

CSE369: Intro to Digital Design

L1: Combinational Logic

Instructor: Naomi Alterman

Teaching Assistants:

Isabel Froelich
[Head TA]

Packard Stephenson

Mark Garcia

Ewan Kae

Violet Monserate

Avesta Rahimi

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!

Isabel



Packard



Mark



- Run your lab demos
- Teach section
- Mentor you through Ed and Office Hours
- An **invaluable** source of information and help

Ewan



Violet



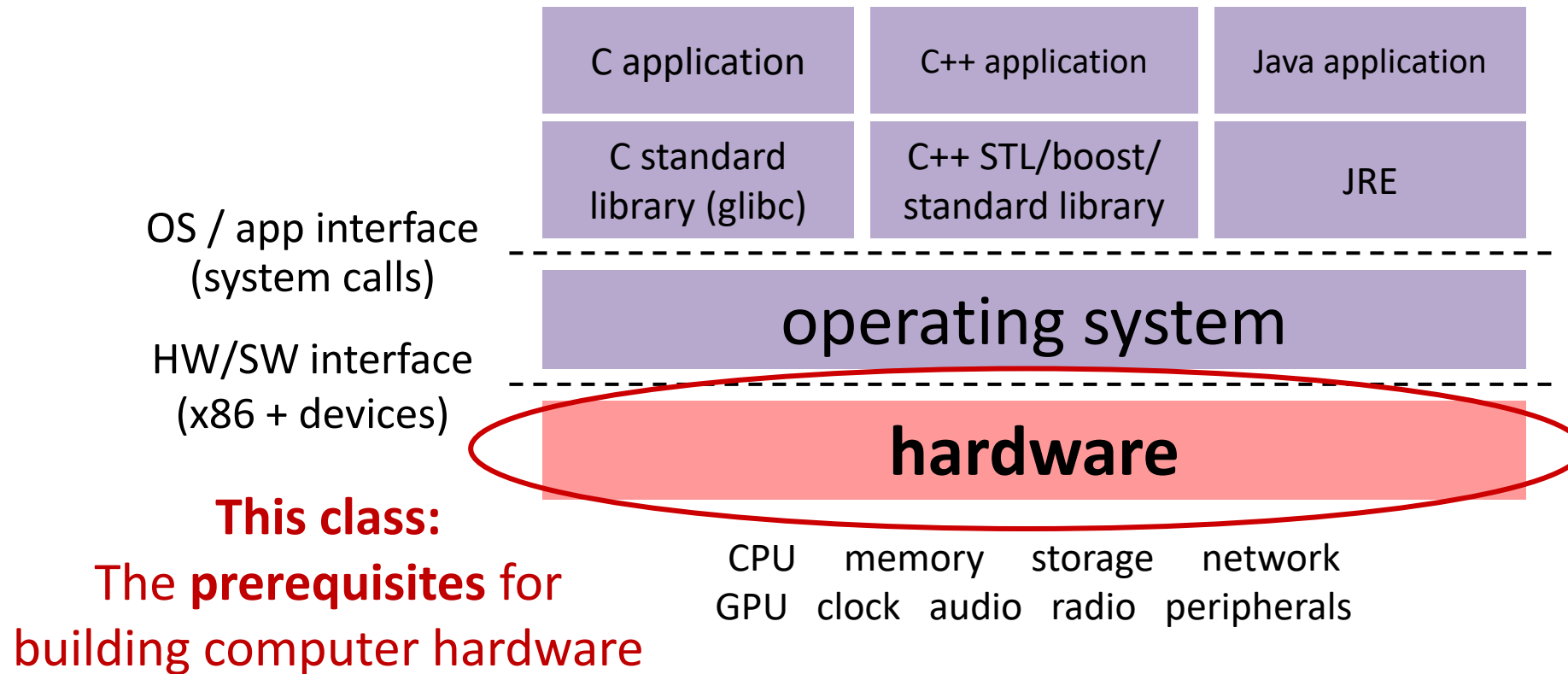
Avesta



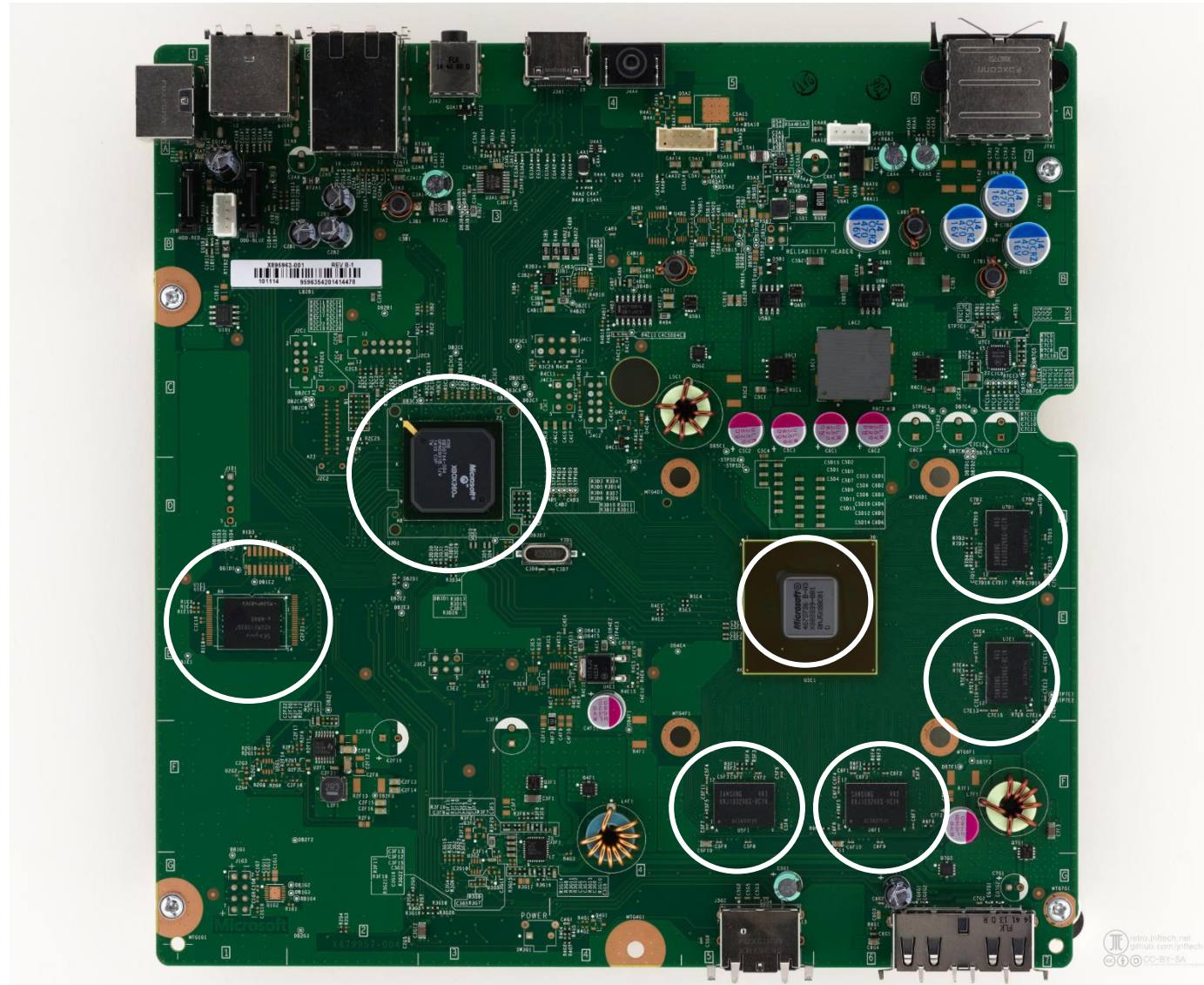
Lecture Outline

- ❖ **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?



