Intro to Digital Design
L1: Combinational Logic

Instructor: Naomi Alterman

Teaching Assistants:

Derek de Leuw
Isabel Froelich
Kevin Hernandez
Sathvik Kanuri
Aadithya Manoj

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

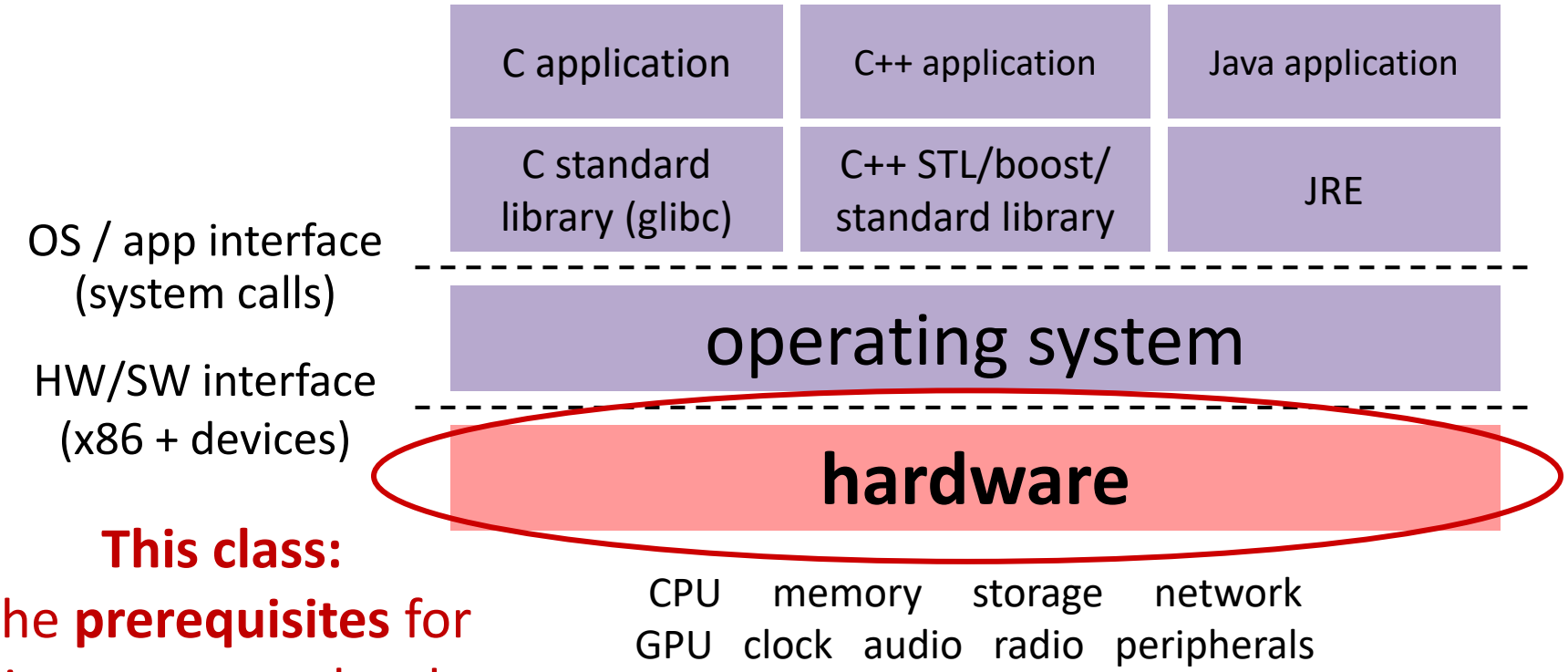
Kevin

Sathvik

Aadithya

  - 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!

