Evaluation of computation-in-memory using traditional (SRAM) and emerging non-volatile devices (memristors)

by

W. Sewnarain

to obtain the degree of Master of Science

at the Delft University of Technology.

Student number: 4494245

Supervisor:	Prof. Dr. Said Hamdioui
Daily supervisor:	Dr. Ing. Anteneh Gebregiorgis
Committee member:	Dr. Chang Gao

TU Delft

Abstract

Modern computer application require large amounts of data processing. Traditional computing models involve constant data transfer between memory and processor. This data transfer is a major contributor to high energy consumption. As these applications scale, the energy demand increases. This poses challenges in terms of sustainability and operational costs. Computation In Memory (CIM) integrates processing within the memory. This reduces the need for data transfer between memory and processor. Potential for drastically lowering energy consumption.

CIM macros are often implemented using modified SRAM cells, though recent literature explores memristor-based CIM designs due to the memristor's low-energy, non-volatile characteristics. However, no comprehensive comparisons between SRAM-based and memristor-based CIM designs exist. While memristor-based designs are hypothesized to be more energy-efficient, this has not yet been proven.

This thesis compares SRAM-based and memristor-based CIM designs to determine which is better suited for CIM applications. This has been achieved by exploring the state of the art of memristive devices, memristor based CIM macros and SRAM based CIM macros. A selection of designs were chosen to compare, including the 1T1R and 8T SRAM design, which are the most popular memristor based and SRAM based CIM designs. The schematics of all the designs were recreated and simulated using as much of the same parameters as possible in all of the designs. A simulation of performing the logic AND and the MAC operation was made. Additionally a layout of the designs was made to extract the area. The designs were compared based on area, energy consumption and delay.

From the results could be concluded that the best device for CIM depends on the application. The memristor design had the smallest area and consumed the least amount of energy for reading, logic and MAC operations. The memristor design also consumed the most amount of energy during writing and the delay for all operations is longer than with the SRAM based designs. If area, energy consumption and delay are equally important for an application, then memristor based CIM would be the better choice only if there are much more logic/read operations than write operations. It could be the better choice for MAC operations if a more energy efficient ADC was used than the one used in this thesis.

Contents

1 Introduction

Most modern computers are made using the Von Neumann architecture as shown in Figure 1.1. A computer using this architecture consist of:

- **A processing unit**. It contains an arithmetic-logic unit (ALU) and registers. The ALU performs the logic operations and the registers contain the data on which the operations need to be performed.
- **A control unit**. The control unit directs the operation of the processor. It contains an instruction register, which contain the instruction that make up the program, and a program counter.
- **Memory unit**. The memory unit is a separate memory which stores data and instructions (the RAM). When the central processing unit (CPU) needs to perform an operation on data which is not present in the CPU registers (a cache miss), it needs to look for the data in the RAM. The RAM is slower than the CPU cache, but is larger.

There is also external mass storage (HDD or SSD) which is even slower than RAM, but has a higher capacity.

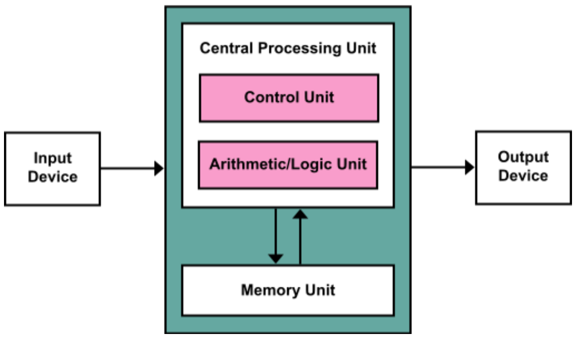


Figure 1.1: Von Neumann Architecture [1].

The Von Neumann architecture is used because of its ability to be easily reprogrammed. Before this architecture was used, computing machines used to be designed for a specific task. It would have a fixed program [2][3].

Since the introduction of the Von Neumann architecture, advancements of computing systems have been made in a couple of ways. The main driver was CMOS scaling [4]. This allowed more transistors to fit on a chip which increases performance, or made more chips to fit on a waver which reduces the cost of a chip. It also made higher clock speeds possible and lowered power usage by lowering the voltage threshold of the transistors. Since around 2005, performance increase has been realized by increasing the number of logical cores. This in combination with clever techniques like adding multi-threading, branch prediction, out of order execution, etc., has been giving us an exponential increase in computing performance. Unfortunately, this trend is declining as can be seen in Figure 1.2 [5].

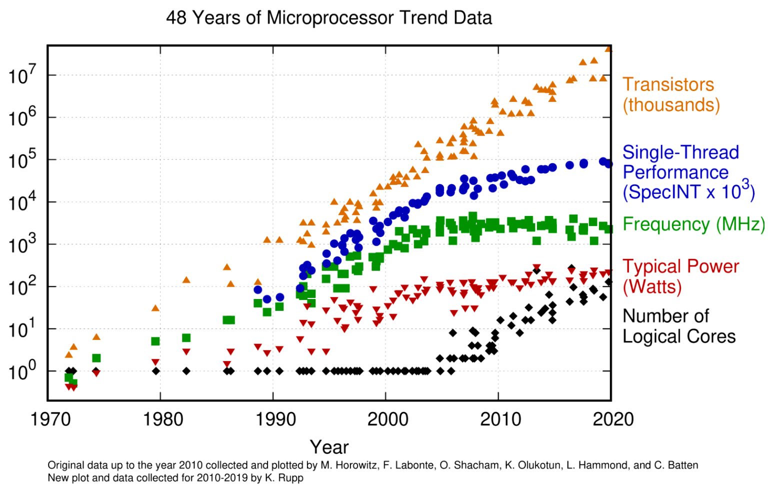


Figure 1.2: Microprocessor trend data [6].

The exponential performance increase of processors is declining because we have hit a multiple walls [7].

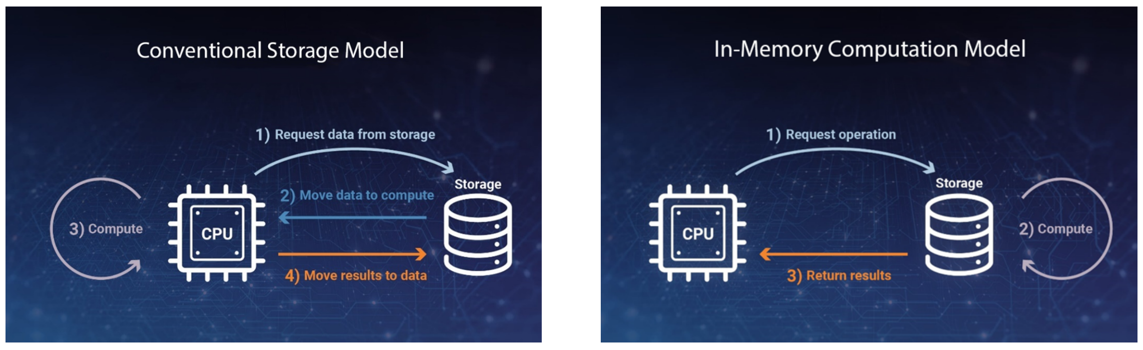
CMOS scaling is reaching its limits
When transistors are becoming smaller, they will have a shorter lifetime and higher failure rate [8]. The threshold voltage decreases, which increases the relative sub-threshold leakage [9]. The static leakage could become so large that it might even exceed the dynamic power [10]. CMOS scaling becomes less profitable because of the increased design complexity and test difficulty [11].

Memory can't keep up with the CPU *The memory wall*
Data-intensive applications have gained more importance in many domains such as health-care, AI, and economics [11]. In a computer made using the Von Neumann architecture, this means that there will be more frequent data transfer between the CPU and the memory. The processing speed of CPUs has been increasing at a higher rate than the speed of the memory [12]. This increasing gap in speed in combination with the limited memory bandwidth has a huge impact on computation performance. It cost a lot of energy to transfer data between the memory and processor, so frequent data transfer also comes with an increase in energy consumption.

We have reached a power limit *The power wall*
For a while, CPUs were operating at higher frequencies every generation in order to increase the performance. After around 2005, the operating frequency more or less stayed the same which can be seen in Figure 1.2. Increasing the clock speed further would increase the power usage and with that the heat generated by the processor. Due to cooling constraints it would be unfeasible to operate at these higher frequencies [11][12]. The cooling constraints are not the only reason for trying to keep the power usage low. There is an increasing demand for low powered microprocessors for battery powered devices like phones and IoT-devices. There might also be environmental or financial incentive to keep the energy usage of computers low.

It becomes harder to utilize all logical cores *The ILP wall*
Processors nowadays consist of multiple logical cores. An easy way to make your code run faster is to make different independent parts of the code run in parallel on different cores. The problem is that we now have so many cores that it becomes difficult to find enough parallelism to make good use of all of the cores [12].

Different computing paradigms and technologies are being explored to try and break the walls that computing systems are facing. One of these paradigms is computation-in-memory (CIM). A computing architecture for in-memory computing can process data in the same location where it is stored. In a computer with a Von Neumann architecture, the CPU would request data from storage. The data will be moved to the CPU. The CPU will perform computations. The results are moved back to the storage. With a CIM architecture, the CPU would request an operation from the storage. The storage will perform the computation and return the results. This is also illustrated in Figure 1.3.



(a) (b)

Figure 1.3: (a) Conventional storage model. (b) CIM model [13].

Recent CIM macros proposed in literature perform either logic or MAC operations. A CIM macro for logic can be useful for logic-heavy application like cryptographic security algorithms, edge devices, DNA pattern matching for bio-informatics applications and hamming distance calculation. A CIM macro for MAC can be useful for application with many vector-matrix multiplication operations, like neural networks, signal processing and image compression [14].

A CIM architecture has the potential to help break the memory wall, because if some of the computations can be performed by the memory, then the processor will need to access the memory less often. It could also help to solve the power wall, because transferring data between the memory and processor cost a lot of energy, so when the processor needs to access the memory less often, it will consume less energy. Although a crossbar of cells with a CIM architecture makes it possible to perform many calculations in parallel, the ILP wall refers to the difficulty to parallelize computer programs and CIM doesn't help with that.

1.1. Motivation

In Figure 1.4 is shown the basics of a CIM macro. A CIM architecture would consist of a crossbar of cells and below each column there would be some additional circuitry that can contain sense amplifiers and ADCs. For reading you would enable just one word-line. For logic operations you would enable two or more word-lines, and for MAC operations you supply a vector of inputs at the world-lines.

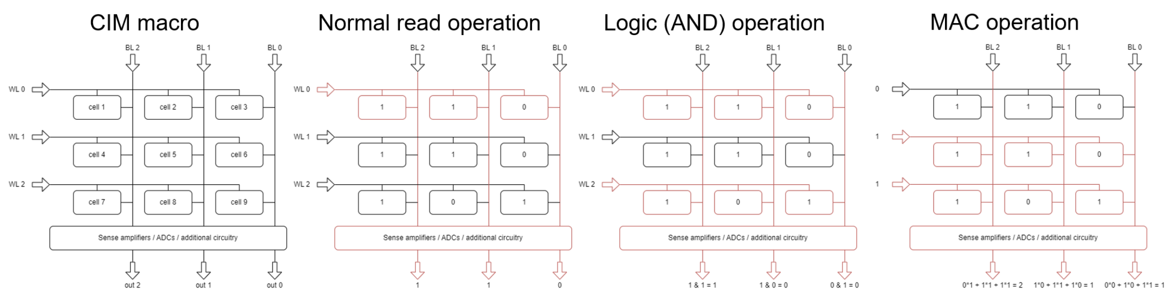


Figure 1.4: Basic idea of CIM.

The cells in the crossbar can be modified SRAM cells. It is also being explored to use a new type of non-volatile memory, called memristors. Non-volatile memory is a type of memory that can retain it's data without needing power. It has advantages over SRAM like not needing constant power and being smaller. The disadvantages include being slower than SRAM and having a lower endurance. More about these types of memory can be read in chapter 2.

1.2. Problem statement

Unfortunately, there are no comparisons of CIM using SRAM and memristors, so it is not clear which technology would be better suited for CIM. Conducting a meta-analysis by combining data from papers on memristor-based CIM and SRAM-based CIM is difficult. In Table 1.1 is an attempt at comparing different CIM macros. Each paper reports different metrics and different technology nodes, computation operations, voltages, crossbar size, clock frequency, etc. are used.

	8T SRAM [15]	**Push-Rule WL-Dec. 6T SRAM [16]**	**WL-Decoupled 4+2T SRAM [17]**	**8T SRAM CIM With ADC [18]**	**6T XNOR SRAM [19]**
Read delay (ps)	9.756				
Read power (nw)	417.4				
Read energy	4.069zj		3.96 fJ		
Write delay (ps)	62.85				
Write power (uw)	1.579				
Write energy (fj)	0.09924		4.02		
Logic latency	1ns		1.3 cycles		
Logic energy (fj/bit)	21.84	0.6	6.6		
Technology node	180nm	28nm	55nm DDC	65nm	28nm
V_{DD}		1 V	Read: 0.25 V Write: 0.35 V Logic: 0.35 V	0.45V / 0.8V	1.0 V
Clock frequency		787 MHz	Read: 4 MHz Write: 2.5 MHz Logic: 15 MHz	200 MHz	25 MHz
Energy/operation(fJ/Op)				2.04	3.33
Energy efficiency (TOPS/W)				490-15.8	
Transistors	8T	6T	6T	8T	6T
Area/cell (μm^2)		0.152		3.35	0.362
Area/array				0.055 mm^2	
Array size		64x64	128x128	128x128	256x64
Operations	Logic	Logic	Logic	MAC	MAC

Table 1.1: Attempt at comparing different CIM macros using data available in literature

A common hypothesis is that memristor-based CIM macros will have higher energy efficiency and occupy a smaller area than SRAM-based CIM macros. However, there is no data proving or disproving this hypothesis. Therefore, it is not possible to make a fair comparison of the two technologies in a CIM macro using the available data in the literature.

1.3. Thesis contribution

This thesis seeks to answer the question: *Which technology (memristor or SRAM) is better suited for computation-in-memory?* This will be accomplished not merely by comparing memristor-based and SRAM-based CIM macros from the literature, but by recreating these designs, simulating them under comparable parameters, and analyzing the simulation results. Through this approach, the following contributions are made:

Implementation of various SRAM-based CIM designs
Three SRAM-based CIM designs were recreated and tested in this work: Push-Rule WL-Decoupled

6T SRAM, WL-Decoupled 4+2T SRAM, and 8T SRAM. These designs were selected based on their emphasis on logic and MAC (multiply-accumulate) operations, which are central to CIM applications. Schematics from prior works were used as a foundation. Each design was implemented using a consistent set of parameters (e.g., technology node, voltages) to ensure a fair comparison. By simulating these designs under identical conditions, this thesis provides a unified basis for comparison.

Implementation of the 1T1R design
The 1T1R design, a popular memristor-based CIM macro, was recreated and analyzed for both logic and MAC operations. This design leverages resistive memory to perform computations in the analog domain, an approach with promising implications for energy efficiency. ReRAM (Resistive Random Access Memory) was chosen as the representative memristor technology due to its scalability and multi-level cell capability. A SPICE model of the ReRAM was used for the recreation of the 1T1R design. While memristor-based designs are hypothesized to excel in energy efficiency and area utilization, empirical comparisons with SRAM-based designs are limited. By implementing a 1T1R design, this research bridges the gap in direct comparisons.

Delay, power, and layout-based area estimation
To ensure a comprehensive evaluation, each design was assessed in terms of delay, power consumption, and area efficiency. These metrics are critical in determining the suitability of CIM designs for real-world applications. Each design was implemented in a simulation environment, where logic and MAC computations were tested. Layouts were created to extract accurate area estimations, while simulation tools measured power and delay metrics during operations.

Quantitative evaluation of SRAM- and memristor-based CIM designs
The core contribution of this thesis lies in the quantitative comparison of SRAM-based and memristor-based CIM designs. This evaluation identifies trade-offs and application-specific advantages of each technology. The results from the individual analyses were combined into a detailed comparison, focusing on area, energy consumption, and delay. An application-specific use case (neural networks) was considered in the analysis. Insights from this work can inform design choices in emerging applications, such as AI accelerators and IoT devices, where energy efficiency and area constraints are critical.

1.4. Thesis organization

chapter 2 presents the state-of-the-art analysis. The first part of this chapter discusses various memristor technologies, their working principles, and their respective advantages and disadvantages. The second part explores different types of memristor-based CIM macros. Finally, the third part examines the various SRAM-based CIM macros. chapter 3 details the designs selected for recreation and comparison, their working principles, and the recreation process. The chapter begins by presenting the design requirements used for selecting the designs. Next, the chosen designs are introduced, followed by an explanation of how each design functions and the methods used for their recreation. The chapter concludes with an overview of additional components used in the designs, explaining their functionality and the process of creating them. In chapter 4, the results are presented. The chapter begins with an overview of the simulation setup, followed by the presentation of the layouts. Finally, the performance and energy consumption results are discussed. Following the results, the discussion and conclusion are presented in chapter 5 and chapter 6 respectively. The thesis concludes with the appendix and bibliography. The approach of answering the question 'Which technology (memristor or SRAM) is better suited for computation-in-memory?', is illustrated in Figure 1.5. Each color represents one of the chapters in this thesis.

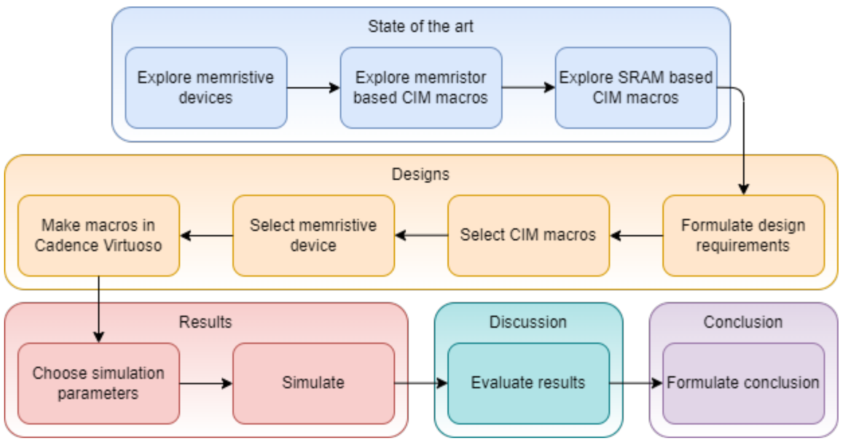

State of the art
Explore memristive devices
Explore memristor based CIM macros
Explore SRAM based CIM macros
Designs
Make macros in Cadence Virtuoso
Select memristive device
Select CIM macros
Formulate design requirements
Results
Choose simulation parameters
Simulate
Discussion
Evaluate results
Conclusion
Formulate conclusion

Figure 1.5: Approach to answer the question: 'Which technology (memristor or SRAM) is better suited for computation-in-memory?'

2
State of the art

The first part of this chapter discusses various memristor technologies, their working principles, and their respective advantages and disadvantages. The second part explores different types of memristor-based CIM macros. Finally, the third part examines the various SRAM-based CIM macros.

2.1. Non-volatile devices (memristors)

Figure 2.1 shows the memory taxonomy from the 2015 International Technology Roadmap for Semiconductors (ITRS) Emerging Research Devices (ERD) chapter [20]. Some of the ”prototypical” NVM’s can also be considered ”emerging” [21] so they will be considered as potential candidates for the CIM design as well. All of these devices are based on different technologies, which results in each of them having different benefits and drawbacks.

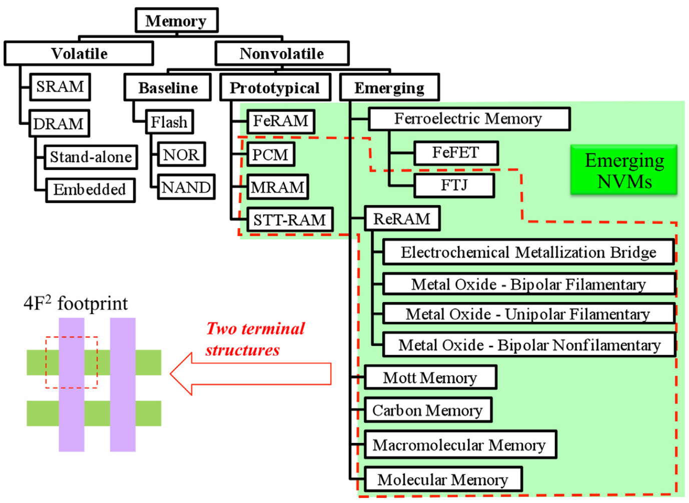


Figure 2.1: Memory taxonomy from the 2015 ITRS Emerging Research Devices (ERD) chapter [20].

Nine of these emerging NVM technologies were evaluated in a workshop organized by ERD in 2014. PCM, STTRAM and RRAM (also referred to as ReRAM) came out to have the most promising performance. FeFET and FTJ was considered the top choice that needs more resources to explore its potential [21].

Phase Change Memory (PCM)
Figure 2.2 shows the cross section of a PCM bitcell. The GST (Ge-Sb-Te) material is a chalcogenide alloy. Crystallisation occurs in this material when a voltage pulse with low amplitude is applied [22]. This is induced because of joule heating. The crystal structure is broken down (amorphization) when a voltage pulse with high amplitude is applied. In the crystalline phase, the device has a low resistance because of a large concentration of carriers. In the amorphous phase, there is a high resistance because of Fermi level pinning at the mid gap. This process can be seen in Figure 2.3. The time it takes to transition from the amorphous phase to the crystalline phase, called the set process, determines the speed of the PCM [23]. The temperature required to to do the reverse, called the reset process, is the power-limiting step.

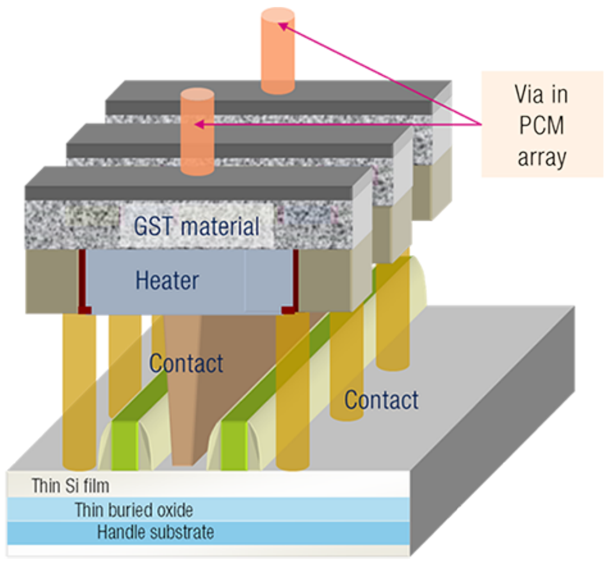


Figure 2.2: Cross section of a PCM bitcell [24].

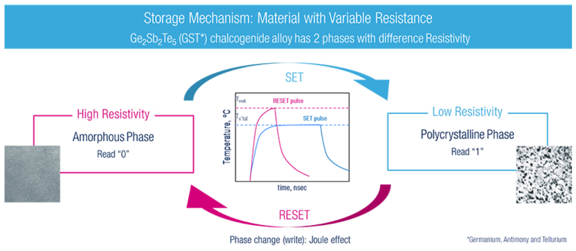

Storage Mechanism: Material with Variable Resistance
$Ge_2Sb_2Te_5$ (GST*) chalcogenide alloy has 2 phases with difference Resistivity
SET
High Resistivity
Amorphous Phase
Read "0"
T_{melt}
RESET pulse
$T_{x'tal}$
SET pulse
Temperature, °C
time, nsec
Low Resistivity
Polycrystalline Phase
Read "1"
RESET
Phase change (write): Joule effect
*Germanium, Antimony and Tellurium

Figure 2.3: Phase change process of PCM [24].

Besides low and high resistance states, it is also possible to to get into intermediate states [11]. This would allow multi-level cell operation. Another advantage of PCMs is that the manufacturing technology is quite mature and is compatible with CMOS. The endurance of PCMs is more than 10^9 [25] [26]. This represents the number of switching cycles before the device becomes unreliable. PCMs show desirable scaling behaviors [21]. Citing from [21], the scaling behaviors are: *e.g., phase transition at highly scaled dimension (several nm in thin-film thickness or nano-particle diameter), higher crystallization temperature (i.e., longer retention) at smaller dimension, decreasing thermal conductivity (i.e., higher power efficiency) with thinner films, linear dependence of threshold voltage on device size, improved endurance at smaller dimension, etc.*

A downside of PCM is the slow programming/switching speed [11][21]. This is due to the slow crystalline process. There is also a resistance drift in the amorphous state that has to be compensated for at circuit level, which further decreases the switching speed.

The performance of PCM devices depends on the phase change materials that are used. They affect the speed and power usage of the device. Doping the GST material might also improve the speed, endurance and data retention at higher temperatures of the device [27]. Scaling also has a (positive) impact on performance as mentioned before.

Spin-Transfer Torque RAM (STTRAM)

The structure of a STTRAM can be seen in Figure 2.4. The STTRAM consists of of two ferromagnetic layers and a tunnel layer, that together forms a a magnetic tunnel junction (MTJ) [11]. The *pinned layer* has a fixed magnetization and the magnetization of the free layer can change directions. The two ferromagnetic layers have either magnetization in the same direction (parallel) or in or in opposite directions. Electrons can easily pass through the stack when both layers have parallel magnetization. Therefore the device is in a low resistance state. There is a low probability that electrons pass through the stack when the layers have an anti-parallel magnetization. This is the high resistance state. The magnetization can be controlled by letting a spin-polarized current flow through the magnetic multilayer [28]. This is called the spin-transfer torque (STT) effect.

