

# Intro to Digital Design

## L8: Memory, Useful Doodads

**Instructor:** Naomi Alterman

**Teaching Assistants:**

Derek de Leuw

Isabel Froelich

Kevin Hernandez

Aarjav Jain

Packard Stephenson

# Administrivia

## ❖ Lab 8 – Project!

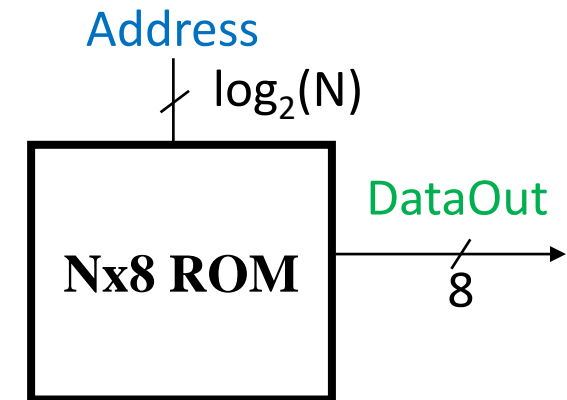
- 2 weeks to work on it – **don't wait to start!**
  - **Milestone 1:** Check-in during lab demo next week
    - Submission **required!** Block diagram of your overall system design and at least one new module in Verilog
  - **Milestone 2:** Turn in code + report + bitfile by end of dead week
  - **Milestone 3:** Demo your bitfile and turn in your lab kit
- 8 suggested projects, or get your own approved
  - Not all are worth the same number of points (“full credit” is 160)
  - Think carefully about what you want to tackle (*e.g.*, complex FSM, LED board, multiple “clock speeds”)
- Bonus points for adding cool features and early finish
  - Up to **20 points for extra features**; up to **10 points for early finish**

# Outline

- ❖ **Memory**
  - ROM
  - RAM
- ❖ Useful Doodads

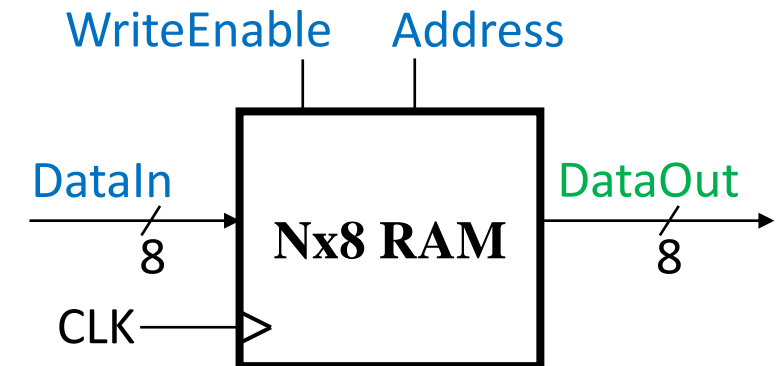
# Storage Element: Idealized ROM

- ❖ “Read Only Memory”
  - $N \times M$ : An array of  $N$   $M$ -Bit words
  - Address input selects the word driven on the output
  - Data “burned in” by the manufacturer (or your bitfile) and retained even if power is lost
- ❖ Difference between a ROM and your seven segment decoder is *density*
  - (and thus underlying circuit implementation)
- ❖ What can we do with a truly *read-only* memory?
  - Math look-up tables (trig functions)
  - FSM state transitions
  - Bitmap images and other simple patterns