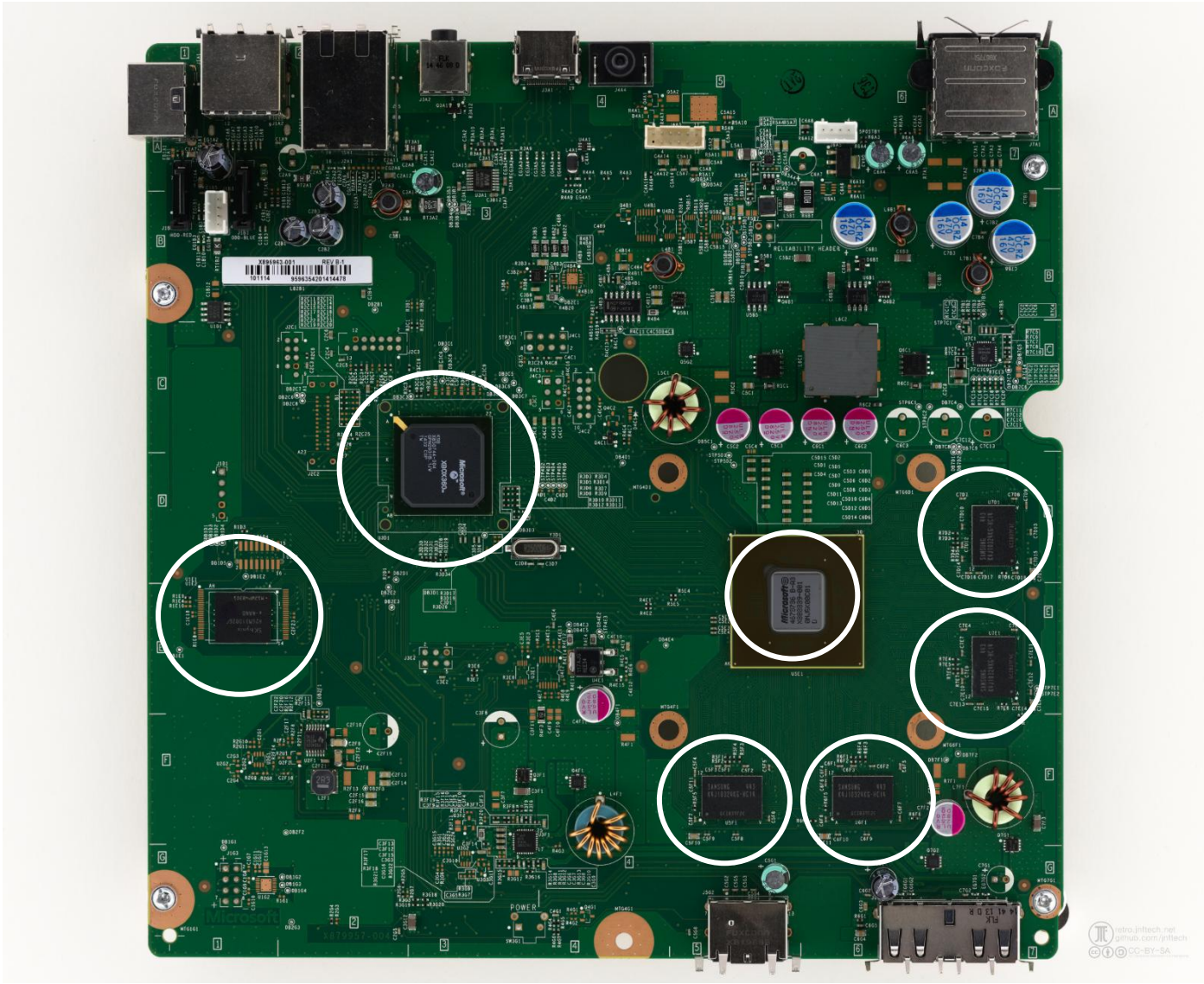
Course Motivation



This class:

The **prerequisites** for building computer hardware

Digital Electronics



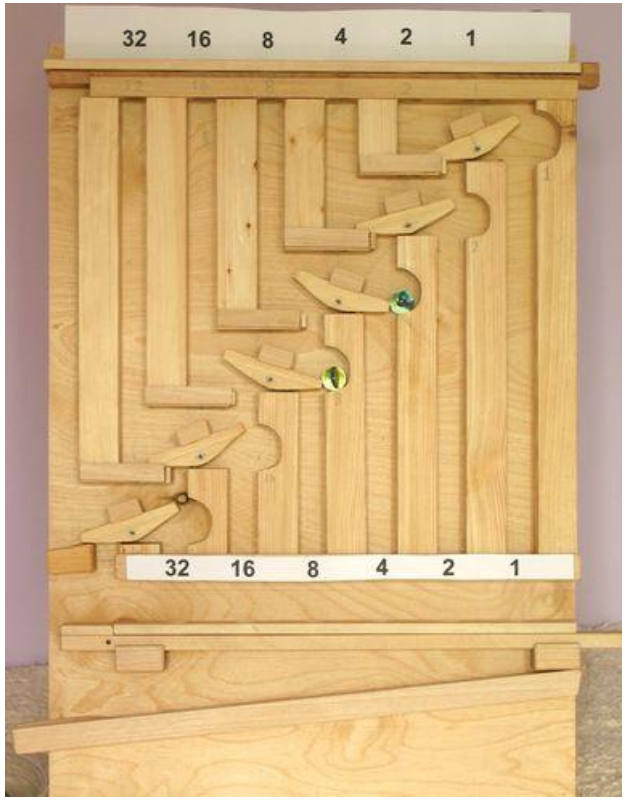
Course Motivation

- 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? 🤔 🤔 🤔

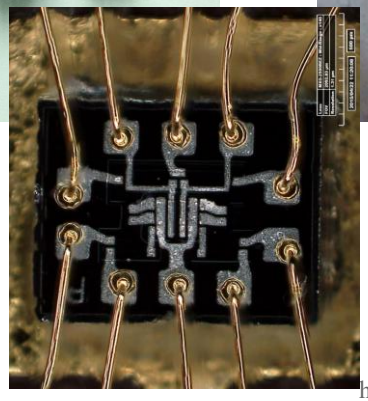
Minecraft?

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

Course Motivation

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

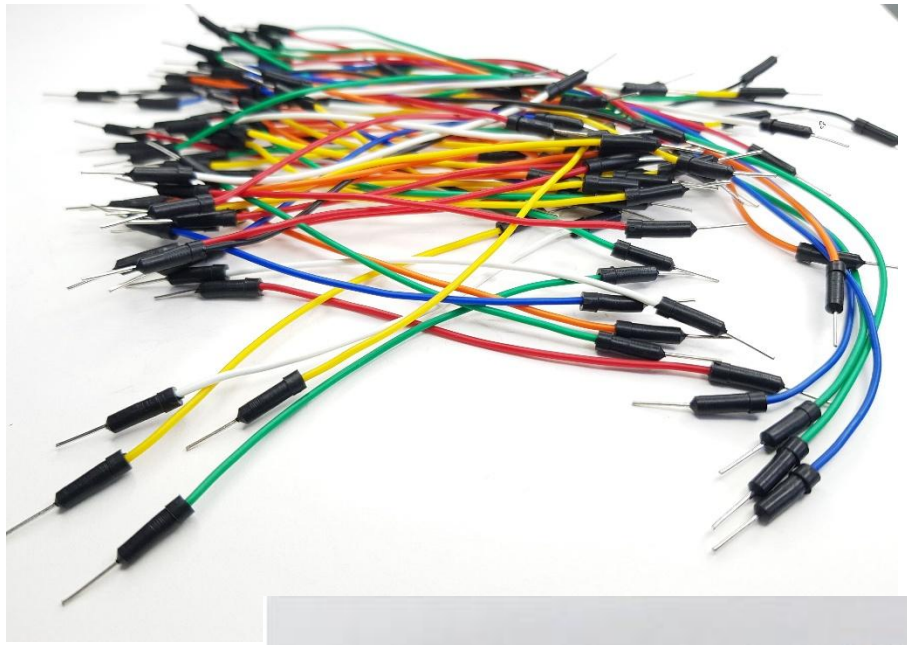
Electronic Transistors!

