

# Memory

**Prof. Hakim Weatherspoon**

**CS 3410, Spring 2015**

Computer Science

Cornell University

See P&H Appendix B.8 (register files) and B.9

# Announcements

Make sure to go to **your** Lab Section this week

Completed Lab1 due **before** winter break, Friday, Feb 13th

Note, a **Design Document** is due when you submit Lab1 final circuit

Work **alone**

## Save your work!

- **Save often.** Verify file is non-zero. Periodically save to Dropbox, email.
- Beware of MacOSX 10.5 (leopard) and 10.6 (snow-leopard)

## Homework1 is out

Due a week before prelim1, Monday, February 23rd

*Work on problems incrementally, as we cover them in lecture (i.e. part 1)*

Office Hours for help

Work **alone**

Work alone, **BUT** use your resources

- Lab Section, Piazza.com, Office Hours
- Class notes, book, Sections, CSUGLab

# Announcements

Check online syllabus/schedule

- <http://www.cs.cornell.edu/Courses/CS3410/2015sp/schedule.html>
- Slides and Reading for lectures
- Office Hours
- ***Pictures of all TAs***
- Homework and Programming Assignments
- **Dates to keep in Mind**
  - **Prelims: Tue Mar 3rd and Thur April 30th**
  - ***Lab 1: Due next Friday, Feb 13th before Winter break***
  - Proj2: Due Thur Mar 26th before Spring break
  - Final Project: Due when final would be (not known until Feb 14t

Schedule is subject to change

# Collaboration, Late, Re-grading Policies

## “Black Board” Collaboration Policy

- Can discuss approach together on a “black board”
- Leave and write up solution independently
- Do not copy solutions

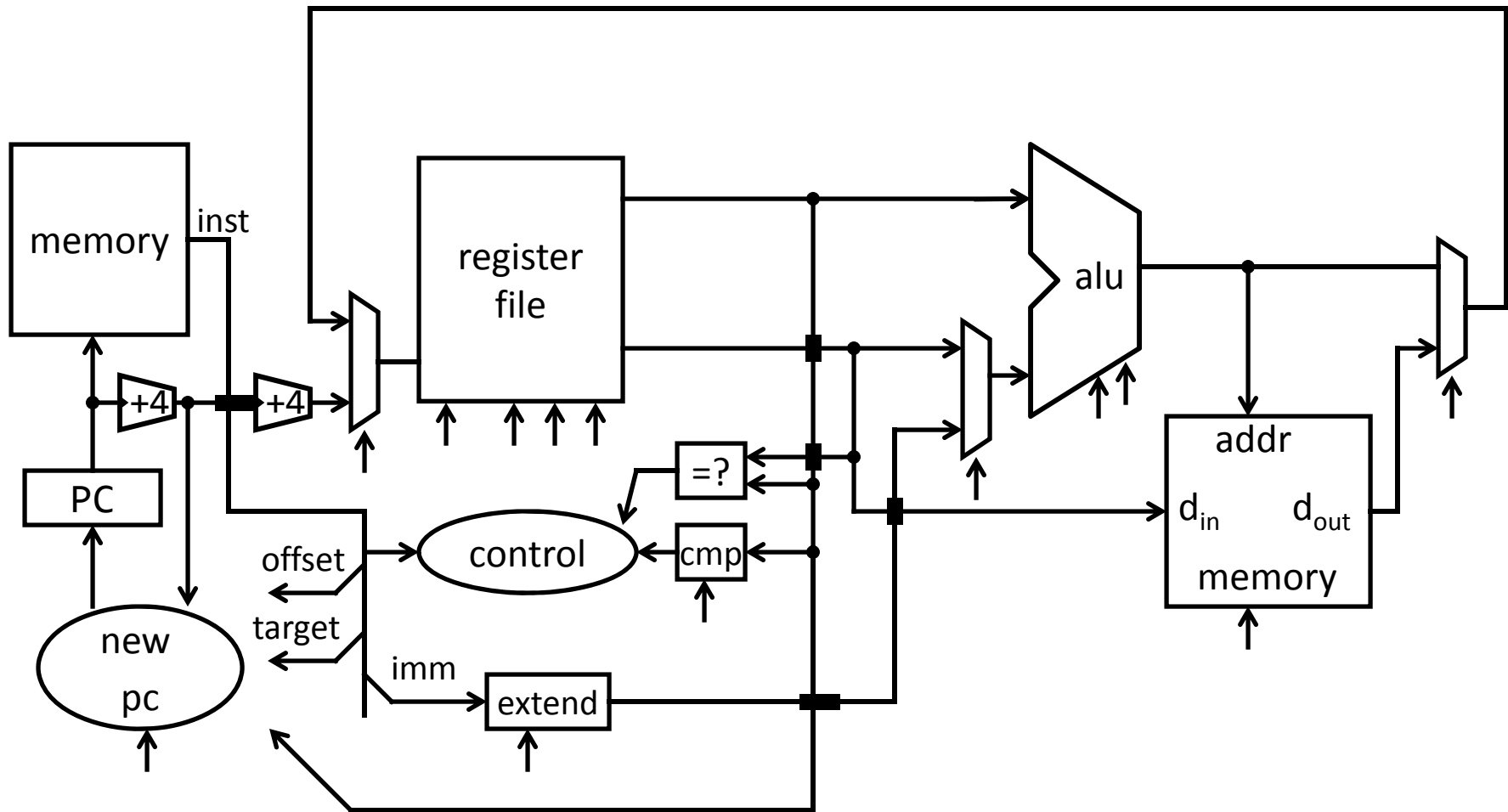
## Late Policy

- Each person has a total of **four** “slip days”
- Max of **two** slip days for any individual assignment
- Slip days deducted first for *any* late assignment, cannot selectively apply slip days
- For projects, slip days are deducted from all partners
- **25%** deducted per day late after slip days are exhausted

## Regrade policy

- Submit written request to lead TA,  
and lead TA will pick a different grader
- Submit another written request,  
lead TA will regrade directly
- Submit yet another written request for professor to regrade.

# Big Picture: Building a Processor



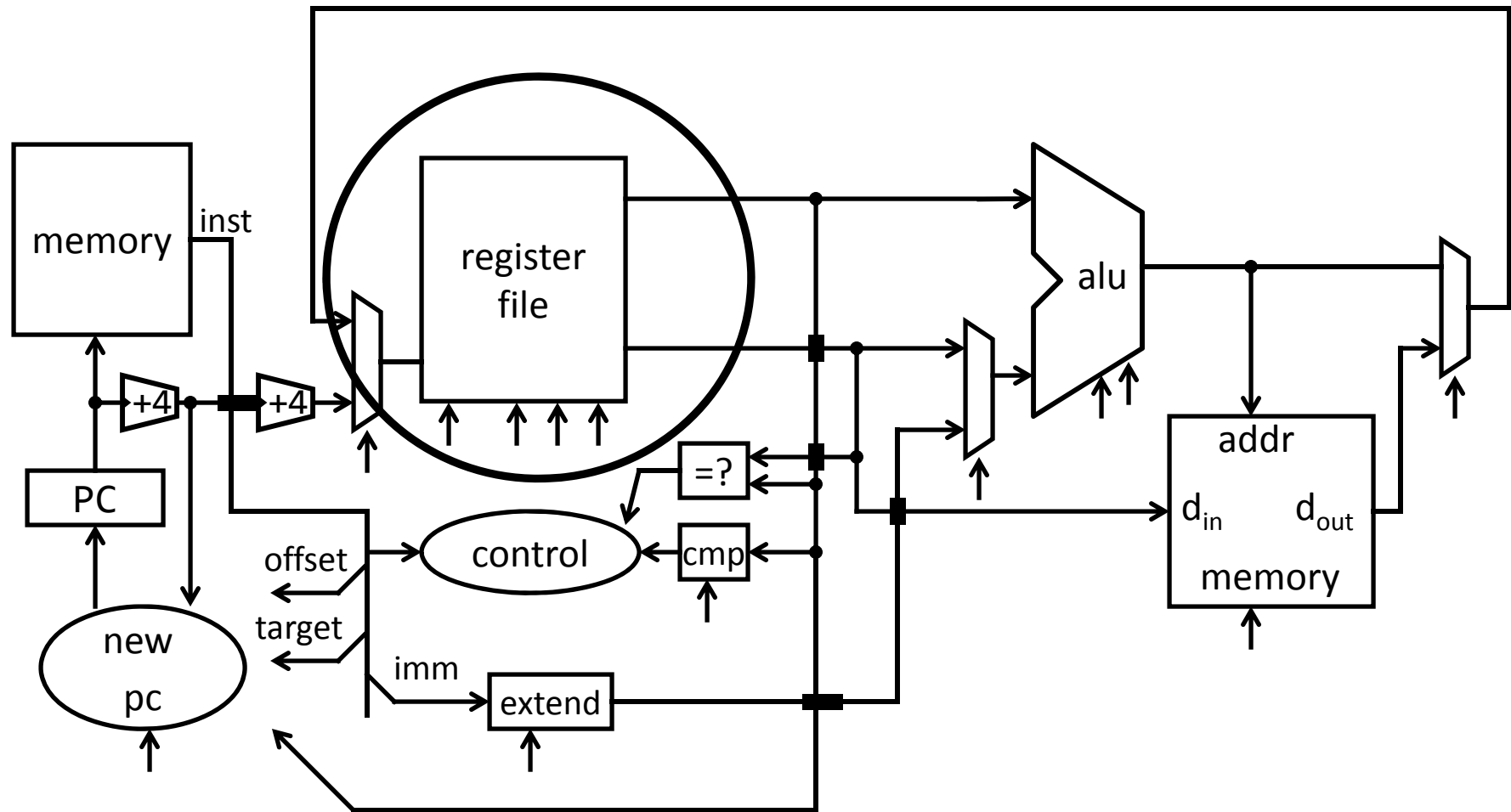
A Single cycle processor

# Goals for today

## Memory

- Register Files
- Tri-state devices
- SRAM (Static RAM—random access memory)
- DRAM (Dynamic RAM)

