In-memory Computing Architectures for Sparse Distributed Memory

Mingu Kang, *Student Member,* and Naresh R. Shanbhag, *Fellow*, *IEEE*

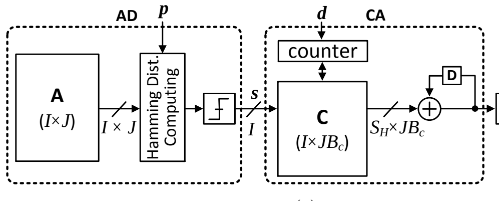
***Abstract*—This paper presents an energy-efficient and high-throughput architecture for Sparse Distributed Memory (SDM) - a computational model of the human brain [1]. The proposed SDM architecture is based on the recently proposed in-memory computing kernel for machine learning applications called *Compute Memory* (CM) [2], [3]. CM achieves energy and throughput efficiencies by deeply embedding computation into the memory array. SDM-specific techniques such as hierarchical binary decision (HBD) are employed to reduce the delay and energy further. The CM-based SDM (CM-SDM) is a mixed-signal circuit, and hence circuit-aware behavioral, energy, and delay models in a 65 nm CMOS process are developed in order to predict system performance of SDM architectures in the auto- and hetero-associative modes. The delay and energy models indicate that CM-SDM, in general, can achieve up to 25$\times$ and 12$\times$ delay and energy reduction, respectively, over conventional SDM. When classifying 16$\times$16 binary images with high noise levels (input bad pixel ratios: 15% ~ 25%) into nine classes, all SDM architectures are able to generate output bad pixel ratios (B_o) $\leq$ 2%. The CM-SDM exhibits negligible loss in accuracy, i.e., its B_o degradation is within 0.4% as compared to that of the conventional SDM.**

***Index Terms*—Associative memory, brain-inspired computing, *Compute Memory*, Machine learning, Pattern recognition, Sparse Distributed Memory**

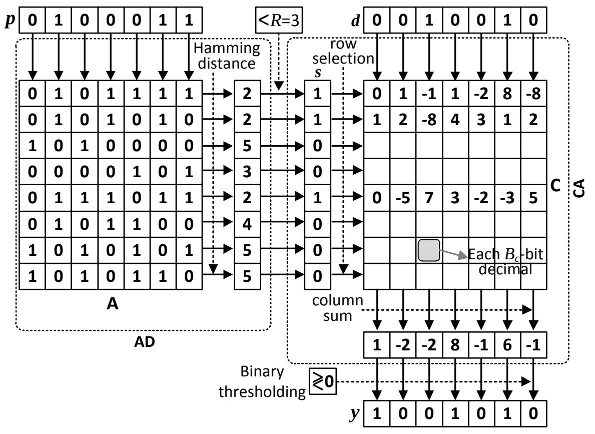
I. Introduction

Emerging applications require the processing of massive data volumes generated by sensor-rich platforms such as wearables, autonomous vehicles, robots, Internet-of-things, and others. These applications require the implementation of statistical signal processing, and machine learning kernels in silicon in order to provide in-situ data analytics capabilities. Such implementations need to be energy-efficient in order to operate with energy constrained sources, in a limited form factor, and with large data volumes. Therefore, there is much interest in exploring brain-inspired models of computation that can provide robust system behavior for inference applications while achieving high energy efficiency [4], [5], [6], [7].

The Sparse Distributed Memory (SDM) [1] (see Fig. 1) is one such computational model of the human brain. A SDM can be trained to remember sparse data vectors and retrieve these when presented with noisy or incomplete versions of the stored vectors. This is similar to human brain's ability to associate related memory given noisy sensory input by conceptualizing/categorizing incomplete information [8].



(a)



(b)

Fig. 1: Sparse distributed memory (SDM): (a) architecture (S_H: number of selected rows), and (b) example of SDM operation with *address decoder* (**AD**) and *counter array* (**CA**) ($I = 8$, $J = K = 7$, and $S_H = 3$).

Being a memory array, the SDM input is a 2-tuple $(\boldsymbol{p}, \boldsymbol{d})$, where $\boldsymbol{p}$ and $\boldsymbol{d}$ are the J-bit address and K-bit data, respectively (we assume $J = K$ in the rest of this paper). In a SDM, data vectors $\boldsymbol{d}$ are first stored (WRITE operation). The *address decoder* (**AD**) projects the J-bit address vector $\boldsymbol{p}$ on to a higher I-dimensional ($I \gg J$) space, and then uses this high dimensional representation $\boldsymbol{s}$ of $\boldsymbol{p}$ as the decoded address into the *counter array* (**CA**), where $\boldsymbol{d}$ is stored in a distributed fashion. In the READ mode, the address $\boldsymbol{p}$ is first decoded by the **AD**, and the decoded address $\boldsymbol{s}$ used to retrieve the stored data from the **CA**. The sparse and distributed nature of data processed and stored in a SDM provides inherent robustness to noise or imprecision in the

This work was supported by Systems on Nanoscale Information fabriCs (SONIC), one of the six SRC STARnet Centers, sponsored by SRC and DARPA.

The authors are with the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: mkang17@illinois.edu; shanbhag@illinois.edu).

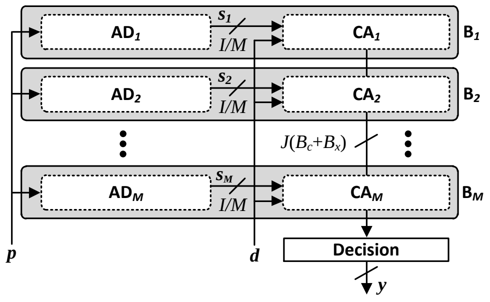
input data. The SDM can also be employed in an auto- or hetero-associative mode to achieve even greater robustness to data errors.

However, a straightforward SDM implementation will consume much energy and will be slow because the SDM operates in a high (hyper)-dimensional space [1], e.g., typical SDM parameters are: $I = 2 \times 10^3$ to 10^6, $J \geq 256$, $B_c \geq 5$, where B_c is a bit precision of each counter in the **CA** [8], [9]. Such an implementation in a 65 nm CMOS process would consume 77 uJ and have a delay of 2 ms per READ. In fact, the dominant (about 80% as shown in Section V) source of energy consumption and delay in the SDM can be attributed to the **AD**. Hence, several high throughput architectures for the **AD** based on SRAM and DRAM have been proposed. These achieve speed-up by parallelizing the **AD** using multiple memory blocks [10]. However, these architectures suffer from an inter-block throughput bottleneck. To remove memory read operation, a shift register-based **AD** architecture [11] has also been proposed. However, this architecture suffers from large dynamic energy consumption and occupies large area as compared to memory-based architectures. Mixed-signal **AD** implementations [11], [12] employ a current mirror to evaluate the Hamming distances in parallel thereby achieving high throughput. However, the large content addressable memory bit-cell dimension (i.e., 11 transistors including the current mirror) results in a loss of storage density, and the bias currents results in high DC power consumption. The design of SDM is also implemented by employing resistive memory devices [9].

