BIMAX: A Bitwise In-Memory Accelerator using 6T-SRAM Structure

1st Nezam Rohbani
Barcelona Supercomputing Center and School of Computer Science, Institute for Research in Fundamental Sciences (IPM)
nrohbani@bsc.es

2nd Mohammad Arman Soleimani
Department of Computer Engineering, Sharif University of Technology
arman.soleimani@sharif.edu

3rd Behzad Salami
Barcelona Supercomputing Center
behzad.salami@bsc.es

4th Osman Unsal
Barcelona Supercomputing Center
osman.unsal@bsc.es

5th Adrian Cristal Kestelman
Barcelona Supercomputing Center
adrian.cristal@bsc.es

6th Hamid Sarbazi-Azad
Department of Computer Engineering, Sharif University of Technology and Institute for Research in Fundamental Sciences (IPM)

***Abstract*—In-memory computing (IMC) paradigm reduces costly and inefficient data transfer between memory modules and processing cores by implementing simple and parallel operations inside the memory subsystem. SRAM, the fastest memory structure in the memory hierarchy, is an appropriate platform to implement IMC. However, the main challenges of implementing IMC in SRAM are the limited operations and unreliable accuracy due to environmental noise and process variations. This work proposes a low-latency, energy-efficient, and noise-robust IMC technique, called Bitwise In-Memory Accelerator using 6T-SRAM Structure (BIMAX). BIMAX performs parallel bitwise operations (i.e., (N)AND, (N)OR, NOT, X(N)OR) as well as row-copy with the capability of writing the computation result back to a target memory row. BIMAX functionality is based on an imbalanced differential sense amplifier (SA) that reads and writes data from and into multiple 6T-SRAM cells. The simulations show BIMAX performs these operations with 52.7% lower energy dissipation compared to the state-of-the-art IMC technique, with 5.7% average higher performance rate. Furthermore, BIMAX is about 5.4× more robust against environmental noises compared to the state-of-the-art.**

***Index Terms*—SRAM, In-Memory Computing, Process Variation, Noise, Sense Amplifier**

I. Introduction

The transfer of data between memory subsystems and processing elements can have a significant impact on both latency and power consumption, a phenomenon referred to as the memory wall effect [4]. To combat this effect, promising solutions have emerged in the form of in-memory computing (IMC) and near-memory computing (NMC) [4], [21], [23]. IMC adds computational capabilities within the memory by making slight modifications to the memory structure, typically at the level of memory arrays [13]. In contrast, NMC uses lightweight processing circuits inside the memory chip or employs 2.5 D or 3 D die stacking. One of the key advantages of IMC over NMC is the ability to utilize the large cell access bandwidth within memory arrays by accessing multiple cells in parallel. However, the dense structure of memory layouts limit IMC operations and make writing results back to the target cells costly.

Regarding diverse features of memory technologies, various IMC techniques have been proposed in the literature, such as those utilizing DRAM, non-volatile memories (NVMs), and SRAM [1]. Out of all memory technologies, SRAM is considered the fastest due to its easy-interfacing and symmetrical read and write latencies [7]. These properties make SRAM a suitable candidate for integration with IMC capabilities [26].

The previously proposed techniques for computing using SRAM structure can be grouped into three categories. Analog in-SRAM computing: Techniques in this category use charge/current sharing for in-memory computing [3], [6], [13], [14], [16], [22], [24]. 1- In these techniques, any voltage level from *VDD* to *GND* is acceptable on the bitlines, depending on the operands values. The focus of these techniques is typically multiply and accumulate (MAC), which is the main operation in neural network applications. Another work in this category uses 8T- and 9T-SRAM cells with an area overhead of approximately 30% to 50% to perform bitwise in-memory operations [2]. One of the main challenges with analog in-SRAM computing is its susceptibility to noise, especially at nano-scale technology sizes, as these techniques rely on small voltage perturbation on bitlines. 2- Semi-digital in-SRAM computing: In these techniques, the permissible voltage levels (noise margin) on the bitlines are the same as those of conventional SRAM [1], [5], [11]. The focus of IMC in this category is bitwise operations. These techniques are more robust against environmental noise compared to analog in-SRAM computing, but the number of implementable operations is limited. 3- Near-SRAM computing: Techniques in this category involve the insertion of simple logic gates [9], [12] or more complex circuits [15], [17], such as adders and multipliers, inside the SRAM structure. These techniques provide a trade-off between robustness against noise and the number of implementable operations, compared to the other two categories. The techniques of this category impose considerable area overhead and are not capable of utilizing the huge internal bandwidth of the SRAM.

This work proposes a Bitwise In-Memory Accelerator using 6T-SRAM Structure (BIMAX), which falls under the semi-digital in-SRAM computing category. BIMAX is capable of performing various bitwise operations, including (N)AND, (N)OR, NOT, X(N)OR, row-copy, and writing back the computation result to any target row in the SRAM array. The functionality of BIMAX is based on a low-power and robust imbalanced differential sense amplifier (SA). The proposed SA can detect equal voltages on its inputs, which is not possible in conventional differential SA. Unlike conventional active-load current mirroring SA, widely used in SRAM memory arrays, the proposed SA can directly write back the computation result to the operational memory cells without the need for extra write drivers. Simulation results

978-3-9826741-0-0/DATE25/© 2025 EDAA

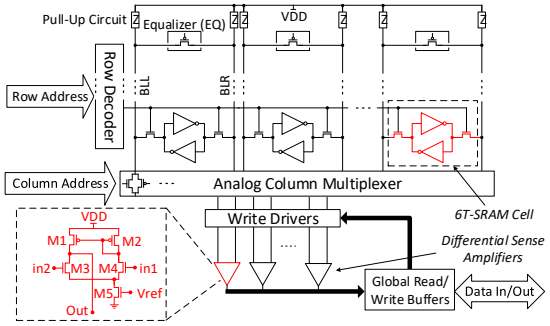

Fig. 1. 6T-SRAM array structure. Active-load current mirroring SA, generally utilized in SRAM architecture, is shown in the dashed box.

indicate that BIMAX consumes 52.7% less energy, on average, for executing bitwise operations on SRAM rows compared to the previous IMC techniques. BIMAX also outperforms the fastest previously proposed IMC technique by 5.7%. Additionally, its robustness against environmental noises is 5.4× greater than the most robust IMC technique in the same category.

