

Intro to Digital Design

L1: Combinational Logic

Instructor: Naomi Alterman

Teaching Assistants:

Derek de Leuw

Isabel Froelich

Kevin Hernandez

Aarjav Jain

Packard Stephenson

Introducing your instructor

- ❖ Professor Naomi
 - Electrical engineer by training
 - Bopped around Silicon Valley hacking on everything from internet backbone routing chips to OS kernels to LIDAR firmware to mobile graphics libraries
 - Discovered in industry that computers are boring
 - But *people* on the other hand...
 - Proud cat mom to Danni (aka Her Royal Majesty Queen Baby)



Introducing your TAs

❖ The dream team!

Derek



Isabel



Kevin



Aarjav



Packard

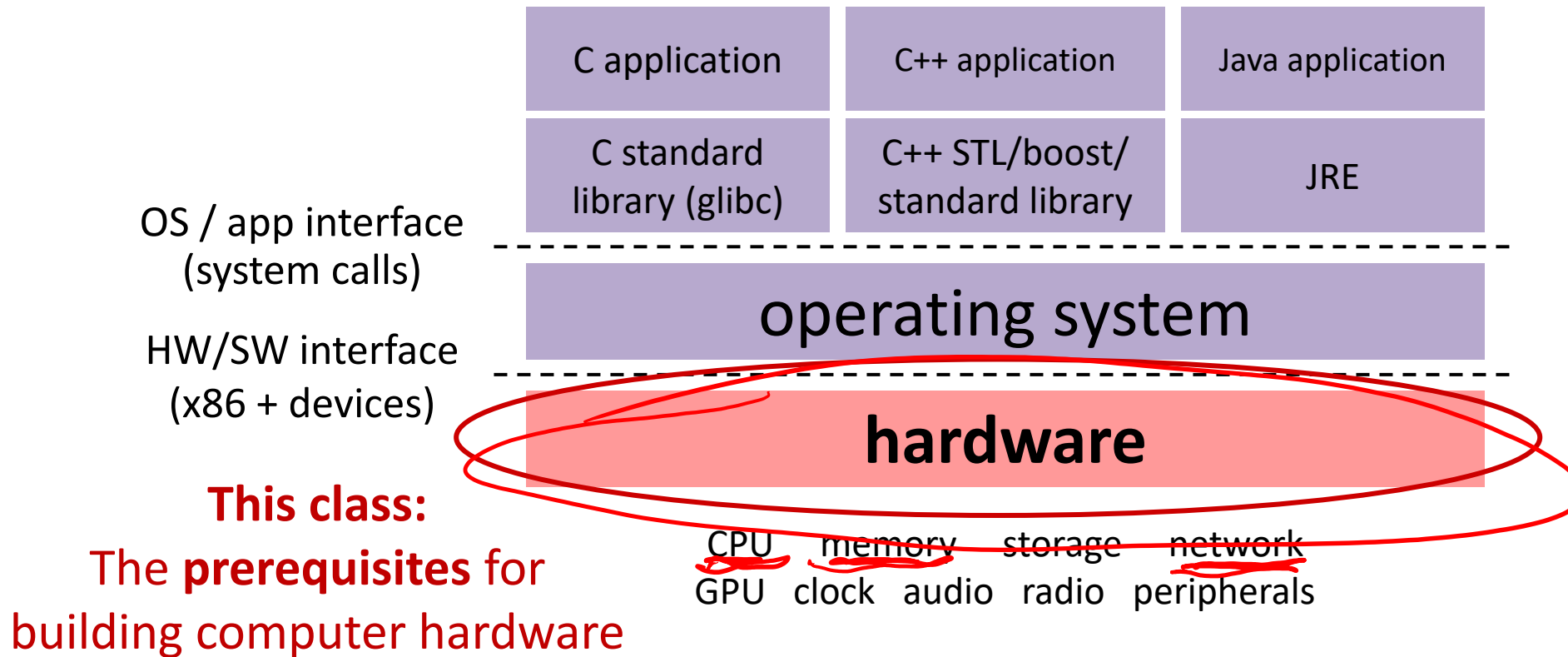


- Run labs and section, around for office hours and on Ed
- An **invaluable** source of information and help
- ❖ Get to know us – we are here to help you succeed!

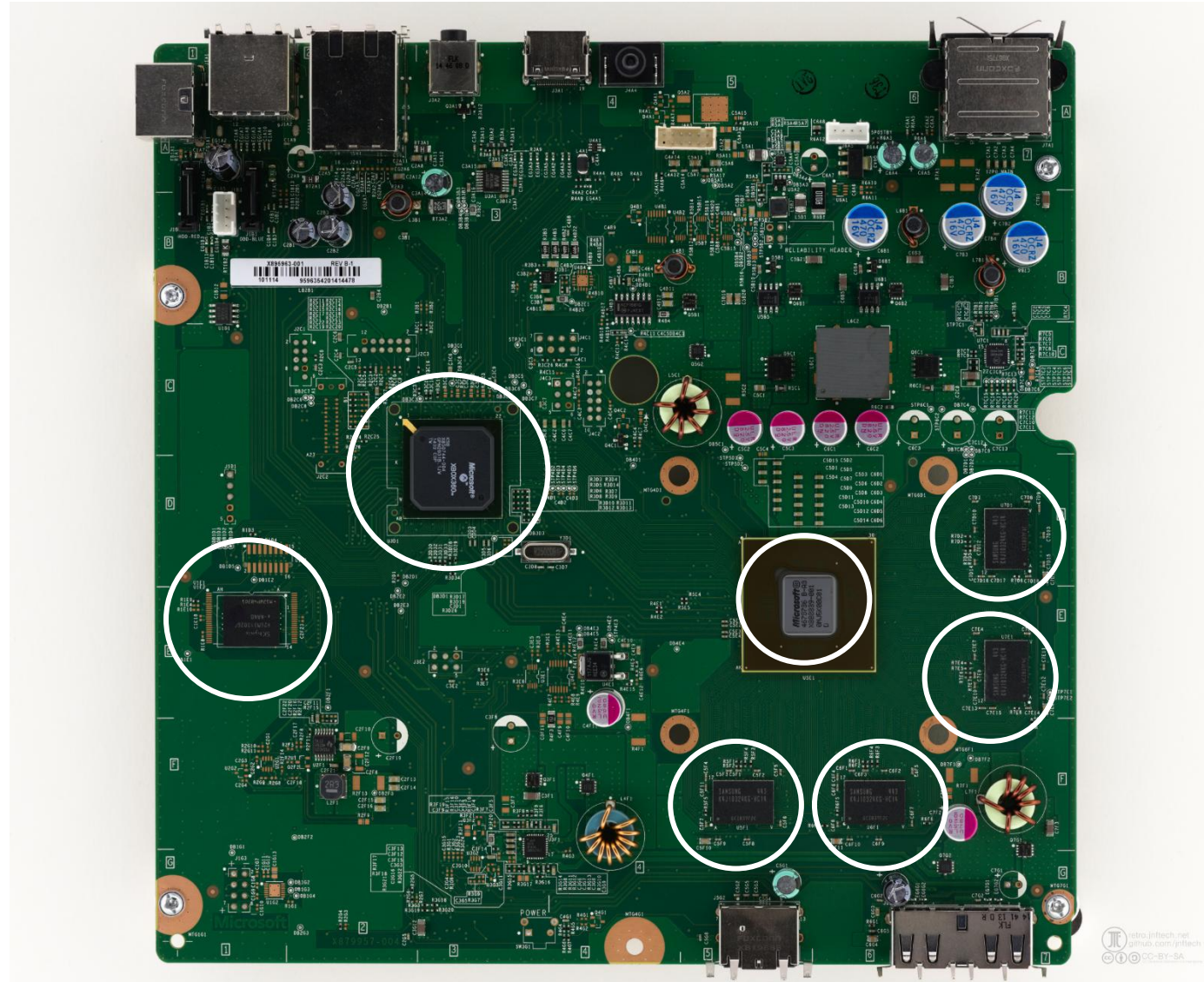
Lecture Outline

- ❖ **What's this class all about?**
- ❖ Course Logistics
- ❖ Combinational Logic Review