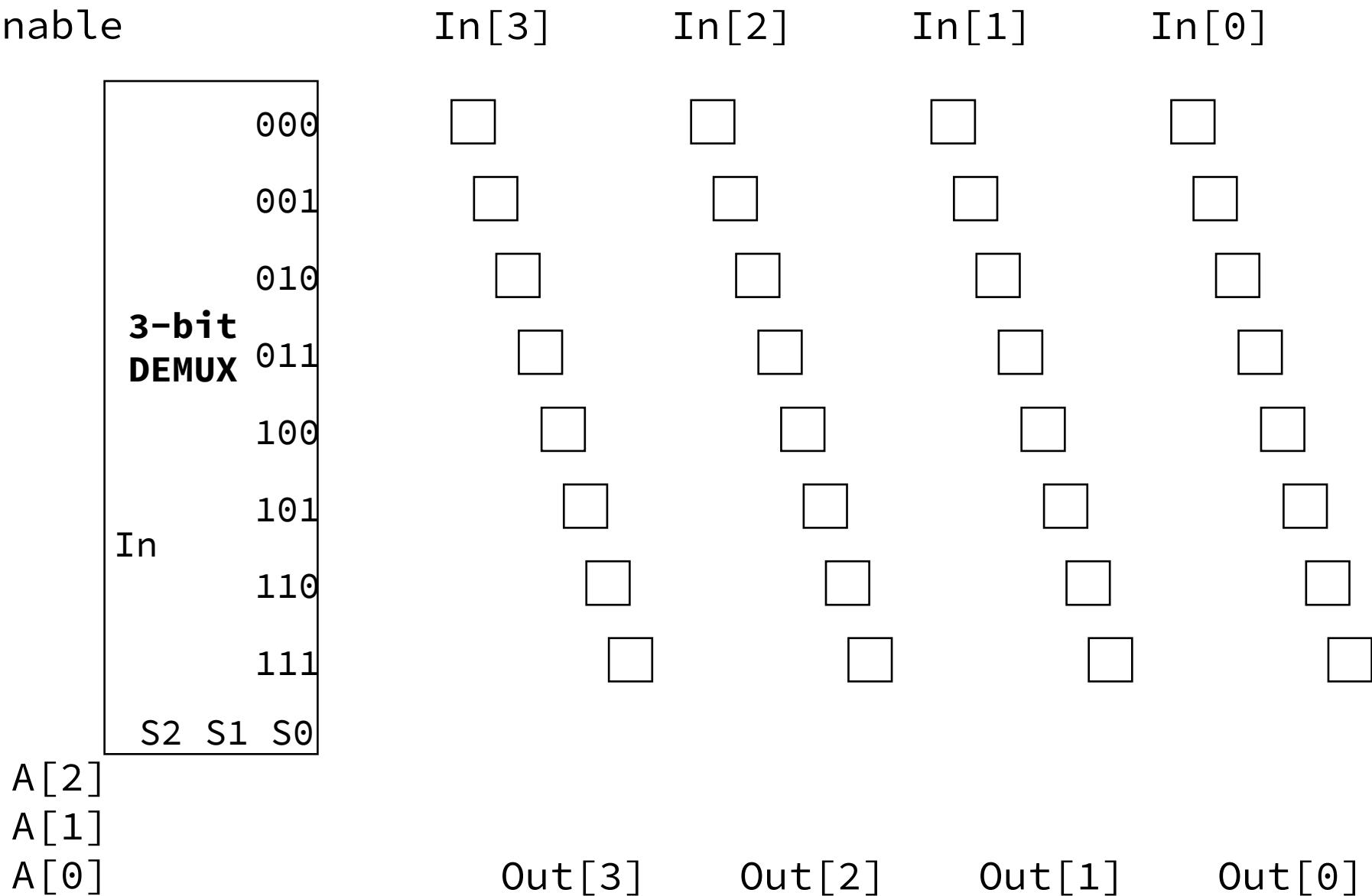
# Storage Element: Idealized RAM

- ❖ “Random Access Memory”
  - Can now **write** as well as read
  - Data lost on reset or power-off
  - On clock edge, **DataIn** written to memory cell at **Address** if **WriteEnable** = 1
  - **DataOut** is still completely combinational (changes to **Address** immediately change the data output)
- ❖ Simplest and most robust implementation: “SRAM” – implemented as an array of flip flops with muxed inputs and outputs
- ❖ Higher-density but more touchy: “DRAM” – implemented with circuit wizardry



