# Memory

**Prof. Kavita Bala and Prof. Hakim Weatherspoon**

**CS 3410, Spring 2014**

Computer Science

Cornell University

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

# Administrivia

Make sure to go to **_your_** Lab Section this week
Completed Lab1 due **_before_** winter break, Friday, Feb 14th
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 24th
*Work on problems incrementally, as we cover them in lecture*
Office Hours for help
Work **alone**

Work alone, **BUT** use your resources
- Lab Section, Piazza.com, Office Hours
- Class notes, book, Sections, CSUGLab

# Administrivia

Check online syllabus/schedule

- http://www.cs.cornell.edu/Courses/CS3410/2014sp/schedule.html

Slides and Reading for lectures

Office Hours

Homework and Programming Assignments

Prelims (in evenings):

- Tuesday, March 4th
- Thursday, May 1th

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
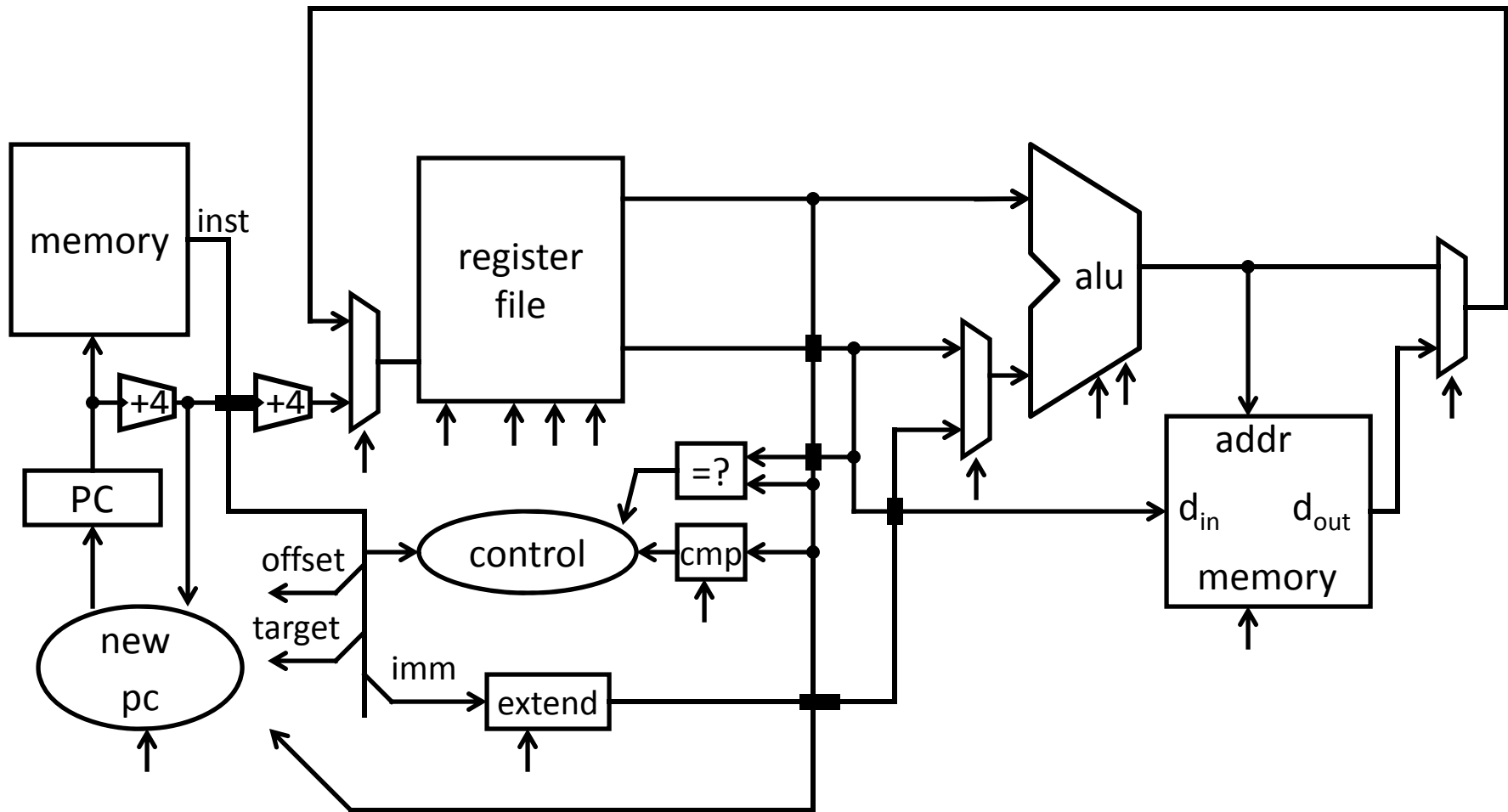