II. Preliminaries

6-TSRAMs are an ideal choice for IMCs due to their rapid memory access, straightforward interfacing, minimal read/write latency, non-destructive readout, and the existence of a mature technology base [7], [20], [26]. Figure 1 illustrates the configuration of a 6T-SRAM cell array and its peripherals. SRAM arrays typically consist of 256×256 up to 1024×1024 cells [1], [7]. Each SRAM cell is composed of two inverters connected in a cross-coupled configuration, creating a positive feedback that holds one bit of data. Two NMOS access transistors connect a cell to the bitline left (*BLL*) and bitline right (*BLR*) in a pair for read/write access.During a read operation, the bitlines are precharged to roughly *VDD* by activating the pull-up circuit (often a P-type power transistor), and their voltage is equalized using an equalizer PMOS transistor (*EQ* in Figure 1). Then, the bitlines are disconnected from each other and VDD, and the selected wordline is activated with a *VDD* pulse of about 150 ps width [7]. This briefly connects all cells in the selected row to the corresponding bitline pairs through the access transistors [8]. Based on the stored data, a voltage perturbation (V_s) of 0.1-0.3 V is developed on the bitline connected to the SRAM cell's internal node with *GND* voltage, while the other bitline's voltage in the pair stays unchanged. It is important to note that the parasitic capacitance of the bitlines (C_{BL}) is very high, around 60 fF for a 512×512 cell memory array [18], due to the long metal line loaded with drain parasitic capacitance from the access transistors of the cells in each column. The voltage of selected bitline pairs is transferred to differential SAs through an analog multiplexer.

Active-load current mirroring SA is a popular choice for SRAM memory arrays due to its simple structure and robust operation. As shown in the dashed box in Figure 1, the active-load current mirroring SA consists of four transistors (*M1-M4*). The *M5* transistor operates as a current source, regulated by the reference voltage V_{ref}, that keeps the current flowing through *M3* and *M4* constant. The gate-to-source voltage V_{GS} of *M1* and *M2* is equal, acting as active loads, and the current flowing through them is the same. When there is a slight difference in the input voltages, *in1* or *in2*, the voltage at the drain of the transistor with a lower input voltage increases, as it cannot sink the current being pumped to its drain through the active-load transistor. This voltage can then be stored in the read/write buffer after being amplified to a suitable level. Long bitlines attract significant

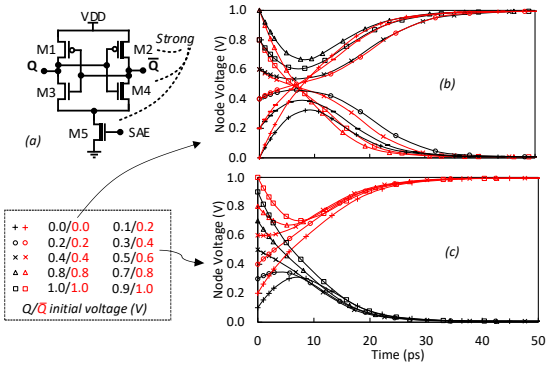

Fig. 2. Proposed imbalanced differential SA, composed of two cross-coupled inverters with different strengths (a), stabilization of the SA with equal initial voltages on inputs (b), and stabilization of the SA with 0.1 V difference on SA internal nodes voltages (c).

environmental noise, so differential sensing of the charge on the bitlines is essential for reliable operation.

For a write operation, the binary input is transformed into a full-differential signal on the I/O lines. These lines then drive the selected bitlines through the analog column multiplexer [7], [21]. The analog multiplexer transmission gates and write drivers are stronger than the cell transistors, allowing the data on the bitline pair(s) to be overwritten to the connected bistable SRAM cell(s) that are activated by the row decoder. The width of the column multiplexer determines the number of bits that can be processed during a read or write operation, with typical widths ranging from 1 to 8 bits [7].

III. Proposed IMC Technique

This section presents a low-power and robust IMC technique referred to as Bitwise In-Memory Accelerator using 6T-SRAM Structure (BIMAX). This approach leverages an imbalanced differential SA to access data in a 6T-DRAM array. This SA is capable of identifying states where bitlines in a pair carry the same voltage, for example, both *VDD*. This SA has the ability to update the bitlines directly, eliminating the requirement for additional write drivers.

A. Imbalanced Differential Sense Amplifier for 6T-SRAM

Active-load current mirroring SAs are used to detect tiny voltage perturbations (voltage-drop) on bitlines in SRAM arrays. These SAs are uni-directional SAs (separated input and output). Furthermore, this type of SA is unable to detect a state where both inputs' voltages are the same (This state never happens in a conventional SRAM). If this state occurs for this type of SA, ambient noise, process variation, or previous values on the bitlines from the former write/read accesses determine the output value.