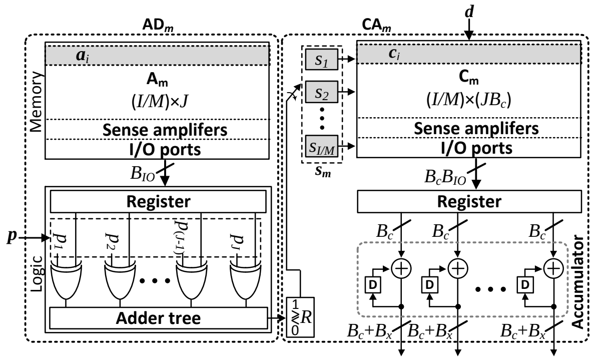
Implementing the SDM model requires large storage capacity closely integrated with computation. Traditional processor-memory architectures separate low-swing memory storage functionality from high-swing logic. This separation exists even in the so-called processor-in-memory architecture [13], [14], and is the source of both a throughput bottleneck and energy consumption. In fact, conventional architectures fail to exploit an important feature of the SDM [8] - the ability to compensate for hardware noise/errors in addition to noise/errors in the input data.

Recently, we proposed *Compute Memory* (CM) [2], [3] an in-memory computing architecture, where both computation and storage are implemented in a low swing/low signal-to-noise ratio (SNR) domain thereby eliminating the processor-memory interface completely, and providing a $5.0\times$ energy reduction and $4.9\times$ throughput enhancement for pattern recognition application in a 45 nm CMOS process. CM preserves the storage density, the conventional SRAM's read/write functionality, and is well-suited for inference kernels such as SDM which can compensate for non-deterministic hardware operations.

In this paper, (preliminary results are in [15]), we describe an architecture and circuit implementation of a CM-based SDM (**CM-SDM**), which incorporates two proposed techniques 1) CM-based **AD** (**CM-AD**), and 2) **CA** with a hierarchical binary decision (**CA-HBD**). Circuit and system



(a)



(b)

Fig. 2: The conventional SDM: (a) a M-parallel block architecture, and (b) architecture of a single block.

simulations in a 65 nm CMOS process show that the **CM-SDM** reduces energy and delay simultaneously by a factor of up to $25\times$ and $12\times$, respectively, over the conventional SDM architecture in the auto- and hetero-associative modes with negligible loss in accuracy.

The rest of paper is organized as follows. Section II provides the necessary background on SDM, CM, and associative memory. Section III describes the proposed **CM-AD** and **CA-HBD** architectures. Section IV develops circuit-aware energy, delay, and behavioral models for the entire signal processing chain. Section V presents circuit and system simulation results demonstrating the performance, delay reduction, and energy savings of the architecture over the conventional SDM.

II. Background

This section introduces the necessary background on the topics of SDM [1] and CM [2], [3].

A. Sparse Distributed Memory (SDM)

Figure 1 shows that the SDM accepts as input a 2-tuple $(\boldsymbol{p}, \boldsymbol{d})$ with a J-bit address $\boldsymbol{p}$ and J-bit data $\boldsymbol{d}$. The SDM architecture (see Fig. 1(a)) includes: 1) a J-bit *address decoder* **AD** to evaluate the Hamming distance between $\boldsymbol{p}$ and the I, J-bit addresses stored in a $I \times J$ memory array **A** in the **AD**, and 2) a *counter array* **CA** with a counter and a memory array **C** to store the IJ B_c-bit counts.

1) WRITE Operation: During the WRITE operation, the **AD** generates an I-bit decoded row address $\boldsymbol{s} = [s_1, s_2, \dots, s_I]$ (see Fig. 1), as follows:

$$s_i = sgn\{R - \sum_{j=1}^{J}(a_{ij} \oplus p_j)\}, \; (i = 1, 2, \dots, I) \quad (1)$$

$$sgn(x) = \begin{cases} 1, \; if\; x \geq 0 \\ 0, \text{ otherwise} \end{cases}$$

where $\oplus$ is the binary EXOR operator, $\boldsymbol{a}_i = [a_{i1}, a_{i2}, \dots, a_{iJ}]$ is the i-th address stored in $\mathbf{A}$, $\boldsymbol{p} = [p_1, p_2, \dots, p_J]$ is the input address, and R is a user-defined radius/threshold.

The decoded row address $\boldsymbol{s}$ and the data $\boldsymbol{d}$ are employed in the **CA** to update the count, as follows:

$$c_{ij} \leftarrow \begin{cases} c_{ij} + s_i, & if\; d_j = 1 \\ c_{ij} - s_i, & \text{otherwise} \end{cases} \quad (2)$$

where $\boldsymbol{d} = [d_1, d_2, \dots, d_J]$. Note: that the contents of $\mathbf{C}$ are updated only when $s_i = 1$, i.e., only the selected rows are updated.

2) READ Operation: During the READ operation, **AD** generates the row address $\boldsymbol{s}$ in the same manner as in the WRITE operation described by (1). Then, the SDM output is read out as:

$$y_j = sgn(\boldsymbol{s} \cdot \boldsymbol{c}_j), \; (j = 1, 2, \dots, J) \quad (3)$$

where $\boldsymbol{c}_j$ is the j-th column vector of $\mathbf{C}$, and $\boldsymbol{y} = [y_1, y_2, \dots, y_J]$ is the output word.

B. Associative Memory

In associative memories, the data read is the stored data that is most strongly associated with the contents of the input rather than a specific address. There are two types of associative memories: 1) auto-associative memory, and 2) hetero-associative memory. The SDM can operate in both modes of associative recall and when it does, the SDM exhibits even stronger robustness to noise/errors in data.

In the auto-associative mode, the SDM is trained by selecting its input 2-tuple $(\boldsymbol{p}, \boldsymbol{d})$ from the training set $\boldsymbol{S}_t = \{(\boldsymbol{t}_1, \boldsymbol{t}_1), (\boldsymbol{t}_2, \boldsymbol{t}_2), \dots\}$, i.e., both the address $\boldsymbol{p}$ and the data $\boldsymbol{d}$ are assigned the same value [16]. On the other hand, in the hetero-associative mode, the SDM is trained by selecting its input 2-tuple $(\boldsymbol{p}, \boldsymbol{d})$ from the training set $\boldsymbol{S}_t = \{(\boldsymbol{t}_{11}, \boldsymbol{t}_{12}), (\boldsymbol{t}_{21}, \boldsymbol{t}_{22}), \dots\}$, i.e., $\boldsymbol{p}$ and the $\boldsymbol{d}$ are assigned different values.

