Energy-efficient Computation-In-Memory Architecture using Emerging Technologies

Bishnoi, Rajendra; Diware, Sumit; Gebregiorgis, Anteneh; Thomann, Simon; Mannaa, Sara; Deveautour, Bastien; Marchand, Cedric; Bosio, Alberto; Deleruyelle, Damien; O'Connor, Ian

DOI
10.1109/ICM60448.2023.10378889

Publication date
2023

Document Version
Final published version

Published in
Proceedings of the 2023 International Conference on Microelectronics, ICM 2023

Energy-efficient Computation-In-Memory Architecture using Emerging Technologies

Rajendra Bishnoi*, Sumit Diware*, Anteneh Gebregiorgis*, Simon Thomann***, Sara Mannaa**, Bastien Deveautour**, Cédric Marchand**, Alberto Bosio**, Damien Deleruyelle**, Ian O'Connor**, Hussam Amrouch***, Said Hamdioui*

*Computer Engineering Laboratory, Delft University of Technology, The Netherlands
**University Lyon, ECL, INSA Lyon, CPE Lyon, Institut de Nanotechnologies de Lyon, France
***Chair of AI Processor Design; Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, Germany

2023 International Conference on Microelectronics (ICM) | 979-8-3503-8082-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICM60448.2023.10378889

***Abstract*—Deep Learning (DL) has recently led to remarkable advancements, however, it faces severe computation related challenges. Existing Von-Neumann-based solutions are dealing with issues such as memory bandwidth limitations and energy inefficiency. Computation-In-Memory (CIM) has the potential to address this problem by integrating processing elements directly into the memory architecture, reducing data movement and enhancing the overall efficiency of the system. In this work, we propose CIM architecture using three distinct emerging technologies. Firstly, a CIM architecture utilizing Ferroelectric Field-Effect Transistors (FeFET) is shown and the resulting errors from the analog compute scheme are injected into the emerging algorithm of Hyperdimensional Computing. Subsequently, we explore Vertical Nanowire Field-Effect Transistors (VNWFETs) based CIM within a 3D computing architecture, demonstrating improved energy efficiency and reconfigurability for CIM. Additionally, we improve the accuracy of the Resistive Random Access Memories (RRAM) based CIM architecture using two mapping-based solutions. These three technologies exhibit non-volatile characteristics, and when integrated into the CIM architecture, they yield significant advantages, including enhanced energy efficiency, reliability, and accuracy in computing processes.**

I. Introduction

Deep neural networks (DNNs) have demonstrated significant advancements across a variety of applications, including image recognition, speech recognition, healthcare and natural language processing [1]–[3]. In general, DNNs are configured using several layers with many inputs that are connected through weights to their outputs and learn useful representations by adjusting their weights algorithmically. This architectural design allows DNNs to effectively capture complex patterns and relationships within data, enabling them to excel in tasks requiring sophisticated decision-making capabilities. However, they face challenges for computation efficiency, especially as they grow in complexity and require substantial computational resources, that hinder their deployment in resource-constrained environments like IoT devices or mobile platforms.

Current computing systems, including Central Processing Units (CPUs), Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), and Tensor Processing Units (TPUs), are developed using CMOS-based von-Neumann architecture [4]–[6]. In these systems, the physical separation of memory and compute units results in a large number of data transfers necessary to execute vector-matrix multiplication (VMM) operations for neural network applications, leading to a degradation in performance and energy efficiency. Additionally, CMOS technology is struggling with challenges such as excessive sub-threshold leakage and scalability issues. Computation-in-memory, where data processing occurs directly within the memory, demonstrates potential for enhancing computation efficiency and addressing the data transfer bottleneck. The CIM architecture, combined with emerging non-volatile technologies, delivers various advantages, including leakage-free storage, high density, high scalability, and faster accesses. These features have the potential to further enhance the computation efficiency of the system.

In this paper, we exploit three emerging technologies, namely Ferroelectric Field-Effect Transistors (FeFET), Vertical Nanowire Field-Effect Transistors (VNWFETs), and Resistive Random Access Memories (RRAM), to enhance Computational-In-Memory (CIM) capabilities for neural network development. The contributions for this paper are as follows:

- We address ferroelectric stochasticity and temperature effects in FeFET-based IMC, proposing innovative strategies like on-chip cooling using thermoelectric devices to ensure reliable computing.
- We highlight the potential of VNWFETs in 3D computing architecture, demonstrating enhanced energy efficiency and reconfigurability advantage for computing-in-memory and approximate computing.
- We present two mapping-based solutions to enhance the accuracy of the RRAM-based CIM architecture for Neural Networks, namely the unbalanced bit-slicing scheme and the mapping-aware biased training methodology.

The rest of this paper is organized as follows. Section II presents the fundamentals of CIM architecture. CIM Using FeFET Technology is discussed in Section III. Section IV and Section V provides details of the VNWFET- and RRAM-based CIM architectures, respectively. Finally, Section VII concludes the paper.

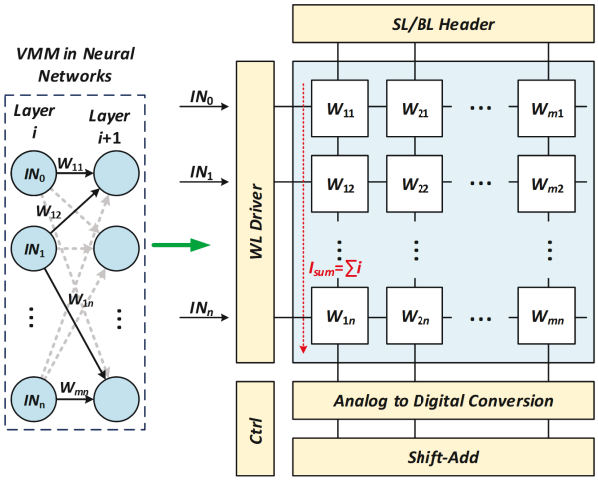


Fig. 1: Computation-In-Memory architecture [7].

II. Computation-in-memory (CIM) Architecture

Computation-in-memory (CIM) proves to be a highly efficient alternative to the conventional von-Neumann architecture for the implementation of Vector-Matrix Multiplication (VMM) in neural network hardware. The mapping of VMM operation between layers onto the CIM architecture is illustrated in Figure 1. In the CIM configuration, data is stored as conductance in memory elements arranged in a grid-like structure called crossbar [8]–[12]. This crossbar operates in the analog domain and interfaces with other digital components in the system through peripheral circuits such as digital-to-analog converters (DACs) and analog-to-digital converters (ADCs). The weight matrix is translated into conductances within the crossbar, and input voltages (IN's) are applied using DACs. This sets off a current flow through all conductances, following Ohm's law and simulating element-wise multiplication of voltages and conductances. In the same column, currents from conductances accumulate according to Kirchhoff's law, generating output currents (I's). Each column, consequently, performs a multiply-and-accumulate operation in the analog domain. Simultaneous output production from all columns allows the VMM operation to be executed with an O(1) time complexity. Following this, the column currents are converted into digital outputs using ADCs and transmitted to other system components for subsequent processing or storage. This process demonstrates the potential advantages of CIM for the neural network computations.