The core of the IMC structure in this work is an imbalanced differential SA that can detect equal voltages on bitlines in each pair in a 6T-SRAM array, with the capability of updating bitlines voltage (write operation). Figure 2-(a) depicts the structure of the proposed SA. *M2* and *M4* transistors construct a stronger inverter, and *M1* and *M3* build a weaker inverter in a cross-coupled positive feedback of the proposed SA. *M5* disconnects the cross-coupled inverters from GND, thus the internal nodes of the SA can be floated by deactivating *M5*. Two possible initial conditions on the internal nodes of the SA, Q and $\overline{Q}$, are acceptable:

1) As shown in the Figure 2-(b), both internal nodes of the SA have the same starting point (before SA activation). In this figure, six equal starting points for the SA are simulated (see Section IV), from 0.0 V to 1.0 V. In equal voltages on Q and $\overline{Q}$ at the initial state, the node Q (black lines) rises/drops to the closest voltage

of *VDD* or *GND*, and the node $\overline{Q}$ (red lines) flips to the opposite voltage after activation of the SA. This is because *M2* and *M4* overpower *M1* and *M3*, as they are wider (stronger) transistors with higher drain current.

2) In the other case, as shown in Figure 2-(c), the voltage on Q and $\overline{Q}$ are not equal initially. In this case, the node with higher voltage, e.g. $\overline{Q}$, rises to *VDD* and the opposite node drops to *GND*. It is worth mentioning that, since in SRAM array the bitlines in each pair connecting to Q and $\overline{Q}$, always have an initial voltage of *VDD* or very close to *VDD* (*VDD*-V_s), the SA is not required to disconnect from VDD rail during the initial (floating) state.

It is important to note that the asymmetric differential SA presented in X-SRAM [2] is designed for 9T-SRAM cells and lacks the ability to directly write back the stored value in the SA to a target memory cell. Further details on the differences between the proposed imbalanced SA and the SA presented in X-SRAM [2] are discussed at the end of this section.

B. BIMAX: A Bitwise In-Memory Accelerator using 6T-SRAM Structure

The functionality of BIMAX is based on single bitline update in bitline pairs, together with imbalanced differential sense amplifying. Figure 3 shows the general structure of BIMAX. The 6T-SRAM cells in this structure are utilized as the storage cells which can operate as normal SRAM cells in the memory subsystem. The difference between the utilized cells with normal 6T-SRAM cells is that each access transistor can be controlled separately with WL_L and WL_R.

One of the inverters in the SA is stronger than the other, distinguished by a hatched fill, which comprises an imbalanced SA. The positive feedback in the SA amplifies voltage perturbation on one of the bitlines and translates it to '0' or '1' at the output. In the case of no voltage perturbation on bitlines, the node Q of the SA rises to *VDD* and the node $\overline{Q}$ drops to *GND*. The SA is connected to the bitlines through four access transmission gates to read the initial voltage on the bitlines and write the computation result back to the target SRAMs. BIMAX is capable of executing (N)AND, (N)OR, NOT, X(N)OR, and row-copy.

1) (N)AND and (N)OR Operations: BIMAX is capable of performing bitwise operations with an unlimited number of operands. For (N)AND operation, both bitlines in a pair are precharged to *VDD*. Then, the first operand cell node Q is shortly connected to *BLL*, by activating one of the access transistors, Figure 3-(a); Active components are shown in red in this figure. This is while $\overline{Q}$ does not connect to *BLR*. If the cell contains '0' (Q = *GND*), voltage drop occurs on the precharged *BLL*. Otherwise, *BLL* voltage remains untouched. After generating the required voltage perturbation on the connected bitline, the access transistor is deactivated.

The identical procedure is applied to the second operand cell, Figure 3-(b). In this manner, the unlimited number of other operands for (N)AND operation can be read in serial. During these activations, *BLR* voltage remains untouched (*VDD*); this is while *BLL* voltage drops, if even one of the operands contains '0'. During operands cells access, bitlines are connected to the imbalanced differential SA through *TG1* and *TG2* for the SA initialization. The SA is deactivated in this time period by disconnecting it from the GND.

After initialization of the SA, it is disconnected from bitlines by deactivating *TG1* and *TG2*, and the SA is activated for the stabilization phase, Figure 3-(c). The SA stabilizes with the result of AND operation of input operands, and the binary result is latched in the SA (node Q of the SA). In the case that all accessed

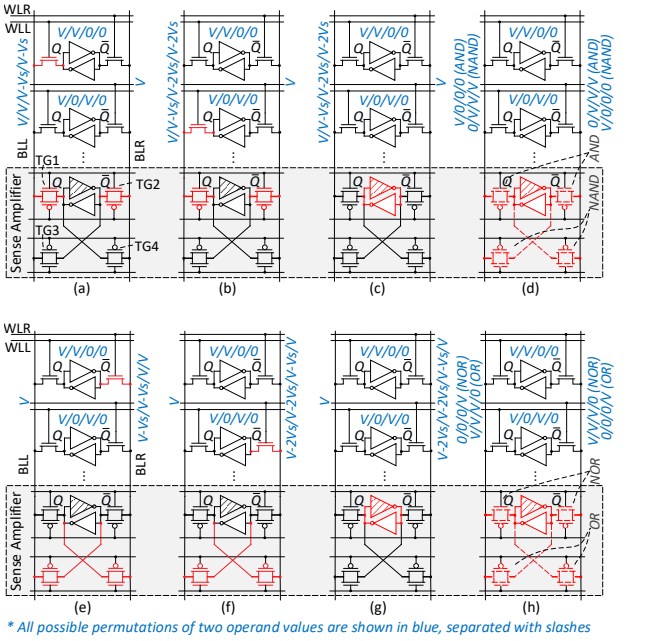

Fig. 3. BIMAX architecture and operations steps. For AND/OR operation, the first operand cell is connected to *BLL*/*BLR* (a/e), then the second operand is connected to *BLL*/*BLR*, while *TG1* and *TG2*/*TG3* and *TG4* are activated (b/f). The SA is disconnected from the bitlines in the next step and activated for the stabilization (c/g). In the end, the result can be written back to the target cell by activating *TG1* and *TG2* for AND or NOR/activating *TG3* and *TG4* for NAND or OR operations (d/h).