A Single cycle processor

# Goals for today

Review

- Finite State Machines

Memory

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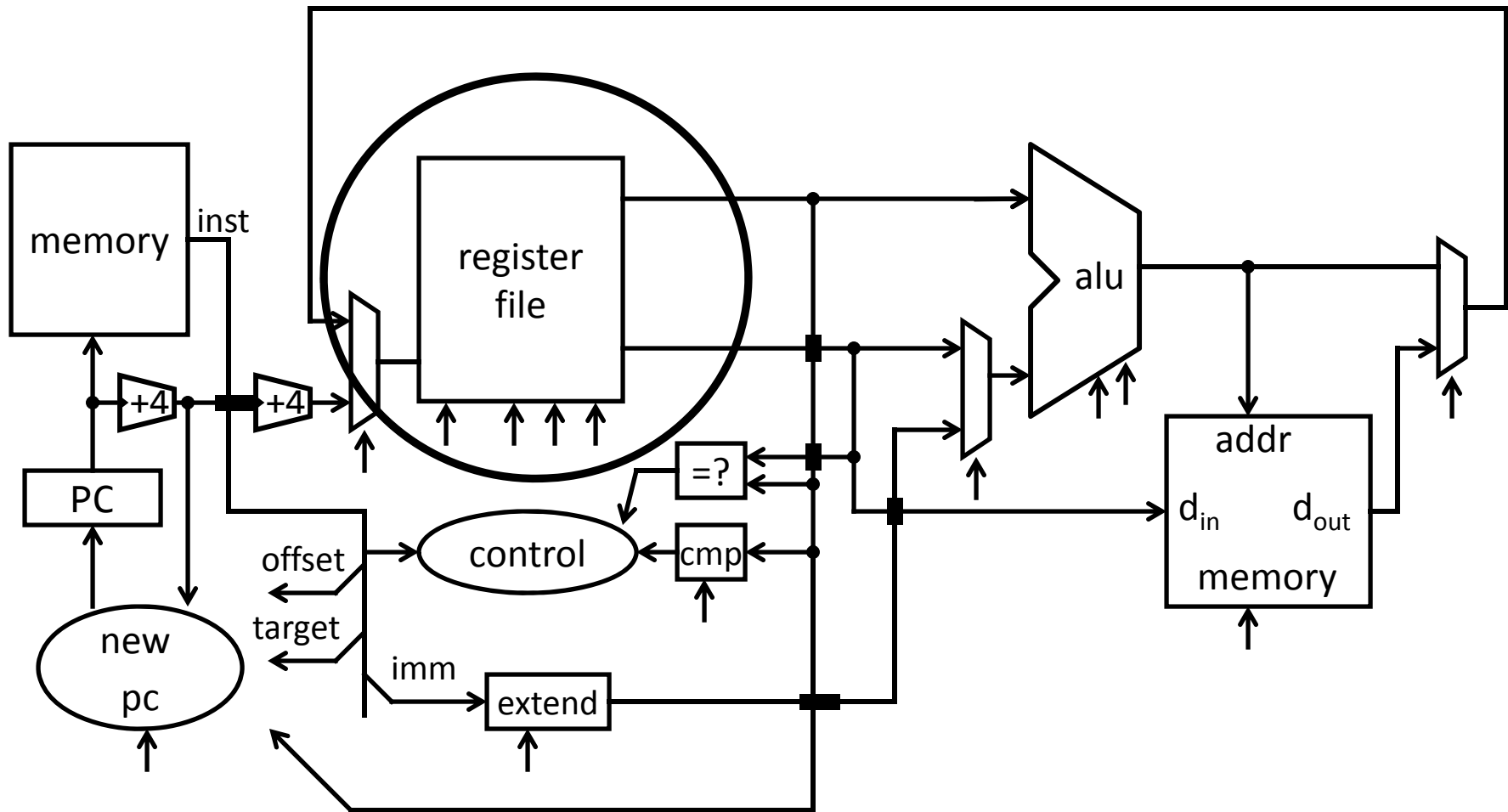- Memory: DRAM (Dynamic RAM)

# Goal:

How do we store results from ALU computations?

How do we use stored results in subsequent operations?

Register File

How does a Register File work? How do we design it?
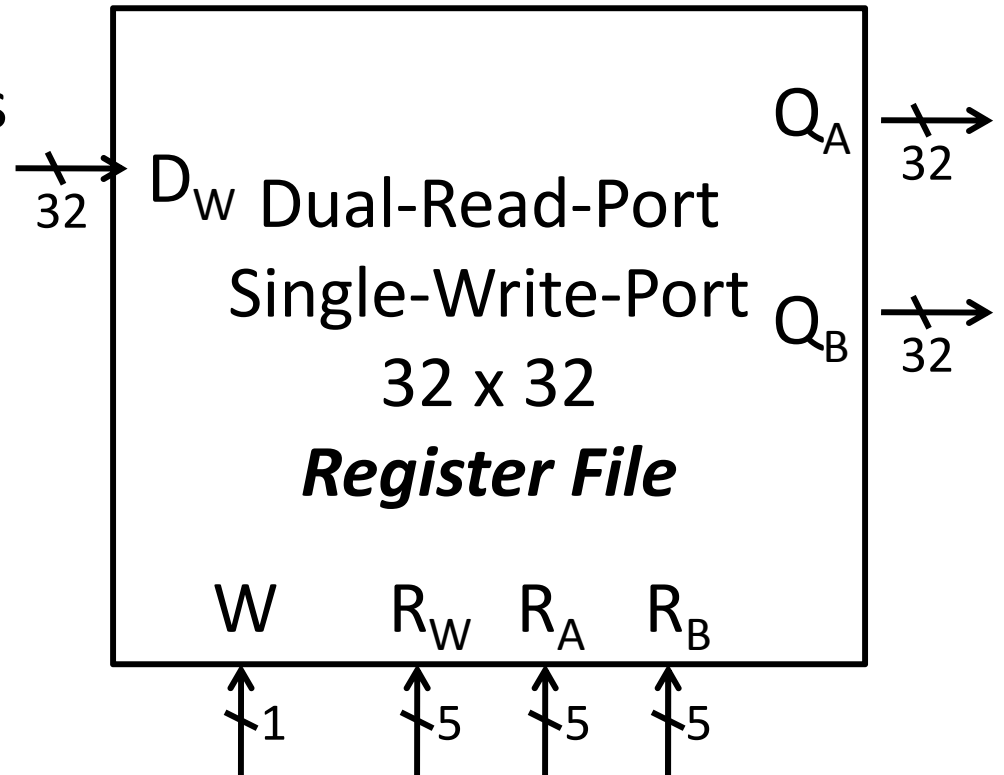
# Big Picture:  Building a Processor