III. CIM using Ferroelectric Field-Effect Transistor

Applications like Deep Neural Networks (DNNs) and big data are heavily data-centric, demanding excessive amounts of on-chip memory, while traditional CMOS-based SRAM is power and area-hungry. At the same time, the slow-down in classical CMOS technology scaling boosted the attention of emerging non-volatile memory (NVM) technologies. The literature has proposed several different technologies like, Spin-Transfer Torque Magnetic RAM (STT-MRAM) [13], Resistive RAM (ReRAM) [14] or Ferroelectric FET (FET) [15]. Classically SRAM needs at least six transistors, whereas NVMs needs only one memory device plus typically one access

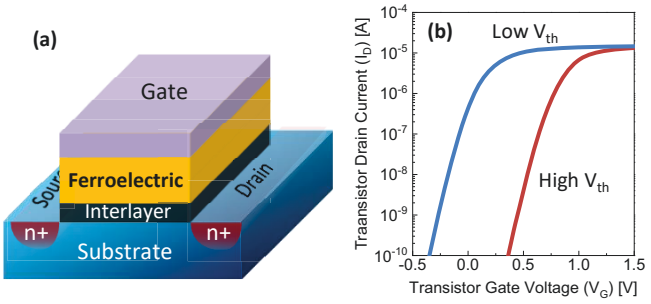


Fig. 2: (a) FeFET where the high-κ layer is replaced by a thick (e.g., 10 nm) layer of ferroelectric material. (b) Two distinguishable states are created by polarizing the ferroelectric layer, i.e., low-V_{TH} and high-V_{TH}, corresponding to high and low currents, respectively.

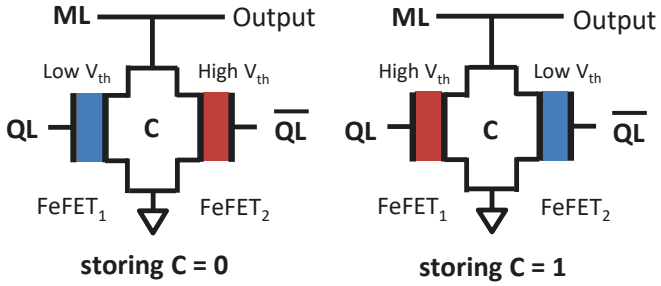


Fig. 3: Circuit of the FeFET-based Ternary Content Addressable Memory (TCAM) cell implementation. A complementary write scheme stores logic 0 and 1 in a cell.

transistor [16]. Additionally, NVMs technologies retain their stored information even when powered off, drastically reducing static power consumption. These advantages enable larger on-chip memories compared to conventional SRAM. However, tight integration with logic is challenging for most NVMs due to their lacking CMOS compatibility [17].

Since 2007, hafnium oxide (HfO_2) has been adopted as a high-κ material in existing manufacturing processes. Later, in 2011, ferroelectricity was discovered in $Hf_{0.5}Zr_{0.5}O_2$. Replacing the high-κ layer of a traditional CMOS transistor with a thick HfO_2-based ferroelectric (FE) layer creates a FeFET (see Figure 2(a)). Except for the addition of two masks, no new process steps or materials are necessary [15], [18], making FeFET a fully CMOS-compatible emerging NVM [18], [19]. Further, as the FE layer only increases the height of the device, the footprint is unchanged, enabling high-density memory.

The non-volatility of FeFET comes from the polarization of the FE layer itself. By applying strong gate voltage biases (e.g., ± 4 V), the polarization can be changed, i.e., a state is *written* into the FE layer. In turn, the polarization of FE layer affects the characteristics of the underlying transistor, where a positive polarization decreases the transistor's V_{TH} (low-V_{TH} state), while a negative polarization increases V_{TH} (high-V_{TH} state). With these two states, shown in Figure 2(b), FeFET can be used to store binary information. To *read* a FeFET, a much smaller probing gate bias is used ($\approx$0.7 V), and the flowing current indicates the stored state within the FeFET.

Using FeFET TCAM cells can be implemented extremely area-efficient, as only two FeFETs are necessary. In compari-

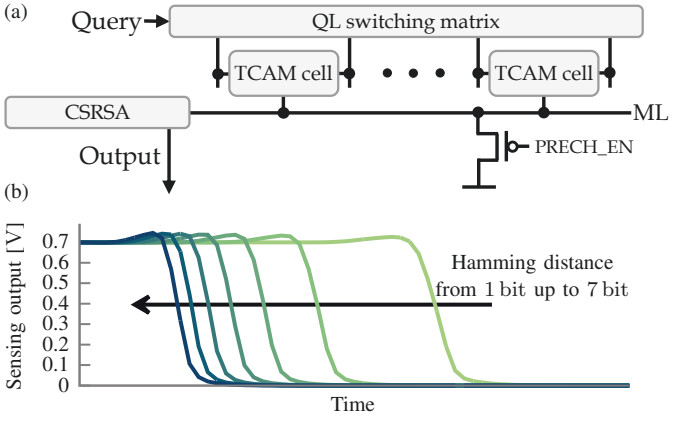


Fig. 4: (a) Circuit of TCAM block. The number of cells (block size) is variable. (b) Example of the output voltage waveforms of the CSRSA for a block size of 15 bit. Data from [20].

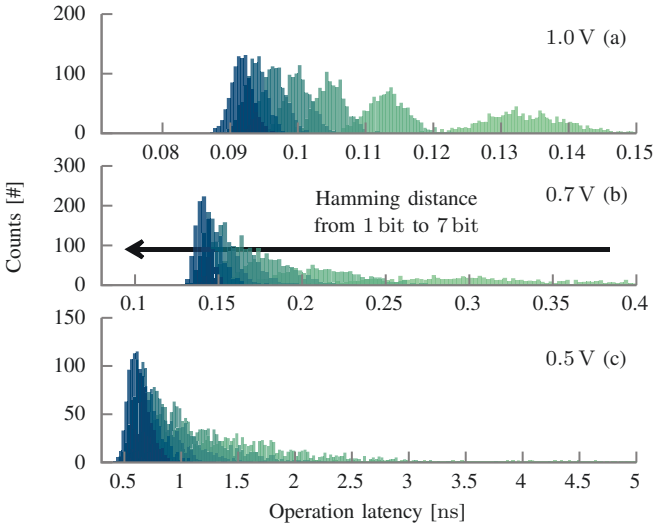


Fig. 5: The operation latencies of FeFET TCAM under process variation at three different supply voltages. The variation for one miss is particularly high at 0.5 V ranging from 0.8 ns to 5.0 ns (not recognizable in the plot). The block size is 15 bit. The results are based on 1000 Monte Carlo SPICE simulations per voltage level and Hamming distance. Data from [20].

son, 16 transistors are needed using conventional technology. Figure 3 shows the circuit of a FeFET-based TCAM cell and how data is stored using a complementary write scheme of the two FeFET. On an abstract level, a TCAM cell will compare the provided query data with the internally stored data (i.e., $Q = C$). The Match Line (ML) (the cell's output) is precharged to the supply voltage during the initialization. When the two data are the same, a match occurs, and the cell will respond with logic `1` and block the path from ML to ground. In case of a miss, when the two data are different, the cell will have *one* conducting path from ML to ground, and the result will be a logic `0`.

Using several of such TCAM cells, a block can be constructed by connecting the cells to a shared ML shown in Figure 4(a). The ML now depends on the miss/match state of all the connected cells, and the number of cells that report a mismatch is proportional to the discharge current of the ML. This is due to the parallel discharge paths that are formed in the cells reporting a mismatch. By using a clocked self-referencing sense amplifier, the discharge current is translated to the temporal domain shown in Figure 4(b). As the discharge current is proportional to the number of cells reporting a miss, the degree of mismatch is equal to the Hamming distance of the bit string stored in the block and an applied query string. From the moment of applying the query data until the sharp output transition of the sense amplifier, an operation delay can be measured, representing the different Hamming distances.

