Memory

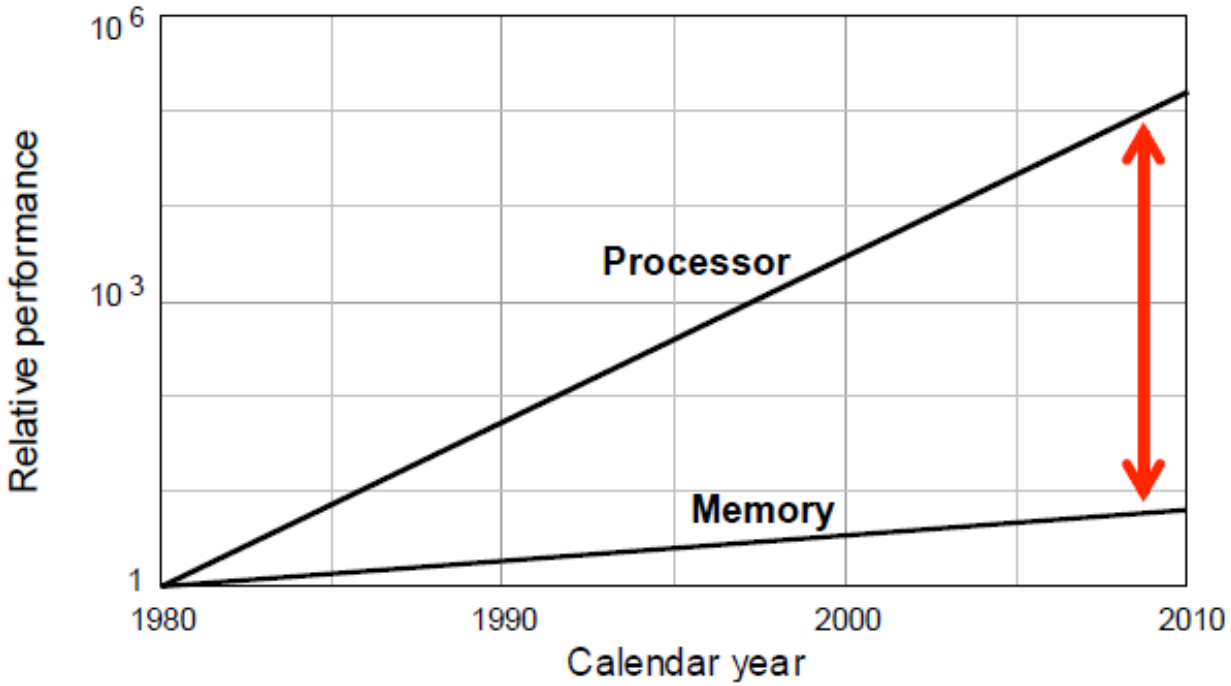
Muhammad Tahir

Lecture 18

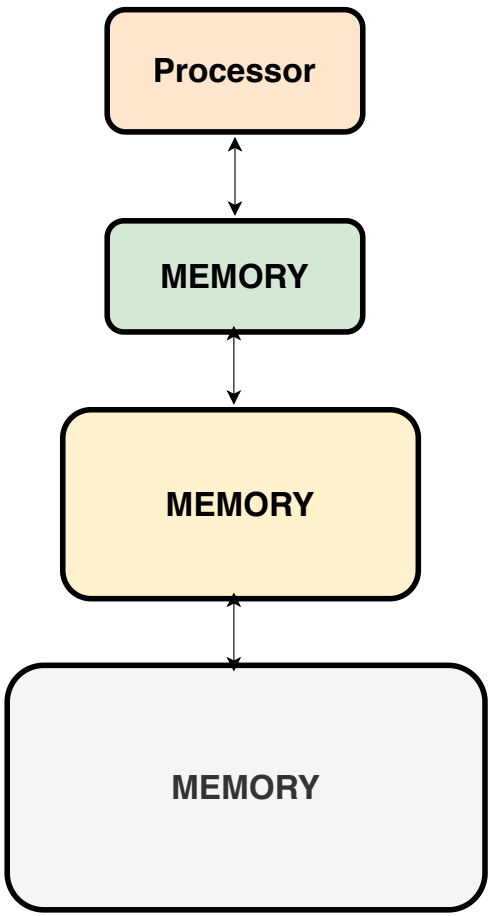
Electrical Engineering Department
University of Engineering and Technology Lahore

