Chapter IV: Memory architectures and technologies

IV Memoires

IV.1 Introduction to Memories

Memory is one of the fundamental components of any computing or embedded system. It plays a central role by ensuring both:

- the permanent or semi-permanent storage of programs (firmware, operating systems, applications) that define the overall behavior of the system;
- the temporary storage of data required for processing by the processor, serving as a dynamic workspace during instruction execution.

Thus, memory serves a dual purpose: it preserves essential information for system operation and provides a fast, flexible area for data processing.

Moreover, it acts as a critical interface between the processor and the rest of the system. Its access speed, storage capacity, and technological characteristics directly influence:

- overall system performance,
- power consumption,
- reliability, and
- the overall cost of the hardware solution.

In embedded systems, memory therefore goes far beyond a simple storage function. It represents a strategic component of both the hardware and software architecture, whose understanding and optimization are essential for designing efficient, reliable, and application-specific systems — particularly those constrained by size, cost, energy autonomy, or real-time requirements.

IV.2 Impact on Device Performance

Memory has a decisive impact on the performance of an embedded system. Its characteristics — access speed, capacity, and technology — directly affect the processor's efficiency in executing instructions and processing data.

A first impact concerns execution speed: if read/write operations are too slow, the processor must wait for data, reducing overall efficiency.

A second factor is storage capacity. Insufficient memory limits the size of programs and data handled. In demanding applications such as artificial vision or real-time signal processing, this limitation may compromise project feasibility.

Another crucial aspect is power consumption. In battery-powered devices — such as wireless sensors, IoT systems, or portable medical devices — memory must be optimized to reduce power usage while maintaining reliable and fast data access.

Finally, memory quality directly affects system reliability and robustness. Poorly adapted or low-quality memory can cause data loss, computation errors, or instability — unacceptable in critical applications.

IV.3 General Organization of Memory

Memory is a fundamental component of digital systems. Its organization is based on three main elements: addressing, decoding, and data management.

As illustrated in Figure 87, a memory unit consists of the following components:

- Address Bus: allows the processor to select a specific memory word. With n address lines, the memory can contain 2^n words.
- Address Decoder: receives the address input and activates a single line corresponding to the selected memory word.

- Memory Matrix: a set of cells organized into 2^n words of m bits each. Each word corresponds to a row selected by the decoder.
- Data Bus: carries information read from or written to the memory. Its width is m bits, matching the size of one memory word.
- Control Signals (Read/Write): define the operation to be performed:
  - *Read*: reads the content of the selected word and outputs it on the data bus.
  - *Write*: writes a new word into the selected row.

Figure 87 – General organization of a memory (decoder, memory matrix, and data bus)

When an address is placed on the address bus, the decoder selects the corresponding word in the memory matrix. The control signal then determines the operation:

- **Read**: the content of the memory word is transferred through a multiplexer to the data bus and sent to the processor.
- **Write**: data from the data bus are transferred through a demultiplexer to the selected memory cell, replacing the old value.

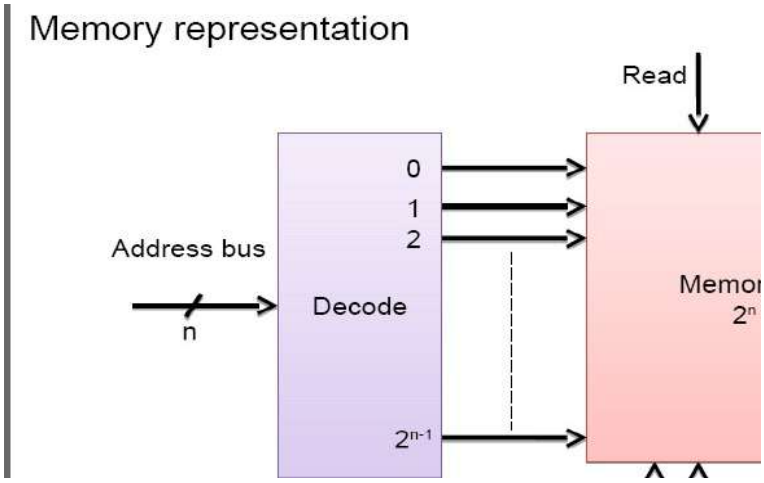

Figure VI. 1 - Read and Write process in memory

IV.4 Memory Read/Write Timing Diagram

The read/write timing diagram describes the **temporal sequence of signals** exchanged between the microprocessor and the memory to ensure correct data transfer. Each operation (read or write) follows a precise, clock-synchronized sequence.

IV.4.1 Steps of a Memory Operation

1. The microprocessor first places the target memory address on the address bus.

2. The Chip Select (CS) signal is activated to indicate that the addressed memory component should respond.
3. Depending on the operation:
    - For a read, the RD (Read) signal is activated, and the memory places the corresponding data on the data bus.
    - For a write, the WR (Write) signal is activated, and the data provided by the processor are stored in the addressed memory cell.
4. The time required for the memory to provide or record the data is called access time (Taccess).
5. The total duration of the operation, including signal setup and data stabilization, defines the memory cycle time (Tcycle).