Analog computing schemes are very sensitive to noise and process variation, leading to errors in the final result of the calculation. Using a TCAM block to calculate the Hamming distance faces similar issues. The fact that emerging technologies like FeFET exhibit even more process variation compared to mature technologies exacerbates the problem further. Figure 5 shows the operation latency distributions at different supply voltages of the TCAM block under process variation. With increasing Hamming distances, the margins between the distributions shrink until they start to overlap. However, with the overlap to the neighboring distributions, the operation latency will be interpreted wrongly, leading to errors in the calculation. This can be abstracted and modeled using the concept of a confusion matrix. Each row represents a true Hamming distance value, and the columns represent the potential output value. The cells denote the probability for a specific value X to result in a value of Y; ideally, all the probability mass is on the main diagonal. Evidently, the error probability of the circuit is very high, which will break error-sensitive algorithms like deep neural networks and other conventional machine learning algorithms [21]. Thus, to make use of such unreliable hardware, an emerging computing paradigm that is able to tolerate errors becomes a must.

A. Hyperdimensional Computing on Analog Hardware

Hyperdimensional Computing (HDC) is an emerging brain-inspired algorithm that uses large vectors with elements in the thousands to represent real-world objects in the hyperspace. By using these large-scaled vectors, HDC does not store information in the conventional sense, where each element represents a unique piece of data, but rather relies on patterns encoded in the vectors [22]. To retrieve data, these patterns are recognized, giving HDC strong error and noise resistance [22]. After the concept was proposed by Kanerva in 2009 [23], HDC has been employed in a wide range of applications. Amongst others, language recognition [22], image classification [24], EMG signal processing [25], wafer map defect pattern classification [26], and voice recognition [27] have been showcased in literature.

In order to encode the complex real-world entities from the example applications, only the three basic HDC operations are necessary. Further, the implementation of these operations depends on the data type that is used for the elements of the vectors, which can be only binary, integers, real, or even complex numbers. The binary case, in particular, is

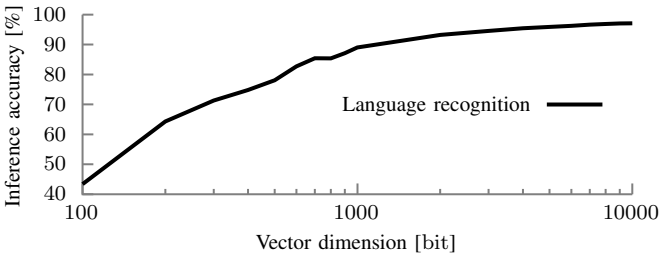


Fig. 6: Relation of inference accuracy and hypervector dimension. Data from [20].

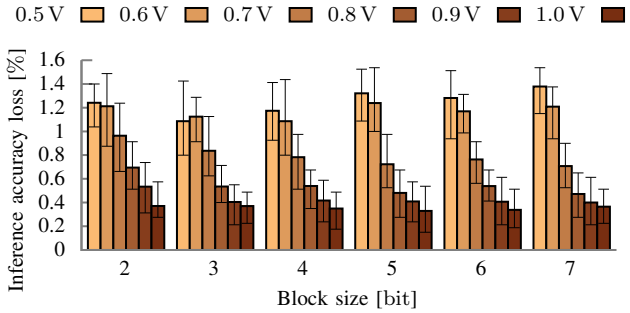


Fig. 7: Inference accuracy loss due to process variation on FeFET-based TCAM block for language recognition. Data from [20].

very computationally lightweight, making HDC very energy efficient. HDC follows a similar flow to other machine learning approaches, where a model is first trained on data and then tested. To infer the model with a query, all the class hypervectors are stored in the associative memory. The result of an inference is the class vector with the highest similarity to the query vector, and the respective class label is selected. Figure 6 shows the accuracy of the language recognition task using binary HDC across different vector dimensions. With increasing vector dimensions, the pattern storage capacity improves, and the accuracy grows higher, peaking around 97 % with a dimension of 10 000 bit.

For binary vectors, the similarity metric is implemented through the Hamming distance. As HDC is very error-resilient, it is a perfect candidate to employ unreliable analog compute schemes whilst maintaining reasonable performance. To test this, we have injected the errors coming from the previously described TCAM block into the inference algorithm of HDC via the abstracted confusion matrices. Figure 7 shows the inference accuracy loss on the test set for the different supply voltages we have tested. As the probabilistic-based error model introduces additional randomness, we have repeated the inference of the whole test set 100 times for each configuration plotted to have sufficient statistical data. Thus, the bars represent the average accuracy loss, and the error bars indicate the standard deviation. From the histograms in Figure 5, we could already estimate the increase in error probability with lower supply voltages. This carries forward, and the peak average accuracy loss is around 1.3 % for 0.5 V V_{DD}. Increasing V_{DD} decreases the average accuracy loss throughout below 0.4 %, showing the trade-off between energy and inference accuracy.

IV. CIM using Vertical Nanowire Field-Effect Transistors

Vertical Nanowire Field Effect Transistors (VNWFETs) represent a promising emerging technology that holds substantial promise for minimizing footprint and, consequently, reducing interconnect capacitance. This has the potential to enhance energy efficiency and seamlessly integrate with advanced 3D integration strategies. In this perspective, the VNWFET emerging technology that may fulfill the energy efficiency, performance, and compact design prerequisites for CIM, serving as viable alternatives to conventional von Neumann machines. Indeed, adding ferroelectric material to the VNWFET gate-stack [28] enables nonvolatile logic as well as nonvolatile reconfigurability.

In previous works such as [29] and [30] we presented a design methodology aimed at bridging the gap between an established (laboratory-scale) VNWFET technology and its corresponding compact model, progressing towards standard static logic cell design and characterization, and ultimately extending to logic synthesis.

A. Vertical Nanowire Field Effect Transistors (VNWFET) : Device & Compact Model

In this study, we employ a VNWFET with a junction-less Gate-All-Around (GAA) structure [31], illustrated in Figure 8. The nanowire channel, homogeneously highly doped and patterned into a boron-doped silicon substrate, controls the current between drain and source contacts. The GAA structure ensures an effective channel length of 14 nm, improving electrostatic control and overcoming scaling issues in conventional planar transistors.

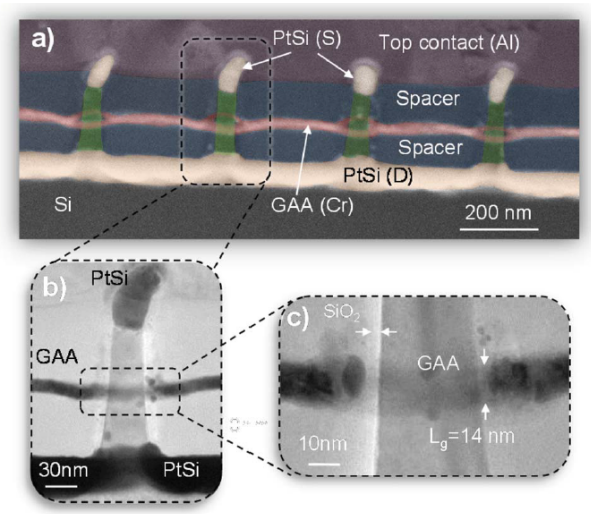


Fig. 8: VNWFET device from [31]: (a) STEM image in cross section of the vertical transistor implemented in nanowire arrays, (b) single VNWFET showing its (c) gate formation

The VNWFET device modeling incorporates carrier transport physics in the junctionless architecture. A SPICE simulation methodology, founded on a unified charge-based control model [32], accurately captures the 3D technology. The compact model addresses depletion and accumulation charges, short channel effects, velocity saturation, Drain-Induced Barrier Lowering (DIBL), Band-To-Band Tunneling (BTBT),

Gate-Induced Drain Leakage (GIDL), and Schottky contact formation [33], [34]. To enhance model accuracy at low drain bias, a semi-empirical field-dependent mobility model is implemented, demonstrating good agreement between measurement and compact model.

