**Chapter 5: Internal Memory**

Table of Contents

[5.1 Semiconductor Main Memory 2](#_Toc48982598)

[RAM 3](#_Toc48982599)

[ROM 6](#_Toc48982600)

[Chip Logic 8](#_Toc48982601)

[Chip Packaging 8](#_Toc48982602)

[Module Organization 9](#_Toc48982603)

[Interleaved Memory 9](#_Toc48982604)

[5.2 Error Correction 10](#_Toc48982605)

## 5.1 Semiconductor Main Memory

In earlier computers, the most common form of random-access storage for computer main memory used doughnut shaped ferromagnetic loops, called cores. Today, we use semiconductor chips, and this is what will be discussed in this section.

The basic element of a semiconductor memory is the memory cell. There are differences in electronic technology, but all semiconductor memory cells have the following properties:

* They have two stable, or semi-stable states, used to represent binary 0 or 1
* We can write onto them, at least once, to set the state
* We can read from them, to retrieve the state

This is essentially how a single memory cell is set up:



There are thee terminals that carry electric signals. The select terminal is used to select a cell for a read or write operation. The control terminal indicates whether the operation is a read or a write. The third terminal either sets the state for a write operation, or outputs the state for a read operation. There are of course, more complicated details of internal organization, functions and timing that depend on which integrated circuit technology is being used with the memory cells, but for now we will assume that individual cells can be selected for reading or writing.

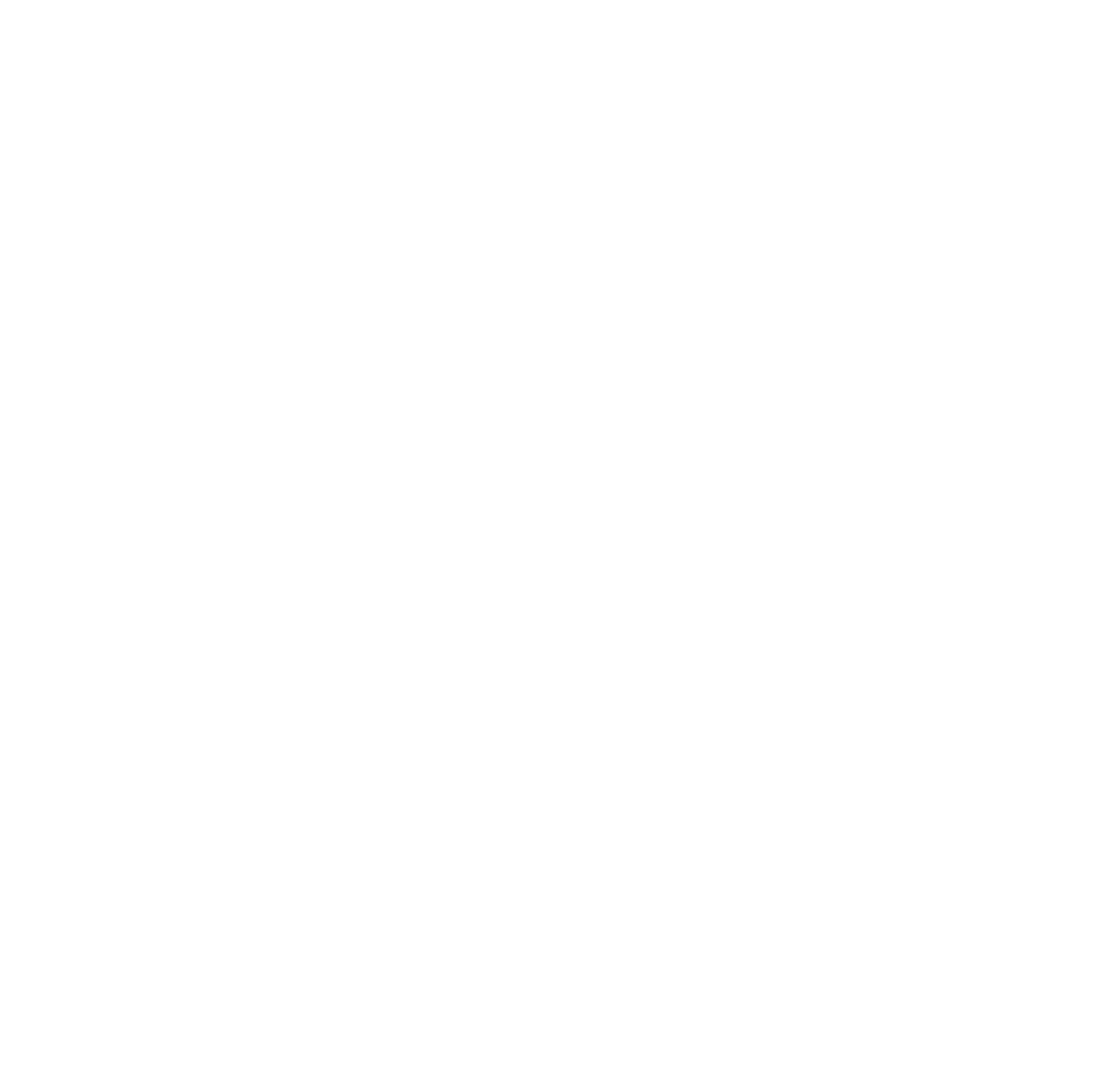
The table below lists different types of semiconductor memory:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Memory Type** | **Category** | **Erasure** | **Write Mechanism** | **Volatility** |
| Random-access memory (RAM) | Read-write memory | Electrically, byte-level | Electrically | Volatile |
| Read-only memory (ROM) | Read-only memory | Not possible | Masks | Non-volatile |
| Programmable ROM (PROM) | Electrically |
| Erasable PROM (EPROM) | Read-mostly memory | UV light, chip-level |
| Electrically Erasable PROM (EEPROM) | Electrically, byte-level |
| Flash Memory | Electrically, block-level |

