POLITECNICO DI TORINO

Master degree course in Electronic Engineering

Master Degree Thesis

SOT STT MTJ Architectures for Logic-In-Memory Computing

Supervisors
Prof. Marco Vacca
Prof. Maurizio Zamboni

Candidates
Michela Graglia

Academic year 2020 – 2021

Ai miei genitori
e a tutti i miei cari

Summary

Von Neumann architectures are based on the information transfer between the memory and the processing units: the associated computational efficiency is hindered by the interconnection delay and by the fact that these two CMOS units work at different speeds. Furthermore, the ongoing advancement in CMOS scaling makes the static power consumption become higher and higher every year. For these reasons, *Logic-in-Memory* (LiM) design is seen as a promising candidate for overcoming such limitations: it consists in the introduction of some computational features inside the memory itself, in order to reduce the data transfer between memory and CPU, increase processing speed and lower the power consumption. Moreover, the exploration of Beyond–CMOS technologies for memory implementation aims to find an efficient alternative to CMOS technology capable of solving the leakage problem and providing better performance.
In this work it is designed, with a Beyond–CMOS technology, a LiM target application, the *Hamming Distance* (HD) *counter*[1]: this unit is able to count, as analog value, the number of mismatches between external data and pre-stored informations inside a memory. In particular, the memory is based on a type of magnetoresistive technology, whose non-volatile storage element is called *Magnetic Tunnel Junction* (MTJ).
The work starts from a Verilog-A Spin Orbit Torque Spin Transfer Torque (SOT STT) MTJ model, found in literature[2] and employed as basic storage block inside a Cadence Virtuoso schematic. The peculiarity of this type of MTJ is the fact that it exploits the double contribution of SOT and STT currents to enhance its switching speed. Through a series of parametric analysis

[1] [1] Rahimi, et al., *A robust and energy-efficient classifier using brain-inspired hyperdimensional computing*, 2016.

[2] [2] http://www.spinlib.com/STT_SOT_MTJ.html

it is found the proper point of work (SOT and STT pulses duration and amplitude) that allows to minimize the writing delay. Then, it is implemented the basic MRAM-like cell, shown in Figure 1, (a).
Peripherals —such as the Sense Amplifier and the Read and Write Drivers— are designed for providing the correct read/write pulses to the schematic, following the chosen point of work. The input signal generation has been taken into account by Cadence SPECTRE *states*, that allow to save the simulation environment for each memory operation. Also a Python script has been written and tested for providing the proper pulses to the cell. The peripherals are then adapted to different sized cells arrays (8x8, 16x16, 32x32) and through Cadence Calculator are defined the function for computing the delays and power consumption (average power, peak power and energy) related to each memory operations. A Python script has been written for calculating the sum of all the power contributions, since the Cadence SPEC- TRE pwr function could not sense the consumption of the SOT STT Verilog-A blocks.
The evaluated performance of this first design are comparable, for what concerns the read/write speed, to a CMOS Static RAM; however, the power consumption associated to the SOT STT MTJ technology is much higher than the one related to CMOS technology (Figure 1, (c) (d)).
Then, the memory design is properly modified for testing the feasibility of different logic approaches: it is chosen the one most suited for the target implementation of a *HD Counter*. The HD has to be evaluated between an input word and pre-stored values inside each memory row: hence, each row results in a different HD depending on the "similarity degree" that the row has with respect to the input word. Two HD counting methods have been tested, both based on CAM architecture: the search operation, in fact, can be though as a bit-a-bit XNOR between the external data and the stored informations. The first method, inspired by a memristive approach found in literature[3], evaluates the HD as different discharging times on the match lines (Figure 1, (e)). The second one, instead, emulates the behaviour of an analog adder: each (properly modified) CAM cell (Figure 1 (b)) is able to generate or not a small amount of current depending on if an XOR = '1' (mismatch) or XOR = '0' (match) takes place. Then, all the currents of each row are summed, and the total V/I can be measured on a load at the end of the row. Such value is proportional to the number of mismatches associated to each row, i.e. to the HD (Figure 1, (f)).

[3] [3] Abbas Rahimi et al., *Exploring hyperdimensional associative memory*, 2017.

Both the counting methods are implemented through the design of SOT STT CAM cells, and then tested for increasing array sizes. As for the SOT STT MRAM, the extracted performance show good speed results and significant power consumption, coherently to the technology selection. It is demonstrated that the *analog adder* approach is faster and hardware saving with respect to the ML discharge one. Such design can be exploited, for example, in combination to minimum value search paradigms, for selecting the minimum HD (i.e. closest similarity) in HD Classifier algorithms, used for text language recognition. The work has been preceded by an extensive research on the Beyond–CMOS technologies working principles, their memory architectures and Logic–in–Memory approaches based on them.

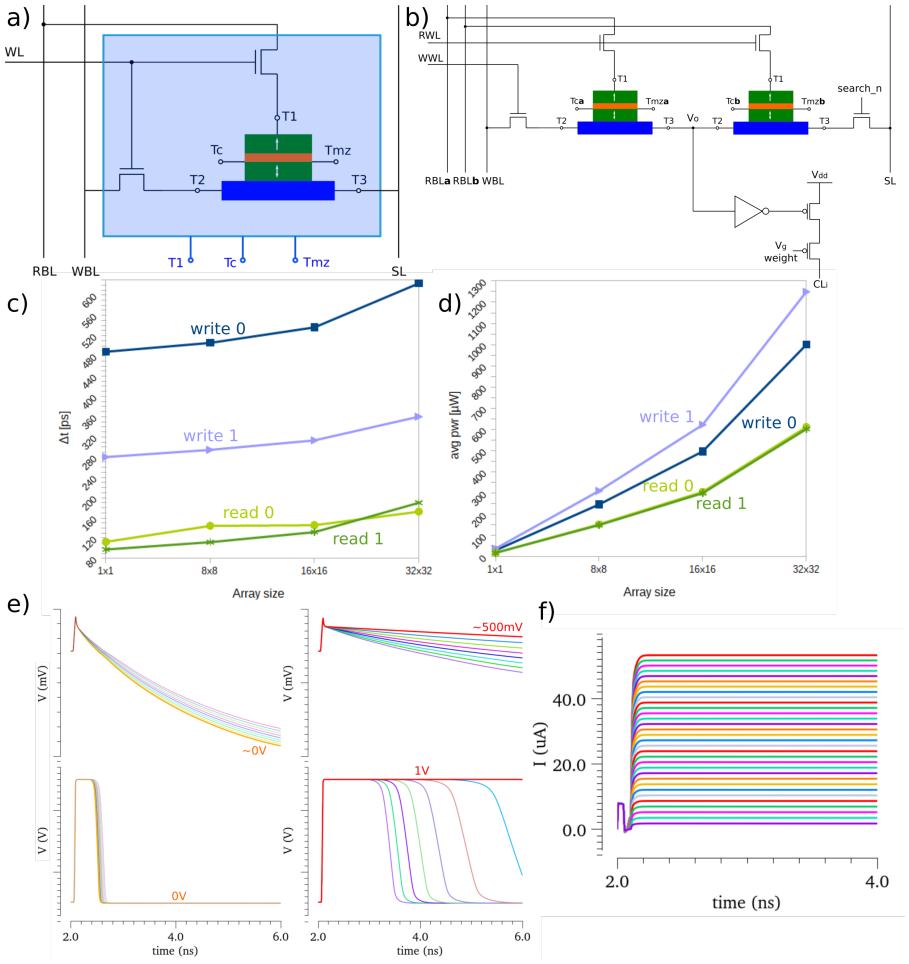


Figure 1. Designed SOT STT MRAM cell (a) and CAM-like cell (b), MRAM evaluated delays (c) and average power consumption (d), ML discharge curves (e) and *analog adder* LiM approach (f)

Contents

Part I

State of the art

Chapter 1

Emerging Memory Technologies: State of the art

1.1 Introduction

This chapter is dedicated to the investigation of the main emerging technologies for implementing memory devices. Such innovative techniques are necessary in order to overcome some of the possible physical limitations related to CMOS world.

1.1.1 Beyond CMOS

The most important Moore's law, that well predicted the doubling number of transistors per integrated circuit every 18 months, revealed its limits since 2000's. From these years on, in fact, microprocessors performance significantly bettered only due to new architectural structures. The main reason is that, in practice, CMOS scaling leads to an non-negligible leakage current increasing; it also implies reliability issues and heating problems.
So, especially in the last decade, researchers focused on innovative implementations of information storing mechanisms. This research field is known as *Beyond CMOS* and it is included in the so–called *more than Moore* trend. The aim is to find a technology that represents a valid alternative to CMOS–based memories, offering at the same time better performance.

1.1.2 Explored technologies

Non–volatile devices represent the most promising candidate as valid alternative to CMOS–based memories. They are able to provide higher reliability, together with reduced power consumption and new potential functionalities [1].

Figure 1.1 shows all the investigated technologies for Non–Volatile Memories (NVM) presented in this chapter. Obviously they are just some of the several solutions that are explored in recent years. For convenience, CMOS–based memories are treated only for performance comparison with respect to the new ones, as the attention would be otherwise deviated to another unwanted focus.
Non–volatile memories are also divided into baseline, prototypical and emerging according to their maturity level (updated in March 2020, see [2]).

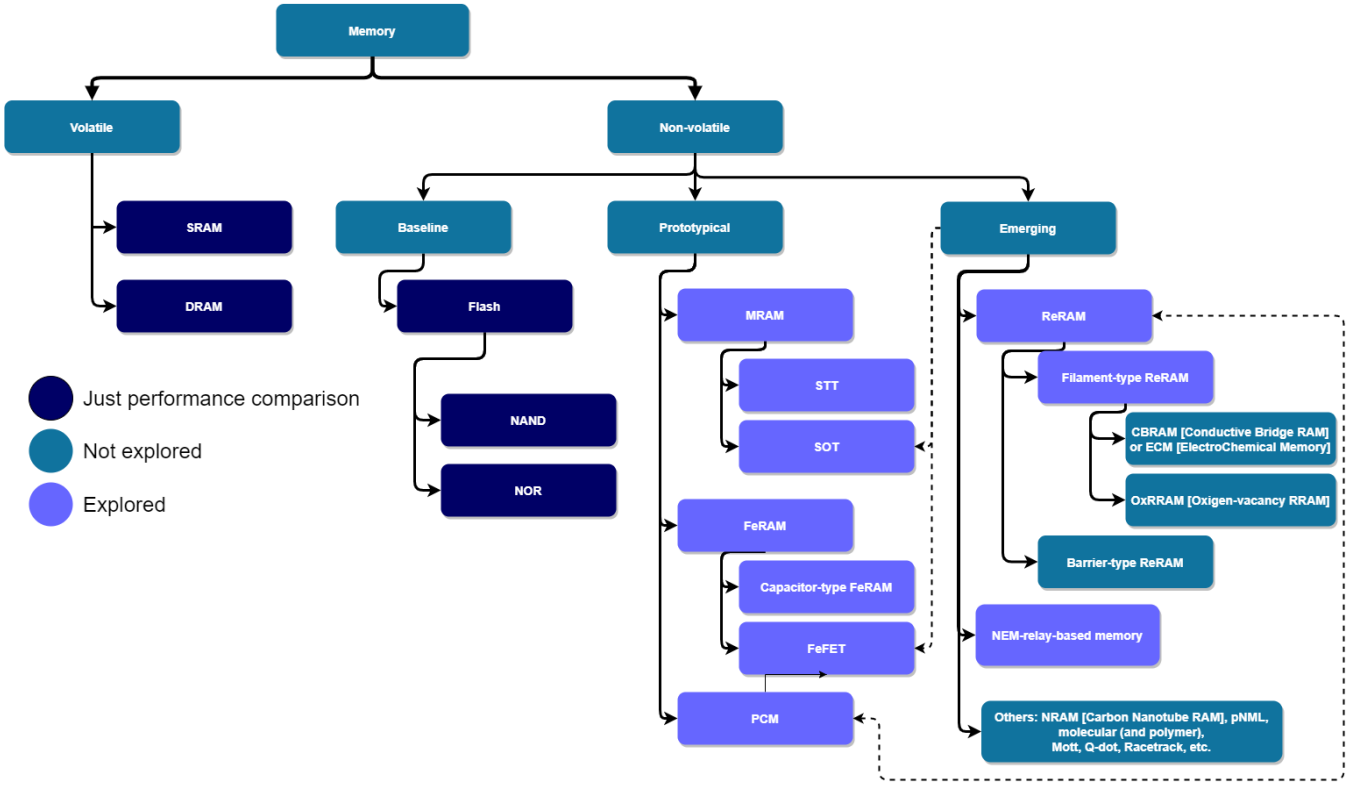


Figure 1.1. Explicative diagram of the analyzed emerging technologies. Color legend on the bottom right.

The sequence in which they will be presented in the next sections is the following: Megnetoresistive RAMs (MRAMs), divided into STT (Spin Transfer Torque) and SOT (Spin Orbit Torque); Resistive RAMs, of which only filament–type —the most common one— will be explored, while barrier–type (e.g. Schottky) will not be treated; Ferroelectric RAMs (FeRAMs), divided into capacitor–type and FET–type (FeFETs); Phase Change Memory (PCM, but also referenced as PCRAM or PRAM), whose working principle is similar to the resistive one: indeed, sometimes it is classified as a type of ReRAM (dashed line in figure 1.1).
NanoElectroMechanical (NEM) relays based memory will also be explored; however, since NEM relays are about 1000 times slower than solid state transistors, they will not be considered in this research for LiM applications.
Just a quick reference will be done for pNML memories, since currently they have not achieved a performance level that can keep up with the other previously mentioned memories yet.

1.2 Magnetoresistive Random Access Memory (MRAM)

1.2.1 Conventional MRAM

Magnetoresistive RAMs are non–volatile memories, whose state information is codified as a resistance value. The fundamental block is a Magnetic Tunnel Junction (MTJ): a stack of ferromagnetic–insulator–ferromagnetic layers constitutes its basic structure. One of the two ferromagnets has a free–to–change magnetization, while the other one is fixed (Figure 1.2, top). When the magnetizations of the two ferromagnets have the same direction (parallel magnetizations) the resistance of the MTJ is low and for convention this corresponds to a logic value “0”. A logic "1" is instead encoded when the two magnetizations are antiparallel (i.e. with opposite directions), matching to an high resistance state [3].
A significant parameter for MTJ is the Tunneling MagnetoResistance (TMR), defined in [3] as:

$$TMR = \frac{R_{AP} + R_P}{R_P}$$

and it quantifies how well the two logic states are distinguishable.

The conventional (or field–switched or only field) MRAM cell is depicted

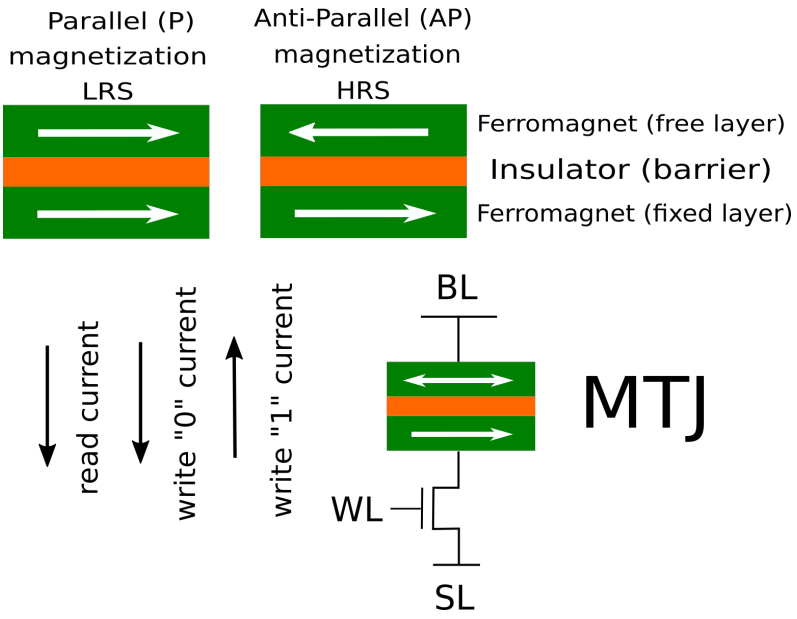


Figure 1.2. Up: MTJ stack for P and AP configurations, down: STT cell with currents directions.

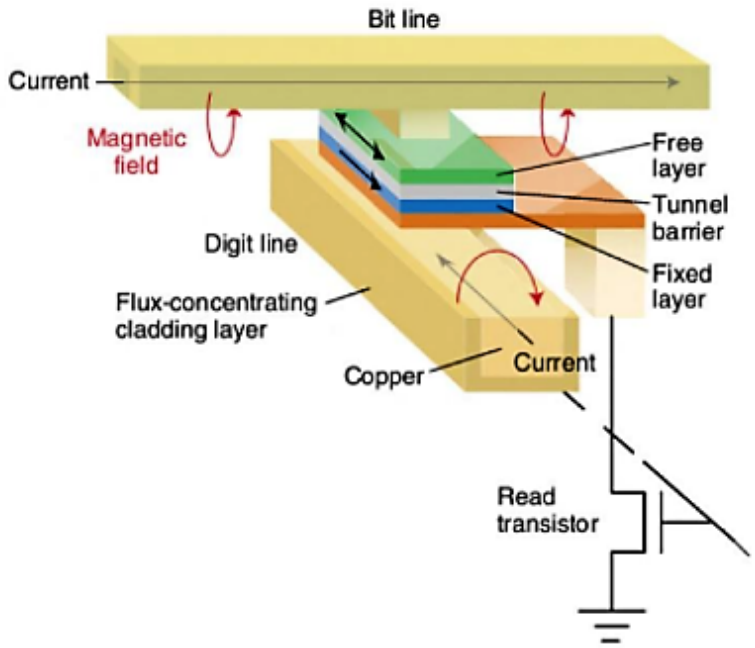


Figure 1.3. Conventional (or field–switched) MRAM cell [4].

in Figure 1.3: here it is well visible the MTJ in series with the access transistor for bit selection during read operation. The MTJ is situated between the bit line (BL) and the bipass line (orange in Figure 1.3). It is clear, just by observing Figure 1.3, that MRAM cell area is down–limited not by MTJ size but by the access transistor one.

Reading operation is performed as a simple measurement of the MTJ resistance value. In order to write information to the cell, instead, two methods are possible: the first one consists in applying an external field that forces a certain state in the cell; the second one is to provide a spin–polarized electric current pulse through the junction. This last one is at the base of the so–called Spin Transfer Torque (STT) memory cell presented in section 1.2.2, that also offers a more simplified and compact cell structure than conventional MRAM [3].

MRAMs can offer a large range of benefits: they can be very fast (reaching hundreds of ps), with an high data retention time (more than 10 years) and a very low power consumption (both static and dynamic). Their endurance is really high and comparable to CMOS–based SRAM and DRAM, being easily above 10^{15} cycles [5]. Nevertheless, conventional MRAMs present some weaknesses: their basic cell is big and structurally complex. This point represent the main reason for which in recent years research has been concentrated on STT–MRAMs rather than field[1] ones (see subsection 1.2.2).

1.2.2 Spin Transfer Torque MRAM improvements

As already anticipated in subsection 1.2.1, STT–MRAM cell has a simpler structure than conventional MRAM one (see figure 1.4). This implies a more compact structure and, hence, a lower fabrication cost. It also draws much less power, and integrability increases since the achievable density is greater. Since the write current scales down when reducing the cell dimensions, the scalability can be high. Furthermore, STT–MRAM intrinsic STT mechanism can provide a better write selectivity [7, 8].

Some reference values are reported in Table 1.1 [5]: it depicts all the performance parameters of conventional MRAM with respect to the STT–type. It

[1]Conventional MRAMs are also called *1st generation* MRAMs or *field–switched* MRAMs

can be noticed that cell size (expressed in feature size F), as already said, is reduced, leading to high density applicability. Moreover, the writing power consumption decreases and the endurance is greater.

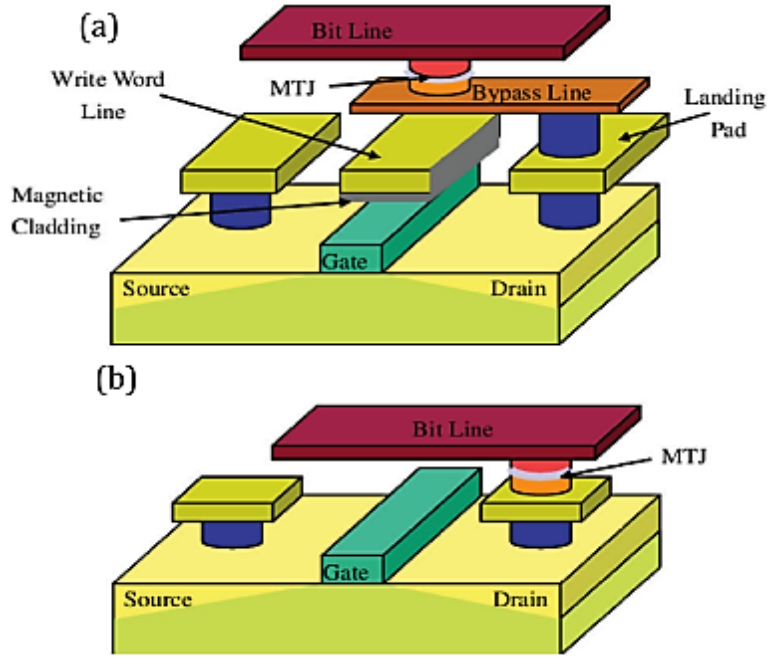


Figure 1.4. Conventional (a) versus Spin Transfer Torque (b) cell structure [6]

Features	**MRAM**	**STT–MRAM**
Cell size [F^2]	Large, ∼25	Small, ∼6÷20
Read time [ns]	3÷20	2÷20
Write time [ns]	3÷20	2÷20
Endurance	$> 10^{15}$	$> 10^{16}$
Write power	Mid to high	Low
Non volatility	yes	yes
Applications	Low density	High density

Table 1.1. Comparison between conventional and STT MRAM performance [5].

Another table is extracted from [9] (Table 1.2): this time, STT–MRAMs features are compared to the the well–known CMOS memories. It can be

Features	SRAM	DRAM	NAND Flash	NOR Flash	STT–MRAM
Cell size [F^2]	>100	6	<4 (3D)	10	6÷20
Cell element	6T	1T1C	1T	1T	1(2)T1R
Voltage [V]	<1	<1	<10	<10	<2
Read time	∼1 ns	∼10 ns	∼10 µs	∼50 µs	<10 ns
Write time	∼1 ns	∼10 ns	100 μs ÷1 ms	10 μs ÷1 ms	<5 ns
Write energy [J/bit]	∼fJ	∼10 fJ	∼10 fJ	100 pJ	∼0.1 pJ
Retention	N/A	∼64 ms	>10 y	>10 y	>10 y
Endurance	$> 10^{16}$	$> 10^{16}$	$> 10^4$	$> 10^5$	$> 10^{15}$
Non volatility	no	no	yes	yes	yes

Table 1.2. Comparison between CMOS memories and STT–MRAM performance [9].

noted that:

- STT–MRAM minimum cell size is comparable to DRAM and FLASH memories;
- reading and writing times are good, much lower than FLASH memories, although not at SRAM levels;
- endurace can be very high (can be greater than 10^{16}).

Some of the STT–MRAMs outlines, e.g. their near–zero leakage and great density, are making them an actual potential post–CMOS technology, mostly for on–chips memories applications. Anyhow, their energy efficiency has a limitation related to the STT switching energy necessary for writing and reading [10].

However, STT–MRAMs have reached an advanced maturity level with respect to other emerging Non–Volatile Memories (NVM), since they already started to be commercially produced; for instance, Everspin Technologies on December 23, 2019, announced its qualification to start producting STT–MRAM chips for its first costumer. It further called out that "since it started MRAM production, it shipped over 120 Toggle MRAM and STT–MRAM devices" [7].
Another example is the Numem, developer of high–performance STT–MRAMs, announcement regarding its selection for NASA AI project, done on 09 October 2020. Numem also stated that its memory allows for a 20÷50 times

lower standby power than SRAMs, and cell dimension can be made two–three times smaller [7].

1.2.3 STT–MRAM characteristics

Once explained the main benefits of this technology (section 1.2.2), it is convenient to present in details its working principle.

STT–MRAM cell is shown in Figure 1.5, together with its circuital representation.

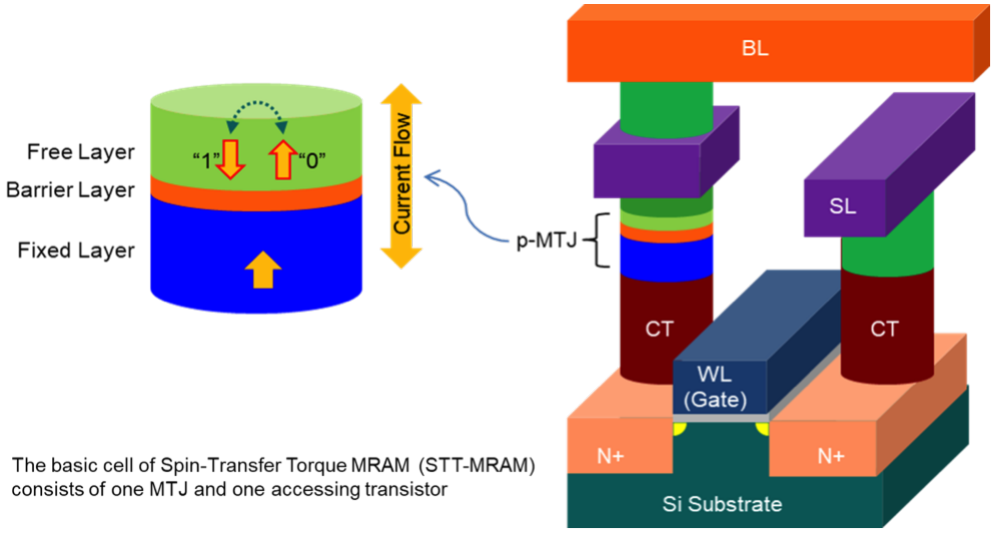


Figure 1.5. STT MRAM cell structure: MTJ element is highlighted [12].

The conventional logic state association to R_P, R_{AP} is the same done for field–switched MRAM. What differs between the two MRAM types is that writing operation for STT memory is based on current injection: the spin associated to electrons, that are flowing through the MTJ, exercise a torque force that is able to modify the angular momentum of the free layer. This change of magnetization implies a change of the cell logic state [11].
In details, write operations is performed with the injection of a current higher than the MTJ critical switching current I_C for a certain duration. Current direction (I_{WRITE0}, I_{WRITE1}) forces a certain logic state in the cell, and it depends on the polarity of the voltage between the BL and the Source Line (SL). The amount of current to inject is also influenced by the MTJ geometry

and barrier material, and by the duration of the writing pulse.
On the other hand, reading implies the WL activation (that makes the nMOS turn ON) and the application of a voltage V_{READ} between BL and SL (Figure 1.5). The resultant current is compared to a reference current by a sense amplifier, in order to detect the value contained in the cell [11].
In Figure 1.2 it is clarified the direction of the various currents involved in STT–MRAM operations.

Just few words about current research status —up to 2019— of STT–MRAMs are spent in this paragraph.
An ultra-high performance MTJ has been developed by researchers of National Taiwan University: it exploits a superlattice barrier layer and two half–metallic magnets instead of usual ferromagnets. This new type of structure constitutes the basic block of the so–called SS–MRAMs (included in STT–MRAM memory class) that is able to reach ultra–low power reads and writes, high and very high writing speed and endurance, respectively [7].
Another significant datum is that antiferromagnetic materials (opposed to the currently–used ferromagnetic ones) can be exploited to build STT–MRAMs, leading to potential high density devices, with fast writes and low currents. This last concept is nowadays investigated by researches from Northwestern University, Illinois [7].

STT–MRAMs, however, present some physical problems that limit device scaling and performance: one of these is the fact that the lower is the switching time (e.g. about nanoseconds), the higher will be the risk to damage the MTJ barrier with high writing current densities. This is due to the fact that switching speed is directly proportional to the switching current.
Moreover, since reading and writing follow the same path through the MTJ, with STT–MRAM cell scaling the write current decreases but not at the same rate of read one. So the difference between the two currents is reduced, and this makes the sensing not so reliable since write errors might occur [1].

These two issues can be mitigated in Spin Orbit Torque (SOT) MRAMs, explained in the next section (1.2.4).

1.2.4 Spin Orbit Torque MRAMs

Spin Orbit Torque (SOT) Magnetoresistive RAMs are introduced in this section starting from their structure (Figure 1.6, (b)). Differently form STT–MRAM cells (Figure 1.6, (a)), SOT–MRAM ones are based on an MTJ placed on a layer of heavy metal, resulting in a three–terminals device.

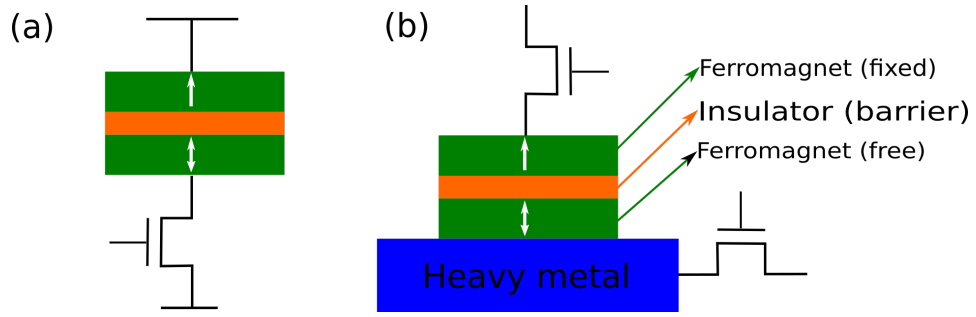


Figure 1.6. Spin Transfer Torque (a) versus Spin Orbit Torque (b) cell structure

The particularity of SOT cell is that its content can be switched by letting a current flow through the metal layer: thanks to the Spin Orbit Torque effect, in fact, the magnetization of the free layer can change leading to a logic status switch. This effect consist in a spin coupling induced by the spin associated to electrons in the current flow; such coupling forces a certain spin angular momentum direction in a particular region of space. In practice this is a generic term that includes various mechanisms derived from SOT interaction: if they rise up from the layer interfaces, they takes the name of *Rashba effect*; if they take place from the bulk, they are referred to as *Spin Hall Effect* (SHE). However, this classification is often just a convention since it is difficult to distinguish between the complex contributions from bulk and interfaces [1].

An additional key characteristic of SOT–MRAMs is the separation of reading and writing paths (Figure 1.7). In fact, while for STT–MRAMs reads and writes share the same path with currents perpendicularly injected through the MTJ, in SOT memory cells they have different routes: writing operation is performed through in–plane current injection, along the writing stripe — the heavy metal film— as in Figure 1.7, (a); reading, instead, is the same as STT–MRAMs one (Figure 1.7, (b)). This fundamental separation allows

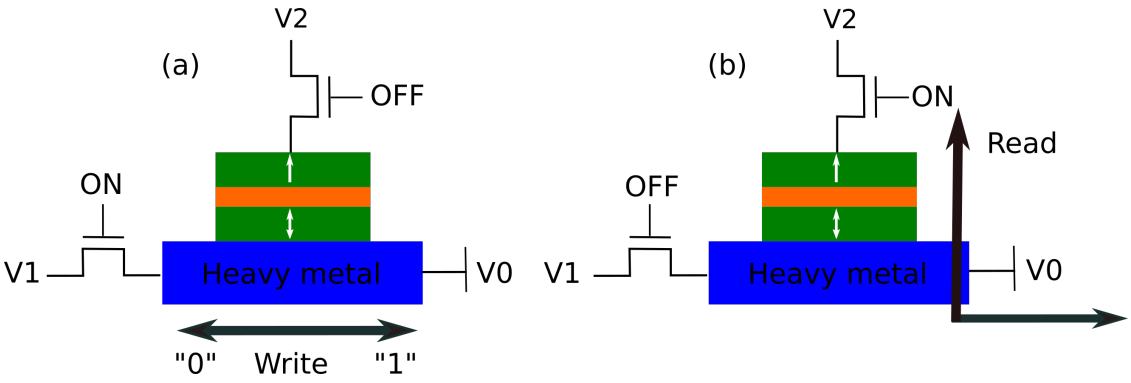


Figure 1.7. SOT MRAM writing (a) and reading (b) paths.

SOT–MRAMs to mitigate MTJ tunnel barrier damaging and to reduce problems related to read/write disturbs.

Features	**STT–MRAM**	**SOT–MRAM**
ON/OFF Ratio	1.5÷2	1.5÷2
Write voltage	<1.5 V	<1.5 V
Write time	<10 ns	<10 ns
Read time	<10 ns	<10 ns
Stand–by power	low	low
Write energy [J/bit]	∼100 fJ	<100 fJ
Drift	No	No
Integration density	High	High
Retention	Medium	Medium
Endurance	10^{15}	$> 10^{15}$

Table 1.3. Comparison between Spin Transfer Torque and Spin Orbit Torque MRAM performance [13].

The emerging consideration that has to be done by observing Table 1.3 is that endurance is surely greater than STT–MRAM one. This is due to the fact that writing current, that is larger than the reading one, in this case does not run across the magnetic tunneling junction, hence it does not damage the barrier and the endurance can be very high.
Furthermore, SOT–MRAM switching duration can be reduced more than the

STT–MRAM one, reaching also some hundreds of picoseconds (experimentally demonstrated, [1]). Also, power consumption decreases a little.
Nevertheless, the great constrain of SOT–MRAM is the large size of its cells: they can easily reach 160 F^2 [14], whereas STT cells are much smaller (6÷20 F^2, Table 1.1). Consequently, SOT–MRAM integration density is lower with respect to STT case.

In conclusion, SOT–MRAMs represent a valid and performing alternative to STT–MRAMs, since they can offer higher speed and endurance; however, they lead to a slightly deteriorated integration density.

Despite all the advantages that SOT–MRAM can provide, up to these days difficulties have been found regarding the development of a material capable of being an high electrical conductor with strong Spin Hall Effect. In 2018, researchers at Tokyo Institute of Technology announced that they built a topological insulator based on bismuth–antimony (BiSb), able of achieving both a huge Spin Hall effect and strong conductivity, hence a potential candidate for SOT–memories implementation [15].

1.3 Resistive Random Access Memory (RRAM or ReRAM)

1.3.1 ReRAM principles

Resistive RAMs are Non–Volatile Memories whose information storing element is a capacitor–like structure called *memristor*, portmanteau of memory and resistor (Figure 1.8). It consists in an insulating material (very thin, usually 2–to–hundreds of nanometers) sandwiched between two metal electrodes. The thickness and material of the insulating layer influences several features such as ON–OFF resistance ratio, memristor working mechanism, amount of voltage to use, etc.. By applying a proper voltage across the insulator its conductivity can be changed: it this way the information —coded as memristor resistance value— varies and a writing operation is performed.

It is considered to be in Initial High Resistance State (IHRS). In filament–based memristors, if it is applied a sufficiently high differential voltage to the electrodes, a soft breakdown into the insulator layer induces the creation of

Figure 1.8. Memristor C–like structure

a conductive path (filament) that electrically links the two metal layers (Figure 1.9). This process is known as *electroforming* (or just *forming*) and it occurs at the so–called forming voltage (V_F), which mainly depends on cell area and dielectric thickness [9].
Once the filament is created, the memristor is switched to a Low Resistance State (LRS), conventionally associated to a logic "1" (ON state). It is possible to switch the memristor to High Resistance State (HRS), through an appropriate reset voltage, breaking–up the conductive path and leading to a logic "0" (OFF). A set process, with its V_{SET} voltage pulse, changes the state back to a low resistance value re–forming the filament [13].
Hence, writing operation consists in changing the cell content through set and reset operations; even after the voltage has been removed, the memristor retains the information about its state (non–volatile device). Reading, instead, is performed by applying a voltage pulse smaller than the ones used for writing. In this way no write disturb occurs and the content of the cell can be determined evaluating the resulting current.

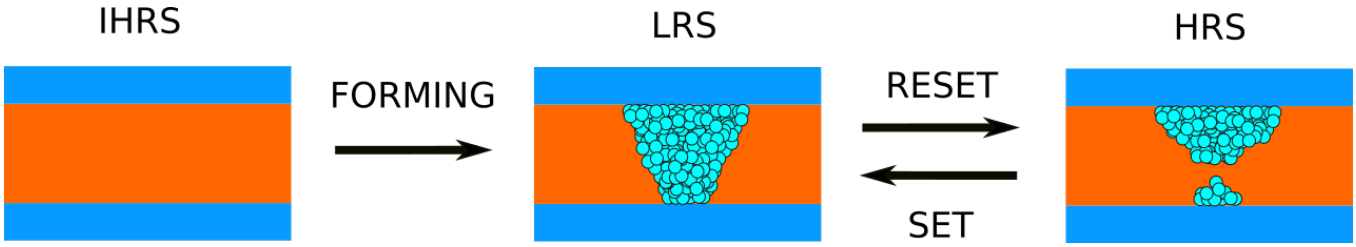


Figure 1.9. Memristor forming, set and reset processes.

1.3.2 ReRAM classification

The previous paragraph made reference to filament–type ReRAMs, based on creation and rupture of aforementioned conductive filaments to change the

resistive state of the memristor. Conduction path can take place thanks to various physical mechanisms, such as movement of vacancies or ions.
As depicted in Figure 1.1, there are two main sub–categories of Resistive RAMs: Oxide–RAM (OxRAM) and Conductive Bridge RAM (CBRAM), also known as Electrochemical Memories (ECRAM or ECM) [16].

Oxide–RAM memristor is a stack of bottom electrode–oxide–top electrode and oxygen vacancies in this oxide layer compose its conductive filament [17]. OxRAM can be further sub–divided into Valence Change ReRAM (VCRAM) and Thermochemical ReRAM (TCRAM): the first ones are more developed, but present stocastic parameters in forming process, require high current and their scalability is problematic. Their theoretical endurance is high (10^{12}) but experimentally are demonstrated some order of magnitude less. The second ones have a similar structure but exploit a current to break the filament through heat; however, they present a much lower endurance.

Conductive Bridge RAMs switching behaviour is instead based on movement of metal ions (usually Cu or Ag) into the solid–electrolyte. Their main advantage is that their ON/OFF resistance ratio can be pretty high ($10^3 \div 10^6$) in comparison to OxRAMs (limited to $10 \div 100$), at the cost of a reduced endurance ($< 10^4$ cycles with respect to OxRAM 10^{12}) [17]. In Table 1.4, updated in 2020, are reported additional performance parameters to compare these two technologies.

Features	**Metal–oxide ReRAM**	**Conductive Bridge ReRAM**
Speed [ns]	5	1
Operation voltage [V]	$\sim$3	$\sim$7
Operation current [µA]	5	10
Endurance [cycles]	10^{12}	10^6
ON/OFF ratio	10^7	10^7
Retention [s] @ 85°C	10^6	10^6
CMOS compatible	Yes	Yes
Fabrication	Easy	Easy
Scalability	Good	Good

Table 1.4. Comparison between Metal–oxide and Conductive Bridge ReRAM performance [9].

Just for completeness, it is noticeable that besides filament–type, memristors can be also barrier type: they present the same structure, but the working principle is based on the modulation of a physical barrier that leads to a resistance change of the memristor. For instance, Schottky barrier–type memristors can change their resistance value according to the potential distribution of their depletion layer [16].
However, this classification is just indicative to introduce memristors, since they can be based on many other physical mechanisms (atomristors, ferroelectric memristors, spintronic memristors, carbon nanotubes memristors are just some examples). Nevertheless, filament–based memristors were examinated more in details in this introdution just because they are the most studied and developed type of memristors up to recent years.

1.3.3 ReRAM advantages and constrains

In terms of performance, ReRAMs are the most promising candidate as CMOS–based memories replacement among all emerging memory technologies. This is due to their several benefits such as high speed (read and write times <10 ns according to Table 1.5, which is the completion of Table 1.2 from [9]) and improved density thanks to small cell area (<4 F^2, Table 1.5); the related costs and power consumption are low; moreover they present adaptability to various applications, also because they are CMOS compatible [9], and they have been theoretically proved to have good endurance (10^6 to 10^{12}, Table 1.5).

Features	SRAM	DRAM	NAND Flash	NOR Flash	ReRAM
Cell size [F^2]	>100	6	<4 (3D)	10	<4 (3D)
Cell element	6T	1T1C	1T	1T	1T(D)1R
Voltage [V]	<1	<1	<10	<10	<3
Read time	∼1 ns	∼10 ns	∼10 µs	∼50 µs	<10 ns
Write time	∼1 ns	∼10 ns	100 µs ÷1 ms	10 µs ÷1 ms	<10 ns
Write energy [J/bit]	∼fJ	∼10 fJ	∼10 fJ	100 pJ	∼0.1 pJ
Retention	N/A	∼64 ms	>10 y	>10 y	>10 y
Endurance	$> 10^{16}$	$> 10^{16}$	$> 10^4$	$> 10^5$	$\sim 10^6 \div 10^{12}$
Non volatility	no	no	yes	yes	yes

Table 1.5. Comparison between CMOS memories and ReRAM performance [9].

However, resistive RAMs still present some challenges that limit their efficiency. First, the variability of the physical parameters involved in switching process. Oxygen vacancies and metal ions (mentioned in subsection 1.3.2 as basic species of OxRAMs and CBRAMs respectively) follow a stochastic behaviour when they move to form the filament. In this way, the shape of the filament can change depending on the specific device and even on the specific cycle in which we are operating. Such dependence influences a lot the memristor resistance, so often write–verify techniques are necessary and the sensing circuit can become more complex to design. This can have heavy consequences on the latency, especially for Multi Level Cell operations. Problems can also arise when the thickness of the filament is too small (even if the writing current involved is usually low, downto $\sim$10 µA), worsening in this way data retention [17].
Furthermore, data in Table 1.5 for endurance and cell size may be in general too optimistic. It happens frequently that the endurance reported in scientific paper refers to the one of the single cell, and not to the one of the entire array (also orders of magnitude lower): the number of cycles for which the memory content is reliable is influenced by leakage, so when this parasitic contribution becomes significant (as when increasing the number of cells in the array) the endurance decreases. Commercial memristors present up to $10^6 \div 10^8$ cycles of endurance.Tantalum–Oxide based ReRAMs can reach high endurance but they have low data retention time [18].
Moreover, a cell size less than 4 F^2 (Table 1.5) probably refers to a 0T1R–array cell: different types of ReRAMs array will be presented in the next chapter, but it is useful to anticipate that this type of cell is smaller since it contains only one memristor without any access device. Unfortunately, 0T1R arrays are only theoretically developed, since no efficient implementation in practice is feasible yet. The main reason is to attribute to significative leakage problems that preclude their fabrication.

1.4 Ferroelectric Random Access Memory (FeRAM, F–RAM or FRAM)

1.4.1 FeRAM basics

In Ferroelectric Random Access Memories the information storing mechanism is based on the hysteretic polarization–to–field dependence, typical of ferroelectric materials.
Just as reminder, ferroelectricity is a property of certain media to intrinsically have a spontaneous electric polarization (formally, the dipole moment per unit volume [C/m^2]). When applying an external electric field, its direction can be reversed. In ferroelectrics the polarization is maintained even in the absence of an electric field, and from this it is derived the non–volatility of FeRAMs.

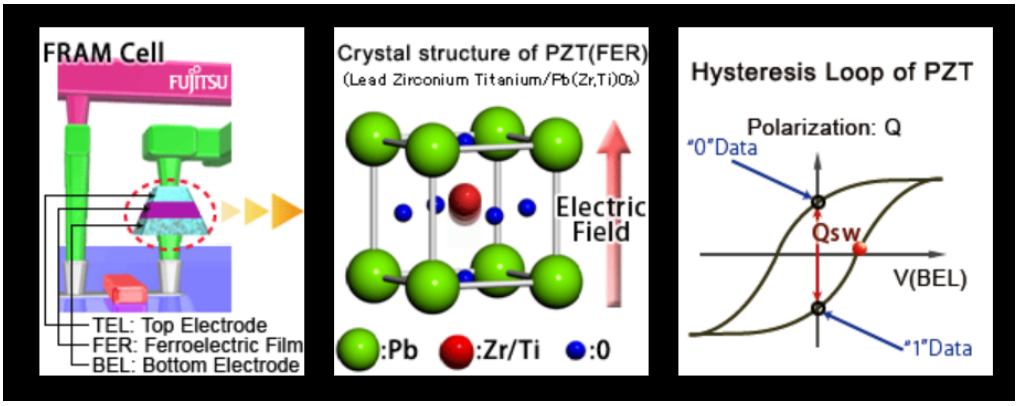


Figure 1.10. FeRAM cell structure (left), lattice element (middle), and P–to–V characteristic (right) [19].

In Figure 1.10 is depicted a Fujitsu simplified capacitor–like cell representation (left), together with the atomic view of the crystal unit cell of its PZT layer (center). The PZT (*lead zirconium titanate*) is a typical ferroelectric material employed in FeRAMs. Zr/Ti ions shift up and down in the unit cell according to the direction of the field applied between the top and bottom electrodes: they can reach two stabilization points on the hysteretic loop (right figure), corresponding to logic "0" and "1" stored in the memory cell. As aforementioned, data are preserved also when the voltage that gives rise to the electric field is no more applied [19].

FeRAMs can be classified in two main subcategories: Capacitor–type (1T1C cell) and FET–type (1T cell). The second one is also called called FeFET and it will be better described in subsection 1.4.2. The corresponding circuital scheme of their basic cells is reported in Figure 1.11. A marked similarity between Figure 1.11 (a) and a standard DRAM cell can be noted; the only differences are the presence of an additional Plate Line (PL) and the fact that the material inside the capacitor is ferroelectric instead of dielectric, making it a so–called ferroelectric capacitor. In Figure 1.11 (b), instead, ferroelectricity is detained by a thin layer of ferroelectric material plugged in the gate stack of a MOSFET, obtaining a ferroelectric FET. Both C–type FeRAM and FeFET physical structures are presented in Figure 1.12.

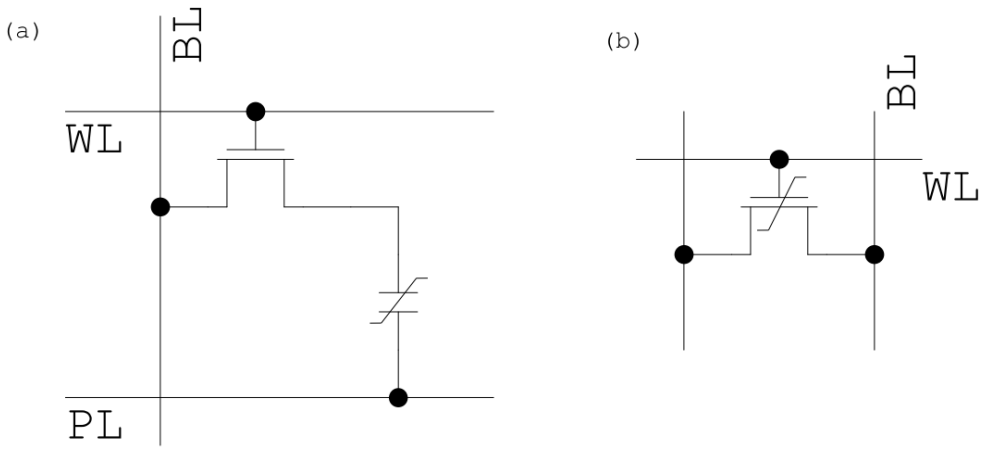


Figure 1.11. FeRAM classification: 1T1C (a) and 1T (b) cells.

Just for completeness, in the last years another type of FeRAM has been developed, the Ferroelectric Tunnel Junction (FTJ). This technology will not be presented here since it is too immature for our purposes, and some requirements that it needs, as a very low reading current, make it not commercially feasible (yet).

1.4.2 Capacitor–type FeRAM

In C–type FeRAMs, as anticipated by Figure 1.10 and related description, writing operation is accomplished by applying a proper voltage that stimulate a electric dipoles reorientation, changing the polarization of the material between the electrodes.

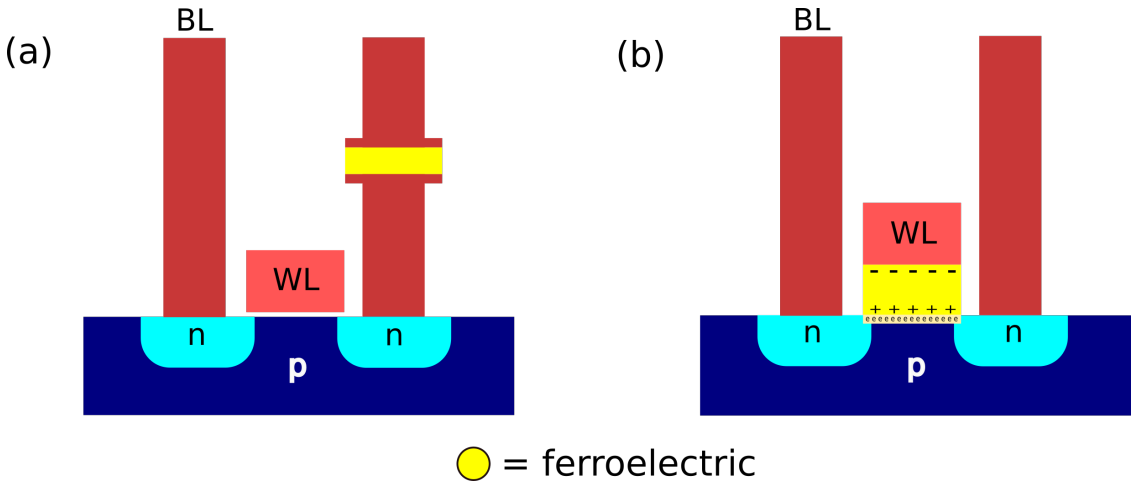


Figure 1.12. FeRAM (a) and FeFET (b) cell structures.

Readout is instead achieved by forcing through the transistor a certain logic state into the memory cell. If, for instance, a '0' is strained, the value '1' that is possibly contained in the cell before the forcing is sensed as a current pulse on the bit line. The problem is that such current is due to the charges that were stored by the ferroelectric capacitor, which is in this way emptied. So reading operation is destructive and the information must be rewritten to avoid data loss. On the contrary, in case of a stored "0" no variation on the bit line occurs, and this distinguishes the two (bitwise) reading results.
In subsection 1.4.4 it will be explained why for FeFETs data loss is not an issue, since reading is not destructive.

From Table 1.6 (completion of Table 1.3, [13]), it can be noticed that reading and writing times for FeRAM arrays are much lower than FLASH memories ones. Furthermore, they offer lower power consumption and a much greater read/write endurance (about 10^{10} for C–FeRAMs, some order of magnitude less for FeFETs).
However, both FeRAMs types present some issues related to scaling: ferroelectricity tends to disappear in too thin materials, and charge density may be too low to be detected. Obviously this depends on the choice of the ferroelectric material. In [20] it is stated that perovskite capacitor–type FeRAMs cannot be scaled under the 130 nm node. Exploiting PZT, scaling can be improved downto 70 nm, that is still not enough compared to CMOS technology. These limitations have prevented traditional C–FeRAM to be widely

industrialized, because they lead to lower storage densities and higher costs. A promising implementation is represented by Hafnium–dioxide (HfO_2) ferroelectric capacitors, since they are CMOS compatible and suited for Atomic Layer Deposition (necessary for reaching lower nanometric nodes). Again in [20] Stefan Müller, chief executive of Ferroelectric Memory Co., stated that HfO_2 ferroelectric films could be thinned downto 5 nm, reaching in this way the latest technology nodes.
Apart from this, research is moving towards 3D integration and stacking for scalability improvement. Furthermore, another interesting field of research is the one of FeFETs, a more recently–developed technology that aims to overcome C–FeRAM limitations, and that will be presented in the next sections (1.4.3, 1.4.4).

Features	**NOR Flash**	**NAND Flash**	**C–FeRAM**	**FeFET**
ON/OFF Ratio	10^4	10^4	$10^2 \div 10^3$	$5 \div 50$
Write voltage	<10 V	>10 V	<3 V	<5 V
Write time	$1 \div 10$ µs	$0.1 \div 1$ ms	∼30 ns	∼10 ns
Read time	∼50 ns	∼10 µs	<10 ns	∼10 ns
Stand–by power	low	low	low	low
Write energy [J/bit]	∼100 pJ	∼10 fJ	∼100 fJ	<1 fJ
Drift	No	No	No	No
Integration density	High	Very high	Low	High
Retention	Long	Long	Long	Long
Endurance	10^5	10^4	10^{10}	$> 10^5$

Table 1.6. Comparison between FeRAMs (C–type and FET–type) and CMOS Flash memories performance [13] (completion of Table 1.3).

1.4.3 FET–type FeRAM (FeFET): advantages and constraints

Ferroelectric Field Effect Transistor–based cell circuital scheme is reported in Figure 1.11 (b). They are theorized as a type of FET in which a ferroelectric film simply substitutes the oxide layer of a traditional MOSFET, as shown in Figure 1.12 and reported for convenience in Figure 1.13 (a). However, this structure leads to fabrication problems due to interdiffusion —into Silicon— of traditional ferroelectric materials [21]. So, in general, more complex structures are employed: an insulating layer between the substrate and

the ferroelectric material (Figure 1.13 (b)) or a ferroelectric–metal–insulator stack (Figure 1.13 (c)) can mitigate interface problems but also heavily limits retention time.

A structure like Figure 1.13 (c) is for example exploited in [22] for implementing a non–volatile and energy efficient Logic–in–Memory application: in fact, since this structure is simply modeled as a ferroelectric capacitance (the yellow layer between two metals) in series with a MOS capacitance (the stack below), the related $C_{FERROELECTRIC}/C_{MOS}$ ratio allows to suitably fit position and width of hysteresis loop for the aimed purposes.

The usage of Hafnium–dioxide (HfO_2) can provide better performance, like for C-FeRAMs, but the consequent retention time is still too limited. Some improvements have been proposed: for instance, in Figure 1.13 (d) is shown a standard Metal–Ferroelectric–Insulator–Semiconductor (MFIS) FeFET, with silicon–doped hafnium dioxide as ferroelectric layer. In the related article [23] the thin dielectric layer of SiO_2 is annealed to obtain SiON, which provides higher permittivity: this allows for lower FeFET switching voltage, mitigated tunneling effect which leads to higher endurance (due to reduced charges trapped in ferroelectric layer), and higher retention time.
Another solution to interface issues consists in exploiting ferroelectric polymers: since they can be deposited at room temperatures, interdiffusion may be reduced. They represent a flexible and cheap alternative to standard FeFETs, but the switching time is much lower [21]. Relatively to this approach, in [24] a hybrid SnO–polymer FeFET reached 5000 s retention time.

Referring to Table 1.6, it is evident how FeFETs can provide —with respect to C–FeRAMs— higher write speed, lower write energy (in the order of femtoJoule) and greater integration density, also due to the reduced cell dimensions. As stated in [25], the reduced writing latency and energy can be attributed to the fact that FeFET switching is field driven (differently from current–driven MRAM, PCM and ReRAM) at the gate, where leakages are smaller. Hence FeFETs can be employed for high–density, fast storage and minimal leakage applications. Moreover, especially HfO_2–based FeFETs yield good CMOS compatibility and scalability, and so thay have a true potential for in–memory computing. GlobalFoundries has already verified 28 nm HfO_2 FeFETs advantages and feasibility [26].

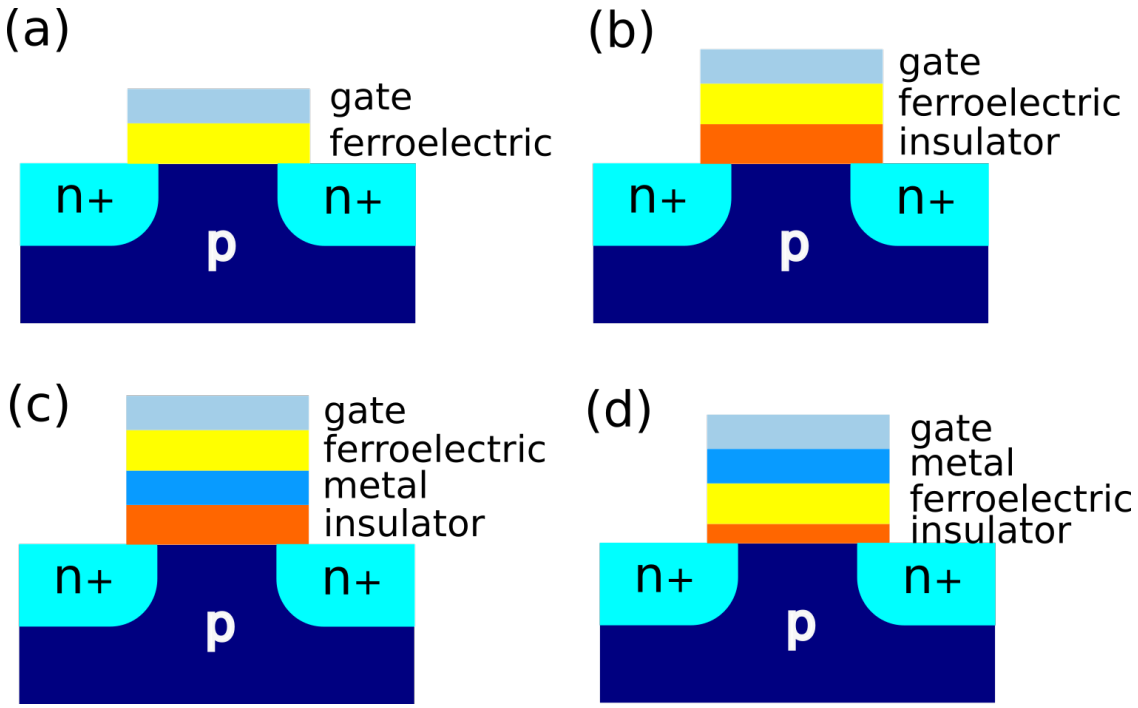


Figure 1.13. Some of the possible FeFET stacks

Nevertheless, a note must be done regarding FeFETs maturity: differently from capacitor–based FeRAMs, FET–types are a relatively "new" technology, since almost all implementations are still in research phase. Anyway, few commercial products are attempted to be created: in 2017 Ferroelelectric Memory Company stated that hafnium dioxide FeFETs, scalable to the last technologic node, were in development [20].

Furthermore, even counting all the benefits that this tecnology can provide, a huge limitation is related to their endurance.

1.4.4 FET–type FeRAM (FeFET): functioning

In FeFETs, the ferroelectric layer in the gate stack is responsible for retaining the transistor state (ON/OFF) even when the external voltage is no more applied. Exactly like in C–FeRAMs, in fact, the ferroelectric hysteretic polarization–to–electric field dependence presents two stable points for logic information.

In Figure 1.14 is displayed a simple n–FeFET schematic of polarization states: in low V_t state (left figure), the polarization of the ferroelectric layer is down–directed and the electrons invert the channel region, making the FeFET turning on. This case corresponds to the left branch of the hysteretic loop. On the other hand, when polarization instead is up–directed (center figure), it is formed a permanent accumulation, turning off the ferroelectric transistor. In the hysteresis loop this corresponds to the right (high–V_t) branch [20].
These two polarization states represent the two possible logic states stored in the cell. So, writing operation simply consist in the reversal of the ferroelectric polarization through an appropriate electrical field, applied by a proper voltage V_g to the gate stack. In such sense, it is very similar to the approach used for C–FeRAMs switching.
What instead differenciate FET and C–FeRAM types is the nature of reading mechanism: this time, logic data can be non–destructively read simply sensing if a drain current is present or not (so if the FeFET is ON or OFF, Figure 1.14).

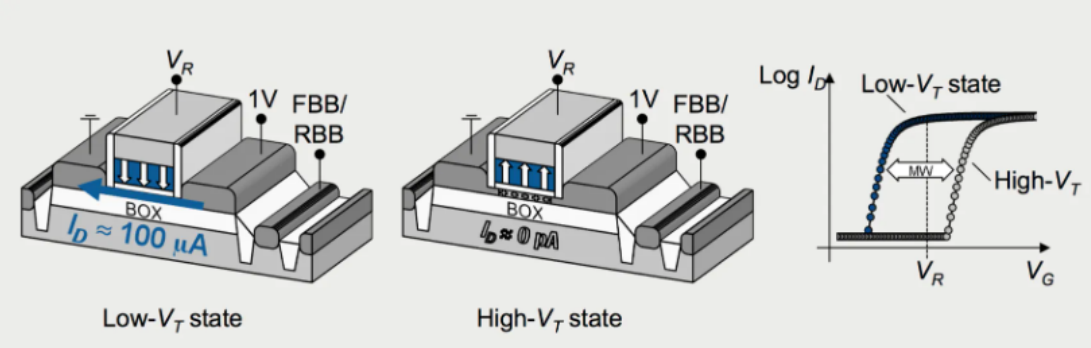


Figure 1.14. FeFET polarization states and hysteresis loop [20].

