Perspectives on Emerging Computation-in-Memory Paradigms

Shubham Rai[1], Mengyun Liu[2], Anteneh Gebregiorgis[3], Debjyoti Bhattacharjee[4], Krishnendu Chakrabarty[2], Said Hamdioui[3], Anupam Chattopadhyay[5], Jens Trommer[6], Akash Kumar[1]
[1]Chair for Processor Design, Technische Universität Dresden, Germany,
[2]Department of Electrical and Computer Engineering, Duke University, USA,
[3]Department of Quantum and Computer Engineering, Delft University of Technology, The Netherlands
[4]imec, Leuven, Belgium, [5]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[6]Namlab gGmbH, Dresden, Germany

***Abstract*—The traditional Von-Neumann architecture is reaching its limits and finding it difficult to cope up with the ever-increasing demands of modern workloads like artificial intelligence. This demand has fueled the search of technologies that can mimic human brain to efficiently combine both memory and computation within a single device. In this work, we present the state-of-the-art research in the domain of computation-in-memory. In particular, we take a look at memristors and its widespread application in neuromorphic computation. We introduce ReRAMs in terms of their novel computing paradigms and present ReRAM-specific design flows. We address the various circuit opportunities and challenges related to reliability and fault tolerance associated with them. Another high-potential candidate to leverage memory and computation from a single device is Ferroelectric Field-effect Transistor (FeFET). Here we present a co-integration of such FeFETs with another emerging nanotechnology concept, called Reconfigurable Field Effect Transistor (RFET) and discuss the impact of the higher amount of states provided by this combination.**

I. Introduction

Technology scaling driven by Moore's law has enabled the semiconductor industry to build transistors that are smaller, faster, and consume less power for successive generations [1]. Unfortunately, the continued scaling has now become extremely costly due to various challenges such as physical, material, thermal and technology challenges [1, 2]. The explosion of data intensive applications and their unprecedented demand for energy efficiency, from data centers to energy-constrained edge and wearable technologies, further exacerbate these challenges [3]. Hence, innovating technologies and architectures that can not only keep up with these demands, but also offer superior performance with limited energy budget are of paramount importance [4, 5]. In this regard, the costly development and deployment of accelerated systems has been amortized by domain-specific architectures to provide dedicated computing architectures for complex algorithms and applications [6]. This has led to a proliferation of accelerators in different application domains [7, 8]. Furthermore, the post-CMOS era has led to the exploration of a new computing paradigm with the help emerging technologies such as non-volatile devices to provide high level of energy-efficiency that are difficult to attain with the conventional CMOS-based architectures [9, 10, 11].

Similarly, the breakthrough in Artificial Intelligence (AI) which led to a booming increase in AI-based applications and services, also poses a significant challenge in terms of energy consumption, memory storage and data transfer bandwidth [3, 12]. As a result, the existing AI processing architectures based on the conventional von-Neumann architecture (see Figure 1a), such as CPU, GPU and TPU, spend excessive time and energy in moving massive amount of data between the memory and data paths [13]. These challenges are indeed imperiling the widespread deployment of AI on resource constrained computing platforms such as edge devices. Therefore, a computing paradigm shift is paramount to unlock the full potential of AI. In this regard, memristor-based computation-in-memory (CIM) has the potential to break the aforementioned challenge by replacing the traditional computing architecture by CIM architectures [14, 15] (as shown in Figure 1b).

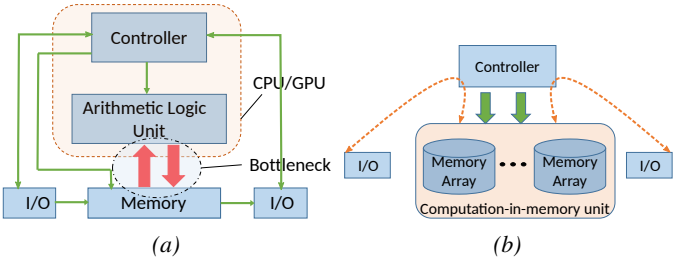


Fig. 1: (a) Typical Von-Neumann Architecture showing the memory-processor communication as the bottleneck (b) Computation-in-memory (CIM)

Memristor-based CIM architecture uses non-volatile devices to store the data while exploiting their inherent capability to perform computation on the stored data and, hence, circumventing the costly data movement of von-Neumann based systems [16]. Several recent works on CIM have demonstrated that multiply-and-accumulate (MAC) operations, which are the fundamental operations in AI applications such as Deep Neural Networks (DNNs), can be successfully implemented on CIM crossbar arrays consisting of memristor devices and boost execution time significantly [17, 18]. In addition, memristor-based CIM architecture has various advantages such as zero leakage, non-volatility and density. CIM architecture has two vital components, namely crossbar array (where the data resides and computation takes place), and peripheral circuits such as Digital-to-Analog Converters (DACs) and Analog-to-Digital Converters (ADCs) for data format conversion [19].

This paper provides a broad overview of emerging computation-in-memory paradigm highlighting state-of-the-art research in the CIM domain. Particularly, the paper investigates memristors and their widespread application in neuromorphic

Architecture	Data movement outside memory core	Computation requirements		Available bandwidth	Memory design efforts			Scalability
		Data Align-ment	Complex function		Cells & Array	Periphery	Controller	
CIM-A	No	Yes	High latency	Max	High	Low/medium	High	Low
CIM-P	No	Yes	High cost	High-Max	Low/medium	High	Medium	Medium
COM-N	Yes	NR	Low cost	High	Low	Low	Low	Medium
COM-F	Yes	NR	Low cost	Low	Low	Low	Low	High

NR: Not required.

TABLE I: Comparison of CIM-A, CIM-P, COM-N and COM-F architectures using different metrics [16]

computing. In this regard, ReRAMs are introduced in terms of their potential for novel computing paradigms and ReRAM-specific design flows are presented. Moreover, we address various circuit opportunities and challenges related to reliability and fault tolerance associated with ReRAM. We also look at an emerging nanotechnology paradigm involving co-integration of ferroelectric FET with reconfigurable field effect transistor (RFET), which has the potential to leverage memory and computation providing multiple states from a single device.

The remainder of this paper is organized as follows: Section II presents CIM architecture classification, background of ReRAM-based CIM architecture and its potentials. Section III describes testing and fault tolerance of CIM designs. EDA for ReRAM-based CIM is covered in IV. Emerging circuit design topologies for CIM using non-volatile reconfigurable FETs are presented in Section V. Finally, the paper is concluded in Section VI.

II. CIM ARCHITECTURE CLASSIFICATION AND ITS POTENTIALS

A. CIM classification and comparison

A memory core consists of one or more cell arrays (used for storage) and peripheral circuits (used to access the cells). Traditionally, the computation takes place in the computational cores. However, with the emergence of CIM, the computation can be done within the memory array as well as the periphery circuits. Hence, computer architectures can be classified into different categories based on where the computation takes place as shown in Figure 2 [16]; if the result is produced within the memory core, the computer architecture is referred to as Computation-In-Memory (CIM); otherwise, the architecture is referred to as Computation-Out-Memory (COM).

1) **CIM:** Computations in CIM takes place either within the memory array or peripheral circuit (labeled as (1) and (2) in Figure 2), which are referred to as Computation-In-Memory Array (CIM-A) and Computation-In-Memory Peripheral (CIM-P), respectively. Both sub-classes impact the design of the memory, either the redesign of cells to support computing within the crossbar, or special circuits in the peripheral circuit such as customized sense amplifiers [20, 21], or the usage of ADCs [22].
2) **COM:** Computation in COM takes place either in the extra logic circuits outside the memory core but inside the memory SiP (as it is the case for commercialized High Bandwidth Memories) or in the traditional computational cores such as CPUs and GPUs (labeled as (3) and (4) in Figure 2), which are referred to as Computation-Out-Memory Near (COM-N) and Computation-Out-Memory Far (COM-F), respectively.

