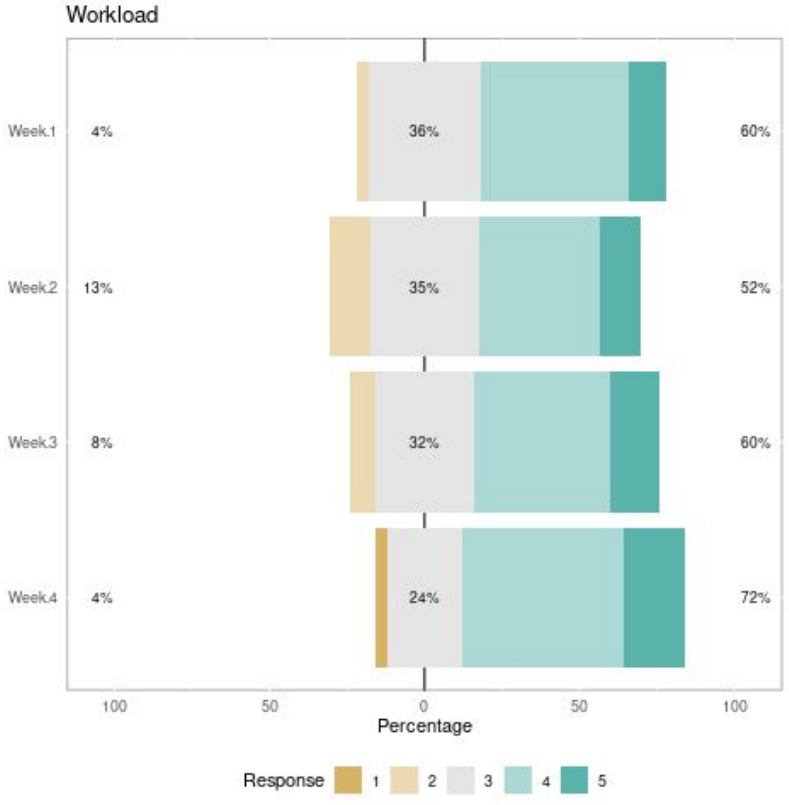
0x08 - Memory and Finite State Machines

ENGR 3410: Computer Architecture

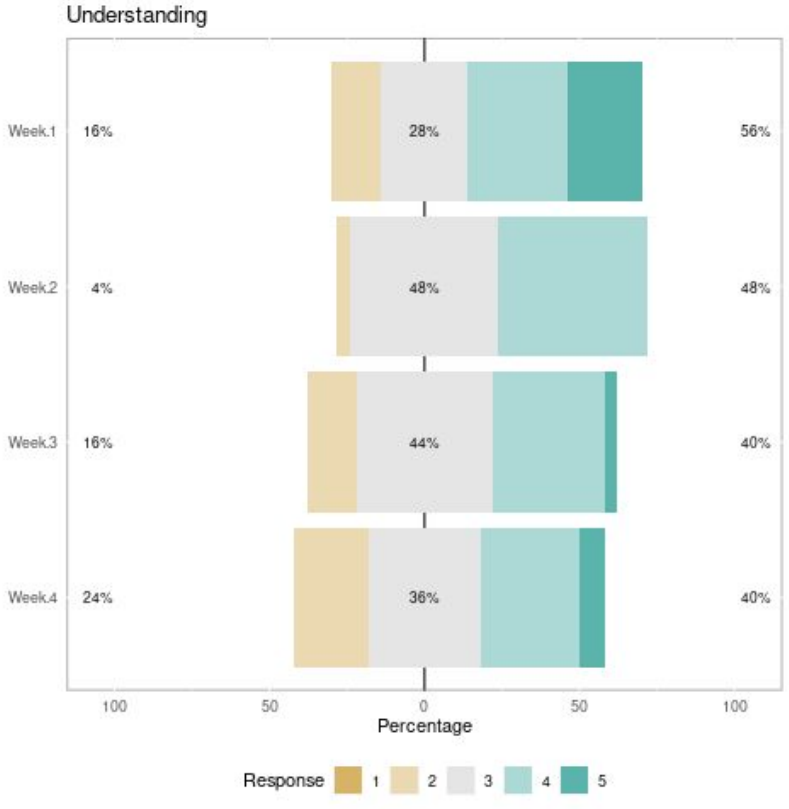
Jon Tse

Fall 2020

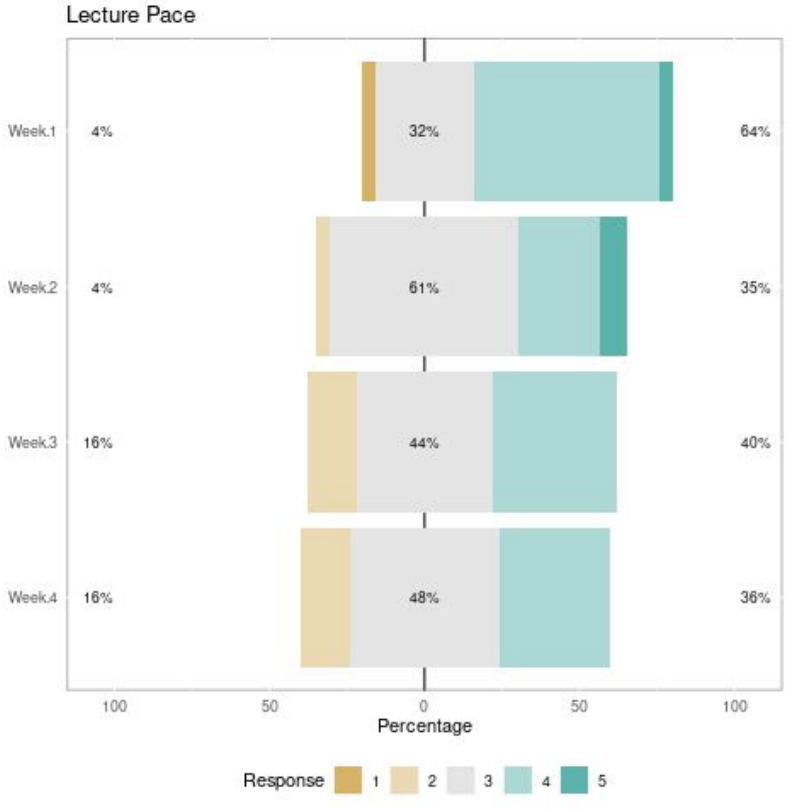
Feedback - Workload



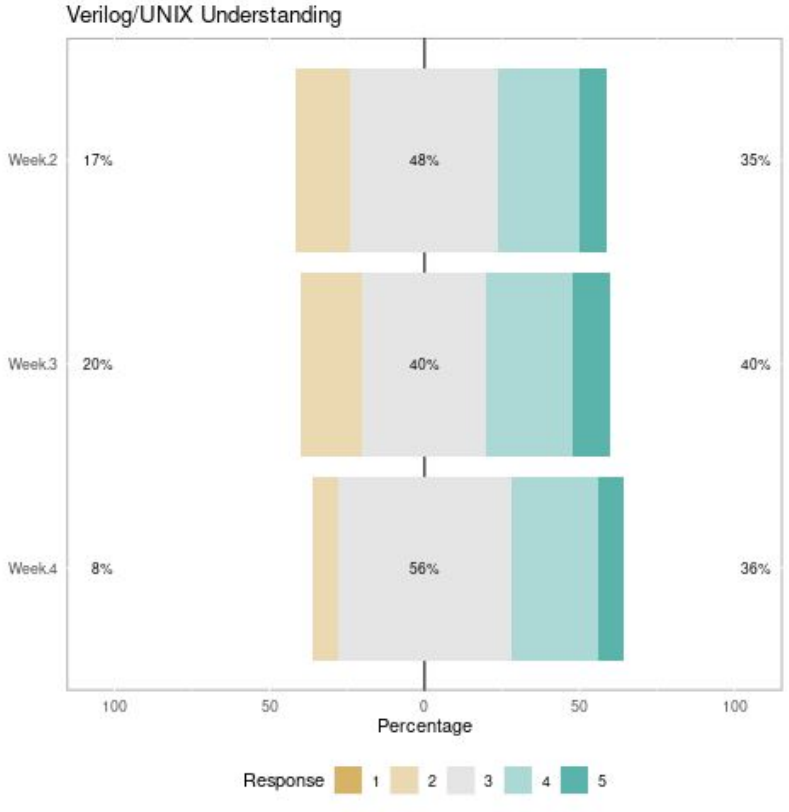
Feedback - Understanding



