad/apple-7nm-a12-chip-iphone.html>

# FPGA

- Field-Programmable Gate Array
- Use LUTs, block RAM, on-die IP cores



Typical FPGA Structure



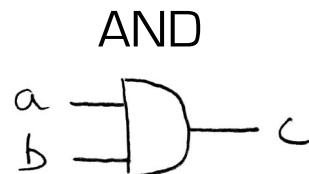
Xilinx Virtex Family

# ASIC vs FPGA

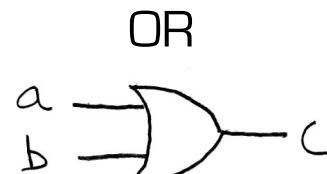
	ASIC	FPGA
Performance	maximum $\uparrow$	lower $\uparrow$
Design cost	higher - non-recurring engineering $\uparrow$	low $\uparrow$
Per-unit cost	low $\uparrow$	higher $\uparrow$
Design time	years $\uparrow$	minutes to program $\uparrow$
Custom blocks	custom analog, etc. $\uparrow$	limited to what is one FPGA
Job perspective	big tech	EE and non-EE fields

# Logic gates

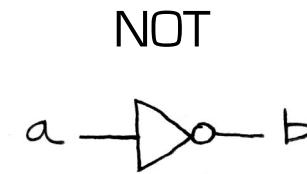
# Simple logic gates



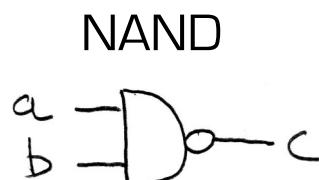
$a$	$b$	$c$
0	0	0
0	1	0
1	0	0
1	1	1



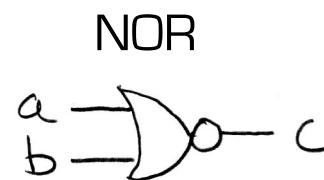
$a$	$b$	$c$
0	0	0
0	1	1
1	0	1
1	1	1



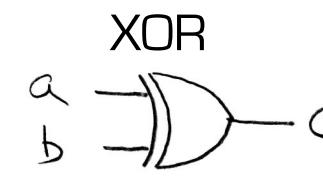
$a$	$b$
0	1
1	0



$a$	$b$	$c$
0	0	1
0	1	1
1	0	1
1	1	0



$a$	$b$	$c$
0	0	1
0	1	0
1	0	0
1	1	0



$a$	$b$	$c$
0	0	0
0	1	1
1	0	1
1	1	0

- Logic gates are often the primitive elements out of which combinational logic circuits are constructed.
  - In some technologies, there is a one-to-one correspondence between logic gate representations and actual circuits (ASIC standard cells have gate implementations).
  - Other times, we use them just as another abstraction layer (FPGAs have no real logic gates).
- How about these gates with more than 2 inputs?
- Do we need all these types?



# Boolean logic

$$\begin{aligned} \rightarrow Y &= A \cdot \bar{B} + \bar{A} \cdot B \\ &= (A \cdot \bar{B}) + (\bar{A} \cdot B) \end{aligned}$$

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

- What basic function does this represent?

$$\text{XOR: } y = A \oplus B = A \wedge B$$

# Boolean expression standard forms

- Same truth table can have many gate realizations
- However, we can find standard forms for a boolean expression
- Two types:
  - Sum of Products (SOP)
  - Product of Sums (POS)

# Sum of products using minterms

minterms	a	b	c	f
a'b'c'	0	0	0	1
a'b'c	0	0	1	1
a'bc'	0	1	0	1
a'bc	0	1	1	0
ab'c'	1	0	0	0
ab'c	1	0	1	0
abc'	1	1	0	0
abc	1	1	1	0

minterm: product term  
consisting of input combos

Sum of minterms = = 1

$$f = a'b'c' + a'b'c + a'bc'$$

# Product of sums using maxterms

Due to De Morgan's theorem:  $(A+B+C)' = A' * B' * C'$  ↪

maxterms	a	b	c	f
a+b+c	0	0	0	1
a+b+c'	0	0	1	1
a+b'+c	0	1	0	1
a+b'+c'	0	1	1	0
a'+b+c	1	0	0	0
a'+b+c'	1	0	1	0
a'+b'+c	1	1	0	0
a'+b'+c'	1	1	1	0