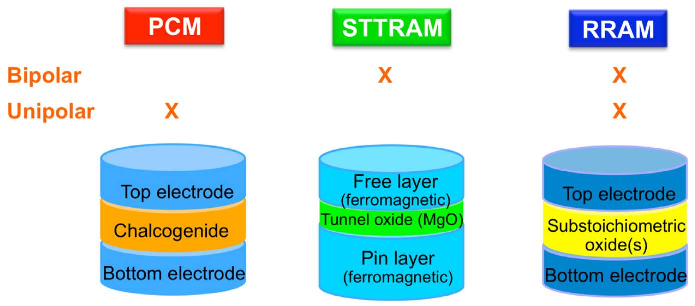


Figure 2.4: Structure and operation polarity of PCM, STTRAM, and RRAM [21].

STTRAM has the fastest switching speed and best reliability (cycle endurance) among all NVMs [21]. Unfortunately it also has a lot of drawbacks in its current state. It has low energy efficiency, the ferromagnetic materials are not easy to handle and it is challenging to make it compatible with CMOS [11]. It has a small on/off ratio, needs well designed reading schemes and has an increased MTJ variability when scaled [21].

The performance of STTRAM devices is strongly influenced by the manufacturing process. Substrate smoothness, etching damage, encapsulation and materials used for the layers all have a big impact on the performance of the device [21].

Resistive RAM (RRAM)

There are three categories of RRAM [29]. You have electrochemical memory (ECM), valence change memory (VCM) and thermochemical memory (TCM). All of them have different mechanisms to generate their resistances. TCMs and ECMs use redoxation and oxidation to build up and break down a structure called filamentary. There are less free electrons when the filamentary is build, so the resistance of the device rises. TCMs are unipolar, which means that polarity doesn't matter when switching states. ECMs are bipolar. Voltage and reversed voltage signals are used to build up and dissolve the metallic filament using redox reactions. VCMs are also bipolar. With them, not only a filament, but also a complete metallic layer is built up and dissolved by the exchange of ions.

Advantages of RRAM are that they offer good scalability [11]. VCM cell sizes can be made in the nanometer range. There is a large difference between the high and low resistance states. This means that simpler CMOS circuits are required to evaluate the resistance. RRAM has fast switching, which is in the nanoseconds range. It also has good compatibility with CMOS manufacturing processes.

RRAM usually has trade-offs between speed-retention, power-speed, endurance-retention, etc [21]. The biggest challenges RRAM has are reliability, variability and failure mechanisms. The resistance and switching voltage varies in each cell and even in each cycle. This is because the location, dimension and composition of the filaments vary in a random manner[30].

RRAM can be made using using different materials for the electrodes and for the resistance changing

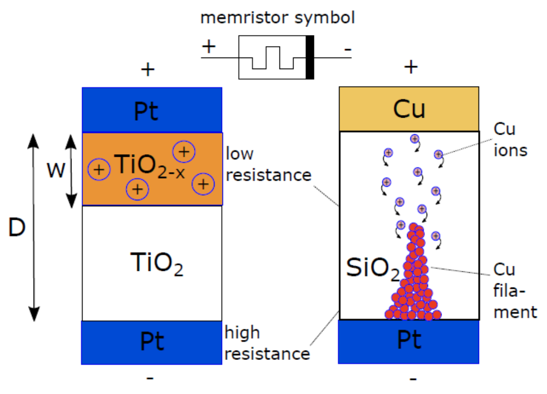
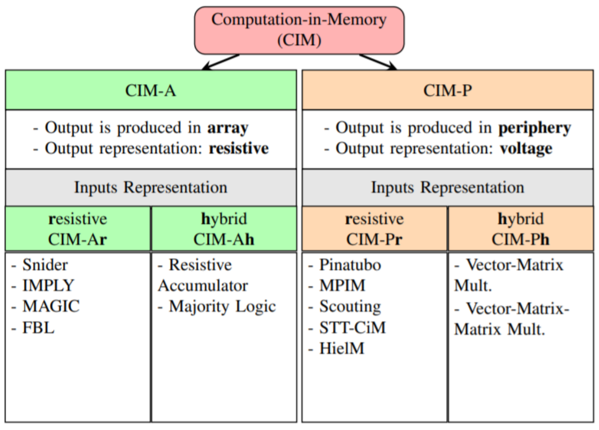


Figure 2.5: ECM (left) and VCM (right) RRAMs [29].

layer. Numerous metal oxides and non-oxide have been used to make RRAM (like NiO_x, HfO_x, TiO_x, silicon, sulfides, chalcogenides). More recent devices have been made using multi oxide stacks or oxide compositions.

2.2. Memristor-based CIM macros

Memristor-based CIM macros can be classified based on input and output data representation [31]. The output of the CIM kernel can be produced in the peripheral circuit in the form of a voltage. Alternatively, the output can be produced in the memory array. When the output is produced in the array, it will be in the form of a resistance in a memristor. The architectures are classified as CIM-P (CIM-Peripheral) and CIM-A (CIM-Array) designs respectively. The location of the input data divides these classes into two sub-classes. CIM designs that require all operands to be in the array, are classified as CIM-Ar and CIM-Pr ('r' for resistive). CIM architectures that only require part of the operands to be in the array are called CIM-Ah and CIM-Ph ('h' for hybrid). The other part of the operands are received via the memory ports as a voltage. An overview of this classification and some popular designs in each class can be seen in Figure 2.6.



Computation-in-Memory (CIM)

CIM-A		CIM-P	
- Output is produced in **array** - Output representation: **resistive**		- Output is produced in **periphery** - Output representation: **voltage**	
Inputs Representation		Inputs Representation	
resistive **CIM-Ar**	**hybrid** **CIM-Ah**	**resistive** **CIM-Pr**	**hybrid** **CIM-Ph**
- Snider - IMPLY - MAGIC - FBL	- Resistive Accumulator - Majority Logic	- Pinatubo - MPIM - Scouting - STT-CiM - HielM	- Vector-Matrix Mult. - Vector-Matrix-Matrix Mult.

Figure 2.6: Classification of memristor-based CIM design styles [31].

Common advantages of CIM-A architectures are that you have maximum parallelism, since the execution in independent of sense amplifiers. You can also have cascading operations without having to feed intermediate results back to the array. The disadvantages are that execution requires the memristive devices to change states. This is a big downside since the endurance of memristive devices is not many cycles. High voltages need to be applied to the memory array during execution, which results in high power consumption and the need of large drivers.

Advantages of CIM-P designs are that they do not change the states of the memristive devices during execution, so computation does not affect the endurance of the device. Low voltages are required to be applied to the array, since the states don't have to change during execution. Because of this, it will have low power consumption and doesn't need large drives. CIP-P designs don't modify the conventional structure of a memory array as much as CIM-A designs, because they focus on modifying the peripheral circuit to realize the operations. CIM-P designs depend on sense amplifiers. A disadvantage because of this is that the level of parallelism depend on the amount of sense amplifiers that is used. A sense amplifier in every column will result in maximum parallelism, but this will also increase the area. Cascading operations is not possible in CIM-Pr architectures since the output is produced as a voltage.

CIM-A architectures change the state of the memristive devices during each computation and memristive devices don't have a high endurance. This makes CIM-A architectures not suitable for any applications using the current state of memristive devices. An example of a CIM-Pr and CIM-Ph architecture is given in the following two paragraphs.

Scouting Logic (CIM-Pr design example)
Scouting logic [32] will be used as an example of an CIM-Pr design. Scouting logic is similar to Pinatubo [33], MPIM [34], STT-CiM [35] and Hielm [36] [31]. With scouting logic, you are able to perform the logic functions OR, AND and XOR. The concept can be seen in Figure 2.7. M_1 and M_2 are the two memristors that have the operands stored in their resistive state. The reference current I_{ref} determines which logical operation is performed. To perform the OR operation, you need to set I_{ref} between $2V_r/R_H$ and V_r/R_L. This works because I_{in} will be equal or larger than V_r/R_L if either M_1 or M_2 are in a low resistance state (logical '1'). If one of the memristors is in a high resistance state, then $I_{in} = V_r/R_L + V_r/R_H$, which is still equal to approximately V_r/R_L since $R_L << R_H$. For the AND operation you need to set I_{ref} between V_r/R_L and $2V_r/R_L$. You need both reference currents for the XOR operation. To read the value of a memristor, you need to only enable V_r on that line.

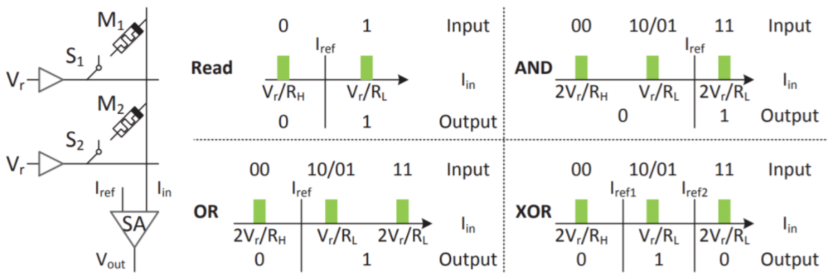


Figure 2.7: Scouting logic [31].

Vector-Matrix Multiplication (CIM-Ph design)
For the CIM-Ph design example we use the design described in [37]. This design is similar to others like [38], [39] and [40]. In Figure 2.8 can be seen how vector V=[$v_1^I, v_2^I, ..., v_N^I$] is multiplied with matrix G=[g_{ij}]. The data of matrix G is stored in the resistive value of the memrstors. Vector V is provided via the wordlines. The input voltages produce currents with the memristors. These currents represent the multiplication. The input v_1^I multiplied with the matrix element $G_{1,1}$ generates the current $I = \frac{v_1^I}{G_{1,1}}$. If $G_{1,1}$ is in the low resistance mode, which represents a '1', then the current is high (if the input was also a '1'). The currents resulting from all the multiplications on the same bitline are added together. Using the reference resistor R_S, this current produces the output v_1^O. The output then needs to be converted back to a digital value using an ADC. The input voltages and matrix resistances also don't necessarily have to be binary values.

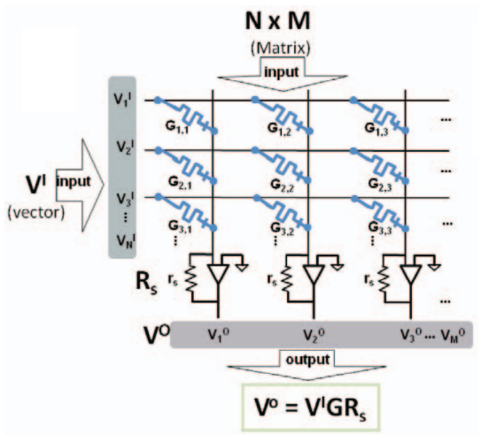


Figure 2.8: Vector-Matrix multiplication [37].

2.3. SRAM-based CIM macros

CIM can also be implemented using traditional SRAM. The main advantages of SRAM-CIM are that they have faster switching times and much larger endurance. Just like with memristor-based CIM designs, most recent papers on SRAM-CIM focus on designs which can be used to perform logic or multiplication-accumulation (MAC) operations [14]. SRAM-CIM designs are made by modifying the standard 6T SRAM cells or by adding transistors.

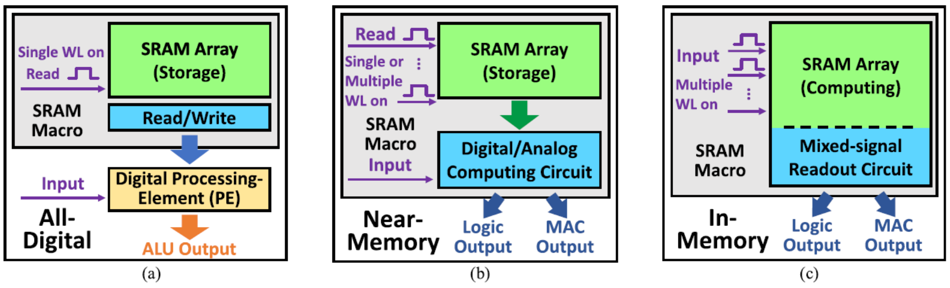
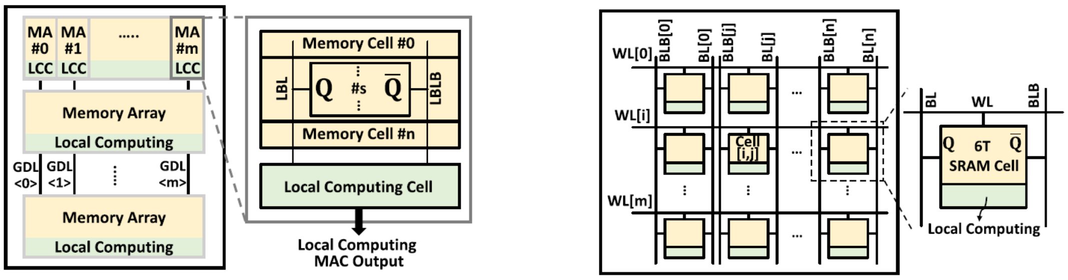


Figure 2.9: Three conceptual approaches to computing: (a) conventional digital computing, (b) near-memory-array computing (NMAC), and (c) in-memory-array computing (IMAC) [14].

Many schemes for SRAM-based CIM have been proposed in literature. They could be categorized based on where in the memory the computation happens. They could be categorized as as near-memory-array computing (NMAC) or in-memory-array computing (IMAC), see Figure 2.9. In an NMAC scheme, the memory cells and array have the same structure as traditional memory. The computation happens in a circuit block close to the memory-array. With IMAC computation happens in the array. IMAC schemes can be further categorized as in-array local computing (IA-LC), in-cell local computing (IC-LC) and non-local computing, see Figure 2.10. With IA-LC multiple SRAM cells share a set of local computing cells. With IC-LC, each cell has computing circuits embedded in them. With non-local computing, traditional SRAM cells and/or the periphery is modified and the computaion is made in the array [14].

A selection of different SRAM-based CIM macros can be seen in Figure 2.11. Each design has their advantages and disadvantages. A design can focus on different features. They include: high cell density, low read disturb, high precision, low power, short latency, fast computing, large signal margin, high throughput, etc. More examples of SRAM-based CIM macros can be seen in Appendix B. A brief explanation on how each of the designs from these figures work and what their advantages and disadvantages are, can be read in [14].



(a) (b)

Figure 2.10: a) In-array local-computing scheme (IA-LC): One local computing cell (LCC) is shared by multiple (n) SRAM cells. b) In-cell local-computing (IC-LC) scheme: Local computing circuits are embedded in each SRAM cell.[14]

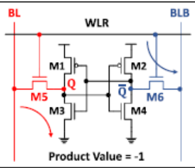
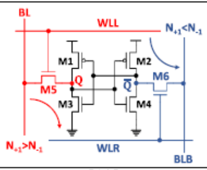
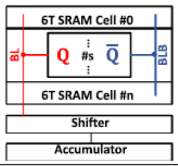
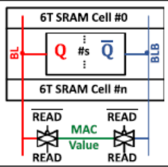
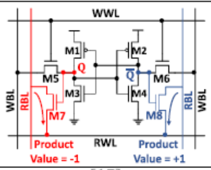
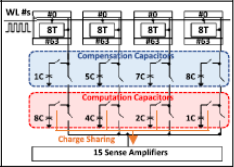
			
Reference	[12]	[13]	[15]
Process	130nm	65nm	55nm
Computing Type	IMAC	IMAC	IMAC
Architecture Type	Analog	Analog	Analog + Digital
Cell Structure	6T	DSC6T	6T
Cell Function	MAC	MAC	MAC
IN/W/OUT (# bits)	5/1/1	1/1/1	8/8/19
Energy Efficiency (TOPS/W)	-	111.6	0.6
Features	Compact Cell Area	Fast Computing	Large Signal Margin
			
Reference	[16]	[17]	[18]
Process	28nm	65nm	7nm
Computing Type	IMAC	IMAC	IMAC
Architecture Type	Analog	Analog	Analog
Cell Structure	Split-WL 6T	8T	8T
Cell Function	XAC	MAC	MAC
IN/W/OUT (# bits)	1/1/1	1/1/5	4/4/4
Energy Efficiency (TOPS/W)	299.71	15.8	351
Features	Compact Cell Area	w/o Read Disturb	Low Power

Figure 2.11: Comparison table of non local computing (IMAC) CIM schemes [14].

3 Designs

This chapter outlines the selected designs for recreation and comparison, detailing their working principles and the recreation process. It begins by presenting the design requirements that guided the selection process. The chosen designs are then introduced, followed by an explanation of their functionality. Additional components used in the designs are described next, with an emphasis on their roles and the methods employed in creating them. The chapter concludes with a discussion of the layout design process.

3.1. Design requirements

In order to compare the CIM macros, they should be able to perform the same operations. Since most CIM macros are made to perform either logic or MAC operations, it should be a requirement that the chosen designs support either logic or MAC operations. For both technologies (SRAM and memristor), there should be at least one macro for logic and one macro for CIM. Additionally, the designs for logic should have at least one logic operation in common.

In order to get a fair comparison, silicon-verified designs should be used.

To be able to accurately recreate the macros, there should be circuit level info available.

Memristors are being used for CIM with the purpose of giving us higher energy efficiency and smaller area compared to SRAM-based CIM. Most SRAM-based CIM designs already aim to have high energy efficiency and small area, but there are still some that focus more on things like fast computing or low read disturb [14]. In order to see if SRAM-based CIM can outperform memristor-based CIM based on energy consumption and area, it should be a requirement to choose SRAM-based CIM designs that focus on having low energy consumption and small area.

In summary, the designs requirements are:

- Must support either logic or MAC operations.
- Logic designs must have at least one logic operation in common.
- Should be silicon-verified.
- The design should have circuit level info for verification.
- Should be made for low energy consumption and small area.

CIM designs can operate in either the digital or analog domain. However, the majority of memristor-based CIM designs found in literature function in the analog domain. Therefore, this thesis focuses exclusively on analog-based designs.

3.2. Chosen designs

Based on the design requirements, the following designs were chosen: Push-Rule WL-Decoupled 6T SRAM by Jeloka et al. [16], WL-Decoupled 4+2T SRAM by Dong et al.[17], 8T SRAM by A. K. Rajput and M. Pattanaik [15], RW-Decoupled 8T SRAM by Yu et al.[18], and 1T1R by Yu et al. [41]. An overview of the chosen designs can be seen in Figure 3.1. A larger version of the designs schematics is shown further down in this chapter.

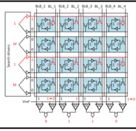
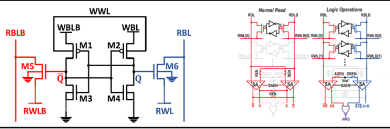
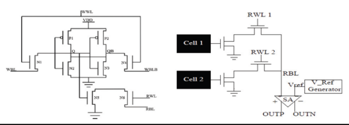
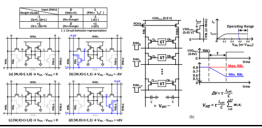
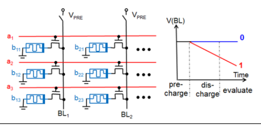
Name	Design schematic	Paper	Operations
Push-Rule WL-Decoupled 6T SRAM		S. Jeloka, N. B. Akesh, D. Sylvester and D. Blaauw, "A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory," in IEEE Journal of Solid-State Circuits, vol. 51, no. 4, pp. 1009-1021, April 2016	AND, NOR
WL-Decoupled 4+2T SRAM		Q. Dong *et al.*, "A 4 + 2T SRAM for Searching and In-Memory Computing With 0.3-V VDDmin," in *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 1006-1015, April 2018	AND, OR, XOR
8T SRAM		A. K. Rajput and M. Pattanaik, "Implementation of Boolean and Arithmetic Functions with 8T SRAM Cell for In-Memory Computation," *2020 International Conference for Emerging Technology (INCET)*, Belgaum, India, 2020, pp. 1-5	NAND, NOR, MAC
RW-Decoupled 8T SRAM		C. Yu, T. Yoo, T. T. -H. Kim, K. C. Tshun Chuan and B. Kim, "A 16K Current-Based 8T SRAM Compute-In-Memory Macro with Decoupled Read/Write and 1-5bit Column ADC," *2020 IEEE Custom Integrated Circuits Conference (CICC)*, Boston, MA, USA, 2020, pp. 1-4	MAC
1T1R		J. Yu, H. A. D. Nguyen, L. Xie, M. Taouil and S. Hamdioui, "Memristive devices for computation-in-memory," *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, Germany, 2018, pp. 1646-1651	AND, OR, MAC

Figure 3.1: Overview of the chosen designs.

Three SRAM designs for logic were chosen. All three are able to perform the AND operation. The 8T SRAM design is made to perform the NAND operation, but you can take the inverted output of the sense amplifier to get the AND operation. Two designs for MAC have been chosen. One of them is also the 8T SRAM, since it can do both logic and MAC operations. The only difference between the 8T design for logic and the 8T design for MAC is that the logic design has a sense amplifier (SA) in the periphery and the MAC design has an analog to digital converter (ADC). One design has been chosen for the memristor-based CIM. The 1T1R macro is able to perform both logic and MAC operations, just like the 8T SRAM design.

3.3. Chosen memristor technology

In chapter 2 was introduced which memristor technologies there are. PCM, STTRAM and RRAM turned out to be the most popular of them. In order to find out which of them would be best suited for CIM, a comparison was made using data from different papers. In Appendix A you will find figures and tables from different papers that were used to make this comparison. In Table 3.1 you will find a summary of the findings. SRAM is also included in this table to show

Each of these devices have their advantages and disadvantages and there is not one that is best at

	SRAM	Resistive switching	Phase change	Spintronics
MLC	No	Yes	Yes	No
3D integration	Good	Good	Good	Limited
Read time (ns)	Fast	Fast	Fast	Fast
Write time (ns)	Fastest	Medium	Slow	Fast
Write power	Low	Low	High	Medium
Write energy	Lowest	Low	High	Low
Variability	Low	High	Medium	Low
Endurance	Best	Bad	Medium	Good

Table 3.1: Characteristics of various memristor technologies and SRAM.

everything. For CIM however, it would be good to have a device that can do MLC, which you can not do with spintronics. PCM has high write power and energy, which would be even more disadvantageous in the large crossbars used for CIM. ReRAM seems to be the best choice for CIM based on this table. Especially when you consider that you can increase the write power to decrease the write time and variability.

3.4. SRAM based designs

This chapter is split into two parts. The first part discusses the three SRAM based designs that are made to perform logic operation. The second part will be about the two designs made for MAC operations.

3.4.1. SRAM macros for logic

In this section, Push-Rule WL-Decoupled 6T SRAM by Jeloka et al. [16], WL-Decoupled 4+2T SRAM by Dong et al.[17] and 8T SRAM by A. K. Rajput and M. Pattanaik [15] are discussed. Their working principles, together with an example are discussed.

Push-Rule WL-Decoupled 6T SRAM

Working principle
In [16], Jeloka et al. used an existing split-wordline 6T SRAM cell structure [42] to implement logic operations and CAM operations. A CAM operation is where you give some data as in input and retrieve at which address it is stored. The logic AND and NOR operation can be performed using the same design. For our purpose, only the logic operations will be used.

The SRAM cell structure can be seen in Figure 3.3a. In this SRAM cell, the pass gates are decoupled and they are connected to two separate wordlines, the WL and WLB. The sense amplifier is a reconfigurable SA, see Figure 3.2. In logic mode, it will look like Figure 3.3b. The bitline (BL) and bitline bar (BLB) are connected to the positive terminal of of a sense amplifier. The negative terminal is connected to a reference voltage. The output of these sense amplifiers are the input of an AND port. The output of the AND port is the output of the operation. In normal read mode, one of the sense amplifiers has the BL at its positive terminal and the BLB at its negative terminal. The output of the SA is the output of the read operation.

Normal read and write operations are performed in the same way as with traditional 6T SRAM. The WL and WLB have the same input and together act like a single WL. When performing the logic operation, multiple WLs or WLBs are enabled depending on which logic operation is performed and which cells are part of the computation.

The advantages of this design are the small area overhead compared to traditional 6T SRAM cells and the ability to perform CAM operations in addition to logic operations. A disadvantage of this design

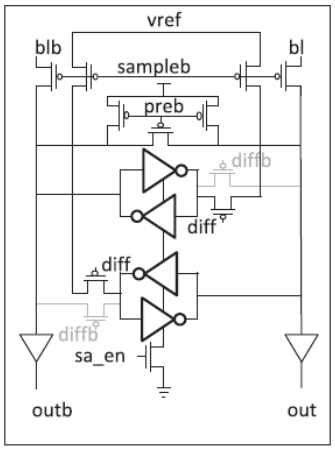
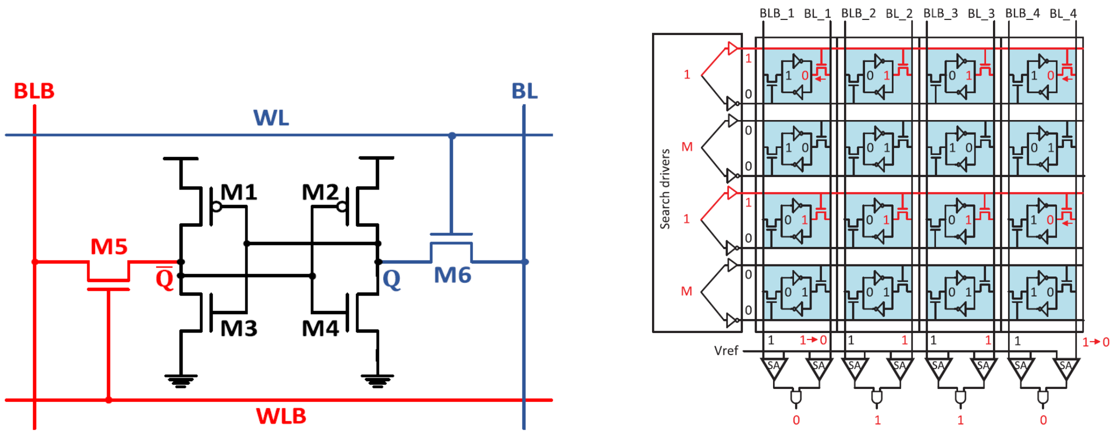


Figure 3.2: Push-Rule WL-Decoupled 6T SRAM reconfigurable sense amplifier.