IV.4.2 Timing Diagram

The figure illustrates the sequence of control and data signals:

- The address bus specifies the target memory position (@x).
- The CS signal enables the memory.
- The R/W signal defines the operation type (read or write).
- The data bus (BUS D) carries the exchanged information.

Two key time periods are distinguished:

- **Taccess**: the moment when data become valid;
- **Tcycle**: the duration of a full memory operation cycle.

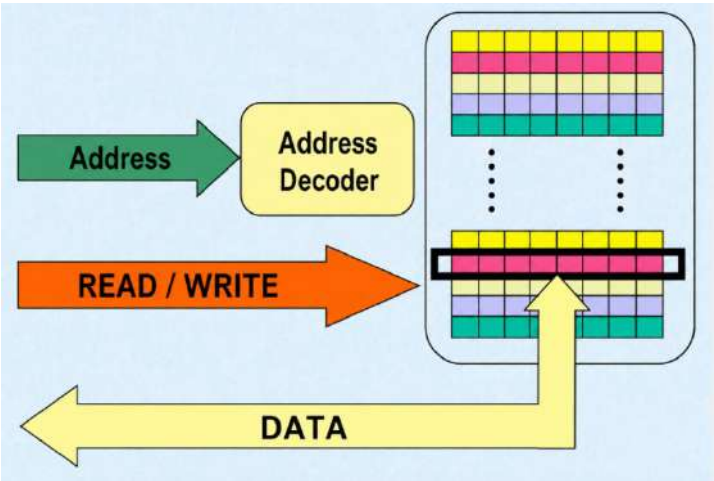

Figure VI. 2 - Memory Read/Write Timing Diagram

Respecting the timing diagram is crucial to ensure:

- proper synchronization between processor and memory,
- stable and reliable data exchange, and
- optimized performance (since long access times slow down the entire system).

In modern architectures, several techniques are used to reduce latency, such as:

- pipelined memory access (anticipating the next data read), and
- cache memories (L1, L2, L3), which store frequently used data to minimize slow main-memory accesses.

IV.5 Memory: Read/Write Timing Diagram

The read/write timing diagram describes the temporal sequence of signals exchanged between the microprocessor and the memory to ensure a correct data transfer. Each operation (read or write) follows a well-defined sequence, synchronized by the system clock.

IV.5.1 Execution of a Memory Operation

- The microprocessor first places the address of the target memory cell on the address bus.
- The Chip Select (CS) signal is activated to indicate that the addressed memory device must respond.
- Depending on the operation type:
  - For a read, the RD (Read) signal is activated, and the memory places the corresponding data on the data bus.
  - For a write, the WR (Write) signal is activated, and the data provided by the processor are stored in the addressed memory cell.
- The time taken by the memory to provide (or record) a data item is called the access time (Taccess).
- The total duration of the operation, including signal setup and data stabilization, is known as the memory cycle time (Tcycle).

IV.5.2 Timing Diagram

The figure below illustrates the sequence of signals during a memory operation:

- The **address bus** specifies the target memory position (@x).
- The **CS** signal enables the selected memory component.
- The **R/W** signal defines the type of operation (read or write).
- The **data bus (BUS D)** carries the data being exchanged.

Two important time intervals can be distinguished:

- **Taccess** – the moment when the data become valid;
- **Tcycle** – the duration of a complete memory operation cycle.

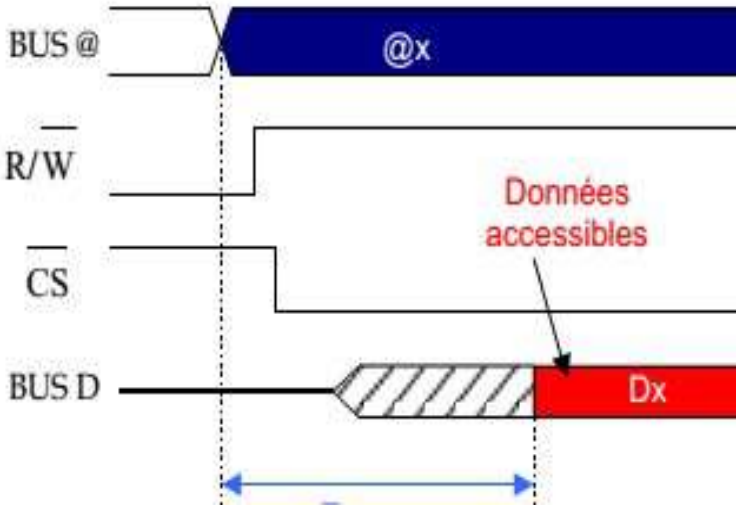

Figure VI. 3 - Memory Read/Write Timing Diagram

Respecting the timing diagram is essential to ensure:

- correct synchronization between the processor and the memory,
- stability of the data being read or written, and
- optimal system performance, since excessive access time slows down the entire system.

