Memory

Hakim Weatherspoon
CS 3410, Spring 2011
Computer Science
Cornell University

See: P&H Appendix C.8, C.9

Announcements

HW1 due today

HW2 available later today

HW2 due in one week and a half

Work alone

Use your resources

- FAQ, class notes, book, Sections, office hours, newsgroup, CSUGLab

Make sure you

- Registered for class, can access CMS, have a Section, and have a project partner
- Check online syllabus/schedule, review slides and lecture notes, Office Hours, early homework and programming assignments

Announcements

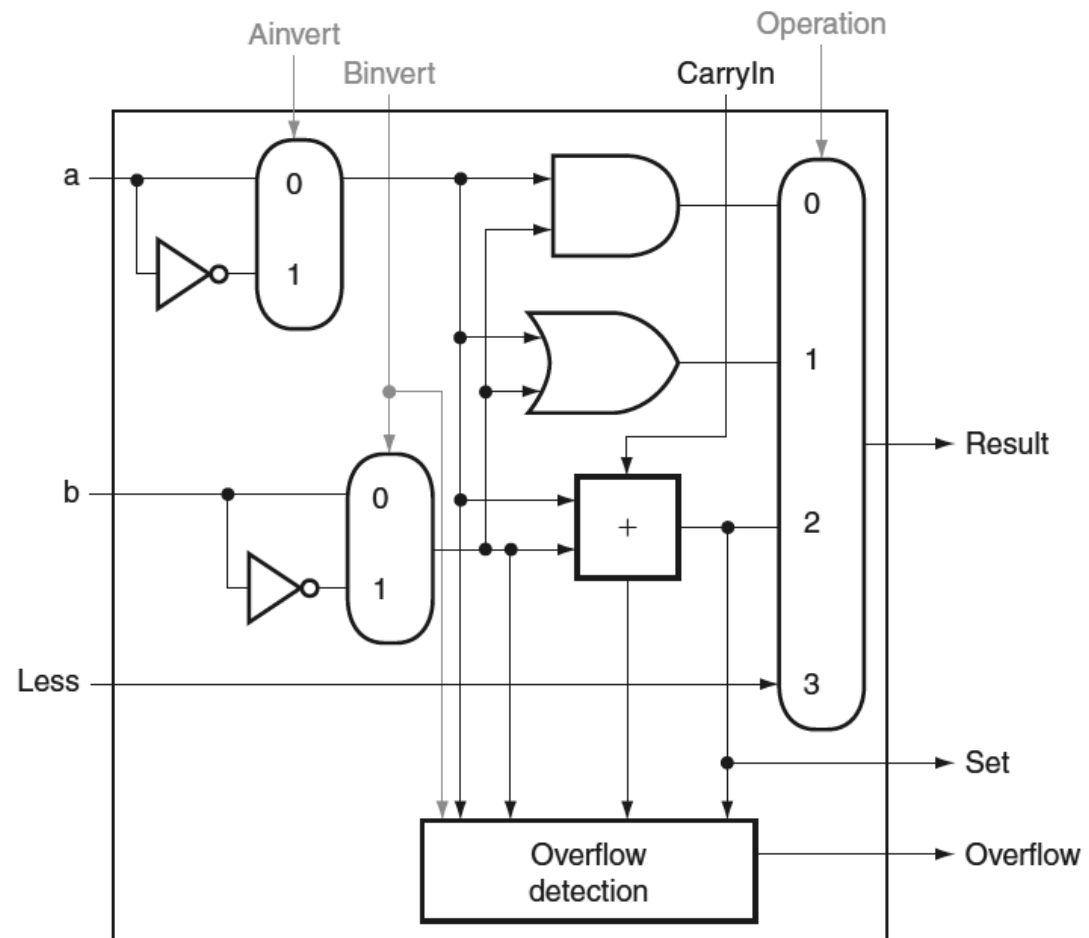Prelims: Evening of Thursday, March 10 and April 28th

Late Policy

1) Each person has a total of four "slip days"

2) For projects, slip days are deducted from all partners

3) 10% deducted per day late after slip days are exhausted

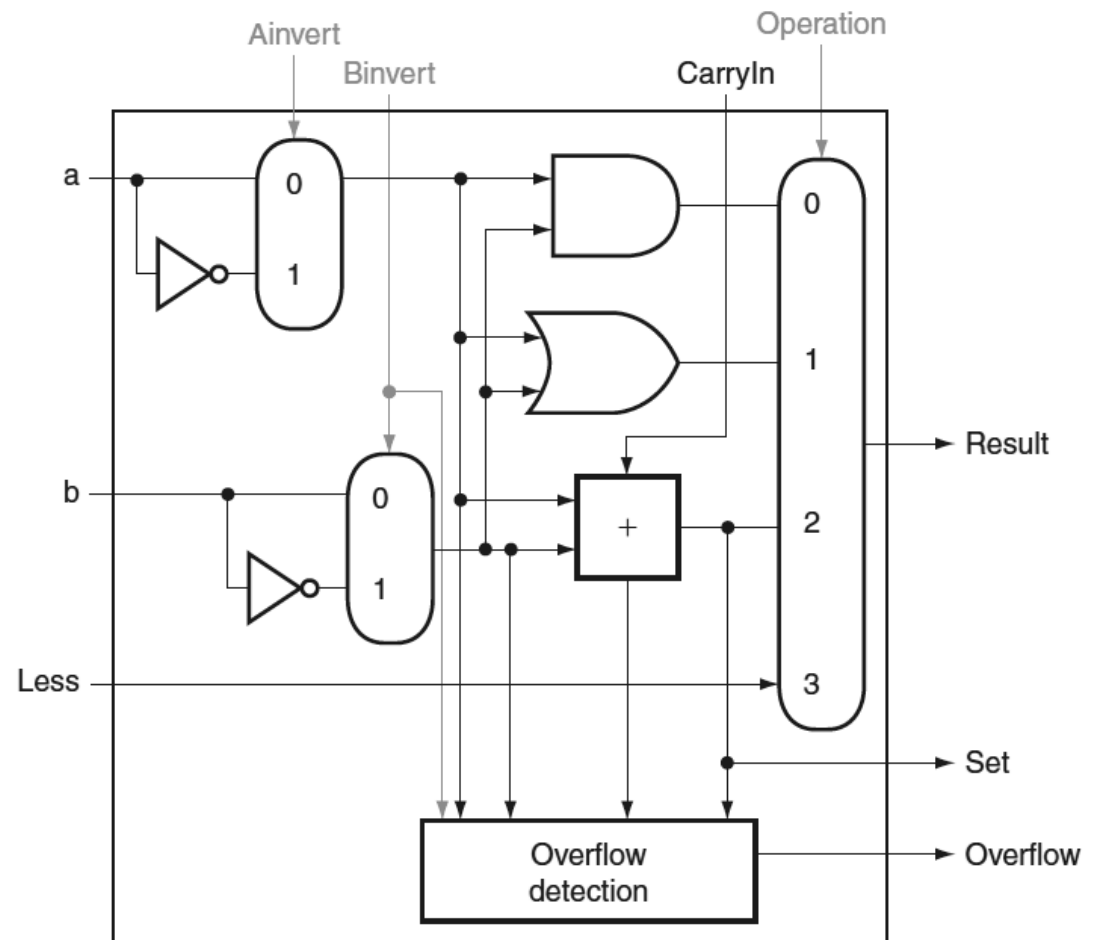# Critical Path

Which operation is the critical path?