### RAM

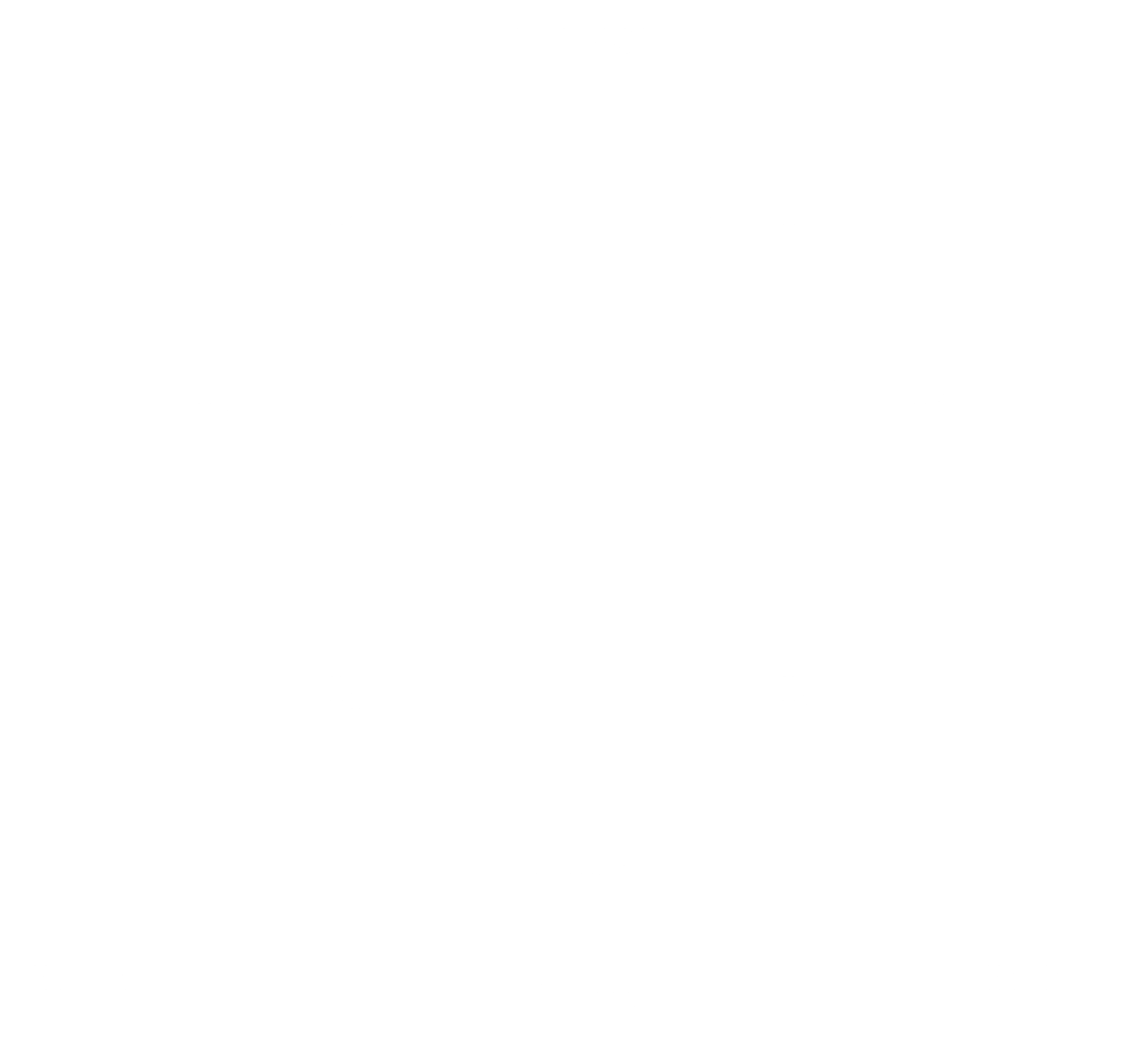
The most common type of memory is random-access memory (RAM). In reality, every type of memory listed in the table is random-access, meaning individual words of memory can be directly accessed through wired-in addressing logic. The distinguishing characteristic of RAM is that we can both read and write data using electrical signals, and we can do it very easily and quickly. RAM is also volatile, meaning that if it loses power, all the data on it is lost. Because of this, RAM can only be used temporary storage. Traditionally, computers use one of two forms of RAM, DRAM or SRAM. There are however, newer forms of RAM that are non-volatile.

Dynamic RAM, or DRAM, is made of cells that store data as charge on capacitors. The presence or absence of charge is interpreted as a binary 1 or 0 respectively. Capacitors however, have a natural tendency to discharge, which means DRAM requires a periodic refreshing of charge to maintain data storage. The term dynamic, refers to this tendency of the stored charge to leak away, even with power continuously being supplied.



The above figure describes the typical structure for an individual cell for DRAM that stores a single bit. When we want to read or write a bit value, the address line is activated. The transistor acts like a switch. It is closed and allows current to flow if voltage is applied to the address line, and is open, preventing current flow, if no voltage is applied to the address line. During a write operation, the bit line is given either a high or low voltage, representing a 1 or 0. This, along with the transistor being switched on, allows charge to be transferred to the capacitor. For a read operation, when the transistor is switched on, the charge from the capacitor flows onto the bit line and onto a sense amplifier, which compares the voltage to a reference value to determine if the cell logic is 1 or 0. The readout discharges the capacitor, so it must be recharged before the operation is complete. We use DRAM to store a single bit, but it is essentially an analogue device. The capacitor could store any charge value within a range, and a threshold value is used to determine if the charge is to interpreted as a 1 or a 0.