During the classification/decision-making phase, in both associative modes, the SDM is operated in an iterative manner where initially $\boldsymbol{p} = \boldsymbol{l}$, where $\boldsymbol{l}$ is a noisy/incomplete version of the stored data. Then, in subsequent iterations, $\boldsymbol{p}$ is set to the current output of the SDM. Thus, the classification phase of the SDM is described as follows:

$$\boldsymbol{p}[n] = \begin{cases} \boldsymbol{l}, & if\; n = 1 \\ \boldsymbol{y}[n-1], & if\; n > 1 \end{cases} \quad (4)$$

where n is the time index, and $\boldsymbol{l}$ is an initial input. The output $\boldsymbol{y}[n]$ converges to an error-free/closest version of $\boldsymbol{l}$ that was stored during the training phase. The SDM's auto- and hetero associative modes can be interpreted as human brain's ability to extract a pattern from noise and locate the next pattern in a certain sequence given the current pattern [8].

C. Conventional SDM Architecture

The conventional SDM architecture is multi-block [10] (see Fig. 2(a)) in order to enhance throughput. The multi-block SDM architecture comprises M blocks ($\mathbf{B}_1$, $\mathbf{B}_2$,. . .,$\mathbf{B}_M$) that operate in parallel, where each block has its own *address decoder* $\mathbf{AD}_m$ with memory array $\mathbf{A}_m$ of size $(I/M) \times J$ bits, a *counter array* $\mathbf{CA}_m$ with memory array $\mathbf{C}_m$ of size $(I/M) \times (JB_c)$ bits, and decoded row address $\boldsymbol{s}_m$ $(m = 1, \dots, M)$. The memory array $\mathbf{A}$ in **AD** is implemented via SRAMs for high throughput, while the memory array $\mathbf{C}$ in **CA** is implemented using DRAM, Flash, and PRAM, in order to achieve high storage densities.

The architecture of each block (see Fig. 2(b)) indicates that $\mathbf{AD}_m$ computes the Hamming distance between the input address $\boldsymbol{p}$ and (I/M) stored addresses in $\mathbf{A}_m$, while $\mathbf{CA}_m$ generates a partial sum, which is then accumulated and thresholded during the READ operation. The each partial sum requires an additional B_x bits in addition to B_c bits per single counter in order to prevent overflow, where B_x depends upon the sparsity of stored data and R. Thus, the multi-block SDM architecture requires $J(B_c + B_x)$ global bit-lines (GBLs) per block to transfer partial sums to the decision block.

The conventional architecture in Fig. 2(a) has a number of drawbacks. Key among these are:

1) The Hamming distance computation in the $\mathbf{AD}_m$ requires access to all the memory locations. The throughput of $\mathbf{AD}_m$ is limited by the SRAM read out bandwidth. In particular, multiple read out cycles are required to read a single $\boldsymbol{a}_i$ in conventional memory and processor architecture.
   This is because conventional SRAMs need to employ column multiplexing, whereby multiple bit-lines (BLs) share a single sense amplifier (SA). Reliability constraints force the SA and other peripheral circuits to be designed with area that is 4×-to-8× of that of a bit-cell, thereby necessitating column multiplexing. Additionally, there is another throughput bottleneck due to limited memory I/O port or bus width in von Neumann architectures [18]. Typically, $J/B_{IO} \geq 4$ read outs are required to read the entire data in single row even in application processors with custom-designed on-chip SRAM [17], where B_{IO} is the bit width of the SRAM I/O port or the bus width.
2) The additional digital blocks in the **AD** such as the adder tree and EXOR gates lead to energy consumption and area overhead.
3) Routing the GBLs in the $\mathbf{CA}_m$ is made difficult because of their large number ($J(B_c + B_x)$ per block),

and because of the small bit-cell area ($4F^2$, F: $1/2$ of BL pitch) due to the use of high density memories [19], [20].

Section III describes how these drawbacks of the conventional architecture can be overcome.

D. Compute Memory (CM)

The recently proposed CM implements data processing functionality of inference algorithms into the periphery of the memory array, e.g., SRAM. The CM incorporates the following features: 1) a *multi-row read* (MR-READ) process where multiple rows are read per BL precharge, and 2) *BL analog signal processing* (BL-ASP) that leverages the differential nature of the SRAM read-out circuitry to implement simple operations such as sum of absolute differences, signed multiplication, and others. CM achieves energy efficiency and high throughput by implementing the bulk of processing (both MR-READ and BL-ASP) in low-swing/low-SNR domain and by eliminating the memory-processor interface completely. Only the final decision is sense amplified to full-swing digital. CM does not modify the core bit-cell array and thus is able to maintain the storage density.

III. Proposed Architecture

In this section, a CM-based SDM (**CM-SDM**) is proposed (see Fig. 3(a)) to address the drawbacks of conventional architecture listed in Section II-C. In particular, **CM-SDM** employs the following key techniques:

- The **AD** is designed using CM (**CM-AD**) (see Fig. 3(b)) in order to overcome its bandwidth limitation and eliminate the use of digital logic.
- The **CA** is implemented using a hierarchical binary decision (HBD) technique (**CA-HBD**) as shown in Fig. 3(c) in order to minimize the routing overhead of GBLs.

A. Compute Memory-based Address Decoder (CM-AD)

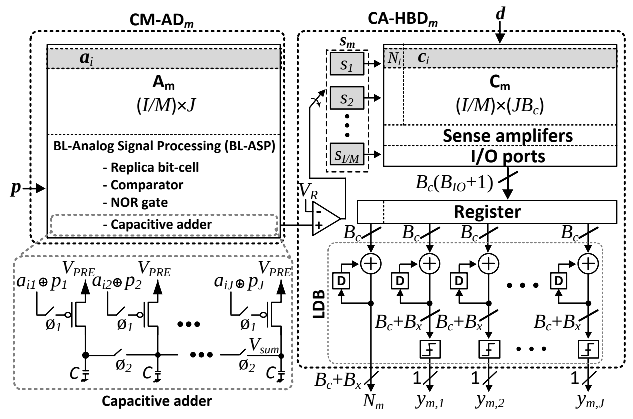
The proposed **CM-AD** generates the Hamming distance per (1) via a three-step process: 1) MR-READ process generates BL voltages V_{BL} and V_{BLB} that are proportional to the sum $a_{ij} + p_j$ over the field of real numbers, followed by 2) the use of BL-ASP to compute $a_{ij} \oplus p_j$ and 3) finally the Hamming distance (via a capacitive adder). These steps are described next.