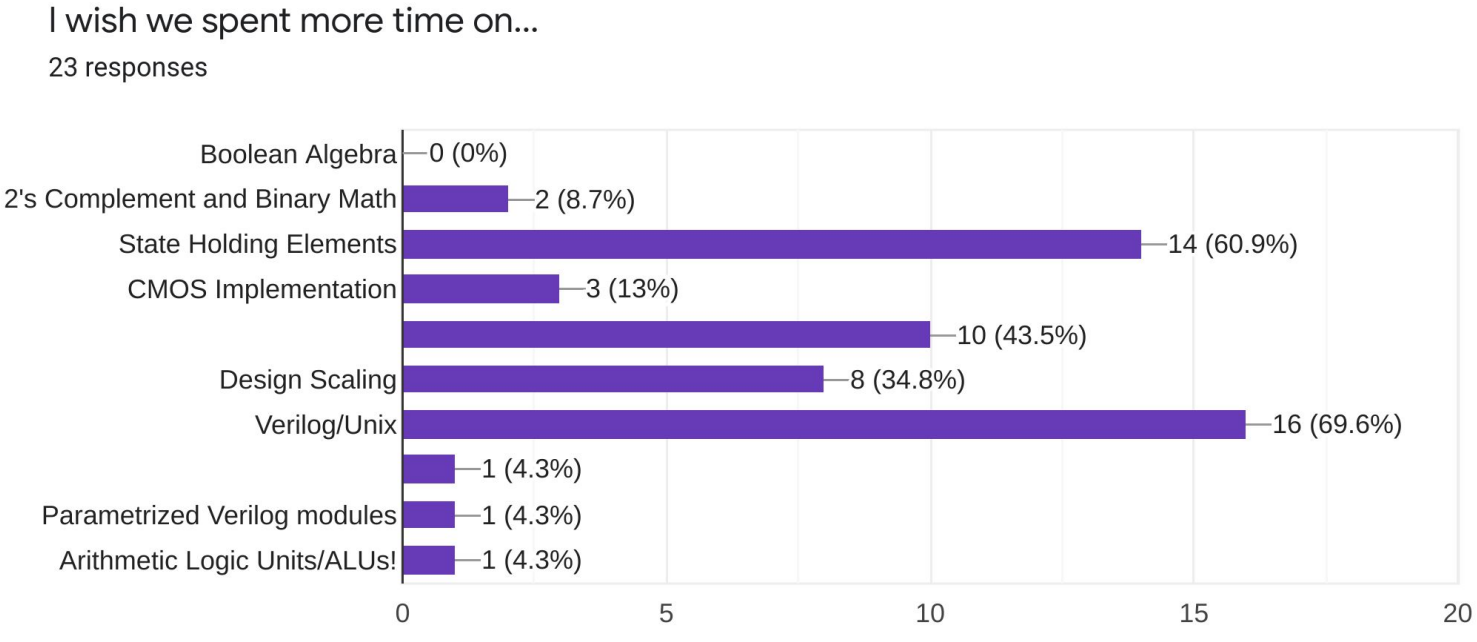
Feedback - Pace



Feedback - Verilog

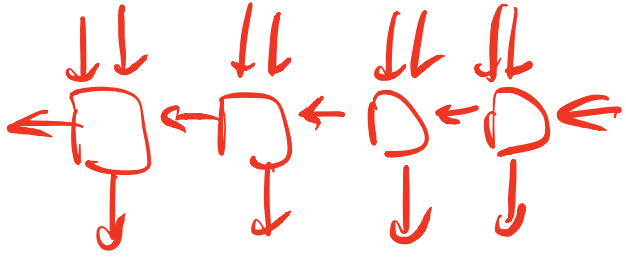


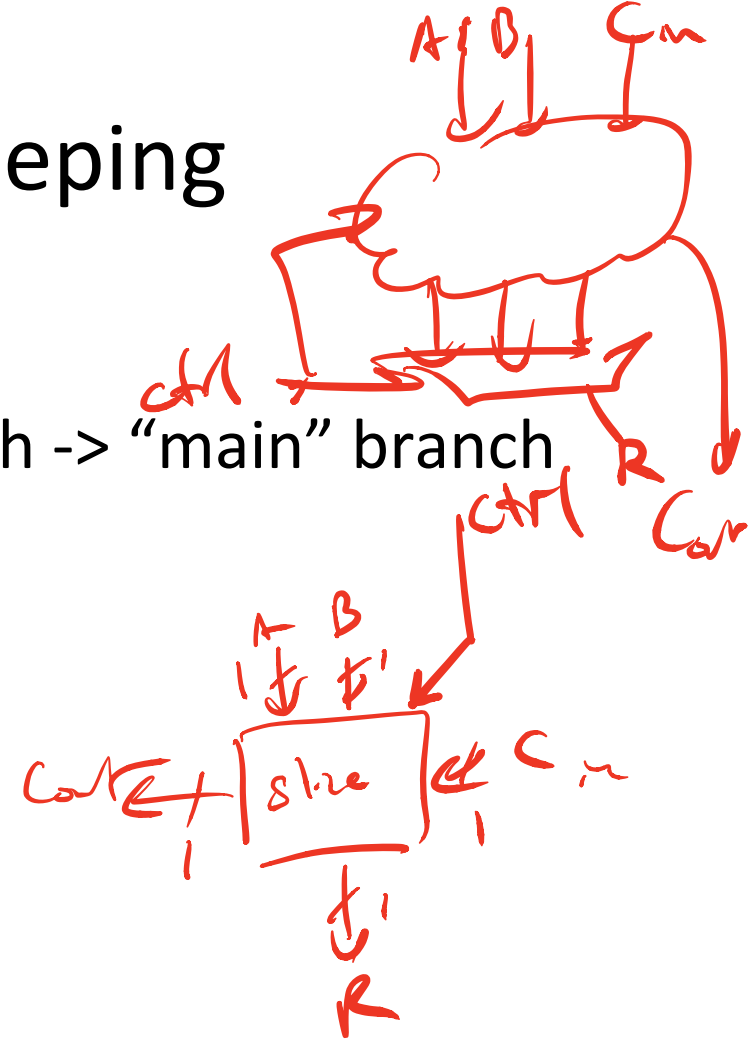
Feedback


I wish we spent more time on...
23 responses
Boolean Algebra
2's Complement and Binary Math
State Holding Elements
CMOS Implementation
Design Scaling
Verilog/Unix