# Big Picture: Building a Processor

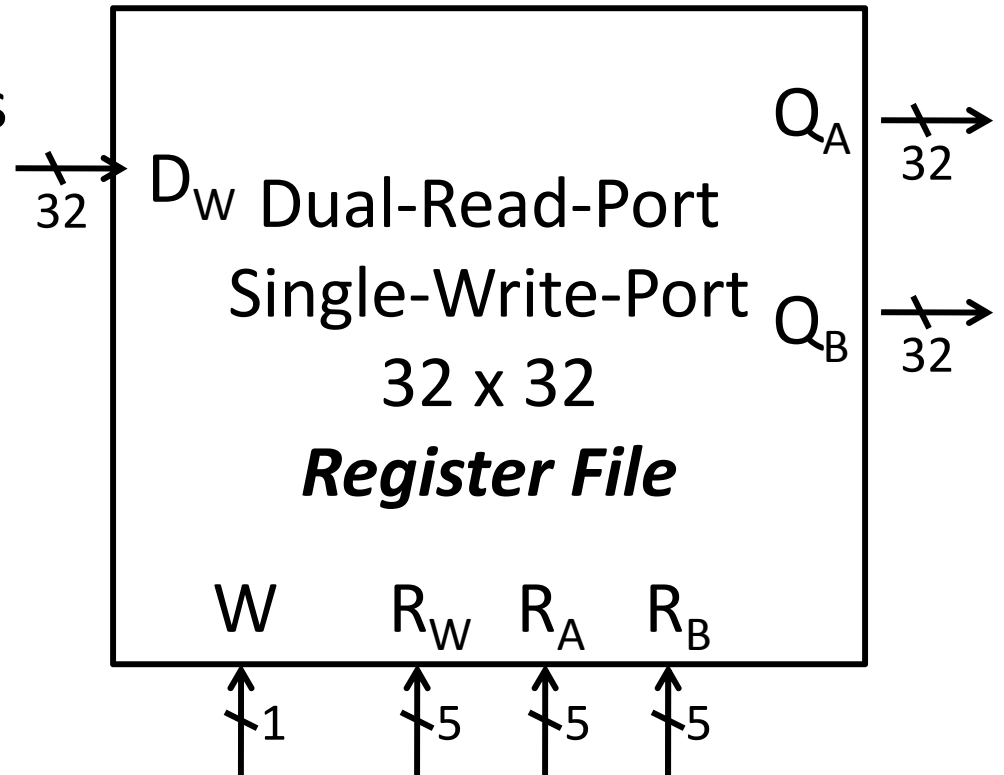


A Single cycle processor

# Register File

## Register File

- N read/write registers
- Indexed by register number





# Tradeoffs

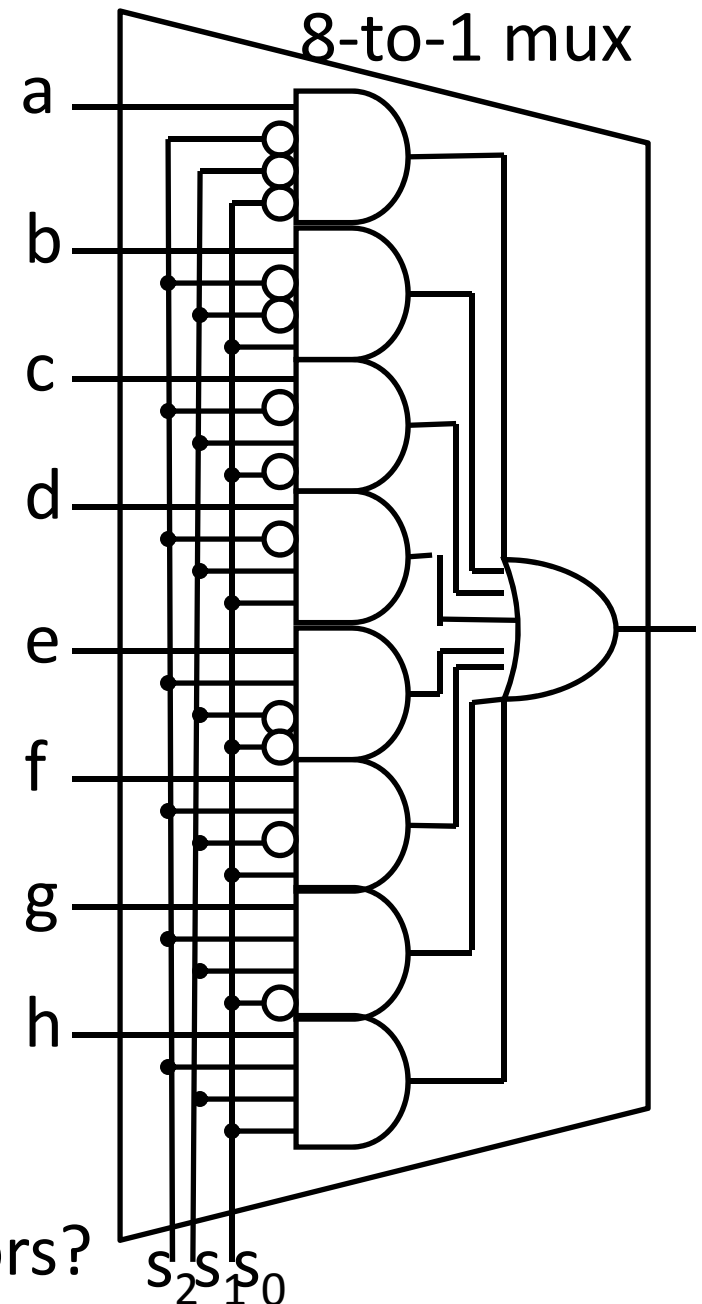
## Register File tradeoffs

- + Very fast (a few gate delays for both read and write)
- + Adding extra ports is straightforward
- Doesn't scale

e.g. 32Mb register file with  
32 bit registers

Need 32x 1M-to-1 multiplexor  
and 32x 20-to-1M decoder

How many logic gates/transistors?



# Takeaway

Register files are very fast storage (only a few gate delays), but does not scale to large memory sizes.

# Goals for today

## Memory

- CPU: Register Files (i.e. Memory w/in the CPU)
- Scaling Memory: Tri-state devices
- Cache: SRAM (Static RAM—random access memory)
- Memory: DRAM (Dynamic RAM)

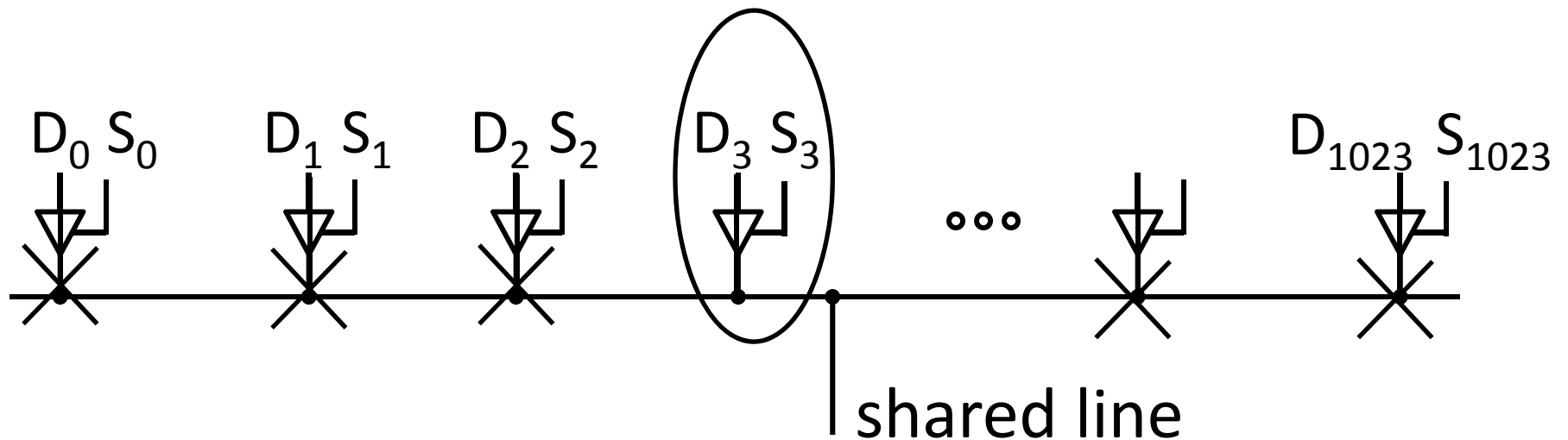
# Next Goal

How do we scale/build larger memories?

# Building Large Memories

Need a shared bus (or shared bit line)

- Many FlipFlops/outputs/etc. connected to single wire
- Only one output *drives* the bus at a time

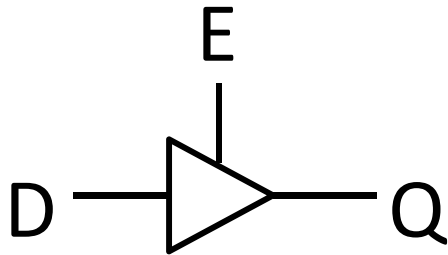


- How do we build such a device?