The MR-READ step begins with the application of access pulses simultaneously to the rows storing a_{ij} and p_j such that the pulse width $T \ll R_{BL}C_{BL}$, where $R_{BL}C_{BL}$ is the RC time constant of BL/BLB [2]. This results in a BL/BLB voltage (see Fig. 4) given by:

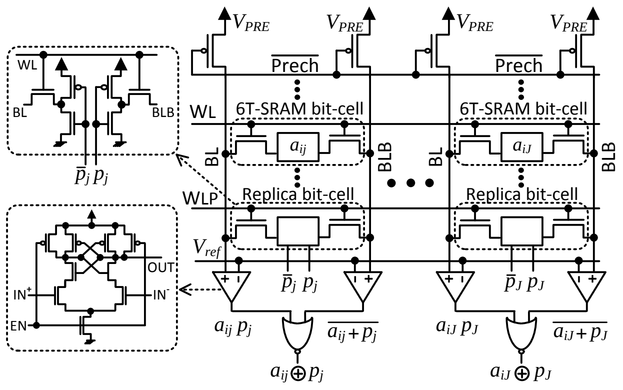
$$V_{BL} = V_{PRE} - (\overline{a}_{ij} + \overline{p}_j)\Delta V_{BL} \quad (5)$$

$$V_{BLB} = V_{PRE} - (a_{ij} + p_j)\Delta V_{BL} \quad (6)$$

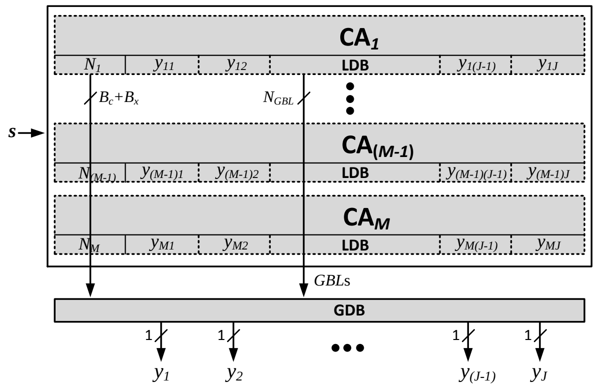
where $\Delta V_{BL} = V_{PRE}(T/R_{BL}C_{BL})$. A replica bit-cell is employed to avoid writing $\boldsymbol{p}$ into the main array $\mathbf{A}$ (see Fig. 3(b)).



(a)



(b)



(c)

Fig. 3: Proposed SDM architecture (**CM-SDM**): (a) architecture of single block, (b) **AD** with CM (**CM-AD**) including deeply embedded mixed signal processing units, and (c) **CA** with hierarchical binary decision (**CA-HBD**) (N_{GBL}: number of GBLs).

The second step (BL-ASP) begins with the BL/BLB provided as inputs to differential comparators [21] sized to fit within a single bit-cell pitch with an appropriately selected reference voltage $V_{ref} = V_{PRE} - \Delta V_{BL}/2$. Doing so results in binary valued comparator outputs:

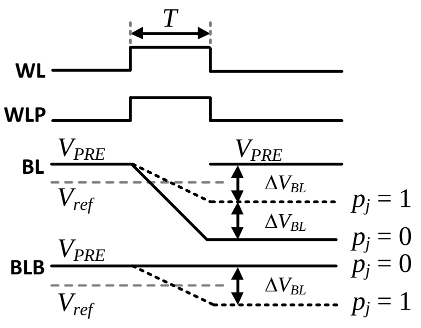


Fig. 4: Multi-row read (MR-READ) for EXOR operation in **CM-AD** when $a_{ij} = 0$.

$$X_{BL} = a_{ij}p_j = sgn(V_{diff,BL}) = sgn\{0.5 - (\overline{a}_{ij} + \overline{p}_j)\} \quad (7)$$

$$X_{BLB} = \overline{a_{ij} + p_j} = sgn(V_{diff,BLB}) = sgn\{0.5 - (a_{ij} + p_j)\}$$

A NOR2 gate that combines the comparator outputs generates $a_{ij} \oplus p_j$ as follows:

$$a_{ij} \oplus p_j = \overline{sgn\{0.5 - (a_{ij} + p_j)\} + sgn\{0.5 - (\overline{a}_{ij} + \overline{p}_j)\}} \quad (8)$$

Next, a J-bit capacitive adder (see Fig. 3(a)) accepts $a_{ij} \oplus p_j$ from the NOR2 gate output and employs charge redistribution to compute the summation in (1) as follows:

$$V_{sum,i} = \frac{1}{J}\sum_{j=1}^{J}(1 - a_{ij} \oplus p_j)V_{PRE} \quad (9)$$

The last step involves an analog comparator that generates the decoded address bit s_i as shown below:

$$s_i = sgn(V_{sum,i} - V_R) \quad (10)$$

This sequence of operations is repeated I/M times.

Thus, **CM-AD** reads $\boldsymbol{a}_i$ in single read cycle (single precharge) and has $\approx J/B_{IO}$ times higher throughput as compared to the conventional **AD**. Additionally, **CM-AD** is more energy-efficient than a digital implementation because the capacitive adder employs small capacitances (i.e., $C = 10\,\text{fF}$) and requires a simple switching operation.

B. Counter Array using Hierarchical Binary Decision (CA-HBD)

The proposed **CA-HBD** architecture minimizes the inter-block data transfer as shown in Fig. 3(c), where **GDB** and **LDB** are global and local decision blocks, respectively. The **CA-HBD** architecture requires to record the row access count N_i, i.e., the number of accesses to each physical address $\boldsymbol{a}_i$ in **A** during the WRITE operation. The row access count N_i is recorded in an additional column in the **CA**.

During the READ operation, the **LDB** of the m^{th} block generates a local binary decision $y_{m,j}$ and N_m as follows:

$$N_m = \sum_{i\in H_m} N_i \quad (11)$$

$$y_{m,j} = sgn(\sum_{i\in H_m} c_{ij}) \quad (12)$$

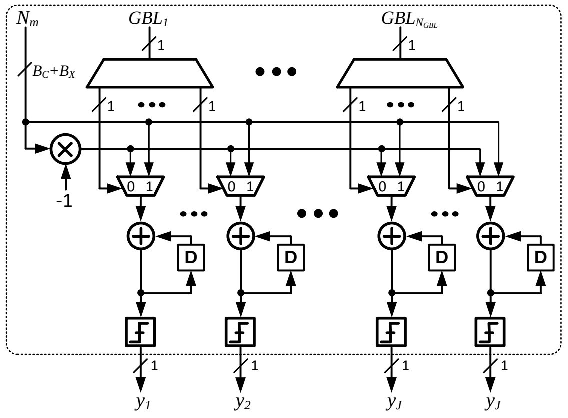


Fig. 5: Global decision block (**GDB**) to incorporate local decisions ($y_{m,j}$) with impact factor N_m.

where H_m is the set of row indices in the m^{th} block $\textbf{CA}_m$ that were selected during READ, and N_m represents the sum of the row access counts for these rows.