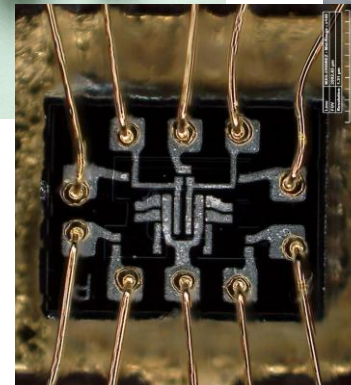
<https://tylerdmace.on.fleek.co/posts/redstone-computers/>

Course Motivation

❖ The real question:

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

**Electronic
Transistors!**



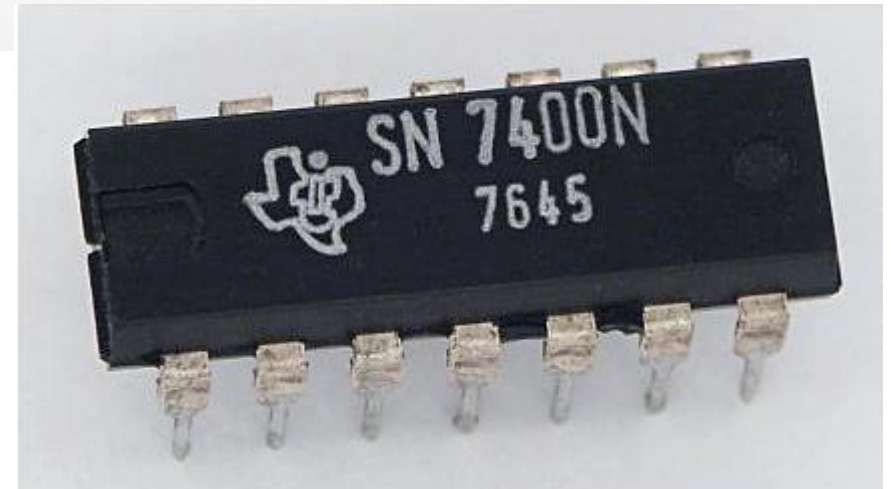
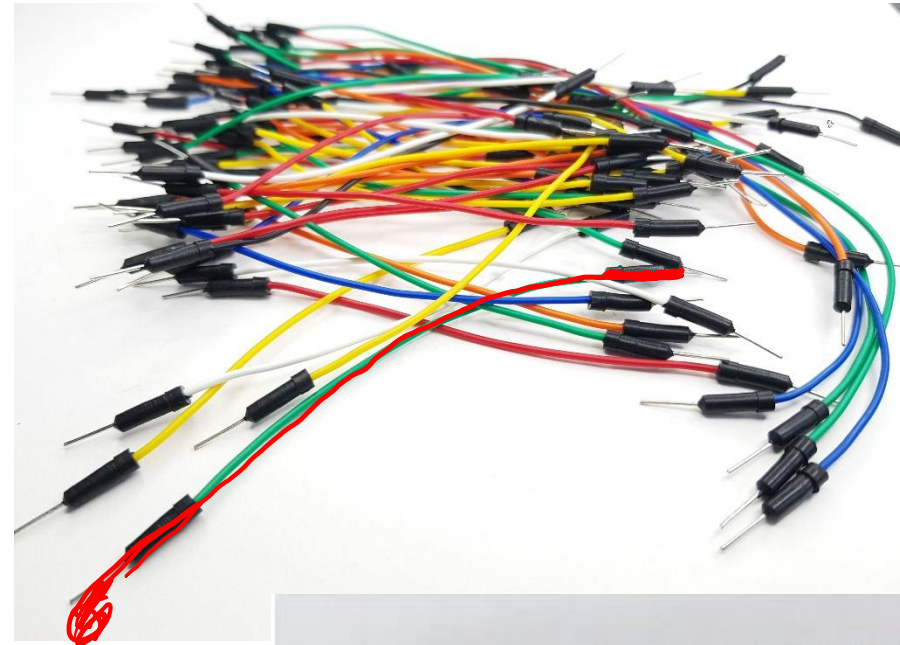
https://en.wikipedia.org/wiki/Vacuum_tube#/media/File:Triody_var.jpg

https://www.youtube.com/watch?v=F8gwexl1I_E

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

Digital Electronics

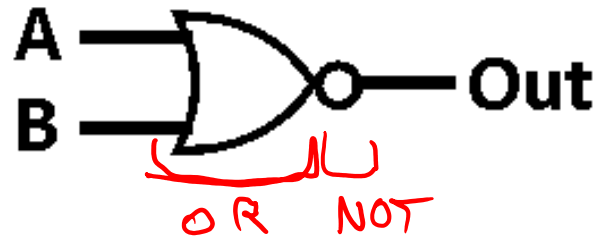
- ❖ Our data:
 - We'll use individual metal wires to move single bits around
 - When the **voltage** on a wire is low, that wire is transmitting a "0". When it's high, it's a "1".
- ❖ Our manipulations:
 - "Logic gates" built out of transistors
 - The output voltage is always some computation based on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



<https://diyaables.io/products/jumper-wires-male-to-male-round-head>
https://en.wikipedia.org/wiki/Logic_gate#/media/File:TexasInstruments_7400_chip,_view_and_element_placement.jpg