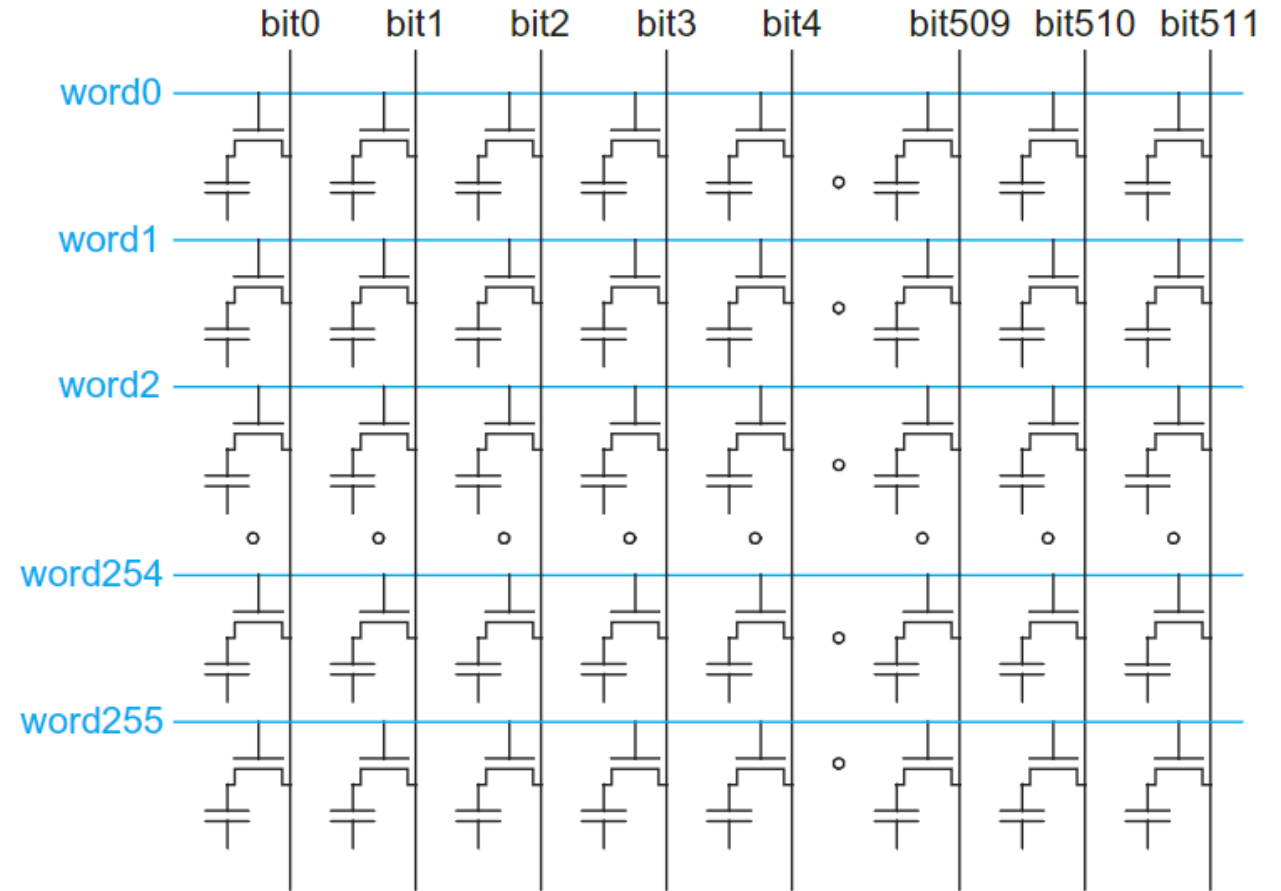
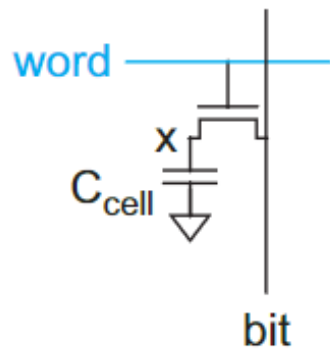
# 8x4 SRAM