Static Ram, or SRAM, on the other hand, is a digital device and uses the same logic elements as a processor. Binary values are stores using traditional flop-flop logic-gate configuration. SRAM can hold its data for as long as power is supplied to it.



This is the figure for a typical SRAM structure for an individual cell. Four transistors, to , are cross-connected to produce a stable logic state. For logic 1, is high and is low. This causes and to be off and and to be on. For logic 0, the opposite happens. Both states are stable as long as the dc voltage is applied. Unlike DRAM, no refresh is needed to retain data. The address line is used to open or close a switch, just like in DRAM. The transistors and are controlled using the address line. When a voltage is applied to the address line, these two transistors are switched on. For a write operation, the bit value is applied to line , while its complement is applied to line . This forces the four transistors to into the proper state. For a read operation, the bit value is read from line .

Both SRAM and DRAM are volatile, so power must continuously be supplied to preserve the bit values. DRAM is simpler and smaller, which means it is denser (more memory cells can be packed into the same area) and less expensive. However, it also needs supporting refresh circuitry. For larger memories, the fixed cost of this circuitry is compensated by the small cost of DRAM cells, making DRAM favourable. SRAM however, is somewhat faster. Thus, DRAM is used for main memory, while SRAM is used for cache memory.

### ROM

Read-Only Memory, or ROM, contains a permanent pattern of data that cannot be changed. ROM is non-volatile, meaning no power source is required to maintain the bit values in memory. It is possible to read from ROM, but not write onto it. The advantage of ROM is that data is permanently stored in main memory and never needs to be loaded from a secondary storage device. Applications of ROM include:

* Micro-programming
* Library subroutines from frequently used functions
* System programs
* Function tables

ROM is created like any other integrated circuit chip, with the data wired into the chip as part of the fabrication process. This causes two problems:

* Data insertion has a large fixed cost, whether one copy of a particular ROM is created, or a thousand
* There can be no errors. If one bit is wrong, the entire batch of ROMs is wasted

When only a small number of ROMs with some particular memory content is needed, a less expensive alternative, programmable ROM or PROM, is used. PROM is also non-volatile and can be written into only once. The writing process is performed electrically, and can be performed by a supplier or customer after the original chip fabrication. Special equipment is needed for this writing, or ‘programming’, process. PROM provides flexibility and convenience, while ROM is attractive for high-volume production.

There is another type of read-only memory called read-mostly memory, which is used for applications where read operations are far more frequent than write operations, but for which non-volatile storage is required. The most common forms of read-mostly memory are EPROM, EEPROM and flash memory.

Optically Erasable PROM, or EPROM, is read and written electrically. Before a write operation, all storage cells must be erased to the same initial state by exposing the packaged chip to UV radiation. This is done using a window designed into the memory chip. The erasure can be performed repeatedly, but takes up to 20 minutes. EPROM can be altered multiple times and holds its data virtually indefinitely. For similar amounts of storage, EPROM is more expensive than PROM, bit it has the advantage of multiple update capability.

Electrically Erasable PROM (EEPROM) is more attractive than EPROM. It can be written into at any time without erasing prior contents, with only the byte or bytes addressed being updated. The write operation takes much longer than the read operation, up to several hundred microseconds per byte. EEPROM combines nonvolatility with the flexibility of being updatable in place, using ordinary bus control, address and data lines. EEPROM is more expensive than EPROM, and is also less dense.

Flash memory is an intermediate between EPROM and EEPROM in terms of cost and functionality. It uses electrical erasing technology, making erasure very fast. An entire flash memory can be erased in a few seconds. It is also possible to erase individual blocks of memory. Flash memory gets its name due to the way the microchip is organized. A section of memory cells can be erased in a single action, or ‘flash’. However, flash memory does not allow byte-level erasure. Like EPROM, flash memory uses only one transistor per bit, thus achieving high levels of density.