- A) AND
- B) OR
- C) ADD/SUB
- D) LT

Critical Path
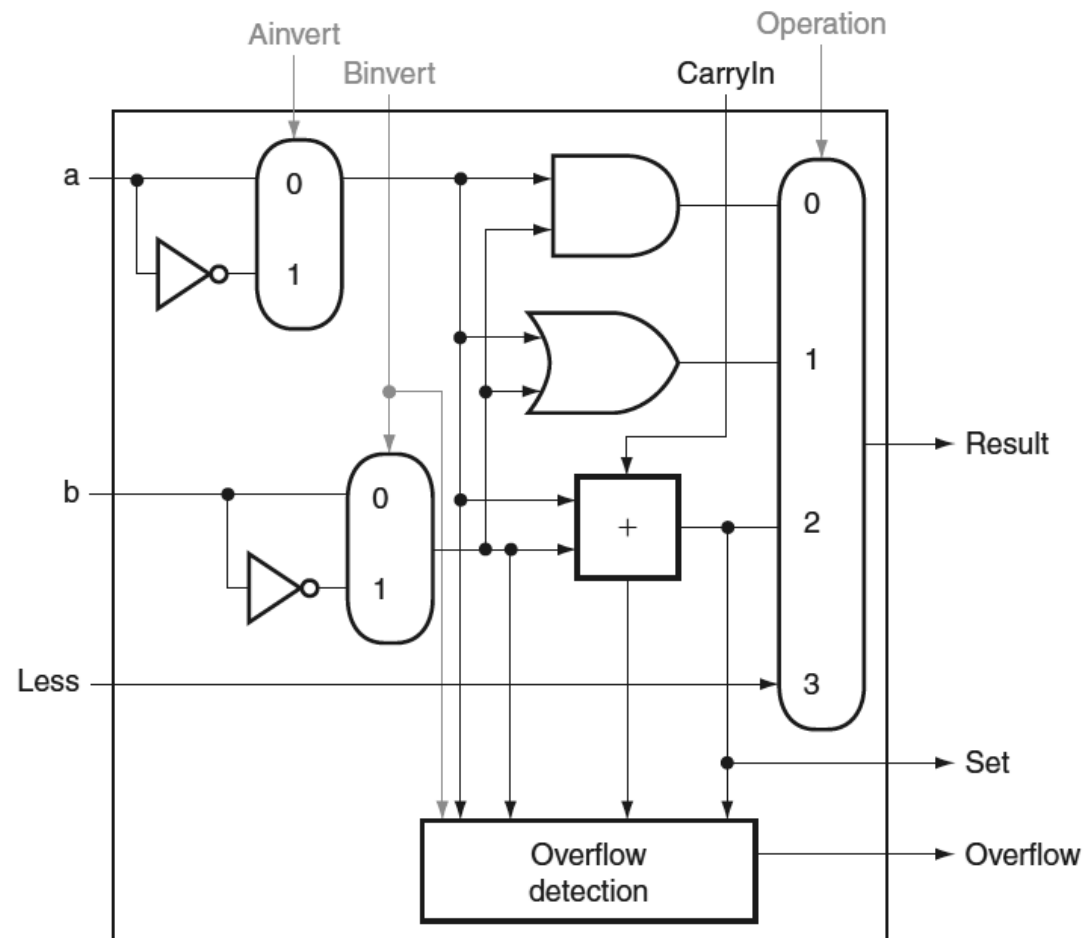
What is the length of the critical path (in gates)?
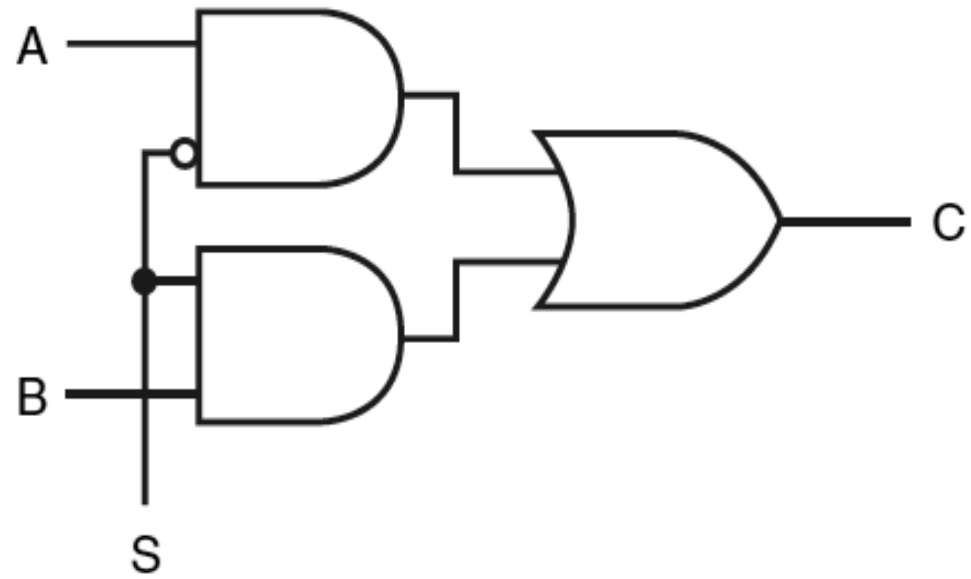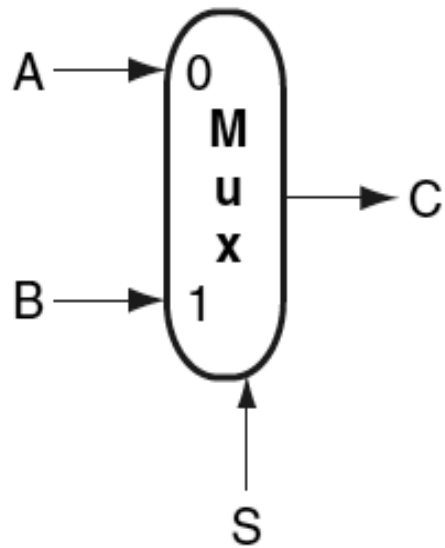
- A) 3
- B) 5
- C) 7
- D) 9

Critical Path

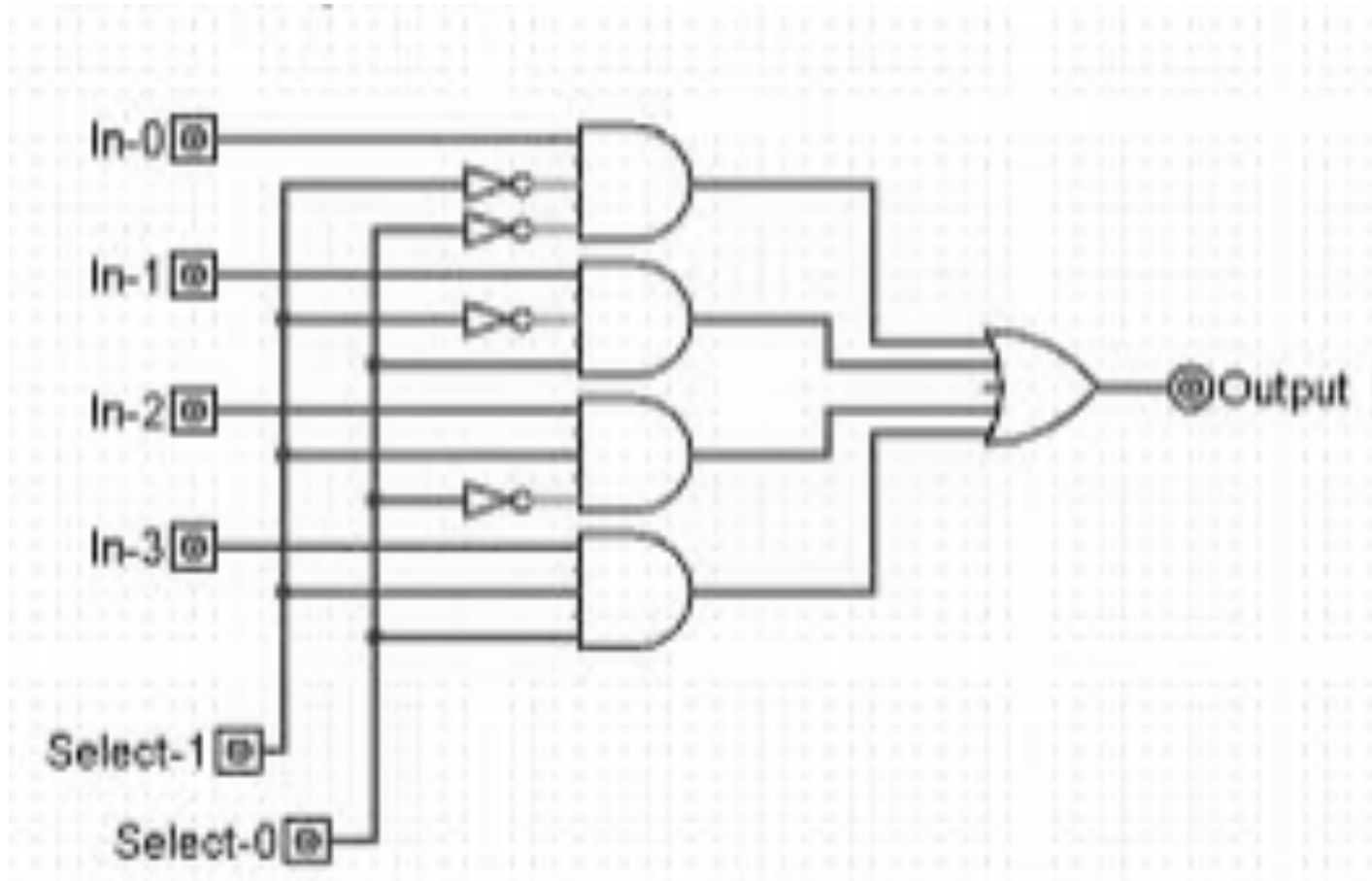What is the length of the critical path for a 32-bit ALU (in gates)?

- A) 9
- B) 32
- C) 64
- D) 69

Multiplexor

Multiplexor

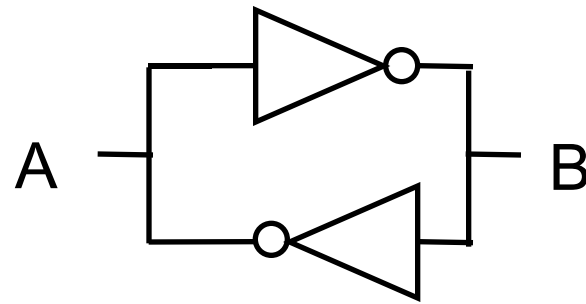# Goals for today

Review

- SR Latches, D Latches, D Flip Flips, and Registers

Memory

- Register Files

- Tri-state devices

- SRAM (Static RAM—random access memory)

- DRAM (Dynamic RAM)
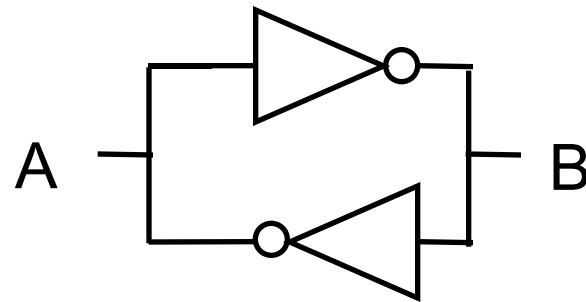
Bistable Devices

- Stable and unstable equilibria?

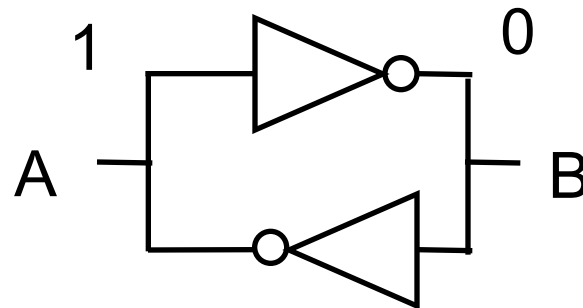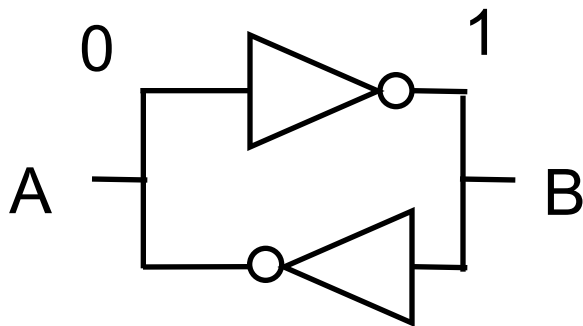A ———| A Simple Device   B

Bistable Devices

- Stable and unstable equilibria?

