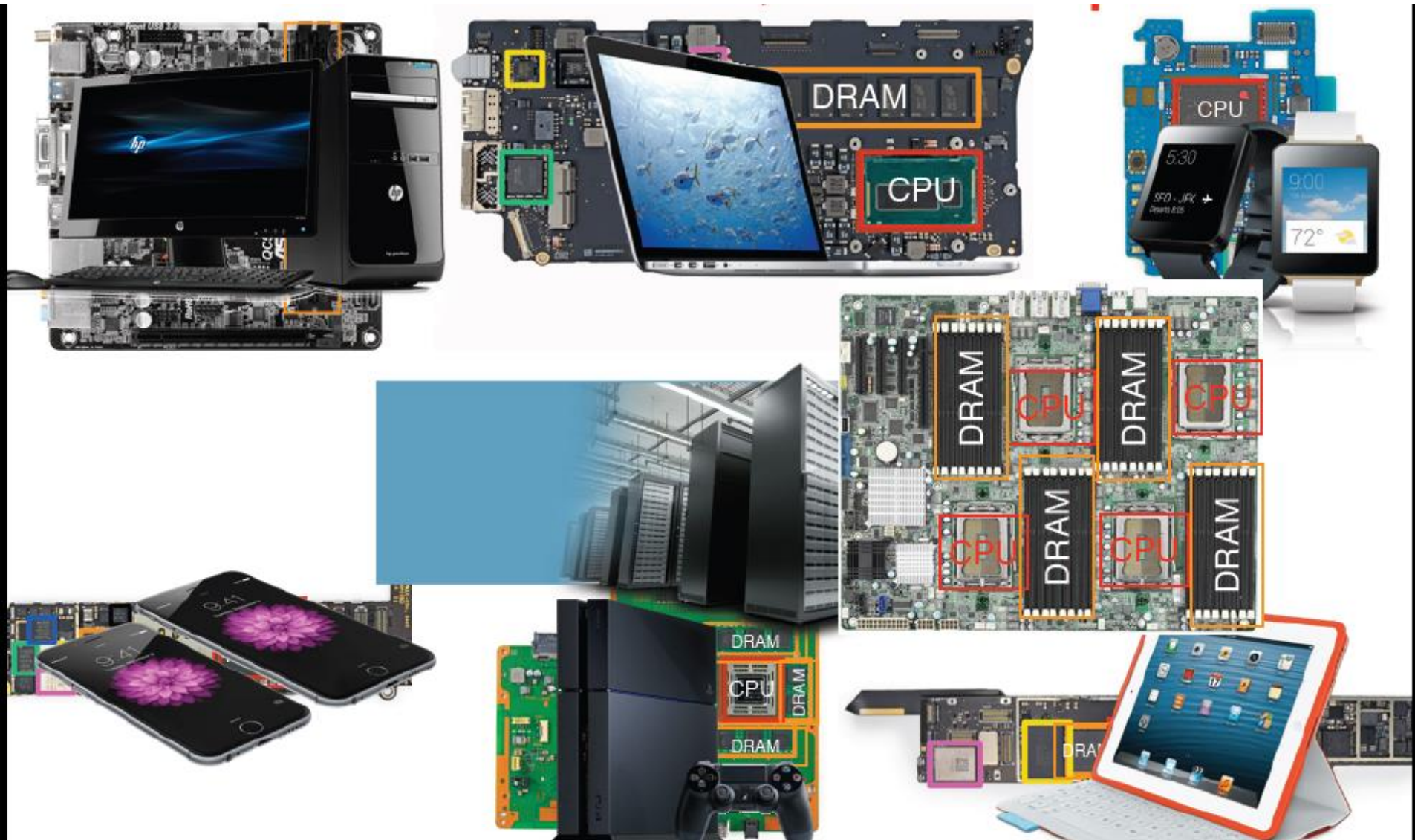
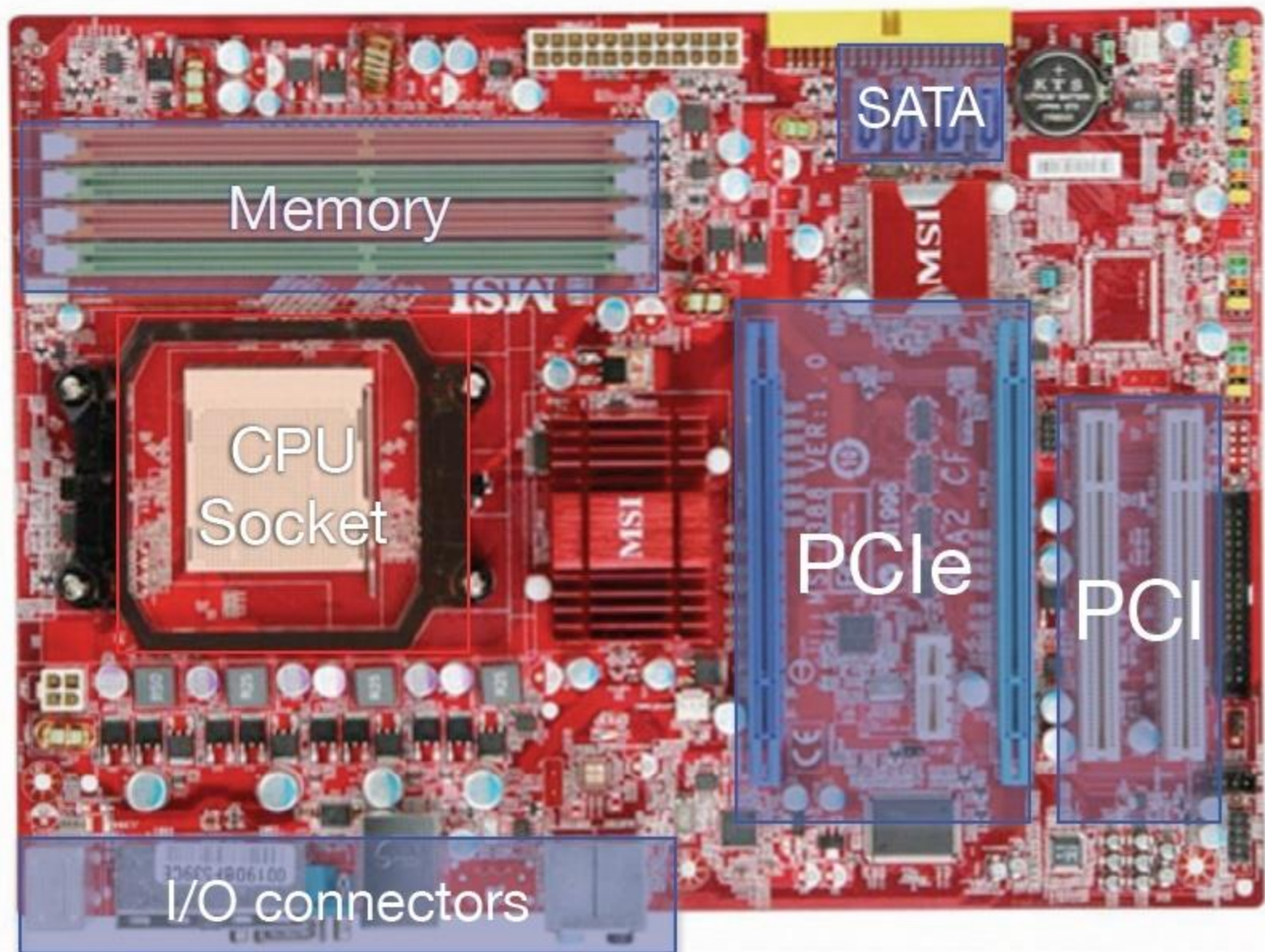