Course Motivation



Digital Electronics



Course Motivation

- ❖ The real question:

- How do you build a machine that interprets assembly code? 🤔 🤔 🤔

- ❖ Let's break that into two smaller questions:

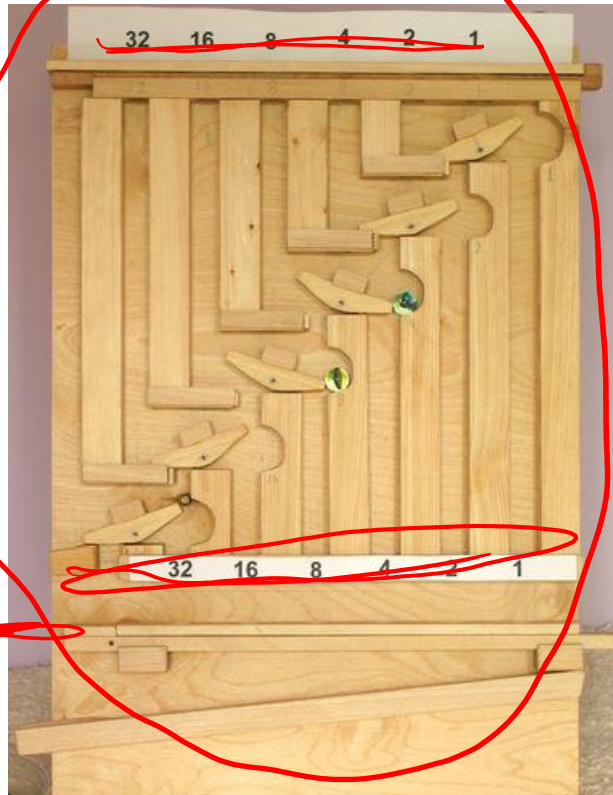
- How do you represent data inside a machine?
- How do you build components that manipulate said data?

Course Motivation

❖ The real question:

- How do you build a machine that interprets assembly code? 🤖 🤖 🤖

Marbles?



<https://woodgears.ca/marbleadd/>



Course Motivation

- ❖ The real question:
 - How do you build a machine that interprets assembly code? 🤔 🤔 🤔

Minecraft?



<https://tylerdmace.on.fleek.co/posts/redstone-computers/>

Course Motivation

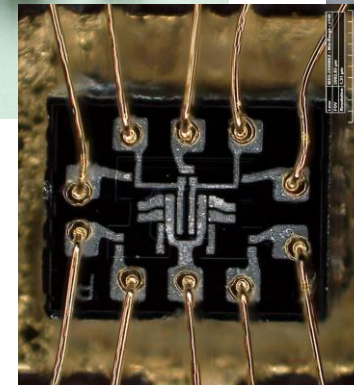
❖ The real question:

- How do you build a machine that interprets assembly code? 🤔 🤔 🤔

Electronic Transistors!



"ICs" →



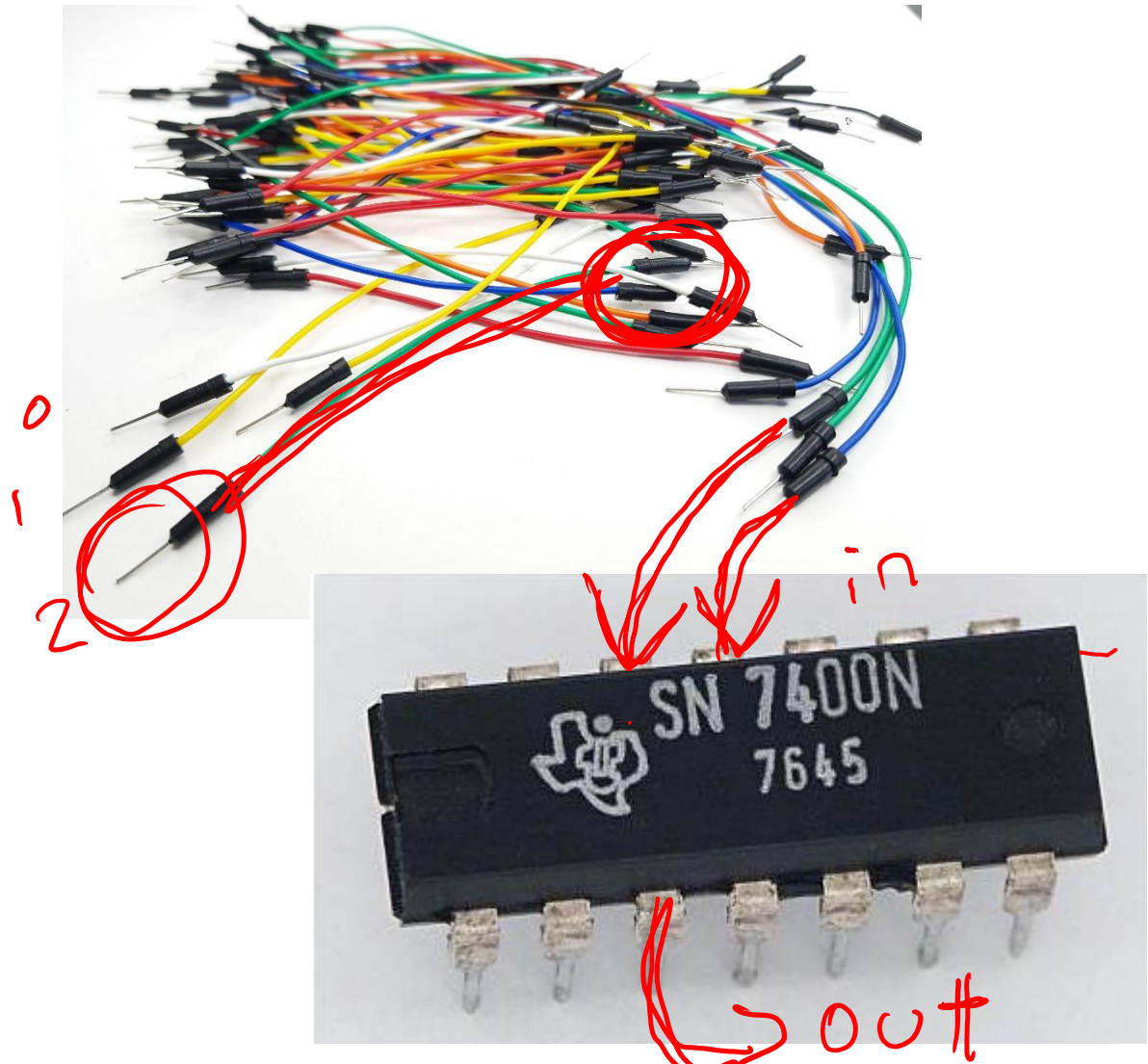
https://en.wikipedia.org/wiki/Vacuum_tube#/media/File:Triody_var.jpg

https://www.youtube.com/watch?v=F8gwex11I_E

<https://www.righto.com/2019/09/a-computer-built-from-nor-gates-inside.html>

Digital Electronics

- ❖ Our data:
 - We'll use individual metal wires to move single bits around
 - When the **voltage** on a wire is low, that wire is transmitting a "0". When it's high, it's a "1".
- ❖ Our manipulations:
 - "Logic gates" built out of transistors
 - The output voltage is always some computation based on the input voltages. **Always. Immediately.**