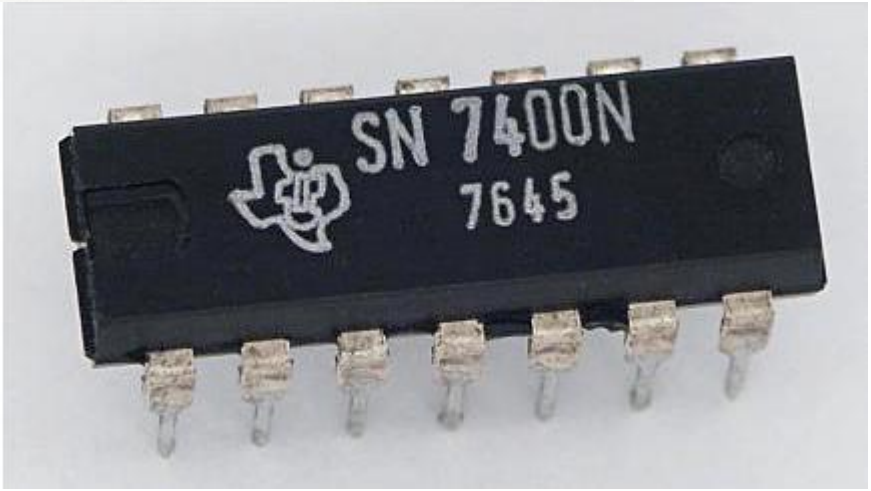


https://en.wikipedia.org/wiki/Vacuum_tube#/media/File:Triody_var.jpg
https://www.youtube.com/watch?v=F8gwexl1I_E
https://www.righto.com/2019/09/a-computer-built-from-nor-gates-inside.html

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.**



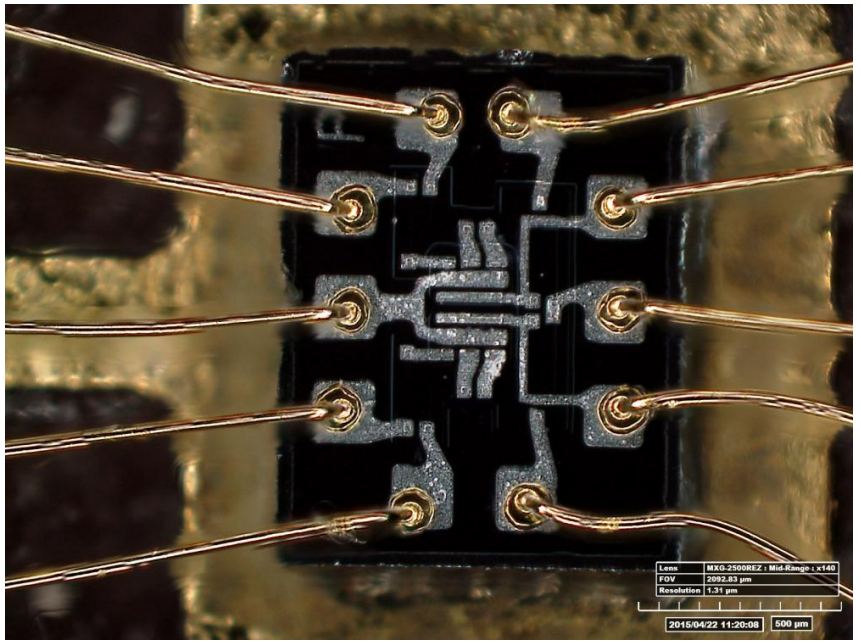


https://diyables.io/products/jumper-wires-male-to-male-round-head
https://en.wikipedia.org/wiki/Logic_gate#/media/File:TexasInstruments_7400_chip,_view_and_element_placement.jpg

Logic gates

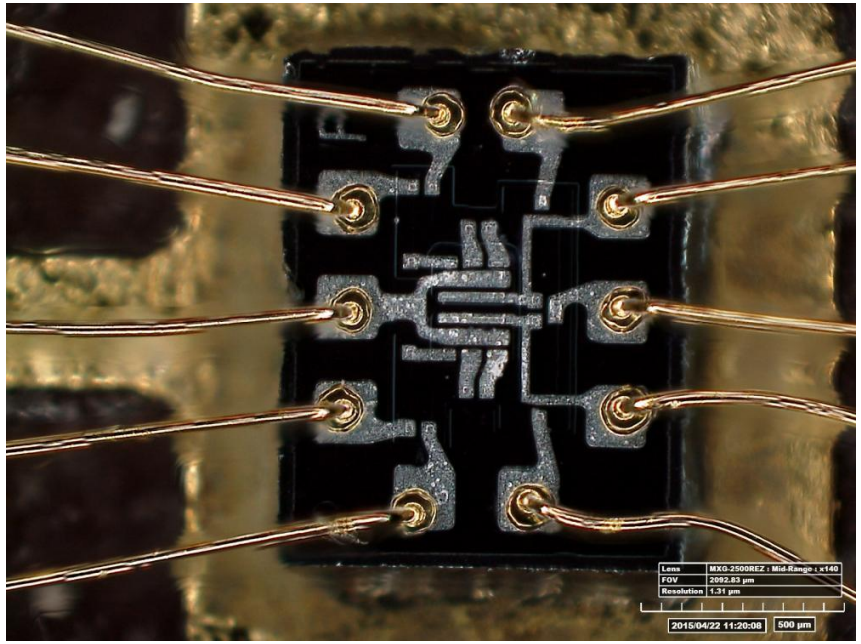
NOR

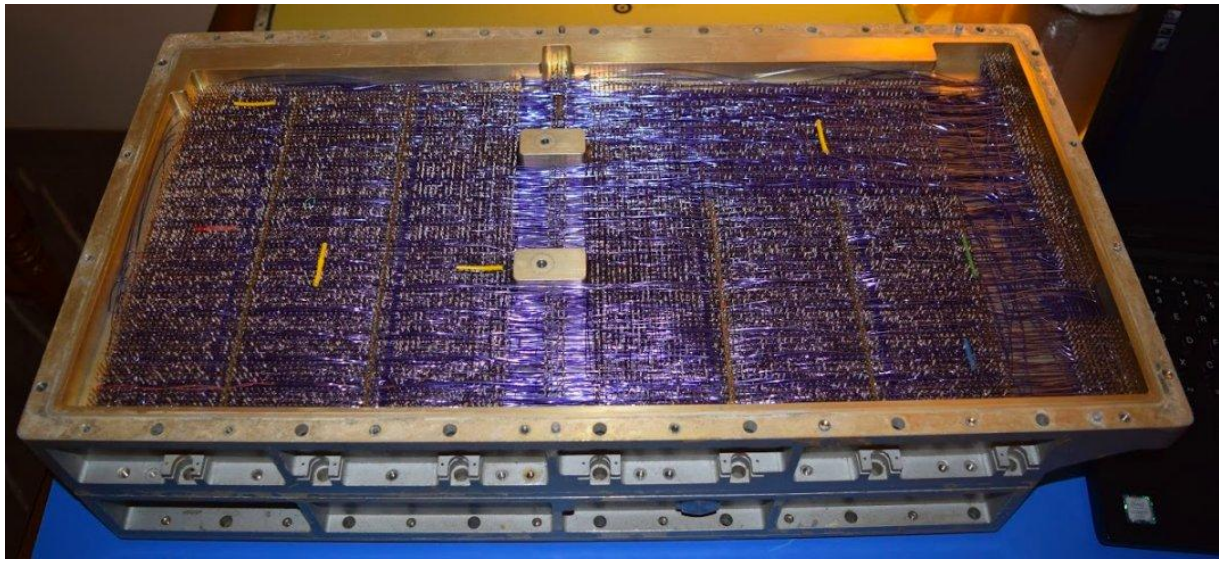
A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0