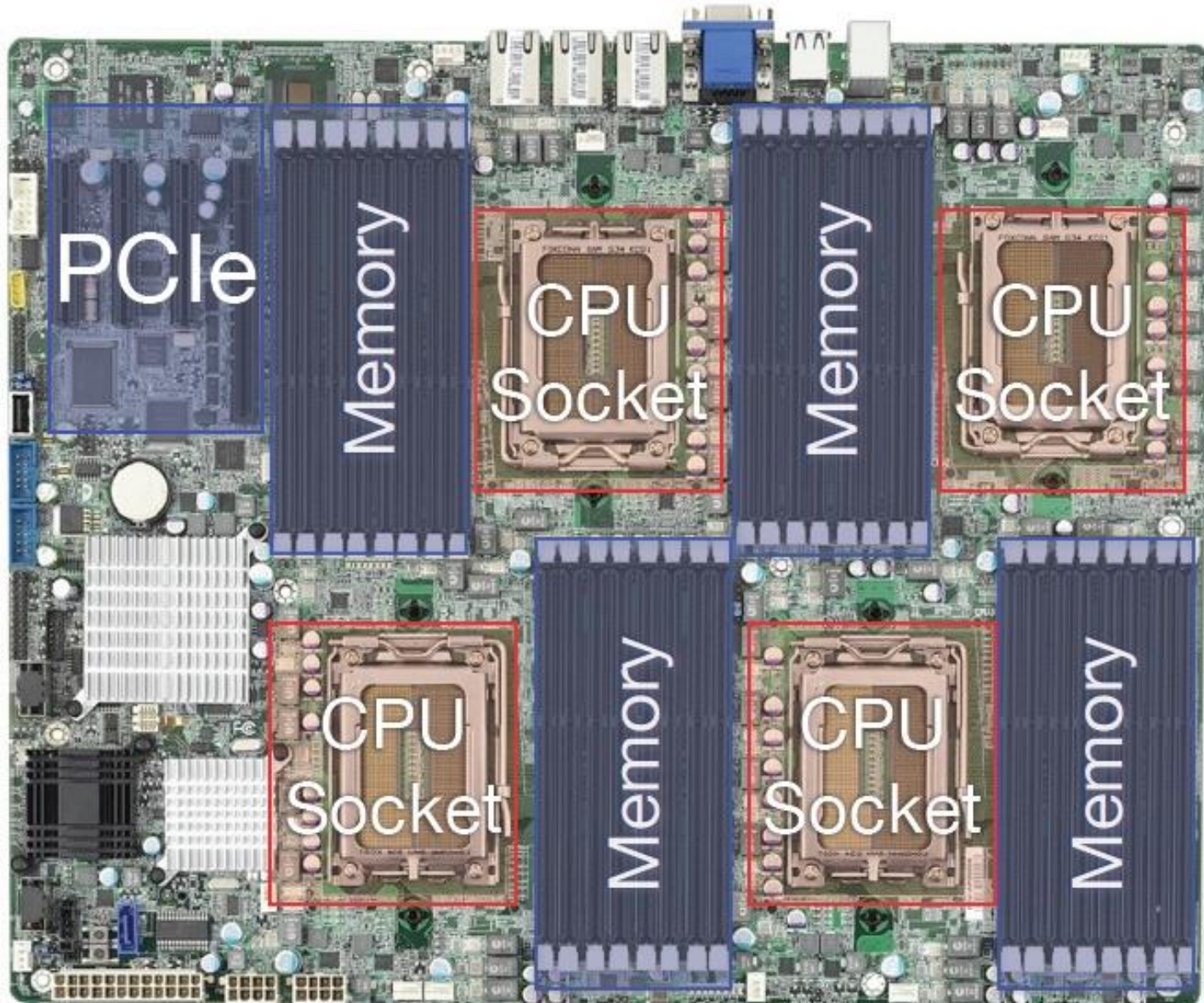
Systems



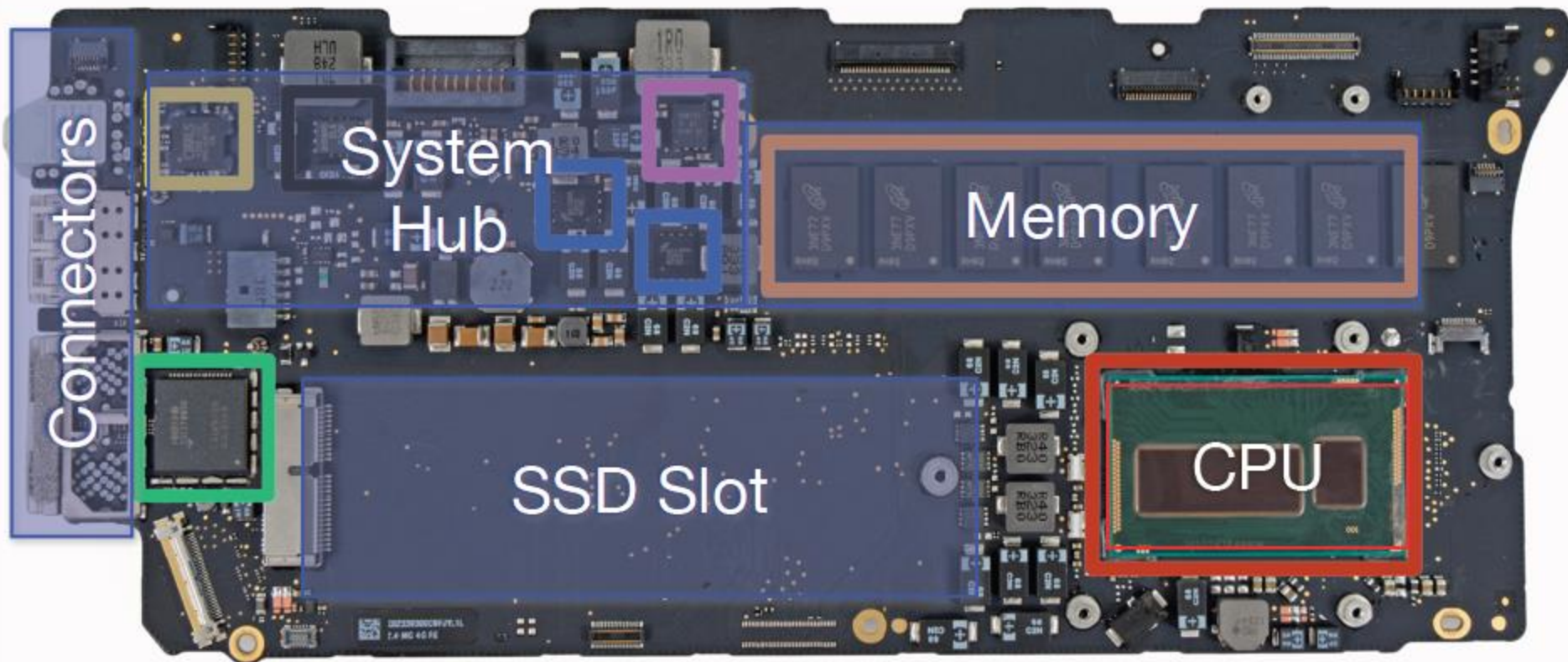
Desktop PC



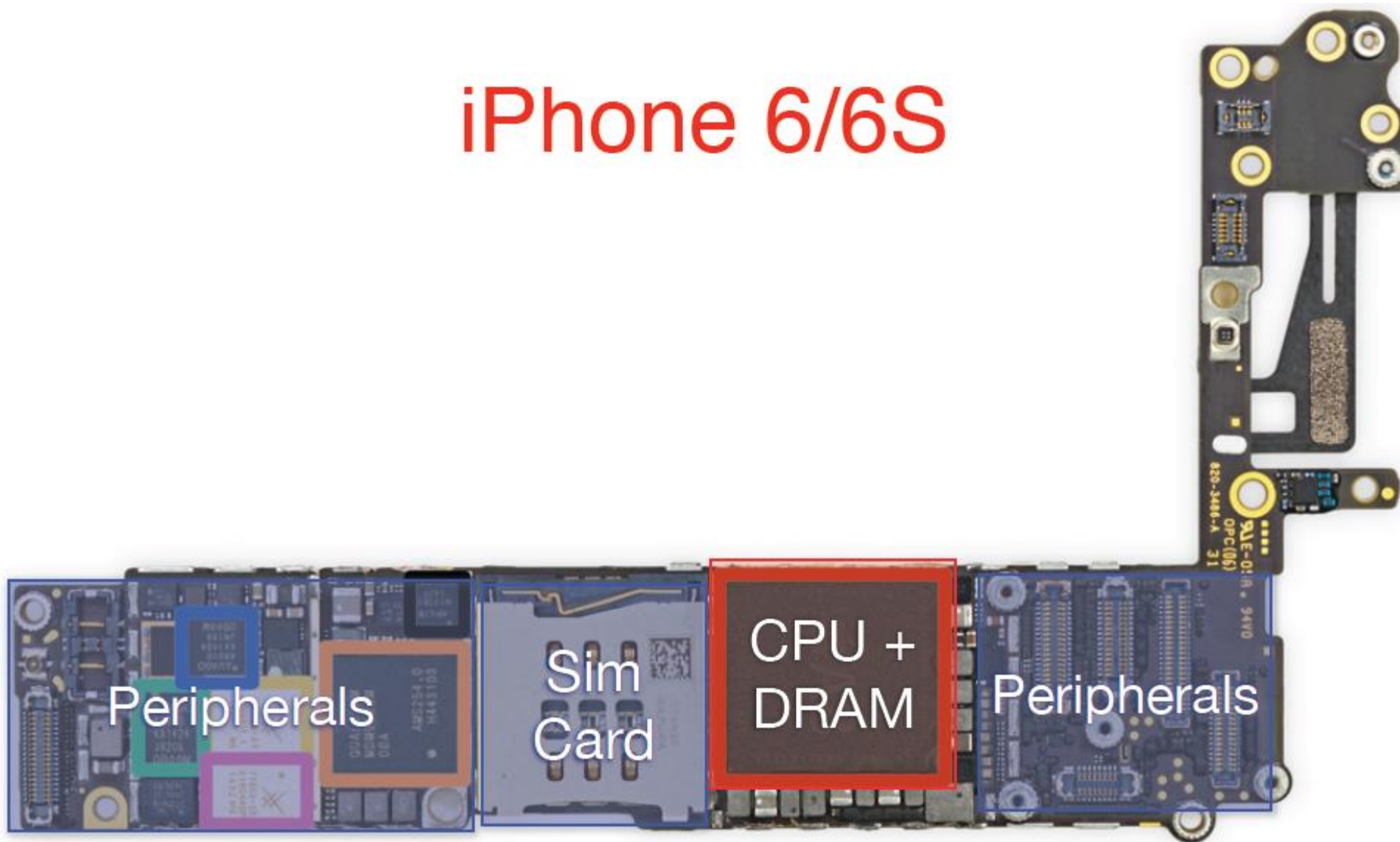
Server



Macbook Pro w/ Retina



iPhone 6/6S

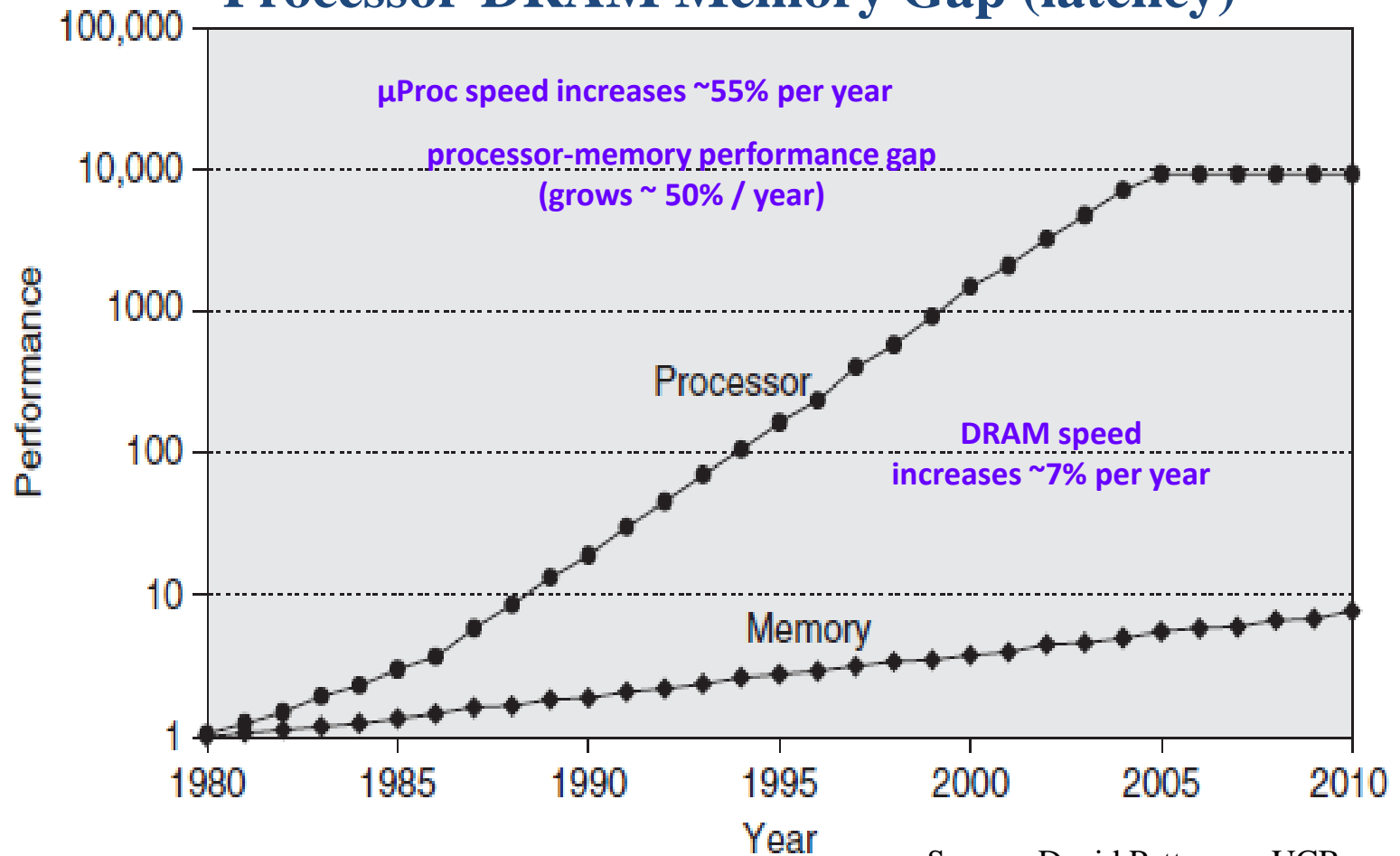


Memory in a Modern System

Overview

- **Review of Memory Technologies**
- **Overview of Memory Hierarchy**
- **Cache Design Principles**
- **Learning Objectives**
 - Why is that some memories slow ?
 - What is memory hierarchy?
 - Why do we need memory hierarchy?
 - What is a Cache?

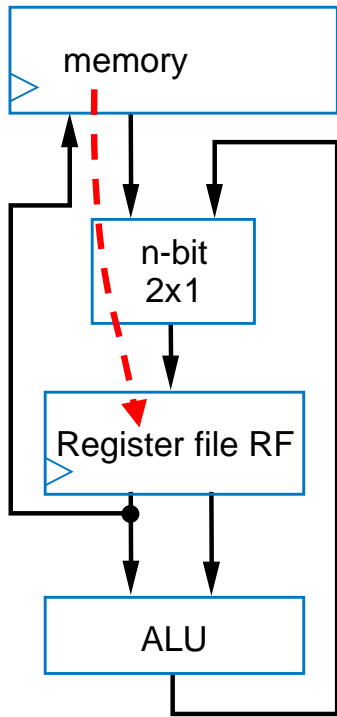
Processor-DRAM Memory Gap (latency)



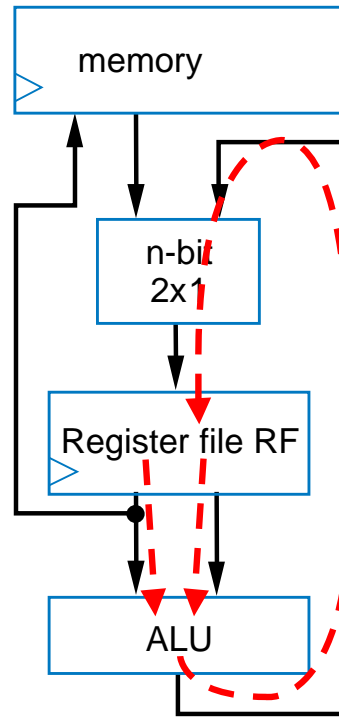
Source: David Patterson, UCB

Review: Datapath Operations

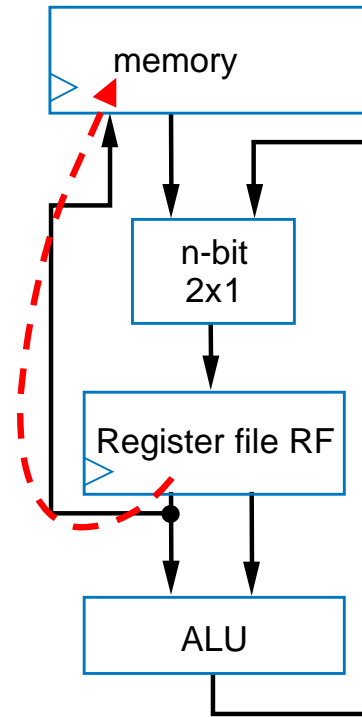
- Load operation: Load data from data memory to RF
- ALU operation: Transforms data by passing one or two RF register values through ALU, performing operation (ADD, SUB, AND, OR, etc.), and writing back into RF.
- Store operation: Stores RF register value back into data memory
- Each operation can be done in one clock cycle



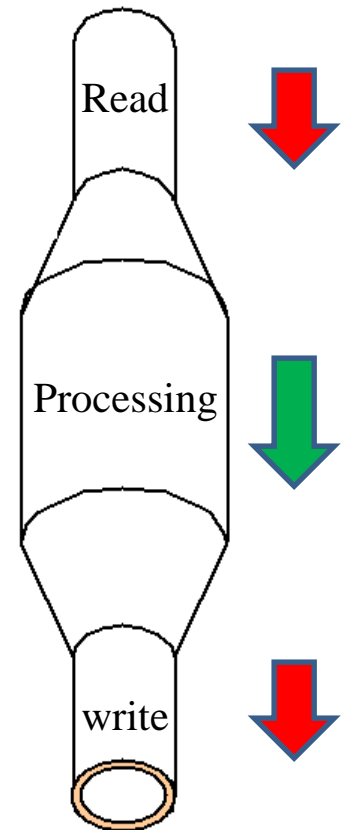
Load operation



ALU operation

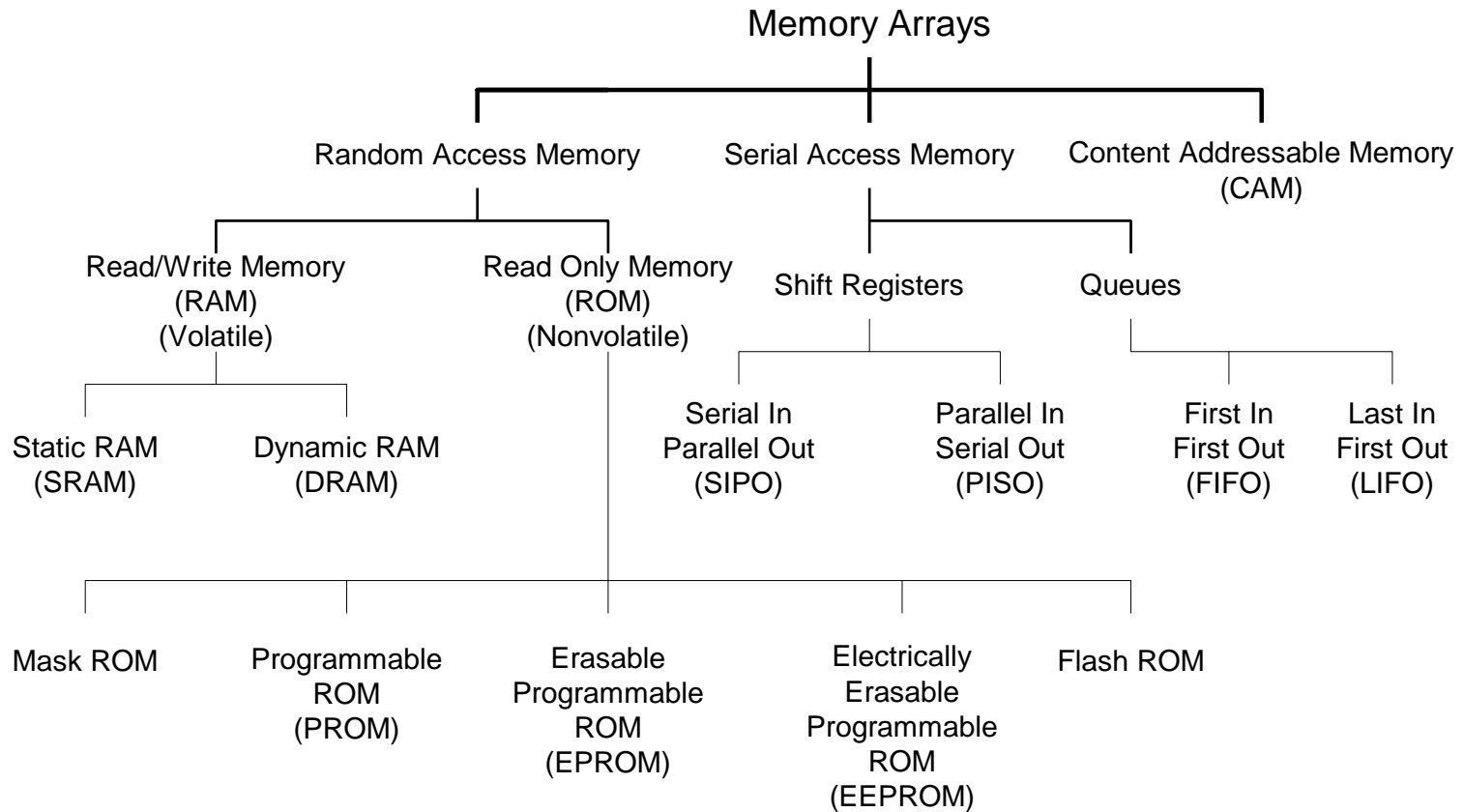


Store operation



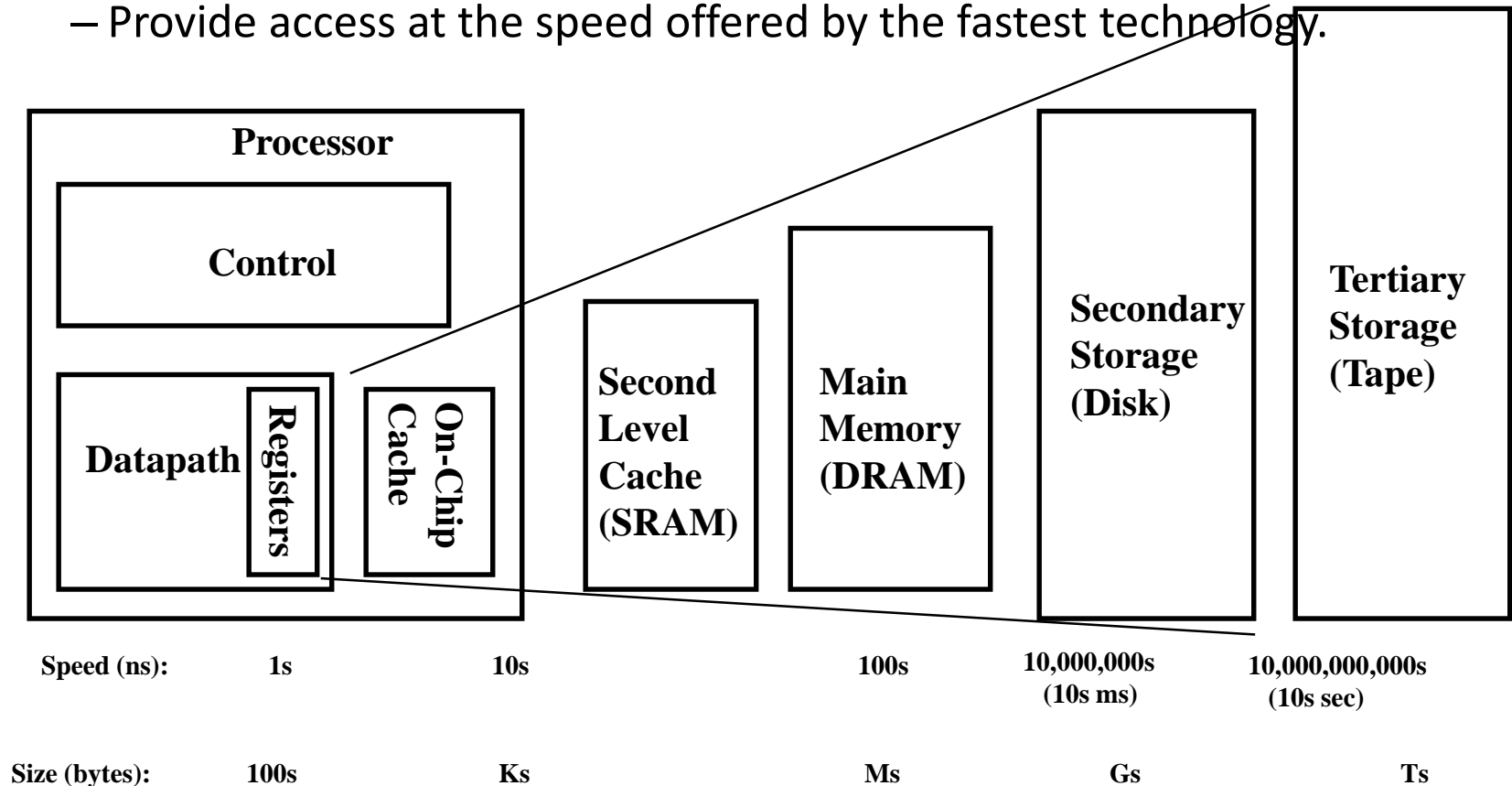
What are the Common Memory Technologies?

Memory



Memory Hierarchy of a Modern Computer System

- By taking advantage of the principle of locality:
 - Present the user with as much memory as is available in the cheapest technology.
 - Provide access at the speed offered by the fastest technology.

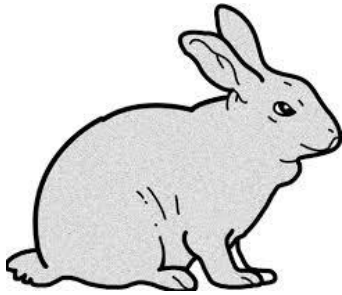


Memory

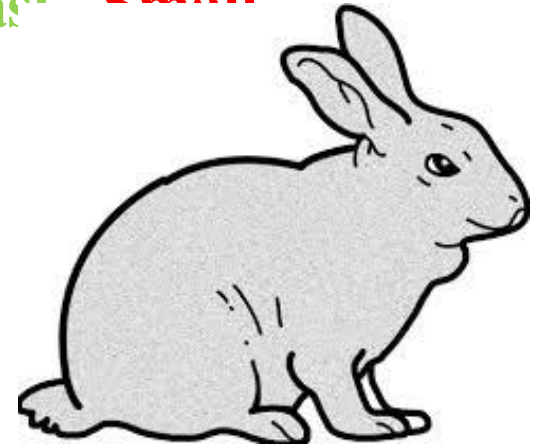
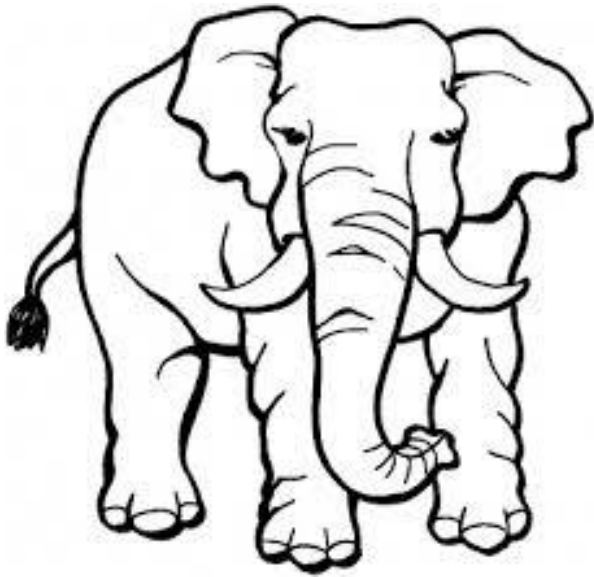
| Address | Data |
|---------|----------|
| 000000 | 00111110 |
| 000001 | 01101011 |
| 000010 | 01011101 |
| 000011 | 01100011 |
| 000100 | 00111110 |
| 000101 | 00000000 |
| 000110 | 11111111 |
| 000111 | 01010101 |
| 001000 | 10101010 |
| 001001 | 00100001 |
| 001010 | 11011010 |

| 64x8 RAM | |
|----------|----|
| A3 | D7 |
| A2 | D6 |
| A1 | D5 |
| A0 | D4 |
| | D3 |
| | D2 |
| | D1 |
| Write | D0 |

Actual Memory Systems



+



Type1: Fast ~~Small~~

Can we achieve?: Fast, Large
Type2: ~~Slow~~, Large

(Cache Principle)