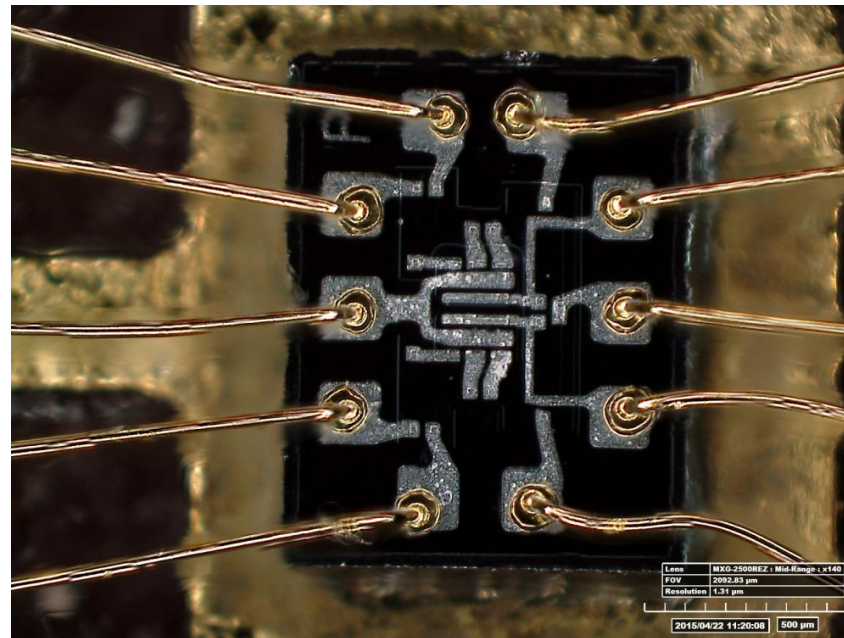
<https://diyables.io/products/jumper-wires-male-to-male-round-head>
https://en.wikipedia.org/wiki/Logic_gate#/media/File:TexasInstrument_s_7400_chip,_view_and_element_placement.jpg

Logic gates

NOR

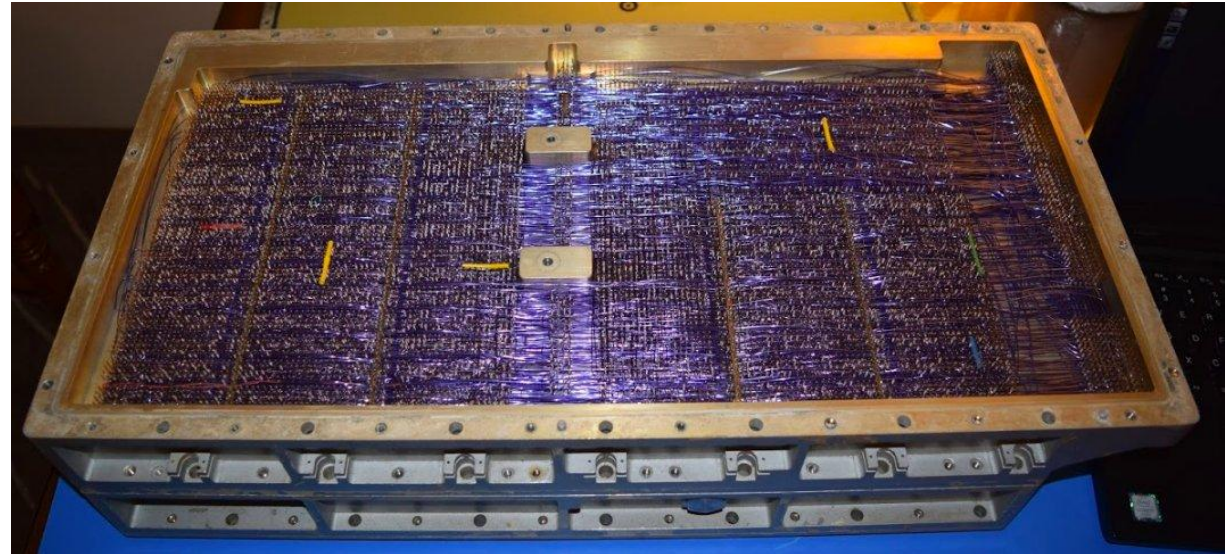
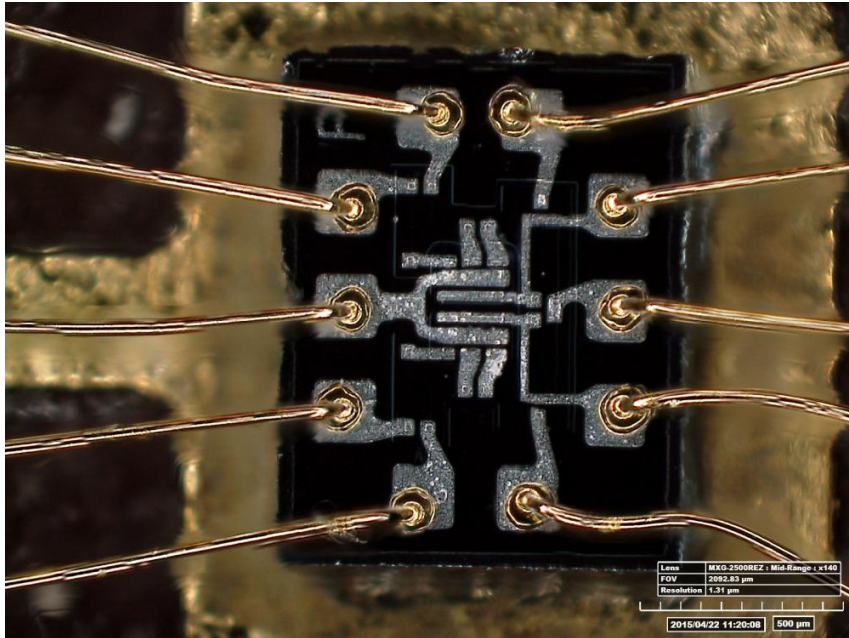


A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0



By the way

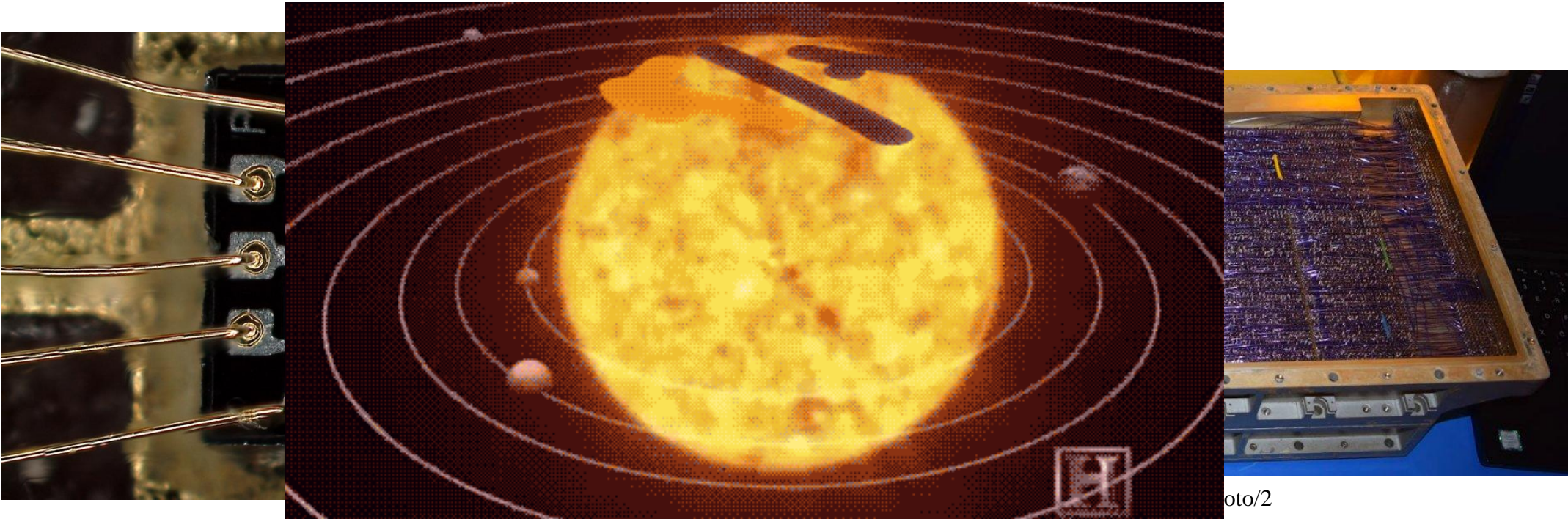
- **This NOR gate was literally the only kind of chip used in the Apollo Guidance Computer**
(there were just thousands upon thousands of them)