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)



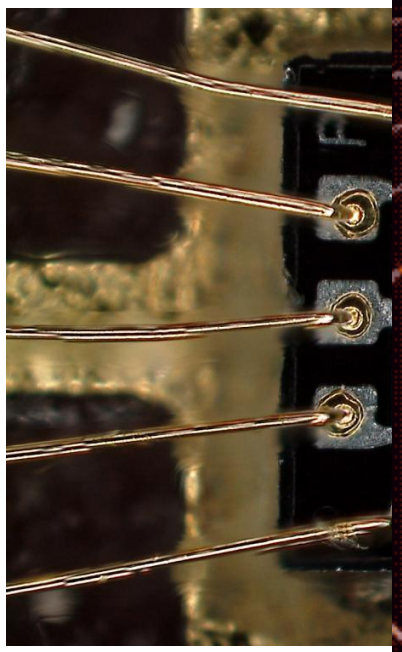


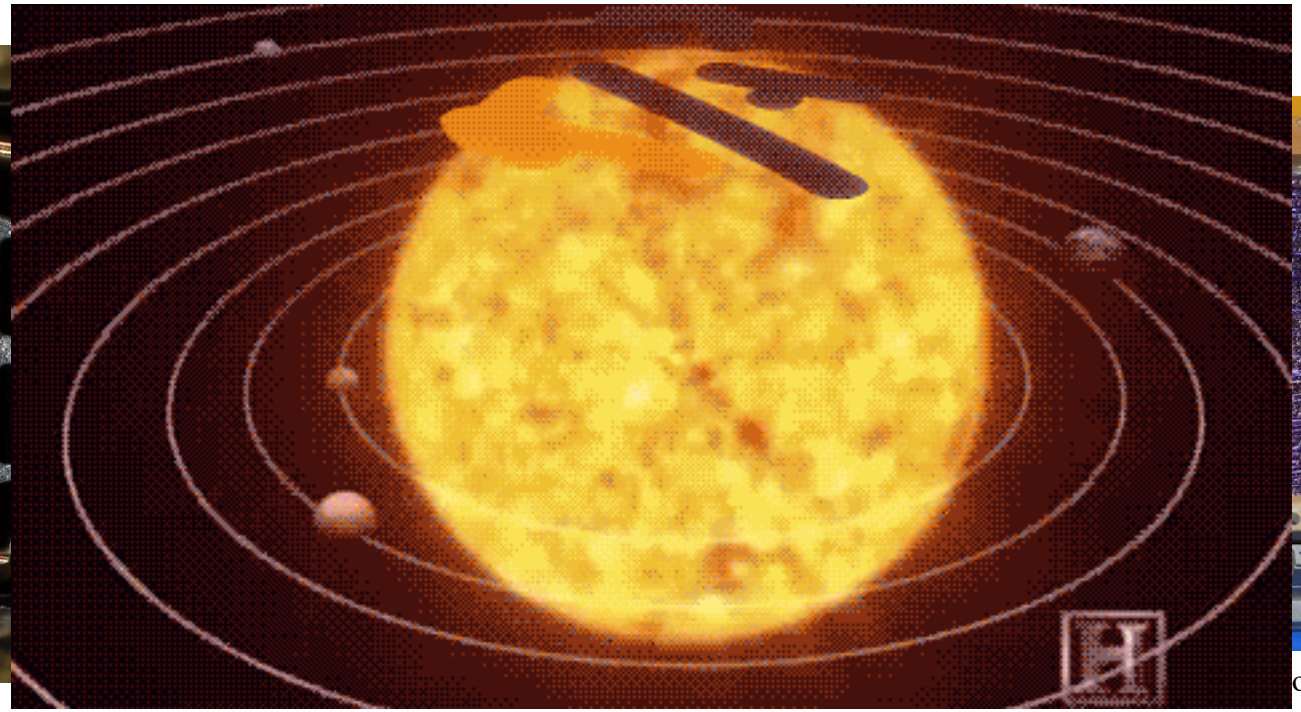
https://x.com/kenshirriff/status/1237818694379556864/photo/2