(a)

(b)

Figure 3.3: (a) Cell structure of Push-Rule WL-Decoupled 6T SRAM (b) AND operation between rows 1 and 3. Rows 2 and 4 are disabled for this operation.[16]

however is that in implementing logic operations, the issue of read disturbance must be considered when computing with many cells on the same bitline.

Example

Reading and writing is done in the same way as with traditional SRAM. WL and WLB have the same input and together act as a single wordline. An example of performing the AND operation will be given in this section.

On the left in Figure 3.4 you can see the initial state of the crossbar. Wordlines 0 until 3 store the values ”0110”, ”0101”, ”1110” and ”1110” respectively. On the right side of this image you can see the cells connected to bitline 1 and to wordlines 0 and 2.

The first step of performing the AND operation is charging the bitlines.

Then you enable the wordlines on which you want to perform the operation. In this example wordlines 0 and 2 are enabled. The wordlines that are not used, are masked. Masking a wordline means that both the WL and WLB are not enabled.

What will happen now is that there is a path from bitline 1 to the ground of the cell connected to wordline 0. There is also a path from bitline 1 to the Vdd of the cell connected to wordline 2. The pull down transistors are stronger than the pull up transistors. This causes the pre charged bitline 1 to discharge.

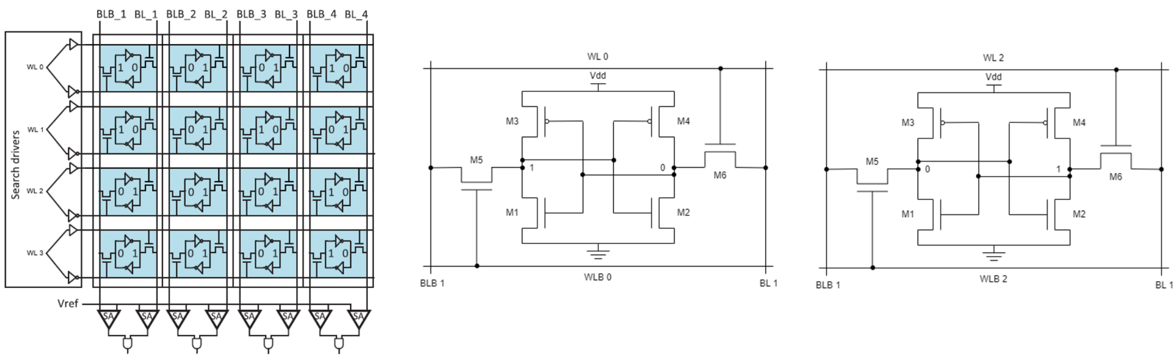


Figure 3.4: Push-Rule WL-Decoupled 6T SRAM logic operation example. Step 0: Initial state.

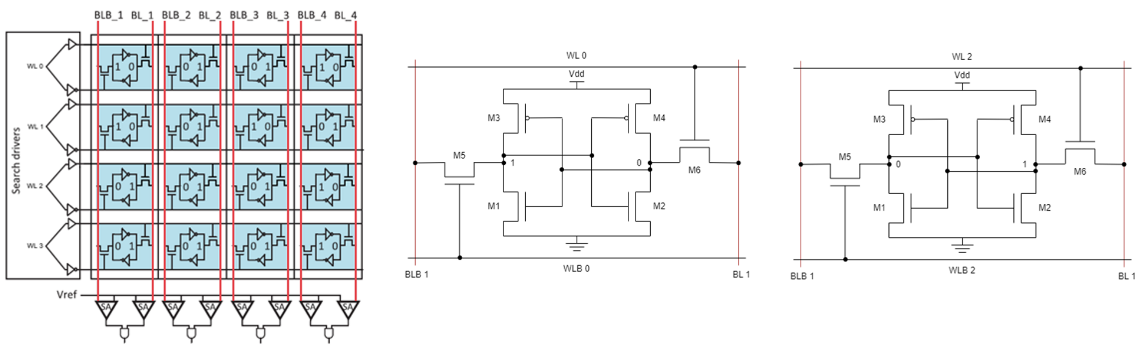


Figure 3.5: Push-Rule WL-Decoupled 6T SRAM logic operation example. Step 1: Pre-charge bitlines.

The voltage on the bitline falls below the reference voltage of the sense amplifier. The output of the SA connected to the BL becomes 0, so the output of the AND operation becomes 0. If the cell connected to WL 2 also stored a 0, then there would be two paths to ground and the output would be the same. If WL 0 stored a 1 instead of a 0, then there would not be a path to ground and the BL would stay charged. The BLB would also stay charged, because WLB was not enabled. This will cause the output of the AND gate to be a 1.

WL-Decoupled 4+2T SRAM

Working principle

Dong et al.[17] proposed a modified SRAM cell that can perform write operations using only four transistors, see Figure 3.7. This is done by using deeply depleted channel technology (DDC). Transistors with a deeply depleted channel have strong body effect. The body effect in transistors is refers to the change in the threshold voltage when there is a voltage difference between the source and the body of the transistor. The strong body effect of DDC allows this scheme to use the N-well of PMOS M1 and M2 as write wordlines (WWL). Two more transistors (M5 and M6) make it possible to perform the read operation and to perform CAM and logic operations. Figure 3.8 shows the voltages that need to be applied at the different terminals to perform each operation.

The periphery of the WL-Decoupled 4+2T SRAM macro can be seen in Figure 3.7b. During a read/logic operation RBL gets discharged if the cell stores a ’1’ and RBLB gets discharged if a cell stored ’0’. Just like with the Push-Rule WL-Decoupled 6T SRAM design, a reconfigurable SA is used. The design is a little bit different though. In read mode, the two sense amplifiers have the same inputs and consequently the same output. The inverted output of the sense amplifier will be the result of the read operation. In logic mode, one of the sense amplifiers has RBL at its positive input and a reference voltage at its negative input. The other SA has the same reference voltage at the positive input and has RBLB at the negative input. The outputs of the periphery in logic mode can be the results of either the AND, or

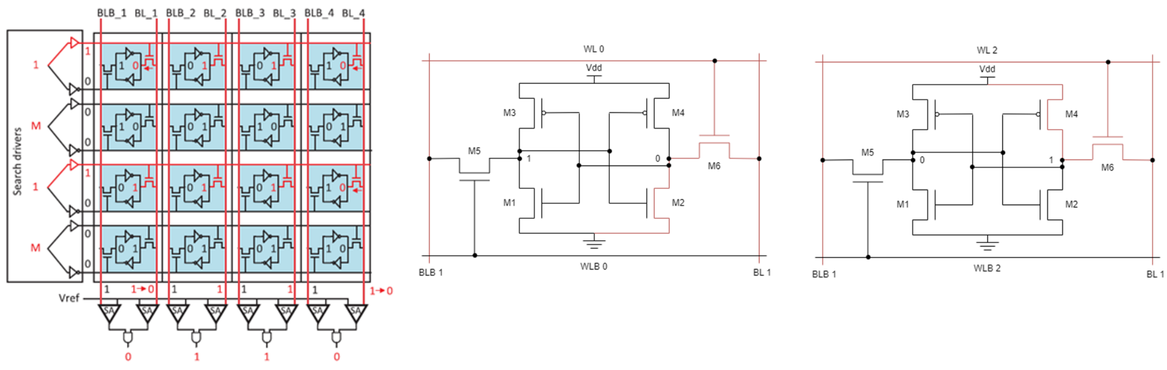


Figure 3.6: Push-Rule WL-Decoupled 6T SRAM logic operation example. Step 2: Enable wordlines.

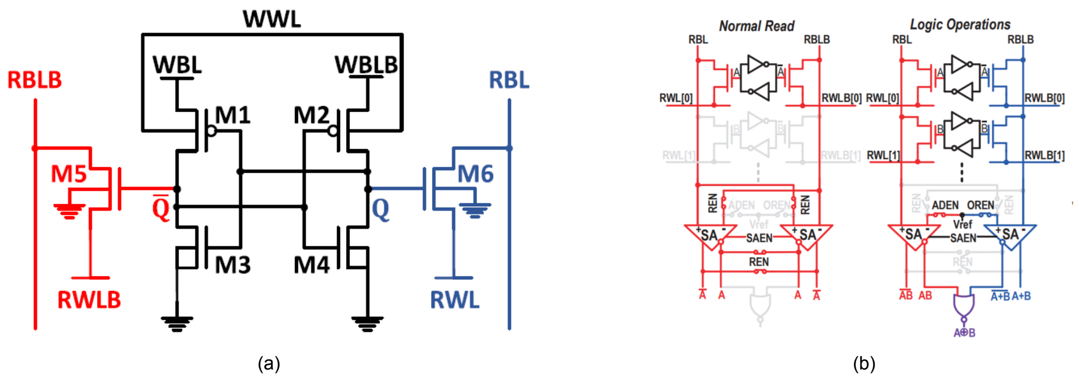
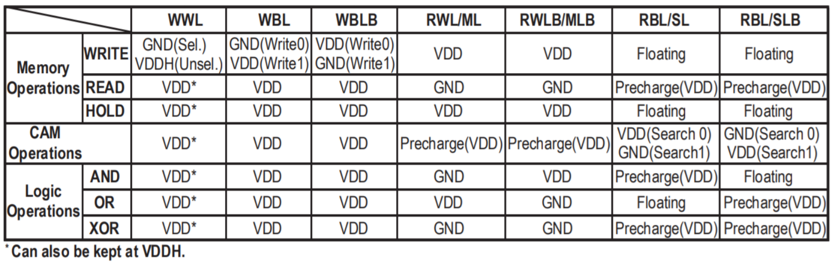


Figure 3.7: (a) WL-Decoupled 4+2T SRAM cell (b) WL-Decoupled 4+2T SRAM read and logic operations[17].

OR operation. The outputs can also be the results of the NAND, AND, XOR, NOR and OR operation, see Figure 3.7b. Which of these two will be the output, depends if you pre-charge RBL, RBLB or both, see Figure 3.8. An example of the write and read/logic operation is given in the next section. This will make the working principle clearer.



		WWL	WBL	WBLB	RWL/ML	RWLB/MLB	RBL/SL	RBL/SLB
Memory Operations	WRITE	GND(Sel.) VDDH(Unsel.)	GND(Write0) VDD(Write1)	VDD(Write0) GND(Write1)	VDD	VDD	Floating	Floating
	READ	VDD*	VDD	VDD	GND	GND	Precharge(VDD)	Precharge(VDD)
	HOLD	VDD*	VDD	VDD	VDD	VDD	Floating	Floating
CAM Operations		VDD*	VDD	VDD	Precharge(VDD)	Precharge(VDD)	VDD(Search 0) GND(Search1)	GND(Search 0) VDD(Search1)
Logic Operations	AND	VDD*	VDD	VDD	GND	VDD	Precharge(VDD)	Floating
	OR	VDD*	VDD	VDD	VDD	GND	Floating	Precharge(VDD)
	XOR	VDD*	VDD	VDD	GND	GND	Precharge(VDD)	Precharge(VDD)

*Can also be kept at VDDH.

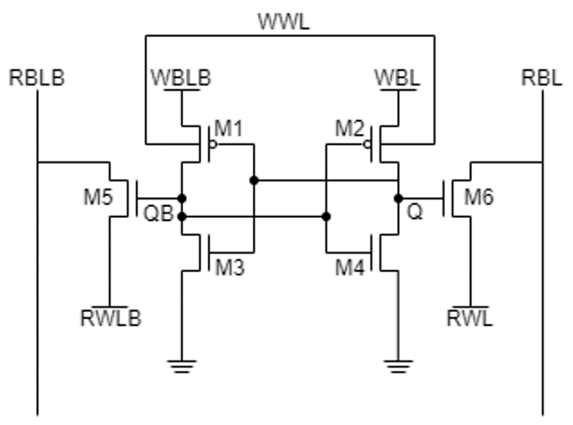
Figure 3.8: 4+2T SRAM operation table [17].

An advantage of this design is the small area overhead compared to a lot of CIM designs like the 8T SRAM. According to [17], it reduces area overhead by 15% compared to conventional 8T SRAM cells. A disadvantage is that since the WWL is connected to the N-well of transistors M1 and M2, the cells are susceptible to half-select disturbance and data flipping. Applying VDDH to unselect the cells during writing is supposed to alleviate the half-select disturbances.

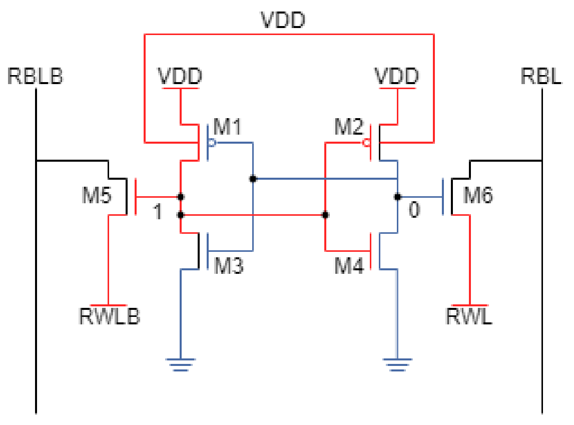
Example

This example consists of two parts. In the first part, an example is given of writing to the cell. In the second part, an example is given of performing the read and the logic operation.

Write example

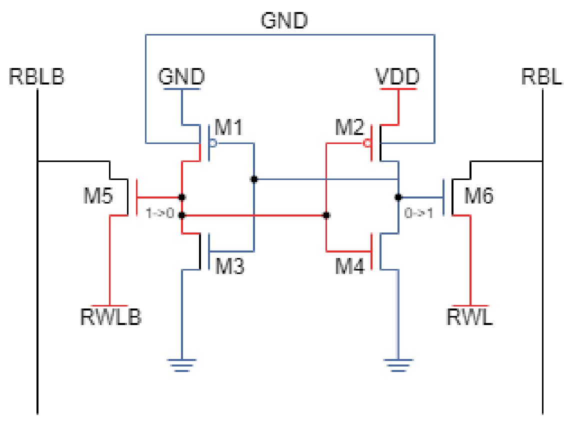


(a) WL-Decoupled 4+2T SRAM cell

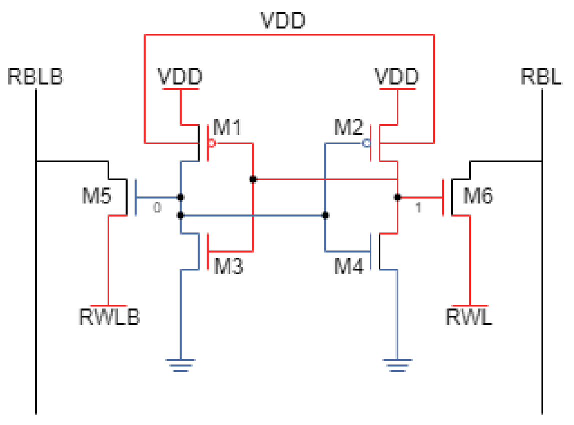


(b) Initial state: ’0’ stored

Figure 3.9: WL-Decoupled 4+2T SRAM write operation example.



(a) Transitioning from ’0’ stored to ’1’ stored.



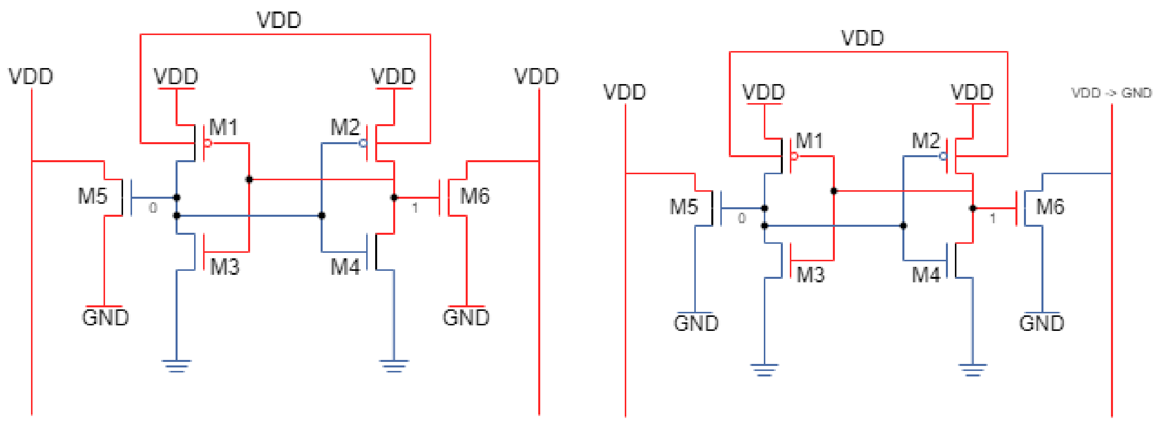
(b) Final write example state / initial read example state: ’1’ stored.

Figure 3.10: WL-Decoupled 4+2T SRAM write operation example.

In this example will be shown how you can write a ’1’ to the cell when a ’0’ is stored initially. In Figure 3.9a, the WL-Decoupled 4+2T SRAM cell can be seen. When writing to the cell, you only use the transistors M1 to M4. Transistors M5, M6, RBL and RBLB are not used for writing. The initial state of the cell can be seen in Figure 3.9b. This corresponds to the HOLD operation shown in Figure 3.8, which simply means that the cell holds it’s stored data and nothing happens. In the figures of this example, red wires mean that there is a high voltage on the wire and blue wires mean that there is a low voltage on there. Black wires mean that it is floating. The cell stores a ’0’ because the ’Q’ wire is blue.

The first step in writing a ’1’ to the cell is connecting the WWL and the WBLB to ground and connecting the WBL to VDD. The WWL of other rows in the crossbar is set to VHHD, which is a higher voltage than VDD. All other wires do not change. This can also be seen in Figure 3.8 and Figure 3.10a. When WWL is set low, the selected PMOS device becomes much stronger due to its forward body bias. This will short the WBL/WBLB with Q/QB, thus writing into the cell. The WWL in other rows is set to VDDH, which weakens their PMOS devices, preventing anything from being written in the cells. In Figure 3.10 is illustrated what happens during the transition from a ’0’ stored in the cell to a ’1’ stored in the cell. The end state when a ’1’ is stored in the cell is also shown in that figure.

Read/logic example
For the read example, we look at a cell in which a ’1’ is stored, see Figure 3.10b. The cell starts in the HOLD state. The reconfigurable SA in the periphery is set in read mode. Both sense amplifiers have RBL at their positive input and RBLB at their negative input. First the RBL and RBLB are pre-charged (Figure 3.11a). Then RWL and RWLB are set to ground (Figure 3.11b). The cell stores a ’1’, so transistor M6 is enabled. Because of this, there is a path from RBL to ground and RBL is discharged. Once the



(a) RBL is pre-charged. (b) RWL is set to ground.

Figure 3.11: WL-Decoupled 4+2T SRAM read operation example.

SA is enabled, the output of the sense amplifiers will be ’0’. The inverted output of the sense amplifiers present the result of the read operation. The next paragraph will give an example of performing the logic AND operation.

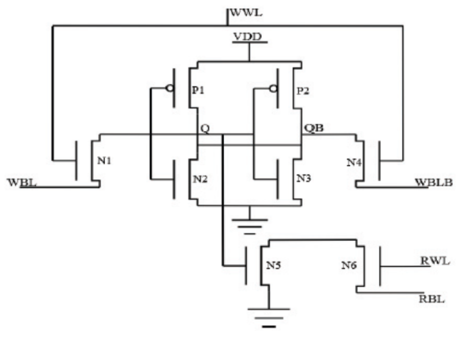
During a logic operation, the reconfigurable sense amplifier is in the logic mode, as described earlier and as shown in Figure 3.7b. Like with the read operation, the cells start in the HOLD state. For the AND operation, RBL needs to be pre-charged. Lastly RWL is connected to ground. The logic operation is performed between two cells in the same column, which means that RWL of both rows need to be connected to ground. If one of the cells stores a ’1’, then RBL is discharged. If both cells store a ’1’, then RBL is discharged, but faster. When only one of the cells contain a ’1’, RBL will be discharged, but it will not have discharged below the reference voltage before the SA is enabled. If both cells store a ’0’, then RBL will not be discharged and it will also remain above the reference voltage when the SA is enabled. In both cases the output of the SA, which represents the NAND operation, will be ’1’. The inverted output of the SA, which represents the AND operation will be ’0’. If both cells contain a ’1’, then RBL will be discharged fast enough that it is below the reference voltage by the time that the SA gets enabled. This will cause the output of the SA to be ’0’.

8T SRAM

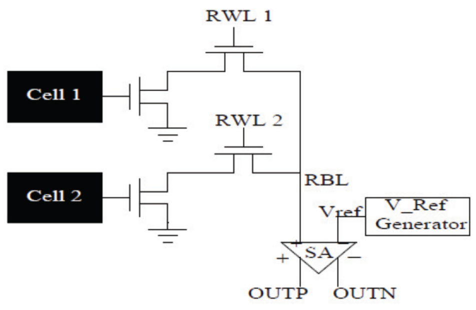
Working principle
In [15], A. K. Rajput and M. Pattanaik present an 8T SRAM cell design that can be used to perform logic operations. The design is made by adding two additional transistors to the standard 6T SRAM cell, see Figure 3.12a. Logic operations NAND and NOR can be performed in a similar way as with Scouting Logic, which was described in chapter 2. The circuitry that makes it possible for the 6T SRAM part of this design to perform logic operations, can be seen in Figure 3.12b. Cell 1 and cell 2 are normal 6T SRAM cells. RWL 1 or RWL 2 (or both) will be high if you want to use cell 1 or cell 2 for the logic operation. RBL is initially pre charged. If a RWL is high and the corresponding cell stores a ’1’, then there will be a path from the RBL to ground and the RBL will be discharged. If multiple cells store a ’1’ and their corresponding RWLs are high, then there will be multiple paths to ground. RBL will then be discharged faster. The reference voltage for the sense amplifier is chosen based on which operation you want to perform.

The advantage of this design is that is has a high read noise margin and energy efficiency according to [15]. A disadvantage is that you require a Vref generator and two sense amplifiers.

Example
Reading and writing is performed in the same way as you would read/write to traditional 6T SRAM. In this example will be shown how to perform the NAND operation. Figure 3.12b will be used to illustrate how this operation is performed. In this example, cell 1 will store a ’0’ and cell 2 will store a ’1’.

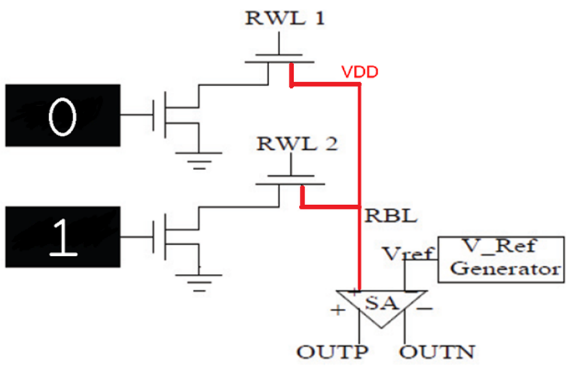


(a)

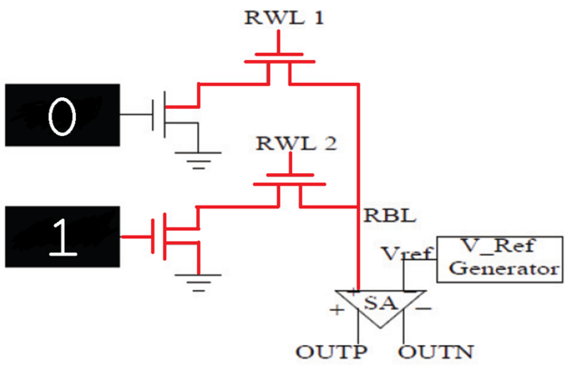


(b)

Figure 3.12: (a) The schematic of 8T SRAM Cell. (b) The circuit schematic for logic operations with 8T SRAM Cell. [15]



(a) Step 1: RBL is pre-charged.



(b) Step 2: RWL 1 and RWL 2 are made high.

Figure 3.13: 8T SRAM NAND operation example.

The first step is to pre-charge the RBL (Figure 3.13a). The second step is to make RWL 1 and RWL 2 high (Figure 3.13b). The transistors connected to RWL 2 and cell 2, are enabled since cell 2 contains a 1 and RWL is high. Because of this, there is a path from RBL to ground and RBL is discharged. The final step is to make RWL 1 and RWL 2 low and to enable the SA. In Figure 3.14 can be seen what the voltage of RBL is, and what Vref for the NAND operation is during the three steps. This is just an illustration of how the voltage of RBL would change during a logic operation. It is not the result of a simulation or a test on silicon. When RWL 1 and RWL 2 were made high, RBL started discharging. At the time of step 3, RBL has been discharged a little bit, but not enough to be below the reference voltage. The resulting output of the sense amplifier will be ’1’, which is the correct result for the NAND operation.

If both cells stored a ’0’, then RWL would not discharge and the SA output would still be ’1’. If both cells stored a ’1’, then RBL would discharge faster and it would be below the reference voltage by the time of step 3. This will result in the SA output to be ’0’. When you want to perform the OR operation, you set a higher Vref voltage. The voltage is chosen in a way that, at the time of step 3, RBL is higher than Vref if both cells contained a ’0’. If one or both cells contain a ’1’, then RBL has been discharged to below Vref at the time of step 3.

3.4.2. SRAM macros for MAC

In this section, 8T SRAM by A. K. Rajput and M. Pattanaik [15] and RW-Decoupled 8T SRAM by Yu et al.[18] are discussed.