Parametrized Verilog modules
Arithmetic Logic Units/ALUs!
0 (0%)
2 (8.7%)
14 (60.9%)
3 (13%)
10 (43.5%)
8 (34.8%)
16 (69.6%)
1 (4.3%)
1 (4.3%)
1 (4.3%)
0
5
10
15
20

Housekeeping

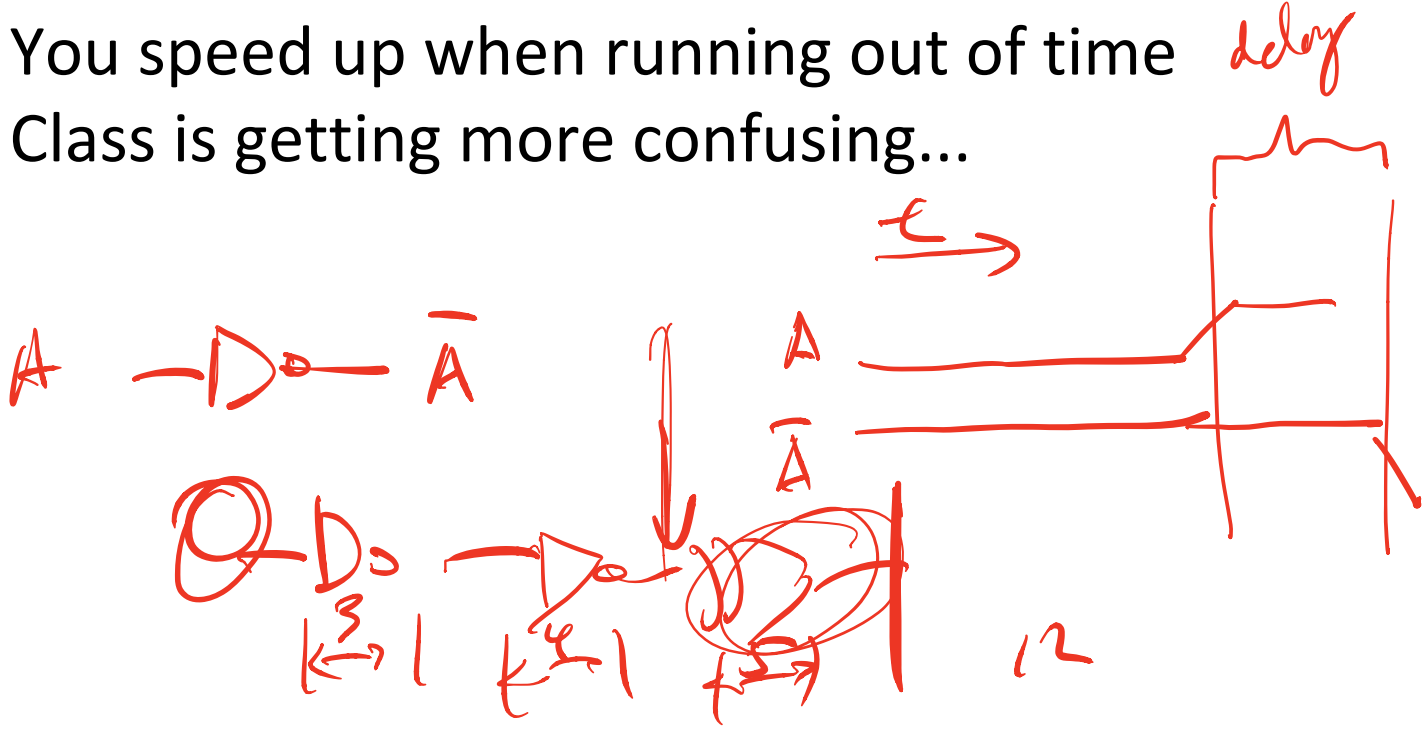
- github - “master” branch -> “main” branch
- Discussion of HW5
- Discussion of Lab 2
- Discussion of Lab 3