# Tri-State Devices

## Tri-State Buffers

- If enabled ( $E=1$ ), then  $Q = D$
- Otherwise,  $Q$  is not connected ( $z = \text{high impedance}$ )

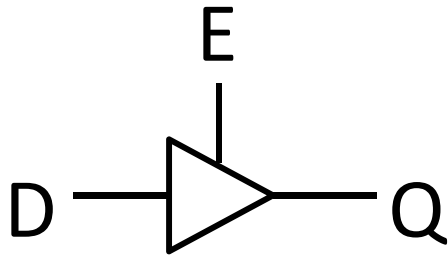


E	D	Q
0	0	z
0	1	z
1	0	0
1	1	1

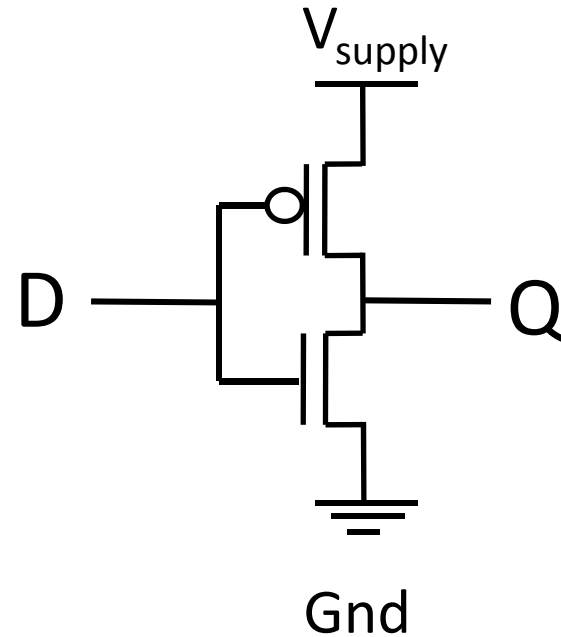
# Tri-State Devices

## Tri-State Buffers

- If enabled ( $E=1$ ), then  $Q = D$
- Otherwise,  $Q$  is not connected ( $z = \text{high impedance}$ )



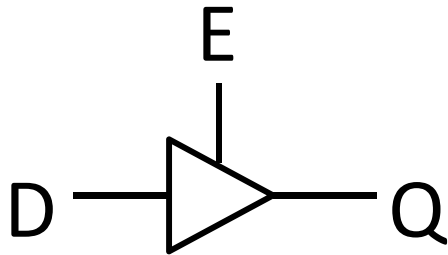
E	D	Q
0	0	z
0	1	z
1	0	0
1	1	1



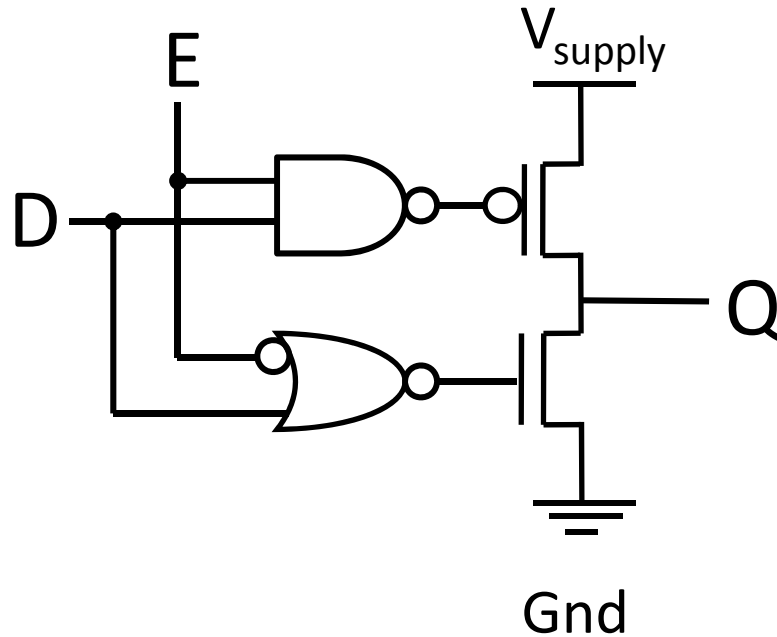
# Tri-State Devices

## Tri-State Buffers

- If enabled ( $E=1$ ), then  $Q = D$
- Otherwise,  $Q$  is not connected ( $z = \text{high impedance}$ )

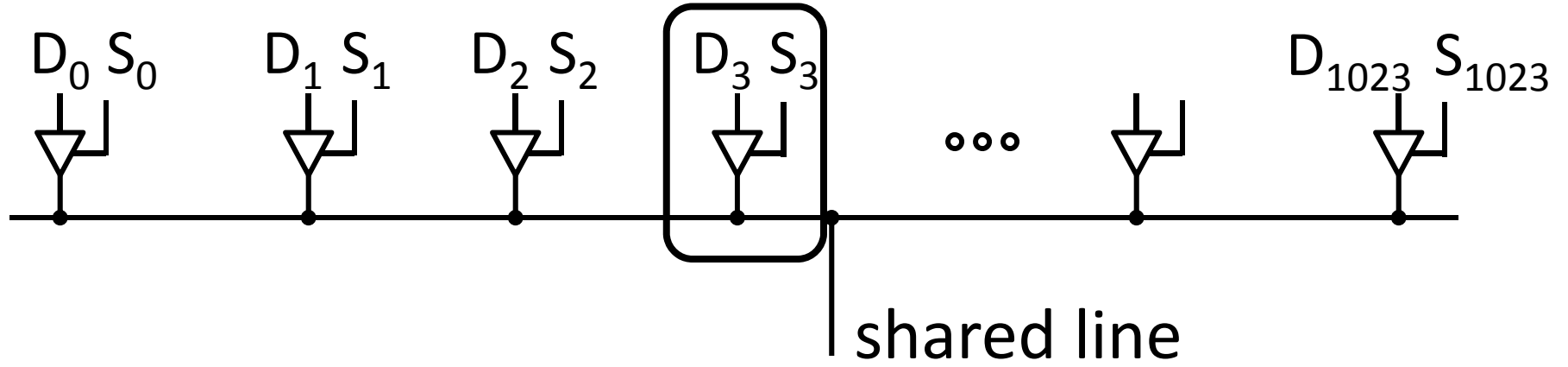


E	D	Q
0	0	z
0	1	z
1	0	0
1	1	1





# Shared Bus



# Takeaway

Register files are very fast storage (only a few gate delays), but does not scale to large memory sizes.

Tri-state Buffers allow scaling since multiple registers can be connected to a single output, while only one register actually drives the output.

# Goals for today

## Memory

- CPU: Register Files (i.e. Memory w/in the CPU)
- Scaling Memory: Tri-state devices
- Cache: SRAM (Static RAM—random access memory)
- Memory: DRAM (Dynamic RAM)

# Next Goal

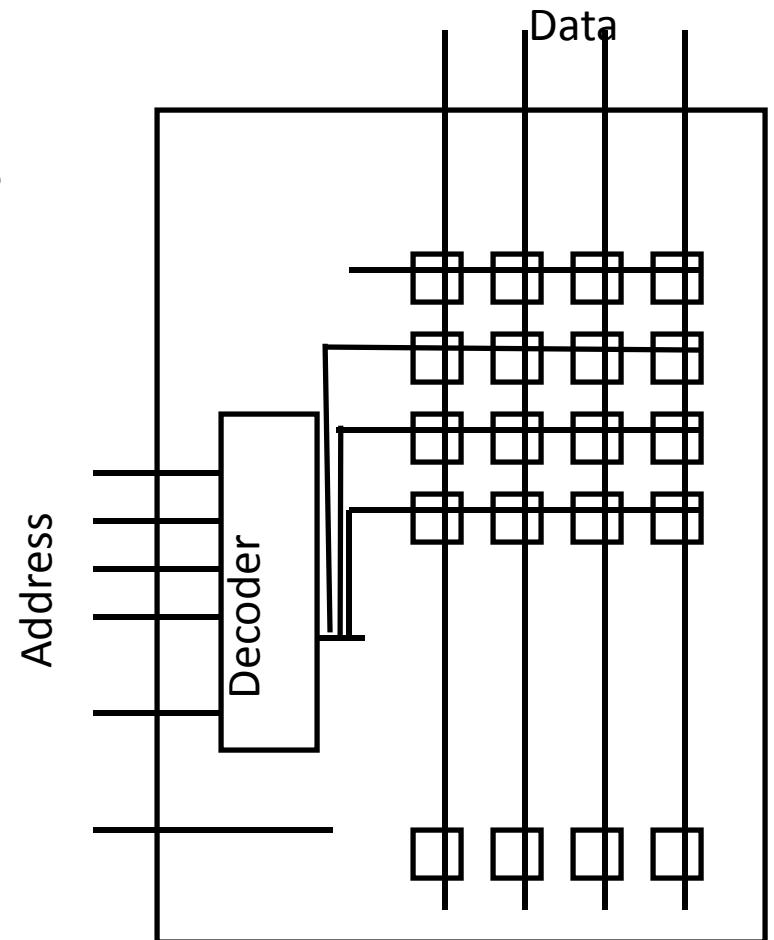
How do we build large memories?

Use similar designs as Tri-state Buffers to connect multiple registers to output line. Only one register will drive output line.