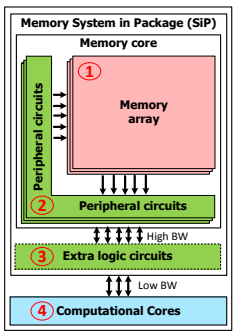


Fig. 2: Computer architecture classification

The CIM and COM classes and their sub-classes can be qualitatively compared based on different criteria. Table I presents the comparison results of CIM-A, CIM-P, COM-N and COM-F architectures, while considering data movement to and from memory core, computational requirement, available bandwidth, memory design efforts and scalability potential [16]. From the table we can observe that both CIM-A and CIM-P architectures does not move data outside the memory core for computation purposes and have relatively higher bandwidth when compared to COM-N and COM-F architectures. However, CIM-A and CIM-P architectures are less scalable, require data alignment and requires more memory design efforts than their COM-N and COM-F counterparts.

B. Basics and key components of CIM architecture

The memory array for CIM architecture can be implemented using different non-volatile memory technologies such as Phase Changing Memory (PCM), Resistive Random Access memory (ReRAM) and magnetic memories (MRAM) as well as conventional volatile memory technologies such as SRAM and DRAM [23]. Irrespective of the memory technology used for CIM architectures, the basic concept of CIM and its core functional units are similar and independent of the adopted memory technology. In this paper we will use ReRAM technology to elaborate on CIM architectures and their potential.

1) Evolution of ReRAM: In 1971, Chua proposed the concept of the memristor (short for memory resistor) as the fourth fundamental element besides the resistor, capacitor and inductor [24]. The physical model of a two-terminal memristor was first proposed by HP Lab in 2008 [25], and the first memristive device was implemented by switching the doping front within a thin TiO_2/TiO_{2-x} filament. Since then, different metal-oxide materials have been studied to realize the programmable resistance of memristors, e.g. TiO_x [26], WO_x [27] and HfO_x [28]. The characteristic of programmable resistance

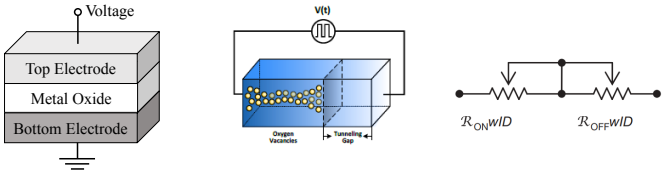


Fig. 3: Diagram with a simplified equivalent circuit of a ReRAM device [25].

leads to two important applications of memristors, i.e., a nonvolatile memory cell and a synapse in a neural network [29], which enable memristor-based computing systems.

The ReRAM was invented as an emerging non-volatile memory device based on the metal-insulator-metal (MIM) structure of the memristor [25]. The conceptual structure of a ReRAM device is shown in Figure 3, and it contains two regions, i.e., the doped region and the undoped region. This two-port MIM structure is equivalent to two serially-connected resistors. Therefore, by applying positive and negative voltages, we can change the length of metal filament inside the structure and further change the resistance value of the tunable insulator. To reduce the effect of random variation, the resistance value is typically quantized into N levels. Noise margin and guard bands are added to each level [30].

2) CIM architecture units: As shown in Figure 4(b), a CIM core has two main architectural units: (1) Memory array commonly known as crossbar array unit and periphery unit. The crossbar array stores the data, and can perform any logic or arithmetic operation. Similarly, the periphery unit converts input/output data formats between analog and digital. Moreover, the periphery unit can also be used to perform basic logical and arithmetic operations.

Crossbar array: Different neuromorphic applications use primitive computational units such as multiply and accumulate (MAC) extensively in order to perform matrix-matrix multiplication (MMM) with large operand sizes [31, 32]. Such primitive units can be easily mapped into a memristive-based crossbar array and perform their operation e.g., MMM in the crossbar unit of a CIM. Figure 4(a), shows a subset of MMM operation *i.e.*, vector-matrix multiplication (VMM) using CIM crossbar array. From Figure 4(a) it can be observed that the VMM is performed by applying a voltage vector $V = V_j$ (where $j \in \{1, m\}$) to a memristor-crossbar matrix of conductance values $G = G_{ij}$ (where $i \in \{1, n\}$, $j \in \{1, m\}$). At any

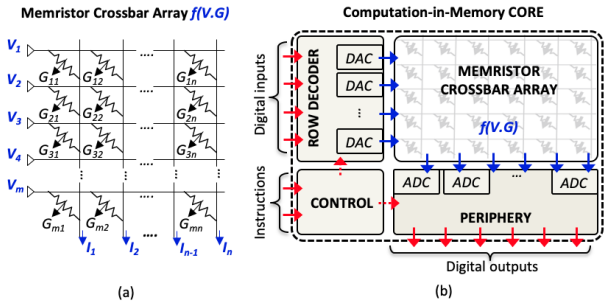


Fig. 4: Basic CIM architecture (a) ReRAM based crossbar operation demo (b) CIM core architecture i.e., Periphery + crossbar array

instance, each column performs a vector-vector multiplication (VVM) or a MAC operation, with the output current vector I, in which each element is $I_i = \Sigma V_j \cdot G_{ij}$. Note that all n MAC operations are performed with O(1) time complexity.

Periphery: A CIM core needs some major modifications to accommodate analog-based computing, as shown in Figure 4(b). The circuit blocks comprising the periphery that supports the bitcell array need to be modified to support CIM operations. For example, the following is needed to perform VMM operation in CIM: 1) Row-decoder becomes complex as it involves enabling several rows in parallel. Also, *1-bit* row or word-line drivers are now replaced by digital-to-analog converters (DACs) that convert multi-bit VMM operands into an array of analog voltages. 2) Column periphery circuits performing read operations need to be replaced by analog-to-digital converters (ADCs). 3) Control block needs to deal with complex instructions such as handling intricacies of multi-operand VMM operations.

C. Early prototypes and results

Neuromorphic hardware implementations for Artificial Intelligence (AI) applications are massively employing CIM architectures to exploit its potential over the conventional von-Neumann architecture, and accelerate AI deployment on resource constrained devices [16]. Although CIM has huge potential over the conventional architecture, it is not widely adopted yet due to limitations in EDA tools and lack of proper test and verification flow. However, there are quite a few prototypes demonstrating CIM potential for different application domains.

Data-intensive Architecture (DIVA) is one of the earliest CIM architecture prototype developed at USC Information Sciences Institute [33, 34]. The architecture consists of a host processor, host memory interface and multiple CIM blocks as co-processors. Similarly, ReVAMP a ReRAM-based VLIW architecture was proposed by [35] to exploit parallelism using majority logic.

D. Potential CIM applications

CIM architectures can be applied in different application segments which have extreme demand in terms of storage, energy and computation efficiency. This subsection presents some of the application domains in which CIM can be applied [36].

1) Neuromorphic computing: Neuromoprhic computing is one of the application domains which can significantly benefit from CIM architecture. The main reason for this is the fact that the main operation employed by neuromorphic systems involves intensive Matrix-Matrix Multiplication (MMM) or Vector-Matrix Multiplication (VMM). Since both MMM and VMM kernels can be easily accelerated using CIM architecture, neuromorphic computing can achieve substantial improvement in energy-efficiency and alleviate data movement problems by employing CIM.

