Sparse Vector-Matrix Multiplication Acceleration in Diode-Selected Crossbars

Nicholas Jao, *Student Member, IEEE*, Akshay Krishna Ramanathan,
John Sampson, *Member, IEEE*, and Vijaykrishnan Narayanan, *Fellow, IEEE*

***Abstract*—Conventional processors suffer from high access latency and power dissipation due to the demand on memory bandwidth for data-intensive workloads such as machine learning, analytic, etc. In-memory computing support for various memory technologies has provided formidable improvement in performance and energy for such workloads, alleviating the repeated accesses and data movement between CPU and storage. While many processing in-memory (PIM) works have been proposed to efficiently compute dot-products using Kirchoff's law, such solutions are unsuitable for many analytic workloads where working data is too large and too sparse to efficiently store in memory. This paper closely focuses on the peripheral circuit design for diode-selected crossbars and configures the compute-embedded fabric to efficiently compute sparse matrix-vector multiplication (SpMV). On average, our proposed end-to-end SpMV accelerator achieves 7.7x speed up and 4.9x energy-savings compared to the state-of-art Fulcrum.**

I. Introduction

As data volume continues to grow in emerging workloads, the performance of machine learning and data analytics applications on conventional von Neumann systems are constrained by memory accesses and data movement between physically separate processing and memory elements [14]. Moreover, such applications contain sparse matrix-vector multiplication (SpMV) operations which suffer from inefficient memory accesses and dominate the run time on conventional computing platforms [34] [35]. In an effort to mitigate the costs of data movement, recent works [47] [25] [24] have explored compute as close to the memory as possible to mitigate the data movement bottleneck.

Emerging non-volatile memory (NVM) technologies such as Phase Change Memory (PCM) [16], Resistive Random Access Memory (ReRAM) [13] and Spin Transfer Torque Magnetic Random Access Memory (STT-MRAM) [22] have enabled the exploration of new memory architectures which offer higher integration density than conventional volatile memories. Prominent among these structures, the crossbar architecture [27] [5] has shown great potential to be cheaper and denser than dynamic random-access memory (DRAM). Furthermore, research in emerging NVMs has demonstrated the possibility of using the resistance storage characteristics to perform analog computation [21]. As opposed to embedding logic near memory, NVM-based crossbars can be designed such that accessing the memory array itself outputs the desired computation. Because of these unique features, the crossbar architecture shows immense potential as computing in-memory solutions for memory-intensive workloads [19].

We propose a crossbar-based accelerator to solve SpMV operation found in data analytics workloads. Our architecture combines both computing in-memory and computing near-memory approaches to independently perform the entire SpMV operation. Our contributions include:

- The proposed crossbar design features a new analog compute operation called index search in addition to the read and write operations. The index search operation leverages the crossbar structure to search across at multiple non-zero matrix elements in parallel.
- We propose a dual sense amplifier (DSA) design which converts the crossbar's analog current into the index search output without using expensive analog-to-digital converters (ADC).
- We develop a parasitic-aware search reference capable of mitigating sense amplifier errors caused by the interconnect parasitics of large crossbars.
- We employ dynamic configurability in the peripheral logic to adjust the array-level parallelism by only turning on a select portion of the available DSAs, effectively reducing the number of outputs from index search at each crossbar. While this architectural feature also offers flexibility in the deployment scenario of the proposed accelerator, we analyze the trade-off between performance and energy-efficiency to show how this tuning knob is optimized based on the characteristics of the dataset.
- When processing multiple rows of the sparse matrix in parallel, our architecture control mechanism achieves significant energy-efficiency over a state-of-art near-memory processing architecture [24] by employing an independent control scheme and buffer local to each tile.

II. Background

In this section, we provide an overview of the different kinds of multiply-accumulate (MAC) operations as well as related works for computing in-crossbars.

A. Overview of Dense MAC

Many state-of-art works [41] [26] [30] employ NVM-based crossbars as dot-product engines to accelerate matrix-vector multiplication (MVM) in convolution neural networks (CNN). The MVM operations are typically the inner product between a dense matrix against a dense vector, where the matrix is stored linearly in the crossbar array in column major order. Prior works [21] [8] [1] propose parallel analog dot-products in ReRAM arrays, applying Ohm's Law across the resistive

device and leveraging Kirchoff's Law to sum the current across all devices sharing the same column. Therefore, the magnitude of the current at each bit line is the result of the MAC operation. However, such works adopt high resolution analog-to-digital converters (ADC) to accurately transform the output current into a binary value, which incurs significant area overhead and dominates the power consumption of these systems. Moreover, these works do not consider current sneak paths and interconnect parasitics in crossbar architectures [27].

In contrast, prototype chips [7] [45] [46] implement the analog MAC operation in ReRAM arrays selected by access transistors and do not suffer from sneak path leakage. These works utilize novel multi-bit sense amplifier topologies [7] [45] and low-precision ADCs [44] [46] to efficiently realize neural network computations without complex and costly peripheral circuit overhead. [33] uses the ReRAM array as a fully-connected perceptron circuit and further reduces the analog-digital conversion into a step-function output, realized by a 1-bit comparator.

Aside the many works in ReRAM technology, [39] propose domain wall (DW) based devices programmed by spin orbit torque (SOT) to serve as both crossbar array and peripheral sensing interface. While ReRAM accelerators require CMOS-based sense amplifiers and power-hungry ADCs, such works [39] [41] leverage the ultra-low current switching characteristics of DW-based devices to replace the ADC and the data buffer, which are the most expensive components in the peripheral circuits.

In summary, many of the in-memory MVM works demonstrate speedup against CMOS accelerators for CNN applications. However, such systems can only achieve performance improvement under the condition that the data is stored linearly using contiguous addresses and all the operands are arranged in the same column to sufficiently take advantage of Kirchoff's Current Law. Furthermore, the analog MAC operation inherently does not support floating point representations and thus cannot be used for CNN training.

B. Overview of Sparse MAC

It is expensive to linearly store a very sparse matrix in memory because the elements storing zero occupy too much space, especially if the dataset is larger than the capacity of memory chip itself. To reduce the memory requirement, many machine learning datasets are typically arranged in a data structure in which only the non-zero values are stored along with their positions in the matrix. Among these data structures, the compressed sparse row (CSR) format [38] constructs an array of non-zero elements for each row of the matrix. Each non-zero element is represented by a column index and a non-zero value, and the array is sorted by column index in ascending order. The CSR format is widely used in many applications due to its storage efficiency, but the compressed structure increases the computational complexity of matrix operations.

State-of-art accelerators [34] [35] have shown that sparse matrix vector multiplication (SpMV) operations in CSR format dominates the execution time of analytic applications during training phase. Among the workloads under study, these works have observed that over 90% of kernelized support vector machines (K-SVM) and over 70% of K-means clustering algorithms are spent on sparse matrix sparse vector multiplication (spMspV) and sparse matrix dense vector multiplication (spMdV) respectively. Such compute patterns in data analytic algorithms provide a strong motivation to reduce the run time of solving SpMV in CSR data structures.

While SpMV accelerators [34] support bit-level data packing schemes to reduce memory accesses of the matrix data stored off-chip, works in computing in-memory [20] [24] propose redesigning the memory itself to accelerate SpMV workloads and further reduce data movement required to access large matrices. Our prior work [20] integrates arithmetic compare logic into ReRAM crossbar peripheral circuits to solve the index matching of non-zero elements in SpMV and only schedules off-chip loads for non-zero values that need to be multiplied. State-of-art Fulcrum [24] integrates an entire arithmetic logic unit (ALU) at every DRAM row buffer to support a variety of memory-intensive benchmarks including SpMV and achieves significant speed-ups against a GPU by avoiding data movement overheads.

Other related works [42] [6] [48] propose ReRAM-based crossbar architectures to accelerate SpMV in graph processing. Both [6] and [48] explores a conditional dot-product engine in the ReRAM crossbar, only activating the analog MAC operation on the rows corresponding to a particular source or destination vertex. To support vertex searching, the vertex indices are physically stored in ternary content-addressable memory (TCAM) arrays which requires two complementary ReRAM devices to represent one bit of data [18] [19]. Like the prior works in Section II-A, the selective analog dot-products performed in ReRAM-based graph processing are limited to integer and fixed-point data representations only.

III. Sparse-Matrix Sparse-Vector Multiplication

In this section, we cover the details of the SpMV operation as well as discuss the additional processing overheads required to access a sparse matrix represented by the CSR data structure.