1.5 Phase Change Random Access Memory

1.5.1 PCM: basics

The Phase Change Random Access Memory (PCRAM or PRAM), also called CRAM (chalcogenide RAM) or simply Phase Change Memory (PCM), is a non–volatile memory whose working principle is based on the particular properties of chalcogenide glasses. Such materials are capable of changing their physical state, from amorphous to crystalline and vice versa, when they are heated. In memory applications, heat is applied through electrical pulses. The most known phase change material for memories is germanium–antimony–tellurium (GeSbTe or GST). The large range of resistance values associated to the switching physical state is at the basis of data storage [27].

Conventionally, the high resistance state (HRS) is associated to a logic "0", and this corresponds to the case in which the chalcogenide material is in amorphous state (high electrical resistivity). Vice versa, when it switches to a low–resistivity crystalline state, a logic "1" is stored into the cell (see Figure 1.15).

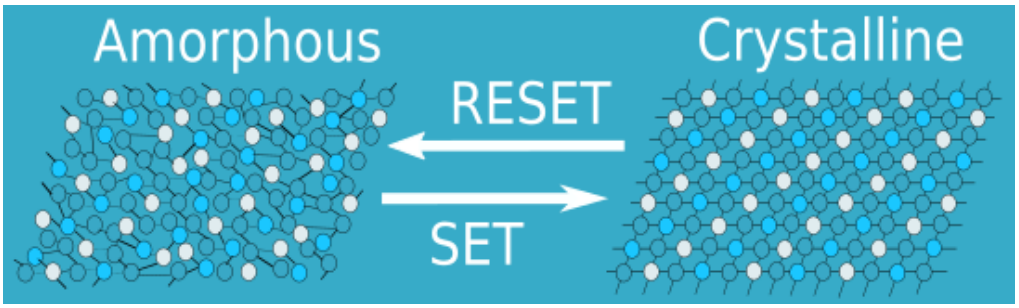


Figure 1.15. Amorphous (HR) and crystalline (LR) states for PC materials.

A standard PCM cell is the so–called 1T1R cell: the transistor "T" is a n–type access MOS while a variable resistance ("R") is used to represent the phase–change chalcogenide glass. A basic circuital scheme is reported in Figure 1.16.

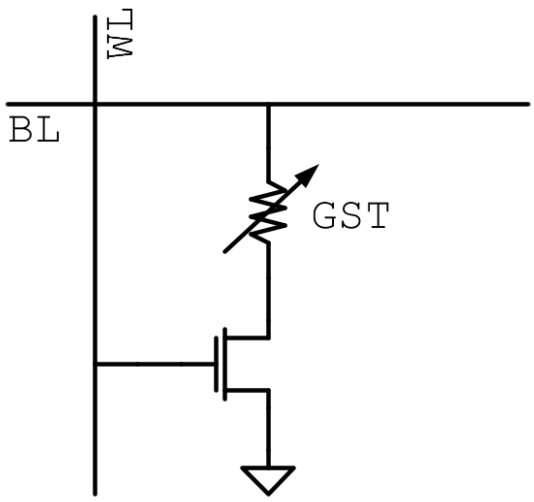


Figure 1.16. PCM standard cell.

1.5.2 Phase Change RAM operations

Writing PCM cells needs different heating/cooling steps depending on the process, SET or RESET, that is performed.
Taking as example the GST glass in an initial crystalline state, it can be heated above its melting point —usually over 600°C— in order to partially break the lattice strong atomic bonds. In this way the material stops being crystalline and, once cooled down, its atomic disorder is frozen into the so–called amorphous state (HRS, "0"). This procedure is known as RESET, and the related current pulse (RESET pulse) is relatively short in time and with a quite high amplitude (minima: $t_{RESET,MIN}$, $I_{RESET,MIN}$).
In the SET process, on the other hand, the crystalline state (LRS, "1"), is obtained by applying a current pulse wider than the previous one
($t > t_{SET,MIN} > t_{RESET,MIN}$) but with lower amplitude
($I_{SET,MIN} < I < I_{RES,MIN}$). This is motivated by the fact that the GST material requires a proper time interval to accumulate enough energy for organizing atoms in an ordered structure, and an amplitude such that the corresponding temperature will be above the crystallization point but below the melting temperature.

In order to read out the information stored by PCRAM cells, instead, it is

simply sensed the current after having applied a small voltage (V_{READ}) between the electrodes. The voltage pulse is high enough to have a sensible current but not too much to prevent writing errors ($I_{READ}<I_{SET,MIN}<I_{RES,MIN}$), and it is usually shorter than the SET pulse.
Set, reset and read operation are well described in [27, 28, 29]. The associated current pulses, as results of Joule heating, correspond to temperature pulses as depicted in Figure 1.17.
In Figure 1.18, instead, is shown a couple of Phase Change RAM cells, respectively in crystalline LRS and amorphous HRS.

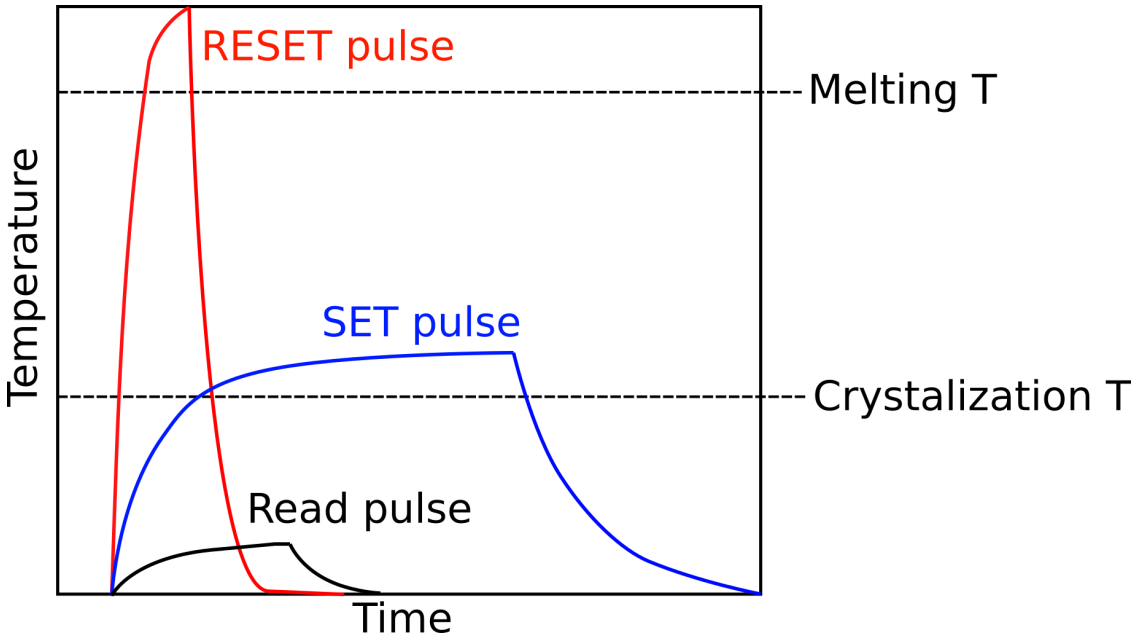


Figure 1.17. Temperature pulses corresponding to reset, set and read processes.

1.5.3 PCM performance

From Table 1.7 it is possible to have an estimation about PCM performance. Reading and writing times are comparable to the other explored memory technologies, so much lower with respect to FLASH memories. Besides, endurance is rather good ($10^6 - 10^9$) and the small PCM cell size ($4 \div 20$ F^2, [9]) allows for high integration density.

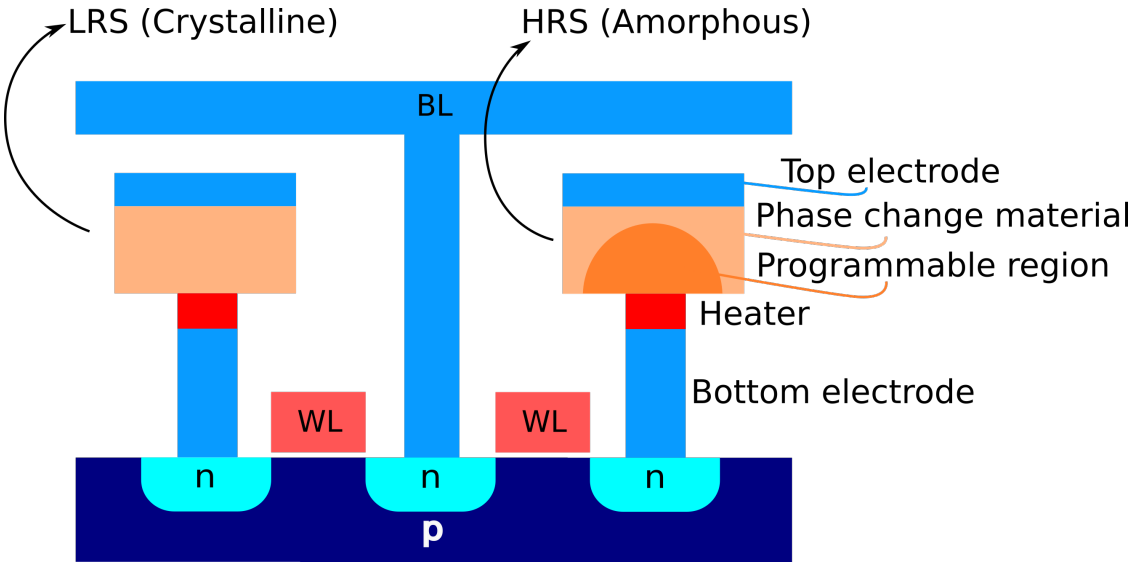


Figure 1.18. Section view of a couple of PCRAM cells.

Features	PCM
ON/OFF Ratio	$10^2 \div 10^4$
Write voltage	<3 V
Write time	$\sim$ 50 ns
Read time	<10 ns
Stand–by power	Low
Write energy [J/bit]	10 pJ
Drift	Yes
Integration density	High
Retention	Long
Endurance	$10^6 \div 10^9$

Table 1.7. Phase Change Memory performance [13] (completion of table 1.3 and 1.6).

However, PCRAMs present degradation with usage, as for FLASH memories (although for different causes), but they their degradation rate is much slower. PCM lifetime is reduced by peculiar physical mechanisms, as the fact that phase–change glasses, while thermally expanding in programming, degrade the memory; moreover, they could lose adhesion with the adjacent

dielectric due to different expansion rates. At high temperatures, the dielectric may present leakage issues; also, metal migration is a problem.

Furthermore, high current required by programming (especially for reset operation) may lead to high energy consumption.

Nevertheless, the main issue in PCRAMs is maybe their long–term threshold voltage and resistance drift, as stated in [30] and mentioned in table 1.7: with time, the resistance associated to the phase–change material in amorphous condition tends to increase. This is a limiting factor for multilevel operations, due to the risk of mixing up different resistance intermediate states. Also the threshold voltage presents a similar behaviour, bringing possible failures even in standard operations.

1.6 NanoElectroMechanical Systems (NEMS) based memory

1.6.1 NEM relays: basics

In this section NanoElectroMechanical (NEM) relays (or NanoElectroMechanical Systems, NEMS) for memory implementations are investigated, although, as anticipated in the introduction (subsection 1.1.2), their switching speed is not even comparable with the technologies already presented, and so they will be not considered for an application for in–memory computing in this case of study.

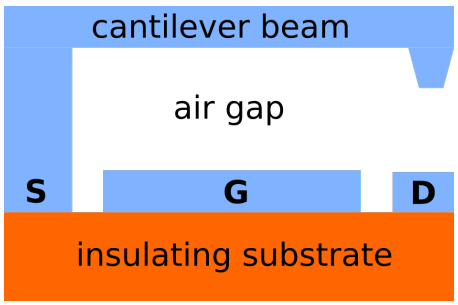


Figure 1.19. Three–terminals NEM relay structure.

The conventional three–terminals structure of a NEM relay is shown in Figure 1.19: source (S) and drain (D) are the so–called input and output terminals, while the gate (G) is known as the actuation terminal [31]. Also two and four terminals structures are possible, however here the focus is centered on three terminals configuration since it is simpler than the structure with four terminals and allows for a better control than two terminals devices.

A three–terminals (plus bulk one) NEMS can circuitally mimic a MOSFET, but obviously with different performance and physical characteristics [32, 33]. Since it is in principle both a switch and a memory element at the same time, it can be treated more or less in a similar way as FeFETs, at least for what concerns the basic 1T cell circuit (Figure 1.11, (b)), and, in theory, the non–destructively read–out method (just by sensing drain–source current).

In literature there are a lot of other different solutions for NEMS–based cells; however, from now on, it is taken in consideration a "1T" cell (where "T" this time stands for NEM relay and not transistor) as proposed for example in [34], for having a clearer and simpler view of the technology.
Just for curiosity, a CMOS circuit implemented with only NEMS is called CNEM [33],

Referring to Figure 1.19, a countilever beam, attached only by one side to the source region (S) of the substrate, can be flexed under the application of a proper electrostatic or magnetostatic force. Through such deformation an electrical connection can be formed between the beam itself and the drain (D). Hence this device simply consists in an electrically actuated switch, which can be employed in memory cells as data storing element [31].

A writing operation in memory cells based on NEM relays simply consist in applying the proper voltage between the beam and the gate region (V_{GB}). The cantilever rest (or OFF) position occurs when an air gap separates the beam and the gate; if it is applied a V_{GB} greater than the so called Pull–In voltage (V_{PI}), the corresponding electrostatic force overcomes the elastic one associated to the beam. Hence the beam itself is bent and put in contact with the drain region (ON position). Such operation is conventionally called Pull–In, and makes a "1T" cell store a logic "1" (NEM relay ON).
On the contrary, a Pull–Out process takes place when a voltage V_{GB} lower than V_{PO} (Pull–Out voltage) is not sufficient to maintain the contact between the beam and the drain. From a physical point of view, this is due to

the fact that the adhesion force alone —since this time the electrostatic one is weak— between the beam and the drain is not capable of facing the elastic force, which tends to make the beam returning in rest position. In this way the connection is lost and a logic "0" is stored in a "1T" cell.
V_{PI} and V_{PO} points are highlighted in the hysteretic dependence of the drain current from the applied voltage in Figure 1.20. The beam–drain voltage V_{GB} can be also called V_{GS} since the source is electrically and physically connected to the beam. The hysteresis loop is at the basis of the non–volatile information storing mechanism, since each voltage value in the range between V_{PI} and V_{PO} is not capable of making the NEM relay switch its logic content. This characteristic is particularly relevant for Random Access Memories applications [31].

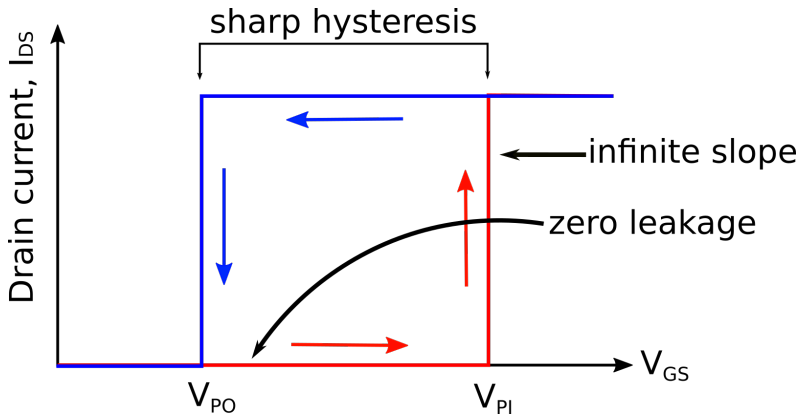


Figure 1.20. NanoElectroMechanical relays sharp hysteresis loop.

Reading operation in "1T" cell instead is simply performed by sensing if the drain current (so the bit line current) of the NEM relay is present or not (NEM relay ON or OFF), exactly as said for FeFETs in subsection 1.4.4.

1.6.2 NEMS–based memories: benefits and drawbacks

NEMS–based memories can benefit of some unique properties related to the mechanical nature of such devices. In particular, NEM relays are able to overcome one of the main bottlenecks of CMOS technology: leakage issues. In fact, when NEM relays are in OFF position, an air or vacuum gap acts as physical separation between the flexible beam and the drain region. Such opening nullifies leakage in the device (Figure 1.20). This leads to high I_{ON}/I_{OFF} ratio, zero OFF–state energy dissipation, and higher retention time: all advantages that are much aspired for next generation technologies.
Referring again to Figure 1.20, it can be noticed that another peculiar characteristic in NEMS is their "sharp" hysteresis loop: the transition between OFF and ON states (and vice versa) is abrupt, since no intermediate states are possible —the contact occurs or not— given the mechanical essence of NEMS [31].

All the advantages notwithstanding, NEMS intrinsic nature can be a double–edged sword: mechanical delays are much greater than electric ones, such that NEMS switching speed can be orders of magnitude lower than solid state relays one [32]. This is the main reason for having neglected this technology as valid candidate for this work: with NEMS fast computation in memory is almost impossible. However, their relatively low resistance may allow for chains of multiple NEM relays performing, with simultaneous switches, a single but complex computation: so a computing system is feasible for applications at low clock frequency that aims to carry out large operations.

Furthermore, NEM relays have to deal with nanoscale effects, such as surface adhesion forces (with special reference to Van Der Waals force), that become problematic when scaling the devices. This last side effect must be taken into account by optimizing growth conditions and geometry of the mechanical switch, as suggested in [35]: in this article, it is presented a singular memory implementation based on NEM switched capacitor structure built with vertical multiwalled carbon nanotubes (CNT), that is able to reach low power consumption and reduced cell size (due to vertical structure development), but with predicted switching speed limited by the natural oscillating frequency of the CNTs.

All the difficulties notwithstanding, some studies still focus on this technology trying to find some speed improvements for memory applications.

In [36] it is claimed to have designed a $4 \div 6$ F^2 NanoElectroMechanical diode device for RAM applications capable of performing writing in 0.27 ns but with limited endurance ($>10^4$).
In [37] instead it is proposed a NV–NEMory (Non–Volatile NEM Memory) for energy–efficient data searching with a further access transistor, that with its 8 F^2 size can reduce programming and reading times respectively under 10 ns and 0.1 ns. However, these predictions may be a little too optimistic since —as aforementioned— electro–mechanical switches are far slower than solid state ones, and usually speed–improved NEM applications sacrifice some other performance aspect to become faster.

1.7 About Nano Magnetic Logic (NML)

In this section it is done just a very quick review on the current state of Nano Magnetic Logic technology.
In few words, rectangular nano–magnets are used as bitwise informations storing elements, since their magnetization assumes only two possible directions; perpendicular (pNML) and in–plane (iNML) configurations are possible depending on the orientation of the magnetization (Figure 1.21). Information propagation is basically achieved by exploiting an external magnetic field together with a clock mechanism, and the direction of data flow is set by the nucleation center: this allows in practice to build all possible types of logic.

A standard pNML RAM cell is designed in [38]: however, due to the required high level of interconnections for row/word selection, area and delays could increase too much. In the same article is proposed a distributed version of such cell, but its performance are not able to keep up with other EMTs: cell size is far greater (about 1280 μm^2) than average EMT cells, while reading and writing times are not so good, respectively about 35.2 µs and 24 µs. However, this can be seen as a consequence of the fact that NML technology is still at a lower degree of maturity and few articles in literature are focusing on NML memory applications. For such reasons, it will be not further presented for Logic in Memory implementations in this work.

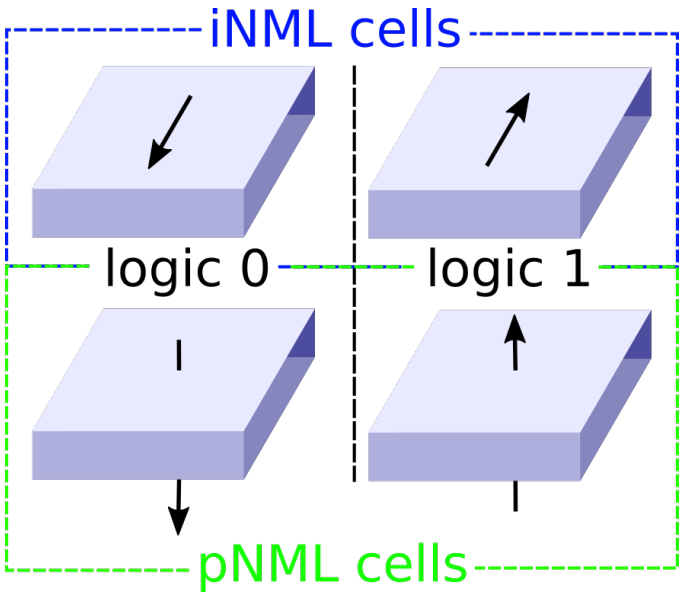


Figure 1.21. Magnetic binary representation for iNML (top) and pNML (bottom).

1.8 Brief on EMT performance comparison

In order to conclude this first introductive chapter, in the following pages are reported four graphs that summarize the presented technologies (excluding NEM relays–based and NML ones) following four different parameters: cell dimensions, reading and writing times, and endurance. Almost all data are taken from the tables reported in the previous sections (Table 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7), that, as can be seen from the references, are taken from articles published in 2020. The only values not updated up to this year are specified in the next paragraphs.

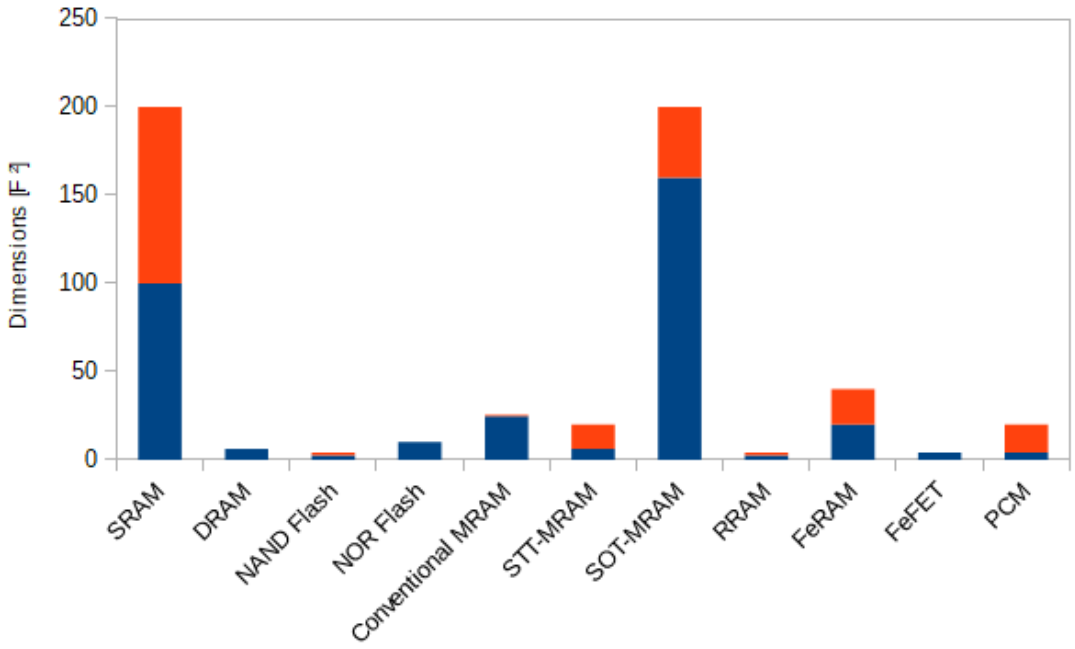


Figure 1.22. Graph comparing EMT memories dimensions.

In graph 1.22 are reported cell size values in feature size units [F^2]. What is immediately evident is that —for dimensions— Spin Orbit Torque MRAM is comparable to CMOS SRAM ([14], 2018), having a cell size greater than the other presented memory technologies, which are instead comparable to CMOS DRAMs. In particular, standard ReRAM offers the smallest cell (<4 F^2, Table 1.5), close only to FeFET size (4 F^2, according to [39], 2010).

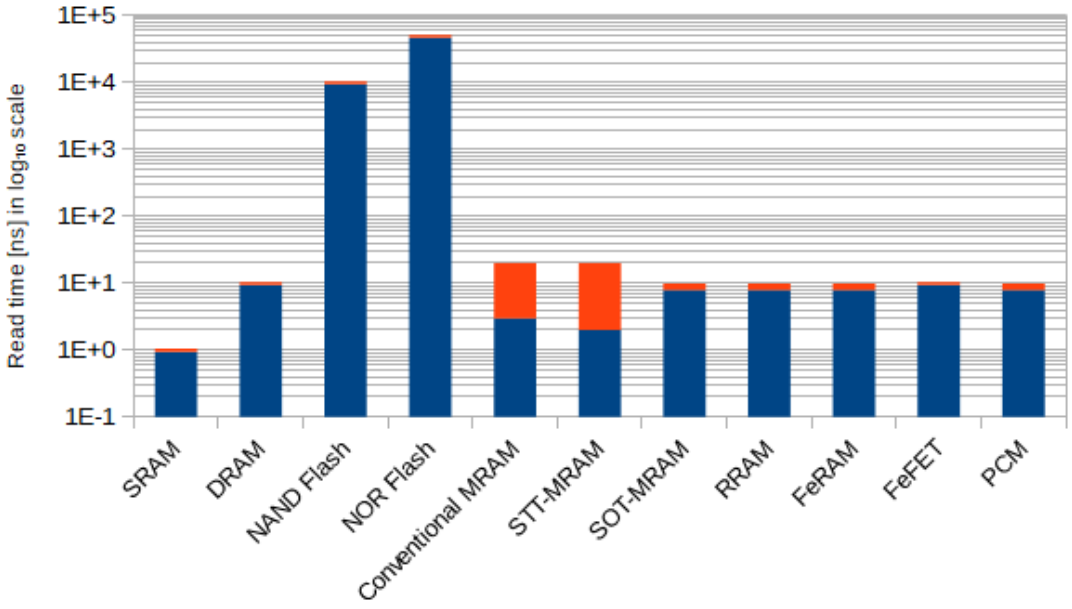


Figure 1.23. Graph comparing EMT memories read times.

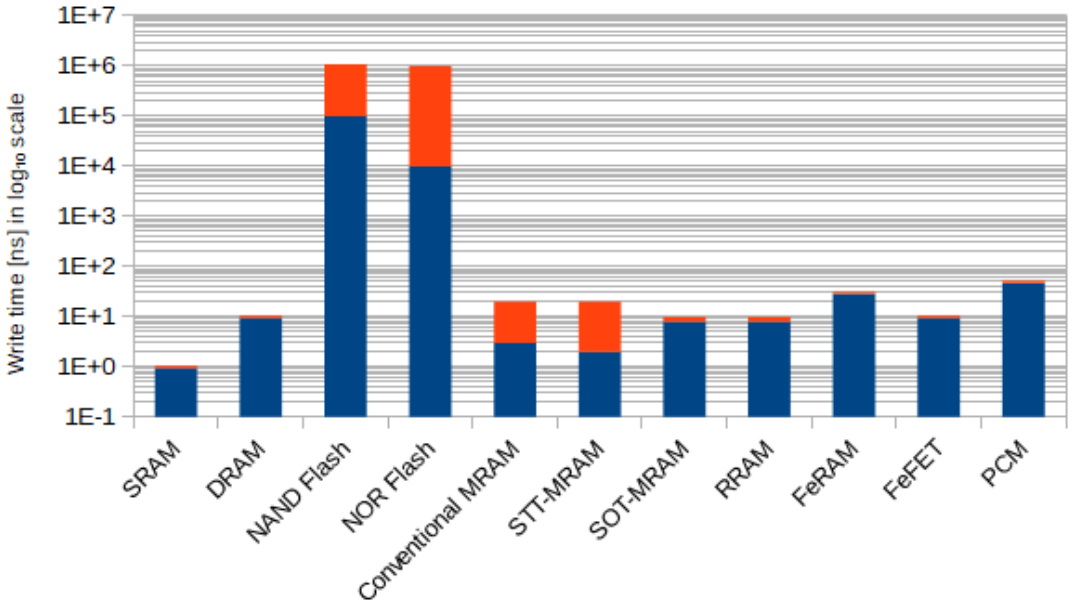


Figure 1.24. Graph comparing EMT memories write times.

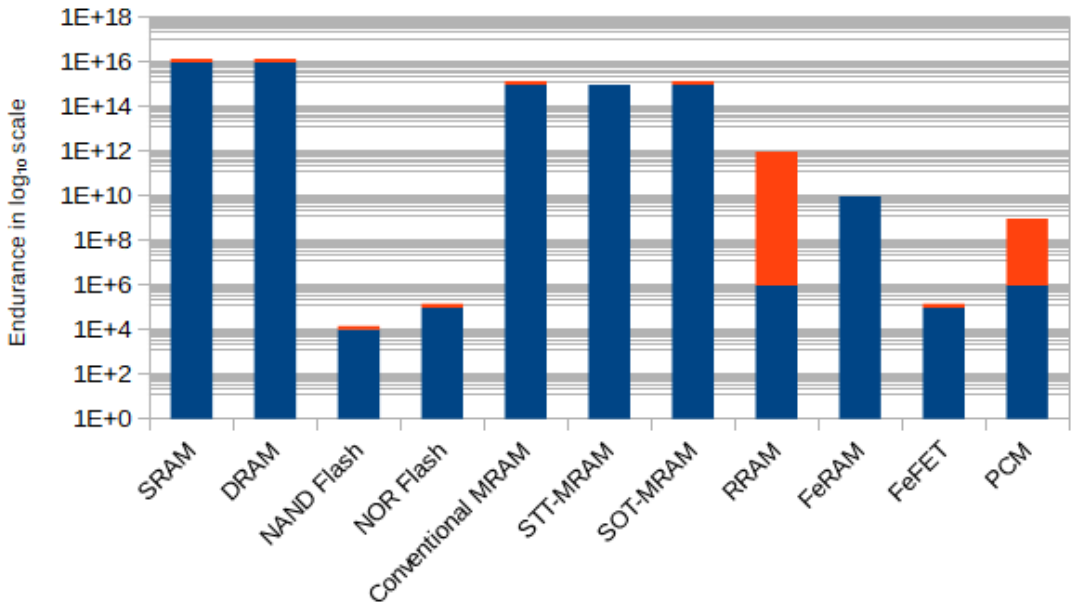


Figure 1.25. Graph comparing EMT memories endurance.

Also STT MRAM and PCM are characterized by rather small cell dimensions (respectively Table 1.2 and article [9], 2020).

Graphs in Figure 1.23 and Figure 1.24, instead, show the average values of reading and writing times, respectively. It can be noticed that the read times are similar for all the analyzed memory technologies, and comparable to DRAM ones (units of nanoseconds, so much lower than FLASH memories ones). NAND FLASH read time is in the order of microseconds ($\sim 10\,\mu s$, Table 1.2) so its value is saturated in order to make the graph more readable. The same is done also for NAND and NOR FLASH memories write times (respectively $\sim 0.1 \div 1$ ms and $\sim 1 \div 10$ µs, Table 1.6). Again, from Figure 1.24 it is clear that writing is performed with almost the same speed for all presented EMTs, besides capacitor–type FeRAM and PCM, that can reach several tens of nanoseconds ($\sim 30\,ns$ for C–FeRAMs and $\sim 50\,ns$ for PCMs, Table 1.6 and Table 1.7). For both reading and writing operations, however, Static RAMs still remain the fastest memories.

Last but not least, in Figure 1.25 are illustrated the indicative values of endurance in logarithmic scale (base 10). What is important to highlight is the limited number of cycles of Resistive RAMs and FET–types FeRAMs:

this is significant also because these two technologies are maybe the most promising candidates for computing–in–memory applications. It is to remind that a low endurance represent a huge limitation for non–volatile memories development. However, both ReRAMs and FeFETs endurance can be theoretically improved up to 10^{12}, as reported respectively in Table 1.5 and in [40].

1.9 Conclusions

This chapter aims to briefly introduce the state of the art of the currently most developed emerging memory technologies, both in working principle as well as in benefits and constrains that they could provide for a future LiM application. The focus is centered on physics and parameters of the different memories at cell–level. Obviously there are a lot of other possible technologies that are arising in research field (Figure 1.1), such as Mott, Q–dot based, Racetrack, Carbon Nanotube (as seen in subsection 1.6.2 it can be considered as a type of NEMS–based memory), molecular, etc.; nevertheless they will not be considered in this work because of both their maturity level and their performance, that are still not comparable to the presented EMT.

A study for peripheral circuits will be done in chapter 2, while chapter 3 will be dedicated to logic applications exploiting EMTs. As will be discussed in section 3.1, in the next chapters four technologies will be selected among the analyzed EMTs since considered the most promising candidate for in–memory computing.

Chapter 2

EMT: arrays and peripheral circuits

2.1 Technology choice

All the presented emerging solutions for CMOS substitution in memory implementations are valid and could reach high level of production in few years from now. Nevertheless, for a better analysis, a choice must be done in selecting only few technologies among them: in fact, it would be too long —for the aim of this work— to take all of them into consideration for a LiM application.
Hence, from this point on, only four technologies will be further discussed: Magnetoresistive RAMs, Resistive RAMs, Phase change RAMs (that can be classified as a type of ReRAMs) and Ferroelectric RAMs (in particular, the one based on FeFETs). This selection is determined not only by discussions on performance and benefits/constraints (made in the last chapter for each technology), but also with respect to the maturity level associated to such memory implementations.

In this context, in Figure 2.1 it is presented a possible EMT distribution at hierarchical level, at least for what it is predicted for the short–term future: while most of PCMs, RRAMs and MRAMs are currently employed as non–volatile embedded memories, future prospects tend to see them occupying an intermediate level between Solid State Drives (SSD) and main memories (DRAM), called Storage Class. Within few years, MRAM may be employed also in lower Cache levels; instead, for L1–L2 Chache memories,

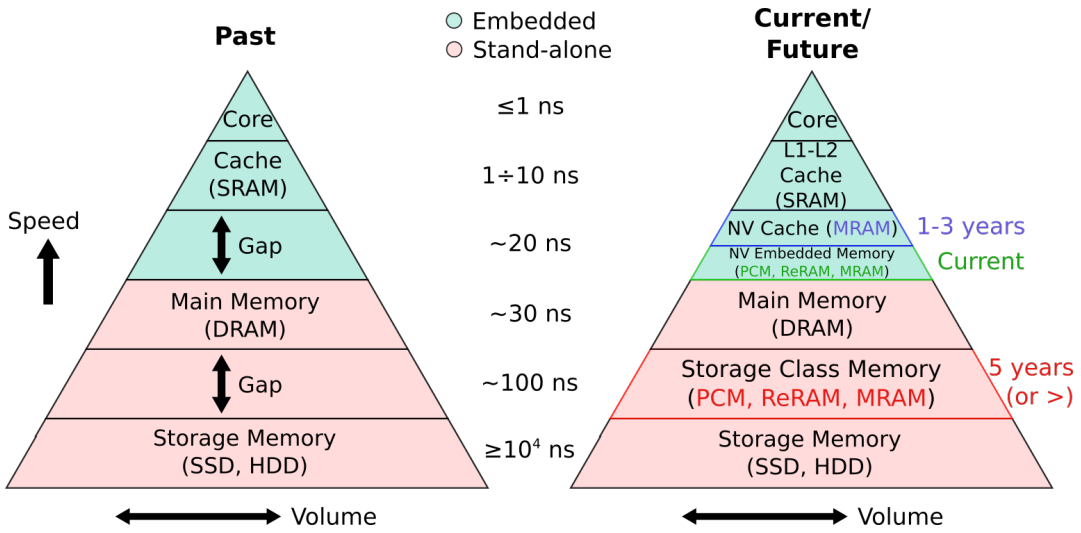


Figure 2.1. Hierarchical memory classification, with past/current status and future predictions.

CMOS SRAMs still maintain the leading role [3].
In addition, also Ferroelectric FETs will be further investigated, due to all the benefits —presented in subsection 1.4.3— that they offer for a possible LiM application.

2.2 STT–MRAM: towards a practical application

It is well known that memory chip organization can be simply divided in cell array and peripheral circuitry: this last one includes row and column decoders, multiplexer, timing control, write driver and sense amplifier (etc.), as discussed for a standard Spin Transfer Torque MRAM architecture in [41]. This subject will be further explored, for STT–MRAMs, in subsection 2.2.1 and subsection 2.2.2, including the considerations about sense amplifier classification done in Appendix A.

2.2.1 STT–MRAM: array and peripherals

A conventional 1T1MTJ STT–MRAM array is depicted in Figure 2.2: it can be noted that such structure is very similar to the one of a DRAM array, since both present transistors that allow access to the content of the cell. The

two arrays differ only for the distinct storage elements, capacitor for DRAM and MTJ for STT–MRAM [42].

In literature, it is more frequently considered an in–plane magnetized MTJ (iMTJ, as the one in Figure 2.2) for array cells implementation. However, according to [42], the perpendicular magnetized MTJ (pMTJ, as depicted in Figure 1.5) is a more recent MTJ variant that needs a much lower current for writing operation than iMTJ one.
The consequences of adopting an iMTJ rather than a pMTJ are further investigated in [43]. The two configurations only have different magnetic anisotropy and do not depend on the shape of the free layer.

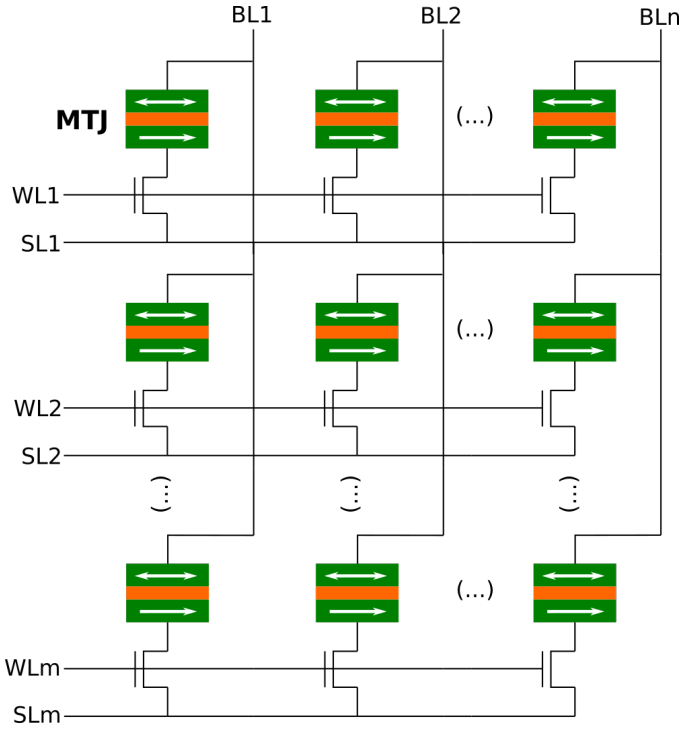


Figure 2.2. Conventional 1T1MTJ STT–MRAM array.

On the other hand, peripheral circuitry design strongly influences all the operations that take place in the memory. The reading performance of a STT MRAM cell, for example, is heavily dependent on the choice of the Sense Amplifier (SA), for which different implementations are possible (see Appendix A), or on the picked reference cell (the standard one is shown in Figure 2.3).

For instance, in [44][1] it is presented a very simple cell programming/reading circuit; however, the low number of employed components and signals may lead to poor control and/or approximate results. In the same article it is reported also a modified sense amplifier (Chung SA, based on current latched SA, see Appendix A), able to provide rather high sensing speed, and that even models parasitic contributions for a more accurate simulation.
In [45][2], instead, it is tested a more complex circuitry for reading and writing, able to provide a reference current I_{REF} equal to I_{AP} or I_P —if the stored information is 1 or 0, respectively— thanks to two reference cells[3]. This improves a lot sensing accuracy, and a better control is offered by means of supplementary signals (Clamping, Precharge, both reading/writing enables); however, the additional components lead to a greater cell area.

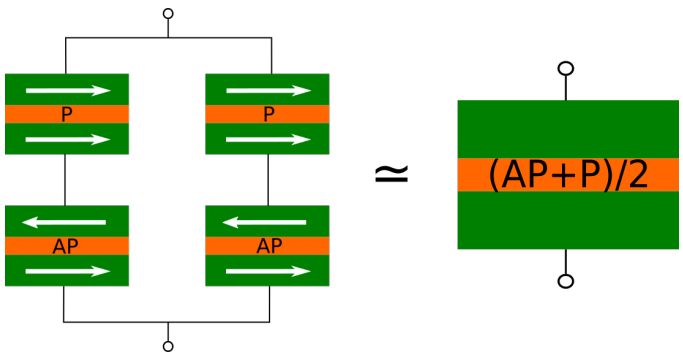


Figure 2.3. Standard reference cell, used in SA circuits to provide the proper reference current for distinguishing —through a comparison with the sensed current— between LRS and HRS. The MTJs bi–parallel on the left is equivalent, in terms of resistance, to the MTJ on the right, i.e. to the average of the resistances of the two (parallel P and antiparallel AP) MTJ states [44].

Obviously, also physical conditions have some impact on memory processes: for instance, the switching circuit works faster at higher temperatures, since their effect is to turn down the energy barrier associated to the MTJ insulating layer. Moreover, a greater voltage between bitline and source line can decrease switching delay, since it increases the current involved in the re–orientation of MTJ magnetization; however, a too high bias could cause

[1]It is intended figure no.7, at p.3 of article [44].

[2]It is intended figure no.10.a, at p.4 of article [45].

[3]In this case they are simple MTJs in P and AP states.

breakdown of the MTJ barrier. Another parameter is the so–called Tunnel MagnetoResistance (TMR) (introduced in subsection 1.2.1): the higher is the distinguishability between MTJ states, the faster —and with a lower power consumption— is reading operation [44].

2.2.2 Considerations on SAs for STT–MRAM

STT–MRAMs are maybe the most flexible technology[4] in terms of choices for sense amplifiers: they do not have the constraint of low operating voltage as for memristor crossbar arrays (see subsection 2.3.2) and do not require compensation mechanisms for GST material peculiarities as for Phase Change Memories (see subsection 2.4.2).

For instance, in [41] it is proposed a basic voltage latch–type SA for STT–MRAM implementation, while in [44] (as already mentioned in subsection 2.2.1) it is reported a "Chung" SA, which is simply a fast current latched SA that includes parasite capacitances for more realistic results.
In [49] a latch–based SA, called Pre–Charged Sense Amplifier (PCSA), is claimed to be capable of high–reliable, low–voltage sensing operation and considered as a good candidate for mixed CMOS/MTJ hybrid logic.

Hence, depending on the performance trade–off (e.g. low voltage/high speed, or other parameters) that is set at design level, there is a wide range of possible choices for STT–MRAM sense amplifiers.

2.3 ReRAM: towards design level

2.3.1 ReRAM arrays

Differently from STT–MRAMs, whose cell in general requires a transistor to access MTJ content, memristive memories can adopt three different types of array: 1T1R, 0T1R (or 0D1R) and 1D1R, where "R" stands for memristor (since it is, in practice, a variable resistor).

A typical 1T1R array is depicted in Figure 2.4 (a) [50]. The dedicated access MOSFET guarantees an easier control of the cell, but it also increases

[4] Among the ones selected in the current chapter.

its size; actually, cell dimensions will be dependent on transistor ones, since memristors are usually much smaller than MOSFETs. This is further aggravated considering that ReRAM access transistors are usually bigger than DRAM ones due to the much higher operating current. Moreover, a larger cell area implies higher fabrication costs.
Besides that, the access MOS offers an optimal insulation of the selected cell avoiding disturbs on other array cells [50, 51]. As a consequence, 1T1R array provide better energy efficiency and lower access time with respect to other arrays.

0T1R and 1D1R (Figure 2.4, (b) e (c)) are instead named crossbar (Xbar) structures, since the ReRAM cells are located at each cross–point of the array; such layout is much simpler than 1T1R, since the cells are just sandwiched between two mutually perpendicular metal wires, but it also induces higher sneak currents and worse noise margin [50, 51].

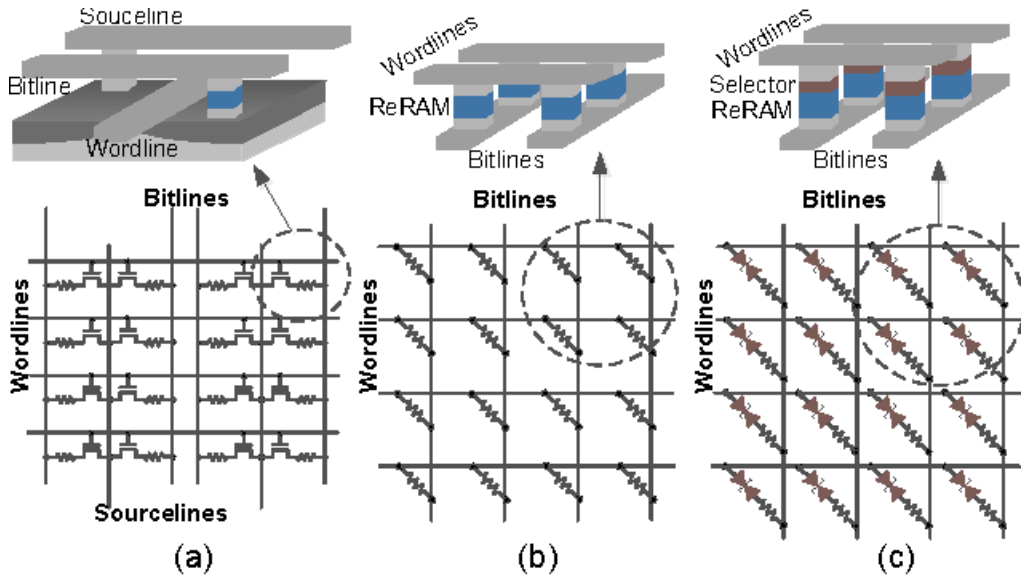


Figure 2.4. ReRAM 1T1R (a), 0T1R (b) and 1D1R1 (c) arrays [50].

The only–memristors array (0T1R) offers the smallest cell structure at all. In absence of any device for accessing the cell, such task is relegated to the intrinsic nonlinear characteristic of memristors. Unfortunately this leads to difficult target cell insulation, since current tends to flow in unselected cells, i.e. considerable sneak currents are present in the array. This contributes in

augmenting energy request per area unit [50, 51] and increases the probability of reading errors. To mitigate such issue, usually special bias schemes are employed, see subsection 2.3.3.

The 1–diode 1–memristor crossbar cell array is depicted in (Figure 2.4, (c)). A bi–directional diode acts as access device to the cell, but with much lower area occupation than 1T1R. With size similar to 0T1R structure, 1D1R array presents the advantage of reducing significantly sneak currents. However, its main limitation is the voltage requested for programming, that is higher than the one for 1T1R and 0T1R arrays. For such matter, charge pumps —that elevate voltage level— are usually employed, but they increase area and design complexity [50, 51]. Also Phase Change 1D1R array presents the same issue (see subsection 2.4.1).

Even if, in ReRAM arrays, the memristor is often represented as a resistor (or variable resistor), its conventional symbol is the one depicted in Figure 2.5. This representation takes into account also the polarity of the device, represented by a thick black line at one of the edges. An example of 0T1R Xbar array that employs such symbol is reported in Figure 2.6.

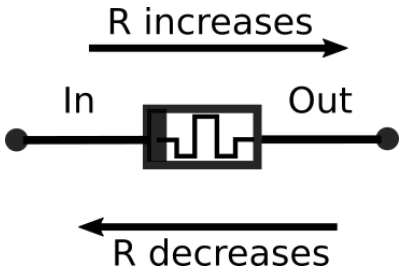


Figure 2.5. Conventional memristor symbol.

2.3.2 ReRAM peripherals

As stated in subsection 2.2.1 for STT–MRAMs, the design of reading/writing interface plays a main role in influencing memory switching and sensing speed. Hence it is important to understand, for example, if there are design constraints that affect only some types of ReRAM arrays.

In [52] it is stated that ReRAMs have to face some criticities in designing peripheral circuits; in particular, a reliable sensing margin is hard to

maintain due to its high sensitivity to process–voltage–temperature (PVT) variations, especially with the scaling of the device.
Furthermore, high-density ReRAMs —with several cells integrated on the same bitline— involve very high BL parasitic capacitance, which worsens sensing speed. For these reasons, in ReRAMs current–mode sense amplifiers are usually adopted, rather than voltage SA. Nevertheless, basic current SA (so excluding ACLSA and High–speed low–power LSA described in Appendix A) cannot work at a too low V_{DD}, especially when sensing speed and margin are reduced.

Moreover, further limitations on the choice of ReRAM sense amplifiers are found for crossbar arrays, as it will be explained in the following. 1T1R structure, instead, seems to be the most flexible for SA choice.
Again in [52] it is reported the scheme of a current SA conventionally employed in 1T1R ReRAMs. In the same article, some modifications to such circuit are introduced (as Amplifier–Assisted load PMOS, AAP, and Dynamic Pre–charge Circuit, DPC), turning it into a so-called Reference Clamping Current Sense Amplifier (RC–CSA). This modified sensing scheme offers improved operating speed and can work at lower V_{DD}.
With reference to 1T1R ReRAMs, in [53] it is proposed a current–mode SA based on a current mirror and thin and thick oxide transistors respectively for accelerating BL charging and for supply shield. What is interesting in this example are the three possible operating modes (sense, set/reset verification) that allow also to verify the cell states during programming.

1D1R arrays —as mentioned in subsection 2.3.1— usually need charge pumps, due to the higher voltage level required for programming. Such circuits make the design more complex and increase peripherals area [51]. For instance, in [54] it is proposed a charge pump control system that tries to optimize power efficiency in 1D1R array.
In the same article, it is also presented a write circuit with leakage compensation: leakage is in fact particularly significative when more cells on the same bitline are programmed into LRS, and may cause sensing errors. In this case the SA simply samples the leakage current before the writing pulse arrives, producing a BL compensation current.

Memristive–only crossbar arrays (0T1R) require in general high resolution sensing circuits; to handle this, methods for offset cancellation are proposed,

but they usually induce area increasing and matching issues. Moreover, memristors reading operation can be performed only at low voltages, and this adds more limitations to the design of a proper sense amplifier [48].
Hence, in order to obtain a fast sensing at low V_{DD} and very low BL voltages, the best solution for memristor–only cells seems to be a VLSA–like sense amplifier (see Appendix A), as stated in [48]. In this article it is designed, indeed, a slightly–modified voltage latched SA in order to deal with 0T1R crossbar array requirements. Here it also claimed that many other SA layouts were experimented, and that several Monte Carlo simulations were performed to support their design.

Other considerations still must be done concerning an SA design suitable for Xbar 0T1R arrays.
Firstly, given the great difference between very small memristor–only cells and SA circuits, it is unreasonable to deliver a dedicate sense amplifier for each ReRAM bitline (e.g. in contrast to DRAMs reading interface). In order to manage such matter, some solutions are proposed. For example, sense amplifiers located between adjacent arrays may be shared between these arrays themselves. In [48], for example, it is specified that one SA needs to be shared between each pair of array columns. Another possibility consists in sensing only a small portion of cells of the selected row, by exploiting BL multiplexers [51].
Secondly, reading informations stored into a ReRAM cell is less accurate when no insulating access device is present, as in 0T1R cells [55]; for this reason complex SA are often employed to provide at the same time both high sensing accuracy and high access speed [56].

In the next section, other 0T1R design implications are presented, but this time they do not directly regard peripherals layout: instead, they are biasing methods necessary for reliably operate with Xbar arrays.

2.3.3 Biasing challenges of 0T1R cross–point structures

In this section the focus will be only on 0T1R Xbar arrays and the expedients that must be adopted in order to perform reliable operations on them. The reasons for concentrating on such arrays are summed up at the end of this section.

The absence of an access device —such as a diode or a transistor— makes

the insulation of the selected cell very difficult. For example, when applying a certain write voltage on the wordline of the target cell, it happens that also other unselected cells of the same WL are biased [51]. To mitigate this effect, specific writing bias schemes such as V/2 and V/3 are usually employed. They are discussed here in order to understand their benefits and constraints. Also the effects of a floating line configuration are taken into consideration. The analysis, however, is done without considering wire resistances in the array.

First of all, for better clarity, it is necessary to divide the array cells into groups, as done in [57]. Group classification is depicted in Figure 2.6: besides the selected cell (whose WL is at writing voltage **V** and BL is grounded), it is possible to distinguish among unselected and half–selected cells. Half–selected cells share the same bitline or wordline of the selected cell.
This subdivision is just done to simplify the analysis of the bias schemes.

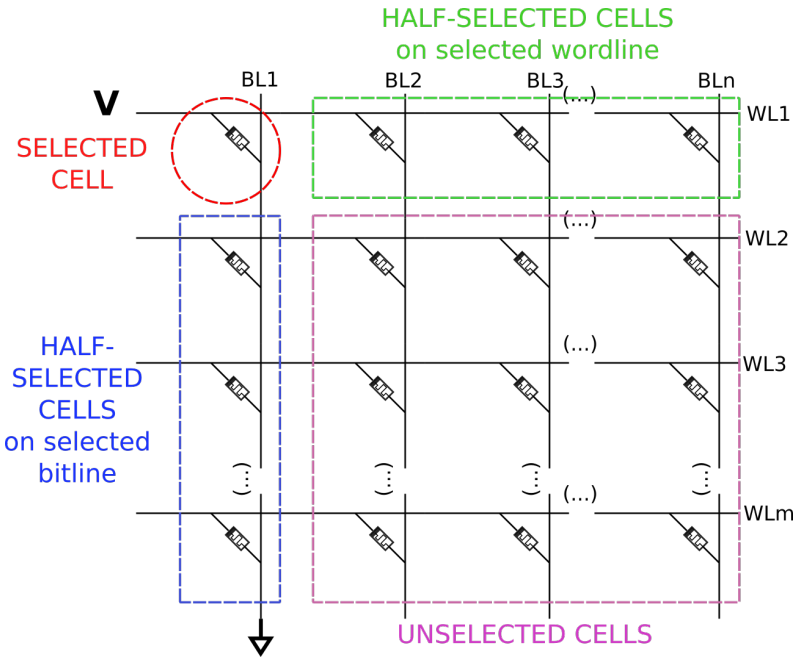


Figure 2.6. 0T1R ReRAM array cells groups.

In floating line configuration, writing operation is performed applying V_{WRITE} and 0 V respectively on the wordline and bitline of the selected cell, while all the other lines are left floating.
As it can be seen from Table 2.1, half–selected and unselected cells undergo a

certain voltage drop that depends on the geometry of the array. This writing disturb voltage can be kept under $V_{WRITE}/2$ only when the number of rows (m) and columns (n) is equal, hence for an array with 1:1 aspect ratio. In this way, the correct functioning of such scheme is heavily dependent on the array structure. Moreover it needs to be mentioned that a voltage near to $V_{WRITE}/2$ may cause unwanted switching of cells after a certain time [57].

Floating Line Scheme	**Voltage Differential**
Selected cell	V_{WRITE}
Half–selected cell on selected BL	V_{WRITE}(m-1)/(m+n-1)
Half–selected cell on selected WL	V_{WRITE}(n-1)/(m+n-1)
Unselected cells	V_{WRITE}/(m+n-1)

Table 2.1. Voltage drop related to cells groups in Floating Line Scheme [57].

It can be noticed that half–selected cells are the ones that have the bigger spurious voltage drop across them (Table 2.1); due to this partial write bias, they leak current, contributing in increasing sneak currents in the crossbar array.
To mitigate leakage, V/2 and V/3 biasing methods are usually adopted. The efficiency of such schemes is given by the fact that, depending on the V–I non–linearity of the memristive element, even a small reduction of voltage drop across half selected and unselected cells may lead to significant decrease of sneak current [50].

V/2 method is the most known scheme used in crossbar arrays to reduce sneak currents.
It consist in supplying both unselected wordlines and bitlines with $V_{WRITE}/2$, with selected row and column respectively at **V** and 0 V. This limits the voltage drops across half–selected cells to $V_{WRITE}/2$, while unselected cells are subjected to a null differential (Table 2.2).
Hence, half selected cells are the only ones, in this scheme, to undergo spurious voltage drop. They represents less than the 50% of the total mxn array. However, the probability of write disturbs is still quite high in V/2 scheme, since errors may occur specially when such differential voltage ($V_{WRITE}/2$) is maintained for a certain amount of time [57].

V/3 bias scheme, instead, consists in applying again V_{WRITE} and 0 V respectively to the selected WL and BL, but this time unselected rows are

V/2 Bias Scheme	Voltage Differential
Selected cell	V_{WRITE}
Half–selected cell on selected BL	$V_{WRITE}/2$
Half–selected cell on selected WL	$V_{WRITE}/2$
Unselected cells	0 V

Table 2.2. Voltage drop related to cells groups in V/2 Bias Scheme [57].

biased with $V_{WRITE}/3$ and unselected columns are at $2V_{WRITE}/3$. This leads to the voltage drops reported in Table 2.3: the maximum voltage drop above half selected cells is reduced to $V_{WRITE}/3$, but the number of cells with such spurious voltage increases, since also unselected cells are subjected to it. So in V/3 scheme more cells contribute to sneak currents, but the probability of write disturbs is reduced with respect to V/2 method.
Although V/3 scheme seems the best option among the proposed biasing methods, it has to face an heavy power consumption due to the increased number of "sneaky" cells and to the necessity of switching a lot between $V_{WRITE}/3$ and $2V_{WRITE}/3$ voltages.

V/3 Bias Scheme	Voltage Differential
Selected cell	V_{WRITE}
Half–selected cell on selected BL	$V_{WRITE}/3$
Half–selected cell on selected WL	$V_{WRITE}/3$
Unselected cells	$V_{WRITE}/3$

Table 2.3. Voltage drop related to cells groups in V/3 Bias Scheme [57].

The three schemes are summarized in Table 2.4, considering this time selected/unselected rows and columns (instead of the initially adopted subdivision groups) together with the associated voltage bias.
Another table is reported in [57][5], which completes this discussion providing specific formulas for minimum and maximum voltages, leakage currents and number of disturbed cells.

Unfortunately, leakage–limiting bias schemes are not the only design impli-

[5]Table no. 4 in the article

	Floating Line Scheme	**V/2 Scheme**	**V/3 Scheme**
Selected row	V_{WRITE}	V_{WRITE}	V_{WRITE}
Selected column	0 V	0 V	0 V
Unselected row	Floating	$V_{WRITE}/2$	$V_{WRITE}/3$
Unselected column	Floating	$V_{WRITE}/2$	$2V_{WRITE}/3$

Table 2.4. Voltage associated to rows/columns for the presented bias schemes.

cation that memristive Xbar arrays require.
Problems arise also when multiple cells are selected on the same row. In this case it is difficult to find a correct value for unselected lines, risking unwanted set or reset operations; furthermore, these two cannot be performed at the same time in a Xbar array. About this, two solutions are proposed in [58], one based on the separation in time of set and reset processes, the other one on erasing the cell before a reset operation.
Another issue regarding 0T1R crossbar array is related to reading operation. Up to now have been considered only the effects of leakage on writing to the array; the problem is that leakage associated to half selected cells may cause also reading errors. A critical case is when the target cell in in HRS while half selected cells are in LRS. Again in [58], two possible solving options are described: one consists in supplying with the same voltage the unselected rows and the column containing the target cell. The other one, more at peripheral level, involves the isolation, through 2–steps sampling, of the sneak current of half–selected cells.

At this point, a spontaneous question arise: why centering the focus on 0T1R crossbar ReRAM, if they imply so much complications?
The answer is simple: even considering all 0T1R ReRAM drawbacks as array (subsection 2.3.1), in peripheral implementation (subsection 2.3.2) and in biasing schemes (subsection 2.3.3), they still represent a huge aspiration for future memory implementations.
Not only among other ReRAM arrays, but among all the emerging memory technologies, 0T1R cells in Xbar arrays can —in theory— reach the smallest area ever. Due to prospective of great integration density, and rather high sensing and switching speed, they are probably the most promising alternative to CMOS technology by far. Unfortunately, for now this advantage is only theoretical, since, as anticipated in subsection 1.3.3, significative leakage problems make them not feasible in practice, yet.

2.4 PCM: towards design level

2.4.1 PCM arrays

As anticipated in subsection 1.1.2, Phase Change Memories are sometimes classified as a type of ReRAMs. Differently from them, however, the majority of optimized PCM architectures do not even consider the possibility for crossbar arrays without access devices [50]. Hence, phase change RAMs commonly exploit arrays whose cells have dedicated selectors, such as MOSFETs, BJTs and diodes.

Among all selector devices, diodes can offer the minimum 4 F^2 size; BJTs, that usually occupy about 8 F^2, can reach 5.5 F^2 by sharing their base contact with multiple cells. MOSFET area is instead heavily dependent on the amount of current necessary for programming the cells.

