Memory

Anne Bracy

CS 3410

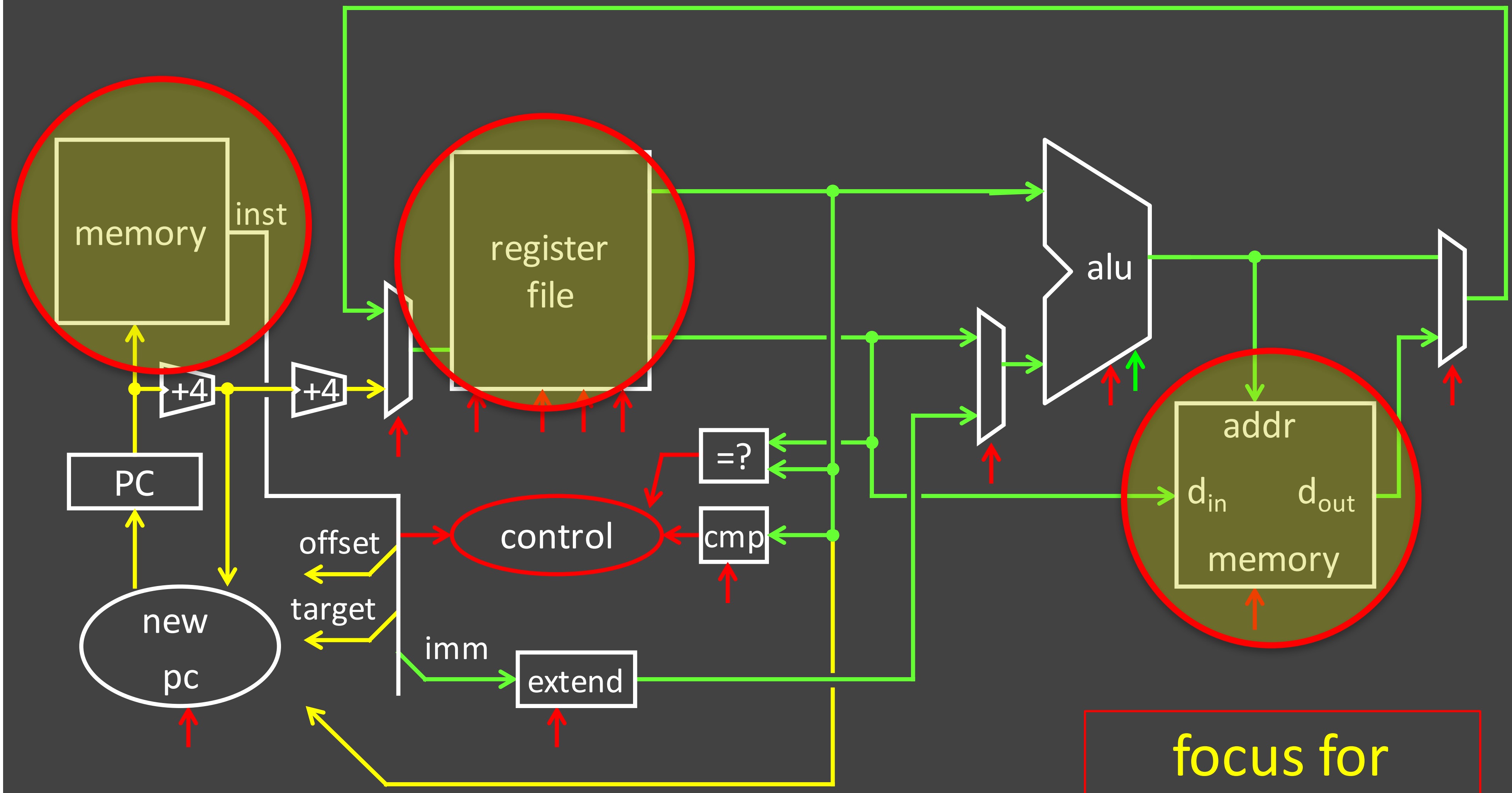
Computer Science

Cornell University

The slides are the product of many rounds of teaching CS 3410 by Professors Weatherspoon, Bala, Bracy, and Sirer.