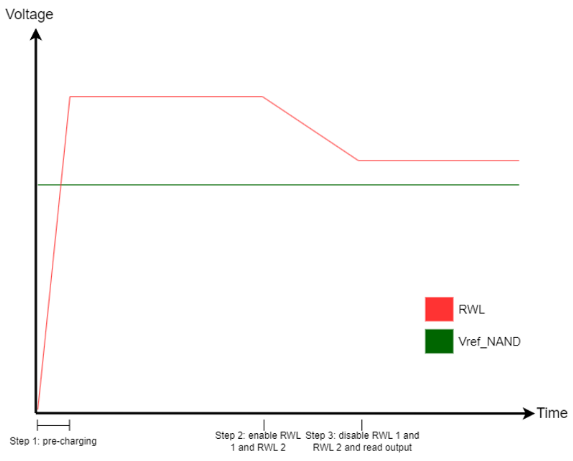


Figure 3.14: 8T SRAM RWL and Vref voltage during logic operation.

8T SRAM

Working principle
The design of the 8T SRAM macro for MAC operations is almost exactly the same as for logic operations. The circuit is the same with only one exception. The sense amplifier used for logic operations is replaced with an analog to digital converter (ADC). The MAC operation is performed in the same way as the logic operations in the 8T SRAM for logic. The difference is in the way that the RWLs are made high. With the 8T SRAM for logic, you select two rows on which you want to perform the logic operation and make their RWLs high. With the 8T SRAM for MAC, the RWLs are made high based on the input. When you want to multiply the data of a cell in a row with '1', you make the RWL high. When you want to multiply with '0', you make the RWL low. RBL gets discharged faster depending on how many cells containing a '1' are multiplied with an input of '1' in the same column. The ADC converts the amount that the RBL is discharged into a digital output. With a crossbar like Figure 3.15, you can perform the computation shown in Equation 3.1. With the 8T SRAM for logic, there were 3 different voltages that RBL could be by the time that the SA is enabled. With the 8T SRAM for MAC with n cells in a column, there are n+1 possible valtages that RBL can be at, at the time that the ADC is enabled. These voltages are ideally evenly spaced apart. A simulation of the different RBL voltages for a column of 128 cells, can be seen in Figure 3.18. Although this figure is from the paper of the RW-Decoupled 8T SRAM, you would ideally get the same output for all of the CIM MAC macros that are discussed in this thesis.

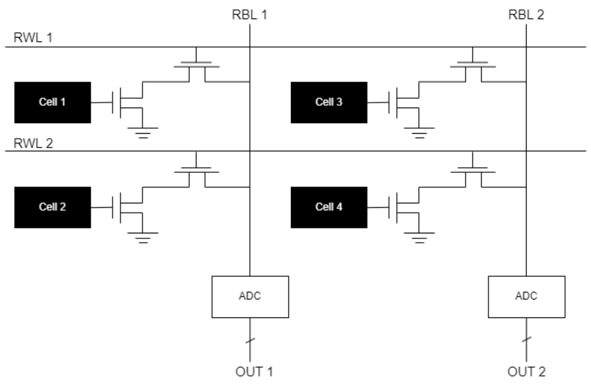
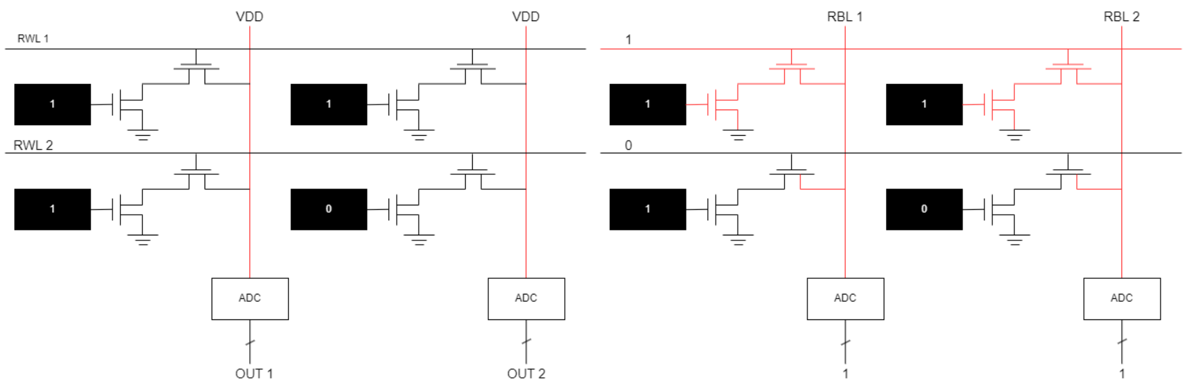


Figure 3.15: 8T SRAM for MAC 2x2 crossbar.

$$\begin{bmatrix} Cell1 & Cell2 \\ Cell3 & Cell4 \end{bmatrix} \cdot \begin{bmatrix} RWL1 \\ RWL2 \end{bmatrix} = \begin{bmatrix} RWL1 \cdot Cell1 + RWL2 \cdot Cell2 \\ RWL1 \cdot Cell3 + RWL2 \cdot Cell4 \end{bmatrix} \tag{3.1}$$



(a) Step 1: RBL is pre-charged.

(b) Step 2: Input is applied to RWL 1 and RWL 2.

Figure 3.16: 8T SRAM MAC operation example.

Example
In this example, the calculation in Equation 3.2 will be performed in the crossbar shown in Figure 3.15.

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.2}$$

The first step is to pre-charge the bitlines (Figure 3.16a). Then the input is applied to the RWLs, which is '1' for RWL 1 and '0' for RWL 2 (Figure 3.16b). The last step is to enable the ADC. The output of the ADC will be the output of the operation.

RW-Decoupled 8T SRAM

Working principle
A CIM design by Yu et al.[18] can be seen in Figure 3.17a. Each bitcell has a weight of -1 or +1 stored. A high voltage at Q in the bitcell means that it has a weight of -1 and a low voltage at Q means a weight of +1. A binary 0 can be assigned to the input (RWL) by applying a high voltage. A 1 can be assigned by applying a short negative pulse as seen in Figure 3.18. Multiple of these bitcells can be vertically stacked to make a column-based neuron, see Figure 3.17b. The RBL and RBLb are pre charged to 0.8V. RBL is discharged a little bit when a negative weight is multiplied with a 1 at the input. The same happens at RBLb when a positive weight is multiplied with a 1 at the input. How much RBL and RBLb discharge depend on how many $1 \cdot 1$ or $1 \cdot -1$ operations are performed in the same column, see Figure 3.18. The voltage difference between RBL and RBLb represent the accumulated value of the multiplications and can be converted in a digital value using a ADC.

An advantage of this design is that the read and write disturb issue that CIM designs often have, is eliminated by adding the two extra transistors to a 6T SRAM cell. A disadvantage of this design, which is shared with all the CIM designs for MAC, is that an ADCs are needed in each column. An ADC is very large compared to cells and they consume a lot of energy. An ADC could be shared with multiple columns using a MUX. Then the results of each column then need to be calculated one at the time. This is a tradeoff between area and latency that can be made. Another tradeoff that could be made is lowering the resolution of the ADC, which will result in a smaller ADC that consumes less energy, at the cost of the accuracy of the MAC result.

Example
Reading and writing to the RW-Decoupled 8T SRAM is done in the same way as with a traditional 6T SRAM cell. In this example, we will look at how a MAC operation is performed. For this example, the 2x2 crossbar shown in Figure 3.19a is used. We will focus on what happens in cell 0, which is shown on the right side in the example images. Cell 0 stores a '1', the input on RWL 0 will be '1' and the input on RWL 1 will be '0'.

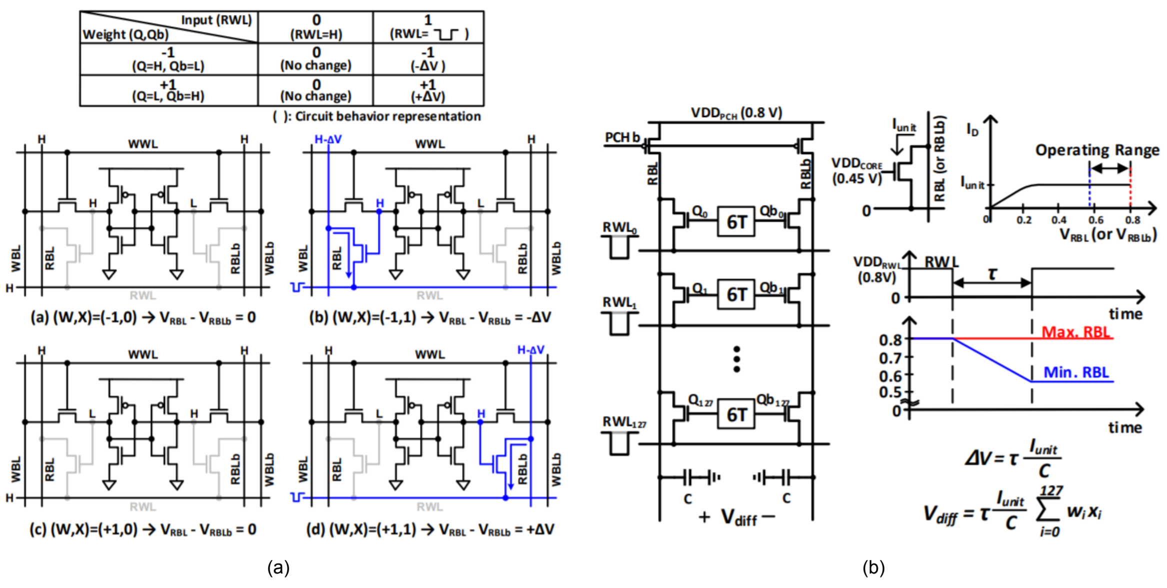


Figure 3.17: (a) RW-Decoupled 8T SRAM basic operating principle. (b) Current accumulation in a column-based neuron[18].

The first step is to apply VDD to RWL and to pre-charge RBL and RBLb (Figure 3.20). Next, a short negative voltage pulse is applied on RWL (Figure 3.21). Cell 0 contains a '1'. Because of this, transistor M7 is enabled and shorts RBL with RWL. The negative voltage pulse applied to RWL 0 causes RBL 0 to be discharged by δV. A graph of the voltages of RWL 0, RBL 0 and RBLb 0 over time, can be seen in Figure 3.22. The final step is to enable the ADCs, whose output will be the output of the computation.

3.5. Memristor based design

The most popular CIM-P RRAM designs are based on a one-transistor-one-resistor (1T1R) design. This type of design has received a lot of attention because it provides a reduction of the sneak current problem happening in array applications [43]. Novel applications, such as neuromorphic calculation and in-memory-computing (IMC) architecture, have also been implemented on the 1T1R array structure in the recent decade [44], [45], [46], [47].

Working principle
The 1T1R by Yu et al. [41] can be used to perform both logic and MAC operations in the same way that the 8T SRAM can. The cell can be seen in Figure 3.23. A schematic of a crossbar using these cells can be seen in Figure 3.24. A VCM ReRAM cell is used. The program circuit is used for writing to the cell.

The WL is made high to select the row on which you want to write. V_{RESET} is applied on the BL to write a '0' to the cell. V_{RESET} is a positive voltage that will start the 'reset' process in the memristor. The reset process will, as described in chapter 2, build up the filament. This will make the device less conductive. The device is then in its high resistance state (HRS), which is interpreted as storing a '0'. V_{SET} is a negative voltage. When applied on the bitline, it will start the 'set' process in the ReRAM. During this process, the filament gets broken down a little bit. This makes the device more conductive. The device is then in its low resistance state (LRS). This is interpreted as storing a '1'.

Logic and MAC operations are performed in the same way as with the 8T SRAM. First the bitlines are pre-charged. Then the wordlines are enabled. If a cell stores a '1', which means that it is in its LRS, then it will discharge the bitline. If a cell is in its HRS (stores a '0'), then ideally the resistance is high enough that no current can flow through it and that the BL is not discharged. More realistic is that the current is small and that the BL is only discharged a small amount. The final is to enable the SA or

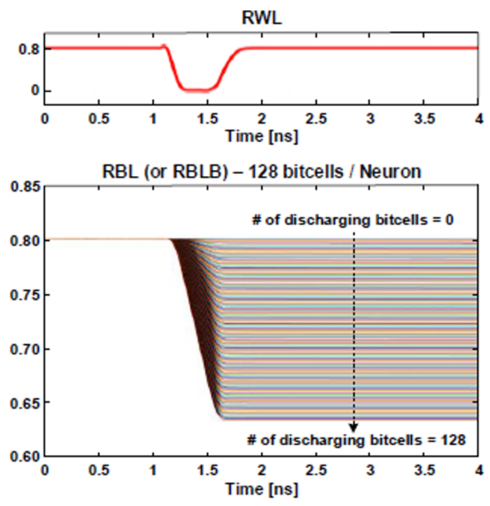
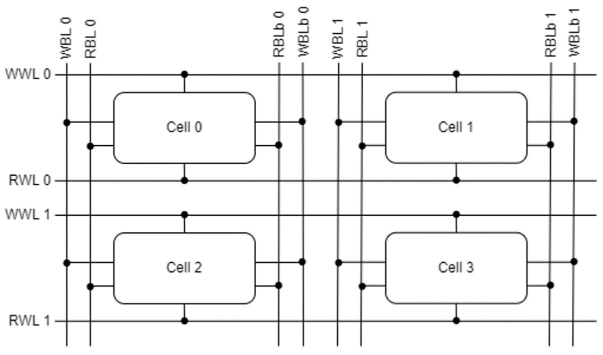
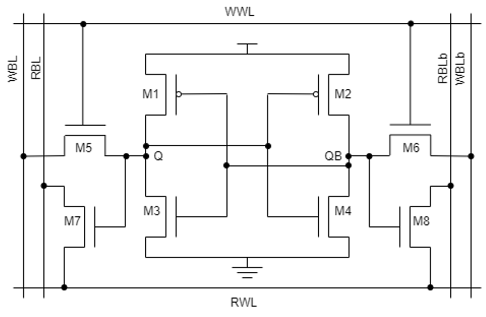


Figure 3.18: Simulated bitline current discharging and the resulting bitline voltage vs. the number of discharging SRAM bitcells in the RW-Decoupled 8T SRAM macro [18].



(a)



(b)

Figure 3.19: RW-Decoupled 8T SRAM 2x2 crossbar and cell.

ADC. Reading is performed that same as a logic operation since you do not have a BLB.

3.6. Additional components

In order to test the macros, some additional components had to be made. These include an input generator, sense amplifier and analog-to-digital converter.

3.6.1. Input generator

A Verilog-A component seen in Figure 3.25a was made to generate the inputs needed to test the CIM macros. These signals include the WL, BL, BLB, the enable signal for SA/ADC, etc.

After pre-charging the bitlines, they need to become floating. The Verilog-A component can't make floating wires. To make that happen, the circuit shown in Figure 3.25b was made. The Verilog-A component generates the "ForceH", "ForceHb", "ForceL" and "ForceLb" signals. These signals are the input of Figure 3.25b. The output of this component can be seen in Table 3.2. A "1" means made high/vdd. a "0" means made low/GND. To pre-charge the bitlines, they are first made floating. Then the signal PCON is made high. This shorts the WBL and WBLB and connects them to VDD. Figure 3.25 is an example for the 8T SRAM, but the same components for generating input signals is used in all designs. The only difference is that the Verilog-A component can have different outputs for different

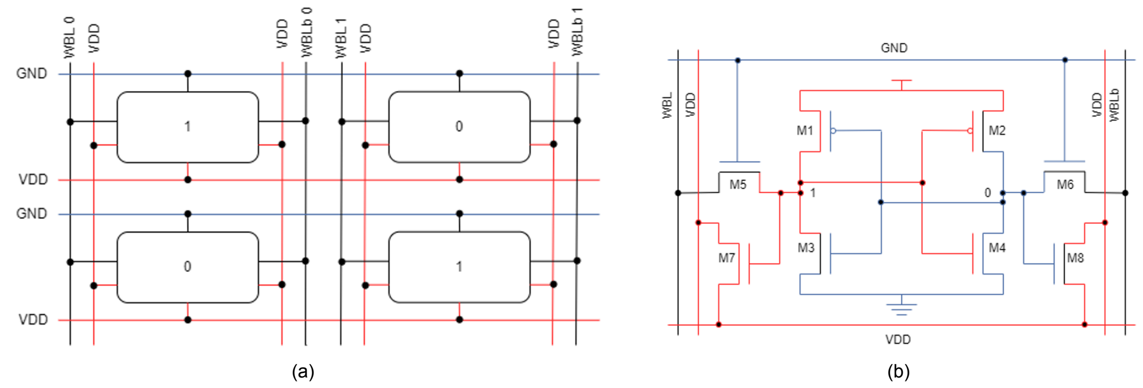


Figure 3.20: RW-Decoupled 8T SRAM example step 1: VDD on RWL and pre-charge RBL and RBLb.

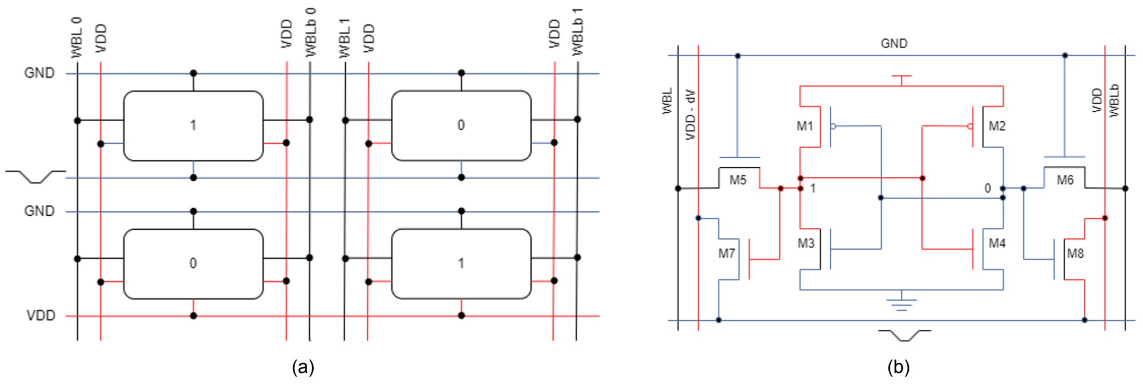


Figure 3.21: RW-Decoupled 8T SRAM example step 2: negative voltage pulse is applied on RWL.

designs.

3.6.2. Sense amplifier

Only the paper for the Push-Rule WL-Decoupled 6T SRAM design has a circuit level description of the SA that is being used. They use a reconfigurable SA that is made using two regular sense amplifiers (Figure 3.2). From the circuit of the reconfigurable SA can be derived what a single SA would look like. This is a commonly used SA, so the same one will be used in all of the designs that need a SA. The schematic of the SA can be seen in Figure 3.26. The wires for pre-charging seen in Figure 3.2 are left out in the SA, since this is already in the component mentioned in the previous paragraph (in Figure 3.25b).

3.6.3. Analog-to-Digital Converter

Out of the papers for the design for MAC, only the paper of the RW-Decoupled 8T SRAM discusses the ADC that is being used. This is not a conventional ADC. In order to keep as much parameters as

Input				Output	
ForceH	ForceHb	ForceL	ForceLb	BL	BLB
1	0	0	1	1	0
0	1	1	0	0	1
0	1	0	1	Floating	

Table 3.2: Inputs and outputs of WBL and WBLB generating circuit.

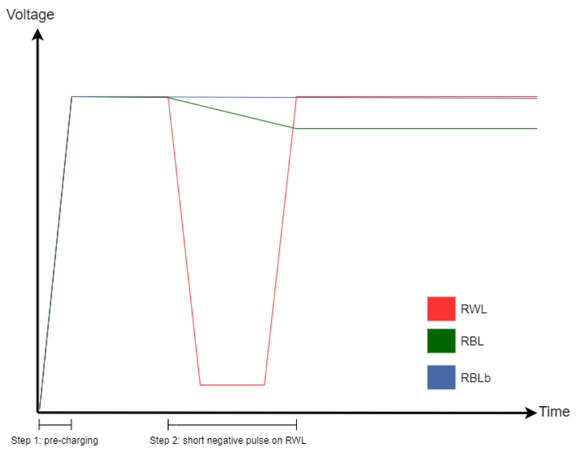
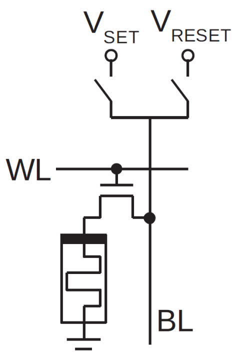
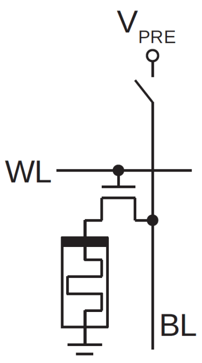


Figure 3.22: Voltages of RWL 0, RBL 0 and RBLb 0 over time during a MAC operation.



(a) Program circuit.



(b) RRAM

Figure 3.23: 1T1R cell [41].

possible equal between the memristor and SRAM based designs, it has been decided that a commonly used ADC will be used for all of the designs.

Commonly used ADCs include: Successive Approximation Register (SAR), Flash and Sigma-Delta ADC. They all have their advantages and disadvantages. For MAC operations, fast reading is important. For CIM, low energy consumption is important. However, fast ADCs have high energy consumption and the energy efficient ADCs are slow. The focus of this thesis is the the comparison between memristor based CIM and SRAM based CIM, and both types of designs will use the same ADC. For this reason, it was decided not to go into too much depth to research the best type of ADC for this application, and use the Flash ADC for its high speed, despite its high energy consumption.

The Flash ADC is made using multiple sense amplifiers based on the resolution required. The amount of sense amplifiers that output a high voltage depends on how high the input of the ADC is. For example the right most SA has a high output when the input is higher than 0.6V, the SA to the left has a high output if the input it higher than 0.7V, etc. A voltage divider is used to have increasingly lower reference voltages at the inputs of the sense amplifiers. The output of the sense amplifiers is then put into a priority encoder. The priority encoder is made using logic gates. The schematic of the SA array and priority encoder can be seen in Figure 3.27 and Figure 3.28. This is only part of the SA array and priority encoder since the full schematic would be to big to be readable. The full priority encoder, although unreadable, can be seen in Figure 3.29 to get a sense of how big the ADC is.

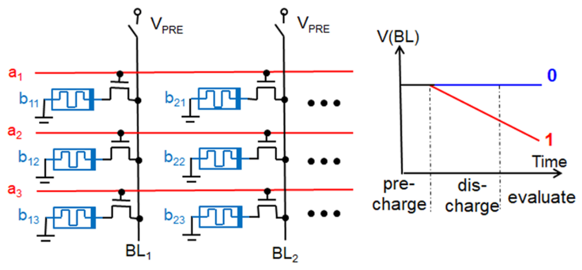


Figure 3.24: Crossbar of 1T1R cells[31].

3.7. Layouts

Layouts of the cells are made in order to find the area of the cells. The cells are made as small as possible while making sure that the designs rules are not violated. The DRC (Design Rule Checking) tool was used to verify that none of the design rules are violated. The LVS (Layout Versus Schematic) tool was used to to verify that the layout matches the schematic.

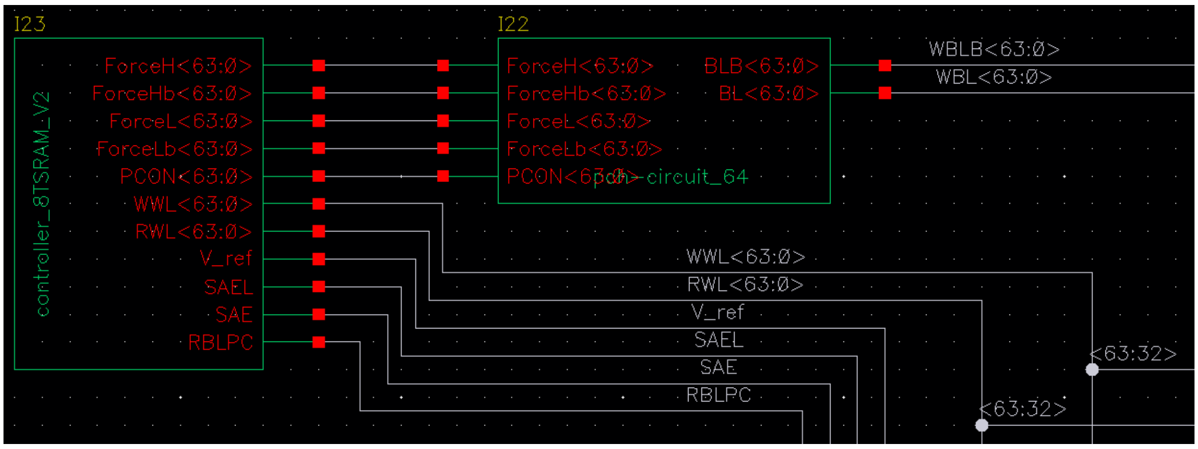
Simply taking the area of the cells that were made and multiplying this with the crossbar size, is not an accurate representation of the size of the crossbar. This is because SRAM memory arrays often use overlapping cells. The concept of overlapping SRAM cells is primarily aimed at reducing the total area occupied by the memory array by sharing common features or layers across neighboring cells. This can be done by letting cells share N-well/P-well, source/drain, contacts, vias, poly gate of by making the bitline or wordline overlap.

To get an accurate area of the crossbar without making the entire crossbar, a 3x3 crossbar was made using the concept of overlapping cells. To do this, a single cell was first made while taking in consideration how it could overlap with neighboring cells. Using the cell, a 3x3 crossbar with overlapping cells was made. The DRC tool is also used on the 3x3 crossbar. The middle cell was used to calculate the area. The middle of the overlapped part of the cell is seen as the edge of the cell. Multiplying this area by the size of the crossbar gives a more accurate representation of the area of the crossbar.