8x4 RAM

Address

Data

000



001



010



011



100



101



110



111

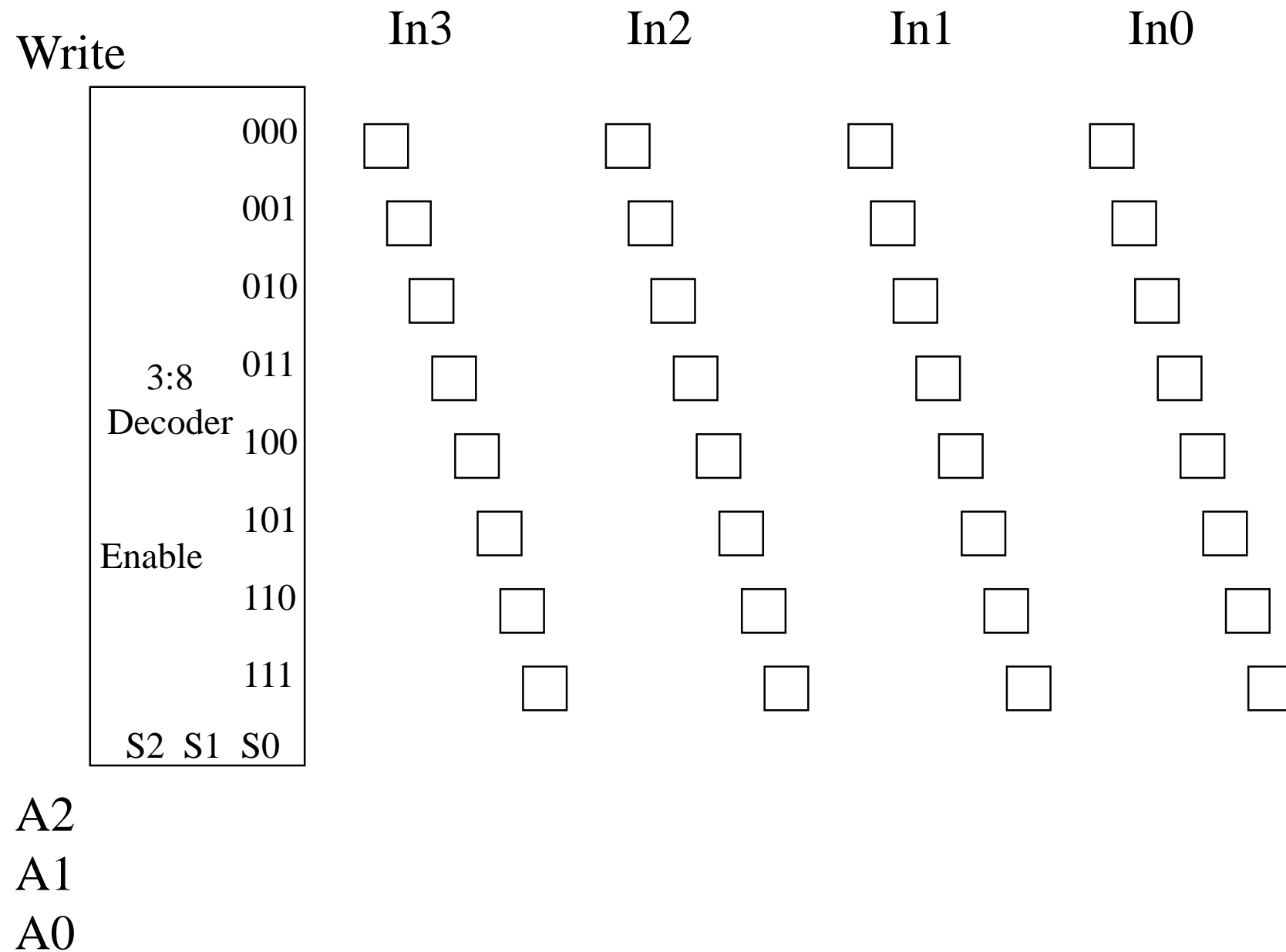


A2

A1

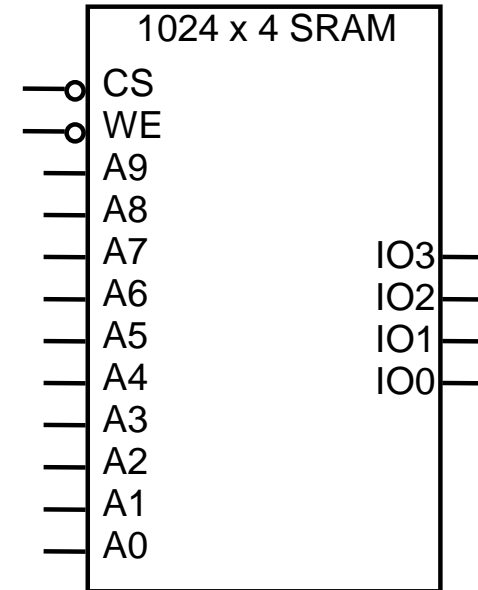
A0

8x4 RAM



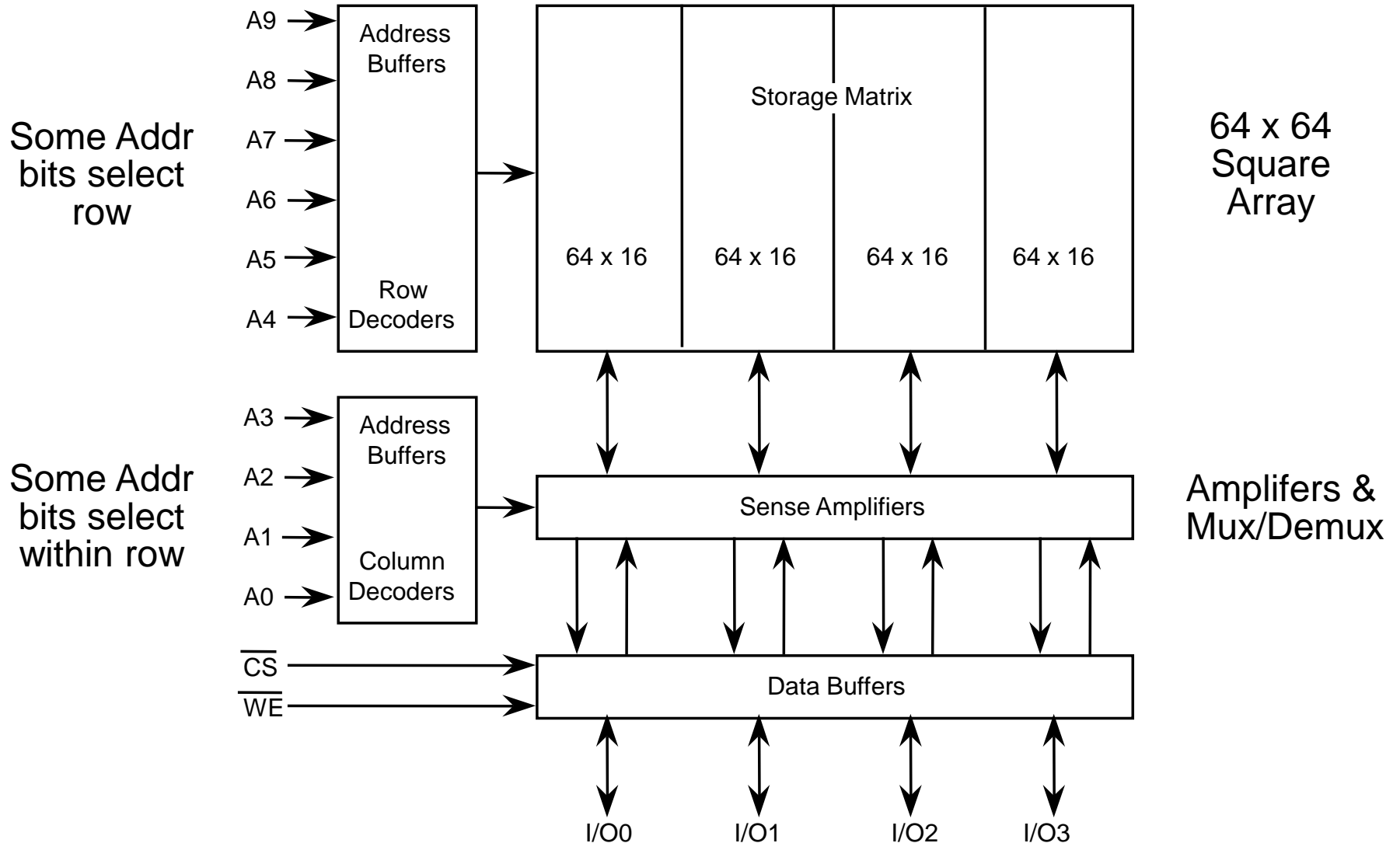
Static RAM Organization

Chip Select Line (active lo)
Write Enable Line (active lo)
10 Address Lines
4 Bidirectional Data Lines

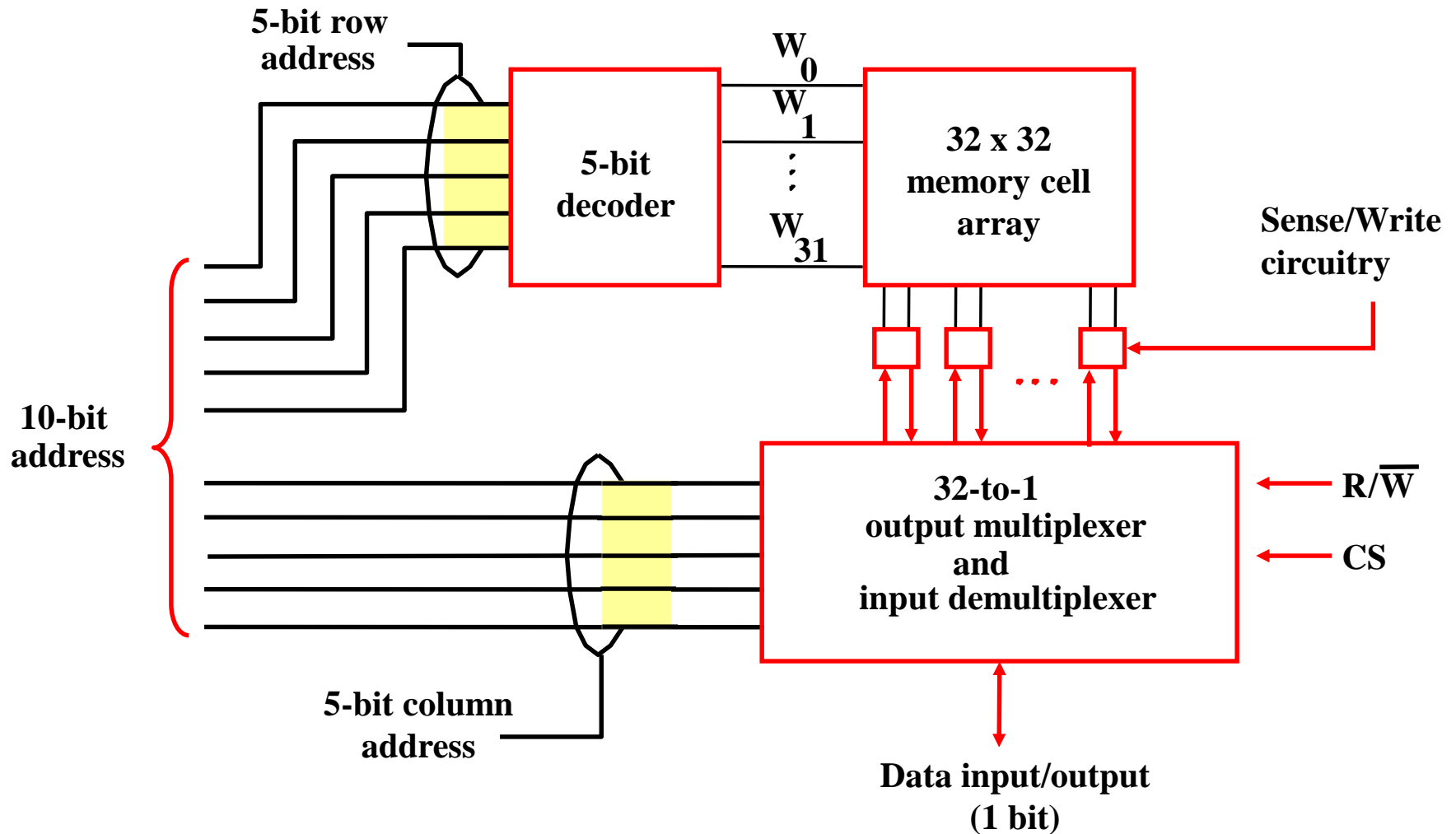


RAM Organization

Long thin layouts are not the best organization for a RAM



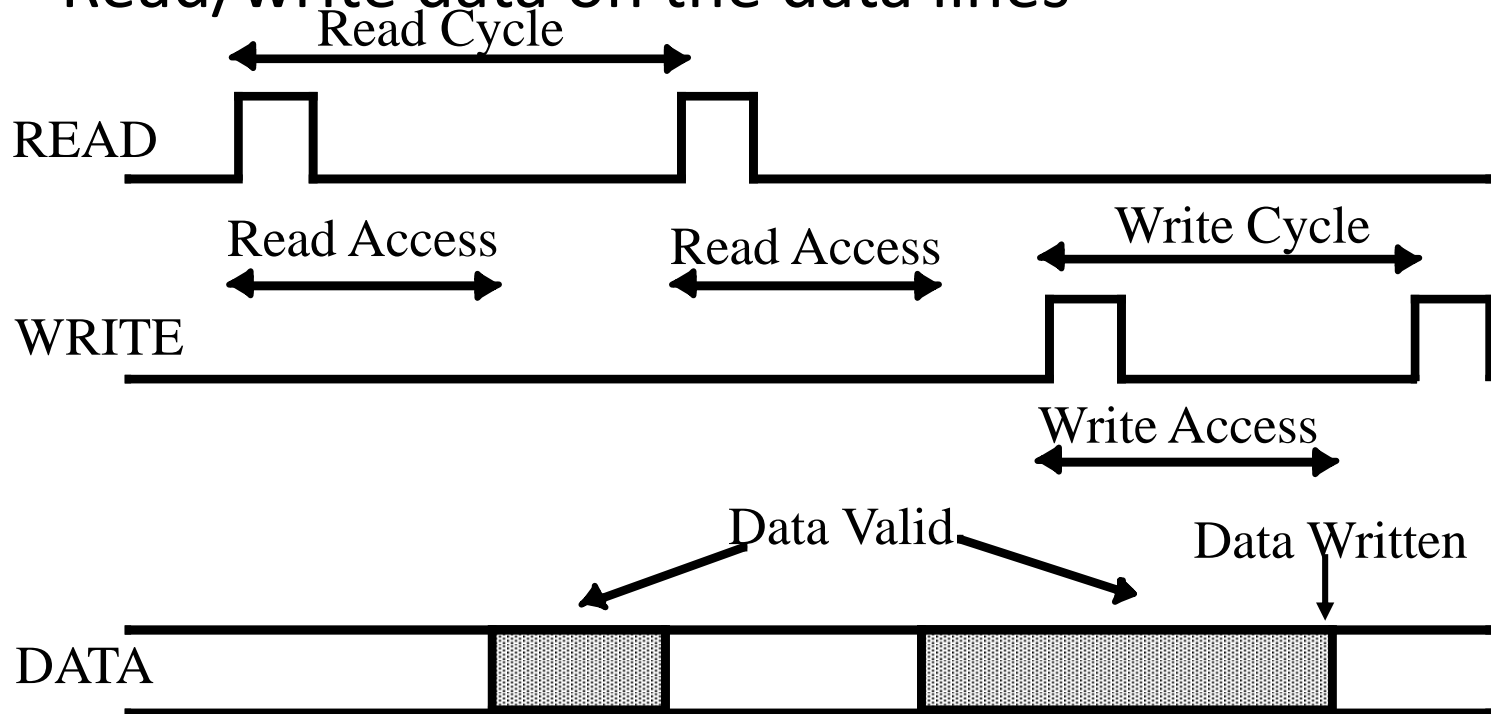
1Kx 1 bit RAM Organization



Organization of a $1K \times 1$ memory chip.

Memory Access Timing: the Big Picture

- Timing:
 - Send address on the address lines, wait for the word line to become stable
 - Read/write data on the data lines



Content of a memory

- Each **word in memory** is assigned an **identification number**, called an **address**, starting **from 0 up to 2^k-1** , where **k** is the **number of address lines**.
- The number of words in a memory with one of the letters $K=2^{10}$, $M=2^{20}$, or $G=2^{30}$.

$$64K = 2^{16} \quad 2M = 2^{21}$$

$$4G = 2^{32}$$

| Memory address | | Memory content |
|----------------|---------|------------------|
| Binary | decimal | |
| 0000000000 | 0 | 1011010101011101 |
| 0000000001 | 1 | 1010101110001001 |
| 0000000010 | 2 | 0000110101000110 |
| | ⋮ | ⋮ |
| 1111111101 | 1021 | 1001110100010100 |
| 1111111110 | 1022 | 0000110100011110 |
| 1111111111 | 1023 | 1101111000100101 |

1024x16 Memory Module

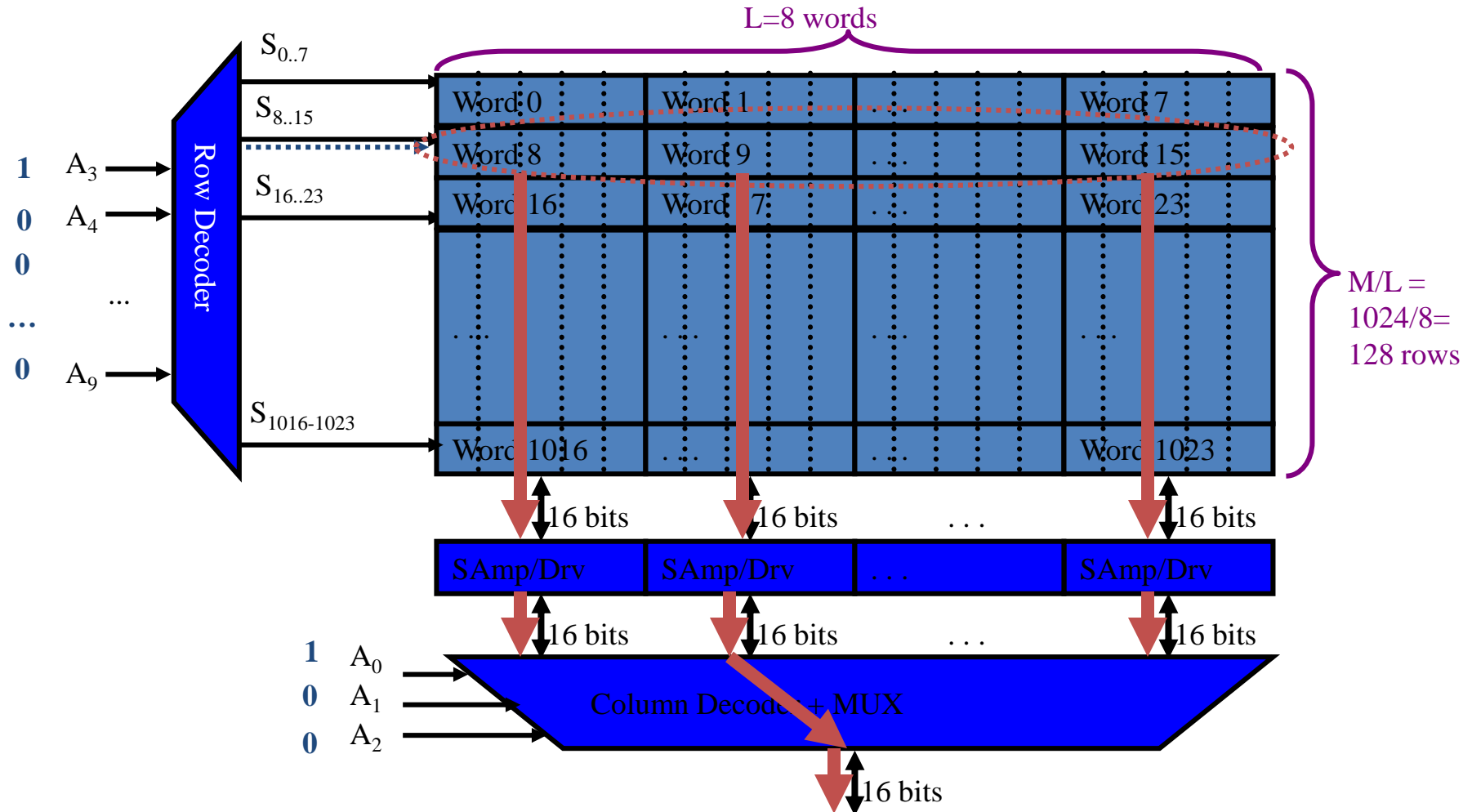
Word-Addressable Memory

- Each 32-bit data word has a unique address

| Word Address | Data | |
|--------------|-----------------|--------|
| ⋮ | ⋮ | ⋮ |
| 00000003 | 4 0 F 3 0 7 8 8 | Word 3 |
| 00000002 | 0 1 E E 2 8 4 2 | Word 2 |
| 00000001 | F 2 F 1 A C 0 7 | Word 1 |
| 00000000 | A B C D E F 7 8 | Word 0 |