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

Big Picture: Building a Processor



focus for today

A Single cycle processor

Goals for today

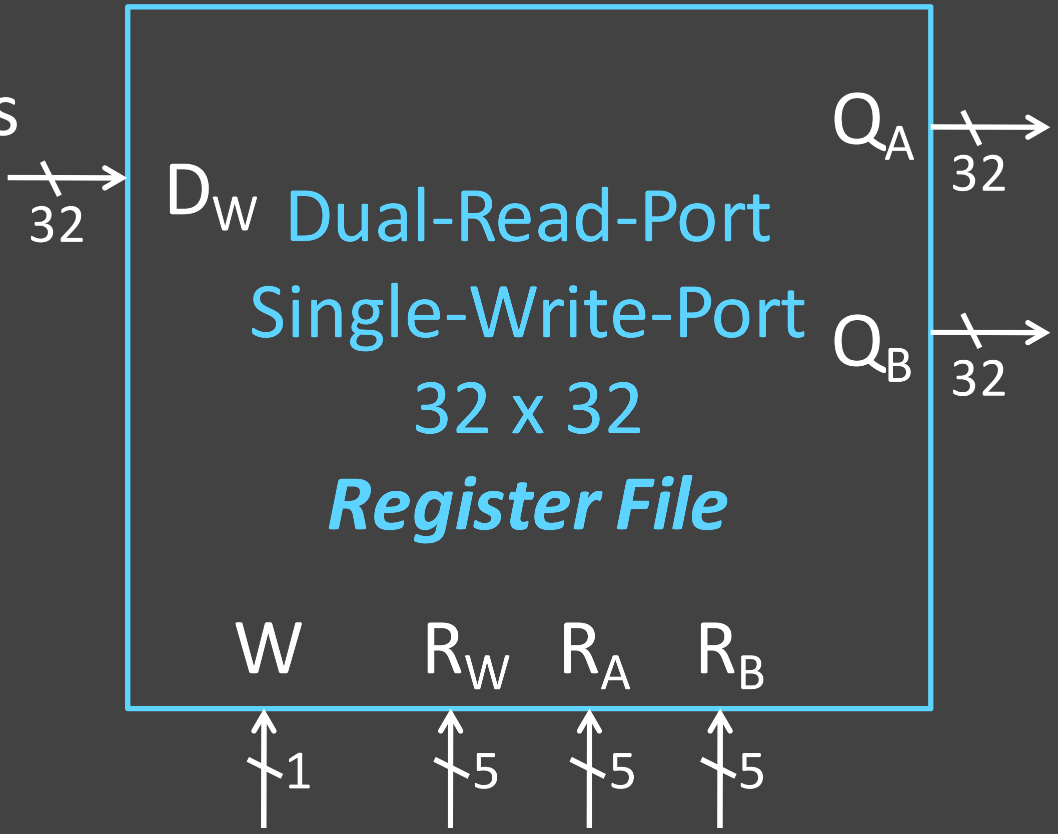
Memory

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

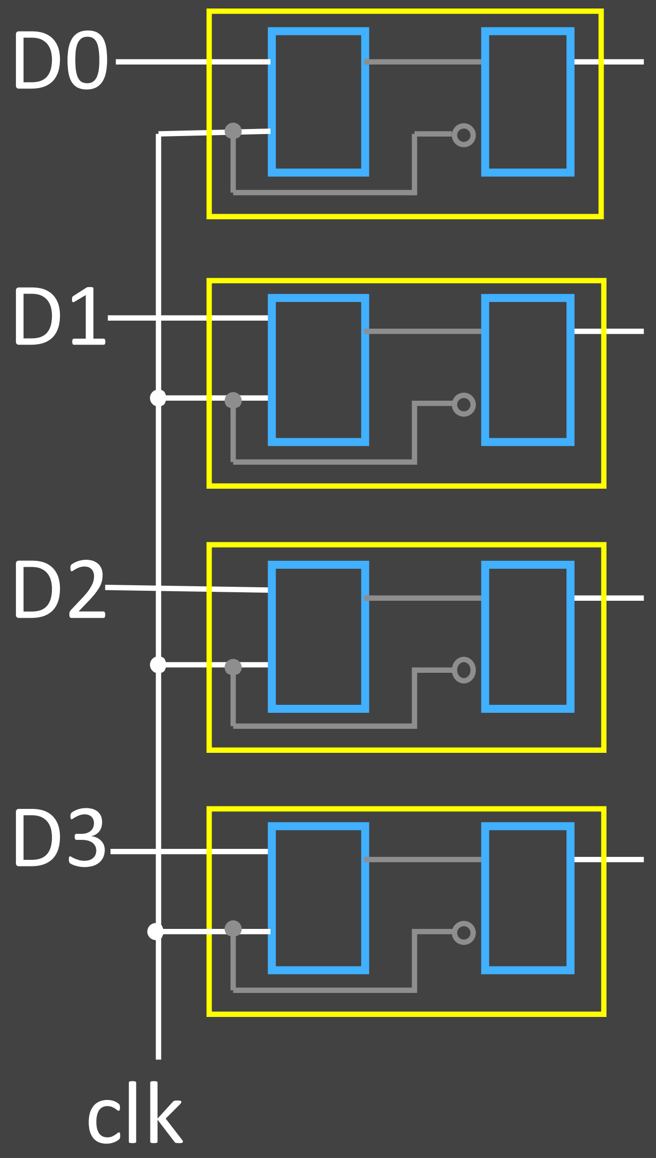
Register File

Register File

- N read/write registers
- Indexed by register number

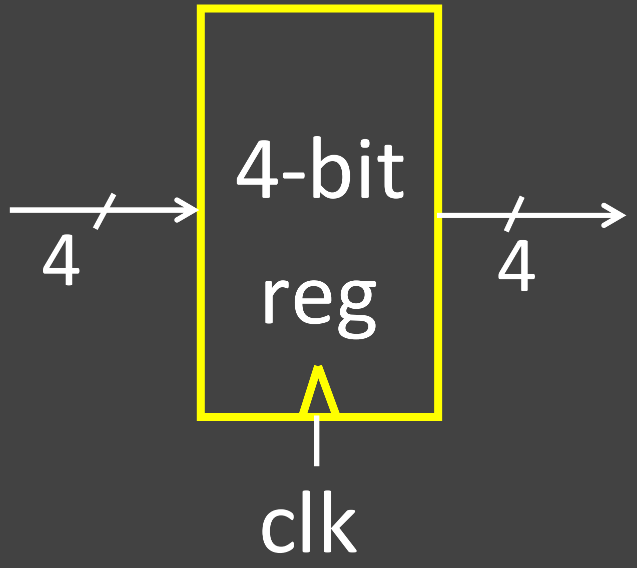


Register File



Recall: Register

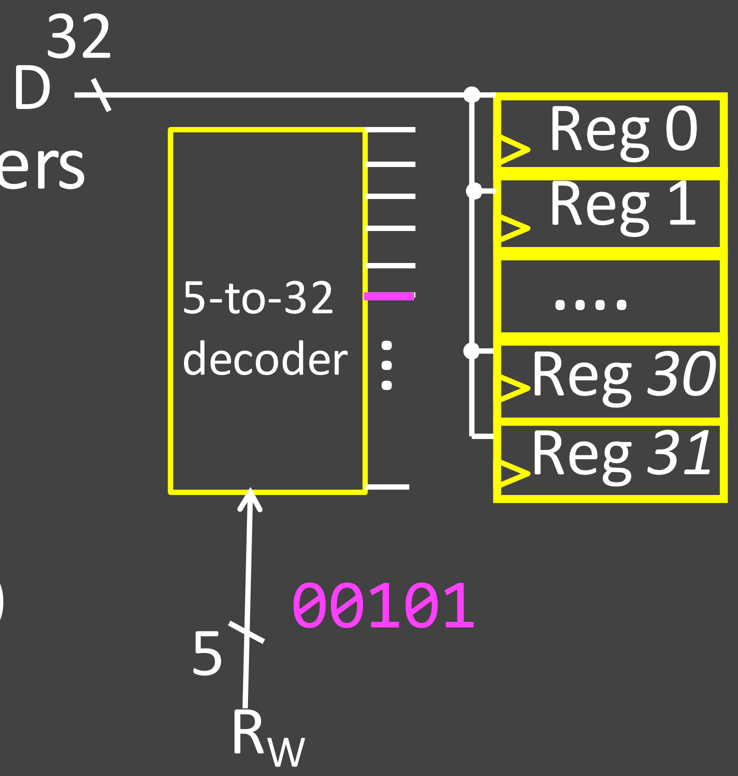
- D flip-flops in parallel
- shared clock
- extra clocked inputs: write_enable, reset, …



Writing to the Register File (1)

Register File

- N read/write registers
- Indexed by register number



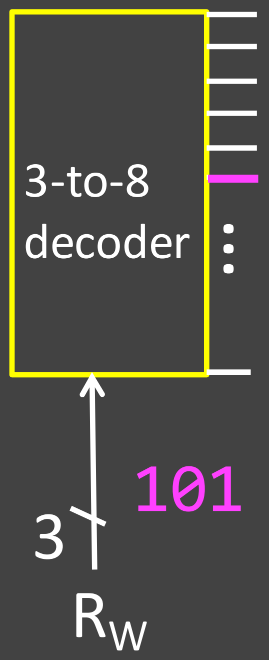
```
addi  r5, r0, 10
```

How to write to ***one*** register in the register file?

- Need a decoder

Activity: 3-to-8 decoder truth table & circuit

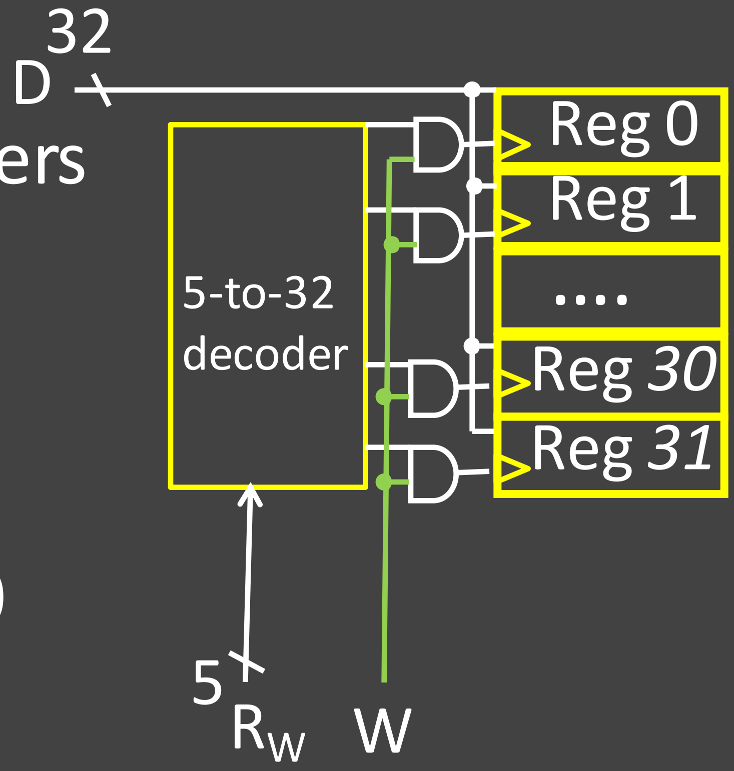
i2	i1	i0	o0	o1	o2	o3	o4	o5	o6	o7
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								



Writing to the Register File (2)

Register File

- N read/write registers
- Indexed by register number



```
addi  r5, r0, 10
```

How to write to ***one*** register in the register file?