B. Logic Cell Characterization: Simulation Flow and resulting delays & Power Consumption

For this study, we implemented 4 basic complementary static: INV1X1, NAND2X1, NOR2X1 and XOR2X1 and a D Flip-Flop. SPICE simulations utilize the compact model implemented as a Verilog-A executable model making use of the vertical nanowire technology and exploring the impact of the variation of critical parameters. Gate physical length (L_g) and nanowire (NW) diameter (d_{nw}) are fabrication-dependent parameters set to $L_g = 18nm$ and $d_{nw} = 22nm$ = 22nm based on experimental devices. This study explores the number of nanowires for p-type and n-type transistors as a key design parameter. Initial steps involve verifying n-type and p-type VNWFET device functionality through DC-sweep simulation, ensuring expected I_{DS}/V_{GS} behavior. Subsequent simulations focus on an elementary inverter gate to determine the optimal ratio between n-type and p-type nanowires for balanced noise margins and well-matched rise and fall times. The number of nanowires are defined as shown in table I

TABLE I
Number of NWs used for each version of a logic gate

Logic Cell	n-type NW values	p-type NW values
INV	4, 24, 44, 64	4, 24, 44, 64
NAND	8, 48, 88, 128	4, 24, 44, 64
NOR	4, 24, 44, 64	8, 48, 88, 128
XOR	8, 48, 88, 128	8, 48, 88, 128

Accurate measurement of latency in individual logic gates is essential. This study evaluates delay in output transition compared to input transition(s), which was measured for both rise and fall transitions according to equation 1.

$$delay = t(V_{out} = 0.5V_{dd}) - t(V_{in} = 0.5V_{dd}) \quad (1)$$

Output rising and falling times are also measured, representing the time for the output voltage to rise/fall from $0.1V_{dd}$ to $0.9V_{dd}$.

Logic cell power consumption comprises static (leakage) and dynamic power. Leakage power arises from cell leakage current in a static state, defined as P_{leak} = $I_{dd}V_{dd}$. Whereas dynamic power includes switching power from load capacitor (C_l) charging and short circuit power during pull-up and pull-down network conduction. Total energy (E_t) consumption per transition is calculated, considering switching energy (E_s) and internal energy as shown in equation 2. In our case, we considered that V_i= $0.1V_{dd}$ and V_f= $0.9V_{dd}$ whereas t_i corresponds to the time at which V_{out}= V_i and t_f corresponds to the time at which V_{out}= V_f.

$$E_{int} = E_t - E_s = V_{dd}\left|\int_{t_i}^{t_f} I_{dd}\, dt\right| - \left|\int_{V_i}^{V_f} C_l V_{out}\, dv_{out}\right| \quad (2)$$

C. Logic Synthesis: from Standard Cell Library Characterization to Synthesis Experiment

This study adopts the standard liberty file format for standard cell library characterization. This hierarchical file includes information on delay models, unit attributes, operating conditions, and lookup tables (LUTs) for timing and power consumption based on input slew and load capacitance. The nonlinear delay model, incorporating input slew rate and load capacitance, necessitates LUTs to store values for synthesis optimization.

The liberty file details timing and power consumption at the output pins of each cell. Matrices reflect values related to output transitions affected by a single input transition, considering the input's effect on the output transition. A time sensing parameter identifies how input transitions affect output transitions, classified as negative unate, positive unate, or non unate.

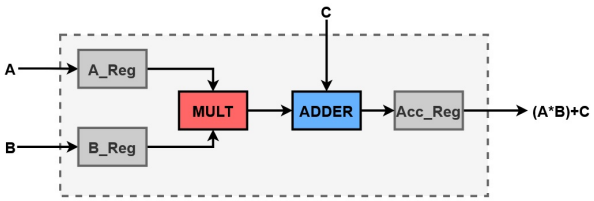


Fig. 9: Processing Element of a systolic array dedicated to MAC operations

We chose to compare the number of cells, delay and power performance of the VNWFET library generated, with two technology libraries: a free 45nm pdk [35] and a 65nm industrial library. We chose to retrain these library equivalent logic cells (INV, NAND, NOR, XOR and D FLip-Flop) to realize a fair comparison. Table II shows how many versions of the same combinational cell is available for each technology library.

TABLE II
Number of employed cells per logic function for each technology library

Library	INV	NAND	NOR	XOR
VNWFET	4	3	3	3
45nm CMOS	4	1	1	1
65nm CMOS	20	15	15	10

The targeted circuit used as benchmark is a Processing Element (PE) within a Systolic Array accelerator designed for matrix-to-matrix multiplication through Multiply and Accumulate operations (MAC). Illustrated in Figure 9 is the PE architecture, comprising a multiplier, an adder, and three D flip-flops responsible for storing and synchronizing operands, weights, and the accumulated result of the operation. Each library is utilized to synthesize PE versions with data sizes of 4, 8, 16, and 32 bits, aiming to assess the scalability of VNWFET in comparison to traditional CMOS technology.

D. Preliminary results on VNWFET performance

The synthesis results reveal two primary aspects: logic optimization and power/timing trade-off. Concerning logic optimization, Table III indicates that the VNWFET exhibits strong performance in comparison to the 45nm library, potentially

attributed to the broader selection of cells within the VNWFET library. However, when compared to the 65nm library, the VNWFET library still performs admirably despite having a less optimal pool of cells. Although the cell count may not explicitly indicate area performance, a reduced number of cells inherently contributes to improved place and route solutions, thereby limiting the overall footprint.

TABLE III
Number cells per circuit based on VNWFET and 45/65 nm CMOS technology library

Circuit	NVWFET	45nm CMOS	65nm CMOS
4-bits PE	230	530	221
8-bits PE	737	1674	705
16-bits PE	2553	5697	2321
32-bits PE	9395	20608	8492

The timing outcomes, as illustrated in Figure 10, indicate that the VNWFET library exhibits commendable performance compared to the 45nm CMOS library, particularly for smaller circuits. However, the trend suggests that the advantage diminishes as circuits grow in size. Additionally, when compared to the 65nm CMOS, which performs optimally, the disadvantage becomes more pronounced. Nevertheless, the minimal timing differences are promising in the context of these preliminary results.