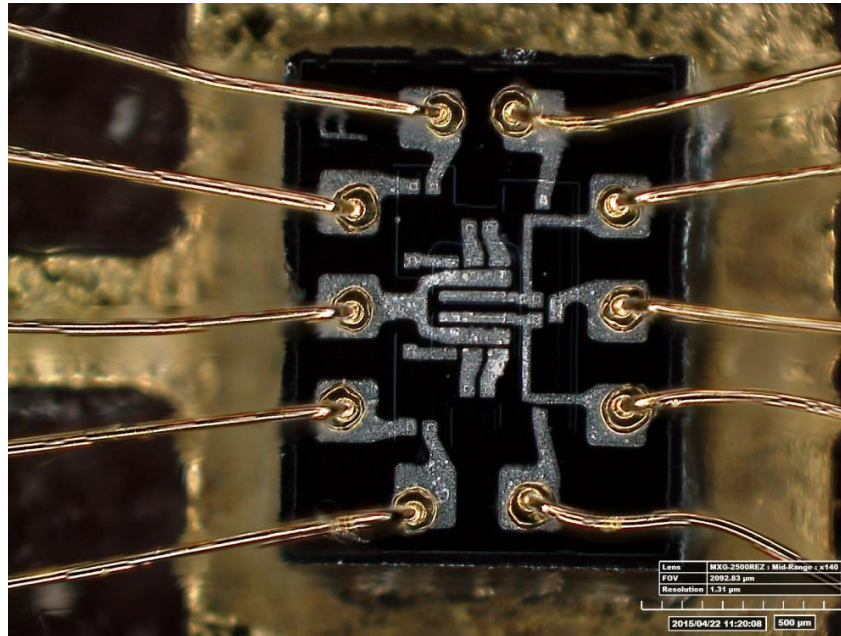
Logic gates

NOR



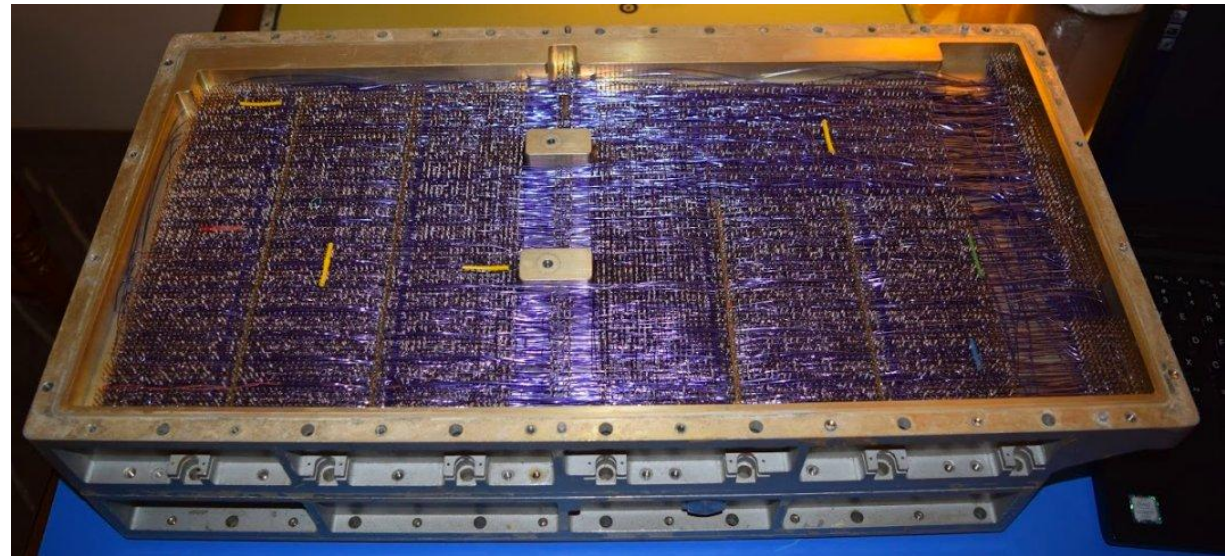
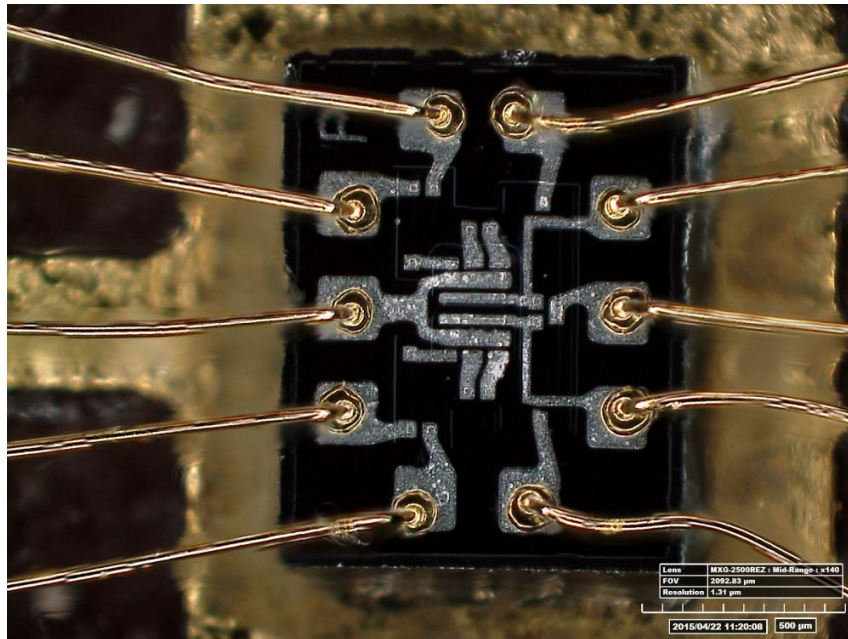
↓

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



By the way

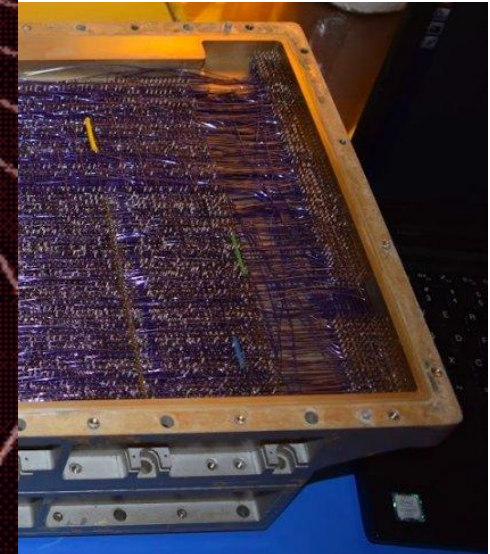
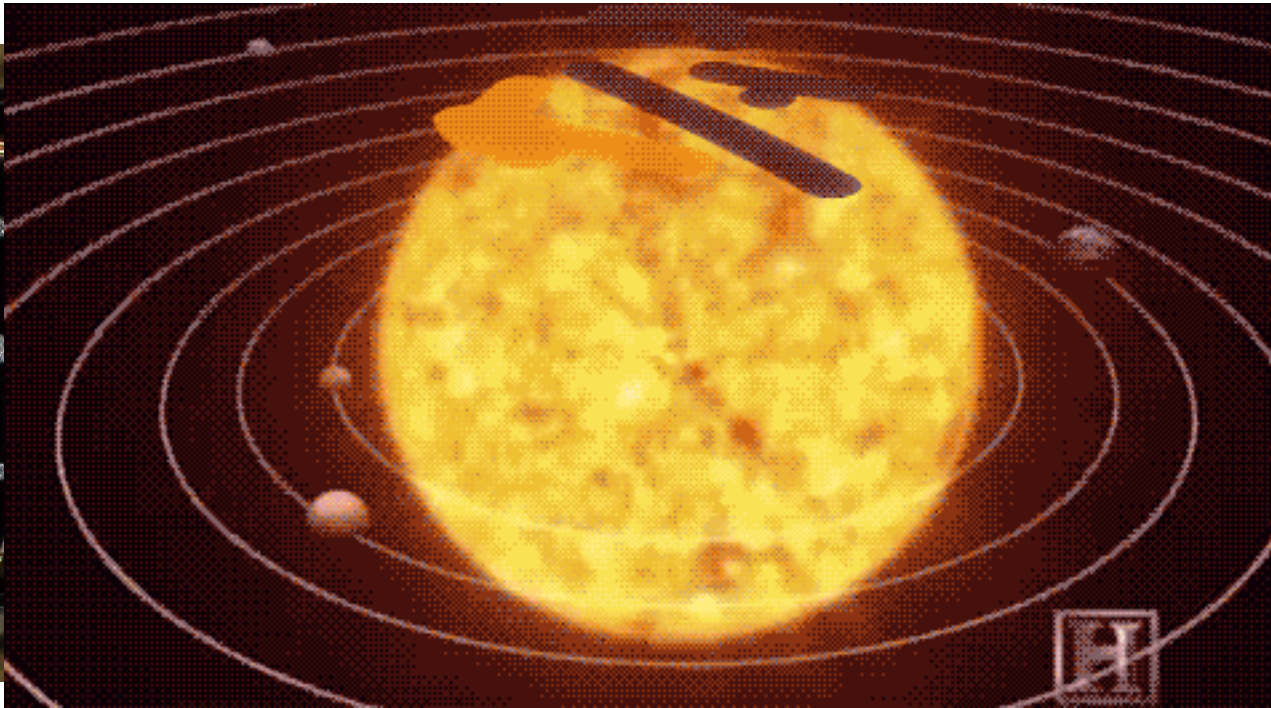
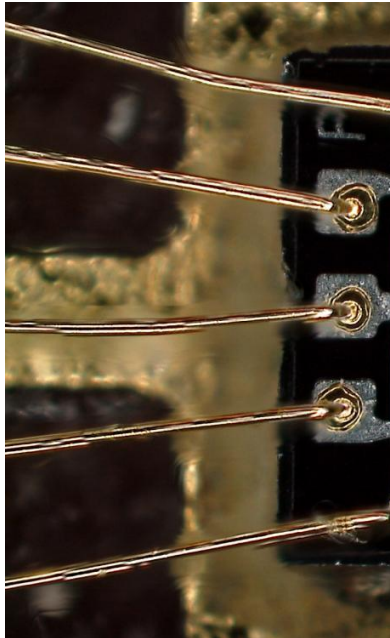
- **This NOR gate was literally the only kind of chip used in the Apollo Guidance Computer**
(there were just thousands upon thousands of them)