Finally, the **GDB** (see Fig. 5) generates the final SDM output bit y_j as follows:

$$y_j = sgn\{\sum_{m=1}^{M} sign(y_{m,j})N_m\}, \text{ where} \quad (13)$$

$$sign(x) = \begin{cases} 1, \; if\; x > 0 \\ -1, \; otherwise \end{cases}$$

Thus, the **GDB** weights $\textbf{CA}_m$'s contribution $y_{m,j}$ by N_ms in order to assign more weight to those blocks which were accessed more frequently during the WRITE phase. In this manner, the **LDB** transmits compressed information to the **GDB** (Note: that $y_{m,j}$s are binary numbers), and thus $\approx J$-bits are required instead of $J(B_c + B_x)$-bits as shown in Fig. 3(c), thereby minimizing the delay and the energy penalty for the data transfer.

IV. Circuit-aware Behavioral, Energy, and Delay Models

The analog-intensive MR-READ and BL-ASP operations of the **CM-AD** are intrinsically vulnerable to various sources of noise due to its low-SNR operation. The dominant sources of noise in the **CM-AD** are: 1) local transistor threshold voltage V_{th}-variation across bit-cells caused by random dopant fluctuations, and 2) input offset of the analog comparator. This section derives behavioral models of the non-ideal behavior of **CM-AD** to predict system performance. Energy and delay models are also provided.

A. Behavioral Model

In the MR-READ operation, the V_{th} variations were modeled as a Gaussian distributed random variable in [2]. In this paper, two binary numbers a and p (we omit indices i and j

TABLE I: Design parameters for SDM.

Parameter	Value	Parameter	Value
$V_{DD}(=V_{PRE})$	1 V	M	4~2048
I/M	512	J	256
$B_c(=B_x)$	4	B_{IO}	8~64
Clock frequency	1 GHz	N_{GBL}	256
C	10 fF	C_{BL}	230 fF

for simplicity) are MR-READ. The impact of V_{th}-mismatch on the BL/BLB voltages is modeled as follows:

$$f_{V_{BL}}(V_{BL};a,p) = \mathcal{N}(V_{PRE} - (\overline{a}+\overline{p})\Delta V_{BL}, (\overline{a}+\overline{a})\sigma^2_{cell})$$
$$f_{V_{BLB}}(V_{BLB};a,p) = \mathcal{N}(V_{PRE} - (a+p)\Delta V_{BL}, (a+p)\sigma^2_{cell}) \quad (14)$$

where $f_{V_{BL}}(V_{BL};a,p)$ and $f_{V_{BLB}}(V_{BLB};a,p)$ are the probability density functions of V_{BL} and V_{BLB}, respectively, parametrized by a and p. $\mathcal{N}(\mu,\sigma^2)$ is the normal distribution with mean μ, and variance σ^2, and σ^2_{cell} is the variance of ΔV_{BL} due to V_{th} variation across the storage array **A**. It is assumed that V_{th} variations for the bit- and replica cells are identical.

The comparator outputs X_{BL} and X_{BLB} are obtained as:

$$X_{BL} = \begin{cases} 0 & if\, V_{BL} < V_{ref} + V_{offset} \\ 1 & otherwise \end{cases}$$
$$X_{BLB} = \begin{cases} 0 & if\, V_{BLB} < V_{ref} + V_{offset} \\ 1 & otherwise \end{cases}$$
$$f(V_{OS}) = \mathcal{N}(0, \sigma^2_{comp}) \quad (15)$$

where an input offset voltage (V_{offset}) of the comparator is modeled as a zero mean Gaussian random variable with variance σ^2_{comp}.

The charge injection noise in the switches and thermal noise/mismatch of capacitors in the capacitive adder are made negligible by ensuring $C > 10\,\text{fF}$ [22]. The single comparator at the output of capacitive adder can be designed to have a small input offset by using large transistor sizes and calibration techniques.

The behavioral models in this section are validated in Fig. 8 of Section V.

B. Delay and Energy Models

The delay per READ of the conventional SDM and the **CM-SDM** are described as follows:

$$\begin{aligned} T_{SDM} &= T_{AD} + T_{CA} \quad (16) \\ T_{AD} &= (I/M)(J/B_{IO})T_{read} \\ T_{CA} &= S_{H,max}(J/B_{IO})T_{read} \\ &\quad + M\left\lceil J(B_c+B_x)/N_{GBL}\right\rceil T_{GBL} \end{aligned}$$

$$\begin{aligned} T_{CM-SDM} &= T_{CM-AD} + T_{CA-HBD} \quad (17) \\ T_{CM-AD} &= (I/M)T_{read} \\ T_{CA-HBD} &= S_{H,max}(J/B_{IO})T_{read} \\ &\quad + M\left\lceil J/N_{GBL}\right\rceil T_{GBL} \end{aligned}$$

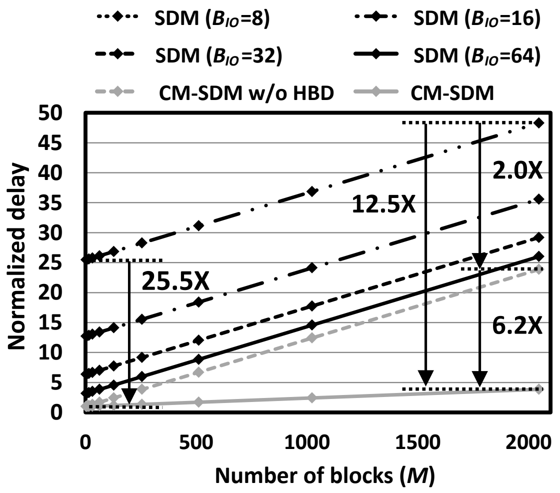


Fig. 6: Normalized delay of single READ operation with $B_{IO} = 8 \sim 64$ ($B_{IO} : J = 1 : 4 \sim 32$).

where T_{AD} (T_{CM-AD}) and T_{CA} (T_{CA-HBD}) are the delay for **AD** (**CM-AD**) and **CA** (**CA-HBD**). In addition, T_{read} is the delay for a single read access for memory arrays **A** and **C**, T_{GBL} is the delay in transferring a single bit via the GBLs, N_{GBL} is the number of GBLs, and $S_{H,max}$ is the maximum number of selected addresses per block. It is assumed that all other blocks are operating in parallel while the memories are being accessed. Hence, the delay of blocks such as the logic blocks in the conventional **AD** and the capacitive adder in **CM-AD** are not included in (16)-(17). The factors (I/M) and (J/B_{IO}) in T_{AD} are equal to the number of rows and the number of read outs per row, respectively, in **AD**$_m$. The partial sums per block are transferred serially through N_{GBL} GBLs, thus requiring $\lceil J(B_c+B_x)/N_{GBL}\rceil$ cycles.