PC memories needs, in general, high RESET current: this is in fact necessary, for the crystalline phase change material, for reaching —thanks to Joule heating— the melting point temperature, in order to be successively cooled–down into amorphous high resistance state (HRS). Temperature pulses are indicatively depicted in Figure 1.17.
Given that, PCM cells generally require high quality BJT and diodes or quite large MOSFETs for facing such current [59]. Examples of 1T1R and 1D1R PCRAM arrays are depicted in Figure 2.7.

Thanks to their potential for higher integration density, diodes seem to be the best option for implementing PCM arrays. It needs to be considered, however, that such devices require an accurate array and peripheral design in order to mitigate the parasitic effects, including the vertically and laterally formed BJTs and thermal crosstalk. Leakages also arise in the cell due to the fact that current directly flows —through the PC material and the 2–terminal diode— from the the selected BL to a grounded WL.
Moreover, diodes in general operate at higher writing and reading voltage than MOSFETs. The difference between the operating voltages of the two devices is about 1 V, as it can be noticed from Figure 2.8 [60].

Furthermore, it can be quite challenging to seek out good diodes featuring both high ON/OFF ratio and large programming current. An alternative could be to search for materials or other typologies of access devices that can provide a proper degree of nonlinearity between low and high voltage levels

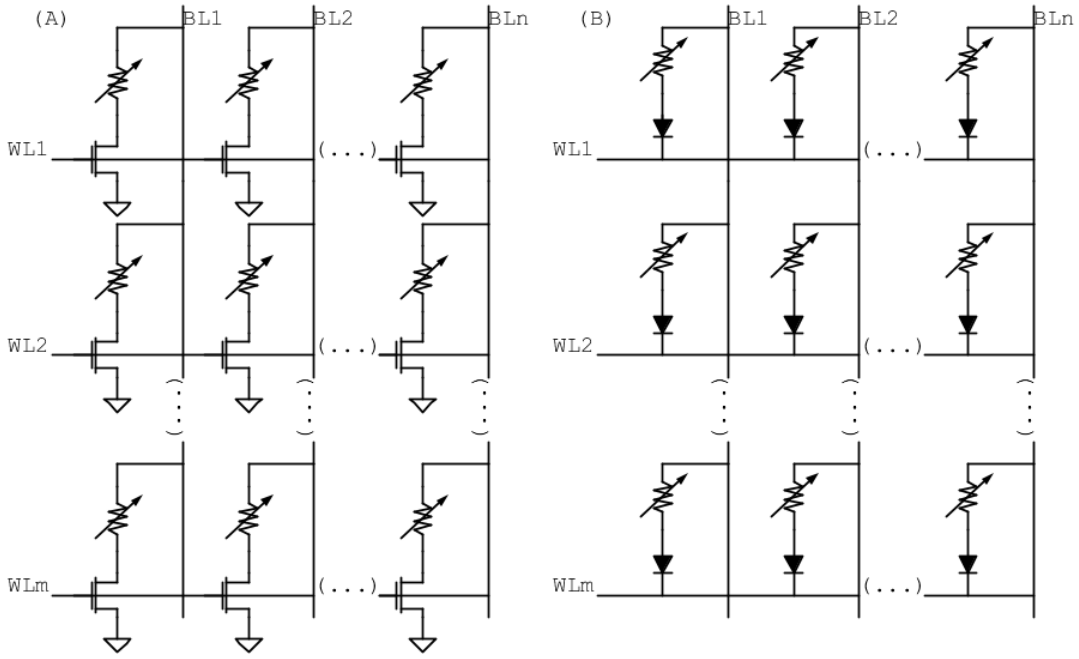


Figure 2.7. PCRAM 1T1R (a) and 1D1R (b) arrays. The variable resistor represents the phase change material.

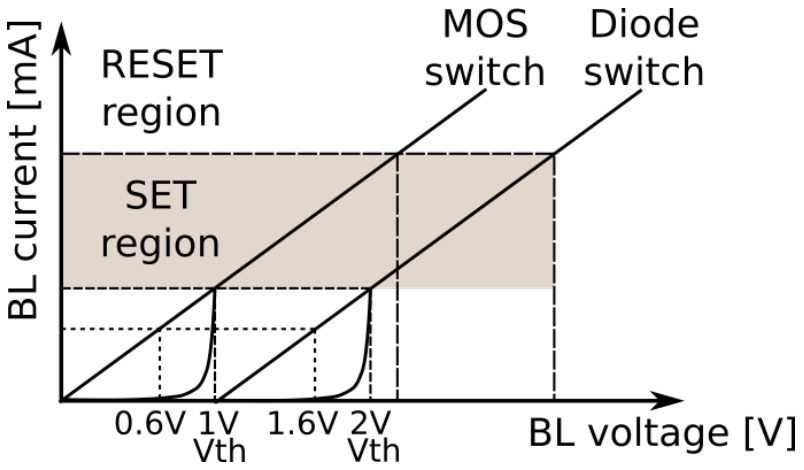


Figure 2.8. BL current–voltage characteristic of MOSFET and diode switches.

[59]. However, reasearch on PCMs has confidence on new (future) methods for providing enough heating to the cells for trying to minimize the high current request. Also, phase change material–only crossbar arrays (0T1R) would be a huge step forward for PCRAMs design [59].

2.4.2 PCM peripherals

High write current requirement for PCMs may be an issue for the design of peripheral circuitry.
To deal with this, a possible approach may be to exploit an external power supply (on–chip or off–chip). However, the usage of charge pump circuits for elevating the voltage level seems to be the best solution: firstly, because PCRAMs need, in general, more than one boosted voltage level in different memory components. This requires the presence of numerous external power rails, that implies the addition of several chip pins, which, in turn, leads to an higher chip implementation cost. Secondly, because a precise and fast control is required for the involved write voltages, and this can not be provided by external power supplies [61].

Charge pump systems are used for turning a lower input voltage into higher output voltages, and are employed both for 1T1R and 1D1R arrays.
In [62], charge pumps operate for increasing the voltage level for the write driver (from 1.8 V to about 4.5 V) during writing in a MOSFET–based PCM array; otherwise, with only an external 1.8 V voltage, the resulting write current would be too low to guarantee a correct switch, due to the presence of parasitic resistances. Boosted voltages are also applied to the gates of selection MOS to lower their channel resistances.
In [60], instead, it is discussed a charge pump system for a diode–based PCM array. A boosted voltage is used as basic voltage for reading and writing operations in the array: it is higher than MOSFETs supply voltage (V_{DD}) by 1 V, difference that takes into consideration the built–in voltage of diodes (Figure 2.8). In a read process, such voltage is reached through a 2–steps operation, that consists in precharging the line to an intermediate V_{DD} level: this reduces the burden of charge pumps.

Unfortunately, charge pumps introduce several issues at design level.
A tipical charge pump circuit includes rather large cascaded capacitors and transistors, that contribute in making the voltage rise, step by step. Such elements imply the presence of large parasitic capacitances, which, during charge and discharge cycles, make the power consumption increase quite a bit. Furthermore, due to the presence of wide transistors and internal high–voltage nodes, leakage power may be significant in the circuit. Another contribution to power consumption is provided by charge pump related peripherals (i.e. drivers, clock signal handling, control elements, etc.) [61].

As a consequence of all these dissipative effects, power efficiency (output power /input power) decreases a lot. According to [60], during reset operations on a diode–based PCRAM with 0.6÷1 mA writing current, only a 20% power efficiency has been obtained. It is to say that, in reset processes, charge pumps operate with larger parasitic power loss.
Moreover, to satisfy high current request a single charge pump unit may be not sufficient, an this leads to additional chip area occupation.
In addition, charge pumps present high latency of charging (to a target output voltage) process, because of their big load capacitance. Sometimes, discharge/charge operations at each memory access end/request are forced for improving the reliability: in fact, frequent charging/discharging can decrease the time interval in which they are exposed to possible dielectric breakdown of their inner gates oxides. Such phenomenon is particularly problematic in charge pumps, since they work with voltages higher than V_{DD} [61].

In [61], simulation results —on proposed PCM design— show that 81% of total power consumption is associated to charge pumps, with 60% related to parasitic contributions. For this reason, in the same article it is discussed a method for decrease power dissipation, called RESET scheduling. It consists in lowering the power peak for writing in PCM arrays by scheduling the reset operation along all the duration of the writing. It is demonstrated that such technique leads to a 70% decrease of charge pumps area and dissipated power.

In summary, charge pumps are suggested for handling the high writing current request by both 1T1R and 1D1R PCM arrays, and an optimized design helps in limiting area occupation and power consumption.

Some other considerations can be done about PCRAM peripherals.
For instance, in [63] it is proposed a modified writing/reading interface, which takes into account the bitline resistance (R_{BL}) dependency on the distance that separates the write driver from the PCM cells.
For what concerns sense amplifier design, different solutions for PCRAMs were observed. Among all, one is particularly relevant for its considerations about PCM requirements in reading process: in [64], in fact, it is proposed an high performance SA that takes into account some peculiarities of PCRAM sensing. It is stated that the chalocogenide material (e.g. GST) suffers of possible "destruction" of its phase state when the heating caused by the current flow overcomes heat dissipation rate of the PCM cells. Another fact may be problematic: when the phase change material is in amorphous high

resistance state, a voltage greater than threshold one may cause breakdown in the material. In this case, the PCM cell abruptively changes to low resistance state even if the phase change material is still in amorphous state. In order to mitigate such effects, a specific SA is designed; however, the resulting circuit it more complex and occupies greater area than "conventional" ones (presented in Appendix A).

In conclusion, PCRAMs present array layouts very similar to ReRAM ones (with the exception of 0T1R crossbar); for what concerns peripheral circuits, they usually employ charge pumps and have some peculiarities that may affect the sense amplifier design. High power consumption is one of the main issues of PCM design.

2.5 FeFET: towards design level

2.5.1 FeFET arrays

The last memory technology explored in this chapter is the FET–type FeRAM.

FeFET memory arrays strictly remind FLASH ones, with the only difference that a ferroelectric FET is at the basis of the cell, instead of a conventional MOSFET.
Nowadays, FeFET array analysis is focused on AND and NAND types. The first architecture is maybe the most studied one [25]: its structure is analogue to a NOR FLASH, differing for the presence of multiple souce lines (SLs), as shown in Figure 2.9. The voltage is directly applied to the FeFETs terminals without the need for selector devices.
The second one (Figure 2.10), differently from AND type, requires selecting transistors at the beginning and at the end of each "string". A string is just a series of all FeFETs on the same column. Such selectors are necessary in order to provide a better insulation from the BL and the Common Source Line (CSL). Hence, only the voltage applied to FeFETs gates is applied directly through the WLs [65].
In particular, NAND arrays —with respect to AND type— allow for smaller FeRAM cell area. This implies the possibility for higher integration density; moreover, NAND array can be used for 3D stacking, as done in [65].

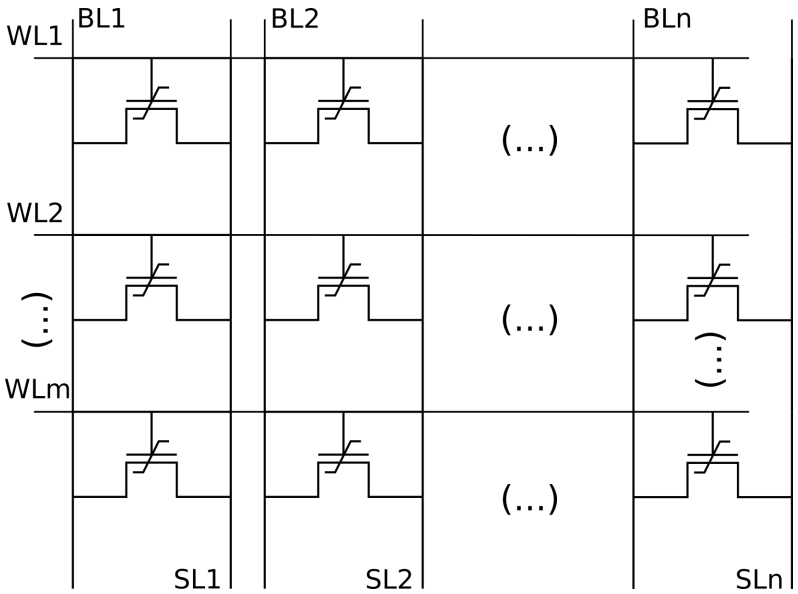


Figure 2.9. FeFET AND array.

FeFET arrays, however, require peculiar biasing schemes in order to minimize the disturb effects associated to writing operation. Such schemes are very similar to the ones exploited in ReRAM crossbar arrays (see subsection 2.3.3).
For instance, V/2 and V/3 biasing for AND arrays are well explained in [66]: exactly as for crossbar ReRAMs, spurious voltage drop in half selected and unselected cells is reduced to V/3 in the homonymous scheme, but in such case the number of disturbed cells is higher than for V/2 bias pattern. For AND FeFET array, the respective biases at each row (WL) and column (BLs and SLs) of the array are the same of Table 2.4.
However, in the aforementioned bias scheme, positive/negative WLs voltages are applied for programming/erasing the cell, neglecting the body influence [25]. To take this effect into consideration, and, at the same time, to prevent such scheme from using negative gate voltages, a peculiar drain–erase all–positive bias scheme is proposed in [65] for both AND and NAND arrays.

2.5.2 FeFET peripherals

Only few examples of peripheral circuits specifically designed for FeFETs memories have been found in literature. The wide range of applications of such devices makes difficult to find articles that are focused on this topic,

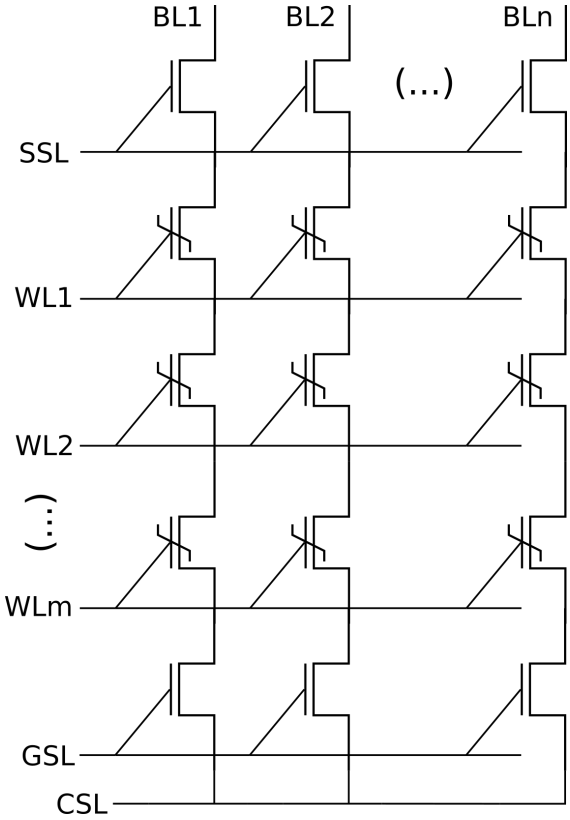


Figure 2.10. FeFET NAND array.

if there are any. For instance, a sense amplifier design for a FeFET memory is described in [40]; however, such circuit is non–conventional, since it is expressly modified for a computation–in–memory application. Another example can be found in [67], where for "FeFET peripheral" is intended a reading pheripheral (again a sense amplifier) implemented with FeFETs instead of standard MOSFETs.

Hence, examples of FeFET–memory peripheral circuits have been found only in articles where the attention has been focused on other aspects of the design, and, for this reason, they were poorly commented or their design has been taken for granted. For instance, in [66], it is stated that, for a 1T–FeFET AND array, a typical choice of sense amplifier is the current–mode one, without justifying why. A rather simple circuit of such SA is reported in the same article, without further descriptions.
In [68], an even simpler SA circuit is presented: in this case, however, the

choice of a current mode SA is explained as for "tap" the significant difference between the drain currents of the FeFETs in ON and OFF states. It is also claimed that, for the described architecture, such choice would minimize sneak currents in unselected cells.

Even if the considerations about peripheral design in FeFET memories are relatively limited, the application flexibility of FeFETs makes them very useful in implementing logic–in–memory; this is mainly due to the fact that, in practice, a FeFET works in a way similar to MOSFETs, but with the additional capability of nonvolatile storage of information. Such concept will be better explained in section 3.5.

Chapter 3

EMT LiM

3.1 Why implementing LiM?

The growing requirement for fast and high–volume data transfer in advanced applications is constrained by the so–called "memory wall": a performance–limiting barrier, mainly caused by the difference between memory and processor speed, the transfer bandwidth between them, and the related communication delay [40, 69].
An innovative solution is represented by Logic–in–Memory (LiM), considered as a valid alternative to conventional Von Neumann architectures [70].
It consists, in practice, in a memory device in which some logic operations, normally carried out by processing units, can be implemented directly into the memory itself. In this way, a smaller amount of data needs to be transferred and, as a consequence, lower latency and energy are involved. For such reasons, LiM is useful especially in data–intensive and/or power–critical memory implementations. However, the range of tasks that LiM is able to perform is still very narrow [40].

LiM implementations with the four technologies selected in section 2.1 will be presented in this chapter; each technology offers peculiar features for in–memory computation. Furthermore, the computations themselves can be performed in various ways. For instance, some approaches are based on modifications on reading/writing interface or on other peripheral circuits; others directly adapt the array cells to make them compute the target operations; furthermore, others do not modify the memory circuitry, and exploit proper current/voltage pulses schemes for logic implementation.

3.2 STT–MRAM LiM

3.2.1 STT–MRAM LiM peculiarities

In addition to the limitations presented in **??**, Spin Transfer Torque MRAMs suffer from the fact that a low TMR (defined in subsection 1.2.1) sensibly affects the reliability of reading operations. Unfortunately, this is even worsened for logic–in–memory applications, but proper correction mechanisms can be exploited to handle it by strengthening the STT–LiM steadiness against process variations [11].

Besides, some unique properties associated to magnetoresistive memories may be useful in LiM design. For instance, their basic cell element, the MTJ, can grow on top of MOSFETs without affecting the cell footprint: this allows magnetoresistive LiM to reach a significative integration level [70]. Furthermore, STT–MRAM basic cells are resistive, and the current involved in writing operation is usually much bigger than reading current. Such features make it possible for multiple array wordlines to be activated at the same time, and, as a consequence, to directly dispose of the results of logic operations by simply reading the informations contained in multiple WLs. Hence, with just one access to the memory, a logic computation can be performed thanks to a properly modified sensing and reference circuitry. This is the case of the first example of LiM application presented in subsection 3.2.2. On the contrary, due to the elevated risk for short–circuits through the array and the resulting information loss, enabling multiple WLs is not feasible in CMOS static RAMs [11].

The simplest way to introduce STT logic is through a series of confrontations among related works: in this sense, in subsection 3.2.2, subsection 3.2.3 and subsection 3.2.4 are presented three selected approaches to introduce STT–LiM.

3.2.2 STT–MRAM LiM, 1st example: modified reading interface

The first example of STT in–memory computation (CiM) is taken from [11]. In this article it is considered a STT–MRAM featuring a conventional 1T1MTJ array, and with modified sensing and global reference generator circuits. Such design exploits the aforementioned (subsection 3.2.1) property of STT MRAMs to allow for multiple wordlines enabling. According to [71], the multiple WLs assertion can be exploited to perform a large range of logic computations in just one single memory access: this represents a strong point for STT–LiM, since it improves performance and energy efficiency. On the contrary, several intermediate access cycles are required by logic approaches such as MAGIC in ReRAMs (subsection 3.3.3).

In Figure 3.1 a couple of standard STT–cells are presented, whose MTJs are modeled by a varying resistance. These represent two target cells that can be read by activating the related wordlines (WL_i, WL_j) and by applying a read voltage V_{READ}. Table 3.1 reports the four possible combinations of resistances (R_i, R_j) associated to two standard STT cells. The corresponding currents (I_i, I_j) are summed, and the resulting current I_{SL}, in turn, assumes four possible values. P and AP labels stand for parallel and antiparallel MTJ states, respectively.

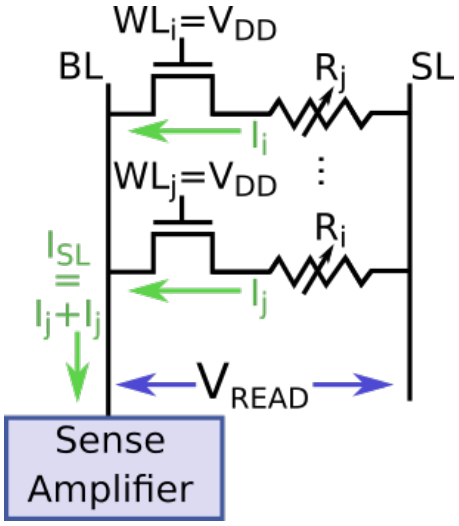


Figure 3.1. Couple of conventional STT–MRAM cells.

(R_i,R_j)	I_{SL}
(R_P,R_P)	I_{P-P}
(R_P,R_{AP})	I_{P-AP}
(R_{AP},R_P)	I_{AP-P}
(R_{AP},R_{AP})	I_{AP-AP}

Table 3.1. Source Line current (I_{SL}) values for the resistance combinations of two STT MTJs.

In this implementation, bitwise OR and AND operations —together with their negations— are obtained as result of a comparison between the sensed current and specific reference values.
For instance, in order to perform an OR operation, I_{SL} and $I_{REF,OR}$ are put at the input of a comparator. $I_{REF,OR}$ is chosen as an intermediate value between $I_{AP,AP}$ and $I_{AP,P}$ (same as $I_{P,AP}$), as depicted in Figure 3.2 (c); hence, the only case for which I_{SL} is lower than $I_{REF,OR}$ is when both the STT cells are in HR antiparallel state ($I_{SL} = I_{AP,AP}$), resulting in a logic '0' as result at the positive output of the comparator. In all the other cases, the result is a logic '1', as expected from an OR operation. The corresponding negated function (NOR) is obtained at the negative output of the comparator, with the same reference current Figure 3.2 (a).
An AND function is implemented again as result of a comparation, but this time the reference $I_{REF,AND}$ is in the middle of the $I_{AP,P}$ ($I_{P,AP}$) to $I_{P,P}$ current range. A logic '1' can be obtained at the comparator positive terminal when the I_{SL} is higher than $I_{REF,AND}$, i.e. only if both the selected cells are in LR parallel configuration. NAND operation, obtained at the negative output of the comparator, exploits $I_{REF,AND}$ as well (Figure 3.2 (b)).

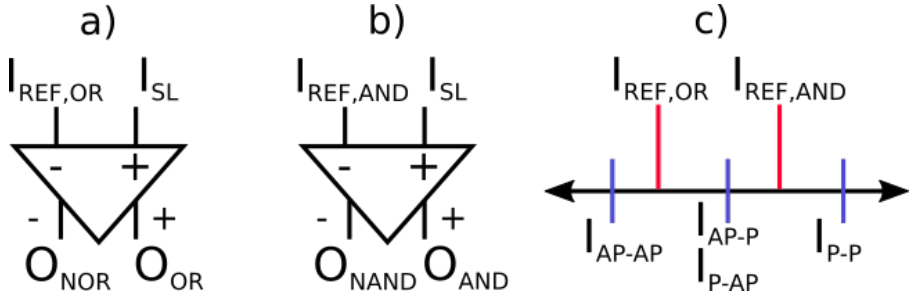


Figure 3.2. a) Bitwise OR sensing scheme, b) Bitwise AND sensing scheme, c) Reference currents.

A potential design complication may be the fact that three different levels of I_{SL} current ($I_{AP,AP}$, $I_{AP,P}$=$I_{P,AP}$, $I_{P,P}$) must be distinguished in order to perform a reliable LiM operation, and not only two as for conventional reading: this means that two reading margins must be considered (instead of one), and that the probability of decision failure increases. Simulations in [11] demonstrate also that the two read margins are different and that the one between $I_{AP,P}$ and $I_{P,P}$ is more predisposed for decision failures.

AND/OR functions can be combined to easily perform more complex operations, such as XOR and ADD. For example, XOR is implemented combining the two comparators in Figure 3.2, such that AND and NOR outputs are given in input to a CMOS NOR gate. Equation 3.1 sums up the corresponding notation.

$$\begin{aligned} A_n, B_\mathrm{n} &= n^{th} \text{ bit of words A, B} \\ O_{AND} &= A_n \cdot B_n \\ O_{NOR} &= \overline{A_n + B_n} \\ O_{XOR} &= A_n \oplus B_n = \overline{O_{AND} + O_{NOR}} \end{aligned} \tag{3.1}$$

A full ADD operation includes a carry input/output (C_{n-1}/C_n) and a sum output (S_n). Since XOR and AND functions (Figure 3.2) can be performed at the same time, only one single access to the STT array is necessary for realizing a full ADD. The details of this operation are recapped in Equation 3.2.

$$\begin{aligned} S_n &= (A_n \oplus B_n) \oplus C_{n-1} = O_{XOR} \oplus C_{n-1} \\ C_n &= ((A_n \oplus B_n) \cdot C_{n-1}) + (A_n \cdot B_n) = \\ &= (O_{XOR} \cdot C_{n-1}) + O_{AND} \end{aligned} \tag{3.2}$$

The overall customized sensing circuit —able to perform OR, AND (and their negations), XOR and ADD— is depicted in Figure 3.3.
On the contrary, no modifications are done to the sense amplifier circuit, that is a Voltage Latched SA (Figure A.3) with additional enable signals.

The principal advantages and disadvantaged of this LiM implementation can be listed as:

- ✓ logic function are simply results of comparations between sensed and reference currents;

✓ more complex LiM functions (such as XOR and ADD) can be easily obtained combining basic functions;

✓ logic operations are performed by a modified sensing circuitry, hence no changes are done on the standard STT array;

✓ since the sensing circuit occupies only a small portion of the overall memory area and power consumption, all the modifications on it do not imply large increase in area occupation and/or power consumption;

✗ an additional input signal[1] has to be addressed at each memory access, which defines the wanted reference current (*rwl*, *rwr* signals in [11][2]) and specifies the type of LiM operation (*sel* signal in Figure 3.3);

✗ three current levels are required, hence the reduced read margins increase the probability for decision failures.

3.2.3 STT–MRAM LiM 2nd example: modified cell and column decoder

Another example of in–memory computation with STT–MRAMs is described in [70].
Logic operations, such as AND and OR, can be performed by comparing the equivalent resistance (R_{MTJ1}+R_{MTJ0}) of two in–series STT–MRAM cells (cell1, cell0) and the sum between a reference cell resistance (R_{REF}) and a logic selection (*logic–sel*) cell resistance ($R_{MTJ,L}$). The last one is responsible for deciding the type of logic function that has to be performed: if $R_{MTJ,L}$ is equal to R_{AP}, an AND operation is selected, while if it is equal to R_P, an OR operation takes place.

The main difference between this LiM implementation and the one presented in subsection 3.2.2 is that in this example the operand cells (*cell1*, *cell0*), as well as the *logic–sel* cell, have a modified architecture, so they can not be considered conventional cells. The modified cell structure is depicted in Figure 3.4: two additional lines (CL and GL) are required to regulate the current flow from the bitline, and their connection to the MTJ is controlled

[1] Called *CiMType* in the reference article [11].

[2] Referring to the overall scheme in Fig. 6 of [11].

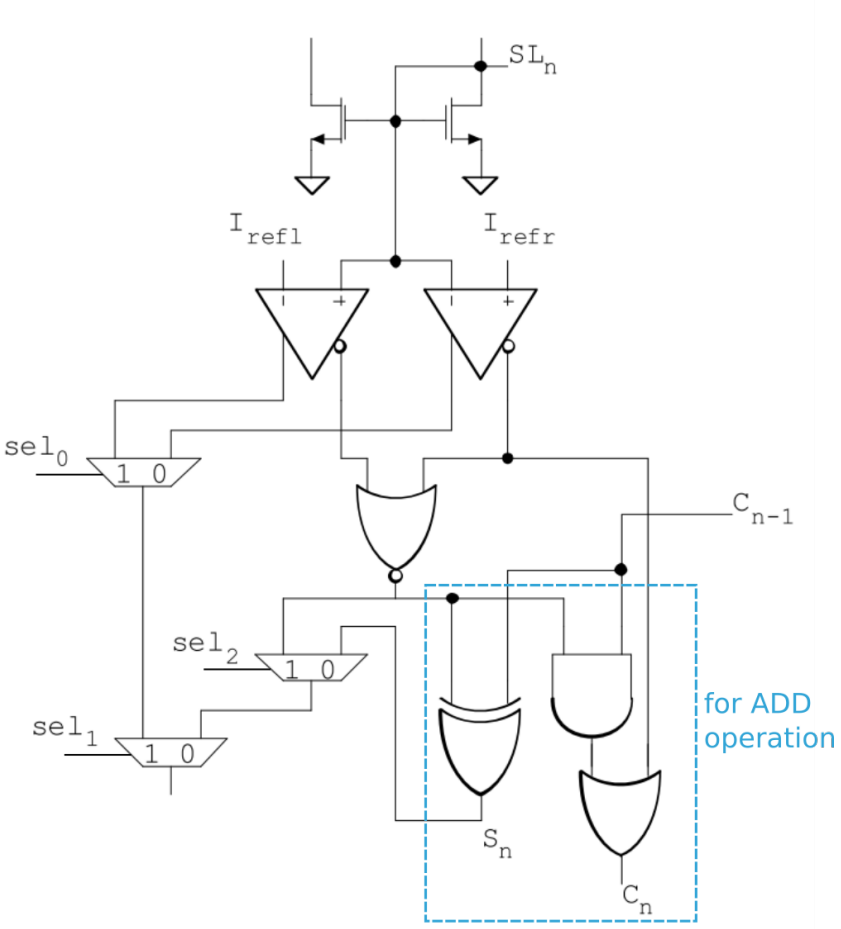


Figure 3.3. Overall modified sensing circuitry for (N)OR, (N)AND, XOR and ADD operations in STT–MRAMs.

by a couple of MOSFETs. The wordline and sourceline are responsible for the transistor switching on/off.

The main disadvantage of such implementation is the use of non–standard array cells. The doubled number of MOSFETs, above all, contributes in increasing the overall area and delay. Moreover, the proposed STT–based LiM requires a modified column controller that manages negated logic and more complex functions such as XOR and ADD. As mentioned before, in fact, only basic AND and OR operations are implemented at cell level.
In practice, what is different with respect to the previous case (subsection 3.2.2) is that here all the computations are handled by unconventional cells together with a customized column controller, instead of a modified

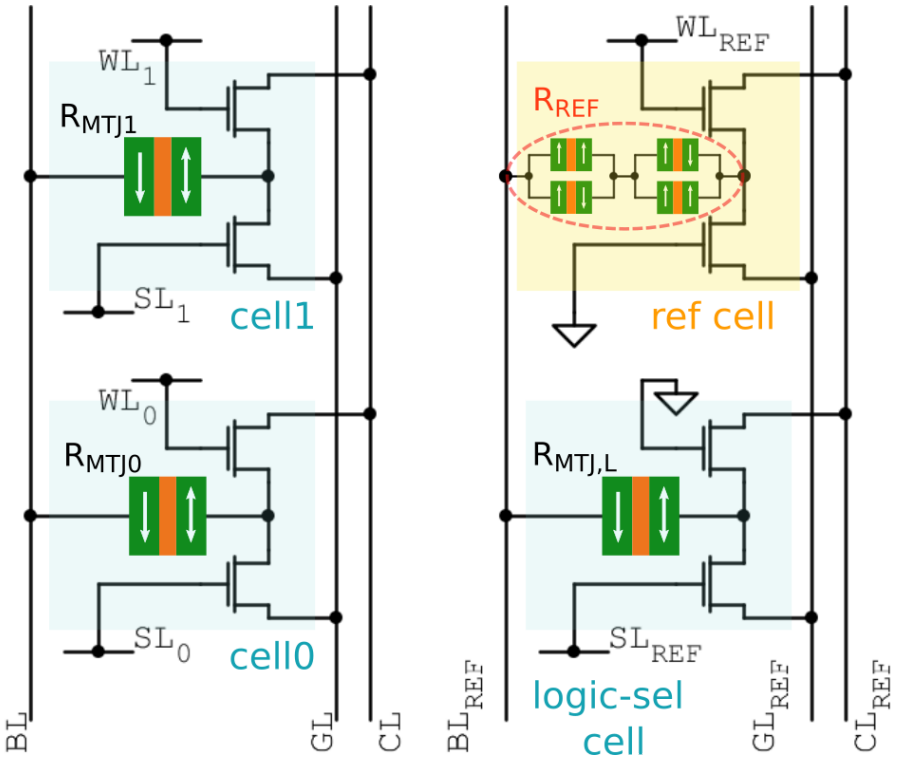


Figure 3.4. Modified STT–MRAM cells for in–memory computing, divided in operand cells (on the left) and reference cells (on the right); *ref cell* is a standard reference cell, as the one in Figure 2.3.

sensing circuitry. In both examples, standard latched sense amplifiers are employed (see Appendix A).

The main benefits and constraints of the presented LiM design are:

✓ logic functions are results of comparations between sensed and reference currents;

✗ the modified cell with larger area and delay (due to the additional transistor) is at the basis of the STT array;

✗ additional control logic in the column selector is required.

3.2.4 STT–MRAM LiM 3rd example: voltage pulses approach

An alternative approach to STT–logic is presented in [72].
It is based on the application of voltage pulses in a certain sequence, depending on the logic operation to perform. In comparison with the previous examples (subsection 3.2.2, subsection 3.2.3) it does not need modified array cells, nor requires additional dedicated peripherals for implementing logic. Hence, a big benefit related to it is the fact that no extra hardware is necessary.

Such LiM application is usually called IMPLY (or just IMP) logic. The reason behind its name is that all the logic operations that can be implemented through it are a combination of logic implications.
For example, a NOR function can be performed in three steps: one state switching (from parallel to antiparallel magnetizations) and two imply operations, through the application of proper voltage/current on the WL/BL of the considered cells.
The previous statement, however, already reveals one of the main constraints of IMPLY logic: even basic functions require multiple serialized operations. This means that such approach may be heavily time–consuming.
Nevertheless, it is noteworthy that IMPLY logic is able to realize a complete set of Boolean operations. IMPLY logic is also employed in crossbar ReRAM LiM applications, as it will be discussed in subsection 3.3.1.
The presented advantages and disadvantages of this approach can be summed up as:

- ✓ no extra hardware nor modifications on standard array cells are required;
- ✓ it can implement a complete set of Boolean functions;
- ✗ it is a high latency approach, due to the high number of steps necessary for performing an operation.

3.3 ReRAM LiM

Due to the relatively good sensing and switching times (Table 1.5), and for the prospective of very high integration density (specially with Xbar arrays, subsection 2.3.1), resistive memories appear to be an attention–getting option for LiM implementations.

Almost all the articles related to this topic are focused on crossbar 0T1R ReRAMs, reconfirming the growing attention on this type of array, as anticipated in subsection 2.3.3.

Several logic designs can be realized through memristors. IMPLY, MAD, MAGIC, hybrid–CMOS, CRS, Zhang et al., are just some of the various ReRAM LiM approaches. Among them, the first three are the most known, and are discussed in subsection 3.3.1, subsection 3.3.2 and subsection 3.3.3, respectively.
Finally, in subsection 3.3.4 it is reported a overall confront among all cited LiM approaches, with Table 3.3 and Table 3.4 comparing the number of steps and the area occupation required to perform basic Boolean operations.

3.3.1 ReRAM IMPLY

In IMPLY approach, all logic functions are based on the homonymous imply operation (reported, for convenience, in Table 3.2); the fundamental memristive circuit able to perform it is depicted in Figure 3.5 (a), together with the corresponding adaptation on a ReRAM 0T1R Xbar array (b).

Case	p	q	p ⇒ q
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1

Table 3.2. Imply truth table.

Referring to Figure 3.5, the inputs of the imply gate are the initial states of memristors p and q. Just as reminder, in ReRAMs a high resistance state (HRS) conventionally corresponds to a logic '0', while low resistance state (LRS) to a '1'. The output of the logic implication is stored back into memristor q: due to the input overwriting, IMPLY approach is considered as a destructive operation [73].

In practice, a logic implication is performed by applying proper voltage pulses, V_{COND} and V_{SET}, respectively to p and q memristors. The voltage magnitudes must satisfy the relation $|V_{COND}| < |V_{SET}|$.
If both the memristors are in HRS (p=0, q=0), when applying the voltage

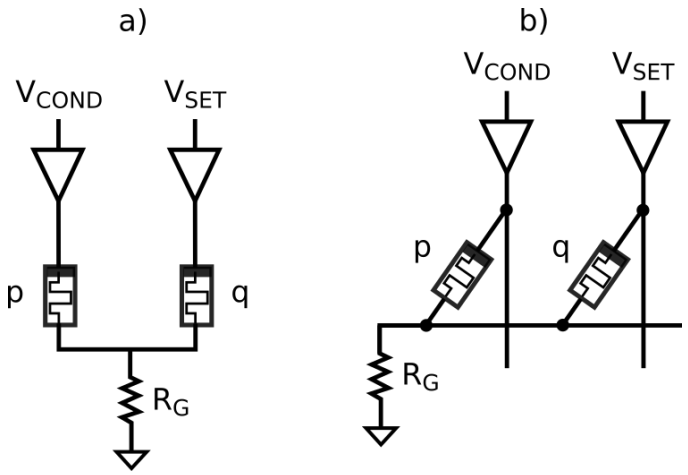


Figure 3.5. IMPLY logic gate (a) and its ReRAM Xbar mapping (b).

pulses the common node remains at very low voltage, and the drop on q is close to V_{SET}; in this way, q undergoes a set process and switches to LRS (q=1).
Instead, when q is already in LRS and p in HRS (p=0, q=1), the voltage ($\sim V_{SET}$) across q leaves it unalterated (q=1).
Finally, when p is in LRS (p=1), the voltage on the load resistor R_G is approximately V_{COND}, and this leads to a voltage drop on q equal to the difference between the two voltages (V_{COND} – V_{SET}). Hence, either if q is in HRS or in LRS (respectively cases 3 and 4 in Table 3.2), the voltage across it is small enough to prevent a change of state [73].

IMPLY logic design is already described in brief in subsection 3.2.4 for STT MRAMs.
What differs from STT case —besides the employed technology— is that, instead of having a dedicated cell to store the information as for the STT example, this time the operation result overwrites one of the input memristors. Moreover, exactly as for STT case, the great limitation of this approach is the fact that each operation is based on the logic imply, increasing a lot the latency: for example, according to [74], a NAND operation requires three imply steps, as shown in Equation 3.3 (a) and reported in Table 3.3.
For more complex operations, like XOR (Equation 3.3, (b)), even 8 steps are required: only four imply operations are necessary, but when $p \Rightarrow q$ is performed, q is rewritten with the implication result. Hence, an intermediate step, that copies the original value of q, is implemented through two

consecutive NOT operations (i.e. two $\Rightarrow 0$ operations). The same is done for p, leading to a total of four additional logic implications. This results in an overall amount of 8 imply steps, as reported in Table 3.3.

$$\begin{aligned} &(a) \quad p \cdot q = (p \Rightarrow (q \Rightarrow 0)) \Rightarrow 0 \\ &(b) \quad p \oplus q = (p \Rightarrow q) \Rightarrow ((q \Rightarrow p) \Rightarrow 0) \end{aligned} \tag{3.3}$$

In conclusion, IMPLY logic in crossbar 0T1R ReRAM share the same characteristics reported in subsection 3.2.4 for STT–MRAM, with the additional constraint related to the overwriting of input memristors.
This section has discussed in details such approach in order to have a clearer view of it and to offer the opportunity for an overall comparison with other ReRAM LiM designs (see subsection 3.3.4).

3.3.2 ReRAM MAD

Memristors–As–Drivers (MAD)[74] has been initially developed with the aim of overcoming some limitations of IMPLY logic, such as the high latency related to serialized steps for performing an operation.
MAD approach is based on a combination of IMPLY approach and threshold logic. An example of circuitry for MAD operation is depicted in Figure 3.6, where an AND gate is implemented through two driver signals (V_{COND}, V_{SET}), three memristors (p, q and s), pull–down resistors (R_G) and a switch [74].

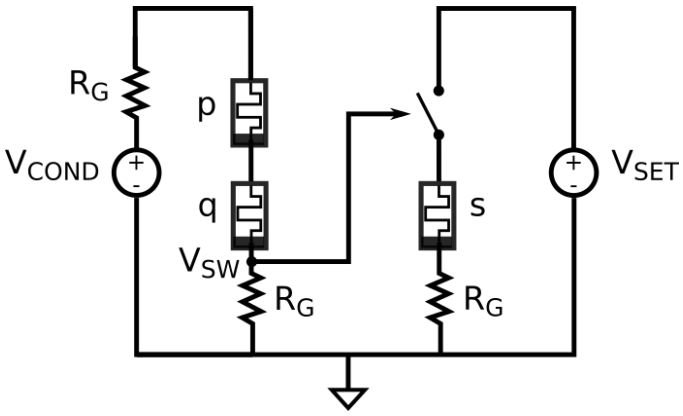


Figure 3.6. ReRAM MAD AND gate.

In MAD logic, two memristors (p, q) act as operands while the third (s) store the result of the operation. For this reason, differently from IMPLY

approach, the logic operation is non–destructive.
The AND operation is implemented by applying a proper sense voltage V_{COND} to the p and q series. Depending on the resistive state of the two memristors, a certain voltage value is sensed at node V_{SW}. Such voltage, if greater than the threshold of the switch linked to memristor s, can make the switch close: consequently, a voltage V_{SET} drops on the ouput memristor. In this way, memristor s undergoes a set process and switches to LRS (logic '1'). This happens only if both the input memristors were originally in LRS, thus making V_{SW} sufficienly high to activate the switch.
In all the other cases, the switch stays opened and memristor s does not change its resistive state. The same gate circuit of Figure 3.6 can also be employed for performing an OR operation, just by changing the threshold level of the switch [74].

As for IMPLY approach, MAD logic can be mapped to a crossbar array.
The mapping, however, is not so direct: in a crossbar array, all rows connects memristors at the same terminal[3]; the opposite terminals, instead, are connected by the array columns.
At this point, it is evident that the standard MAD AND gate of Figure 3.6 must be slightly modified. The modified circuit is shown in Figure 3.7 (b). The red line follows the path of voltage V_{COND}, and it is the only part that differs from Figure 3.7 (a): the two input memristors are not connected in series anymore, and share the same terminals at node V_{SW}. Even if, from a circuital point of view, the two gates in Figure 3.7 are not equivalent, the performed logic operation does not change: in fact the AND output depends only on the resistive value stored in memristor q. This implies that only the switch threshold and the voltage at node V_{SW} determine the switching of q state. Hence, a proper design of the pull–down resistors (R_G) and choice of the switch threshold allow the correct logic function implementation [74].

The final mapping of the AND gate on a Xbar array is depicted in Figure 3.8. Blue and green lines correspond to V_{SW} and V_{SET} signals paths, respectively. When performing the logic operation, both p and q BLs and WLs are selected, with V_{COND} applied to their BLs and with their WLs connected to ground. Then the sensed voltage V_{SW} drives the the column and

[3]It is intended one of the two edges of the memristor; conventionally, memristors are connected to the WLs on the black side (as in Figure 2.6), which marks the memristor polarity (the thick black line in Figure 2.5)

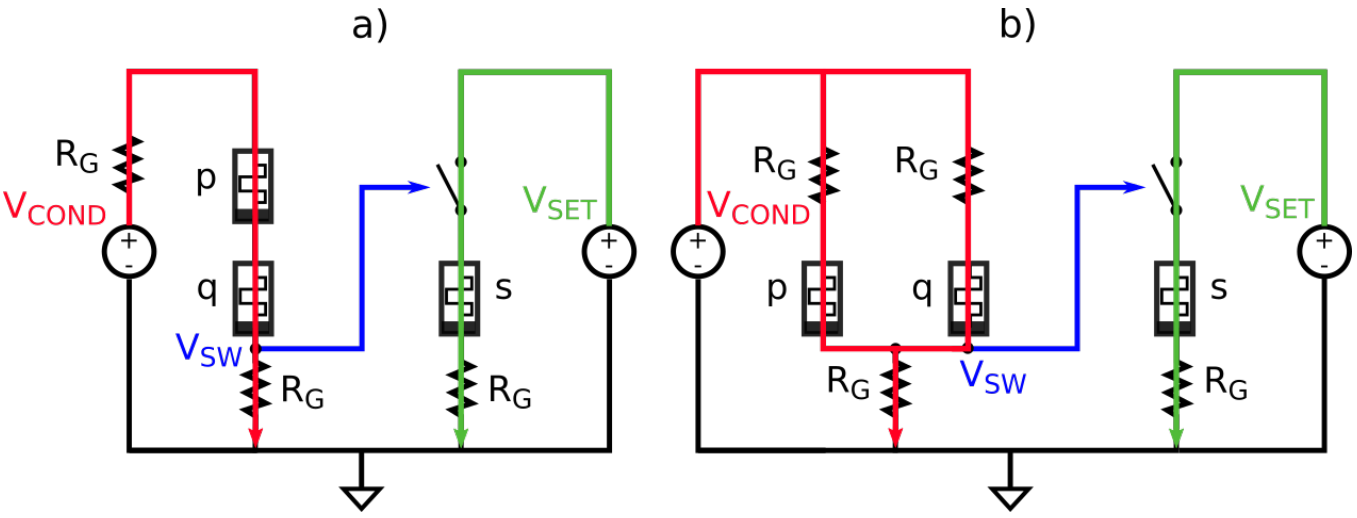


Figure 3.7. V_{COND} (red), V_{SW} (blue) and V_{SET} (green) voltage paths on ReRAM MAD gate (a) and circuit modifications for enabling Xbar mapping (b).

row access transistors of the target memristor (s) that stores the result.
The same crossbar mapping can be employed also for implementing XOR and XNOR functions, with a slight modification: the voltages at the nodes connecting p and q to the respective BLs are the ones that drive the access transistors of the output memristor BL and WL, respectively [74].

As for IMPLY logic, MAD approach is able to implement a complete set of Boolean operations. However, as it can be noted from Table 3.3, the amount of steps required for performing an operation is generally lower than IMPLY one. Also the area occupation in Xbar arrays is smaller for almost each logic function.
Furthermore, thanks to the dedicated output memristor, MAD logic avoids the overwriting of input memristors and it is considered, for this reason, a non–destructive LiM approach. Other considerations on MAD benefits are discussed in subsection 3.3.4.

3.3.3 ReRAM MAGIC

Memristor Aided LoGIC (MAGIC) [74] is maybe the most known approach for memristor–only logic. Differently from IMPLY, but in a similar way than MAD approach, it exploits a dedicated output memristor for storing the result of the logic operation.

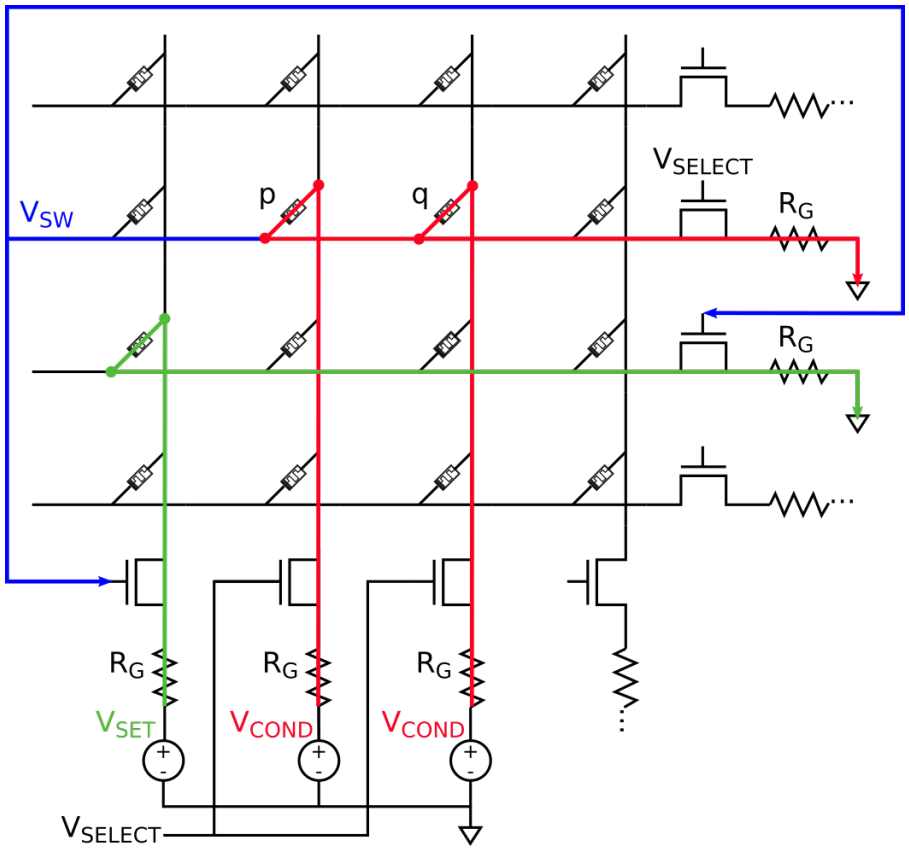


Figure 3.8. ReRAM Xbar mapping of MAD AND gate.

From a computational point of view, MAGIC design can implement almost all Boolean equations: the XOR operation, in fact, cannot be performed with standard MAGIC gates. Hence, sometimes a hybrid–CMOS solution is used to implement such operation [74].

However, this is not an issue for Logic–in–Memory applications. As IMPLY and MAD logic paradigms, also MAGIC can be mapped to a crossbar ReRAM array; through a Xbar, MAGIC can implement a complete Boolean logic set, including XOR operation.
An explanation is necessary. In MAGIC, only the NOR function can be mapped succesfully to a Xbar array. Together with NOT, NOR function is able to implement a complete set of logic operations: it means that a proper combination of only NOT and NOR gates can set up all logic functions, including XOR. Even if this may be seen as a solution, in practice it turns into a huge constraint: since a sequence of NOT and NOR operation is needed

for MAGIC LiM operations, the related latency increases a lot. Even if not quantified, in [74] it is claimed that MAGIC delays in Xbar implementations are much higher than MAD ones (see Table 3.3).
Hence, although MAGIC and MAD logic gates are similar in latency and complexity, their Xbar counterparts are very different; in addition to the increased delays, also the area occupation of MAGIC LiM is higher than MAD one.
In addition, according to [71], MAGIC approach needs several intermediate access cycles that limit LiM performance.

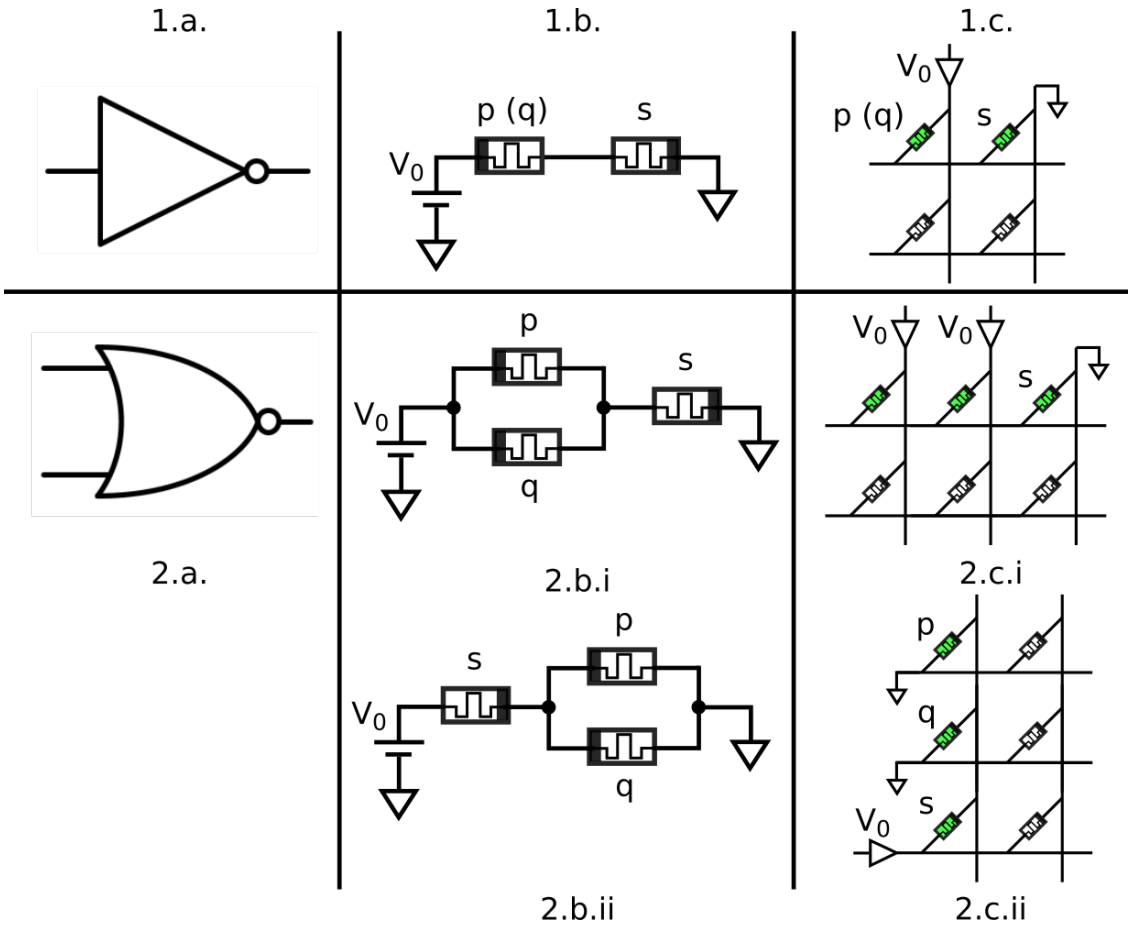


Figure 3.9. NOT (1.) and NOR (2.) symbols (a.), MAGIC gates (b.) and mapping on ReRAM Xbar array (c.); (i) and (ii) indicate the two versions of logic NOR implementation.

The two fundamental MAGIC gates, NOT and NOR, are depicted in Figure 3.9, together with their crossbar mapping.
The NOT gate, for example, exploits a simple voltage divider between the input (p or q) and output (s) memristors. The structure consists in a series of such memristors with opposite polarity (Figure 3.9, 1.b.). The output memristor is initialized to a logic '1'. Then, a voltage V_0 is applied to the series: if the input memristor is in HRS (logic '0'), the voltage drop on it is greater than the output memristor one, which remains under the switching threshold. This prevents memristor s from undergoing reset process, thus keeping it in LRS (logic '1'). Instead, if the input memristor is in LRS, most of the V_0 voltage drops on the output memristor, making it switch its resistive state to a logic '0'.

The NOR gate, on the other hand, is based on a parallel between the input memristors, which is connected to a third memristor that stores the result (Figure 3.9, 2.b.i). The output memristor is initialized to a logic '1'. When a voltage V_0 is applied, the output memristor is crossed by an amount of current that strictly depends on the two memristors resistance. If both the inputs are in HRS ('0'), the current pulse is not sufficient to make the output memristor change its state ('1'). In all the other cases, the current overcomes the threshold of the output memristor, making it switch to an HRS ('0') [75].

An alternative NOR implementation is reported in [76], where the output memristor and the inputs parallel are swapped, as shown in Figure 3.9 2.b.ii; this configuration is adopted for implementing the AND gate of Figure 3.10, explained in the following.
Crossbar mapping of MAGIC NOT, NOR gates are shown in Figure 3.9, 1.c. and 2.c. (i and ii versions), respectively [75].

For the sake of completeness, in Figure 3.10 is reported an example of gate that combines both NOT and NOR functions: the AND gate, in fact, can be implemented with a NOR gate with negated p and q as inputs (Figure 3.10, a.). Consequently, not only the amount of steps for performing the operation is greater, but also the number of required memristors is increased. Two additional memristors, n_1 and n_2, are in fact necessary for storing the intermediate outputs of NOT gates. This leads to a total of 5 memristors needed for AND operation (Figure 3.10, b. and c.). More complex operations, such as XOR and XNOR, require 7 and 8 memristors, respectively. Hence, the

implementation of MAGIC approach needs a significantly large area occupation in Xbar ReRAMs: from Table 3.3 it can be observed that the area required for each logic operation is generally larger than MAD logic one and similar to IMPLY logic one.

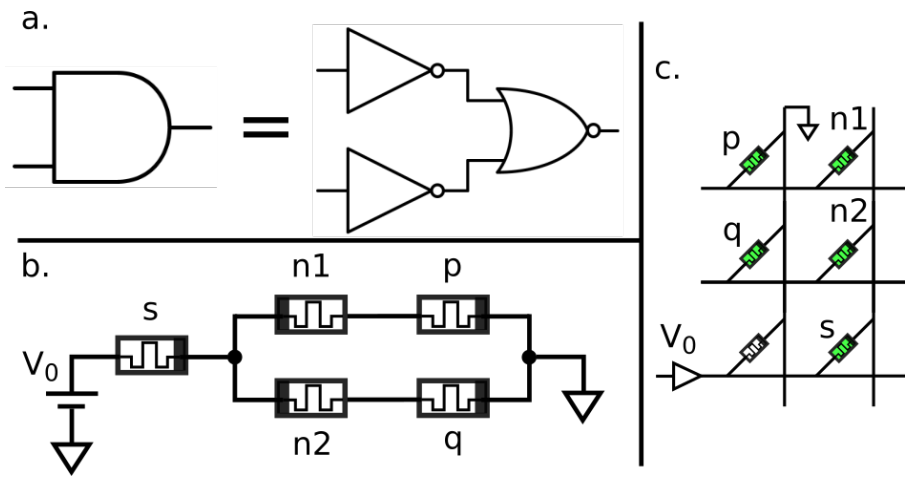


Figure 3.10. MAGIC AND obtained by combining NOT and NOR functions: in symbols (a.), as memristive gate (b.) and crossbar mapping (c.)

The complete set of Boolean functions mapped to crossbar arrays is presented in [76]. In this article it is also proposed a compact Xbar mapping, that allows for a more uniform distribution of the operating memristors thanks to a so–called *transpose array*.
Besides the compact mapping, that is not described here, it may be worthy to spend some words on transpose crossbar arrays. In conventional Xbar arrays, the access to the memory cells is done only from one direction. This is due to the fact that decoders and sensing circuits are usually linked just to one side of the array. For instance, a MAGIC NOR gate is mapped to a row in a conventional Xbar array. On the contrary, transpose arrays exploits extra pheripherals in order to increase the flexibility of the array, allowing access to the cells from both rows and columns. In this sense, a MAGIC NOR gate becomes mappable also to columns [76, 77].
Such array design may be useful in LiM implementations even for technologies different than resistive memories.

3.3.4 ReRAM LiM comparisons and other ReRAM LiM approaches

The aim of this section is to sum up benefits and constraints of the presented ReRAM LiM approaches, and to discuss in a more quantitative way some of their aspects. In particular, latency and area occupations are utilized as comparative parameters. Table 3.3 and Table 3.4 quantify them respectively as amount of steps required to perform the logic operation and as number of involved memristors. The logic designs to which they refer are the —previously described— IMPLY (subsection 3.3.1), MAGIC (subsection 3.3.3), MAD (subsection 3.3.2), together with two new approaches, Complementary Resistive Switch (CRS) and Zhang et al.. These last ones will be only briefly treated, for the reasons explained later in this section [74].

As discussed in subsection 3.3.1, for the IMPLY approach each operation different from IMPLY[4] requires more than one step to be performed; besides the latency, also the number of employed memristors is quite large, since each operation is a combination of logic implications. For instance, an XOR function implemented on an Xbar array needs 8 steps and 7 memristors.
Another constraint of IMPLY logic is that the result of the logic operation is stored back into one of the two input memristors; besides implying the loss of input data (destructive operation), this makes it necessary, for input and output memristors, to lie on the same row of the array.
Nevertheless, IMPLY logic is computationally complete and can be entirely mapped to Xbar arrays [74].

Also the MAD approach is able to perform a complete set of Boolean operations mappable to crossbar ReRAMs. Differently from IMPLY, however, MAD logic is not based on serialized operations, and thus the number of steps required for performing logic is kept low (2 steps for each operation, see Table 3.3).
Furthermore, MAD logic does not need any memristor to store the partial results of intermediate logic steps: the only memristors involved in a logic operation are the memristors storing input and output data of that gate. Such feature allows MAD to occupy a very low area on crossbar arrays: for all Boolean operations, just 3 memristors are necessary (2 input and 1 output), with the exception of NOT gate that requires only 2 memristor (since

[4]Including logic NOT, which is simply an IMPLY 0 operation.