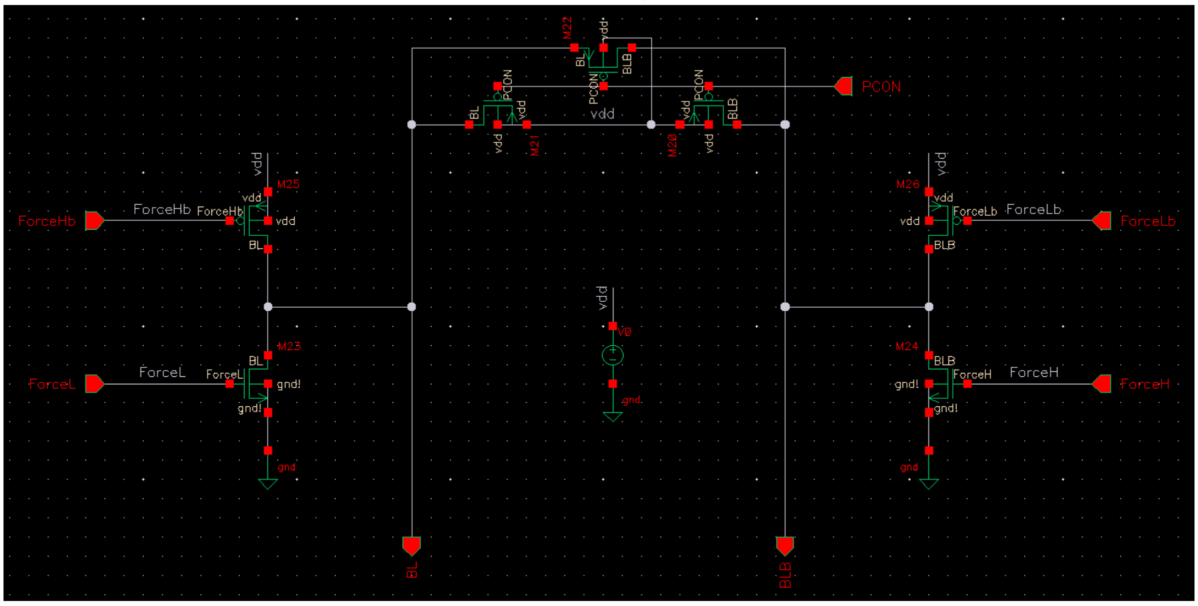
The paper of the WL-Decoupled 4+2T SRAM included an image of the cell layout, so this was recreated. Since the 8T SRAM cell is very popular, there were enough images of it's optimal layout available online, so it was recreated. The layout of a traditional 6T SRAM cell was used as a reference in creating the layouts for the Push-Rule WL-Decoupled 6T SRAM and RW-Decoupled 8T SRAM.

A layout for the 1T1R can not be made in the same way as for the SRAM cells, because the ReRAM model that is used, is a Verilog-A component. For this reason, the layout of the 1T1R crossbar in [48] is used for the comparison with the SRAM cells. In [48], a 3x3 crossbar of a 1T1R cell was developed using the same type of memristor and transistor technology node that is used for the designs in this paper. This layout can be seen in Figure 3.30.

The layout of the periphery of the CIM designs for logic, are not made. This is because they are comparable in size in all the designs and because they are very small compared to the crossbar. So they can be neglected. The ADC is the same size in all of the CIM designs for MAC. Its size however is large, even compared to the crossbar. For this reason it can not be ignored. Due to time constraints, a complete layout of the ADC is not made. Instead, an estimation is made in the following way. The ADC is made using the logic gates included in the TSMC 40nm pdk library. These logic gates come with their layout included. The total size of the ADC is calculated by adding the areas of the logic gates that are used.



(a) Verilog-A component on the left to generate input signals. On the right is an additional component to make it possible to make WBLB and WBL floating.



(b) Circuit to make make floating WBL and WBLB.

Figure 3.25: Input generating components of the 8T SRAM 64x64 crossbar.

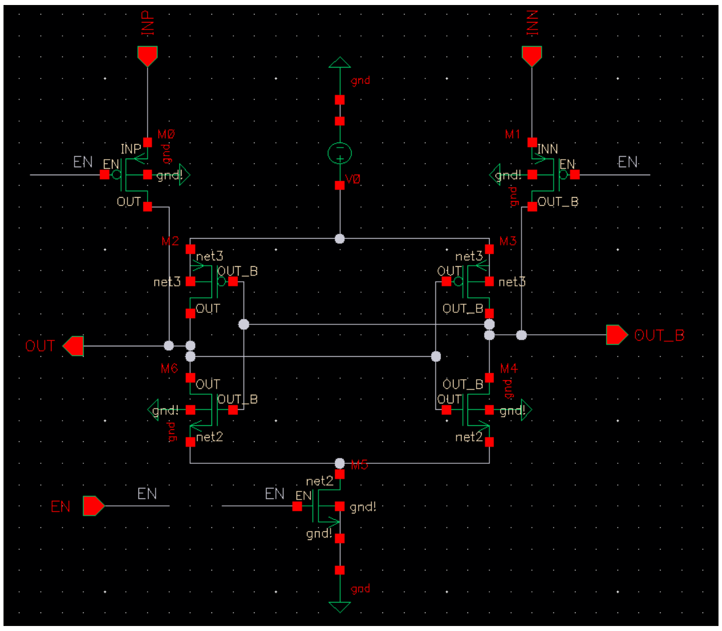


Figure 3.26: Sense amplifier schematic.

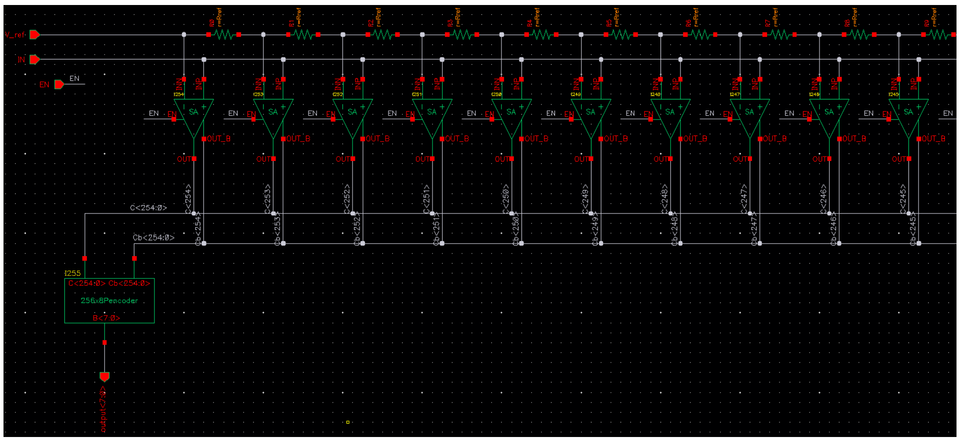


Figure 3.27: Sense amplifier array of the ADC.

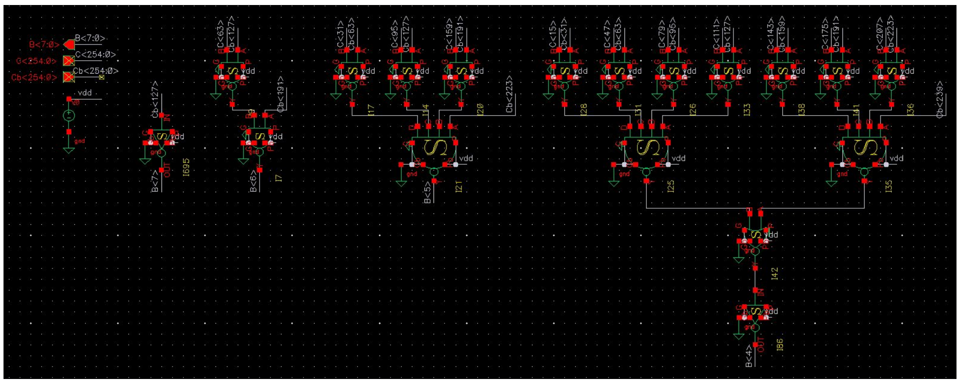


Figure 3.28: Part of the 8-bit priority encoder of the ADC.

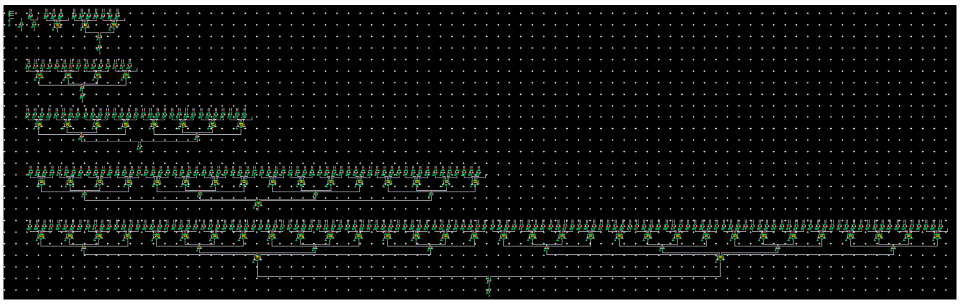


Figure 3.29: Full 8-bit priority encoder of the ADC.

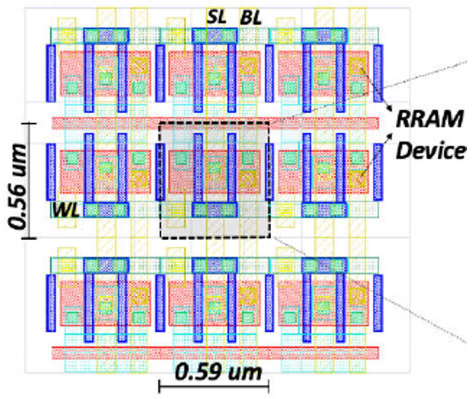


Figure 3.30: 1T1R layout[48].

4 Results

In this chapter, the results are presented. The chapter begins with an overview of the simulation setup, followed by the presentation of the layouts. Finally, the performance and energy consumption results are discussed.

4.1. Simulation setup

All the schematics, layouts and simulations are made in Cadence Virtuoso. ReRAM is used for the memristor device as explained in section 3.3. A SPICE model of a VCM ReRAM device from the university of Aachen is used [49] [50]. TSMC's 40nm pdk is used for the transistors. In the remainder of this section, the simulation parameters, methodology and some caveats are discussed.

4.1.1. Parameters

The parameters of all designs are kept equal if possible in order to make the comparison fair. As discussed in chapter 3, the same sense amplifier is used in all designs that use it and the same ADC is used for all designs for MAC. VDD of 1V is used for all designs.

The parasitics of the designs in this thesis are not included in the schematics, which means that there is no wire capacitance in the bitlines, which is needed to pre-charge them. In [48], a 1T1R crossbar was developed using the same type of memristor and the same technology node for the transistor that is used in the 1T1R design of this paper. The parasitics of the developed were extracted. A wire capacitance of 0.3fF was found for the bitlines. A capacitor of 0.3fF is connected to the bitlines of the CIM designs for logic, in order to make it possible to pre-charge them. A capacitor was also added to the bitlines of the CIM designs for MAC. The capacitance of these capacitors is bigger than 0.3fF and is different for each of the designs. It was not possible to get the correct MAC output if the same capacitance was used in all the CIM designs for MAC.

All SRAM based CIM designs except for the WL-Decoupled 4+2T SRAM, use a traditional 6T SRAM at its base, with added transistors or decoupled WLs to make CIM possible. This 6T SRAM base has the same transistor ratios in all designs. The pull up transistors are 1.5 times the (width/length) of the access transistors. The pull down transistors are 1.5 times the (width/length) of the pull up transistors. These ratios have been chosen because they provide stable read operation, which is important for CIM [51]. Other transistors made as small as possible for fast speed and low energy consumption.

The designed SA could detect a voltage difference less than 1mV, which is not realistic. This is most likely because the parasitics were not included in the schematic. In [48] a similar sense amplifier was used, which was made using the same TSMC 40nm pdk. The SA in [48] was developed and extracted to find out that it had a minimum differential sensing margin of 40mV. For this reason, the SAs used in this theses are enabled only after a voltage difference of 40mV is found.

The CIM designs for logic are evaluated by simulating the read, write and logic operations on a 64x64 crossbar. The CIM designs for MAC are evaluated by simulating the MAC operation. The weights for the cells that are used, are weights extracted from the fully connected layer (fc layer) of a CNN model trained on the MNIST and CIFAR-10 dataset. The crossbar using the weights of the CNN trained on the MNIST dataset, is 250x10. The crossbar using the weights of the CNN trained on the CIFAR-10 dataset is 64x10.

The SPICE model of a ReRAM device from the university of Aachen has many parameters that had to be chosen, see Figure 4.1. Each of these parameters can have an influence on the voltage required to start the ’SET’ or ’RESET’ operation. They can also have an effect on the high resistance state and low resistance state. Parameters have been chosen to minimize the ’SET’ and ’RESET’ voltage, to minimize the LRS, and to maximize the HRS. The parameters do not exceed the range of possible values seen in Figure 4.1a. The tables in Figure 4.1 were found in the user guide of the ReRAM device [52]. The chosen parameters led to a HRS/LRS of ≈100.

4.1.2. Methodology

For each design the performance and energy consumption of read and write operations are found by simulating these operations in Cadence Virtuoso. For logic operations, the AND operation between two wordlines is simulated. Each operation is performed multiple times and the average is taken.

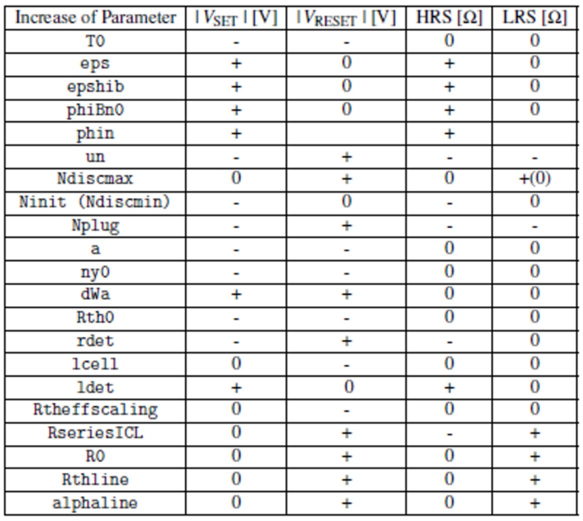
The performance and energy consumption for the MAC operation is found by simulating a real use case for a CIM design for MAC operations. This use case is performing the computations made in the fully connected layer of a convolutional neural network (CNN). The RedBit CNN [53] is used for this. With the RedBit CNN it is possible to use quantized weights and activations. Simulations are performed using 1, 2, 4 and 8 bit activations and weights quantization. However, there are some caveats in using this method. They can be read in subsection 4.1.3.

The RedBit CNN is trained using the python code found on the RedBit CNN github page. The weights are extracted from the trained model. Then inference is run on the trained model and the inputs and outputs of the fully connected layer are extracted. The inputs are are used as the activations of the crossbar. The outputs of the fc layer are compared to the output of the MAC operation from the CIM macro. The accuracy of the CIM macro is measured by comparing the output of the fc layer with the output of the CIM macro.

4.1.3. Caveats of using the RedBit CNN

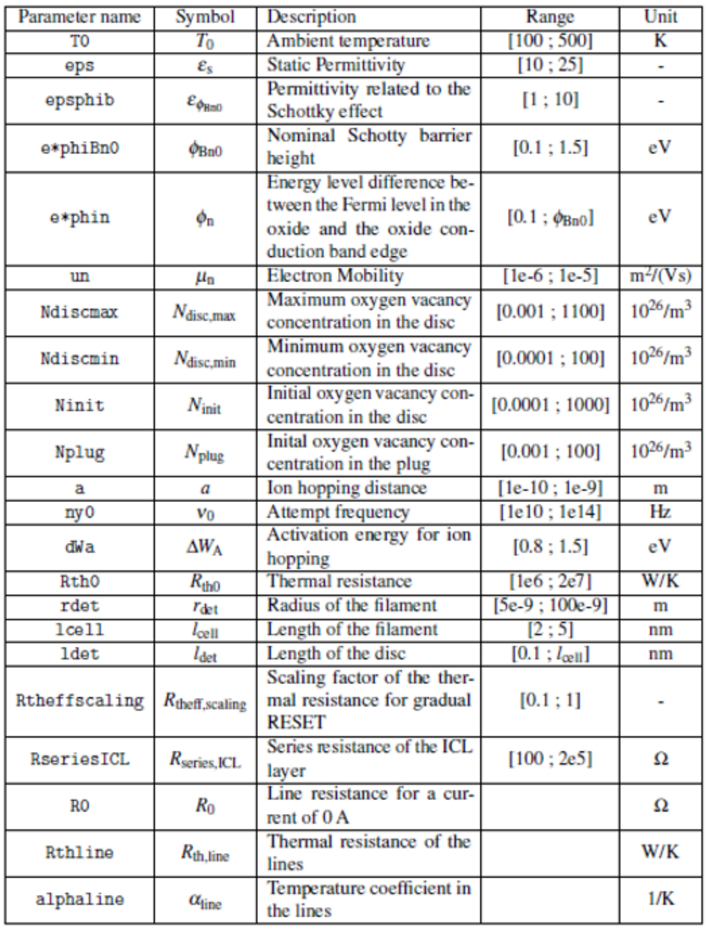
For 1-bit quantization you can use the schematics shown in chapter 3. For 2-bit quantization, you will need a copy of the crossbar. Each crossbars will store one of the two bits. The MAC operation is performed twice on both crossbars. Once for each bit of the input. The output of each of the operations will need some post processing to get the correct output. This circuitry has not been made since we are only interested in the performance/energy consumption of the CIM macro.

The RedBit CNN uses positive and negative weights/activations and the MAC designs in this thesis are only able to work with ones and zeroes (except for the RW-Decoupled 8T SRAM which can use negative weights). To still be able to get the accurate result, you could perform the MAC operation four times: (1) with only the positive weights and activations, (2) with only the absolute value of the negative weights and negative activations, (3) with the only positive weights and the absolute value of the negative activations, (4) with only the absolute values of the negative weights and the positive activations. The activations/weights that are not used, are replaced with a 0. The results from operations (1) and (2) should added together and the results from operation (3) and (4) should be subtracted from this.



Increase of Parameter	$\vert V_{\mathrm{SET}}\vert$ [V]	$\vert V_{\mathrm{RESET}}\vert$ [V]	HRS [Ω]	LRS [Ω]
T0	-	-	0	0
eps	+	0	+	0
epshib	+	0	+	0
phiBn0	+	0	+	0
phin	+		+	
un	-	+	-	-
Ndiscmax	0	+	0	+(0)
Ninit (Ndiscmin)	-	0	-	0
Nplug	-	+	-	-
a	-	-	0	0
ny0	-	-	0	0
dWa	+	+	0	0
Rth0	-	-	0	0
rdet	-	+	-	0
lcell	0	-	0	0
ldet	+	0	+	0
Rtheffscaling	0	-	0	0
RseriesICL	0	+	-	+
R0	0	+	0	+
Rthline	0	+	0	+
alphaline	0	+	0	+

(a)



Parameter name	Symbol	Description	Range	Unit
T0	T_0	Ambient temperature	[100 ; 500]	K
eps	ε_s	Static Permittivity	[10 ; 25]	-
epsphib	$\varepsilon_{\phi_{\mathrm{Bn0}}}$	Permittivity related to the Schottky effect	[1 ; 10]	-
e*phiBn0	ϕ_{Bn0}	Nominal Schotty barrier height	[0.1 ; 1.5]	eV
e*phin	ϕ_{n}	Energy level difference between the Fermi level in the oxide and the oxide conduction band edge	[0.1 ; ϕ_{Bn0}]	eV
un	μ_{n}	Electron Mobility	[1e-6 ; 1e-5]	$\mathrm{m^2/(Vs)}$
Ndiscmax	$N_{\mathrm{disc,max}}$	Maximum oxygen vacancy concentration in the disc	[0.001 ; 1100]	$10^{26}/\mathrm{m}^3$
Ndiscmin	$N_{\mathrm{disc,min}}$	Minimum oxygen vacancy concentration in the disc	[0.0001 ; 100]	$10^{26}/\mathrm{m}^3$
Ninit	N_{init}	Initial oxygen vacancy concentration in the disc	[0.0001 ; 1000]	$10^{26}/\mathrm{m}^3$
Nplug	N_{plug}	Inital oxygen vacancy concentration in the plug	[0.001 ; 100]	$10^{26}/\mathrm{m}^3$
a	a	Ion hopping distance	[1e-10 ; 1e-9]	m
ny0	ν_0	Attempt frequency	[1e10 ; 1e14]	Hz
dWa	ΔW_{A}	Activation energy for ion hopping	[0.8 ; 1.5]	eV
Rth0	R_{th0}	Thermal resistance	[1e6 ; 2e7]	W/K
rdet	r_{det}	Radius of the filament	[5e-9 ; 100e-9]	m
lcell	l_{cell}	Length of the filament	[2 ; 5]	nm
ldet	l_{det}	Length of the disc	[0.1 ; l_{cell}]	nm
Rtheffscaling	$R_{\mathrm{theff,scaling}}$	Scaling factor of the thermal resistance for gradual RESET	[0.1 ; 1]	-
RseriesICL	$R_{\mathrm{series,ICL}}$	Series resistance of the ICL layer	[100 ; 2e5]	Ω
R0	R_0	Line resistance for a current of 0 A		Ω
Rthline	$R_{\mathrm{th,line}}$	Thermal resistance of the lines		W/K
alphaline	α_{line}	Temperature coefficient in the lines		1/K

(b)

Figure 4.1: (a) Effect of different parameters on the memristor. (b) Range and meaning of different parameters in the memristor model.

Additional circuitry is needed for this, which again is not made. The output of these operations are added manually and the result is compared with what the output should have been in order to calculate the accuracy of the CIM macro.

The RedBit CNN is used despite its caveats, because of its ability to use 1, 2, 4 and 8-bit quantized weights and activations. No other CNN has been found with this feature or with the ability to use only positive weights/activations. Making a custom CNN would be out of the scope for this thesis.

4.2. Schematics

The schematics of the designs can be seen in Figure 4.2 until Figure 4.6. Each figure shows a single cell, periphery and crossbar of the design. The crossbar is made using the cells. A copy of the periphery is connected to each column of the crossbar.

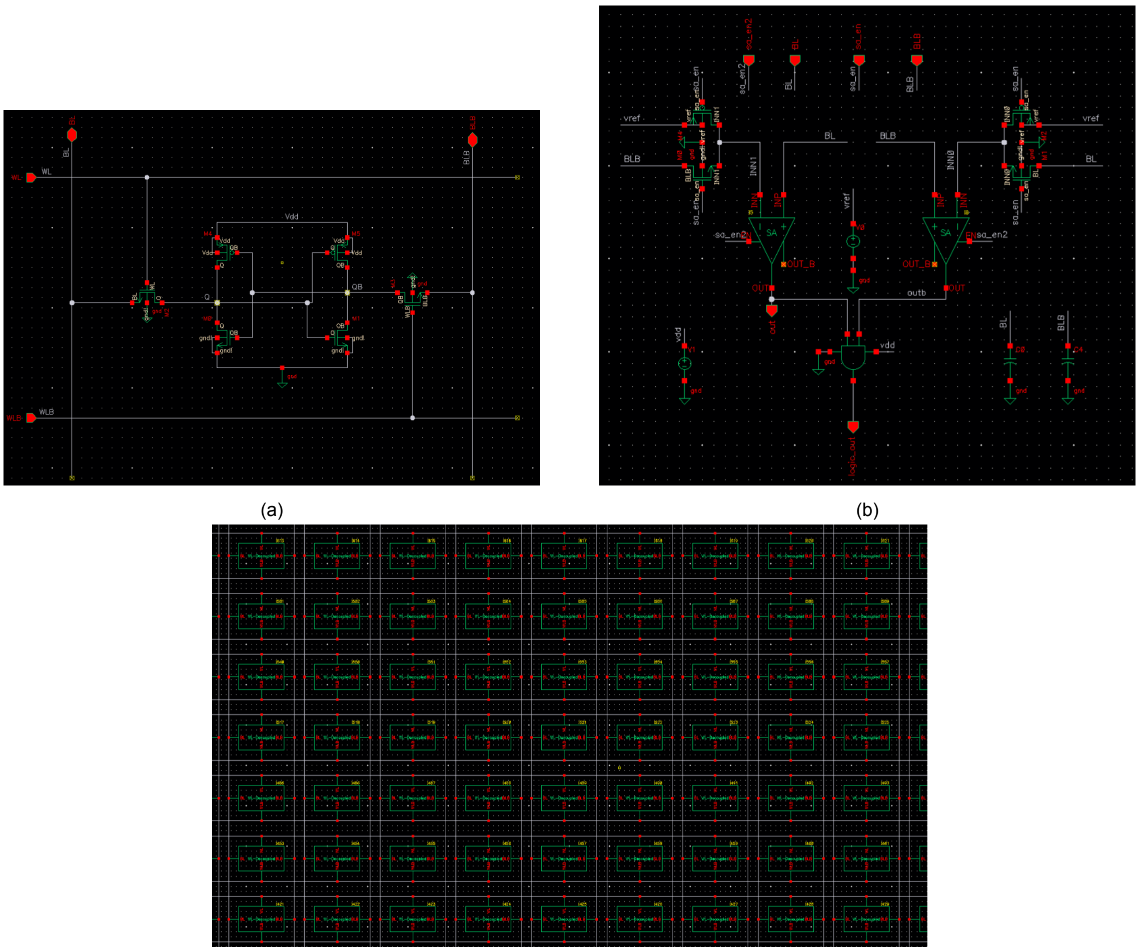
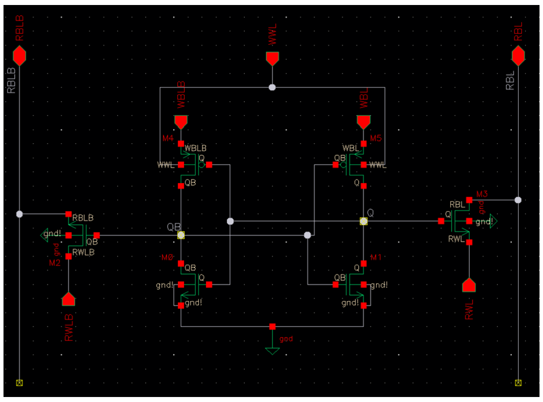
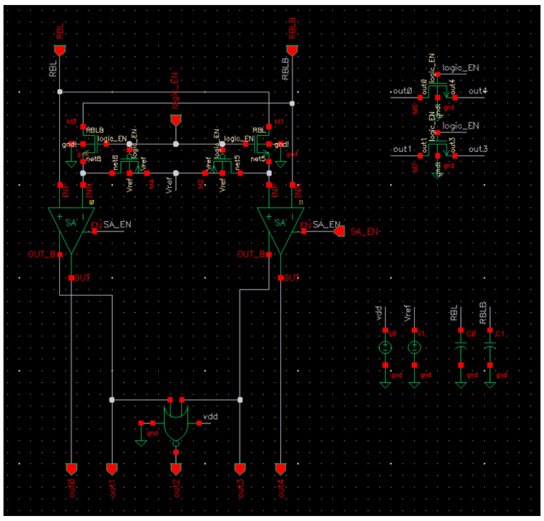


Figure 4.2: Push-Rule WL-Decoupled 6T SRAM schematics. (a) Single cell. (b) Periphery. (c) Crossbar.