The throughput enhancement of **CM-SDM** over SDM derives from: 1) $J/B_{IO} \geq 4$ in T_{AD}, and 2) $B_c + B_x \geq 8$ in T_{CA}. The delay models in (16)-(17) are plotted in Fig. 6, where it is assumed that T_{read} and T_{GBL} take two clock cycles. **CM-SDM** demonstrates a 25× smaller delay compared to SDM with $B_{IO} = 8$ due to high bandwidth of **CM-AD** when $M = 4$. The benefit of HBD at $M = 2048$ is evident as there is a 3.2× additional delay reduction as compared to **CM-SDM** without HBD.

The energy consumption per READ of the conventional SDM and the **CM-SDM** are modeled as follows:

$$\begin{aligned} E_{SDM} &= E_{AD} + E_{CA} \quad (18) \\ E_{AD} &= I[(J/B_{IO})(E_{PRE}+E_{leak}) + JE_{SA} + E_{logic}] \\ E_{CA} &= S_H B_c[(J/B_{IO})(E_{PRE}+E_{leak}) + JE_{SA}] \\ E_{PRE} &= JC_{BL}\Delta V_{BL}V_{PRE} \\ E_{leak} &= IJP_{leak\_cell}T_{read} \end{aligned}$$

$$\begin{aligned} E_{CM-SDM} &= E_{CM-AD} + E_{CA-HBD} \quad (19) \\ E_{CM-AD} &= I(2E_{PRE} + E_{leak} + 2JE_{comp} + E_{a\_add}) \\ E_{CA-HBD} &< E_{CA} \end{aligned}$$

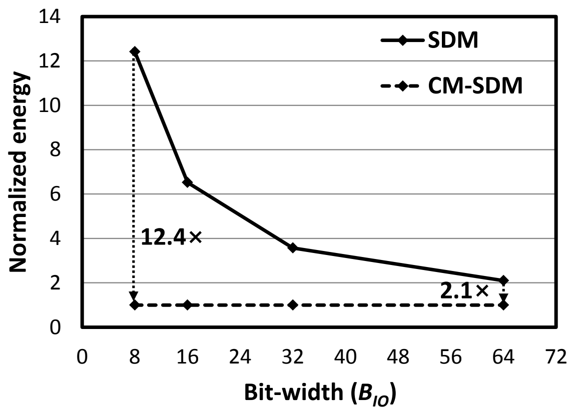


Fig. 7: Normalized energy based on models (18), (19) with $\Delta V_{BL} = 75\,\mathrm{mV}$ and $125\,\mathrm{mV}$ (obtained from Fig. 10) for SDM and **CM-SDM**, respectively.

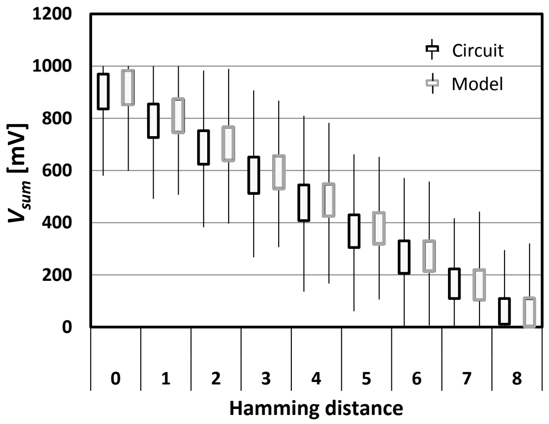


Fig. 8: V_{sum} from circuit simulations and system simulations with behavioral models (14) to (15) with $J = 8$ and $\Delta V_{BL} = 50\,mV$.

where E_{AD} (E_{CM-AD}) and E_{CA} (E_{CA-HBD}) are the energy consumptions of the **AD** (**CM-AD**) and **CA** (**CA-HBD**), respectively. E_{PRE} and E_{leak} are the energy consumptions of the precharge and bit-cell leakage for the entire memory array **A**, respectively, and E_{SA} (E_{comp}) is for a single unit of sense amplifier (analog comparator). P_{leak_cell} is the leakage power of each bit-cell. The energy consumptions of the analog capacitive adder in the **CM-AD** and logic blocks in the conventional **AD** per single Hamming distance computation are denoted by E_{a-add} and E_{logic}, respectively. Energy consumptions from other blocks such as WL drivers [23], **CA**'s decision blocks are assumed to be negligible. It is assumed that the **AD** and **CA** can be placed into a deep sleep mode independently [17]. Note that $E_{AD} \gg E_{CA}$ because $I \gg S_H B_c$. The E_{CM-AD} has a scaling factor of two for the first and third terms as **CM-AD** reads a_{ij} and p_j, and employs two comparators per bit-cell column.

The energy efficiency of **CM-AD** derives from the fact that: 1) the first term in E_{AD} and E_{CM-AD} is the largest, and because $J/B_{IO} \geq 4$, 2) $E_{a\_add} \ll E_{logic}$ as the capacitances in the capacitive adder are very small, e.g., $10\,\mathrm{fF}$, and the capacitive adder requires only simple switching operations, and 3) leakage energy in E_{CM-AD} is smaller than that in E_{AD} because the high-throughput (see delay models (16)-(17)) of **CM-SDM** permits it to be placed into a deep sleep mode much quicker than SDM [17].

The energy models (18) and (19) using typical design parameters from Table I are plotted in Fig. 7. The component values of (18) and (19) obtained from Fig. 11. Figure 7 indicates that **CM-SDM** achieves energy reductions of 2.1× to 12.4× over SDM.

V. Simulation Results

In this section, we apply SDM for hand-written digit recognition. Monte Carlo circuit (HSPICE) simulations in $65\,\mathrm{nm}$ CMOS process technology are employed to validate the models in (14) and (15). These models are employed in system simulations to estimate the output bad pixel ratio (B_o). Energy and throughput benefits are demonstrated via circuit simulations and the energy/throughput models (16), (17), (18), and (19).

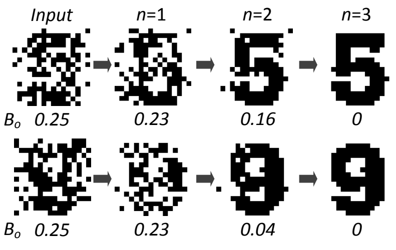
A. System Configuration

Nine 16×16 binary shapes of numbers from 1 to 9 are employed to generate $\boldsymbol{p}$ and $\boldsymbol{d}$ as shown in Fig. 9(a) and (b). For each of the nine patterns, 225 noisy copies with input bad pixel ratio $B_i = 0.25$ are generated by randomly flipping 25% of the bits. These images form the training data set $\boldsymbol{S}_t$ of size $255 \times 9 = 2295$ and are written into the SDM in the auto-associative mode ($\boldsymbol{p} = \boldsymbol{d}$).