NOT is a single input gate), as shown in Table 3.4). On the contrary, logic approaches such as IMPLY, MAGIC and CRS need several dedicated memristors to store intermediate results. Such memristors are a sort of phantom cells that do not store any useful data, occupy additional array area and contribute to increase the power consumption. For these reasons, MAD logic design is considered as a low–latency and low–area approach for both logic gates and LiM applications.
Moreover, thanks to a dedicated output memristor, MAD logic avoids input overwriting when storing the operation result. In this way, the input memristors maintain their information and can be read or reused again for other logic operations. This also allows in/out memristors to be located on different rows and BLs; however, the activation of multiple rows in the Xbar array may cause the sneak currents to increase. In contrast, IMPLY, Zhang et al. and CRS are input–destructive approaches.

Another comparative factor, significant for memory design, is the usage of multiple voltages. The voltage sources required by MAD logic are three: V_{COND}, V_{SET} and V_{RESET}, as for IMPLY. For Zhang et al. and CRS, instead, the number increases to four. MAGIC NOR only needs one voltage signal, but it is to remind that all MAGIC LiM operations are based on NOR gate, hence the number of voltages may vary depending on the operation to perform.
However, a possible constraint is related to the fact that MAD logic requires different switch threshold voltages, according to the implemented logic operation. This introduces additional complexity on the crossbar array design [74].

Finally, both IMPLY and MAD approaches may exploit the concept of pipelined operations for reducing their latency. According to [74], this is possible thanks to the fact that they exploit a logic based on voltage drivers: the results of operations and driven voltage pulses move through the system, allowing for immediate reuse of computational elements just after the end of their operation.

As anticipated in subsection 3.3.3, MAGIC and MAD gates have similar latency and complexity, with the exception of XOR operation that cannot be implemented through MAGIC. At array level, instead, the differences between the two logic approaches becomes substantial: the only MAGIC function that a crossbar array can map is NOR, while MAD has at its disposal a complete set of Boolean operations. Hence, the only way for which MAGIC can be mapped to a Xbar array is through a combination of NOR

Operations	IMPLY	MAD	MAGIC	CRS	Zhang et al.
p NAND q	2	2	N/A	3	3
p AND q	3	2	N/A	6	2
p NOR q	5	2	1	3	3
p OR q	4	2	N/A	6	2
p XOR q	8	2	N/A	6	N/A
NOT p	1	2	N/A	3	2

Table 3.3. Latency comparison among the presented ReRAM LiM approaches, measured in number of steps. All the values are taken from [74]. As explained in subsection 3.3.4, data in MAGIC column are missing since the reference article does not consider the possibility of combining NOT/NOR gates for deriving all the other Boolean operations.

Operations	IMPLY	MAD	MAGIC	CRS	Zhang et al.
p NAND q	3	3	6	6	2
p AND q	4	3	5	8	2
p NOR q	6	3	3	6	2
p OR q	6	3	4	8	2
p XOR q	7	3	7	8	N/A
NOT p	2	2	2	6	2

Table 3.4. Area occupation comparison among the presented ReRAM LiM approaches, measured in number of involved memristors. All the values are taken from [74], except from the ones in MAGIC column, taken from [76].

and NOT functions, that, together, are able to implement all Boolean operations. The combination of logic gates makes both latency and number of memristors involved for MAGIC LiM to increase.
The results of such considerations are reported in Table 3.3 and Table 3.4; even if not quantified, in [74] it is stated that MAGIC latency in Xbar implementations is much higher than MAD one. Unfortunately, no specific latency values were found in literature: this is also due to the fact that some articles do not even consider the NOT/NOR approach, claiming the impossibility for MAGIC to be mapped to a Xbar. Instead, data regarding the area occupation in crossbar arrays were taken from [76], counting the number of memristors for each NOT/NOR combination. Such number is quite large

also because, differently from MAD, MAGIC approach requires the aforementioned "phantom" memristors to store the result of intermediate steps. Furthermore, MAGIC exploits a dedicated output memristor for performing non–destructive logic operations.

Complementary Resistive Switch (CRS) and Zhang et al. LiM approaches are also mentioned; these are considered only for area and delay comparison with respect to other logic designs, without going into details. This is justifiable since the CRS is based on non–standard 0T1R memory cells, while Zhang et al. is not able to implement a complete set of Boolean functions. Furthermore, this chapter is only a state of the art on LiM approaches, and their discussion would not follow the aims of this thesis.

CRS exploits a cell based on a series of two memristors in opposite polarity configuration. Such approach is able to perform only AND, OR and their negations, hence all other Boolean operations must be derived through their combinations. As a consequence, each logic function is completed in $N+2$ steps, with N equal to the number of NAND and NOR operations. This contributes in increasing the delays, together with the fact that all operands need to be initialized before a logic operation. Also, the area occupation is quite large, being for certain operations more than the double of MAD logic one (see Table 3.4). Furthermore, CRS approach is input destructive since one of the input memristors is overwritten for storing the result of logic operation [74]. However, an advantage of such approach is that a NAND operation with n inputs requires only 3 steps, in contrast with $n+1$ steps necessary in other logic approaches [78].

Zhang et al. LiM approach aims to optimize the number of memristor involved in logic operations. It has been originally introduced in [79]. It is based on the combination of a newly–designed OR gate —mappable to a Xbar array— with IMPLY NOT and AND gates, for implementing all other Boolean functions. This approach is promising in terms of latency and area occupation on the array. The amount of steps needed for performing an operation is the same of MAD one, with the exception of NAND and NOR functions, for which it is required an additional step (Table 3.3). Moreover, for each logic operation only 2 memristors are involved (Table 3.4). However, Zhang et al. approach is not Boolean–complete (XOR/XNOR is not implemented) and it is input–destructive.

This section proposed a comparison, based on few performance parameters, among different types of ReRAM LiM approaches. Hybrid–CMOS logic designs were not treated, in order to entirely focus on emerging memory technologies. The confrontation does not pretend to be complete, since there are a lot of other criteria that may influence the choice of a certain logic approach: for instance, the values of driven voltages, switch thresholds, and pull–down resistances play an important role at design level. However, this may be a good starting point for a future LiM design.

3.4 PCM LiM

3.4.1 PCM logic potential

There are two main features that make Phase Change Memories a good candidate for logic–in–memory applications.

Firstly, the so–called *analog storage capability*, that consists in the skill of assuming a continuum of resistance values, instead of just two resistive states. This is possible thanks to the volumetric phase transition of the material: differently from filament–type ReRAMs, for which HR-to-LR switch —associated to the conductive filament formation— is quite abrupt, in PCMs it is possible to obtain a series of intermediate R values just by the application of proper current pulses with varying amplitude[5], as shown in Figure 3.11 (a). The depicted curve (called *programming curve*) is bidirectional, in the sense that such pulses are able to make the resistance increase or decrease, depending on their current modulation.
The analog storage capability is particularly useful for enabling matrix–vector multiplication. For instance, in $Ax = b$ computation the elements of matrix A are mapped to the conductances of crossbar PCM cells, while vector x elements are mapped to the durations or amplitudes of read voltage pulses applied to PCM WLs. The vector b, containing the results of multiplication, is obtained by sensing the BLs currents of the array [80].

The second PCM property that can be exploited for LiM applications is the

[5]In filament ReRAM this would be —in practice— almost impossible, since the transition is more or less comparable to the one of a switch: the intermediate resistance values could be achieved only with infinitely precise and small current pulses.

so–called *accumulative behaviour*. It consists in the capability of reducing the resistance value in a progressive way, by applying an increasing number of current pulses with the same amplitude. The corresponding resistance curve (reported in Figure 3.11 (b)) is called *accumulation curve*, and it is unidirectional.
Accumulative behaviour enables in–memory computations such as addition, multiplication, division and subtraction, with simultaneous processing and storing of the results. For instance, an sum operation may be performed by encoding the addends values with the corresponding number of voltage pulses, and setting a certain conductance threshold corresponding to the numerical base of the addition; in this way, when the threshold is reached, the number of additional pulses will give the result.
However, accumulative behaviour involves a certain degree of randomness caused by crystallization dynamics. This contributes in increasing the variability of HR ad LR states values, particularly relevant for large arrays [80].

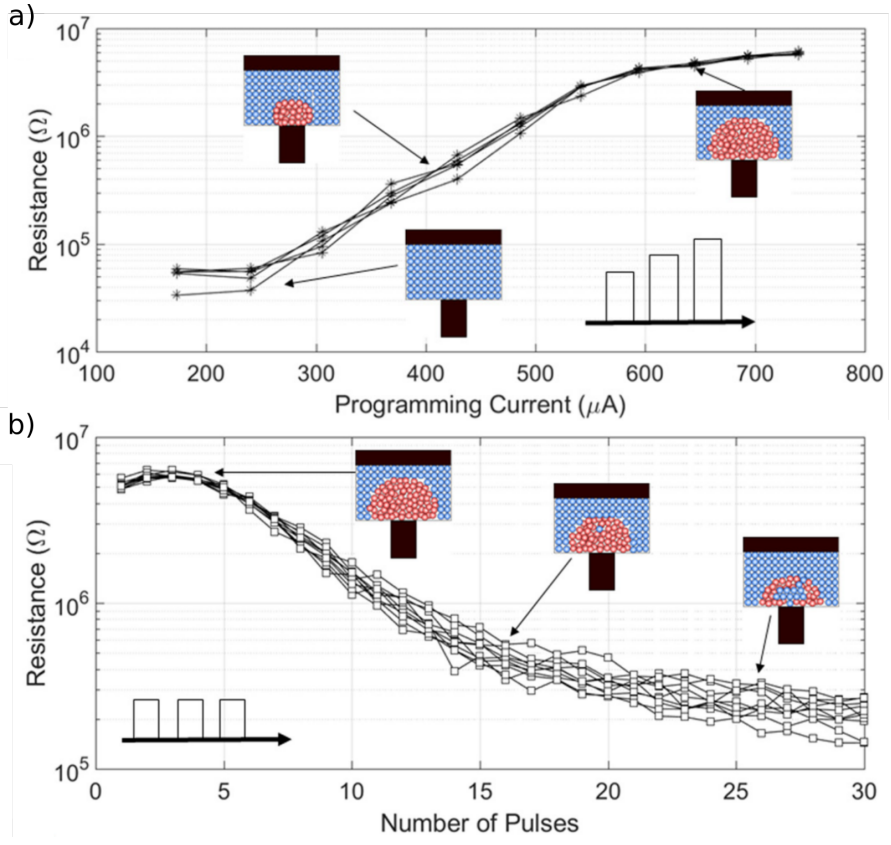


Figure 3.11. Programming curve (a) and accumulation curve (b) [80].

3.4.2 PCM logic

A very useful concept for discussing PCM LiM is *statefulness*. Such term is used to describe all types of logic applications for which the information is encoded as resistive state: in this sense, approaches like IMPLY, MAD and MAGIC —already introduced for ReRAM LiM applications— can be considered stateful designs. In such approaches, the same elements (memristors) are exploited for inputs/outputs storing and for logic computing.
It often happens that stateful logic is considered only a type of memristive logic; however, such concept can be employed also in PCM logic applications, even if not so many related articles can be found in literature. As a consequence, IMPLY, MAD and MAGIC can be in theory used also for PCM LiM.
Nevertheless, PCMs are even able to perform a non–stateful logic, by exploiting their control properties over physical processes (crystallization and melting), as explained in subsection 3.4.1. Non–stateful logic differs from stateful one for the fact that the inputs are provided as external voltage pulses, and only the outputs are stored as resistive states. For this reason, in order to carry out consecutive computations, a resistance–to–voltage conversion is required. However, an advantage of this type of logic is that is it possible to perform different computations with the same PCM cells, just by changing the applied pulses [80].

Another method for implementing PCM logic is through an enhanced sense amplifier, i.e. a sense amplifier able to distinguish very small differences of resistance in PCM cells. In this way, operations like AND/OR can be executed juts by reading, at the same time, multiple memory cells attached to the same SA. As consequence, the number of times the cells are programmed is considerably reduced, and the relatively low endurance of PCMs is no longer an issue [80]. However, the design of the modified SA may be challenging (see subsection 2.4.2).

Examples of stateful logic have already been presented in subsection 3.3.1, subsection 3.3.2 and subsection 3.3.3.
On the other hand, an implementation of non–stateful logic is presented in [81]. In this application, the phase change material is considered to be in an initial crystalline state, and the applied pulses modulate the melting process. Two Boolean operations, NOR and NAND, are presented respectively in Figure 3.12 (a)–(c) and (b)–(d). The logic inputs are mapped as a sequence of

voltage pulses, for which a low amplitude (LO) corresponds to a logic '0', and vice versa an high amplitude (HI) represents a logic '1'.
In this way, the set of pulses LO–LO, LO–HI, HI–LO, HI–HI corresponds to logic inputs '00', '01', '10', '11'. If the pulses are not separated by any time interval (Figure 3.12 (a)), a NOR operation is performed. Instead, if a time interval (of about 100 ns) separates the two inputs (Figure 3.12 (b)), a NAND is computed. This is due to the fact that the two pulses configurations make the ouput PCM cell undergo phase transition in two volumetrically different ways. In other words, different portions of volume of the phase change material are melted, corresponding to different resistive values. Such resistances are then classified as HRS or LRS, depending on if they are above or below a certain threshold (R_{REF}). In PCM logic, conventionally, HRS corresponds to a logic '0', while LRS to a logic '1' [81].

Such logic approach is advantageous for the re–usability of the PCM cells, able to perform different operations just by changing the input pulses. However, it is to consider that a certain amount of time is necessary for applying the logic pulses, for both pulses duration and eventual time separation. As a consequence, the overall latency required for the logic operations increases. For this reason, non–stateful logic is usually considered as a high latency approach.

In conclusion, volumetric switching is a point of force for PCM cells, since it is controllable by properly applied pulses. Such controllability is at the basis of the two fundamental properties of PCMs, i.e. the analog storage capability and the accumulative behaviour, introduced in subsection 3.4.1. These properties are exploited in non–stateful logic approaches, where inputs are given as voltage pulses and the output is the resulting resistance state of the target PCM cell. However, such approach would lead to a time consuming LiM application. Other possible PCM LiM options are stateful logic (IMPLY, MAD, MAGIC discussed for ReRAMs) and enhanced SA approaches (that may make complex the design of the SA).

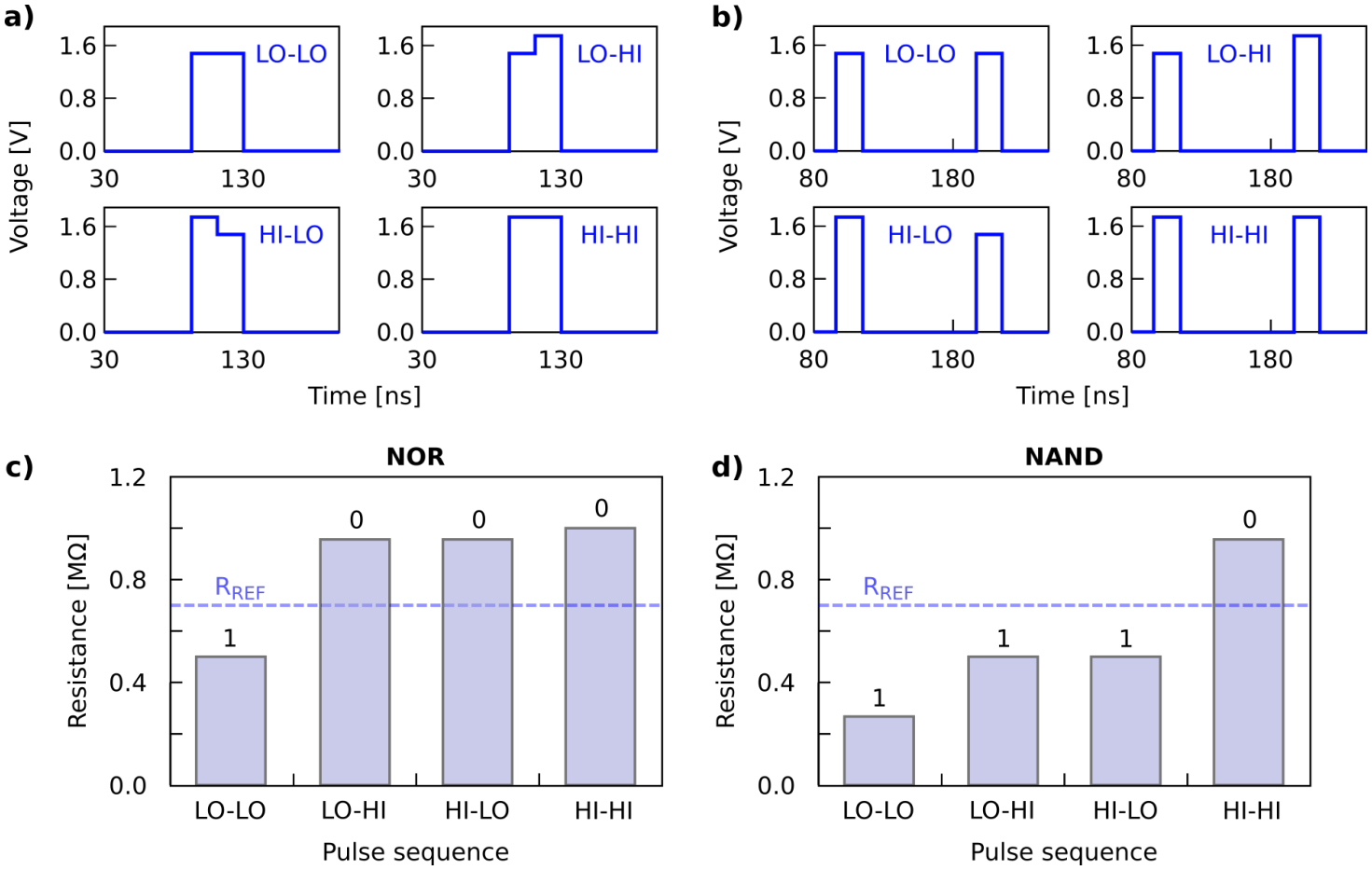


Figure 3.12. Input voltage sequence without (a) and with (b) a time separation for implementing NOR (c) and NAND (d) operations; the resistance states represent the computing outputs [81].

3.5 FeFET LiM

3.5.1 Why FeFET LiM?

Ferroelectic FETs present unique characteristics that distinguish them from all other technologies: since they can be exploited, in practice, both as traditional transistors and for non–volatile information storage, they represent a good candidate for low power and high density LiM implementations.
In addition to the FeFETs structural similarity with respect to MOSFETs, both transistors share the same order of magnitude of I_{ON}/I_{OFF} ratio ($\sim 10^6$). This allows FeFET based memories to adopt simpler (and less expensive) voltage–mode sensing schemes (see Appendix A).
Furthermore, the three–terminals structure of FeFETs offers separate paths for reading and writing: such operations are performed respectively through I_{DS} sensing and V_{GS} application (for switching the ferroelectric polarization), as described in subsection 1.4.4. This could represent an advantage for FeFET LiM, since, differently from two–terminal resistive memories (like STT–MRAM, ReRAM and PCM), no current is necessary in write operations: such feature reduces considerably the writing energy consumption (Table 1.6), since, in resistive memories, write currents are usually large in order to mitigate read disturbs [40].

In comparison with the STT LiM implementation in subsection 3.2.2, FeFET LiM guarantees an high distinguishability between its logic states, mitigating the issues related to multiple WLs assertion. Unfortunately, single–phase computations in FeFET memories are possible only with the use of bipolar voltages, thus reducing the energy efficiency. Furthermore, FeFET LiM often needs multiple references for sensing and computing, increasing design complexity [71].

3.5.2 FeFET logic approaches

Many different approaches to FeFET logic are possible: some of them are based on modifications of standard FeFET memory cells, others delegate logic computations to the sensing circuitry. Various articles even propose the substitution, with FeFETs, of regular MOSFET for implementing logic in SRAMs; 3D implementations are also feasible [65, 82, 83, 87].
In this section, it is discussed a type of FeFET logic that exploits conventional FeFET memory cells (see Figure 1.11 (b)) and that can be integrated

on AND or NAND FeFET arrays (presented in subsection 2.5.1). Then, it is proposed a qualitative classification that lists some examples of other FeFET logic approaches that can be found in literature (of the last few years).

The first example of FeFET logic is described in [82]. Such approach is based on the consideration that, together with the gate voltage V_g, the polarization state of the ferroelectric (FE) layer can be exploited as input of the logic operation. Hence, the first input is a pre–initialized non–volatile FE polarization, that corresponds to a certain transistor threshold level (V_T). Conventionally, high threshold ($V_{T,H}$)/low threshold ($V_{T,L}$) stand for logic '0'/'1', respectively. This can be clearly seen looking at the hysteretic I_d–V_g characteristic of the FeFET (Figure 3.13): the first input (input A) represents one of the two branches of the hysteresis. The second input (input B), instead, corresponds to the gate voltage V_g on the horizontal axis: this time, a low/high voltage level results in a logic '0'/'1'. The output of the logic function is the sensed drain current of the FeFET.
In particular, in Figure 3.13 (a) it is depicted the hysteresis loop corresponding to a (N)OR operation: I_d is maintained high (logic '1' at output) if at least one of the two inputs are logic '1', i.e. when the threshold voltage is low and/or when the V_g level is high. Otherwise, the FeFET current is heavily reduced (logic '0' at output, I_{d1}). This behaviour maps an OR function. If, instead of I_d, it is measured the corresponding output voltage, an inverted signal is obtained: hence, a NOR operation can be obtained starting from the same inputs and the same FeFET by sensing the output voltage instead of the current. However, in order to measure the voltage of the gate, a pull–up element (a resistor or a pMOS in resistive region) must be connected in series to the FeFET, as shown in Figure 3.14.
In Figure 3.13 (b) it is depicted the FeFET hysteresis loop for implementing a (N)AND logic function. The loop is obtained simply by horizontally shifting the one in Figure 3.13 (a) with the application a proper source voltage V_s or back bias voltage V_{bb}. When applying V_s>0 or $|V_{bb}|$>0, the FeFET conducts a different amount of current I_d for the same[6] voltage V_g applied to its gate. In this configuration, the drain current results high (logic '1', I_{d4}) only if both the logic inputs are '1', i.e. if the threshold voltage is low and the gate voltage is high. In all the other cases, a low I_d is obtained (I_{d1}, I_{d2}, I_{d3}). In this way, a logic AND is implemented when sensing the drain current, while

[6] i.e. of the (N)OR case

a NAND function is obtained by observing the output voltage.
The truth table corresponding to (N)OR and (N)AND configurations is reported in Table 3.5, together with the involved current and voltage signals.

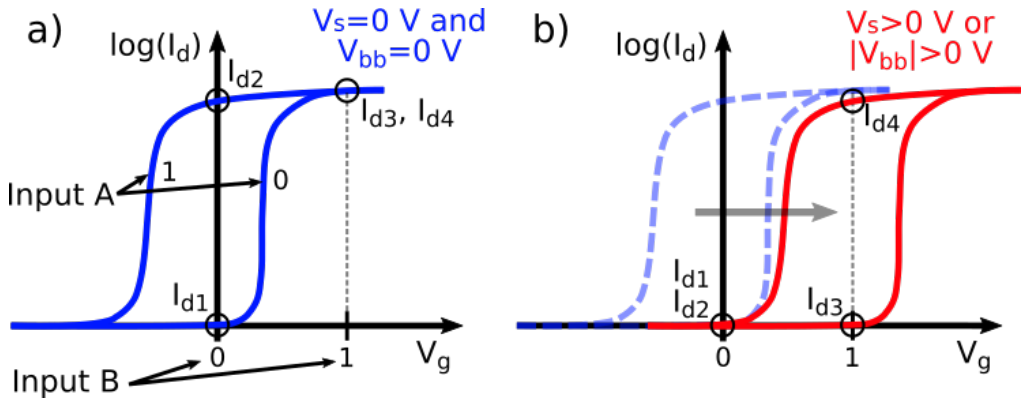


Figure 3.13. Hysteresis loop related to FeFET logic (N)OR (a) and (N)AND (b) operations.

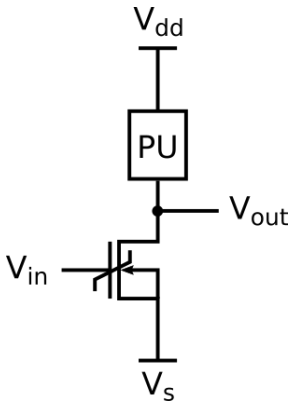


Figure 3.14. Electrically reconfigurable FeFET logic NAND/NOR gate. V_{in} corresponds to the gate voltage (V_g) of the n–type FeFET, while V_s represents the source bias and PU the pull–up device.

The described FeFET logic approach offers a great computational potential: with just one single FeFET, together with a pull–up device, up to four Boolean operations can be implemented (OR, NOR, AND, NAND). This is due to the FeFET intrinsic reconfigurability that such design exploits: just by applying a V_s or V_{bb} voltage, the I_d–V_g FeFET characteristic is shifted, allowing to increase the number of logic functions that can be implemented [82].

In	FE	(N)OR		(N)AND	
		Out	Out	Out	Out
V_g		I_d	V_{out}	I_d	V_{out}
0	0	0	1	0	1
0	1	1	0	0	1
1	0	1	0	0	1
1	1	1	0	1	0

Table 3.5. Truth table of (N)OR and (N)AND FeFET operations, together with the involved signals.

The logic potential is even higher when connecting two FeFETs in parallel or in series. The connection of multiple FeFETs introduces an additional level of computing: for example, when two parallel FeFETs in AND configuration are connected in parallel, an OR operation is performed between them. The resulting function is $(A \cdot B) + (C \cdot D)$, with (A, C) internal FeFETs polarization states and (B, D) externally applied V_g voltages. Then, if (C, D) are substituted by $(\overline{A}, \overline{B})$ respectively, an XNOR operation is performed between (A, B) inputs (Figure 3.15, (a)).
On the other hand, a XOR function can be implemented by connecting in series two OR FeFETs[7], such that $(A+B)\cdot(C+D)$, and then by substituing (C, D) with $(\overline{A}, \overline{B})$ (Figure 3.15, (b)). In this way, multiple input AND and OR gates simply consists in parallel AND FeFETs and in series OR FeFETs, respectively. Such results are obtained by measuring the output current I_{out}, while negated functions are implemented by observing the output voltage V_{out}.
The described FeFET logic gates can be mapped to AND and NAND memory arrays (depicted in subsection 2.5.1). For structural reasons, parallel connected FeFETs are better arranged to AND arrays, while NAND arrays are preferable for mapping in series FeFETs [83].

Also 3D FeFET memories are feasible, as anticipated in subsection 2.5.1. In [65] it is discussed a 3D NAND FeFET array for high density storage required by deep neural network applications. The structure of a single memory

[7] i.e. two FeFET, each implementing an OR function.

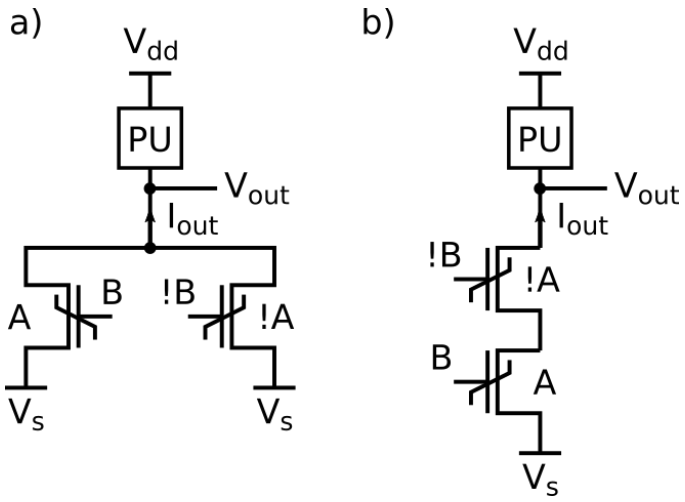


Figure 3.15. FeFET logic XNOR (a) and XOR (b) functions are obtained by measuring the output current I_{out}; negated operations —XOR (a) and XNOR (b)— are implemented by measuring the output voltage V_{out}.

block can be described considering a reference xyz space: multiple NAND arrays are aligned in parallel to yz planes, with all FeFETs oriented in z direction[8]. All the bitlines are parallel to y direction, while wordlines are aligned to x direction, and all the gates of FeFETs in the same xy plane share the same WL. String and ground select transistors are located at the structure top and bottom xy planes, respectively: all bottom select transistors gates share a common ground select line (GSL), while different string select lines (SSL) links the gates of the homonymous transistors. Both GSLs and SSLs are aligned to x direction. The overall 3D array, for a single memory block, is depicted in Figure 3.16. The BLs of one block are connected to the BLs of another block, respecting their enumeration (BL0 of block1 to BL0 of block2, etc.). WLs are instead independent among different blocks.
Such structure is described here since it is very useful for a particular in–memory computation: the *vector–matrix multiplication* (VMM). In order to implement the VMM, a voltage vector is given in input to the WLs associated to FeFETs sharing the selected xy layer, in multiple memory blocks. The weight matrix simply corresponds to the FeFETs channel conductances. The resulting output vector is obtained by sensing the BLs currents from the various blocks [65].

[8] i.e. with S and D of a single FeFET sharing the same line along x direction.

A VMM performed by a 3D NAND FeFET array is also described in [84]. However, as discussed in the following, such operation can also be implemented in 2D memories, with proper modifications of the conventional FeFET cell.

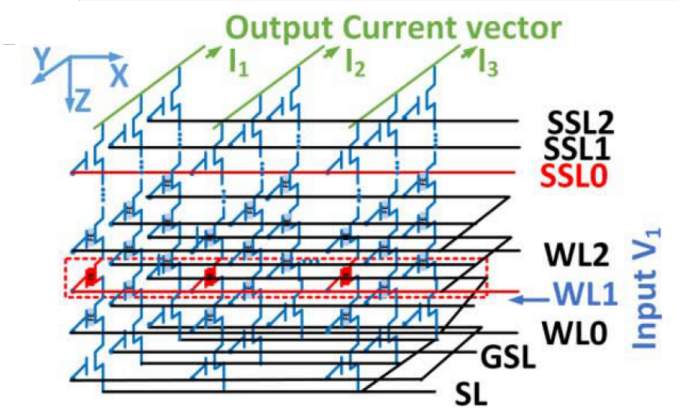


Figure 3.16. 3D FeFET NAND array for VMM operation [65].

In some cases, logic functions can be introduced in a FeFET–based memory through the sensing circuitry. For example, in [40] it is described a modified latched sense amplifier able to perform (N)AND, (N)OR, X(N)OR, NOT and ADD operations. The designed SA is capable of working both in current mode and in voltage mode, in order to implement more efficient FeFET LiM functions.

However, a consistent number of FeFET LiM implementations is based on modification of the standard 1FeFET cell.
For instance, in [85] two different design of ternary content addressable memory (TCAM) cells are proposed. Such memory is able to carry out parallel searches among tables of saved data for finding a match with a certain given data. For such capability, TCAMs are considered as a type of LiM circuit. However, both the presented TCAM designs completely revolutionize the structure of the conventional FeFET memory cell: up to 5 regular MOSFETs and 2 FeFETs assemble a cell, and one of the two designs also requires an additional voltage rail for negative supply.
An interesting FeFET LiM implementation is described in [86]. The designed 2D FeFET array, with properly modified memory cell, is able to perform

a VMM[9] in the signed ternary regime: the costumized FeFET cells, called ternary compute–enabled memory cells (TeC–Cells), allow for ternary weight storing and massively parallel signed VMM between the stored weights and ternary inputs, through the assertion of multiple WLs. Such design can be employed in ternary Deep Neural Network (DNN) applications.

Another possible approach to FeFET LiM consists not only in modifying the cell structure, but also the FeFET device itself: in [71] it is proposed a 4T–R[10] memory cell that includes two access transistors and two cross–coupled reconfigurable FeFETs (R–FEFETs). The cross-coupled configuration —not easily implemented with conventional FeFETs— is possible for R–FeFETs due to their capability of switching between volatile and non–volatile modes. Such features is enabled through a modification of the ferroelectic layers in the gate stack of standard FeFETs.
Furthermore, the presented memory design allows for differential access to the cell, which improves the sense margin during reading; in addition, AND and NOR operations between two cells are achievable just asserting two WLs and sensing the cell outputs. All the other Boolean operations can be performed through a proper compute module, such as the one designed in the reference article.

Lastly, it must be noticed that, sometimes, it is considered a type of FeFET logic also the case in which FeFETs substitute regular MOSFETs in standard CMOS logic gates. While the logic circuits remain the same, peculiar FeFET properties (e.g. non–volatile information storage) are introduced in the design. With this approach, in [87] logic NOT and NAND functions are implemented for building an all–FeFETs BL and block selector for a ferroelectric NAND memory array.

[9] already mentioned for 3D FeFET.

[10] Acronym for 4 Transistors Reconfigurable.

Part II

SOT STT LiM development

Chapter 4

SOT STT Memory implementation

4.1 Technology choice

Among all the in–memory logic approaches presented in chapter 3, a magnetoresistive–based one has been selected for a more practical application. In this chapter, the following steps will be described:

1. the research of a Verilog–A model for simulate in Cadence Virtuoso environment (section 4.2);
2. the development of a memory array and writing/peripherals without computational features (section 4.3);
3. the discussion of simulations results of such memory design (section 4.4).

4.2 The model

Interesting Verilog–A models, developed by Spintronics Interdisciplinary Center (SIC) of Beihang University, are presented in [88].

Initially, a traditional Spin Transfer Torque Perpendicular Magnetic Anisotropy (STT PMA) MTJ model has been investigated [89]: after some simulations, however, it has been noticed a discontinuity in the current across the MTJ, even when the resistive state is stable; this peculiarity is predicted by the model manual, but it is present even if no resistance or switching duration variations are introduced; furthermore, relatively long ($\geq 10\,\text{ns}$) writing

pulses are required for achieving a complete state switching.

For these reasons, but also for the attractive perspective offered by a Spin Orbit Torque (SOT) switching (already presented in subsection 1.2.4), another model —found in [90]— has been selected.
Such model exploits the combined action of STT and SOT effects to enhance MTJ switching speed, as qualitatively displayed in Figure 4.1: for identical STT input currents amplitudes, the SOT STT–assisted switching (blue curve) is much faster than the STT–only switching (purple curve). Furthermore, the required STT pulse duration is, in the first case, much shorter (2 ns) than the second case one (5.5 ns). The complete MTJ switching cannot be achieved by SOT–only case (green curve), since the SOT current only plays an assisting role in the perpendicular MTJ STT switching.

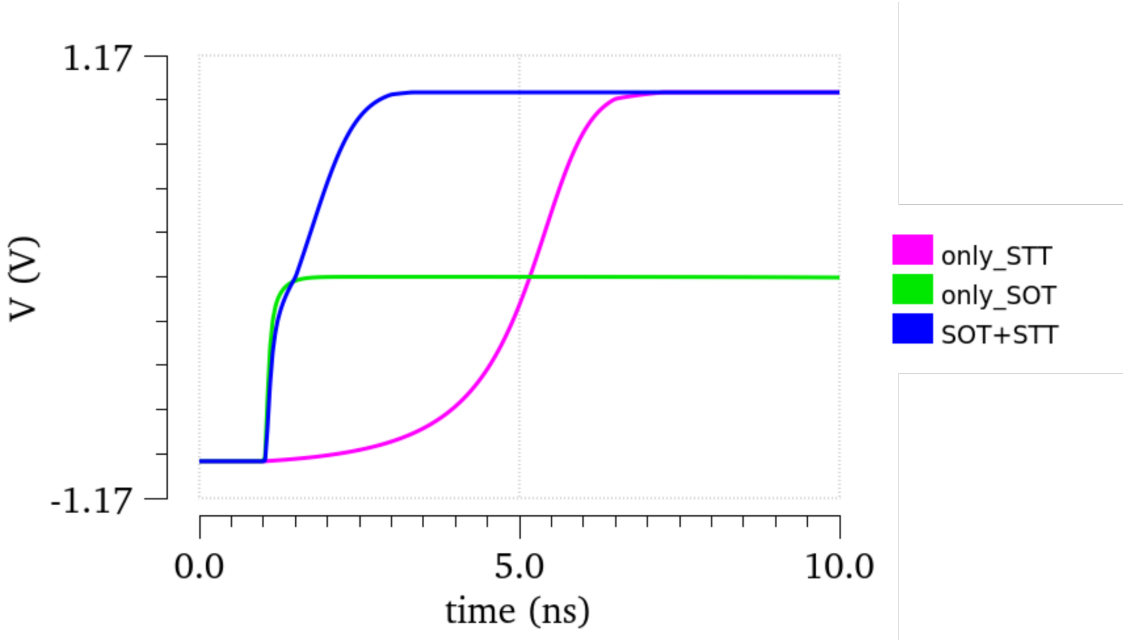


Figure 4.1. Practical simulations performed in Cadence Virtuoso by appending the Tmz (z–component of magnetization of the MTJ free layer) curve of three different simulations (with only SOT, only STT, both STT and SOT inputs) done on the same MTJ.

A SOT–MRAM based on PMA MTJs faces some difficulties in reaching a stable perpendicular magnetization: for this reason, it usually exploits the contribution of a magnetic field for achieving the complete switching. However, the field assistance to SOT in–plane current (for MTJ switching)

increases the device complexity and lowers the thermal stability of the MTJ [91].
Another way for breaking the magnetization asymmetry is the one proposed by [90], where the STT current substitutes the role of the magnetic field in enhancing SOT switching. This allows to inherit the SOT advantage of low power consumption[1] and, at the same time, to speed up the STT–only MTJ switching. The model takes into account only the Spin Hall Effect (SHE) contribution to SOT current since "the Rashba Effect"[2] —as declared— "is still subjected to debate" [91].

The original SOT STT PMA MTJ model [90] consists in a Verilog–A script associated to a symbol, testable into Cadence Virtuoso environment. Three versions of the model are provided: a first standard one, a second one where some deviation factors are introduced, and a third one where a testbench is set up to simulate some cases of study.

The basic 3–terminals SOT STT element, equal for all the three model versions (except for some Verilog-A parameters), is reported in Figure 4.2. It includes a PMA MgO–barrier MTJ over a a β–W Heavy Metal (HM) strip, whose default sections are circular (that corresponds to the Verilog-A parameter *shape* = 2, settable through the *Component Description Format* (*cdf*) in Cadence) and rectangular, respectively. The MTJ section can be also set to rectangular or elliptical by changing *shape* to 0 or 1, respectively. All the physical and geometrical parameters adopted for simulations are reported in the model manual [90], while the system of equations describing the model (modified Landau–Lifshitz–Gilbert (LLG) equations) behaviour is discussed in [91].

The SOT STT device can be seen as a trio of resistances: the variable one represents the MTJ, while the other two corresponds to the HM layer and are set to 0.5 kΩ. The R_{MTJ} —in the two logic states— is computed through the Cadence calculator by considering the current flowing through it and the voltage between T1 and TC points: in this way, the resistance of both parallel and anti–parallel magnetization states can be measured.
T1, T2 and T3 are the real pins corresponding to the three terminals of the

[1] With respect to STT one.

[2] Spin Hall and Rashba effects were briefly introduced in subsection 1.2.4.

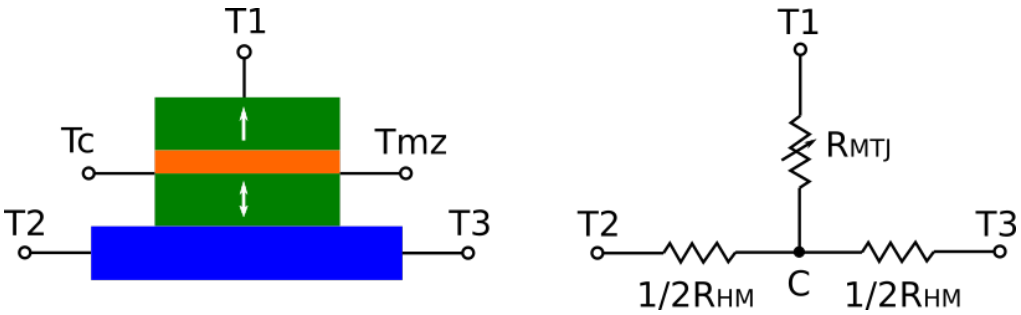


Figure 4.2. SOT STT MTJ model representation (left) together with its equivalent resistive circuit (right) [90].

device, while TC and Tmz are virtual pins: the first one is used to detect the voltage in C point, while the second one indicates the perpendicular (along z) magnetization state. A Tmz $= -1$ V matches a parallel configuration between the free and fixed MTJ magnetizations (LRS, logic '0'), while the anti–parallel states corresponds to a Tmz $= 1$ V (HRS, logic '1').

The provided testbench schematic is shown in Figure 4.3: the SOT and STT currents are supplied by ideal current generators, while T3 is connected to a dc voltage set to 0 V in order to avoid an eventual model bug when connecting it directly to ground (reported also by the manual [90]). Four preset states are associated to the testbench: *asymmetry*, *field–like*, *only–SHE* and *only–STT*. Unfortunately, the states of the testbench were locked and not directly usable into Cadence, hence the parameter values for each states have been set in new states by observing the provided (but not usable) states files.

The asymmetry configuration takes into account the different propensity of STT current to write '0' or '1', by regulating the cfd parameter *asy* (when set to 1 no asymmetry is introduced, otherwise it is considered for 1.05÷1.1) and *SAS* (when 1 it takes into account the *asy* factor, if 0 it does not).
A field–like simulation is instead set up by modifying the *fac_FL* cfd parameter that allows to decide if a field–assisted torque is considered or not.
The parameter *Bext* offers the possibility to introduce an external magnetic field and regulate its size (if *Bext* is equal to 0, a purely electrical simulation takes place).

For the sake of completeness, variations of the oxide (*tox*) and free layer (*tsl*)

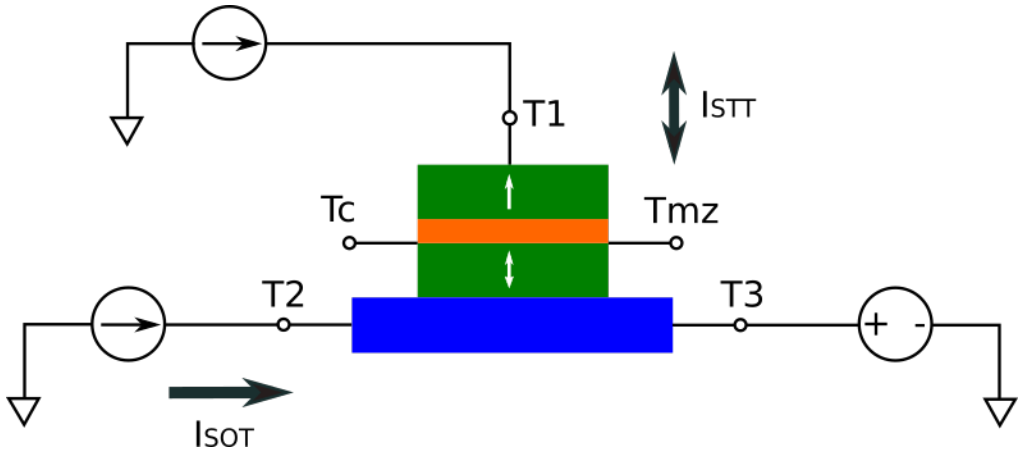


Figure 4.3. Pre–set testbench provided by [90] for simulating the SOT STT MTJ model. The voltage generator on T3 is set to vdc = 0 and it is present only for preventing a model bug discussed in the model manual.

thicknesses and of the TMR can be introduced by setting the *STDS* parameter to 1 (uniform distribution) or 2 (Gaussian distribution) and by regulating the respective percentage variations (*DEV_tox*, *DEV_tsl*, *DEV_TMR)*.

All these parameters are discussed in the model manual [90], and the various states were tested on the Cadence Virtuoso schematic to verify the device functioning. In particular, *only–STT* and *only–SHE* configurations (without asymmetry and field assistance) confirmed the trend in Figure 4.1.

However, the most interesting configuration —i.e. the SOT STT one— is not provided among the testbench input states. The article related to the model [91] is useful in order to have an idea of the pulses used for this case. In Figure 4.4 it is shown the SOT STT MTJ behaviour when applying SOT and STT pulses of different duration: it can be noticed that the switching speed is directly dependent on the SOT pulse duration, hence such pulse must be shortened as much as possible. The article proposes a STT duration of 4 ns and a SOT one of 0.5 ns.

On the other hand, the amplitudes of SOT and STT pulses adopted by the article are dependent on the setting of the geometric parameters (i.e. the sizes of the MTJ and the HM layer), that slightly larger than the ones used in the model manual; it has to be considered that the geometric setting used in this work is the same of the one reported in the manual [90].

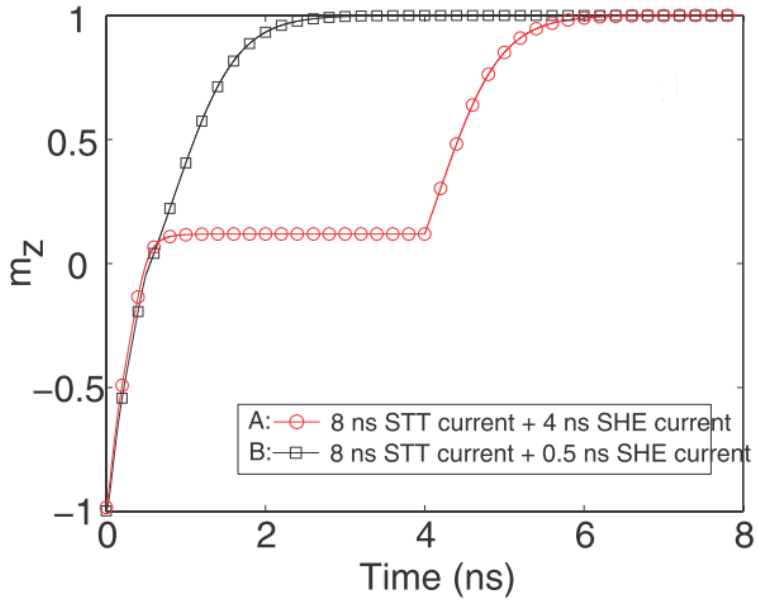


Figure 4.4. Magnetization along z for different duration of SOT STT pulses [91].

Nonetheless, the pulses durations adopted by the manual are different than the ones of the article (only SOT is specified in the manual, 0.75 ns long); hence, a re–computation of the working point has been performed not only to replicate the right functioning of the device, but also to try to optimize the pulses duration with respect to the switching delay and the power consumption.

With a series of parametric analyses, a good compromise is found by adopting $1.5\,\mathrm{MA\,cm^{-2}} \cdot 2\,\mathrm{ns}$ / $50\,\mathrm{MA\,cm^{-2}} \cdot 0.5\,\mathrm{ns}$ for STT / SOT pulses: a SOT current density higher in modulus with respect to the one of the article [91] and shorter than the one of the manual, but that allows the STT pulse to be reduced to 2 ns (instead of 4 ns) with the same amplitude of the one proposed in the paper. As it can be seen from Figure 4.5, it is evident that the switching duration is comparable to both the article and the manual one, with shorter STT pulse. The STT current polarity determines the direction of the Tmz switching, and, hence, the resistive state of the MTJ. The SOT current, instead, plays an assisting role in the switching process regardless of its direction.

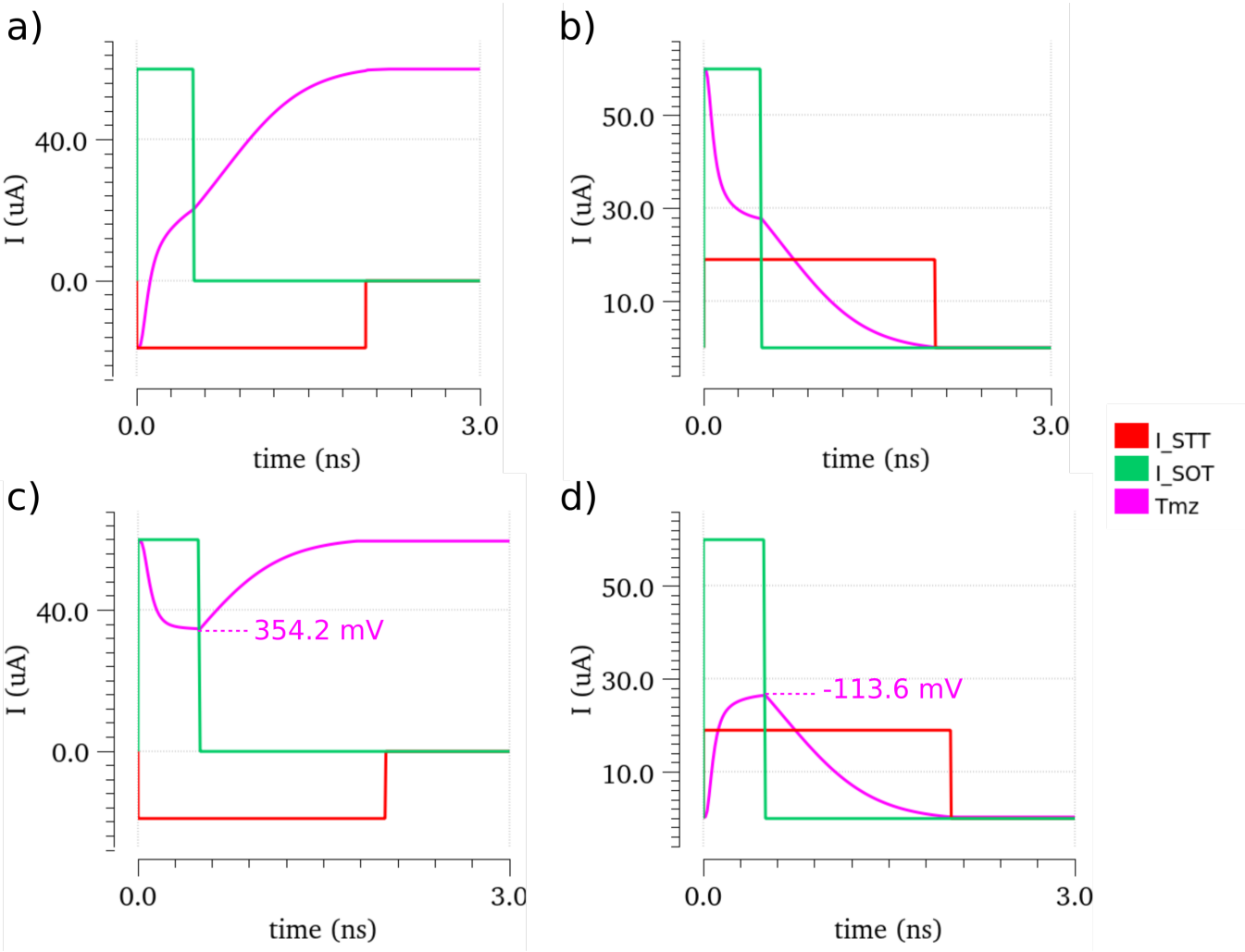


Figure 4.5. Tmz curves and input current pulses for write '0' (a, c) and write '1' operation (b, d). The ordinate axis is related to the current pulses, while the Tmz voltage axis (hidden in the figure) goes from -1 V (logic '1') to 1 V (logic '0'). The initial MTJ logic state (*PAP*) is equal to 1 for a),d) and to 0 for b),c). The c) and d) plots display the slight Tmz oscillation that takes place when a writing operation is performed on a MTJ that already has the target logic value (i.e. a write '0'/'1' on logic '0'/'1').

However, with the introduction of the real drivers (subsection 4.3.3) and the logic implementation (chapter 5) it has been is preferred to provide voltage inputs instead of current ones, adopting the same pulses duration and an equivalent working point.

In this work it is adopted the second version of the MTJ model (PMA_SOT_STT_DEV), but no deviation of the parameters is introduced during the simulations. However, it is left the possibility to introduce parameter variations at cell and array level, as discussed in subsection 4.3.5.

4.3 SOT STT memory design

In this section it is described the design and testing of a SOT STT MTJ–based memory, without logic functions implemented in the array. The design includes the basic cell (subsection 4.3.1), organized into arrays of different sizes (8x8, 16x16, 32x32, subsection 4.3.4), the sensing interface (Sense Amplifier and read driver, subsection 4.3.2) and the write drivers (subsection 4.3.3). The complete architecture is tested for read and write operations, with delays and power consumption results discussed in section 4.4.

All the simulations are performed in Cadence Virtuoso employing the ST FD–SOI 28nm technological library.

4.3.1 The PMA SOT STT cell

The adopted cell design is reported in Figure 4.6: it is inspired by SOT MRAM cell proposed in [92], although the working principle is different. The cell employs a couple of nMOS transistors for accessing two of the three MTJ terminals. The n–MOSFETs aspect ratio is as small as possible to maximise the switching speed, even at the cost of an higher resistance in ON state, and for limiting the cell area.

In a standard SOT cell, the RBL (Read Bit Line)–to–SL (Source Line) and the WBL (Write Bit Line)–to–SL represent the two separate signal paths for reading and writing, respectively. This is also the reason for which some SOT cell designs [93, 94] adopt separate WWL (Write Word Line) and RWL (Read Word Line) instead of an unique WL to activate the access transistors. In SOT STT case, instead, both RBL–to–SL and the WBL–to–SL paths are

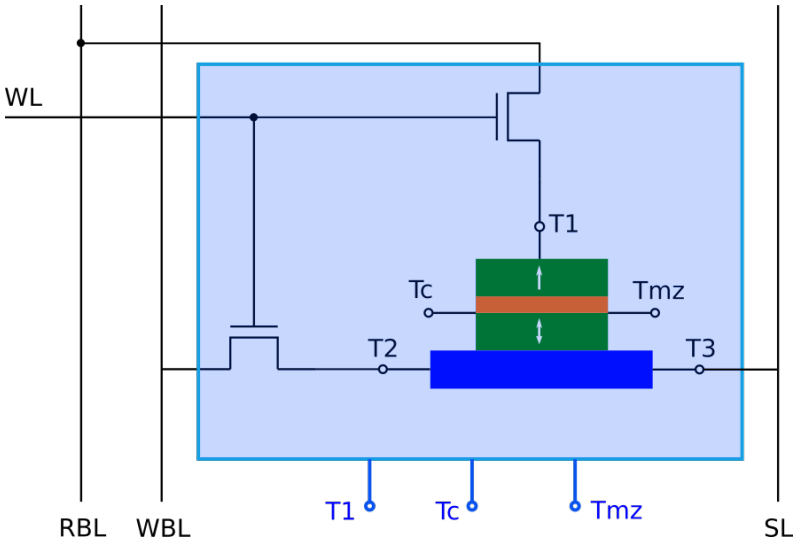


Figure 4.6. Proposed SOT STT MTJ cell.

employed for performing a write operation, and correspond to the STT and SOT currents paths, respectively. On the other hand, only RBL–to–SL path is necessary for a reading operation. Both operations are summarized in Table 4.1.

Operations	**Write "1" ("0")**	**Read**
WL	V_{dd}	V_{dd}
RBL	V_{STT}>0 V (<0 V)	I_{sense}
WBL	V_{SOT}>0	floating
SL	0	0

Table 4.1. Overview of the voltages and currents involved in writing and reading operations.

When performing a write operation, proper voltages are applied to both RBL and WBL. As anticipated in section 4.2, it is the polarity of the STT pulse (on RBL) that determines the resulting logic state of the cell. In standard SOT cells, the negative writing voltage is provided by applying the corresponding positive voltage to the SL while grounding the opposite line: this is possible since only one current is employed for writing, hence the resistive path between the lines is symmetrical.
In SOT–STT case, instead, both the access transistors are turned on, and

both RBL and WBL are supplied: since the resistive path is no more symmetrical, the negative voltage has to be provided by a dedicated negative supply rail and adapted by properly designed write drivers (subsection 4.3.3).
From this point on, in almost all memory and LiM operations, the shared SL will be mainly used only for giving a ground reference.

4.3.2 The sensing interface

The sensing interface includes both Sense Amplifier (SA) and read driver design.

As anticipated in Table 4.1, a reading operation in SOT STT cell consists in activating the WL while providing a proper sense current (I_{sense} or I_{ref}) that flows from the RBL to the (grounded) SL of the selected cell.
The WBL is left floating during such operation, as it can be seen from the WBL driver description in subsection 4.3.3; this is due to the fact that, differently from SOT cells like the ones in [93, 94], in the proposed design there is not a dedicated WWL signal that makes the WBL access transistor turn off during reading. Hence, the direct grounding of WBL would add in parallel to the $1/2{\cdot}R_{HM}$ resistance between Tc and T3 terminals (Figure 4.2) the R_n + $1/2{\cdot}R_{HM}$ series[3] of the WBL–to–Tc path.
The WBL grounding, however, would not completely preclude the read operation, but would decrease the already small sensed voltage of the cell (V_{sense}). This is also one of the reasons for which the simulation results discussed in subsection 4.4.3 are directly referring to the use of "real" drivers, and not just voltage generators: in the second case, in fact —during both writing (after the 0.5 ns pulse) and reading— the WBL would be put to ground by the turned off voltage generator, and not disconnected from the cell.

The overall read interface is represented in Figure 4.7. For convenience, it is depicted only the selected cell, instead of the entire column to which the SA is attached.
As previously discussed, a sensing current I_{sense} is provided to the cell through the RBL: this current generates a voltage V_{sense} on the RBL that is proportional to the resistance of the MTJ (i.e. its logic value) and that can be sensed by the SA. The second input to the sense amplifier is the voltage V_{ref} produced by the same I_{sense} current flowing on a fixed resistance, that

[3] R_n is the ON resistance of the WBL nMOS access transistor.

represents the reference cell.
With respect to the standard STT reference cell discussed in Figure 2.3, a parallel of SOT STT MTJ is not directly feasible, unless by leaving floating one of the two heavy metal resistances; furthermore, a simple fixed resistance involves much smaller dimensions. The resistance is sized such that the voltage on it falls approximately in the middle of the interval between the high and low sensing voltages on RBL, i.e. $V_{ref} \approx (V_{sense,H} + V_{sense,L})/2$. In this way, the SA provides a logic '1' only when $V_{sense} > V_{ref}$, otherwise a logic '0', as it can be seen in Figure 4.8.
The read driver is the circuitry that provides the proper I_{sense} current to both the reference cell and the selected SOT STT cell.

Different circuits for the SA were tested. In particular, the proposed SOT STT memory requires a voltage–mode SA (Appendix A) capable of distinguish between low voltages (few hundreds of millivolts). The fastest voltage sensing is achieved by latch–based SAs, as the VLSA in Figure A.3 which is often employed in SRAM sensing. However, in this SA scheme the inputs are provided at the gates of a couple of nMOS: hence, due to the small amplitudes of the voltages (V_{sense}, V_{ref}) involved in reading operation, the discharging time of such transistors may be quite long (several hundreds of picoseconds) and heavily affects the sensing delay. For this reason, it is preferred a SA circuit which provides the input voltages directly to the latch, through the source–drain path of two couples of access nMOS–pMOS (M5–M6, M7–M8 in Figure 4.9). The SA circuit is taken from [95]: however, the original scheme provided a Sense Enable (SE) signal as supply voltage of the latch, while in this work it is preferred to provide a V_{dd} supply voltage through a pull–up pMOS enabled by the negated SE.

This SA circuit is particularly suited for working with very low input voltages and a small voltage difference between these. The design proposed in the article [95] is based on a 65 nm CMOS process, hence all transistor dimensions are adapted for a 28 nm process. The final sizing is reported in Figure 4.9: here, the access transistors (M5,M6,M7,M8) are equally–sized in order to make nMOS transistors stronger than pMOS ones, since the input voltages are closer to ground voltage than supply one.