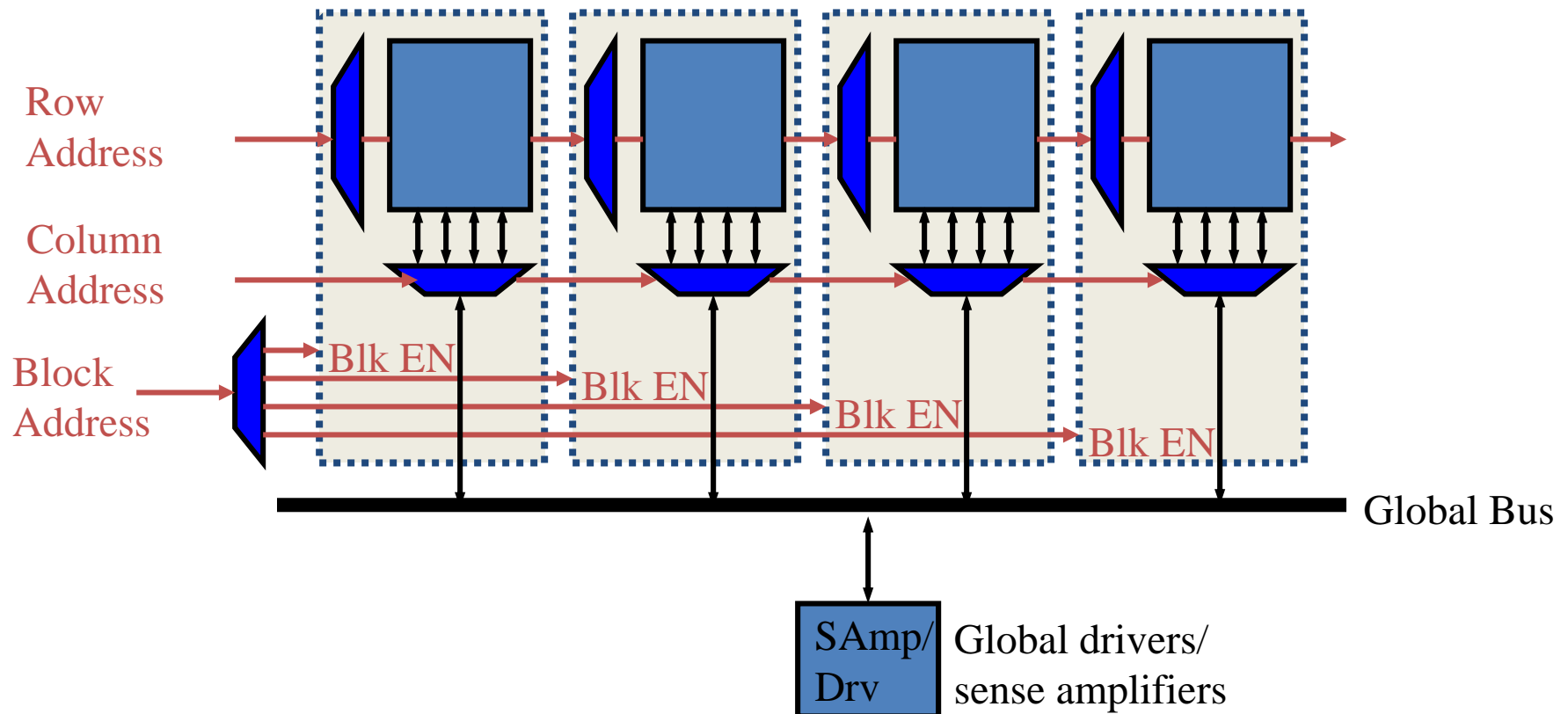
Memory Cell Array Access Example

- word=16-bit wide(N), row=8 words(L), address=10 bits (k)
- Accessing word 9= 0000001001_2

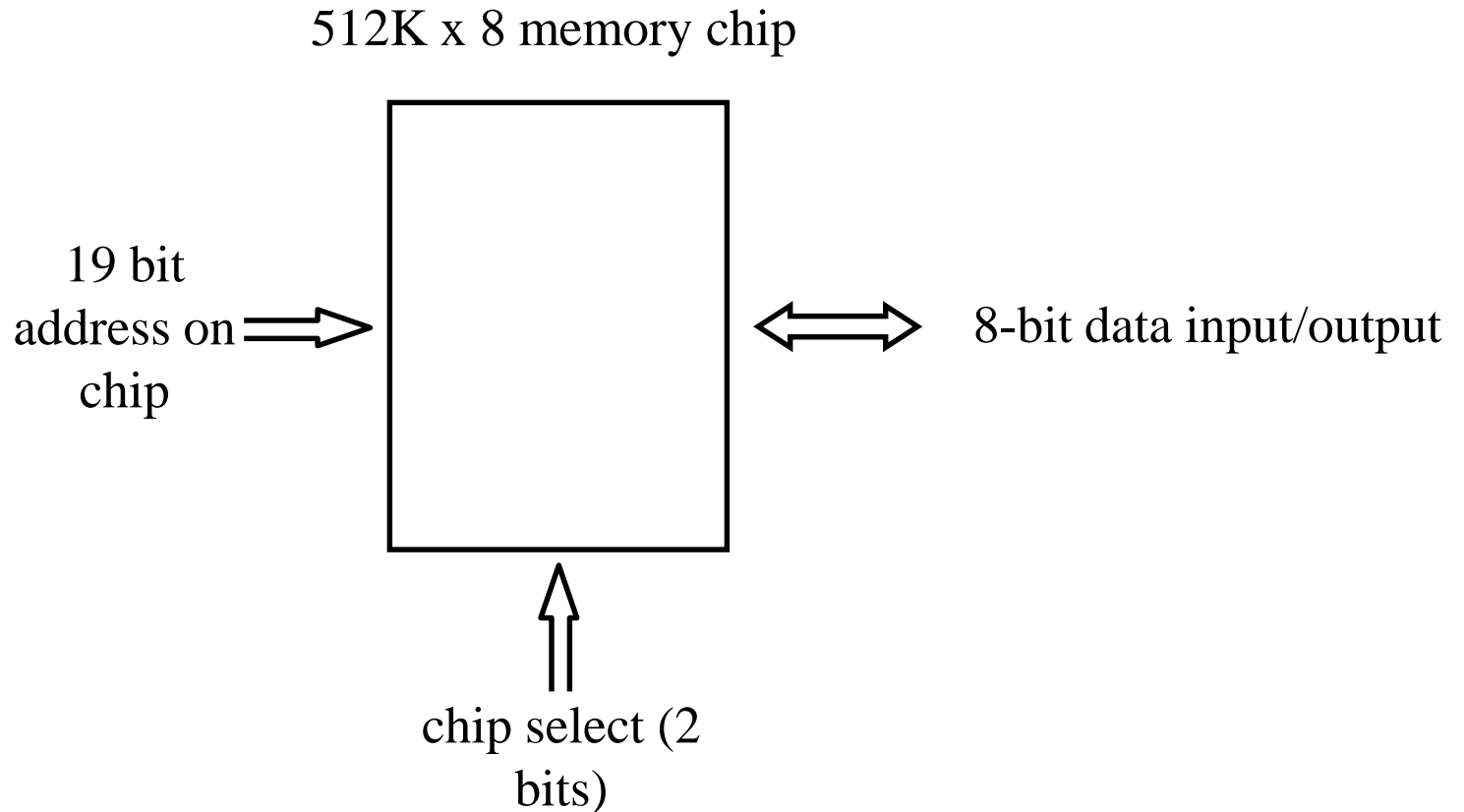


Memory Structures

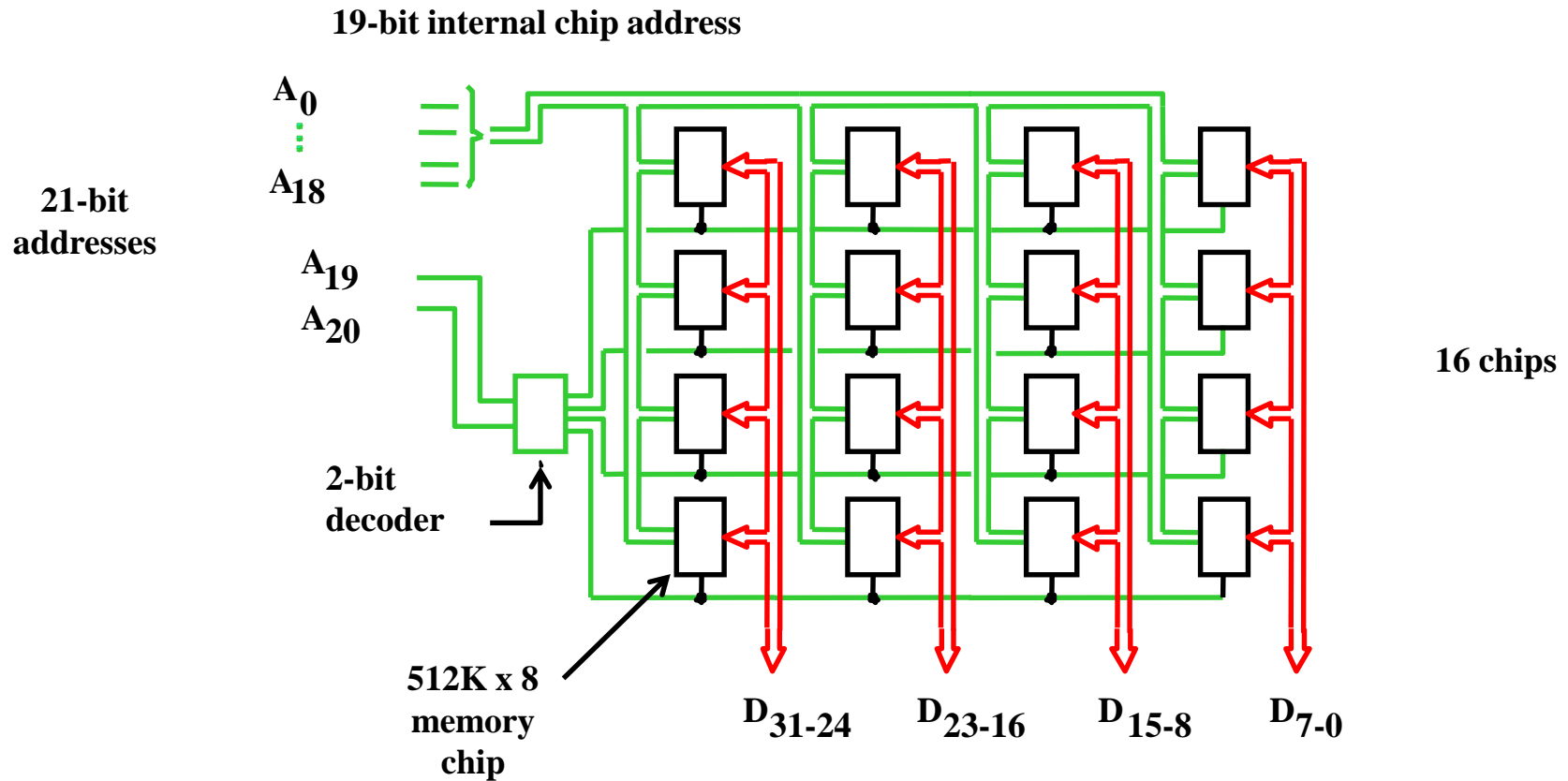
- Taking the idea one step further
 - Shorter wires within each block
 - Enable only one block addr decoder → power savings



Larger Memories Using Multiple Chips

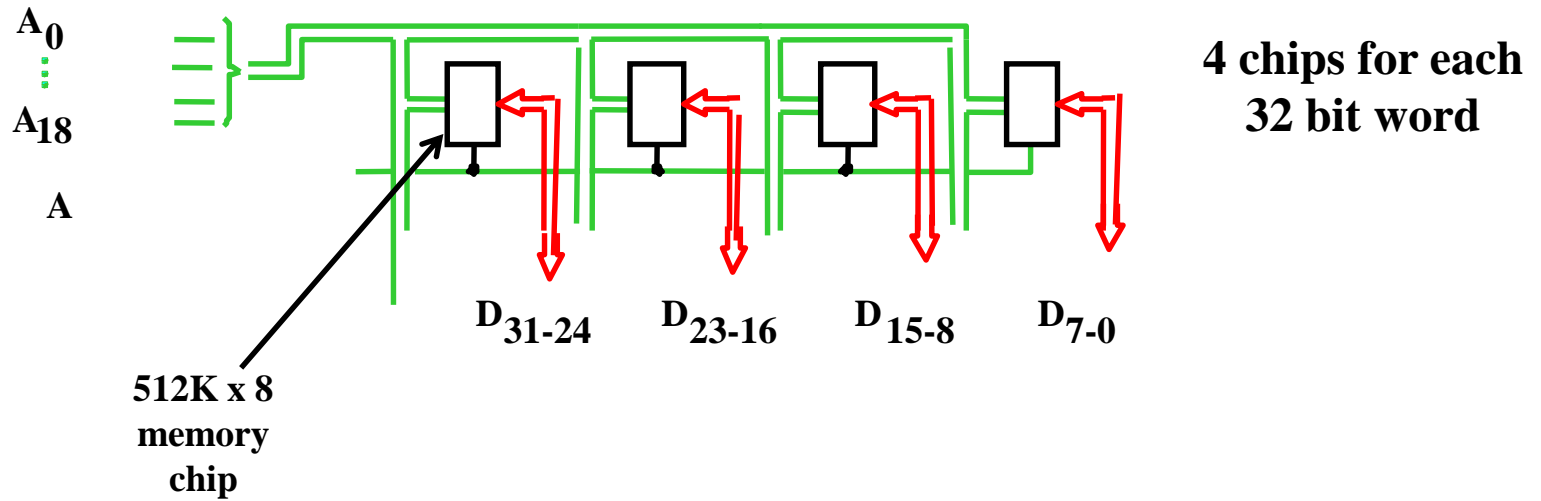


Question ? Design a 2Mx32 given: 512Kx8

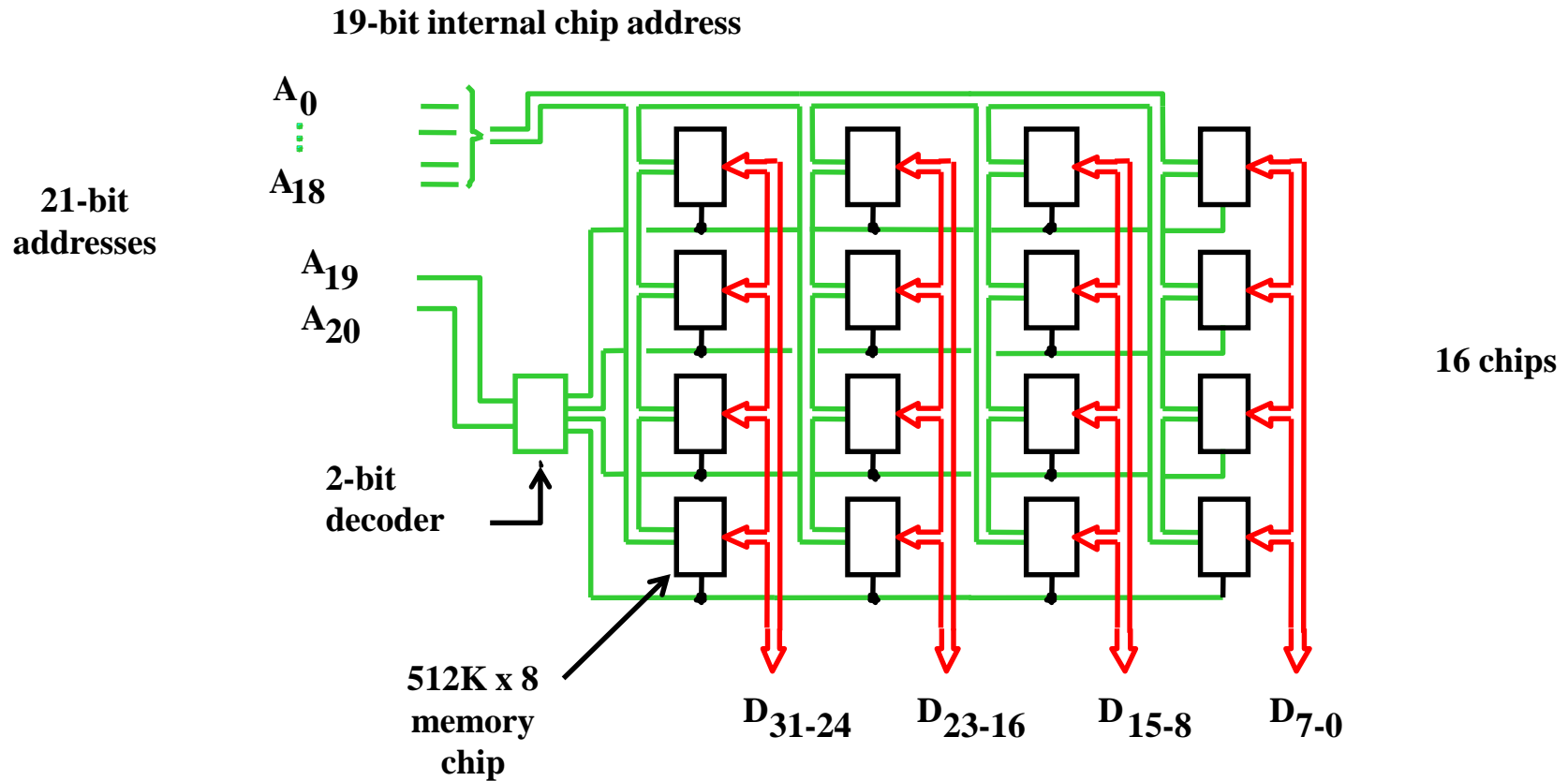


Organization of a $2M \times 32$ memory module using $512K \times 8$ static memory chips (16 chips).

19-bit internal chip address



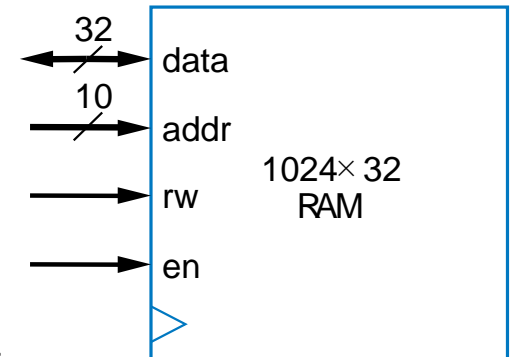
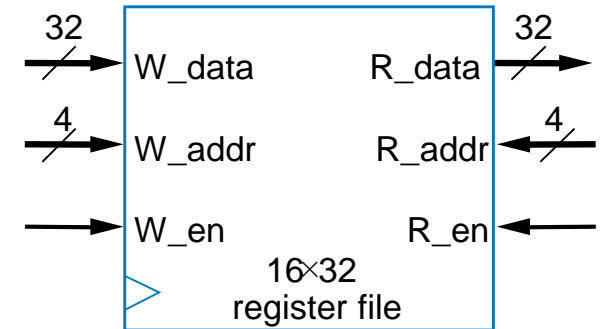
Organization of a $2M \times 32$ memory module using $512K \times 8$ static memory chips (16 chips).



Organization of a $2M \times 32$ memory module using $512K \times 8$ static memory chips (16 chips).