A Simple Device

- In stable state, $\bar{A} = B$

- How do we change the state?

SR Latch
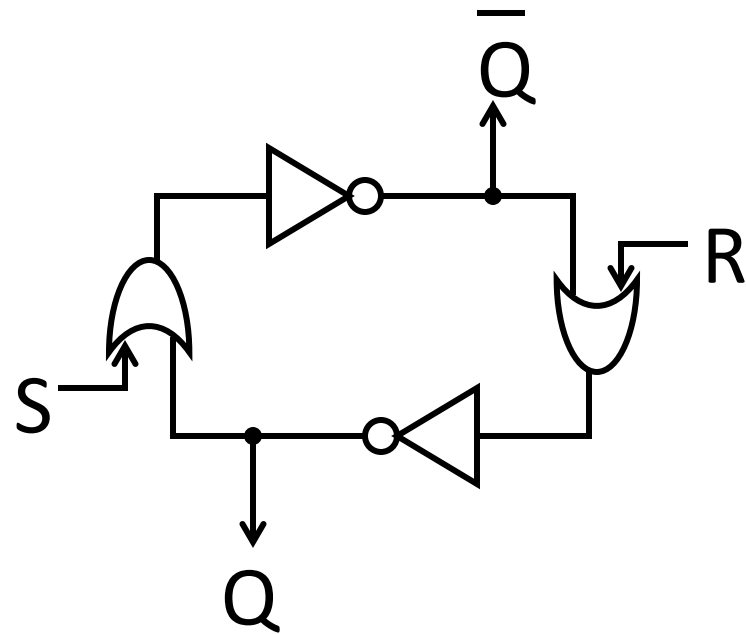
SR Latch

Set-Reset (SR) Latch

Stores a value Q and its complement $\overline{Q}$

S	R	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1		
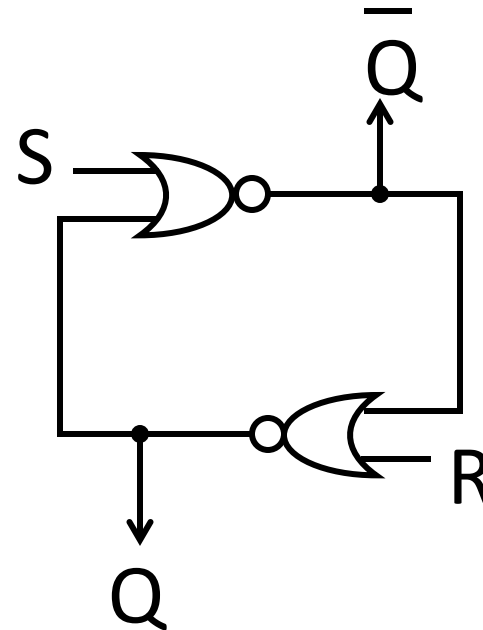
SR Latch

Set-Reset (SR) Latch

    Stores a value Q and its complement $\overline{Q}$
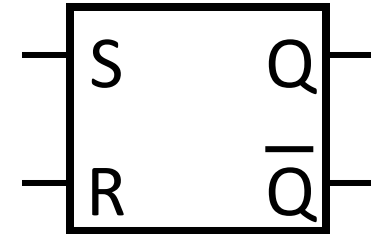
S	R	Q	$\overline{Q}$
0	0	Q	Q
0	1	0	1
1	0	1	0
1	1	forbidden	
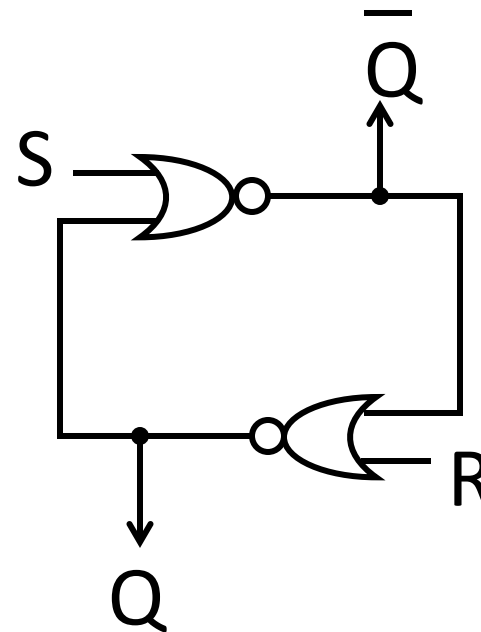
Unclocked D Latch
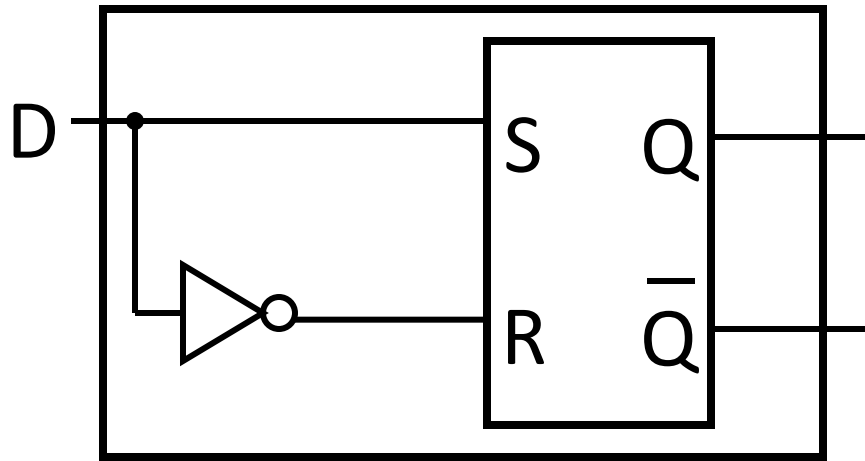
Data (D) Latch

D	Q	$\overline{Q}$
0		
1		

Unclocked D Latch
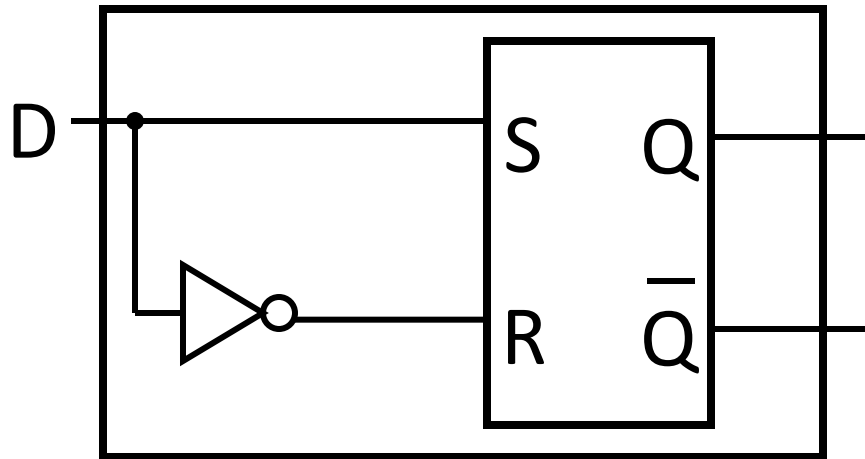
Data (D) Latch

D	Q	$\overline{Q}$
0	0	1
1	1	0

Data Latch
- Easier to use than an SR latch
- No possibility of entering an undefined state

When D changes, Q changes
  - ... immediately (after a delay of 2 Ors and 2 NOTs)

Need to control when the output changes

D Latch with Clock
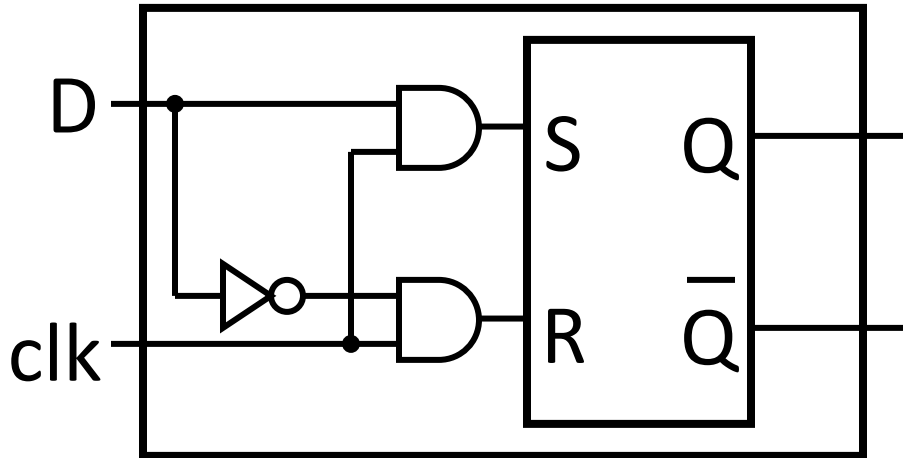
D	Q	$\overline{Q}$
0	0	1
1	1	0

Level Sensitive D Latch

Clock high:

   set/reset (according to D)

Clock low:

   keep state (ignore D)