- Need a decoder

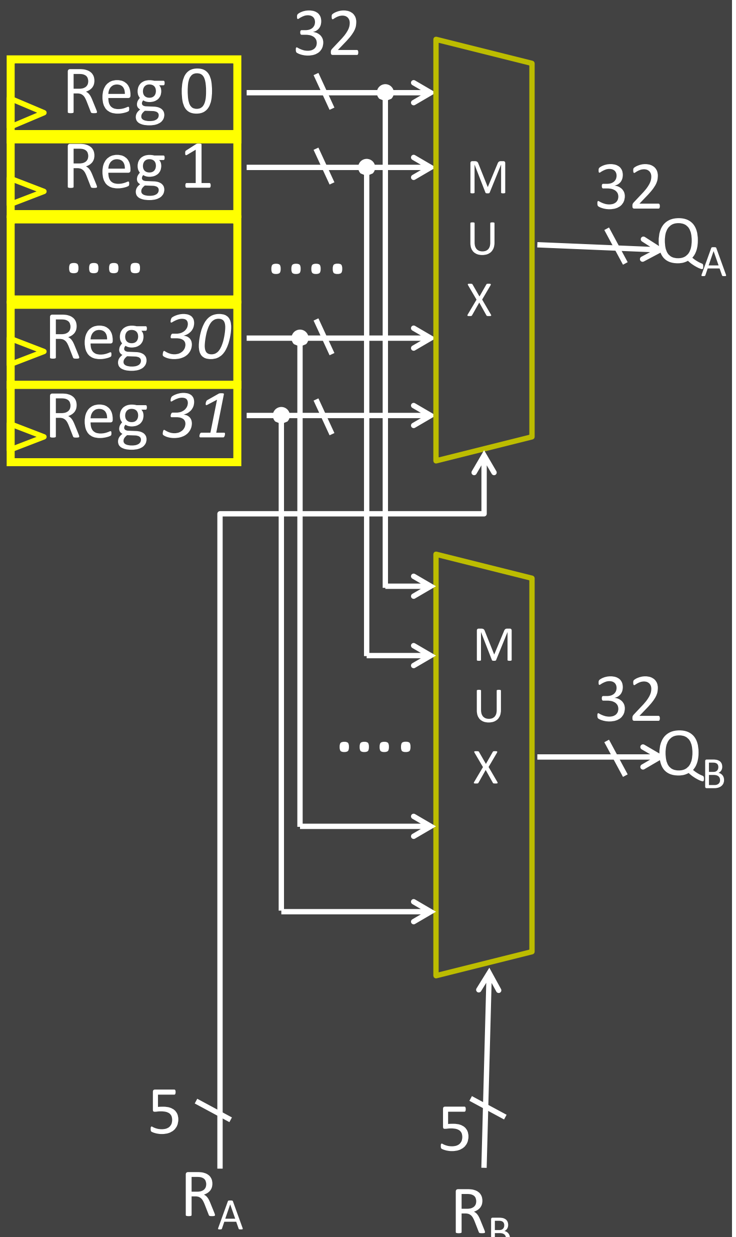
Reading from the Register File

Register File

- N read/write registers
- Indexed by register number

How to read from two registers?

- Need a multiplexor



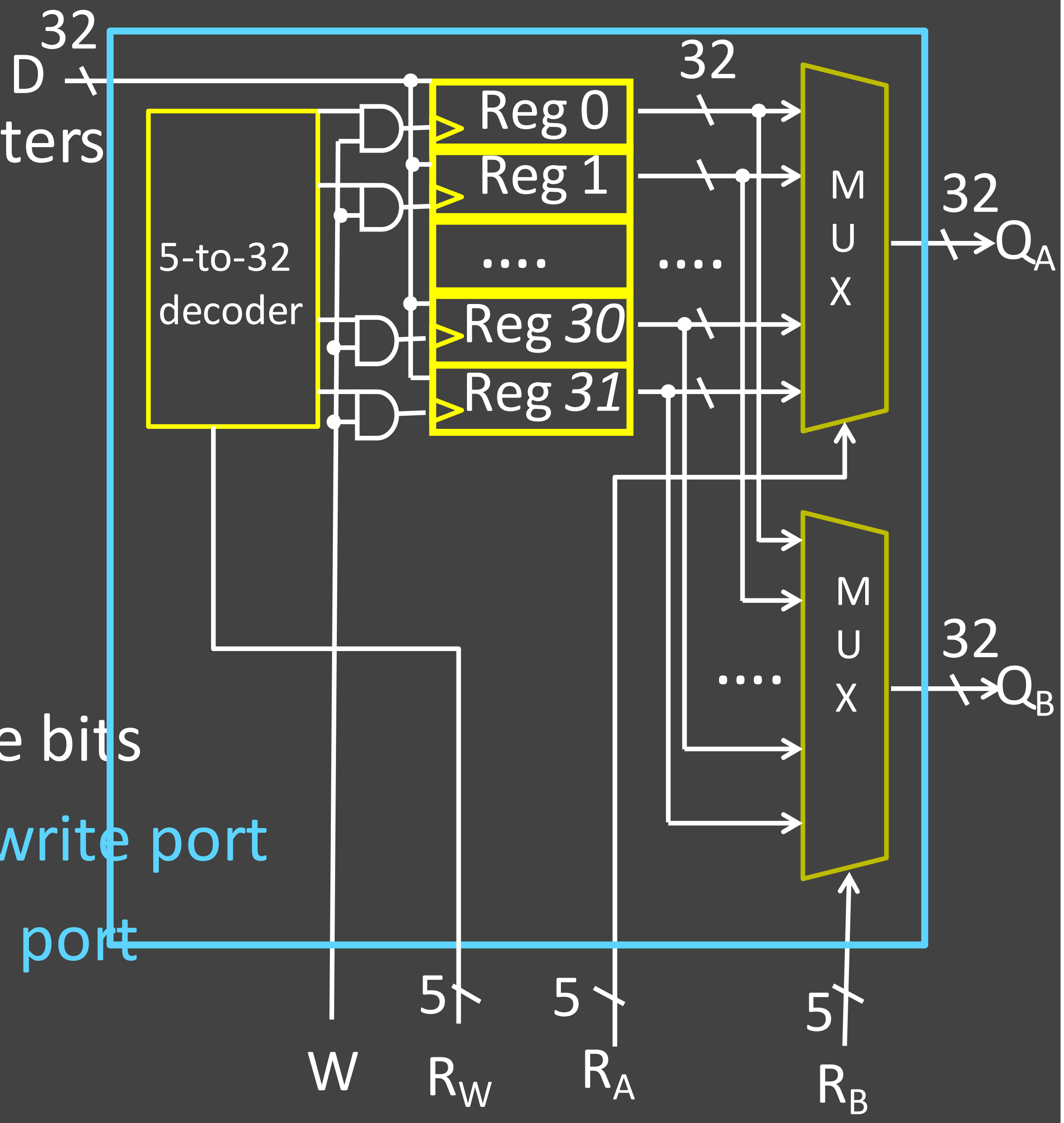
Register File

Register File

- N read/write registers
- Indexed by register number

Implementation:

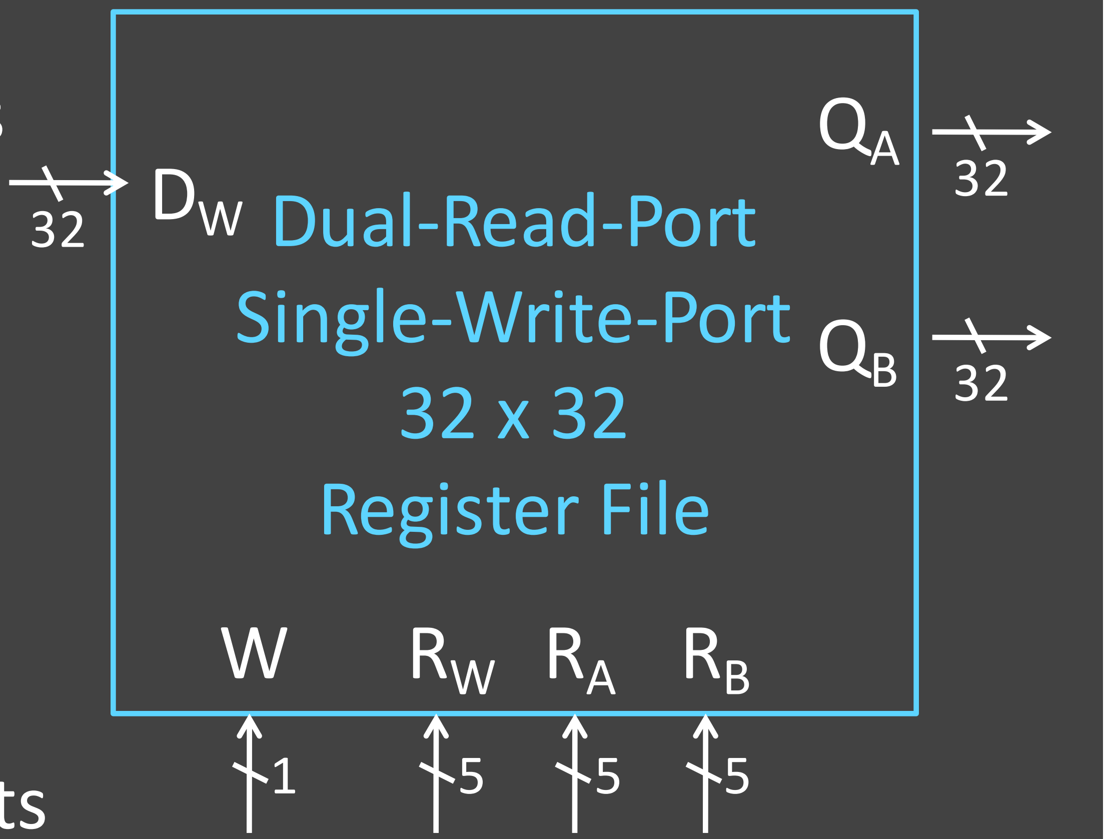
- D flip flops to store bits
- Decoder for each write port
- Mux for each read port