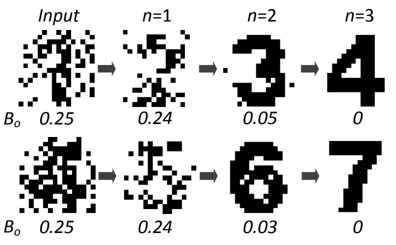
In hetero-associative mode, during the training phase, $\boldsymbol{p}$ and $\boldsymbol{d}$ are assigned images corresponding to consecutive numbers, e.g., if $\boldsymbol{p}$ is assigned image corresponding to 4 then $\boldsymbol{d}$ is assigned the image corresponding to 5. Thus, during the READ operation, the SDM retrieves the image corresponding to the number that is one greater than the input number, as shown in Fig. 9(b).

After the training, 100 contaminated copies of each pattern (total of 900 inputs) with $B_i = 0.15,\ 0.25,\ 0.3$ are generated and provided as the address $\boldsymbol{p}$ for classification. Four READ iterations of the auto- and hetero-associative memory are performed. The error immunity against faulty hardware increases with larger R in (1) as each data is distributed across greater number of physical addresses. On the other hand, R needs to be small enough not to create excessive intersection between physical addresses for different number's images. To balance this trade-off, R for WRITE and READ operations are set to 79 and 82, respectively.

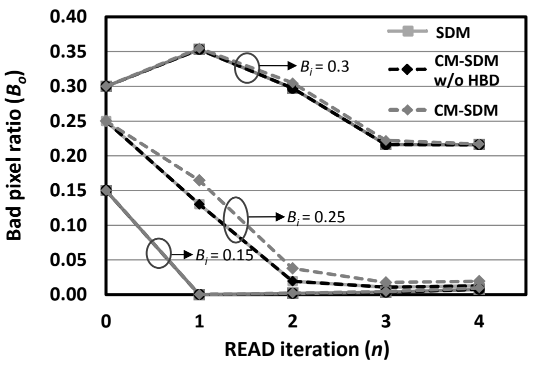
The block size $I/M = 512$ is chosen to balance the read out delay and area efficiency of memory. The value of $B_{IO} = 64$ is chosen as its typical value of J/B_{IO} ranges from 4 to 32 in conventional SRAM architectures [17], in order to permit the maximum bandwidth. In this specific application,



(a)



(b)



(c)

Fig. 9: Behavior of SDM, **CM-SDM** (without HBD) and **CM-SDM** with $M = 4$ in: (a) auto-associative mode, (b) hetero-associative mode, and (c) the output bad pixel ratio $B_o[n]$ in the auto-associative mode.

$M = 4$ and $S_H \approx 0.1I$, which will be used in the rest of this section. The design parameters (other than B_{IO} and M) used in the simulations are summarized in Table I.

B. Model Validation

Monte-Carlo circuit simulations show that $\sigma_{cell}/\Delta V_{BL} = 6.5\%$ and the analog comparator has an input offset $\sigma_{comp} = 18\,\mathrm{mV}$.

The behavioral models of the entire analog signal processing chain from the bit-cell to the final output V_{sum} are validated as shown in Fig. 8, where the results of Monte-Carlo circuit simulations are compared with those from system simulations employing the behavioral models (14)-(15). Nine different combinations of $\boldsymbol{p}$ and $\boldsymbol{a}_i$ (with $J = 8$) are chosen as inputs. Figure 8 indicates that the maximum modeling error is 4.5% of the dynamic range of V_{sum}. This level of accuracy is sufficient for system performance

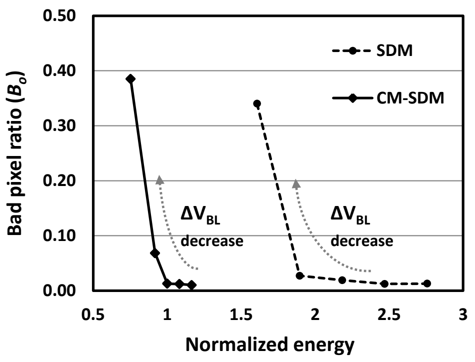


Fig. 10: Energy vs. B_o trade-off with $n = 4$. Here $\Delta V_{BL} = 25\,\mathrm{mV}$~$125\,\mathrm{mV}$ for SDM and $75\,\mathrm{mV}$~$175\,\mathrm{mV}$ for **CM-SDM**.

estimation, as it is much less than the smallest non-zero Hamming distance.

C. System Performance

The output bad pixel ratio $B_o[n]$ in the n-th READ iteration is computed as:

$$B_o[n] = \frac{1}{900J}\sum_{k=1}^{900} H_k[n] \tag{20}$$

where $H_k[n]$ is the Hamming distance between the SDM output $\boldsymbol{y}_k[n]$ at time index n and the ideal output for the k^{th} input image.

Figure 9(c) shows that the conventional SDM, the **CM-SDM** (without HBD) and the **CM-SDM**, all converge to achieve a B_o less than 2% for $n \geq 3$ when $B_i \leq 25\%$. Similar results were observed for the hetero-associative mode as well. Furthermore, SDM and **CM-SDM** were found to achieve $B_o[n]$ that were within $< 5\%$ from each other for $n \leq 3$ (the B_o of **CM-SDM** is slightly worse), for all three values of B_i. The B_o of **CM-SDM** was higher than SDM by only 0.4% for $n = 4$ and $B_i = 25\%$ indicating that the non-ideal behavior of **CM-SDM** is successfully compensated by the inherent noise immunity of SDM and the associative mode of operation. The B_o degradation of the **CM-SDM** can be reduced by increasing the number of blocks M with large I so that more averaging can occur.

D. Delay and Energy Savings

The conventional SRAM read access and MR-READ require two clock cycles, and the data transfer from the LDB to the GDB also requires two cycles. The proposed **CM-SDM** achieves $3.1\times$ smaller delay over SDM as shown in Fig. 6 due to high bandwidth of **CM-AD** with $M = 4$.