maxterm: sum terms consisting  
of <sup>inverted</sup> input combos

find maxterms = 0

$$f = (a+b'+c')(a'+b+c)$$

$$(a'+b+c')(a'+b'+c)$$

$$(a'+b'+c')$$

# Simplification – Boolean algebra laws

Operations with 0 and 1 (annulment/identity laws)	$x + 0 = x$ $x + 1 = 1$	$x \cdot 0 = 0$ $x \cdot 1 = x$
Idempotent law	$x + x = x$	$x \cdot x = x$
Involution (double negation) law	$\bar{\bar{x}} = x$	
Complementarity	$x + \bar{x} = 1$	$x \cdot \bar{x} = 0$
Commutativity	$x + y = y + x$	$x \cdot y = y \cdot x$
Associativity	$(x + y) + z = x + (y + z)$	$(x \cdot y) \cdot z = x \cdot (y \cdot z)$
Distributivity	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	$x + (y \cdot z) = (x + y) \cdot (x + z)$
Simplification	$x \cdot y + x \cdot \bar{y} = x$ $x + x \cdot y = x$	$(x + y) \cdot (x + \bar{y}) = x$ $x \cdot (x + y) = x$

Example:  $a'b'c' + a'b'c + a'bc' = a' (b'c' + b'c + bc')$

$$\begin{aligned}
 &= a' (b' (c' + c) + bc') \\
 &= a' (b' + bc') = a'b' + a'bc'
 \end{aligned}$$

# Verilog

# Hardware description language (HDL)

- Used to describe a digital system for your tools to synthesize
- Keep reminding yourself: **I'm describing hardware, I'm not writing a program!**
- Always sketch out your circuit, even if just at a high level (modules, ports, connections)
- Get hands on practice! Don't spend too much time only reading references

# Verilog design types

## ✗ • Structural Verilog

- Directly describe the schematic in text format
- • Typically combinational logic circuits build out of basic gates and module instances

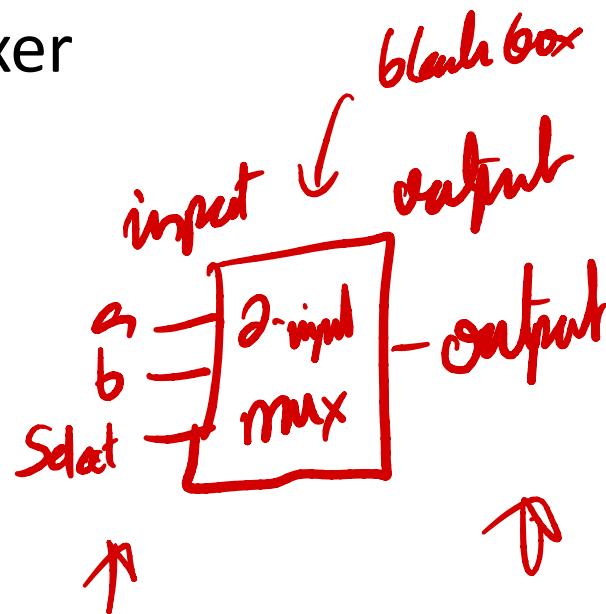
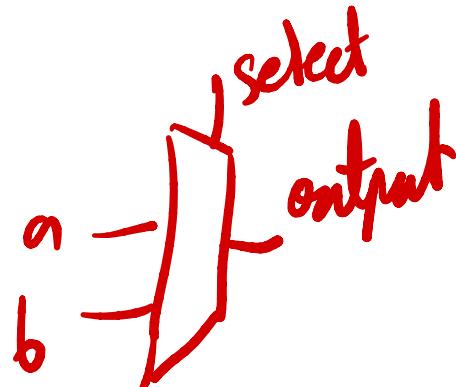
AND ( $a, b, c$ )

## ✗ • Behavioral Verilog

- Describe the desired functionality
- Use assign statements for continuous assignment
- • Use always@ blocks to describe how a circuit behaves under certain conditions

# Verilog modules

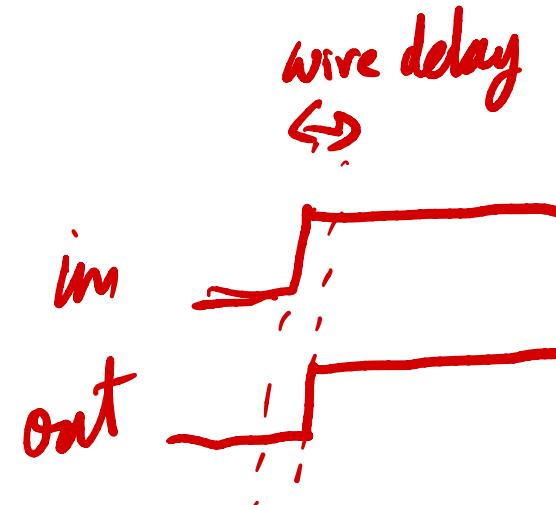
- Black box design unit with a specific purpose
- Designer specifies inputs, outputs, behavior
- Example: 2-input multiplexer