operands contain '1' (*VDD* at node Q), no voltage perturbation occurs on the bitlines, and both bitlines carry *VDD* charge. Consequently, both internal nodes of the SA remain charged to *VDD*. By activating the SA, node Q of the SA remains *VDD* (after a short glitch, see Figure 2-(b)) while node $\overline{Q}$ promptly drops to *GND*. In the other case, if even one of the connected operands is '0' (*GND* at node Q of the operand cell), a voltage drop occurs on node Q of the SA, which after activation of the SA, node Q stabilizes to *GND* and $\overline{Q}$ voltage remains *VDD* after a short glitch. see Figure 2-(c).

After stabilization of the SA, by activating *TG1* and *TG2*/*TG3* and *TG4*, AND/NAND operation result can be stored in the target cell, Figure 3-(d). Any target row can be selected to write the result. To accomplish this, the stabilized SA is connected to the bitlines first to discharge the parasitic capacitance of one of the bitlines to *GND* and hold (charge) the voltage on the other bitline to *VDD*. After discharging/charging the bitlines, the target cell row is activated to capture the result.

For (N)OR operation the procedure is more or less the same as (N)AND operation; however, the right access transistors to operand cells are activated and connect the operand cells to *BLR*. In contrast with (N)AND operation, *TG3* and *TG4* are activated to set the starting point of the SA. See Figure 3-(e,f). In this scenario, the only case that both bitlines carry the same value (i.e. *VDD*) is when both/all operands contain '0' (*VDD* on $\overline{Q}$ of both/all operand cells). In this scenario, the initial state of the SA is the same as the case of all operands containing '1' in (N)AND operation, and node Q of the SA remains *VDD*, while $\overline{Q}$ drops to *GND*. Otherwise, Q voltage drops to *GND* and $\overline{Q}$ voltage remains *VDD*, Figure 3-(g). By activating *TG1* and *TG2*/*TG3* and *TG4*, after the stabilization of the SA, NOR/OR operation result can be stored in the target cell (Figure 3-(h)).

2) Copy and NOT Operations: For copy/NOT operation, the source (operand) cell is connected to *BLL* and *BLR* by activating

the corresponding access transistors, together with activating *TG1* and *TG2* to initialize the SA. After initialization and activation of the SA, the SA node corresponding to the accessed cell node containing '0' drops to *GND*, and the other node of the SA is elevated to (or remains at) *VDD*. To write the result back to the target cell, *TG1* and *TG2*, or *TG3* and *TG4* are activated for copy or NOT operation, respectively. The target cell is connected to the bitlines after bitlines voltages stable. The write-back operation can be performed on more than one target cells, if needed.

3) X(N)OR Operation: XOR operation is implemented as $\overline{XNOR}$ in BIMAX as Eq. 1 describes.

$$XOR = \overline{XNOR} = \overline{AB + \overline{A + B}} \quad (1)$$

For the X(N)OR operation, AB is computed as a normal single AND operation and the result is stored in a temp SRAM cell. Then, *BLL* and *BLR* are precharged to *VDD* for $\overline{A + B}$ operation. The result of $\overline{A + B}$, stored in the SA is utilized to generate V_s on the bitlines for the NOR operation between AB and $\overline{A + B}$. After the SA stabilization ($\overline{A + B}$ result), a NOR operation is performed between the temp SRAM cell and the stored value in the SA. For this aim, in parallel with the stabilization of the SA during computing $\overline{A + B}$, the bitlines are precharged to *VDD*. After generating the voltage perturbation on the *BLR*, the SA is disabled in parallel with shortly connecting the temp SRAM cell to the *BLR*. The remaining operations is just like a normal NOR/OR in BIMAX to compute X(N)OR operation.

The main differences between the proposed IMC and the technique presented in X-SRAM [2] are that first of all, X-SRAM requires highly accurate timing of wordlines in 8T- and 9T-SRAMs for precise computation. Second of all, because the computation uses values between *VDD* and *GND*, this technique is more susceptible to process variations and environmental noise. Furthermore, For (N)AND and (N)OR operations, X-SRAM requires different sense amplifiers (SAs), resulting in approximately 2× more area overhead compared to the SA structure presented in BIMAX. Moreover, for X(N)OR operations, X-SRAM necessitates a hard-wired OR gate, and to write back the computation result to the operational SRAM cells, extra write drivers are needed for each bitline.

The main advantages of BIMAX over previously proposed IMC techniques using SRAM technology are:

- Differential sensing with rail-to-rail voltage access, ensuring robustness against bitline noise and process variations.
- Immediate write-back capability to any target SRAM cell without the need for additional write drivers.
- An SA per bitline is sufficient for all bitwise operations, with an area comparable to conventional active-load current mirroring SAs in SRAMs.
- BIMAX utilizes conventional wordline activation timing which is simpler and more robust than the precise timing needed in other SRAM-based IMC techniques, which adds complexity and sensitivity.

IV. Experimental Evaluations

This section compares BIMAX with the state-of-the-art bitwise in-SRAM computing techniques presented in [1], [11], [10], and [2]; we refer to these works as *Compute Caches*, *Fast SRAM*, *Configurable Memory*, and *X-SRAM* respectively, for better readability. To fairly evaluate and compare these techniques, we utilize 14 nm multi-gate transistors technology model in HSPICE. Table I shows simulation parameters.