In modern architectures, several techniques are used to reduce this delay, such as:

- pipelined memory access (anticipating the next data read), and
- the addition of cache memories (L1, L2, L3), which store frequently accessed data to reduce slow main memory accesses.

IV.6 Memory Characteristics

The performance and role of a memory device depend on several technical parameters:

- **Capacity:**
  - Represents the size of the memory, i.e., the amount of information it can store.
  - Expressed in bits, bytes (KB, MB, GB, TB), or in the number of memory words.
  - Example: a 4 GB memory can store approximately 4 billion bytes.
- **Access Time (Taccess):**
  - The interval between a data request and its actual availability.
  - In some memories, read and write access times differ.
  - The lower the access time, the faster the memory.
- **Memory Cycle Time (Tcycle):**
  - The minimum time separating two consecutive accesses to memory.
  - Ideally close to the access time, but may include additional delays (refresh, synchronization, etc.).
- **Data Rate (or Memory Bandwidth):**
  - The amount of information transferred per second, expressed in bits/s or bytes/s.
  - Example: DDR4-3200 can reach a theoretical bandwidth of about **25.6 GB/s**.
- **Volatility:**
  - Defines whether the memory retains data when power is removed.
  - Volatile memory: RAM (data lost when power is off).
  - Non-volatile memory: ROM, Flash, EEPROM (data retained even when unpowered).

IV.7 Memory Classification

Memory occupies a central place in every embedded system. It handles the storage of programs, data, and instructions required for processor operation.
Its classification is based on several fundamental criteria:

1. Volatility: This criterion distinguishes between memories that retain information after power loss and those that lose data once power is turned off. This separation helps identify memories used for temporary storage versus those intended for permanent data retention.
2. Access Mode: Depending on their design, some memories allow sequential access (data read in a fixed order), while others support random access (direct access to any address). This feature directly affects both speed and the method of reading or writing data.

3. Functional Role: Some memories serve as working space for ongoing operations (temporary storage), while others are dedicated to permanent storage of programs and system data.
4. Technology: Memories can be built using magnetic, optical, or semiconductor technologies, depending on the desired performance, capacity, and cost.

This diversity of criteria allows memory to be grouped into two main families:

- Volatile memories, used for fast, temporary data processing.
- Non-volatile memories, designed for long-term information storage.

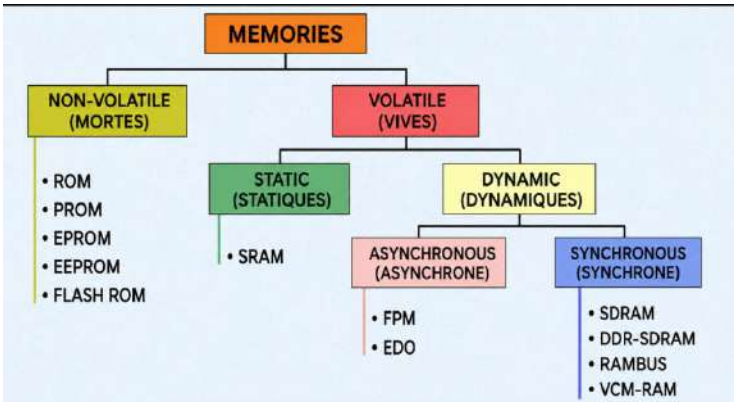

Figure VI. 4 - General Classification of Memories in Embedded Systems

IV.8 Non-Volatile Memories

Non-volatile memories are those that retain stored data even after power is turned off. They play a crucial role in embedded systems, as they store the main program (firmware), user parameters, and system boot information.

Unlike volatile memories, they do not require continuous power to preserve data, which ensures the persistence of embedded software and the stability of the device.

These memories are divided into several categories according to their structure, programming method, and rewriting capability. The main types include Read-Only Memories (ROM), mass storage memories, and Flash memories.

Read-Only Memories, or ROM, are designed to hold permanent data. They are generally programmed once and used to store the main program code or the instructions required for system initialization.

In embedded systems, ROM ensures reliable operation by maintaining a stable content, even after multiple power cycles. Different types of ROM exist depending on the programming and rewrite capabilities.

A) ROM

ROM (Read-Only Memory) is a non-volatile memory whose content is permanently defined during the manufacturing process. It is the simplest and most stable form of read-only memory. The data stored in ROM are written by the manufacturer and cannot be modified by the user. This property guarantees high reliability, since the information remains intact even without electrical power. In embedded systems, ROM is typically used to store:

- Boot programs (Bootloaders);
- Processor or microcontroller initialization routines;
- Constant tables or microcodes required for internal CPU operation.