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

By the way

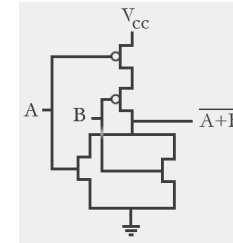
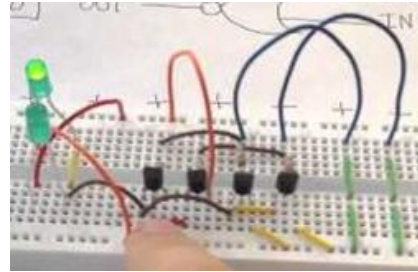
- This NOR gate was literally the only kind of chip used in the Apollo Guidance Computer



oto/2

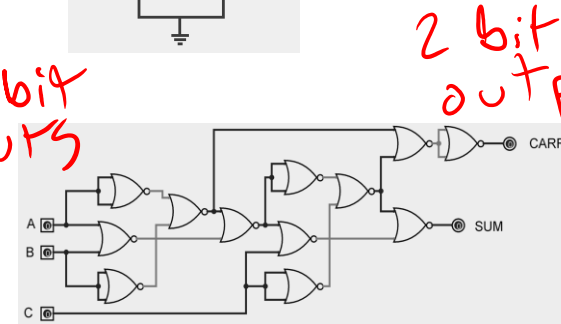
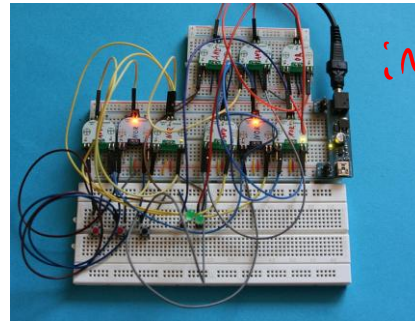
https://www.sccs.swarthmore.edu/users/06/adem/engin/e77vlsi/lab3/logic_nor.gi
<https://www.youtube.com/watch?v=TnltHE0Wp8Y>
<https://www.ahirlabs.com/2017/06/09/adder-and-subtractor/>
http://www.justgeek.de/wp-content/uploads/2014/07/IMG_0173.jpg
https://www.researchgate.net/figure/ALU-block-diagram_fig1_312203298
<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



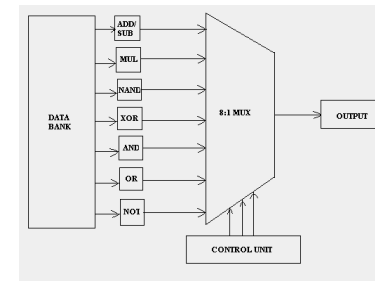
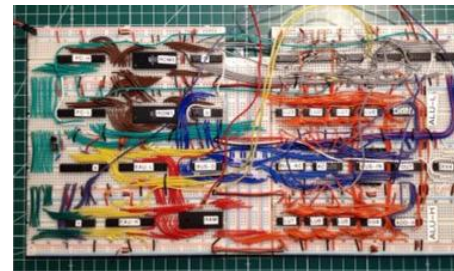
NOR gate

~ x60

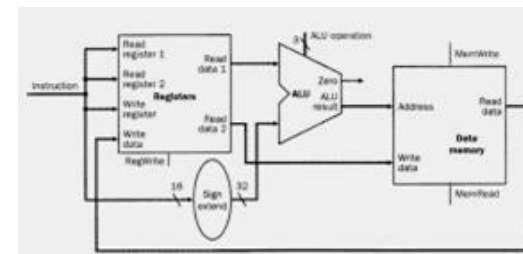
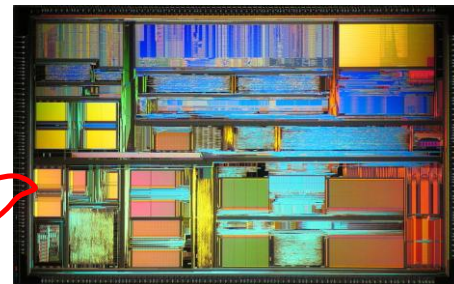