Random Access Memory (RAM)

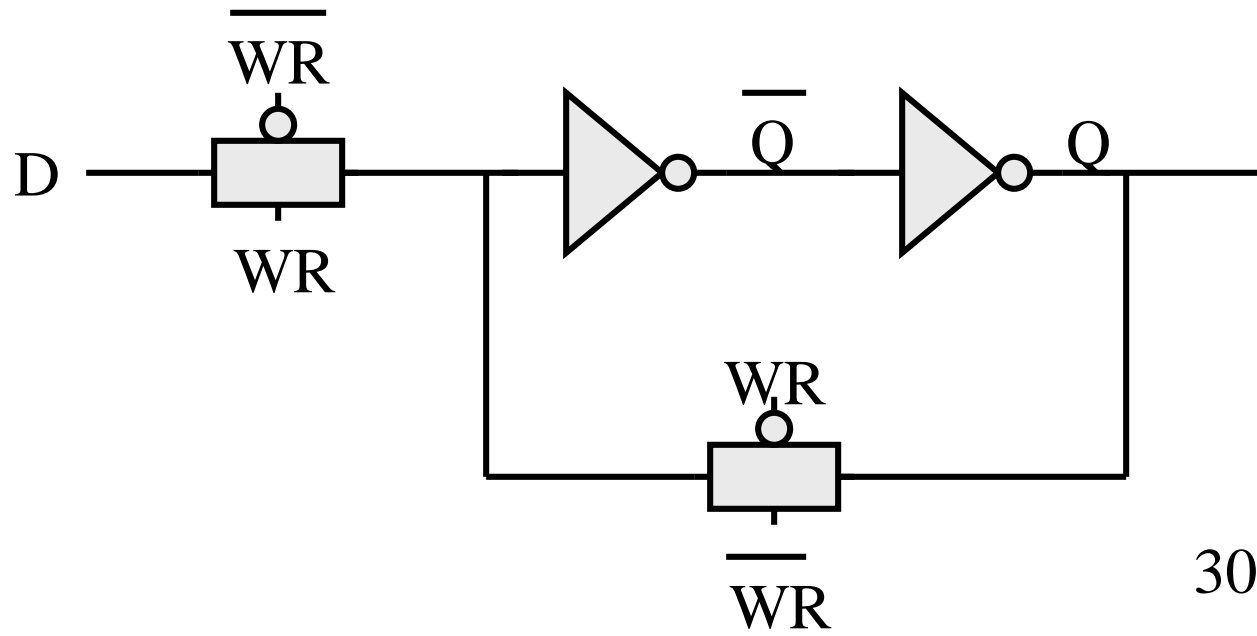
- RAM – Readable and writable memory
 - “Random access memory”
 - Strange name—Created several decades ago to contrast with sequentially-accessed storage like tape drives
 - Logically same as register file—Memory with address inputs, data inputs/outputs, and control
 - RAM usually one port; RF usually two or more
 - RAM vs. RF
 - RAM typically larger than *about* 512 or 1024 words
 - RAM typically stores bits using a bit storage approach that is more efficient than a flip-flop
 - RAM typically implemented on a chip in a square rather than rectangular shape—keeps longest wires (hence delay) short



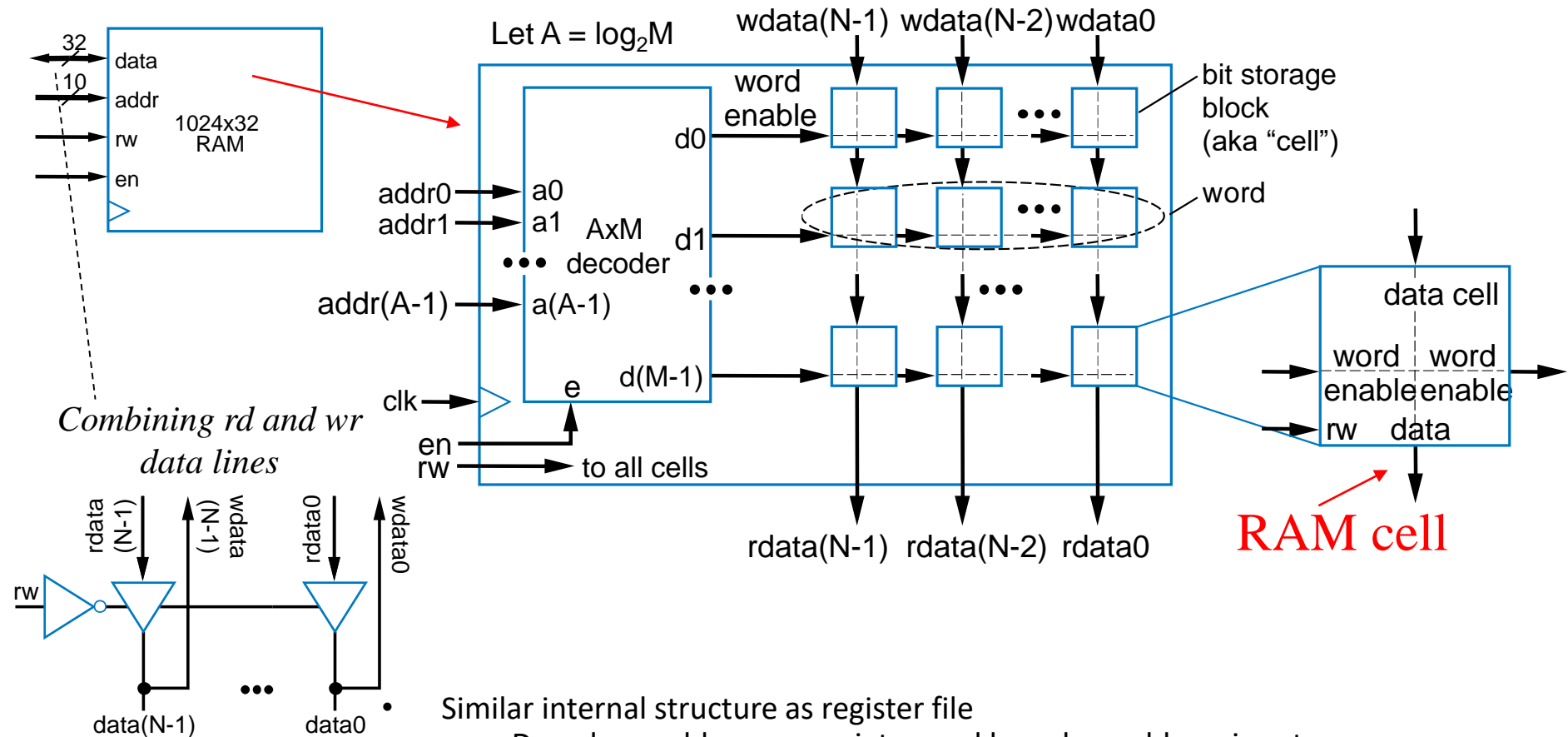
RAM block symbol

Implementing Registers in CMOS

- Uses transmission gate
 - When “WR” asserted, “write” operation will take place
 - Stack D latch structures to get n-bit register



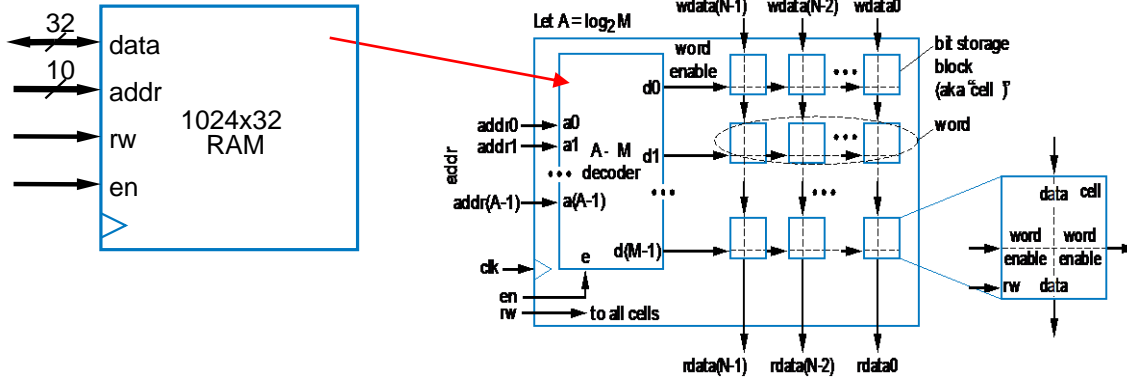
RAM Internal Structure



Similar internal structure as register file

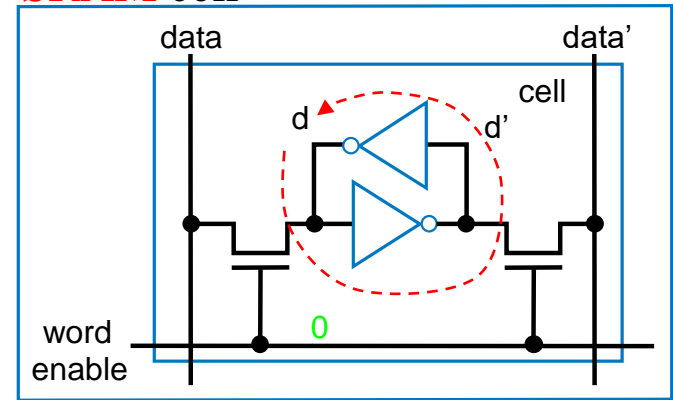
- Decoder enables appropriate word based on address inputs
- rw controls whether cell is written or read
- rd and wr data lines typically combined
- Let's see what's inside each RAM cell

Static RAM (SRAM)

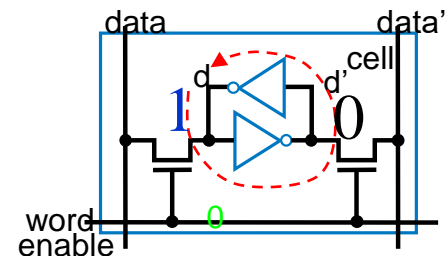
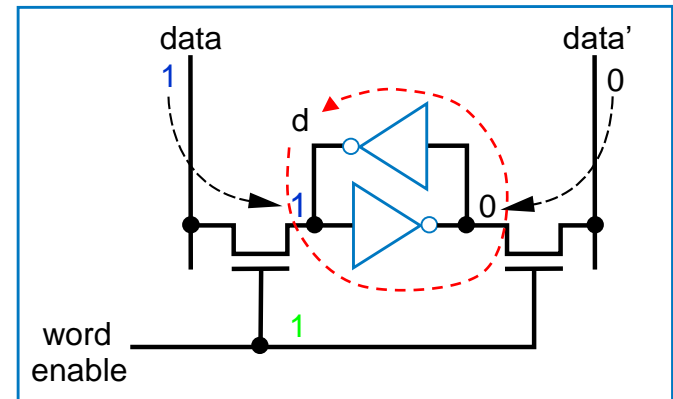


- “Static” RAM cell
 - 6 transistors (recall inverter is 2 transistors)
 - Writing this cell
 - *word enable* input comes from decoder
 - When 0, value *d* loops around inverters
 - That loop is where a bit stays stored
 - When 1, the *data* bit value enters the loop
 - *data* is the bit to be stored in this cell
 - *data'* enters on other side
 - Example shows a “1” being written into cell

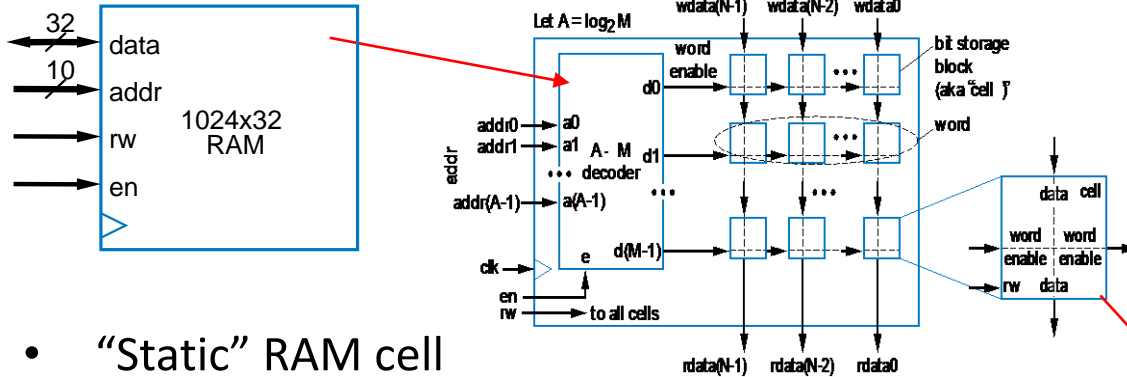
SRAM cell



SRAM cell

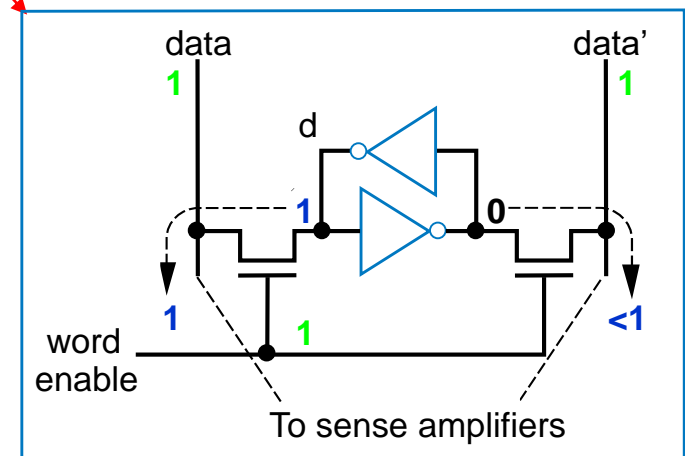


Static RAM (SRAM)



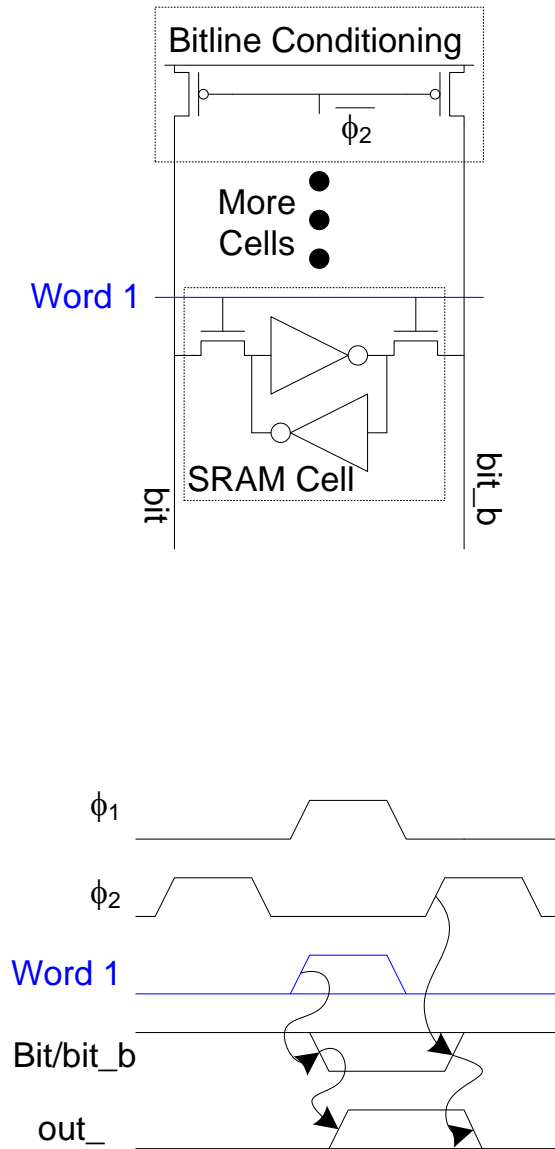
- “Static” RAM cell
 - Reading this cell
 - Somewhat trickier
 - When *rw* set to read, the RAM logic sets both *data* and *data'* to 1
 - The stored bit *d* will pull either the left line or the right bit down slightly below 1
 - “Sense amplifiers” detect which side is slightly pulled down
 - The electrical description of SRAM is really beyond our scope – just general idea here, mainly to contrast with *DRAM*...

SRAM cell

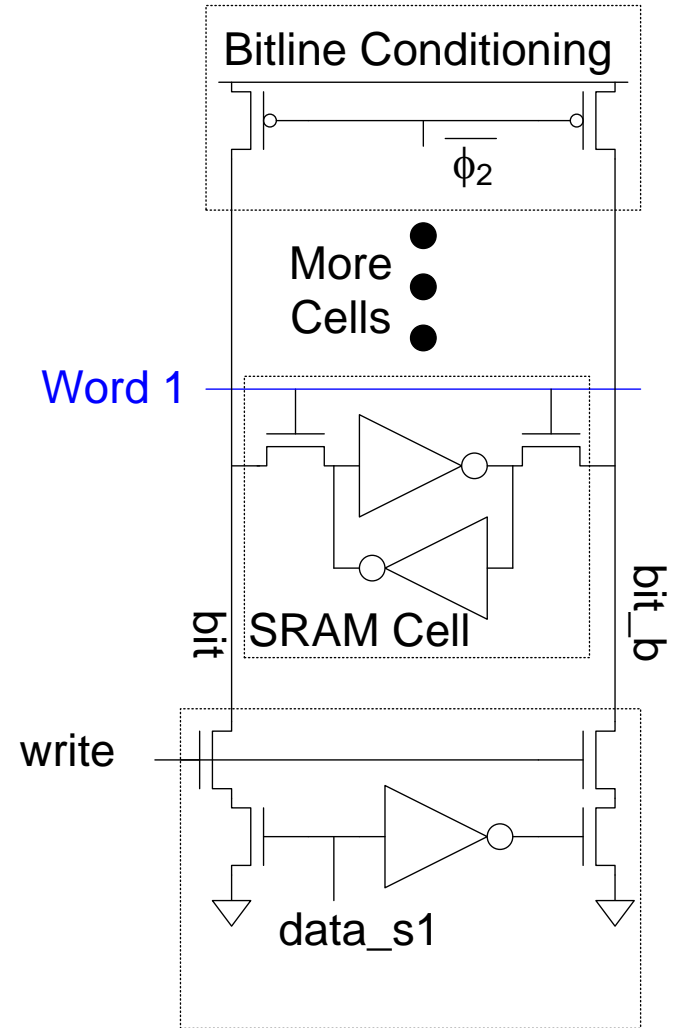


SRAM Column Example

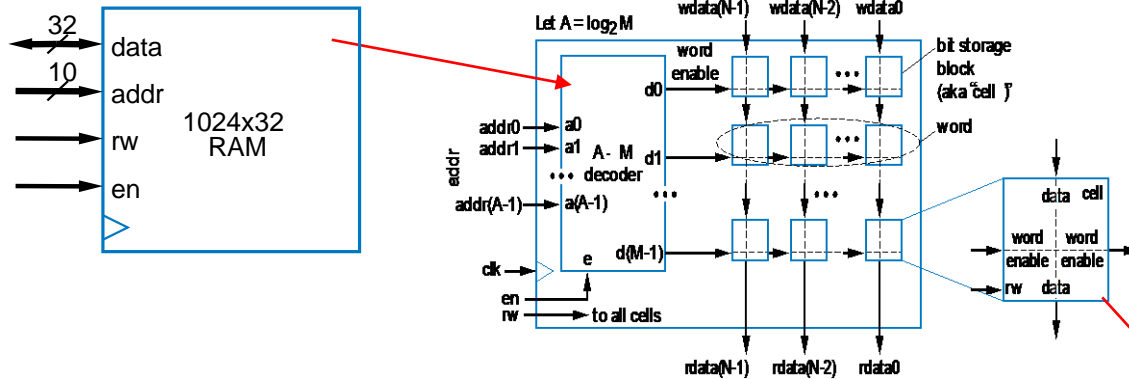
Read



Write

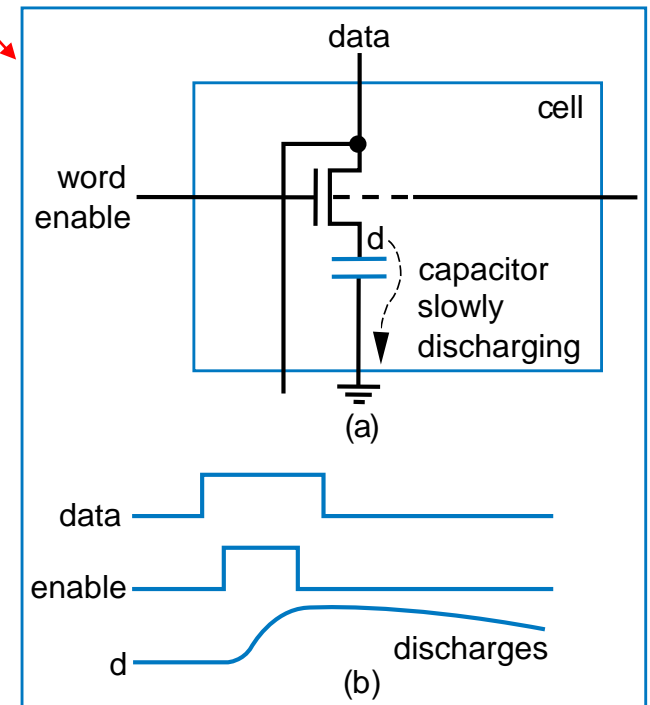


Dynamic RAM (DRAM)