Feedback

- Delay in Verilog is confusing
- Issues with Makefiles
- You speed up when running out of time
- Class is getting more confusing...

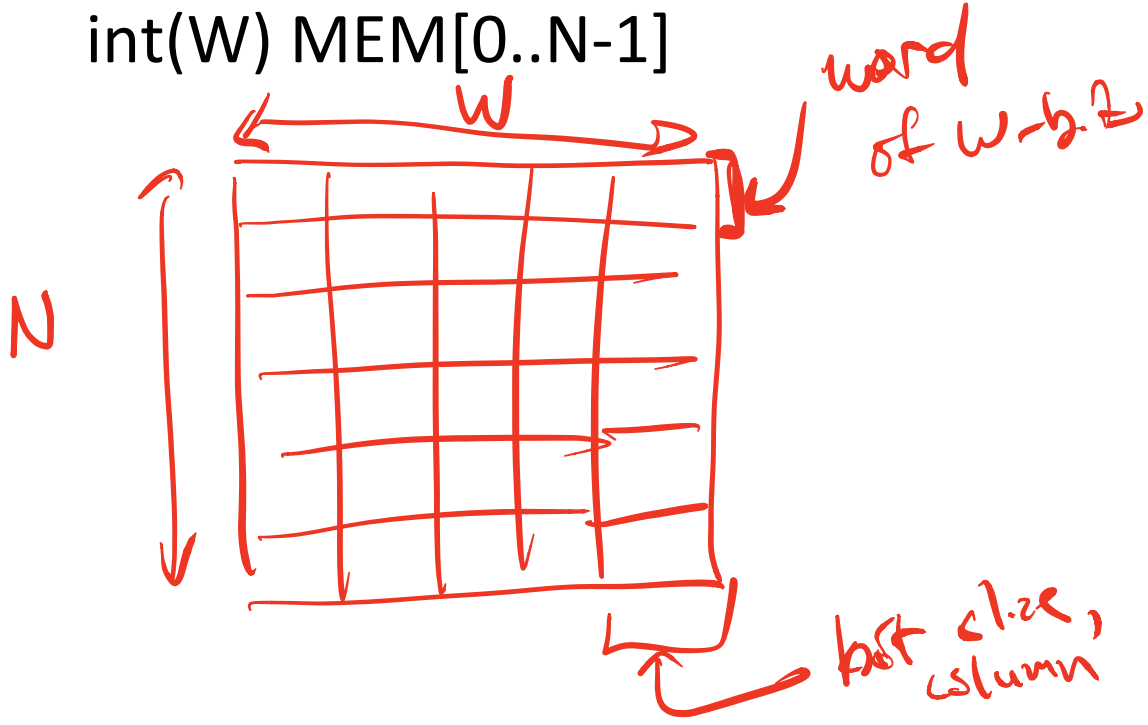