Full Adder

~ x1500



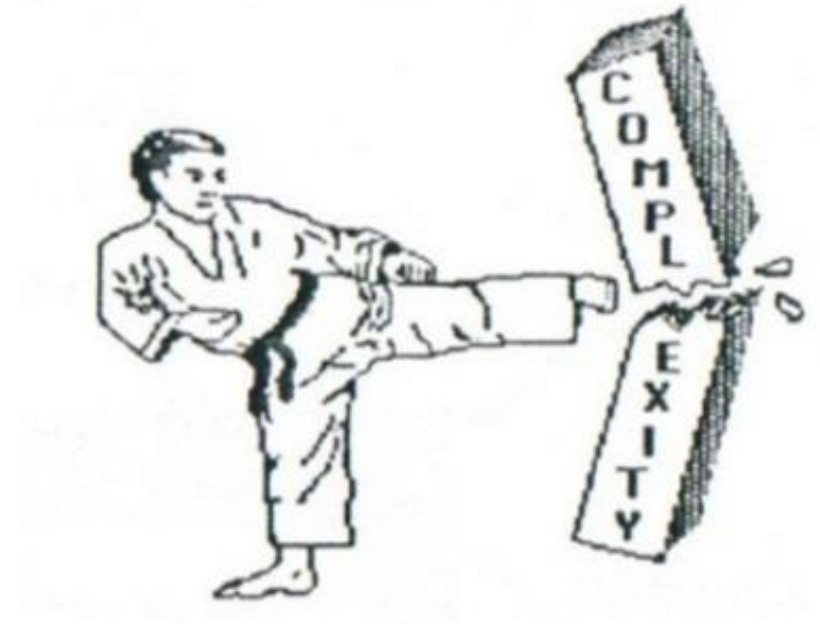
ALU



CPU

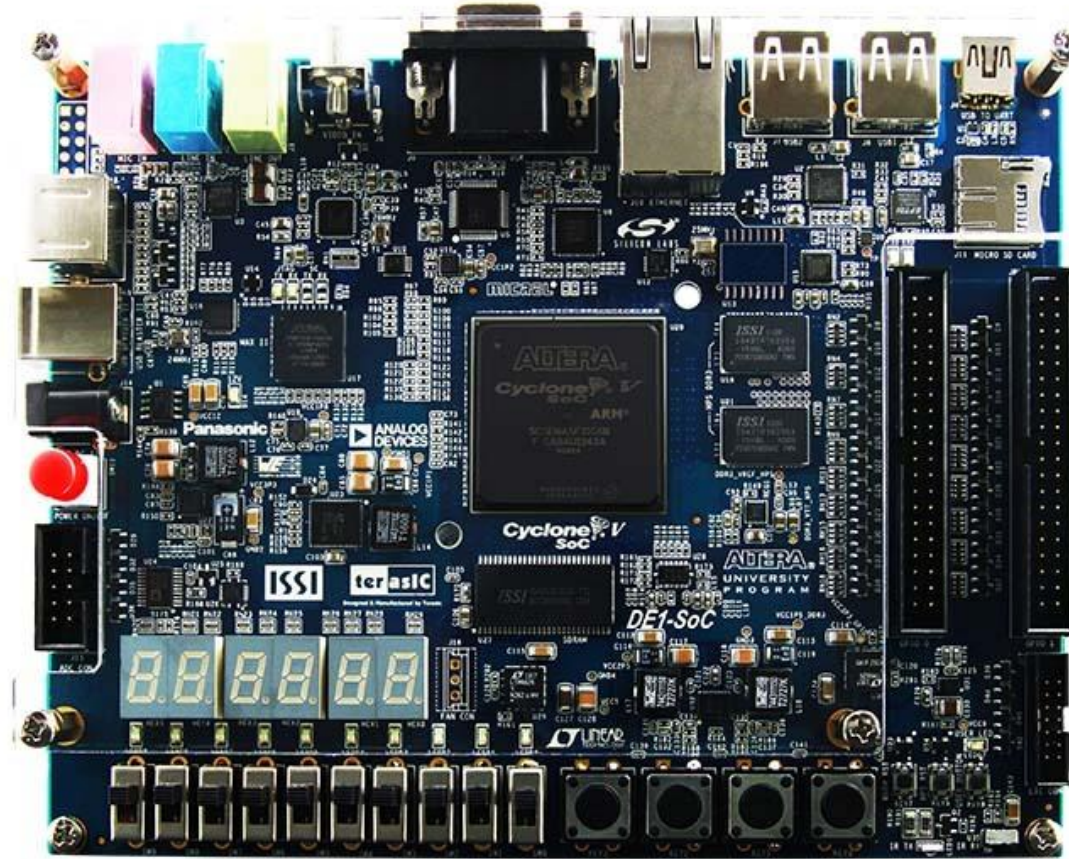
VLSI

- ❖ **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**)



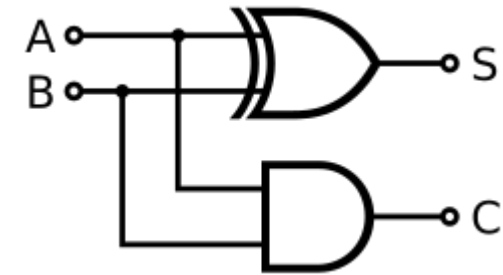
<http://hartenstein.de/KARL/KARL-folder.pdf>

FPGAs: our prototyping hardware

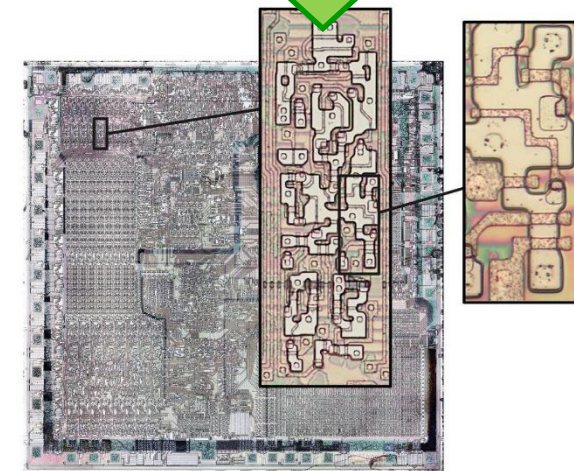


Digital design methodology

- ❖ 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
```



Verilog

- ❖ A hardware description language (**HDL**)
 - Define circuit schematics using text editors
 - Simulate behavior before (wasting time) implementing
 - Find bugs early
- ❖ ***Syntax*** is like C/C++/Java, but **meaning** is *very* different
 - Borrows heavily from early concurrency-focused languages like *Modula*
 - VHDL (the other major HDL) borrows from Ada (per...the DoD??)
- ❖ Modern version is **SystemVerilog**
 - Superset of previous; cleaner and more efficient

Hardware Ain't Software

- ❖ Although it looks like code:

```

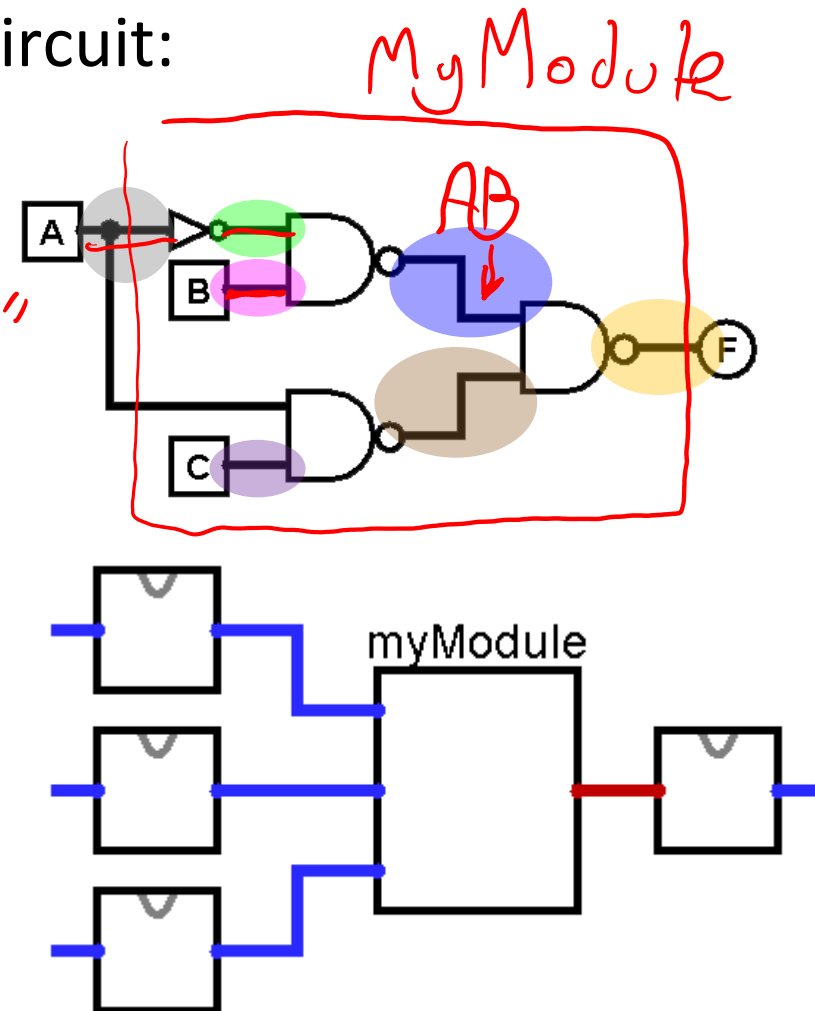
    module myModule (F, A, B, C);
    output logic F;
    input logic A, B, C;
    logic AN, AB, AC;