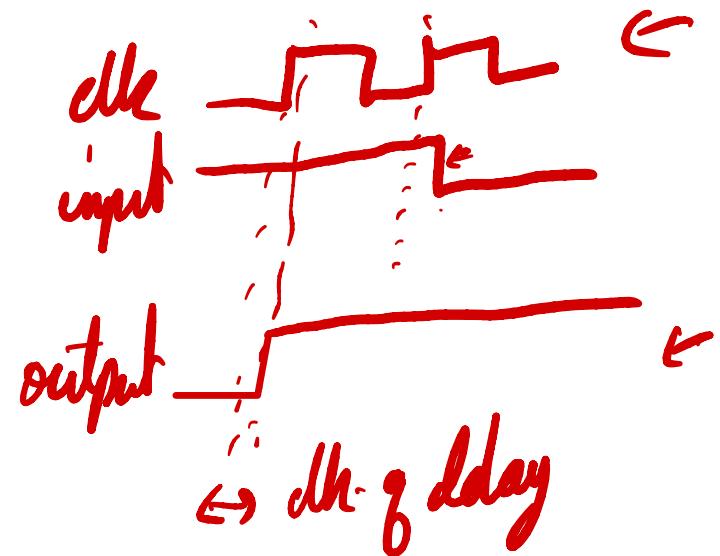
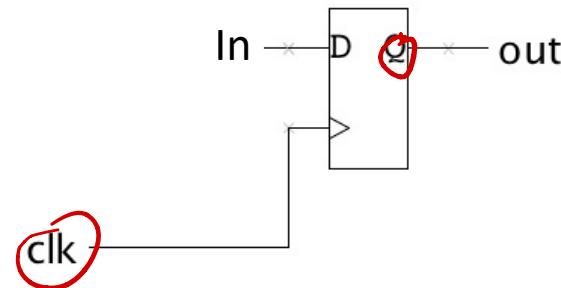
# Wire vs. Register

- Wire

In — Out



- Register



# Examples (with common syntax errors)

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

→ *declarations*

→ *cond ? if true : if false*

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

→ *select*

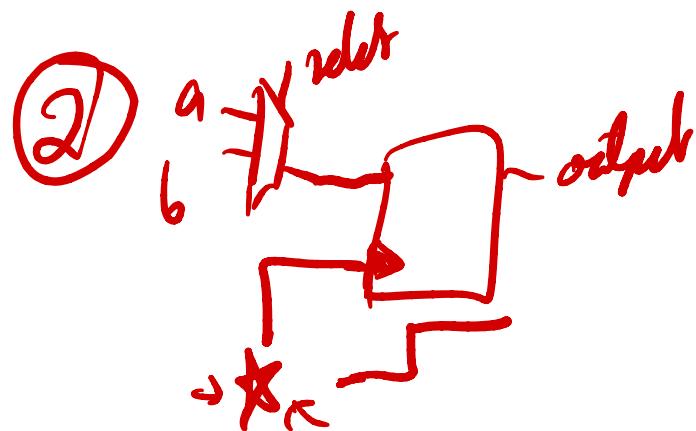
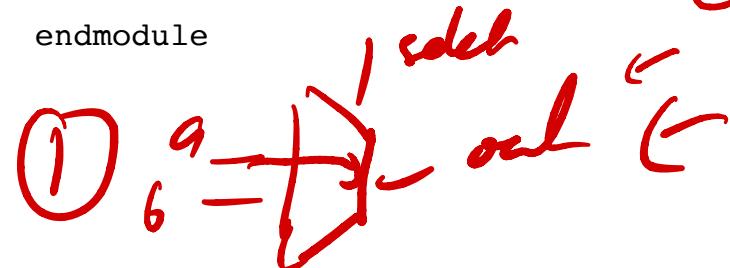
→ *dedata*

→ *\* begin begin*

→ *end end*

# Examples (corrected)

```
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
```

days @ (posedge clk)