D Latch with Clock
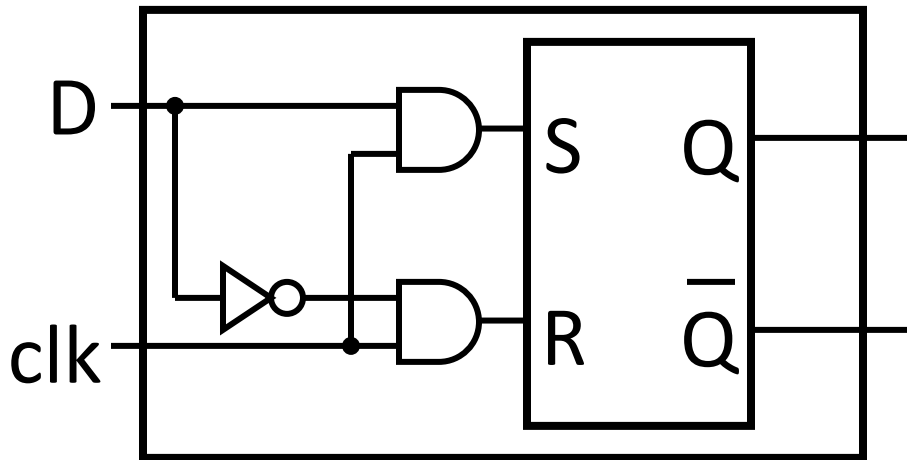
D	Q	$\overline{Q}$
0	0	1
1	1	0

S	R	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	0	1
1	0	1	0
1	1	forbidden	

clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

D Latch with Clock
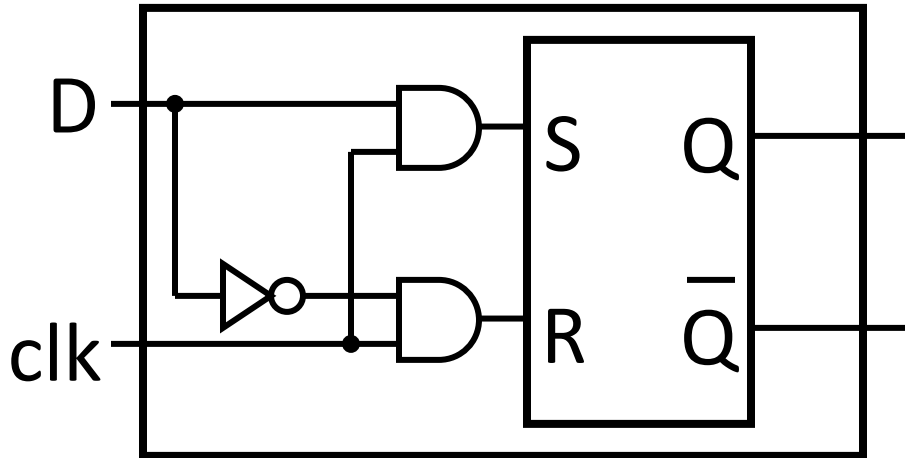
D	Q	$\overline{Q}$
0	0	1
1	1	0
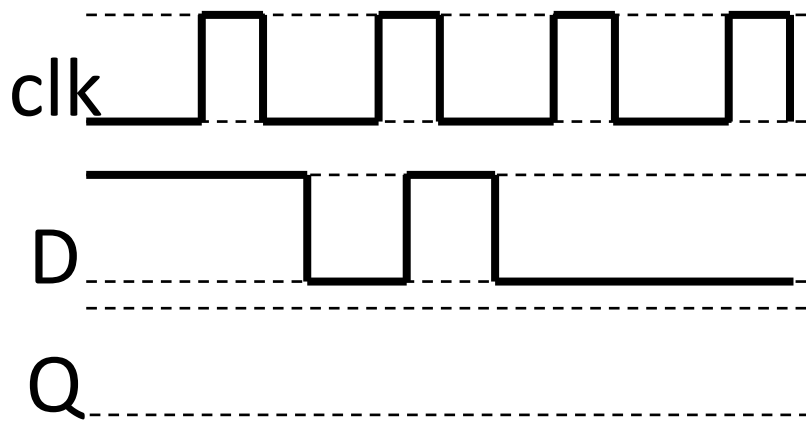
clk	D	Q	$\overline{Q}$
0	0	Q	$\overline{Q}$
0	1	Q	$\overline{Q}$
1	0	0	1
1	1	1	0

Edge-Triggered D Flip-Flop

D Flip-Flop

- Edge-Triggered
- Data is captured when clock is high
- Outputs change only on falling edges

Registers

D0

D1

D2

D3

clk
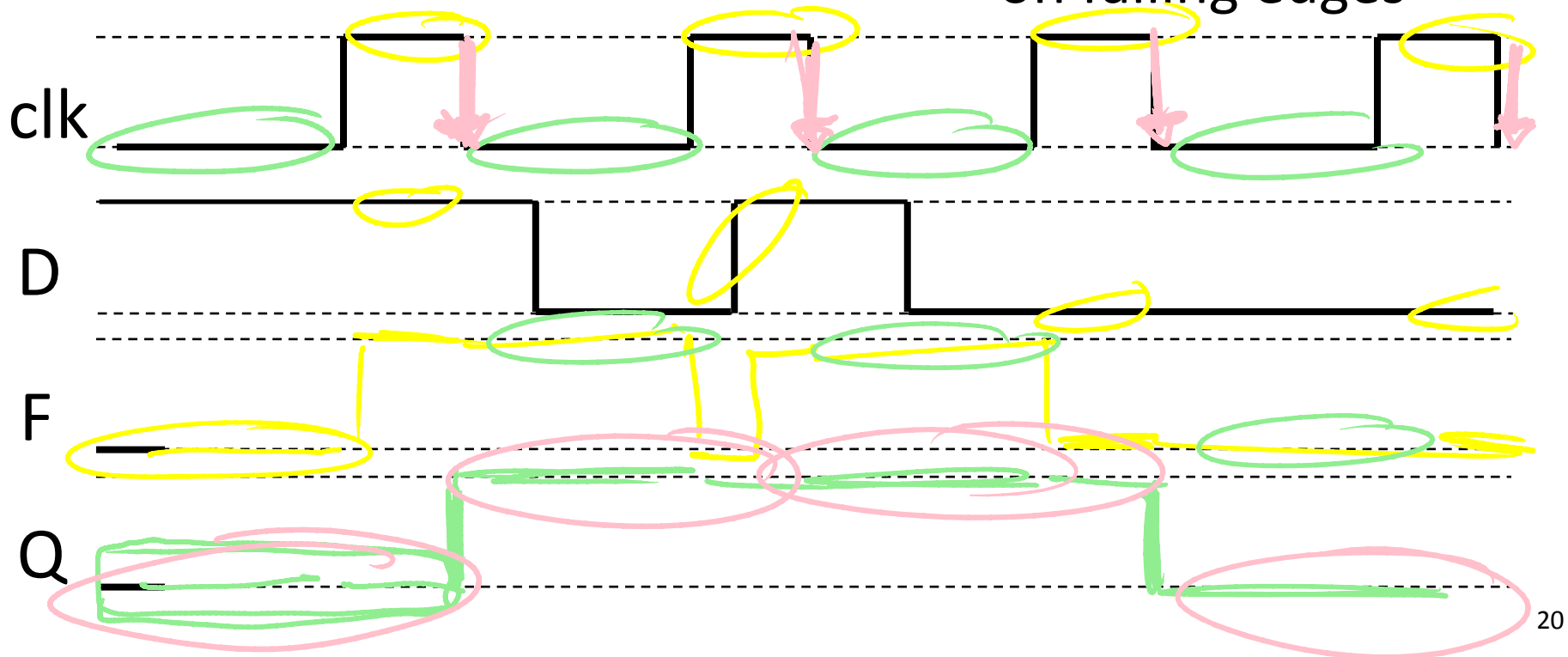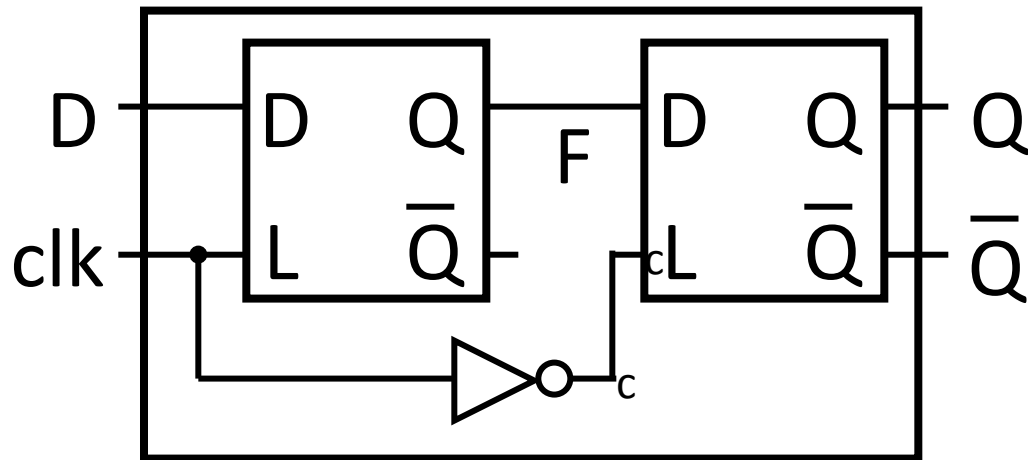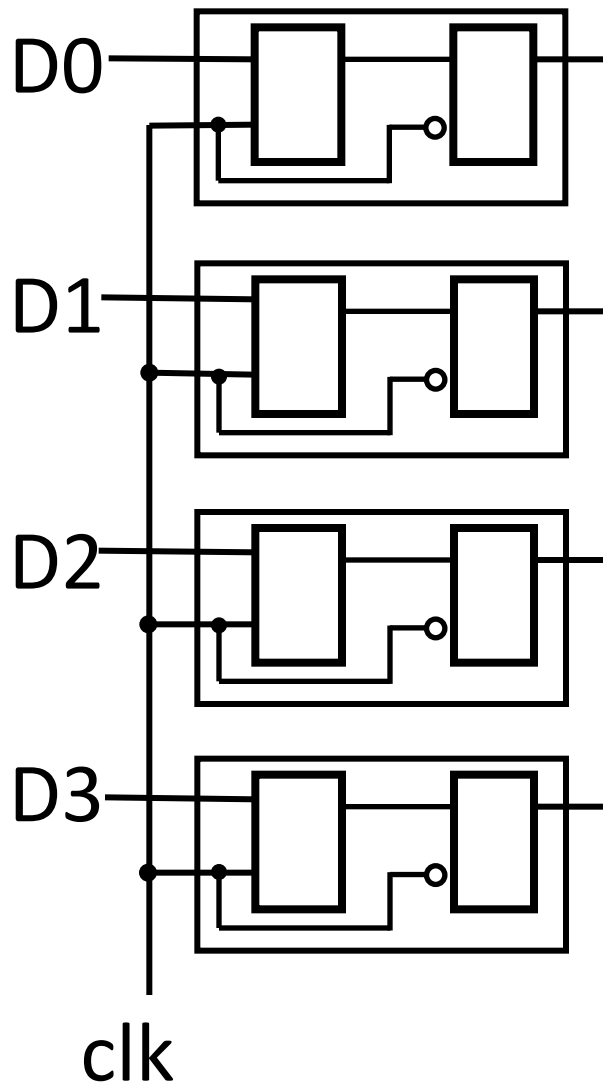
Register