2) Sparse coding: Sparse coding of information is a powerful mean to perform feature extraction on high dimensional data and it is of vital importance for wide range of application segments such as object recognition, computer vision, signal processing and etc. Sparse coding can be used to implement

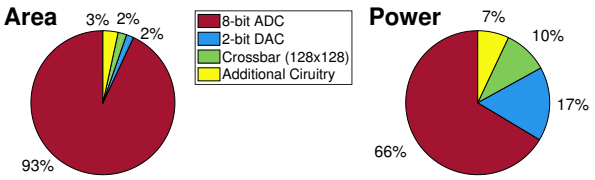


Fig. 5: Area and Power share of CIM design blocks [32].

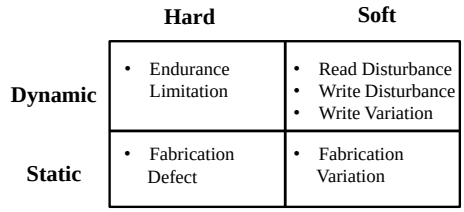


Fig. 6: Classification of different fault types in ReRAM cells.

energy-efficient bio-inspired neuromorphic applications as well. Since sparse coding mainly rely on bulky matrix-vector multiplication operation, it can directly benefit from CIM to accelerate the matrix-vector multiplication operation in an efficient manner.

3) Threshold logic: Threshold logic is a basic operation that uses a threshold gate which takes n inputs (x_1,x_2, . . ., x_n) and generates single output y. A threshold logic has a threshold θ and each input x_i is associated with a weight w_i. Since weighted sum operation is the core operation involved in threshold logic, it can be easily accelerated using CIM.

E. CIM challenges

In CIM architectures, the operations are performed in an analog manner as shown in Figure 4, and the result is converted to digital signal using Analog-to-Digital Converter (ADC) at the periphery of the CIM architecture. However, the conversion performed by ADC is very critical and challenging due to 1) Analog signals have low noise margin and hence, can lead to erroneous output; 2) Analog computation heavily relies on memristor and CMOS device strengths along the column, therefore these variations induce variation in output current; 3) Quantization error in ADC increases as we increase the number of levels or reduce the resolution. In addition, area/power increases drastically as we do so and speed reduces along with accuracy. For instance, substantiating the importance of ADC design in CIM-based implementation of machine learning algorithms such as CNN and DNN, Figure. 5 shows that the ADC alone typically dominates CIM die area (>90%) and power consumption (>65%). Thus, efficient ADC design is imperative to efficiently deploy CIM architecture in different resource constrained systems.

III. Testing of CIM Systems

Emerging computation-in-memory systems provide attractive hardware solutions for neuromorphic computing by reducing time complexity and improving energy efficiency dramatically. However, these emerging devices are vulnerable to faults. The computational accuracy of practical circuits is limited by device faults [37]. For example, previous studies have shown that the classification accuracy for a typical ImageNet testbench with random stuck-at-0 faults is reduced by 35% when the yield drops to 80% [38]; clearly this accuracy drop is not acceptable. If the yield is lower than 80%, the classification accuracy is even lower. Besides the fabrication faults, faults may occur during data processing. The accuracy is reduced further if we take into account process-induced variations, aging, and other sources of errors. In order to recover to an acceptable level of accuracy in CIM applications, fault detection and fault tolerance are necessary.

A. Fault Models

The ReRAM crossbar structure is similar to traditional RAM structures, thus we can reuse most of the fault models used for testing RAMs. These fault models include the Stuck-At-Fault (SAF), Transition Fault (TF), and Address Decoder Fault (ADF) [39]. Besides these memory-based faults, several unique fault models based on the physical mechanism of ReRAM cells have also been introduced, e.g., the read disturbance fault. The read disturbance fault [39, 40] may appear when a read current is applied during read operations, which may bias the state of the cell.

Faults in a single ReRAM cell can be classified into soft faults and hard faults [41]. For soft faults, the actual resistance of the ReRAM cell deviates from the targeted value, but the resistance can still be tuned. Soft faults are caused by variations associated with both fabrication techniques and write/read operations. For hard faults, the resistance of an ReRAM cell is stuck at a fixed state which cannot be tuned anymore, e.g., the stuck-at-0 (SA0) and stuck-at-1 (SA1) faults. Although the conductance of an ReRAM cell can take any value between *High Resistive State* (HRS) and *Low Resistive State* (LRS), the ReRAM cells with stuck-at faults tend to get stuck at the highest and lowest value, i.e., SA0 or SA1 [42, 43]. The stuck-at faults are caused by fabrication defects [39] and limited endurance [44]. For example, a broken word-line in a ReRAM crossbar array leads to the SA1 behavior.

Faults in a single ReRAM cell can also be classified into dynamic faults and static faults, as shown in Fig. 6. Static faults are generated during the process of fabrication, which includes fabrication defects that cause hard faults and variations that lead to soft faults. For example, a fresh ReRAM cell needs to undergo a forming process to switch from its initial resistance state to the normal LRS. During the forming process, an unstable external voltage may lead to over-forming defects. Dynamic faults are typically generated during read and write operations in ReRAM cells which passed fabrication tests, e.g., the write variation, write disturbance and read disturbance faults. For example, in write operations, positive and negative voltage pulses are applied to ReRAM cells to change their resistance values. Because of the stochastic characteristic of this process, it is impossible to recreate an exactly same filament in a ReRAM device every time [41]. As a result, write variation always exists while programming a ReRAM cell and we end up writing to the cell from a certain conductance distribution, instead of a specific conductance value. The impact of various process variations and manufacturing defects like oxide-pinholes on ReRAM and associated defect-to-fault mapping have been explored in [45].

B. Testing Methods

A typical test method to detect fault is to program all the ReRAM cells to a target conductance state, and then measure the conductance variations. Based on this idea, a March test algorithm, named as March C^*, was proposed for ReRAM fault detection in [39]: $\{\Uparrow (r0, w1); \Uparrow (r1, r1, w0); \Downarrow (r0, w1); \Downarrow (r1, w0); \Uparrow (r0)\}$. By applying the test pattern in this designed order, each ReRAM cell provides a six-bit signature from the six read operations in the algorithm. These signatures can detect stuck-at faults, transition faults, coupling faults, address decoder faults, and read-1 disturbance faults. However, the March C^* test checks each ReRAM cell sequentially. Despite achieving very high fault coverage, it requires a long test time.

In order to accelerate the test process, a sneak-path technique is proposed in [46] to increase test parallelism by testing a group of adjacent ReRAM cells simultaneously. This method utilized the inherent sneak-path mechanism in the ReRAM crossbar array. Because of the resistive and bidirectional characteristics of ReRAM cells, the current through both the targeted ReRAM cell and adjacent unintended paths [46]. In this way, when tests are applied to one ReRAM cell, the defect information of the adjacent ReRAM cells in the region of detection can be detected simultaneously by measuring the output currents. In order to test the entire crossbar array, tests are applied to a subset of ReRAM cells one by one. However, the test time required by the sneak-path technique increases linearly with the array size, remaining unacceptably high for on-line test.

C. On-Line Testing

Studies have shown that even among ReRAM chips that pass manufacturing test, many faults appear in the field during read and write operations [47, 48] because of the limited write endurance of ReRAM cells. Various efficient approaches have been proposed for online fault detection.