# SRAM

## Static RAM (SRAM)—Static Random Access Memory

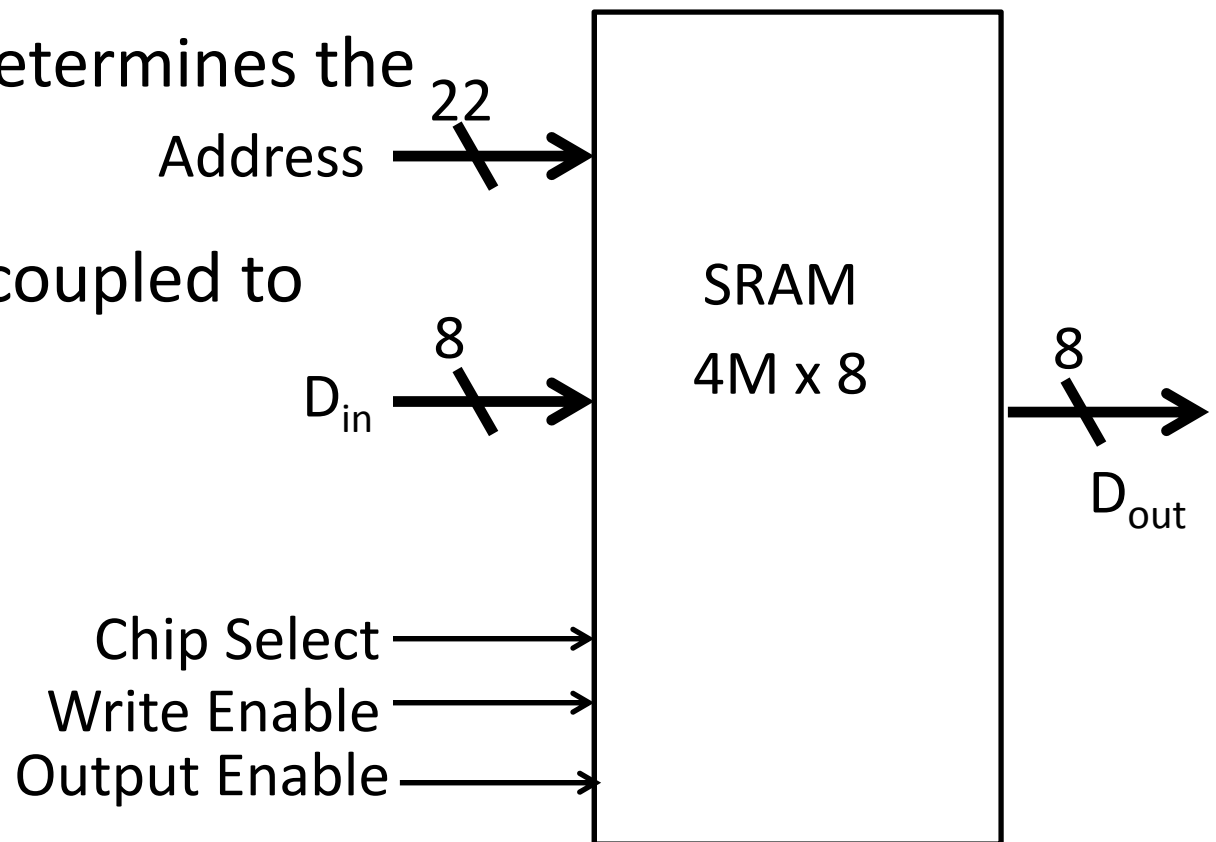
- Essentially just D-Latches plus Tri-State Buffers
- A decoder selects which line of memory to access (i.e. word line)
- A R/W selector determines the type of access
- That line is then coupled to the data lines



# SRAM

## Static RAM (SRAM)—Static Random Access Memory

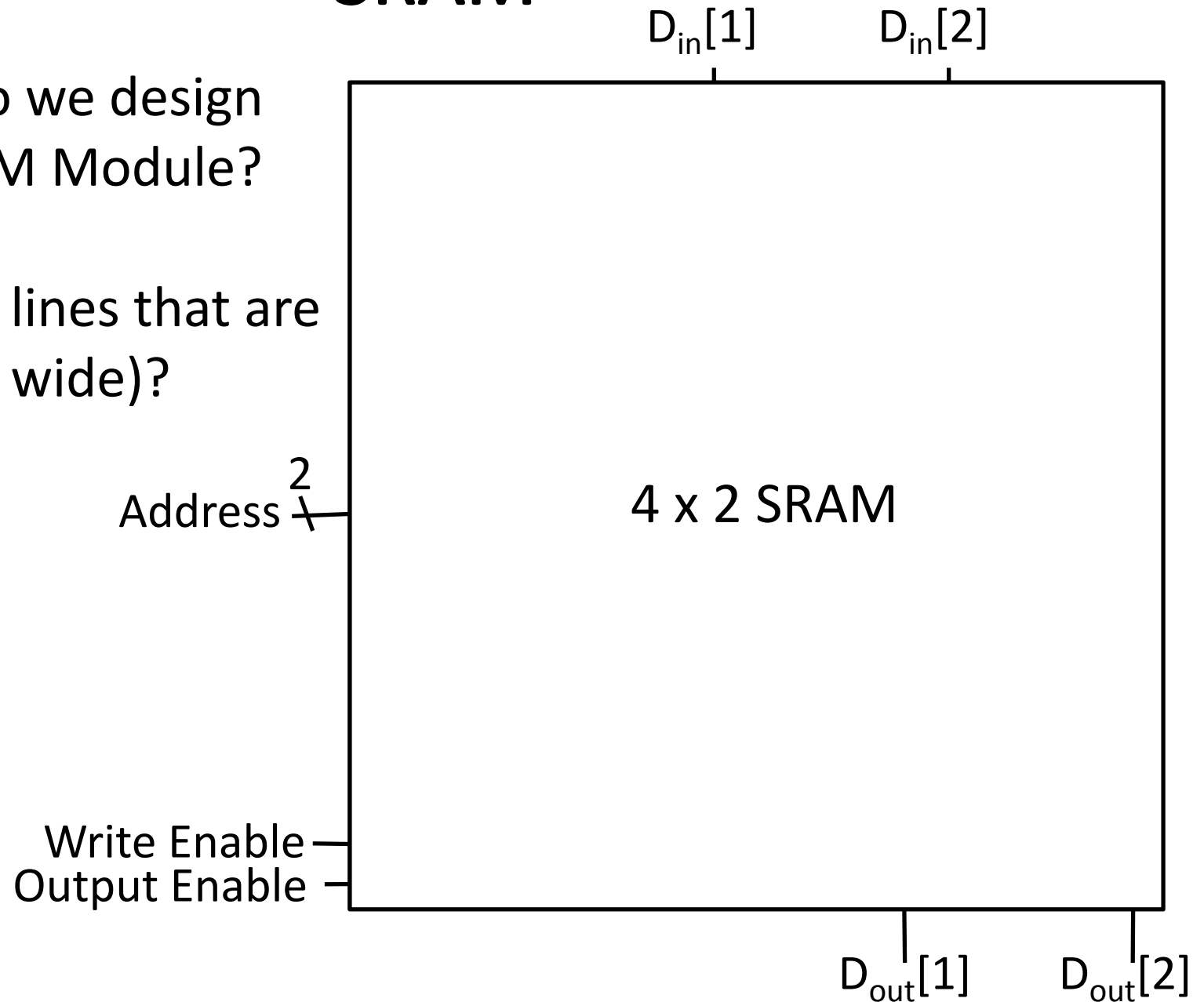
- Essentially just D-Latches plus Tri-State Buffers
- A decoder selects which line of memory to access (i.e. word line)
- A R/W selector determines the type of access
- That line is then coupled to the data lines



# SRAM

E.g. How do we design  
a 4 x 2 SRAM Module?

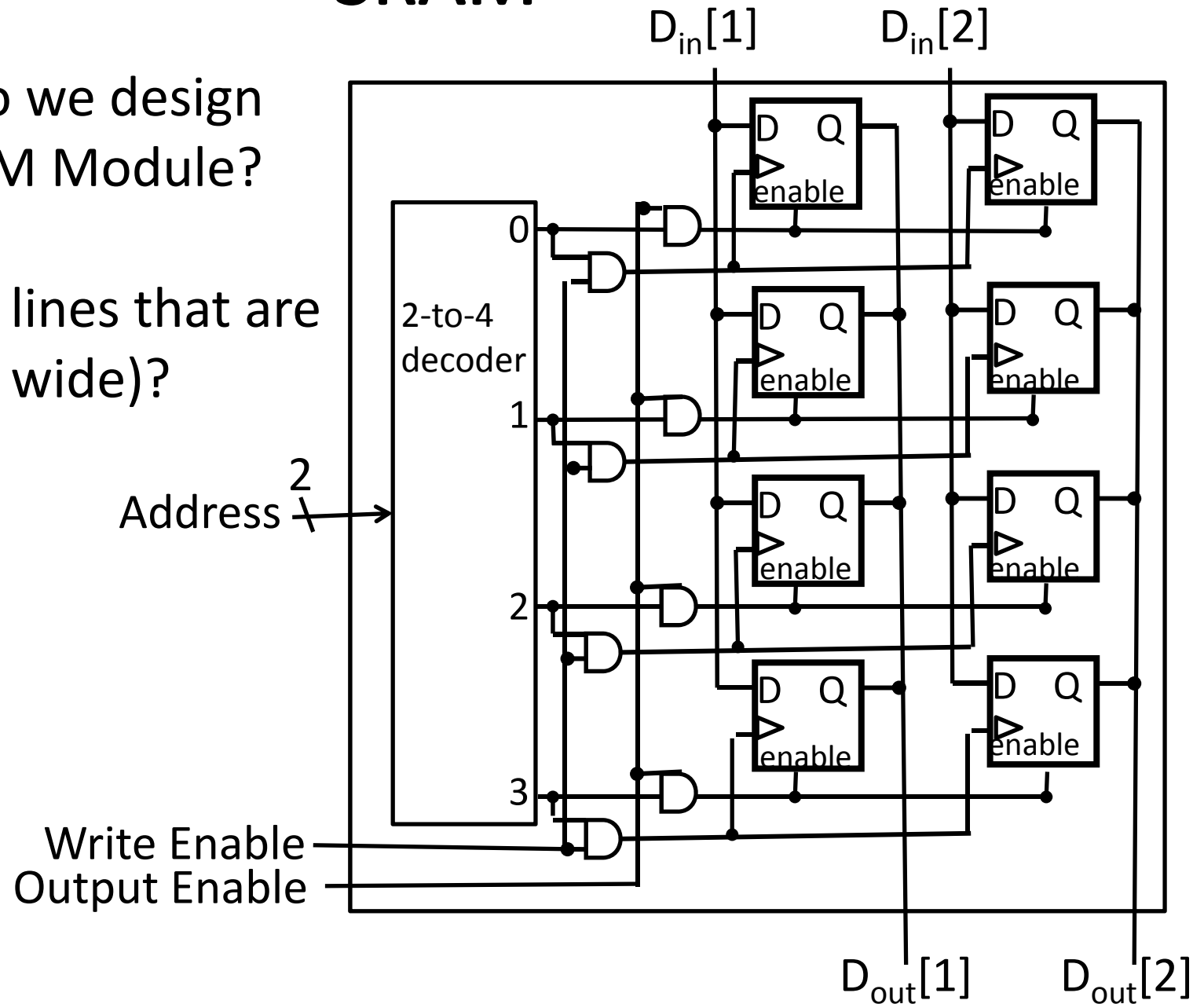
(i.e. 4 word lines that are  
each 2 bits wide)?