The schematic of the Push-Rule WL-Decoupled 6T SRAM cel in Figure 4.2a is the same as a traditional 6T SRAM, except the access transistors are decoupled. The periphery seen in Figure 4.2b is made using two of the sense amplifiers from Figure 3.26. There are four transistors that let the input of the SAs be either the bitline or the reference voltage. The AND port is taken from the TSMC library. Two capacitors are connected to the bitlines to simulate the capacitance that these wires could have on a chip. These capacitances are necessary to be able to pre-charge the bitlines. The crossbar in Figure 4.2c is made using the cells in Figure 4.2a


(a)


(b)

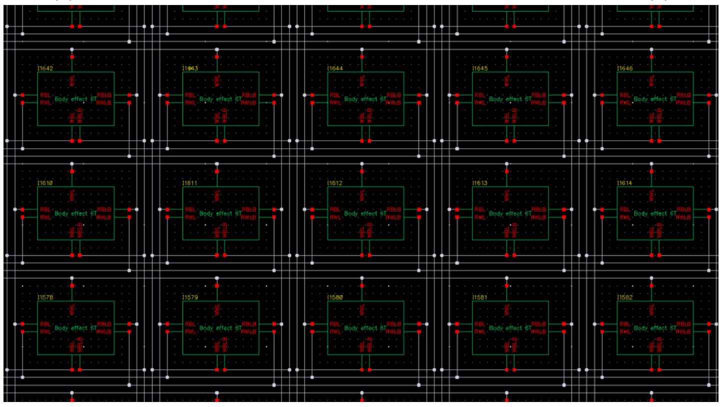
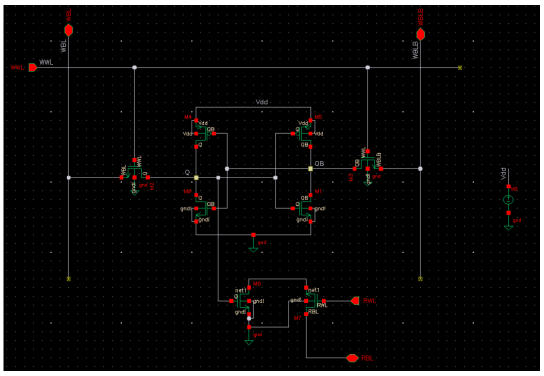

(c)

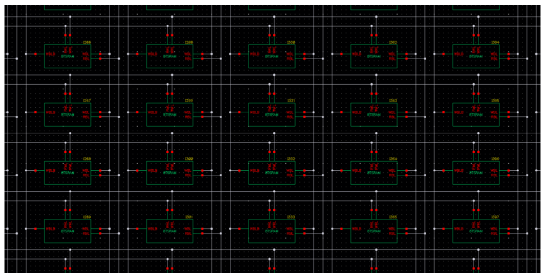
Figure 4.3: WL-Decoupled 4+2T SRAM. (a) Single cell. (b) Periphery. (c) Crossbar.

The schematic of the WL-Decoupled 4+2T SRAM cell seen in Figure 4.3a has the same cross coupled inverters as a traditional 6T SRAM cell. But the n-well of the pull up transistors is connected to the WWL. There are no access transistors and the transistors M2 and M3 are used for reading and logic operations by discharging one of the bitlines. The periphery in Figure 4.3b is similar to that of the Push-Rule WL-Decoupled 6T SRAM. It has two sense amplifiers with four transistors that let on of the inputs be either a bitline or a sense amplifier. The OR gate is taken from the TSMC library. There are two more transistors that connect out0 with out4 and out1 with out3 when you are performing a read operation. This is because those outputs will have the same output during a normal read operation. By shorting the wires, the correct output will become available faster because the voltage is supplied by two sense amplifiers.

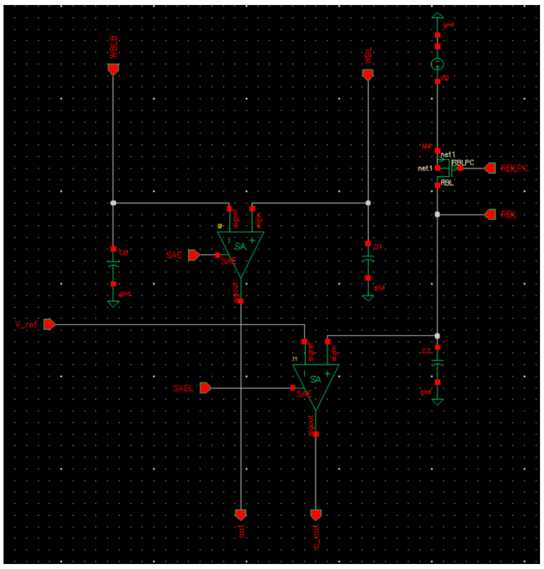
The 8T SRAM cell seen in Figure 4.4a is a traditional 6T SRAM cell with two additional transistor for performing logic operations. The periphery of the 8T SRAM, seen in Figure 4.4c, contains two SAs. One is used for logic operations. The other is used for regular read operations. Like with all the designs, a copy of this periphery is connected to each column of the crossbar. The periphery of the 8T SRAM for MAC in Figure 4.4d is a little different, since an ADC is needed. Other than the ADC, there is no difference. The cells and crossbar is the same as for the 8T SRAM logic design.



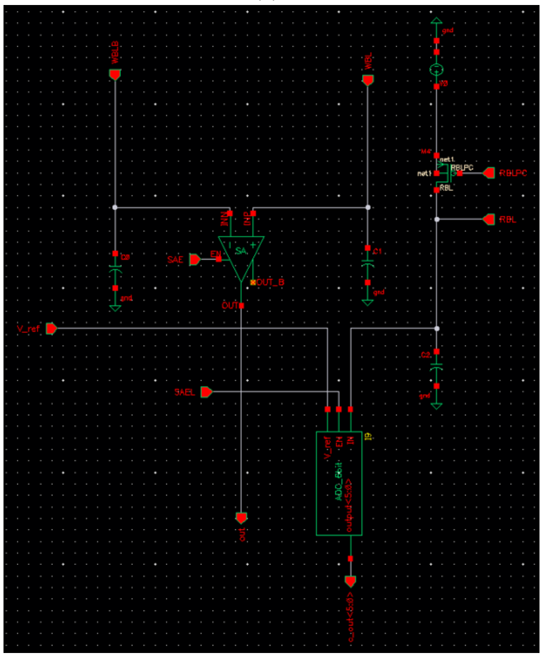
(a)



(b)



(c)

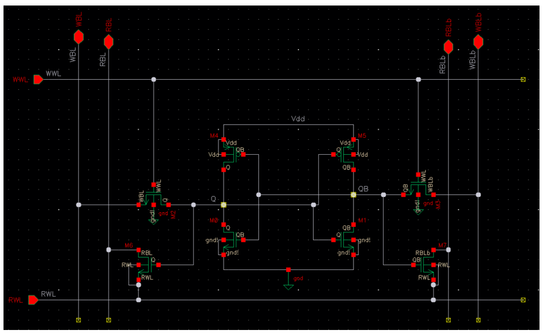


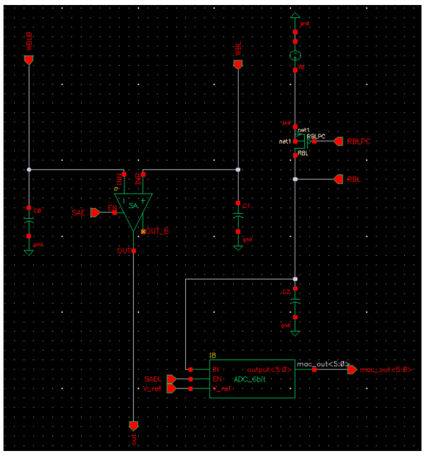
(d)

Figure 4.4: 8T SRAM. (a) Single cell. (b) Crossbar. (c) Logic periphery. (d) MAC periphery

The schematic of the RW-Decoupled 8T SRAM cell seen in Figure 4.5a is same same as a traditional 6T SRAM, but with two additional transistors for performing logic operations. Just like the 8T SRAM. The periphery in Figure 4.5b consists of two parts. The part with the sense amplifier is used for performing regular read operations. The part with the ADC is for performing the MAC operation.

The schematic of the 1T1R SRAM cell in Figure 4.6a contains a SPICE model of an ReRAM device and a transistor. The periphery seen in Figure 4.6b is the same as it would be for a traditional 6T SRAM, except a reference voltage is used instead of a BLB. This is for the design for logic. The periphery for the design for MAC is the same, except the SA is replaced with an ADC. A copy of this periphery is connected to each column of the crossbar.


(a)


(b)

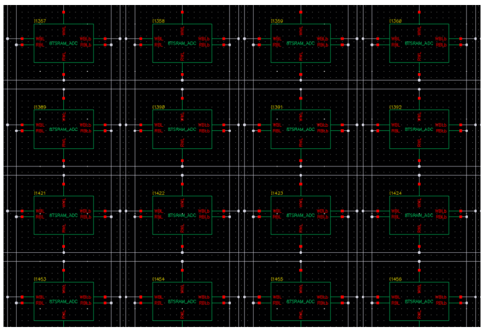
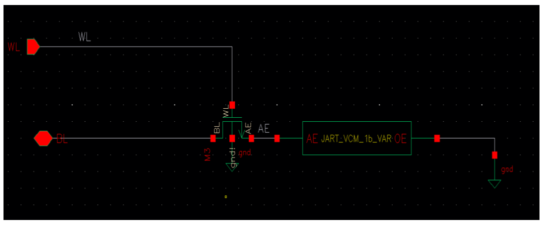
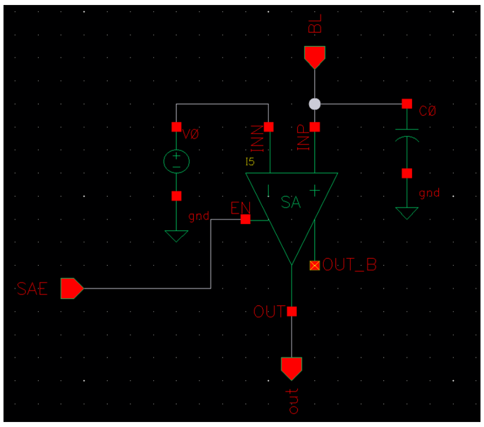

(c)

Figure 4.5: RW-Decoupled 8T SRAM. (a) Single cell. (b) Periphery. (c) Crossbar.



(a)



(b)

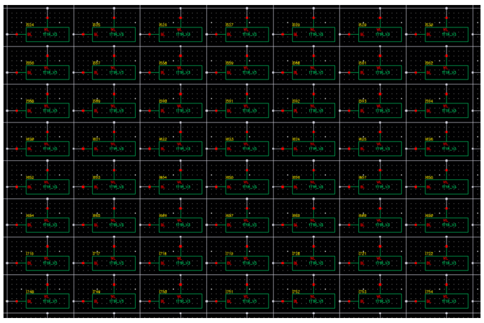
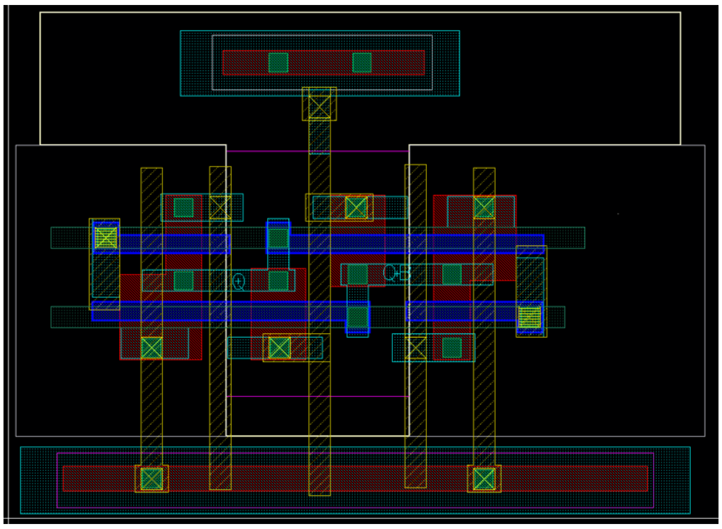
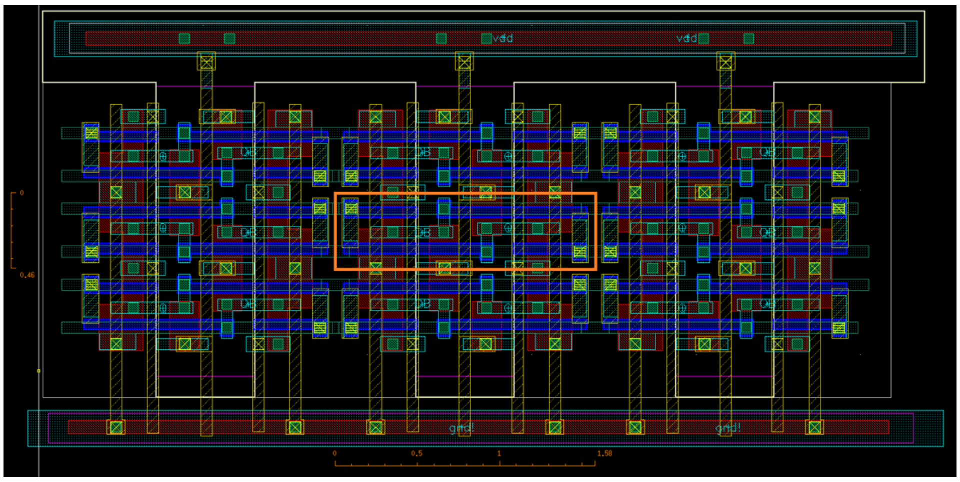

(c)

Figure 4.6: 1T1R. (a) Single cell. (b) Periphery. (c) Crossbar.


(a)



(b)

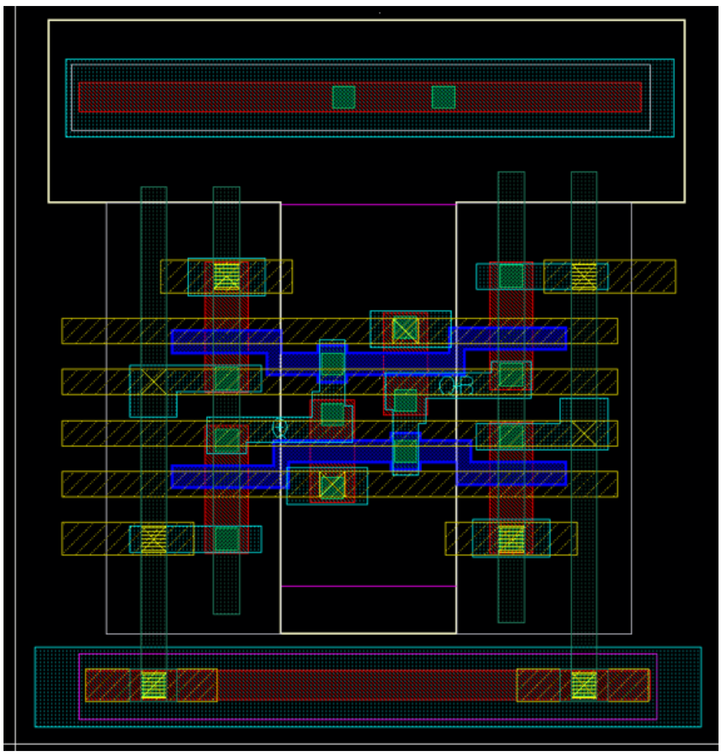
Figure 4.7: Push-Rule WL-Decoupled 6T SRAM. (a) cell layout. (b) 3x3 crossbar layout. Cell area: 0.73 μm^2.

4.3. Layouts

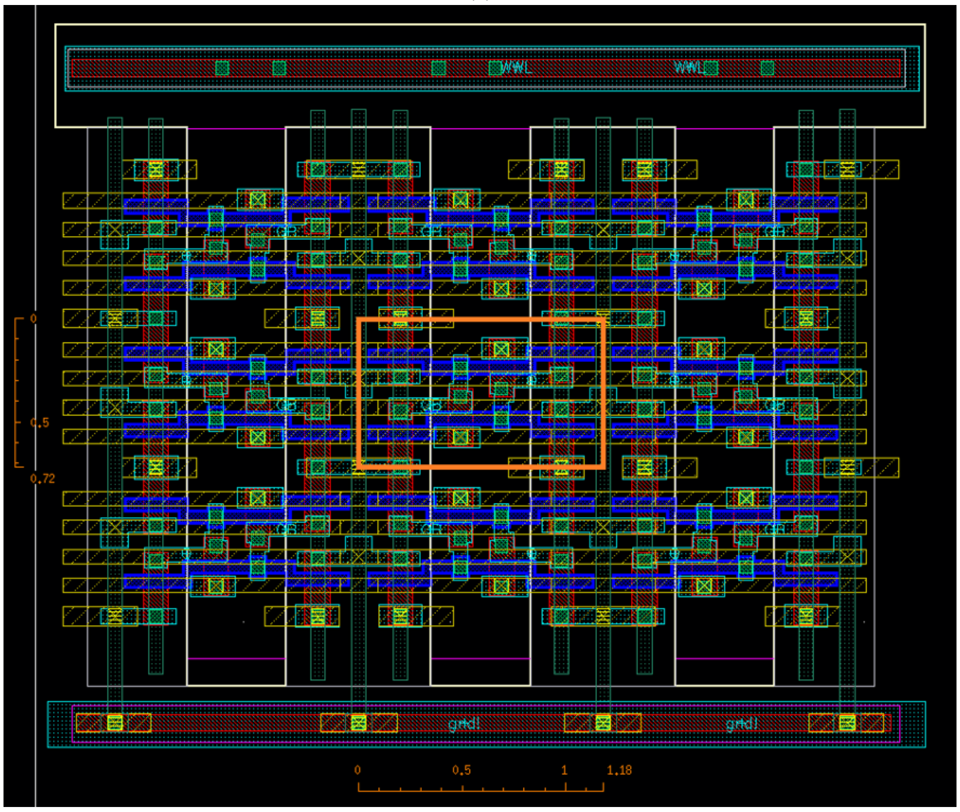
The layouts of the individual cells, as well as the cells in a 3x3 crossbar can be seen in Figure 4.7 until Figure 4.11. The figures with the crossbar layout contain an orange border in the middle cell. This is the part that is measured to get the area of the cell. The DRC and LVS tools have been used on all the layouts to ensure that they do not break any design rules and to verify that the layout is the same as the schematic. The Push-Rule WL-Decoupled 6T SRAM shown in Figure 4.7 is made using a traditional 6T SRAM layout as a reference. Some parts of the cells in the crossbar have been merged with neighboring cells to reduce space. The connection point of the BLs and BLBs of neighboring cells are merged, the ground terminal of the pull down transistors of neighboring cells are merged and the VDD terminal of the pull up transistors of neighboring cells are merged. The area of the cell, which is the area within the orange border in Figure 4.7, is 0.73 μm^2

The layout of the WL-Decoupled 4+2T SRAM seen in Figure 4.8 has been made using the layout in [17] as a reference. The ground terminal of the pull down transistors of neighboring cells are merged and the BLs and BLBs of neighboring cells are merged. The area of the cell is 0.85 μm^2

There was no figure of the layout of the 8T SRAM in [15]. There are however many to be found online since it is a commonly used design. The layout of the 8T SRAM shown in Figure 4.9 was made using

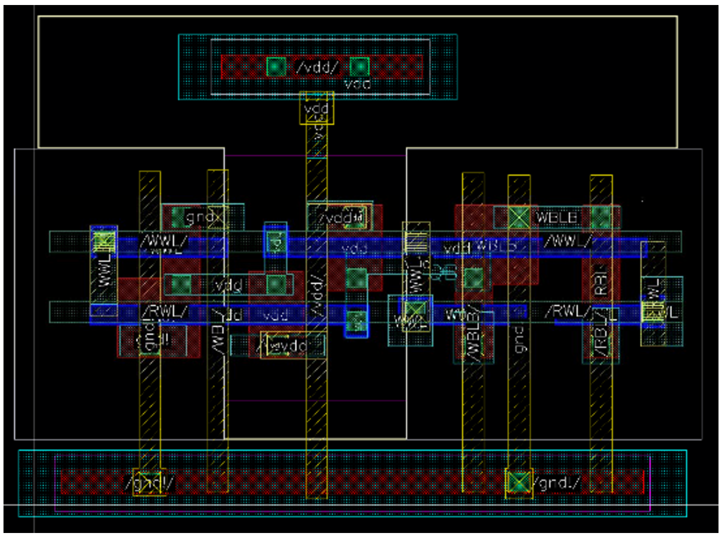


(a)

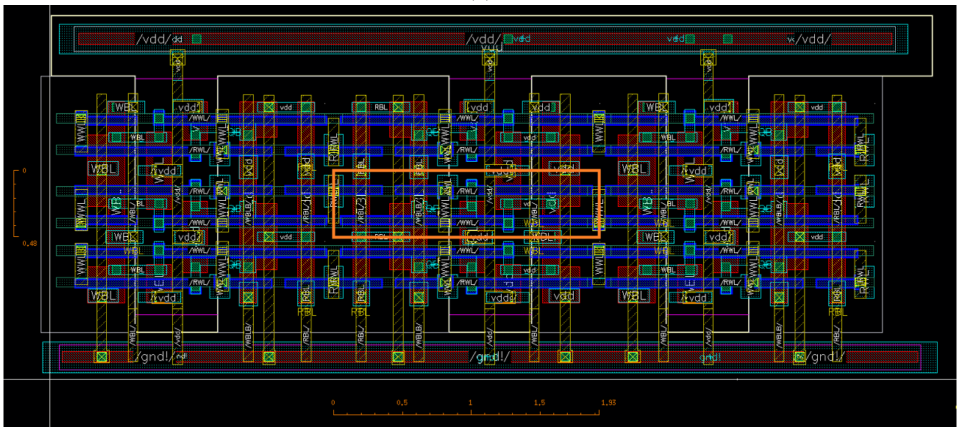


(b)

Figure 4.8: WL-Decoupled 4+2T SRAM. (a) cell layout. (b) 3x3 crossbar layout. Cell area: 0.85 μm^2.

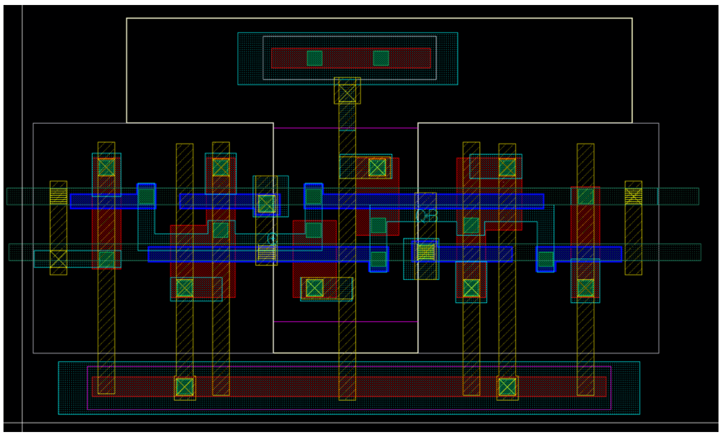


(a)

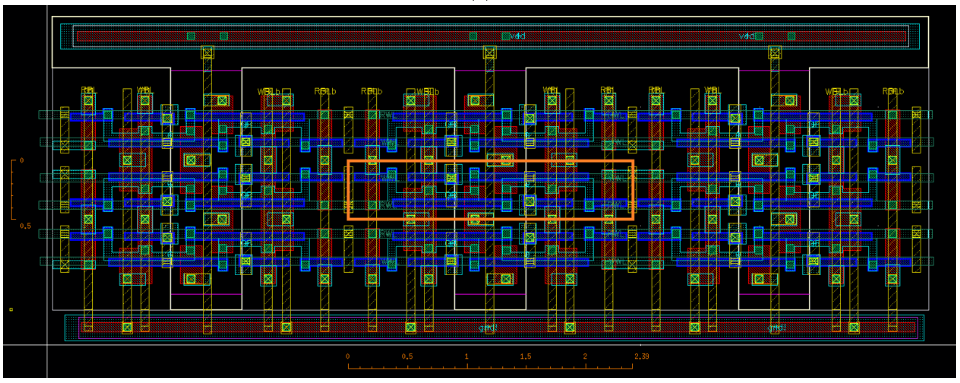


(b)

Figure 4.9: 8T SRAM. (a) cell layout. (b) 3x3 crossbar layout. Cell area: 0.93 μm^2.



(a)



(b)

Figure 4.10: RW-Decoupled 8T SRAM. (a) cell layout. (b) 3x3 crossbar layout. Cell area: 1.20 μm^2.

the layout in [54] as a reference. The VDD of pull up transistors, ground of pull down transistors, WBL and WBLB, WWL and RBL of neighboring cells are merged to save space. The area of the cell is 0.93 μm^2.