### Chip Logic

Like other integrated circuit products, semiconductor memory comes in packaged chips. Each chip contains an array of memory cells. The trade-offs we saw in memory hierarchy as a whole are also present in organization of memory cells and functional logic on a chip. One of the key design issues is how many bits can be read or written at a time. One way is an organization where the physical arrangement of cells in the array is the same as the logic arrangement, so the array is divided into words of bits each, where is the word size the system works with. The complete opposite method is to simply read or write data one bit at a time.

### Chip Packaging

An integrated circuit is mounted on a package that contains pins for connection. These pins support the following signal lines:

* Address of word being accessed ()
* Data to be read out ()
* Power Supply ()
* Ground ()
* Chip Enable () pin – this indicates whether or this chip is being used
* Program Voltage () supplied during programming (for EPROM)

RAM chips have a few differences such as data pins being used for input and output, and write enable () and output enable ) pins to indicate whether it is a write or read operation. Ram is also organized differently, in rows and columns, so instead of the pin, there are row address select () and columns address select ) pins.

### Module Organization

Module organization deals with how the memory chips are organized. For example, if we have a RAM chip that contains only one bit per word, then we will need at least a number of chips equal to the number of bits per word, since data is loaded a word at a time into RAM. If our memory module consists of 256,000 8-bit words, then we need an 18-bit address. This address is presented to the 8 separate sets of 256,000 1-bit chips, each of which provide the input or output of one bit.

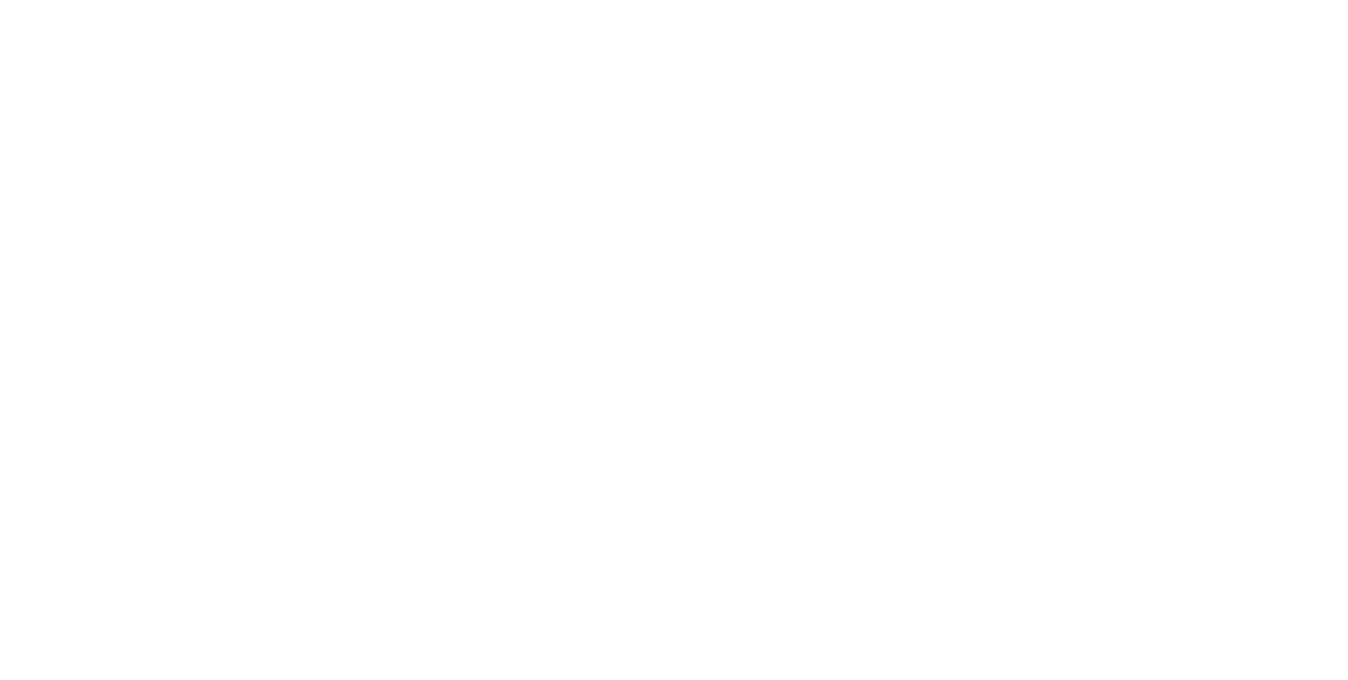
If the word size was larger, say 1M, then we would need to arrange he chips in an array, so that we can create 1M with 8-bits per word.