A Single cycle processor

# Register File

## Register File

- N read/write registers
- Indexed by register number

$D_W$ Dual-Read-Port Single-Write-Port 32 x 32 **Register File**

$Q_A$ — 32

$Q_B$ — 32

32 →

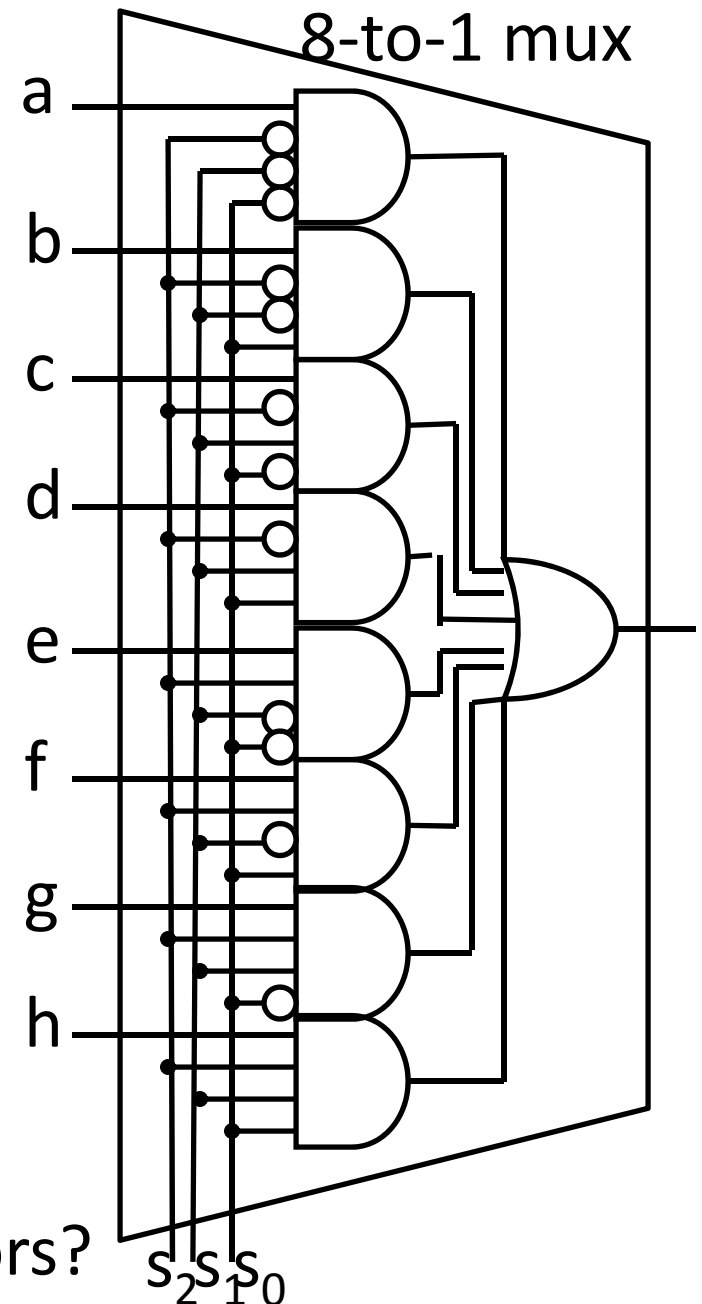W    $R_W$    $R_A$    $R_B$

1      5      5      5

# 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?

8-to-1 mux

a

b

c

d

e

f

g

h

$s_2 s_1 s_0$

# Takeway

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)
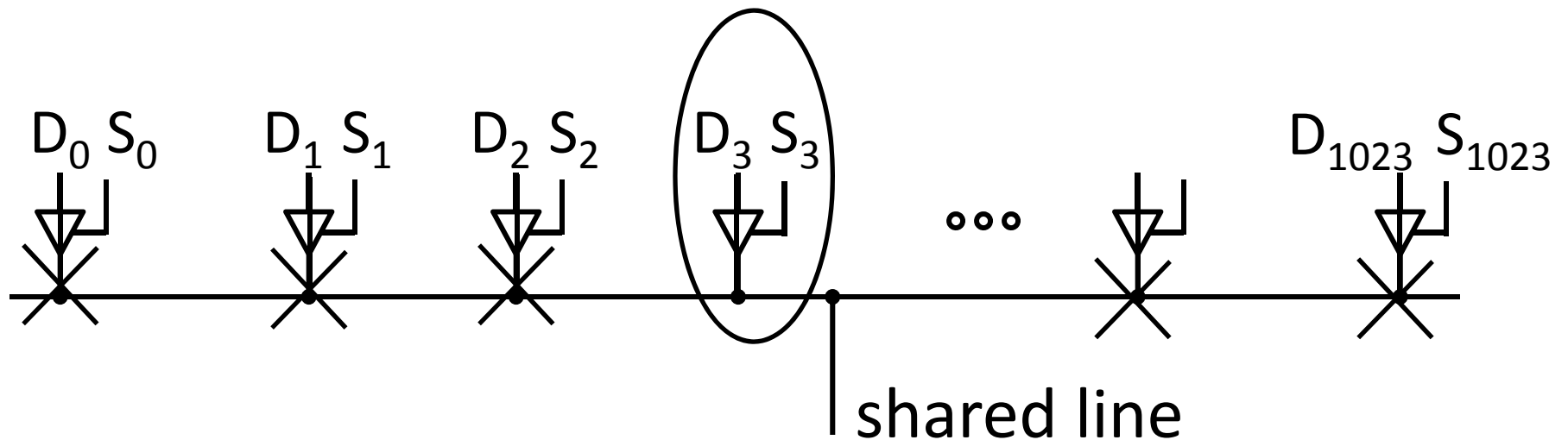- Memory: DRAM (Dynamic RAM)

# 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
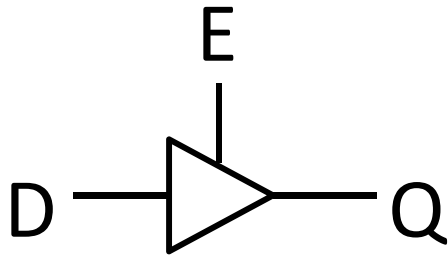- Only one output *drives* the bus at a time



shared line

- 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 = 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 = high impedance)