Compute Caches [1] uses two active-load current mirroring SAs per each bitline pair. Each SA compares one of the bitlines voltages in a pair with a reference voltage to determine voltage perturbation. The reference voltage is common between all the SAs, and its voltage is set to $\frac{VDD-V_s}{2}$. Two extra write drivers per bitline are employed for write-back. Fast SRAM [11] adds two high-skewed inverters, two low-skewed inverters, two low-threshold voltage NMOS transistors, and two normal-threshold voltage NMOS transistors per bitline pair. The bitlines analog voltage is directly fed into the high-skewed inverters. If a voltage drop from *VDD* occurs on the corresponding bitline, the output of the high-skewed inverter rises to *VDD*, otherwise the output of the high-skewed inverter remains *GND*. Configurable Memory [10] exploits one extra wordline per SRAM cells row, together with two active-load current mirroring SA per bitline pair. This technique is not capable of storing computation results back to the memory, but instead can perform content addressable memory operations. X-SRAM [2] utilizes 8T- and 9T-SRAM for IMC, incorporating a shifted sense amplifier for each operation. Since this technique relies on a different memory technology, we just focused our evaluation on the impact of process variation on the 6T-SRAM variant of this technique, which is described in the appendix of [2].

A. Functionality and Performance

The voltage waveforms of operand cells A and B, bitlines, and the SA are presented in Figure 4 for AND operation. The OR, NOR, and NAND operations are not shown here, but their operation steps are similar. The voltage perturbation on *BLL*/*BLR* for (N)AND/(N)OR operations depends on the activation pulse width on the operand wordline. A shorter pulse width leads to smaller voltage change on the floated bitline and a longer pulse width can lead to reading disturbance on the operand cell [7], [25]. Thus, an optimum pulse width is selected for the SRAM cell read. This value in conventional SRAM memories is 100 ps to 300 ps, depending on the technology, and reliability and performance requirements of the SRAM memory array [7]. The pulse width on bitlines is considered 150 ps in our evaluations, which connects cell A and cell B to *BLL* at time 0 s and 150 ps, respectively.

As Cell A and Cell B waveform plot in Figure 4 shows, during the time period that the cells are connected to the bitline, a glitch occurs on Q and $\overline{Q}$ of the cells, if the voltage of connected node to the bitline is *GND* (red lines with circle and cross marks in cell A waveform and black line with rectangle and cross marks in cell B waveform). However, this glitch is recovered as soon as the cell is disconnected from the bitline. If only one of the two operands is '0', by connecting the cell to *BLL*/*BLR* for (N)AND/(N)OR operation, the connected bitline voltage drops to 911 mV in our setup. It is worth mentioning that since the voltage perturbation (drop) on the bitlines (V_s = 89 mV) is less than NMOS access transistor threshold voltage, the voltage drop on the bitline is not compensated if the voltage of connected node of the second

TABLE I
CIRCUIT-LEVEL SIMULATIONS PARAMETERS.

Technology node	14 nm PTM-MG
VDD	1.0 V
Array size	512×512 cell [7]
C_{BL}	60 fF [18]
SRAM cell size (W×L nm^2)	Pull-Up: 14×14, Pull-Down: 35×14, Access: 21×14
SA components dimensions (W×L nm^2)	Pull-Up: 49×14, Pull-Down: 84×14, Access NMOS: 21×14, Access PMOS: 28×14
Noise Simulation Parameters	Gaussian distribution: σ = 100 mV, 150 mV 200 mV, 250 mV and 300 mV. m = 0
Process Variation Drift Parameters	Gaussian distribution: σ = 2.5 %, 5 % 7.5 %, and 10 % of respective parameters. m = 0

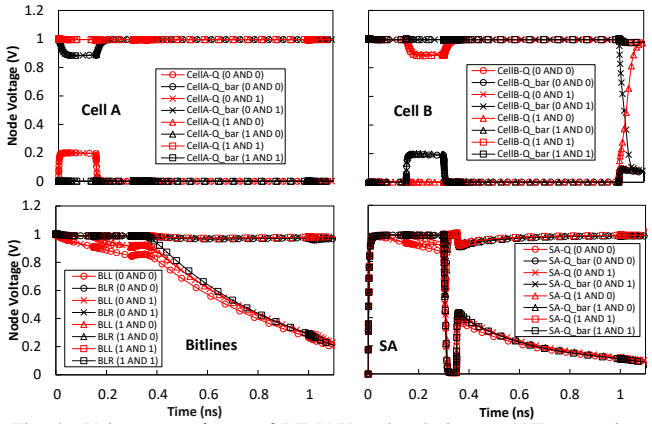

Fig. 4. Voltage waveforms of BIMAX nodes during an AND operation.

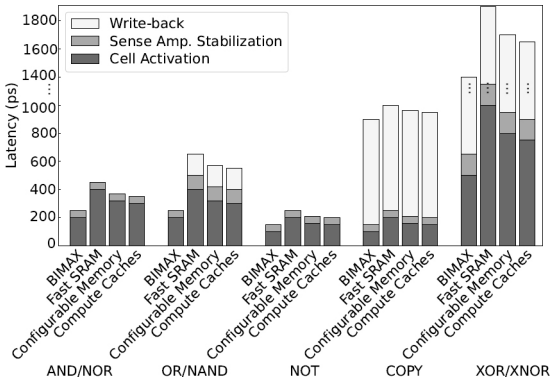

Fig. 5. Comparison of latency breakdown among BIMAX, Compute Caches [1], Fast SRAM [11], and Configurable Memory [10].

operand cell is *VDD*. If the voltage of both connected nodes is *GND*, after initializing the bitlines, the voltage on the connected bitline will be 843 mV (V_s = 157 mV). This voltage perturbation is slightly less than twice of that of the former case, because the drain current of the second operand cell access transistor reduces by decreasing its drain-to-source voltage (see the red line with circle marks in bitlines voltages waveform in Figure 4). The nodes Q and $\overline{Q}$ of SA follow the connected bitlines voltage till time 300 ps.