Register File

Register File

- N read/write registers
- Indexed by register number



Implementation:

- D flip flops to store bits
- Decoder for each write port
- Mux for each read port

Tradeoffs

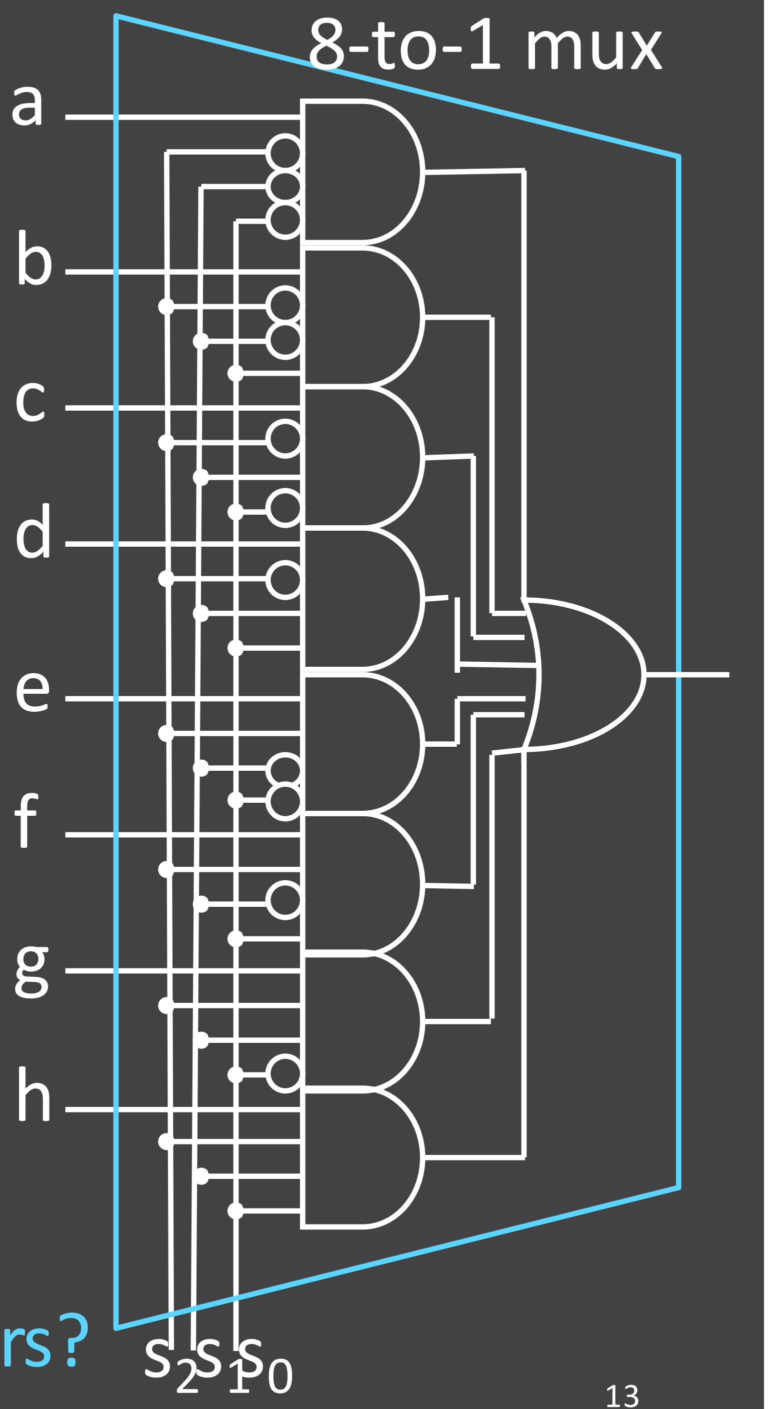
Register File tradeoffs

- + 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?



Goals for today

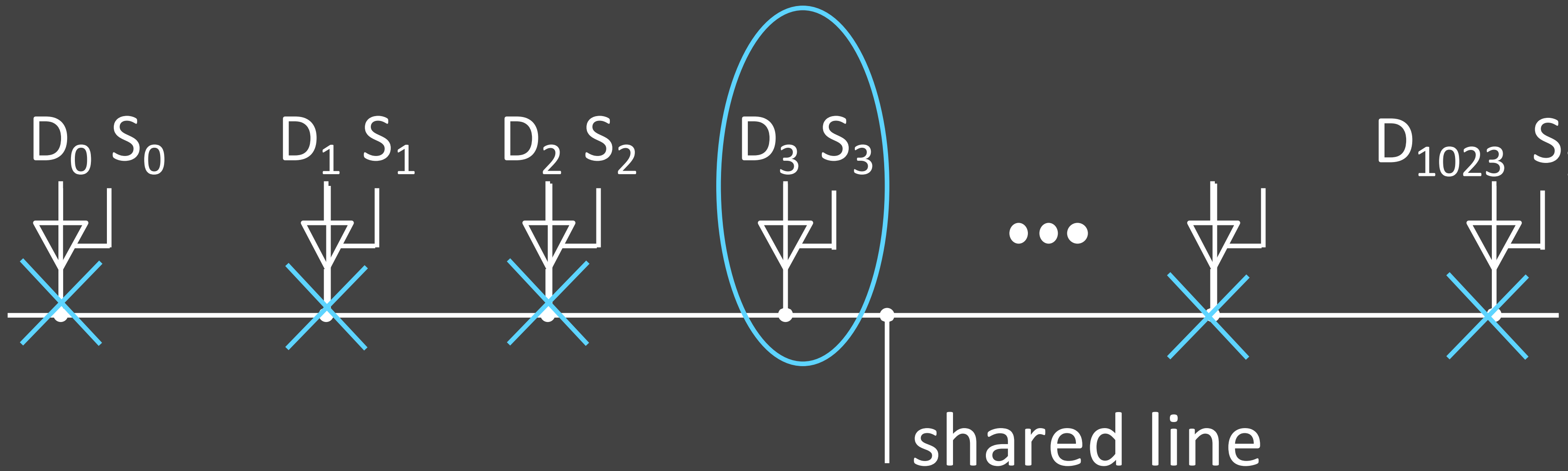
Memory

- 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)

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

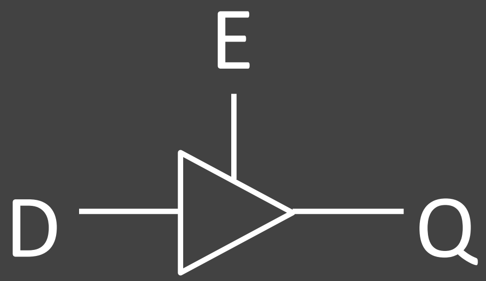


- How do we build such a device?

Tri-State Devices

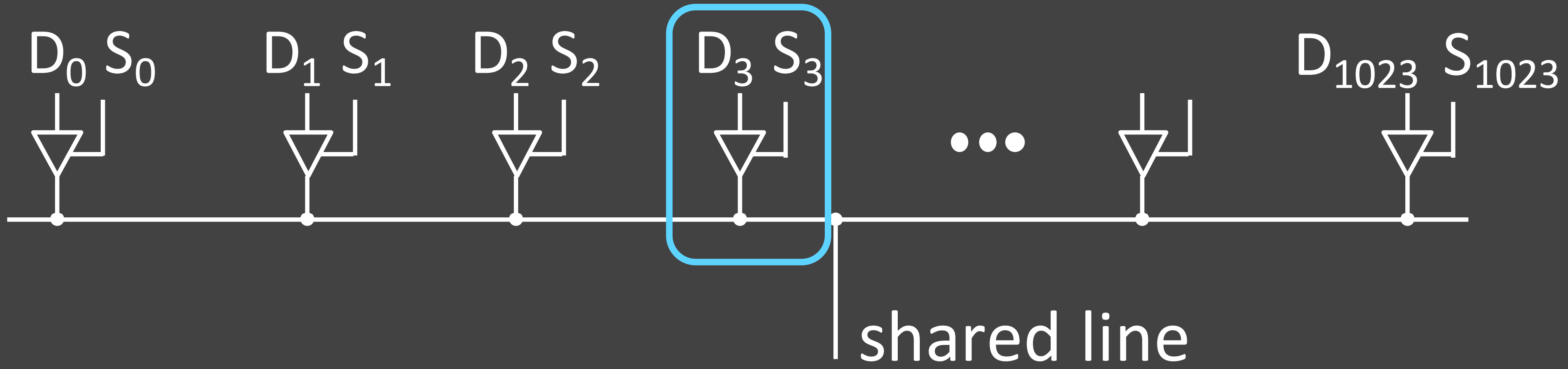
Tri-State Buffers

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



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

Shared Bus



Takeways

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.