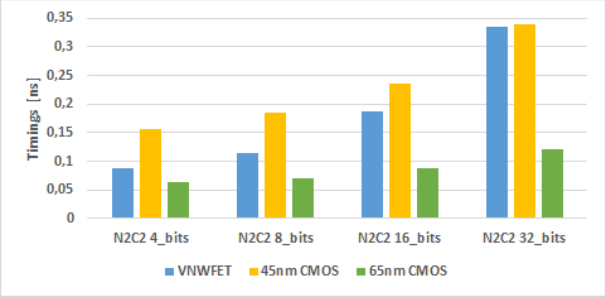


Fig. 10: VNWFET timing results vs 45/65 nm CMOS libraries

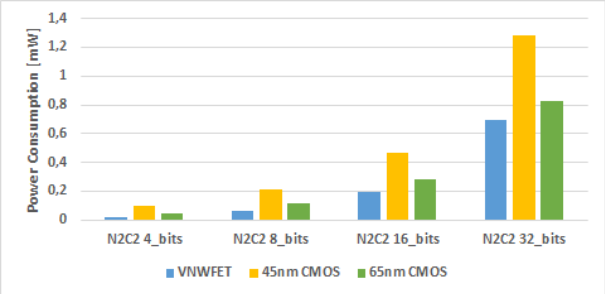


Fig. 11: VNWFET power consumption results vs 45/65 nm CMOS libraries

An equal comparison is made for power consumption results obtained. It is important to emphasize that these results are derived from estimates obtained through the synthesis tool. The power consumption estimates are illustrated in Figure 11.The VNWFET technology library exhibits lower power consumption across each size version of the PE. The scaling trend of VNWFET aligns with that of the CMOS libraries, indicating that these results are likely to remain favorable at higher integration rates.

To push further, the next step will be to add Ferroelectric layer (see Section ÌII) to to VNFET to obtain VNFeFET. We thus plan to have Non-Volatile Boolean gates [36] and operators meaning that we can permanently store a given input data (i.e., an operand) in order to reduce the data transfer and obtain a hybrid approach between CIM and the classical von Neumann paradigm

V. CIM USING RESISTIVE RADOM-ACCESS MEMORY

A. RRAM technology

RRAM devices function as storage elements where data is stored in resistance states. The operational mechanism of RRAM devices is centered around the reversible creation or disruption of a Conductive Filament (CF) within a resistive layer, leading to high and low resistance states. The structural composition of RRAM devices is depicted in Figure 13. It comprises of a metallic oxide that is sandwiched between a Top Electrode (TE) and a Bottom Electrode (BE) as described in the figure. The magnitude of the CF determines the resistance state of the device, which, in turn, is dependent upon the polarity of the applied supply voltage [37]–[41]. To elaborate, when a sufficiently high positive voltage (exceeding the set threshold voltage, V_{set}) is applied, bonds between ions break, creating a conductive filament (CF) comprising vacancies capable of conducting current. This amplifies the CF's size, corresponding to the low resistance state. Conversely, when a negative voltage (below the reset threshold voltage, V_{reset}) is applied, some ions migrate back into the oxide region, thereby diminishing the CF's size. Consequently, the device attains a high resistance state. These RRAM devices are non-volatile in nature, scalable, compact, with reduces access latency and energy consumption as well as compatible with the CMOS technology.

B. CIM architecture using bit-slicing

CIM architectures encounter a challenge in meeting the high bit-precision requirements of neural network applications. This is primarily because the bit capacity of RRAM devices is typically insufficient compared to the bit size needed for neural network weights. Additionally, achieving full-precision inputs and outputs in neural networks necessitates the use of high-resolution digital-to-analog converters (DACs) and analog-to-digital converters (ADCs). However, these components are costly in terms of both energy consumption and physical space, significantly diminishing the hardware efficiency advantages of CIM [42]–[44]. To address these issues, bit-slicing techniques [8], [9], [45] are implemented within CIM architectures.

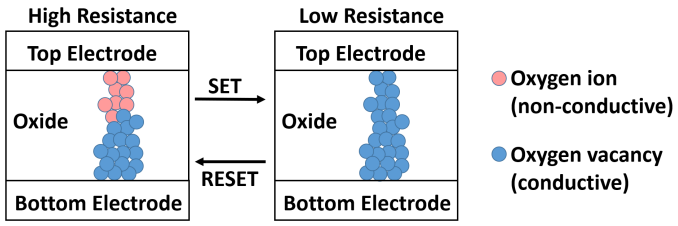


Fig. 12: RRAM technology.

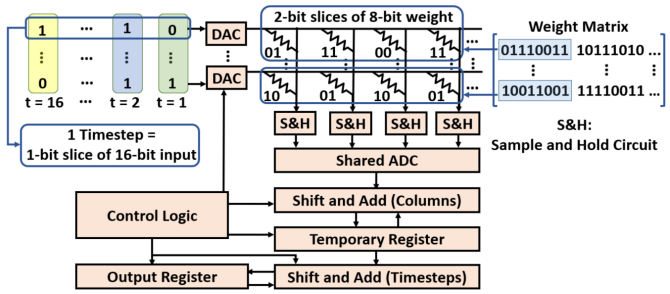


Fig. 13: RRAM-based CIM architecture with bit-slicing.

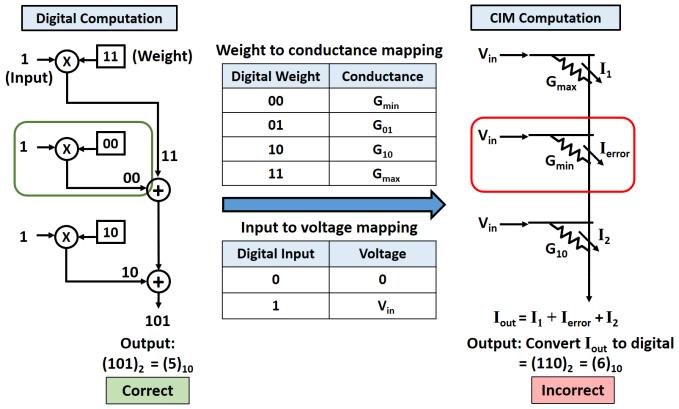


Fig. 14: Demonstration of non-zero G_{min} error.

Bit-slicing involves breaking down the full-precision neural network weights and inputs into smaller bit-size segments known as slices, illustrated in Figure. For example, 2-bit slices of an 8-bit weight are transformed into conductances and assigned to distinct columns in the RRAM crossbar. Simultaneously, 1-bit slices of a 16-bit input are converted into voltages and distributed across different time steps, sequentially applied to the crossbar. Column currents resulting from time-multiplexed voltage inputs are then converted to digital values. Through shift-and-add operations across both columns and time steps, the complete full-precision output is obtained. Nevertheless, such architecture faces the issue of finite conductance states (Non-zero G_{min} error) as well as conductance variations.

Non-Zero G_{min} error: In Bit-slicing CIM architectures, a zero weight-slice in the neural network is represented by an RRAM device with non-zero conductance equal to the minimum device conductance value (G_{min}). In the digital domain, multiplying any non-zero input with a zero weight must result in a zero output. However, when it mapped to CIM architecture (see Figure 14), a non-zero output current is produced when a non-zero input in the form of a voltage is applied to an RRAM with G_{min} conductance. This is known as non-zero Gmin error, which creates a functional mismatch between digital output and CIM output resulting in erroneous VMM operation and degraded neural network accuracy.

Conductance variations: The programmed conductance of a RRAM deviates from its target value due to the stochastic nature of oxygen vacancy creation/depletion and fabrication imperfections like variable oxide thickness [18]. This phenomenon is called conductance variation, shown in Fig. 3. It leads to incorrect weight storage as RRAM conductance, resulting in poor accuracy. In this paper, we improve the accuracy of RRAM-based neural network architectures in the presence of conductance variation.

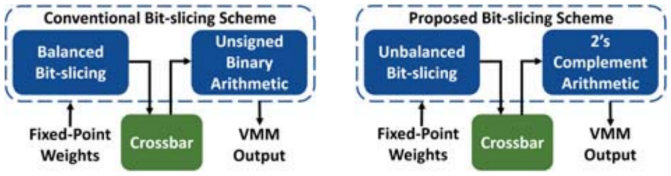


Fig. 15: Overview of conventional and proposed bit-slicing schemes [46].