<https://x.com/kenshirriff/status/1237818694379556864/photo/2>

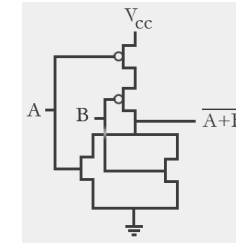
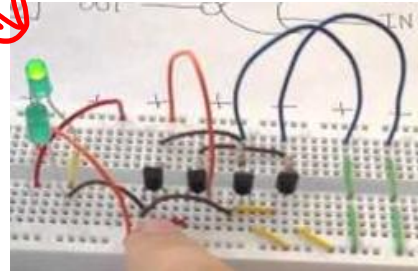
By the way

- This NOR gate was literally the only chip used in the Apollo Guidance Computer



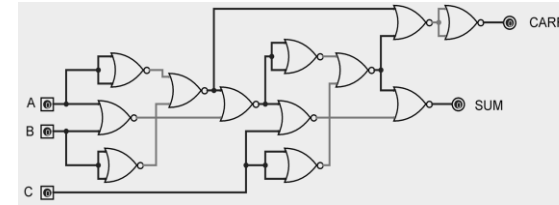
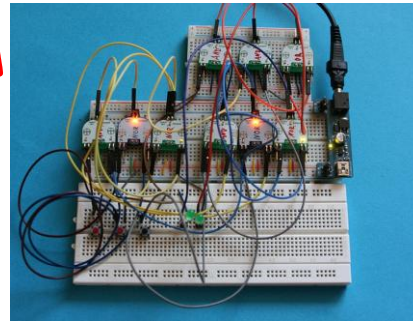
- https://www.sccs.swarthmore.edu/users/06/adem/engin/e77vlsi/lab3/logic_nor.gi
- <https://www.youtube.com/watch?v=TnltHE0Wp8Y>
- <https://www.ahirlabs.com/2017/06/09/adder-and-subtractor/>
- http://www.justgeek.de/wp-content/uploads/2014/07/IMG_0173.jpg
- https://www.researchgate.net/figure/ALU-block-diagram_fig1_312203298
- <https://hackaday.com/2017/06/16/homemade-computer-from-1970s-chips/>
- <https://www.cise.ufl.edu/~mssz/CompOrg/CDA-proc.html>
- https://commons.wikimedia.org/wiki/File:AMD_K5_PR75_die.JPG

x12



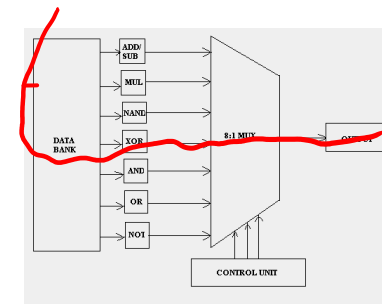
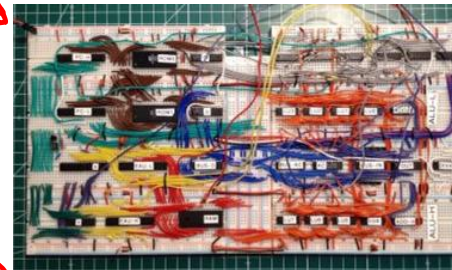
NOR gate

~ x60

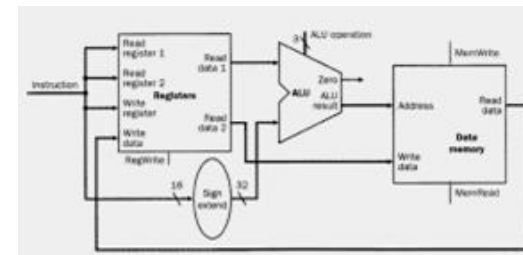
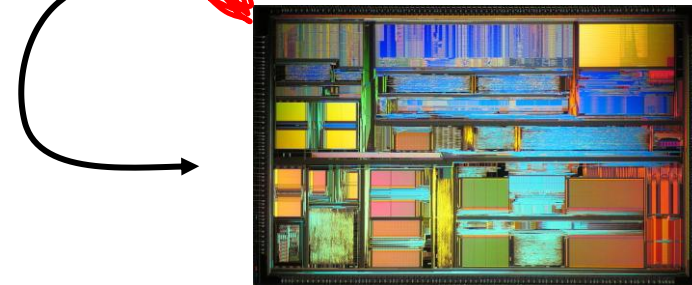


Full Adder

~ x1500



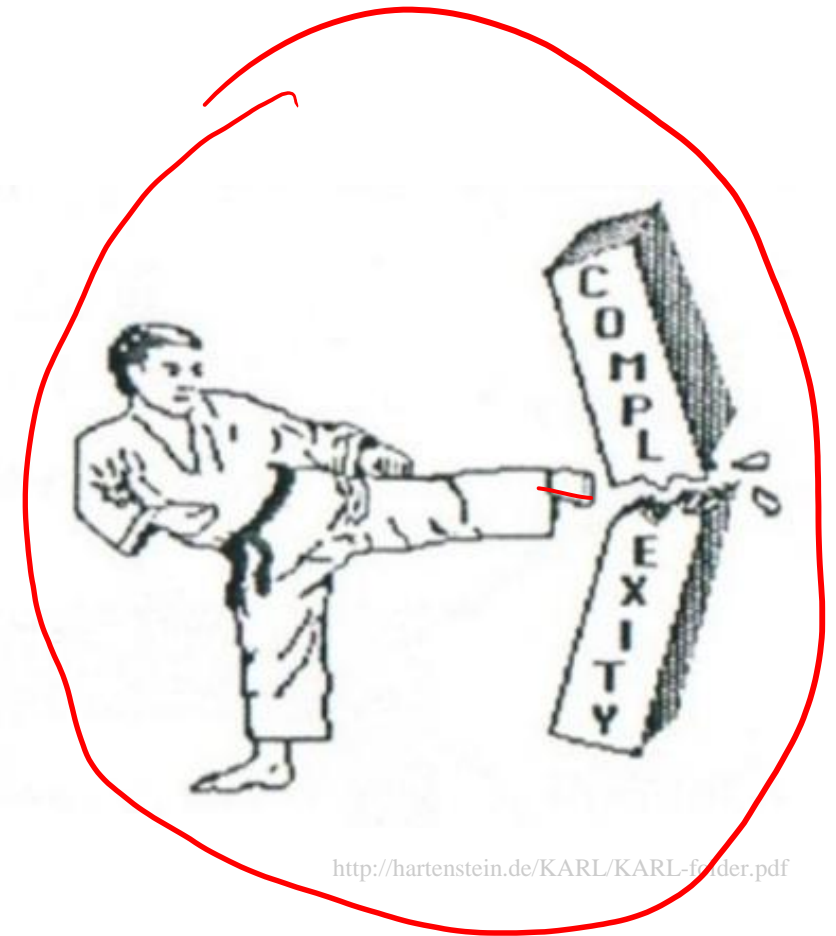
ALU



CPU

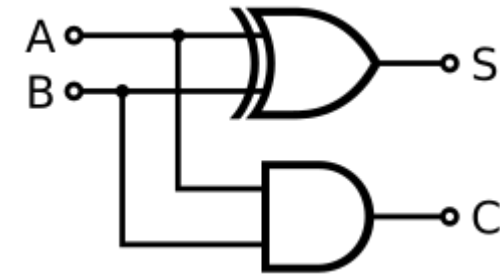
VLSI

- ❖ **Very Large Scale Integration** is the term used for the modern process of making computer chips (since the mid 80s)
- ❖ **The idea:** as our machines get more complex, we need to find **methodological and technological solutions** to mitigate that complexity
- ❖ This means in this class we're **not** working at the transistor level (**thank god**)