Contents

Processor-Memory Performance Gap

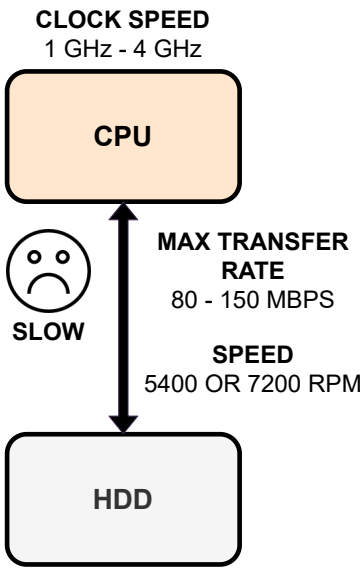


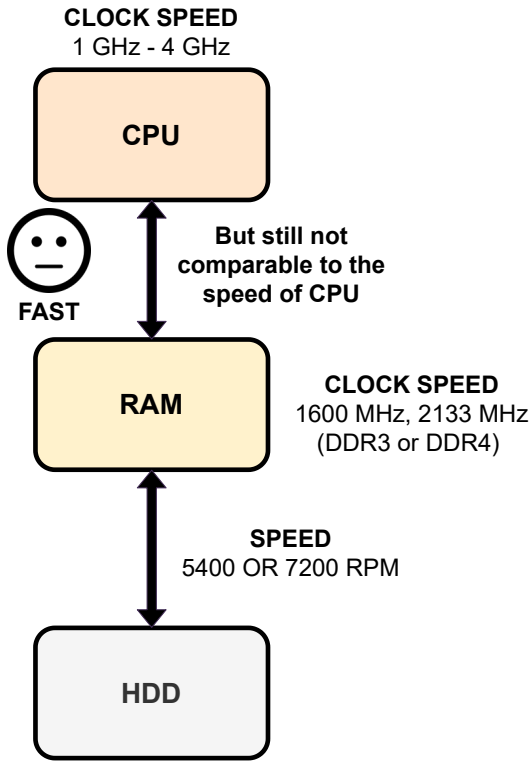
Memory Hierarchy

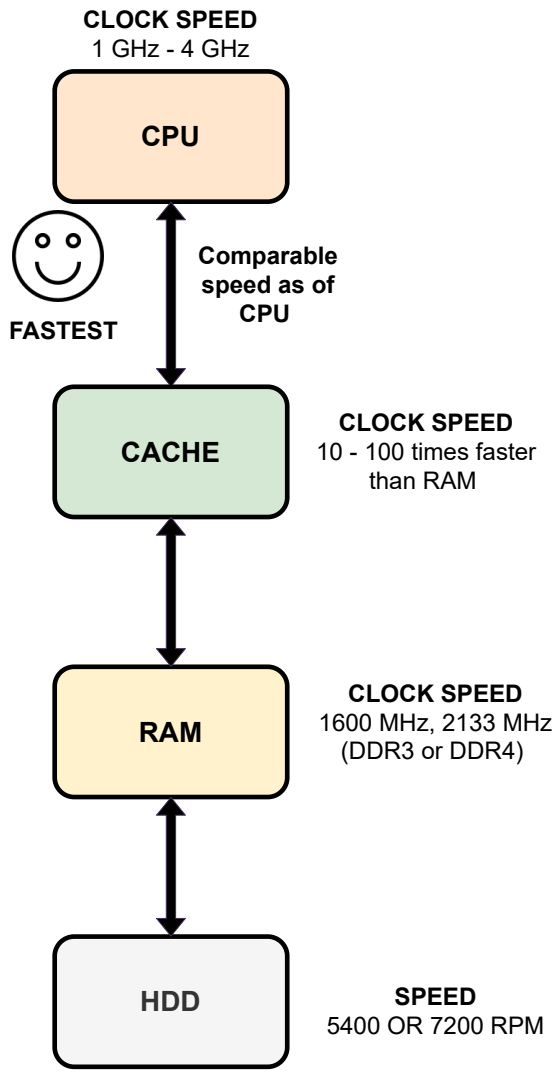


SPEED	CAPACITY	COST ($/GB)	CURRENT TECHNOLOGY
Fastest	MB's	$100's	SRAM
	GB's	$10's	DRAM
Slowest	TB's	$1	Magnetic Disk