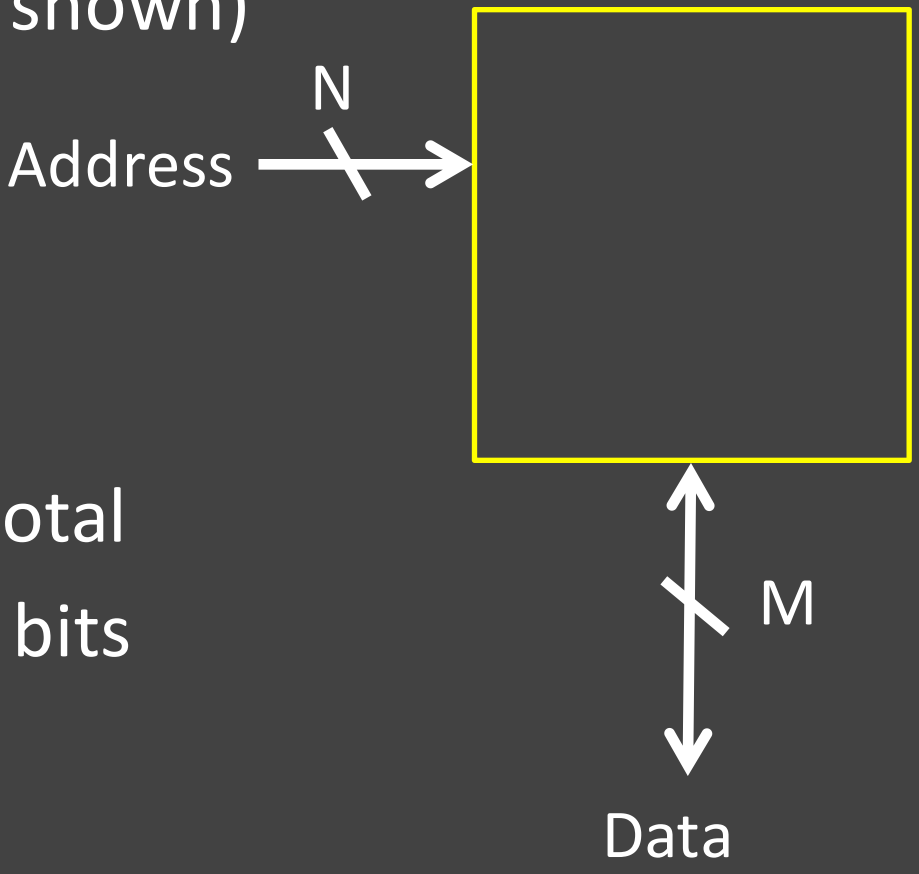
Goals for today

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)

Memory

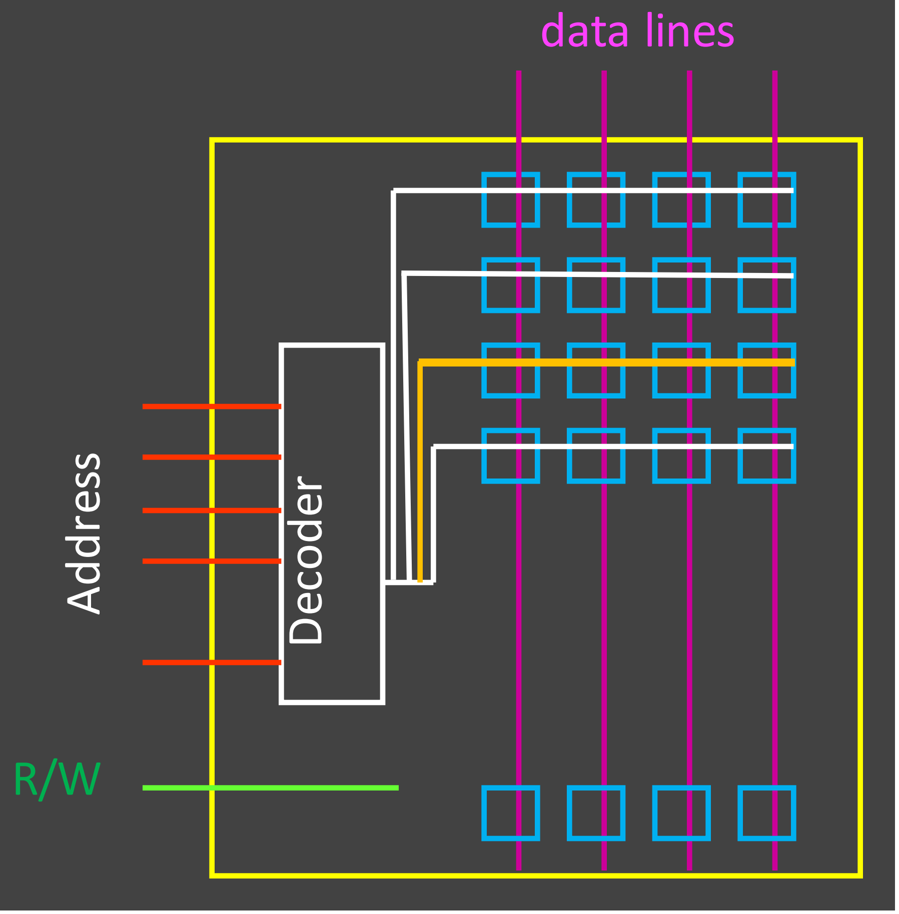
- Storage Cells + plus Tri-State Buffers
- Inputs: Address, Data (for writes)
- Outputs: Data (for reads)
- Also need R/W signal (not shown)



- N address bits → 2^N words total
- M data bits → each word M bits

Memory

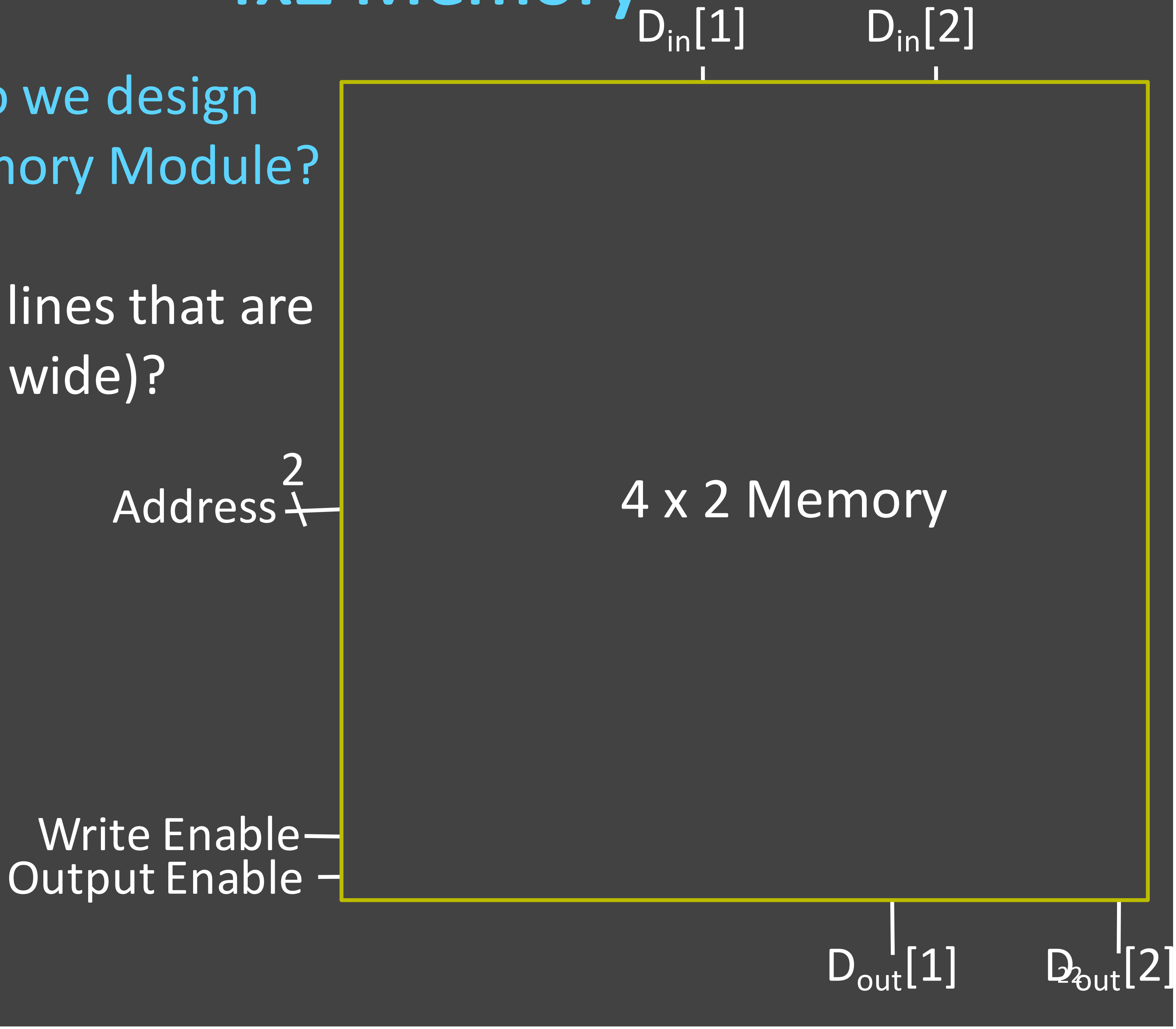
- Storage Cells + plus Tri-State Buffers
- Decoder selects a word line
- R/W selector determines access type
- Word line is then coupled to the data lines