Memory

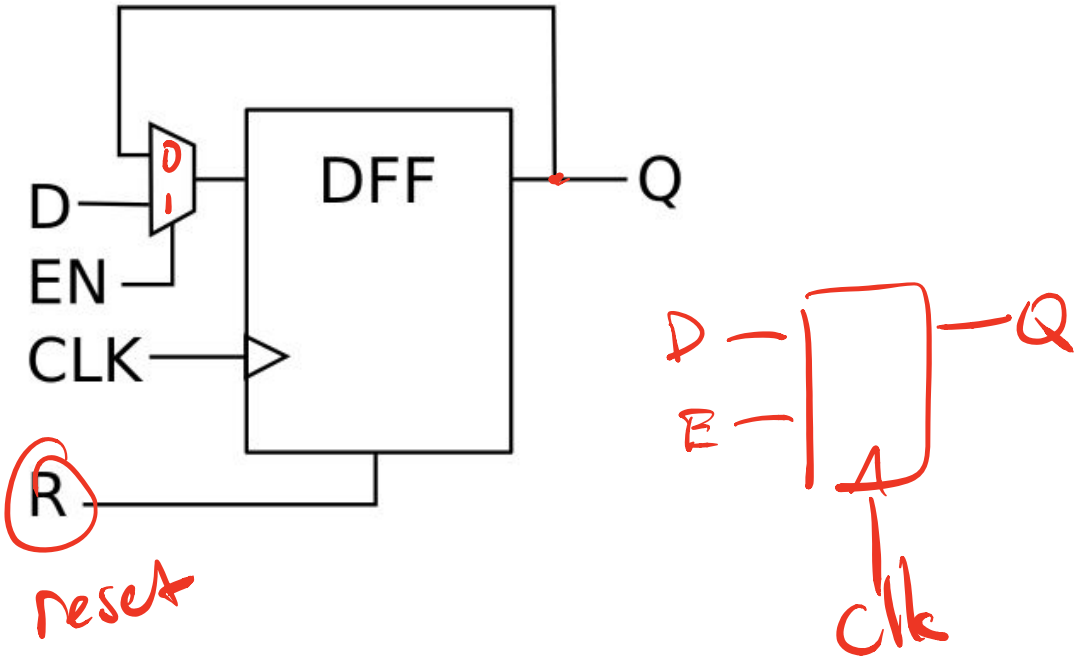
Array Model

wire [W-1:0] MEM [N-1:0]

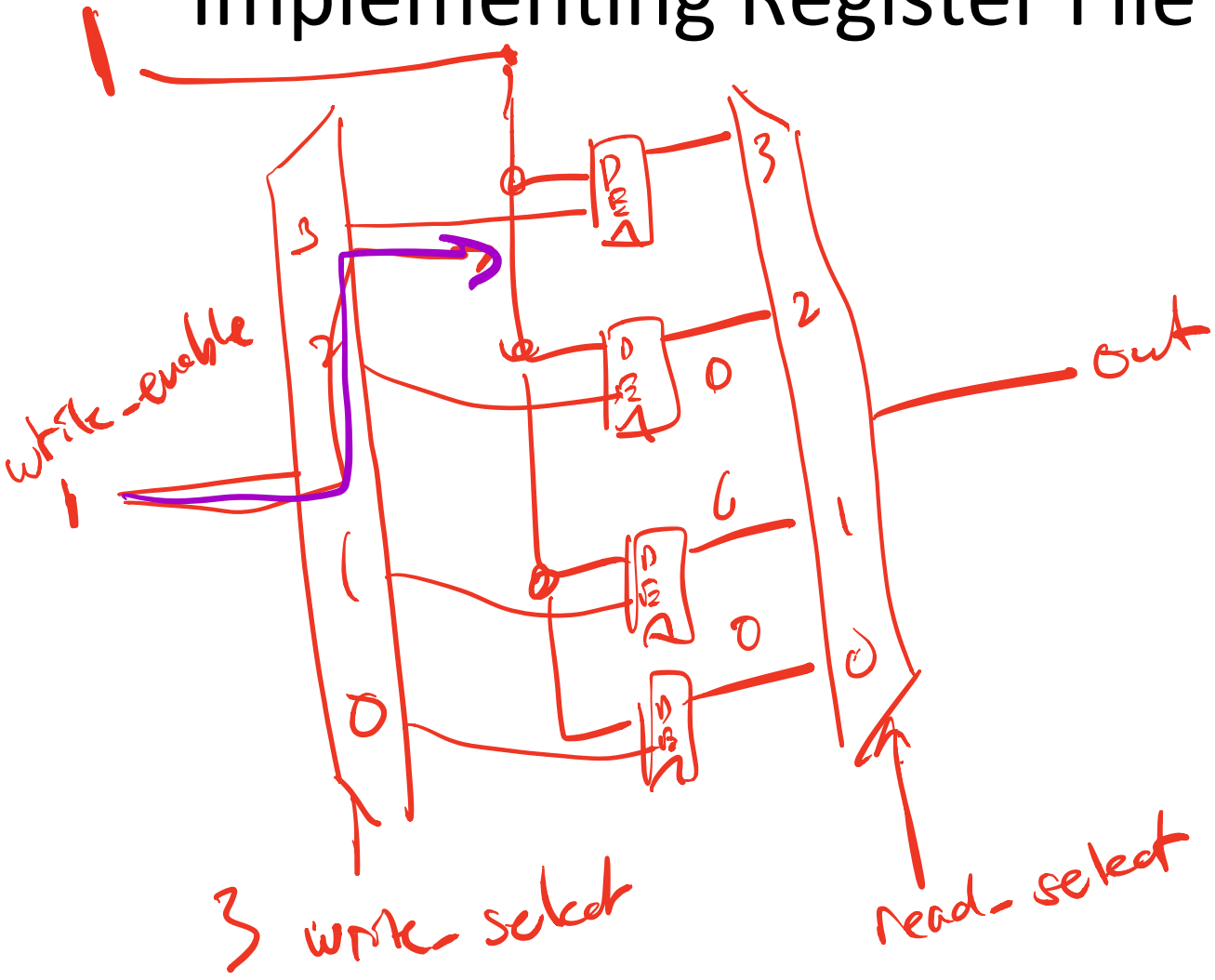
int(W) MEM[0..N-1]