After stabilization of the SA, Bitlines voltage is updated in time period of 350 ps to 1 ns to perform the write-back operation. By connecting the SA nodes to the precharged bitlines with large parasitic capacitance, a glitch occurs on the internal nodes of the SA; however, the glitch is recovered over time by discharging one bitline to the *GND*.

At 1 ns, by activating the target cell, e.g. cell B in Figure 4, the computation result is written back into the cell in less than 100 ps. To the best of our knowledge, no SA structure, proposed in the previous works, is capable of storing computation results directly to the target cell. Previous proposals use some additional peripheral circuits, such as extra write drivers, to store the results in the target cell.

Figure 5 presents a comparison of the computation latency of BIMAX with that of the state-of-the-art previous work. The latency of AND/NOR operation of BIMAX, Compute Caches, Fast SRAM, and Configurable Memory are 250 ps, 350 ps, 450 ps, and 370 ps, respectively. These results demonstrate a 40% improvement in performance compared to the fastest previously proposed technique (Compute Caches). The delay comprises two SRAM row activations and one SA stabilization latency. The use of positive feedback in BIMAX's SA and its reliable operation make it possible to decrease the activation time. The OR/NAND

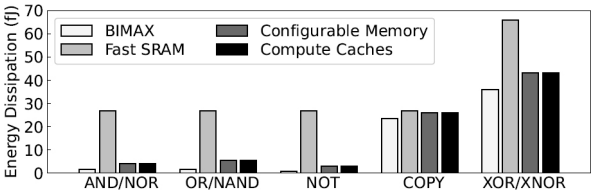

Fig. 6. Energy dissipation per operation.

operation values were found to be 250 ps, 550 ps, 650 ps, and 570 ps, respectively, resulting in a 120% performance improvement compared to the fastest previously proposed technique. Furthermore, Fast SRAM and Configurable Memory techniques lack the ability to perform OR and NAND operations, and cannot combine multiple base operations like AND and NOR to perform OR and NAND operations. To ensure a fair comparison, we included the necessary extra components required for these techniques to perform these operations in our evaluations. The NOT operation, consisting of one row activation and one SA stabilization, and the copy operation, which includes a write-back sub-operation in addition to the NOT, were found to be 50 ps faster than the state-of-the-art, due to the shorter required activation time. The writeback operation in copy takes 750 ps for all techniques.

We extended previously proposed techniques sub-operations sequences to execute X(N)OR operations in the fastest possible way. This delay is composed of four row activation, three SA stabilization, and two write-back latencies. For X(N)OR operations, intermediate calculated values must be stored in temporary SRAM rows, necessitating 750 ps to write back these values to the cells. Notably, BIMAX is also 250 ps faster than the state-of-the-art for X(N)OR operation. On average, BIMAX is 35%, 21%, and 17% faster than Fast SRAM, Configurable Memory, and Compute Caches, respectively.

B. Energy Dissipation

The main source of energy dissipation during in-memory computing is bitlines voltage charge-discharge cycles; Thus, full charge-discharge cycles on bitlines are avoided as much as possible in SRAM cell access. The evaluation results of energy consumption per operation, using HSPICE simulations, are presented in Figure 6 for BIMAX compared to the previous works. The energy dissipation for (N)AND and (N)OR operations is 1.3 fJ for BIMAX, which is 60.7% less than Configurable Memory and Compute Caches and 17.3× less than Fast SRAM. For OR and NAND operations, BIMAX consumes 71.3% less energy than Compute Caches, while Fast SRAM and Configurable Memory are not capable of performing OR/NAND operation. Energy dissipation of BIMAX for NOT operation is 0.6 fJ; 79.8% less than Configurable Memory and Compute Caches, and 44.6× less than Fast SRAM. Copy and X(N)OR operations are performed in BIMAX with 9.1% and 16.4% less energy consumption compared with Compute Caches.

C. Robustness

Dependable operation is the most significant attribute for real-world implementation of IMC techniques [19]. In this section we evaluate the two most important dependability parameters in nano-scale technology node size, 1) robustness against process variation and 2) noise immunity.

1) Process Variation: The operational SRAM cells and SA transistors threshold voltages are randomly selected in other 10 K Monte Carlo simulation rounds. The standard deviation of the

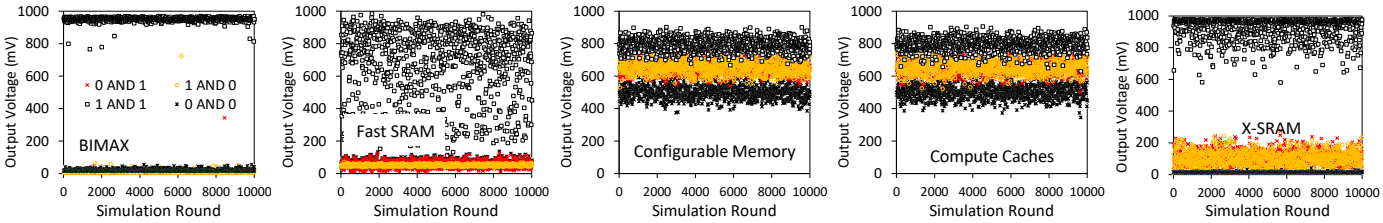

Fig. 7. SA output voltage for AND operation with 10% process variation.

TABLE II
NUMBER OF FAULTY OPERATIONS UNDER DIFFERENT PROCESS VARIATION LEVELS IN TEN THOUSAND SIMULATION ROUNDS.