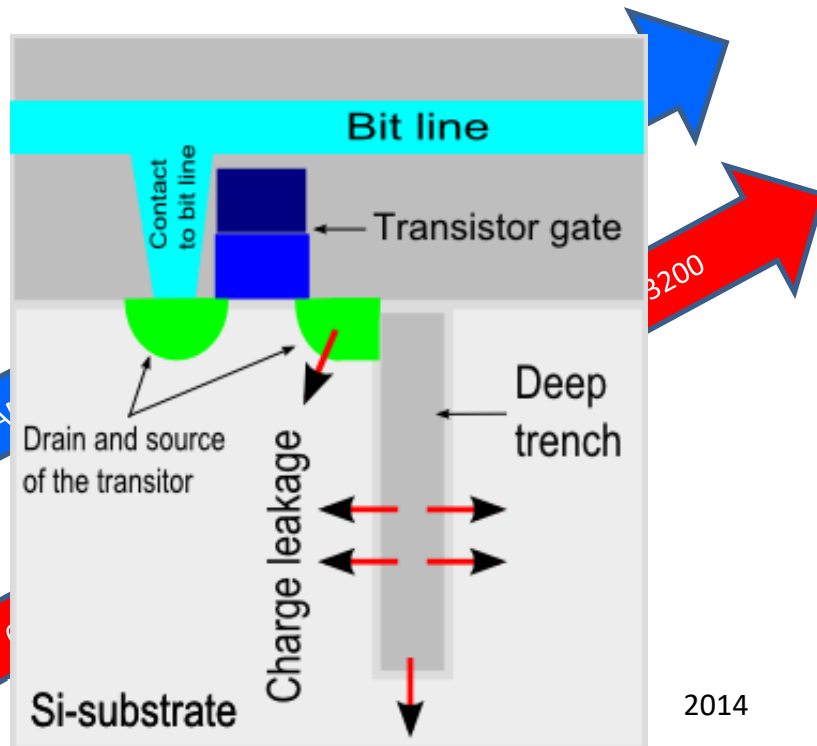
DRAM cell

- “Dynamic” RAM cell
 - 1 transistor (rather than 6)
 - Relies on *large* capacitor to store bit
 - Write: Transistor conducts, data voltage level gets stored on top plate of capacitor
 - Read: Just look at value of d
 - Problem: Capacitor discharges over time
 - Must “refresh” regularly, by reading d and then writing it right back

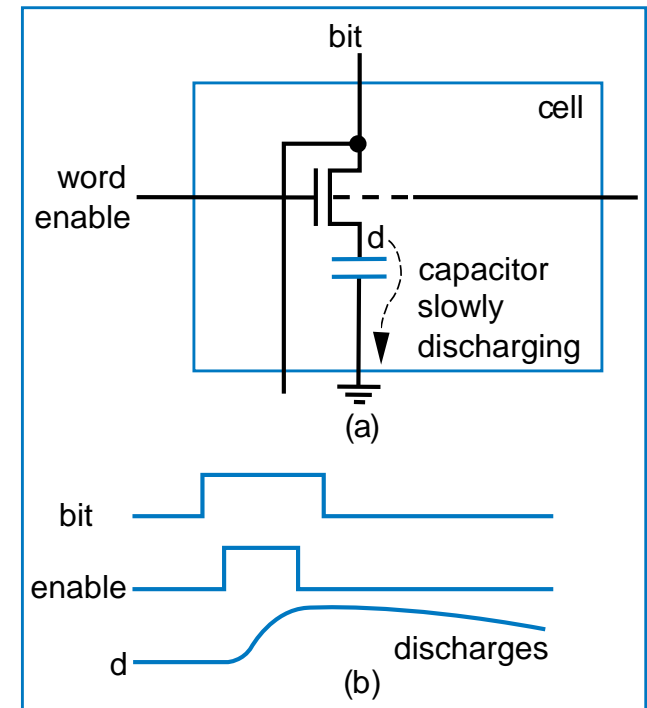


Dynamic RAM (DRAM)

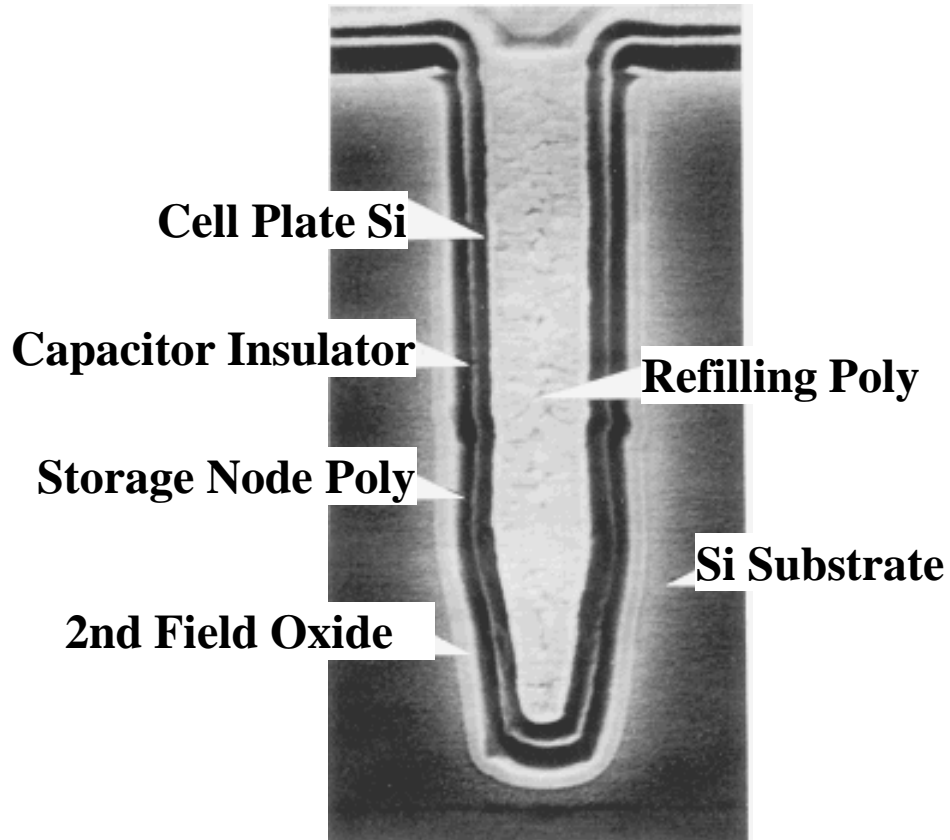
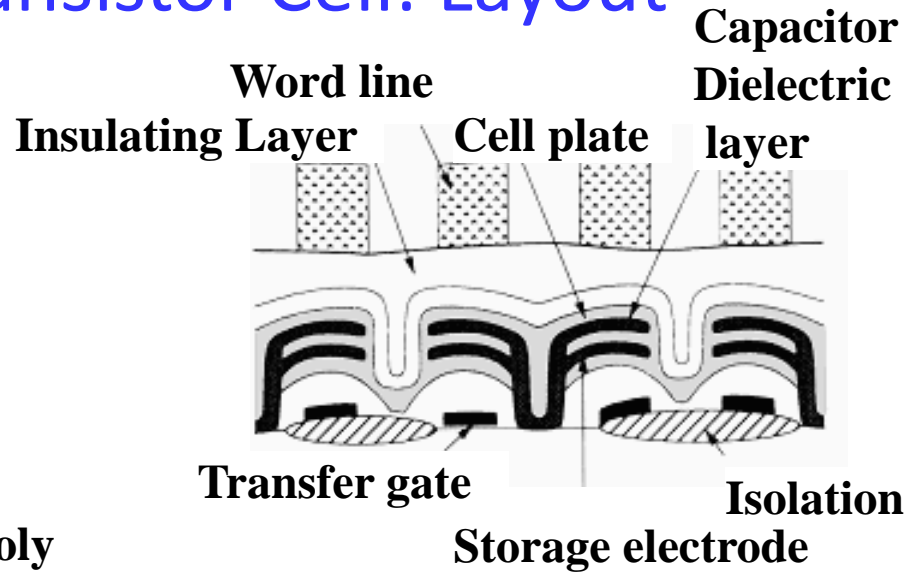
- “Dynamic” RAM cell
 - 1 transistor (rather than 6)
 - Relies on capacitor to store bit



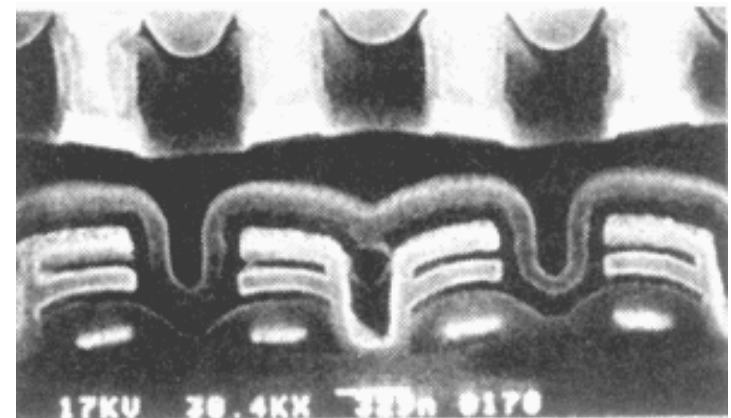
DRAM cell



Dynamic RAM 1-Transistor Cell: Layout



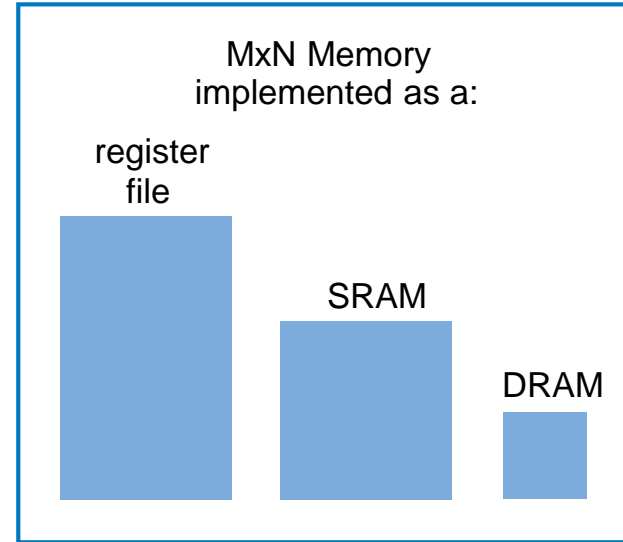
Trench Cell



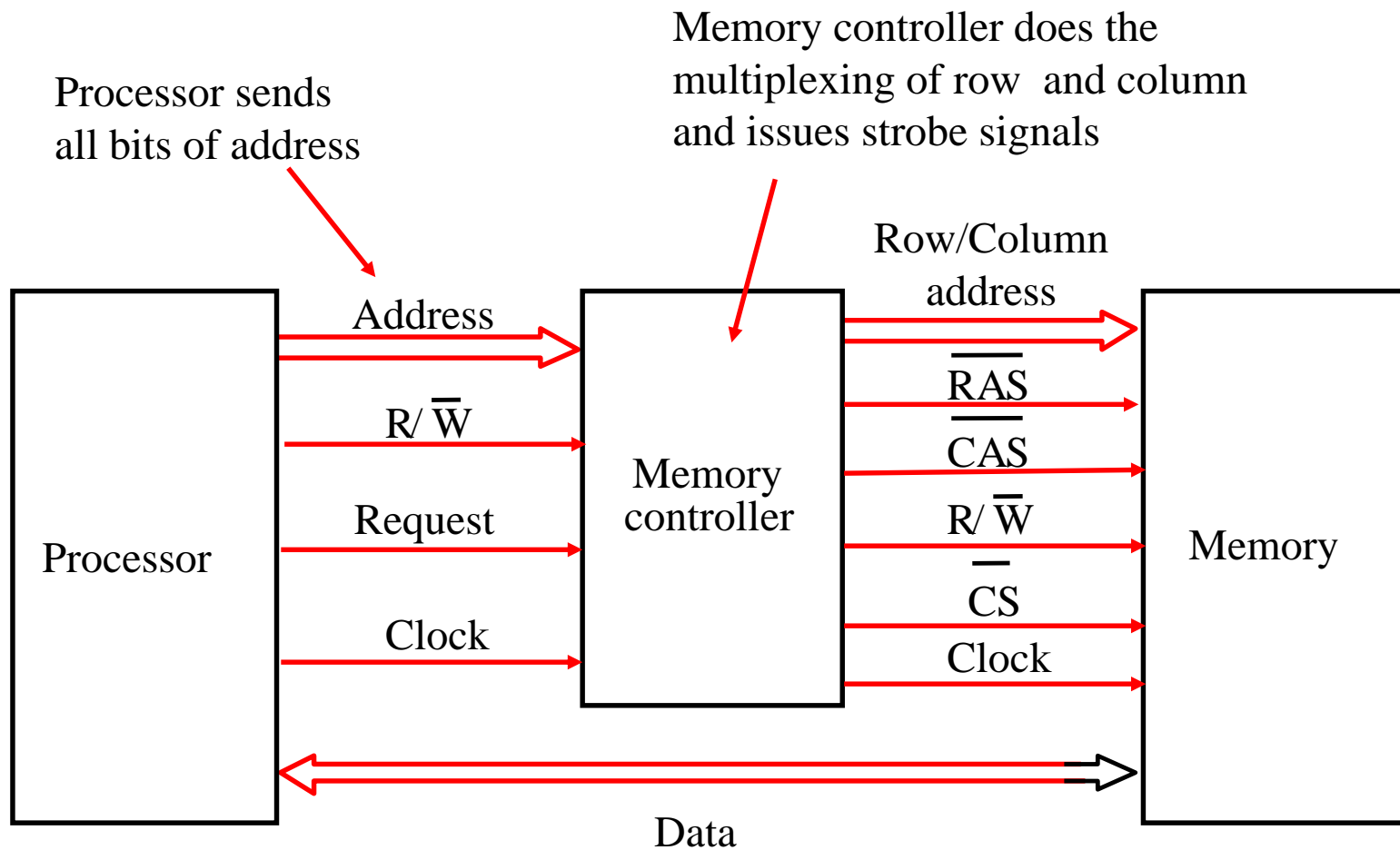
Stacked-capacitor Cell

Comparing Memory

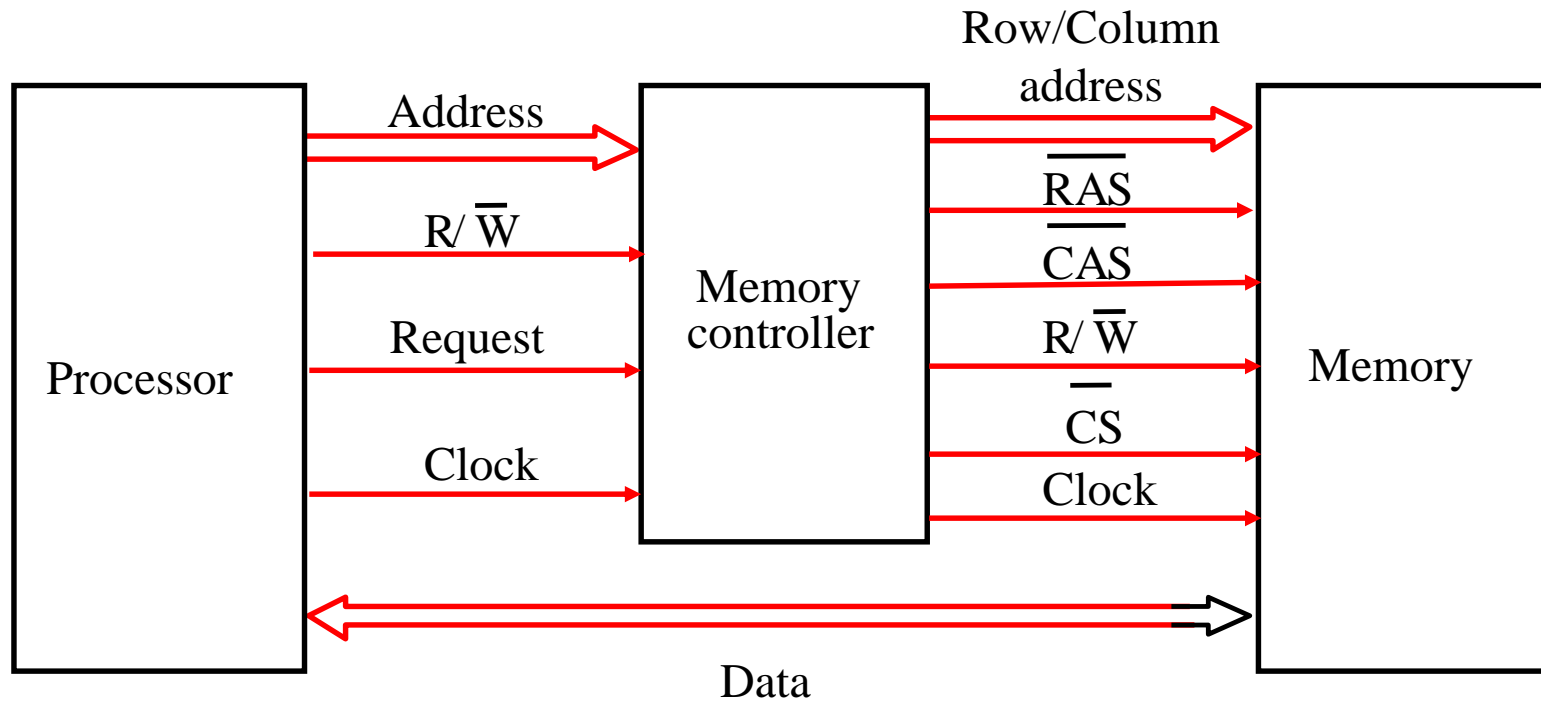
- Register file
 - Fastest
 - But biggest size
- SRAM
 - Fast
 - More compact than register file
- DRAM
 - Slowest (capacitor)
 - And refreshing takes time
 - But very compact (lower cost)
- Use register file for small items, SRAM for large items, and DRAM for huge items
 - Note: DRAM's big capacitor requires a special chip design process, so DRAM is often a separate chip



Size comparison for same number of bits (not to scale)



Use of a memory controller.