# SRAM

E.g. How do we design a 4 x 2 SRAM Module?

(i.e. 4 word lines that are each 2 bits wide)?

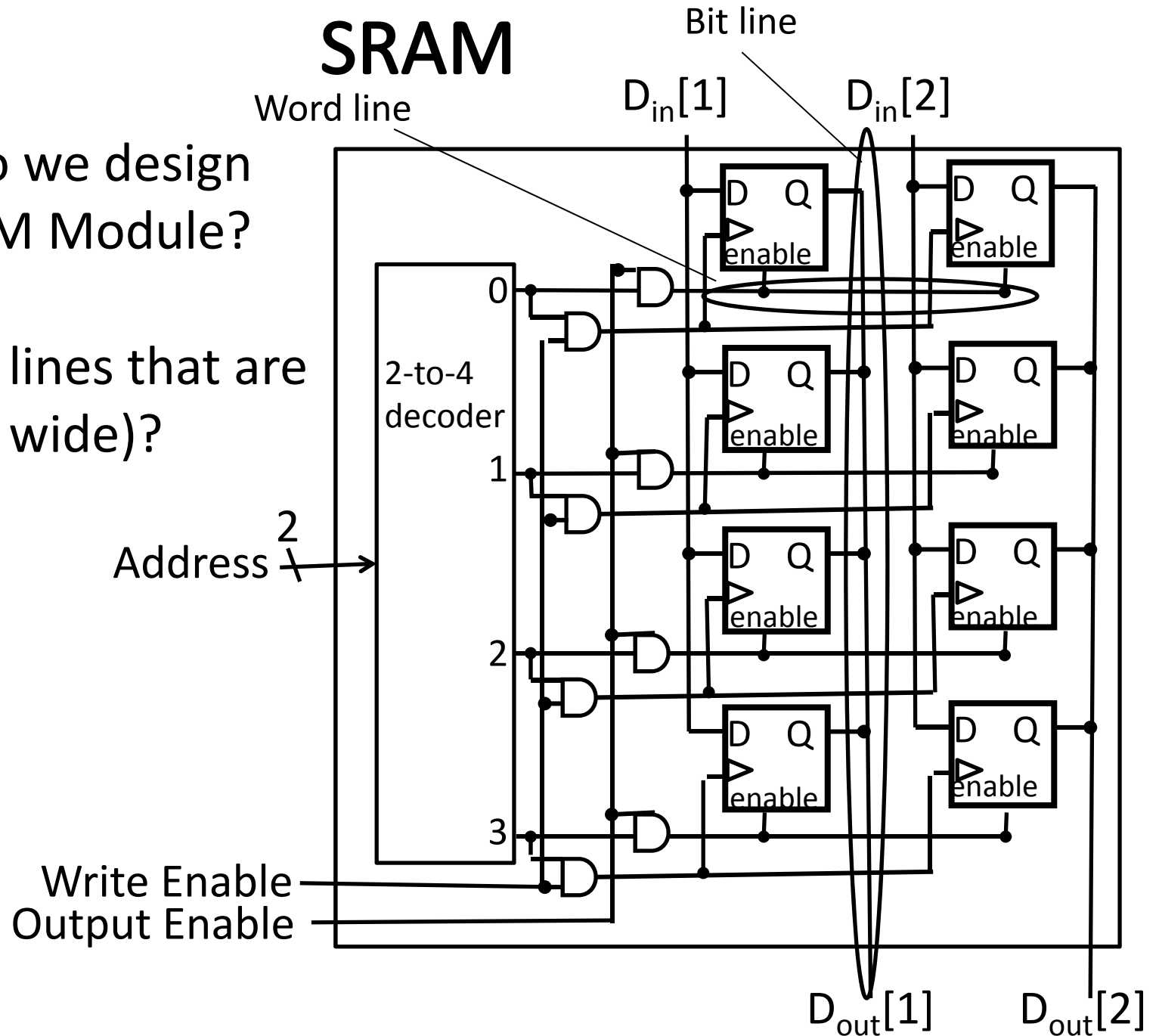




# SRAM

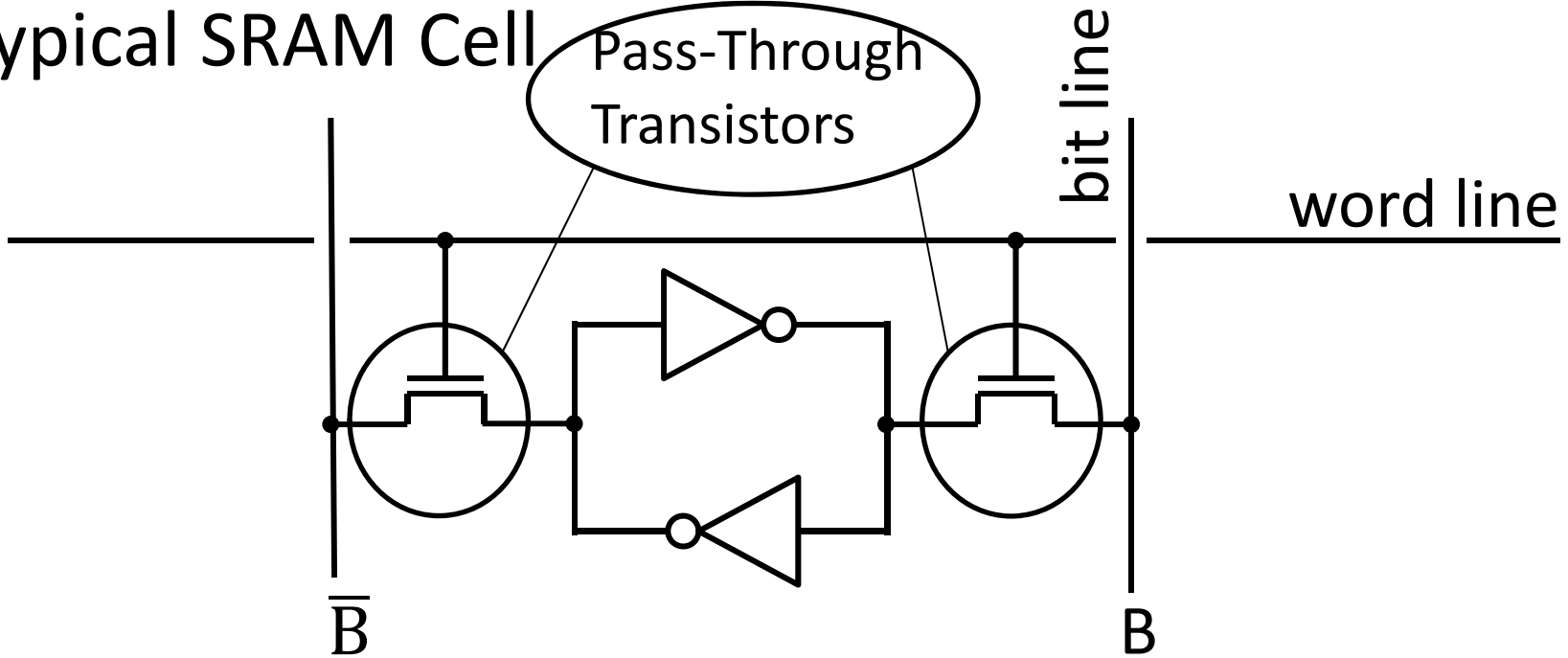
E.g. How do we design a 4 x 2 SRAM Module?

(i.e. 4 word lines that are each 2 bits wide)?