σ	BIMAX	X-SRAM	Fast SRAM	Configurable Memory	Compute Caches
2.5%	0	0	567	76	37
5%	0	0	728	293	220
7.5%	2	11	790	544	437
10%	15	47	835	712	645

distribution model (σ) is considered 2.5%, 5%, 7.5%, and 10% of the mean value of each parameter (m).

Table II shows the number of faulty operations under process variation levels in 10K simulation rounds. As it is expected, Fast SRAM is the most susceptible IMC technique with about 5.7% at σ=2.5% that rises to 8.6% at σ=10%. This is because of the skewed single-input SA, utilized in this technique, which is highly susceptible to process variation. X-SRAM, Compute Caches, and Configurable Memory are much less susceptible than Fast SRAM, but their faulty operations rate rises to 0.5%, 6.5%, and 7.1%, respectively, as σ elevates to 10%. This is while even with a high process variation rate, with σ=10%, the faulty operations rate in BIMAX is less than 0.2%. Although X-SRAM is more robust compared to previously proposed techniques, it remains sensitive to corner cases, such as when one of the operands is '0'.

Figure 7 presents the SA output voltage for the execution of AND operation on random inputs for different IMC techniques. For better readability, one out of every ten rounds' results is depicted in this figure. As results show, in BIMAX, the output voltage levels of the SA are well separated for '0' and '1', with the standard deviation of 23 mV and 939 mV noise margin between output voltages corresponding to '0' and '1'. The standard deviation of output voltages for X-SRAM, Fast SRAM, Compute Caches, and Configurable Memory are 70 mV, 220 mV, 87 mV, and 88 mV with the noise margin of 823 mV, 613 mV, 138 mV, and 136 mV, respectively.

As the results show, the standard deviation and noise margin of SA output for BIMAX is 3.04× and 14.0% better than the previously proposed techniques. This is achieved by the high-gain positive feedback in the proposed SA, constructed by two cross-coupled inverters, and direct control of the internal nodes' voltage of the SA by the bitlines.

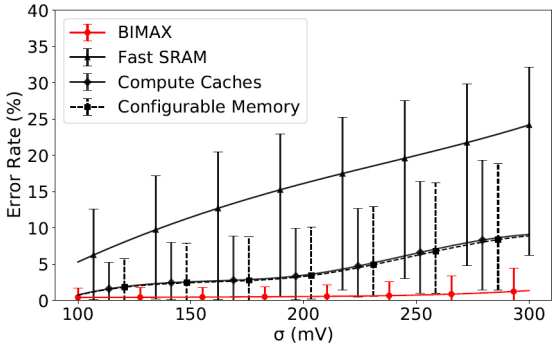

Fig. 8. Impact of ambient noise on the error rate.

2) Noise Immunity: Since the bitlines in memory structures pass over all memory cells of a column in the memory array, long bitlines are susceptible to environmental noises (behave as an antenna), e.g. electromagnetic noises. BIMAX uses a fully-differential SA which greatly cancels out the noise on bitlines in each pair, the same as cell access in conventional 6T-SRAMs [7]. This is while the other state-of-the-art techniques of in-SRAM computing use nondifferential skewed inverters [11] or double-ended differential SAs [1], [23] with noise sensitive inputs.

10 K Monte Carlo simulation rounds with five different noise voltage distribution levels on SRAM bitlines are considered. The injected voltage levels of noises follow Gaussian distribution with standard deviation (σ) of 100 mV to 300 mV, with a mean value (m) of 0 mV (refer to Table I). Figure 8 shows the average error rate of bitwise operation execution with random operand values, in the presence of ambient noise. With the σ of 100 mV, the error rate on BIMAX, Fast SRAM, Compute Caches, and Configurable Memory, is 0.4%, 5.3%, 0.8%, and 0.7%, respectively. BIMAX error rate is about 0.5%, even by increasing the σ to 200 mV. This is while Fast SRAM, Compute Caches, and Configurable Memory error rates sharply increase to 16.1%, 4.6%, and 5.0%, respectively (the same trend can be observed for the σ of 300 mV). The results show that BIMAX is 19.5×, 5.4×, and 5.7× more immune against noise compared to Fast SRAM, Compute Caches, and, Configurable Memory respectively, on average.

D. Area Overhead

The area overhead imposed by applying BIMAX to an SRAM array is limited to just one sense amplifier per bitline. For a 512×512 cell memory array, the overhead of BIMAX is less than 0.6%. The sense amplifier size is approximately 2.7× of the area of a single operational SRAM cell.

V. CONCLUSIONS

In this work, we proposed an imbalanced differential sense amplifier (SA) that is capable of detecting equal voltages at its inputs, as the third state, and generating appropriate output. Our proposed IMC technique, called Bitwise In-Memory Accelerator using SRAM Structure (BIMAX), is capable of executing (N)AND, (N)OR, X(N)OR, NOT, and row-copy operations. The simulation results showed that BIMAX dissipates 52.7% less energy than its state-of-the-art IMC competitor, while it outperforms the fastest previously proposed in-SRAM processing technique by 5.7%. Furthermore, the error rate of BIMAX is 5.4× less than the most robust IMC in the literature.

ACKNOWLEDGMENT

This paper is co-financed by the Barcelona Zettascale Laboratory with reference REGAGE22e00058408992 which is financed by the Ministry for Digital Transformation and Public Services within the framework of the Resilience and Recovery Facility - and the European Union - NextGenerationEU.

REFERENCES

[1] S. Aga, S. Jeloka, A. Subramaniyan, S. Narayanasamy, D. Blaauw, and R. Das, "Compute caches," in *IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 481–492.

[2] A. Agrawal, A. Jaiswal, C. Lee, and K. Roy, "X-sram: Enabling in-memory boolean computations in cmos static random access memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 12, pp. 4219–4232, 2018.