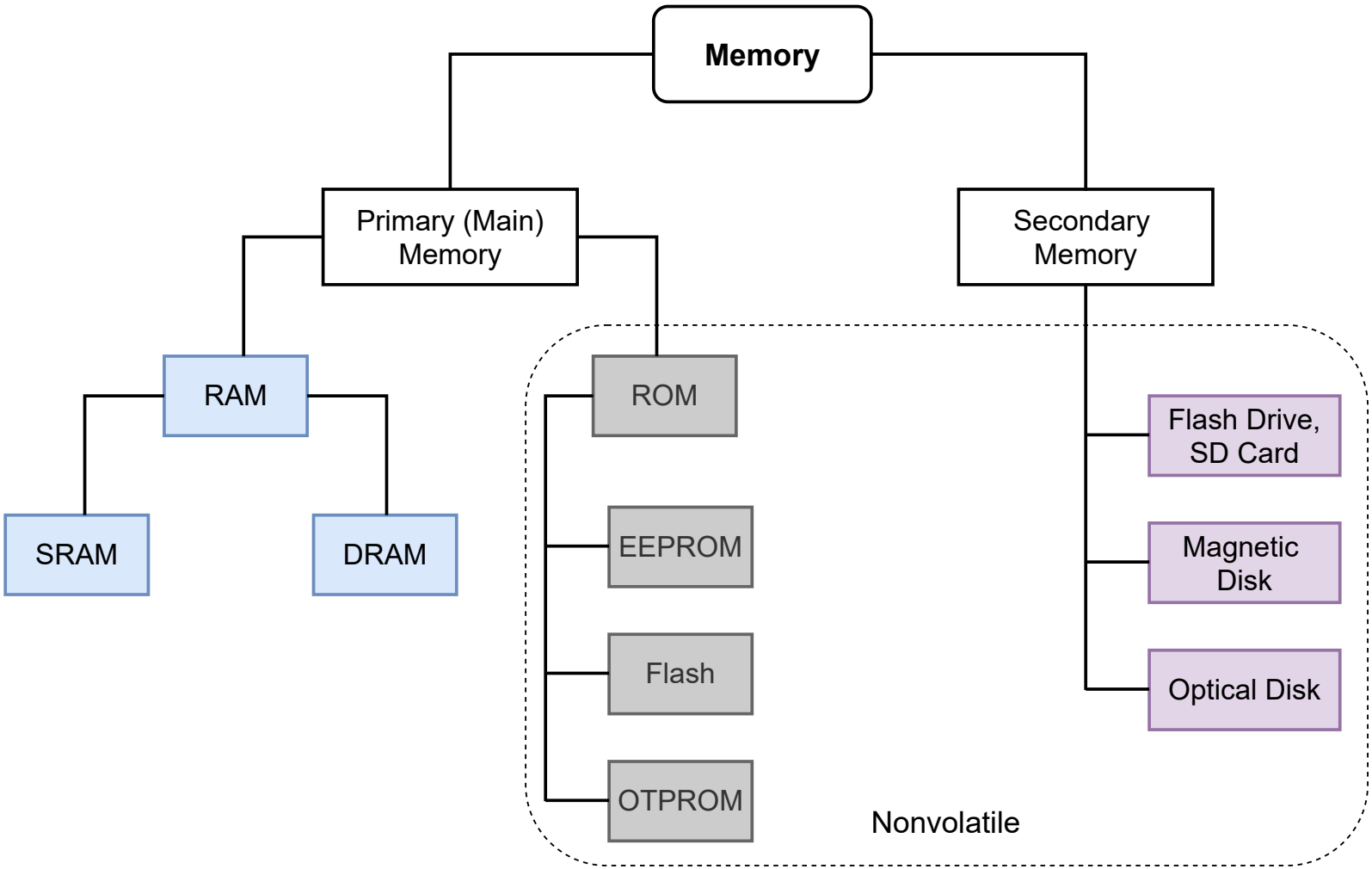
Memory Hierarchy Cont'd







Memory Types



Non-Volatile Memory (Flash Type)