4x2 Memory

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

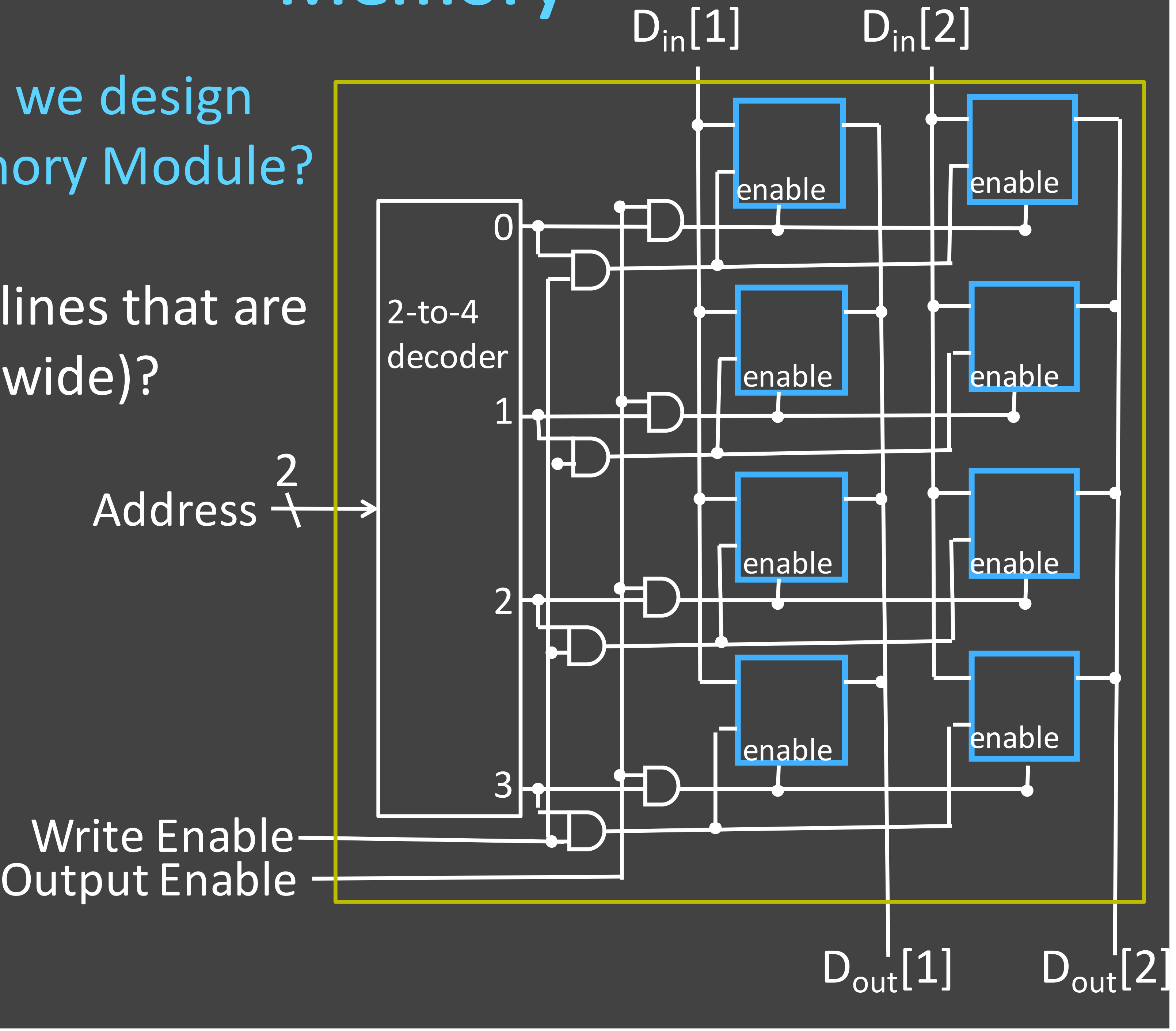
(i.e. 4 word lines that are each 2 bits wide)?



Memory

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

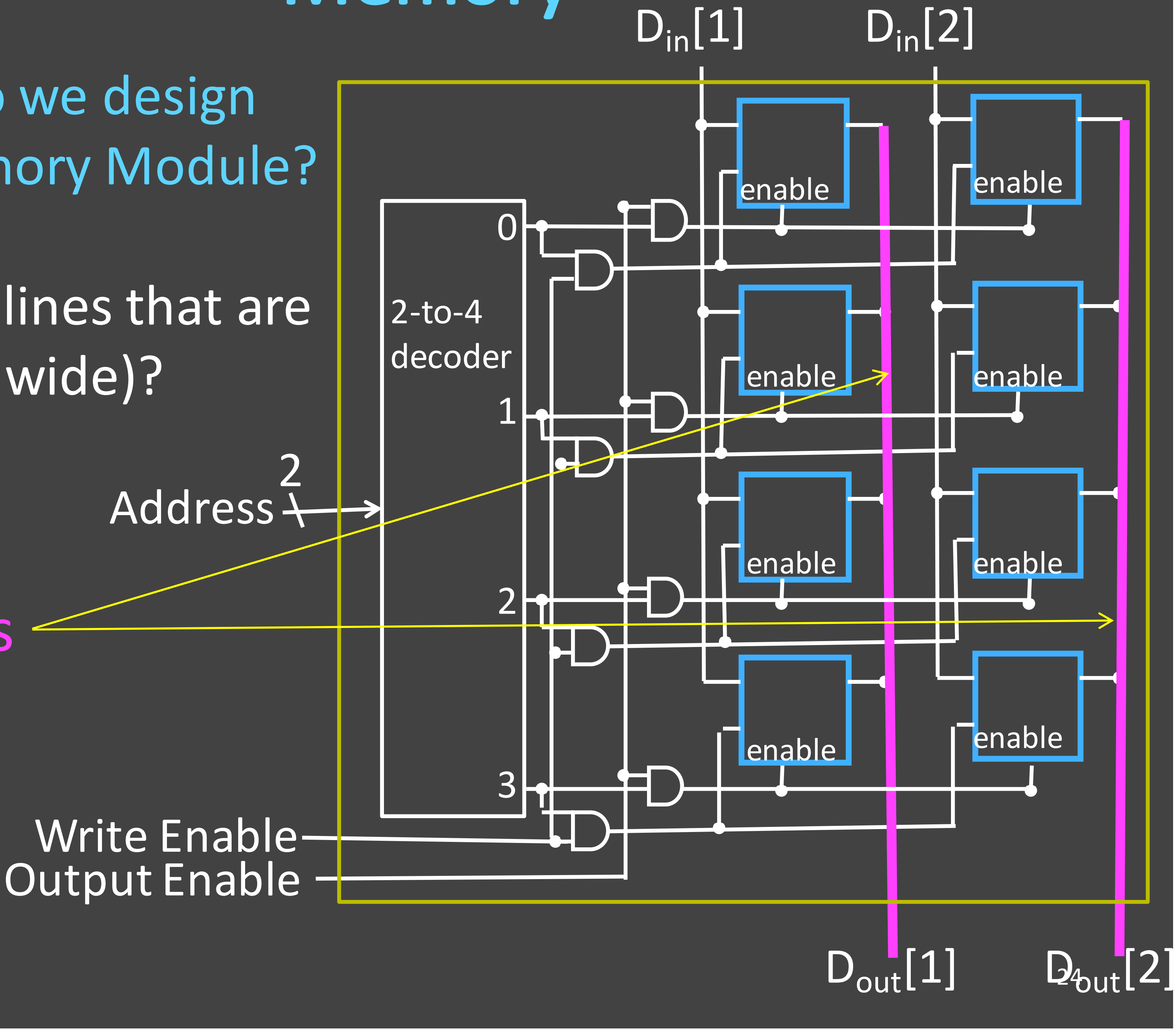
(i.e. 4 word lines that are each 2 bits wide)?