E

D — ▷ — Q

| E | D | Q |
|---|---|---|
| 0 | 0 | z |
| 0 | 1 | z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$V_{supply}$

D — Q

Gnd

# Tri-State Devices

Tri-State Buffers

- If enabled (E=1), then Q = D
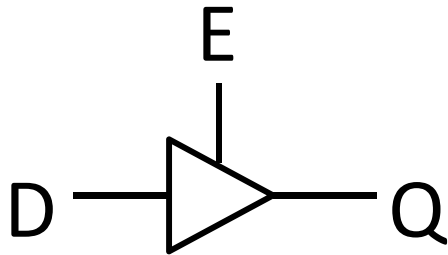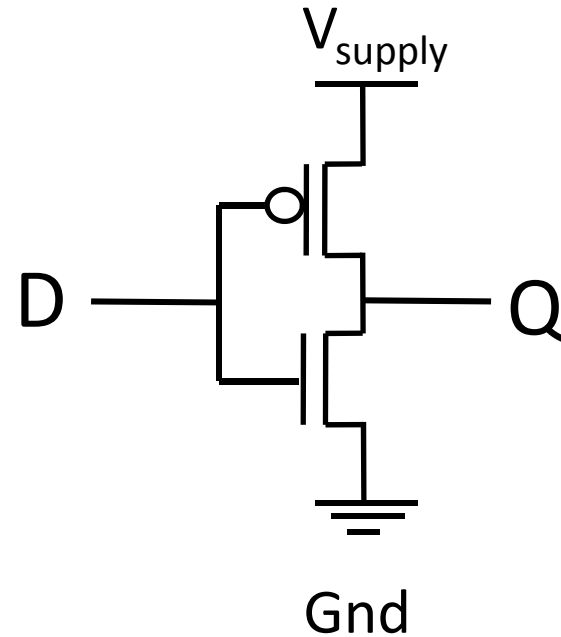- Otherwise, Q is not connected (z = high impedance)



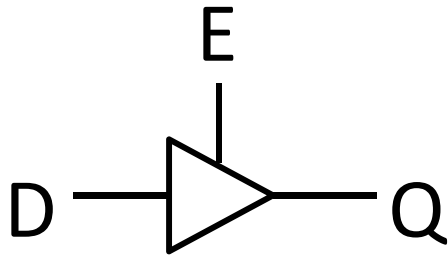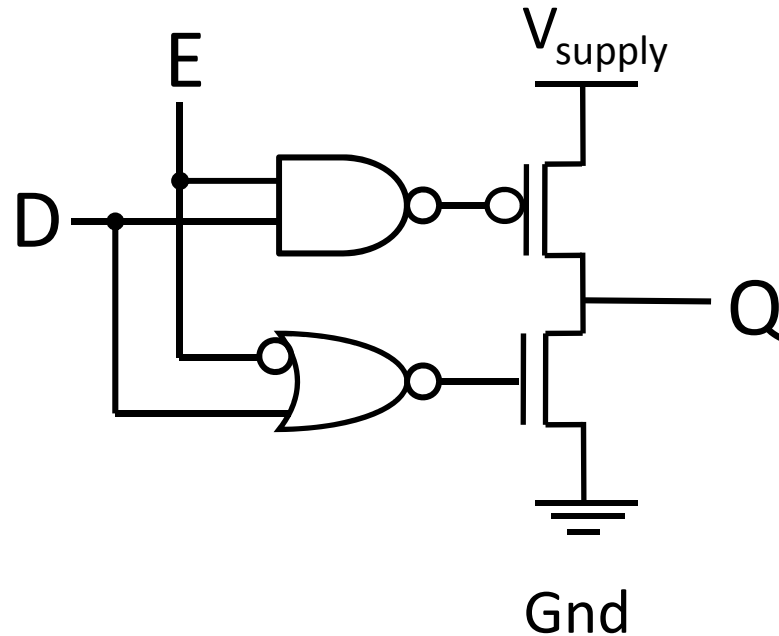| E | D | Q |
|---|---|---|
| 0 | 0 | z |
| 0 | 1 | z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# Shared Bus
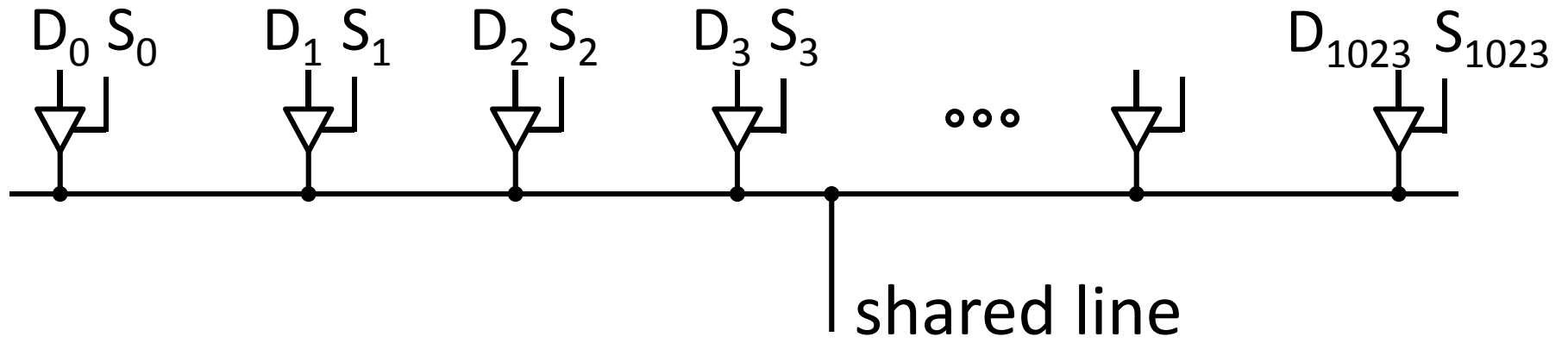
$D_0$ $S_0$     $D_1$ $S_1$     $D_2$ $S_2$     $D_3$ $S_3$     $D_{1023}$ $S_{1023}$