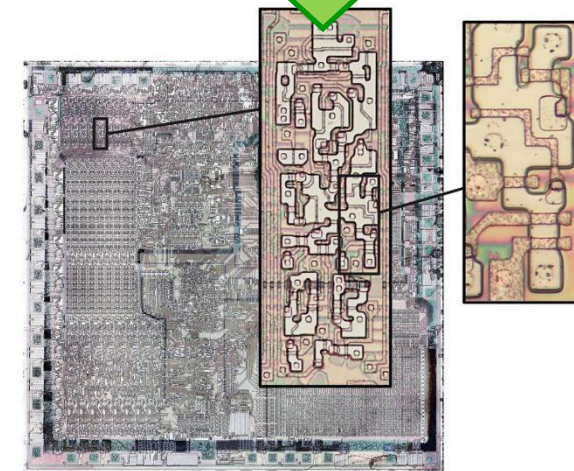


Digital design methodology

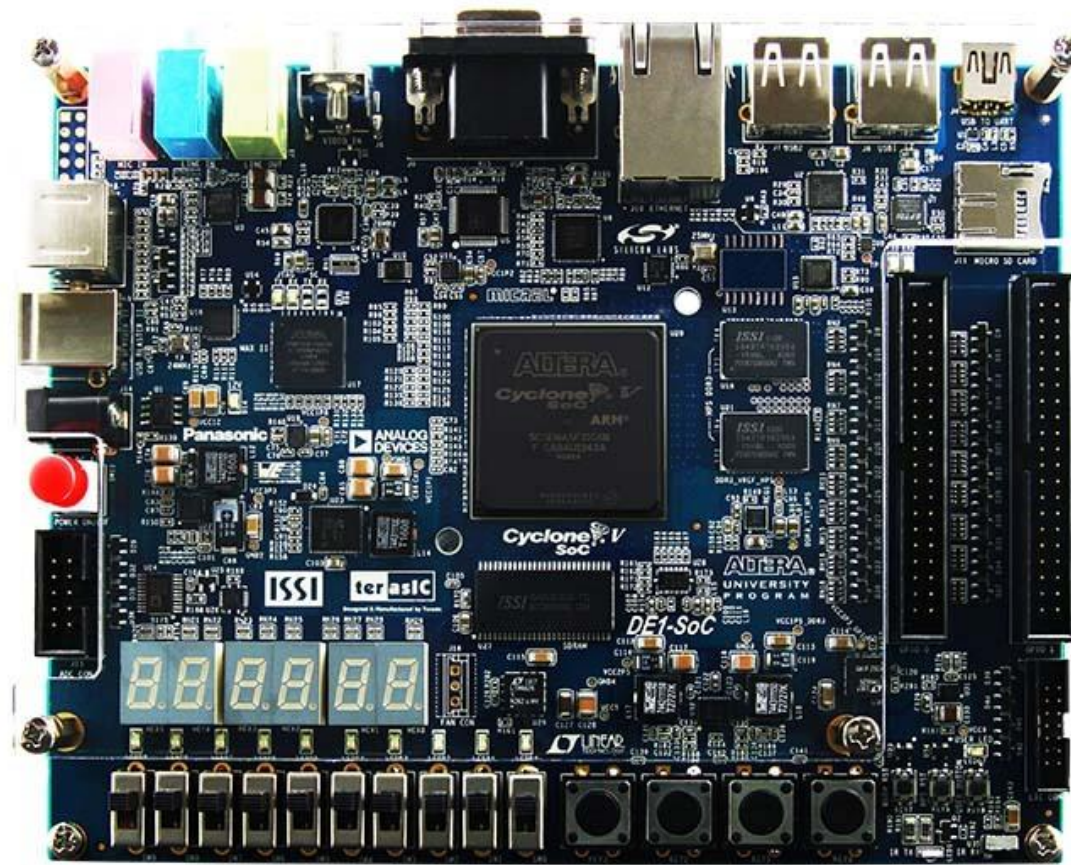
- ❖ Workflow:
 - Draw a “**block diagram**” / “**schematic**” at the **logic gate level**
 - Translate this diagram into a **hardware description language (HDL)** called **Verilog**
 - 🚨 🚨 It’s not computer code, even though it looks a lot like it 🚨 🚨
 - Use **electronic design automation (EDA)** tools to “**synthesize**” the Verilog into a physically realizable electronic circuit
 - Spend an easy couple mil to **etch** it into silicon \$\$\$
 - ..or be a cheapskate and load the circuit onto a \$50 **Field-Programmable Gate Array (FPGA)**



```
module half_adder (input logic a,
input logic b, output logic carry,
output logic sum)
    assign carry = a & b;
    assign sum = a ^ b;
endmodule
```



FPGAs: our prototyping hardware



Creativity within constraints

- ❖ Digital design is in some ways more art than a science
 - The creative spirit is in combining primitive elements and other components in new ways to achieve a desired function
- ❖ However, unlike art, we have objective measures of a design (*i.e.*, constraints):
 - Performance
 - Power
 - Cost

General digital logic learning progression

369
971

1. Learn the building blocks and theory for building machines that manipulate binary data

- Work at the “logic gate” level of abstraction

469

2. Learn how to use said building blocks to build a CPU that reads and executes machine code

- Still at the “logic gate” level

~~567~~

3. Learn how to build logic gates (and other digital components) out of transistors

- We start paying attention to electricity here

567
EX-476

Can do this in Minecraft 🙌

Cannot do this in Minecraft 😬

Miso Moment



Lecture Outline


- ❖ What's this class all about?
- ❖ **Course Logistics**
- ❖ Combinational Logic Review

Bookmarks



- ❖ Website: <https://cs.uw.edu/369/>
 - Schedule (lecture slides, lab specs), weekly calendar, other useful documents
- ❖ Ed Discussion: <https://edstem.org/us/courses/97192/>
 - Announcements, questions and answers – staff will monitor and contribute
- ❖ Gradescope: <https://www.gradescope.com/courses/1288555/>
 - Lab submissions, Quiz grades, regrade requests
- ❖ Canvas: <https://canvas.uw.edu/courses/1881169>
 - Grade book, lecture recordings

Grading

- ❖ Labs (66%)
 - 7 regular labs – 1 week each
 - Labs 1-4: ~50-60 points each, 5-7: ~100 points each
 - 1 “final project” – 2 weeks
 - Lab 8 Check-In: 10 points, Lab 8: 150 points
- ❖ 3 Quizzes (no final exam)
 - Quiz 1 (10%): 20 min in class on April 28th
 - Quiz 2 (10%): 30 min in class on May 19th
 - Quiz 3 (14%): 60 min in class on June 2nd
- ❖ This class uses a straight scale (> 95% → 4.0) 
 - Extra credit points count the same as regular points

Labs

- ❖ **Design** digital logic circuits using industry-standard methodologies and then **implement** them with Verilog HDL and an FPGA prototyping board

- ❖ You've got three graded deliverables:
 1. A written lab report outlining your design and showing simulation results
 2. Your design's Verilog "code"
 3. An in-person lab demo showcasing said code functioning on an FPGA

Getting Help

- ❖ Naomi's OH: Wednesdays 1:00-2:00p (or by appointment) in her office (CSE458)
 - Great for conceptual questions and big picture stuff (“🤔 🤔 🤔 ???”)
 - She can help debug things, but she's not gr8 at it

- ❖ TA OH: Always in lab (CSE003). See website calendar for times.

- Great for helping you deal with Verilog bugs and Quartus issues (“aRARHGG 🤔 🤔!”)

- ❖ Ed board:

- Great for generalizable questions about any course material
- Answer questions if you think you can! Peer learning, woo
- Make a private thread if you have questions about your specific Verilog code

- ❖ Check out this guide to Asking good questions

- 👍 Helpful: “I need help with my comparator code. The block diagram looks fine, but the simulation outputs the wrong value if the numbers are negative. Here's an example waveform.”
- 🙅 Less helpful: “I need help with lab 3, it doesn't always work.”

“How do i demo professionally 😎”



Section

- ❖ Discussion section on **Fridays 1:30-2:20** in **CSE2 271**
- ❖ Run by TAs
- ❖ Highlights the material that shows up on quizzes and the tips and tricks you'll need to keep your labs from becoming a living nightmare

 **Highly** recommended if you can make it.

- This was a recently added component to the course to fill learning gaps often reported by students

Lab Kits

- ❖ You'll get a lab kit for the quarter to test your designs on
 - Pick one up from CSE 003 this week (Wed + Thurs 2:30p – 5:20p)
 - Kit contains an FPGA board, USB cable, power adapter and “bonus” (~~unused~~) LED matrix
- ❖ Class software available here
 - Tragically, not compatible with ~~MacOS~~ or Apple Silicon. Make use of Windows computers in the lab, instead.