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

4-bit reg

4

4

Voting Machine

Goals for today

Review

- SR Latches, D Latches, D Flip Flips, and Registers

Memory

- Register Files

- Tri-state devices

- SRAM (Static RAM—random access memory)

- DRAM (Dynamic RAM)

Register File

Register File
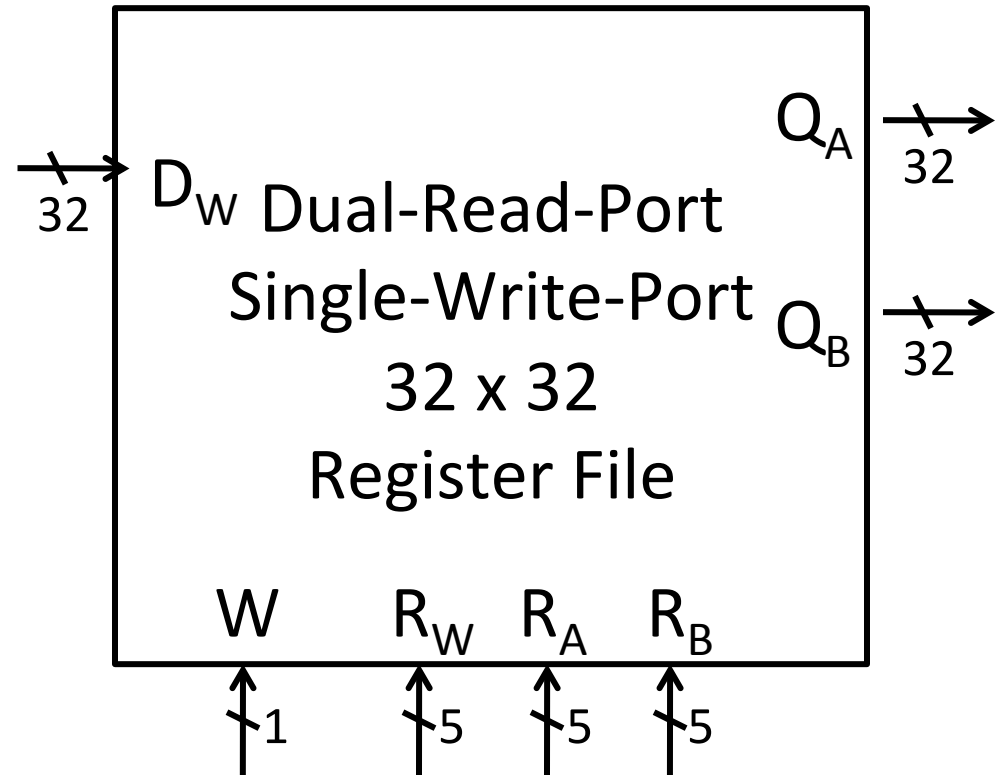
- N read/write registers
- Indexed by register number

```
                    ┌──────────────────────────────┐
                    │                              │  Q_A  ↗→
          ↗→ D_W    │   Dual-Read-Port             │       32
          32        │   Single-Write-Port     Q_B  │  ↗→
                    │       32 x 32                 │       32
                    │     Register File            │
                    │                              │
                    │   W      R_W    R_A    R_B    │
                    └───↑───────↑──────↑──────↑─────┘
                       ↑1      ↑5     ↑5     ↑5
```
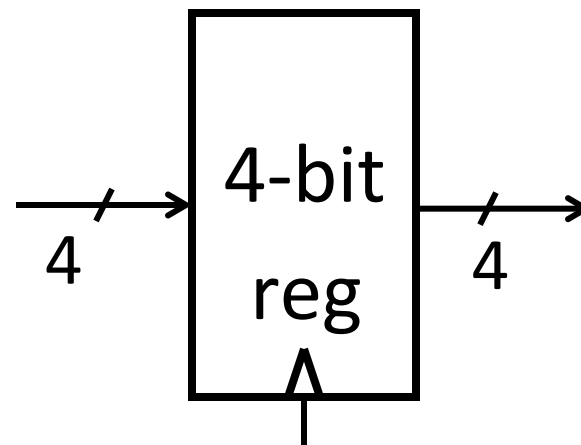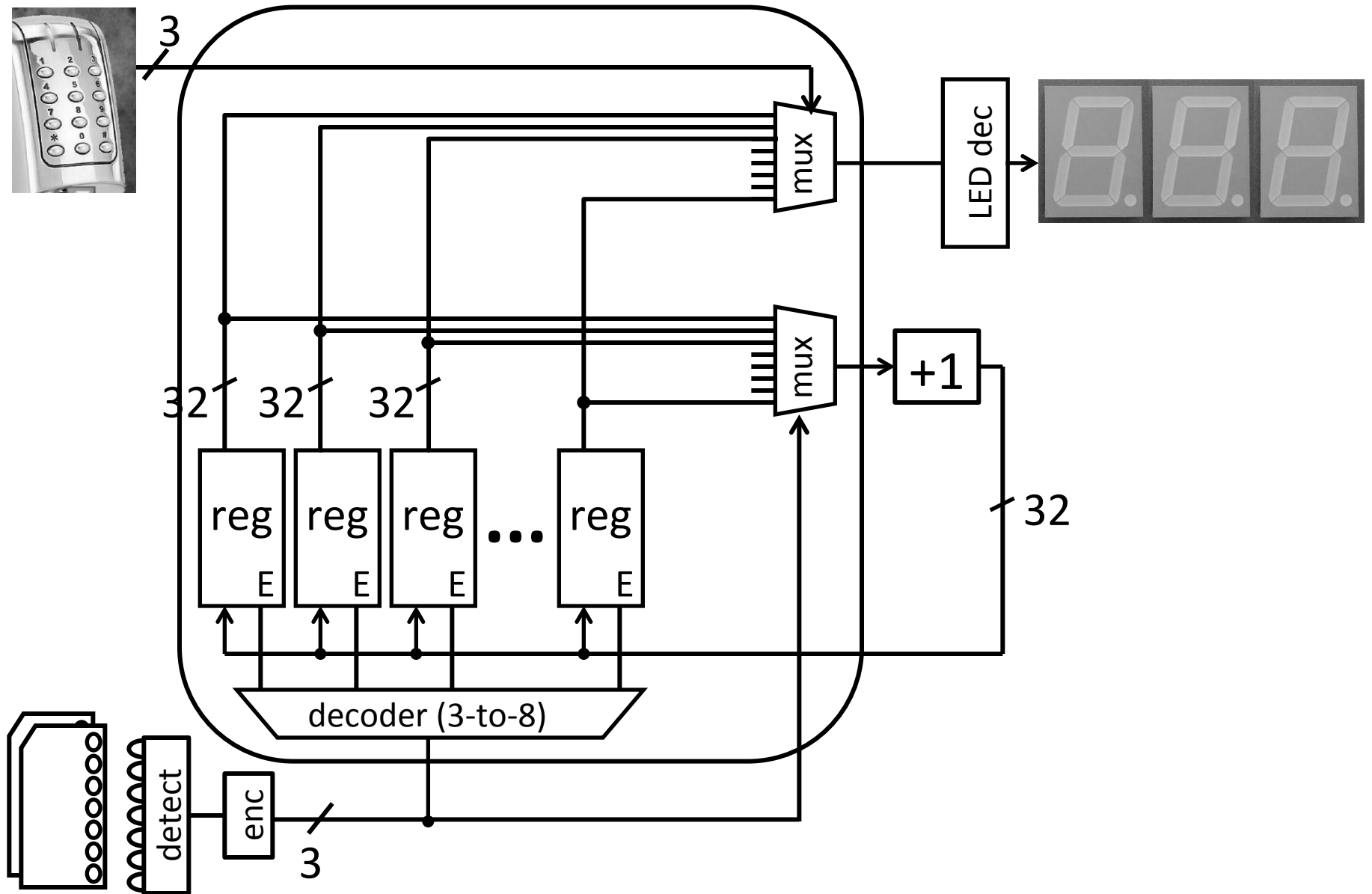
Where D_W, Q_A, Q_B are the 32-bit data ports and W, R_W, R_A, R_B are the address/control ports (1, 5, 5, 5 bits respectively).