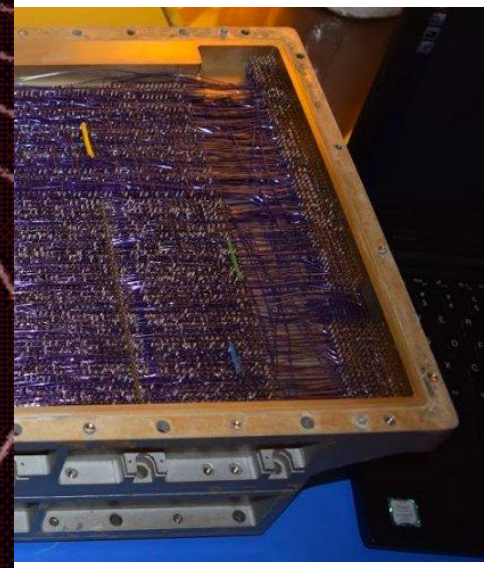
https://airandspace.si.edu/file/9687

By the way

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





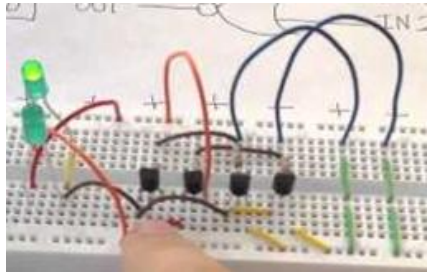


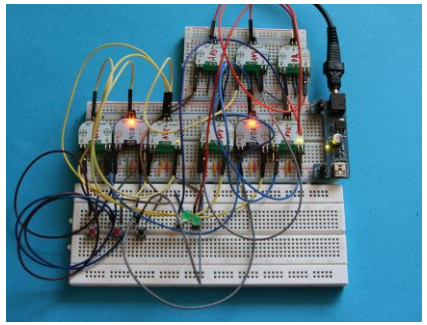
oto/2

https://airandspace.si.edu/file/9687

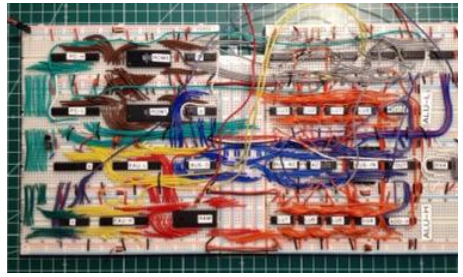
https://www.sccs.swarthmore.edu/users/06/adem/engin/e77vlsi/lab3/logic_nor.gi
https://www.youtube.com/watch?v=TnltHE0Wp8Y
https://www.ahirlabs.com/2017/06/09/adder-and-subtractor/
http://www.justgeek.de/wp-content/uploads/2014/07/IMG_0173.jpg
https://www.researchgate.net/figure/ALU-block-diagram_fig1_312203298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

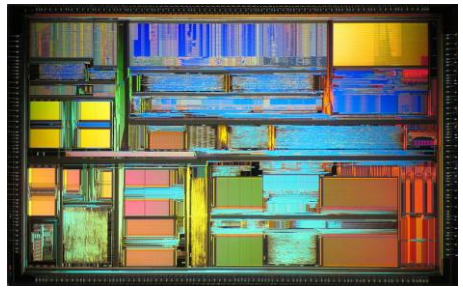


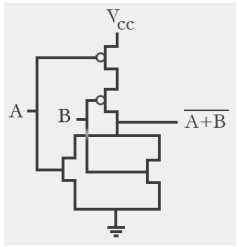


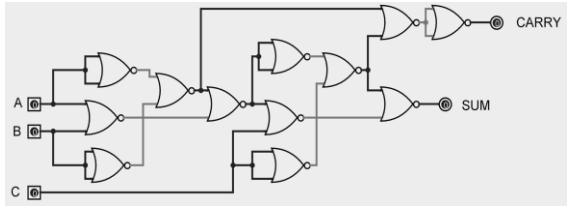
~ x60

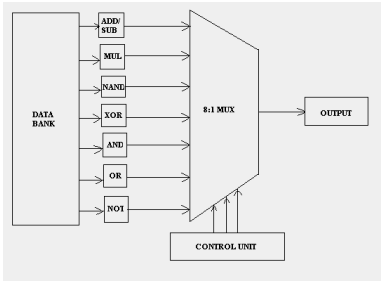


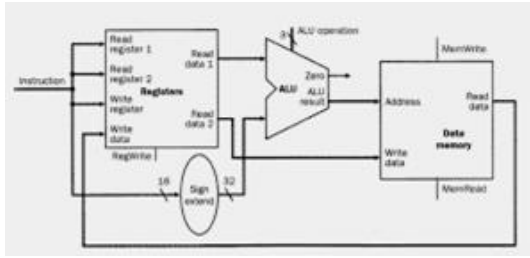
~ x1500











NOR gate

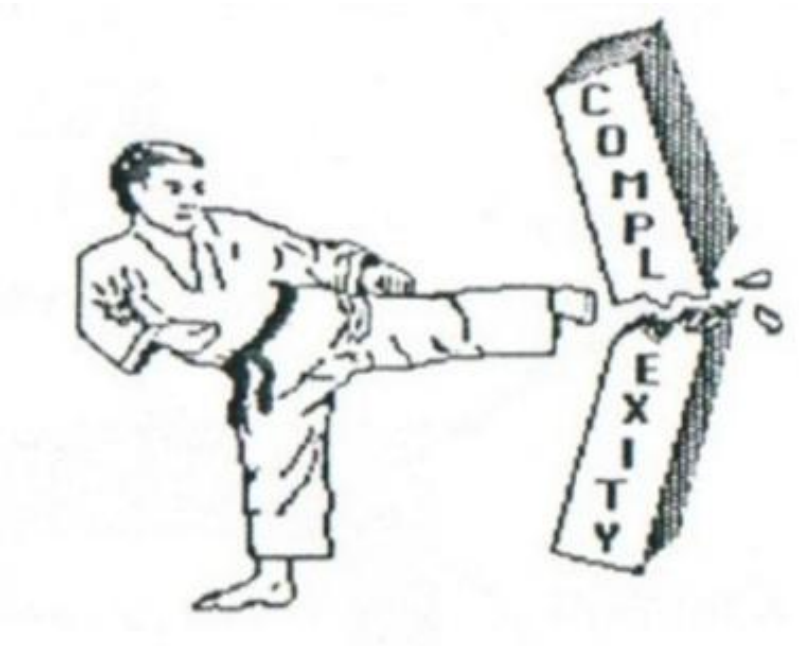
Full Adder

ALU

CPU

VLSI

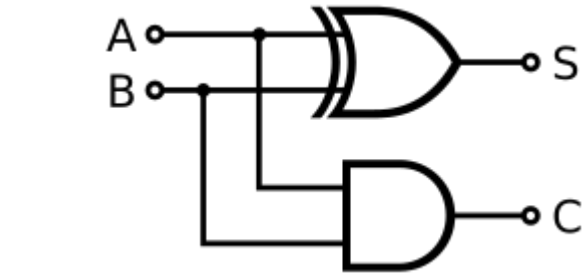
- **Very Large Scale Integration** is the term used for the modern process of making computer chips (since the mid 80s)
- **The idea:** as our machines get more complex, we need to find **methodological and technological solutions** to mitigate that complexity
- This means in this class we're **not**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



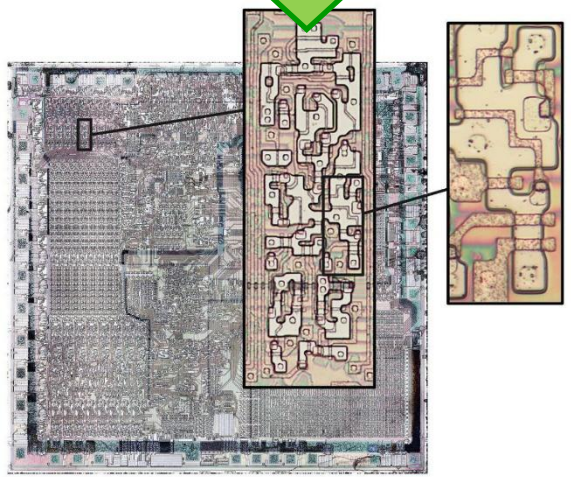
http://hartenstein.de/KARL/KARL-folder.pdf