A. SpMV Basics

Fig. 1 illustrates an example of how the SpMV operation is executed on conventional computing systems, where a sparse matrix $\mathbf{A}$ is multiplied by vector $\mathbf{x}$ to generate the output vector $\mathbf{y}$. Fig. 1(a) shows the original dense representation and its corresponding sparse representation of the example SpMV operation, highlighting the dot product between first row of $\mathbf{A}$ with the column vector $\mathbf{x}$ to produce the first element in the $\mathbf{y}$ vector, $\mathbf{y_0} = \mathbf{A_0} \bullet \mathbf{x}$.

Matrix $\mathbf{A}$ is converted from a 4-by-4 square to a CSR structure in which every row contains only the non-zero elements. For instance, row $\mathbf{A_0}$ contains the value 3 at column 0 [0,3] and the value 2 at column 3 [3,2] as outlined in Fig. 1(a). The non-zero elements in vector $\mathbf{x}$ is represented

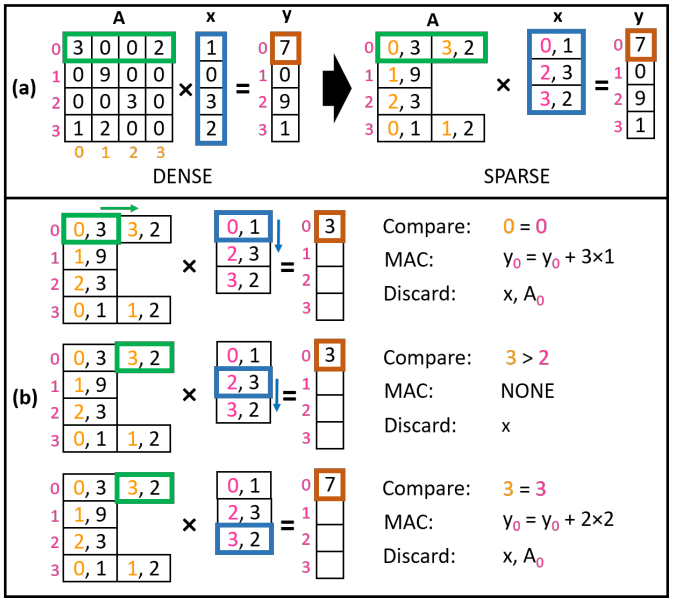

Fig. 1: (a) Example SpMV showing the conversion from the original to sparse (CSR) data format (b) Irregular access patterns and compute structures required to perform SpMV in CSR format

with the row index and the non-zero value at that row. The SpMV example shows that rows 0, 2 and 3 corresponds to the values 1, 3 and 2 respectively. In conventional MVM, the number of columns in matrix **A** must be the same as the number of rows in vector **x** as the columns needs to be multiplied against the corresponding rows. However when the non-zero elements are compressed to index-value structure, the indices of **A** and **x** need to be compared to determine which non-zero elements are to be multiplied and added to resulting element, $\mathbf{y_0}$. We refer to this comparison process as the index search operation which bottlenecks the execution time of SpMV.

Fig. 1(b) shows a step by step process on how SpMV is typically performed in software. Initially, all elements in the output vector **y** begin at zero. The example given in Fig. 1(b) demonstrates the computation flow required to resolve the first element $\mathbf{y_0}$ as shown. The first non-zero elements from $\mathbf{A_0}$ and **x** are accessed in a sequential manner for comparison and the index search operation finds that the column index from $\mathbf{A_0}$ matches the row index of **x**. Because an index match indicates that both $\mathbf{A_0}$ and **x** holds a non-zero element in the same position, the non-zero values, 3 and 1 respectively, need to be multiplied and accumulated to the $\mathbf{y_0}$. Then, the elements [0,3] and [0,1] are discarded because no other element can occupy position 0 and the subsequent elements [3,2] and [2,3] are accessed. Here the the indices do not match, and the MAC operation is skipped as a result. Because the CSR format is sorted by index in ascending order, only the element with the lower index is discarded as the next element in **x** could possibly match the element in $\mathbf{A_0}$. Lastly, the next element in **x** matches that of $\mathbf{A_0}$ and the corresponding values are multiplied and added to the final result. In this example, both $\mathbf{A_0}$ and **x** have reached the last non-zero element of the list. In most cases, this process completes when one finishes before the other.

As a result, the index search component of SpMV controls which non-zero elements are the multiplied, accumulated and discarded. While the original dense format enables direct linear access of the elements and achieves computation parallelism in dot-products, the throughput of SpMV is limited by the sequential nature of index search required properly operate on the compressed sparse data. This work proposes an in-memory solution to solve index search and demonstrates how to leverage the PCM-based crossbar structure to achieve parallel compare operations and speed up execution time of SpMV.

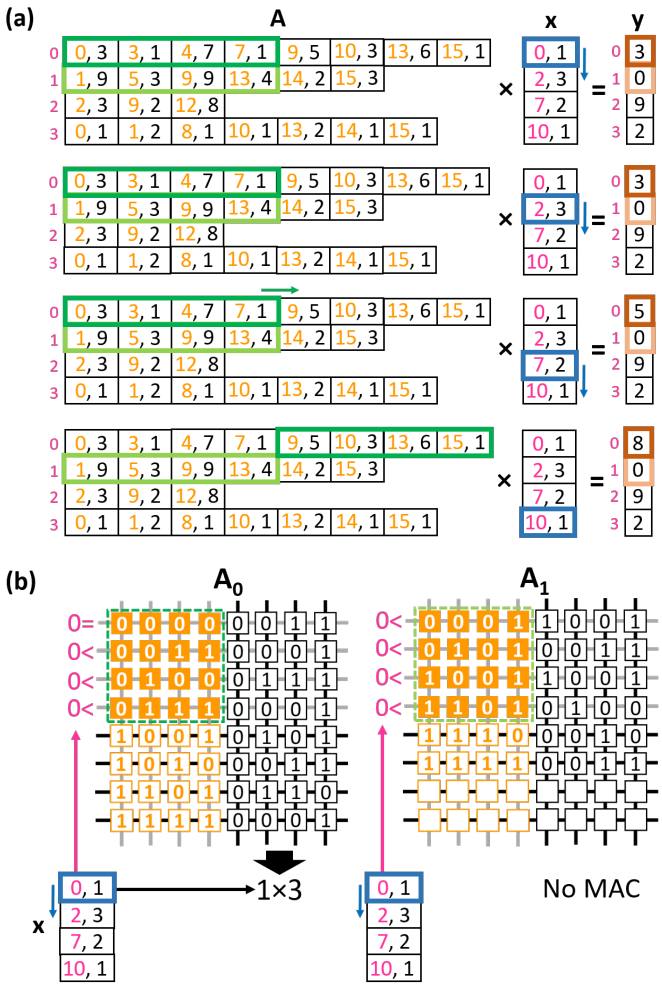

Fig. 2: (a) Example of proposed SpMV in CSR format (b) Index search operation performed in-situ in PCM-based crossbars which physically stores the non-zero elements of row 0 and row 1 of matrix **A** on separate tiles in binary

B. Proposed Index Search

Fig. 2(a) goes through step by step how the proposed approach to solves SpMV. In this example, matrix **A** represents

a 4×16 matrix in CSR format and is multiplied with a 16×1 sparse vector, vector **x**, to produce a 4×1 output, vector **y**. In contrast to Fig. 1, we use a larger matrix to showcase index search operations using a cluster. Fig. 2(a) highlights the first two rows of Matrix **A**, i.e. $\mathbf{A_0}$ and $\mathbf{A_1}$, being processed simultaneously since rows $\mathbf{A_0}$ and $\mathbf{A_1}$ both need to be multiplied with the same vector **x** to generate outputs $\mathbf{y_0}$ and $\mathbf{y_1}$ respectively. Moreover, the proposed approach compares each non-zero element from vector **x** to a cluster of non-zero elements from Matrix **A** to increase the chance of an index match in a single index search cycle. The example in Fig. 2(a) shows both rows $\mathbf{A_0}$ and $\mathbf{A_1}$ selecting 4 non-zero elements simultaneously, amounting to 8 index comparisons per cycle.