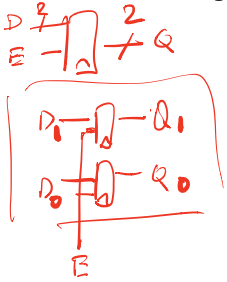
Enabled D-Flip Flop



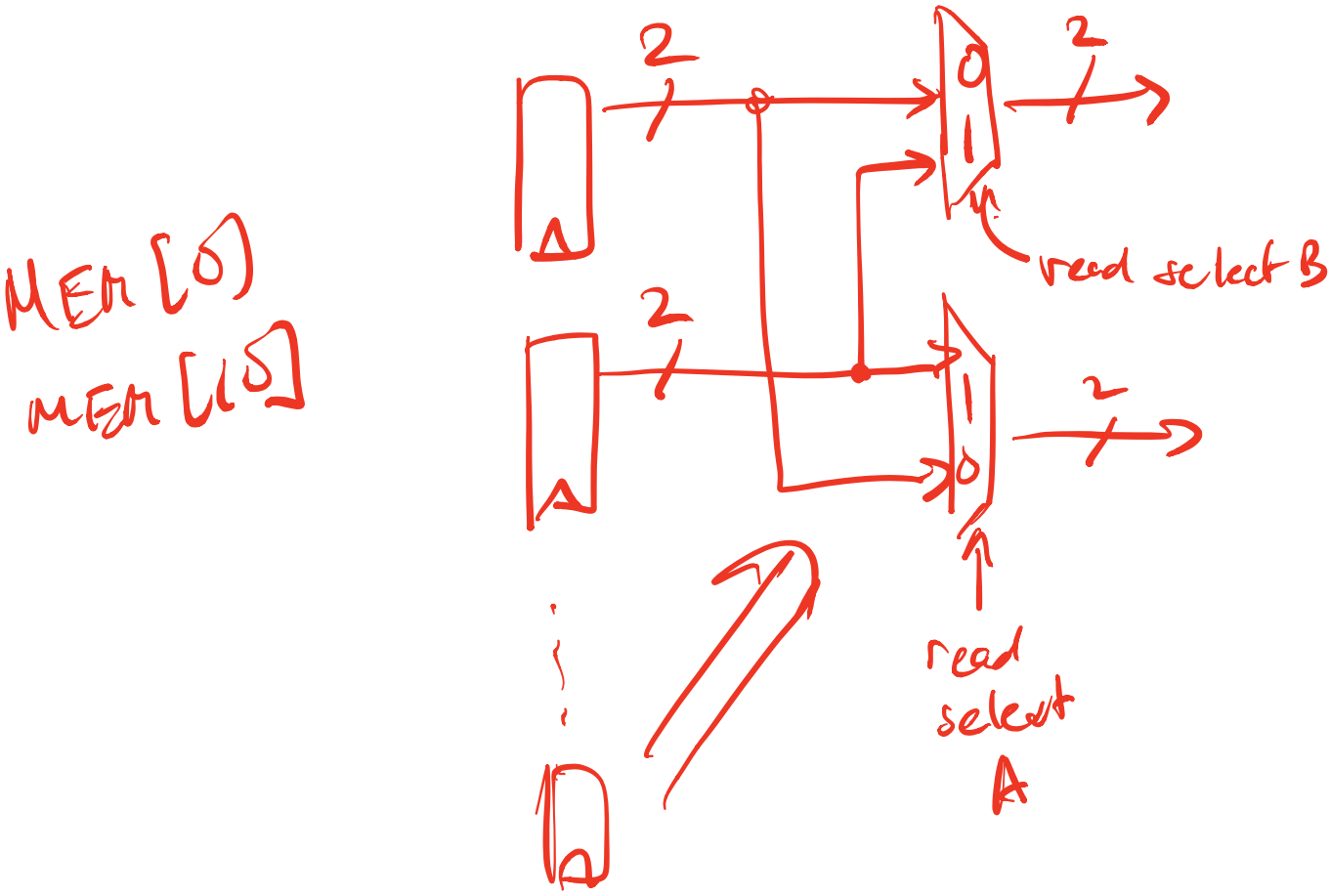
Implementing Register File



Implementing Register File



Implementing Register File

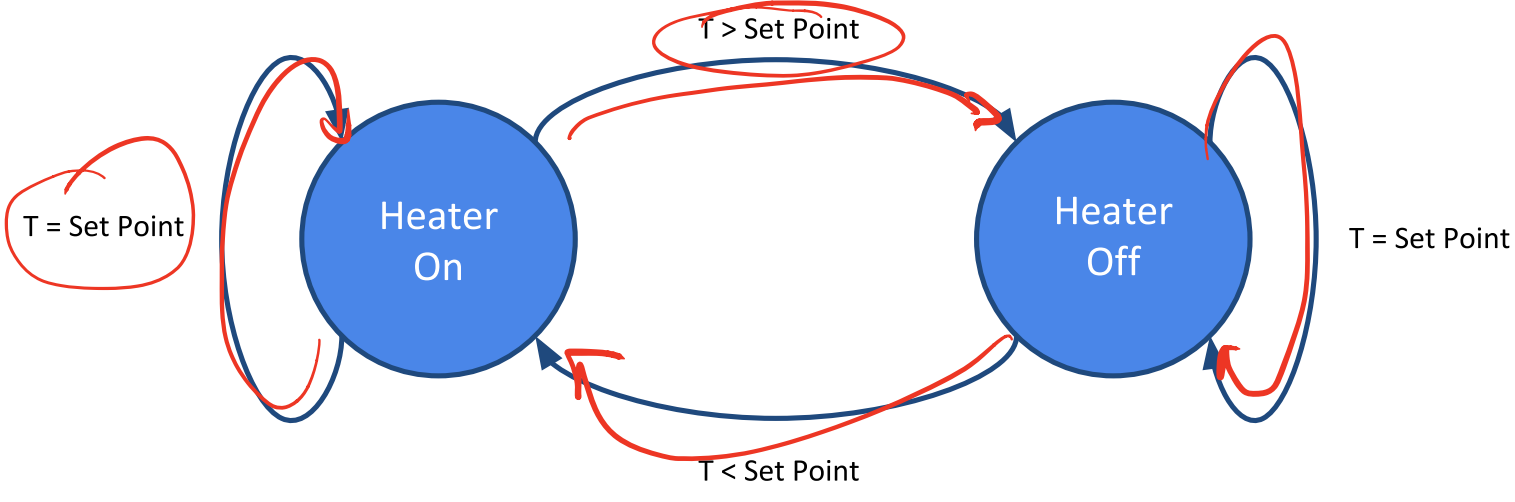


Finite State Machines

Collection of States (Node) and Transitions (Edge)