ROM is organized as a matrix of memory cells arranged in rows and columns. Each cell represents one bit (0 or 1), and its state is determined by the presence or absence of a physical connection (such as a diode or transistor).

The general operation of a ROM involves three main blocks:

- **Address decoder:** receives an *N*-bit binary address and selects the corresponding row in the matrix.
- **Memory matrix:** contains the pre-programmed cells located at the intersections of rows and columns.
- **Output multiplexer:** gathers the data read from the selected columns and sends them to the data bus.

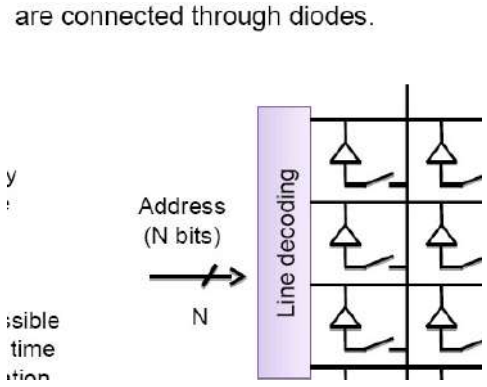

Figure VI. 5 - Simplified structure of a ROM showing address decoding and data reading

When an address is applied at the input, the decoder activates the corresponding row, allowing the multiplexer to retrieve the stored bit values (0 or 1) from the associated cells.

Thus, ROM performs only read operations, with no possibility of writing or later modification.

IV.8.1 PROM

PROM (Programmable Read-Only Memory) is an evolution of classical ROM. Unlike standard ROM, it is blank at manufacture and can be programmed once by the user using a device called a PROM programmer.

This memory type is particularly useful when the manufacturer needs to record specific programs or data after production, such as for client configurations or industrial prototypes.

PROM operates with a matrix of memory cells similar to ROM, but with electrical connections that can be permanently modified during programming. Each memory cell contains a fuse or antifuse, which is the key element used to store data, as shown in the figure below.

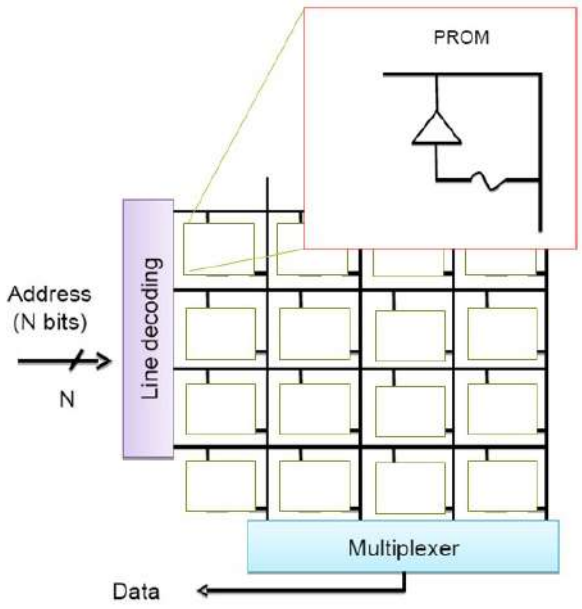

Figure VI. 6 - Simplified structure of a PROM – information encoding using fuses

In a PROM, each cell corresponds to one bit of information:

- When the fuse is intact, it represents a logical 1.

- When a high electrical pulse is applied by the programmer, the fuse blows, breaking the circuit, which corresponds to a logical 0.

This electrical burning process is irreversible: once a fuse is blown, it cannot be restored. PROMs are therefore one-time programmable (OTP) memories.

The essential components of a PROM cell are:

- The selection transistor, which identifies the cell to be programmed.
- The fuse, which stores the binary value depending on its state (intact or blown).

PROMs combine the stability of classical ROM with limited flexibility for post-manufacturing customization. However, because they cannot be reprogrammed, their use is restricted to applications where data must never change, such as industrial embedded systems, identification circuits, and fixed-configuration microcontrollers.

IV.8.2 EPROM

EPROM (Erasable Programmable Read-Only Memory) is a non-volatile memory that, unlike PROM, can be erased and reprogrammed multiple times.
It was developed to provide a flexible solution for embedded system designers, allowing them to modify the memory content without manufacturing a new chip for each update.

The key element of an EPROM is the floating-gate transistor, as illustrated in the figure below.