The first non-zero element [0,1] from vector **x** searches for a matching index in rows $\mathbf{A_0}$ and $\mathbf{A_1}$. The first element of $\mathbf{A_0}$ matches with index 0 and the corresponding values, 1 and 3, are multiplied and added on to $\mathbf{y_0}$ while no match exists in the selected cluster from $\mathbf{A_1}$. Because index 0 is less than the largest index in the $\mathbf{A_0}$ and $\mathbf{A_1}$ clusters, indices 7 and 13 respectively, element [0,1] from vector **x** is discarded for both rows. Next, the same index search pattern follows for the next non-zero element [2,3] from vector **x**, no matches occur and element [2,3] is discarded for both rows as well. When the selected element in vector x moves on to [7,2], the largest index element in the $\mathbf{A_0}$ cluster matches with index 7 and the MAC operation is performed on the corresponding values, 1 and 2. In this specific case, the entire non-zero cluster in $\mathbf{A_0}$ is discarded and the index search selects the next four non-zero elements in that row. On the other hand, row $\mathbf{A_1}$ does not match again and the cluster is not discarded because the element of highest index is still greater than index 7. In the final step of fig. 2(a), $\mathbf{A_0}$ cluster matches with index 10 and $\mathbf{A_1}$ does not match. The index search operation ends because the last element in vector x is discarded after multiplying the corresponding value with element [10,3] from row $\mathbf{A_0}$ and accumulating it on to element $\mathbf{y_0}$. Based on the varying discard patterns showed in $\mathbf{A_0}$ and $\mathbf{A_1}$, our proposed approach requires an independent control scheme for each row of matrix **A** to process them in parallel.

Note that conventional Von Neumann systems would require several digital comparators and a large memory bandwidth to process such a large portion of the sparse matrix at once. Thus, such solutions are unsuitable for our proposed approach especially when the index search scales up to operating on larger number of rows and non-zero clusters in parallel. Fig. 2(b) demonstrates the advantages the crossbar structure exploits when searching across multiple indices sharing the same column. To enable both rows of matrix **A** to operate independently, the contents of rows $\mathbf{A_0}$ and $\mathbf{A_1}$ are stored in separate crossbar arrays. Consistent with the first step of Fig. 2(a), both crossbars select the first 4 indices and compare whether the stored value is greater than, less than or equal to the search input, index 0. Based on the results of each comparison, the corresponding control unit for each crossbar determines which non-zero elements are discarded for the next index search. For the case shown in Fig. 2(b), element [0,1] from vector **x** is discarded for both crossbars. Then the queue head is updated to element [2,3] for the next index search. Because the CSR format sorts the non-zero elements by index in ascending order, the index-value pairs are stored in ascending order from the top to the bottom of each crossbar, allowing the control scheme to easily identify the largest index in the selected cluster. Lastly, the control unit enables the corresponding value for each index match to be multiplied and accumulated into an output register.

IV. Crossbar Design and Analysis

To support the proposed index search described in Section III-B, the voltage biasing and memory technology are the key design parameters such that the crossbar can properly convert its stored contents into analog currents via Kirchoff's current law. Voltage biasing design space becomes even more critical when the access transistors are eliminated for higher integration density and therefore relies on two-terminal selector devices to access the desired memory cells without interference from unaccessed cells during operation [5]. In such transistor-free crossbar memory systems, the voltage biasing scheme are optimized based on the current-voltage characteristics of the selector such that sneak path currents are effectively minimized and the unaccessed cells remain undisturbed. Among the many selector candidates, selector diodes [23] [36] have been well-studied due to its low process cost [17], high current density and orders of magnitude ON-OFF ratio. The primary advantage of diode-based selectors from a crossbar design standpoint is the device's intrinsic rectification characteristics during reverse bias which can be used to effectively cancel current sneak paths and enable more flexible voltage biasing schemes for compute in-memory applications. However, the rectification property limits the memory cells to only support unipolar switching. While novel back-end-of-line (BEOL) diode structures have been explored to potentially overcome these limitations [15], we choose PCM memory technology as the most suitable candidate to design diode-selected crossbars for index search support.

A. Phase Change Memory Basics

The data stored inside PCM devices are represented by the state of matter retained by the chalcogenide material. While prior work demonstrates the multi-bit storage potential with intermediate states [16], our work focuses on the PCM technology as a conventional single-level cell (SLC) device. PCMs use a temperature-based phase transition property to switch between amorphous (high resistance) and crystalline (low resistance) states of the material, which represents 0 and 1 bits, respectively.

When reading a single bit stored inside a PCM device, the voltage applied across the cell generates a current based on the electrical resistance of the material and that current is fed into a sense amplifier circuit to distinguish which of the two resistance states is stored in the cell. For the index search operation, we design the applied voltage across the resistance-based storage to produce a current proportional to the index

data stored in the array. To ensure the largest index produces the largest current, the voltage applied across cell storing the most significant bit (MSB) needs to be largest and vice versa for the least significant bit (LSB). Section IV-B goes through how the voltage-biasing scheme is designed to ensure (1) the accessed cells are transformed into the appropriate analog current (2) sneak path currents do not offset the result of the index search operation.

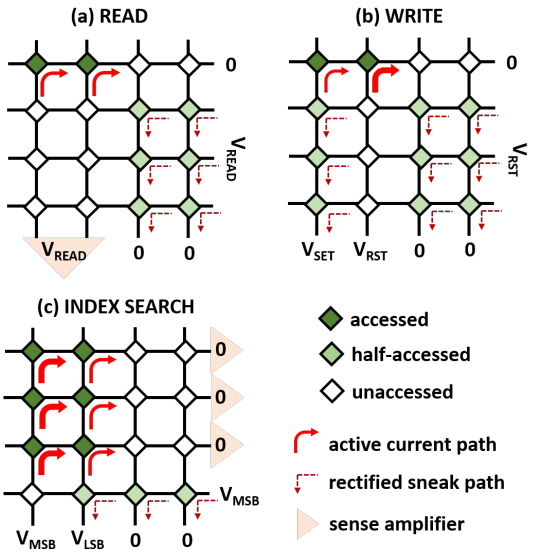

Fig. 3: Voltage biasing schemes for all supported operations

B. Voltage Biasing Scheme

Fig 3 shows the voltage biasing schemes for read, write and index search operations respectively. The read operation in Fig 3(a) applies the ground voltage on the selected word line (WL) and the state of the selected cells are sensed at the bit lines (BL) which are tied to V_{READ}. While there is no voltage drop across the cells sharing the same WLs and BLs as the accessed word, there is a reversed voltage drop across the unaccessed cells in which the current is effectively rectified by the reverse bias of the selector diode. Fig 3(b) shows the write operation following a similar voltage biasing pattern, driving the selected WL to ground. Here, a SET voltage, V_{SET}, is applied to the BLs of the cells writing 1 and the RESET voltage, V_{RESET}, to the BLs of the cells writing 0. Because the melting temperature is higher than that of the crystallization temperature of the device, the V_{RESET} is typically larger than the V_{SET} for the same pulse duration. Therefore, the unselected WLs are set to V_{RESET} to ensure none of the unaccessed cells are in forward bias.

Fig 3(c) shows the proposed voltage biasing scheme for the index search operation, which leverages the crossbar structure to sense the current at the WLs as opposed to the conventional read operation. Each selected BL is driven to a unique voltage level for the cells to generate current based on the bit position. In other words, V_{MSB} applied to the left-most bit produces the largest cell current and V_{LSB} applied to the right-most bit produces the smallest cell current when the respective PCM devices are storing 1, following Ohm' Law. However, when the PCM device stores 0, the resistance is so high that the cell yields insufficient current no matter what voltage is applied. In this way, the magnitude of the current on the WL represents the binary value stored, adding up the cell currents from MSB to LSB based on Kirchoff's Current Law. The crossbar architecture also allows the operation to search across multiple WLs and the unselected WLs are driven to V_{MSB} to ensures unaccessed cells stay below the threshold of the selector diode. The following sections go into detail on the peripheral circuit design to efficiently sense the WL currents while averting use of expensive ADC circuitry.