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

Register File

Register File

- N read/write registers
- Indexed by register number

Implementation:

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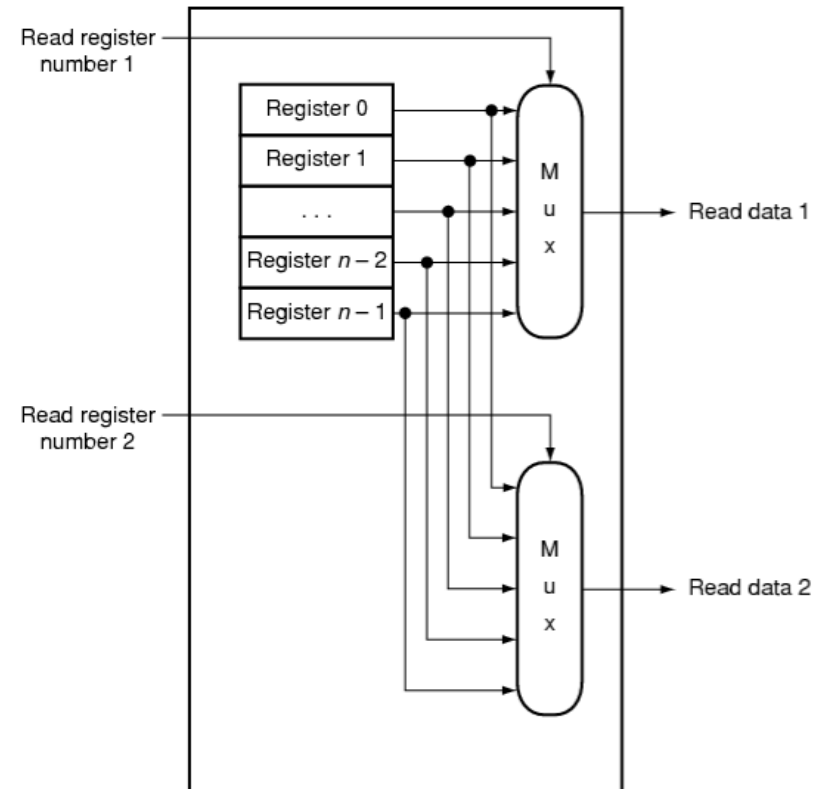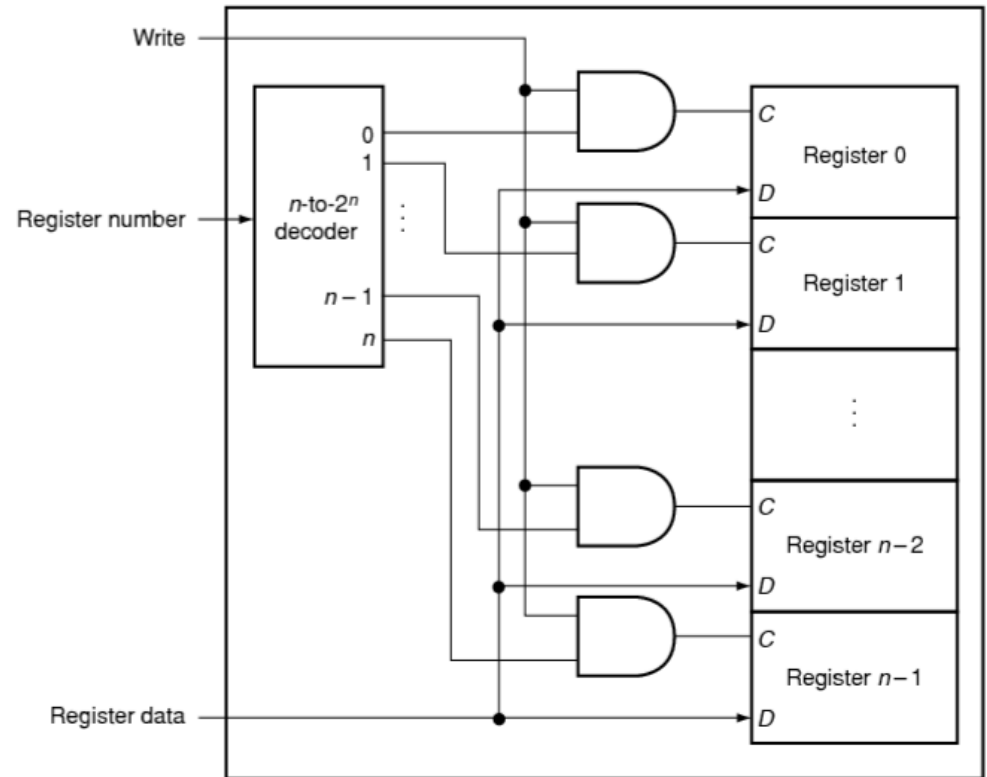- Mux for each read port

Tradeoffs

Register File tradeoffs

+ Very fast (a few gate delays for both read and write)

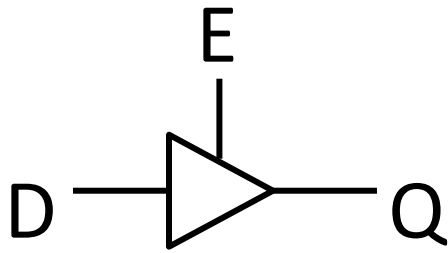+ Adding extra ports is straightforward

– Doesn't scale

Building Large Memories

Need a shared bus (or shared bit line)
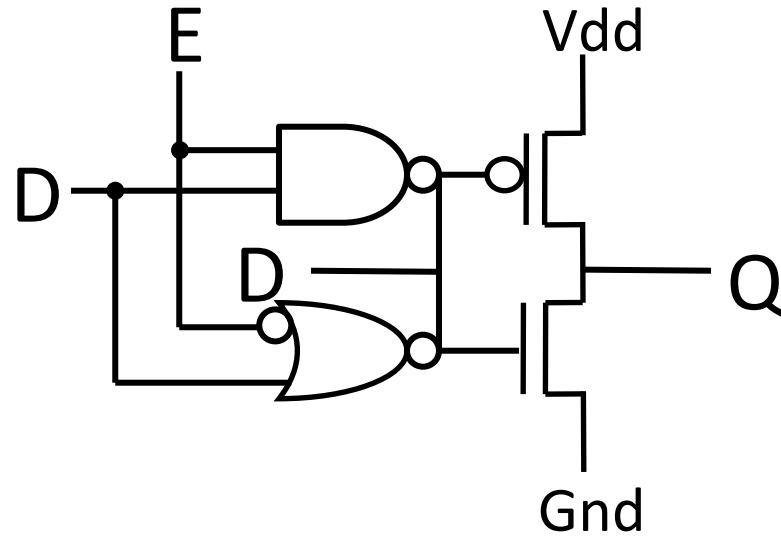
- Many FFs/outputs/etc. connected to single wire
- Only one output *drives* the bus at a time

Tri-State Devices

Tri-State Buffers

E

D ▷ Q

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

E

D

Vdd

D

Q

Gnd

Shared Bus

D_0 S_0 D_1 S_1 D_2 S_2 D_3 S_3 D_{1023} S_{1023}
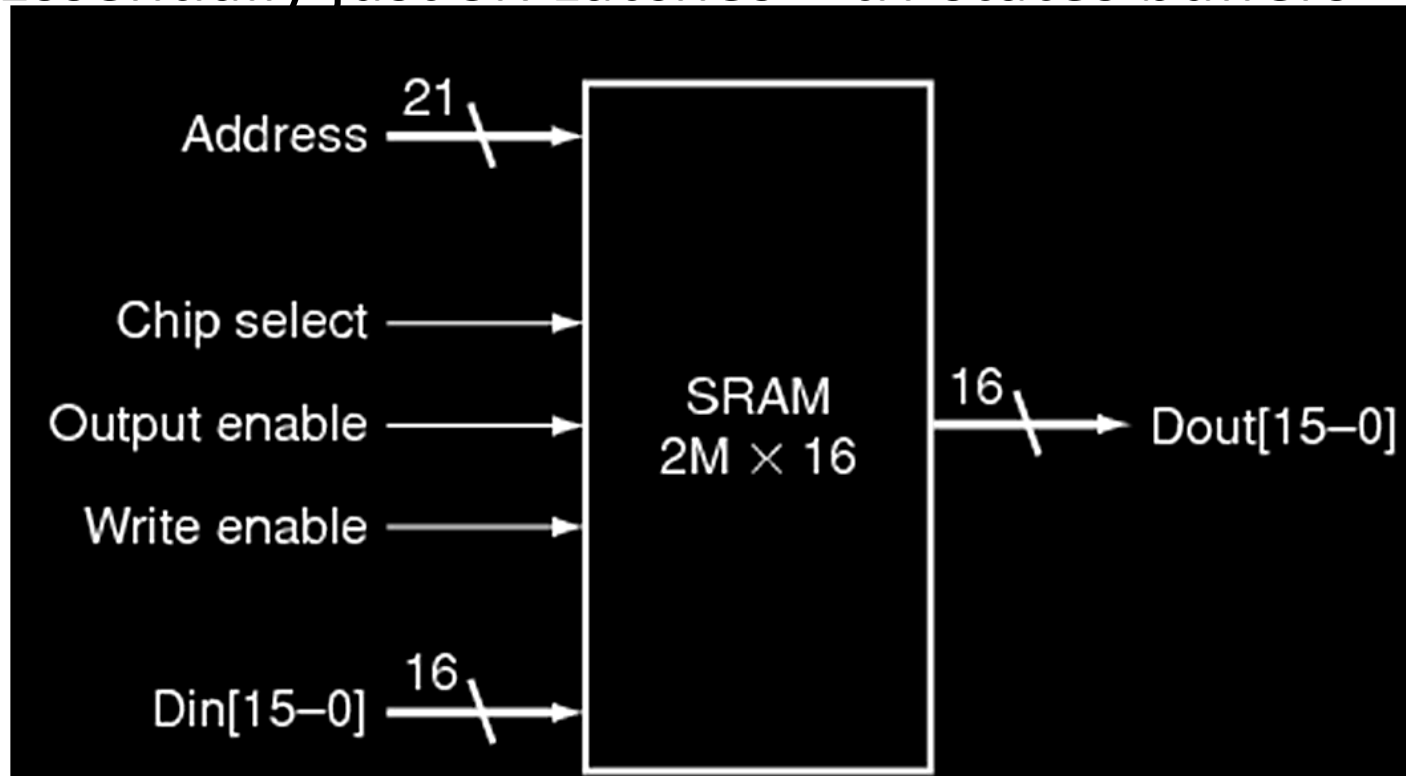
o o o