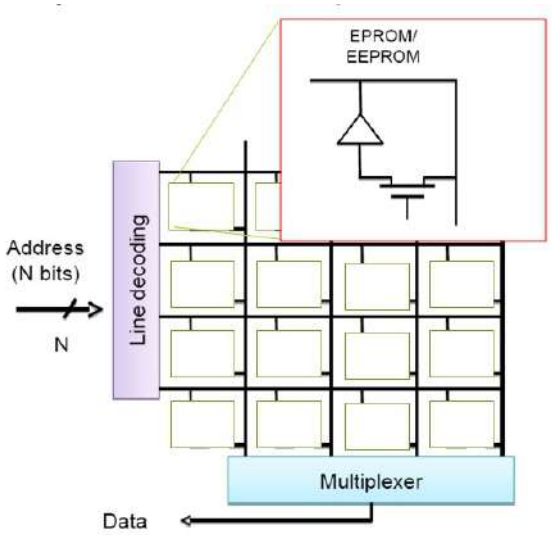

Figure VI. 7 - Simplified structure of an EPROM/EEPROM – floating-gate memory cell

Each EPROM cell is made up of a modified MOS transistor with two gates:

- a control gate, connected to the memory's control circuitry;
- a floating gate, electrically isolated by an oxide layer and capable of storing electric charges.

During programming, a high voltage is applied, causing electrons to become trapped on the floating gate. These electrons remain stored as long as no erase operation is performed, thereby modifying the transistor's conduction threshold. Depending on the presence or absence of electrons on the floating gate, the cell is interpreted as a logical 0 or 1.

The erasure of an EPROM's content is achieved using ultraviolet (UV) radiation with a wavelength of approximately 253 nm. To do this, the chip is exposed to UV light through a small quartz window integrated into its package. This radiation releases the trapped electrons in the floating gate, effectively resetting the memory for a new programming cycle.

In the figure, the EPROM cell includes:

- a selection transistor, allowing access to the cell according to the specified address;
- a floating-gate transistor, which stores the electrical charge corresponding to the data bit.

These cells are arranged in a matrix, controlled by an address decoder and connected to an output multiplexer, similar to other ROM architectures.

EPROM is therefore a non-volatile, reusable memory after complete erasure by UV exposure.
It offers excellent data stability and good charge retention, but its UV-based erasure is relatively slow (several minutes) and requires physical exposure, which limits its use in systems that need frequent updates.

Thanks to its reprogrammability and low cost, EPROM was widely used during the 1980s and 1990s for the development and prototyping of microcontrollers, before being gradually replaced by EEPROM and Flash technologies, which are faster and more convenient.

IV.8.3 EEPROM

EEPROM (Electrically Erasable Programmable Read-Only Memory) is a non-volatile memory that allows electrical erasure and reprogramming, without exposure to ultraviolet

light. It also uses floating-gate transistors capable of storing an electric charge that represents a data bit.

During programming, an appropriate voltage injects electrons into the floating gate, changing the logical state of the cell. Erasure is achieved by applying a reverse voltage that releases these electrons, restoring the cell to its initial, writable state.This operation can be performed selectively, cell by cell, without erasing the entire memory.

EEPROM offers several advantages:

- it retains data without power,
- it allows electrical updating of the content,
- it supports a large number of write and erase cycles.

However, it has slower access times and a lower storage capacity than other memory types.

EEPROMs are commonly used in embedded systems to store configuration parameters, calibration values, or persistent information that must be preserved even after a power loss.

IV.8.4 Flash Memory

Flash memory is a type of non-volatile memory derived from EEPROM technology. It retains data without electrical power and provides high storage density, fast access speeds, and quick reprogramming capabilities. Unlike EEPROM, which allows cell-by-cell erasure, Flash memory performs erasure by blocks, which reduces erase/write time and simplifies internal control.

Flash memory also relies on floating-gate transistors that store electrical charges representing binary data. During programming, a high voltage injects electrons into the floating gate, modifying the transistor's threshold voltage. Erasure is performed by applying a reverse voltage to remove the trapped electrons, restoring the cells for new programming.

Two main architectures exist depending on the organization of the memory cells: NOR and NAND.

- In NOR Flash, the cells are connected in parallel, enabling direct and random access to data. This configuration allows byte-level read and write operations,

making it ideal for executable code storage (firmware). However, it takes up more space and offers lower storage density.

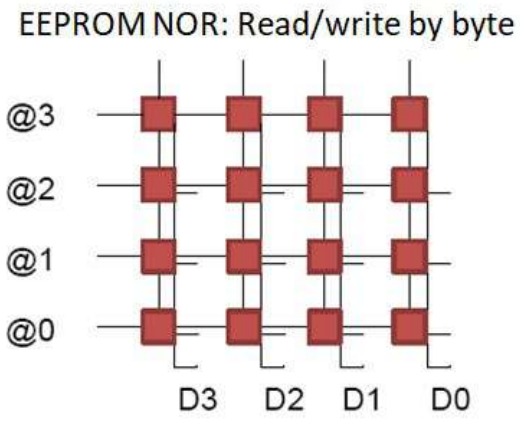

Figure VI. 8 - Organization of a NOR-type EEPROM memory – byte-level read/write operations

- In NAND Flash, the cells are connected in series within chains or blocks, allowing sector-level read/write operations. This design increases storage density and reduces manufacturing cost, while improving write speed. It is particularly well-suited for mass storage applications such as SD cards, USB drives, and SSD disks, where large amounts of data must be accessed quickly.