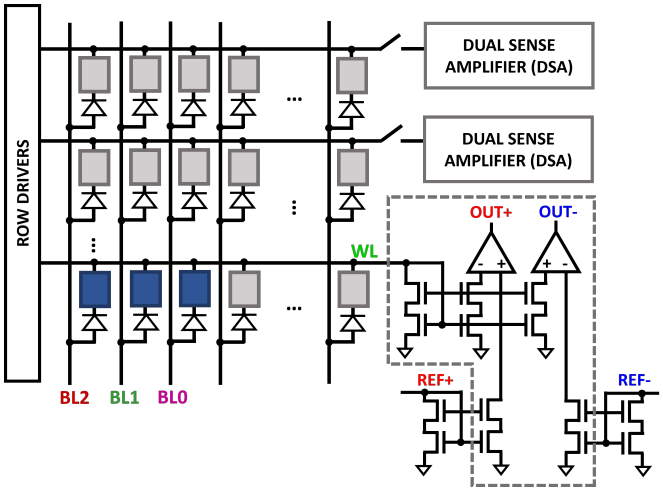

Fig. 4: Circuit diagram of the dual sense amplifier (DSA)

C. Dual Sense Amplifier

As discussed in Section III, the output of index search is primarily used to schedule the MAC and discard operations in the SpMV compute pattern. Consequently, the ADCs are unnecessary because the control unit only needs to know whether the stored data are less than, greater than or equal to the search input to make decisions. Shown in Fig. 4, we propose a dual current comparator circuit which comprises of two current-latch current sense amplifiers (CL-CSA) [29] [44] to transform the WL current into digital output signals, **OUT+** and **OUT-** respectively. Signal **OUT+** indicates that the stored index is less than or equal to search input and signal **OUT-** indicates that the stored index is greater than or equal to the search input. In other words, **OUT+** is logic 1 when the WL current is less than the **REF+** threshold and same with **OUT-** when the WL current is greater than the **REF-** threshold. When the magnitude of the WL current falls right in between **REF+** and **REF-** thresholds, both **OUT+** and **OUT-** output logic 1 and this indicates an index match. Therefore, the comparison output information is simply represented with 2 bits.

While voltage-mode signals can directly be connected to multiple comparators, the current sense amplifiers used in our design cannot be directly connected to the same WL for both comparators to operate simultaneously. Thus, the WL is connected to a conventional low voltage cascode current mirror

(LVCCM) to effectively copy the WL current to both CL-CSAs. Similarly, the **REF+** and **REF-** currents are also copied via LVCCMs to broadcast the search input to multiple dual sense amplifiers (DSAs). In this way, the DSAs provide the necessary interface to directly extract the compare information from the stored data. The primary disadvantage of the using the LVCCM to replicate the WL, **REF+** and **REF-** currents is that the power consumption is copied as well, resulting in additional direct current paths from the power rail to ground in the DSA design. In our design, the additional power overhead is mitigated by only activating the CL-CSAs for a short pulse after the crossbar array is fully charged. While the charging time of the WL current is determined by the crossbar size and interconnect parasitics, the pre-charge time for each CL-CSA is based on how fast the LVCCM turns on and is independent of the array size. For instance, it takes roughly 2 ns for all the WLs to charge to steady-state in simulation according to the crossbar parameters given in I. By enabling the CL-CSAs to operate for 0.5 ns and sample the WL currents at steady state, our design effectively reduces the energy dissipation of DSA, which accounts for 80% of the total crossbar energy.

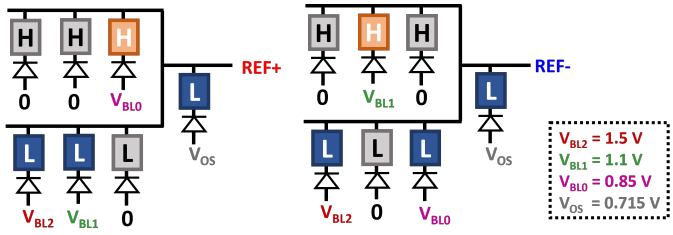

Fig. 5: Applied voltage bias on reference cells to generate **REF+** and **REF-** currents for a 3-bit index search corresponding to the binary number 110_2

In terms of peripheral circuit area, each DSA is pitch-matched across 16 WLs of the array and fits a total of 64 DSAs at the edge of a 1024x1024 crossbar. Consequently, the maximum number of comparisons that can be supported in a single search cycle is 64. Although searching across a larger cluster of non-zero elements will improve the throughput of the index search operation, the activation of many WLs and DSAs in parallel is very power hungry. Therefore, we enable the crossbar to switch between two configurations, high performance (HP) mode and low power (LP) mode. The HP mode aggressively turns on all 64 DSAs and strives to complete the SpMV computation flow as quickly as possible. On the other hand, the LP mode sequentially turns only 4 of 64 DSAs at a time and limits the broadcast capability on the crossbar to only a cluster of 4 per search. We evaluate the system-level performance benefits of both modes in section V.

D. Parasitic-Aware Search Reference

Another critical circuit design challenge is the generation of the appropriate **REF+** and **REF-** currents for the corresponding search input. The reference thresholds need to be chosen such that only one index can produce a match. Parallel to the how the WL currents are accumulated in the crossbar, PCM cell replicas with the same bias voltages are employed to produce the reference currents. Every reference cell comprises of a fixed PCM resistance in series with a selector diode, replicating the current-voltage characteristics of a crossbar cell.

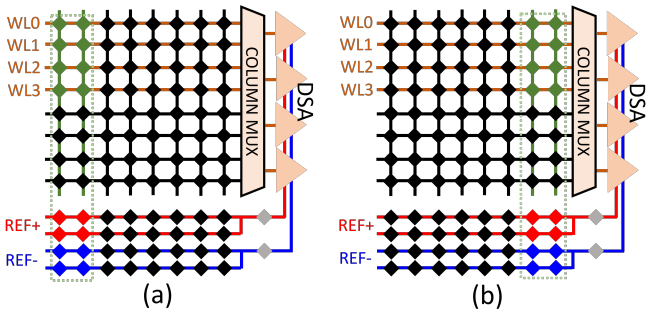

Fig. 6: Proposed parasitic-aware reference generation, activating the 2-bits which are (a) far away from the DSA (b) close to the DSA

Fig. 5 shows an example of how **REF+** and **REF-** are generated for a search input of the value 6. The voltage biasing pattern on the LRS replica cells is the primary mechanism that converts the input binary into analog current. In the example, the majority of the **REF+** current comes from the higher-order reference cells while the LSB reference cell is turned off, yielding a current that matches 110_2. To account for HRS leakage currents in the WL current, HRS reference cells are employed with the inverse voltage biasing of the LRS reference cells. An additional LRS reference cell, biased to $\mathbf{V_{OS}}$, is used to ensure that **REF+** is half LSB larger than the matching current such that the CL-CSA asserts **OUT+** when the index is less than or equal to that threshold. For the **REF-** threshold to be half LSB less than 110_2, the input to the reference cell drivers are decremented to 101_2 and employs the same half LSB reference cell to achieve the greater than or equal to functionality. By design, a matching WL current would fall between the two reference thresholds.

However, the parasitic resistance of the WL connection degrades the output current at large array sizes. Because interconnect resistance grows larger over routing distance, the accumulated current from cells far away from the DSA are lower than the expected value. In other words, the mean of the WL current falls closer to the **REF-** threshold for such cases and results in a higher probability of error when distinguishing an index match. Our solution modifies the reference current generator to adjust the thresholds based on the physical location of the accessed cells. To implement this, the entire WL is replicated twice for each reference threshold, one containing only LRS cells and the other containing only HRS cells. Fig. 6 shows how the WL replication is used to balance out the interconnect parasitic on both sides of the DSA, matching the degradation effect of the WL in the reference generation to minimizing errors. At the end of each reference generator, there is an additional LRS cell driven to $\mathbf{V_{OS}}$ to offset the reference currents by half LSB as elaborated

Parameter	Mean (μ)	Variation (σ/μ)
CMOS Supply V_{DD}	1.1V	0.64%
V_{MSB}-V_{LSB}	1.5 V, 1.1V, 0.85 V, 0.715 V	0.64%
R_{WIRE}, C_{WIRE}	0.52 Ω/μm, 0.048 fF/μm	10%
PCM R_{LRS}, R_{HRS} [16]	30 kΩ, 1000 MΩ	10%
Selector R_{DIODE}	5.8 kΩ	5%
Selector V_{TH}	0.2 V	2%

TABLE I: Default Simulation and Variation Parameters

in Fig. 5. The only overhead of our proposed parasitic-aware search reference is the additional peripheral circuit area of four entire WLs of dummy cells. However, the area overhead is less significant as the crossbar size increases. For our crossbar design, reference generator only accounts for 0.39% of a 1024x1024 array size.

E. Variation Analysis

To analyze the robustness of our design, we perform rigorous Monte-Carlo simulations on the index search operation and find the number of DSA errors which is referred to as search error rate in this work. TABLE I lists the memory device, selector device and array parameters as well as their coefficients of variance used in our design and simulation. For supply and bias voltages, we assume the variation parameter found from prior works in bandgap references [28] [37] [40] which have temperature coefficients as low as 34 ppm/°C. The selector variation parameters are chosen to be 2-5% according to [9].