To reduce on-line testing time, a voltage-comparison method was proposed in [38] to detect stuck-at faults by comparing the real output voltage with the expected voltage. This method includes four key steps. First, the conductance values of ReRAM crossbars are read and stored off-chip. Second, a fixed increment or decrement is written to all ReRAM cells for detecting SA0 and SA1 faults, respectively. Third, test voltages are applied to a group of rows at a time, and output voltages are obtained at all column output ports concurrently. Fourth, output voltages are compared with the corresponding reference voltages, which are calculated with the assumption that all the ReRAM cells can be tuned successfully. A discrepancy between an output voltage and the corresponding reference voltage denotes that at least one of the ReRAM cells in the selected rows and columns has a stuck-at fault. By carrying out this fault-detection method bidirectionally, faults can be located.

To detect deviation faults and correct errors online, a signature-based method called X-ABFT method was proposed in [49, 50]. The basic idea of the X-ABFT method is to encode matrices with checksums (the sum of each row or column) and compute using both original and encoded data. Thus, faults can be detected when discrepancies exist between the checksums and the sum of the cells. Moreover, this method periodically applies test-input vectors to extract signatures, and uses signatures for fault localization and correction.

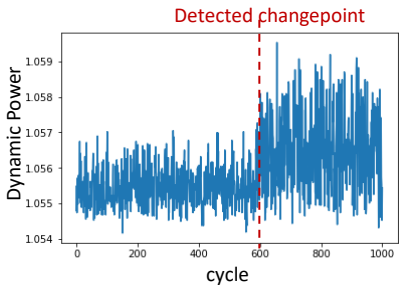


Fig. 7: A changepoint is detected when faults are inserted in a ReRAM crossbar after cycle 600 *[52].*

However, these methods are all based on a Pause-and-Test mechanism. The Pause-and-Test mechanism periodically interrupts the regular computation and applies test-specific operations to detect the existence of faults. The time-consuming fault detection process is carried out every fixed cycles, even if there is no faults in ReRAM crossbars. Error-correction codes (ECC) can also be used in ReRAM memory [51], when the bit error rate (BER) is small (e.g., $< 10^{-5}$). However, due to the limited endurance, more devices will be worn out over time and eventually the number of hard faults will exceed the ECCs correction capability.

An efficient online fault-detection method for ReRAM-based computing systems was proposed in [52]. This method exploit the fact that ReRAM faults affect the dynamic power consumption of ReRAM crossbars; therefore, it monitors the dynamic power consumption of each ReRAM crossbar and determines the occurrence of faults when a changepoint is detected in the monitored power-consumption time series, as shown in Fig. 7. Moreover, when faults are detected, this method estimates the percentage of faulty cells in a faulty ReRAM crossbar by training a machine learning-based estimation model. To train the estimation model, the statistics of the power-consumption profile as independent variables, and the percentage of faulty cells as dependent variables. In this way, the computationally expensive fault localization and error-recovery steps are carried out only when a high fault rate is estimated.

IV. Enabling EDA for Emerging Memory Devices

To allow computing of arbitrary Boolean functions using CIM platforms, development of design automation flows (EDA) is of critical importance. Typically for ReRAMs, EDA flows consist of multiple phases. Technology-independent logic synthesis is the first step, where the input Boolean function is restructured without any specific technology constraints, which is generally followed by a technology-dependent optimization phase, where technology-specific hints are used for optimization of the data structure obtained from the first step. The final step is technology mapping, which takes the optimized function representation to implement it using technology-specific constraints. In this section, a short overview of the EDA flow (Fig. 8) with various mapping objectives is presented.

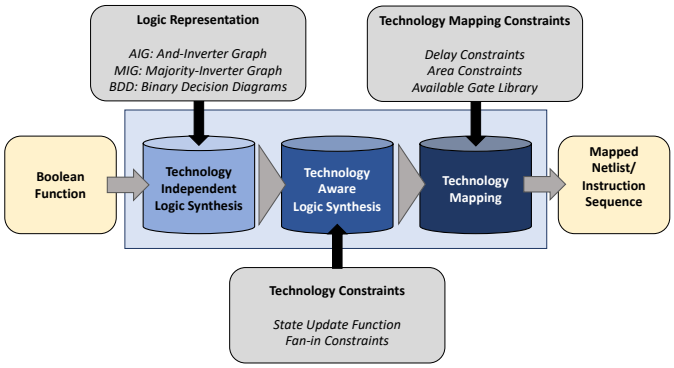


Fig. 8: Representative EDA flow for various technologies.

A. Boolean Operator Set

ReRAMs permit stateful logic, where the logical states are represented as resistive state of the devices and at the same time, are capable of computation. Multiple functionally complete logic families have been successfully demonstrated using ReRAM devices. In the following, three prominent logic families are presented.

Material Implication Logic: Consider two ReRAM devices p and q with internal states S_p and S_q respectively. By applying voltages to the terminal, material implication can be computed, with the next state (NS) of device p set to the result of computation.

$NS_p = S_p \rightarrow S_q$

Majority Logic: In this approach, the wordline voltage (V_{wl}) and bitline voltages (V_{bl}) act as logic inputs, while the internal resistive state (S_x) of the device x acts a third input. The next state of device (NS_x) in this case is a function of three inputs as shown below in the following equation.

$NS_x = M_3(S_x, V_{wl}, \overline{V_{bl}})$

Memristor-Aided loGIC (MAGIC): MAGIC allows computation-in-memory operation by using the internal resistive state of single or multiple ReRAM devices as input. The exact number of inputs (k) depends on the specific device used for computation. The result of computation is written to a new device (r). The internal resistive state of the input devices remain unchanged. Using MAGIC operations, multi-input NOR and NOT can be realized.

B. Logic Synthesis

Logic synthesis [53] converts the input Boolean function into an intermediate representation, such as And-Inverter Graph (AIG) [54], Majority-Inverter Graph (MIG) [55], Exclusive Sum of Product (ESOP) [56], Binary Decision Diagrams (BDD) [57]. This is optimized according to some criteria, such as reduction in the number of vertices or number of logic levels, and fed onward to the technology mapping phase. A variety of tools have been developed both by industrial as well as in academic efforts, for logic synthesis and verification [54, 58, 56, 54, 59, 60]. For ReRAM logic synthesis flows, these representations are further optimized by using technology specific constraints [61, 62].

C. Technology Mapping

The technology mapping phase determines a sequence of *instructions*, for application as input to the wordlines and bitlines of the ReRAM devices or a crossbar array to compute the target function. The quality of the technology mapping phase is determined by the *delay* and *area* of the mapping. The *delay* of the obtained mapping is equal to the number of steps that the mapping contains. The number of devices used during technology mapping determines the *area* of the solution.

For devices realizing material implication, a preliminary approach was proposed [63] that considers two sets of devices—one for storage of inputs and the other for computation. The lower bound on working memristors was thereafter reduced to two [64]. Implementation of various Boolean operators (e.g., NOT, NAND, XOR) and functions through Imply Sequence Diagram was suggested in [65]. A multi-stage mapping technique for Boolean functions, starting from an Or-Inverter Graph representation and heuristics for device count and delay reduction was proposed in [66].

For ReRAM devices realizing majority, an MIG can be mapped with optimal delay (equal to the number of levels in the MIG + 1) when the number of available devices is not constrained [67]. A compiler for minimizing the number of devices for an architecture performing majority operations sequentially was developed in [68]. A delay-constrained mapping heuristic was proposed for mapping onto a crossbar array, while considering the possibility of multiple majority operations per row [69]. A lower bound on the size of crossbar array (3 wordlines and 2 bitlines) required to map a Boolean function in *Exclusive Sum-of-Product* representation was introduced [69]. Using this bound as a building block, an LUT-based, area-constrained mapping approach was proposed.