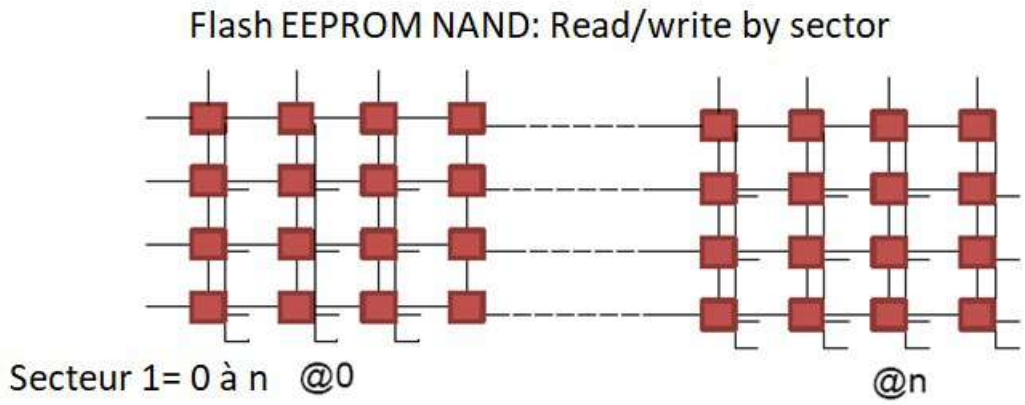

Figure VI. 9 - Organization of a NAND-type Flash memory – sector-level read/write operations

Flash memory provides numerous advantages:

- data retention without power,
- fast access,
- compact size, and

- low power consumption.

However, its lifespan is limited by the number of write/erase cycles supported by each cell, and reprogramming requires full block erasure.

In embedded systems, Flash is often integrated into microcontrollers and used to store the main program, configuration parameters, or critical data. Thanks to its stability and performance, Flash memory has become an essential component of modern embedded architectures.

IV.8.5 Mass Storage Memories

Mass storage memories are devices designed to store large amounts of data permanently, even without power. Unlike main memories such as RAM, which are used for temporary data processing, mass storage provides long-term data retention.

They are used to store files, programs, operating systems, or databases in complex embedded systems. The operation of mass memories is based on different technologies, mainly distinguished by the storage medium and access mode. The three main families are:

- Magnetic memories, such as hard disk drives (HDDs) and magnetic tapes, which use magnetic particle polarization to represent bits. These offer large capacity at low cost, but are sensitive to shocks and consume more energy, making them less suitable for modern embedded systems.
- Optical memories, such as CDs, DVDs, and Blu-ray discs, which use a laser beam to read or write microscopic patterns on the surface of the disc. They are durable and mainly used for archival storage, but less suited for active storage in embedded devices.
- Semiconductor memories, such as SSDs, SD cards, and USB drives, based on Flash technology. These have largely replaced magnetic media in most embedded applications due to their compactness, fast access, shock resistance, and low energy consumption.

In embedded systems, mass memories are essential for storing system logs, configuration data, and sensor-acquired information. Their selection depends on factors such as required capacity, access speed, reliability, and energy efficiency.

Thus, mass storage memories represent the permanent storage level of embedded systems, ensuring the preservation and availability of essential operational data.

IV.9 Volatile Memories

Volatile memories are temporary storage devices that lose their contents once power is removed. They constitute the working memory of an embedded system and play a crucial role in program execution. The processor uses them to store intermediate data, temporary variables, and currently processed instructions. Their access speed is a key factor in the system's overall performance, as they are involved in every execution cycle.

The most common type of volatile memory is RAM (Random Access Memory), which exists in several variants depending on the underlying technology and system requirements.

RAM is a *random-access* memory, meaning that each cell can be directly accessed through its address, without scanning other cells. It serves as the main memory in an embedded system and provides the workspace for the processor. Unlike non-volatile memories, it is cleared each time the system restarts.

Its organization is based on a matrix of cells arranged in rows and columns. Each cell stores a bit of information — typically using a transistor and a capacitor (in DRAM) or only transistors (in SRAM).

Two main families are distinguished:

- **DRAM (Dynamic RAM)**
- **SRAM (Static RAM)**

These differ in internal structure and performance characteristics.

IV.9.1 SRAM

SRAM (Static Random Access Memory) is a very fast volatile memory used primarily in embedded systems that require instant data access. Unlike DRAM, it does not need periodic refresh cycles to retain data, as the information remains stable as long as power is supplied.

Each SRAM cell consists of six MOS transistors forming a bistable latch capable of storing one bit. Transistors T3 and T4, combined with T5 and T6, form two cross-coupled inverters that maintain the logical state. Transistors T1 and T2 act as access gates for reading and writing operations.

When a Word Line (WL) is activated, these transistors connect the cell to the Bit Lines (BL), enabling data transfer without disturbing the stored value.