The various components of the energy models in (18) and (19) are measured via HSPICE simulations. To do so, the parasitic capacitance of BL ($C_{BL} = 230\,\mathrm{fF}$) is extracted from the layout of an SRAM bit-cell. These energy components are

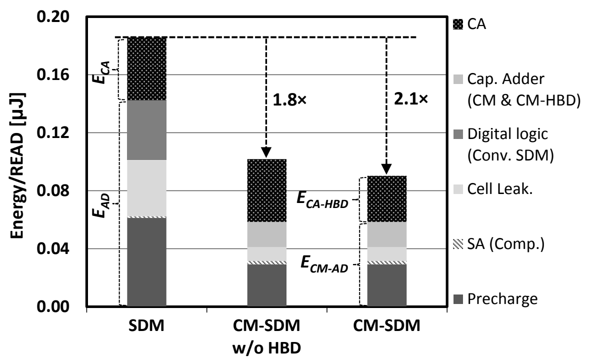


Fig. 11: Energy breakdown for a single READ operation with $B_{IO} = 64$, and $\Delta V_{BL} = 75\,\mathrm{mV}$ for SDM and $125\,\mathrm{mV}$ for **CM-SDM**.

a function of the BL swing ΔV_{BL}. The intrinsic robustness of SDM and the associative mode of operation enable a lower value of ΔV_{BL} to be employed as compared to a typical value in standard SRAM, thereby resulting in even greater energy savings.

Figure 10 shows the trend of B_o with ΔV_{BL} scaling, and the $B_o > 2\%$ when $\Delta V_{BL} < 75\,\mathrm{mV}$ and $125\,\mathrm{mV}$ in the conventional SDM and **CM-SDM**, respectively. Thus, energy-optimal ΔV_{BL}s are applied to both conventional and **CM-SDM** in order to obtain the energy breakdowns in Fig. 11. This figure shows that **CM-SDM** achieves approximately 2.1× reduced energy as compared to SDM.

VI. Conclusion

SDM provides great potential to address stochastic and unreliable behavior of nanoscale fabrics due to its inherent robustness. However, not many SDM architectures have been proposed due to the challenges in achieving high throughput and energy-efficiency. An in-memory computing platform, *Compute Memory* can be a possible solution to such memory-intensive algorithms/applications. The **CM-SDM** achieves aggressive energy-efficiency and high throughput by allowing hardware error/noise introduced by low-SNR operation of CM and approximate decisions from HBD. The non-ideal behavior of CM and HBD is successfully tolerated by inherent robustness of SDM and its associative memory operation. The benefits of **CM-SDM** increases with data volume in big data applications and with high density memory devices. Extensions to SDM architectures based on emerging memory topologies are potential future directions.

References

[1] P. Kanerva, *Sparse Distributed Memory*. Cambridge Massachusetts: MIT press, 1988.
[2] M. Kang, M.-S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An energy-efficient VLSI architecture for pattern recognition via deep embedding of computation in SRAM," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 8326–8330.
[3] M. Kang, S. K. Gonugondla, M.-S. Keel, and N. R. Shanbhag, "An energy-efficient memory-based high-throughput VLSI architecture for Convolutional Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2015.
[4] J. M. Cruz-Albrecht, M. W. Yung, and N. Srinivasa, "Energy-efficient neuron, synapse and STDP integrated circuits." *IEEE Trans. Biomedical Circuits and Systems*, vol. 6, no. 3, pp. 246–256, 2012.
[5] S. Brink, S. Nease, P. Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, "A learning-enabled neuron array IC based upon transistor channel models of biological phenomena," *IEEE Trans. Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 71–81, 2013.
[6] S. Ramakrishnan, R. Wunderlich, J. Hasler, and S. George, "Neuron array with plastic synapses and programmable dendrites," *IEEE Trans. Biomedical Circuits and Systems*, vol. 7, no. 5, pp. 631–642, 2013.
[7] V. Garg, R. Shekhar, and J. G. Harris, "Spiking neuron computation with the time machine," *IEEE Trans. Biomedical Circuits and Systems*, vol. 6, no. 2, pp. 142–155, 2012.
[8] P. J. Denning, *Sparse distributed memory*. Research Institute for Advanced Computer Science (NASA Ames Research Center), 1989.
[9] E. Lehtonen, J. H. Poikonen, M. Laiho, and P. Kanerva, "Large-scale memristive associative memories," *IEEE Trans. VLSI Syst.*, vol. 22, no. 3, pp. 562–574, 2014.
[10] M. Lindell et al., "Configurable Sparse Distributed Memory hardware implementation," in *IEEE Int. Symp. Circuits and Systems. (ISCAS)*, 1991, pp. 3078–3081.
[11] J. Saarinen et al., "VLSI architectures of Sparse Distributed Memory," in *IEEE Int. Symp. Circuits and Systems. (ISCAS)*, 1991, pp. 3074–3077.
[12] J. D. Keeler et al, "Notes on implementation of Sparsely Distributed Memory," in *NASA Research Institute for Advanced Computer Science*, August 1986.
[13] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, "Intelligent RAM (IRAM): Chips that remember and compute," in *IEEE Int. Solid-State Circuits Conf. (ISSCC)*, February 1997, pp. 224–225.
[14] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz, "Smart Memories: A modular reconfigurable architecture," in *ACM Proc. Int. Symp. Computer Architecture*, June 2000, pp. 161–171.
[15] M. Kang, E. P. Kim, M.-S. Keel, and N. R. Shanbhag, "Energy-efficient and high throughput Sparse Distributed Memory architecture," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2015.
[16] S.-I. Chien, I.-C. Kim, and D.-Y. Kim, "Iterative autoassociative memory models for image recalls and pattern classifications," in *IEEE International Joint Conference on Neural Networks (IJCNN)*, 1991, pp. 30–35.
[17] M. Yamaoka et al, "A 300-MHz 25-μA/Mb-leakage on-chip SRAM module featuring process-variation immunity and low-leakage-active mode for mobile-phone application processor," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 186–194, 2005.
[18] J. Backus, "Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs," *Communications of the ACM*, vol. 21, no. 8, pp. 613–641, 1978.
[19] I. Kim et al, "High performance PRAM cell scalable to sub-20nm technology with below $4F^2$ cell size, extendable to DRAM applications," in *IEEE Symp. VLSI Technology (VLSIT)*, 2010, pp. 203–204.
[20] S. Aritome, "Advanced Flash memory technology and trends for file storage application," in *Int. Electron Devices Meeting. (IEDM)*, 2000, pp. 763–766.
[21] T. Kobayashi, K. Nogami, T. Shirotori, and Y. Fujimoto, "A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architecture," *IEEE J. Solid-State Circuits*, vol. 76, no. 5, pp. 863–867, 1993.
[22] A. Verma and B. Razavi, "Frequency-based measurement of mismatches between small capacitors," in *IEEE Custom Integrated Circuits Conference (CICC)*, September 2006, pp. 481–484.
[23] K. Kim, H. Mahmoodi, and K. Roy, "A low-power SRAM using bit-line charge-recycling technique," in *Int. Symp. Low Power Electronics and Design. (ISLPED)*, 2007, pp. 177–182.