TABLE II show the search error rates of 3-bit and 2-bit resolutions with the default variations applied. For the 3-bit case, each digital combination is independently tested under 200 iterations, 100 simulations close to the DSA and 100 simulations far from the DSA to ensure the effect of interconnect parasitics are also captured. To ensure the statistical significance of the results, the 2-bit case is tested under 2000 Monte-Carlo iterations. A search error rate of 0% (which has been reported in several locations of TABLE II) means that none of DSA distinguished incorrectly throughout all the iterations simulated. As observed, the 3-bit index search is more sensitive to variations than the 2-bit index search due to the smaller the current margin between adjacent digital codes, which can cause the DSA to misinterpret the stored resistance states in the WL. It is important to note that the sense margin between adjacent levels becomes tighter as the stored value increases. Consequently, the observed non-linearity in the output current distribution worsens as the index search resolution increases as shown in the high error rate regions of the 3-bit index search reported in TABLE II. To understand whether or not these error rates are acceptable from a system-level, we further evaluate the effect of DSA errors in Section V.

F. Accelerator Architecture

Due to robustness concerns of the index, the maximum index search resolution is limited to 2-bits (or 3-bits) per WL as discussed in IV-E. However, sparse matrices in many analytic workloads are require indices greater than 16 bits to

	10°C	25°C	40°C
3-bit Index	Search Error Rate		
000	0%	0%	0%
001	0%	0%	0%
010	0%	0%	3%
011	1%	0.5%	0.5%
100	27.5%	26.5%	42.5%
101	43%	34.5%	27.5%
110	45%	38.5%	38.5%
111	7.5%	4%	3%

	10°C	25°C	40°C
2-bit Index	Search Error Rate		
00	0%	0%	0%
01	0%	0%	0%
10	1.6%	0.85%	3%
11	0%	0%	0%

TABLE II: Monte-Carlo simulation on index search error rate for 3-bit and 2-bit resolutions under the coefficient of variation parameters from TABLE I

represent millions of columns. To overcome this limitation, our architecture slices the index into 2-bit segments and simultaneously operate on each segment to construct a full comparison. In other words, a 24-bit unsigned integer is distributed across 12 crossbar arrays, each performing the index search operation on their respective 2-bit segments. Because the higher-order segments are weighted heavier than the lower-order segments, a priority logic is required the merge the results of each DSA and output the final comparison information for the control unit.

Fig. 7 shows the architectural block diagram of all the required components to accelerate SpMV. The tile shown in Fig. 7(a) functions as both a memory and a processing element. The crossbars serve as the primary building blocks for index search and the priority logic constructs the final comparison results based on the segment stored in each crossbar. If comparing the most significant segments resulted in a greater than or less than, then the rest of the lower-order data are don't cares. If the result was equal, then the priority logic would propagate to the result of the next most significant segment. The control unit keeps track of which portions of the crossbars are activated, the output port of the FIFO and the operation of the floating-point (FP) MAC unit. Upon detecting an index match, the control unit finds the corresponding floating-point values to be multiplied and added on to the result stored in the buffer. Shown in Fig. 7(a), the value associated with the matched index is read from the crossbar and fed into to the FP MAC unit. The control unit is also responsible for the discard process, handling the read pointer of the FIFO as well as the selected rows and columns of the crossbars at every search cycle. Because the control unit dynamically configures the tile based on the output of each index search, it is critical that the crossbars are designed properly and DSAs generate the correct comparison results.

The FIFO component not only serves as a buffer for the index-value pairs to be compared, but also interfaces with other tiles to minimize the costs of data movement. Fig. 7(b) shows

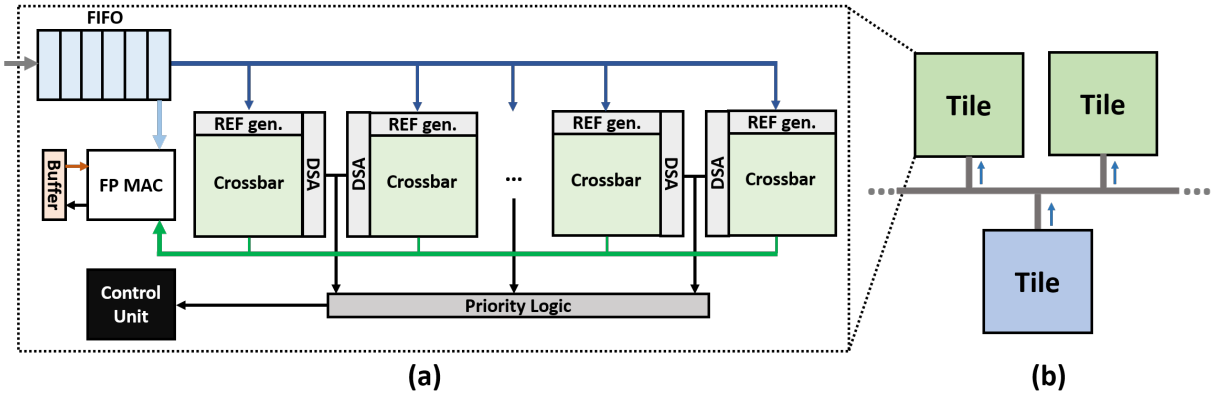

Fig. 7: Block diagram of (a) a single tile (b) broadcasting to multiple tiles

the tiles are connected together via bus-like routing, following the same wiring organization as a conventional random-access memory module. This inter-tile routing, which transfers load and store data during memory mode, is also used to internally broadcast non-zero data to multiple tiles during SpMV. As the number of tiles increases, the latency of the broadcast bottlenecks the performance of the index search. As long as the FIFO buffer is not empty, the tile continuously performs index search with the available elements in the local buffer and effectively hides the latency of the broadcast. To avoid redundant broadcasts, the tile sending the elements of vector **x** must keep track of the FIFO buffers of all active tiles operating on different rows of matrix **A** simultaneously, only sending new elements once when all receiving buffers are not full. If one tile completes the index search early, the sender no longer needs to send any more data to that tile or keep track of its FIFO status. Note that the FIFO memory is configurable as a conventional row buffer shared across multiple crossbars for memory transactions, supporting the distributed read and write of 4-bit segments to each crossbar.

Crossbar (Sub-Tile) Breakdown					
Component	**Detail**	**Area (mm2)**	**Operation**	**Delay (ns)**	**Energy (pJ)**
Row Driver	10:1024 Decoder	0.00154	R/W	0.090	0.201
Multi-Voltage Level Driver	4-wide 256-way	0.000464	Index Search	0.088	0.086
Crossbar	PCRAM 1024x1024 (2-bit / 3-bit)	0.2621	Read	1.1	0.57
			Write	30	10,752
			Index Search	1.581 / 1.75	2.14 / 2.49
Col. Mux	64-wide 16-way	0.000278	R/W	0.054	0.068
Data SA	64 bit	0.000554	Read Only	0.069	1.42
Row Mux	64-wide 16-way	0.00845	Index Search	0.054	0.068
Dual SA	64x2	0.0018	Index Search	0.069	6.22 / 6.59
Ref. Array	PCRAM 4x1024	0.001024	Index Search	-	0.0883
TOTAL	(2-bit / 3-bit)	**0.2763**	**Read**	**1.26**	**2.19**
			Write	**30.1**	**10,752**
			Index Search	**1.79 / 1.96**	**8.60 / 9.32**

Tile Breakdown (2-bit only)					
Component	**Detail**	**Area (mm2)**	**Operation**	**Delay (ns)**	**Energy (pJ)**
FIFO Buffer	1 Kb	0.0253	Index Search	0.069	17.43
Crossbars	12 Sub-Tiles	3.316	Index Search	1.79	103.2
Priority Logic		0.00138	Index Search	0.073	0.64
Control Unit		0.00256	Index Search	0.4	0.54
FP MAC	32 bit	0.0211	Multiply-Add	3.3	11.1
TOTAL		**3.366**	**Index Search**	**2.33**	**121.8**

System Components			
Component	**Detail**	**Delay (ns)**	**Energy (pJ)**
Routing	512 bit	9.582	163.6

TABLE III: Component Breakdown and Specifications

V. Evaluation

A. Experimental Methodology

Table III lists all the extracted delay, energy and area of the circuit and architecture components described in Section IV. The crossbar peripheral circuits and analog signals are simulated using HSPICE. In terms of layout analysis, all peripheral components make up only 4.6% of the crossbar sub-tile area. The digital logic and architecture components of the tile are implemented via register transfer level (RTL) synthesis. The routing between tiles are extracted from nvsim [11] configured with low swing wiring strategy and 2 GB storage capacity. All components are designed using 45nm transistor process design kit (PDK) [43] across all simulations.