Memory controller provides the refresh control if not done on the chip

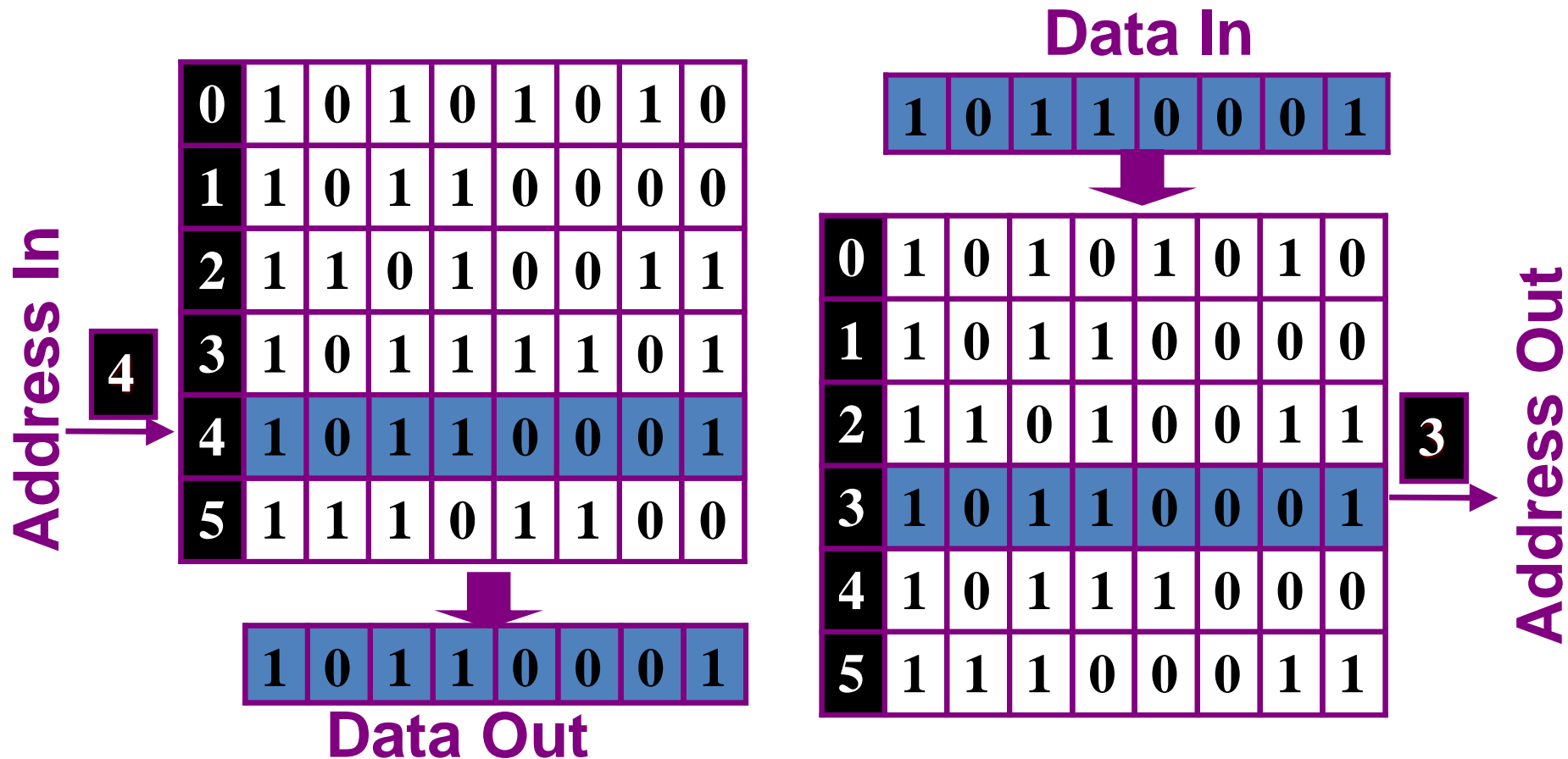
Refreshing typically once every 64 ms. At a cost of .2ms

Less than .4% overhead

Use of a memory controller.

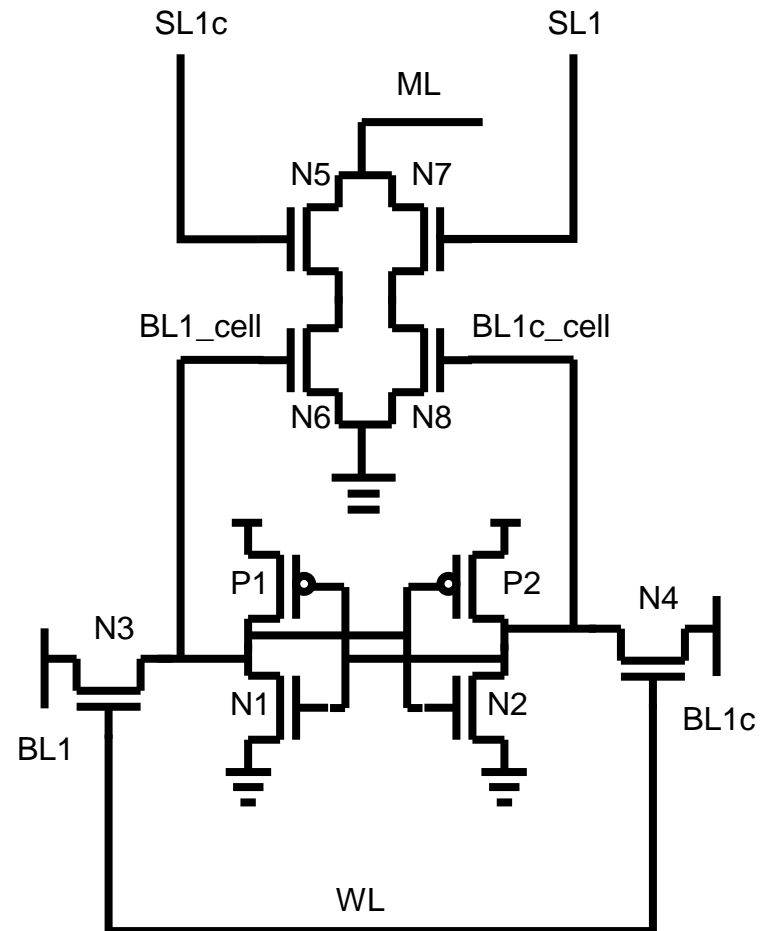
CAM

- CAM vs. RAM



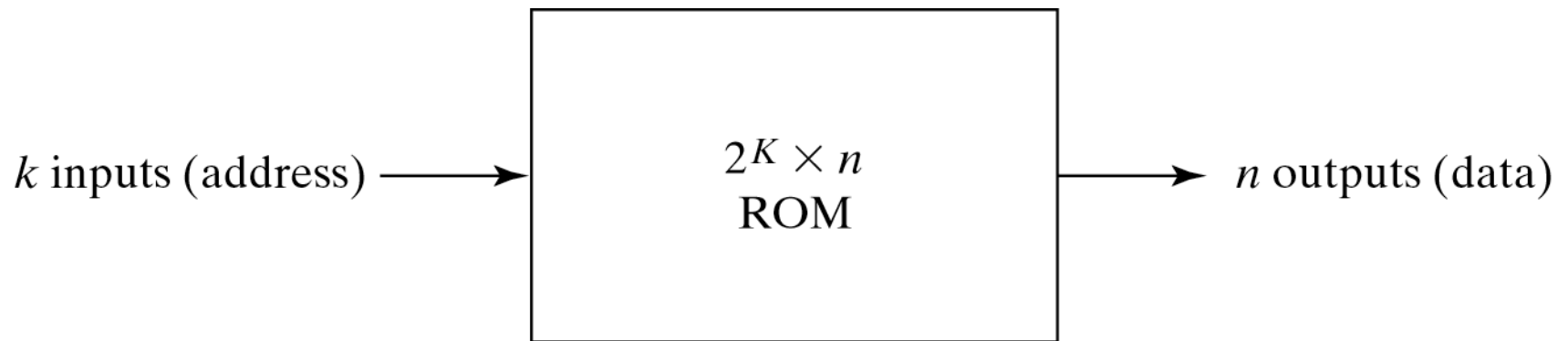
CAM

- Binary CAM Cell
 - ML pre-charged to V_{DD}
 - Match: ML remains at V_{DD}
 - Mismatch: ML discharges



Read-Only Memory

- A block diagram of a ROM is shown below. It consists of k address inputs and n data outputs.
- The number of words in a ROM is determined from the fact that k address input lines are needed to specify 2^k words.



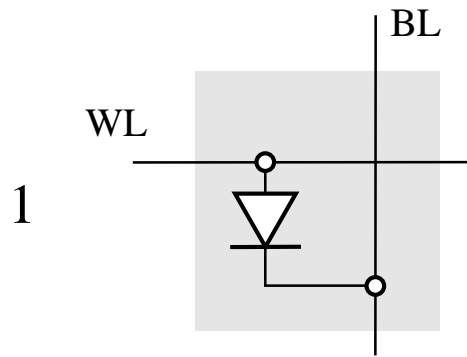
ROM Block diagram $2^k \times n$ Module

Read-Only Memory Cells

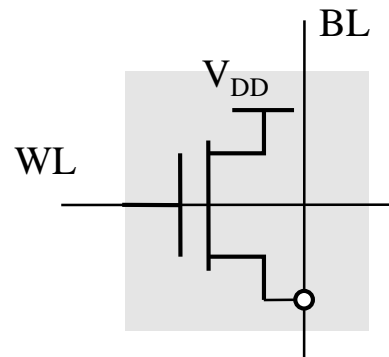
Bit line (BL) is resistively clamped to the ground, so its default value is 0

Diode disadvantage – no electrical isolation between bit and word lines

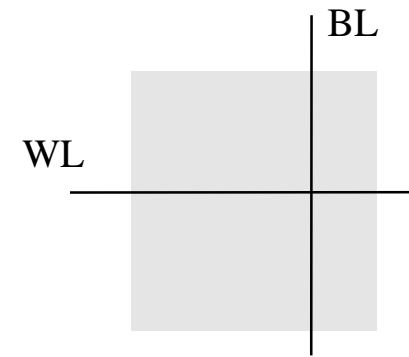
BL is resistively clamped to VDD, so its default value is 1



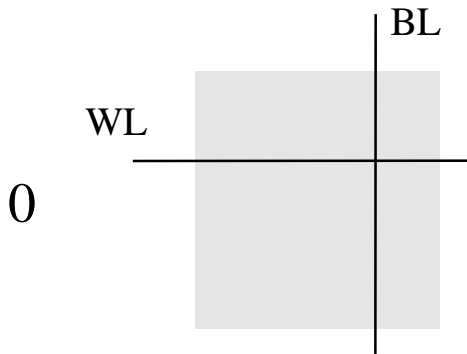
Diode ROM



MOS ROM 1

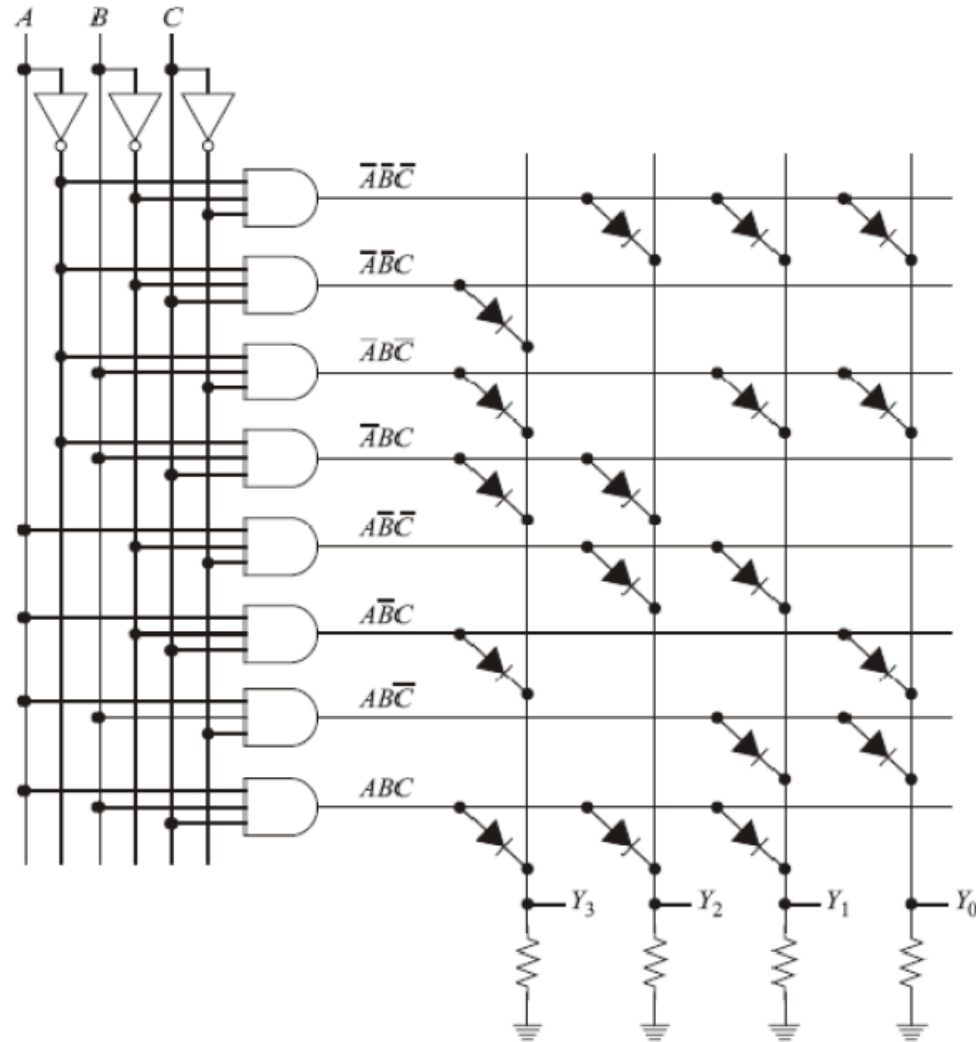


MOS ROM 2



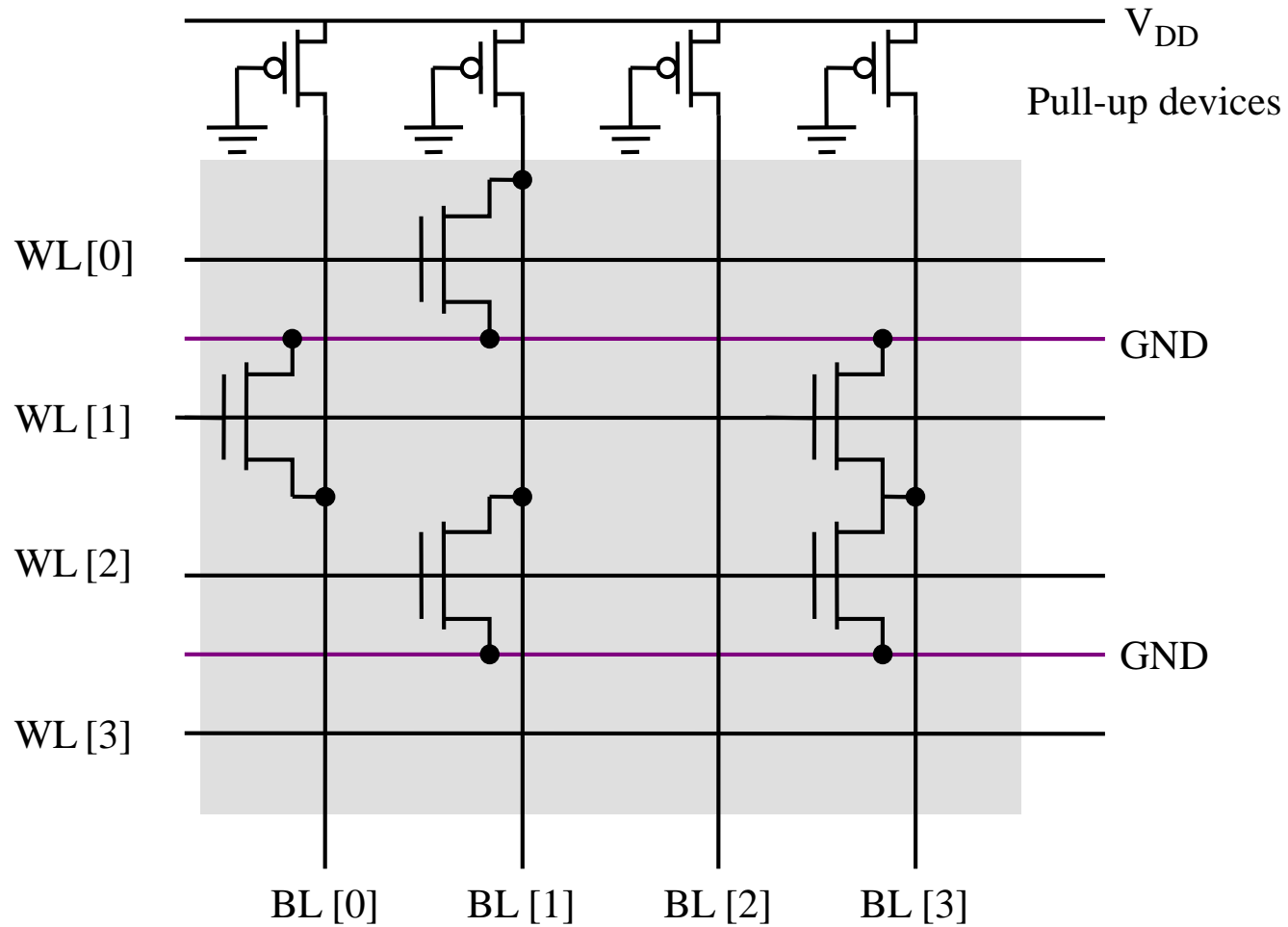
Nonvolatile Memory

ROM
PROM
EPROM

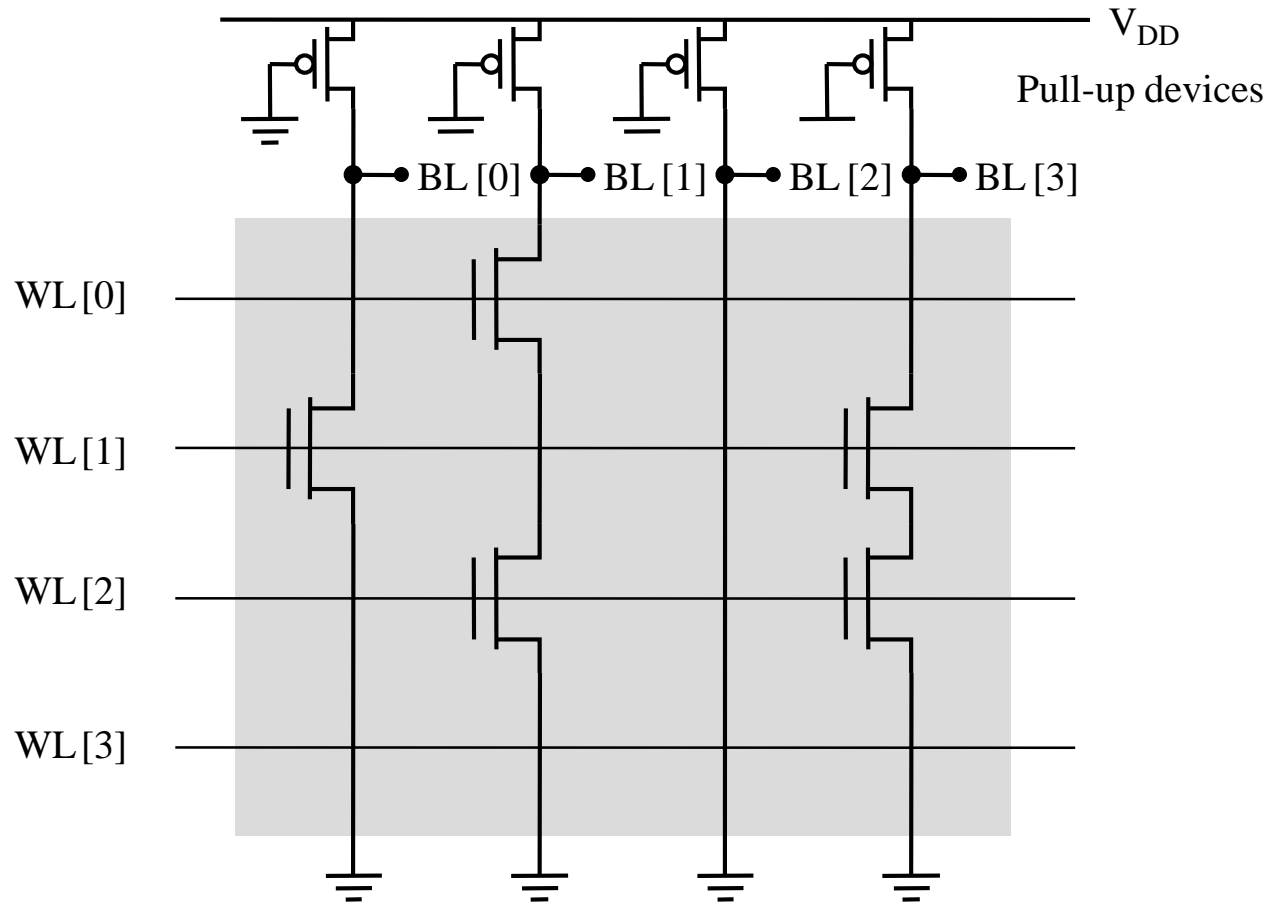


Read-only memory organization, with the fixed contents shown on the right.