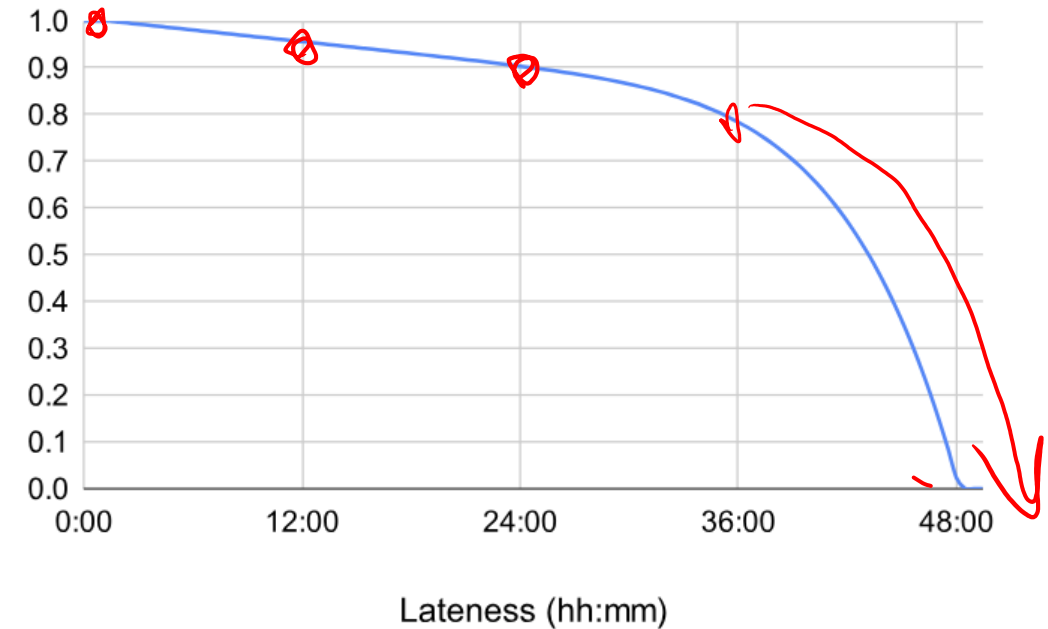
Lab Submissions and Demos

- ❖ Reports and code are due on **Wednesdays @ 2:30p regardless of your demo time**
 - Submitted on Gradescope
- ❖ Lab Demos:
 - 10 minute session with a TA to show off your working design and answer a few questions
 - Hours: Wed & Thu 2:30-5:20 pm (CSE 003)
 - We'll assign you a recurring time slot on based on your reported availability in the [class presurvey](#)



Late policy

- ❖ Late lab reports hit with an exponential penalty after the due date:
 - 5 minutes after the due date? No penalty
 - 1 day after due date? 90% of your score
 - 1.5 days after due date? 80% of your score
 - 2 days? 0% 🐱
- ❖ Exact penalty equation posted on syllabus page
- ❖ No late tokens
- ❖ You can reschedule your lab demo if you end up turning in your report late (eg, after when you'd have been demoing anyway)



Collaboration Policy

- ❖ Labs and project are to be completed *individually*
 - Goal is to give every student the hands-on experience
 - Violation of these rules is grounds for failing the class

- ❖ **OK:**
 - Discussing lectures and/or readings, studying together
 - *High-level* discussion of general approaches
 - Help with debugging, tools peculiarities, etc.

- ❖ **Not OK:**
 - Developing a lab together
 - Giving away solutions or having someone else do your lab for you

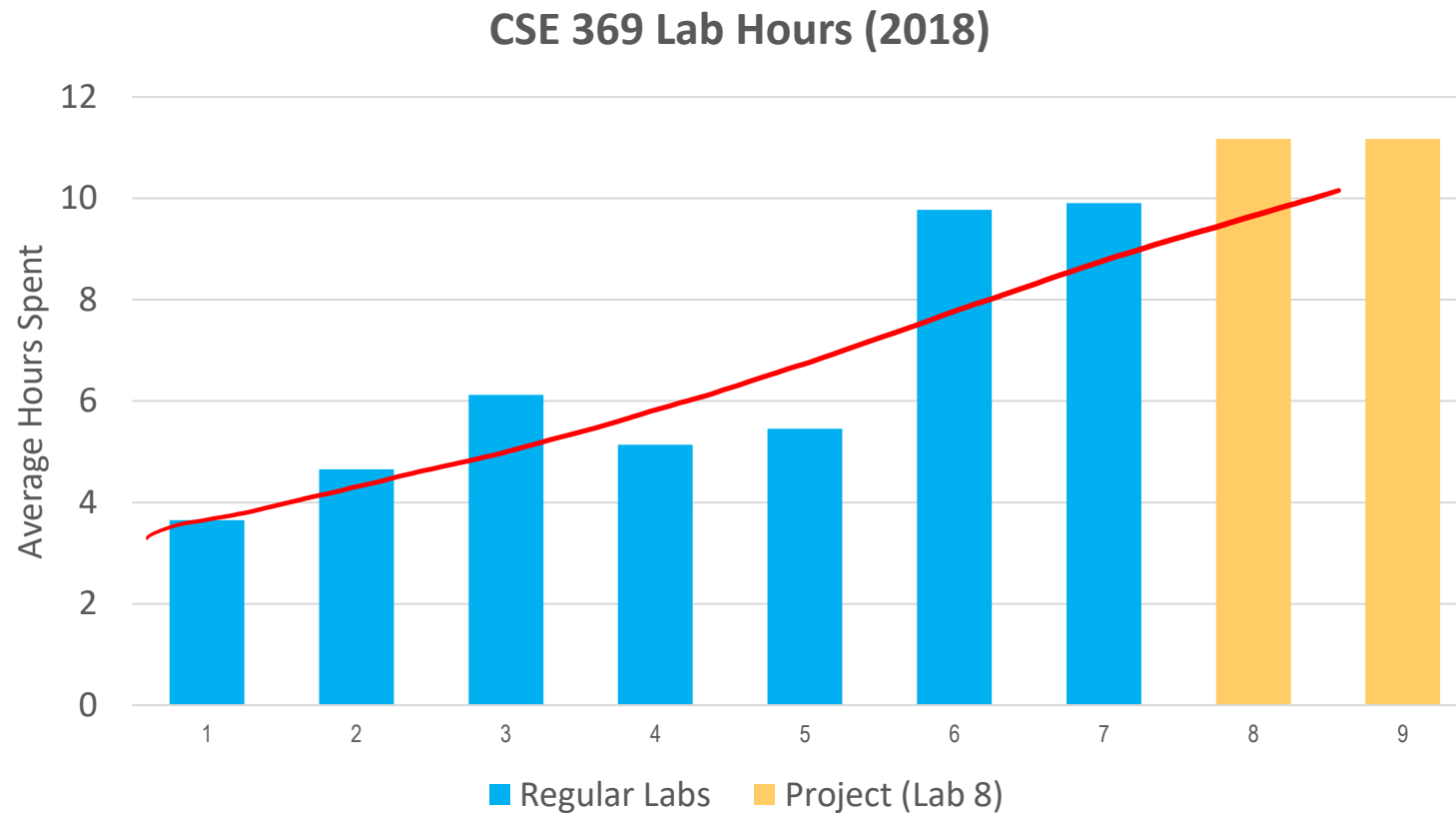
AI Policy

- ❖ Verilog code is the very last step of the digital design process; ideally, it's a 1:1 translation from the schematics you'll draw ahead of time into code.
 - The goal of this class is to help you build the skills to design these schematics yourself, from scratch.
 - In the real world you'll need these skills to develop designs, *regardless* of how (or if) an LLM is involved in their ideation.
- ❖ **TL;DR:** Effectively, same as the collaboration policy
- ❖ Examples of what is and isn't ok on [our website](#).