All cells are organized in a matrix, addressed through row and column decoders. The row decoder selects the word line, while the column decoder identifies the bit line. Data then passes through a multiplexer/demultiplexer (Mux/Demux) that manages data flow during read or write operations.

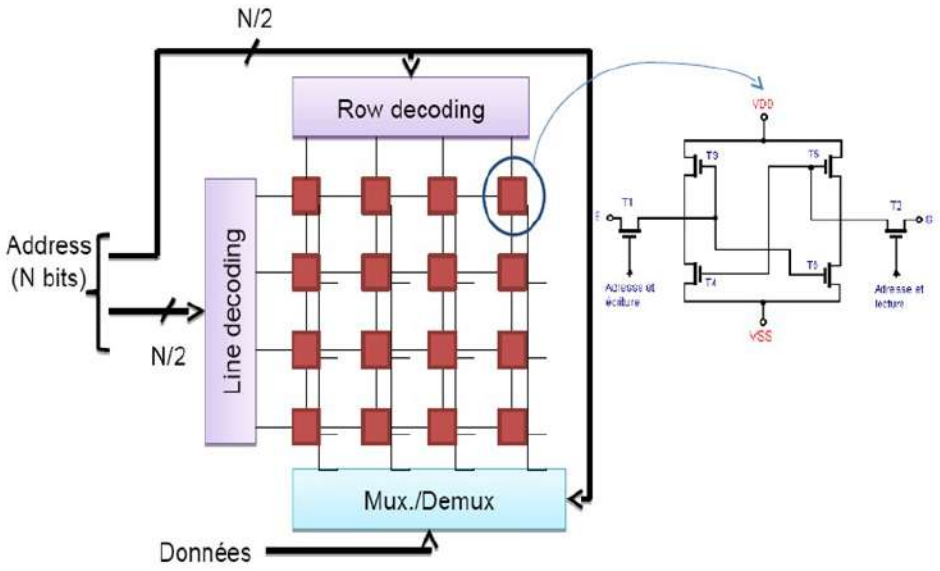

Figure VI. 10 - Internal organization of an SRAM and 6T basic memory cell

This architecture ensures high speed and data stability, at the cost of lower density and higher cost per bit compared to DRAM.

Thanks to its speed and reliability, SRAM is widely used in:

- processor cache memory (L1, L2, L3),
- internal registers,
- high-speed communication buffers, and
- embedded memories in microcontrollers, FPGAs, and ASICs.

It is thus an essential component for high-performance embedded applications.

IV.9.2 DRAM

DRAM (Dynamic Random Access Memory) is a type of volatile memory in which each cell is made up of one transistor and one capacitor. The transistor acts as a switch controlling access to the cell, while the capacitor stores the electrical charge that represents the data bit:

- a **charged capacitor** corresponds to a logical **1**,
- a **discharged capacitor** corresponds to a logical **0**.

Because the capacitor's charge gradually leaks away, the contents of DRAM must be refreshed periodically to avoid data loss. This refresh operation makes DRAM slower than SRAM, but allows greater density and lower cost per bit.