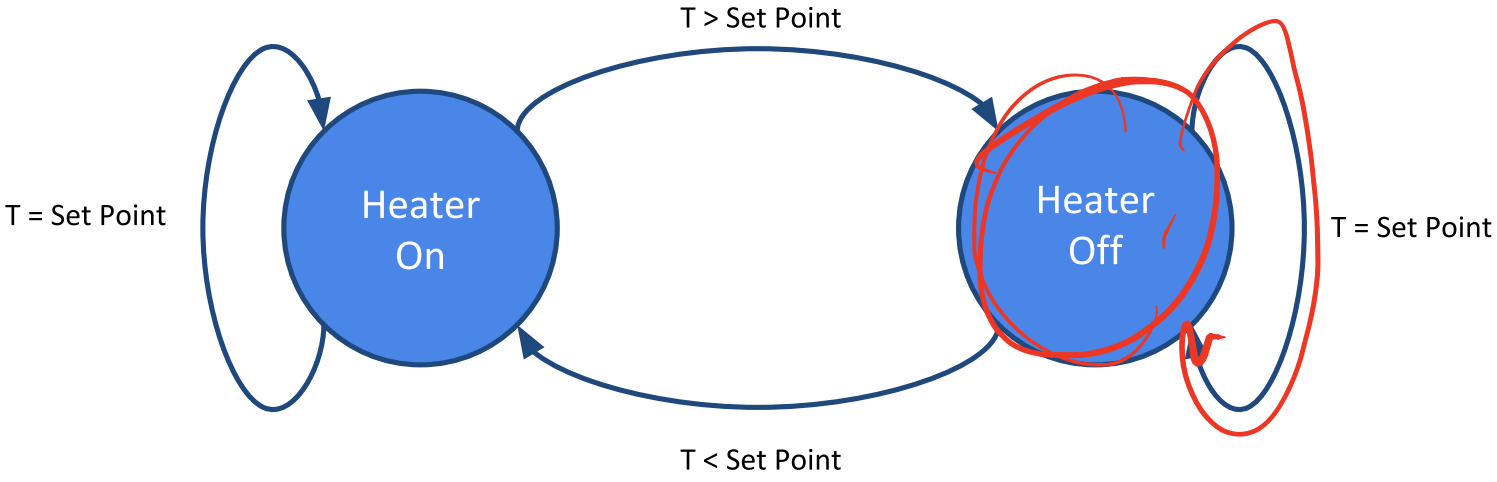
Transitions are guarded by conditions.

Example: ~~Refrigerator~~ Heater



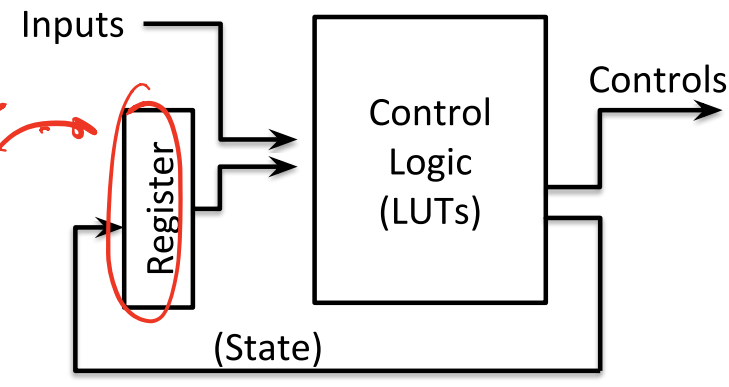
Finite State Machines

- In Computer Architecture, FSMs:
  - Usually transition on a clock edge
  - Are Complete
    - All states define transitions for all inputs
  - Are deterministic



FSM Implementation

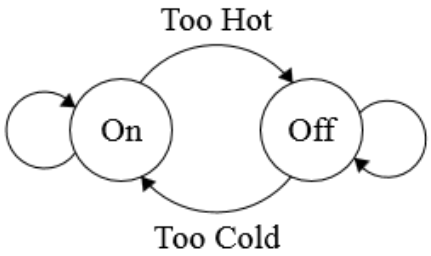
- Register to hold current state
- Wires to provide inputs (arguments)
- Look Up Table(s) to map transitions



Current State	Inputs	Resulting State
Heater Off	Too Cold	Heater On
Heater Off	---	Heater Off
Heater On	Too Hot	Heater Off
Heater On	---	Heater On

Transitions -> LUT

Current State	Inputs	Resulting State
Heater Off	Too Cold	Heater On
Heater Off	---	Heater Off
Heater On	Too Hot	Heater Off
Heater On	---	Heater On

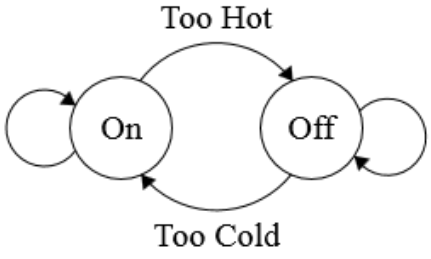


input output

State	Cold?	Hot?	Next State
Off	Yes	Yes	X
Off	Yes	No	On
Off	No	Yes	Off
Off	No	No	Off
On	Yes	Yes	X
On	Yes	No	On
On	No	Yes	Off
On	No	No	On