The layout of the RW-Decoupled 8T SRAM seen in Figure 4.10 is similar to that of the Push-Rule WL-Decoupled 6T SRAM. Cells are merged with the cells above and below them. The VDD of the pull up transistors, ground of the pull down transistors, RBL, RBLb, WBL and WBLb are merged with the neighboring cells. The area of the RW-Decoupled 8T SRAM is 1.20 μm^2.

The figure of the layout of the 1T1R design, shown in Figure 4.11, is taken from [48]. This figure shows a 3 by 3 crossbar. A single cell is shown within the dotted lines. The area of that cell is 0.33 μm^2.

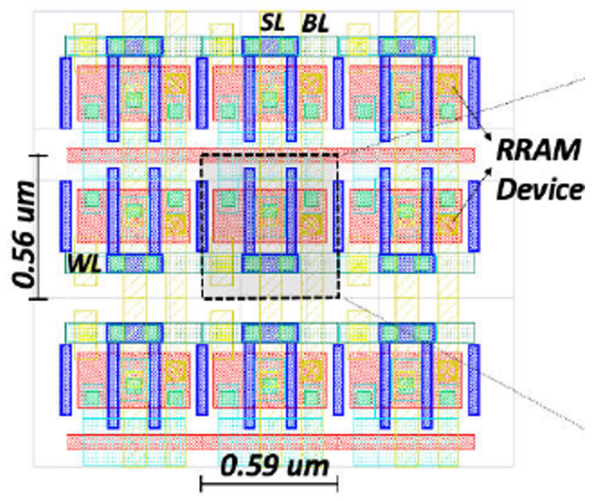


Figure 4.11: 1T1R 3x3 crossbar layout[48]. Cell area: 0.33 μm^2.

4.4. Area

The area of each design can be seen in Figure 4.12 and Figure 4.13. The area of the cells is measured in the layouts shown in section 4.3. The area of the crossbars has been calculated by multiplying the area of the cell with the amount of cells in the crossbar. The area of the ADC has been calculated by adding the area of each of the logic gates. Layouts of the logic gate are included in the TSMC library in Cadence Virtuoso. The calculation of the ADC area can be seen in Table 4.1 and Table 4.2.

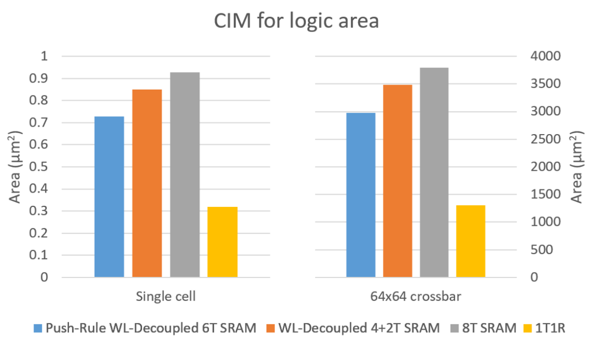


Figure 4.12: Area of CIM for logic designs.

In Figure 4.12 can be seen that the area of the memristor based design for logic in much smaller than the SRAM based designs for logic. The same can be said about the area of the designs for MAC shown in Figure 4.13. The relative difference in area is a lot smaller when the area of the ADC is included.

Logic gate	Area (μm^2)	Amount	Total area (μm^2)
NAND2	0.71	59	41.63
NAND4	1.23	15	18.52
INV	0.53	3	1.59
NOR2	0.71	1	0.71
NOR4	1.23	3	3.70
6-bit priority encoder area:			66.15

Table 4.1: Area of 6-bit priority encoder for 6-bit ADC used by crossbar for CIFAR-10.

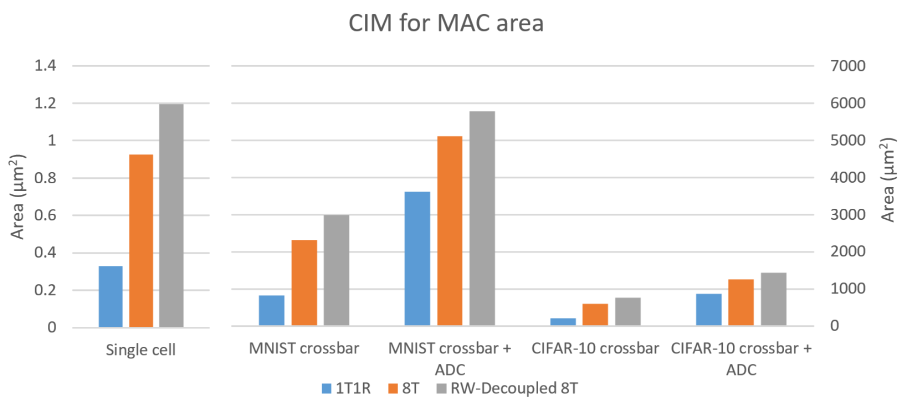


Figure 4.13: Area of CIM for MAC designs.

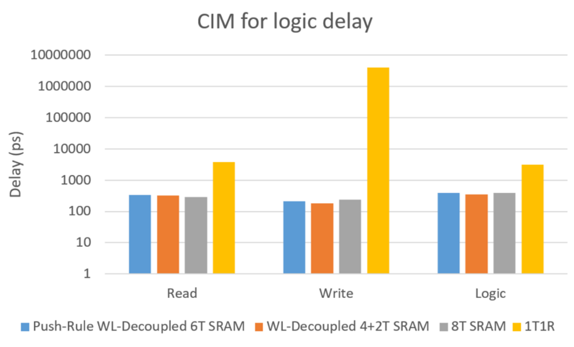
Logic gate	Area (μm^2)	Amount	Total area (μm^2)
NAND2	0.71	249	175.69
NAND4	1.23	66	81.50
INV	0.53	4	2.12
NOR2	0.71	2	1.41
NOR4	1.23	15	18.52
8-bit priority encoder area:			279.24

Table 4.2: Area of 8-bit priority encoder for 8-bit ADC used by crossbar for MNIST.

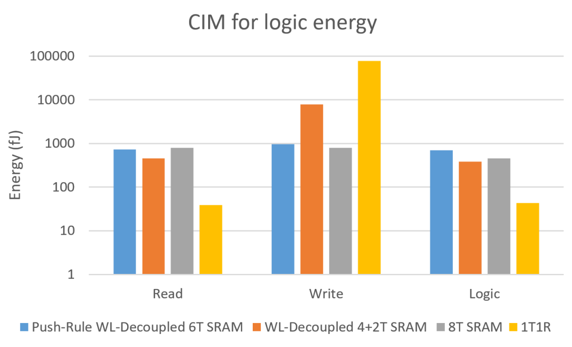
4.5. Simulation results

The simulation results can be seen in Figure 4.14 until Figure 4.16. The accuracy in Table 4.3 is calculated using Equation 4.1. This is done for the output of each of the columns. The average is then taken. This is done 10 times for 10 different inputs. The average of those calculations is then considered as the accuracy and is what is seen in Table 4.3.

$$Accuracy = (1 - \frac{|expected\ output - simulation\ output|}{expected\ output}) * 100\% \tag{4.1}$$



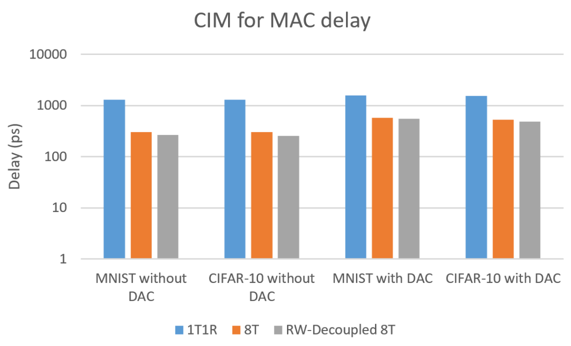
(a)



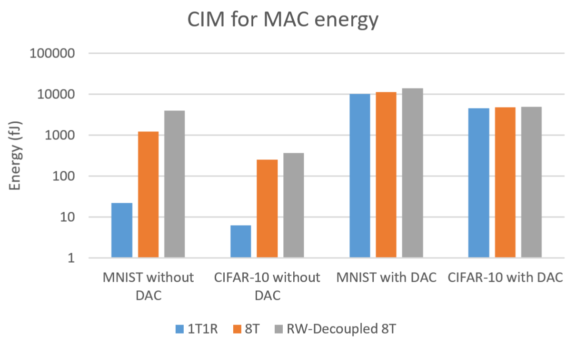
(b)

Figure 4.14: Simulation results of the delay and energy consumption during a single read, write and logic operation.

In Figure 4.14a can be seen that the delay of the memristor based CIM design for logic is a lot higher than that of the SRAM based designs. Especially the write delay. The energy consumption of the logic designs is seen in Figure 4.14b. In this figure can be seen that the write energy of the memristor based design is higher than that of the SRAM based designs. The read and logic energy is lower than that of the SRAM based designs.



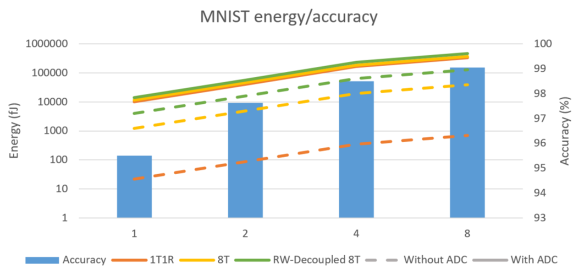
(a)



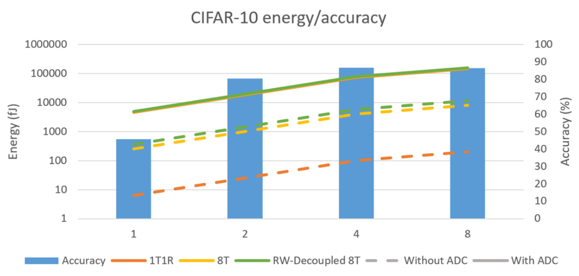
(b)

Figure 4.15: Simulation results of the delay and energy consumption of computing the MAC operation in the fully connected layer of a CNN.

The delay of the MAC designs, seen in Figure 4.15a, is the highest for the 1T1R design. The ADC does not have a large impact on this difference. In Figure 4.15b can be seen that the energy consumption of the memristor based CIM design for MAC, is much lower than that of the SRAM based designs. However with the ADC included, the relative difference becomes very small.



(a)



(b)

Figure 4.16: Energy consumption and accuracy for different number of bits of quantization.

In Figure 4.16 can be seen that the accuracy increases with higher number of bits for quantization. The energy consumption also increases exponentially. This increase in accuracy is the highest when switching from one to two bits of quantization. In this figure can also be seen that there is very little difference in the energy consumption when the ADC is included. The accuracy shown in Figure 4.16 is the reported accuracy of the python code. In Table 4.3, the accuracy refers to the accuracy of the output of the MAC operation in the CIM macro compared to the output of the inference that in run using the python code. In this table can be seen that all designs have a high accuracy.

	1T1R (%)	8T (%)	RW-Dec. (%)
MNIST	98.4	99.2	96.2
CIFAR-10	89.5	90.2	90.7

Table 4.3: Accuracy of CIM macro output compared to the fully connected layer output in software inference.

5 Discussion

To find out whether memristors or SRAM is better suited for CIM, schematics of memristor based and SRAM based CIM designs have been made in Cadence Virtuoso, a Layout of the cells were made to find the area and simulations have been performed on the designs to find the delay, energy consumption and accuracy.

From the results can be seen that the area of the memristor based CIM design is smaller than the SRAM based CIM designs (Figure 4.12 and Figure 4.13). This was expected since the 1T1R uses only one transistor compared to at least six in the SRAM designs. Additionally, a memristor is used, which is also not very large, see Figure 4.11.

In the CIM designs for MAC, ADCs are used, which occupy a large area. An ADC is used for both the memristor and the SRAM based designs. The area of the ADCs is bit bigger than the crossbar of the SRAM designs and they are more than three times the size of the crossbar of the memristor design. This diminishes the advantage that the memristor design has. Without the ADC, the memristor design is only 36% the size of the smallest SRAM based design. With the ADCs included, the memristor design is only around 70% the size of the SRAM design. Even with the ADC, the memristor based design is much smaller than the SRAM based design for MAC. In the designs for logic, which do not require ADCs, the advantage of the memristor design is even larger.

In the simulation results of the designs for logic can be seen that the memristor design performs worse than the SRAM based designs (Figure 4.14). The read, write and logic operations of the 1T1R design are all much slower than on the SRAM based designs. The read, write and logic delay of the 1T1R design is 11, 17k and 8 times respectively higher than the slowest SRAM design. The write energy consumption of the 1T1R design is 10 times higher than the write energy of the SRAM based design with the highest energy consumption and 97 times higher energy consumption than the SRAM based design with the lowest energy consumption. The read and logic operations consume 11 times less energy in the 1T1R design compared to the SRAM based design with the lowest energy consumption.

In the CIM for logic designs, writing to the 1T1R cell, consumes 97 times as much energy as writing to the 8T SRAM cell. Performing the logic operation with two 1T1R cells consumes 11 times less energy compared to the 8T SRAM cell. After writing once, the 1T1R design consumes less energy than the 8T SRAM only after performing 185 logic operations.

In the simulation results of the designs for MAC can be seen that the delay of the 1T1R is 3 times higher than in the SRAM based designs (Figure 4.15a). The ADC does not influence this delay much. The energy consumption of performing the MAC operation is 42-56 times lower with the 1T1R design if no ADC is used (Figure 4.15b). The energy consumption of the ADC is so high that all designs have similar energy consumption when it is included.

In Figure 4.16 can be seen that the accuracy of the CNN increases a lot when increasing the quantization from 1 to 2 bits. Increasing the number of bits exponentially increases the energy consumption. For the CNN trained on the MNIST dataset, increasing the number of bits for quantization has diminishing returns, but the accuracy is already high. The accuracy of the CNN trained on the CIFAR-10 dataset is low and probably needs more than 8 bits precision to get a higher accuracy. In Figure 4.16 can also be seen that the energy consumption with the ADCs is very similar in all of the designs.

In Table 4.3 can be seen that the outputs of all the CIM designs are very close to what the output should be. None of the designs has a significantly higher accuracy than the others. From this can be concluded that both the SRAM and memristor based are equally suitable for performing MAC operations.

If no ADCs were used and delay and energy consumption would be equally important, then the 1T1R design is better for MAC operations in applications like CNNs. This is because writing will be only once. Read delay is only 3-4 times longer, while read energy is 42-181 times lower. The area of the 1T1R is also smaller. The advantage of lower read energy increased exponentially with higher number of bits for quantization. This is because with multiple bits, multiple crossbars can be used in parallel. The delay remains the same except for the overhead of combining the results of the different crossbars. Higher number of bits for quantization is beneficial since it increases the accuracy of the neural network.

This advantage is mostly in analog domain, because with the ADC, the read energy is only 1.1-1.4 times lower. The 1T1R only has 10% lower energy consumption and consumes 30% less area than the 8T SRAM, when the ADCs are included. This while being 3 times as slow.

Since it is necessary to convert the analog output into a digital signal, it will be better to use SRAM based CIM for MAC operations if area, delay and energy consumption is equally important. If area and energy consumption is more important, then memristor based CIM might be better. For logic operations it would only be better to use the 1T1R design in an application where there are at least 185 times more logic operations performed than write operations. And only if speed is not important.

Future research could improve or add to the work of this thesis in a few ways.

The parasitics of the layouts could be extracted and added to the schematics. Then the simulations can be performed on the schematics with the parasitics included. This will give a more accurate representation of how a real chip would behave. Even better would be if the CIM macros were manufactured.

In [48], Singh et al. were able to write to the memristor in only 2ns, while 4μs was used for the simulations in this paper. The energy consumption of this write operation is not mentioned in the paper. For future research, the parameters of the memristor could be tuned, which might lead to better performance and/or energy consumption.

The ADCs used in the CIM designs for MAC, are large and consume a lot of energy. This diminishes the advantages that a memristor based design could have over a SRAM based design. Future research could explore the use of ADCs with energy consumption, speed and area, that is better suited for CIM designs.

The 8T SRAM and the 1T1R design using ReRAM are the most popular designs for CIM at the time of writing this thesis. The 8T SRAM also had the best overall performance and energy consumption of the SRAM designs. There might however be novel SRAM and memristor based CIM design that perform better. This could be using ReRAM, or using a different technology. The same research as in this thesis could be done in the future using different CIM designs and memristor technologies and the conclusion could be different from this thesis.

In this thesis, Memristor and SRAM based CIM designs are compared to each other based on energy consumption, speed and area. However, there are more metrics that they could be compared with, like: endurance, production cost, accuracy, 3D integration, etc.

6 Conclusion

The exponential performance increase that processor have had, is declining because of three ’walls’. The memory wall, where the memory can’t keep up with the speed of the CPU. The power wall, where increasing power consumption becomes expensive and makes cooling more difficult. The ILP wall, where is becomes harder to utilize all logical cores because of the difficulty to parallelize the code. Additionally, CMOS scaling is reaching it’s limit.

Computation in memory (CIM) could help to solve the memory and power wall by performing calculations in the memory instead of sending data between the CPU and the memory. Less data request to the memory elevates the memory wall and less data transfer reduces the power consumption. CIM is often made possible using modified SRAM cells.

Using memristors instead of SRAM could add to these benefits of CIM even more. Memristors could consume less energy because of their non-volatile nature and are smaller than an SRAM cell. It is difficult to prove this hypothesis using data available in literature. There are no comparisons of using memristor based CIM designs versus using SRAM based CIM designs. Trying to compare papers about SRAM based CIM with papers about memristor based CIM will not give an accurate result. This is because the authors of different papers test their designs using different metrics and parameters.

In this paper, different SRAM based CIM designs are compared with a popular memristor based CIM design, the 1T1R design. They are compared by recreating the designs using as much of the same parameters as possible and then simulating both logic and mac computations. ReRAM was found te be the most suitable memristor device for CIM, so a SPICE model of it was used for the simulations. A layout of the designs is made to compare the area of the designs.

From the layouts could be seen that the area of the memristor based CIM design for logic operations is more than 60% smaller than the smallest SRAM based designs. For MAC operations, an ADC is required. The area of the ADC is so big that the memristor based design is only 30% smaller than the smallest SRAM based design.

From the simulations could be seen that the memristor based design for logic had 11 times longer read, 17k times longer write and 8 times longer logic delay than the slowest SRAM based designs for logic. The write energy is 97 times higher than that of the SRAM design with the lowest energy consumption. The energy consumption for performing read and logic operations is 11 times lower with the memristor design.

The delay of the memristor based design for MAC, is 3 times higher than the SRAM based designs. The energy consumption of performing the MAC operation is 42-56 times lower with the 1T1R design if the energy consumption of the ADC is excluded. With the energy consumption of the ADC included, the 1T1R only has 1.1-1.4 times lower energy consumption.

From these results can be concluded that the most suitable technology for CIM depends on the application. If area, delay and energy consumption are equally important, then SRAm based CIM would be better for MAC operations. If area and energy consumption are more important, then memristor based CIM could be better if the additional delay is acceptable. For logic operations it would only be better to use the 1T1R design in an application where there are at least 185 times more logic operations performed than write operations. And only if speed is not important.

The simulations in this thesis were performed on schematics that do not include the parasitics, due to time constrains. Although the simulations give a good representation of the performance and energy consumption of the designs, it would be more accurate in simulations of schematics that include the parasitics. Testing different ADCs for the MAC designs was out of the scope for this thesis. However, a different existing or emerging ADC could make memristor based CIM much more favorable if the ADC has low energy consumption without adding too much delay.

The findings of this thesis can be used to choose whether to use a memristor based or SRAM based CIM design for applications that use logic and/or MAC operations. It gives an insight in the importance of choosing the right ADC for a CIM application. And it shows the areas that need improvement in memristive devices in order for it to become a better alternative to SRAM base CIM.

A

Comparison of NVMs in various works

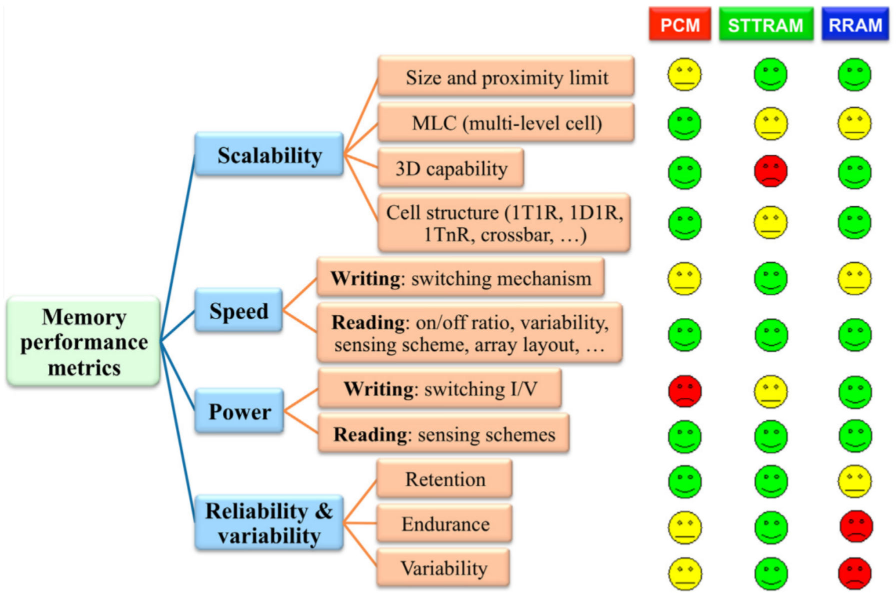
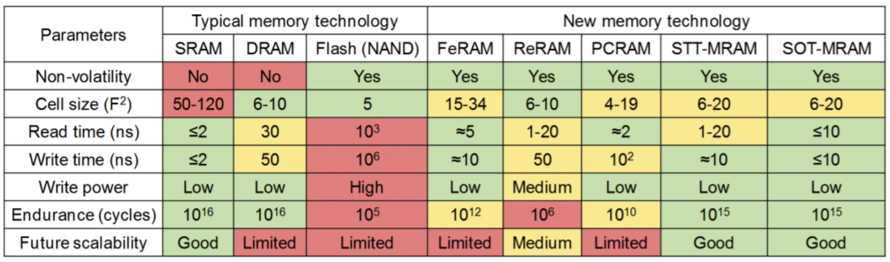

PCM
STTRAM
RRAM
Memory performance metrics
Scalability
Size and proximity limit
MLC (multi-level cell)
3D capability
Cell structure (1T1R, 1D1R, 1TnR, crossbar, …)
Speed
Writing: switching mechanism
Reading: on/off ratio, variability, sensing scheme, array layout, …
Power
Writing: switching I/V
Reading: sensing schemes
Reliability & variability
Retention
Endurance
Variability

Figure A.1: Key metrics for memory performance assessment and a qualitative comparison of STTRAM, PCM, and RRAM based on the metrics [21].



Parameters	Typical memory technology			New memory technology				
	SRAM	DRAM	Flash (NAND)	FeRAM	ReRAM	PCRAM	STT-MRAM	SOT-MRAM
Non-volatility	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Cell size (F^2)	50-120	6-10	5	15-34	6-10	4-19	6-20	6-20
Read time (ns)	≲2	30	10^3	≈5	1-20	≈2	1-20	≲10
Write time (ns)	≲2	50	10^6	≈10	50	10^2	≈10	≲10
Write power	Low	Low	High	Low	Medium	Low	Low	Low
Endurance (cycles)	10^{16}	10^{16}	10^5	10^{12}	10^6	10^{10}	10^{15}	10^{15}
Future scalability	Good	Limited	Limited	Limited	Medium	Limited	Good	Good

Figure A.2: Benchmark table of the performance of emerging memories and their comparison with typical memories [55].

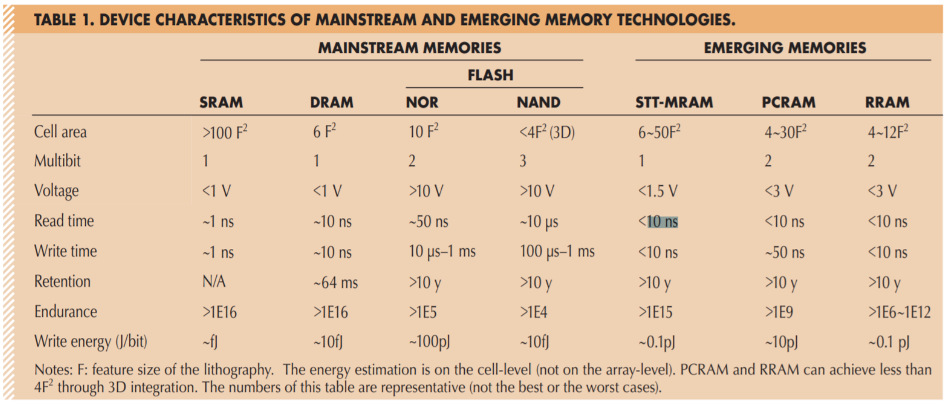
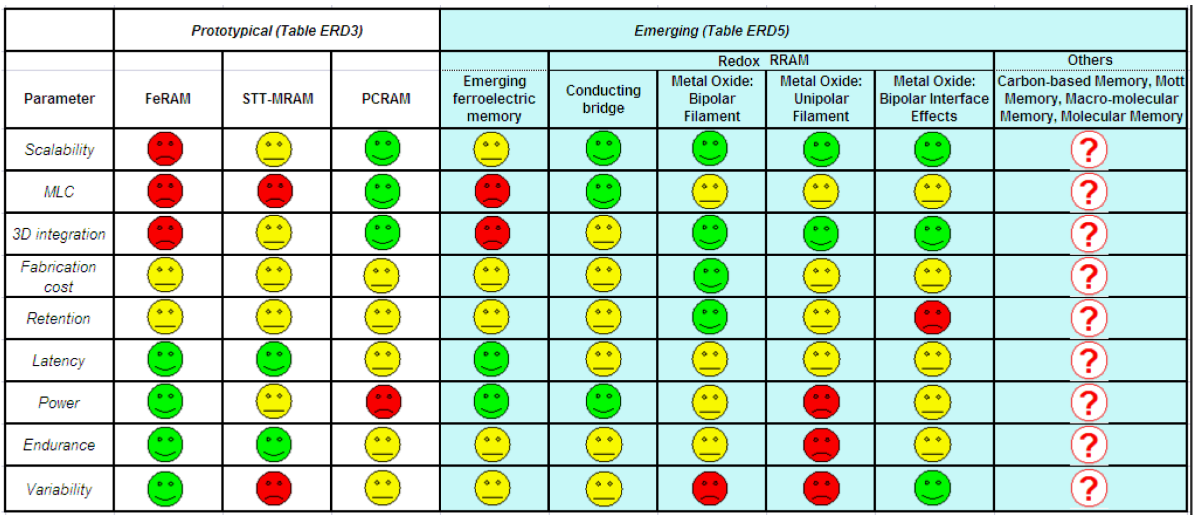


TABLE 1. DEVICE CHARACTERISTICS OF MAINSTREAM AND EMERGING MEMORY TECHNOLOGIES.