shared line
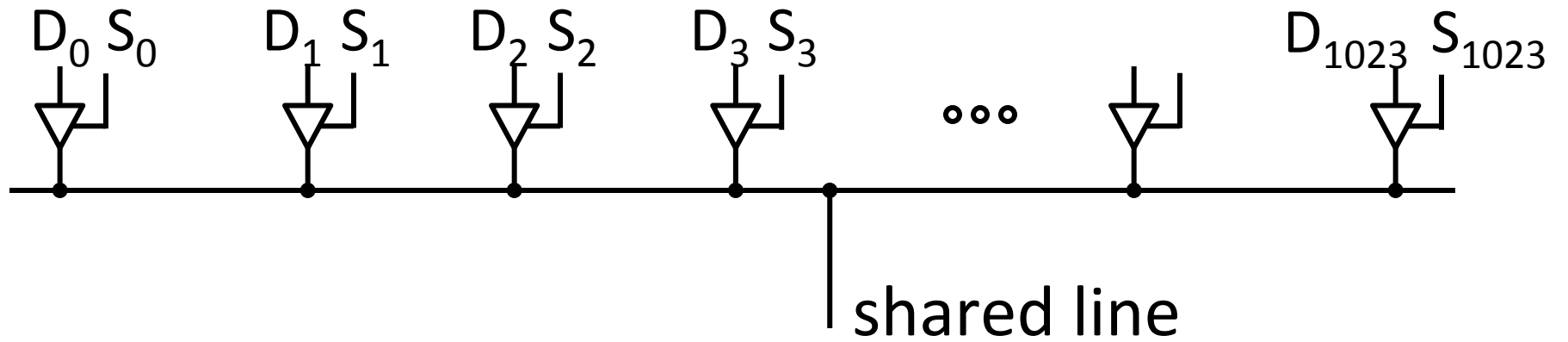
SRAM

Static RAM (SRAM)

- Essentially just SR Latches + tri-states buffers

SRAM Chip
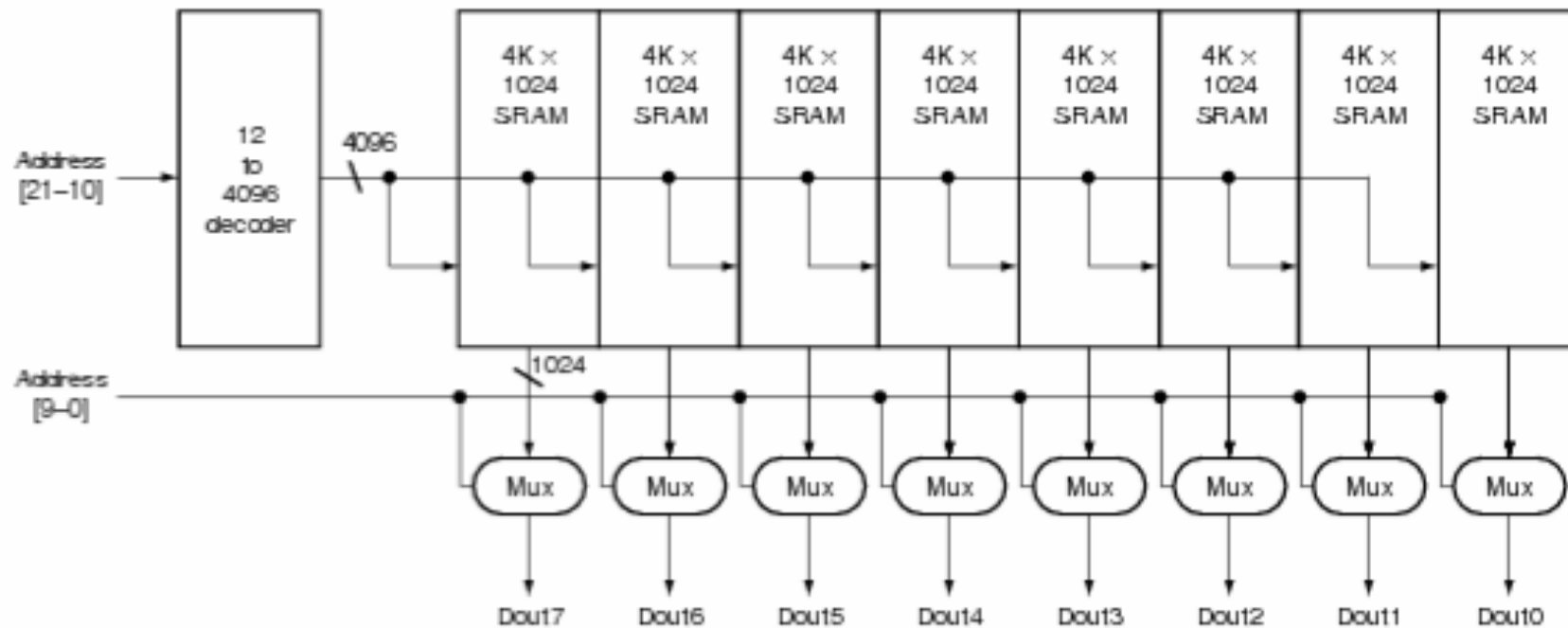
FIGURE B.9.4   Typical organization of a 4M x 8 SRAM as an array of 4K x 1024 arrays. The first decoder generates the addresses for eight 4K x 1024 arrays; then a set of multiplexors is used to select 1 bit from each 1024-bit-wide array. This is a much easier design than a single-level decode that would need either an enormous decoder or a gigantic multiplexor. In practice, a modern SRAM of this size would probably use an even larger number of blocks, each somewhat smaller.