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 CAD 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)**

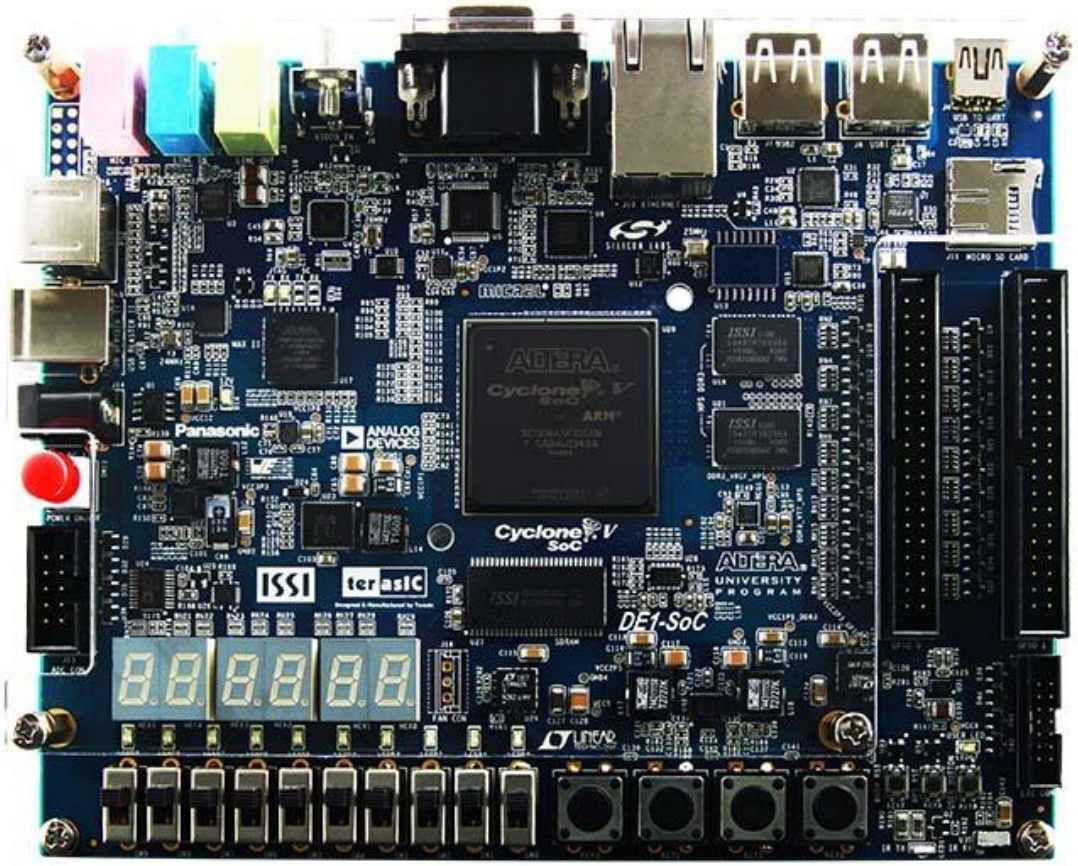



```
module half_adder (input logic a,
input logic b, output logic carry,
output logic sum)
   assign carry = a & b;
   assign sum = a ^ b;
endmodule
```



https://www.righto.com/2020/08/reverse-engineering-adder-inside-intel.html

FPGAs: our prototyping hardware



Digital Design: What's It All About?

- Come up with an implementation using a set of primitives given a functional description and 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

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

Can do this in Minecraft 🤘

Cannot do this in Minecraft 🥺

Miso Moment

Lecture Outline

- **Course Logistics**
- Combinational Logic Review
- Combinational Logic in the Lab

[🔖 Bookmarks 🔖]

- Website: https://cs.uw.edu/369/
  - Schedule (lecture slides, lab specs), weekly calendar, other useful documents
- Ed Discussion: https://edstem.org/us/courses/90073/
  - Announcements, questions and answers – staff will monitor and contribute
- Gradescope: https://www.gradescope.com/courses/1201886/
  - Lab submissions, Quiz grades, regrade requests
- Canvas: https://canvas.uw.edu/courses/1860370
  - Grade book, lecture recordings

Grading

- Labs (66%)
  - 6 regular labs – 1 week each
    - Labs 3-4: 60 points each, Labs 1&2, 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 February 4th
  - Quiz 2 (10%): 30 min in class on February 25th
  - Quiz 3 (14%): 60 min in class on March 11th
- This class uses a straight scale (> 95% → 4.0)
  - Extra credit points count the same as regular points

Getting Help

- Naomi's OH: Wednesdays 2:30-3:30 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 😤 👿 !")
  - Starting in week 2
- 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."

Section

- Optional discussion section on **Fridays 1:30-2:20** in **CSE 203**
- Run by TAs
- Focused on connecting the theory covered in lecture to the concrete work you in lab
- *Highly* recommended if you can make it.
  - This was a recently added component to the course to fill learning gaps often reported by students

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

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.