- NOR flash: bit cell like a NOR gate
  - Random read/write access
  - Used for instruction memory in embedded systems
- NAND flash: bit cell like a NAND gate
  - Denser (bits/area), but block-at-a-time access
  - Cheaper per GB
  - Used for USB keys, media storage
- Flash bits wears out after 1000's of accesses
  - Not suitable for direct RAM or disk replacement
  - Wear leveling: remap data to less used blocks

Non-Volatile Memory (Magnetic/Optical)

Magnetic	**Optical**
Data stored magnetically	Data stored optically
Storage is based on magnetic alignment	Storage is based on height variations (pits & bumps)
Access time $<$ 10 ms	Access time ¡ 200 ms
Faster data read/write	Slower data read/write
Always readable and rewrite-able	Readable, write-able and may be rewrite-able

Volatile Memory

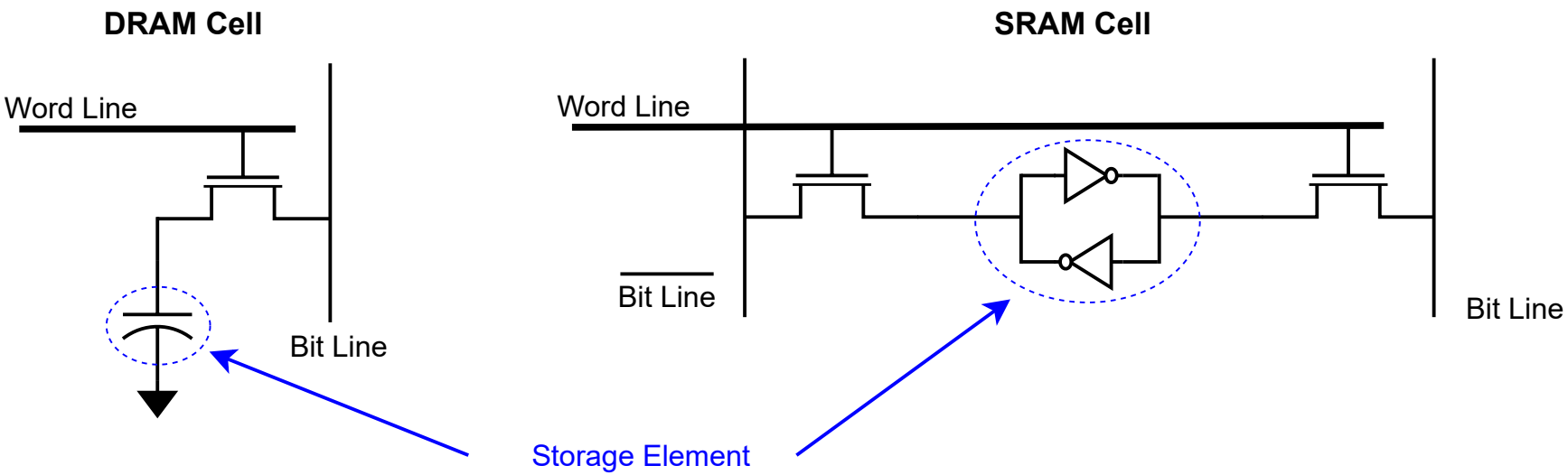
- Random Access Memory (RAM)
  - Any byte of memory can be accessed arbitrarily
  - RAM is the most common type of memory found in computers and other digital devices
  - There are two main types of RAM
- DRAM
  - Needs to be "refreshed" regularly (~ every 8 ms)
  - Refreshing accounts for 1 to 2 % of the active cycles of the DRAM
  - Used as Main Memory
- SRAM

Volatile Memory Cont'd

- Random Access Memory (RAM)
  - Any byte of memory can be accessed arbitrarily
  - RAM is the most common type of memory found in computers and other digital devices
  - There are two main types of RAM
- DRAM
- SRAM
  - Will last until power turned off
  - Low density (6 transistor cells), high power, expensive, fast
  - Used for Caches