### Interleaved Memory

Main memory is made of a collection of DRAM memory chips. A number of such chips can be grouped together to form a memory bank. It is possible to organize the memory banks in a way called interleaved memory. Each bank is able to independently service a memory read or write request, so a system with banks can service requests simultaneously, thus increasing memory read and write speeds by a factor of . If consecutive words of memory are stored in different banks, then the transfer of a block of memory is speeded up.

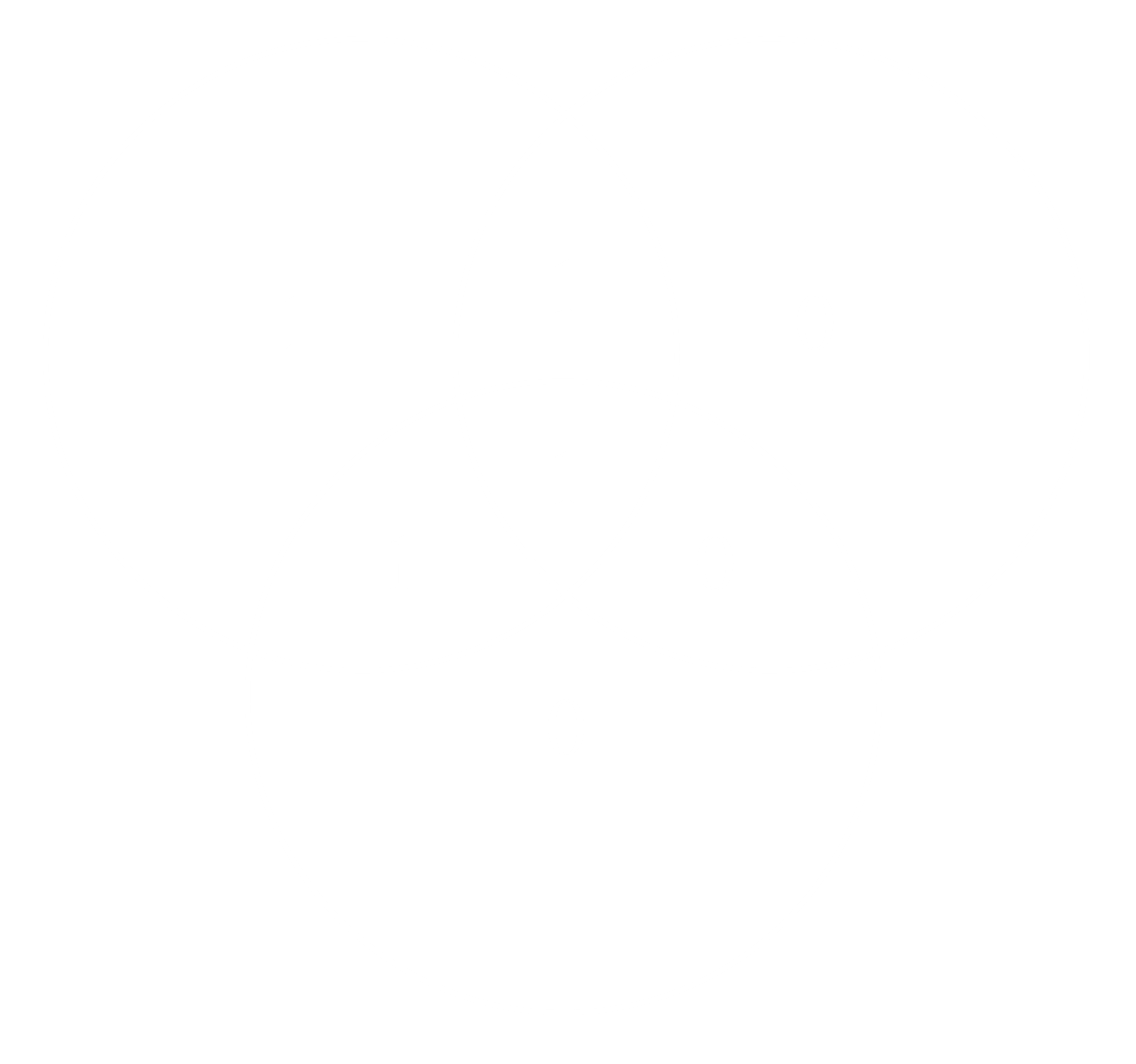
## 5.2 Error Correction

A semiconductor memory system may sometimes get errors. These can be categorized into hard failures and soft errors. Hard failures are permanent physical defects in memory cells that prevent them from reliably storing data. They become stuck at 0 or 1 or switch frequently and erroneously between the two values. Hard errors are caused by harsh environmental abuse, manufacturing defects or normal wear over long periods of usage. Soft errors are random, non-destructive events that alter the contents of one or more memory cells without damaging the memory physically. These are caused by power supply problems or alpha particles, that arise from radioactive decay from nearly all materials. Most modern main memory systems include logic to detect and correct errors.

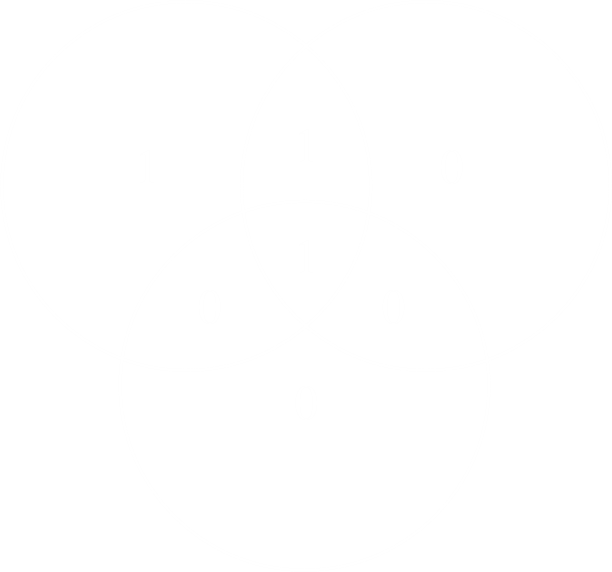


When an bit word of data is being stored, a function is performed on it to produce a code of bits which is stored with the word. When the word is read, a new code is generated by and is compared with the old code. If the codes are the same, then there are no errors. Otherwise, either the detected error is corrected or, if it is not possible to correct, the error is reported. If it can be corrected, then the word is passed to the corrector, along with error correction bits that give information about what type of error has occurred. Code that helps to correct erroneous data is called error correcting code.

The simplest error correcting code is Hamming code. This can be described using the following Venn diagram:



For a 4-bit word, , we put each bit in one of the inner compartments. In each circle, the outer compartment is if the number of s in the inner compartments related to it is odd, and if the number of s is even. The bits in the outer compartments are called parity bits. Now say an error occurs and a single bit in the 4-bit word changes. The Venn diagram becomes like this:



The top left outer compartment tells us that there should be an odd number of s in its connected inner compartments, but there are an even number. The top left outer compartment tells us that there should be an even number of s in its connected inner compartments, which we do have. The bottom outer compartment tells us that there should be an even number of s in its connected inner compartments, but we have an odd number. Since the top left and bottom compartments are showing errors, the erroneous bit must be shared between them. Since the top right compartment is showing no error, the erroneous bit must be outside its reach. There is only one bit that is like this, the highlighted one. We have detected the erroneous bit. Notice that we can only detect single bit errors in this way. If there were multiple errors, we would be unable to detect which bits had been changed.