	MAINSTREAM MEMORIES				EMERGING MEMORIES		
			FLASH				
	SRAM	DRAM	NOR	NAND	STT-MRAM	PCRAM	RRAM
Cell area	>100 F^2	6 F^2	10 F^2	<4F^2 (3D)	6~50F^2	4~30F^2	4~12F^2
Multibit	1	1	2	3	1	2	2
Voltage	<1 V	<1 V	>10 V	>10 V	<1.5 V	<3 V	<3 V
Read time	~1 ns	~10 ns	~50 ns	~10 μs	<10 ns	<10 ns	<10 ns
Write time	~1 ns	~10 ns	10 μs–1 ms	100 μs–1 ms	<10 ns	~50 ns	<10 ns
Retention	N/A	~64 ms	>10 y	>10 y	>10 y	>10 y	>10 y
Endurance	>1E16	>1E16	>1E5	>1E4	>1E15	>1E9	>1E6~1E12
Write energy (J/bit)	~fJ	~10fJ	~100pJ	~10fJ	~0.1pJ	~10pJ	~0.1 pJ

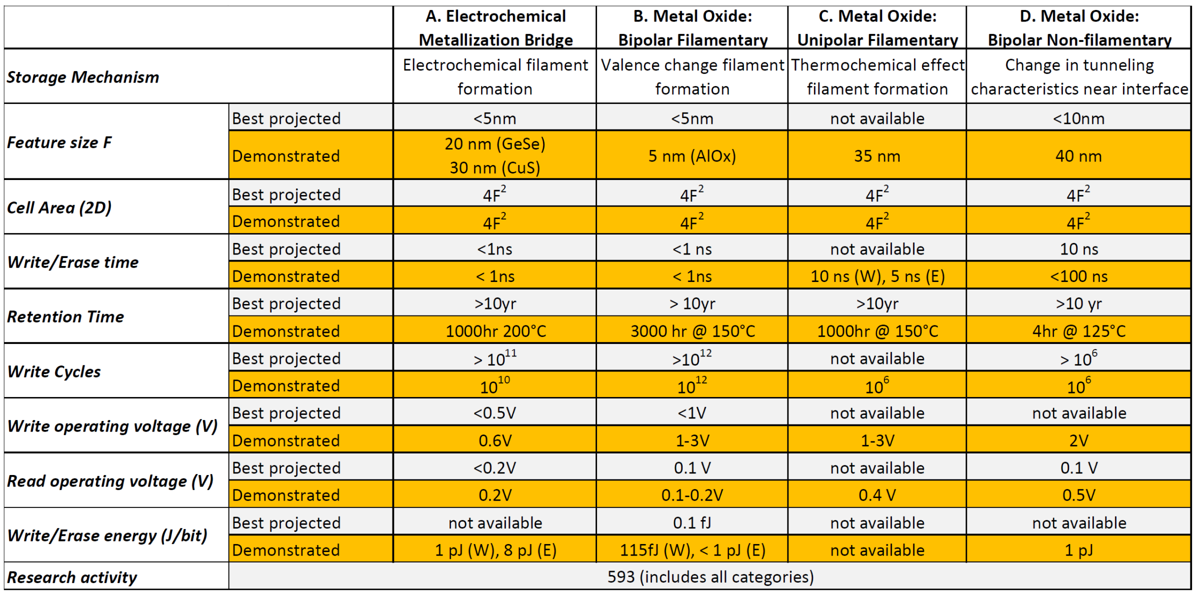
Notes: F: feature size of the lithography. The energy estimation is on the cell-level (not on the array-level). PCRAM and RRAM can achieve less than 4F^2 through 3D integration. The numbers of this table are representative (not the best or the worst cases).

Figure A.3: Device characteristics of mainstream and emerging memory technologies[56].



	Prototypical (Table ERD3)			Emerging (Table ERD5)					
					Redox RRAM				Others
Parameter	FeRAM	STT-MRAM	PCRAM	Emerging ferroelectric memory	Conducting bridge	Metal Oxide: Bipolar Filament	Metal Oxide: Unipolar Filament	Metal Oxide: Bipolar Interface Effects	Carbon-based Memory, Mott Memory, Macro-molecular Memory, Molecular Memory
Scalability	🔴	🟡	🟢	🟡	🟢	🟢	🟢	🟢	❓
MLC	🔴	🔴	🟢	🔴	🟢	🟡	🟡	🟡	❓
3D integration	🔴	🟡	🟢	🔴	🟡	🟢	🟢	🟢	❓
Fabrication cost	🟡	🟡	🟡	🟡	🟡	🟢	🟡	🟡	❓
Retention	🟡	🟡	🟡	🟡	🟡	🟢	🟡	🔴	❓
Latency	🟢	🟢	🟡	🟢	🟡	🟡	🟡	🟡	❓
Power	🟢	🟡	🔴	🟢	🟢	🟡	🔴	🟡	❓
Endurance	🟢	🟢	🟡	🟡	🟡	🟡	🔴	🟡	❓
Variability	🟢	🔴	🟡	🟡	🟡	🔴	🔴	🟢	❓

Figure A.4: .



		A. Electrochemical Metallization Bridge	**B. Metal Oxide: Bipolar Filamentary**	**C. Metal Oxide: Unipolar Filamentary**	**D. Metal Oxide: Bipolar Non-filamentary**
Storage Mechanism		Electrochemical filament formation	Valence change filament formation	Thermochemical effect filament formation	Change in tunneling characteristics near interface
Feature size F	Best projected	<5nm	<5nm	not available	<10nm
	Demonstrated	20 nm (GeSe) 30 nm (CuS)	5 nm (AlOx)	35 nm	40 nm
Cell Area (2D)	Best projected	4F^2	4F^2	4F^2	4F^2
	Demonstrated	4F^2	4F^2	4F^2	4F^2
Write/Erase time	Best projected	<1ns	<1 ns	not available	10 ns
	Demonstrated	< 1ns	< 1ns	10 ns (W), 5 ns (E)	<100 ns
Retention Time	Best projected	>10yr	> 10yr	>10yr	>10 yr
	Demonstrated	1000hr 200°C	3000 hr @ 150°C	1000hr @ 150°C	4hr @ 125°C
Write Cycles	Best projected	> 10^{11}	>10^{12}	not available	> 10^6
	Demonstrated	10^{10}	10^{12}	10^6	10^6
Write operating voltage (V)	Best projected	<0.5V	<1V	not available	not available
	Demonstrated	0.6V	1-3V	1-3V	2V
Read operating voltage (V)	Best projected	<0.2V	0.1 V	not available	0.1 V
	Demonstrated	0.2V	0.1-0.2V	0.4 V	0.5V
Write/Erase energy (J/bit)	Best projected	not available	0.1 fJ	not available	not available
	Demonstrated	1 pJ (W), 8 pJ (E)	115fJ (W), < 1 pJ (E)	not available	1 pJ
Research activity		593 (includes all categories)			

Figure A.5: ReRAM characteristics from ITRS Emerging Research Devices Chapter [20].

B

Various SRAM-based CIM macros found in literature

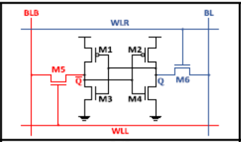
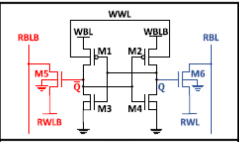
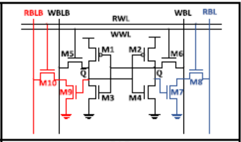
			[]	[]	[]
Reference			[6]	[7]	[8]
Process			28nm	55nm	40nm
Cell Structure			Split-WL 6T	4+2T	10T
Logic Operation	AND		v	v	v
	NAND			v	
	OR			v	v
	NOR		v	v	
	XOR			v	v
Applications			Logic/BCAM/TCAM	Logic/BCAM/TCAM	Cryptographic Processor
Features			Compact Cell Area	Compact Cell Area	w/o Read Disturb

Figure B.1: Comparison table of logic-in-memory CIM schemes [14].

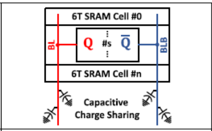
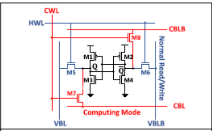
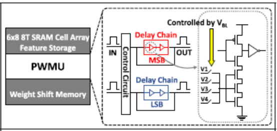
			
Reference	[9]	[10]	[11]
Process	65nm	28nm	28nm
Computing Type	NMAC	Logic IMAC + ALU NMAC	NMAC
Architecture Type	Analog	Analog + Digital	Analog + Digital
Cell Structure	6T	8T	8T
Cell Function	Weight Storage	Weight Storage	Input Storage
IN/W/OUT (# bits)	8/8/8	64-bit ALU	8/1/8
Energy Efficiency (TOPS/W)	3.125	-	119.7 (Peak) / 46.6 (AlexNet)
Features	High-Precision Input and Weight; Low Power	General Purpose Computing	Short Input Latency

Figure B.2: Comparison table of near-memory-computing (NMAC) CIM schemes [14].

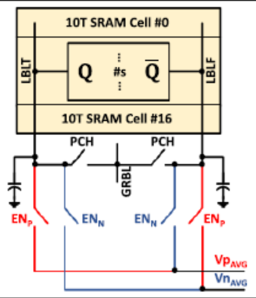
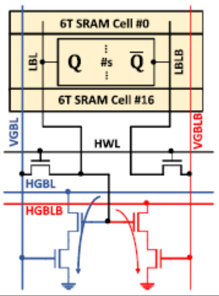
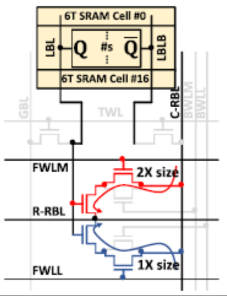
			
Reference	[19]	[20]	[21]
Process	65nm	28nm	28nm
Computing Type	IMAC	IMAC	IMAC
Architecture Type	Analog	Analog + Digital	Analog + Digital
Cell Structure	10T	6T	6T
# of MCs per LCC	16	16	16
LCC Function	Accumulate	Accumulate	Accumulate
IN/W/OUT (# bits)	7/1/7	8/8/20	8/8/20
Energy Efficiency (TOPS/W)	28.1	16.63	7.6
Features	Low Power	Large Signal Margin; High-Precision Input and Weight	Large Signal Margin; High-Precision Input and Weight

Figure B.3: Comparison table of in-array local computing (IA-LC) CIM schemes [14].

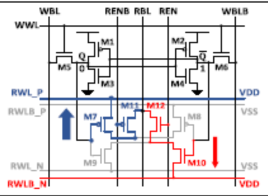
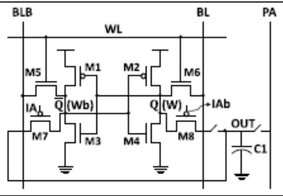
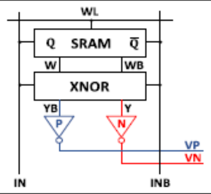
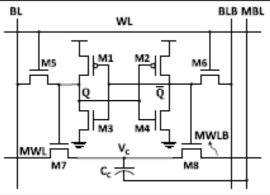
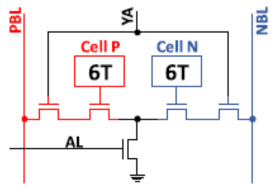
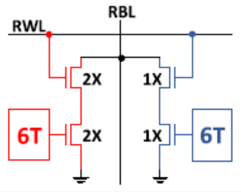
			
Reference	[22]	[23]	[24]
Process	65nm	65nm	65nm
Computing Type	IMAC	IMAC	IMAC
Architecture Type	Analog	Analog	Analog + Digital
Cell Structure	6T + 6T	8T + 1C	6T + XNOR + 2 × INV
LCC Function	XAC	XAC	XAC
IN/W/OUT (# bits)	Ternery/1/3.46	1/1/6	1/1/5
Energy Efficiency (TOPS/W)	403	658	87
Features	Low Power	High Throughput	w/o Read Disturb
			
Reference	[25]	[26]	[27]
Process	65nm	12nm	55nm
Computing Type	IMAC	IMAC	IMAC
Architecture Type	Analog	Analog	Analog + Digital
Cell Structure	8T + 1C	6T × 2 + 5T	T8T
LCC Function	XAC	XAC	MAC
IN/W/OUT (# bits)	Ternary/1/5	1/Ternary/1	4/5/7
Energy Efficiency (TOPS/W)	671.5	79.3	18.37
Features	w/o Read Disturb	Large Capacity	Large Signal Margin

Figure B.4: Comparison table of in-cell local computing (IC-LC) CIM schemes [14].

Bibliography

[1] Kapooht. (2013) Von neumann architecture. [Online]. Available: https://commons.wikimedia.org/wiki/File:Von_Neumann_Architecture.svg

[2] J. von Neumann, “First draft of a report on the edvac,” *IEEE Annals of the History of Computing*, vol. 15, no. 4, pp. 27–75, 1993.

[3] Wikipedia, “Von Neumann architecture — Wikipedia, the free encyclopedia,” https://en.wikipedia.org/wiki/Von_Neumann_architecture, 2021, [Online; accessed 27-December-2021].

[4] M. T. Bohr and I. A. Young, “Cmos scaling trends and beyond,” *IEEE Micro*, vol. 37, no. 6, pp. 20–29, 2017.

[5] Y. Sun, N. B. Agostini, S. Dong, and D. R. Kaeli, “Summarizing cpu and gpu design trends with product data,” *ArXiv*, vol. abs/1911.11313, 2019.

[6] K. Rupp. (2018) Microprocessor trend data. [Online]. Available: https://github.com/karlrupp/microprocessor-trend-data

[7] P. Siegl, R. Buchty, and M. Berekovic, “Data-centric computing frontiers: A survey on processing-in-memory,” in *MEMSYS 2016 October 3–6, 2016, Washington, DC, USA*, ser. MEMSYS ’16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/2989081.2989087

[8] A. Kerber, “Reliability of metal gate / high-k devices and its impact on cmos technology scaling,” *MRS Advances*, vol. 2, pp. 1–10, 07 2017.

[9] Z. Abbas and M. Olivieri, “Impact of technology scaling on leakage power in nano-scale bulk cmos digital standard cells,” *Microelectronics Journal*, vol. 45, no. 2, pp. 179–195, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S002626921300253X

[10] B. Hoefflinger, *The Energy Crisis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 421–427. [Online]. Available: https://doi.org/10.1007/978-3-642-23096-7_20

[11] J. Yu, “Computation-in-memory: From circuits to compilers,” Ph.D. dissertation, TU Delft, The Netherlands, May 2021. [Online]. Available: https://doi.org/10.4233/uuid:9f2a640e-0f19-4d4d-9feb-e27e3e963fcb

[12] S. Hamdioui, S. Kvatinsky, G. Cauwenberghs, L. Xie, N. Wald, S. Joshi, H. M. Elsayed, H. Corporaal, and K. Bertels, “Memristor for computing: Myth or reality?” in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 722–731.

[13] S. Lee, “Why is computational storage inevitable?” Jun 2022. [Online]. Available: https://phisonblog.com/why-is-computational-storage-inevitable-2/

[14] C.-J. Jhang, C.-X. Xue, J.-M. Hung, F.-C. Chang, and M.-F. Chang, “Challenges and trends of sram-based computing-in-memory for ai edge devices,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 5, pp. 1773–1786, 2021.

[15] A. K. Rajput and M. Pattanaik, “Implementation of boolean and arithmetic functions with 8t sram cell for in-memory computation,” in *2020 International Conference for Emerging Technology (INCET)*, 2020, pp. 1–5.

[16] S. Jeloka, N. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm configurable memory (tcam/bcam/sram) using push-rule 6t bit cell enabling logic-in-memory," *IEEE Journal of Solid-State Circuits*, vol. 51, pp. 1009–1021, 2016.

[17] Q. Dong, S. Jeloka, M. Saligane, Y. Kim, M. Kawaminami, A. Harada, S. Miyoshi, D. Blaauw, and D. Sylvester, "A 0.3v vddmin 4+2t sram for searching and in-memory computing using 55nm ddc technology," in *2017 Symposium on VLSI Circuits*, 2017, pp. C160–C161.

[18] C. Yu, T. Yoo, T. T.-H. Kim, K. C. Tshun Chuan, and B. Kim, "A 16k current-based 8t sram compute-in-memory macro with decoupled read/write and 1-5bit column adc," in *2020 IEEE Custom Integrated Circuits Conference (CICC)*, 2020, pp. 1–4.

[19] J. Kim, J. Koo, T. Kim, Y. Kim, H. Kim, S. Yoo, and J.-J. Kim, "Area-efficient and variation-tolerant in-memory bnn computing using 6t sram array," in *2019 Symposium on VLSI Circuits*, 2019, pp. C118–C119.

[20] (2015) Itrs emerging research devices chapter.

[21] A. Chen, "A review of emerging non-volatile memory (nvm) technologies and applications," *Solid-State Electronics*, vol. 125, pp. 25–38, 2016, extended papers selected from ESSDERC 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0038110116300867

[22] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electronics*, vol. 1, 06 2018.

[23] G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. A. Lastras, A. Padilla, B. Rajendran, S. Raoux, and R. S. Shenoy, "Phase change memory technology," *Journal of Vacuum Science & Technology B*, vol. 28, no. 2, pp. 223–262, 2010. [Online]. Available: https://doi.org/10.1116/1.3301579

[24] A. et al., "Truly innovative 28nm fdsoi technology for automotive microcontroller applications embedding 16mb phase change memory," presented at the IEDM conference, San-Francisco, CA, Dec. 3–5, 2018.

[25] I. Kim, S. Cho, D. Im, E. Cho, D. Kim, G. Oh, D. Ahn, S. Park, S. Nam, J. Moon, and C. Chung, "High performance pram cell scalable to sub-20nm technology with below 4f2 cell size, extendable to dram applications," in *2010 Symposium on VLSI Technology*, 2010, pp. 203–204.

[26] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, 2016.

[27] H. Y. Cheng, M. Brightsky, S. Raoux, C. F. Chen, P. Y. Du, J. Y. Wu, Y. Y. Lin, T. H. Hsu, Y. Zhu, S. Kim, C. M. Lin, A. Ray, H. L. Lung, and C. Lam, "Atomic-level engineering of phase change material for novel fast-switching and high-endurance pcm for storage class memory application," in *Technical Digest - International Electron Devices Meeting, IEDM*, 2013, pp. 30.6.1–30.6.4, cited By :15. [Online]. Available: www.scopus.com

[28] I. H. Im, S. J. Kim, and H. W. Jang, "Memristive devices for new computing paradigms," *Advanced Intelligent Systems*, vol. 2, no. 11, p. 2000105, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aisy.202000105

[29] H. D. Nguyen, J. Yu, L. Xie, M. Taouil, S. Hamdioui, and D. Fey, "Memristive devices for computing: Beyond cmos and beyond von neumann," in *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2017, pp. 1–10.

[30] A. Fantini, L. Goux, R. Degraeve, D. Wouters, N. Raghavan, G. Kar, A. Belmonte, Y.-Y. Chen, B. Govoreanu, and M. Jurczak, "Intrinsic switching variability in hfo<inf>2</inf> rram," in *2013 5th IEEE International Memory Workshop*, 2013, pp. 30–33.

[31] M. A. Lebdeh, U. Reinsalu†, H. A. D. Nguyen, S. Wong, and S. Hamdioui, "Memristive device based circuits for computation-in-memory architectures," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[32] L. Xie, H. Du Nguyen, J. Yu, A. Kaichouhi, M. Taouil, M. AlFailakawi, and S. Hamdioui, "Scouting logic: A novel memristor-based logic design for resistive computing," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2017, pp. 176–181.

[33] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie, "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2016.

[34] M. Imani, S. Gupta, Y. Kim, M. Zhou, and T. Rosing, "Digitalpim: Digital-based processing in-memory for big data acceleration," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 429–434. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/3299874.3319483

[35] S. Jain, A. Ranjan, K. Roy, and A. Raghunathan, "Computing in memory with spin-transfer torque magnetic ram," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 26, no. 3, p. 470–483, mar 2018. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1109/TVLSI.2017.2776954

[36] F. Parveen, Z. He, S. Angizi, and D. Fan, "Hielm: Highly flexible in-memory computing using stt mram," *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 361–366, 2018.

[37] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[38] M. Hu, H. H. Li, Q. Wu, and G. S. Rose, "Hardware realization of bsb recall function using memristor crossbar arrays," *DAC Design Automation Conference 2012*, pp. 498–503, 2012.

[39] A. Velasquez and S. K. Jha, "Parallel boolean matrix multiplication in linear time using rectifying memristors," in *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE Press, 2016, p. 1874–1877. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1109/ISCAS.2016.7538937

[40] ——, "Computation of boolean matrix chain products in 3d reram," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[41] J. Yu, H. A. D. Nguyen, L. Xie, M. Taouil, and S. Hamdioui, "Memristive devices for computation-in-memory," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 1646–1651.

[42] M.-F. Chang, C.-F. Chen, T.-H. Chang, C.-C. Shuai, Y.-Y. Wang, and H. Yamauchi, "17.3 a 28nm 256kb 6t-sram with 280mv improvement in v<inf>min</inf> using a dual-split-control assist scheme," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1–3.

[43] P.-H. Chen, Y.-T. Su, W.-C. Huang, and C.-W. Wu, "Improving high resistance state in one-transistor-one-resistor (1t1r) structure resistance random access memory with a body-biased method," *IEEE Transactions on Electron Devices*, vol. 70, no. 3, pp. 1014–1018, 2023.

[44] Y. Okuda, J. Kawakita, T. Taniuchi, H. Shima, A. Shimizu, Y. Naitoh, K. Kinoshita, H. Akinaga, and S. Shin, "Operando observation of analog resistance change in a buried metal/oxide interface by a laser-excited photoemission electron microscope," *Japanese Journal of Applied Physics*, vol. 61, no. SM, p. SM1001, Jun. 2022. [Online]. Available: http://dx.doi.org/10.35848/1347-4065/ac5721

[45] H. Aziza, *Oxide-based Resistive RAM Analog Synaptic Behavior Assessment for Neuromemristive systems*. IntechOpen, Sep. 2023. [Online]. Available: http://dx.doi.org/10.5772/intechopen.1002782

[46] W. Choi, W. Ji, S. Heo, D. Lee, K. Noh, C. Lee, J. Woo, S. Kim, and H. Hwang, "Exploiting read current noise of tiox resistive memory by controlling forming conditions for probabilistic neural network hardware," *IEEE Electron Device Letters*, vol. 43, no. 9, pp. 1571–1574, 2022.

[47] W. Banerjee, “Challenges and applications of emerging nonvolatile memory devices,” *Electronics*, vol. 9, no. 6, p. 1029, Jun. 2020. [Online]. Available: http://dx.doi.org/10.3390/electronics9061029

[48] A. Singh, M. Fieback, R. Bishnoi, F. Bradarić, A. Gebregiorgis, R. V. Joshi, and S. Hamdioui, “Accelerating rram testing with a low-cost computation-in-memory based dft,” in *2022 IEEE International Test Conference (ITC)*, 2022, pp. 400–409.

[49] F. Cüppers, S. Menzel, C. Bengel, A. Hardtdegen, M. von Witzleben, U. Böttger, R. Waser, and S. Hoffmann-Eifert, “Exploiting the switching dynamics of HfO2-based ReRAM devices for reliable analog memristive behavior,” *APL Materials*, vol. 7, no. 9, p. 091105, 09 2019. [Online]. Available: https://doi.org/10.1063/1.5108654

[50] C. Bengel, A. Siemon, F. Cüppers, S. Hoffmann-Eifert, A. Hardtdegen, M. von Witzleben, L. Hellmich, R. Waser, and S. Menzel, “Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using spice level compact models,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4618–4630, 2020.

[51] S. Nalam and B. H. Calhoun, “Asymmetric sizing in a 45nm 5t sram to improve read stability over 6t,” in *2009 IEEE Custom Integrated Circuits Conference*, 2009, pp. 709–712.

[52] C. Bengel, D. K. Zhang, R. Waser, and S. Menzel, “Jart vcm v1 verilog-a compact model user guide,” Sep 2022. [Online]. Available: https://www.emrl.de/JART.html

[53] A. Santos, J. D. Ferreira, O. Mutlu, and G. Falcao, “Redbit: An end-to-end flexible framework for evaluating the accuracy of quantized cnns,” 2023. [Online]. Available: https://arxiv.org/abs/2301.06193

[54] A. R. Jaiswal, I. Chakraborty, A. Agrawal, and K. Roy, “8t sram cell as a multibit dot-product engine for beyond von neumann computing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, pp. 2556–2567, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:3535689

[55] E. Liu, “Materials and designs of magnetic tunnel junctions with perpendicular magnetic anisotropy for high-density memory applications,” Ph.D. dissertation, KU Leuven, 11 2018.

[56] S. Yu and P.-Y. Chen, “Emerging memory technologies: Recent trends and prospects,” *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, 2016.