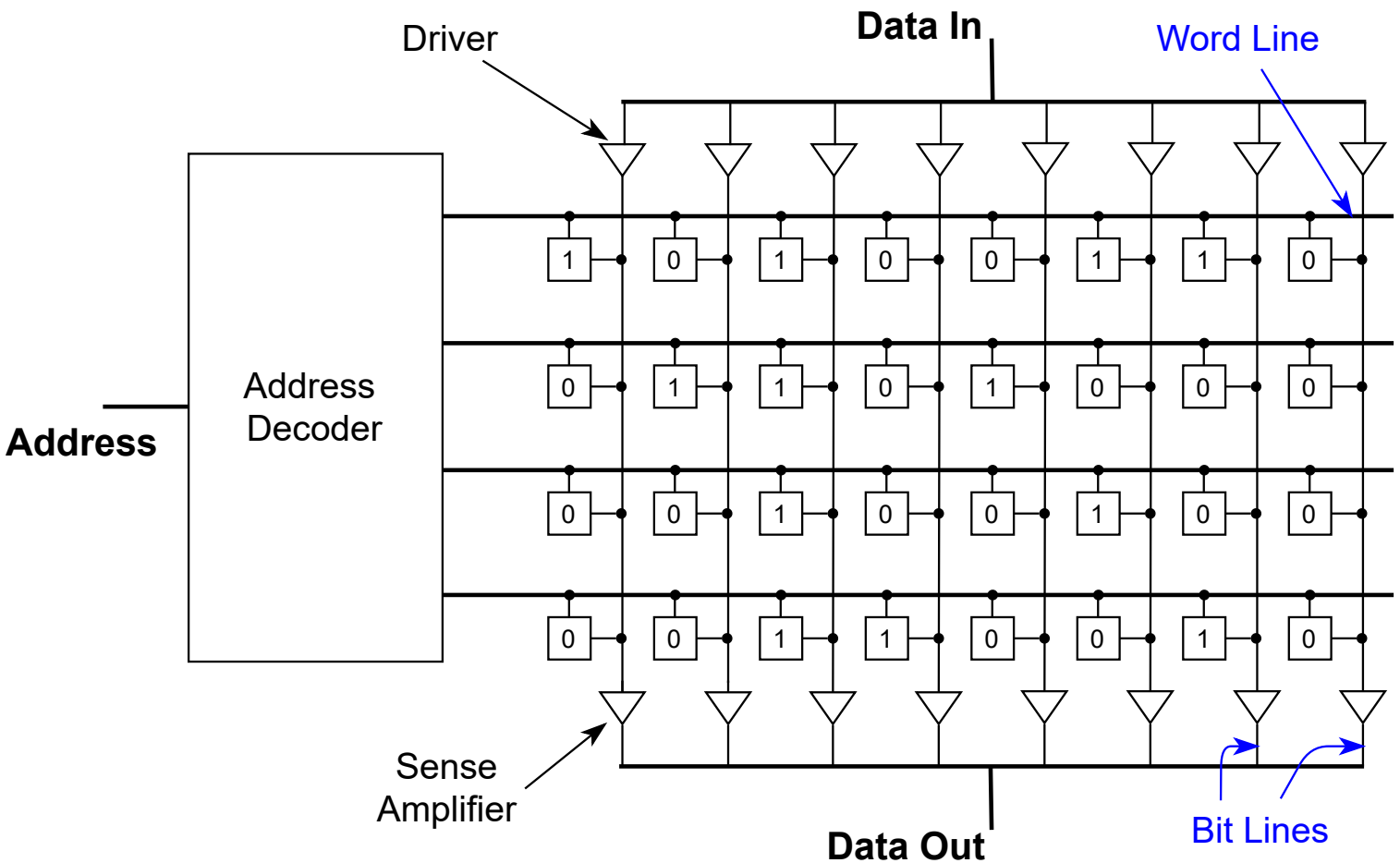
Volatile Memory Cont'd

- Single-transistor DRAM cell is considerably simpler than six-transistor SRAM cell