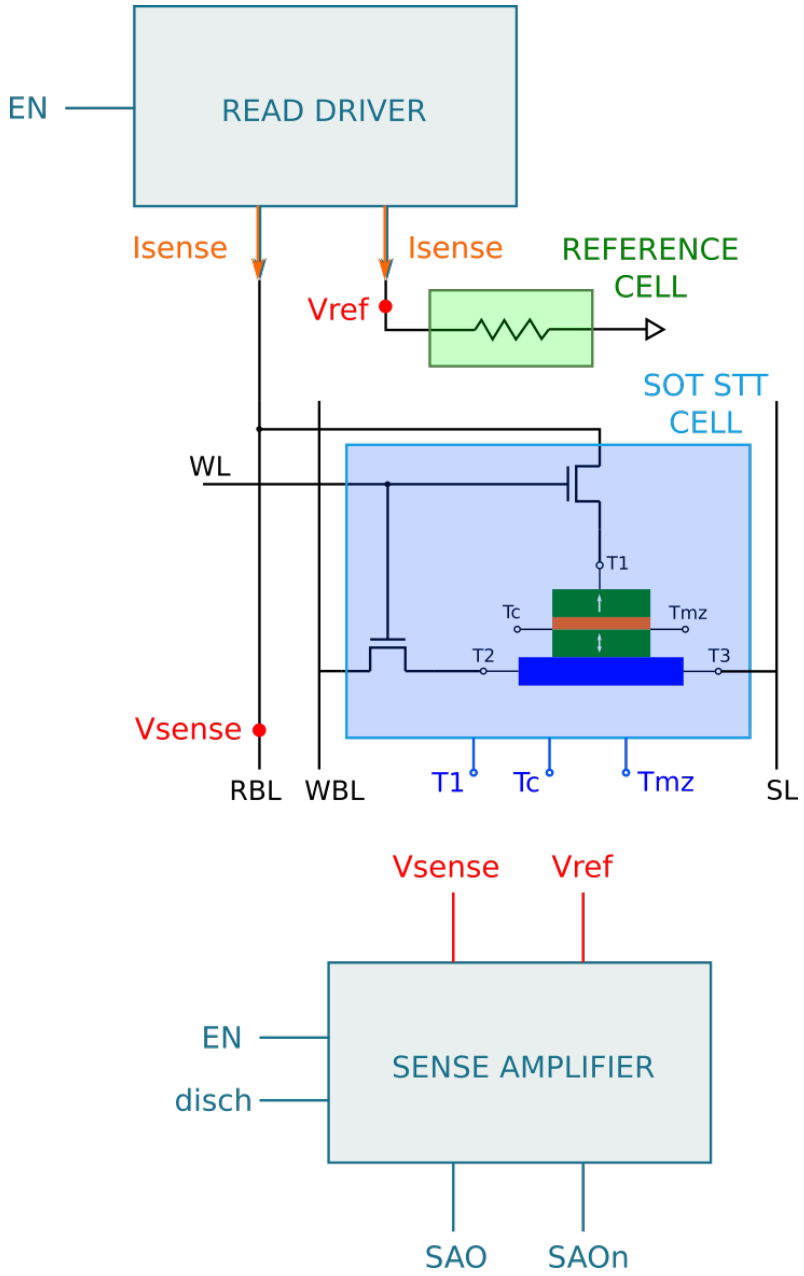


Figure 4.7. Sensing interface, that includes both the read driver and the sense amplifier, which are shared by all the cells of the column.

The nMOS transistors M10–M11 are exploited for discharging the SA outputs to ground, in order to avoid any possible voltage bias after a read operation. Since the output nodes and the input ones are separated by the

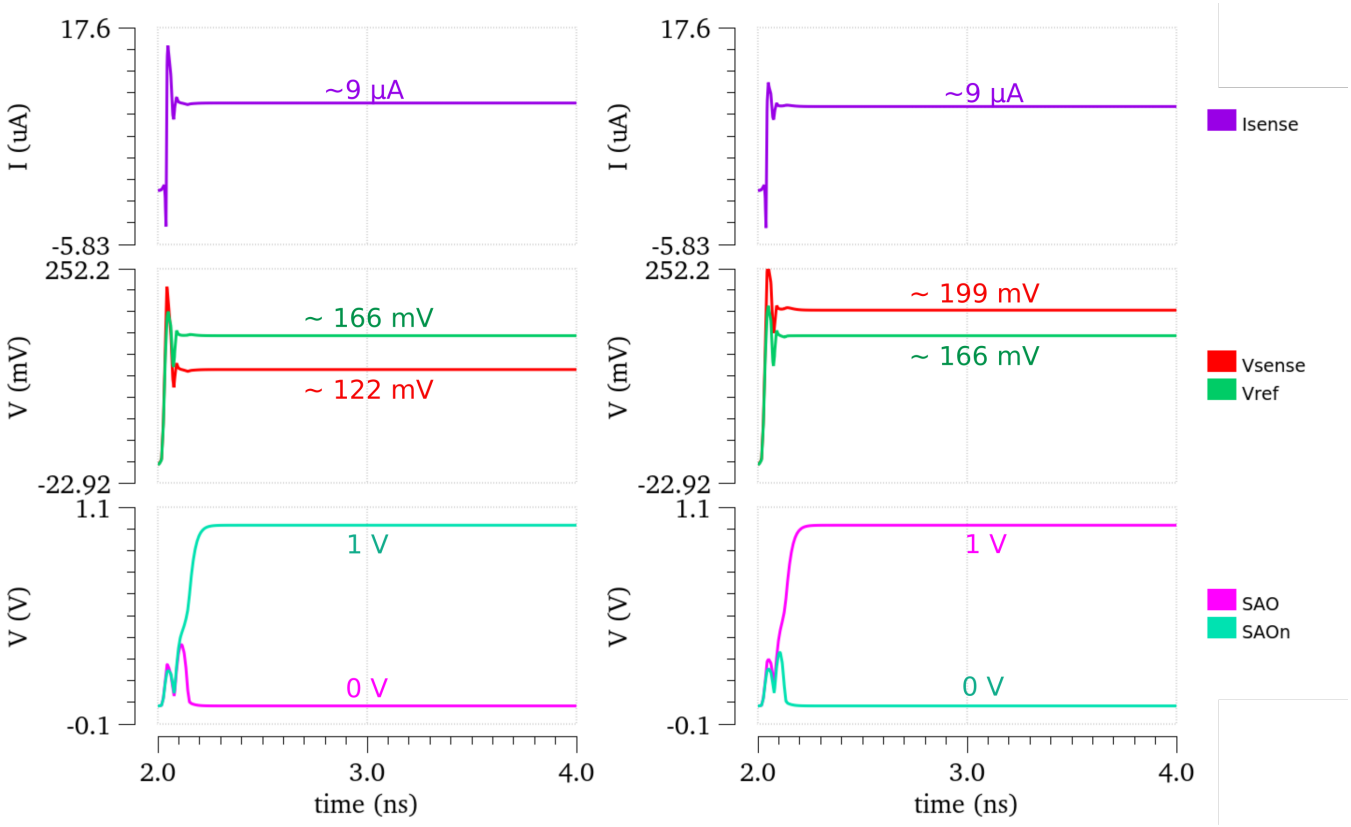


Figure 4.8. Example of reading operation for PAP = '0' (left) and PAP = '1' (right), where PAP is the initial MTJ logic state. The curves are directly extracted from a read simulation on a single cell performed in Cadence Virtuoso.

S–D path of the access transistors, that are in ON state when the SE is disabled, the M10–M11 transistors are useful for discharging the RBL line when no operation is performed in the array (i.e. an idle state). For this reason the discharge pulse duration is extended for the whole idle cycle, instead of just a single small spike at the end of the reading operation[4].

For the proposed SA scheme, no precharge to V_{dd} of the output lines is necessary. After activating the selected WL, a small time interval (about tens of picoseconds) is required for sampling the input voltage on the internal nodes of the latch, during which the SE is disabled. Such interval (called

[4]Differently from [95].

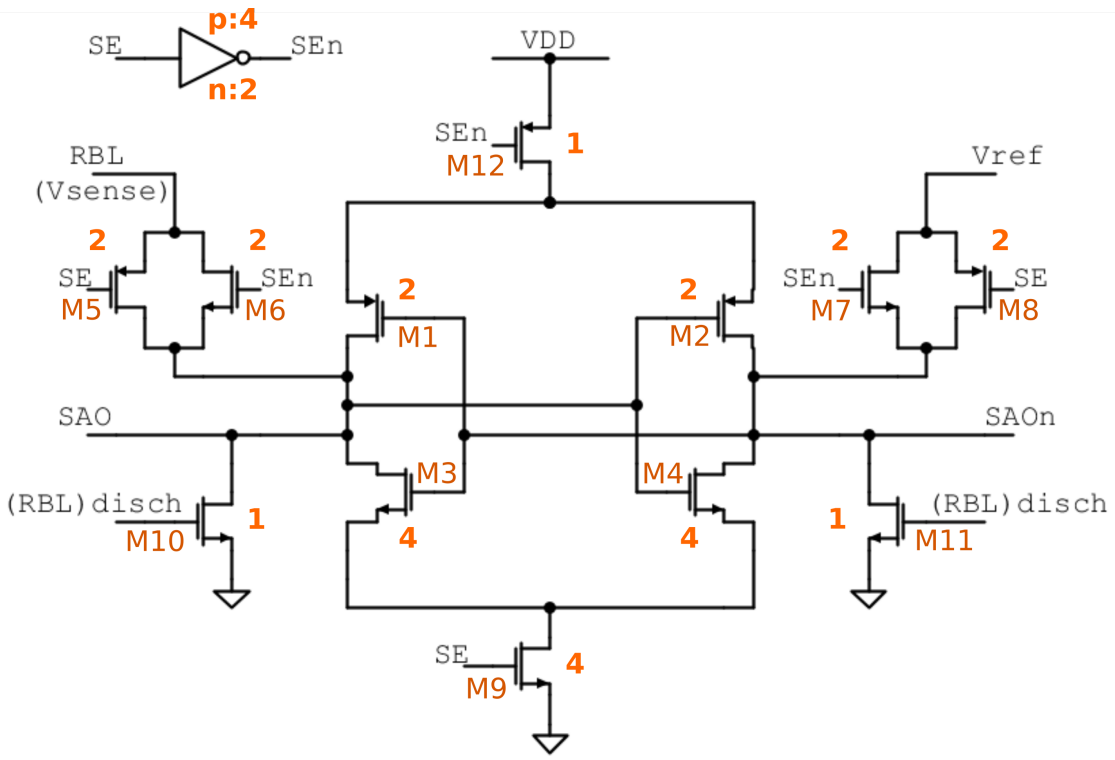


Figure 4.9. Adopted sense amplifier, adapted from [95].

WL–to–SE delay) is longer for larger arrays, hence it contributes in increasing the overall read delay, as shown in subsection 4.4.3. When the SE is activated, the SA connection to the RBL and reference cell is removed, and the sampled voltage —already present in the latch— is amplified, leading to logic '1' or '0' at the SA outputs.

The initial design of the read driver consisted in a parallel between a reference resistance and an ideal current generator (I_{sense}): the resulting voltage across the resistance (V_{ref}) represented the second input of the SA, along with V_{sense} of the selected cell.
With such configuration, however, a problem occurrs when computing the leakage power during read operation. In theory, if a sensing is performed on a single selected cell in the array, the power consumption increases with the increasing of the array size, due to leakage contribution. When using an ideal current generator, instead, such power contribution seems to decrease for larger arrays. This is due to the fact that, in each memory column, the unselected cells represent a parallel of very high resistances: when adding more

cells in parallel, the equivalent resistance decreases, while the current I_{sense} (provided by the read driver to each column) is constant since the current generator is ideal. In this way, the resulting power consumption $P = R \cdot I^2$ (where R is the equivalent resistance of the cells in the column and I is the sensing current), computed by Cadence, decreases for larger array height (i.e. larger number of cells in the column).
Such scenario does not occur when the input is directly provided by an ideal voltage generator, instead of a current one. In this case, in fact, the fixed V input and decreasing column resistance R makes the power consumption $P = V^2/R$ higher for longer columns.

For this reason, in order to have a reliable estimation of the power consumption, it is necessary to adopt a current source whose value depends on the load to which it is attached (i.e. the column resistance). In order to provide such a "real" current input, a pre–amplifier circuit as the one proposed in [96] is adopted: a current mirror with ad additional couple of nMOS which, in this work, are exploited just for enabling or not the current supply[5]. The enable signal of these transistors is called SE_2 since it represents a second Sense Enable, that differs from the first one[6] since it has not to be delayed with respect to the activation of the WL (the previously discussed *WL–to–SE delay*); this is due to the fact that when $SE\_2 = 1$, the same sensing current is made flowing through both the selected cell and the reference cell, thus generating the V_{sense} and V_{ref} SA inputs respectively. Hence, V_{sense} and V_{ref} must be already available at the SA latch input before the SE (enable signal of the SA) is activated.

The read driver circuit scheme is reported in Figure 4.10; the "strong" and "weak" sides of the current mirror are equally sized in order to guarantee the same I_{sense} to both the cells. The pMOS are sized as small as possible (higher resistance) since the sensing current is quite very low, for avoiding unwanted writing operations.

[5]Instead of clamping function as described in the article.

[6]The first SE is the one in input to the SA.

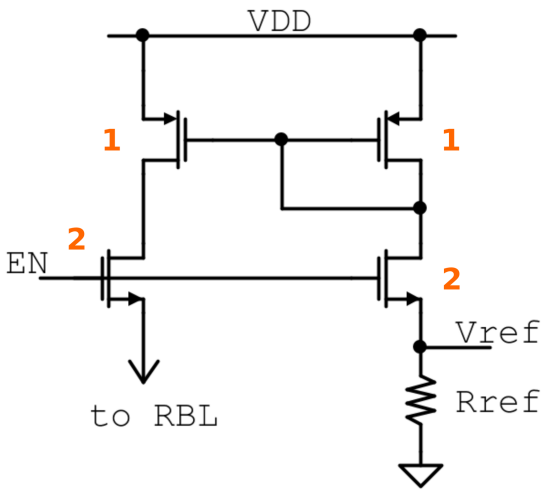


Figure 4.10. Read driver used to provide the proper I_{sense} to both the reference cell (R_{ref}) and the selected cell (through the corresponding RBL), generating the two SA inputs V_{ref} and V_{sense}, respectively. In the schematic, the *SE2* signal (Table 4.2) is linked to the EN input. It has been adapted from [96].

4.3.3 The WL and BL write drivers

Real drivers for both RBL/WBL write inputs generation (hereinafter called *BL write drivers*) and WL access are introduced in this section for two main reasons:

1. if ideal voltage generators on both the bitlines are used during write operations, they are turned off when a reading is performed, since such operation is current–based. As a consequence, both RBL and WBL are grounded, interfering with the path of the sensing current generated by the read driver (subsection 4.3.2): if RBL is connected to ground, a $V_{sense} = 0$ is fixed at the input of the SA, hence the reading operation cannot be fulfilled;

2. since the read operation is already based on a real driver (for the reasons explained in subsection 4.3.2), more coherent results can be obtained in the performance evaluation (subsection 4.4.3) by using real drivers also for writing and accessing the cell.

The WL driver is designed as a simple buffer consisting in a couple of inverters; no enable signal has to be provided to the driver since the WL voltage must be always defined for turning on/off the cell access transistors. The

reference scheme is reported in Figure 4.11.

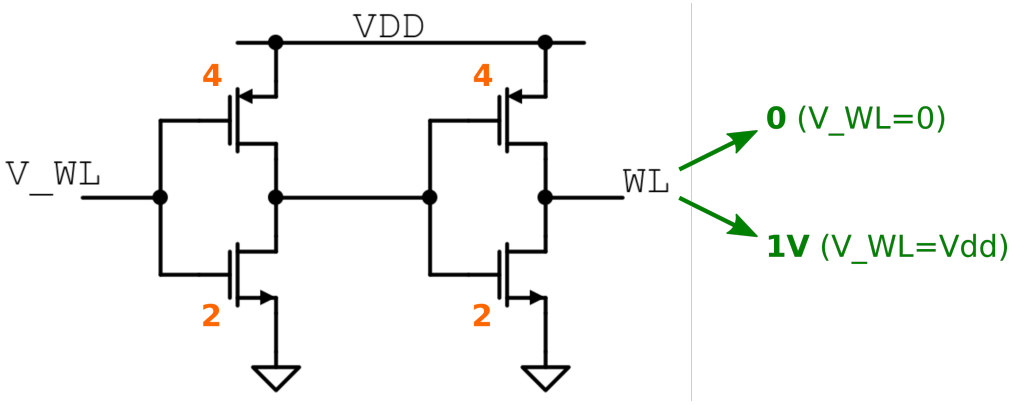


Figure 4.11. A simple buffer for wordline activation.

The RBL write driver is responsible for generating the proper STT write pulse, as the ones in Figure 4.5. The circuital scheme is shown in Figure 4.12: it consists in a double inverter stage provided with an enabling signal.
The particularity of such driver is the bipolar supply: a negative rail V_{ss}=−0.5 V must be given to the driver in order for it to generate the desired negative/positive write voltage pulses. The negative voltage requirement for SOT STT cell writing is discussed in subsection 4.3.1. The IN signal corresponds to the word bit that has to be written in the cell.
The nMOS and pMOS sizing is decided, after a series of tests, to be the one in Figure 4.12 for ensuring the optimal inputs amplitude in order to minimize the MTJ switching duration. If an higher dynamic is required (for example, in other driver applications), it is sufficient to increase the transistors size, or to put additional MOSFETs in parallel to the pull–up or pull–down transistors of the second inverting stage, for decreasing the RBL–to–V_{dd}/V_{ss} resistive paths.
Another peculiarity of such circuit is that both the enable signal EN and the input IN must be bipolar as well (i.e. V_{dd}/V_{ss} for logic '1'/'0'); otherwise, when the enable is at 0 V, the pulldown nMOS sees a $V_{gs} = V_{ss}$ that prevents it from turning off, and the RBL remains dependent on the IN value even if the driver is disabled.

The function of the WBL write driver, on the other hand, is to provide the correct 0.5 ns–long SOT voltage pulse for speeding up the MTJ switching

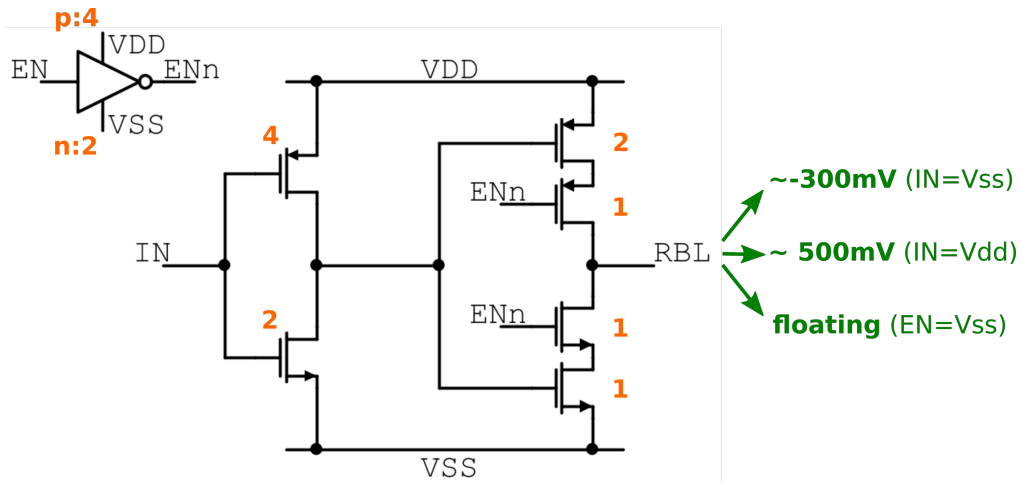


Figure 4.12. RBL write driver: the nMOS and pMOS ad hoc sizing (higher/lower R) controls the amplitude of STT writing voltages (~ -300 mV to write '0', ~ 500 mV to write '1')

operation (Figure 4.5).
The circuit is depicted in Figure 4.13. The pMOS connected to V_{dd} is the one that delivers the necessary SOT voltage to the WBL.
Two signals control the behaviour of the driver: the enable WBL_EN (different from the one of RBL driver) and the discharge WBL_disch. The WBL_disch signal has a role similar to the discharge signal provided to the sense amplifier[7] (Figure 4.9): it can be used to discharge the WBL when no operation is performed (i.e. idle state). Hence, during a writing operation, it is usually set to '0'. The WBL_EN signal, instead, is the one that activates or disables the pull–up pMOS: since the SOT pulse duration (0.5 ns) is shorter than STT one (2 ns), when EN is put to 0 —after the 0.5 ns pulse— it turns off the pMOS, disconnecting the driver from the WBL (since the nMOS is already detached by WBL_disch = 0).

[7]In that case, the discharging is performed on the RBL.

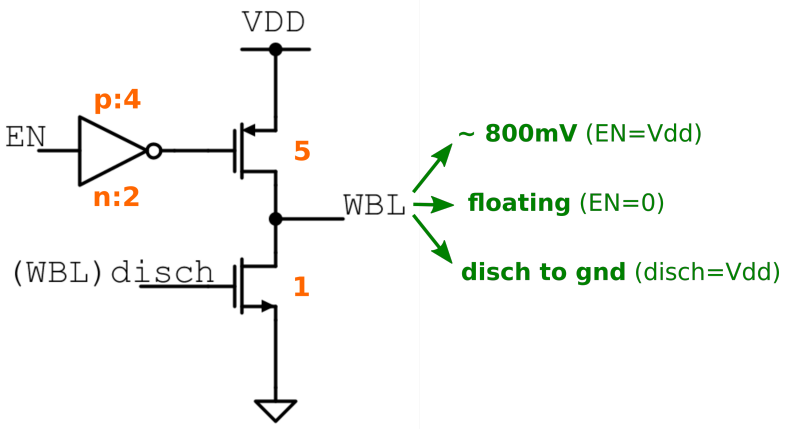


Figure 4.13. WBL write driver: the pMOS sizing controls the amplitude of SOT writing voltage ($\sim$ 800 mV for both write '0' ad '1'). The discharge signal allows to put the WBL to ground when no operation is performed

4.3.4 The array organization

The SOT STT cells were firstly tested in a single 8–bit column and a single 8–bit row with all ideal drivers. Then, for the reasons discussed in subsection 4.3.3, real drivers were introduced, and their functioning has been verified in 8x8, 16x16 and 32x32 array configurations together with all the other designed peripherals.

A qualitative representation of the array organization is shown in Figure 4.14. As it can be seen from the figure, RC circuits are introduced in order to simulate the interconnection parasitics. The RC interconnections represent an important contribution to delays in all the operations performed in the memory: for example, a cell placed at the end of the row (i.e. farthest from the WL drivers, see Figure 4.17) is the last one to be accessed by WL signals, while a cell in the last row (i.e. farthest from the BL drivers) is the slowest to be written, and so on. However, the parasitic resistances and capacitances are sized to be small (5 Ω and 100 aF respectively) for preventing the delays measures to be overturned by the presence of interconnections and obtain more significant results.

Firstly, an 8x8 array is designed with 64 bit–cell SOT STT components, by repeating the previously designed 8–bit row patterns and connecting them

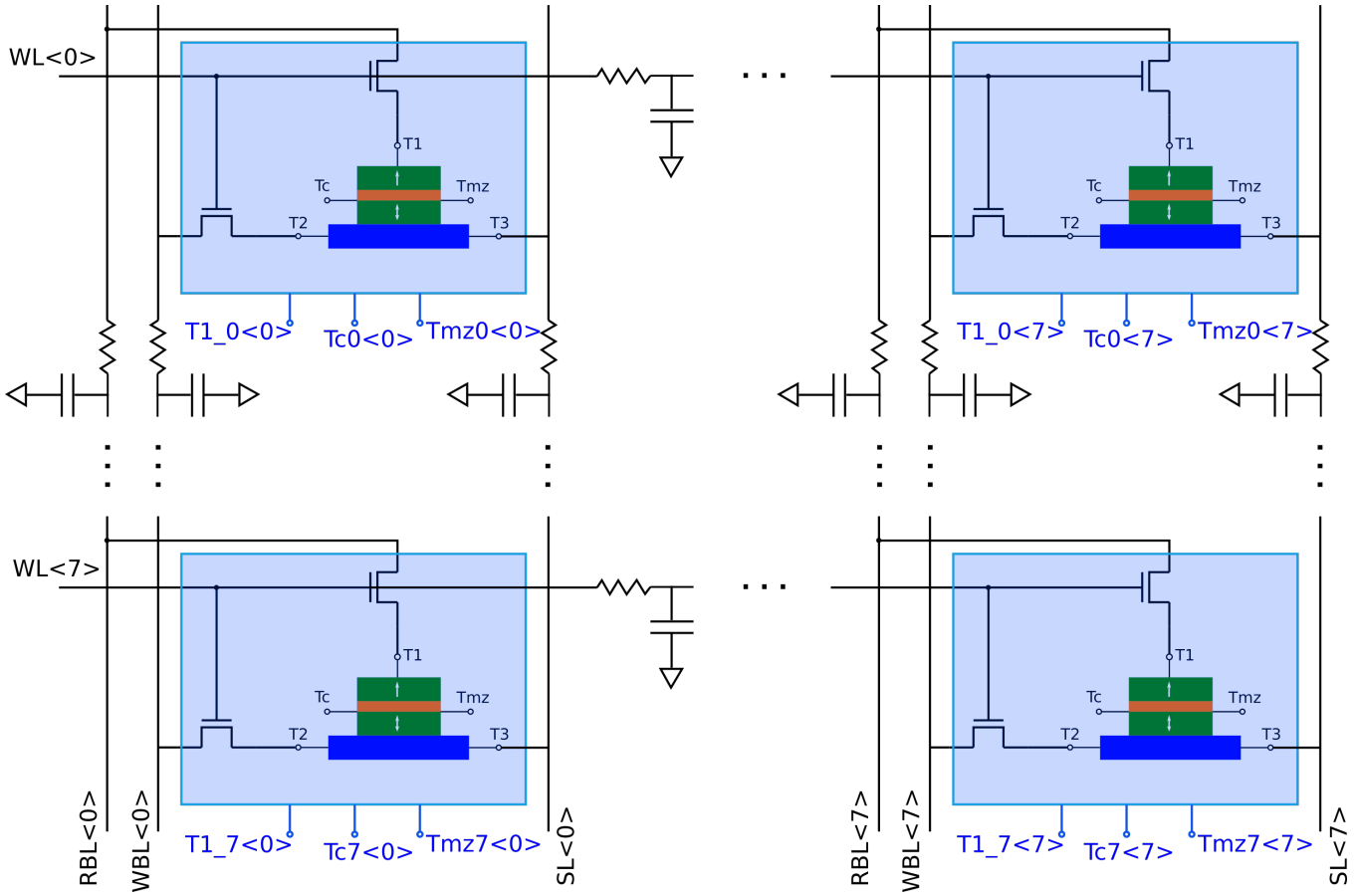


Figure 4.14. Parasitic contributions (due to the interconnections) distributed in all the lines of an 8x8 array.

with each other[8].
Then, a block representation (*symbol*) of the array is extracted through Cadence (Figure 4.15): wordlines/bitlines pins are added in top–bottom/left–right positions of the array for leaving the possibility of connecting the array (ex. with other ones) in both vertical and horizontal directions, respectively. Data pins (Tmz) of all the bit–cells of the array are also made available as outputs (for check purposes), even if only first and last row Tmz are considered for evaluating the array performance.

[8]For "pattern" is intended the row schematic, and not its symbol; in fact, for convenience, the 8–bit row symbol has not been created, since some tests have been done by modifying a single cell in the entire array for verifying the correct functioning of all the memory operations in different parts of the array.

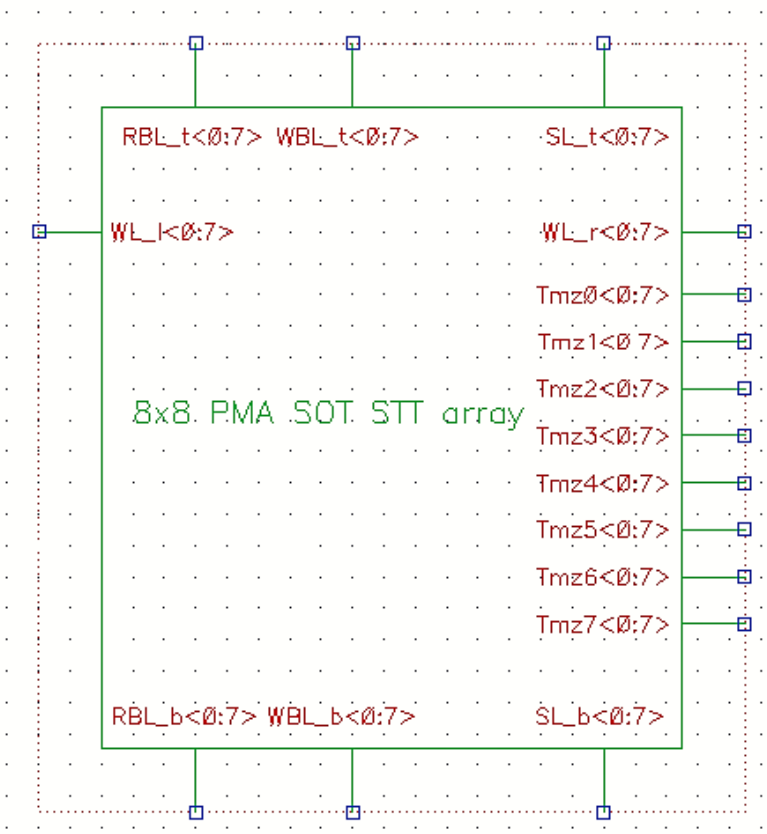


Figure 4.15. Cadence symbol of the SOT STT 8x8 array

In order to obtain a 16x16 array, four 8x8 array symbols were combined as shown in Figure 4.16.

The overall schematic for testing the memory is reported in Figure 4.17. The real drivers are linked to the top of the bitlines and provide the correct voltage/current inputs to the array. The sense amplifiers read the voltage V_{sense} at the bottom of the array. The WL drivers, instead, are attached to the left side of the array.

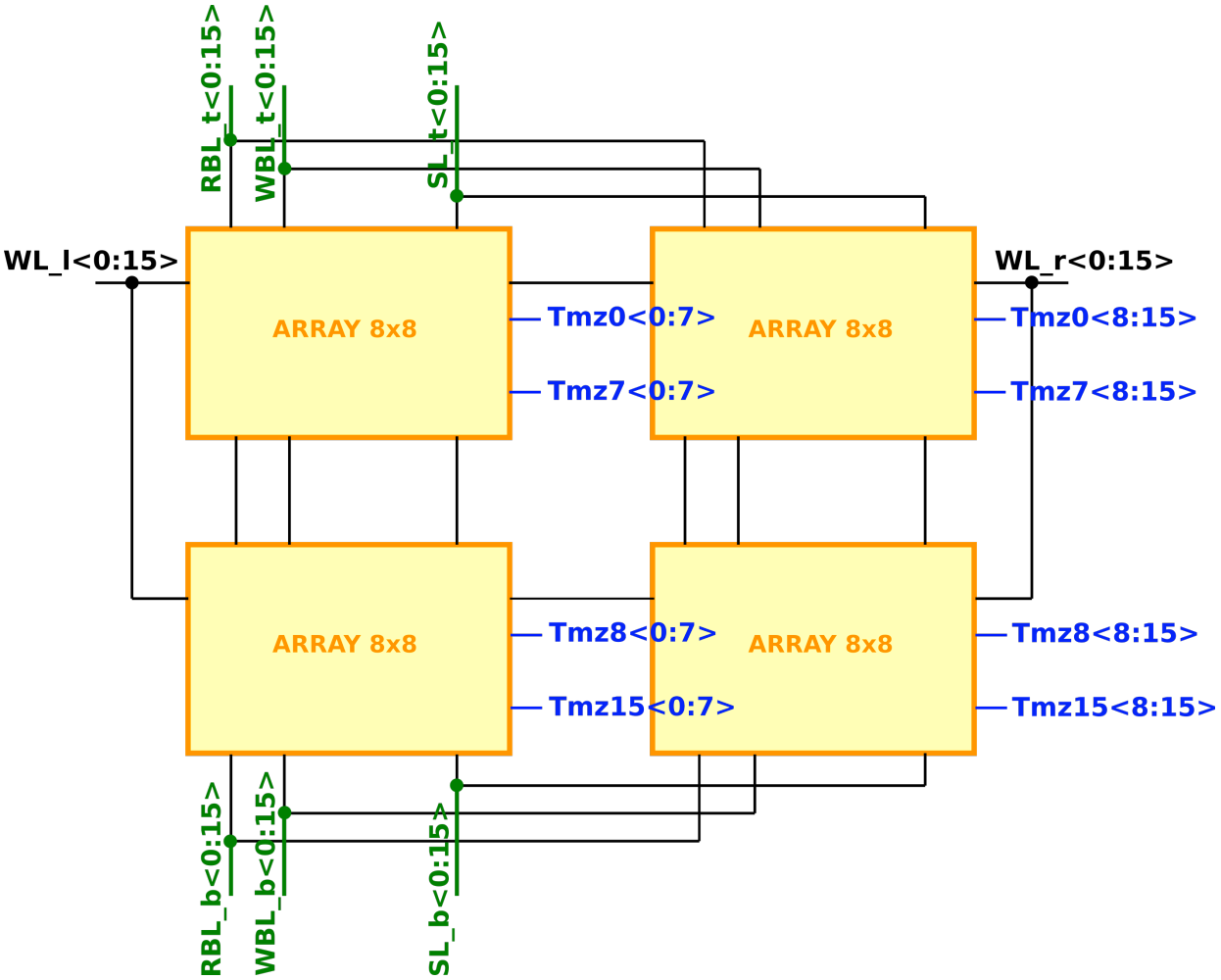


Figure 4.16. 16x16 array obtained by combining 8x8 arrays components.

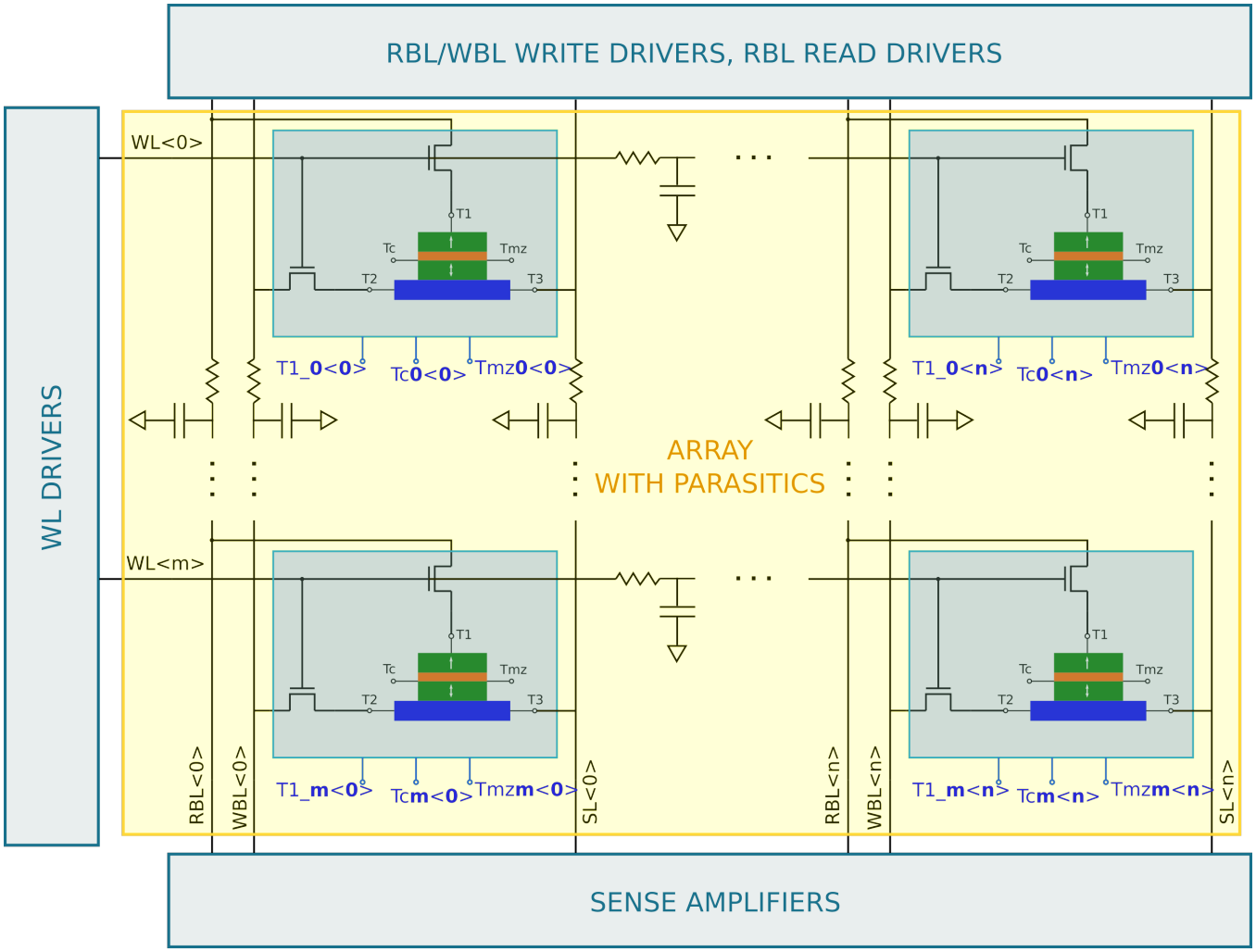


Figure 4.17. SOT STT memory array and peripherals organization.

4.3.5 Testbench states

The signals that control the functioning of the memory are listed in Table 4.2, together with a brief description. Such signals represent the entire list of inputs that must be provided to the schematic for testing all the memory operations. They are supplied through a series of ideal voltage generators whose parameters (delay, width and value, see Table 4.4) are left as variables and then set in the Cadence simulation environment. The simulation settings are then saved as states (for all memory operations) that can be reused for successive simulations.

Signal	Write "0"	Write "1"	Read	Idle	Signal description
V_WL	V_{dd}	V_{dd}	V_{dd}	0	wordline activation
IN	V_{ss}	V_{dd}	V_{ss}	V_{ss}	data input (to be written)
RBL_EN	V_{dd}	V_{dd}	V_{ss}	V_{ss}	RBL write driver enable
WBL_EN	V_{dd}	V_{dd}	0	0	WBL write driver enable
WBL_disch	0	0	0	V_{dd}	WBL discharge (WBL write driver)
SE	0	0	V_{dd}	0	SA enable
SE2	0	0	V_{dd}	0	RBL read driver enable
disch	0	0	0	V_{dd}	RBL discharge (SA)

Table 4.2. Input signal values for all the memory operations.

The WBL_EN signal in Table 4.2 is the only one that has to be provided for a duration lower (0.5 ns) than the other ones (2 ns), since it corresponds to the duration of the SOT pulse. The write operation requires a minimum of 2 ns to be performed (for the Tmz to be switched completely), while the read cycle can be reduced down to 1 ns. All the signals presented in Table 4.2 are considered for power consumption calculation, as discussed in subsection 4.4.2.

The other signal of interest are listed in Table 4.3, followed by a brief description and the indication of the operation for which they are relevant to be observed for. Tmz, SAO/n and V_{ref} correspond to output signals of the single cell, sense amplifier and read driver, respectively; V_RBL and V_WBL are instead voltages taken directly from the bitlines of interest. The signals in both Table 4.2 and Table 4.3 can be observed in the Cadence output waveform plots after running the state corresponding to the selected operation.

Signal	Relevance	Signal description
V_WBL	write	SOT voltage pulse
V_RBL	write/read	STT voltage pulse/V_{sense}
V_{ref}	read	Voltage on reference cell
SAO	read	Sense Amplifier output
SAOn	read	Negated Sense Amplifier output
Tmz	write/read*	MTJ magnetization state

Table 4.3. Other signals of interest involved in memory operations. Note: *It may be interesting to observe the Tmz for checking that no state disturbs occur during reading operation.

In Table 4.4 are listed all the variables that can be set in the selected simulation state.
The PAP parameter is particularly relevant since it allows to change the initial logic state of the SOT STT MTJ: for example, when loading 'write1' state, Figure 4.5 (b) is obtained for PAP = '0' while Figure 4.5 (d) for PAP = '1'. The same procedure is valid for 'read 0' (PAP = '0') or 'read 1' (PAP = '1') operations (Figure 4.8). This variable is equally set for all the cells of the array.

The $fall\_ris\_time$ variable represents the falling and rising time of all the waveforms generated by ideal voltage generators in the schematic. R_{ref} is the value of the reference cell resistance and, if changed, it modifies the minimum *WL–to–SE_delay* discussed in subsection 4.3.2, since it introduces an asymmetry in the read 1/0 operations[9].
The last three variables of Table 4.4 are referred to a generic signal of Table 4.2: in fact, in the simulation states are set all the input signals (WL, IN, WBL_EN etc.) amplitudes according to Table 4.2 together with their duration (WL_pulse_width, IN_pulse_width, $WBL\_EN\_pulse\_width$ etc.) and their delay (WL_delay, IN_delay, WBL_EN_delay etc.).

The IN signal amplitudes are specified for each column and row by an appendix with the corresponding column/row number, for leaving the possibility of testing all possible input word to be written: for example, in an 8x8

[9]Due to the fact that it changes the V_{ref}, that is set to be exactly in the middle of high–low V_{sense} interval.

array, a word = "01000111" can be written in the selected WL by loading the *write* simulation state and simply changing the variables $IN\_0 = -0.5\,\mathrm{V}$, $IN\_1 = 1\,\mathrm{V}$, $IN\_2 = -0.5\,\mathrm{V}$, ... , $IN\_7 = 1\,\mathrm{V}$. When an *_all* appendix follows the signal name, it means that the amplitude is set equally to all columns/rows without a numerical appendix: for example, in the 8x8 array, SE_all corresponds to the sense enable amplitude for all the column SA from 0 to 6, while SE_7 is specified for the last column SA. This is done for leaving the possibility to quickly activating all the SE values ($SE\_all = 1$, $SE\_7 = 1$) or just the last one ($SE\_all = 0$, $SE\_7 = 1$), in order to compute the performance of reading operation on one single cell[10]. However, if there is any doubt, by selecting the wanted variable in the loaded state and clicking "find", Cadence will immediately visualize the location of the selected variable in the schematic.

The asy,$Bext$, fac_FL, SAS, $SDTS$ parameters are already described in section 4.2 and are left as variable in case of, in future, device variation factors are wanted to be taken into account at array level [11]. The *seedin* variable is a parameter required by the SOT STT model, and it is used as random number to initialize a random distribution function inside the Verilog-A file.

A Python script (section B.1) for input waveforms generation has been written and successfully tested in a SOT STT cell schematic, simulating all the possible memory operations in a single time analysis; the results are shown in Figure 4.19. Light blue and orange vertical stripes corresponds to write '0'/'1' and read '0'/'1', respectively. The white stripes that separate writing and reading cycles are 0.5 ns–long idle cycles, useful for exploiting both RBL and discharge signals[12] for removing any residual bias voltage. It is chosen a Tck = 0.5 ns that allows the generation of the SOT writing pulse (WBL_EN), and the read cycle is reduced at its minimum (1 ns) for testing how much it can be shortened. For the results reported in subsection 4.4.3, instead, it is chosen a 2 ns–long read cycle, since the increasing WL–to–SE delay w.r.t. the array size may cause read errors for too short reading pulses. It is interesting to note how the sense amplifier output (SAO) is dependent on the RBL voltage (during writing cycles it sees exactly the writing voltage

[10] The one on the last column for the reasons explained in subsection 4.4.1.

[11] As stated in section 4.2, no device deviation is considered for all the performed simulations.

[12] see subsection 4.3.2 and subsection 4.3.3.

Variable	Value	Variable description
PAP	0/1	Initial MTJ logic state
Cparasitic	500 aF	Interconnection parasitic C
Rparasitic	5 Ω	Interconnection parasitic R
fall_ris_time	50 ps	All signal falling and rising time
R_{ref}	18 kΩ	Fixed R of reference cell
V_{dd}	1 V	Positive supply voltage
V_{ss}	−0.5 V	Negative supply voltage
asy	1	section 4.2
Bext	0	section 4.2
fac_FL	0.5	section 4.2
SAS	0	section 4.2
SDTS	0	section 4.2
seedin	10	section 4.2
signal	Table 4.2	Signal of Table 4.2 amplitude
signal_pulse_width	0.5 ns/2 ns*	Signal of Table 4.2 pulse width
signal_delay	2 ns**	Signal of Table 4.2 delay

Table 4.4. Complete list of all the variables used for simulating the designed SOT STT memory in Cadence. Notes: *the only signal 0.5 ns–long is the WBL_EN, as explained before. **2 ns of delay is just set for convenience, since during testing phase it has been observed also the signal values before the performed operation: it can be freely changed together with the simulation time interval [note: the delay–to–stop–time interval of the simulation regulates the integration interval for power consumption calculation.

pulses driven by the RBL driver, subsection 4.3.3): this is self–evident by observing that only an activated pMOS separates the RBL from the *SAO* (Figure 4.9).

However, a different type of simulation flow is adopted, that correspond to the one illustrated in Figure 4.18. The Cadence simulation environment can be saved into a *state*, which includes the whole variables setting, the type of analysis, the outputs to plot and all the functions that have to be computed (ex. delays, power consumption etc.) in a single file. Different states are saved for performing an operation each: by simply loading and running the state *write1*, for example, all output waveforms and performance evaluation corresponding to the homonymous operation will be displayed in Cadence.

The reasons behind the choice of such type of simulation flow are the following:

1. a lot of time is saved when one operation at time is performed, since the simulation duration can be reduced a lot (minimum of 1 ns for a read cycle): a loaded state directly sets the analysis interval for the desired operation and, in addition, it plots only the output signals of interest corresponding to such operation;

2. the *state* approach allows to perform parametric simulation with time parameters, such as delays and pulse widths, since they are saved as variables in the schematic. This is a crucial point, for example, for evaluating the minimum *WL–to–SE_delay*: different parametric analyses are performed by slowly changing the SE delay (and the corresponding pulse width) and observing the SA outputs during a read operation. The minimum SE_delay that makes the SA read the correct cell value corresponds to the *WL–to–SE_delay* of the selected schematic[13].
This approach is also useful in deciding a proper point of work with different pulse widths of SOT and STT pulses (see section 4.2).

3. for performing the simulations of point 2), all the input voltage generators are already set for each array schematic. Such generators must have specific names for being considered by the Python script used in power consumption evaluation (see subsection 4.4.2). If it is considered that just a single cell requires up to 8 signals (Table 4.2), each provided by a dedicated generator, it is evident how much time–expensive would be to substitute all the generators (ex. in a 32x32 array design) considering that:

   - the newly introduced generators are *vpwlf* components, able to take the time information from a file of a given folder path. The correct folder path and file name have to be provided to each vpwlf generator introduced in the schematic.
   - the new generators must maintain the same component names of the previous ones in order to avoid problems with the Python script for memory consumption computing.

[13] As anticipated in subsection 4.3.2, the WL–to–SE_delay is dependent on the array size

For such reasons, a *state* approach is preferred for providing inputs to the Cadence schematic.

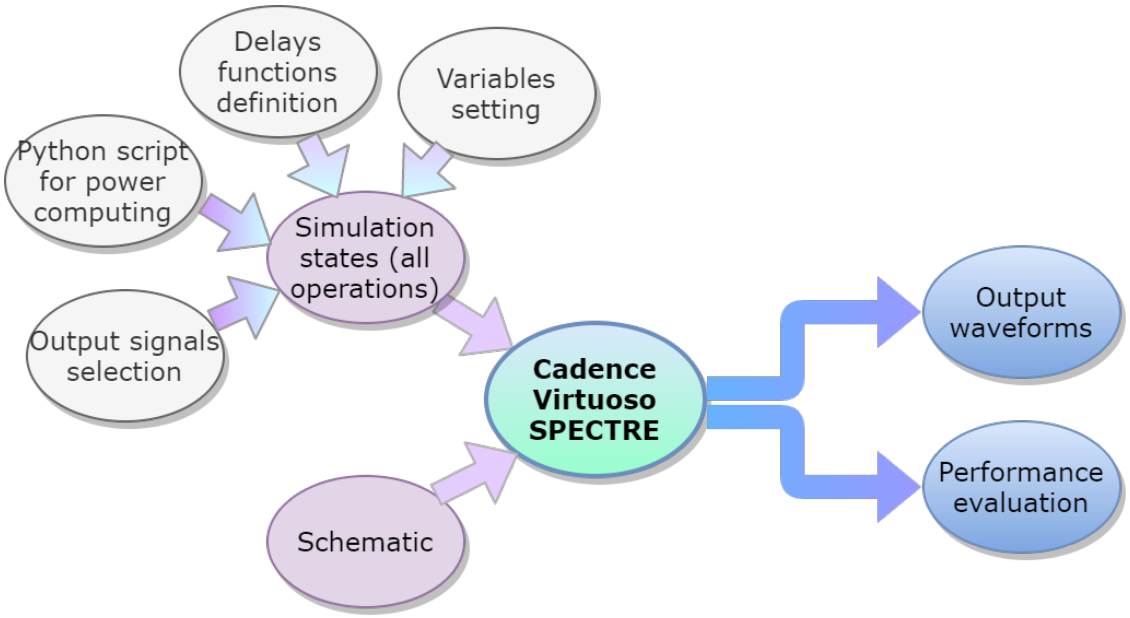


Figure 4.18. Simulation flow.

4.4 Performance evaluation

This section is committed to explain how the delays and power consumption are computed for all the memory operations. In both cases, dedicated functions are defined in Cadence environment and directly computed by the program when simulating an operation.

4.4.1 Delays computing

The read delay is defined as the 50% delay between the WL activation signal and the SA output (*SAO* if a logic '1' has to be read, *SAOn* for '0', Figure 4.20).
As anticipated in subsection 4.3.4, the location of the selected cell in the array is important: increasing the distance from both the WL drivers and the SAs/write drivers, it increases the capacitive load associated to the row/column, hence it increases the delay of the involved signals. It is good

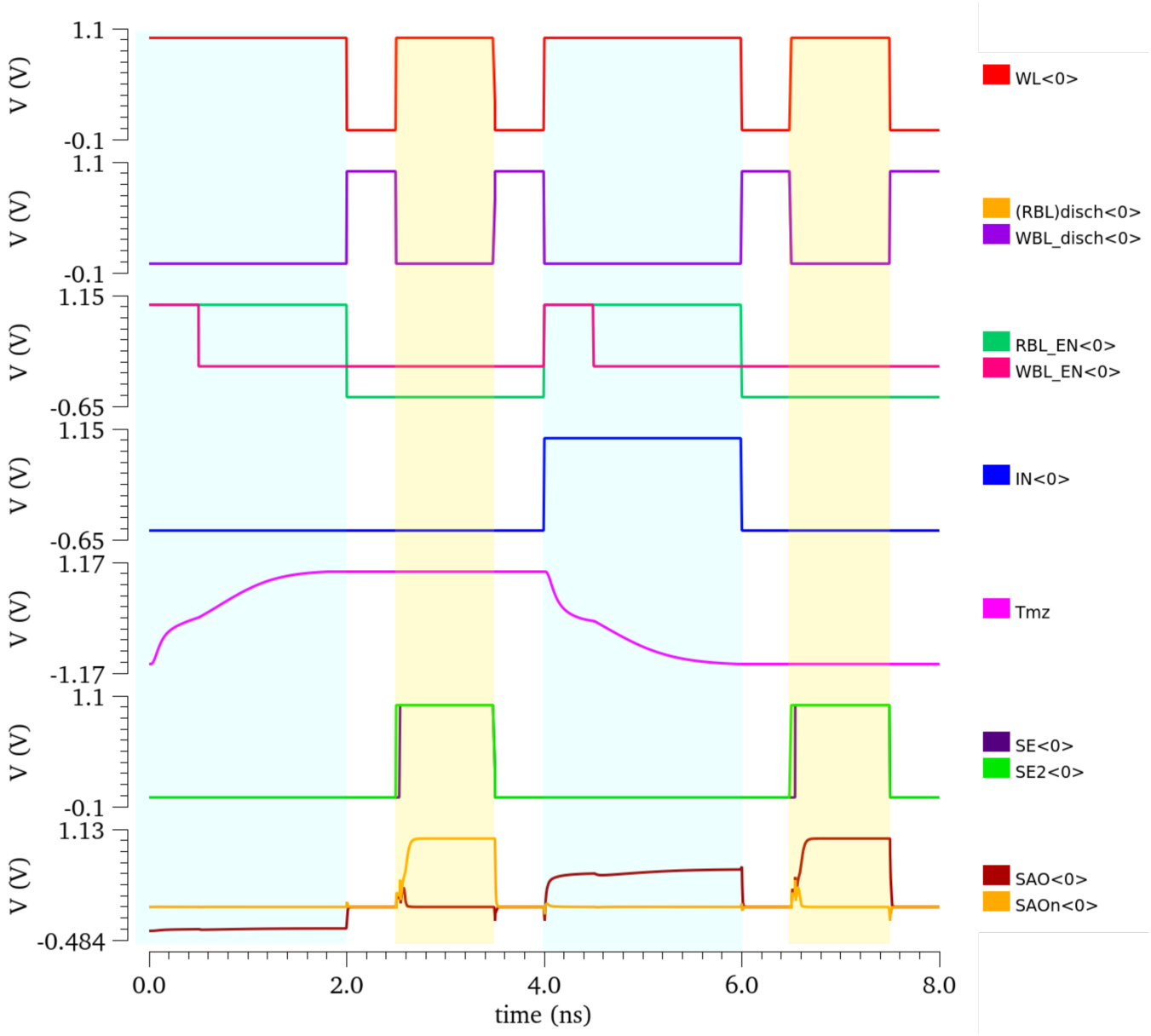


Figure 4.19. All the memory operations performed in a single time simulation through Python generated inputs (section B.1). The plotted signals are selected among the ones listed in Table 4.2 and Table 4.3.

practice, for this reason, to evaluate the worst case delay for both reading and writing operation.

The write delay definition is displayed in Figure 4.21 for both write '0'/'1' cases. It corresponds to the 50% delay between the WL activation signal

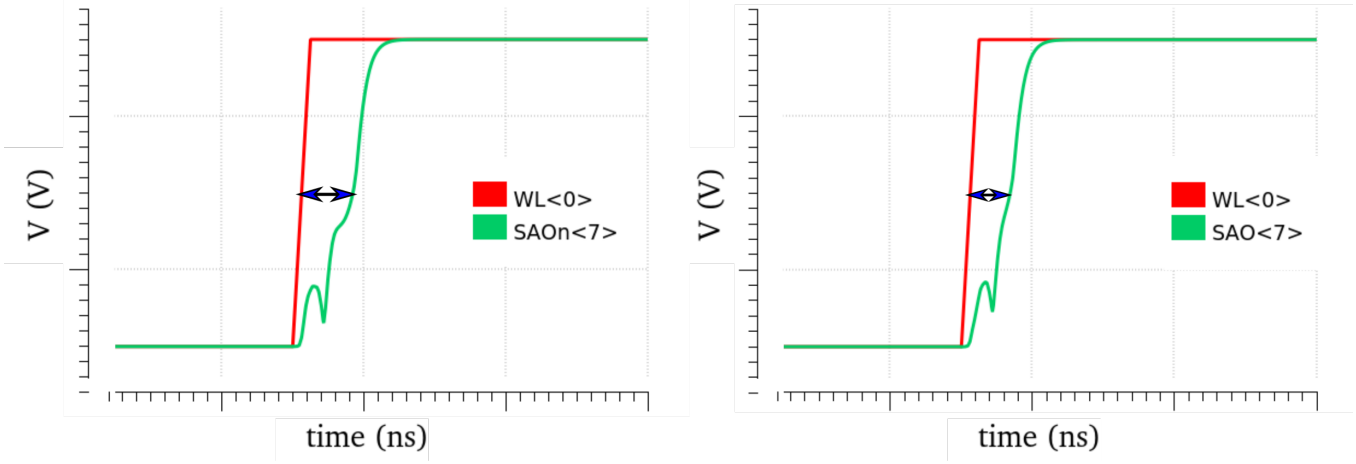


Figure 4.20. Read "0" (left) and read "1" (right) delays example for an 8x8 array. The read "1" delay is sligthy lower than the read "0" delay for all the tested array sizes, as reported in Figure 4.23.

and the Tmz (vertical magnetization state) signal. Since the Tmz dynamic covers a $[-1\,\mathrm{V} \div 1\,\mathrm{V}]$ interval, the delay indicates the time distance between the 50%WL (WL = 0.5 V) and the Tmz that reaches 0 V with rising (write '0') or falling (write '1') edge.

The largest write delay has to be evaluated in the last row of the array, since, in this memory organization, it represents the farthest point from the write drivers; moreover, it is to be considered the last cell (the rightmost one) of the row since it is the farthest from the WL drivers.

The same concepts can be applied for the worst read delay, considering the highest distance from the SAs (at the bottom of the array) and the WL drivers. Hence, it is to consider the last cell (the rightmost one) of the first row. In this case, it can be noticed that also the RBL read drivers (subsection 4.3.3) —located on the top of the array— are involved in the reading operation; however, the SE2 signal (responsible for the activation of the read drivers) is activated at the beginning of the reading cycle, before the activation of the SE (enable signal of the SAs) [14]. Hence, the cell–to–SAs distance represents the higher contribution to read delay.
Both delays are set up as functions with Cadence calculator and directly computed when simulating an operation.

[14]The *SE*, in fact, has to be delayed according to the *WL–to–SE_delay*.

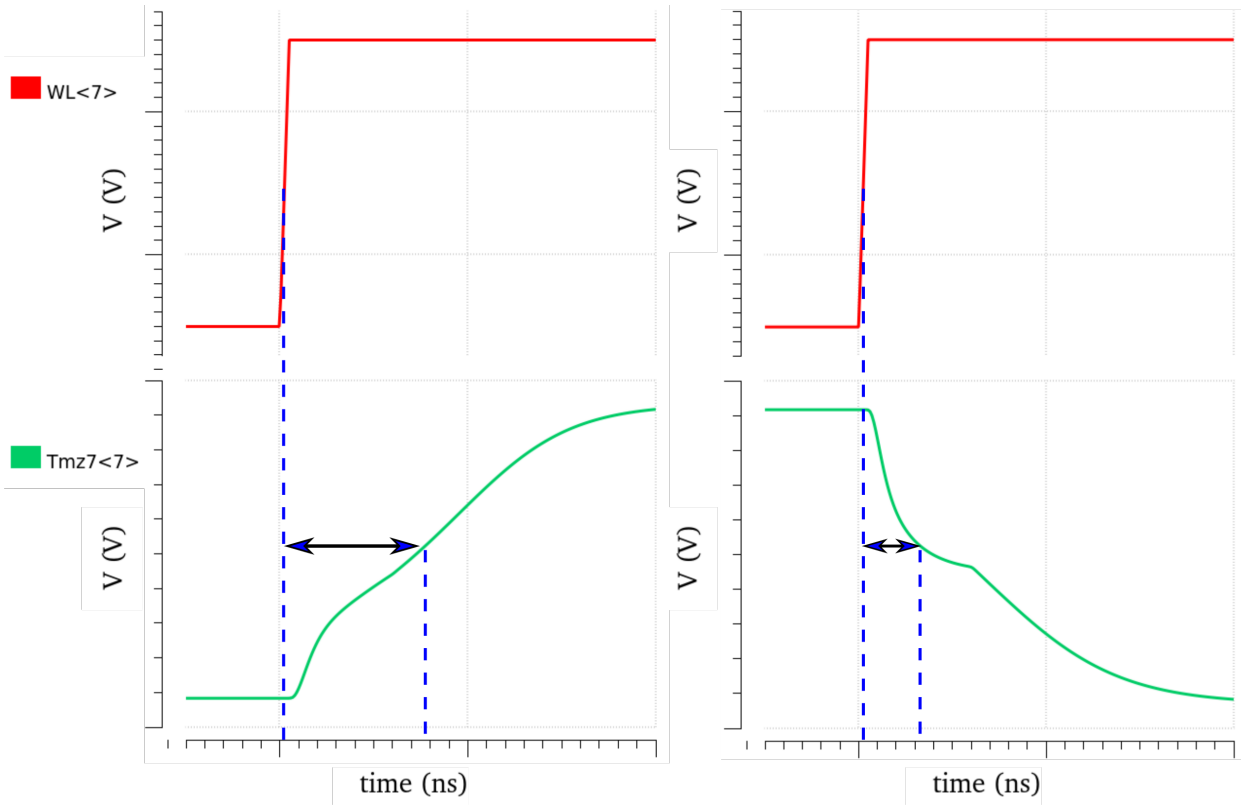


Figure 4.21. Write "0" (left) and write "1" (right) delays example for an 8x8 array. The write "1" operation is generally faster than the write "0" operation for all the tested array sizes, as reported in Figure 4.22.

4.4.2 Power consumption computing

In order to have an estimation of the power consumption of each memory operation, four functions have been defined with Cadence calculator and computed when simulating: instantaneous power $p(t)$, average and peak power consumption, and energy consumption.

By default, Cadence simulator allows to extract the instantaneous power resulting from a "tran" (temporal) analysis: in practice, the instantaneous power of the entire schematic can be plotted after running a simulation by simply displaying the *pwr* function automatically generated in Cadence *Result Browser* tool.
However, after a series of tests on the schematic, one can notice that the consumption associated to all the Verilog–A–defined components (i.e. all the

SOT STT MTJs) is equal to zero. In order to bypass this problem, the instantaneous power is computed as summation of the power contribution of all the generators in the schematic. To deal with this sum, that for a small array starts to have a consistent number of terms, a Python script (section B.2) is employed, which is capable of generating the correct function (to insert in Cadence calculator) for different array sizes.
The script is based on the fact that the schematic is organized in such a way that all wires of one type (ex. the ones connecting RBL_EN signals to the read drivers) are named with the same iterated label (ex. RBL_EN<0>...<n>), and that the generators that supply those wires have the same iterated names (ex. V_RBL_EN0...n). This is due to the fact that the script writes on a file the summation of all $V \cdot I$ power contributions considering V = voltage on the wire attached to the generator and I = current on the "minus" pin of the generator[15] (ex. VT("/RBL_EN<0>")*IT("/V_RBL_EN0/MINUS") + ... etc.). Hence, if the wires attached to the generators or the generators names of the schematic are re–named, the Python script has to be modified accordingly.