Memory

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

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

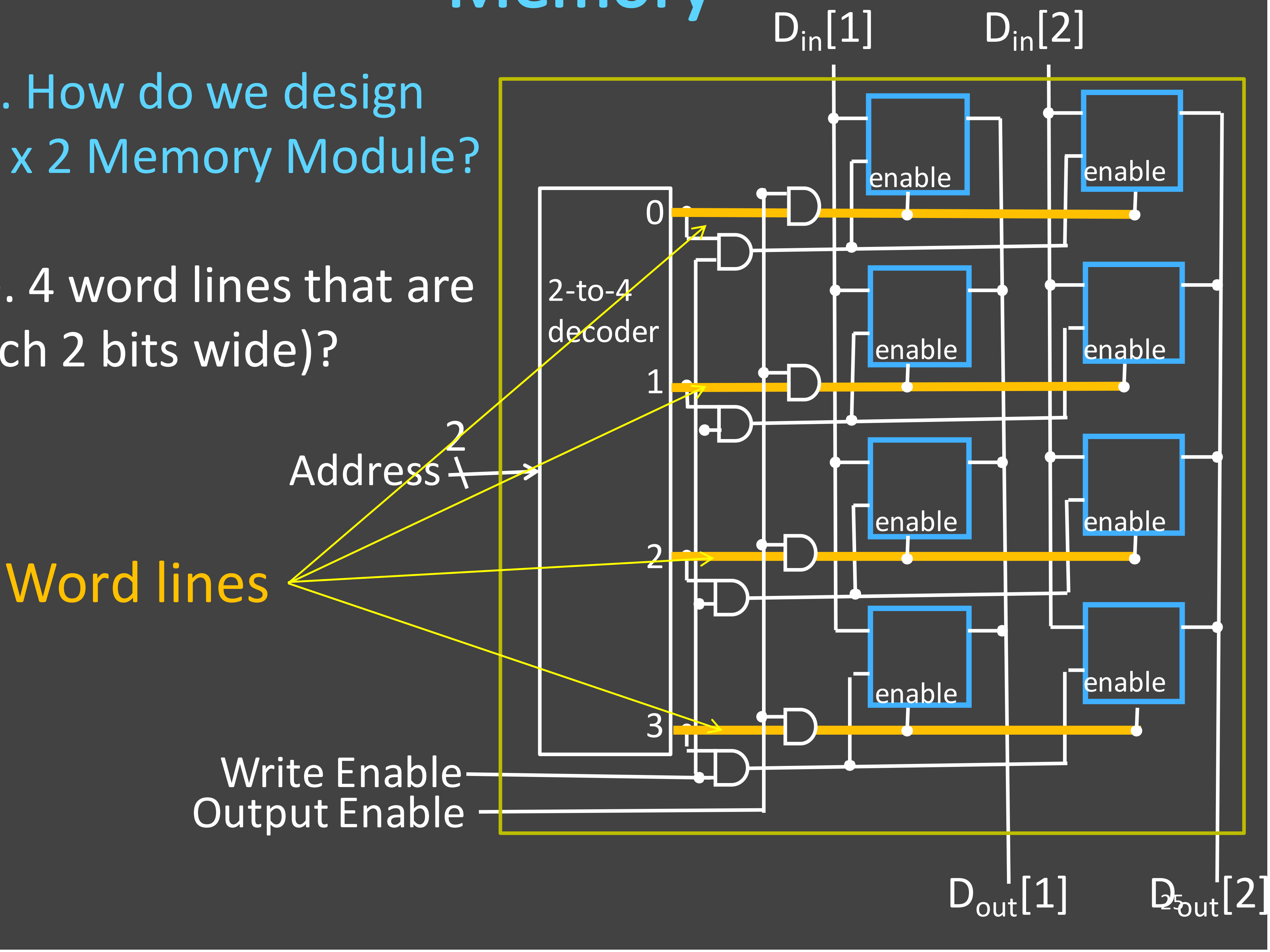


Bit lines

Memory

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

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



iClicker Question

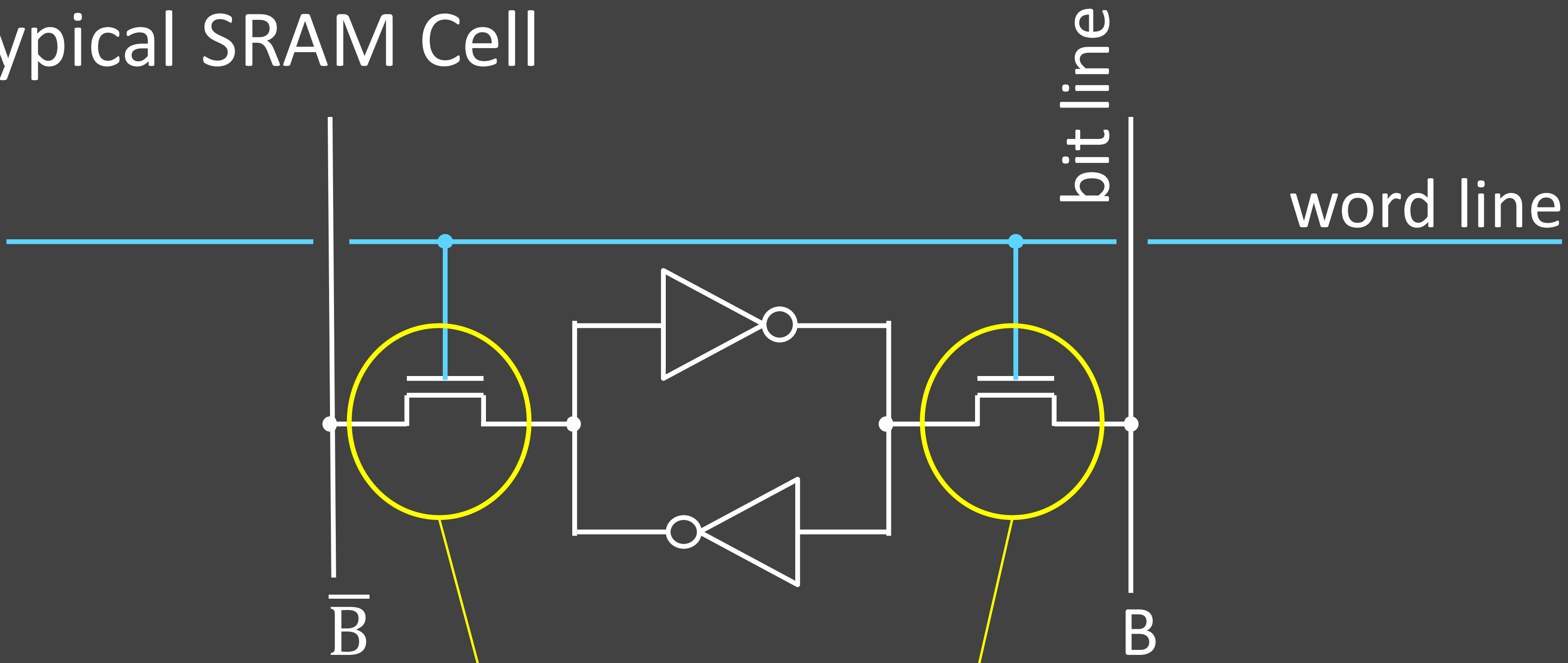
Frequency should be set to **AA**

What's your familiarity with memory (SRAM, DRAM)?