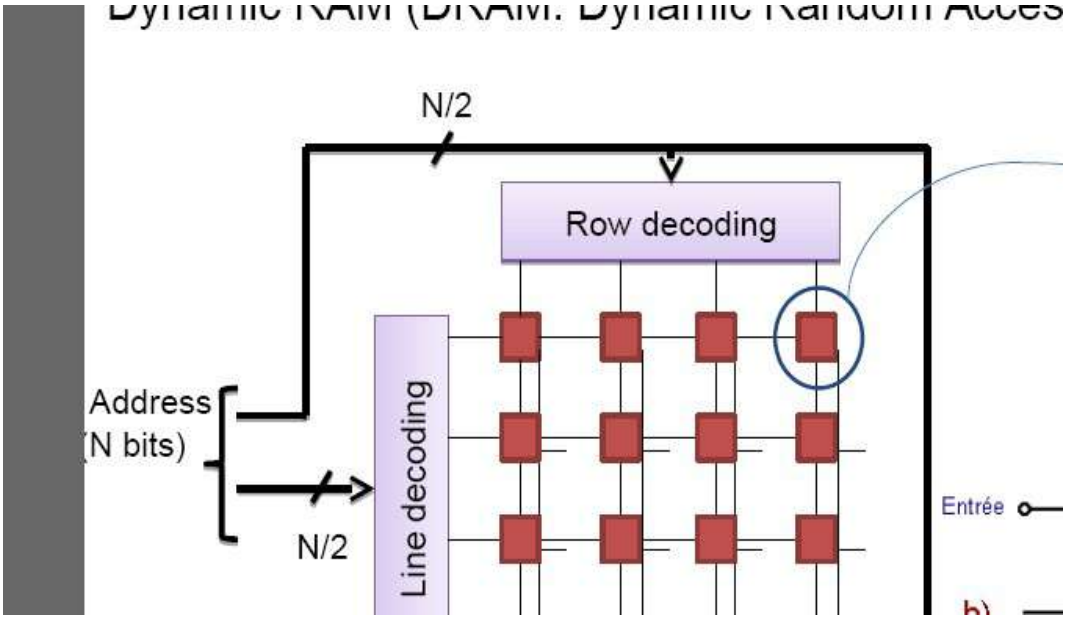

Figure VI. 11 - Structure of a DRAM and its basic 1T/1C memory cell

The DRAM matrix is composed of numerous cells arranged in rows and columns, which are selected by decoding circuits. A multiplexer/demultiplexer manages read and write operations, while the cell structure on the right shows the dynamic storage principle.

This architecture makes DRAM the preferred choice for main memory in many embedded systems requiring large capacity at low cost.

There are two primary types of DRAM, distinguished by their operational mode:

- Asynchronous DRAM, where read and write operations are not synchronized with a clock signal. The processor must wait for one cycle to complete before starting another.
- Synchronous DRAM (SDRAM), which operates in sync with the processor clock, enabling fast and continuous data access.

Because of its simple yet efficient design, DRAM remains ubiquitous in embedded systems, offering a balance between capacity, cost, and compatibility with a wide range of processors and microcontrollers.

A) Asynchronous DRAM

Asynchronous DRAM is the oldest and simplest form of dynamic memory. Its name comes from the fact that its operation is not synchronized with any external clock. Read and write operations are controlled by two main signals:

- **RAS (Row Address Strobe)**: activates the target row,
- **CAS (Column Address Strobe)**: activates the target column.

In an asynchronous DRAM, the processor must wait for each access cycle (read or write) to complete before initiating the next one. This results in longer access times, since each operation must be fully processed before moving to the next.

However, the simplicity and low cost of this architecture made it popular in early embedded systems and in applications where speed was not critical. Notable variants include:

- FPM DRAM (Fast Page Mode) – improves performance by keeping a row active for multiple consecutive accesses.
- EDO DRAM (Extended Data Out) – further reduces cycle time by preparing the next data access while the current one is still being read.

These improvements paved the way for Synchronous DRAM (SDRAM), which operates in coordination with the system clock, providing significantly better performance while maintaining the same basic structure.

B) Synchronous DRAM (SDRAM)

SDRAM (Synchronous DRAM) is a major improvement over asynchronous DRAM, as it operates synchronously with the processor clock. This synchronization allows ordered read/write operations, eliminating idle waiting periods and enabling seamless data transfers. Multiple accesses can be chained together without interruption, significantly increasing transfer speed and memory bandwidth.

Structurally, SDRAM retains the dynamic memory principle — a cell made of one transistor and one capacitor — but adds clock-synchronized control circuits.

These circuits manage data flow efficiently through pipelining mechanisms, which prepare one access while the previous one is still being executed.

Successive generations of SDRAM have brought major improvements:

- **DDR (Double Data Rate SDRAM)** – doubles throughput by transferring data on both the rising and falling edges of the clock.
- **DDR2, DDR3, DDR4, and DDR5** – progressively increase frequency, reduce supply voltage, optimize power efficiency, and expand storage density.

Thanks to these developments, SDRAM and its derivatives have become the standard memory in modern embedded and computing systems. They are widely used in development boards (e.g., Raspberry Pi, high-end STM32), embedded processors, and real-time data processing systems such as robotics, computer vision, and communication equipment.

IV.10 Common Memory Issues in Embedded Systems

Embedded systems can face several critical problems related to memory management.

One major issue is memory fragmentation, which occurs when free space becomes divided into small, non-contiguous blocks, preventing the allocation of new memory areas even if total capacity seems sufficient. This often happens in systems using dynamic memory allocation.

Another frequent problem is memory leakage, which occurs when allocated memory blocks are not released after use. Over time, this reduces available memory, leading to slower performance or system crashes. To prevent these issues, good management practices are essential:

- minimize dynamic allocation,
- favor static data structures,
- monitor pointer integrity, and
- use memory debugging tools to detect errors.

In critical systems, strict memory management is indispensable to ensure long-term reliability and stability.

IV.11 Advanced Aspects of Memory Management

Modern embedded architectures integrate advanced mechanisms to optimize memory use and efficiency.

Virtual memory allows simulation of a capacity greater than physical memory by using secondary storage, enhancing flexibility and multitasking capabilities. Prefetching techniques anticipate processor needs by preloading data likely to be requested soon, thereby reducing wait times during execution.

Memory access optimization also involves:

- instruction and data caching,
- minimizing access conflicts, and
- hierarchical memory organization (L1, L2, L3).

These methods improve system responsiveness while reducing power consumption. In demanding embedded applications — such as automotive systems, robotics, and real-time processing — mastering these memory management techniques is key to achieving high performance and reliability.