# SRAM Cell

Typical SRAM Cell

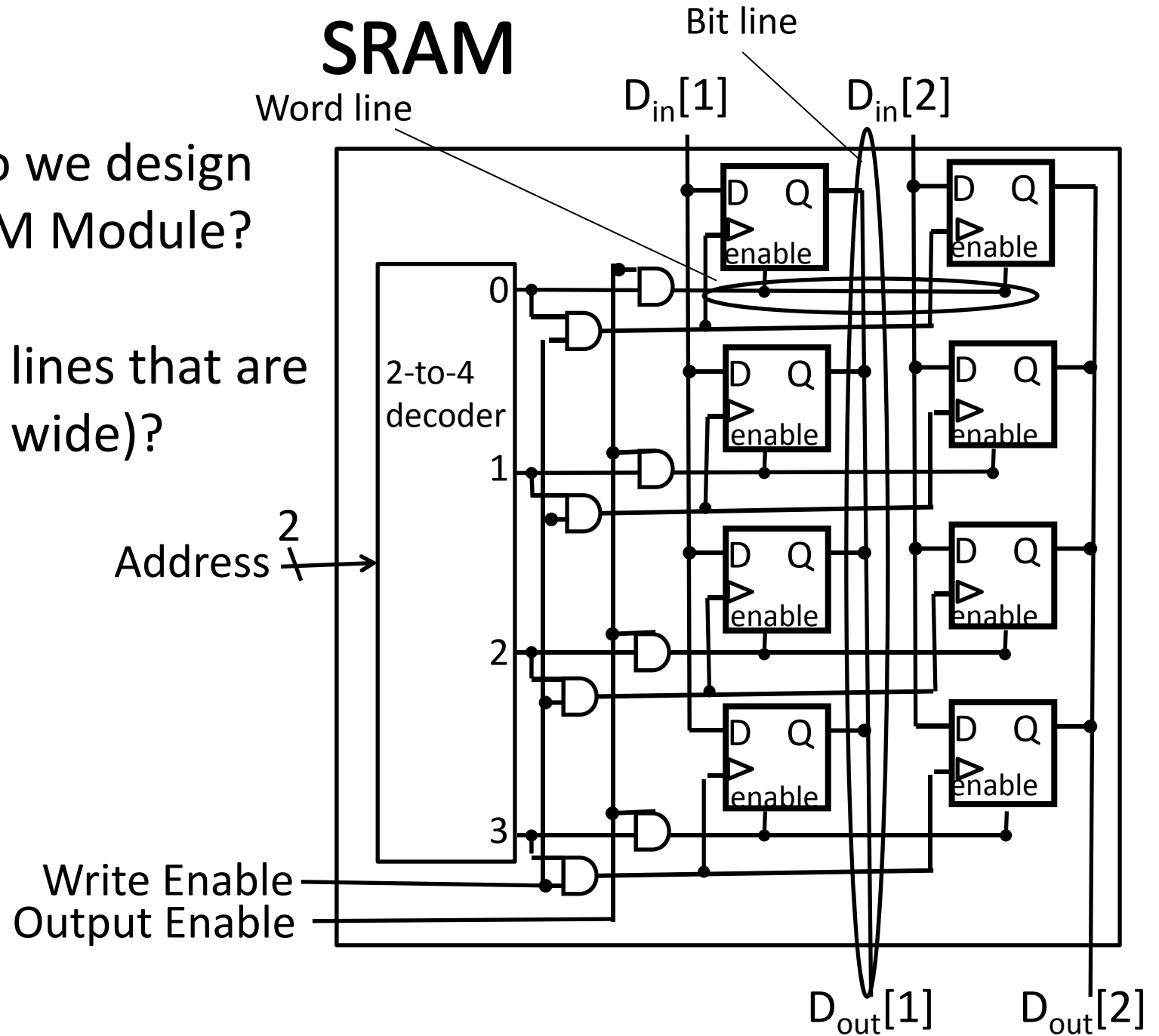


Each cell stores one bit, and requires 4 – 8 transistors (6 is typical)

# SRAM

E.g. How do we design a 4 x 2 SRAM Module?

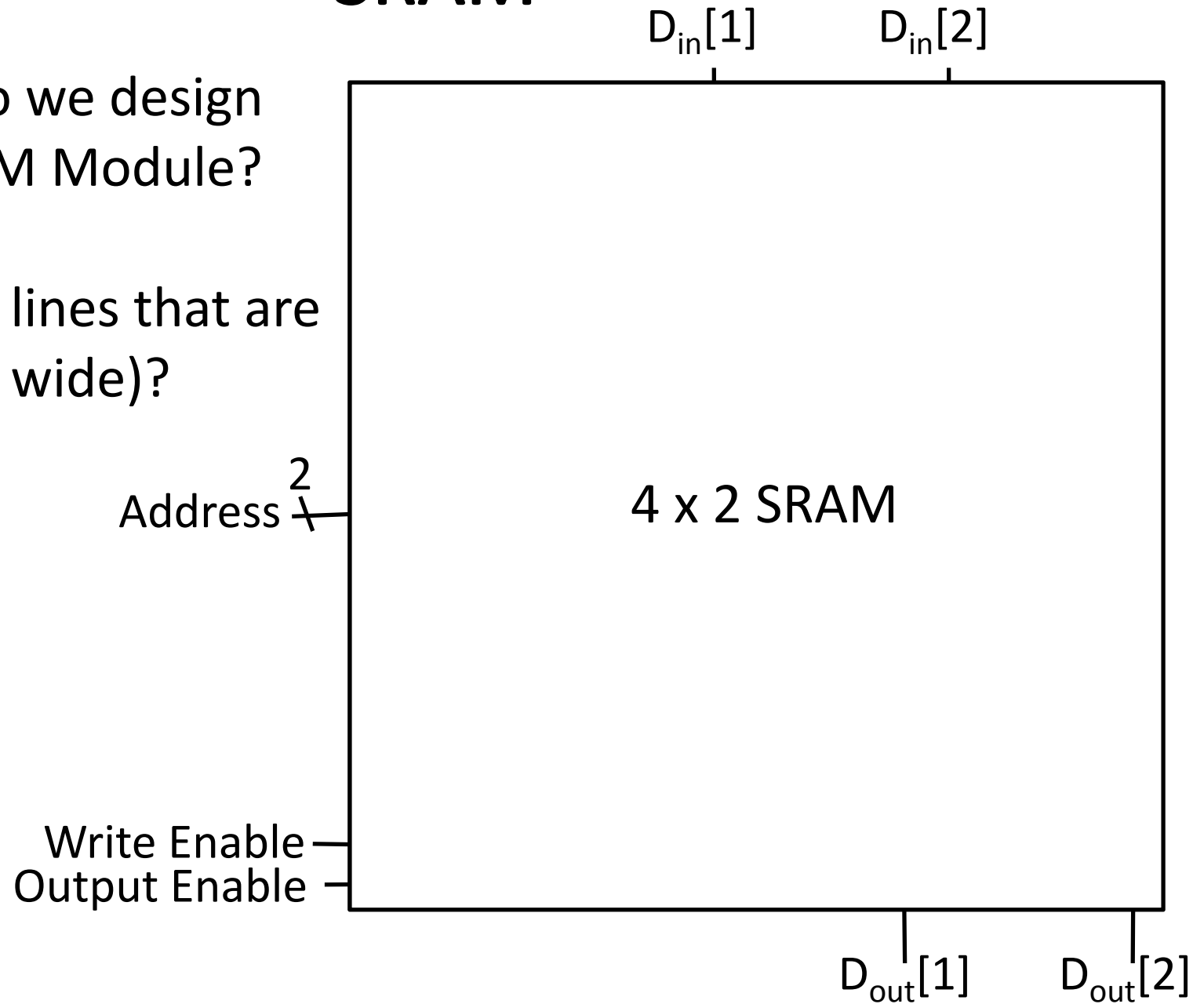
(i.e. 4 word lines that are each 2 bits wide)?



# SRAM

E.g. How do we design  
a 4 x 2 SRAM Module?

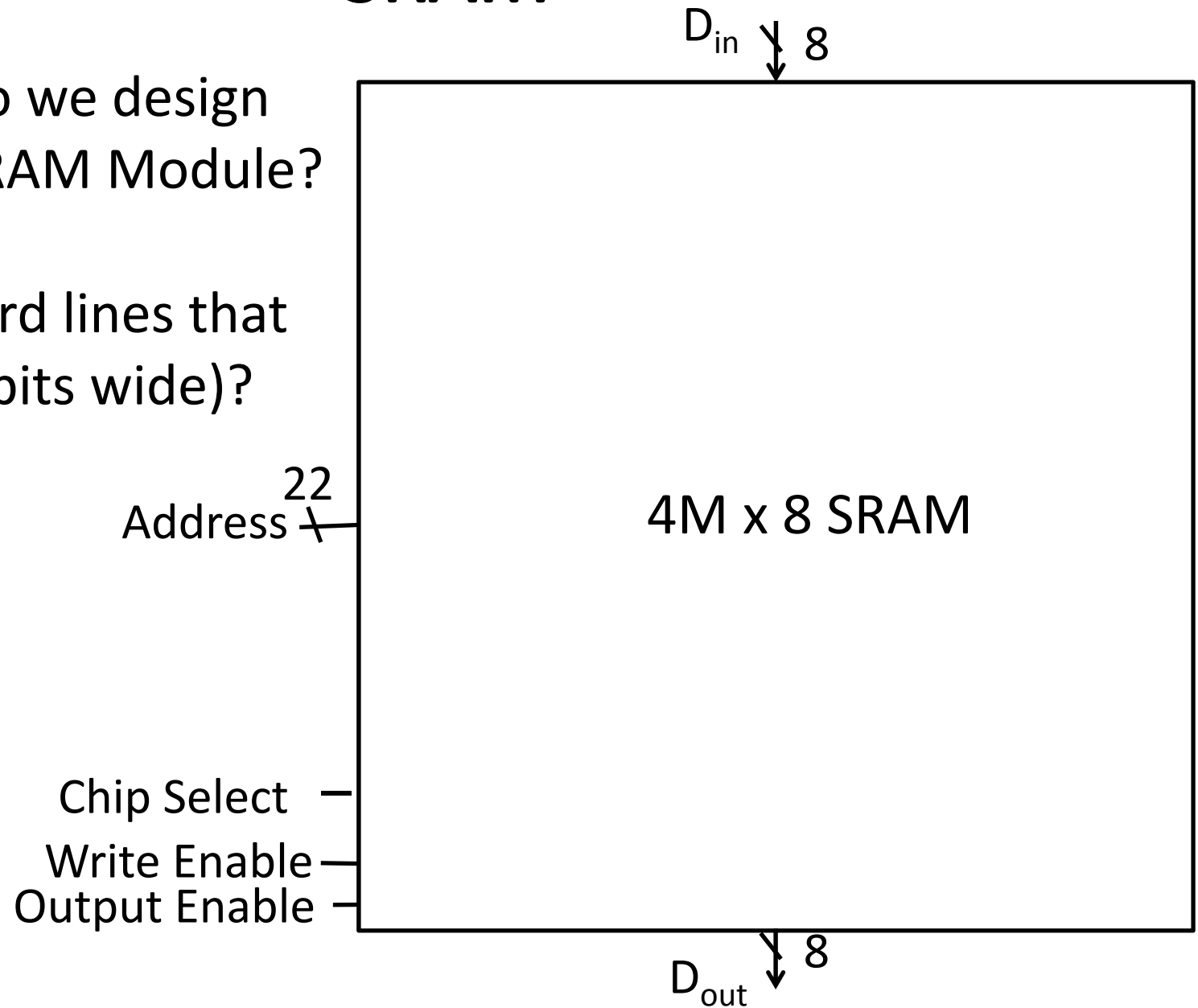
(i.e. 4 word lines that are  
each 2 bits wide)?