WriteEnable



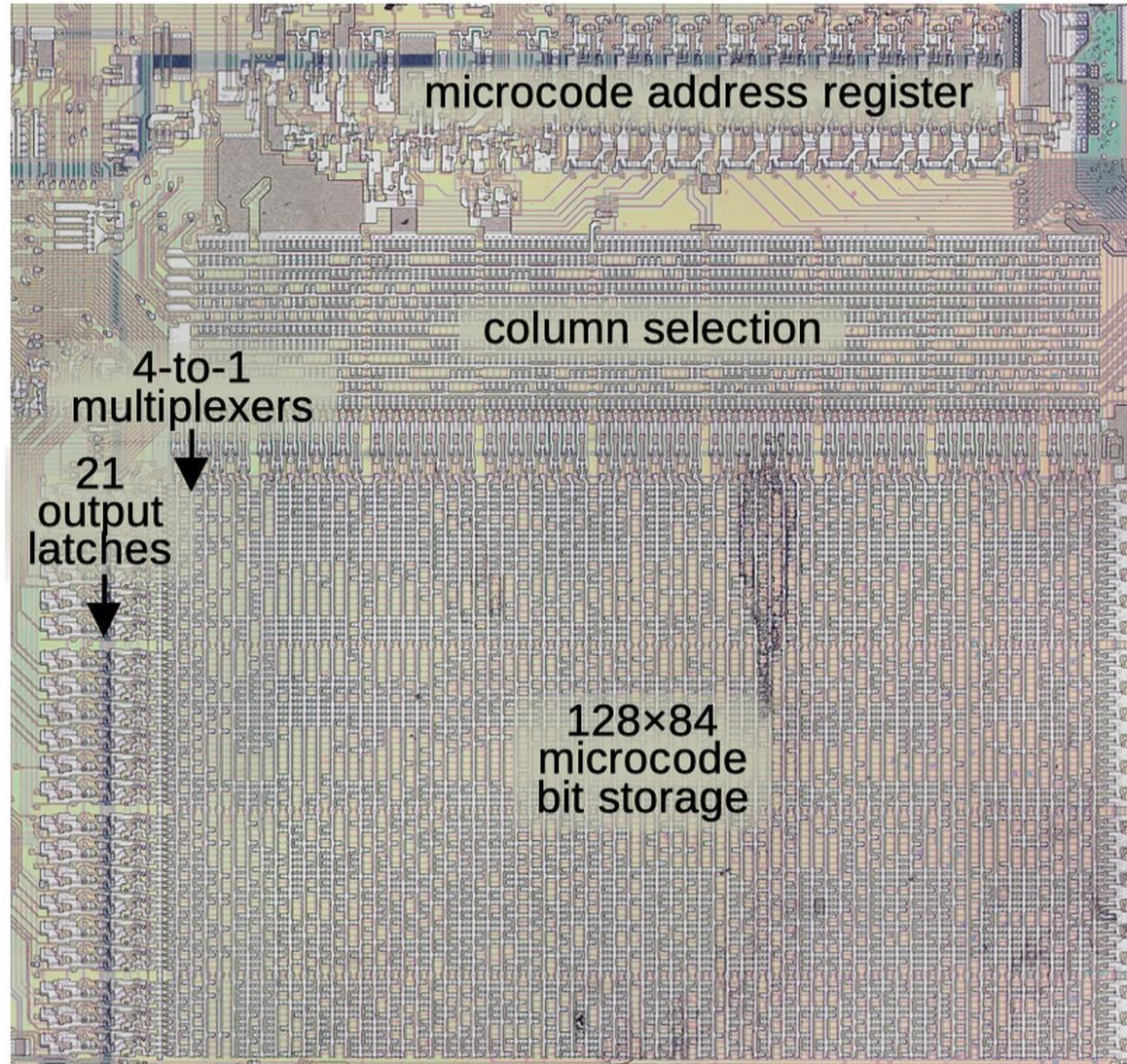
# Sneaky Peaky: DRAM

- ❖ Highest memory density, but very touchy
- ❖ Bits “decay” over time and need to be refreshed (read and written)
- ❖ One bit cell:



Diagrams from CMOS VLSI Design: A Circuits and Systems Perspective (4<sup>th</sup> ed) by Weste & Harris

# Memory IRL: Microcode ROM in the original x86



# Outline

- ❖ Memory
- ❖ **Useful Doodads**
  - Comparators
  - Clock domains

# Comparators (Multibit)

- ❖ Zero comparator  $A == 0$ 
  - NOR all of A's bits
- ❖ Equality ( $A == B$ )
  - XNOR corresponding bits of A and B, then AND together
  - Compute  $A - B$ , then check if it  $== 0$
- ❖ Comparator ( $A < B$ ,  $A == B$ ,  $A > B$ )
  - $A < B$ : MSB of  $A - B$
  - $A == B$ : NOR of all bits of  $A - B$
  - $A > B$ : NOT of MSB of  $A - B$

# “Multiple Clocks” Via Enable Signals and Counters

- ❖ Even with that `clock_divider` module, we **only ever want to run our circuit on *one* clock if we can help it**
  - Logic stages that share a clock are called a **clock domain**
  - Crossing clock domains carelessly is a recipe for metastability
- ❖ So how can we slow things down?
  - Use flip flops with an Enable signal
  - Comparator on counter output generates an Enable pulse once every N cycles
  - Keeps all logic technically on the same clock domain, while allowing some things to happen “more slowly” than others