We know that to detect an error, we compare two codes of bits each. Each bit of the codes is compared and a result is generated. If the bits are the same, then the result for that bit is . If they are different, the result is . We will thus get a resulting code of bits. This code is called the syndrome word and indicates the position of the error.

Notice that the syndrome word will thus have a range between and . The error could occur in any of the data bits or the check bits. Thus, . From here, we can find the size of needed. For an 8-bit code, if , and . Since , must be larger. If , and . Thus, .

We now need to concentrate on how our bits are arranged. If an error occurs in the check bits, we can just ignore that after all, since the check bits don’t actually matter. In general, the 4-bit syndrome word must follow these conditions:

* If it contains all s, no error has occurred
* If it contains a single , the error is in one of the four check bits
* If it contains multiple s, the error is in the data bits. In this case, the numerical value of the syndrome indicates the position of the erroneous data bit, and that data bit is simple inverted to correct it.

To achieve these characteristics, the data and check bits are arranged in the following manner:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bit position | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| Position number | 1100 | 1011 | 1010 | 1001 | 1000 | 0111 | 0110 | 0101 | 0100 | 0011 | 0010 | 0001 |
| Data bit | D8 | D7 | D6 | D5 |  | D4 | D3 | D2 |  | D1 |  |  |
| Check bit |  |  |  |  | C8 |  |  |  | C4 |  | C2 | C1 |

The bit positions that are powers of 2 are assigned to the check bits. The check bits are labelled based on their bit position, thus , , and . The rest of the bits are assigned to the data bits.

The value of each data bit is found by performing the XOR function on each of the data bits that have a 1 in the same position as the check bit does. For example, , , , , and all have a 1 in the least significant bit position. Thus,

Similarly,

Let’s look at an example. For the 8-bit data , using the table above, , , and . Now say the third bit has an error and is changed to . Now, the values become , , and . Comparing the hamming codes, and , if we XOR the bits in each position, the resulting syndrome word is . Again, using the table above, we can see that position number is for , the third data bit. Thus, we need to invert the third bit. Visually, the entire process was:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

The code we just described is called a single-error-correcting (SEC) code ([here’s another Indian dude explain everything simply](https://www.youtube.com/watch?v=Q8xO29WZcXE)). More commonly, semiconductor memory is equipped with single-error-correcting, double-error-detecting (SEC-DED) code ([video](https://www.youtube.com/watch?v=Qnr_6ytX0Mw)).

In SEC-DED code, there is one extra bit, called the General Parity Bit (GPB). The GPB is based on the overall number of 1s in the data bits. For example, for the data bits , GBP = 0 since there are an even number of 1s. If there had been an odd number of 1s, GBP would be 1.

Regardless of whether there is a single error or a double error, we would still get a syndrome code. In the case of a double error however, the syndrome code would be worthless. This is where the GBP comes in. If there is one error, the erroneous code would have a different GBP. For example, if one of the zeroes in became a 1, then there would be an odd number of 1s in the data bits, which give us a GBP of 1, different from the original GBP. However, if there are 2 errors, then we would still have an even number of 1s. Even if a zero became a one and a one became a zero, we would still get a syndrome code and we would still get the same GBP, since the number of 1s is still even. Thus, when the there is no change in the syndrome code or the GBP, then there are no errors. If the GBP and syndrome code both change, there must be a single error, at the position indicated by the GBP. If however, there are two errors, we will get a syndrome code but the GBP will remain the same. In this case, there are two errors.