    nand gate1(AB, AN, B);
    nand gate2(AC, A, C);
    nand gate3(F, AB, AC);
    not not1(AN, A);
    endmodule
  
```

Handwritten annotations for the code:

- name* (under `myModule`)
- ports* (under `(F, A, B, C)`)
- direction* (under `output logic F;` and `input logic A, B, C;`)
- "instantiations"* (under the three `nand` and `not` lines)
- module type* (under `module`)
- module name* (under `myModule`)
- port connections* (under the arguments of the gate instantiations)

- ❖ It directly represents a physical circuit:



General digital logic learning progression

1. Learn the building blocks and theory for building machines that manipulate binary data *369, 371*

- Work at the “logic gate” level of abstraction

2. Learn how to use said building blocks to build a CPU that reads and executes machine code *469*

- Still at the “logic gate” level

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

- We start paying attention to electricity here

567, ee 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

Course Infrastructure

- ❖ ✨ 📌 Course website: <https://cs.uw.edu/369/> 📌 ✨
 - Schedule, lecture slides, lab specs, software tutorials, links to everything else
- ❖ 🗣️ [Ed forum](#) for **all** communications
 - Announcements, lecture discussion, technical content Q&A
 - Make private threads to communicate with Naomi and TAs
- ❖ ☑️ [Gradescope](#) for work submission and grades
- ❖ 📹 [Canvas](#) for lecture recordings
- ❖ 👨‍🔬 CSE003 (room in Allen basement) for **lab demos** and **office hours**

Course Components and 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 November 3rd
- Quiz 2 (10%): 30 min in class on November 24th
- Quiz 3 (14%): 60 min in class on ~~November 8th~~ December
- You must take all three quizzes to pass the class

❖ Weekly optional practice problems (0%)

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

Section

- ❖ Discussion section on **Fridays 3:30-4:20p** in **ECE 045**
- ❖ Run by your TAs
- ❖ Highlights:
 - Material that shows up on quizzes
 - Tips and tricks to keep your labs from becoming a living nightmare
- ❖ ***Highly*** recommended if you can make it.
 - These sections were originally added 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, LED matrix
- ❖ Class software:
 - On Windows machines in CSE 003