SRAM Chip

row decoder

A_{21-10}

A_{9-0}

column selector, sense amp, and I/O circuits

CS

R/W

Shared Data Bus
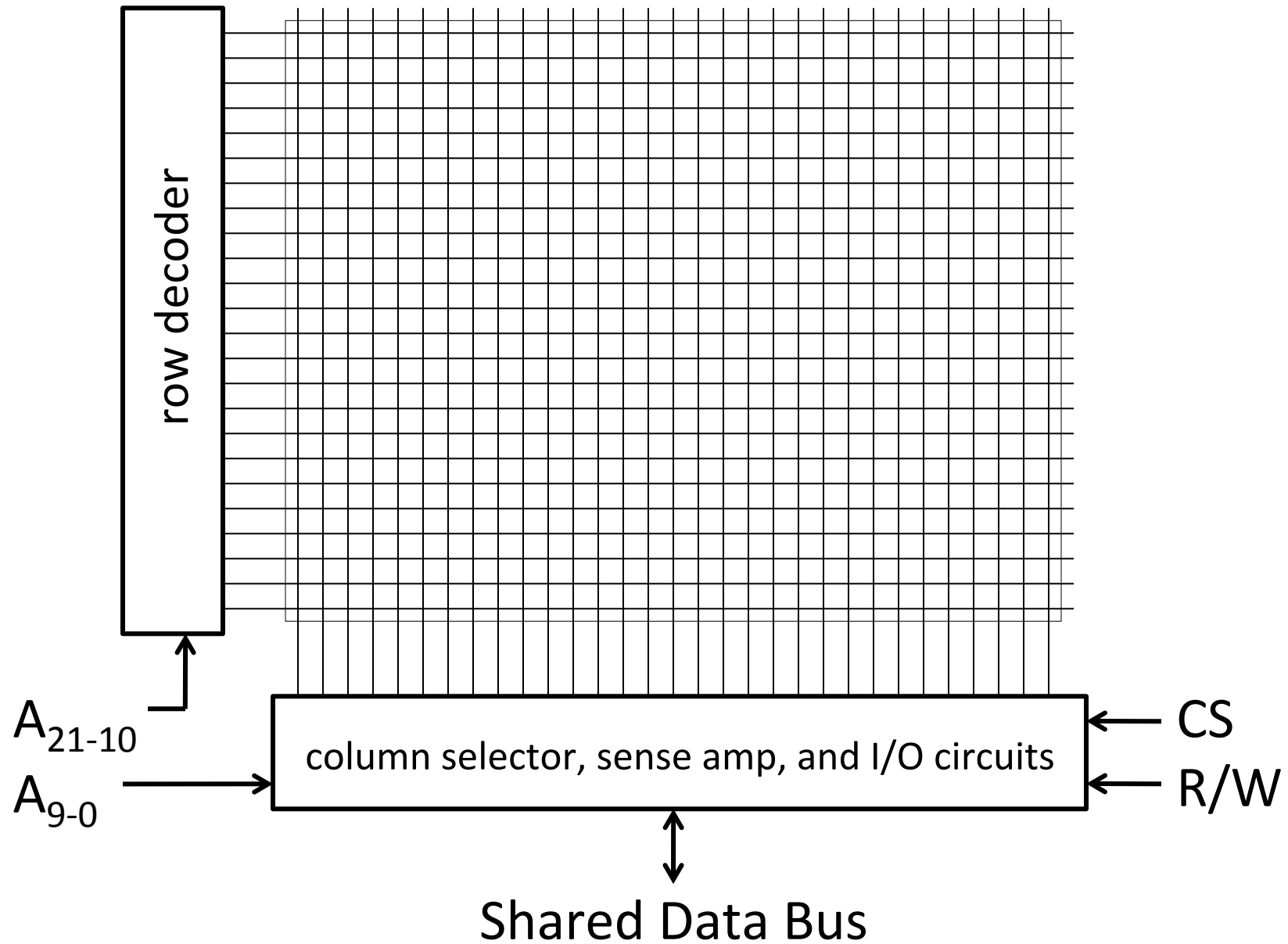
SRAM Cell

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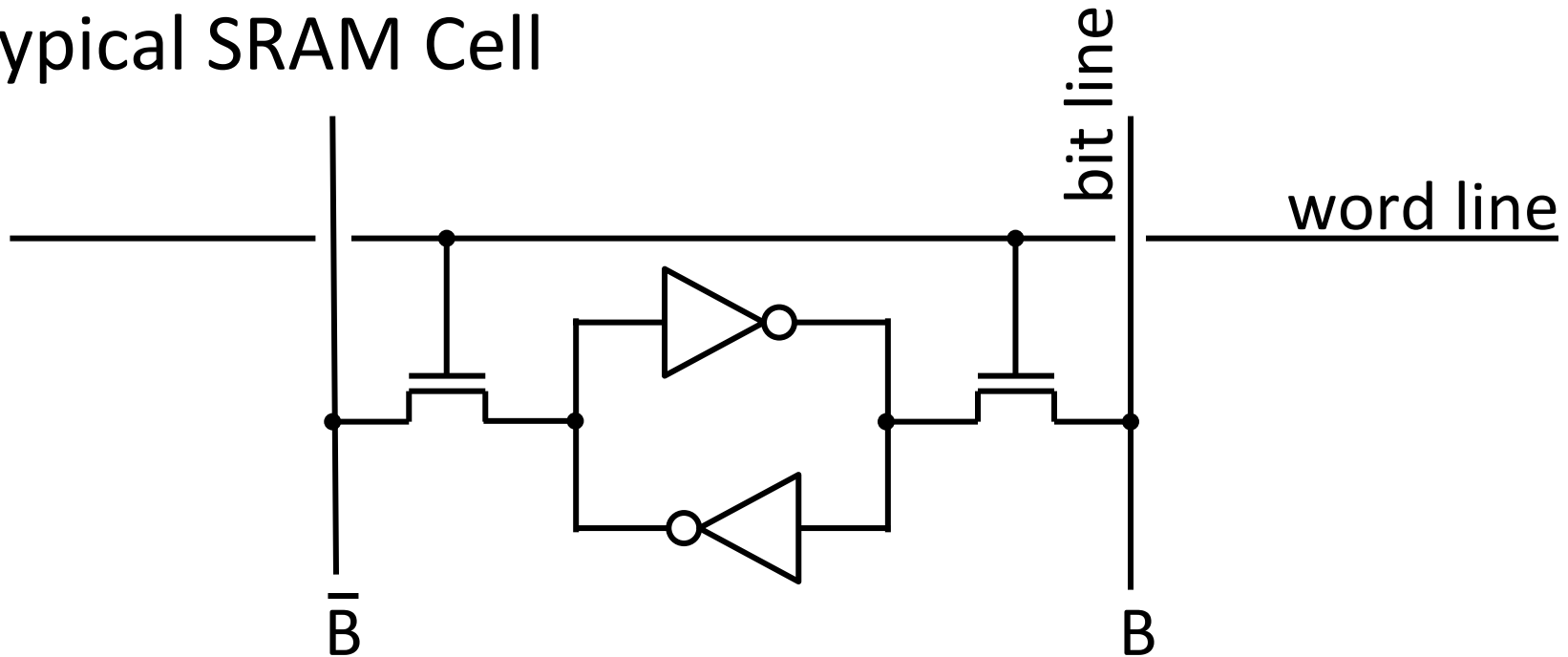
Each cell stores one bit, and requires 4 – 8 transistors (6 is typical)

Read:
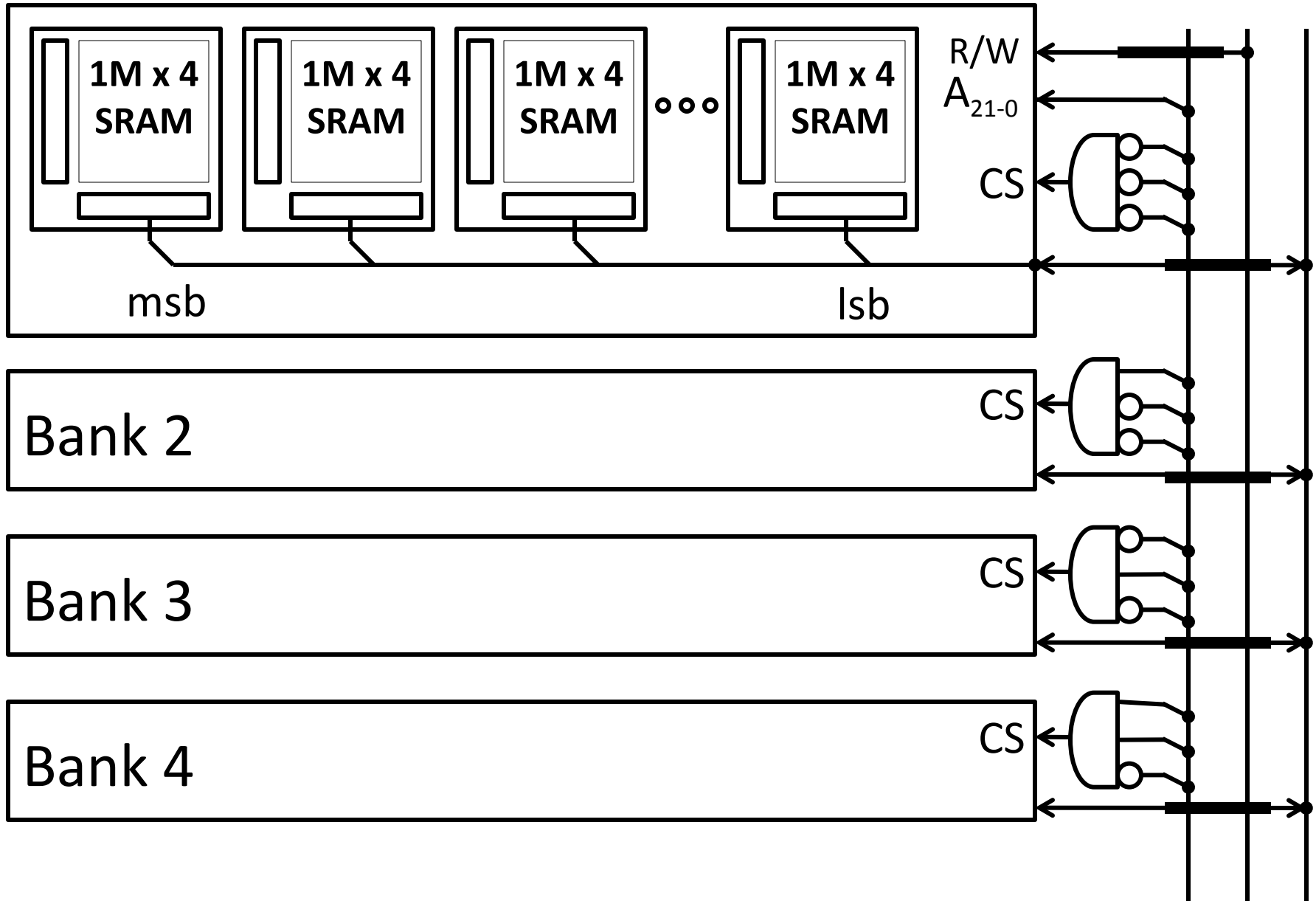- pre-charge B and $\overline{B}$ to Vdd/2
- pull word line high
- cell pulls B or $\overline{B}$ low, sense amp detects voltage difference

Write:
- pull word line high
- drive B and $\overline{B}$ to flip cell

SRAM Modules and Arrays

| 1M x 4 SRAM | 1M x 4 SRAM | 1M x 4 SRAM | ○○○ | 1M x 4 SRAM |

R/W

A_{21-0}

CS

msb   lsb
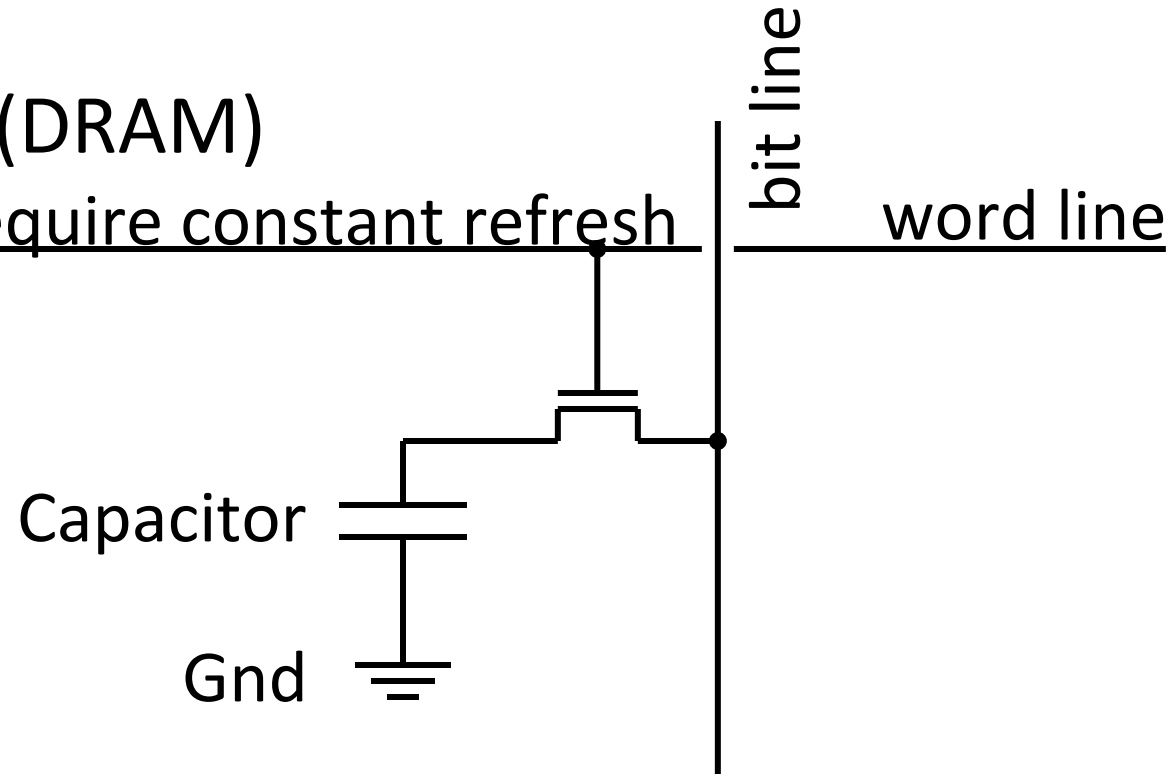
Bank 2

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)
- Data values require constant refresh

bit line

word line

Capacitor

Gnd

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, ...)