MOS NOR ROM

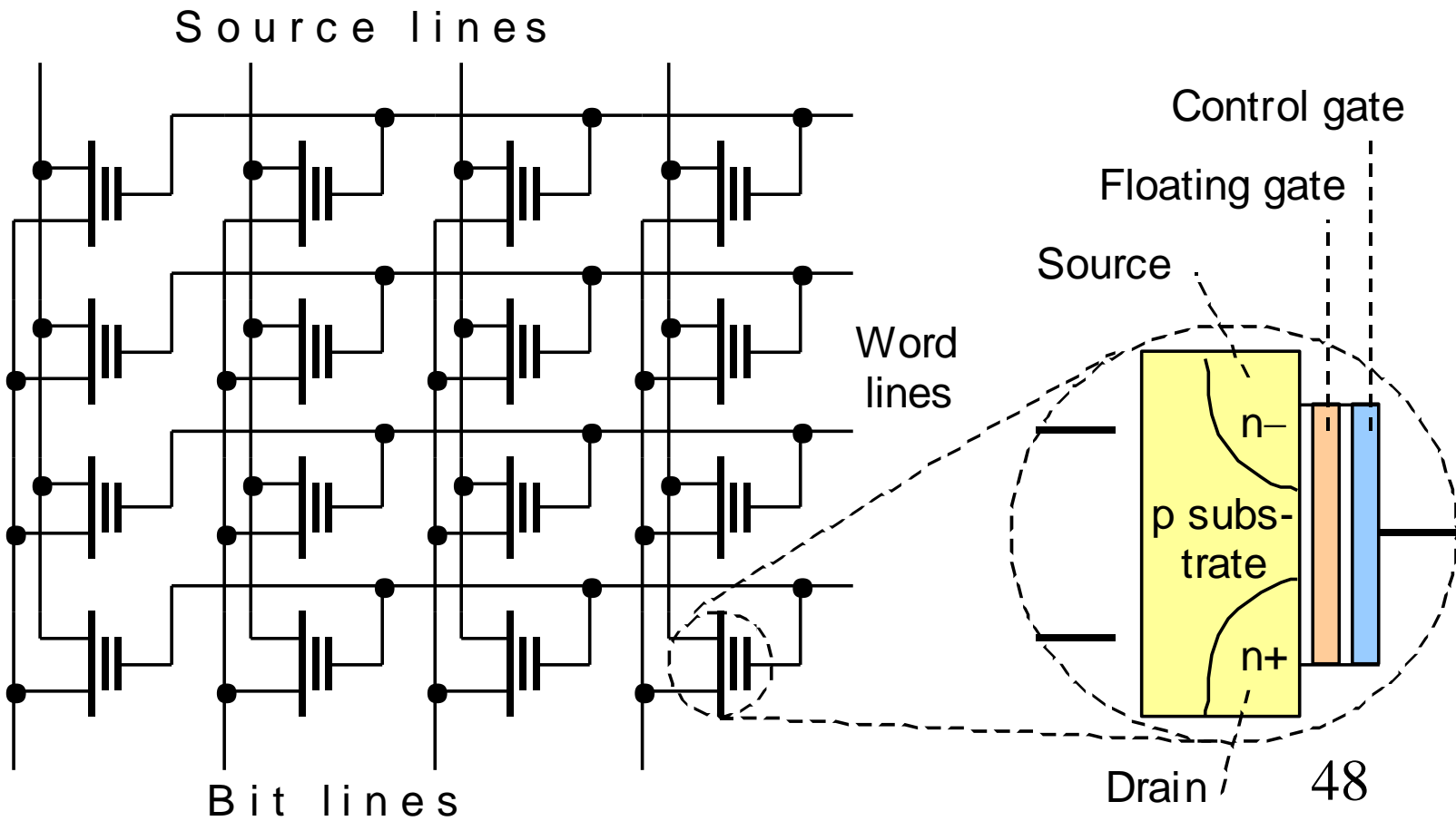


MOS NAND ROM



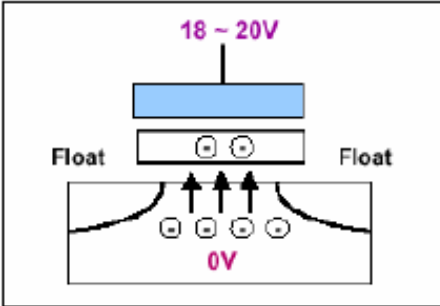
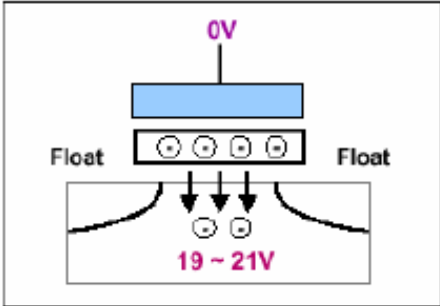
All word lines high by default with exception of selected row

Flash Memory

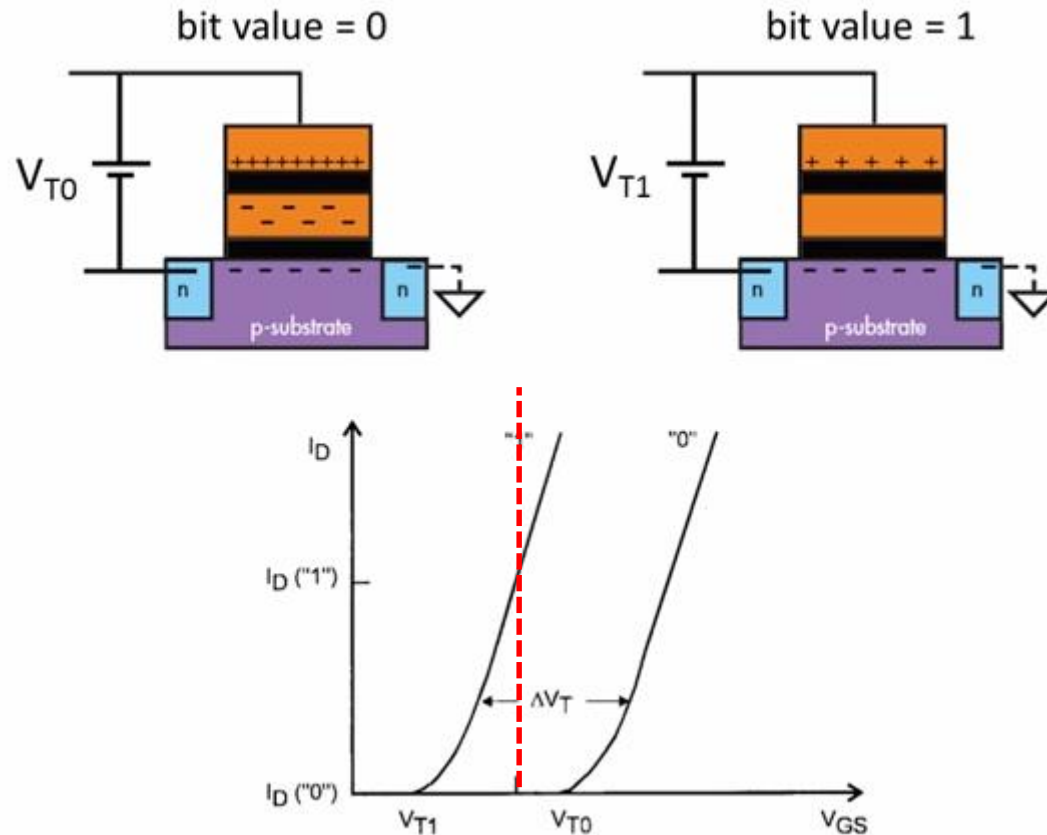


Flash memory

- Flash memory
 - Non volatile read only memory (ROM)
 - Erase Electrically or UV (EPROM)
 - Uses F-N tunneling for program & erase
 - Reads like DRAM (~ns)
 - Writes like DISK (~ms).

| Program | Erase |
|---|--|
|  |  |
| <ul style="list-style-type: none">● Use F-N Tunneling● Channel Inversion | <ul style="list-style-type: none">● Use F-N Tunneling● Channel Accumulation |

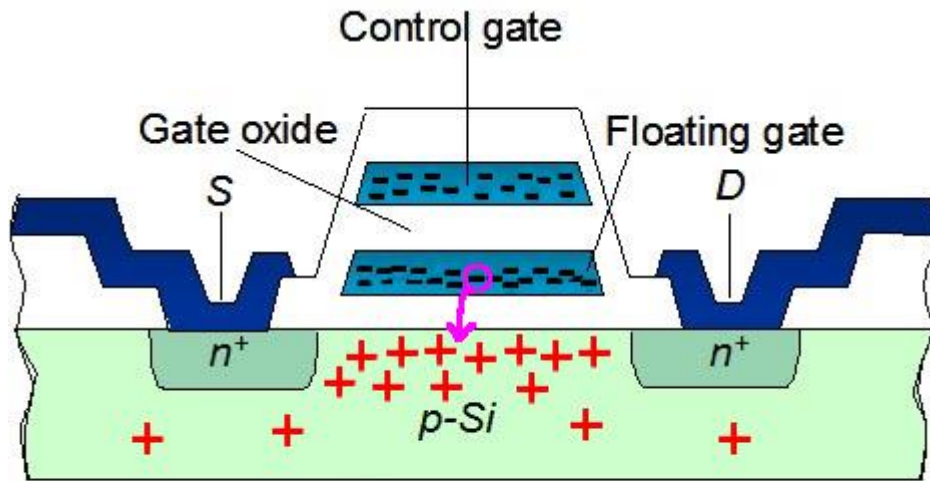
Reading Memory State



Change in Threshold Voltage due to **Screening Effect** of Floating Gate

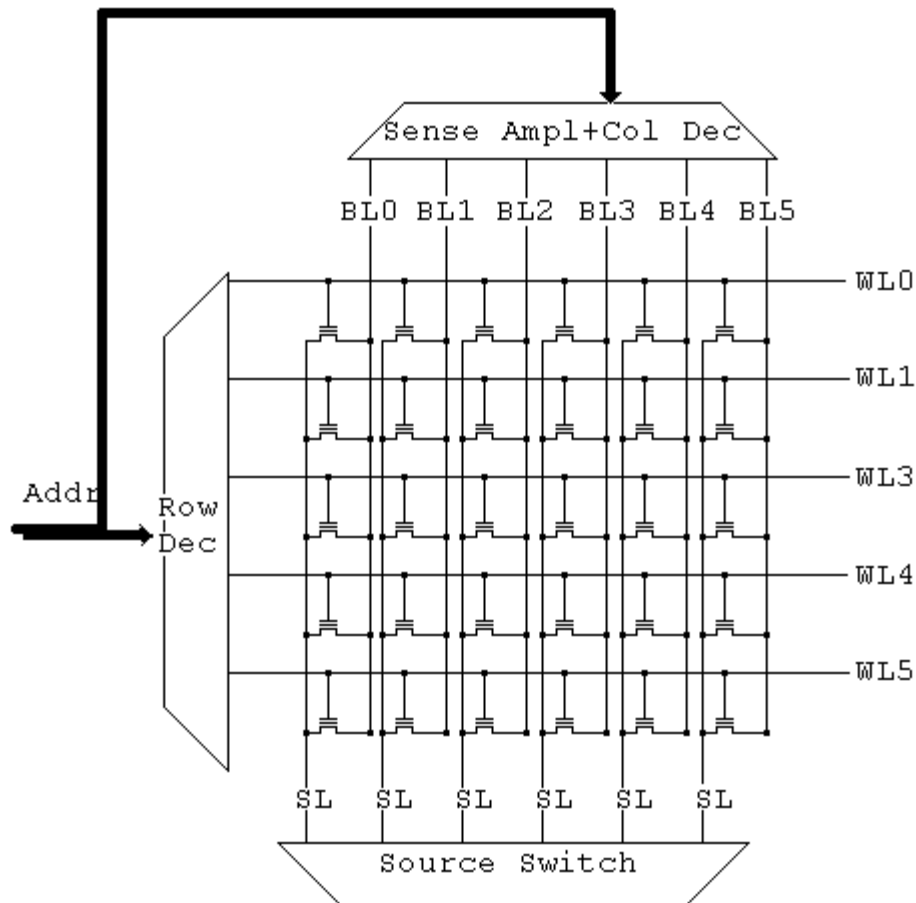
Read mode: Apply intermediate voltage, check whether current is flowing or not

Writing Memory State



Control gate voltage determines whether electrons are injected to, or push/pulled out of floating gate.

NOR Array

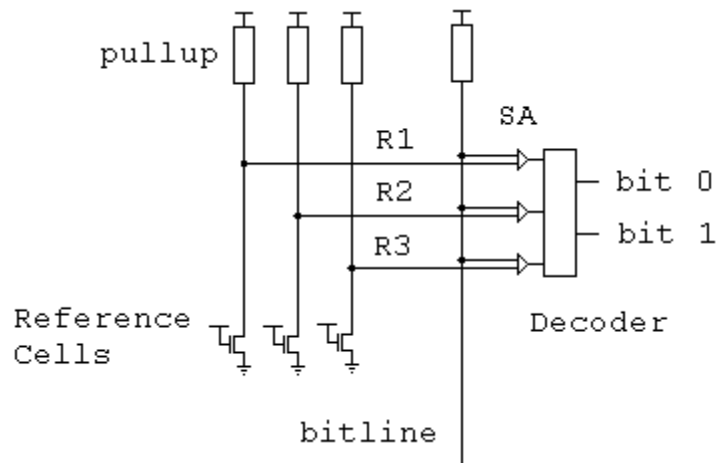


Reading:

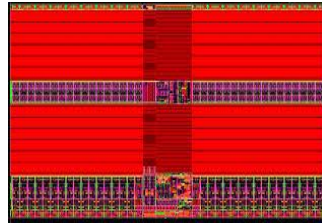
Assert a single word line.
The source lines are asserted and the read of the bitline gives the contents of the cell.

Multi-Levels

- By using reference cells set at given levels and comparing them to the value from the bitline, we can determine the value stored.



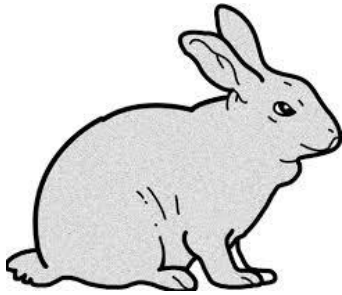
Review of memory technologies



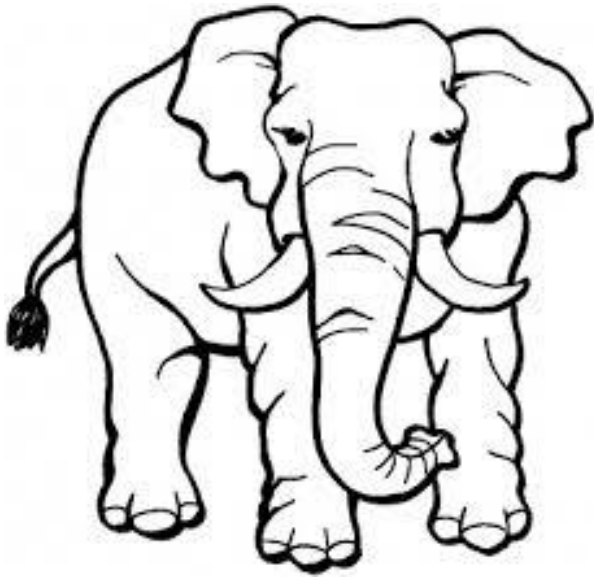
| Memory type | SRAM | DRAM | Flash |
|-------------|-----------------|-----------|--------------|
| Speed | Very fast | Slow | Slow |
| Density | Low | High | Very high |
| Power | High | Low | Very low |
| Refresh | No | Yes | No |
| Mechanism | Bi-stable latch | Capacitor | FN tunneling |

Memory Hierarchy

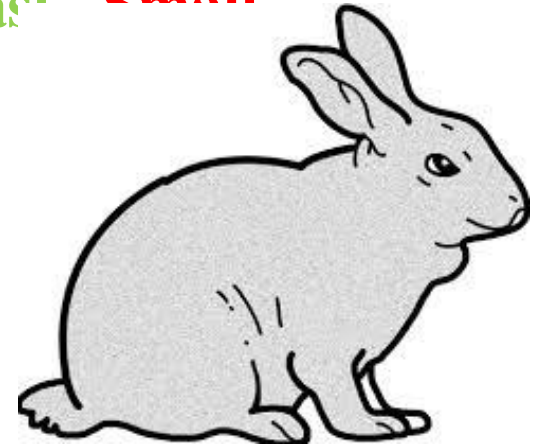
Actual Memory Systems



+



Type1: Fast Small



Can we achieve?: Fast, Large

Type2: Slow, Large

(Cache Principle)

Summary

- **SRAM** is fast but expensive and not very dense:
 - Good choice for providing the user FAST access time.
- **DRAM** is slow but cheap and dense:
 - Good choice for presenting the user with a BIG memory system
- **Flash Memory : Read** is fast but write is slow

-