For crossbars realizing MAGIC operations, optimal and heuristic solutions to map Boolean functions from NOR/NOT netlist onto single row was proposed, with the goal of optimizing throughput by *Single Instruction Multiple Data* (SIMD) like operations [70]. For delay-optimal technology mapping on crossbar array, a *Satisfiability modulo theories* (SMT) based approach was proposed [71]. This was improved in [72], where an LUT-based technology mapping approach was proposed to minimize delay and maximize parallel execution of LUTs [72]. Recently, an area-constrained technology mapping flow was proposed for crossbar memories, which uses $A*$ search of optimal data movement within the array [73]. The proposed approach outperforms existing technology mapping flows [74, 75, 76, 77], when compared on the basis of the area-delay product metric.

V. Emerging Circuit Design Topologies

Besides ReRAM [78, 62], PCM [79] and MRAM [80], ferroelectric field effect transistor (FeFET) based memories [81] have gained a high amount of traction for CIM appliations within the last years [82, 83, 84, 85]. FeFETs conveniently merge the two areas of computation and data storage, by placing a non-volatile storage element directly into the gate stack of a classical MOSFET.

In this section, we take the approach one step further and discuss about the possible co-integration of FeFETs with

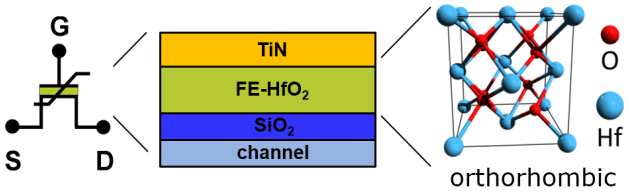


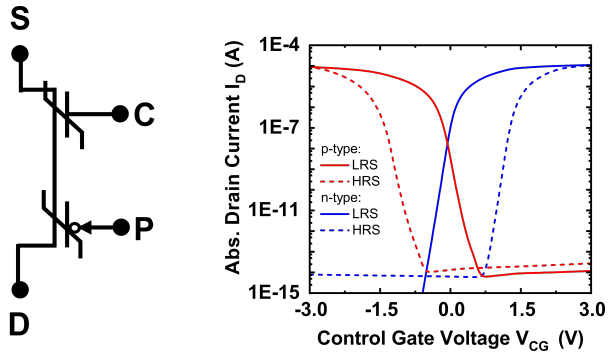
Fig. 9: Schematic of the integration of an ferroelectric (Fe) Hafniumoxide with an remanent polarization into the gate stack of an FET. Diagonal line indicates Fe-behavior

emerging reconfigurable field effect transistors transistors (RFETs) [86, 87, 88]. This nanotechnology provides the opportunity to switch its conduction on-the-fly from p-type to n-type. This feature offers new potential for a large number of new circuit topologies. For example a simple NAND gate can be dynamically switched to a NOR functionality [89]. Albeit some pioneering studies have demonstrated that the RFET functionality can be programmed in a non-volatile fashion [90, 91, 92], most circuit design activities so far have considered only volatile circuit operation [93, 94, 95, 11, 96]. Here we discuss the expected non-volatile device characteristics of the combined *Ferroelectric Reconfigurable Field-Effect Transistors* (FeRFET) and how they impact circuit design topologies for both *Memory-In-Logic* as well as *Logic-In-Memory* approaches and the potential benefit they offer to memory-intensive applications.

A. Ferroelectric Reconfigurable Field-Effect Transistors

RFETs are typically based on ambipolar conduction in Schottky-barrier transistors, allowing for electron and hole transport. A physical structure with multiple independent gates is then used to program the transistor and control the current. Typically, two [88] or three independent gates [87] are used. The RFET structure can simply be enhanced by a ferroelectric (Fe) storage function via placing a doped HfO_2 material [97, 98] directly into the gate stack (Fig. 9). This introduces an internal *remanent* polarization acting superimposed with the applied external potential.

The effect of the ferroelectric material on the device characteristics is different for control (C) gate and program (P) gate as shown in (Fig. 10). The state stored at the program gate puts the transistor either into p-type or n-type transistor. The programmed operation is retained after the applied voltage is withdrawn. The effect at the control gate is different. Depending on the stored polarization orientation, the device is operated either with a high or low threshold voltage, which translate into a high-resistive state (HRS) or low-resistive state (LRS), respectively. This way overall four individual operation states are generated which are shown by the exemplary simulation data in Fig. 10(b). The simulation is based on a TCAD model from [94], adding a Fe layer in the gate stack. Note, that the voltage for programming has to be two to three times larger than the typical operation voltage. This is inherent to the Fe storage mechanism, where the same terminals are operated for storing a state and readout.



(a) *(b)*

Fig. 10: Operation of an ferroelectric RFET. (a) schematic circuit symbol of a dual gated devices with source (S), drain (D), control (C) and program (P). (b) Exemplary TCAD simulations of a dual gated device with 24 nm gate lengths. For both non-volatile programmed configurations (n-type and p-type) low resistive state (LRS) and high resistive state (HRS) are indicated.

B. Memory-In-Logic Cell Topologies

In general two paradigms can be distinguished for the combination of ferroelectric memory with logic. In the first one, FeFETs are implemented within an existing logic circuit to enhance the functionality or locally store data [99, 100]. Naturally, this *Memory-In-Logic* approach can be transferred to RFETs to fix the logic gate functionality in a non-volatile fashion. An example cell design, employing this strategy is shown in Fig. 11. The cell comprises four transistors with three gates each. Notably, the ferroelectric is just present at all outer gates (program gates) which are combined to bias the transistors to either p- or n-type operation. The cell is based on the volatile design as proposed in [101]. As compared to the original design, P and $\overline{P}$ are not used as data inputs, but configure the gate to either compute the XOR or XNOR function of the inputs A and B. Note, that the cell is built for a static, pass-transistor-like style of operation, i.e. there are data inputs located at the source and drain terminals of some transistors. The output is calculated purely combinatorial. The output switches directly, if one of the volatile input signals is switched. The big benefit of this cell is, that the data paths for programming and operation are completely separated. This way, the overhead for providing both program and operation signals with different voltage levels is reduced.

C. Logic-In-Memory Cell Topologies

The second paradigm discussed here is the direct execution of logic inside a given memory array. Here the state stored by the ferroelectric layer at the control gate is exploited as logic input. In order to do so the logic operation is conducted sequentially. This is exemplarily discussed on a simple AND-array-like cell design in Fig. 12(a). In a first step, the LRS or HRS of the control gate is selected by applying a high set voltage at the word line (WL). The stored state serves as an input A. Secondly, input B is applied in volatile fashion at the same WL using a distinctive smaller V_{DD}. Simultaneously, the program line has to be biased, to enable a dynamic readout at

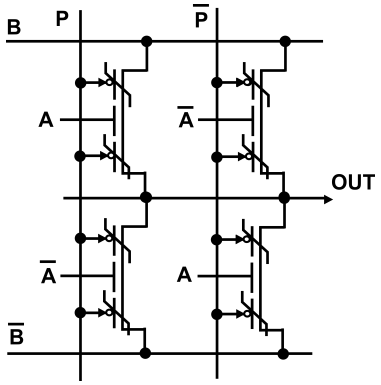
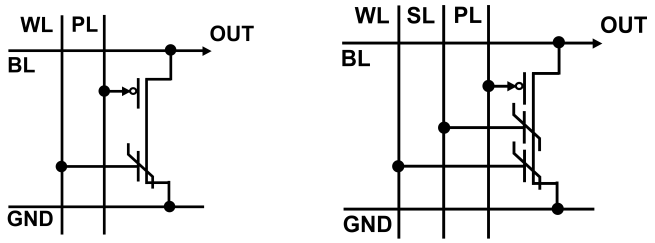


Fig. 11: A programmable XOR/XNOR cell based on four ferroelectric RFETs with three gates each. The program signals P and $\overline{P}$ are used to program the funtionality in a non-volatile fashion. A and B are volatile logic inputs.