[3] A. Agrawal, A. Kosta, S. Kodge, D. E. Kim, and K. Roy, "Cash-ram: Enabling in-memory computations for edge inference using charge accumulation and sharing in standard 8t-sram arrays," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 3, pp. 295–305, 2020.

[4] J. Chen, W. Zhao, Y. Wang, and Y. Ha, "Analysis and optimization strategies toward reliable and high-speed 6t compute sram," *Transactions on Circuits and Systems I*, vol. 68, no. 4, pp. 1520–1531, 2021.

[5] J. Chen, W. Zhao, Y. Wang, Y. Shu, W. Jiang, and Y. Ha, "A reliable 8t sram for high-speed searching and logic-in-memory operations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 6, pp. 769–780, 2022.

[6] Z. Chen, Z. Yu, Q. Jin, Y. He, J. Wang, S. Lin, D. Li, Y. Wang, and K. Yang, "Cap-ram: A charge-domain in-memory computing 6t-sram for accurate and precision-programmable cnn inference," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 6, pp. 1924–1935, 2021.

[7] K. Itoh, *VLSI memory chip design*. Springer Science & Business Media, 2013, vol. 5.

[8] S. Jain, L. Lin, and M. Alioto, "±cim sram for signed in-memory broad-purpose computing from dsp to neural processing," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 10, pp. 2981–2992, 2021.

[9] A. Jaiswal, A. Agrawal, M. F. Ali, S. Sharmin, and K. Roy, "i-sram: Interleaved wordlines for vector boolean operations using srams," *Transactions on Circuits and Systems I*, vol. 67, pp. 4651–4659, 2020.

[10] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm configurable memory (tcam/bcam/sram) using push-rule 6t bit cell enabling logic-in-memory," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1009–1021, 2016.

[11] J. Jeong and J. Park, "Fast 6t sram bit-line computing with consecutive short pulse word-lines and skewed inverter," in *2020 International SoC Design Conference (ISOCC)*. IEEE, 2020, pp. 292–293.

[12] H. Jiang, X. Peng, S. Huang, and S. Yu, "Cimat: A compute-in-memory architecture for on-chip training based on transpose sram arrays," *IEEE Transactions on Computers*, vol. 69, no. 7, pp. 944–954, 2020.

[13] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3sram: An in-memory-computing sram macro based on robust capacitive coupling computing mechanism," *Journal of Solid-State Circuits*, no. 7, pp. 1888–1897, 2020.

[14] N. Kang, H. Kim, H. Oh, and J.-J. Kim, "Taim: Ternary activation in-memory computing hardware with 6t sram array," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, ser. DAC '22. Association for Computing Machinery, 2022, p. 1081–1086.

[15] H. Kim, T. Yoo, T. T.-H. Kim, and B. Kim, "Colonnade: A reconfigurable sram-based digital bit-serial compute-in-memory macro for processing neural networks," *Journal of Solid-State Circuits*, pp. 2221–2233, 2021.

[16] J. Lee, D. Shin, Y. Kim, and H.-J. Yoo, "A 17.5-fj/bit energy-efficient analog sram for mixed-signal processing," *Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2714–2723, 2017.

[17] K. Lee, J. Jeong, S. Cheon, W. Choi, and J. Park, "Bit parallel 6t sram in-memory computing with reconfigurable bit-precision," in *57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2020, pp. 1–6.

[18] R. Mathur, M. Bhargava, S. Salahuddin, P. Schuddinck, J. Ryckaert, S. Annamalai, A. Gupta, Y. K. Chong, S. Sinha, B. Cline, and J. P. Kulkarni, "Buried bitline for sub-5nm sram design," in *2020 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2020, pp. 20–2.

[19] J. Mu, H. Kim, and B. Kim, "Sram-based in-memory computing macro featuring voltage-mode accumulator and row-by-row adc for processing neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 6, pp. 2412–2422, 2022.

[20] N. Rohbani and M. Ebrahimi, "Sram gauge: Sram health monitoring via cells race," in *2021 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 2021, pp. 1–6.

[21] X. Si, J.-J. Chen, Y.-N. Tu, W.-H. Huang, J.-H. Wang, Y.-C. Chiu, S.-Y. Wei, X. Sun, R. Liu, S. Yu, R.-S. Liu, C.-C. Hsieh, K.-T. Tang, Q. Li, and M.-F. Chang, "A twin-8t sram computation-in-memory unit-macro for multibit cnn-based ai edge processors," *Journal of Solid-State Circuits*, vol. 55, no. 1, pp. 189–202, 2019.

[22] H. Wang, R. Liu, R. Dorrance, D. Dasalukunte, D. Lake, and B. Carlton, "A charge domain sram compute-in-memory macro with c-2c ladder-based 8-bit mac unit in 22-nm finfet process for edge inference," *IEEE Journal of Solid-State Circuits*, pp. 1–14, 2023.

[23] J. Wang, X. Wang, C. Eckert, A. Subramaniyan, R. Das, D. Blaauw, and D. Sylvester, "A 28-nm compute sram with bit-serial logic/arithmetic operations for programmable in-memory vector computing," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 1, pp. 76–86, 2019.

[24] K. Xiao, X. Cui, X. Qiao, J. Song, H. Luo, X. Wang, and Y. Wang, "A 28nm 32kb sram computing-in-memory macro with hierarchical capacity attenuator and input sparsity-optimized adc for 4b mac operation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2023.

[25] C. Yu, T. Yoo, K. T. C. Chai, T. T.-H. Kim, and B. Kim, "A 65-nm 8t sram compute-in-memory macro with column adcs for processing neural networks," *IEEE Journal of Solid-State Circuits*, pp. 1–1, 2022.

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