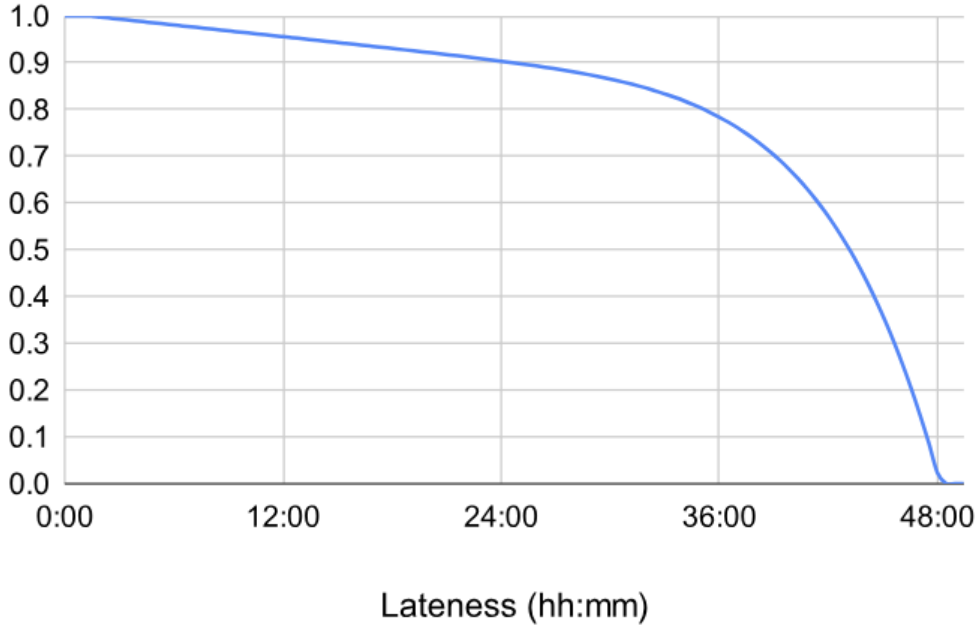
Lab Submissions and Demos

- Reports and code are due on **Wednesdays before 2:30 pm regardless of your demo time**
  - Submit 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
- No penalties on lab demos, but must be done by EOD Friday

Collaboration Policy

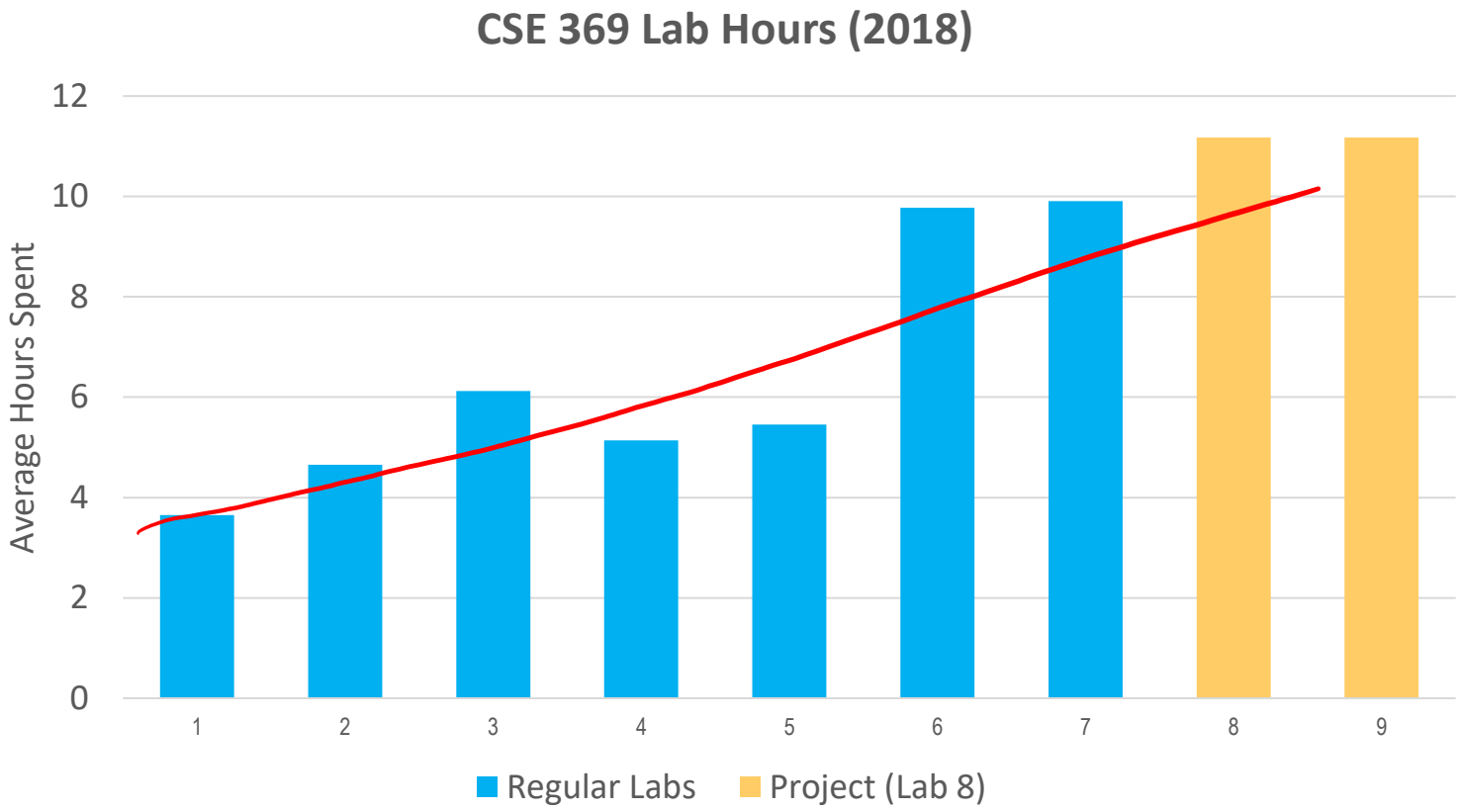
- Labs and project are to be completed *individually*
  - Goal is to give every student the hands-on experience
  - Violation of these rules is grounds for failing the class
- **OK:**
  - Discussing lectures and/or readings, studying together
  - *High-level* discussion of general approaches
  - Help 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:

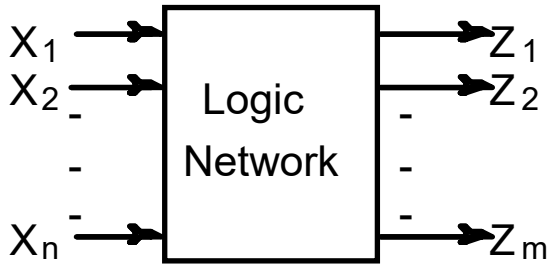


Lecture Outline

- Course Logistics
- **Combinational Logic Review**
- Combinational Logic in the Lab

Combinational vs. Sequential Logic

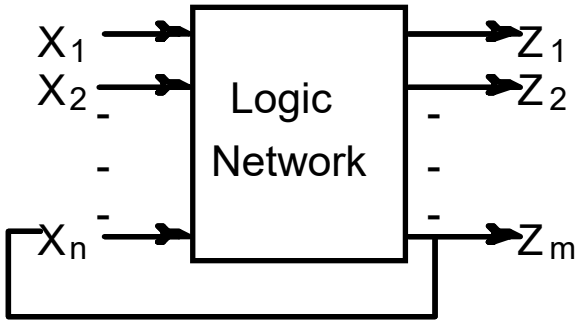
- **Combinational Logic (CL)**



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.