In order to compute the average power consumption related to a single operation, the *avg* special function is set in Cadence calculator. This function calculates the integral of the instantaneous power p(t) over the entire range for which it is defined, and it divides the integral for the same range. It means that, by setting the simulation time equal to the operation cycle duration, it is computed the average power of that single operation. If it is desired to weigh such value with respect to the total average power, it is sufficient to multiply it for the operation duration and divide it for the simulation time.

The peak power is evaluated by simply applying the *ymax* calculator special function to the instantaneous power; it computes the maximum value of $p(t)$ inside the simulation interval. It has to be noticed that the peak power is dependent on the falling and rising time of the input signals. In fact, when this parameter has a lower power, the signal edge transition is slower and "relaxed" in a larger interval: this means that also the instantaneous power is more distributed, hence its maximum value is lower.
The energy consumption is instead calculated through the *integ* function, that integrates the instantaneous power along the specified range (set equal

[15]Such pin is chosen according to the current sign convention in Cadence schematic: if taken on the "plus" pin of the generator, the current would be with negative sign

to the operation interval). It can be also obtained by multiplying the operation average power for the operation cycle duration.

4.4.3 Measurements and results

The performance measurements related to the single cell and different array sizes are reported in this section.

The results are obtained by applying the functions discussed in section 4.4 for each memory operation.
Both write '0' and '1' performance are discussed due to the slight asymmetry of the Tmz switching curve in the two cases.
Also the reading operation is considered for low and high state, since the V_{ref} voltage (subsection 4.3.2) is only approximately at the midpoint of the $V_{sense,L} \div V_{sense,H}$ interval. This is mainly due to the choice of the reference cell resistance and the read driver design: with the adopted scheme, for higher R_{ref} it decreases the current exiting from the read driver, hence it decreases the $V_{sense,L} \div V_{sense,H}$ interval. This is due to the fact that the current mirror (at the basis of the read driver) is dependent on its load. In this way, when decreasing the R_{ref} value, the V_{sense} dynamic enlarges but the V_{ref} is no more at the exact middle of such dynamic, hence it enhances the asymmetry of read '0'/'1' delays. The chosen R_{ref} is a good compromise for maintaining a relatively low asymmetry and an appreciable V_{sense} dynamic (see Figure 4.8).

Except from when it is specified differently (ex. Figure 4.30), all the performance evaluation in this section are computed for all the cells in the same row performing the same operation: for example, the results in Figure 4.22 (left) are referring to all the cells on the bottom row[16] are performing a write '0' operation, i.e. when a "00...00" word is written in the last row. This is done for taking into account the worst (or best) case for each memory operation.

From Figure 4.22 it is noticed that the write '1' operation tends to be faster than the write '0' one; this may be due to different reasons, such as the dependence of the TMR (defined in subsection 1.2.1) on the bias voltage, or the difference among positive/negative voltage amplitudes in output from the RBL write drivers.

[16]For the reasons explained in subsection 4.4.1.

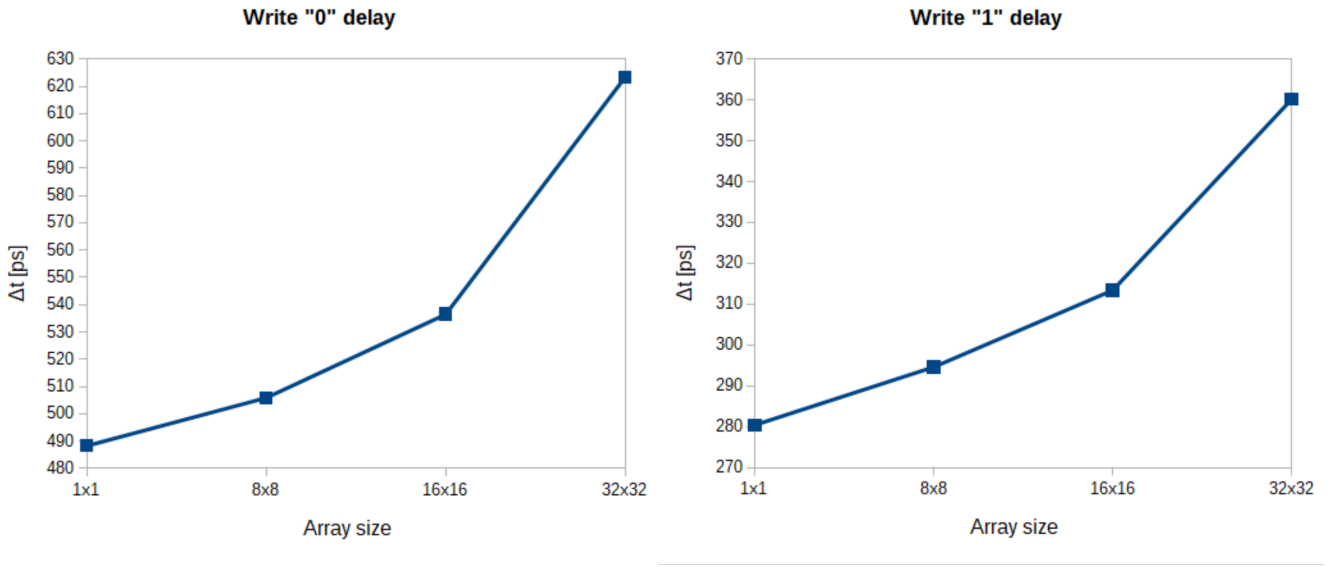


Figure 4.22. Delays results related to write '0' and '1' operations, for different array sizes

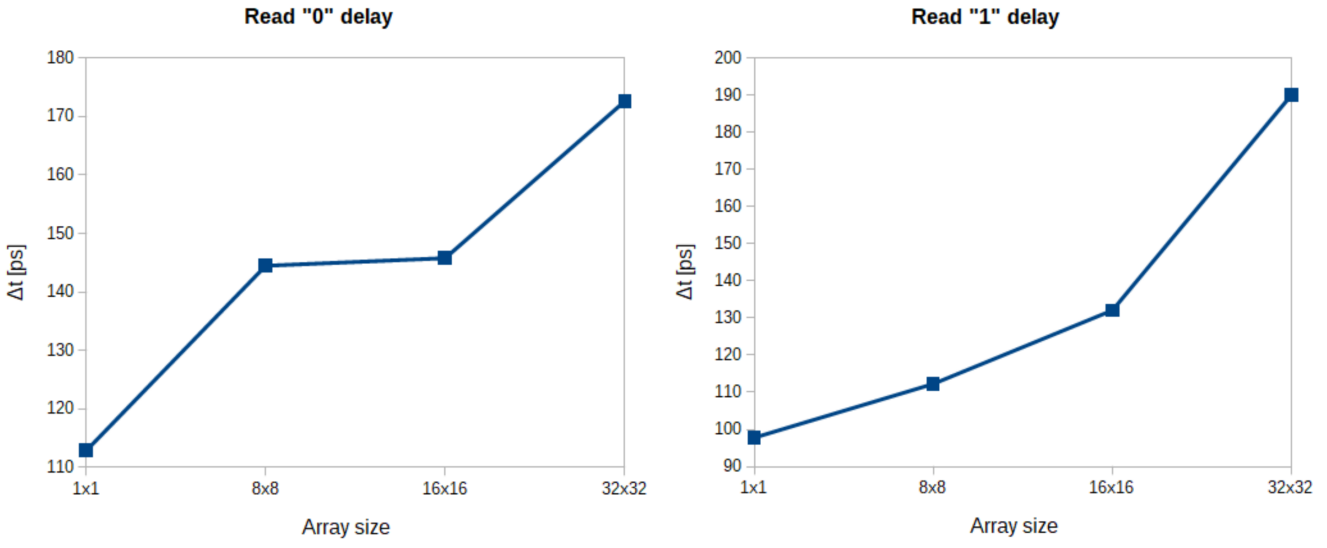


Figure 4.23. Delays results related to read '0' and '1' operations, for different array sizes

The Figure 4.23 confirms the asymmetry of reading operations discussed at the beginning of this subsection. The read delay, in particular, is heavily

dependent on the *WL–to–SE delay* (discussed in subsection 4.3.2), that increases for larger arrays.

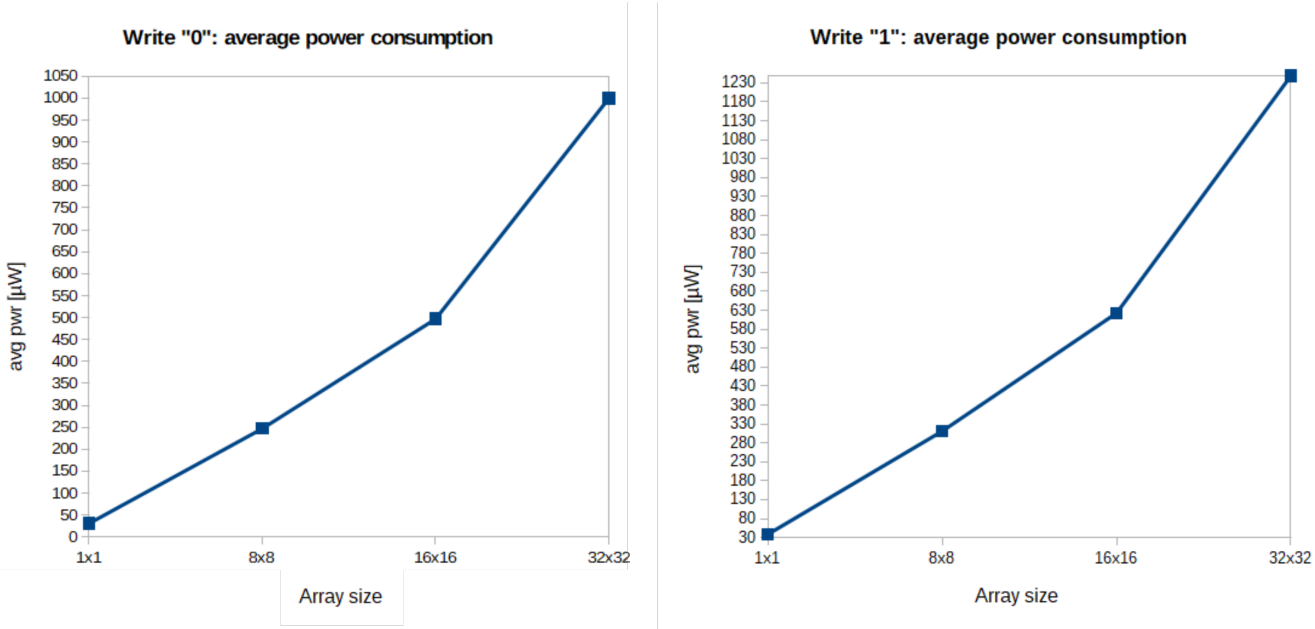


Figure 4.24. Average power consumption related to both writing cases, for increasing array size

The average power consumption during writing operation (Figure 4.24) is similar for both '0' and '1' cases, and it is higher than the one associated to the reading operation. This is predictable since the writing pulses (SOT, STT) required for switching the MTJ state have an higher amplitude with respect to the STT one used for reading the logic state: hence, the (WBL, RBL) write drivers spend much more energy than the (RBL) read drivers. The MTJ itself requires a certain amount of power for changing state, that is taken into account when following the procedure for power calculation discussed in (subsection 4.4.2).
The same considerations can be done referring to Figure 4.26 and Figure 4.27.

The output data extracted by Cadence when computing the power consumption include also the average power associated to the V_{dd} and V_{ss} rails, evaluated by computing the $V \cdot I$ product of the relative dc voltage generators. Such values may be interesting for noticing, for example, the amount of power consumption associated to the negative voltage rail during a write

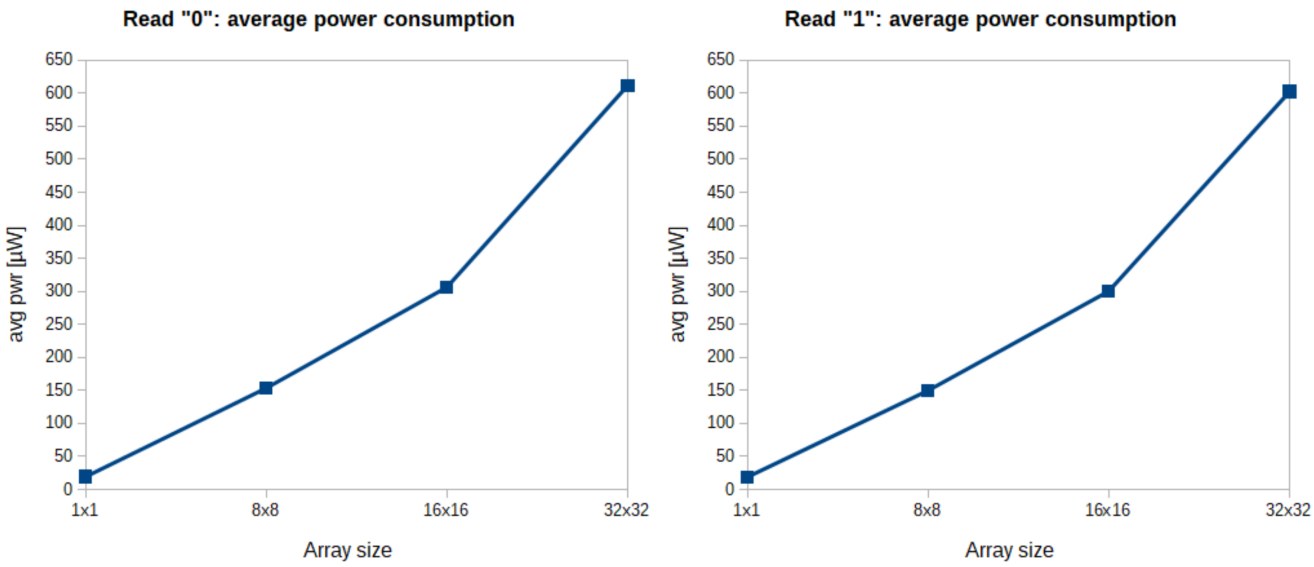


Figure 4.25. Average power consumption related to both reading cases, for increasing array size

'0' operations, due to the fact that in such case the STT pulses (driven by the write peripherals that are connected to V_{ss}) have a negative amplitude.

The peak power (Figure 4.28, Figure 4.29), as discussed in subsection 4.4.2, is dependent from the rising and falling time of the signal involved in the operation. In order to maintain it in a reasonable range, a fall_ris_time = 50 ps is used in all the simulations. Otherwise, if fall_ris_time is decreased downto 1 ps, the peak power can reach maximum values around 20 mW.

In general, the designed SOT STT memory has good performance for what concerns the speed of the operations (especially in the read case), that is comparable to a CMOS Static RAM. The main weakness of such design is the great amount of power required for performing the memory operations, particularly for writing the SOT STT cells. This may be due to technology itself or the selected point of work, that is chosen with the aim of optimizing the operations speed.
However, the energy values seem to be coherent to the ones reported in Table 1.2 and Table 1.3, for what concerns the "write energy" field of a single memory cell.

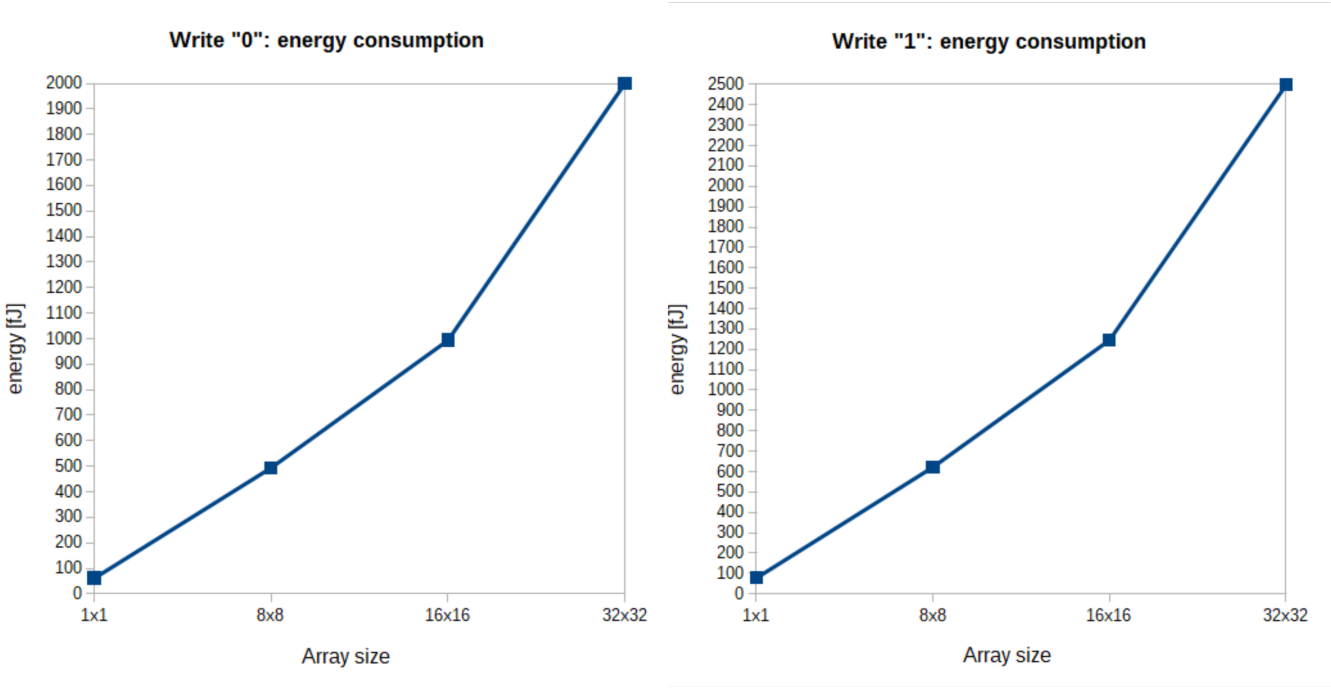


Figure 4.26. Energy consumption evaluation for write '0' and '1' cases

Last but not least, from Cadence Virtuoso are extracted also the power consumption results relative to a single cell (in all the array) that performs the selected operation. This is done in order to highlight the leakage power due to all the unselected cells of the array. As it can be seen from Figure 4.30, the leakage contribution is small, also because it is limited by the fact that both the discharge signals (on the RBL, subsection 4.3.2, and WBL subsection 4.3.3) are activated for all the other unselected columns, eliminating any residual bias from the lines.

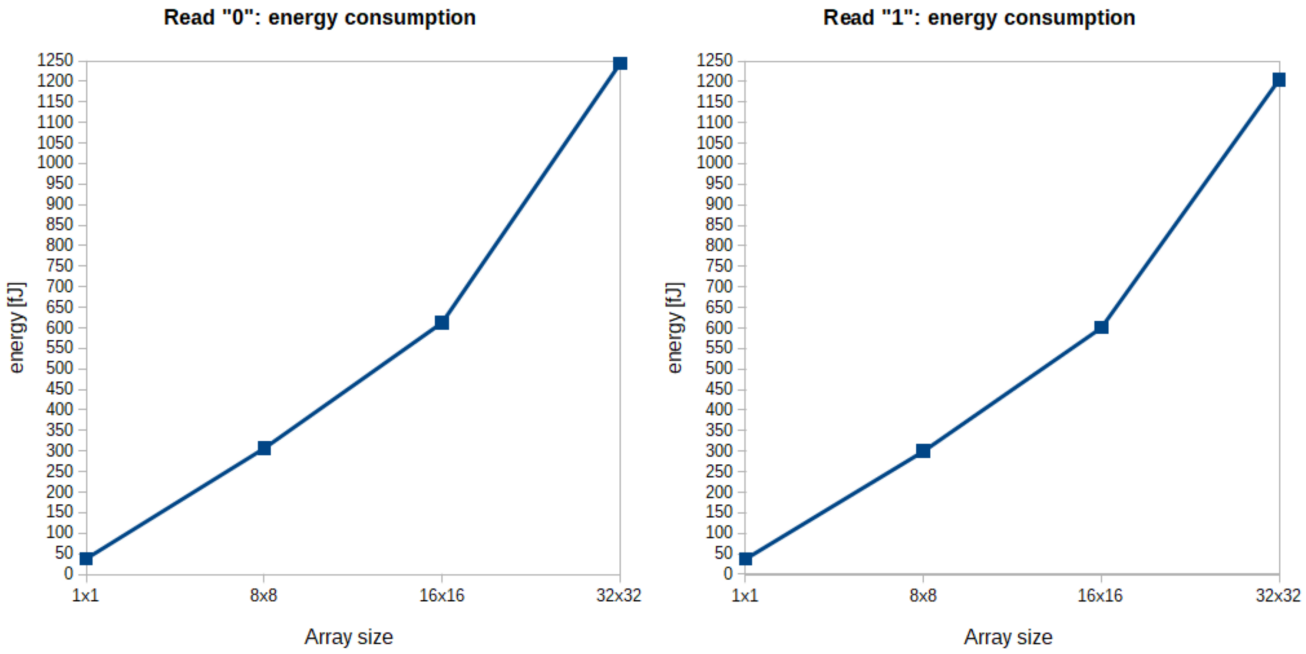


Figure 4.27. Energy consumption evaluation for write '0' and '1' cases

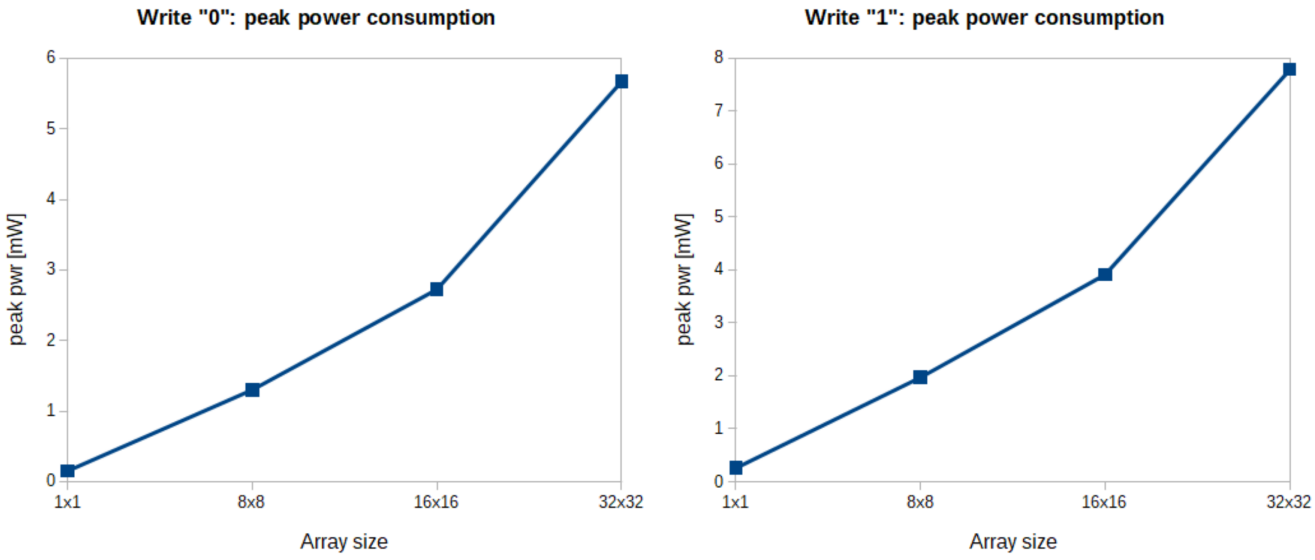


Figure 4.28. Peak power related to both writing cases, for different array sizes

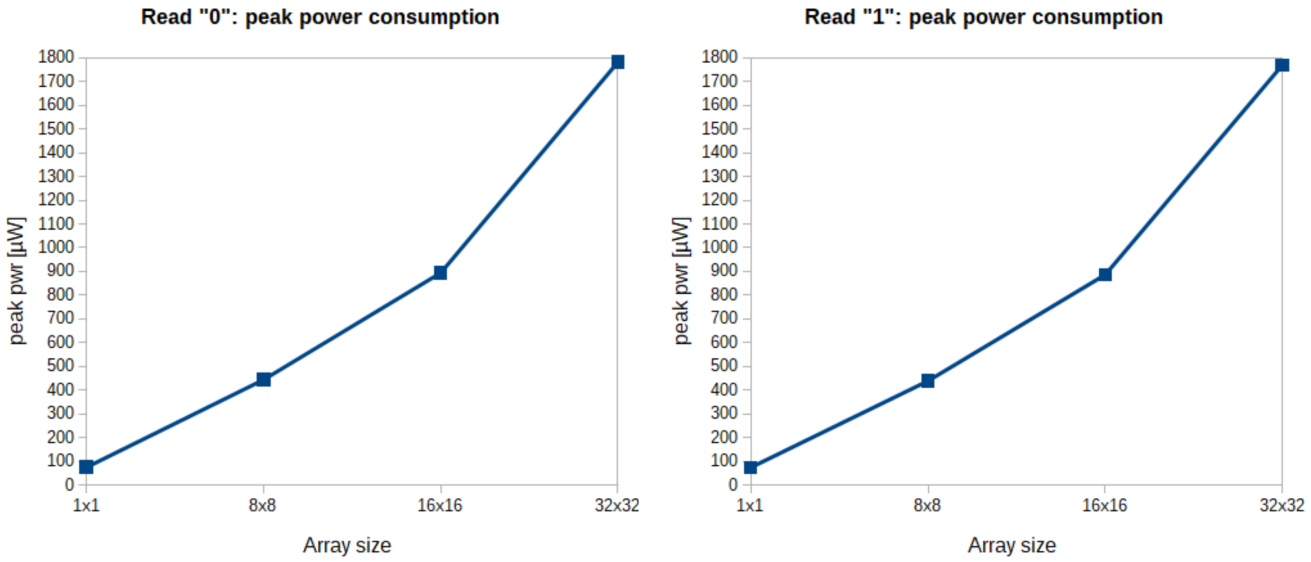


Figure 4.29. Peak power related to both reading cases, for different array sizes

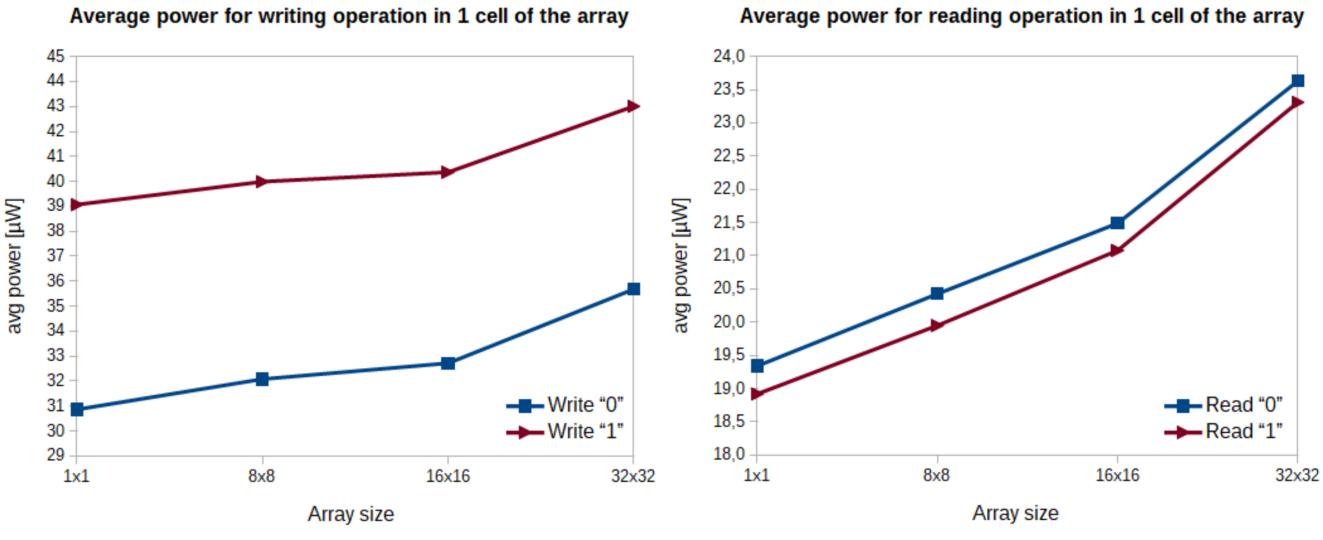


Figure 4.30. Only one cell in the whole array is selected and performs the specified operation: this is done in order to highlight the average leakage power that increases for higher array sizes

Chapter 5

SOT STT Logic–in–Memory implementation

5.1 Logic approaches from literature

This chapter discusses possible Logic–in–Memory solutions with the SOT STT magnetoresistive technology discussed in chapter 4, starting from logic approaches found in literature for *STT–only* or *SOT–only* based memories (section 5.2); in the same section are also proposed potential implementations with the SOT STT memory designed in section 4.3.
Then, it is presented an algorithm of interest (section 5.3), together with a SOT STT implementation (subsection 5.4.2) inspired by a memristive approach found in literature (subsection 5.4.1); finally, it is proposed an entirely new approach for the implementation of the target algorithm (section 5.5).

5.2 Logic approaches from literature

The SOT STT MTJ–based memory is an hybryd of the more known *STT–only* or *SOT–only*[1] magnetoresistive technologies. Hence, with the exception of the article related to the model [92], no specific approach for the introduction of logic in this type of technology has been found in literature. However,

[1] more frequently named simply as *STT* or *SOT*

since it is a magnetoresistive technology, in theory all the approaches presented in section 3.2 for a STT–only memory can be adopted for introducing logic capabilities.
Nonetheless, due to the higher structural affinity (3–terminals component), SOT–only memory is further investigated for searching proposed logic approaches in literature. It is found that the proposed SOT LiM solutions can be subdivided into two main categories:

1. stateful[2] reconfigurable boolean logic approach [92];
2. logic–in–peripherals (mainly sense amplifiers) approach [11].

The two logic approaches are presented in subsection 5.2.1 and subsection 5.2.2 respectively, together with a proposed implementation with the SOT STT technology.

5.2.1 SOT stateful logic

The first SOT LiM approach is presented in [92]. It consists, in practice, in providing a proper sequence of writing pulses that emulate the logic inputs; the result of such combination is the final resistive state of the SOT MTJ.
According to the article, a complete set of boolean functions can be implemented through such approach: True/False, Transfer, NOT, (N)OR, (N)AND, (N)IMP, X(N)OR, etc. The operations are performed by inizializing the MTJ in a target logic state, and then re–write it according to the provided input voltages, that represents the variables *In1* and *In2*. The complete combination of initial state/writing pulses is reported in [92] (table I) for all the listed operations.

Such approach can be implemented in the SOT STT memory designed in section 4.3: the input pulses correspond to SOT and STT writing pulses, with $In1 = V_{SOT}$ and $In2 = V_{STT}$. In particular, $In1 = 0\ /\ 1$ for $V_{SOT} = 0\ /\ >0$, while $In2 = 0\ /\ 1$ for $V_{STT} <0\ /\ >0$[3]. With this configuration, all the aforementioned operations can be performed: just for illustrative purposes, (N)AND and (N)OR functions, together with the corresponding voltage pulses, are displayed in Table 5.1. The operations of Table 5.1 are also

[2] see subsection 3.4.2.

[3] The amplitude of both V_{SOT}, V_{STT} can be equal to the ones proposed in subsection 4.3.3 (Figure 4.12, Figure 4.13) for maximizing the switching speed.

tested into a memory cell for verifying its functioning, and the results confirm the feasibility of this logic approach for the designed memory.

Function	Initialization		PAP	Writing				Final state
	In1	In2		In1		In2		
AND	$1\rightarrow V_{SOT}>0$	$0\rightarrow V_{STT}<0$	$\rightarrow 0$	p	$0\rightarrow 0$ $0\rightarrow 0$ $1\rightarrow V_{SOT}>0$ $1\rightarrow V_{SOT}>0$	q	$0\rightarrow V_{STT}<0$ $1\rightarrow V_{STT}>0$ $0\rightarrow V_{STT}<0$ $1\rightarrow V_{STT}>0$	0 0 0 1
NAND	$1\rightarrow V_{SOT}>0$	$1\rightarrow V_{STT}>0$	$\rightarrow 1$	p	$0\rightarrow 0$ $0\rightarrow 0$ $1\rightarrow V_{SOT}>0$ $1\rightarrow V_{SOT}>0$	$\overline{q}$	$1\rightarrow V_{STT}>0$ $0\rightarrow V_{STT}<0$ $1\rightarrow V_{STT}>0$ $0\rightarrow V_{STT}<0$	1 1 1 0
OR	$1\rightarrow V_{SOT}>0$	$1\rightarrow V_{STT}>0$	$\rightarrow 1$	$\overline{p}$	$1\rightarrow V_{SOT}>0$ $1\rightarrow V_{SOT}>0$ $0\rightarrow 0$ $0\rightarrow 0$	q	$0\rightarrow V_{STT}<0$ $1\rightarrow V_{STT}>0$ $0\rightarrow V_{STT}<0$ $1\rightarrow V_{STT}>0$	0 1 1 1
NOR	$1\rightarrow V_{SOT}>0$	$0\rightarrow V_{STT}<0$	$\rightarrow 0$	$\overline{p}$	$1\rightarrow V_{SOT}>0$ $1\rightarrow V_{SOT}>0$ $0\rightarrow 0$ $0\rightarrow 0$	$\overline{q}$	$1\rightarrow V_{STT}>0$ $0\rightarrow V_{STT}<0$ $1\rightarrow V_{STT}>0$ $0\rightarrow V_{STT}<0$	1 0 0 0

Table 5.1. Some of the boolean functions that can be performed with stateful approach into the designed SOT STT memory. The *PAP* field refers to the MTJ state after the initialization cycle.

The main advantage of such approach in a SOT STT memory is the fact that it can exploit its fast switching for performing boolean operations. On the other hand, two write cycles (initialization+writing) are necessary for performing almost all the functions[4].
However, the great weakness of the stateful logic solution is that it is destructive. It is, in practice, a writing–based approach, which overwrites the initial cell content with the function result.

This logic approach shares the same working principle of the STT–based one discussed in subsection 3.2.4, although there are some differences (serialized

[4]The basic ones, such as NOT, require only the initialization cycle.

steps, IMPLY–based approach).

5.2.2 SOT STT logic–in–peripherals approach

The introduction of logic inside the peripheral circuitry is the most widespread SOT LiM approach among the explored articles. Most of the time, the logic implementation concerns the sensing circuitry; in this sense, such approach is very similar to the one presented in subsection 3.2.2 for *STT–only* memories [11].

The basic mechanism of this type of logic is to employ different voltage thresholds (ex. $V_{ref,OR}$, $V_{ref,AND}$) as second input to the sense amplifier[5] according to the operation that has to be performed.
AND and OR functions can be implemented with a single SA that uses different resistances (R_{OR}, R_{AND}) to weigh the V_{ref} value in order to provide the wanted threshold. Their negation (NAND, NOR) are obtained at the negated output ($SAOn$) of the SA.
More complex operations, such as the XOR, can be performed by adding CMOS logic gates at the output of two SAs connected to the same column: in this way, the XOR can be computed with the combination XOR = NOR (NOR, AND) (alternatively XOR =AND(NAND, OR)).

The main advantage of this approach is that it exploits the possibility of magnetoresistive memories for multiple rows activation without loss of data; in this way, the SA represents a good mean for logic introduction, since it is shared by all the cells of a single column.
With the aim of maximize the LiM parallelism (i.e. the number of rows processed at once), a possibility would be to perform the same boolean operation to all the cells of the same row, such as an m–inputs AND gate with just a single SA. In this case, the RBL read driver (see subsection 4.3.2) should be modified in order to supply a proper sensing current, due to the high number of cells to read[6].

Another interesting improvement to such approach would be to exploit the

[5]The first one remains the V_{sense} read on the RBL, that is dependent on the MTJ state of the selected cell, see subsection 4.3.2.

[6]A m–long column with all the cells selected represents a parallel of m resistances, that behaves as current divider for the I_{sense}.

SA output to perform a writing operation, depending on the result of the logic operation. This approach has been successfully tested with a bipolar–supplied SA, taking into consideration a couple of SOT STT cells; with the logic state of the two cells as input of two sense amplifiers (for performing the aforementioned XOR), a third cell can be written with the function result by exploiting the same read cycle.
This is achieved by connecting the XOR (2 SAs + CMOS gate) output to the RBL write driver (as enable signal) (see subsection 4.3.3), which performs a write '0' or '1' on the third cell according to the XOR result. According to the requirements of the write driver, the XOR output must be bipolar as well (i.e. XOR = V_{ss} for logic '0' and XOR = V_{dd} for logic '1', as shown in Figure 5.2) for enabling a writing operation.
The proposed design for such approach is reported in Figure 5.1.

In Figure 5.2 are reported the waveforms of interest for showing the correct working of the schematic. At the top of the figure are displayed the four combinations of magnetization states of the two SOT STT cells. Then, in the middle, the related output of the designed XOR. At the bottom, it is depicted the magnetization state of the third cell (Tmz3), for an initial logic state '0' ($PAP = 0$) in all the cases. The same results can be obtained for PAP =1.

5.3 The target algorithm

In this section it is presented an algorithm of interest, that has to be performed though LiM approach with the same technology discussed in chapter 4.

The algorithm is taken from [97]. It consists in a language classifier based on the so–called *hyperdimensional computing*: highly dimensional vectors (*hypervectors*, with dimensions in the thousands) are exploited to encode samples of text for language recognition.

From a practical point of view, the aim is to demonstrate the feasibility of such approach without operating with so large vectors. This is mainly due to the fact that the algorithm will be implemented by hand through Cadence Virtuoso, which is time–consuming (in terms of schematic setup and simulations) already for a "small" 32x32 array. However, the proposed solutions

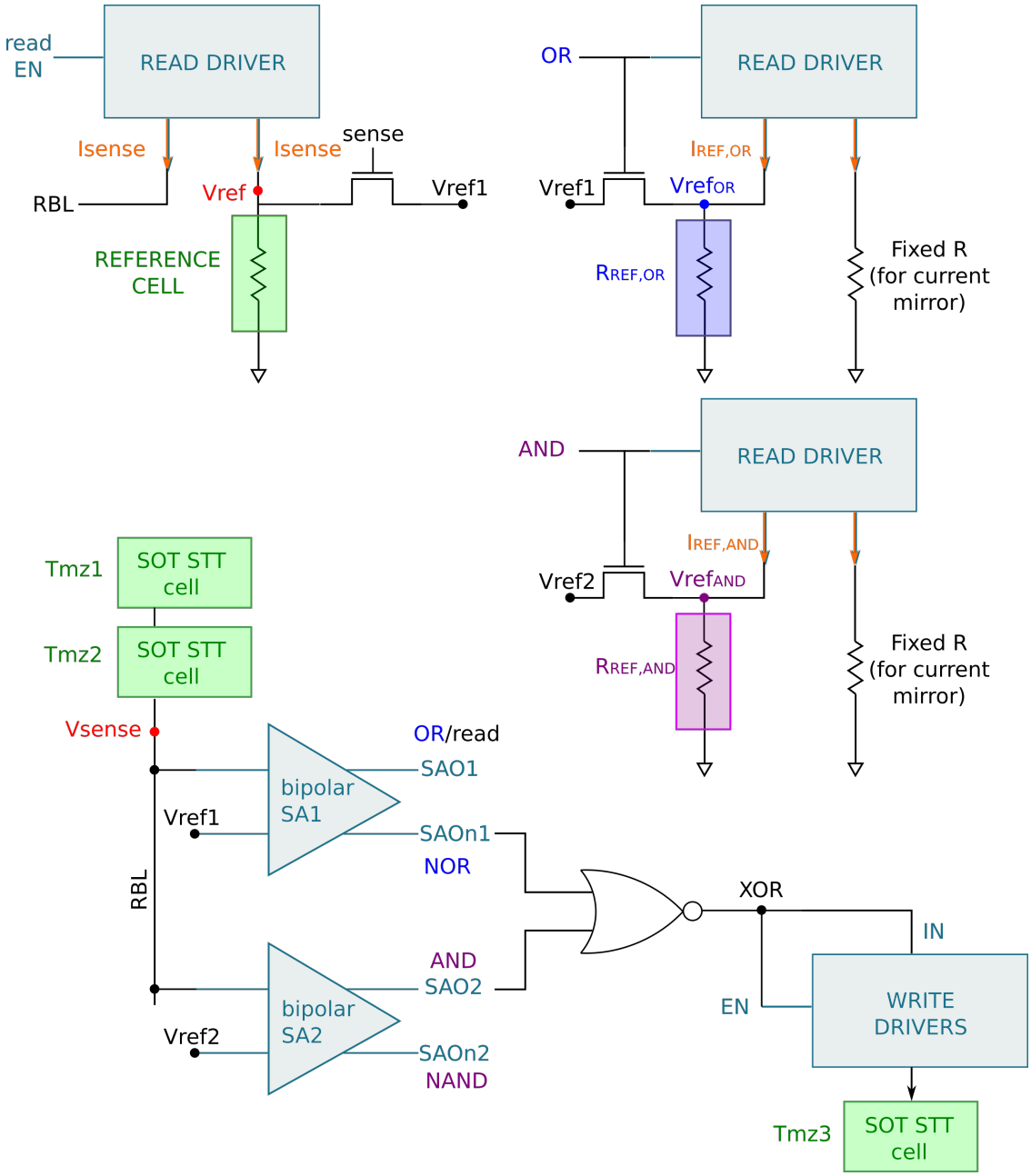


Figure 5.1. Design of the SA–based logic approach implemented in Cadence Virtuoso for implementing (N)OR, (N)AND, and XOR between two SOT STT cells (Tmz1,2) with writing of a third one (Tmz3) with the XOR result in the same cycle The reference SOT STT cell is the one designed in subsection 4.3.1.

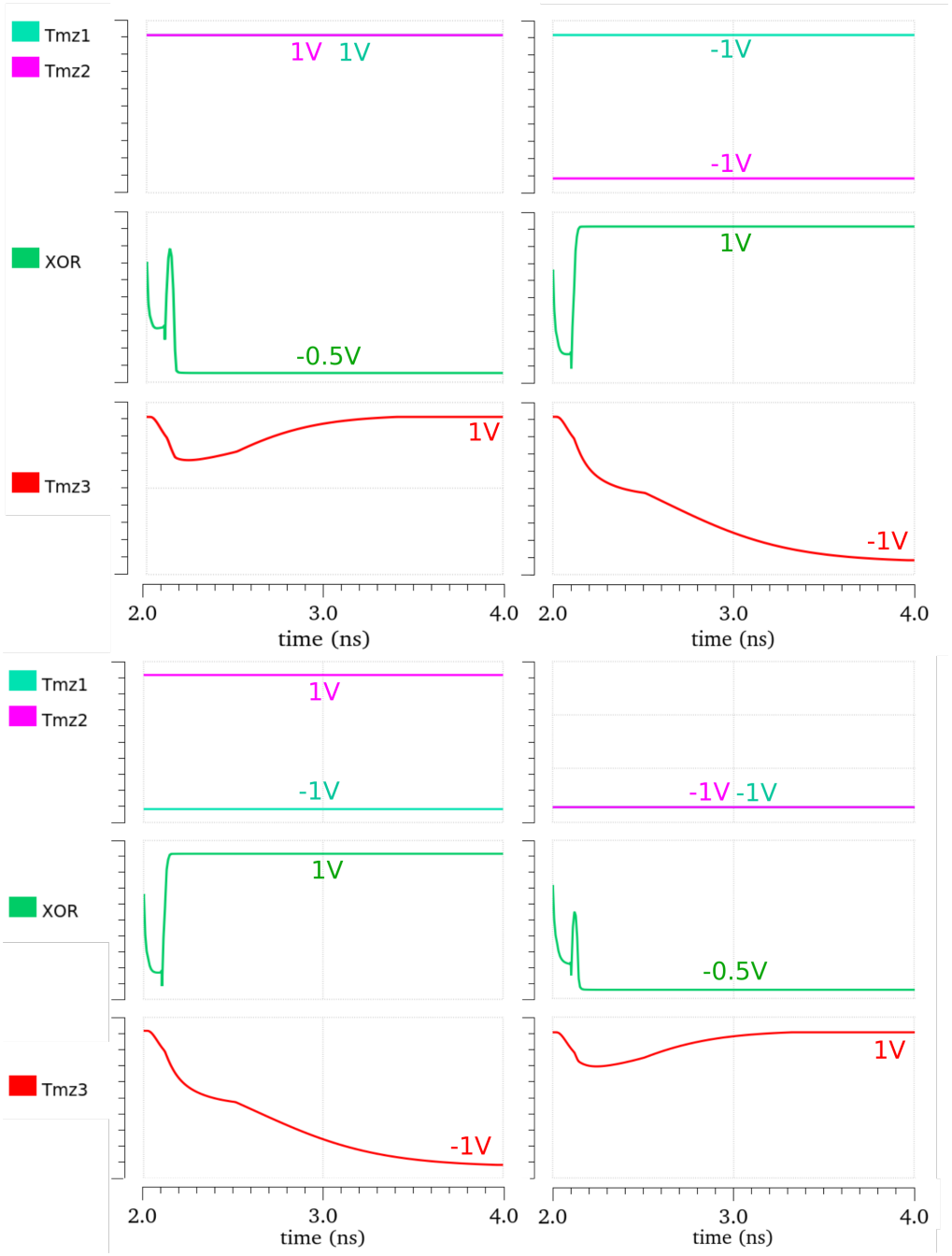


Figure 5.2. Cadence simulation plots related to the implementation of the SA–based XOR (all cases) between a couple of SOT STT cells (Tmz1,2), and writing of a third cell (Tmz3) with the XOR result exploiting the same cycle.

take into account the prospect of highly dimensional vectors, making considerations by observing the results of increasing array sizes (1x1, 8x8, 32x32).

In particular, the classifier is divided into two main modules: the *encoder* and the *search module* (Figure 5.3).
The first one is responsible for the coding of a series of *trigrams* (i.e. a group of three letters) into a final *text hypervector*. The letters (corresponding to randomly associated hypervectors) are given in input to this module one at time; the encoder performs a series of multiplications ($\cdot$) and permutations (ρ), for obtainining a unique final *trigram hypervector* that takes into account the order with which the letters are provided (ex. trigram A–B–C is different from B–A–C). The multiplication between D–long hypervectors is computed through D XOR gates[7], while the permutation consist in a rotation of the hypervector in the space: the hypervector undergoes a right shift by 1 position of all its components. A double–rotated hypervector ($\rho\rho$(L)) is shifted to right by two positions.
The resulting function performed by the encoding module is $\rho\rho(\mathrm{L}_1)\cdot\rho(L_2)\cdot\mathrm{L}_3$ for the trigram L_1–L_2–L_3. All the trigrams hypervectors are then summed up in a text hypervector. Such vector represents the query hypervector in input to the search module.

The search module has the function to find the text sample that has the closer similarity with respect to known languages. In practice, the module stores a group of so–called *language hypervectors* (one for each row) that are pre–computed by the encoding module during an initial training phase; they simply consists in text hypervectors created from known languages.
The search operation consists in a comparation of the encoded query hypervector with respect to all the language hypervectors.

The same article ([97]) proposed a method for finding the closer query–to–language match based on the estimation of the so called *Hamming Distance* (HD). This quantity can be defined as the number of bits of which two binary number (in this case, hypervectors) differ.
The HD is computed through a number of XOR gates equal to the dimension of the hypervectors (hence, a very big number). Furthermore, the search is

[7]Since the hypervectors under study are binary.

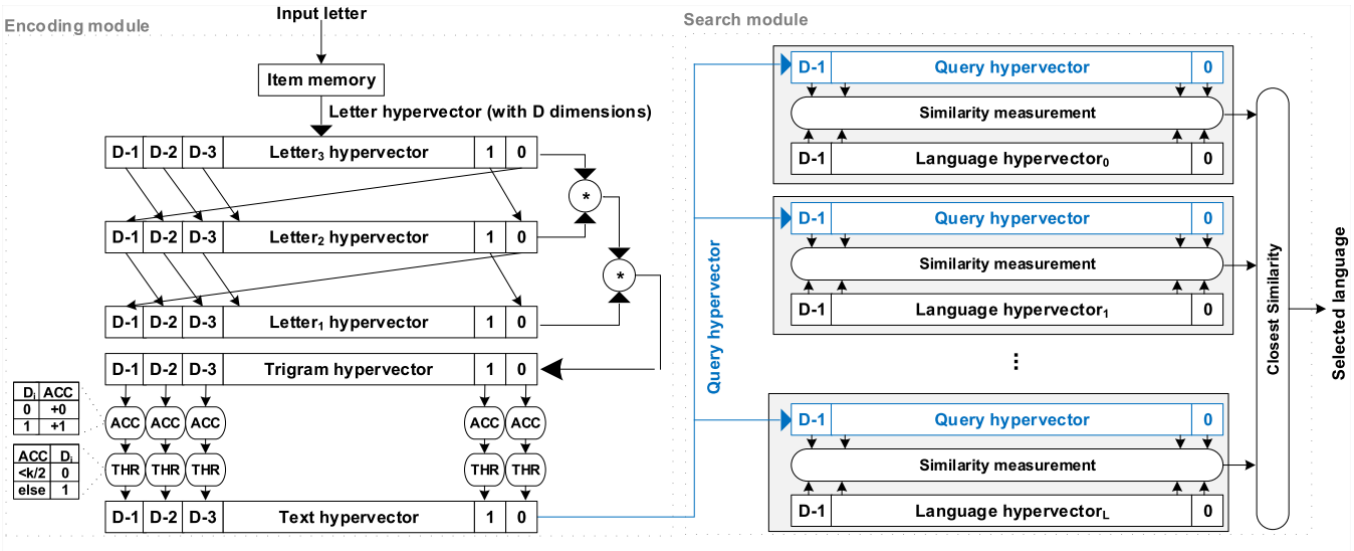


Figure 5.3. HD classifier proposed in [97].

done comparing just one element for each clock cycle, resulting in $O(D)$ cycles for counting the HD between two hypervectors. This XOR block, called similarity measurement block, is repeated in the search module a number of times equal to the number of its rows, as shown in Figure 5.3.
After all the comparisons, the search module selects the language hypervector that has the minimum HD from the query one.

The most interesting part of the algorithm to consider for a LiM approach is the design of the similarity search module. This module can be implemented through a memory storing all the pre–computed text samples. It is clear that such design, for being efficient, requires an high degree of parallelism: the aim is to realize a module that perform a XOR bit a bit in parallel in all the cells, between an external data (the query text hypervector) and a pre–stored data (the language hypervectors).

The stateful approach presented in subsection 5.2.1 is a writing–based logic: this means that the XOR operation — required for computing the HDs — would be performed in one row at time, covering a number of cycles equal to $(N_{rows}) \cdot (t_{writecycle} + t_{initializationcycle}$[8]$)$.

[8]explained in section subsection 5.2.1

Moreover, the XOR would be computed between two external data (writing pulses) that would overwrite the search module content with the result of the operation.

Also the peripheral–based logic discussed in subsection 5.2.2 does not represent the best solution: even if non destructive, such approach would require $(N_{rows}) \cdot (t_{readcycle})$ time for search in all the memory. Furthermore, since a SA-based logic requires the operands to be stored in the same columns, it would be necessary to store the external data (query hypervector) in the search module as well. The "good" news is that the SAs (two for each column for performing an XOR operation) can be re–used for XOR operation in all the rows of the memory: hence, there is no need for replicating them N_{rows} times as the aforementioned similarity measurement block.

A good idea for maximising the XOR parallelism would be to introduce logic at cell level: similar approaches are usually based on the fact that one or more transistors, placed inside each cell, can be activated or turned off depending on the logic state of the storing element.
In this sense, the magnetoresistive technology proposed in chapter 4 has some constraints: a single SOT STT MTJ can provide a low voltage dynamic[9], limited by the TMR[10] and the fact that too high voltages may cause write errors. This is also the reason for which almost all the logic approaches related to STT and SOT technologies are based on introducing logic at peripheral level or exploiting destructive writing–based solutions (see subsection 3.2.4, subsection 5.2.1).
These limitations are in part mitigated when working with Content Addressable Memory (CAM) cells: since their basic cells includes a couple of storing elements (ex. two memristors in the design proposed in subsection 5.4.1, two SOT STT MTJs in subsection 5.4.2) the aforementioned voltage dynamic can be increased to exploitable values.

The first proposed design (subsection 5.4.2) — for implementing the presented algorithm — takes inspiration from the memristive logic approach discussed in (subsection 5.4.1), but with some modifications (in primis, the

[9]Not sufficient for turning on/off a transistor.

[10]The Tunnel MagnetoResistance is defined in subsection 1.2.1.

different technology requirements). As described in subsection 5.4.3, however, it remains a limited approach for different reasons.
A second design is instead discussed in section 5.5; the efficiency of such approach is discussed in subsection 5.5.2.

Both the proposed solutions are based on CAM arrays. CAMs, in fact, are capable of searching an input word in parallel to all the memory: the so–called *Match Line* (ML), that links all the cells of the same row, is discharged to ground or is kept at a pre–charged voltage depending on the result of the bit–a–bit "match" (i.e. if the word bit is equal to the cell content). In this way, the ML that remains in high voltage state is the one associated to the "all–matches" case, hence it corresponds to the row containing the searched word.
The match (or search) operation can be thought as a XNOR operation: if the input and stored data are different, the ML is discharged to logic '0', otherwise it remains at '1'.

In this sense, a CAM works as a *zero Hamming Distance finder*: it is capable of locate only the row associated to a 'HD = 0', that corresponds to the high–state ML (all matches). On the other hand, the search module design intended by the algorithm has to work as a *minimum Hamming Distance finder*: the language hypervector (stored in the memory) with the closer similarity to the query one corresponds to the row with the maximum number of matches.

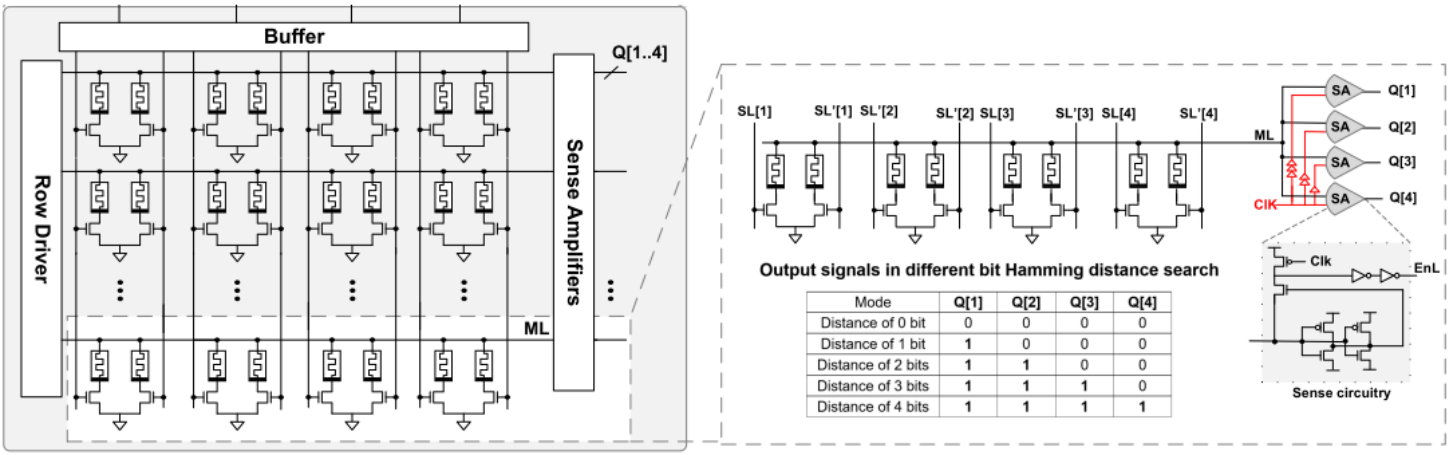
In the proposed approaches (subsection 5.4.2 and section 5.5) it is designed what can be called as a *Hamming Distance counter*, that in practice implements the core function of a *minimum HD finder*: in fact, the fist step has to be the computation (counting) of all the HDs of each row, and only then it can be chosen the best suited minimum–search algorithm.

5.4 Timing–based HD counter

This section discusses the memristive CAM–based HD counting method found in literature (subsection 5.4.1), and a proposed SOT–STT CAM design (subsection 5.4.2) for verifying the feasibility of such approach with magnetoresistive technology.

5.4.1 Memristive CAM SA approach

As anticipated in section 5.3, this approach is based on a memristive CAM array presented in [98]. It exploits the idea that the CAM MLs discharge faster or slower depending on the number of matches in each row: in this sense, it can be considered as a "timing–based" HD counter.
The discharging time is measured through a set of SA attached to the same row, each responsible for identifying a certain HD. In fact, SAs[11] work as threshold comparators for a given time instant: when the sense enable (SE) is activated, they amplify the two voltages that are in input in that precise moment. Any input variation after the SE activation (during the same read cycle) is not considered. Hence, each SA is able to detect different numbers of mismatches (i.e. the HD) if its enable signal is properly delayed.
The reference design is reported in Figure 5.4.



Mode	Q[1]	Q[2]	Q[3]	Q[4]
Distance of 0 bit	0	0	0	0
Distance of 1 bit	1	0	0	0
Distance of 2 bits	1	1	0	0
Distance of 3 bits	1	1	1	0
Distance of 4 bits	1	1	1	1

Figure 5.4. Memristive CAM discharge approach [98].

This working principle is applied to the memristive CAM by dividing the array in 4 bit wide blocks, and connecting 4 SAs for each row of such blocks: in this way, each SA samples 1 bit of HD. Then, all the SAs outputs are connected to a counter located at the end of each memory row. The counter will take in account the sum of all HDs of the 4–bit wide blocks, and a final set of comparators will evaluate the lower HD among all the rows.

[11] It is to be intended the latch–based sense amplifiers; ex. differential amplifiers have a different working principle.

Such approach is clearly hardware expensive: in order to count all the bit mismatches of a single row, it is required one SA for each memory cell. However, it is worthwhile to test its feasibility with the SOT STT magnetorestistive technology discussed in chapter 4; furthermore, it may be interesting to observe the behaviour of the MLs discharging when considering blocks larger than 4 bits (ex. 32 bit long rows), trying to proposing new solutions.

5.4.2 SOT STT CAM adaptation

In order to test the working of the approach proposed in subsection 5.4.1 with SOT STT technology, it is necessary, first of all, to design a SOT STT CAM cell.

A SOT STT MTJ shares a higher structure similarity with respect to SOT MTJs (three terminals, heavy metal layer presence) than to STT MTJs[12]. For this reason, it would be very useful to start from a SOT CAM cell for implementing a SOT–STT CAM. However, no articles are found in literature proposing a SOT CAM design, and very few related to STT CAMs: one of them [99], however, is very useful in order to understand the basic working principle of magnetoresistive CAM, that is displayed in Figure 5.5. The two variable resistances represent two STT MTJs, one in opposite logic state with respect to the other. They represent a voltage divider for the differential voltage on the search lines (SL and SLn). If V_H and V_L are the high and low voltages in input to the search lines, the voltage at point Vo can be computed as:

$$\begin{aligned} Vo_H &= V_{\mathrm{H}} + \frac{V_L - V_H}{2 + TMR} \\ Vo_L &= V_L + \frac{V_H - V_L}{2 + TMR} \end{aligned} \tag{5.1}$$

where $Vo_H > Vo_L$, TMR defined in subsection 1.2.1.
The obtained Vo dynamic may be quite small, or insufficient to completely turn on/off the nMOS transistor, depending on the MTJ TMR.

[12]Both STT and SOT MTJs structures are reported in Figure 1.6

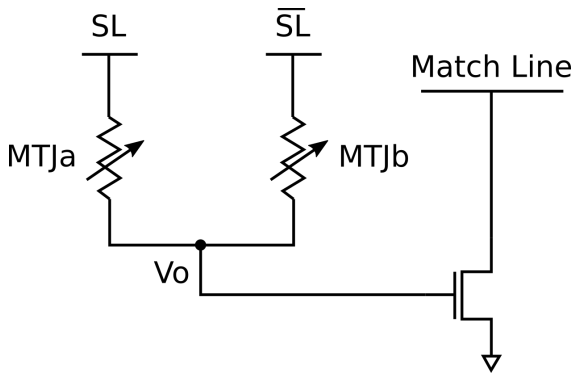


Figure 5.5. STT CAM working principle [99]

For this reason, in [99] it is proposed the introduction of a couple of pMOS/nMOS transistors between the two MTJ in order to amplify the Vo dynamic. This idea is based on the R_p/R_n dependence on the V_{gs} of each transistor, that at its time depend on the search lines voltage and the MTJs resistance. Unfortunately, this approach has not been successfully realized since, when introduced in the circuit, the pMOS presents a much higher resistance than the nMOS one, even if the pMOS is set to a larger size. It is as if, in both high and low SL input cases, a residual V_{gs} on the pMOS does not allow it to turn on completely: hence the voltage partition remains unbalanced for the high and low logic states, preventing a correct voltage dynamic amplification.
This does not represent a real problem since the the SOT STT MTJs have a sufficient TMR for turning on/off the nMOS ML transistor, even if the on–switching is slower with respect to a properly biased transistor. Indeed, the slower switching is useful for the implemented timing–based approach, since it allows a better distinction among ML discharging curves. However, for the new CAM cell designed in section 5.5, an inverter stage — which introduces the same number of transistors of the approach proposed in [99] — is linked to the Vo point for amplifying the voltage dynamic.