We evaluate our proposed design on several sparse machine learning datasets from [12] which require SpMV operations in K-SVM training applications. The datasets under study are RCV1 [2], URL Reputation (URL REP) [31], and GV LDA [32]. RCV1 is a text categorization of multilingual documents from Reuters with the original english collection containing 18,757 documents and up to 21,531 features per document. URL REP is a time-series collection of 20 thousand URLs with over 3.2 million features used for classification of whether a URL is malicious or benign. Lastly, GV LDA contains 97,935 videos uploaded to YouTube in the gaming category in which each instance is a 1000-dimensional latent Dirichlet allocation (LDA) topic model on the title, description and tag information. The sparsity information of the datasets under study are summarized in Table IV.

While the benefits of our prior work [20] against an SpMV accelerator are due to reducing cost of data movement, state-of-art Fulcrum [24] mentioned in section II-B performs better than the GPU by almost 10x for the SpMV operation. Though Fulcrum is implemented in 3D-stacked DRAM, [24] claims that same simplified control and access mechanism can be employed in SRAM and NVM technologies as well. Therefore, we evaluate our proposed in-situ accelerator against the state-of-art Fulcrum with the simplified control built around conventional PCM crossbars called Fulcrum NVM. For quantitative comparison against our accelerator, we estimate the latency

Dataset	Matrix Size	% Non-Zeros
RCV1	18,757 x 21,531	0.35 %
URL REP	20,000 x 3,231,961	0.0036 %
GV LDA	97,935 x 1,000	2.2 %

TABLE IV: Summary of Datasets Under Study

and energy of the logic components using RTL synthesis tool and nvsim. Assuming the same memory organization and aggressive latency-optimized design strategy, we verify that Fulcrum NVM operates very close to the frequency of 199 MHz as reported in [24].

B. Performance Evaluation

Fig. 10 compares the performance benefits of our proposed accelerator against Fulcrum NVM of executing a single SpMV over the three K-SVM datasets under study. We fix the tile configuration of both accelerators to 16 active tiles in parallel, limiting the internal broadcasting of vector **x** to 16 rows of Matrix **A** at a time. For run time analysis, our accelerator outperforms Fulcrum NVM by 7.7x on average across all bit configurations, achieving up to 16.1x for the RCV1 dataset. The benefits of our accelerator are two-fold. First, Fulcrum NVM does not store broadcasted values of vector **x** local to each ALU which performs the index and value processing. Consequently, the clock cycle of Fulcrum NVM is based on broadcast latency which is longer than our accelerator's index search time according to table III. Second, Fulcrum NVM computes SpMV by sequentially broadcasting the index followed by the corresponding value, resulting in two cycles per non-zero element. This results in wasting cycles broadcasting data to that does not match in any active tile as also mentioned in [24]. On the contrary, our accelerator architecture stores the non-zero elements in the local FIFO buffer and handles the MAC and discard operations in the SpMV compute pattern independently at each tile, only stalling the index search process when a match exists.

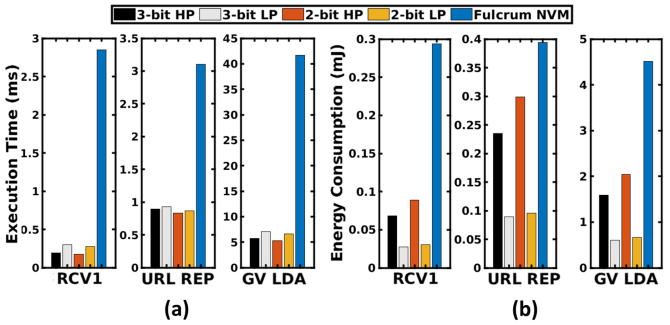

Fig. 8: Evaluation of our accelerator in HP/LP modes and Fulcrum NVM comparing the total (a) execution time and (b) energy consumption of a single SpMV operation.

In general, the GV LDA dataset took much longer to complete than that of RCV1 and URL REP. This is because the matrix size of GV LDA is much larger as the storage requirement is roughly an order of magnitude larger than that of RCV1 and URL REP. Relative to Fulcrum NVM, our accelerator gains less speed up on URL REP than RCV1 and GV LDA. Although URL REP has the highest sparsity among the datasets under study, we found that the URL REP experiences the highest rate of index matches, 63% on average, according to our statistical analysis. Mentioned in Section IV-F, the control unit stalls the index search for an additional two cycles to multiply and add corresponding floating-point values upon detecting an index match. As a result, our accelerator spends many cycles performing the MAC operation on URL REP dataset.

Similarly, our accelerator achieves an average energy savings of 2.61x in the HP mode and 7.17x in the LP mode over Fulcrum NVM. While the 93% of energy is dissipated in the crossbars and DSAs in our accelerator, Fulcrum NVM consumes substantial energy shifting the contents of the row buffer as input into the ALU as well as broadcasting vector **x** data to all tiles at every cycle. We also observe that the LP mode consumes lower energy across all 3 datasets. The HP mode aggressively looks for a match over a larger range of indices but consumes almost an order of magnitude higher static power during index search. While the HP mode speeds up the discard process by saving search cycles, the LP mode enjoys higher energy-efficiency by eliminating the redundant analog comparisons.

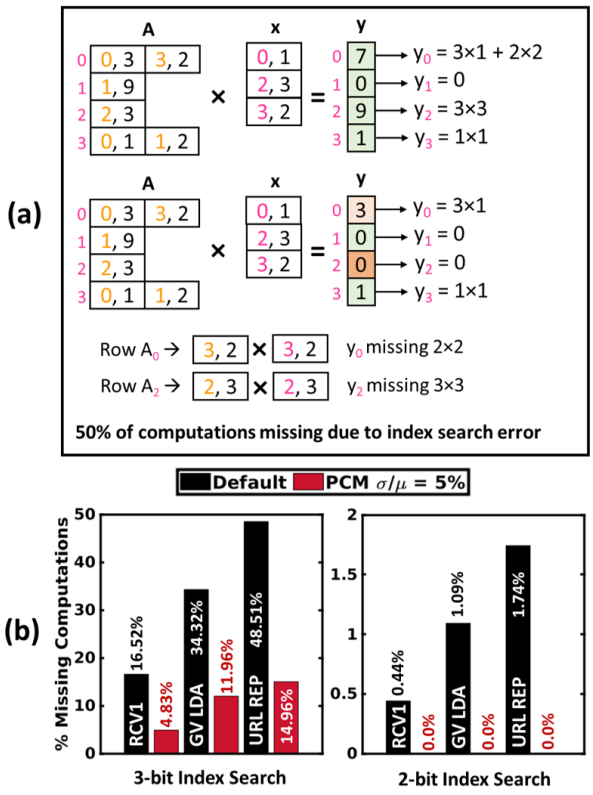

Fig. 9: (a) Example of missing computations in SpMV due to index search error (b) Evaluation of computation loss in SpMV operation due to the variation parameters given in TABLE I, showing the impact of PCM resistance variation on the accuracy of SpMV across various datasets.

It is observed that the 2-bit index search resolution performs

slightly better than 3-bit in execution time due to shorter search cycle. On the other hand, the 2-bit mode comes with higher energy consumption because it requires higher number of active crossbars to represent a 24-bit index. While the performance of both options remain relatively similar, there exists a huge difference in robustness of the index search operation between the 2-bit and 3-bit resolutions as shown in TABLE II. Fig. 9(a) illustrates a simplified example of how index matching errors can cause expected computations to be lost in an SpMV operation. In this example, the output vector **y** dropped 2 terms out of 4 total MAC operations, resulting in 50% missing computations. Fig. 9(b) analyzes the system-level impact of 2-bit and 3-bit search errors due to the default variation parameters from TABLE I at room temperature. Under default variations, the 3-bit resolutions suffer major computation losses for GV LDA and URL REP datasets and the 2-bit resolution is preferred to maintain the quality of output vector. By tuning the PCM resistance coefficient of variance (CoV) from 10% to 5%, we observe an average improvement of 3.1x in missing computations for the 3-bit case and no errors are observed for the 2-bit resolution. Therefore, it is crucial to have precise control over the non-volatile memory resistance states in order to operate the accelerator at higher bit resolutions.