# SRAM

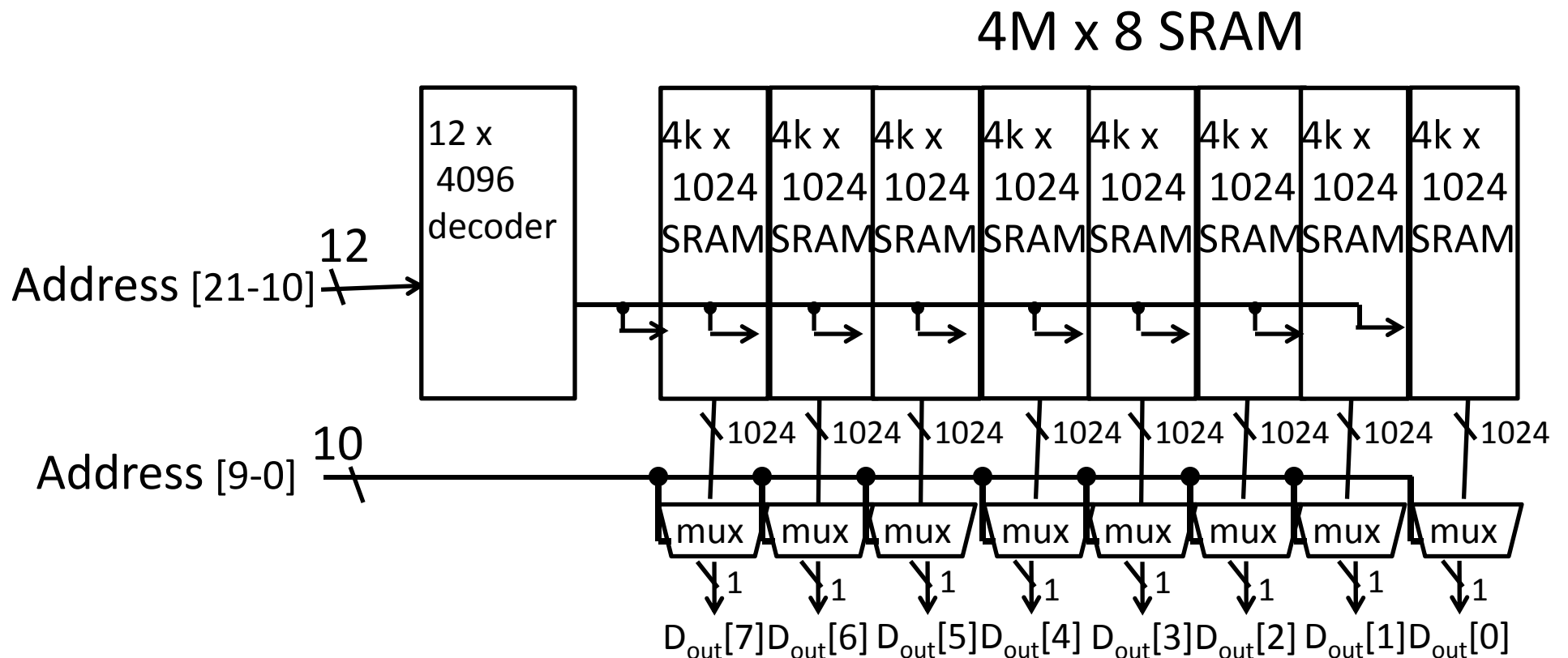
E.g. How do we design a **4M x 8** SRAM Module?

(i.e. 4M word lines that are each 8 bits wide)?



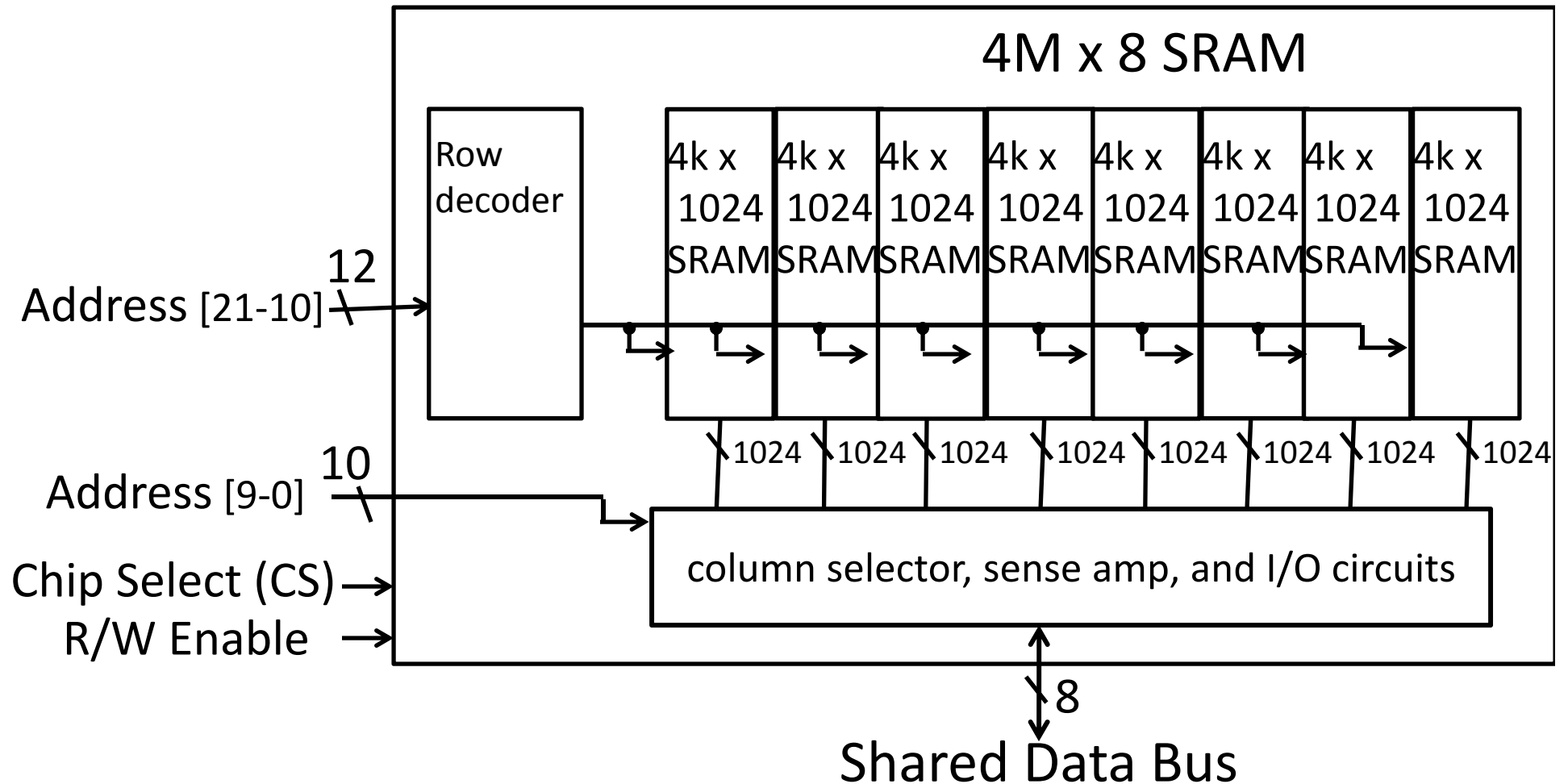
# SRAM

E.g. How do we design  
a **4M x 8** SRAM Module?