The (first) SOT STT CAM cell design is shown in Figure 5.6, considering the resistive equivalent of the two SOT STT MTJs. The cell structure is "similar" to the one proposed in the last chapter (Figure 4.6) in the sense that a single WL signal turns on/off the access transistors on the write and read bitlines. The RBL is split into two lines, which allows to provide the correct write pulses to both the MTJs. The nMOS between the second MTJ

and the SL[13] allows to detach the ground reference from the cell when a search operation is performed: in this way, it is obtained almost the same circuit as the one in Figure 5.5, for $search\_n = 0$. The nMOS connected to the ML is activated only when the pre_n signal is high, hence when the ML precharging phase is concluded.

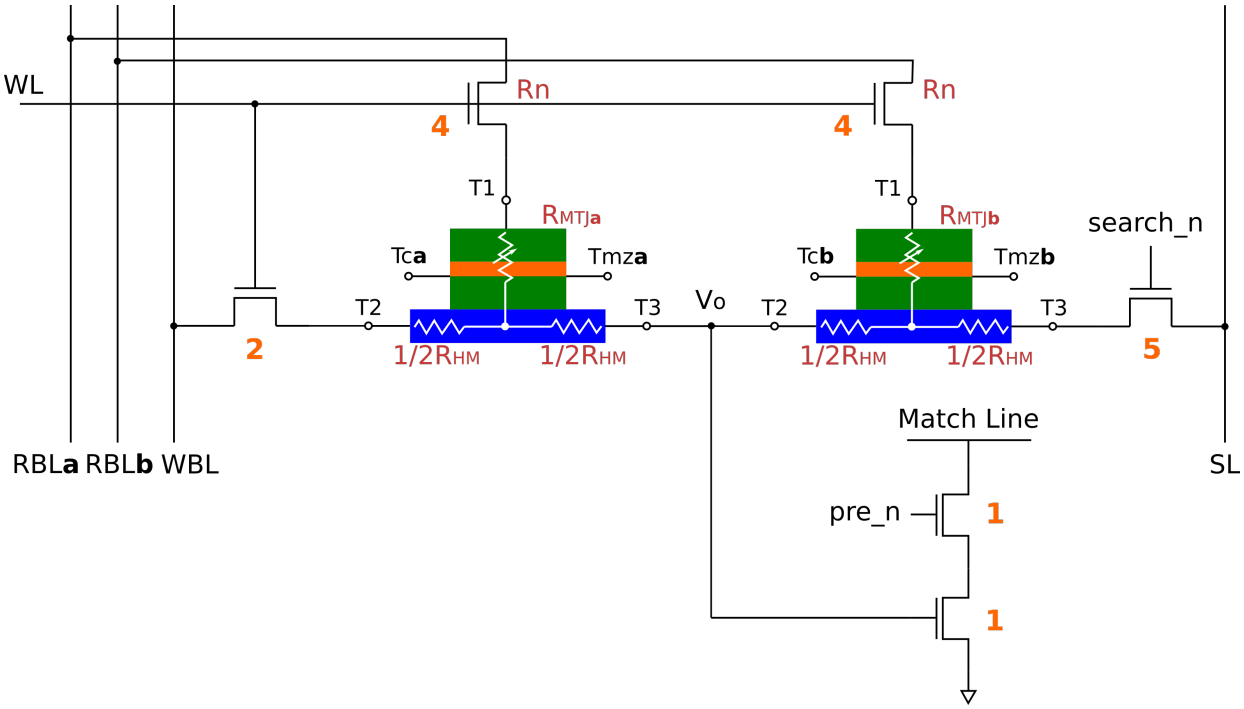


Figure 5.6. SOT STT CAM cell designed for implementing the timing–based approach discussed in section 5.4, with transistors sizing and MTJs resistive equivalent

The cell is able to perform writing and reading operations in a similar way with respect to the cell presented in subsection 4.3.1, by exploiting the same reading interface (SA, read driver in subsection 4.3.2) and WBL write driver (for the SOT pulse generation, subsection 4.3.3); the STT write voltage pulse, this time, is provided by ideal voltage generators on both RBLa and RBLb lines. With the proper pulses duration (0.5 ns for SOT and 2 ns for STT, exactly as done in Figure 4.5) and voltages amplitude (for SOT

[13]The SL is different from the one in Figure 5.5, since, as in Figure 4.6, it is used to provide the ground reference to the memory.

see Figure 4.13, while for STT $-200\,\text{mV}/500\,\text{mV}$ or $500\,\text{mV}/-200\,\text{mV}$ on RBLa/RBLb for write '0'/'1' cases) a writing operation can be performed simultaneously on both MTJs. In fact, for SOT STT CAM cells the two MTJs must be in a differential configuration (i.e. in opposite resistive states) for representing a logic state; in particular, a logic '0' corresponds to the first MTJ in LRS[14] ('0') and the second one in HRS[15] ('1'), viceversa MTJ_1 = HRS and MTJ_2 = LRS for a logic '1'. Hence a write '0' operation, for example, writes a '0' in the first MTJ, while the second one is written to '1'; a write '1' operation writes '1' and '0' in the first and second MTJ, respectively.

The input voltages and currents used for writing and reading the cell are summed up in Table 5.2: for these operations, the *search_n* signal is kept high, in order to provide (through SL) the proper ground reference to the cell circuit.

Line	**Operations**		
	Write '0'	**Write '1'**	**Read**
WL	V_{dd}	V_{dd}	V_{dd}
RBLa	$-200\,\text{mV}$	$500\,\text{mV}$	I_{REF}
RBLb	$500\,\text{mV}$	$-200\,\text{mV}$	0
WBL	$\sim 800\,\text{mV}$	$\sim 800\,\text{mV}$	floating
SL	0	0	0

Table 5.2. List of the involved voltages and currents for writing and reading operations; the RBL pulses are provided by ideal voltage generators, while WBL ones are generated by the WBL write driver shown in Figure 4.13. For both writing and reading, the *search_n* signal is kept equal to V_{dd}.

The high and low Vo states referred to Figure 5.6 can be computed as:

$$\begin{aligned} Vo_H &= V_H - (V_H - V_L) \cdot \frac{R_n + R_{MTJa,L} + R_{HM}}{R_{MTJa,L} + 2R_{HM} + R_n + R_{MTJb,H}} \\ Vo_L &= V_L + (V_H - V_L) \cdot \frac{R_n + R_{MTJa,L} + R_{HM}}{R_{MTJa,L} + 2R_{HM} + 2R_n + R_{MTJb,H}} \end{aligned} \tag{5.2}$$

Since it is evident the Vo_H,Vo_L dependence on the access nMOS resistances, such transistors must be sized properly for maximising the Vo dynamic. The

[14]Low Resistance State.

[15]High Resistance State.

designed sizing is reported in Figure 5.6.
The four match cases can be realized through the signal configuration reported in Table 5.3.

In1	In2	In1		In2		Vo	ML
		PAPa	PAPb	V_RBLa	V_RBLb		
0	0	0	1	0	500 mV	L	1
0	1	0	1	500 mV	0	H	0
1	0	1	0	0	500 mV	H	0
1	1	1	0	500 mV	0	L	1

Table 5.3. Search (or match) operation for all the possible input combinations. The two inputs are the initial logic state of the MTJ (*PAP*) and the voltage pulses configuration on the RBLs (V_RBLa, V_RBLb). For 'H' and 'L' in the Vo field are intended the high and low extremis of Vo dynamic; the ML field reports the logic value associated to the voltage on the match line ('0' corresponds to 0 V, while '1' corresponds to the ML pre–charge voltage).

Practical results can be observed in Figure 5.7, where is shown the behaviour of the pre–charged ML related to a single cell for all the cases listed in Table 5.3. In order to obtain it, the voltage pulses on RBLa, RBLb must be 4 ns–long; if shorter (ex. 2 ns) the match is performed correctly but a higher bias voltage ($\sim$ 100 mV) remains as residual at the end of the 'ML = 0' cases. This is due to the aforementioned "slow" ML discharge: it is however an advantage for this timing–based method, since the ML goes in input to a buffer (shared with all the cells of the same row) that works as amplifier: the buffer output switches to '0' (for all the cases except from the "all matches" case) at different time instants, that are further apart from each other (for all the HD cases) when the ML discharge is slower.

For the single cell simulation, a match ('ML=1') between the initial MTJs state and the RBL voltage pulses corresponds to a zero Hamming Distance contribution[16], while a mismatch ('ML=0') represents 1 bit contribution to HD.

The logic approach is then tested in rows of 8 bits and 32 bits; the row

[16]In the sense that, when the cell will be introduced in a row, its contribution to the total HD is zero.

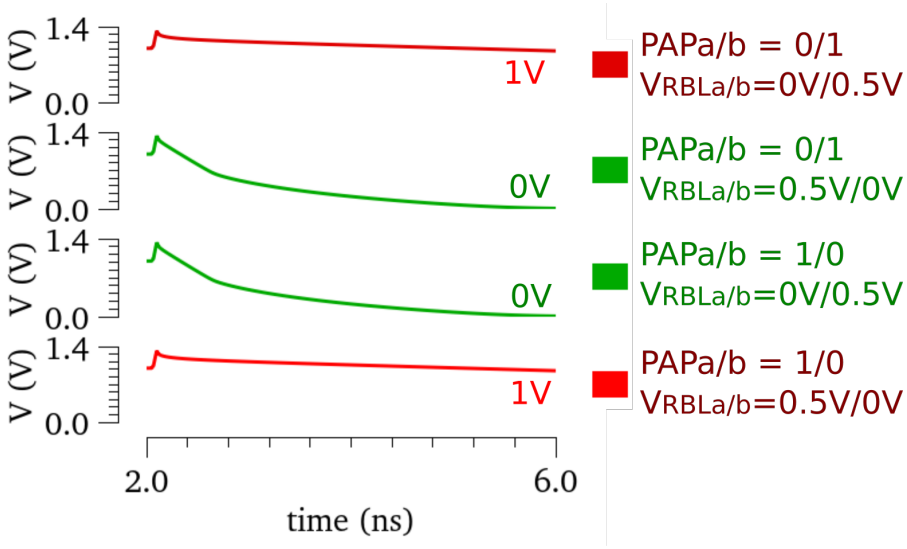


Figure 5.7. Cadence results extracted from a parametric analysis for all the possible match cases, corresponding to Table 5.3.

structure is qualitatively shown in Figure 5.8. This is done for analyzing the effects on the ML for rows longer than the 4 bits one proposed in the reference memristive CAM article [98].

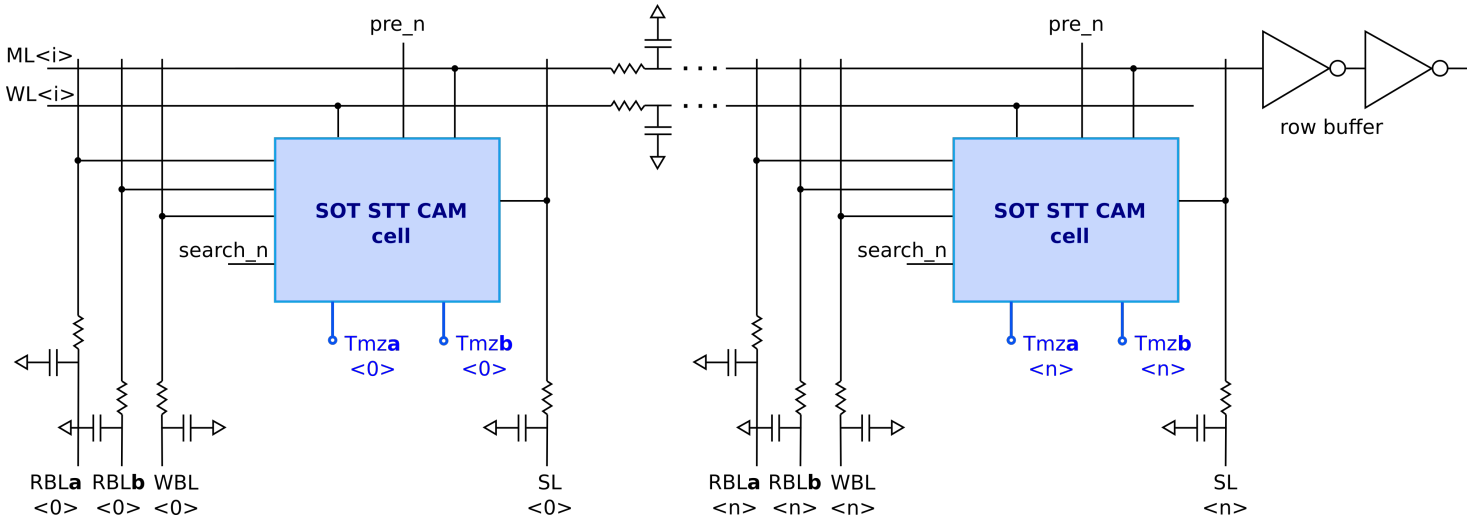


Figure 5.8. SOT STT CAM row organization (design suited for the timing based–approach).

In Figure 5.9 it is displayed the Cadence simulation result for a parametric

analysis on a 8 bit row: the RBL voltage pulses (*in2*) are made varying in such a way that all the possible HD cases (from HD = 0 to HD = 8) are reported in the plot. In the upper part, it is shown the behaviour of the pre–charged (to 500 mV[17]) ML, for the different discharging times: in almost all the cases the ML do not reach the 0 V voltage in the 4 ns interval, but this is not a problems, since the ML is fed to an buffer (a double inverter located at the end of the row). The output of such buffer is displayed in the bottom part of Figure 5.9, where the different discharging times are well defined for all the HD cases.
Hence, an 8 bit row still performs well with respect to the presented logic approach.

The timing–based approach is then implemented in a 32 bit row.
Unsurprisingly, the ML discharging curves associated to the different HDs become more and more closer to each other. This is problematic for a timing–based approach: at this point, it is almost impossible to distinguish and count the number of mismatches, especially for the "worst cases" of near–to–maximum HDs displayed in Figure 5.10 (left).
In particular, Figure 5.10 includes a couple of Cadence simulation plots for the eight "worst" (left) and "best" (right) cases for the counting of mismatches, i.e. for HD = 24÷32 and HD = 0÷8, respectively. As for Figure 5.9, the upper part of Figure 5.10 reports the ML discharging curves, while the bottom one corresponds to the buffer (at the end of the ML) output.

5.4.3 Approach limitations

The feasibility of the timing–based approach proposed in [98] has been successfully verified in subsection 5.4.2 for the SOT STT magnetoresistive technology; the approach, as in the reference article, is however limited by the following aspects:

1. for increasing memory row sizes, the shared ML discharging curves get closer and closer to each other. Hence, if an horizontal threshold is given

[17]It is chosen 500 mV instead of 1 V to make easier the sizing of the buffer attached to the ML: in fact, the buffer has to be designed in such a way that its threshold falls just under the "all matches" case, that corresponds to the ML that remains at the precharge voltage.

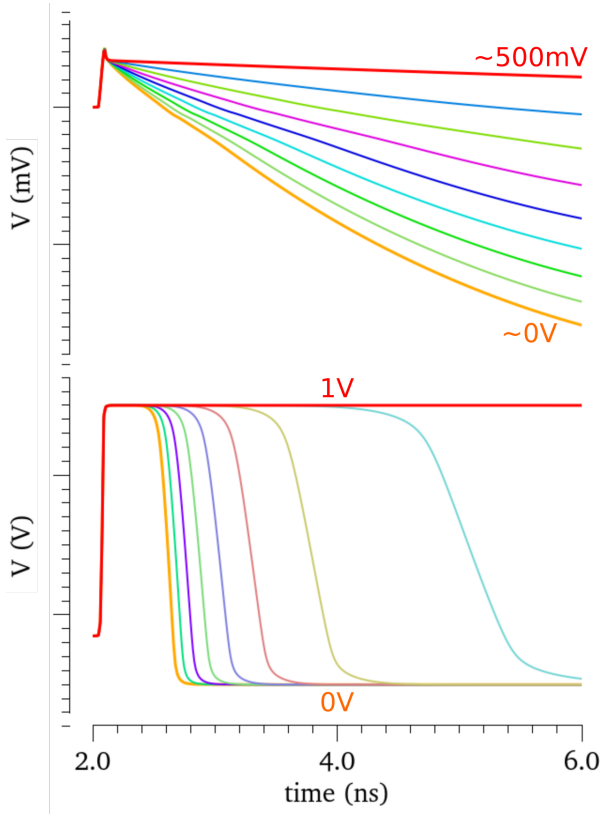


Figure 5.9. Cadence simulation of timing–based approach for 8 bit–long SOT STT CAM row: ML discharging curves (upper part) and buffer output (bottom part) for all HD parametric. The red curve represents the *all matches* case, while the maximum HD distance corresponds to the orange curve.

(ex. by a buffer at the end of the ML), the faster curves (Figure 5.10, left) are almost indistinguishable among them. The same is also tested for other types of threshold: for example, it is designed a simple differential amplifier whose inputs are the ML voltage and a "triangular"[18] reference voltage. The result is that the amplifier outputs are more equally time–spaced than the buffer outputs; however, for increasing row length, they still remain very close to each other.
For this reason, it is confirmed the same limitation of the memristive CAM approach (subsection 5.4.1) also for SOT STT magnetoresistive CAM approach (subsection 5.4.2).
The timing–based approach represents a good solution if the ML is

[18]It is intended an ideal linearly increasing voltage.

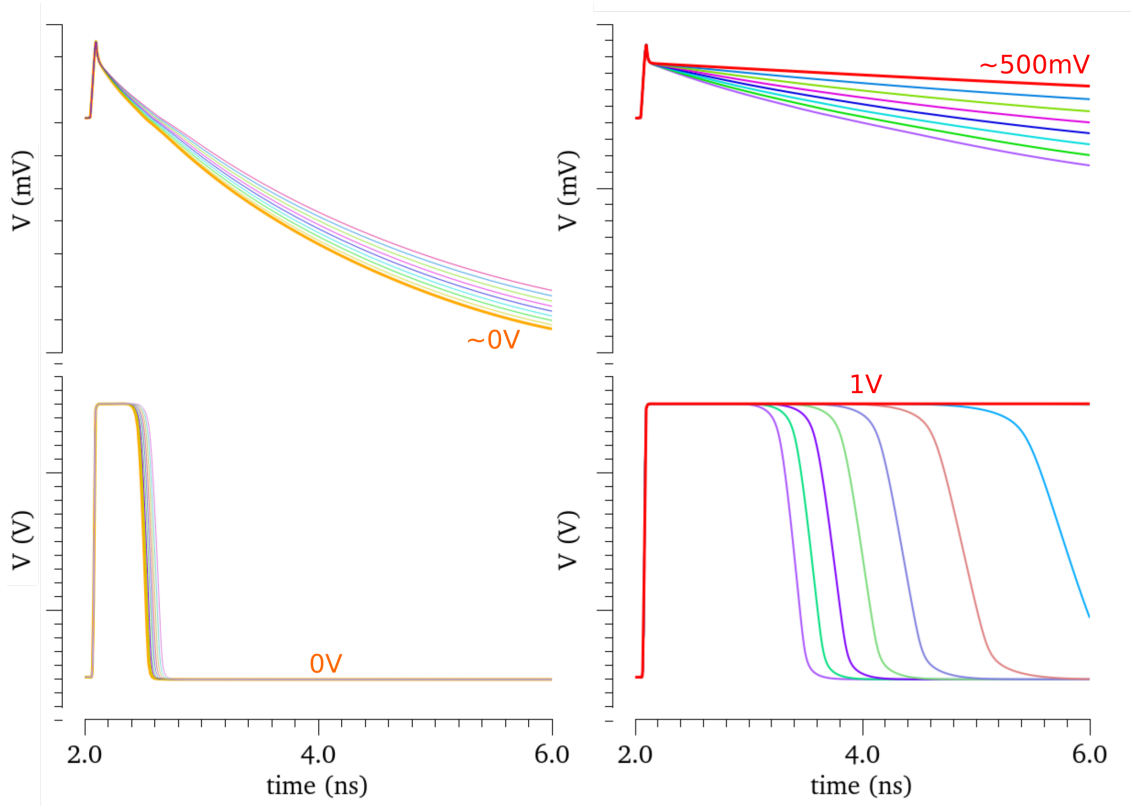


Figure 5.10. The eight worst (left) and best (right) cases —obtained through a parametric analysis— in a 32 bit SOT STT CAM row. In the upper part are shown the ML discharging curves, while in the bottom one the buffer output for all the cases. The red curve represents the *all matches* case, while the maximum HD distance corresponds to the orange curve.

shared only among a reduced number of cells (ex. 4÷8 bit). A possible implementation may be the the design of a memory in which the wordlines/bitlines are connected as for a "normal" CAM array for the whole row/column length, but in which the match lines are subdivided in 4÷8 bit–long segments connected to dedicated sampling SAs and then to a final counter (similarly to the design proposed by [98]).

2. Both memristive and magnetoresistive CAM approaches are quite heavy in terms of hardware requirements: in fact, in addition to the developed CAM structure, it is necessary an hight amount of extra circuitry, like the sampling SAs and counters mentioned in subsection 5.4.1, for finally obtaining a working *HD classifier*.

5.5 An *analog adder* approach

A new solution for implementing the *Hamming Distance counter* discussed in section 5.3 is proposed.
As the approach presented in section 5.4, it exploits the working principle of CAMs for performing a bit–a–bit XOR between the text hypervector (input RBL voltages) and the language hypervector (stored data).
This time, however, the HD counting method is not based on the SAs time sampling of the match lines discharging curves. Instead, the basic idea is to measure the HD of each memory row as a voltage/current value on a load attached to the end of the line. That is to say that on this load it is wanted a V/I proportional to the number of mismatches between the input RBL data and the stored values. For this reason, from this point on, this will be referenced as the *analog adder* approach.

5.5.1 New CAM–like cell

The design of the CAM–like cell used to implement the aforementioned *analog adder* approach is shown in Figure 5.11. It is based on the CAM principle displayed in Figure 5.5, for which the Vo voltage is low (L) or high (H) according to Table 5.4. The Vo amplitude is in the order of few hundreds of millivolts for both L and H cases; for this reason, an inverter stage is introduced for amplifying the Vo voltage in order to be able to turn on/off the pull–up pMOS transistor[19]. This transistor behaves like a generator activated by the Vo value: if Vo is 'L', it means that the inverter output is a logic '1', which keeps the pMOS off. As a consequence, and a zero current is present on the *Current Line* (CL).
On the contrary, when the Vo voltage is 'H', the inverting stage output is '0', which turns on the pull–up pMOS: this links the supply voltage V_{dd} to the source of a second pMOS. Such transistor has a "weighting" role: it behaves like a fixed resistance whose value can be regulated by a proper gate input voltage. In all the simulations, such voltage has been kept equal to 0.4 V. The to–V_{dd} resistive path formed by the two pMOS generates a current on the CL.

[19]In the design proposed in Figure 5.6, it was not necessary to enlarge the Vo dynamic since the nMOS was capable of turning on/off (with a slower switching) even without an amplification.

Since Vo is dependent on the match operation result, it means that the designed schematic behaves like a CAM cell that is able to generate or not a certain I/V (on its output CL) according to whether the 1–bit HD contribution is 0 or 1, respectively.

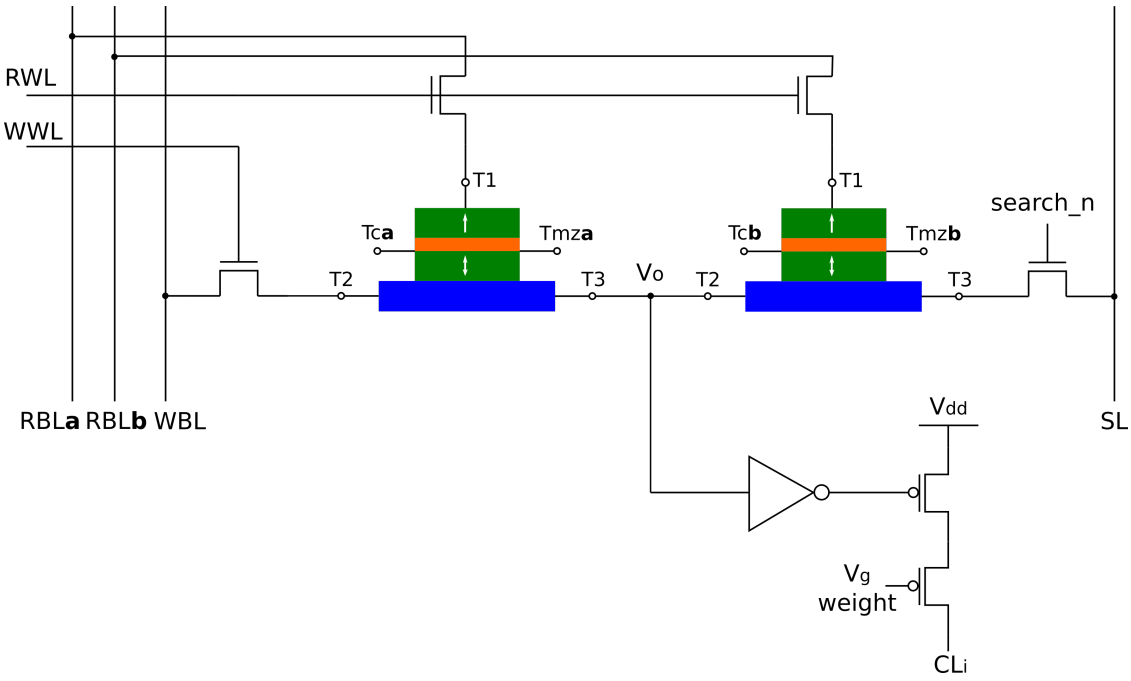


Figure 5.11. New SOT STT CAM cell design for implementing the *analog adder* approach.

In1	In2	In1		In2		Vo	CL_i	XOR
		PAPa	PAPb	V_RBLa	V_RBLb			
0	0	0	1	250 mV	550 mV	L	0	0
0	1	0	1	550 mV	250 mV	H	∼1.7 µA	1
1	0	1	0	250 mV	550 mV	H	∼1.7 µA	1
1	1	1	0	550 mV	250 mV	L	0	0

Table 5.4. XOR operation cases for all the input combinations: the XOR = '1' cases are traduced as a little amount of current flowing on the CL. The CL current value is the one corresponding to the selected sizing and supply of the weight pMOS (inside the cell) and the load nMOS attached to the end of the row; PAPa/b, instead, are the logic states of MTJa/b, respectively.

The XOR = '0' and XOR = '1' cases, for PAPa = 0 and PAPb= 1 configuration, are shown as example in Figure 5.12

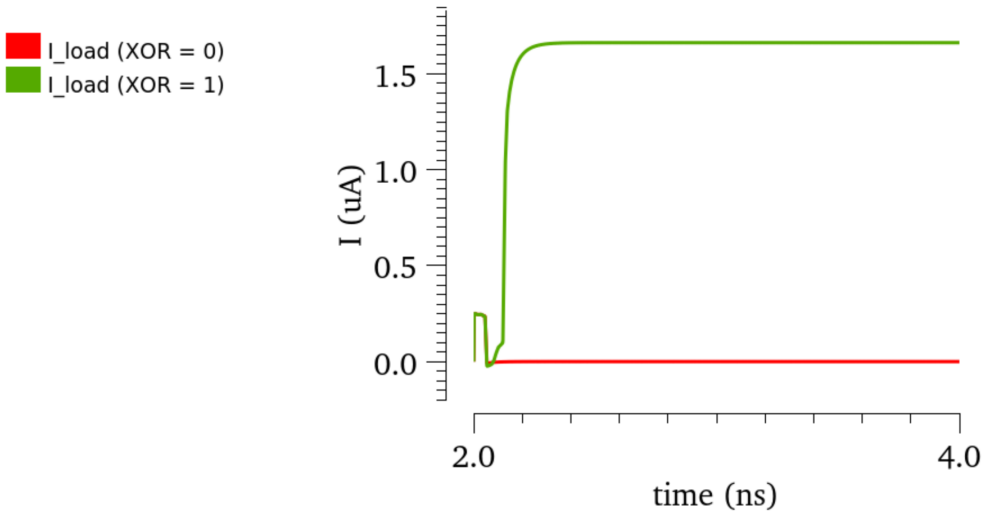


Figure 5.12. Cadence simulation plot for the first two cases of Table 5.4, with the cell displayed in Figure 5.11. The start of the search cycle is at 2 ns (arbitrary value): it is evident that the XOR operation is quite fast.

The other operations (write, read) are exactly the same as the ones described in subsection 5.4.2 and summed up in Table 5.2.

A noticeable difference between the cell in Figure 5.11 and the one in Figure 5.6 is the splitting of the wordline into read and write wordlines (RWL, WWL); in fact, when not divided, the single WL is responsible for the activation of both RBL and WBL access transistors. This does not represent a problem duting writing and reading since these operations are performed by activating only one WL at time[20]. However, the search operation implemented in CAM arrays has to be performed in parallel to all the rows: this means that all the wordlines must be activated, for providing the same RBL search pulses (Table 5.4) to each CAM cell of the same column. By doing this, also the WBL is shared among all the cells of the same column: this fixes a certain voltage on the MTJ 'T2' terminals (Figure 5.11) of each cell, precluding the implementation of a correct search operation.

[20]In fact, in the RAM–like SOT STT cell described in subsection 4.3.1, the WL has not been split into RWL and WWL.

Instead, when the WL is divided into RWL and WWL, it is possible to correctly implement the search operation in all the CAM in parallel: in this case, while activating the RWLs, the WWLs can be turned off detaching all the MTJ 'T2' terminals from the shared WBL[21].

The designed CAM–like cell is then tested for 8x8 and 32x32 arrays. The array organization is shown in Figure 5.13; the current lines of the cells of the same row are linked all together and fed to a properly–sized load nMOS. The CAM–like performance evaluation, for all the operations (read, write, XOR) is discussed in subsection 5.5.2 for the designed arrays.

The gate voltages of both the cells weigth pMOS transistors (p_weigth) and the row load nMOS (n_load) can be regulated in order to control the current divider, together with their sizing. It is important to force a $Rp\_weigth >> Rn\_load$, in order to maintain low the leakage current that goes inside XOR = '0'–performing cells.

[21] This has not been noticed before since the cell designed in Figure 5.6 has been tested only for different sized rows, and not for arrays.

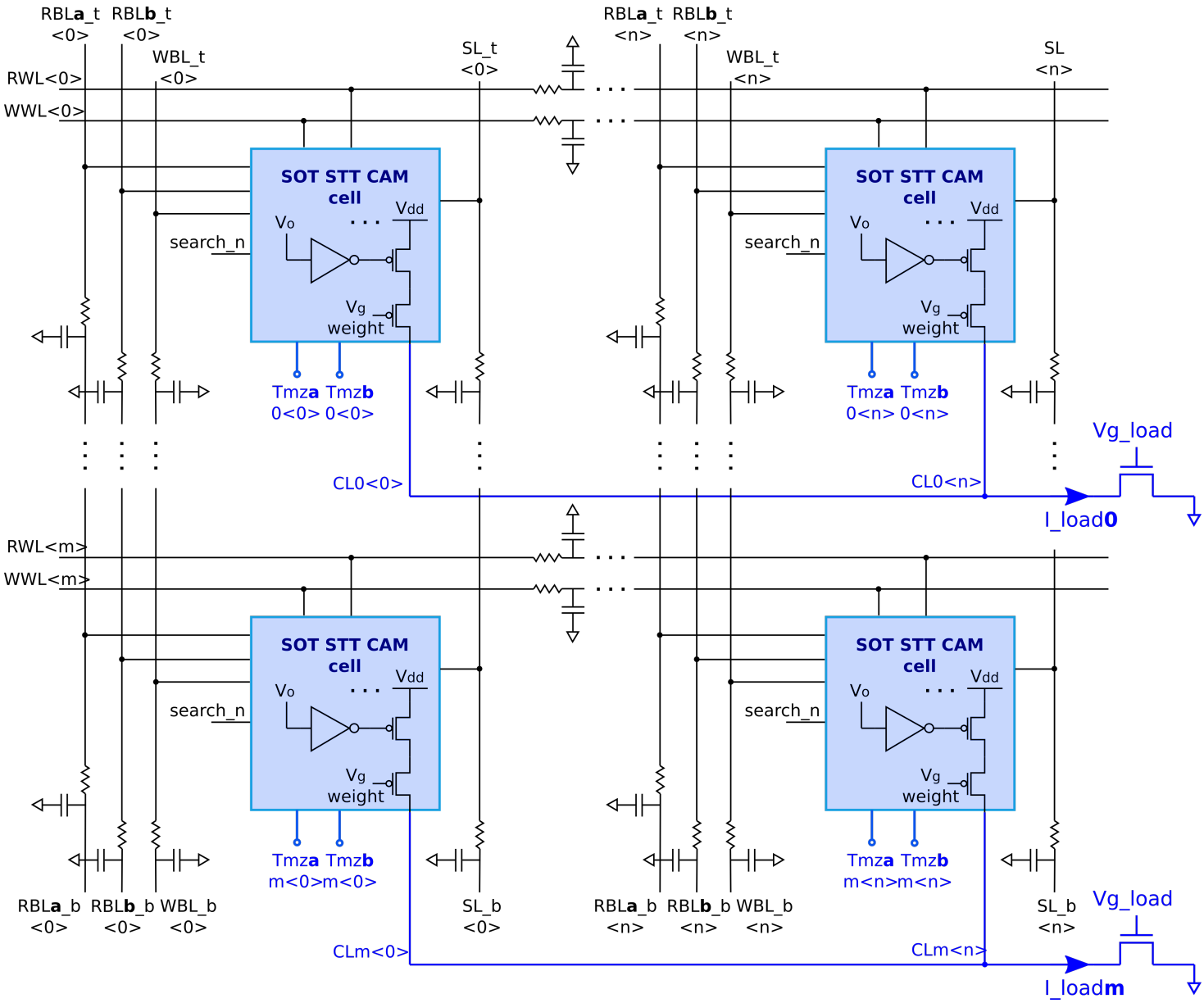


Figure 5.13. SOT STT CAM array organization for implementing the *analog adder* approach. The basic cell is the one displayed in Figure 5.11.

5.5.2 Results

In this section are presented the results of the HD counting method implemented by the *analog adder* approach and the related performance evaluation (delay, power consumption) for 1 cell, 8x8 and 32x32 arrays.

All the results are obtained through the same simulation flow discussed in subsection 4.3.5. The Python script for the power function computation is reported in section B.3.
The signals that have to be provided to the schematic are listed in Table 5.5, together with a brief description. The other signals of interest are the same of Table 4.3, in addition to the current on the load at the end of the CL (*I_load*) that is relevant for search operation.
Also the list of simulation variables is the same of Table 4.4 with the unique difference that, in this case, R_{ref} is equal to 15 kΩ.

Signal	Write'0'	Write'1'	Read	Search'0'	Search'1'	Idle	Signal description
V_RWL	V_{dd}	V_{dd}	V_{dd}	V_{dd}	V_{dd}	0	RBL nMOS activation
V_WWL	V_{dd}	V_{dd}	0	0	0	0	WBL nMOS activation
V_RBLa	−200 mV	500 mV	0	250 mV	550 mV	0	RBLa input (ideal)
V_RBLb	500 mV	−200 mV	0	550 mV	250 mV	0	RBLb input (ideal)
WBL_EN	V_{dd}	V_{dd}	0	0	0	0	WBL write driver enable
WBL_disch	0	0	0	0	0	V_{dd}	WBL discharge
SE	0	0	V_{dd}	0	0	0	SA enable
SE2	0	0	V_{dd}	0	0	0	RBL read driver enable
disch	0	0	0	0	0	V_{dd}	RBL discharge (SA)
search_n	V_{dd}	V_{dd}	V_{dd}	0	0	V_{dd}	Search nMOS activation*
Vg_pMOS	V_{dd}	V_{dd}	V_{dd}	0.4 V	0.4 V	V_{dd}	Vgate of weigth pMOS*
Vg_load	0	0	0	V_{dd}	V_{dd}	0	Vgate of load nMOS

Table 5.5. Complete list of signals that have to be provided to the CAM schematic, for all the operations. Note: *The search nMOS and weigth pMOS are present inside each cell, see Figure 5.11.

The simulation plot resulting for a search operation performed in parallel to all the 8x8 array is reported in Figure 5.14. It shows the *I_load* currents related to all the possible HD cases (only the '0' case is missing, because the HD=0÷8 cases counts 9·*I_load* curves for 8 rows). This plot can be obtained

in two ways:

1. by performing a parametric analysis on an array with all equal PAPa,b[22] for all the RBLa, RBLb input search voltage combinations (Table 5.4), in such a way that *I_load* on the first row (*I_load0*) has the maximum number of mismatches (i.e. of XOR = '1' cases) and the *I_load* on the last row (*I_load7*) has the minimum (not zero) number of mismatches. This method is quite time consuming for increasing array sizes (just for the 32x32 array, such parametric simulations is very slow);

2. by performing a unique simulation on a properly designed array (just for testing purposes), as the one displayed in Figure 5.15: it consists in an array with half of the total cells (plus the ones on the diagonal) with initial MTJs states equal to the variables *PAPa,b* that can be directly set in the simulation state. The lower part of the array cells (under the diagonal) is instead set to *PAP0,1* variables. This allows to obtain all the HD cases (from 1 to 8 or 0 to 7, depending on how the *PAPa,b* and *PAP0,1* are set) in just one single simulation with the same RBLa,b search input voltages. For convenience, it is chosen this method, for both 8x8 and 32x32 arrays, since much faster than 1) after having implemented Figure 5.15.

In Figure 5.16 it is instead reported the simulation plot for the 32x32 array.
In both cases, the results are good and the design correctly realizes the target task described in the introduction to section 5.5: the HD is translated to an analogue current (or voltage) value following the *analog adder* approach. This means that a working analog HD counter has been realized. The currents (or voltages) on the row loads can be fed to an ADC stage that converts them into digital voltages, or other additional peripherals for selecting the lower current value for finding the minimum HD. Hence, this HD counter module can be exploited for implementing the basic function of the *search module* discussed in section 5.3.

The main advantages of such implementation are the following:

1. as it can be seen from Figure 5.14 and Figure 5.16, the HD count is very fast: the mismatch delay is quite small and remains under the hundred

[22]Initial a,b MTJs logic state.

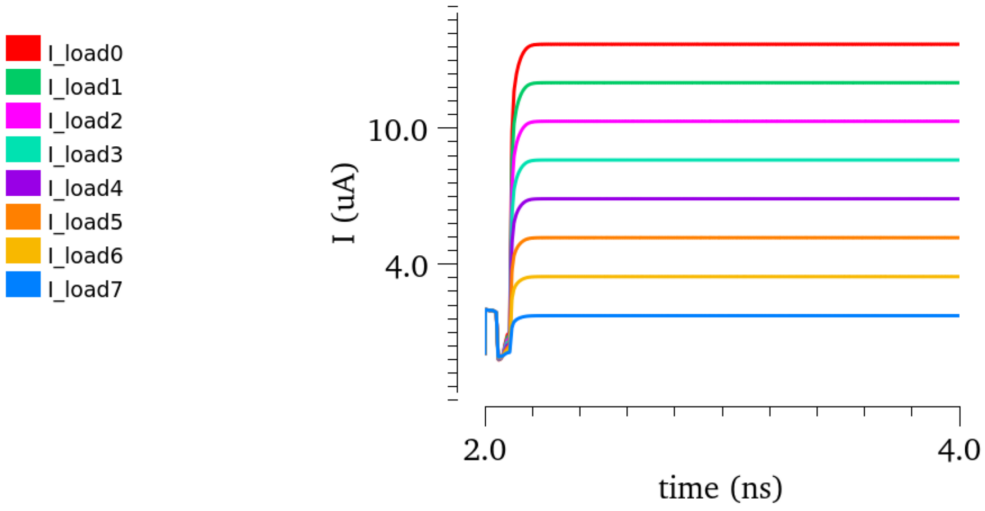


Figure 5.14. All HD cases (except '0') corresponding to different current values on the 8x8 array CLs.

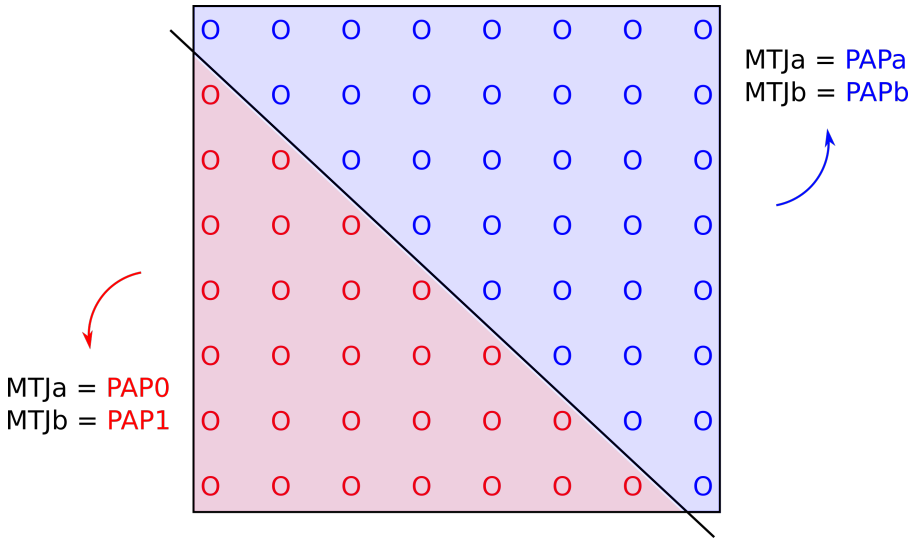


Figure 5.15. Image for clarifying the *PAP* variables (*PAP0,1* and *PAPa,b*) distribution on the array elements, used for for plotting all the *I_load* HD cases in both 8x8 and 32x32 arrays (Figure 5.14, Figure 5.16).

of picoseconds for all the tested array sizes. It has to be noticed that no parasitic contributions are inserted in the CLs since the load nMOS could be attached, for example, at the middle of the row (instead of at the end) and so the path that the current has to cross would be reduced.

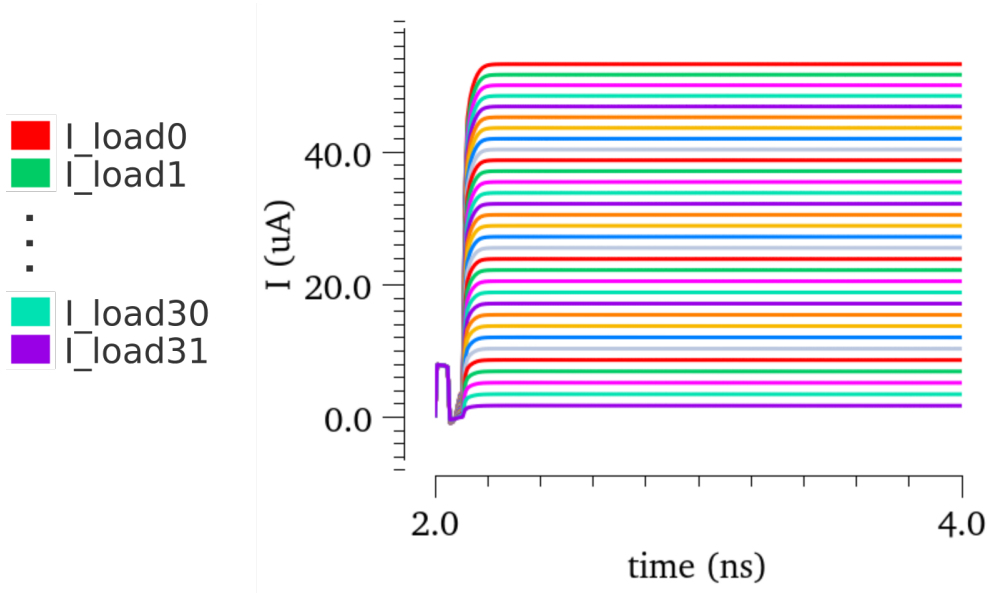


Figure 5.16. All HD cases (except '0') corresponding to different current values on the 32x32 array CLs.

Hence, it is preferred to estimate only the ideal mismatch delay without considering parasitic contributions;

2. the realized approach is hardware–saving with respect to the one proposed in section 5.4: even if an ADC would be inserted at the end of each row, it would be however less invasive than inserting almost one time–sampling SA for each bitcell, as instead done for the timing–based approach.

For the performance evaluation are valid the same considerations done in subsection 4.4.3; the measurements —extracted from Cadence simulations— include delays, average and peak power consumption and energy consumption for all the *analog adder–like* CAM operations (write, read, XOR). The simulated array sizes are 1x1 (1 cell), 8x8 and 32x32.

In Figure 5.17 are reported the delays computation for the two write cases. It can be noticed that, in both cases, the results are generally better than the ones obtained in Figure 4.22 for the MRAM–like arrays: this may be due to the fact that, for the CAM designed in section 5.5, the RBLa and RBLb write voltages are driven by ideal voltage generators, instead of the RBL real

driver designed in subsection 4.3.3.

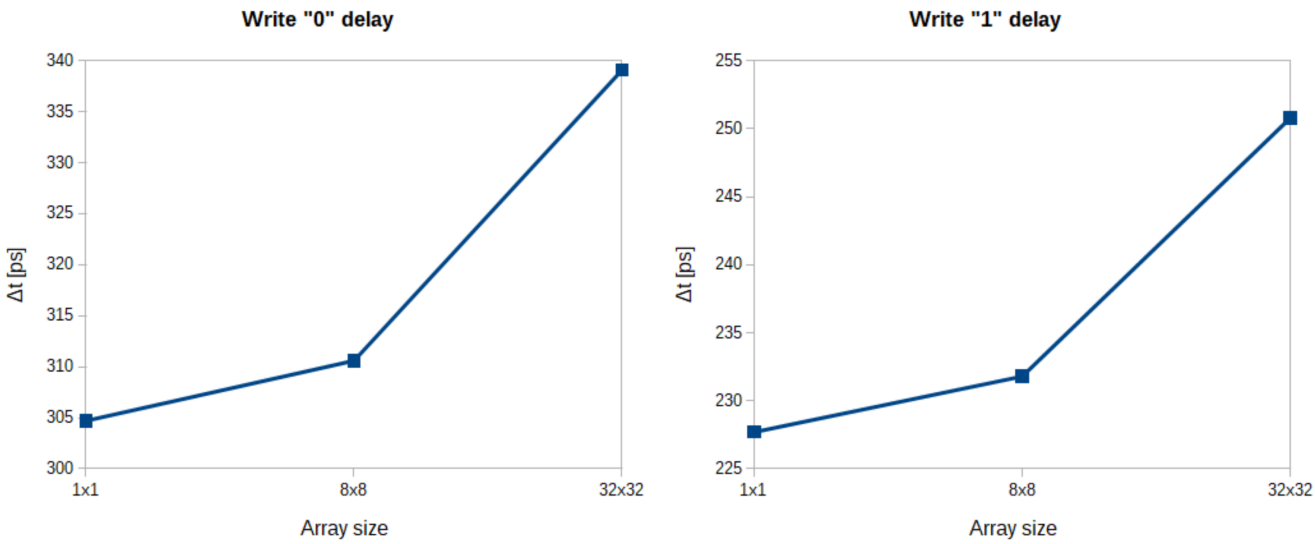


Figure 5.17. Delays results for both writing cases, for increasing array size

The read delays reported in Figure 5.18 are instead slightly higher than the ones computed in the memory without logic (Figure 4.23). One of the reasons may be the different R_{ref} (i.e. reference cell resistance) setting, that changes a little the amplitude of the current generated by the read driver.

The XOR = 0/1 delay results are not reported in table format for the reasons explained previously in this section: no parasitic contributions are introduced in the CLs, hence the delays results for different array sizes are ideal. However, as it can be seen from Figure 5.12, Figure 5.14 and Figure 5.16, the XOR operation is fast: if the XOR delay is defined as the 50% delay between the WL activation signal and the *I_load* current signal, the maximum HD case is associated to a 85.8 ps÷87.3 ps delay.

Both write (Figure 5.19) and read (Figure 5.20) average power consumption are comparable to the ones in subsection 4.4.3. The write power, for both '0'/'1' cases, is slighly higher than the one in Figure 4.24: this is due to the fact that, for equal array sizes, a doubled number of MTJs has to be written in the *analog adder* CAM.
The read average power consumption is instead very similar to the one of Figure 4.25.

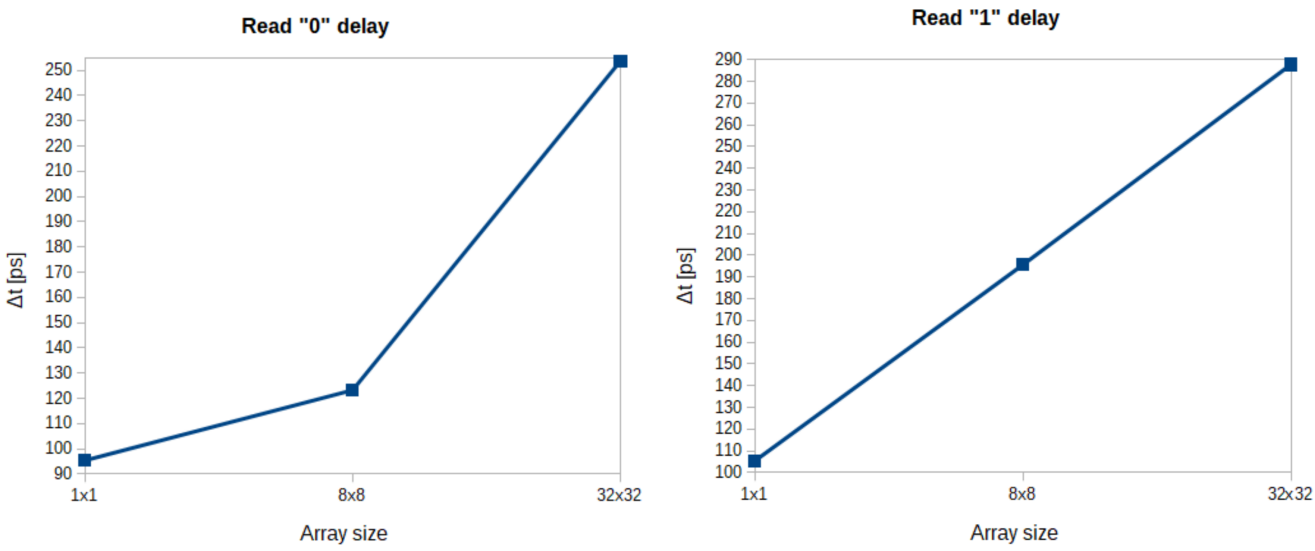


Figure 5.18. Delays results for both reading cases, for increasing array size

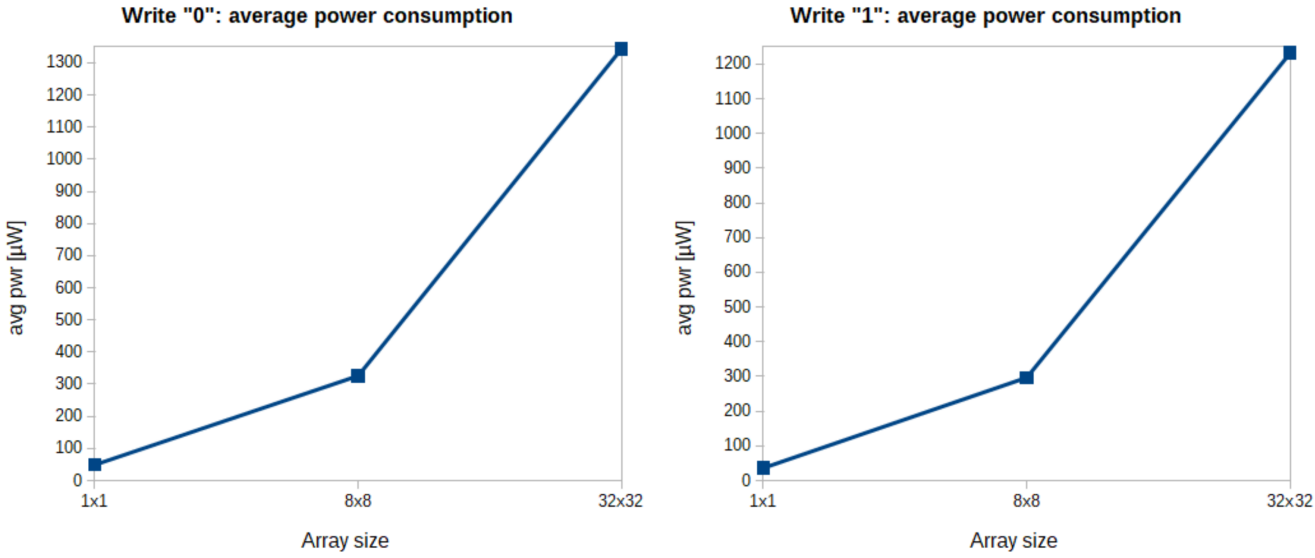


Figure 5.19. Average power consumption related to both writing cases, for increasing array size

The average power consumption associated to the XOR operation increases faster —for higher array sizes— than the other operations, as it can

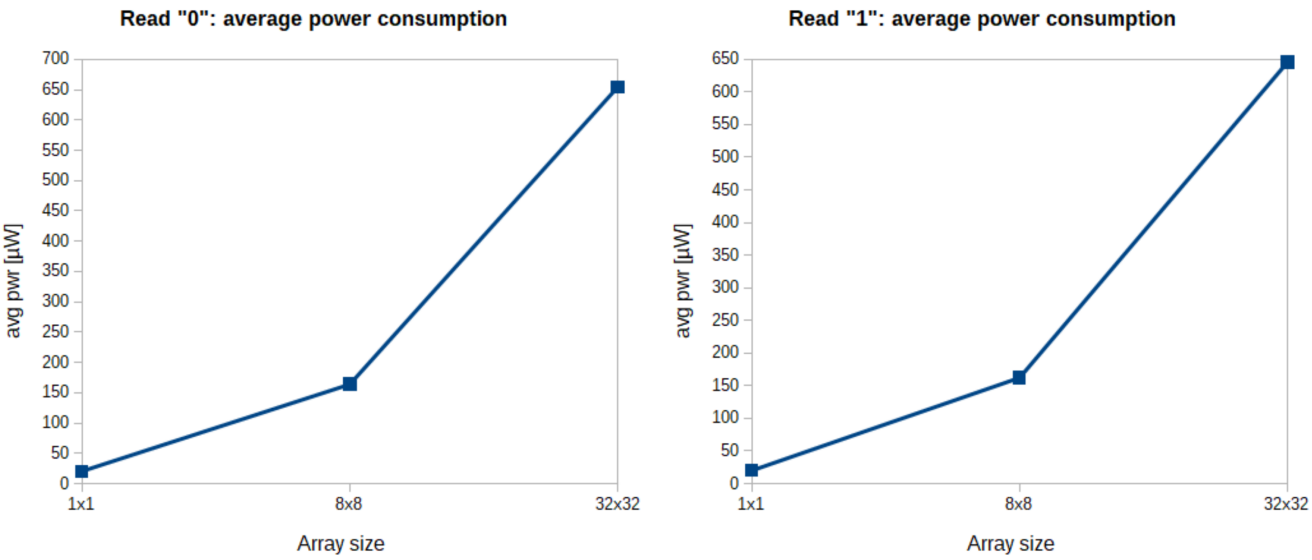


Figure 5.20. Average power consumption related to both reading cases, for increasing array size

be seen from Figure 5.21: the reason is that a search operation is performed on all the memory cells in parallel, and not only with one wordline activated for cycle, as done instead for reading and writing operations. However, the XOR = '1' (mismatch) operation performed on a single CAM cell consumes less than 7 µW: a lower average consumption than both read and write operations.

The same considerations done about the average power consumption can be applied also for the energy consumption results (Figure 5.22, Figure 5.23, Figure 5.24).

In Figure 5.25 and Figure 5.26 are reported the peak power results for write and read operations, respectively. The falling and rising time of all the signals involved in the operations is set to 50 ps, exactly as done in subsection 4.4.3: for this reason, the results of the two designs are comparable. Moreover, by observing Figure 4.28 and Figure 4.29, it can be noted a remarkable similarity between the two couples of plots.

In Figure 5.27, instead, is displayed the plot of the peak power consumption related to the XOR operation cases. As for the XOR average power plots (Figure 5.21), the high consumption that can be observed for the 32x32 array is justifiable since it is to consider that all the 1024 cells perform the XOR operation simultaneously.