∘∘∘

shared line

# Takeway

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)
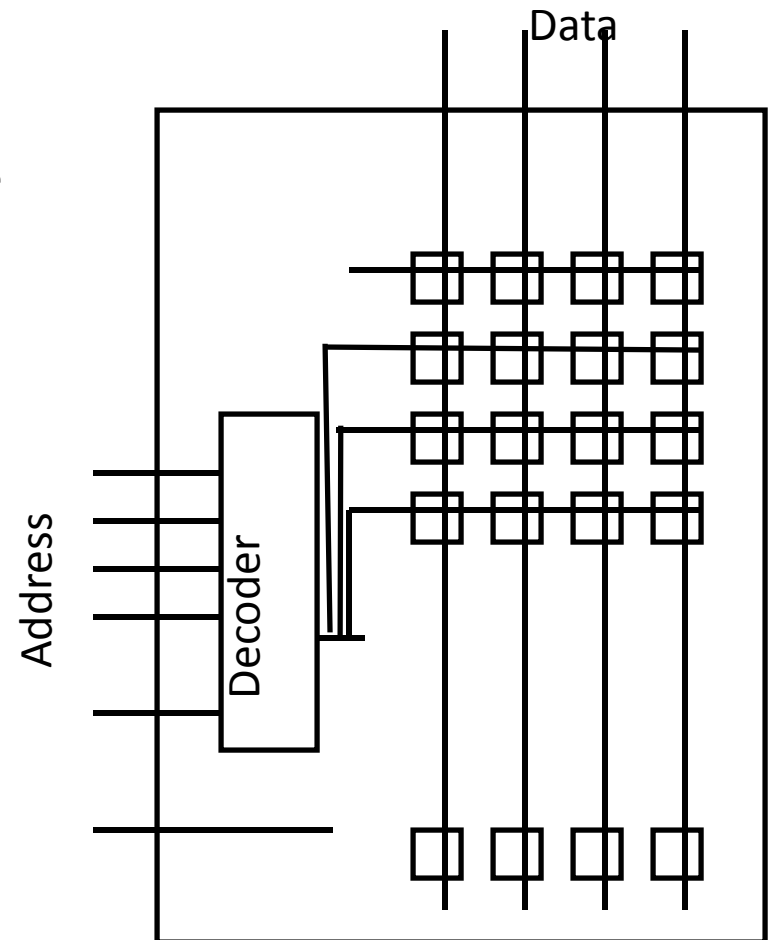- Memory: DRAM (Dynamic RAM)

# 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

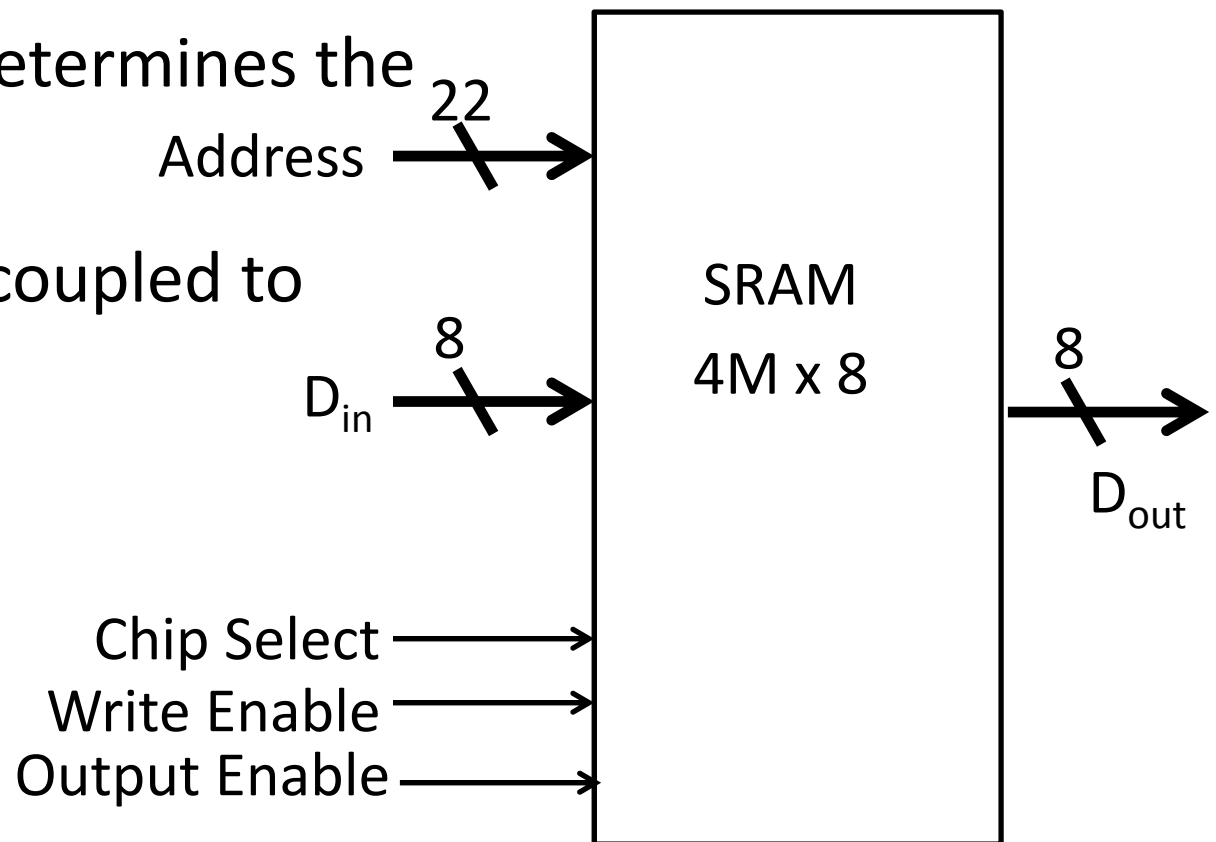Static RAM (SRAM)—Static Random Access Memory