C. Unbalanced bit-slicing scheme

A bit-slicing scheme consists of two fundamental components: 1) bit-slicing logic which determines how the slices are created, and 2) arithmetic which determines how the partial outputs from sliced columns are combined. The *balanced* bit-slicing (BBS) logic in state-of-the-art bit-slicing schemes provides low sensing margin resulting in significant impact of non-zero G_{min} errors, while unsigned binary arithmetic in these schemes leads to high accumulative non-zero G_{min} error on combining the partial outputs as shown in Figure 15.

We have proposed an unbalanced bit-slicing (UBS) scheme in [47] which changes the way in which neural network weights are mapped to bit-slicing CIM crossbar to mitigate the impact of non-zero G_{min} error. The proposed UBS scheme provides high sensing margin for important bits (MSBs) by using an RRAM with *n*-bit maximum capacity as an *m*-bit memory-cell (slice) where $m<n$. This provides sufficient sensing margin to the MSB column output and make them immune to non-zero G_{min} error as shown in Figure 16. This suffices for good accuracy due to the robustness of neural networks to minor computational fluctuations i.e. errors in less important bits (LSBs). Moreover, use of 2's complement arithmetic in [47] leads to reduction in accumulative non-zero G_{min} error after combining the partial outputs due to weighted subtraction as shown in Eq. 3a and Eq. 3b obtained using 8-bit weights with maximum 2 bits/RRAM as an example. The conductance subscripts indicate binary slice value. The digital outputs of the columns are denoted by D_i, while the accumulated digital outputs are indicated by D_f. $D_i{=}T_i+E_i$, where T_i is the ideal output value and E_i the error due to non-zero G_{min}. Similarly, $D_f{=}T_f+E_f$, where T_f is the ideal accumulated output value and E_f the accumulated error due to non-zero G_{min}.

$$E_f = 64{\cdot}E_1 + 16{\cdot}E_2 + 4{\cdot}E_3 + E_4 \quad (3a)$$

$$E_f = (-128){\cdot}E_1 + 64{\cdot}E_2 + 16{\cdot}E_3 + 4{\cdot}E_4 + E_5 \quad (3b)$$

Owing to the cumulative effect of high MSB sensing margin and 2's complement arithmetic on non-zero G_{min} error, unbalanced bit-slicing scheme achieves up to 8.8× and 1.8× classification accuracy compared to state-of-the-art CIM architectures for single-bit memristors and two-bit memristors respectively, at reasonable energy overheads arising due to extra columns.

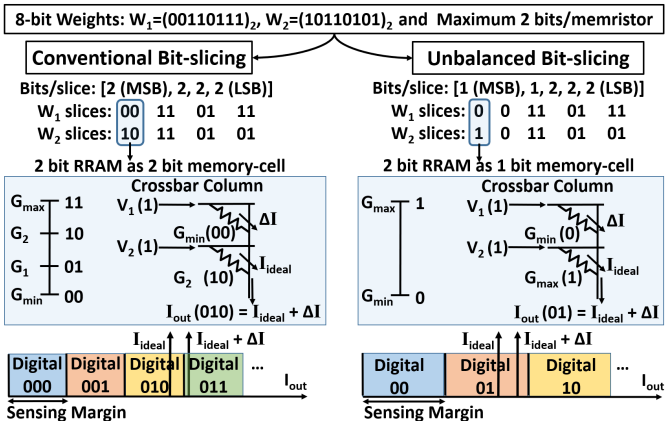


Fig. 16: Impact of sensing margin on bit-slicing schemes.

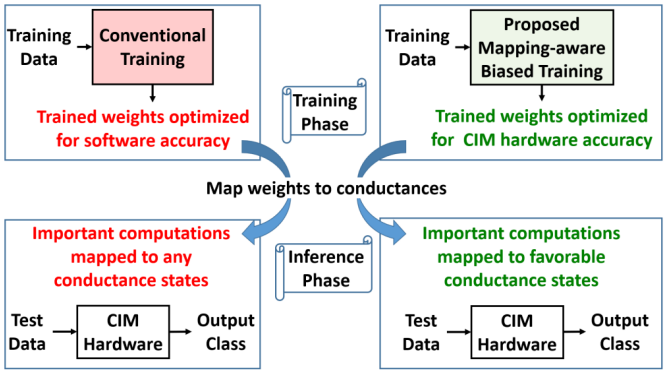


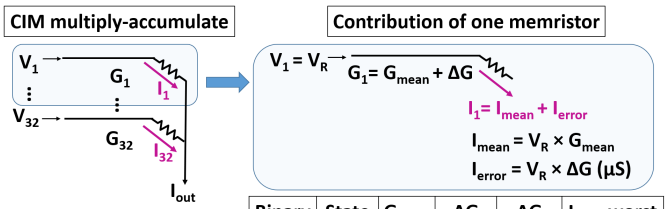
Fig. 18: Overview of the conventional and proposed biased training methodologies

D. Mapping-aware biased training methodology

Figure 17 shows a CIM-based multiply-accumulate operation, where I_{error} is the error current in a single RRAM device due to conductance variation. As small I_{error} is desirable, the preference order of states in Fig. 4 is: G00 (best), G01, G11, G10 (worst). Despite having a higher variation percentage, G00 and G01 are preferred over G11 and G10 as their small mean values result in small I_{error}. Hence, the preference order of conductance states must be based on I_{error} contribution instead of the variation percentage.

The deployment of a neural network on CIM hardware for inference involves two phases as shown in Figure 18: i) Training the neural network weights to obtain high classification accuracy. ii) Mapping the trained weights to RRAM device conductances for inference on CIM hardware. Conventional training can result in the mapping of weights to conductance states having a high variation impact (unfavorable states). This can lead to low hardware accuracy despite high software accuracy. A mapping-aware biased training, as proposed in [49], restricts the neural network weights during training, so that their post-training values directly get mapped to conductance states having a low variation impact (favorable states). However, restricting too many weights hinders backpropagation and leads to low software accuracy. This in turn results in low hardware accuracy, as it is upper bounded by software accuracy. Conversely, if too few weights are restricted, the hardware accuracy will be poor as many RRAM devices can get mapped to unfavorable states. Hence, our proposed mapping-aware biased training only restricts the important weights. This leads to high software accuracy due to the adaptability of non-important weights and also provides high hardware accuracy as important weights get mapped to favorable RRAM states.

We first train the neural network in a standard (hardware-unaware) manner. These weights are used as initial weights for mapping-aware biased training for faster convergence. We then determine a favorability constraint on the weights to ensure the mapping of desired weight bits to favorable conductance states. It depends on RRAM device bit capacity and CIM mapping scheme details like fixed-point format, underlying CIM architecture, etc. For example, consider 2-bit RRAM devices (slices), 8-bit fixed-point weights (6-bit fraction), and CIM architecture in [9]. The mapping scheme first converts trained weights to 2's complement fixed-point format. It then shifts the 2's complement weight range by 2^7 to overcome the difficulty in isolating the sign contribution from a multi-bit slice [9]. Figure 19 then shows the favorability constraint to map the most significant 2-bit slice to favorable states (G00, G01, and G10) for this example. We now perform a new epoch of backpropagation using training data and then determine



Binary value	State name	G_{mean} (μS)	ΔG (%)	ΔG (μS)	I_{error} worst case (μA)
00	G_{00}	2	38.5	0.8	1.6
01	G_{01}	67	24.8	16.6	6.6
10	G_{10}	134	17.2	23.1	9.2
11	G_{11}	200	9.3	18.6	7.4

Fig. 17: Favorable conductance states analysis for a 2-bit memristor (four conductance states). The used conductance variation data is obtained from [48].