 - On personal Windows machines: [guide here](#)
 - On attu (mac users): [guide here](#)



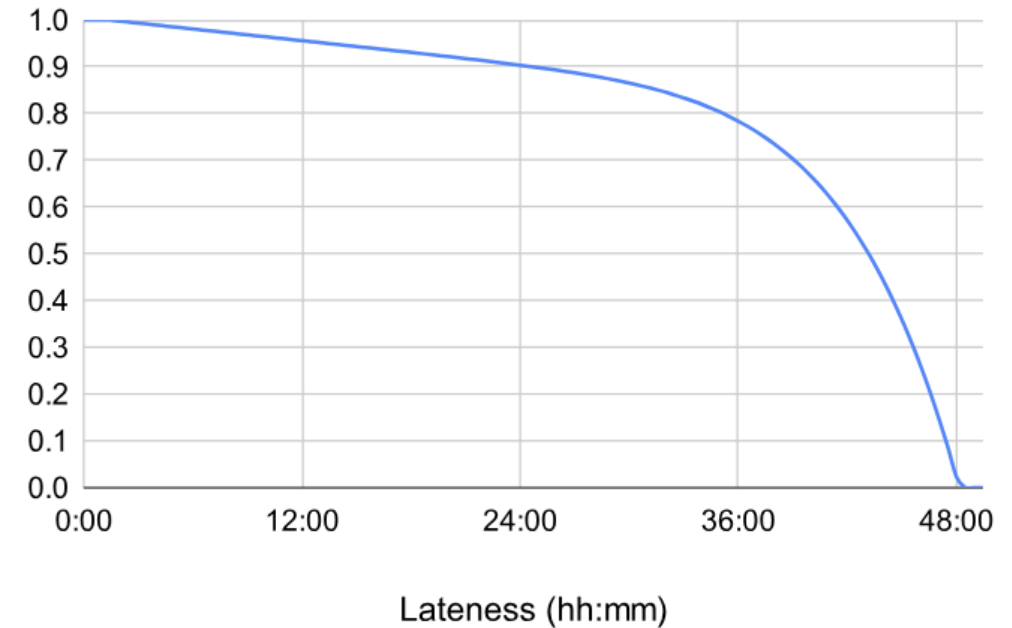
Lab Submissions and Demos

- ❖ 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

Class AI Policy



“AI as a Reference Tool”

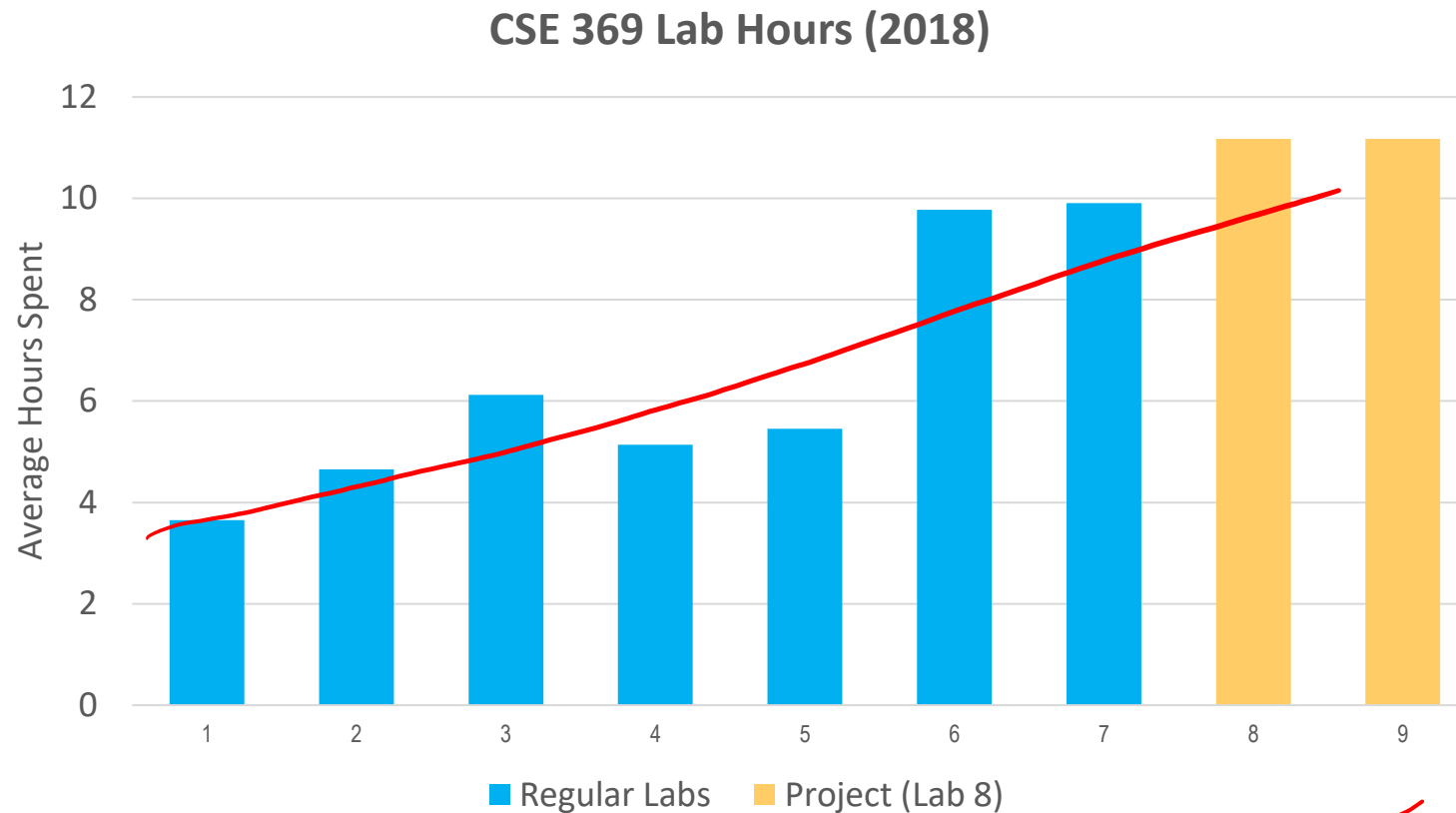
You may use AI for general course questions and ungraded work, but not for graded assignments, quizzes, exams, or any deliverable that counts toward your grade.

- ❖ Consider first whether you have the time to ask your TAs, the professor, or peers for help and then use AI for subsequent clarification.
- ❖ Use AI as a tutor, not a substitute for your own work.
- ❖ Ask for explanations, hints, or examples rather than full solutions.
- ❖ Verify AI output against course materials and official documentation.
- ❖ If AI starts doing the work for you, stop and switch to a more limited prompt.

TL;DR – Same as collaboration policy, but with AI tools instead of friends

Course Workload

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



Lecture Outline

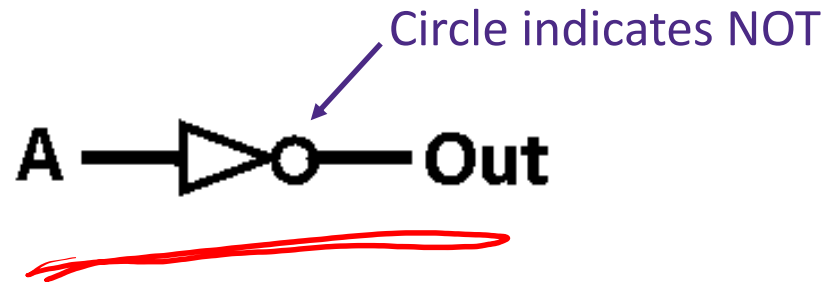
- ❖ What's this class all about?
- ❖ Course Logistics
- ❖ **Combinational Logic Review**

311 Reminder: Logic Gates (1/2)

❖ Special names and symbols:

$\neg$!

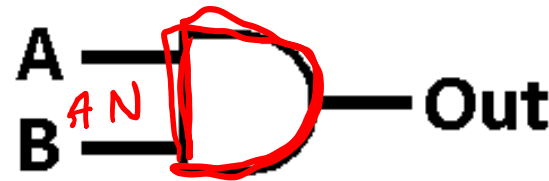
NOT



A	Out
0	1
1	0

$\&$ *

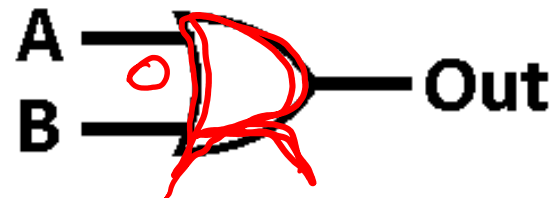
AND



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

$+$ $\vee$

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



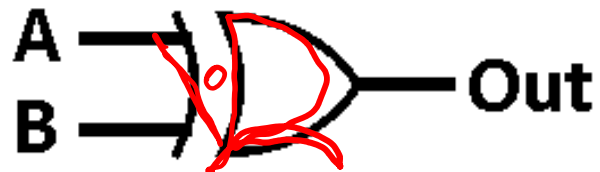
A	B	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR



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

XOR



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

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



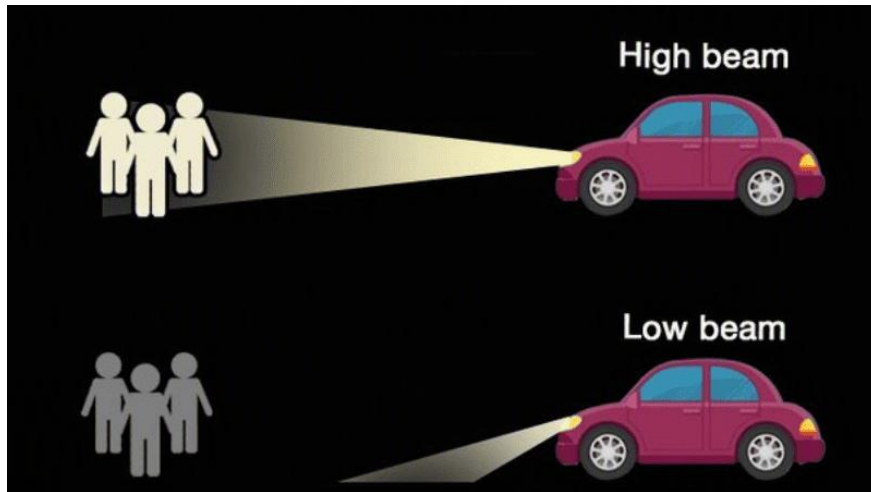
	DDo	PDo	DA
(0)	0	0	0
(1)	0	1	1
(2)	1	0	1
(3)	1	1	1



$$DA = DDO + PDo$$

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

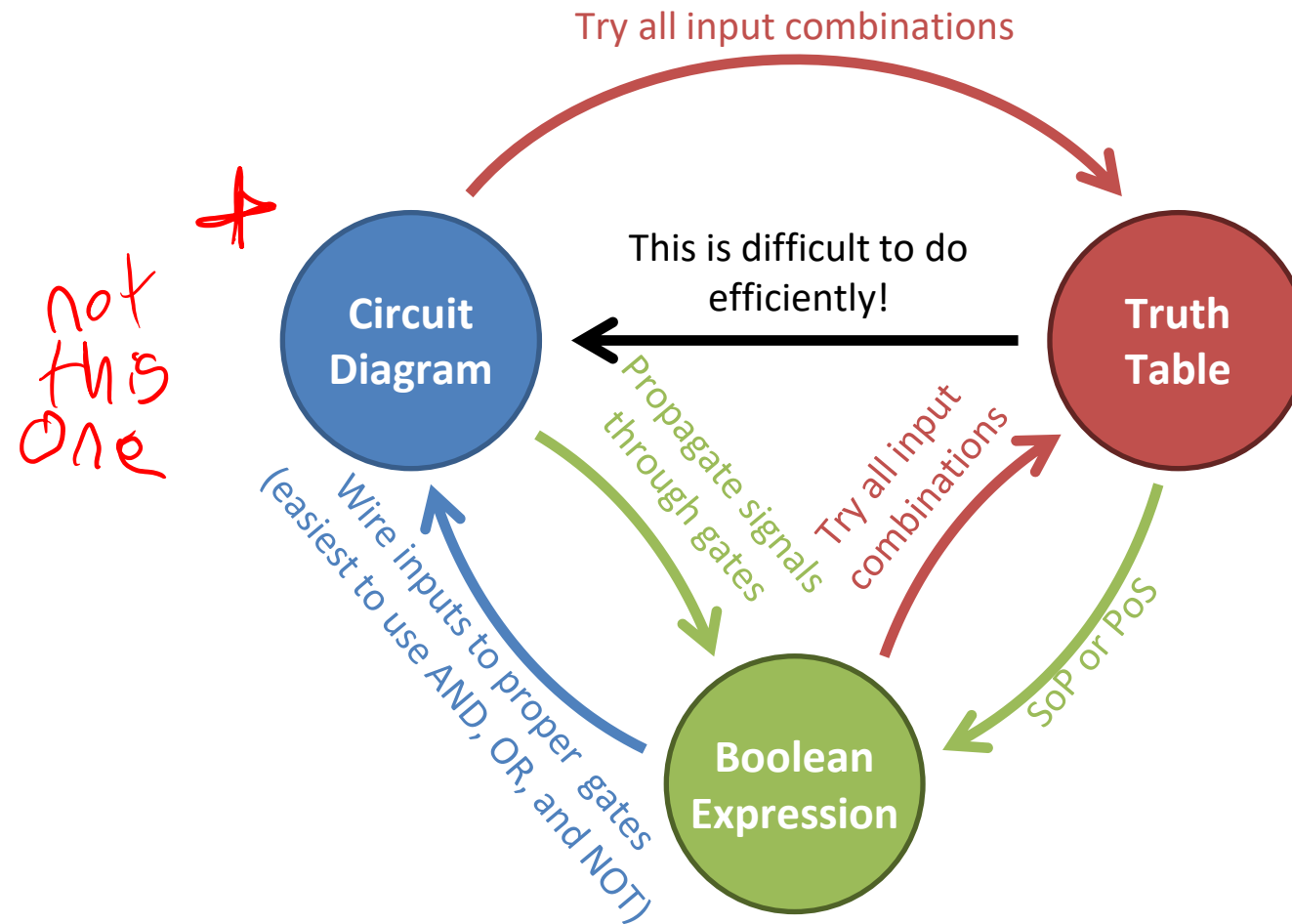
- ❖ Output: **S**eat**B**elt**L**ight
- ❖ Inputs: **D**riverBeltIn, **P**assengerBeltIn, **P**assenger



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

D	PB	P	SBL
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Our three equivalent forms



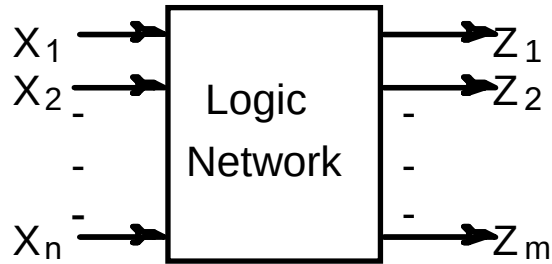
Car Electronics (in Verilog)