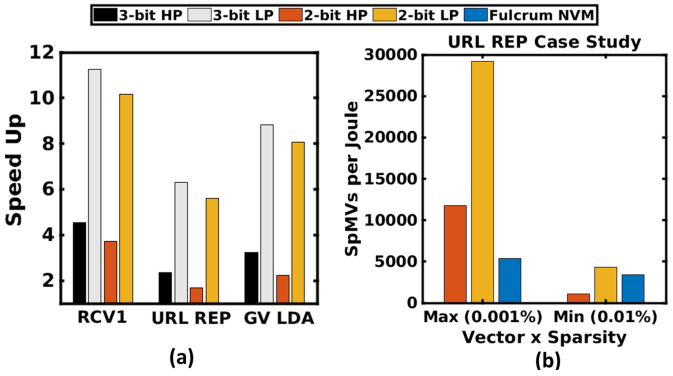

Fig. 10: (a) Throughput normalized to Fulcrum NVM at iso-power of 100 mW (b) Energy-efficiency analysis of the URL REP dataset for best and worst case sparsity (% non-zeros) in vector **x**

Fig. 10(a) evaluates the throughput of the HP and LP modes of our accelerator under a budget of 100 mW average power. Because the number of active tiles is the primary tuning knob to increase the parallelism of the SpMV operation, understanding how the accelerator performs at iso-power gives us insight into how suitable it is as an edge computing device for Internet of Things (IoT) applications. In a power-aware computing scenario, our accelerator achieves much higher throughput than Fulcrum NVM across all datasets. In particular, the LP mode achieves an average of 8.4x and 2.8x against Fulcrum NVM and the HP mode, respectively. The LP mode achieves highest throughput because it can provision more resources under the same power budget.

Fig. 10(b) captures the effect of the vector **x** sparsity on our proposed accelerator compared to Fulcrum NVM. For a deeper analysis of the results of Fig. 10(a), we choose URL REP as a representative workload for more detailed sparsity analysis. We evaluate SpMV operation using the sparsest, 0.001% non-zeros, and the densest, 0.01% non-zeros, vector **x** from the URL REP working set. We observe that the higher the sparsity of the vector **x** in $\mathbf{y} = \mathbf{A} \bullet \mathbf{x}$, the more energy efficient our proposed accelerator is. When multiplied against the sparsest vector, the 2-bit HP and 2-bit LP modes achieve 2.2x and 5.4x higher performance per watt than Fulcrum NVM respectively.

However, the energy-efficiency of the 2-bit HP mode is 3x worse than that of Fulcrum NVM when vector **x** is less sparse than matrix **A**. Since vector **x** is broadcasted to tiles which contain less non-zero elements than itself, the HP mode wastes energy performing the one-to-many compare operation at each tile. Consequently, the LP mode mitigates the energy overhead of the index search and is more suitable for accelerating spMdV operations, executing 27% higher operations per joule over Fulcrum NVM at the lowest vector **x** sparsity.

VI. Conclusions

While state-of-art Fulcrum [24] and our prior work [20] leverage computing near-memory architectures to accelerate SpMV, we propose a novel in-situ design which uses the crossbar's analog readout mechanism to accelerate the computationally-intensive index search of the SpMV pattern. Our proposed crossbar design avoids the use of expensive ADC and employs a circuit-level technique in the reference generator to overcome the sense margin challenges associated with supporting analog computing in large array sizes. While our technique addresses the impact of interconnect parasitics on the output signal, the impact of device variation remains an important concern and requires further investigation. Therefore, a future work can analyze the trade-offs of implementing existing solutions [4] [3] [10] to further improve the search error rate of the index search operation.

Our accelerator architecture efficiently processes the index-value data with a simple control unit local to each tile, only reading the data when an index match is detected. While state-of-art in-situ approaches rely on tile parallelism to broadcast values to a large portion of the unstructured matrix, our index search approach takes advantage of the crossbar structure to extract comparison information over a cluster of stored non-zero elements and ultimately reduces the number of search cycles required to complete each active tile. Moreover, our accelerator features dynamic configurability in the index search cluster size, allowing flexibility in the accelerator reap the benefits of both the HP and LP modes in various scenarios. In datacenters where performance and throughput are critical, the HP mode achieves the fastest SpMV execution on large datasets when the maximum tiles in the system are employed. On the other hand, the energy-efficiency of the LP mode is more suitable for power-aware computing when the number of active tiles are proportional to the power available.

VII. Acknowledgements

This work was supported in part by Semiconductor Research Corporation (SRC), Center for Research in Intelligent Storage and Processing in Memory (CRISP), National Science Foundation (NSF) Expeditions in Computing CCF-1317560 and NSF Special Projects CCF-1955815.

References

[1] A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *ISCA '16*, 2016, pp. 14–26.

[2] M. R. Amini, N. Usunier, and C. Goutte, "Learning from multiple partially observed views -an application to multilingual text categorization," in *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, ser. NIPS'09. Red Hook, NY, USA: Curran Associates Inc., 2009, p. 28–36.

[3] M. Awasthi, M. Shevgoor, K. Sudan, B. Rajendran, R. Balasubramonian, and V. Srinivasan, "Efficient scrub mechanisms for error-prone emerging memories," in *IEEE International Symposium on High-Performance Comp Architecture*, 2012, pp. 1–12.

[4] M. Awasthi, M. Shevgoor, K. Sudan, R. Balasubramonian, B. Rajendran, and V. Srinivasan, "Handling pcm resistance drift with device, circuit, architecture, and system solutions," in *Nonvolatile Memories Workshop*, 03 2011, p. 2.

[5] A. Aziz, N. Jao, S. Datta, and S. K. Gupta, "Analysis of functional oxide based selectors for cross-point memories," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2222–2235, 2016.

[6] N. Challapalle, S. Rampalli, L. Song, N. Chandramoorthy, K. Swaminathan, J. Sampson, Y. Chen, and V. Narayanan, "Gaas-x: Graph analytics accelerator supporting sparse data representation using crossbar architectures," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 433–445.

[7] W. Chen, K. Li, W. Lin, K. Hsu, P. Li, C. Yang, C. Xue, E. Yang, Y. Chen, Y. Chang, T. Hsu, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang, "A 65nm 1mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors," in *ISSCC2018*, 2018, pp. 494–496.

[8] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "PRIME: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA2016*, 2016, pp. 27–39.

[9] H.-L. Chiang, T.-C. Chen, M.-Y. Song, Y.-S. Chen, J.-P. Chiu, K. Chiang, M. Manfrini, J. Cai, W. J. Gallagher, T. Wang, C. H. Diaz, and H.-S. P. Wong, "Design space analysis for cross-point 1s1mtj mram: Selector–mtj cooptimization," *IEEE Transactions on Electron Devices*, vol. 67, no. 8, 2020.

[10] J. Choi, J. Jang, and L. Kim, "Dc-pcm: Mitigating pcm write disturbance with low performance overhead by using detection cells," *IEEE Transactions on Computers*, vol. 68, no. 12, pp. 1741–1754, 2019.

[11] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *TCAD2012*, vol. 31, pp. 994–1007, 2012.

[12] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[13] H. L. et al., "Beol based rram with one extra-mask for low cost, highly reliable embedded application in 28 nm node and beyond," in *IEDM '17*, 2017, pp. 2.4.1–2.4.4.

[14] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. Popescu, A. Ailamaki, and B. Falsafi, "Clearing the clouds: A study of emerging scale-out workloads on modern hardware," *SIGPLAN Not.*, vol. 47, no. 4, p. 37–48, Mar. 2012.

[15] S. Ghosh, R. Jha, A. Iyengar, and R. Govindaraj, "Design space exploration for selector diode-sttram crossbar arrays," *IEEE Transactions on Magnetics*, vol. 54, no. 6, pp. 1–5, 2018.

[16] M. He, D. He, H. Qian, Q. Lin, D. Wan, X. Cheng, M. Xu, H. Tong, and X. Miao, "Ultra-low program current and multilevel phase change memory for high-density storage achieved by a low-current set pre-operation," *IEEE Electron Device Letters*, vol. 40, no. 10, pp. 1595–1598, 2019.

[17] J. G. Hughes and G. W. Neudeck, "The total process cost of selective epitaxial growth (seg) dielectric isolation process as compared to locos," in *Twenty First IEEE/CPMT International Electronics Manufacturing Technology Symposium Proceedings 1997 IEMT Symposium*, 1997, pp. 88–92.

[18] M. Imani, S. Gupta, S. Sharma, and T. S. Rosing, "Nvquery: Efficient query processing in nonvolatile memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 628–639, 2019.