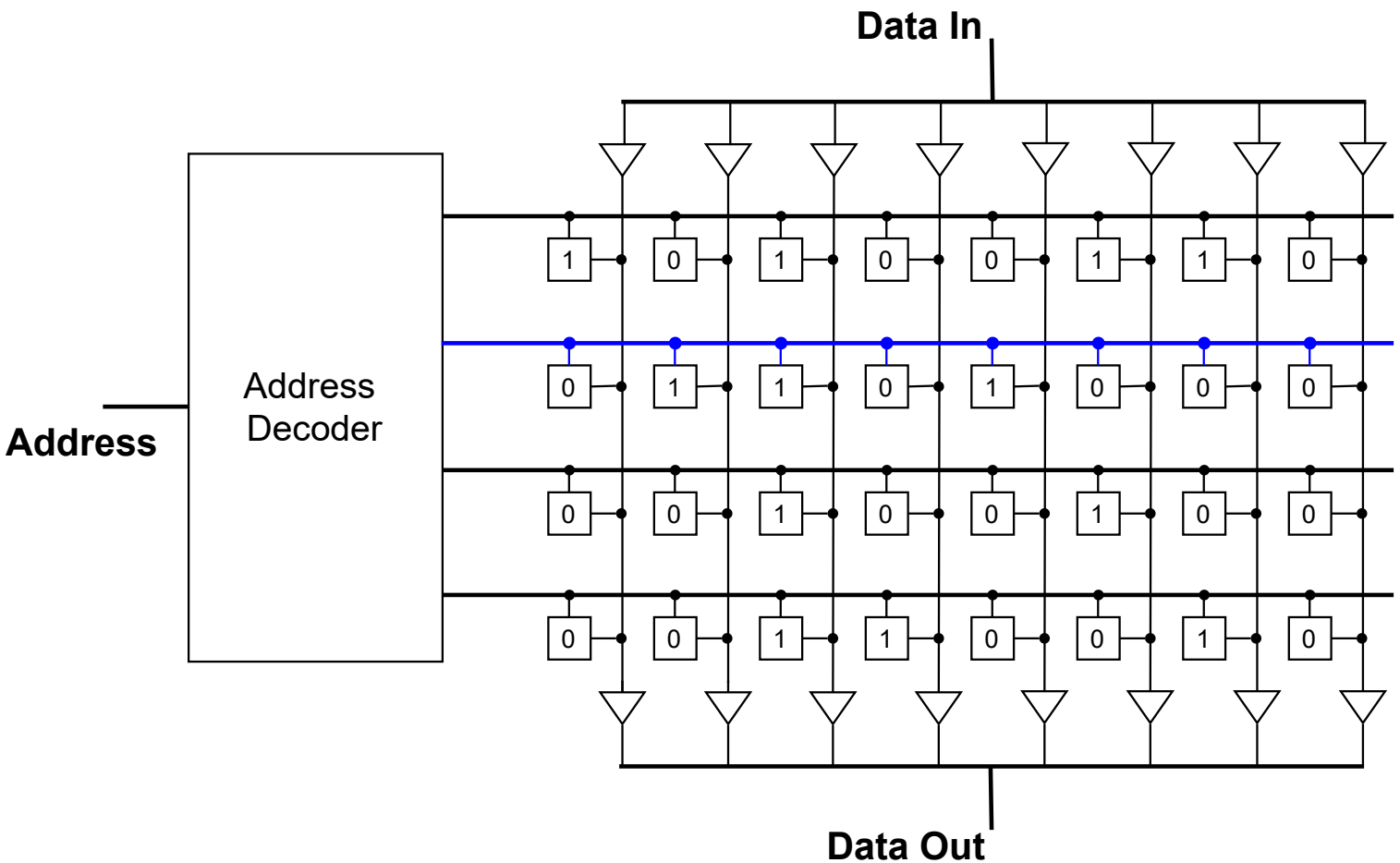
RAM Memory Organization

- Memory organized as arrays



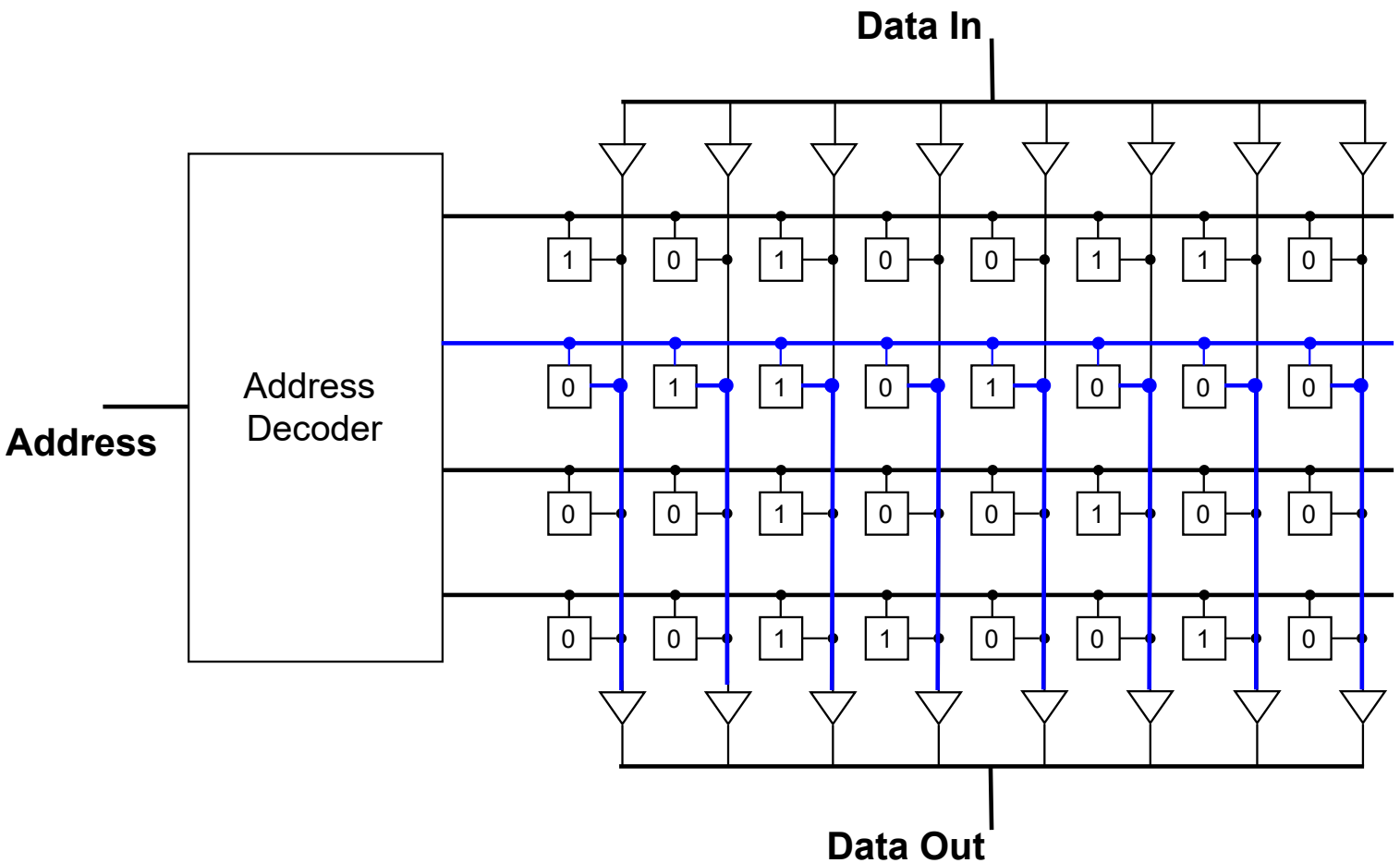
RAM Memory Organization Cont'd

- Only one Wold-line activated by address decoder



RAM Memory Organization Cont'd

- Contents of selected word-line are available at output

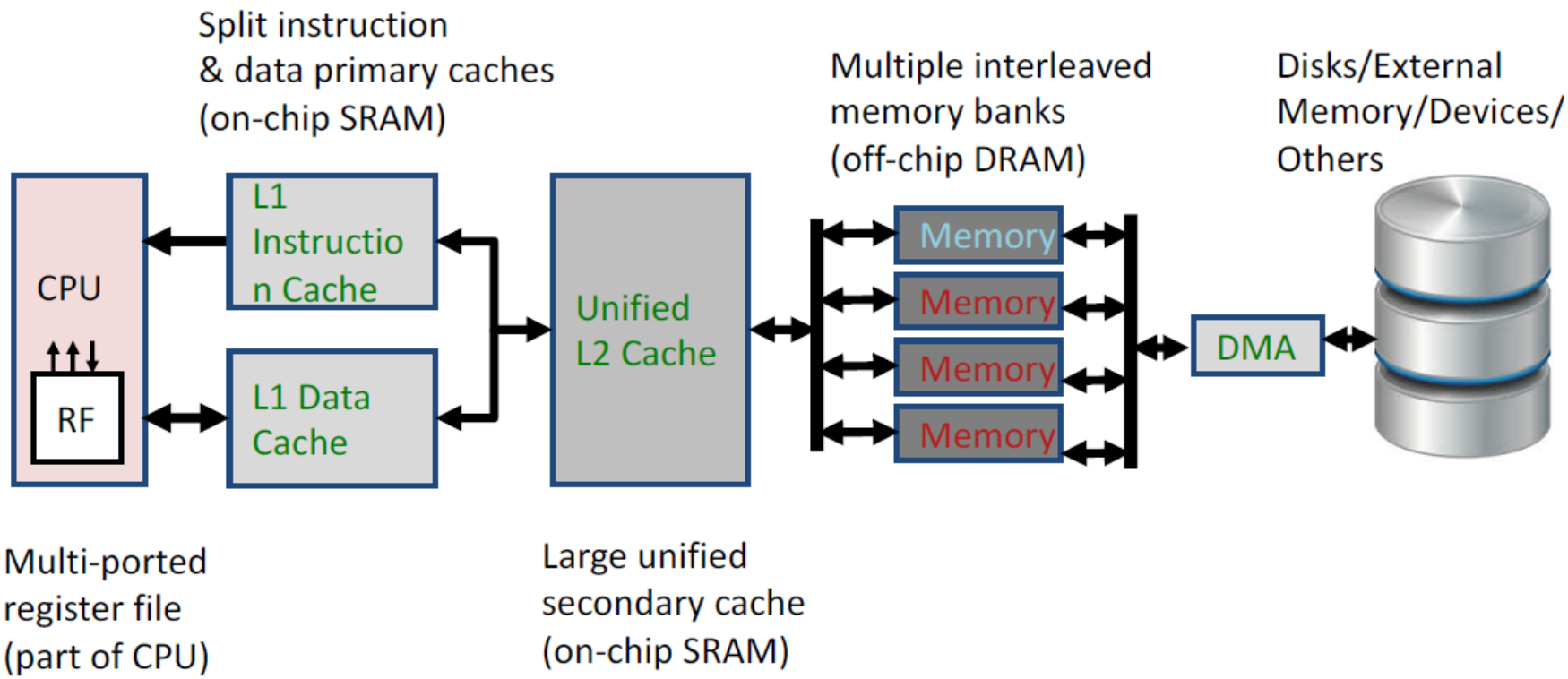


Building Large Memories

- Problem: (a) Larger is slower AND (b) Faster is expansive
- How to make large memory fast too?
- Solution (a): Divide large memory into multiple banks, which can be accessed independently as well as simultaneously
  - Each bank is smaller than the entire memory
  - Different memory banks can be accessed in parallel

Building Large Memories Cont'd

- Solution (b): Organize memory hierarchically



Memory Optimizations