LUT -> Circuit

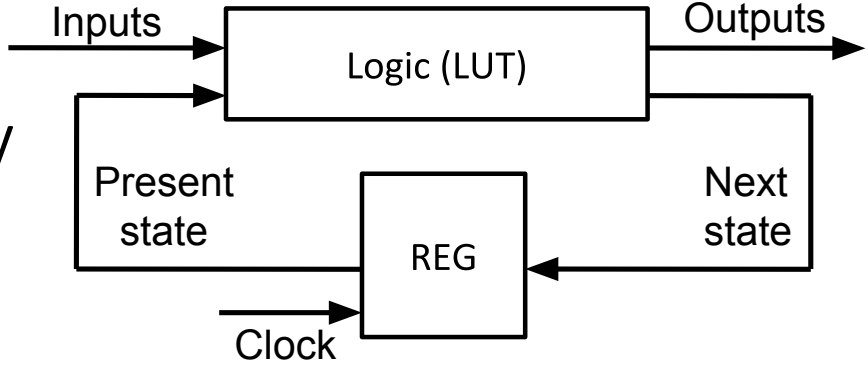
State	Cold?	Hot?	Next State
Off	Yes	Yes	X
Off	Yes	No	On
Off	No	Yes	Off
Off	No	No	Off
On	Yes	Yes	X
On	Yes	No	On
On	No	Yes	Off
On	No	No	On



$$q_{NEXT} = (Cold) + \overline{(Hot)}q$$

FSM Process Summary

- Design Requirements
- Define States
- Define Transitions
- Assign Values / Binary Codes
- Calculate Widths
- Make a LUT
- Connect Registers
- (Optional) Simplify



Binary Encoding

- States are stored as binary numbers
- Requires $\lceil \log_2 N \rceil$ registers for N states
- LUT requires a decoder (per usual)

One Hot Encoding

- States are stored as one bit being “hot”
- Requires N registers for N states
- LUT doesn’t require a decoder

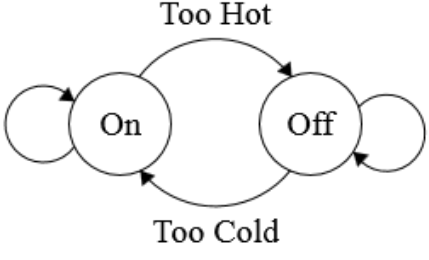
Binary	One Hot
00	0001
01	0010
10	0100
11	1000

One Hot Construction

- LUT Method works:
  - N states = N LUT address bits (+ input bits)
  - This creates many “don’t care” LUT rows
    - 2^N rows, only care about N of them
- LUT Simplification
  - Same process as before
  - Apply LOTS of simplification

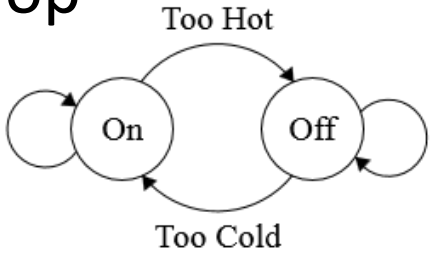
One Hot Construction

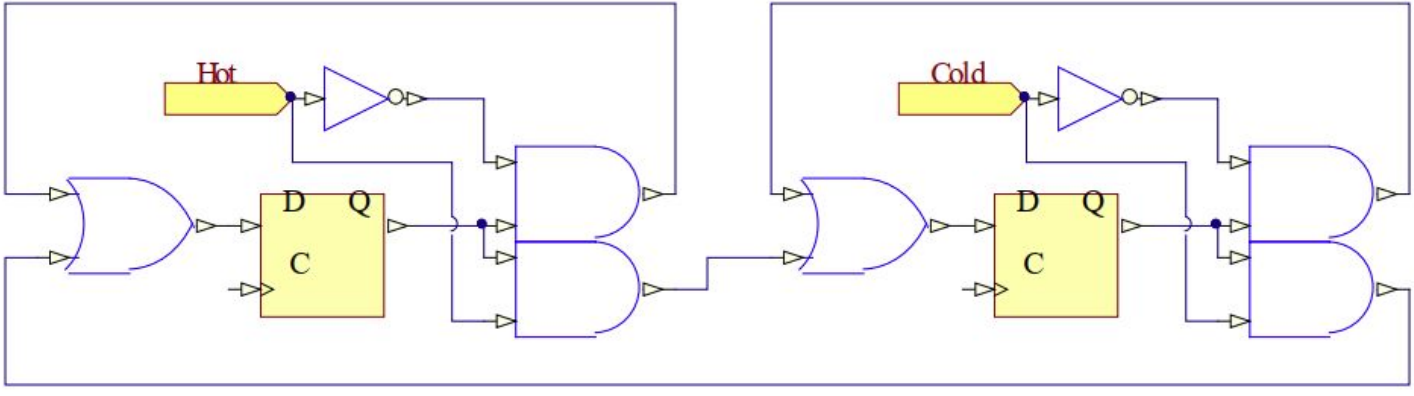
- Start with Flow Diagram
- Replace each state with a D Flip Flop
- D input is an OR gate
  - Each input is a transition



One Hot Construction

- Start with Flow Diagram
- Replace each state with a D Flip Flop
- D input is an OR gate
  - Each input is a transition
    - Source State AND transition condition





Encoding Comparison

Binary

- Fewer Flip Flops
- Slower
- Wider LUTs

One Hot

- More Flip Flops
- Faster Clock Speeds
- Simpler Logic
- Many Illegal States
  - What if 2 bits are hot?
  - LOTS of “Don’t Cares” in logic

Wait States

- Typical Data Flow:
  - FSM creates control signals for a clocked element
  - Other element takes N cycles to finish
  - FSM proceeds to next task
- Wait States delay the FSM
  - Very Simple Single Transition
  - One Hot Just a D Flip Flop, no logic!