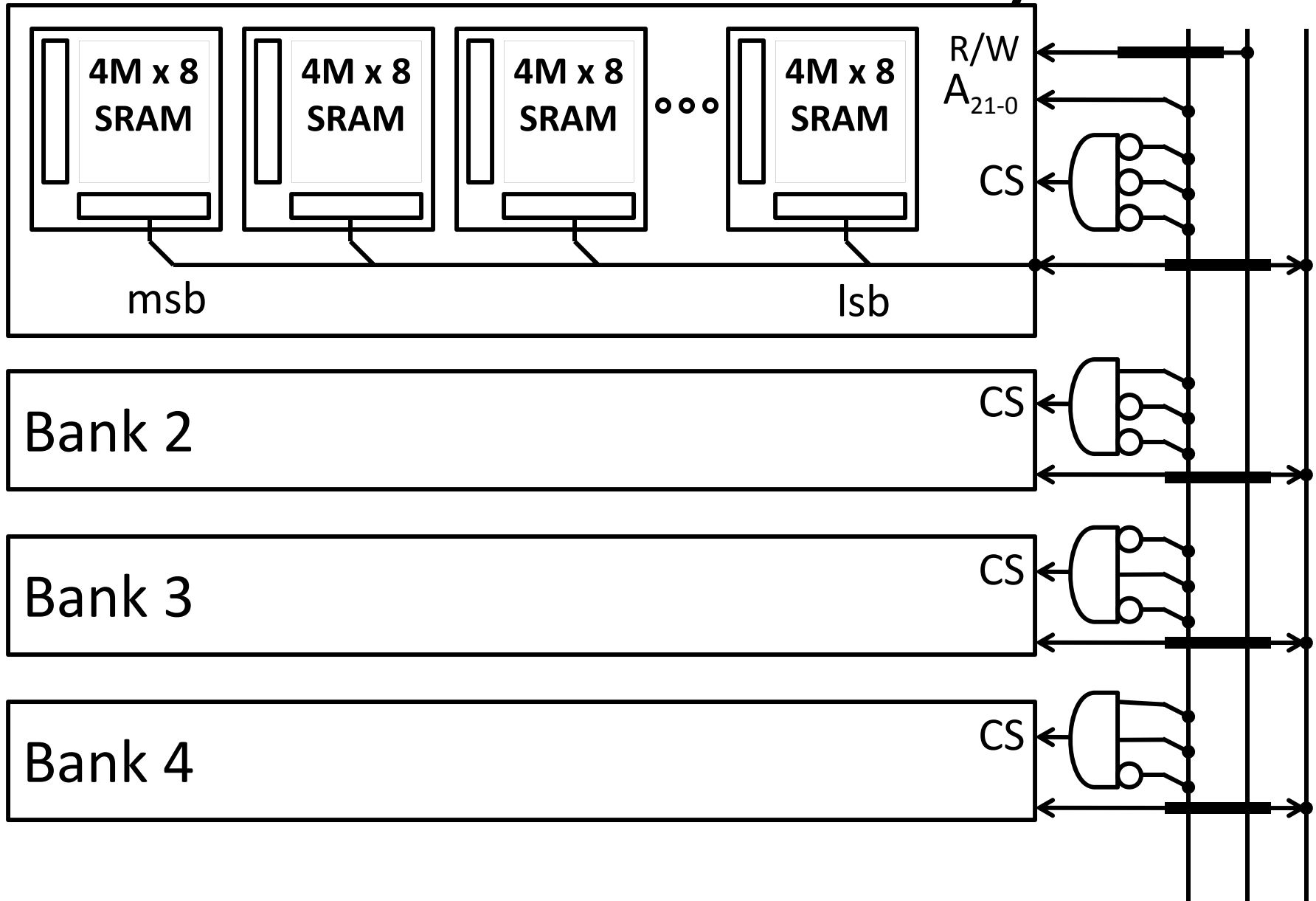


# SRAM

E.g. How do we design  
a **4M x 8** SRAM Module?



# SRAM Modules and Arrays





# SRAM Summary

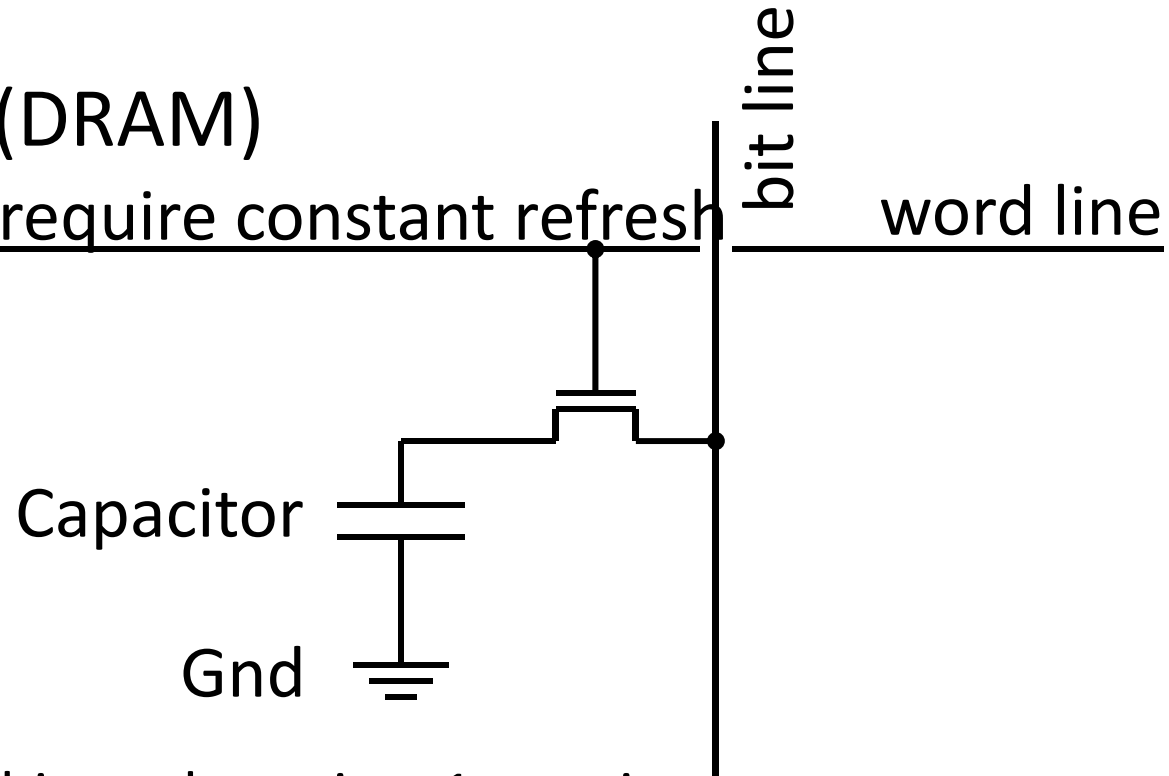
## SRAM

- A few transistors ( $\sim 6$ ) per cell
- Used for working memory (caches)
  
- But for even higher density...

# Dynamic RAM: DRAM

## Dynamic-RAM (DRAM)

- Data values require constant refresh

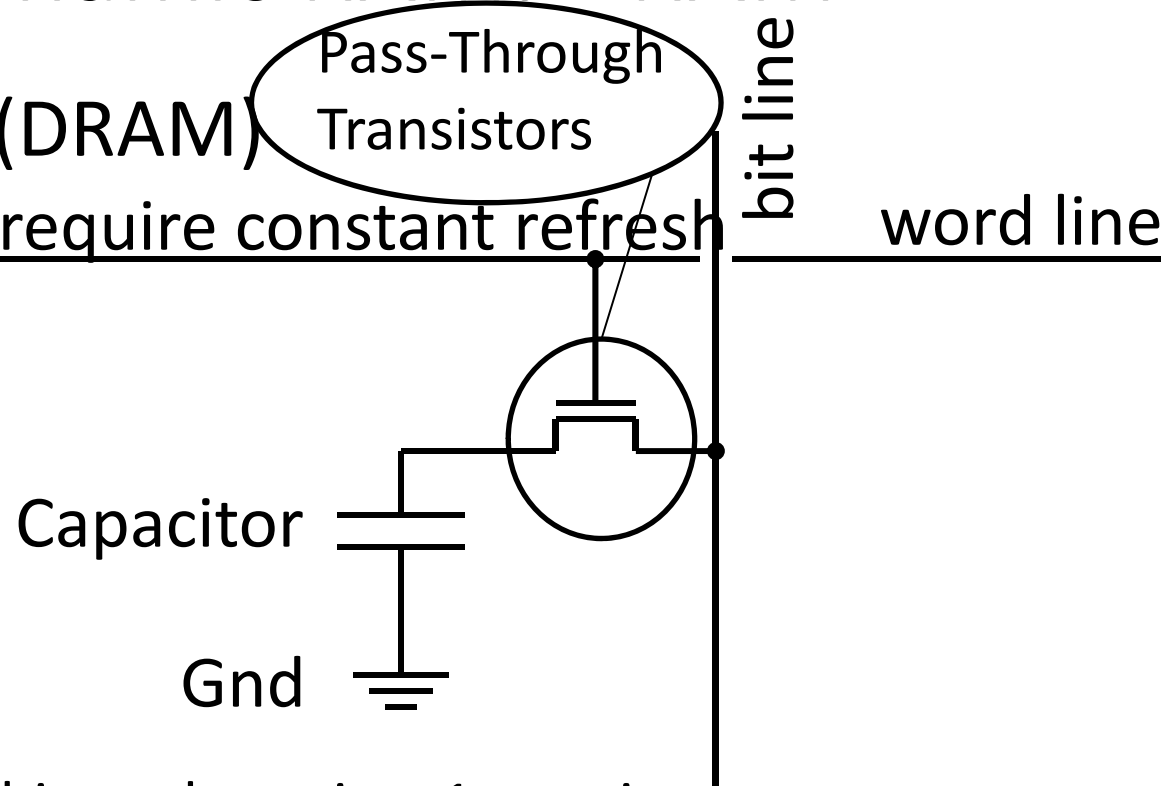


Each cell stores one bit, and requires 1 transistors

# Dynamic RAM: DRAM

Dynamic-RAM (DRAM)

- Data values require constant refresh



Each cell stores one bit, and requires 1 transistors

# DRAM vs. SRAM

Single transistor vs. many gates

- Denser, cheaper (\$30/1GB vs. \$30/2MB)
- But more complicated, and has analog sensing

Also needs refresh

- Read and write back...
- ...every few milliseconds
- Organized in 2D grid, so can do rows at a time
- Chip can do refresh internally

Hence... slower and energy inefficient

# Memory

## Register File tradeoffs

- + Very fast (a few gate delays for both read and write)
- + Adding extra ports is straightforward
- Expensive, doesn't scale
- Volatile

## Volatile Memory alternatives: SRAM, DRAM, ...

- Slower
- + Cheaper, and scales well
- Volatile

## Non-Volatile Memory (NV-RAM): Flash, EEPROM, ...

- + Scales well
- Limited lifetime; degrades after 100000 to 1M writes

# Summary

We now have enough building blocks to build machines that can perform non-trivial computational tasks

Register File: Tens of words of working memory

SRAM: Millions of words of working memory

DRAM: Billions of words of working memory

NVRAM: long term storage

(usb fob, solid state disks, BIOS, ...)

Next time we will build a simple processor!