(a) *(b)*

Fig. 12: Array cells for possible exploitation of ferroelectric RFETs for Logic-In-Memory applications. Word line (WL), bit line (BL), select line (SL) and polarity line (PL) are used to address the cells. (a) AND-array-like cell design. (b) NOR-array-like cell design.

the bit-line. The output will compute the (N)OR function of A and B. Note, that either a sense amplifier or a respective pull-up/down-network is needed to transform the current response into an inverted voltage response.

More options arise for RFETs with multiple independent gates, which intrinsically support a wired-AND functionality [102]. The corresponding cell can be seen as a NOR-array-like design as proposed by [103] and is shown in Fig. 12(b). The middle gate can either be used as access transistor or to perform more complex logic operations in a similar operation scheme as described for simple AND-array example. This way, the cell can perform an AND-OR-INVERT or XNOR function in a dynamic fashion [104]. Finally, the static XOR/XNOR in Fig. 11 can also be realized in array if four such cells are coupled, given that the GND line can be reallocated as signal input.

D. FeRFET Circuit Design Challenges and Opportunities

The co-integration of a Fe with MOSFETs already allows for various functionalities in terms of logic gate design. FeFET memory cells have already been demonstrated to provide all needed basic functions, such as NAND, [105] and XOR [104], flip-flops [106] and LUTs [100, 107]. In-array operation of various half- and full-adder designs [103] has been shown experimentally. Inter-coupled arrays can be used for flexible computation, bit-passing and data storage [108]. Beyond that, biologic-inspired behavior [109, 110, 111, 112] and analog signal processing [113] have been demonstrated using FeFET cells as well. Consequently, the main challenge for exploiting a co-integration of FeFETs with reconfigurable nanotechnologies is to find applications where the added benefit of the reconfiguration is worth the overhead of the additional polarity signal lines. One such target application are binary neural networks [114]. Particularly the very efficient XOR and XNOR implementation enabled by the RFET base technology is suitable to be employed for this type of computing paradigm [115]. The Fe layer allows non-volatility which can be used to store weights as well as other parameters. In contrast to memristors, which carry out computation in analog domain, FeRFETs can enable logic computation in the digital domain without the need of an extensive peripheral circuits. Hence, CIM paradigms based on FeRFETs can in principle be integrated as add-on functionality into the front-end of conventional CMOS circuits. Following this path, it might be possible to deduce paradigms exploiting both *Logic-In-Memory* and *Memory-In-Logic* concepts in intermixed fashion.

While this poses FeRFETs as a promising alternative to ReRAM technologies, one major limitation until now is that RFETs are less mature technology in terms of commercialization. Thus extensive work on design enablement is needed for FeRFET based approaches. For example EDA for RFETs [116, 117] needs to be extended in order to support the added capabilities of non-volatility to enable commercial integration of FeRFETs within circuit.

VI. Conclusions and Discussions

In the present work, we explored various emerging architectures and solutions which enable computation-in-memory. We looked at two main technologies which can contribute to solve the Von-Neumann bottleneck of data transport between separate compute and memory blocks. We gave an overview of how memristors have added new computing paradigms enabling computation-in-memory. We introduced CIM architecture and explored how the integrated core comprising of crossbar arrays and peripheral units can carry out logical and arithmetic operations along with storing data. Such capabilities are widely applicable in neuromorphic compute loads. Compute-intensive loads such as vector multiplication can be easily offloaded to CIM architectures with low power and latency overheads.

However, CIM also comes with its own set of challenges. Firstly, due to the requirements of extra analog computation, CIM demands periphery circuits for data format conversion such as ADC. Since ADC imposes significant area and power overhead, efficient ADC design is crucial to harness the full potential of CIM architectures.

Secondly, wider adoption of CIM paradigm critically depends on the support of effective design automation flows. To that effect, there is a bottom-up growth in the research of such tools, starting from technology mapping. The tools are connected to standard logic synthesis flows, and have been fine-tuned for optimization of delay, device count and crossbar constraints. Future research in this space includes connecting system-level design automation flows and also extending the support towards testing and verification. This requires streamlining design automation process and the fabrication process to achieve

higher yields. Thirdly, the immature fabrication techniques result in reliability problems. Thus, to take advantages of CIM in practical applications, efficient fault-detection and effective fault-tolerance techniques need to be explored across the various abstraction levels.

Fourthly, within CIM paradigms, the unavoidable requirement of different voltages for read and write can lead to excessive power requirements. Further, this skewed voltage for read and write also requires different voltage drivers and can put extra burden on the physical resources within the circuit implementation. Research in this direction is an integral component to make the whole concept of CIM more feasible and to enable co-existence with conventional CMOS-based circuits.

Lastly, while most of the research discussed here has been on ReRAMs, new emerging technologies need to be explored to augment the CIM reqirements. The newly proposed combination of ferroelectrics on emerging reconfigurable transistors opens up new avenues in CIM applications. The co-integration with CMOS allows for an easy integration with the existing manufacturing process. The devices allow for dynamically changing the data and control paths within a circuit, opening up extended functionalities for both computation and storage within the same circuit structure.

ACKNOWLEDGEMENTS

Anupam Chattopadhyay gratefully acknowledge the National Research Foundation (NRF), Singapore for the NRF-CRP grant NRF-CRP21-2018-003.