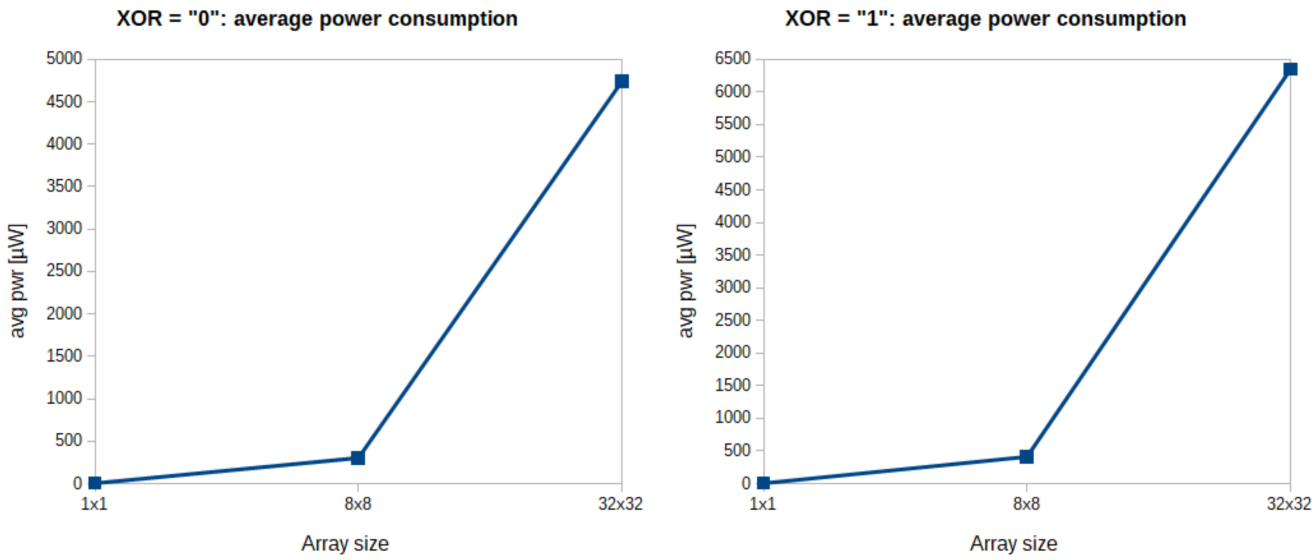


Figure 5.21. Average power consumption related to both XOR cases, for increasing array size

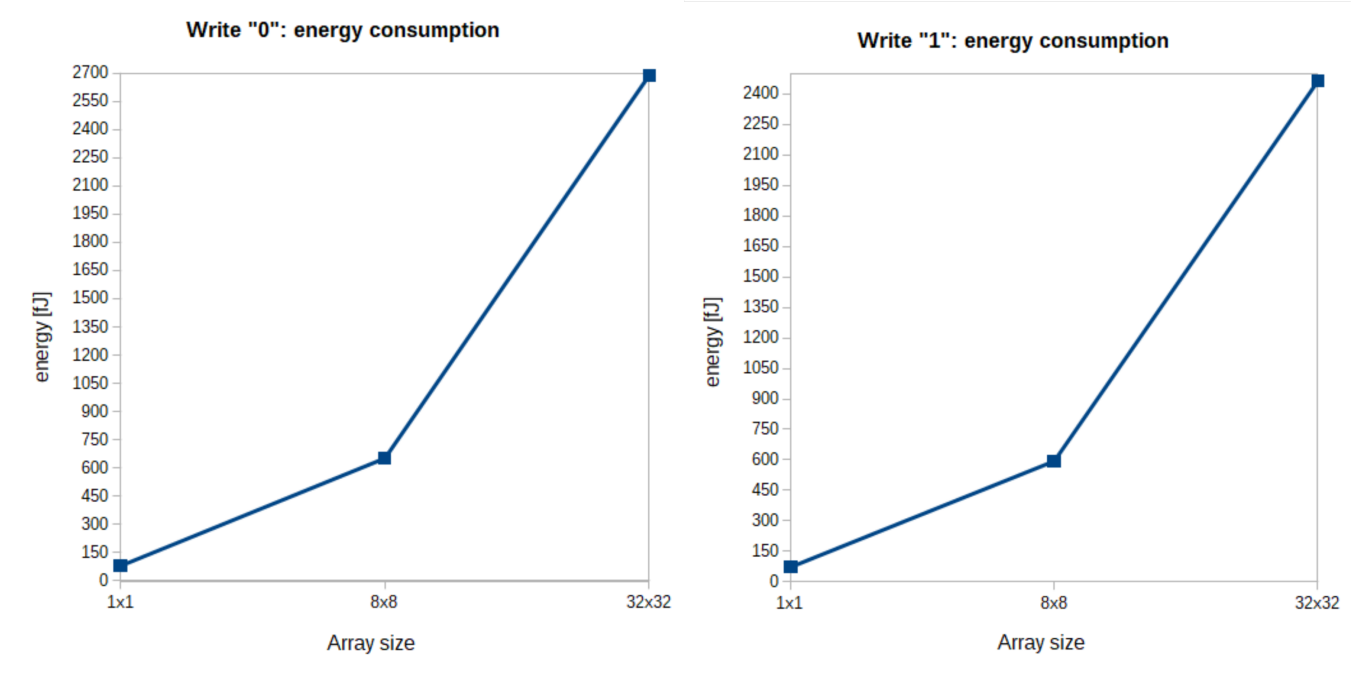


Figure 5.22. Energy consumption evaluated for write '0' and '1' cases

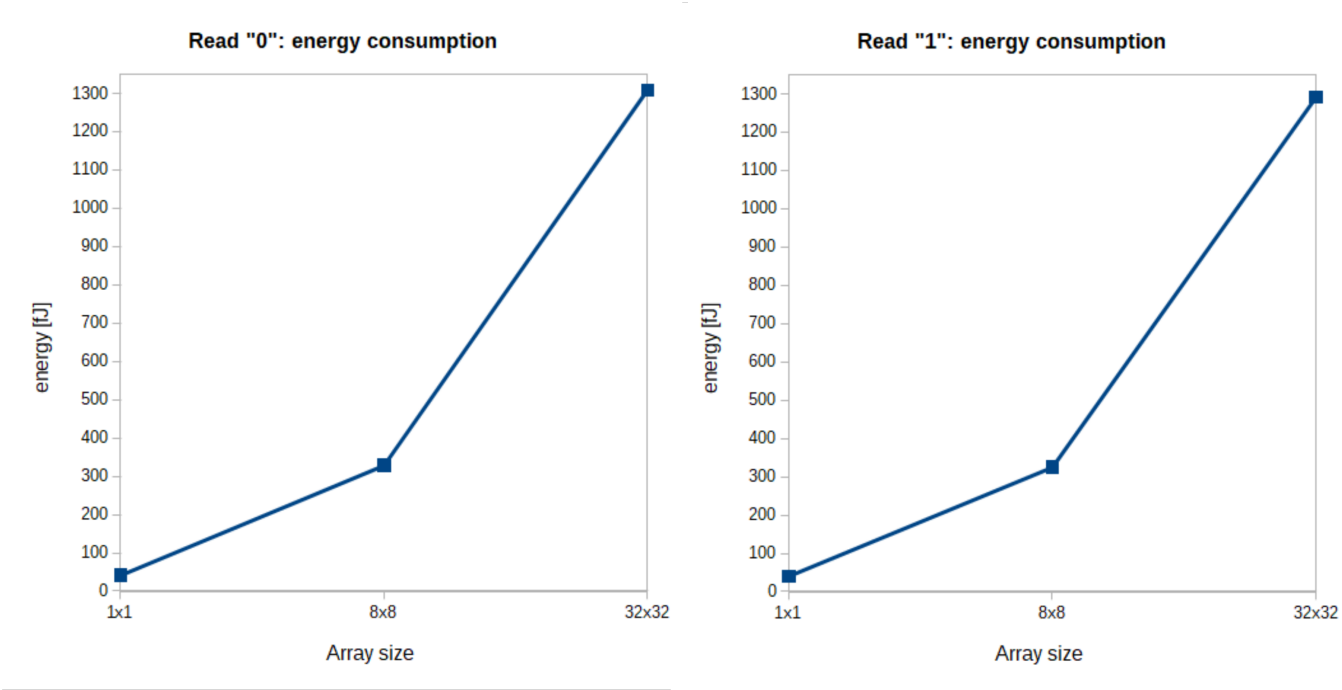


Figure 5.23. Energy consumption evaluated for read '0' and '1' cases

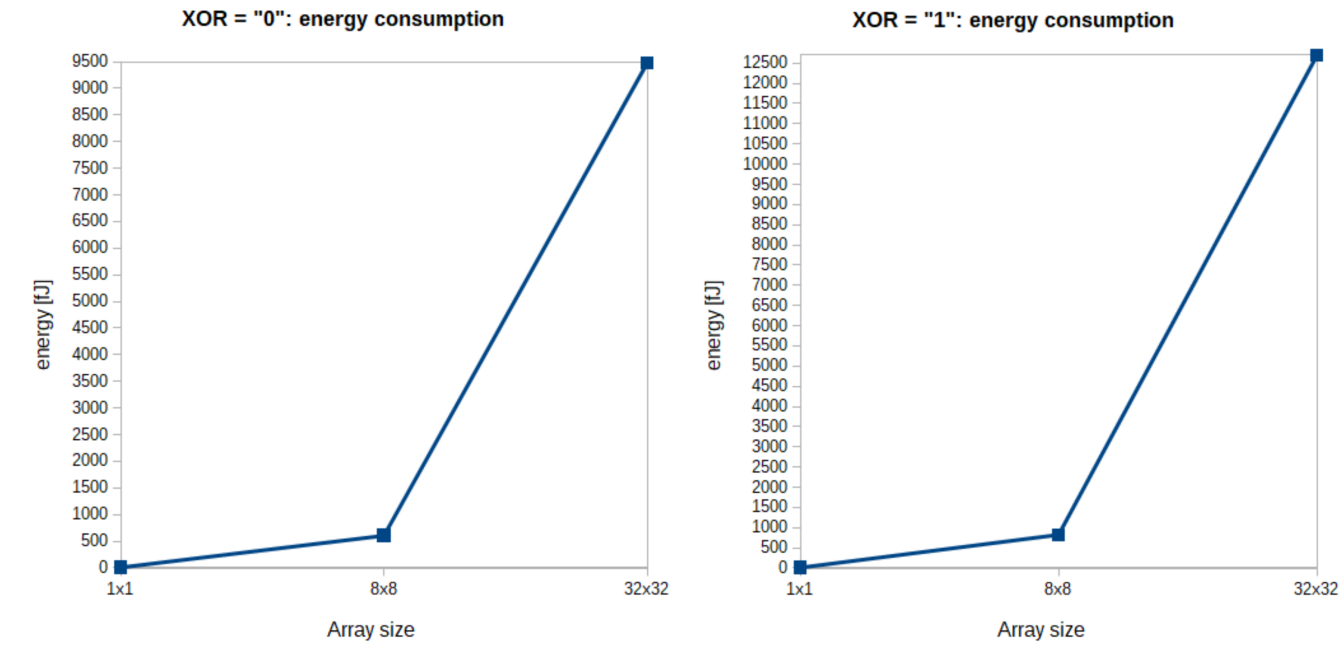


Figure 5.24. Energy consumption evaluated for XOR ='0' and '1' cases

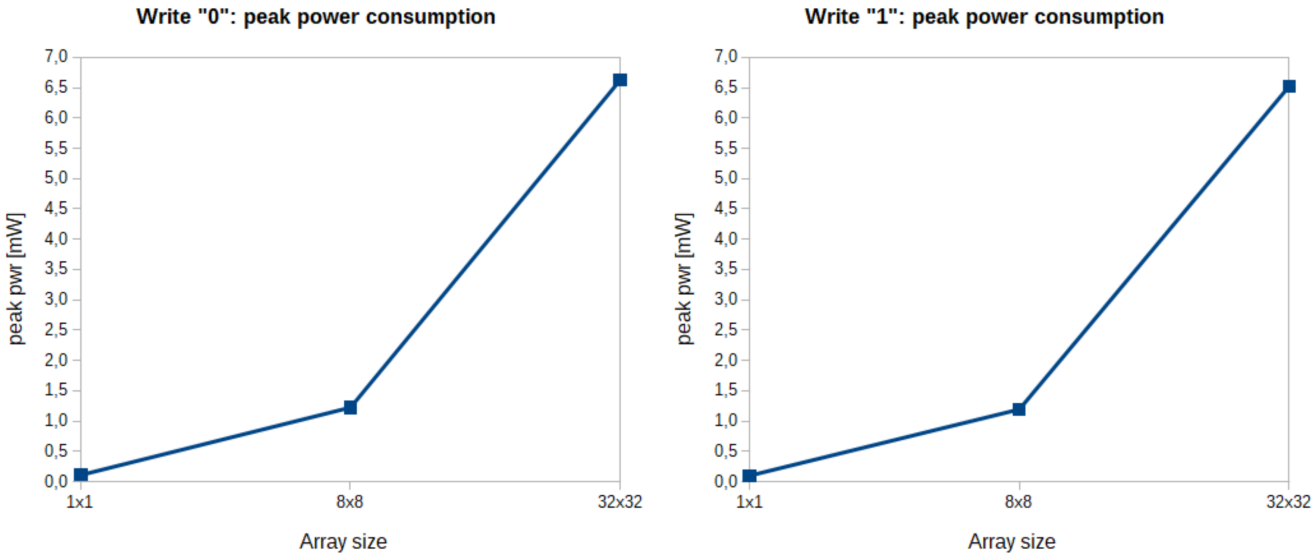


Figure 5.25. Peak power related to both writing cases, for different array sizes

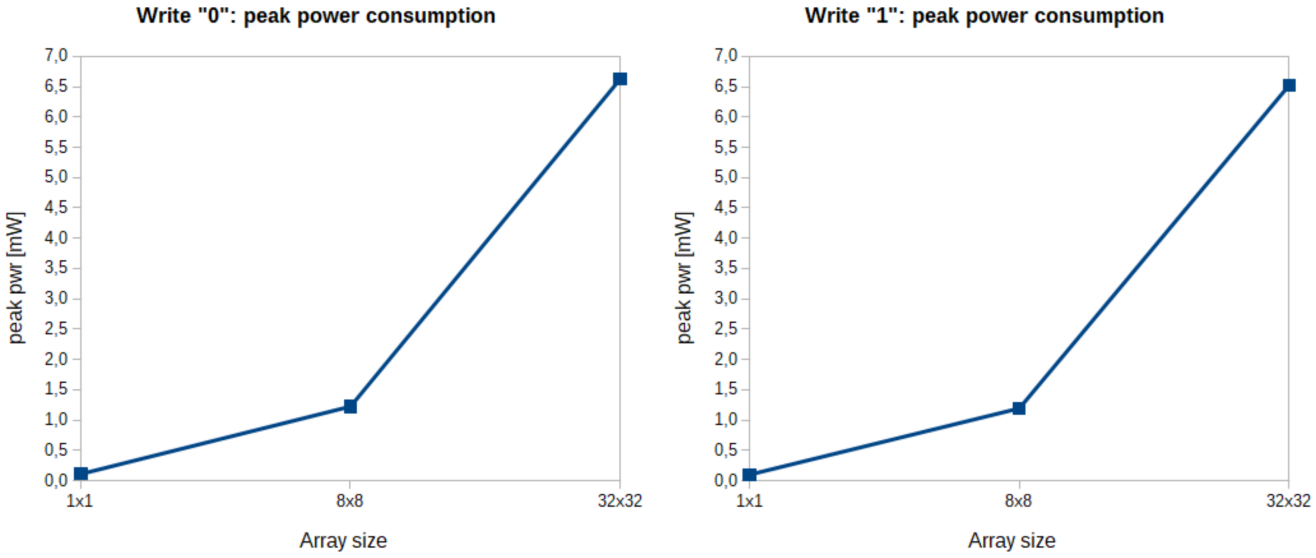


Figure 5.26. Peak power related to both reading cases, for different array sizes

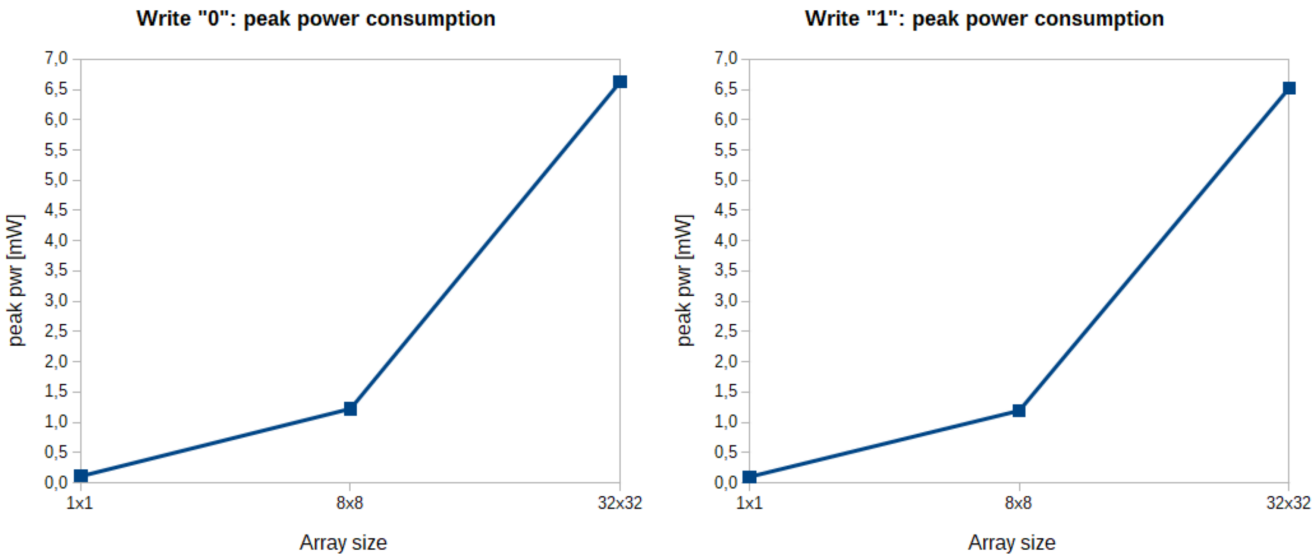


Figure 5.27. Peak power related to both XOR cases, for different array sizes

5.6 Conclusions and future prospects

This work is divided into two parts.
The first one is dedicated to the understanding of the working principle of Beyond–CMOS memory technologies (chapter 1), the analysis of their architectures and peripherals (chapter 2), and the research of feasible Logic–in–Memory approaches exploiting such technologies (chapter 3).
The second one includes the development, through Cadence Virtuoso, of a magnetoresistive RAM–like memory without logic functions (chapter 4). Starting from this first design, a CAM–based LiM (chapter 5) for the counting of Hamming Distance (HD) has been realized, following two different approaches: the *timing–based* one (section 5.4) and the *analog adder* (section 5.5) one.

The magnetoresistive (in particular, based on both Spin Transfer Torque and Spin Orbit Torque effects) technology choice allows to realize a memory that is very fast in both reading and writing operations, but that involves a consistent power consumption (especially for writing), with respect to a Static RAM realized through CMOS technology (subsection 4.4.3).
Almost the same characteristics are obtained with the development a SOT STT CAM; such design allows to exploit the additional search operation for counting the number of mismatches, through properly modified CAM cells. This peculiarity can be exploited for implementing the core function of a *HD classifier*, presented in section 5.3.

The *HD classifier* has been taken into account as LiM case of study for the selected SOT STT magnetoresistive technology.
The design proposed in section 5.5, in particular, is able to count the HDs —between external data and stored information inside the memory— as an analog value; this is done in parallel to all the CAM rows and results in a very fast operation subsection 5.5.2. Such approach is faster than the memristive one found in literature subsection 5.4.1; moreover, it heavily reduces the hardware requirement, since it does not need the presence of one Sense Amplifier per cell[23] for the HD counting.

[23]See explanation in subsection 5.4.1.

However, different improvements can be introduced:

- the optimization of the SOT and STT pulses combination for reducing the power consumption during read operation;
- the realization of real RBL drivers for the SOT STT CAM designed in section 5.5[24];
- the synthesis of the designed Cadence schematics into a program able to simulate them for larger array sizes
- the development of a minimum analog value search algorithm optimized for finding the lower HD among the ones computed by the designed *HD counter*;
- the research and test of other target algorithms for comparing the efficiency of such approach.

Thank you for the attention!

[24]The real drivers has been realized for the SOT STT MRAM design in section 4.3, but not for the SOT STT CAM.

Appendix A

Sense Amplifiers classification

A discussion about SA typologies is introduced since, as mentioned in subsection 2.2.1, peripherals give a great contribution to memories performance; it is worthwhile to do some considerations, anyway:

1. the presented classification is valid for all memory technologies, since it is derived from basic analog electronics;
2. it makes reference to standard SA circuits. Quite often in literature modified or hybrid SA implementations are found, but their basic scheme can (almost) always be referred to one typology included in such classification. In this way it is possible to find out if a certain EMT presents a trend in selecting a specific SA;
3. by understanding the benefits and limitations of the various SA types, it will be simpler to make a choice when designing a target LiM application;
4. one way to implement in–memory computation is, actually, to introduce logic at the level of SA, so it is good to know advantages and constraints of the various typologies.

In Figure A.1 it is summarized a SA categorization according to the working mode (voltage, current or charge transfer based); it can be further subdivided if we consider the analog circuit (differential, latch, current mirror, and modifications) at the basis. All the related circuital schemes and working principles can be found in [46].

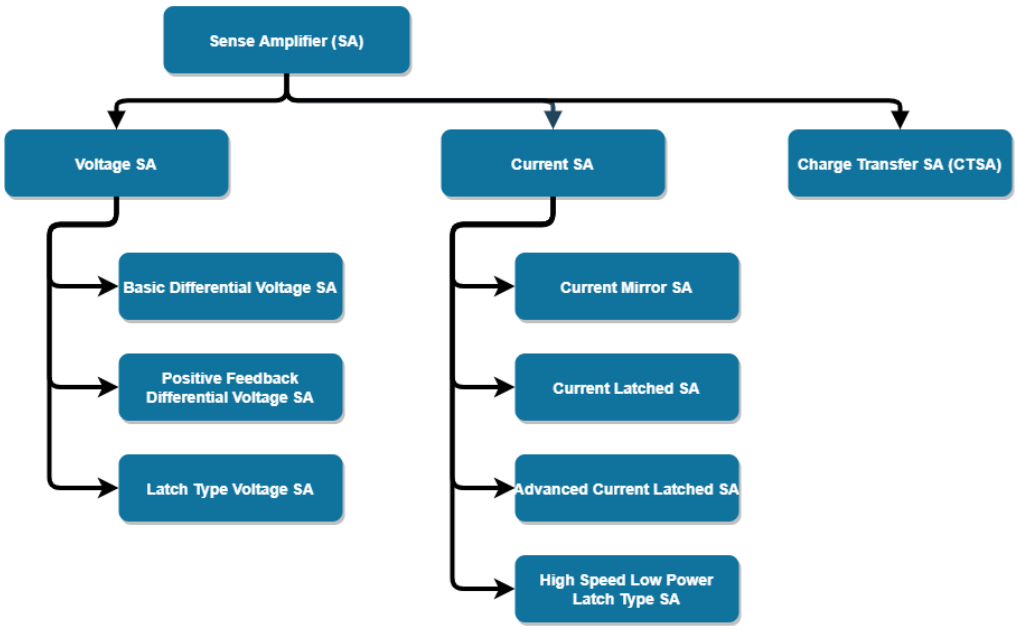


Figure A.1. SA simplified classification.

Voltage mode Sense Amplifiers are based on the comparison between the voltage on the bitline V_{BL} and a reference voltage V_{REF}. They are usually characterized by an higher BL input impedance than current SA , that make possible to offer an high voltage gain with rather simple circuits. Nonetheless, for large arrays, so with increased BL capacitance and limited voltage swing, voltage SA could become less reliable and require higher power.
The most common types of voltage SA[1] are the following:

- Basic Differential Voltage SA: due to its high offset, high power consumption and limited speed it is usually not employed in memory applications;
- Positive Feedback Differential (also called Conventional Latch Type) Voltage SA (Figure A.2): thanks to its positive feedback, it provides really high differential gain that increases sensibility and sensing speed. It can also rewrite data that are read in a destructive way. It is employed for example in [41] for STT–MRAM, see subsection 2.2.2;
- Latch Type Voltage Sense Amplifier (VLSA, Figure A.3): it offers high

[1] According to the classification proposed in [46, 47]

speed with low power dissipation, and for this reason it is often utilized as SA for LiM applications. It is usually employed when it is required fast sensing at low supply voltage and with BL voltages near to ground level. It is however sensible to noise [46, 47, 48]. For instance, a VLSA is described in [48] for 0T1R ReRAM, see subsection 2.3.2.

Current mode Sense Amplifiers are instead based on the comparison between the sensed current I_{CELL} and a reference current I_{REF}. Their input impedance, smaller than voltage SA one, allows them to have lower delays and cross–talk, together with smaller substrate currents.
They are commonly classified[2] as:

- Current Mirror Sense Amplifier (CMSA, Figure A.4): it can be easily controlled, and it can be sped up by enhancing the operating current (so with more power); for these reasons it is often employed in memories. Examples of SAs that exploit current mirror (CM) stage can be found —more or less— for all the EMTs selected in chapter 2. However, it is not certain that an SA including a CM can be classified as a CMSA: this is justified by the fact that, frequently, the CM is not the basic element of the circuit and other stages contribute in defining the final behaviour of the SA.

- Current Latched Sense Amplifier (CLSA)[3]: it cannot work at too low voltage, since otherwise the differential current would be reduced to the point that it could not be used for fast sensing anymore. To avoid this, pre–charge is needed to rise BL voltage close to V_{DD} to keep the input nMOS transistors in saturation [48]. Nonetheless, this type of SA has reduced power dissipation. STT–MRAM Chung's SA, discussed in [44], can be classified as a CLSA.

- Advanced Current Latched Sense Amplifier (ACLSA): it has a more complex structure, but requires lower power dissipation and, through switching circuits, it can operate at smaller voltage.

- High speed low power latch type sense amplifier: in theory it has the same power dissipation of CLSA, but it provides both improved performance and lower operating voltage. In terms of same power consumption, it can work faster than ACLSA and CLSA. Nevertheless, it has a

[2] According to the classification proposed in [46, 47]

[3] According to [46], its circuit is the same of VLSA but with current inputs

large number of transistors and it is subjected to process variations and current mismatch, that may lead to sensing errors.

Finally, Charge Transfer Sense Amplifiers (CTSA) working principle is the redistribution of charge, taken from the high BL capacitance to the low output capacitance of the SA. This mechanism intrinsically offers high speed and requires low power. It is faster and consume less energy than a voltage SA. The great disadvantage of charge transfer SAs is, however, the complexity of design [46, 47].

Due to the increased speed, higher than voltage sense amplifiers one, current mode SAs seem to be the best choice for reading interface for fast emerging memory implementation.
Charge transfer SAs could be a good alternative but their complexity represents a challenge not so easily manageable.

However, sometimes VSA are still used —for instance— when a high voltage–swing is required (for better accuracy) despite the higher energy required, or when high speed is required at low voltage levels, e.g. in 0T1R ReRAM (see subsection 2.3.2).
When a fast speed of operation is required, latch–type CSAs are usually employed, but they are not good for low voltage applications. This is due to the fact that CLSA needs a voltage on the BLs close to V_{DD} in order to maintain the input nMOS pair in saturation: out of such region, CLSA works much slower and so it loses its main benefit [48].

In chapter 2 some examples of SAs for EMTs are presented; obviously, they do not pretend to be the only solutions for a specific EMT, also because the articles from which they are extracted may be application–driven (i.e. high speed, low voltage required by the case of study, etc.). The reading key is the trend in using that SA, instead of others, for that EMT in last year applications (2015–2020), trying to explain the reasons for each choice, in order to understand which sensing scheme is the most appropriate for the target application.

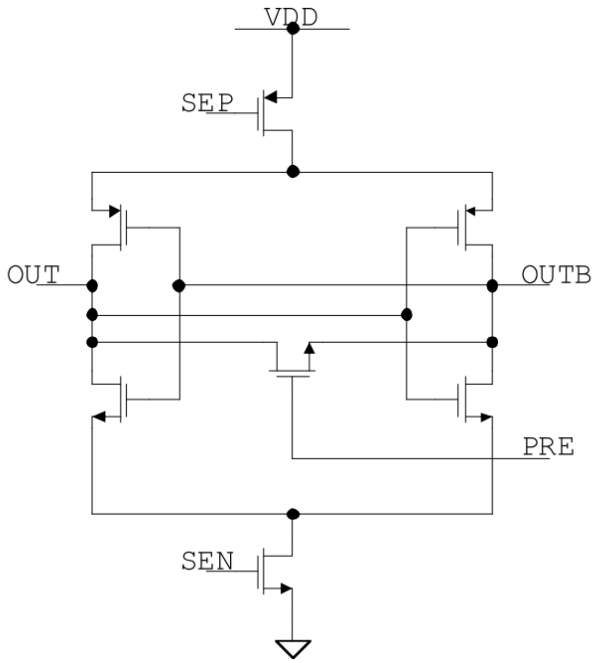


Figure A.2. Positive Feedback Differential (or Conventional Latch Type) Voltage Sense Amplifier.

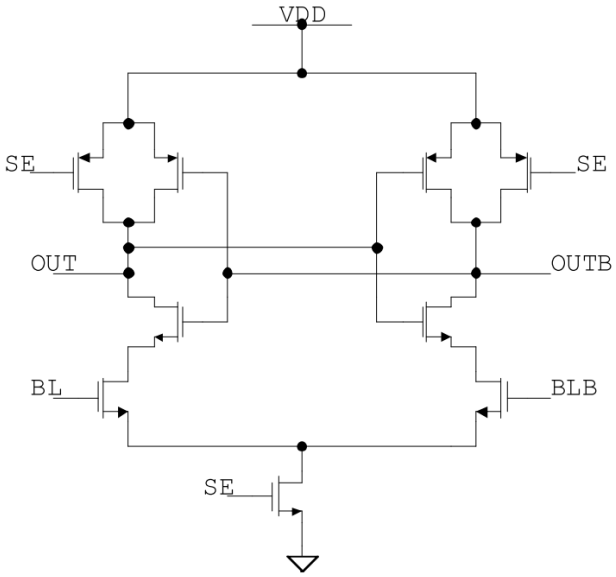


Figure A.3. Latch type Voltage Sense Amplifier (VLSA).

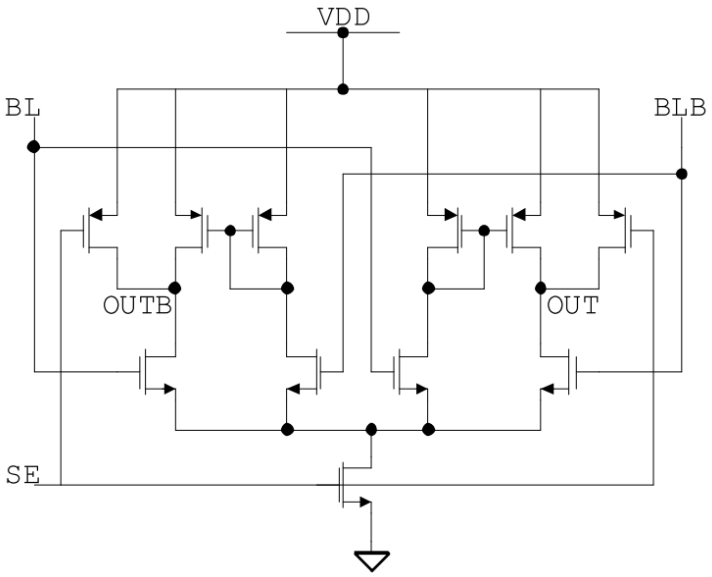


Figure A.4. Current Mirror Sense Amplifier (CMSA).

Appendix B

Python scripts

B.1 Python code for inputs generation

The following Python code is able to drive the proper signal pulses to the SOT STT MRAM cell (subsection 4.3.1) Cadence schematic following the specified operations sequence *operations* (line 60).
In practice, it writes into different files (1 file for each signal generator) a list of time values and amplitude values that describe the signals. These files are taken in input by *vpwlf* generators (i.e. generators in the Cadence schematic that take the input from a file), which drive the proper signal durations and amplitudes.
The script is adapted from the one reported in [100] for CMOS SRAM input generation. With this script, it is obtained the signals plot in Figure 4.19.

```
1  # Python script for input signal generation for all the memory operations
2
3  vdd = 1
4  vss = -0.5
5  T_ck = 5e-10 # 0.5ns, if larger, the 0.5ns-long SOT pulse is a glitch
6  fall_ris_time = 1e-12
7
8  op = "idle"
9  jmax = 0 # used to set the cycle duration with respect to T_ck
10 WL_to_SE_delay = 4e-11 # 40ps
11
12 #cycles for writing the signal values to files
13 def write_generators_files(i, k, genfile_ptr, value):
```

```
14      print(i)
15      jmax = 1
16      if (signal_set[key] == signal_set['SE']) and (op == "read"): # it
        ↪  consider the WL_to_SE delay
17          if i == 0:
18              genfile_ptr.write('0' + " " + '0' + "\n")
19          genfile_ptr.write(str(i * T_ck + WL_to_SE_delay) + " " + '0' +
            ↪  "\n")
20          genfile_ptr.write(str(i * T_ck + WL_to_SE_delay + fall_ris_time) +
            ↪  " " + str(value) + "\n")
21      else:
22          if i == 0:
23              genfile_ptr.write('0' + " " + str(value) + "\n")
24          else:
25              genfile_ptr.write(str(i * T_ck + fall_ris_time) + " " +
                ↪  str(value) + "\n")
26      if op == "write0" or op == "write1": # since the write cycle is 4*T_ck
        ↪  = 2ns long
27          jmax = 4
28      elif op == "read": # since the read cycle is 2*T_ck = 1ns long
29          jmax = 2
30      for j in range(jmax): # writes all the signals
31          print(j)
32          if (signal_set[key] == signal_set['WBL_EN']) and (op == "write0"
            ↪  or op == "write1") and (j == 1):
33              value = 0
34              genfile_ptr.write(str((i+1) * T_ck + fall_ris_time) + " " +
                ↪  str(value) + "\n")
35
36          else:
37              genfile_ptr.write(str((i + 1 + j) * T_ck) + " " + str(value) +
                ↪  "\n")
38      return jmax
39
40  # create the group of signals as a dictionary in python:
41  signal_set = {'WL0': {'genfile_ptr': None, 'value': 0, 'default': 0},
42                'RBL': {'genfile_ptr': None, 'value': vss, 'default': vss},
43                'RBL_EN': {'genfile_ptr': None, 'value': vss, 'default':
                  ↪  vss},
```

```
44                'RBL_disch': {'genfile_ptr': None, 'value': vdd, 'default':
                  ↪  vdd},
45                'WBL_EN': {'genfile_ptr': None, 'value': 0, 'default': 0},
46                'WBL_disch': {'genfile_ptr': None, 'value': vdd, 'default':
                  ↪  vdd},
47                'SE': {'genfile_ptr': None, 'value': 0, 'default': 0},
48                'ref_EN': {'genfile_ptr': None, 'value': 0, 'default': 0}}
49
50  path = "C:\\Users\\HP\\Desktop\\PYTHON\\gen_files_python\\" # the path to
    ↪  the folder where the files are saved
51
52  # open all files in writing:
53  for key in signal_set:
54      signal_set[key]['genfile_ptr'] = open((path+key+".csv"), "w+")  # open
        ↪  files for both reading and writing
55      # print(key)
56
57  # -->open RBL.csv, WBL.csv, SE.csv, etc.
58
59  # tupla for including all memory operations:
60  operations = ("write0", "idle", "read", "idle", "write1", "idle", "read",
    ↪  "idle") # example operations sequence
61
62  #other example sequences:
63  # operations = ("read", "idle", "read")
64  # operations = ("write0", "idle", "read", "idle")
65
66  i = 0  # first cycle
67  j = 0
68
69  for op in operations: # it associated the proper signal value depending on
    ↪  the selected operation
70          for key in signal_set:
71              signal_set[key]['value'] = signal_set[key]['default']
72          if op == "write0":
73                  signal_set['WL0']['value'] = vdd
74                  signal_set['RBL_EN']['value'] = vdd
75                  signal_set['RBL_disch']['value'] = 0
76                  signal_set['WBL_EN']['value'] = vdd
```

```
77                  signal_set['WBL_disch']['value'] = 0
78
79          elif op == "write1":
80                  signal_set['WL0']['value'] = vdd
81                  signal_set['RBL']['value'] = vdd
82                  signal_set['RBL_EN']['value'] = vdd
83                  signal_set['RBL_disch']['value'] = 0
84                  signal_set['WBL_EN']['value'] = vdd
85                  signal_set['WBL_disch']['value'] = 0
86
87          elif op == "read":
88                  signal_set['WL0']['value'] = vdd
89                  signal_set['RBL_disch']['value'] = 0
90                  signal_set['WBL_disch']['value'] = 0
91                  signal_set['SE']['value'] = vdd
92                  signal_set['ref_EN']['value'] = vdd
93
94          elif op == "idle": # keeps the signals to their default values
95              pass
96
97          else:
98              print("Error")
99              exit(1)
100
101         for key in signal_set:
102             print(key)
103             print(op)
104             jmax = write_generators_files(i, k,
                ↪  signal_set[key]['genfile_ptr'], signal_set[key]['value'])
105         i = i + jmax
106
107 for key in signal_set: # close all the files
108     signal_set[key]['genfile_ptr'].close()
```

B.2 Python code for power consumption computing in SOT STT MRAM arrays

The following Python script is used for generate the power function —that has to be included among the Cadence simulation outputs— for compute the power consumption of the SOT STT MRAM memory, as discusses in subsection 4.4.2.

```
# Python script for generating the sum of all the V*I contribution in the SOT STT MRAM schematic (for all array sizes)
import sys

array_size = int(sys.argv[1])

file_ptr = open(("power_calc_no_logic_"+str(array_size)+"x"+str(array_size)+".txt"), 'w')
for i in range(array_size):
    file_ptr.write("VT(\"/WL<"+str(i)+">\")*IT(\"/V_WL"+str(i)+"/MINUS\") + ")
    file_ptr.write("VT(\"/IN<" + str(i) + ">\")*IT(\"/V_IN" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/RBL_EN<" + str(i) + ">\")*IT(\"/V_RBL_EN" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/WBL_EN<" + str(i) + ">\")*IT(\"/V_WBL_EN" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/WBL_disch<" + str(i) + ">\")*IT(\"/V_WBL_disch" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/disch<" + str(i) + ">\")*IT(\"/V_disch" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/SE<" + str(i) + ">\")*IT(\"/V_SE" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/SE2<" + str(i) + ">\")*IT(\"/V_SE2_" + str(i) + "/MINUS\") + ")
file_ptr.write("VT(\"/vdd!\")*IT(\"/V_vdd/MINUS\") + ")
file_ptr.write("VT(\"/vss!\")*IT(\"/V_vss/MINUS\")")

file_ptr.close()
```

B.3 Python code for power consumption computing in SOT STT *analog adder* CAM-like arrays

The following Python script is used for generate the power function —that has to be included among the Cadence simulation outputs— for compute the power consumption of the SOT STT *analog adder* CAM-like arrays (section 5.5).

```
# Python script for generating the sum of all the V*I contribution in the SOT STT CAM-like schematic for the implemntation of the analog adder LiM approach (for all array sizes)
import sys

array_size = int(sys.argv[1])

file_ptr = open(("power_calc_"+str(array_size)+".txt"), 'w')
for i in range(array_dim):

    file_ptr.write("VT(\"/RWL<"+str(i)+">\")*IT(\"/V_RWL"+str(i)+"/MINUS\") + ")
    file_ptr.write("VT(\"/WWL<" + str(i) + ">\")*IT(\"/V_WWL" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/RBLa<" + str(i) + ">\")*IT(\"/V_RBLa" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/RBLb<" + str(i) + ">\")*IT(\"/V_RBLb" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/WBL_EN<" + str(i) + ">\")*IT(\"/V_WBL_EN" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/WBL_disch<" + str(i) + ">\")*IT(\"/V_WBL_disch" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/search_n<" + str(i) + ">\")*IT(\"/Vsearch_n" + str(i) + "/MINUS\") + ")
    file_ptr.write("VT(\"/disch<" + str(i) + ">\")*IT(\"/V_disch" + str(i) + "/MINUS\") + ")
```

```
16      file_ptr.write("VT(\"/SE<" + str(i) + ">\")*IT(\"/V_SE" + str(i) +
        ↪  "/MINUS\") + ")
17      file_ptr.write("VT(\"/SE2<" + str(i) + ">\")*IT(\"/V_SE2_" + str(i) +
        ↪  "/MINUS\") + ")
18  file_ptr.write("VT(\"/vg_load\")*IT(\"/Vvg_load/MINUS\") + ")
19  file_ptr.write("VT(\"/vg_pMOS!\")*IT(\"/Vvg_pMOS/MINUS\") + ")
20  file_ptr.write("VT(\"/vdd!\")*IT(\"/V0/MINUS\")")
21
22  file_ptr.close()
```

Bibliography

[1] Guillaume Prenat, Kotb Jabeur, Gregory Di Pendina, Olivier Boulle, Gilles Gaudin, *Beyond STT-MRAM, Spin Orbit Torque RAM SOT-MRAM for High Speed and High Reliability Applications*, Introduction, 2015.

[2] Valery Lapshinsky, L. N. Patrikeev, *Emerging resistive random-access memory for 'fog' computing and IoT: materials and structural options taxonomy*, International Journal of Nanotechnology, March 2020.

[3] Dr. Meng Zhu, Dr. Roman Sappey, Jeff Barnum, *MRAM Process Development And Production Briefing*, 10 March 2020. https://semiengineering.com/mram-process-development-and-production-briefing/

[4] Gan Fuxi, Wang Yang, *Data Storage at the Nanoscale -Advances and Applications*, p. 284, 2015.

[5] Jagan Singh Meena, Simon Min Sze, Umesh Chand, Tseung-Yuen Tseng, *Overview of emerging nonvolatile memory Technologies*, 25 September 2014.

[6] Gan Fuxi, Wang Yang, *Data Storage at the Nanoscale -Advances and Applications*, p. 311, 2015.

[7] MRAM-Info: the MRAM experts, *STT-MRAM: Introduction and market status*, 19 February 2019.

[8] Y. Huai, *Spin-Transfer Torque MRAM (STT-MRAM): Challenges and Prospects*, 2008.

[9] Furqan Zahoor, Tun Zainal Azni Zulkifli, Farooq Ahmad Khanday, *Resistive Random Access Memory (RRAM) an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (mlc) Storage, Modeling, and Applications*, 22 april 2020.

[10] Ashish Ranjan, Swagath Venkataramani, Zoha Pajouhi, Rangharajan

Venkatesan, Kaushik Roy, Anand Raghunathan, *STAxCache: An Approximate, Energy Efficient STT-MRAM Cache*, Purdue University, (Abstract), 2017.

[11] Shubham Jain, Ashish Ranjan, Kaushik Roy, Anand Raghunathan, *Computing in Memory with Spin-Transfer Torque Magnetic RAM*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 26, No. 3, 70-483, 2017.

[12] Avalanche technology, *MRAM Technology*, 2020. https://www.avalanche-technology.com/technology/mram-technology/

[13] Valerio Milo, Gerardo Malavena, Christian Monzio Compagnoni, Daniele Ielmini, *Memristive and CMOS Devices for Neuromorphic Computing*, 1 January 2020.

[14] Tetsuo Endoh, Hiroaki Honjo, *A Recent Progress of Spintronics Devices for Integrated Circuit Applications*, Tohoku University, 13 November 2018.

[15] MRAM-Info: the MRAM experts, *SOT-MRAM: introduction and market status*, 17 September 2020.

[16] S. G. Hu, S. Y. Wu, W. W. Jia, Q. Yu, L. J. Deng, Y. Q. Fu, Y. Liu, T. P. Chen, *Review of Nanostructured Resistive Switching Memristor and Its Applications*, 2014.

[17] Shimeng Yu and Pai-Yu Chen, *Emerging Memory Technologies -Recent Trends and Prospects*, 21 June 2016.

[18] W. Kim, S. Menzel, D. J. Wouters, R. Waser, V. Rana, *3-Bit Multi Level Switching by Deep Reset Phenomenon in Pt/W/TaOX/Pt-ReRAM Devices*, 2016.

[19] Fujitsu>Memory Products, *What is FRAM?*, 2020. https://www.fujitsu.com/global/products/devices/semiconductor/memory/fram/overview/structure/

[20] Mark Lapedus, *What Are FeFETs?*, Semiconductor Engineering, 16 February 2017.

[21] H. Kohlstedt, Y. Mustafa, A. Gerber, A. Petraru, M. Fitsilis, R. Meyer, U. Böttger and R Waser, *Current status and challenges of ferroelectric memory devices*, 2005.

[22] Xunzhao Yin, Ahmedullah Aziz, Joseph Nahas, Suman Datta, Sumeet Gupta, Michael Niemier, Xiaobo Sharon Hu, *Exploiting Ferroelectric FETs for Low-Power Non-Volatile Logic-in-Memory Circuits*, 2016.

[23] T. Ali, P. Polakowski, S. Riedel, T. Büttner, T. Kämpfe, M. Rudolph, B. Pätzold, K. Seidel, D. Löhr, R. Hoffmann, M. Czernohorsky, K. Kühnel, P. Steinke, J. Calvo, K. Zimmermann, J. Müller, *High Endurance Ferroelectric Hafnium Oxide-Based FeFET Memory Without Retention*

Penalty, 2018.

[24] J. A. Caraveo-Frescas, M. A. Khan, H. N. Alshareef, *Polymer ferroelectric field-effect memory device with SnO channel layer exhibits record hole mobility*, 2014.

[25] Panni Wang, Zheng Wang, Wonbo Shim, Jae Hur, Suman Datta, Asif Islam Khan, Shimeng Yu, *Drain-Erase Scheme in Ferroelectric Field-Effect Transistor-Part I: Device Characterization*, February 2020.

[26] FMC -The ferroelectric Memory Company, *Major differentiation to competition*, 2020. https://ferroelectric-memory.com/technology/major-differentiation-to-competition/

[27] O.D. Alao, J.V. Joshua, D.O. Kehinde, E.O. Ehinlafa, M.O. Agbaje, J.E.T Akinsola, *Emerging Memory Technologies*, p.63, Babcock University, 2016.

[28] Scott W. Fong, Christopher M. Neumann, and H.-S. Philip Wong, *Phase-Change Memory—Towards a Storage-Class Memory*, 2017.

[29] Stefania Braga, Alessandro Cabrini and Guido Torelli, *An Integrated Multi-Physics Approach to the Modeling of a Phase Change Memory Device*, University of Pavia, 2008.

[30] I. V. Karpov, M. Mitra, D. Kau, G. Spadini, Y. A. Kryukov, V. G. Karpov, *Fundamental drift of parameters in chalcogenide phase change memory*, Journal of Applied Physics, 2007.

[31] Kerem Akarvardar, H.-S. Philip Wong, *Nanoelectromechanical Logic and Memory Devices*, Stanford University, 2009.

[32] Fred Chen, Hei Kam, Dejan Markovic, Tsu-Jae King Liu, Vladimir Stojanovic, Elad Alon, *Integrated Circuit Design with NEM Relays*, 2008.

[33] Dimitrios Tsamados, Adrian Ionescu, Kerem Akarvardar, H.-S. Philip Wong, Elad Alon, Tsu-Jae King Liu, *Nanoelectromechanical Switches (NEM Relays NEMFETs)*, 2008.

[34] Tasuku Nagami, Yoshishige Tsuchiya, Ken Uchida, Hiroshi Mizuta, Shunri Oda, *Scaling Analysis of Nanoelectromechanical Memory Devices*, Tokyo Institute of Technology, 2010.

[35] Jae Eun Jang, Seung Nam Cha, Young Jin Choi, Dae Joon Kang, Tim P. Butler, David G. Hasko, Jae Eun Jung, Jong Min Kim, Gehan A. J. Amaratunga, *Nanoscale memory cell based on a nanoelectromechanical switched capacitor*, 23 December 2007.

[36] W. Kwon, *Nano-electromechanical random access memory (RAM) devices*, University of California Berkeley, 2014.

[37] Kimihiko Kato, Vladimir Stojanovic, Tsu-Jae King Liu, *Non-Volatile Nano-Electro-Mechanical Memory for Energy-Efficient Data Searching*,

January 2016.

[38] Antonino Ferrara, Umberto Garlando, Luca Gnoli, Giulia Santoro, Maurizio Zamboni, *3D Design of a pNML Random Access Memory*, Politecnico di Torino, 2017.

[39] Jason Hoffman, Xiao Pan, James W. Reiner, Fred J. Walker, J. P. Han, Charles H. Ahn, T. P. Ma, *Ferroelectric Field Effect Transistor for Memory Applications*, 2010.

[40] Dayane Reis, Michael Niemier, X. Sharon Hu, *Computing in memory with FeFETs*, Proceedings of the International Symposium on Low Power Electronics and Design, 2018.

[41] Gan Fuxi, Wang Yang, *Data Storage at the Nanoscale -Advances and Applications*, pp. 327-328, 2015.

[42] Kazi Asifuzzaman, Rommel Sánchez Verdejo, Petar Radojkovic, *Enabling a Reliable STT-MRAM Main Memory Simulation*, Proceedings of the International Symposium on Memory Systems, 2017.

[43] Insik Yoon, Ashwin Chintaluri, Arijit Raychowdhury, *EMACS: Efficient MBIST architecture for test and characterization of STT-MRAM arrays*, 2016 IEEE International Test Conference (ITC), IEEE, 2016.

[44] Jin Woong Kwak, Andrew Marshall, Harvey Stiegler, *28nm STT-MRAM Array and Sense Amplifier*, 2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), IEEE, 2019.

[45] Jongyeon Kim, An Chen, Behtash Behin-Aein, Saurabh Kumar, Jian-Ping Wang, Chris H. Kim, *A Technology-Agnostic MTJ SPICE Model with User-Defined Dimensions for STT-MRAM Scalability Studies*, 2015 IEEE custom integrated circuits conference (CICC), IEEE, 2015.

[46] Meenu Rani Garg, Anu Tonk, *A Study of Different Types of Voltage Current Sense Amplifiers used in SRAM*, International Journal of Advanced Research in Computer and Communication Engineering 4, No. 5, 30-35, 2015.

[47] Subhodip Maulik, Mili Sarkar, Srismrita Basu, *Comparative Study on Different Types of Sense Amplifiers for Delay and Power Dissipation Calculation.*

[48] Mesbah Uddin, Garrett S. Rose, *A Practical Sense Amplifier Design for Memristive Crossbar Circuits (PUF)*, 31st International System-on-Chip Conference (SOCC), Arlington, VA, USA, September 2018.

[49] Weisheng Zhao, Claude Chappert, Virgile Javerliac, Jean-Pierre Nozière, *High Speed, High Stability and Low Power Sensing Amplifier for MTJ/CMOS Hybrid Logic Circuits*, IEEE Transactions on Magnetics, Vol. 45, No. 10, October 2009.

[50] Cong Xu, Dimin Niu, Naveen Muralimanohar, Rajeev Balasubramonian, Tao Zhang, Shimeng Yu, Yuan Xie, *Overcoming the Challenges of Crossbar Resistive Memory Architectures*, 2015.

[51] Dimin Niu, Cong Xu, Naveen Muralimanohar, Norman P. Jouppi, Yuan Xie, *Design of Cross-point Metal-oxide ReRAM Emphasizing Reliability and Cost*, 2013.

[52] Jiahao Yin, Chunmeng Dou, Danian Dong, Jie Yu, Xiaoxin Xu, Qing Luo, Tiancheng Gong, Lu Tai, Peng Yuan, Xiaoyong Xue, Ming Liu, and Hangbing Lv, *A 0.75 V reference clamping sense amplifier for low-power high-density ReRAM with dynamic pre-charge technique*, IEICE Electronics Express, Vol.16, No.12, 1-6, 2019.

[53] Richard Fackenthal, Makoto Kitagawa, Wataru Otsuka, Kirk Prall, Duane Mills 1 , Keiichi Tsutsui, Jahanshir Javanifard, Kerry Tedrow, Tomohito Tsushima, Yoshiyuki Shibahara, Glen Hush, *A 16Gb ReRAM with 200MB/s Write and 1GB/s Read in 27nm Technology*, 2014 IEEE International Solid-State Circuits Conference, 2014.

[54] Tz-yi Liu, *A 130.7-mm^2-Layer 32-Gb ReRAM Memory Device in 24-nm Technology, IEEE Journal od Solid-State Circuits*, Vol. 49, No. 1, January 2014.

[55] Frederick Perner, *Sense amplifier for reading a crossbar memory array*, U.S. Patent No. 8,472,262, 25 June 2013.

[56] Yi-Chung Chen, Hai (Helen) Li, Wei Zhang, *A Novel Peripheral Circuit for RRAM-based LUT*, 2012 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, 2012.

[57] Adedotun Adeyemo, Abusaleh Jabir, Jimson Mathew, *Minimising Impact of Wire Resistance in Low-Power Crossbar Array Write Scheme*, Journal of Low Power Electronics 13, No. 4 , pp. 649-660, 2017.

[58] Cong Xu, Xiangyu Dong, Norman P. Jouppi, Yuan Xie, *Design Implications of Memristor-Based RRAM Cross-Point Structures*, 2011 Design, Automation Test in Europe, IEEE, 2011.

[59] H-S. Philip Wong, Simone Raoux, SangBum Kim, Jiale Liang, John P. Reifenberg, Bipin Rajendran, Mehdi Asheghi, Kenneth E. Goodson, *Phase change memory*, Proceedings of the IEEE 98, No. 12, 2201-2227, 2010.

[60] Kwang-Jin Lee, Beak-Hyung Cho, Woo-Yeong Cho, Sangbeom Kang, Byung-Gil Choi, Hyung-Rok Oh, Chang-Soo Lee et al., *A 90 nm 1.8 V 512 Mb diode-switch PRAM with 266 MB/s read throughput*, IEEE Journal of Solid-State Circuits 43, No. 1, 150-162, 2008.

[61] Lei Jiang, Bo Zhao, Jun Yang, Youtao Zhang, *A low power and reliable charge pump design for phase change memories*, ACM SIGARCH Computer Architecture News 42, No. 3, 397-408, 2014.

[62] Sangbeom Kang, Woo Yeong Cho, Beak-Hyung Cho, Kwang-Jin Lee, Chang-Soo Lee, Hyung-Rok Oh, Byung-Gil Choi et al., *A 0.1 μm 1.8 V 256 Mb Phase-Change Random Access Memory (PRAM) With 66 Mhz Synchronous Burst-Read Operation*, IEEE Journal of Solid-State Circuits 42, No. 1, 210-218, 2006.

[63] W. Y. Cho, B. H. Cho, B. G. Choi et al., *A 0.18 μm 3 V 64 Mb Non-Volatile Phase-Transition Random Access Memory (PRAM)*, ISSCC-Digest of Technical Papers 2.1., 2004.

[64] Xi Li, Hou-Peng Chen, Zhi-Tang Song, *Design and analysis of a high-performance sense amplifier for Phase-Change Memory*, 2011 3rd International Conference on Computer Research and Development, Vol. 3, pp. 318-321. IEEE, 2011.

[65] Panni Wang, Wonbo Shim, Zheng Wang, Jae Hur, Suman Datta, Asif Islam Khan, Shimeng Yu, *Drain-Erase Scheme in Ferroelectric Field Effect Transistor -Part II: 3-D-NAND Architecture for In-Memory Computing*, IEEE Transactions on Electron Devices 67, No. 3, 962-967, 2020.

[66] Kai Ni, Xueqing Li, Jeffrey A. Smith, Matthew Jerry, Suman Datta, *Write disturb in ferroelectric FETs and its implication for 1T-FeFET AND memory arrays*, IEEE Electron Device Letters 39, No. 11, 1656-1659, 2018.

[67] Ya Qin, Ying Xiong, Kai Li, Minghua Tang, *Simulation of FeFET-based basic logic circuits and current sense amplifier*, In 2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), pp. 1-3. IEEE, 2014.

[68] Sumitha George, Kaisheng Ma, Ahmedullah Aziz, Xueqing Li, Asif Khan, Sayeef Salahuddin, Meng-Fan Chang et al., *Nonvolatile memory design based on ferroelectric FETs*, In Proceedings of the 53rd Annual Design Automation Conference, pp. 1-6, 2016.

[69] Pilin Junsangsri, Jie Han, Fabrizio Lombardi, *Logic-in-memory with a nonvolatile programmable metallization cell*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24, No. 2, 521-529, 2015.

[70] Dooho Cho, Kyungmin Kim, Changsik Yoo, *A Programmable Logic-in-memory (LiM) based on Magnetic Tunneling Junction (MTJ)*, Journal of Semiconductor Technology and Science 18, No. 5, 586-592, 2018.

[71] Sandeep Krishna Thirumala, Shubham Jain, Anand Raghunathan,

Sumeet Kumar Gupta, *Non-volatile memory utilizing reconfigurable ferroelectric transistors to enable differential read and energy-efficient in-memory computation*, In 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), pp. 1-6. IEEE, 2019.

[72] Hiwa Mahmoudi, Thomas Windbacher, Viktor Sverdlov, Siegfried Selberherr, *MRAM-based logic array for large-scale non-volatile logic-in-memory applications*, In 2013 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), pp. 26-27, IEEE, 2013.

[73] Shahar Kvatinsky, Guy Satat, Nimrod Wald, Eby G. Friedman, Avinoam Kolodny, Uri C. Weiser, *Memristor-based material implication (IMPLY) logic: Design principles and methodologies*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 22, No. 10, 2054-2066, 2013.

[74] Sundarapandian Vaidyanathan, Christos Volos, eds., *Advances in memristors, memristive devices and systems*, Vol. 701, pp. 131-156 , Springer, 2017.

[75] Shahar Kvatinsky, Dmitry Belousov, Slavik Liman, Guy Satat, Nimrod Wald, Eby G. Friedman, Avinoam Kolodny, Uri C. Weiser, *MAGIC - Memristor aided logic*, IEEE Transactions on Circuits and Systems II: Express Briefs 61, No. 11, 895-899, 2014.

[76] Rahul Gharpinde, Phrangboklang Lynton Thangkhiew, Kamalika Datta, Indranil Sengupta, *A scalable in-memory logic synthesis approach using memristor crossbar*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 26, No. 2, 355-366, 2017.

[77] Nishil Talati, Saransh Gupta, Pravin Mane, Shahar Kvatinsky, *Logic design within memristive memories using memristor-aided loGIC (MAGIC)*, IEEE Transactions on Nanotechnology 15, No. 4, 635-650, 2016.

[78] Yuanfan Yang, Jimson Mathew, Salvatore Pontarelli, Marco Ottavi, Dhiraj K. Pradhan, *Complementary resistive switch-based arithmetic logic implementations using material implication*, IEEE Transactions on nanotechnology 15, No. 1, 94-108, 2015.

[79] Yang Zhang, Yi Shen, Xiaoping Wang, Yanwen Guo, *A novel design for a memristor-based or gate*, IEEE Transactions on Circuits and Systems II: Express Briefs 62, No. 8, 781-785, 2015.

[80] Abu Sebastian, Manuel Le Gallo, Evangelos Eleftheriou, *Computational phase-change memory: Beyond von Neumann computing*, Journal of Physics D: Applied Physics 52, No. 44, 443002, 2019.

[81] Desmond Loke, Jonathan M. Skelton, Wei-Jie Wang, Tae-Hoon Lee,

Rong Zhao, Tow-Chong Chong, Stephen R. Elliott, *Ultrafast phase-change logic device driven by melting processes*, Proceedings of the National Academy of Sciences 111, No. 37, 13272-13277, 2014.

[82] E. T. Breyer, H. Mulaosmanovic, T. Mikolajick, S. Slesazeck, *Reconfigurable NAND/NOR logic gates in 28 nm HKMG and 22 nm FD-SOI FeFET technology*, In 2017 IEEE International Electron Devices Meeting (IEDM), pp. 28-5. IEEE, 2017.

[83] Evelyn T. Breyer, Halid Mulaosmanovic, Stefan Slesazeck, Thomas Mikolajick, *Demonstration of versatile nonvolatile logic gates in 28nm HKMG FeFET technology*, In 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1-5. IEEE, 2018.

[84] Panni Wang, Feng Xu, Bo Wang, Bin Gao, Huaqiang Wu, He Qian, Shimeng Yu, *Three-dimensional NAND flash for vector-matrix multiplication*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27, No. 4, 988-991, 2018.

[85] Xunzhao Yin, Xiaoming Chen, Michael Niemier, Xiaobo Sharon Hu, *Ferroelectric FETs-based nonvolatile logic-in-memory circuits*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems 27, No. 1, 159-172, 2018.

[86] Sandeep Krishna Thirumala, Shubham Jain, Sumeet Kumar Gupta, Anand Raghunathan, *Ternary compute-enabled memory using ferroelectric transistors for accelerating deep neural networks*, In 2020 Design, Automation Test in Europe Conference Exhibition (DATE), pp. 31-36, IEEE, 2020.

[87] Xizhen Zhang, Mitsue Takahashi, Shigeki Sakai, *FeFET logic circuits for operating a 64 kb FeNAND flash memory array*, Integrated Ferroelectrics 132, No. 1, 114-121, 2012.

[88] School of Microelectronics, Beihang University, *Spinmodel Library, Know more about spin!*, http://www.spinlib.com/index.html.

[89] School of Microelectronics, Beihang University, *Spinmodel Library, Know more about spin!*, http://www.spinlib.com/STT_PMA_MTJ.html.

[90] School of Microelectronics, Beihang University, *Spinmodel Library, Know more about spin!*, http://www.spinlib.com/STT_SOT_MTJ.html.

[91] Wang, Zhaohao, Weisheng Zhao, Erya Deng, Jacques-Olivier Klein, and Claude Chappert, *Perpendicular-anisotropy magnetic tunnel junction switched by spin-Hall-assisted spin-transfer torque*, Journal of Physics D: Applied Physics 48, no. 6 (2015): 065001.

[92] Zhang, He, Wang Kang, Lezhi Wang, Kang L. Wang, and Weisheng

Zhao, *Stateful reconfigurable logic via a single-voltage-gated spin Hall-effect driven magnetic tunnel junction in a spintronic memory*, IEEE Transactions on Electron Devices 64, no. 10 (2017): 4295-4301.

[93] He, Zhezhi, Shaahin Angizi, Farhana Parveen, and Deliang Fan, *High performance and energy-efficient in-memory computing architecture based on sot-mram*, In 2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH), pp. 97-102. IEEE, 2017.

[94] Fan, Deliang, and Shaahin Angizi, *Energy efficient in-memory binary deep neural network accelerator with dual-mode SOT-MRAM*, In 2017 IEEE International Conference on Computer Design (ICCD), pp. 609-612. IEEE, 2017.

[95] Uddin, Mesbah, and Garrett S. Rose, *A practical sense amplifier design for memristive crossbar circuits (PUF)*, In 2018 31st IEEE International System-on-Chip Conference (SOCC), pp. 209-214. IEEE, 2018.

[96] Na, Taehui, Seung H. Kang, and Seong-Ook Jung, *STT-MRAM Sensing: A Review*, IEEE Transactions on Circuits and Systems II: Express Briefs (2020).

[97] Rahimi, Abbas, Pentti Kanerva, and Jan M. Rabaey, *A robust and energy-efficient classifier using brain-inspired hyperdimensional computing*, In Proceedings of the 2016 International Symposium on Low Power Electronics and Design, pp. 64-69. 2016.

[98] Imani, Mohsen, Abbas Rahimi, Deqian Kong, Tajana Rosing, and Jan M. Rabaey, *Exploring hyperdimensional associative memory*, In 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 445-456. IEEE, 2017.

[99] Xu, Wei, Tong Zhang, and Yiran Chen, *Spin-transfer torque magnetoresistive content addressable memory (CAM) cell structure design with enhanced search noise margin*, In 2008 IEEE International Symposium on Circuits and Systems, pp. 1898-1901. IEEE, 2008.

[100] Graziano, Mariagrazia, Marco Vacca, Maurizio Zamboni, and Fabrizio Ottati, *ALiAS-Analog Logic-in-Memory Arrays Synthesizer*, 2020.