- 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

Address $\xrightarrow{22}$

SRAM
4M x 8

$D_{in}$ $\xrightarrow{8}$

$\xrightarrow{8}$ $D_{out}$

Chip Select →
Write Enable →
Output Enable →

# SRAM

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

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

$D_{in}[1]$   $D_{in}[2]$

Address — $2$

4 x 2 SRAM

Write Enable
Output Enable

$D_{out}[1]$   $D_{out}[2]$

# 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

Bit line

Word line

$D_{in}[1]$  $D_{in}[2]$

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

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

2-to-4 decoder

Address $\overset{2}{/}$

Write Enable
Output Enable

$D_{out}[1]$  $D_{out}[2]$

# SRAM

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

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

$D_{in}[1]$  $D_{in}[2]$

Address $\underset{2}{\not}$

4 x 2 SRAM

Write Enable
Output Enable

$D_{out}[1]$  $D_{out}[2]$
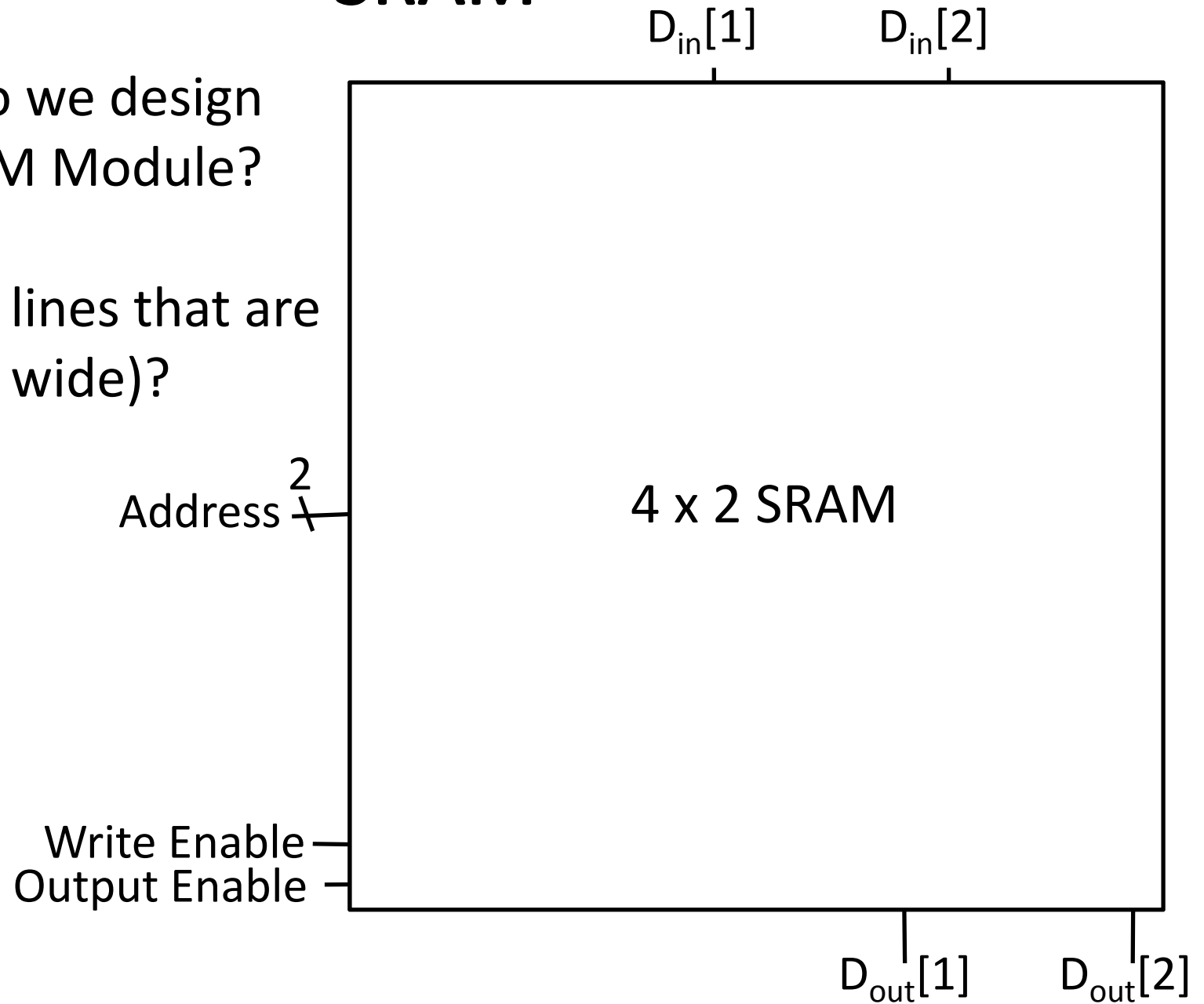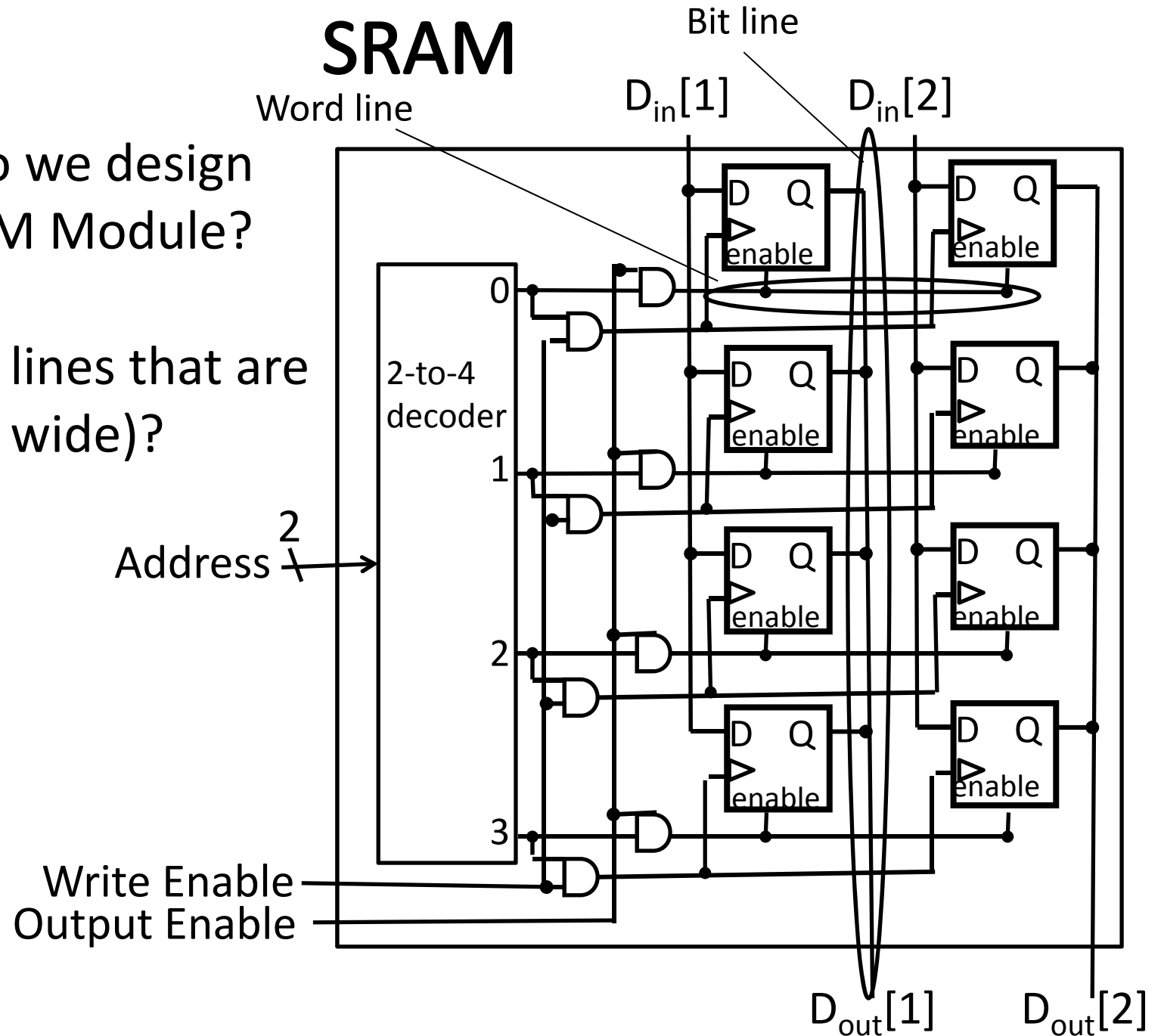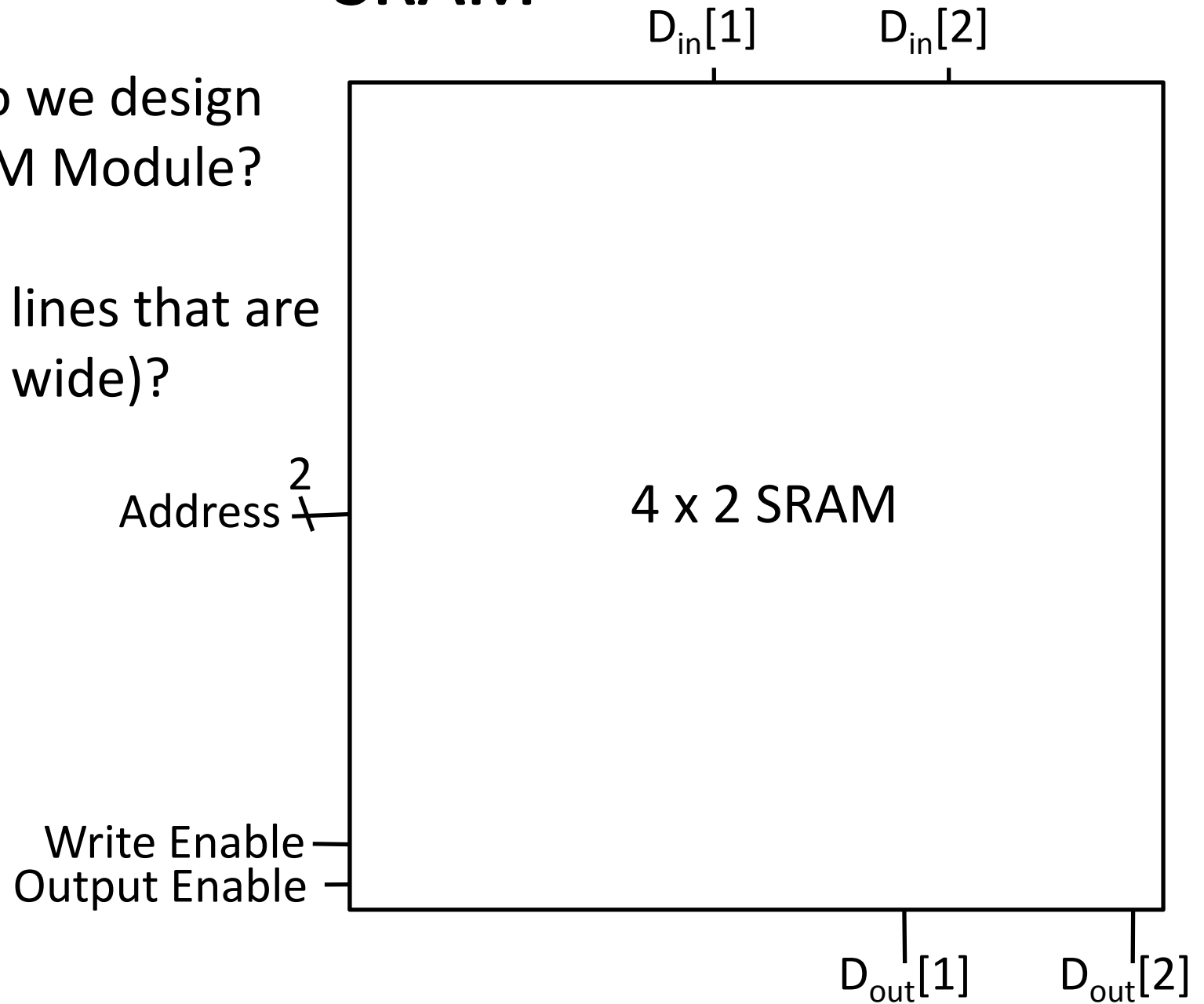
# SRAM