REFERENCES

[1] Nor Zaidi Haron et al. "Why is CMOS scaling coming to an END?" In: *Design and Test Workshop*. 2008.
[2] Tze-Chiang Chen. "Overcoming research challenges for CMOS scaling: Industry directions". In: *Conference on Solid-State and IC Technology Proceedings*. 2006.
[3] Yi Sun et al. "Deepid3: Face recognition with very deep neural networks". In: *arXiv preprint 1502.00873* (2015).
[4] Mark T Bohr et al. "CMOS scaling trends and beyond". In: *IEEE Micro* (2017).
[5] Mohammed A Zidan et al. "The future of electronics based on memristive systems". In: *Nature Electronics* (2018).
[6] Lisa Wu Wills et al. "Guest Editorial: IEEE TC Special Issue on Domain-Specific Architectures for Emerging Applications". In: *History of Computing* (2020).
[7] William J Dally et al. "Domain-specific hardware accelerators". In: *Communications* (2020).
[8] Maria Malik et al. "Architecture exploration for energy-efficient embedded vision applications: From general purpose processor to domain specific accelerator". In: *ISVLSI*. 2016.
[9] Muath Abu Lebdeh et al. "Memristive device based circuits for computation-in-memory architectures". In: *ISCAS*. 2019.
[10] Anne Siemon et al. "Memristive Device Modeling and Circuit Design Exploration for Computation-in-Memory". In: *ISCAS*. 2019.
[11] Shubham Rai et al. "Designing efficient circuits based on runtime-reconfigurable field-effect transistors". In: *TVLSI* (2018).
[12] Ping Chi et al. "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory". In: *Computer Architecture News* (2016).
[13] Said Hamdioui et al. "Memristor based computation-in-memory architecture for data-intensive applications". In: *DATE*. 2015.
[14] Hsu et al. "AI Edge Devices Using Computing-In-Memory and Processing-In-Sensor: From System to Device". In: *IEDM*. 2019.
[15] Zhi Zhou et al. "Edge intelligence: Paving the last mile of artificial intelligence with edge computing". In: *IEEE* (2019).
[16] Hoang Anh Du Nguyen et al. "A classification of memory-centric computing". In: *JETC* (2020).
[17] Manuel Le Gallo et al. "Mixed-precision in-memory computing". In: *Nature Electronics* (2018).
[18] Daniele Ielmini et al. "In-memory computing with resistive switching devices". In: *Nature Electronics* (2018).
[19] Bing Chen et al. "Efficient in-memory computing architecture based on crossbar arrays". In: *IEDM*. 2015.
[20] Lei Xie et al. "Scouting Logic: A Novel Memristor-Based Logic Design for Resistive Computing". In: *ISVLSI*. 2017.
[21] Shuangchen Li et al. "Pinatubo: A processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories". In: *DAC*. 2016.
[22] Daichi Fujiki et al. "In-Memory Data Parallel Processor". In: *ICASPLOS*. 2018.
[23] Said Hamdioui et al. "Memristor for computing: Myth or reality?" In: *DATE*. 2017.
[24] Leon Chua. "Memristor-the missing circuit element". In: *Circuit Theory* (1971).
[25] Dmitri B Strukov et al. "The missing memristor found". In: *Nature* (2008).
[26] Kyungah Seo et al. "Analog memory and spike-timing-dependent plasticity characteristics of a nanoscale titanium oxide bilayer resistive switching device". In: *Nanotechnology* (2011).
[27] Ting Chang et al. "Short-term memory to long-term memory transition in a nanoscale memristor". In: *ACS Nano* (2011).
[28] Z Fang et al. "Multilayer-Based Forming-Free RRAM Devices With Excellent Uniformity". In: *Electron Device Letters* (2011).
[29] Mohammad Javad Sharifi et al. "General SPICE models for memristor and application to circuit simulation of memristor-based synapses and memory cells". In: *Circuits, Systems, and Computers* (2010).
[30] Cory E Merkel et al. "Reconfigurable N-level memristor memory design". In: *IJCNN*. 2011.
[31] Alvaro Velasquez et al. "Parallel boolean matrix multiplication in linear time using rectifying memristors". In: *ISCAS*. 2016.
[32] Ali Shafiee et al. "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars". In: *ISCAS*. 2016.
[33] Jaffrey Draper et al. "A prototype processing-in-memory (PIM) chip for the data-intensive architecture (DIVA) system". In: *Journal of VLSI signal processing systems* (2005).
[34] Jeff Draper et al. "The architecture of the DIVA processing-in-memory chip". In: *conference on Super computing*. 2002.
[35] Debjyoti Bhattacharjee et al. "ReVAMP: ReRAM based VLIW architecture for in-memory computing". In: *DATE*. 2017.
[36] Said Hamdioui et al. "Applications of computation-in-memory architectures based on memristive devices". In: *DATE*. 2019.
[37] Said Hamdioui et al. "Testing Computation-in-Memory Architectures Based on Emerging Memories". In: *ITC*. 2019.
[38] Lixue Xia et al. "Fault-tolerant training with on-line fault detection for RRAM-based neural computing systems". In: *DAC*. 2017.
[39] Ching-Yi Chen et al. "RRAM defect modeling and failure analysis based on march test and a novel squeeze-search scheme". In: *TC* (2015).
[40] Moritz Fieback et al. "Testing Scouting Logic-Based Computation-in-Memory Architectures". In: *ETS*. 2020.
[41] R Degraeve et al. "Causes and consequences of the stochastic aspect of filamentary RRAM". In: *Microelectronic Engineering* (2015).
[42] Lerong Chen et al. "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar". In: *DATE*. 2017.
[43] Wenqin Huangfu et al. "Computation-oriented fault-tolerance schemes for RRAM computing systems". In: *ASP-DAC*. 2017.
[44] Karsten Beckmann et al. "Nanoscale hafnium oxide RRAM devices exhibit pulse dependent behavior and multi-level resistance capability". In: *MRS Advances* (2016).
[45] Arjun Chaudhuri et al. "Analysis of Process Variations, Defects, and Design-Induced Coupling in Memristors". In: *ITC*. 2018.
[46] Sachhidh Kannan et al. "Sneak-path testing of crossbar-based nonvolatile random access memories". In: *Nanotechnology* (2013).
[47] Boxun Li et al. "ICE: inline calibration for memristor crossbar-based computing engine". In: *DATE*. 2014.
[48] Jue Wang et al. "i2WAP: Improving non-volatile cache lifetime by reducing inter-and intra-set write variations". In: *HPCA*. 2013.
[49] Mengyun Liu et al. "Fault Tolerance for RRAM-Based Matrix Operations". In: *ITC*. 2018.
[50] Mengyun Liu et al. "Algorithmic Fault Detection for RRAM-based Matrix Operations". In: *TODAES* (2020).
[51] Mingqing Wang et al. "Theory study and implementation of configurable ECC on RRAM memory". In: *NVMTS*. 2015.
[52] Mengyun Liu et al. "Online Fault Detection in ReRAM-Based Computing Systems by Monitoring Dynamic Power Consumption". In: *ITC*. 2020.

[53] Giovanni De Micheli. *Synthesis and optimization of digital circuits*. 1994.

[54] Robert Brayton and Alan Mishchenko. "ABC: An Academic Industrial-Strength Verification Tool". In: *Computer Aided Verification*. 2010.

[55] Luca Amarù et al. "Majority logic synthesis". In: *ICCAD*. 2018.

[56] Patrick McGeer et al. "ESPRESSO-SIGNATURE: A new exact minimizer for logic functions". In: *DAC*. 1993.

[57] Chang-Yeong Lee. "Representation of switching circuits by binary-decision programs". In: *The Bell System Technical Journal* (1959).

[58] Ellen M Sentovich et al. "SIS: A system for sequential circuit synthesis". In: (1992).

[59] *Synopsys Design Compiler*. https : / / www . synopsys . com / implementation-and-signoff/rtl-synthesis-test/dc-ultra.html. 2018.

[60] *Xilinx Design Suite*. https : / / www . xilinx . com / products / design - tools/vivado.html. 2018.

[61] Debjyoti Bhattacharjee et al. "Technology-aware logic synthesis for ReRAM based in-memory computing". In: *DATE*. 2018.

[62] Saeideh Shirinzadeh et al. "Fast logic synthesis for RRAM-based in-memory computing using majority-inverter graphs". In: *DATE*. 2016.

[63] Eero Lehtonen et al. "Stateful implication logic with memristors". In: *Symposium on Nanoscale Architectures*. 2009.

[64] E. Lehtonen, and others. "Two memristors suffice to compute all Boolean functions". In: (2010).

[65] Anika Raghuvanshi et al. "Logic Synthesis and a Generalized Notation for Memristor-Realized Material Implication Gates". In: *ICCAD* (2014).

[66] Anupam Chattopadhyay et al. "Combinational logic synthesis for material implication". In: *VLSI-SOC*. 2011.

[67] Debjyoti Bhattacharjee et al. "Delay-optimal technology mapping for in-memory computing using ReRAM devices". In: *ICCAD*. 2016.

[68] M. Soeken et al. "An MIG-based Compiler for Programmable Logic-in-Memory Architectures". In: *DAC*. 2016.