- Double Data Rate (DDR): Transfer data on both rising and falling edge of the clock signal.
- Quad Data Rate (QDR): Two DDRs with separate inputs and outputs
- Multiple banks on each DRAM/SDRAM device

Principle of Locality

- Hierarchical organization of memory works primarily because of the `Principle of Locality` (POL)
- POL works because a program accesses a relatively small address space at a given time instant
- Stated otherwise, as a well known saying, that a processor spends 90% of the time on 10% of the code

Principle of Locality Cont'd

Two types of `Locality`

- `Temporal Locality`: If an item is referenced, it is going to be referenced again soon (e.g., loops, reuse)
- `Spacial Locality`: If an item is referenced, items at nearby addresses will tend to be referenced soon (e.g., straight-line code, array access, loops etc.)

Principle of Locality Example

- `Locality in Instructions`:
  - Instructions are referenced sequentially (Spacial locality)
  - Cycling through the loop (Temporal locality)
- `Locality in Data`:
  - Variable `sum` is referenced in each iteration (Temporal locality)
  - Accessing array elements (Spacial locality)

Example program.

```
sum = 0;
for (i = 0; i < j; i++)
    sum = sum + data[i];
```

Suggested Reading

- Read relevant sections of Chapter 5 of [Patterson and Hennessy, 2021].

Acknowledgment

- Preparation of this material was partly supported by Lampro Mellon Pakistan.

References

Patterson, D. and Hennessy, J. (2021).

Computer Organization and Design RISC-V Edition: The Hardware Software Interface, 2nd Edition.

Morgan Kaufmann.