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

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

$D_{in}$ ↓ 8

Address ⟋ 22

4M x 8 SRAM

Chip Select —
Write Enable —
Output Enable —

$D_{out}$ ↓ 8

# SRAM

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

4M x 8 SRAM

| 12 x 4096 decoder | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM |

Address [21-10]  12

Address [9-0]  10

1024 1024 1024 1024 1024 1024 1024 1024

mux  mux  mux  mux  mux  mux  mux  mux

1 1 1 1 1 1 1 1

$D_{out}[7]$ $D_{out}[6]$ $D_{out}[5]$ $D_{out}[4]$ $D_{out}[3]$ $D_{out}[2]$ $D_{out}[1]$ $D_{out}[0]$

# SRAM

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



4M x 8 SRAM

Row decoder

4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM | 4k x 1024 SRAM

12

Address [21-10]

1024  1024  1024  1024  1024  1024  1024  1024

10

Address [9-0]

Chip Select (CS)

R/W Enable

column selector, sense amp, and I/O circuits

8

Shared Data Bus

# SRAM Modules and Arrays



4M x 8 SRAM    4M x 8 SRAM    4M x 8 SRAM  ° ° °  4M x 8 SRAM

R/W
A$_{21-0}$

CS

msb                                          lsb

Bank 2        CS

Bank 3        CS

Bank 4        CS
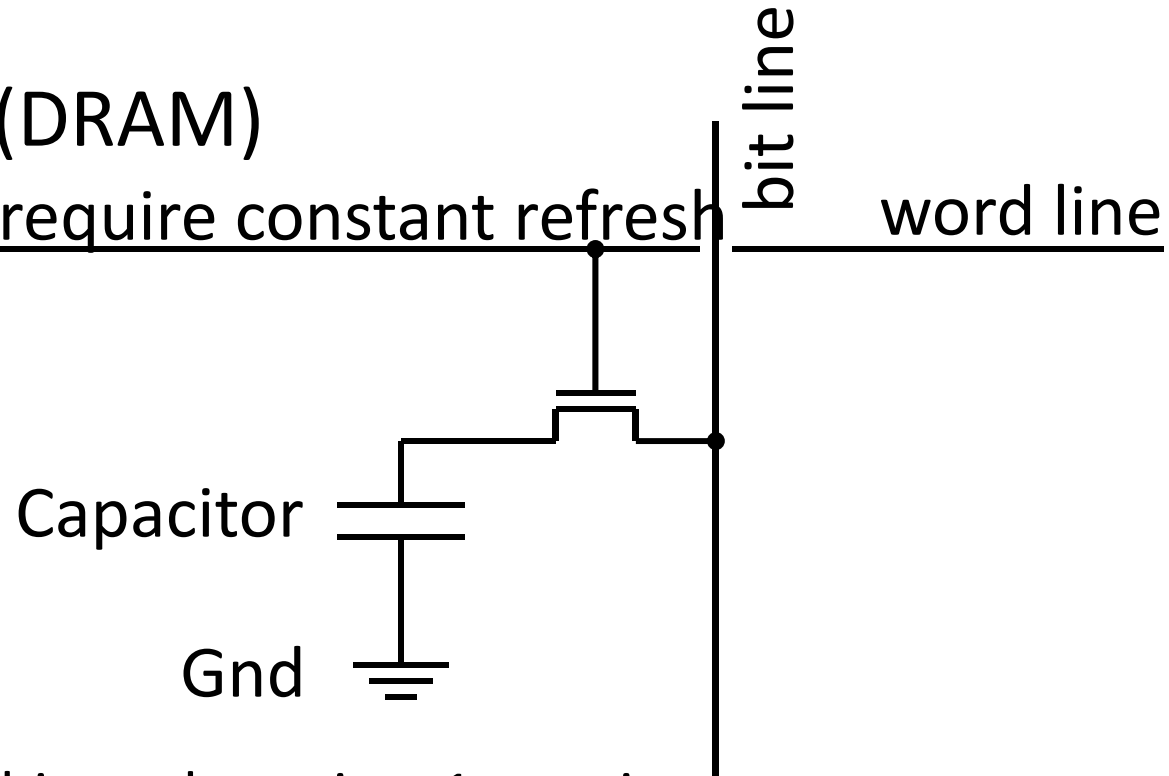
# SRAM Summary

SRAM

- A few transistors (~6) per cell

- Used for working memory (caches)

- But for even higher density…

# Dynamic RAM: DRAM

## Dynamic-RAM (DRAM)

- <u>Data values require constant refresh</u>

bit line

word line

Capacitor

Gnd

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!