[69] Debjyoti Bhattacharjee et al. "Crossbar-constrained technology mapping for ReRAM based in-memory computing". In: *TC* (2020).

[70] Rotem Ben-Hur et al. "Simpler magic: Synthesis and mapping of in-memory logic executed in a single row to improve throughput". In: *TCAD* (2019).

[71] Rotem Ben Hur et al. "SIMPLE MAGIC: Synthesis and in-memory mapping of logic execution for memristor-aided logic". In: *ICCAD*. 2017.

[72] Valerio Tenace et al. "SAID: A Supergate-Aided Logic Synthesis Flow for Memristive Crossbars". In: *DATE*. 2019.

[73] Debjyoti Bhattacharjee et al. "CONTRA: Area-Constrained Technology Mapping Framework For Memristive Memory Processing Unit". In: *ICCAD*. 2020.

[74] Nishil Talati et al. "Logic design within memristive memories using memristor-aided loGIC (MAGIC)". In: *Nanotechnology* (2016).

[75] Rahul Gharpinde et al. "A scalable in-memory logic synthesis approach using memristor crossbar". In: *TVLSI* (2017).

[76] Phrangboklang L Thangkhiew et al. "Scalable in-memory mapping of Boolean functions in memristive crossbar array using simulated annealing". In: *Journal of Systems Architecture* (2018).

[77] Dev Narayan Yadav et al. "Look-ahead mapping of Boolean functions in memristive crossbar array". In: *Integration* (2019).

[78] Pierre-Emmanuel Gaillardon et al. "The programmable logic-in-memory (PLiM) computer". In: *DATE*. 2016.

[79] Marco andothers Cassinerio. "Logic computation in phase change materials by threshold and memory switching". In: *Advanced Materials* (2013).

[80] Wang Kang et al. "In-memory processing paradigm for bitwise logic operations in STT–MRAM". In: *Transactions on Magnetics* (2017).

[81] T Mikolajick et al. "The Past, the Present, and the Future of Ferroelectric Memories". In: *TED* (2020).

[82] Dayane Reis et al. "Computing in memory with FeFETs". In: *ISLPED*. 2018.

[83] Yun Long et al. "A Ferroelectric FET-Based Processing-in-Memory Architecture for DNN Acceleration". In: *Journal on Exploratory Solid-State Computational Devices and Circuits* (2019).

[84] Dayane Reis et al. "A Computing-in-Memory Engine for Searching on Homomorphically Encrypted Data". In: *Journal on Exploratory Solid-State Computational Devices and Circuits* (2019).

[85] Mingyen Lee et al. "FeFET-based low-power bitwise logic-in-memory with direct write-back and data-adaptive dynamic sensing interface". In: *ISLPED*. 2020.

[86] Thomas Mikolajick et al. "The RFET–a reconfigurable nanowire transistor and its application to novel electronic circuits and systems". In: *Semiconductor Science and Technology* (2017).

[87] Michele De Marchi et al. "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs". In: *IEDM*. 2012.

[88] André Heinzig et al. "Reconfigurable silicon nanowire transistors". In: *Nano Letters* (2012).

[89] Jens Trommer et al. "Elementary aspects for circuit implementation of reconfigurable nanowire transistors". In: *Electron device letters* (2013).

[90] Jun-Mo Park et al. "High-density reconfigurable devices with programmable bottom-gate array". In: *Electron Device Letters* (2017).

[91] So Jeong Park et al. "Reconfigurable Si nanowire nonvolatile transistors". In: *Advanced Electronic Materials* (2018).

[92] Violetta Sessi et al. "A Silicon Nanowire Ferroelectric Field-Effect Transistor". In: *Advanced Electronic Materials* (2020).

[93] Jian Zhang et al. "Configurable circuits featuring dual-threshold-voltage design with three-independent-gate silicon nanowire FETs". In: *TCAS I* (2014).

[94] Jens Trommer et al. "Functionality-enhanced logic gate design enabled by symmetrical reconfigurable silicon nanowire transistors". In: *Transactions on Nanotechnology* (2015).

[95] Michael Raitza et al. "Exploiting transistor-level reconfiguration to optimize combinational circuits". In: *DATE*. 2017.

[96] Shubham Rai et al. "Emerging reconfigurable nanotechnologies: Can they support future electronics?" In: *ICCAD*. 2018.

[97] TS Böscke et al. "Phase transitions in ferroelectric silicon doped hafnium oxide". In: *Applied Physics Letters* (2011).

[98] Min Hyuk Park et al. "Review and perspective on ferroelectric HfO 2-based thin films for memory applications". In: *Communications* (2018).

[99] Xunzhao Yin et al. "Exploiting ferroelectric FETs for low-power non-volatile logic-in-memory circuits". In: *ICCAD*. 2016.

[100] Evelyn T Breyer et al. "Ultra-dense co-integration of FeFETs and CMOS logic enabling very-fine grained logic-in-memory". In: *ESSDERC*. 2019.

[101] Ogun Turkyilmaz et al. "Self-checking ripple-carry adder with ambipolar silicon nanowire FET". In: *ISCAS*. 2013.

[102] Maik Simon et al. "A wired-AND transistor: Polarity controllable FET with multiple inputs". In: *Device Research Conference*. 2018.

[103] Evelyn T Breyer et al. "Compact FeFET circuit building blocks for fast and efficient nonvolatile logic-in-memory". In: (2020).

[104] Evelyn T andothers Breyer. "Demonstration of versatile nonvolatile logic gates in 28nm HKMG FeFET technology". In: *ISCAS*. 2018.

[105] ET Breyer et al. "Reconfigurable NAND/NOR logic gates in 28 nm HKMG and 22 nm FD-SOI FeFET technology". In: *IEDM*. 2017.

[106] Danni Wang et al. "Ferroelectric transistor based non-volatile flip-flop". In: *ISLPED*. 2016.

[107] Xiaoming Chen et al. "The Impact of Ferroelectric FETs on Digital and Analog Circuits and Architectures". In: *Design & Test* (2019).

[108] Evelyn T Breyer et al. "Flexible Memory, Bit-Passing and Mixed Logic/Memory Operation of two Intercoupled FeFET Arrays". In: *ISCAS*. 2020.

[109] Y Nishitani et al. "Three-terminal ferroelectric synapse device with concurrent learning function for artificial neural networks". In: *Journal of Applied Physics* (2012).

[110] H Mulaosmanovic et al. "Novel ferroelectric FET based synapse for neuromorphic systems". In: *Symposium on VLSI Technology*. 2017.

[111] Halid Mulaosmanovic et al. "Mimicking biological neurons with a nanoscale ferroelectric transistor". In: *Nanoscale* (2018).

[112] S Dutta et al. "Biologically plausible ferroelectric quasi-leaky integrate and fire neuron". In: *Symposium on VLSI Technology*. 2019.

[113] Halid Mulaosmanovic et al. "Reconfigurable frequency multiplication with a ferroelectric transistor". In: *Nature Electronics* (2020).

[114] Mohammad Rastegari et al. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *Computer vision*. 2016.

[115] Jong-Ho Bae et al. "Reconfigurable Field-Effect Transistor as a Synaptic Device for XNOR Binary Neural Network". In: *Electron Device Letters* (2019).

[116] Shubham Rai et al. "A physical synthesis flow for early technology evaluation of silicon nanowire based reconfigurable FETs". In: *DATE*. 2018.

[117] Shubham Rai et al. "Technology mapping flow for emerging reconfigurable silicon nanowire transistors". In: *DATE*. 2018.