Course Workload

- ❖ The workload (3 credits) ramps up significantly towards the end of the quarter:



Lecture Outline

- ❖ What's this class all about?
- ❖ Course Logistics
- ❖ **Combinational Logic Review**

311 Reminder: Logic Gates (1/2)

schematic

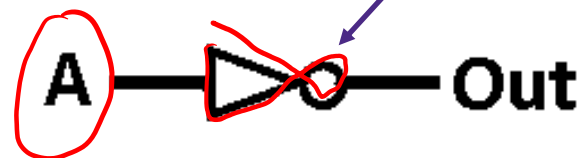
❖ Special names and symbols:

0/1
1/0
Circle indicates NOT

truth table

A

NOT

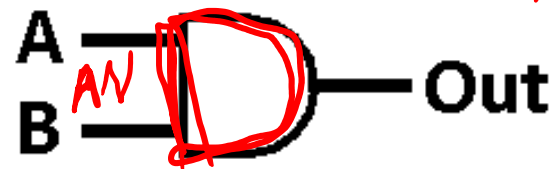


→

A	Out
0	1
1	0

*A * B / A AND B*

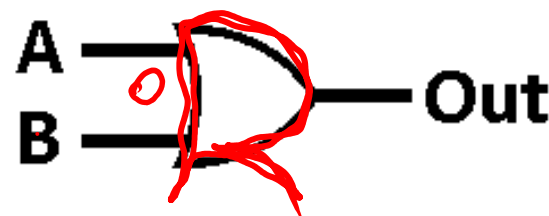
AND



A	B	Out
0	0	0
0	1	0
1	0	0
1	1	1

A + B / A OR B

OR

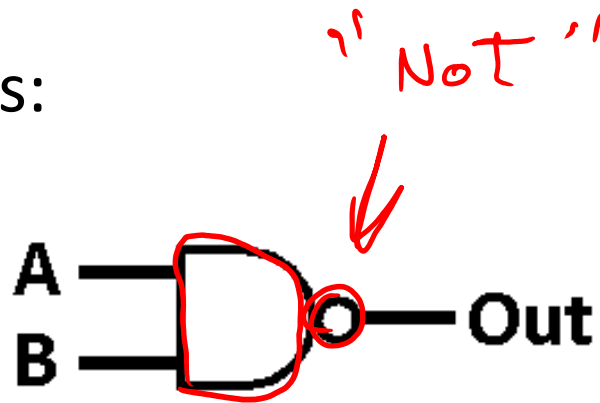


A	B	Out
0	0	0
0	1	1
1	0	1
1	1	1

311 Reminder: Logic Gates (2/2)

❖ Special names and symbols:

NAND

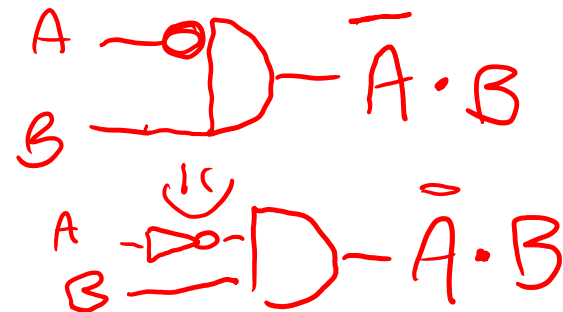


NOR



$A \oplus B$

XOR



A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

A	B	Out
0	0	0
0	1	1
1	0	1
1	1	0

} only 1 input is true

But this isn't 311...

- ❖ Let's *apply* Boolean logic



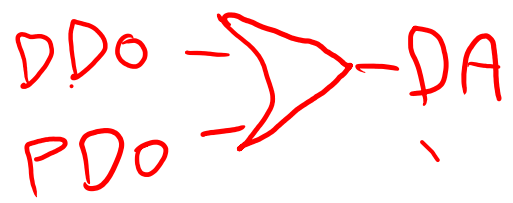
Example: Simple Car Electronics

- ❖ Output: **D**oor **A**jar
- ❖ Inputs: **D**river**D**oor**O**pen, **P**assenger**D**oor**O**pen



DDO	PDO	DA
0	0	0
0	1	1
1	0	1
1	1	1

$DA = DDO + PDO$



■ $DA = DDO + PDO$

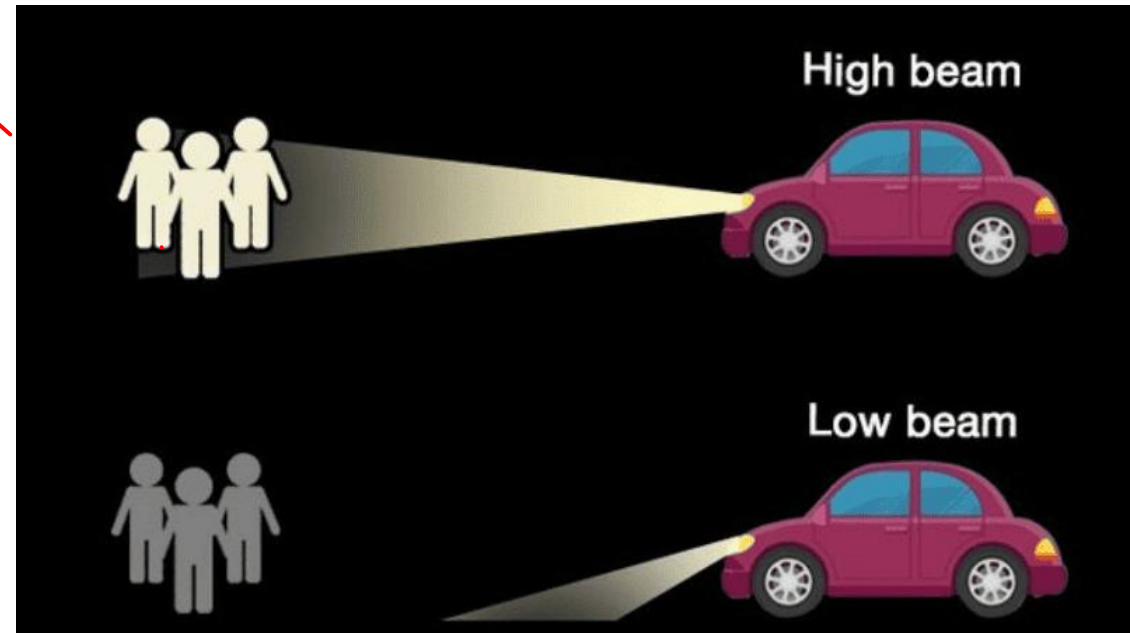


Example: Simple Car Electronics

- ❖ Output: **H**igh **B**eam **I**ndicator
- ❖ Inputs: **L**ights**O**n, **H**igh**B**eam**O**n

LO	HBO	HBI
0	0	0
0	1	0
1	0	0
1	1	1

■ $HBI = LO \cdot HBO$



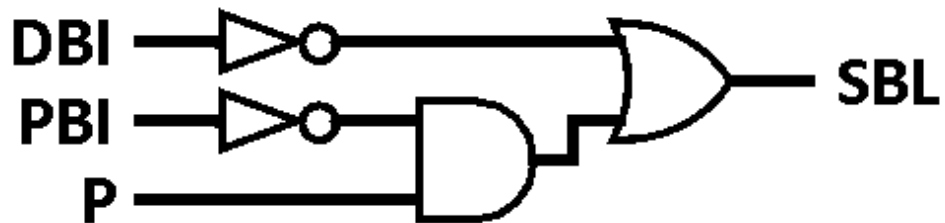
Example: Simple Car Electronics

- ❖ Output: **S**eat **B**elt **L**ight
- ❖ Inputs: **D**river **B**eltIn, **P**assenger **B**eltIn, **P**assenger