```
module DashboardController (  
    input logic DriverDoorOpen, input logic PassengerDoorOpen,  
    output logic DoorAjar,  
    input logic LightsOn, input logic HighBeamOn,  
    output logic HighBeamIndicator,  
    input logic DriverBeltIn, input logic PassengerBeltIn, input logic Passenger,  
    output logic SeatBeltLight,  
);  
  
    OR door_gate(DoorAjar, DriverDoorOpen, PassengerDoorOpen);  
  
    assign HighBeamIndicator = LightsOn & HighBeamOn;  
  
    logic PassengerWithoutBelt;  
    AND passenger_gate(PassengerWithoutBelt, Passenger, ~PassengerBeltIn);  
    assign SeatBeltLight = ~DriverBeltIn | PassengerWithoutBelt;  
endmodule
```

equivalent
coding styles

Combinational vs. Sequential Logic

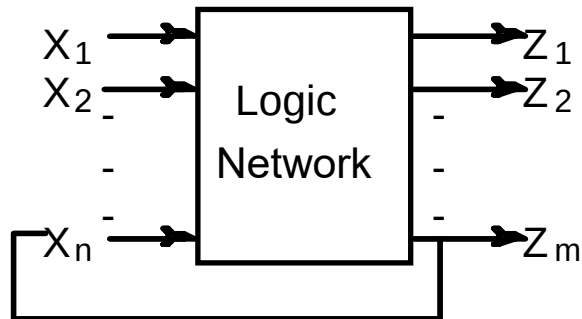
❖ Combinational Logic (CL)



Network of logic gates without feedback.

Outputs are functions only of inputs.

❖ Sequential Logic (SL)



← week 5

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

Circuits that can “remember” or store information.

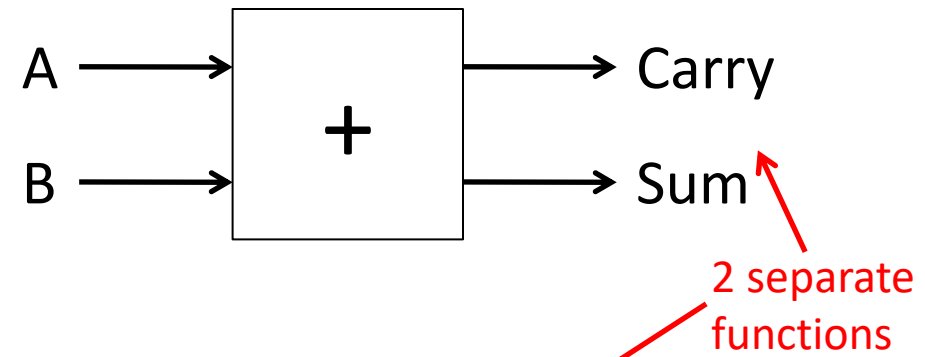
More Complicated Truth Tables

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



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

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

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)