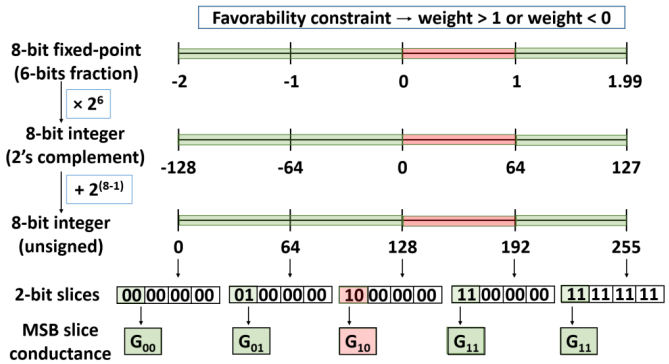


Fig. 19: Illustration of favorability constraint derivation for mapping MSB slice of 8-bit weight to favorable conductance states in a 2-bit RRAM device

which weights are important for high hardware accuracy. In CIM hardware design for the same network, instead of individual weights, some crossbar columns (groups of weights) are more important than others for high hardware accuracy. Weights in these columns are restricted as per the favorability constraint and test accuracy is evaluated. This process is repeated for a given number of biased training epochs and weights with the best post restriction test accuracy are mapped to CIM hardware. The proposed biased training achieves up to 2.4× hardware accuracy and up to 2.4× correct operations per unit energy compared to the conventional training, without incurring any hardware overhead. Such high accuracy and energy efficiency can facilitate the deployment of CIM-based neural networks for edge-AI.

VI. Conclusions

Deep neural networks demand a lot of computation efficiency, and the existing hardware is not suitable as they have separated processor units and memory units that can impose significant latency and bandwidth constraints, limiting the overall speed and effectiveness of deep neural network computations. Computation-in-memory has the potential to considerably enhance deep neural network efficiency by integrating processing and storage within the same module, thereby reducing latency, minimizing data transfer bottlenecks. In this paper, we have presented CIM architecture using three emerging technologies. Initially, a CIM structure incorporating Ferroelectric Field-Effect Transistors (FeFET) is introduced to calculate the Hamming distance function using an analog computing scheme. To test the error resiliency of the emerging HDC algorithm, the errors from the unreliable hardware have been modeled and injected into the inference step of HDC. Following this, we investigated a CIM system employing Vertical Nanowire Field-Effect Transistors (VNWFETs) within a 3D computing framework, demonstrating enhanced energy efficiency and reliability through CIM and approximate computing. Furthermore, we have presented the improvement in the precision of the Resistive Random Access Memories (RRAM) based CIM architecture by implementing two mapping-based solutions, namely the unbalanced bit-slicing scheme and the mapping-aware biased training methodology. These findings underscore the transformative potential of CIM architectures in overcoming current hardware limitations and advancing the capabilities of deep neural network computations.

Acknowledgment

We thank Paul R. Genssler for his work on Hyperdimensional Computing. This work was partly supported by the EU H2020 grant “DAIS” (grant agreement No. 101007273) and the EU HORIZON-JU-RIA grant ”NEUROKIT2E” (grant agreement No. 101112268), and by the FVLLMONTI European Union’s Horizon 2020 research and innovation programme under grant agreement No 101016776.

References

[1] C. Szegedy *et al.*, “Going deeper with convolutions,” in *IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[2] Google, Facebook and microsoft are remaking themselves around AI. [Online]. Available: https://www.wired.com/2016/11/google-facebook-microsoft-remaking-around-ai/

[3] S. Diware *et al.*, “Severity-based hierarchical ecg classification using neural networks,” *TBioCAS*, vol. 17, no. 1, pp. 77–91, 2023.

[4] Intel processor family. [Online]. Available: https://www.intel.in/content/www/in/en/products/processors/core.html

[5] NVIDIA turing architecture GPUs. [Online]. Available: https://www.nvidia.com/en-in/geforce/turing/

[6] N. Jouppi *et al.*, “TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings,” in *International Symposium on Computer Architecture*, 2023.

[7] M. A. Yaldagard and other, “Read-disturb detection methodology for rram-based computation-in-memory architecture,” in *International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2023, pp. 1–5.

[8] A. Ankit *et al.*, “PUMA: A Programmable Ultra-efficient Memristor-based Accelerator for Machine Learning Inference,” in *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2019.

[9] A. Shafiee *et al.*, “ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars,” in *International Symposium on Computer Architecture*, 2016.

[10] A. Singh *et al.*, “A 115.1 tops/w, 12.1 tops/mm 2 computation-in-memory using ring-oscillator based adc for edge ai,” in *International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2023, pp. 1–5.

[11] A. Singh, M. Fieback *et al.*, “Accelerating rram testing with a low-cost computation-in-memory based dft,” in *International Test Conference (ITC)*, 2022, pp. 400–409.

[12] A. Singh *et al.*, “Referencing-in-array scheme for rram-based cim architecture,” in *DATE*, 2022, pp. 1413–1418.

[13] A. V. Khvalkovskiy, D. Apalkov, S. Watts, R. Chepulskii, R. S. Beach, A. Ong, X. Tang, A. Driskill-Smith, W. H. Butler, P. B. Visscher, D. Lottis, E. Chen, V. Nikitin, and M. Krounbi, “Basic principles of STT-MRAM cell operation in memory arrays,” *Journal of Physics D: Applied Physics*, vol. 46, no. 7, p. 074001, feb 2013. [Online]. Available: https://iopscience.iop.org/article/10.1088/0022-3727/46/7/074001

[14] H. Akinaga and H. Shima, “Resistive random access memory (reram) based on metal oxides,” *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2237–2251, Dec 2010.

[15] S. Dünkel, M. Trentzsch, R. Richter, P. Moll, C. Fuchs, O. Gehring, M. Majer, S. Wittek, B. Müller, T. Melde, H. Mulaosmanovic, S. Slesazeck, S. Müller, J. Ocker, M. Noack, D. . Löhr, P. Polakowski, J. Müller, T. Mikolajick, J. Höntschel, B. Rice, J. Pellerin, and S. Beyer, “A fefet based super-low-power ultra-fast embedded nvm technology for 22nm fdsoi and beyond,” in *2017 IEEE International Electron Devices Meeting (IEDM)*, Dec 2017, pp. 19.7.1–19.7.4.

[16] D. Reis, M. Niemier, and X. S. Hu, “A computing-in-memory engine for searching on homomorphically encrypted data,” *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, pp. 1–1, 2019.

[17] J. G. Alzate, U. Arslan, P. Bai, J. Brockman, Y. J. Chen, N. Das, K. Fischer, T. Ghani, P. Heil, P. Hentges, R. Jahan, A. Littlejohn, M. Mainuddin, D. Ouellette, J. Pellegren, T. Pramanik, C. Puls, P. Quintero, T. Rahman, M. Sekhar, B. Sell, M. Seth, A. J. Smith, A. K. Smith, L. Wei, C. Wiegand, O. Golonzka, and F. Hamzaoglu, “2 mb array-level demonstration of stt-mram process and performance towards l4 cache applications,” in *2019 IEEE International Electron Devices Meeting (IEDM)*, Dec 2019, pp. 2.4.1–2.4.4.