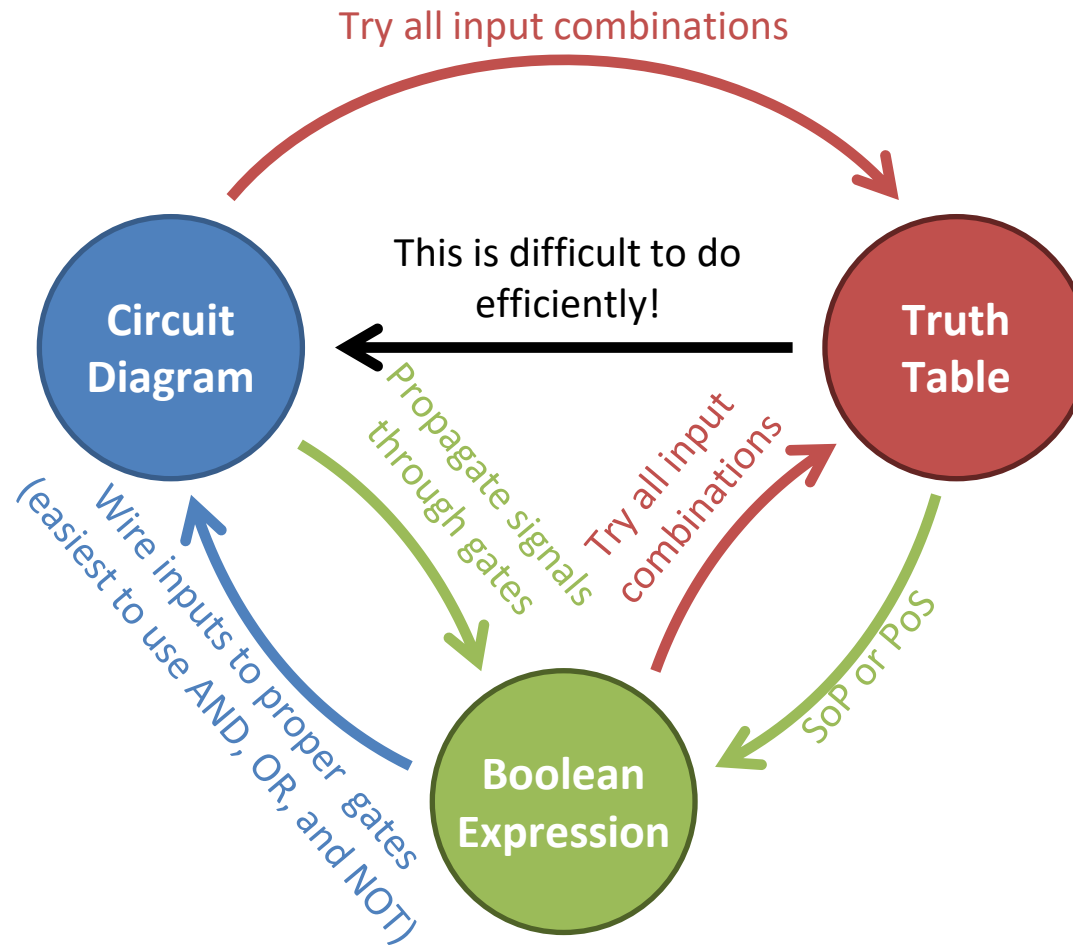
DB	PB	P	SBL
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0



■ $SBL = \overline{DBI} + (P \cdot \overline{PBI})$



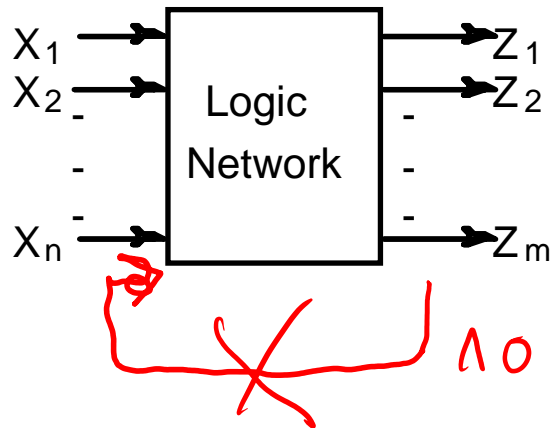
Our three equivalent forms



Combinational vs. Sequential Logic



Combinational Logic (CL)

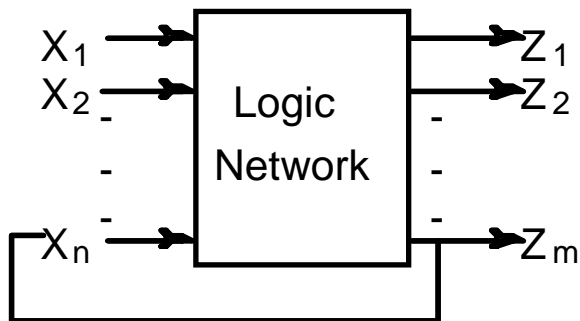


Network of logic gates without feedback.

Outputs are functions only of inputs.



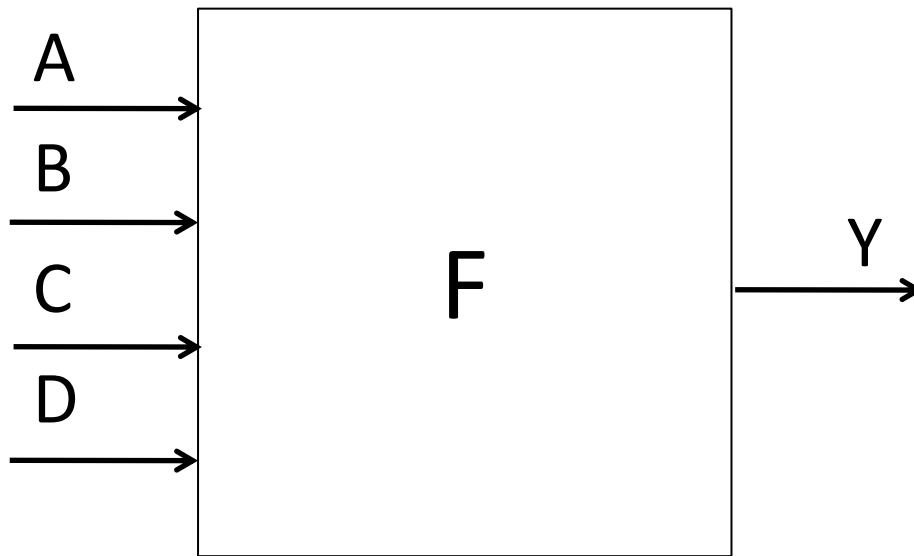
Sequential Logic (SL)



The presence of feedback introduces the notion of “state.”

Circuits that can “remember” or store information.

CL General Form



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)
1	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)

Q1: If N inputs, how many rows does a truth table have?

Q2: If N inputs, how many distinct functions F do we have?

2^{2^N}

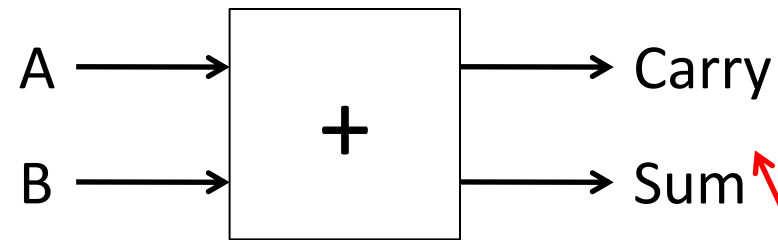
More Complicated Truth Tables

3-Input Majority

How many rows?

A	B	C	Out
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

1-bit Adder



2 separate functions

A	B	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$A \cdot B$ $A \oplus B$

More 311 review: Truth Table to Boolean Expression

- ❖ Terms of equation come from rows of table
 - For 1, write variable name
 - For 0, write complement of variable
- ❖ Sum of Products (SoP)
 - From CSE311, “DNF” (disjunctive normal form)
 - Take truth table rows that output 1:
 - AND the inputs together, OR the rows together
- ❖ Product of Sums (PoS)
 - From CSE311, “CNF” (conjunctive normal form)
 - Take truth table rows that output 0:
 - OR the complemented inputs together, AND the rows together

$$\text{SoP: } C = \bar{A}B + \bar{B}A$$

$$\text{PoS: } C = (A + B) \cdot (\bar{A} + \bar{B})$$

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

TODOs for youse

- ❖ Come to lab on Weds or Thurs to check out a kit (open times on website)
- ❖ **FILL OUT THE PRESURVEY!**
- ❖ Get started on Lab 1 🖱️ (released tonight)