[19] N. Jao, A. K. Ramanathan, A. Sengupta, J. Sampson, and V. Narayanan, "Programmable non-volatile memory design featuring reconfigurable in-memory operations," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[20] N. Jao, S. Srivinasa, A. Ramanathan, M. Kim, J. Sampson, and V. Narayanan, "Technology-assisted computing-in-memory design for matrix multiplication workloads," in *2019 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, 2019, pp. 1–6.

[21] Y. Kim, Y. Zhang, and P. Li, "A digital neuromorphic vlsi architecture with memristor crossbar synaptic array for machine learning," in *2012 IEEE International SOC Conference*, 2012, pp. 328–333.

[22] K. Lee, J. H. Bak, Y. J. Kim, C. K. Kim, A. Antonyan, D. H. Chang, S. H. Hwang, G. W. Lee, N. Y. Ji, W. J. Kim, J. H. Lee, B. J. Bae, J. H. Park, I. H. Kim, B. Y. Seo, S. H. Han, Y. Ji, H. T. Jung, S. O. Park, O. I. Kwon, J. W. Kye, Y. D. Kim, S. W. Pae, Y. J. Song, G. T. Jeong, K. H. Hwang, G. H. Koh, H. K. Kang, and E. S. Jung, "1gbit high density embedded stt-mram in 28nm fdsoi technology," in *2019 IEEE International Electron Devices Meeting (IEDM)*, 2019, pp. 2.2.1–2.2.4.

[23] K.-S. Lee, D.-H. Yoo, J.-J. Han, Y.-W. Hyung, S.-S. Kim, C.-J. Kang, H.-S. Jeong, J.-T. Moon, H. Park, H. Jeong, K.-R. Kim, and B. Choi, "Selective epitaxial growth of silicon for vertical diode application," *Japanese Journal of Applied Physics*, vol. 49, no. 8, p. 08JF03, aug 2010. [Online]. Available: https://doi.org/10.1143/jjap.49.08jf03

[24] M. Lenjani, P. Gonzalez, E. Sadredini, S. Li, Y. Xie, A. Akel, S. Eilert, M. Stan, and K. Skadron, "Fulcrum: A simplified control and access mechanism toward flexible and practical in-situ accelerators," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 02 2020, pp. 556–569.

[25] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "Drisa: A dram-based reconfigurable in-situ accelerator," in *2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017, pp. 288–301.

[26] W. Li, P. Xu, Y. Zhao, H. Li, Y. Xie, and Y. Lin, "Timely: Pushing data movements and interfaces in pim accelerators towards local and in time domain," 2020.

[27] J. Liang and H. . P. Wong, "Size limitation of cross-point memory array and its dependence on data storage pattern and device parameters," in *2010 IEEE International Interconnect Technology Conference*, 2010, pp. 1–3.

[28] Y. Liu, C. Zhan, and L. Wang, "An ultralow power subthreshold cmos voltage reference without requiring resistors or bjts," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 1, pp. 201–205, 2018.

[29] C. Lo, W. Lin, W. Lin, H. Lin, T. Yang, Y. Chiang, Y. King, C. Lin, Y. Chih, T. J. Chang, M. Ho, and M. Chang, "Embedded 2mb reram macro with 2.6ns read access time using dynamic-trip-point-mismatch sampling current-mode sense amplifier for ioe applications," in *2017 Symposium on VLSI Circuits*, 2017, pp. C164–C165.

[30] Y. Luo and S. Yu, "Accelerating deep neural network in-situ training with non-volatile and volatile memory based hybrid precision synapses," *IEEE Transactions on Computers*, vol. 69, no. 8, pp. 1113–1127, 2020.

[31] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious urls: An application of large-scale online learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 681–688. [Online]. Available: https://doi.org/10.1145/1553374.1553462

[32] O. Madani, M. Georg, and D. A. Ross, "On using nearly-independent feature families for high precision and confidence," *Machine Learning*, vol. 92, pp. 457–477, 2013, published online 30 May 2013, [Web Link].

[33] R. Mochida, K. Kouno, Y. Hayata, M. Nakayama, T. Ono, H. Suwa, R. Yasuhara, K. Katayama, T. Mikawa, and Y. Gohou, "A 4m synapses integrated analog reram based 66.5 tops/w neural-network processor with cell current controlled writing and flexible network architecture," in *2018 IEEE Symposium on VLSI Technology*, 2018, pp. 175–176.

[34] E. Nurvitadhi, A. Mishra, and D. Marr, "A sparse matrix vector multiply accelerator for support vector machine," in *2015 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, Oct 2015, pp. 109–116.

[35] E. Nurvitadhi, A. Mishra, Y. Wang, G. Venkatesh, and D. Marr, "Hardware accelerator for analytics of sparse data," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2016, pp. 1616–1621.

[36] J. H. Oh, J. H. Park, Y. S. Lim, H. S. Lim, Y. T. Oh, J. S. Kim, J. M. Shin, J. H. Park, Y. J. Song, K. C. Ryoo, D. W. Lim, S. S. Park, J. I. Kim, J. H. Kim, J. Yu, F. Yeung, C. W. Jeong, J. H. Kong, D. H. Kang, G. H. Koh, G. T. Jeong, H. S. Jeong, and K. Kim, "Full integration of highly manufacturable 512mb pram based on 90nm technology," in *2006 International Electron Devices Meeting*, 2006, pp. 1–4.

[37] Y. Osaki, T. Hirose, N. Kuroki, and M. Numa, "1.2-v supply, 100-nw, 1.09-v bandgap and 0.7-v supply, 52.5-nw, 0.55-v subbandgap reference circuits for nanowatt cmos lsis," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 6, pp. 1530–1538, 2013.

[38] H. Pabst, B. Bachmayer, and M. Klemm, "Performance of a structure-detecting spmv using the csr matrix representation," in *2012 11th International Symposium on Parallel and Distributed Computing*, 2012, pp. 3–10.

[39] A. Sengupta, Y. Shim, and K. Roy, "Proposal for an all-spin artificial neural network: Emulating neural and synaptic functionalities through domain wall motion in ferromagnets," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 6, pp. 1152–1160, 2016.

[40] C.-Z. Shao, S.-C. Kuo, and Y.-T. Liao, "A 1.8-nw, 73.5-db psrr, 0.2-ms startup time, cmos voltage reference with self-biased feedback and capacitively coupled schemes," *IEEE Journal of Solid-State Circuits*, vol. 56, no. 6, pp. 1795–1804, 2021.

[41] S. Singh, A. Sarma, N. Jao, A. Pattnaik, S. Lu, K. Yang, A. Sengupta, V. Narayanan, and C. R. Das, "Nebula: A neuromorphic spin-based ultra-low power architecture for snns and anns," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 2020, pp. 363–376.

[42] L. Song, Y. Zhuo, X. Qian, H. Li, and Y. Chen, "Graphr: Accelerating graph processing using reram," 2017.

[43] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Freepdk: An open-source variation-aware design kit," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, 2007, pp. 173–174.

[44] X. Sun, S. Yin, X. Peng, R. Liu, J. Seo, and S. Yu, "XNOR-RRAM: A scalable and parallel resistive synaptic architecture for binary neural networks," in *DATE2018*, 2018, pp. 1423–1428.

[45] C. Xue, W. Chen, J. Liu, J. Li, W. Lin, W. Lin, J. Wang, W. Wei, T. Chang, T. Chang, T. Huang, H. Kao, S. Wei, Y. Chiu, C. Lee, C. Lo, Y. King, C. Lin, R. Liu, C. Hsieh, K. Tang, and M. Chang, "24.1 a 1mb multibit ReRAM computing-in-memory macro with 14.6ns parallel mac computing time for CNN based AI edge processors," in *ISSCC2019*, 2019, pp. 388–390.

[46] S. Yin, X. Sun, S. Yu, and J. S. Seo, "High-throughput in-memory computing for binary deep neural networks with monolithically integrated rram and 90-nm cmos," *IEEE Transactions on Electron Devices*, vol. 67, no. 10, pp. 4185–4192, 2020.

[47] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski, "Top-pim: Throughput-oriented programmable processing in memory," in *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 85–98. [Online]. Available: https://doi.org/10.1145/2600212.2600213

[48] L. Zheng, J. Zhao, Y. Huang, Q. Wang, Z. Zeng, J. Xue, X. Liao, and H. Jin, "Spara: An energy-efficient reram-based accelerator for sparse graph analytics applications," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 696–707.