[18] S. Beyer, S. Dünkel, M. Trentzsch, J. Müller, A. Hellmich, D. Utess, J. Paul, D. Kleimaier, J. Pellerin, S. Müller, J. Ocker, A. Benoist, H. Zhou, M. Mennenga, M. Schuster, F. Tassan, M. Noack, A. Pourkeramati, F. Müller, M. Lederer, T. Ali, R. Hoffmann, T. Kämpfe, K. Seidel, H. Mulaosmanovic, E. T. Breyer, T. Mikolajick, and S. Slesazeck, “Fefet: A versatile cmos compatible device with game-changing potential,” in *2020 IEEE International Memory Workshop (IMW)*, 2020, pp. 1–4.

[19] T. Mikolajick, S. Slesazeck, M. H. Park, and U. Schroeder, “Ferroelectric hafnium oxide for ferroelectric random-access memories and ferroelectric field-effect transistors,” *MRS Bulletin*, vol. 43, no. 5, p. 340–346, 2018.

[20] S. Thomann, P. R. Genssler, and H. Amrouch, “Hw/sw co-design for reliable tcam-based in-memory brain-inspired hyperdimensional computing,” *IEEE Transactions on Computers*, vol. 72, no. 8, pp. 2404–2417, 2023.

[21] L. Liu, Y. Guo, Y. Cheng, Y. Zhang, and J. Yang, "Generating robust dnn with resistance to bit-flip based adversarial weight attack," *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 401–413, 2022.

[22] A. Rahimi, P. Kanerva, and J. M. Rabaey, "A Robust and Energy-Efficient Classifier Using Brain-Inspired Hyperdimensional Computing," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2016.

[23] P. Kanerva, "Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors," *Cognitive computation*, vol. 1, pp. 139–159, 2009.

[24] D. Kleyko, E. Osipov, A. Senior, A. I. Khan, and Y. A. Şekerciogğlu, "Holographic graph neuron: A bioinspired architecture for pattern processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 6, pp. 1250–1262, 2017.

[25] G. Karunaratne, M. Le Gallo, G. Cherubini, L. Benini, A. Rahimi, and A. Sebastian, "In-memory hyperdimensional computing," *Nature Electronics*, vol. 3, 2020.

[26] P. R. Genssler and H. Amrouch, "Brain-inspired computing for wafer map defect pattern classification," in *IEEE International Test Conference*, 2021.

[27] M. Imani, D. Kong, A. Rahimi, and T. Rosing, "Voicehd: Hyperdimensional computing for efficient speech recognition," in *2017 IEEE International Conference on Rebooting Computing (ICRC)*, 2017.

[28] H. Fujisawa, K. Ikeda, and S. Nakashima, "Nonvolatile operation of vertical ferroelectric gate-all-around nanowire transistors," *Japanese Journal of Applied Physics*, vol. 60, no. SF, p. SFFB10, 2021.

[29] S. Mannaa, A. Poittevin, C. Marchand, D. Deleruyelle, B. Deveautour, A. Bosio, I. O'Connor, C. Mukherjee, Y. Wang, H. Rezgui, M. Deng, C. Maneux, J. Müller, S. Pelloquin, K. Moustakas, and G. Larrieu, "3-d logic circuit design-oriented electrothermal modeling of vertical junctionless nanowire fets," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 9, no. 2, pp. 116–123, 2023.

[30] S. Mannaa, C. Marchand, D. Deleruyelle, B. Deveautour, I. O'Connor, and A. Bosio, "Vnwfet-based technology: From device modelling to standard cell library," in *2023 IEEE 23rd International Conference on Nanotechnology (NANO)*, 2023, pp. 576–581.

[31] G. Larrieu and X.-L. Han, "Vertical nanowire array-based field effect transistors for ultimate scaling," *Nanoscale*, vol. 5, no. 6, pp. 2437–2441, 2013.

[32] A. Hamzah, R. Ismail, N. E. Alias, M. L. P. Tan, and A. Poorasl, "Explicit continuous models of drain current, terminal charges and intrinsic capacitance for a long-channel junctionless nanowire transistor," *Physica Scripta*, vol. 94, no. 10, p. 105813, 2019.

[33] C. Mukherjee, M. Deng, F. Marc, C. Maneux, A. Poittevin, I. O'Connor, S. Le Beux, C. Marchand, A. Kumar, A. Lecestre *et al.*, "3d logic cells design and results based on vertical nwfet technology including tied compact model," in *2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC)*. IEEE, 2020, pp. 76–81.

[34] C. Mukherjee, A. Poittevin, I. O'Connor, G. Larrieu, and C. Maneux, "Compact modeling of 3d vertical junctionless gate-all-around silicon nanowire transistors towards 3d logic design," *Solid-State Electronics*, vol. 183, p. 108125, 2021.

[35] A. B. Kahng, H. Lee, and J. Li, "Horizontal benchmark extension for improved assessment of physical cad research." Association for Computing Machinery, 2014, p. 27–32.

[36] A. Bosio, M. Cantan, C. Marchand, I. O'Connor, P. Fiser, A. Poittevin, and M. Traiola, "Emerging technologies: Challenges and opportunities for logic synthesis," in *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, 2021, pp. 93–98.

[37] W. Kim, A. Chattopadhyay, A. Siemon, E. Linn, R. Waser, and V. Rana, "Multistate memristive tantalum oxide devices for ternary arithmetic," *Scientific reports*, vol. 6, no. 1, p. 36652, 2016.

[38] M. Fieback *et al.*, "Testing scouting logic-based computation-in-memory architectures," in *ETS*, 2020, pp. 1–6.

[39] R. Bishnoi *et al.*, "Special session–emerging memristor based memory and cim architecture: Test, repair and yield analysis," in *VTS*, 2020, pp. 1–10.

[40] C. Bengel *et al.*, "Reliability aspects of binary vector-matrix-multiplications using reram devices," *Neuromorphic computing and engineering*, vol. 2, no. 3, p. 034001, 2022.

[41] M. Fieback *et al.*, "Defects, fault modeling, and test development framework for rrams," *JETC*, vol. 18, no. 3, pp. 1–26, 2022.

[42] A. Singh *et al.*, "Srif: Scalable and reliable integrate and fire circuit adc for memristor-based cim architectures," *TCAS-I*, vol. 68, no. 5, pp. 1917–1930, 2021.

[43] M. Mayahinia *et al.*, "A voltage-controlled, oscillation-based adc design for computation-in-memory architectures using emerging rerams," *JETC*, vol. 18, no. 2, pp. 1–25, 2022.

[44] A. Singh *et al.*, "Low-power memristor-based computing for edge-ai applications," in *ISCAS*, 2021, pp. 1–5.

[45] A. Gebregiorgis *et al.*, "Dealing with non-idealities in memristor based computation-in-memory designs," in *VLSI-SoC*, 2022, pp. 1–6.

[46] S. Diware, A. Gebregiorgis *et al.*, "Unbalanced bit-slicing scheme for accurate memristor-based neural network architecture," in *AICAS*, 2021.

[47] S. Diware *et al.*, "Accurate and energy-efficient bit-slicing for rram-based neural networks," *Transactions on Emerging Topics in Computational Intelligence*, vol. 7, no. 1, pp. 164–177, 2022.

[48] A. Prakash and H. Hwang, "Multilevel cell storage and resistance variability in resistive random access memory," *Physical Sciences Reviews*, vol. 1, no. 6, 2016.

[49] S. Diware *et al.*, "Mapping-aware biased training for accurate memristor-based neural networks," in *International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2023, pp. 1–5.