A. I've never heard of any of this.
B. I've heard the words SRAM and DRAM, but I have no idea what they are.
C. I know that DRAM means main memory.
D. I know the difference between SRAM and DRAM and where they are used in a computer system.

SRAM Cell

Typical SRAM Cell



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

Pass-Through Transistors

SRAM Summary

SRAM

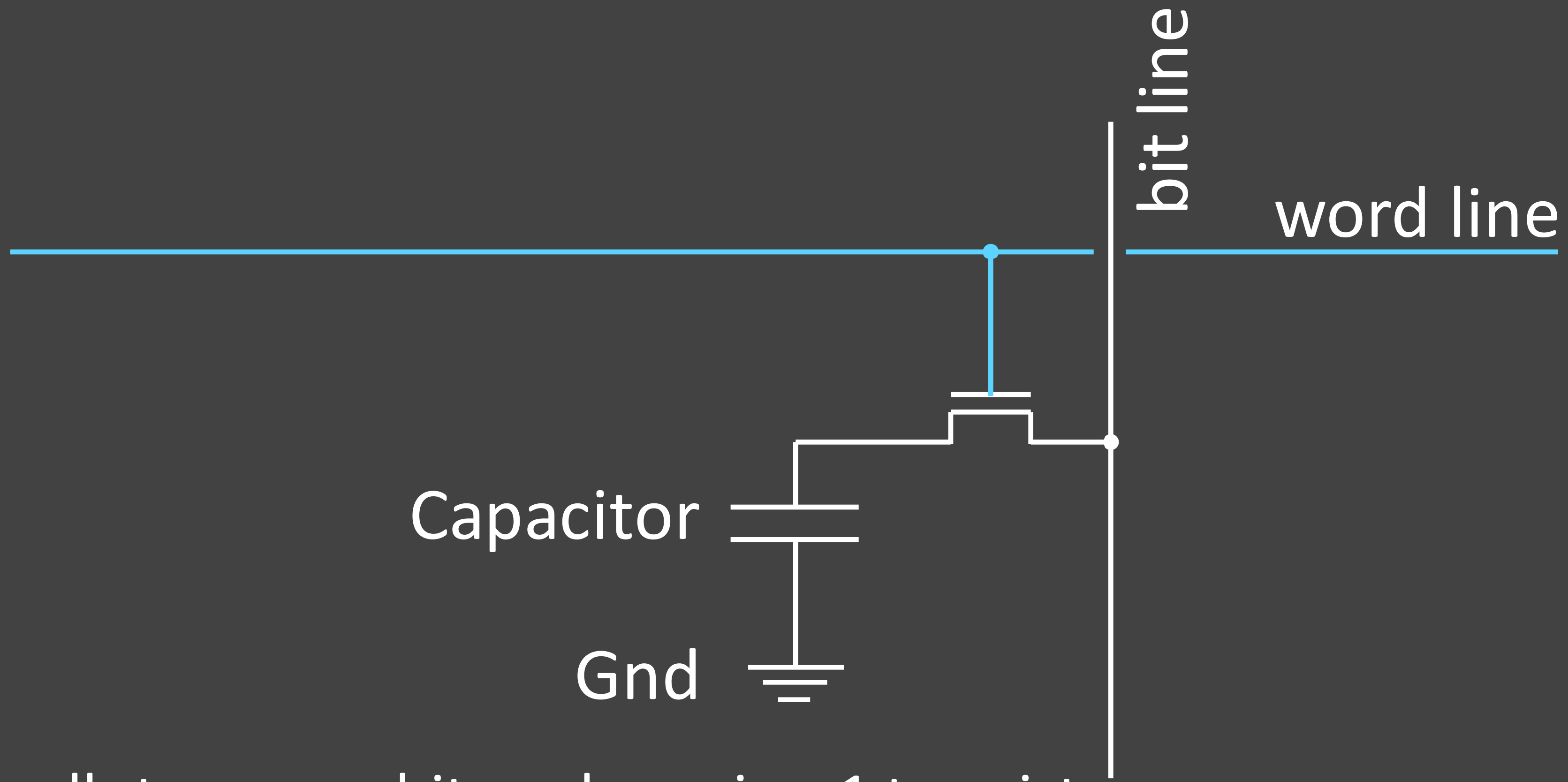
- A few transistors (~6) per cell
- Used for working memory (caches)

- But for even higher density…

Dynamic RAM: DRAM

Dynamic-RAM (DRAM)

- Data values require constant refresh

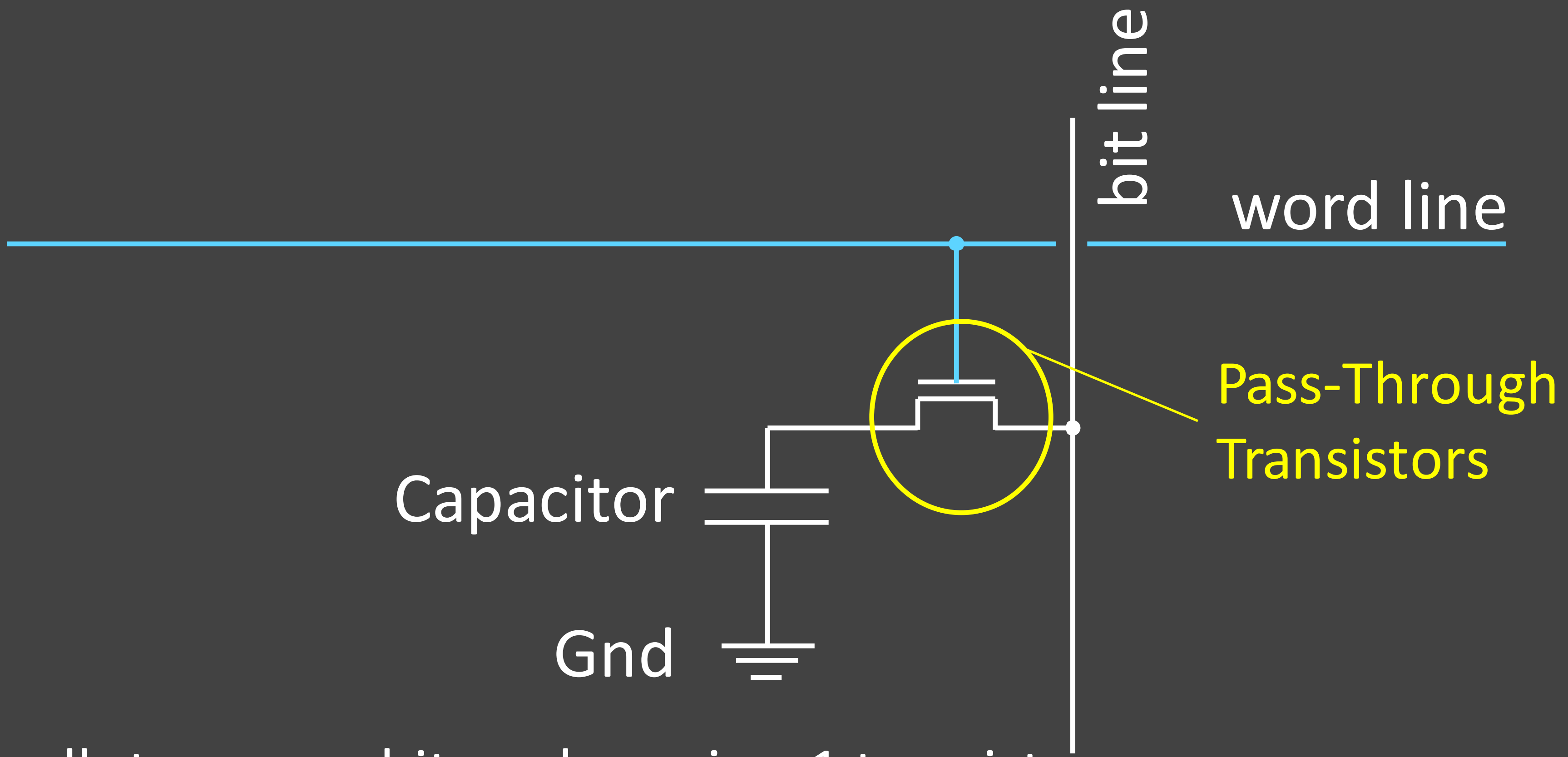


Each cell stores one bit, and requires 1 transistors

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

Finally have the 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!