Representations of Combinational Logic

- Text Description
- Circuit Description
  - Transistors — Not covered in 369
  - Logic Gates
- Truth Table
- Boolean Expression

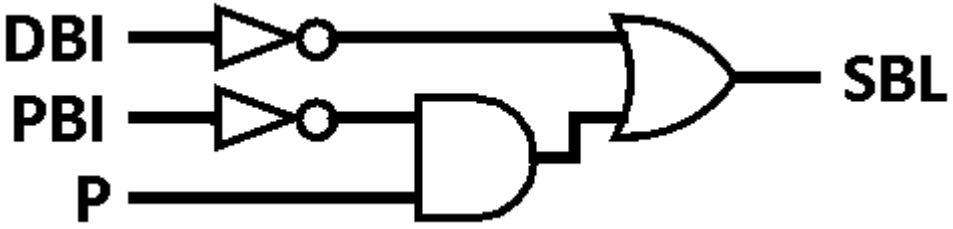
- 😳 ***All are equivalent!*** 😳

Example: Simple Car Electronics

- Door Ajar (DriverDoorOpen, PassengerDoorOpen)
  - $\mathrm{DA} = \mathrm{DDO} + \mathrm{PDO}$

- High Beam Indicator (LightsOn, HighBeamOn)
  - $\mathrm{HBI} = \mathrm{LO} \cdot \mathrm{HBO}$

- Seat Belt Light (DriverBeltIn, PassengerBeltIn, Passenger)
  - $\mathrm{SBL} = \overline{\mathrm{DBI}} + (\mathrm{P} \cdot \overline{\mathrm{PBI}})$

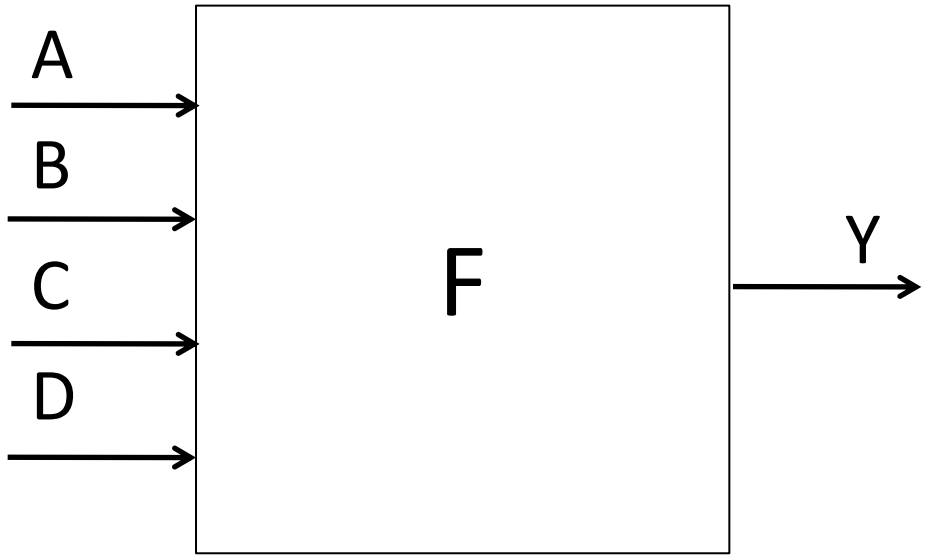


Truth Tables

- Table that relates the inputs to a combinational logic (CL) circuit to its output
  - Output *only* depends on current inputs
  - Use abstraction of 0/1 instead of high/low voltage
  - Shows output for *every* possible combination of inputs ("black box" approach)

- How big is the table?
  - 0 or 1 for each of N inputs
  - Each output is a separate function of inputs, so don't need to add rows for additional outputs, so 2^N rows

CL General Form



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

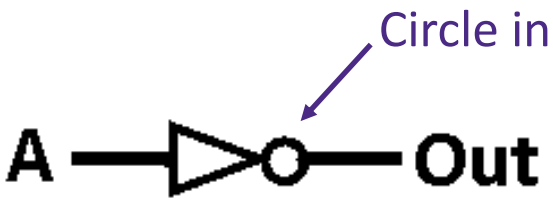
Function maps each row to 0 or 1, so $2^{(2^N)}$ possible functions

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)

Logic Gates (1/2)

- Special names and symbols:

NOT



A	Out
0	1
1	0

AND

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

OR

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

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

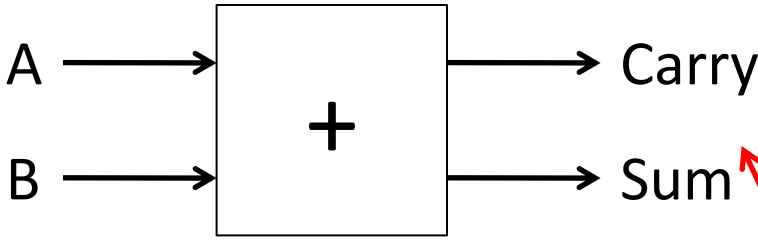
More Complicated Truth Tables

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

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

Boolean Algebra

- Represent inputs and outputs as variables
  - Each variable can only take on the value 0 or 1
- Overbar is NOT: "logical complement"
  - If A is 0, then $\overline{\mathrm{A}}$ is 1 and vice-versa
- Plus $(+)$ is 2-input OR: "logical sum"
- Product $(\cdot)$ is 2-input AND: "logical product"
- All other gates and logical expressions can be built from combinations of these
  - *e.g.*, $\mathrm{A\ XOR\ B} = \mathrm{A} \oplus \mathrm{B} = \overline{\mathrm{A}}\mathrm{B} + \overline{\mathrm{B}}\mathrm{A}$

Truth Table to Boolean Expression

- Read off of table
  - For 1, write variable name
  - For 0, write complement of variable
- *Sum of Products (SoP)*
  - Take rows with 1's in output column, sum products of inputs
  - $C = \overline{A}B + \overline{B}A$
- *Product of Sums (PoS)*
  - Take rows with 0's in output column, product the sum of the *complements of the inputs*
  - $C = (A + B) \cdot (\overline{A} + \overline{B})$

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

We can show that these are equivalent!