Resistive GP-SIMD Processing-In-Memory

AMIR MORAD, Technion
LEONID YAVITS, Technion
SHAHAR KVATINSKY, Technion
RAN GINOSAR, Technion

GP-SIMD, a novel hybrid general purpose SIMD architecture, addresses the challenge of data synchronization by in-memory computing, through combining data storage and massive parallel processing. In this paper, we explore a resistive implementation of the GP-SIMD architecture. In resistive GP-SIMD, a novel resistive row and column addressable 4F^2 crossbar is utilized, replacing the modified CMOS 190F^2 SRAM storage previously proposed for GP-SIMD architecture. The use of the resistive crossbar allows scaling the GP-SIMD from few millions to few hundred millions of processing units on a single silicon die. The performance, power consumption and power efficiency of a resistive GP-SIMD are compared with the CMOS version. We find that PiM architectures and specifically, GP-SIMD, benefit more than other many-core architectures from using resistive memory. A framework for in-place arithmetic operation on a single multi-valued resistive cell is explored, demonstrating a potential to become a building block for next generation PiM architectures.

Categories and Subject Descriptors: **C.1.2 Multiple Data Stream Architectures (Multiprocessors), C.1.3 Other Architecture Styles**

General Terms: Design, Performance, Energy

Additional Key Words and Phrases: GP-SIMD, SIMD, Processing In Memory, PIM, In-Memory Computing, Memristor, Resistive RAM

ACM Reference Format:

TBD

1. INTRODUCTION

GP-SIMD is a highly-parallel, processing-in-memory SIMD architecture. In this paper, we explore a resistive-memory implementation of GP-SIMD (ReGP-SIMD), which achieves significantly higher integration than the CMOS-based GP-SIMD. In resistive GP-SIMD, a novel resistive row and column addressable 4F^2 crossbar is utilized, replacing the modified CMOS 190F^2 SRAM storage previously proposed for GP-SIMD architecture. We find that for the same silicon area, ReGP-SIMD integrates about 25× more memory and PUs and achieves about 5× better performance, but dissipates about 15 times more power. Thus, while higher density and performance are enabled by resistive technology, power density is increased.

GP-SIMD [29] is a novel, hybrid general purpose SIMD computer architecture that addresses the issue of data synchronization (the data transfer between the sequential

This research was funded in part by the Intel Collaborative Research Institute for Computational Intelligence (ICRI-CI), the Viterbi Fellowship, and by Hasso-Plattner Institute (HPI).

Author's addresses: Amir Morad (*), E-mail: amirm@campus.technion.ac.il; Leonid Yavits (*), E-mail: yavits@campus.technion.ac.il; Shahar Kvatinsky (*), E-mail: shahar@ee.technion.ac.il; Ran Ginosar (*), E-mail: ran@ee.technion.ac.il; (*) Authors are with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 3200003, Israel.

Permission to make digital or hardcopies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credits permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© TBD

DOI:TBD

and parallel processors) by in-memory computing, through combining data storage and massively parallel processing. Figure 1 details the architecture of the GP-SIMD processor, comprising a sequential CPU, a shared memory with two-dimensional access, instruction and data caches, a SIMD coprocessor, and a SIMD sequencer. The SIMD coprocessor contains a large number of fine-grain processing units, each comprising a single bit ALU (for performing bit-serial computations), single bit function generator, and quad, single bit register file. The GP-SIMD processor is thus a large memory with massively parallel processing capability. No data synchronization between the sequential and parallel segments [44] is required since both the general purpose sequential processor and SIMD co-processor access the very same memory array. The execution time of a typical vector operation in GP-SIMD does not depend on the vector size, thus allowing efficient parallel processing of very large vectors. The GP-SIMD is shown to achieve better power efficiency than conventional parallel accelerators [28][29]. In general, applications with low arithmetic intensity (the ratio of arithmetic operations to memory access [17][44]), such as sparse linear algebra kernels, are likely to benefit from GP-SIMD implementation.

Resistive RAM (ReRAM) is a non-volatile random-access memory that works by changing the resistance across a dielectric solid-state material [1][11][18]. ReRAM is shown to have key advantages over SRAM: (a) ReRAM memory cells are significantly smaller than CMOS SRAM ($4F^2$ *vs.* $190F^2$), facilitating very high memory densities; (b) ReRAM is non-volatile, incurring near-zero leakage power; and (c) ReRAM cells may be placed in metal layers above CMOS circuits, and consequently ReRAM memory array may be placed above logic, incurring near-zero chip area. ReRAM consists of a normally insulating dielectric material that can be made to conduct through a filament or conduction path formed after the application of a sufficiently high voltage. Once the filament is formed, it may be RESET (broken, resulting in high resistance state) or SET (re-formed, resulting in lower resistance state) by an appropriately applied voltage. Only when sufficient high-voltage current flows through the ReRAM cell in one direction, the electrical resistance increases; it decreases when current flows in the opposite direction. When the current stops, the cell retains its resistance. ReRAM requires similar write energy but suffers from finite endurance, relative to CMOS memories. Both SRAM and ReRAM memories have a similar architecture (row- and bit-line structure), and similar operation modes.

This paper extends the GP-SIMD architecture [29]. The contributions of this paper include:

- *Improvement of shared memory array for GP-SIMD:* (a) Motivating the replacement of CMOS shared memory by NVM; (b) Arguing that ReRAM is the most suitable NVM technology for GP-SIMD shared memory; (c) Arguing that GP-SIMD PiM architecture, being mostly memory based, benefits more than other many-core architectures from using resistive memory;
- *Resistive GP-SIMD:* (a) Exploiting the unique characteristics of ReRAM crossbar to enable 2D symmetric access (that is, read/write access from both the rows and the columns of the crossbar); (b) Replacing CMOS shared memory array (cell size $190F^2$) by a novel symmetric ReRAM memory array (cell size $4F^2$), enabling hundreds of millions of Processing Units (PUs) on a single silicon die and enhancing effective parallelism and performance; (c) Discovering that dynamic power dissipation becomes the limiting factor of ReGP-SIMD integration, and (d) Outlining efficient levels of integration according to power limitations ;
- *Resistive RAM multi-valued, in-cell arithmetic:* (a) Design and analysis of multi-valued addition and subtraction in a ReRAM cell, and (b) Design and analysis of

multivalued multiplication and division in resistive GP-SIMD (a multi-valued cell may store more than a single bit in its memory).

The rest of this paper is organized as follows. Section 2 discusses the GP-SIMD architecture. Section 3 provides a detailed description of the ReGP-SIMD resistive memory array and compares it with CMOS GP-SIMD. Section 4 discusses the ReGP-SIMD performance, power consumption and power efficiency as extracted from cycle accurate simulations, and compares it to CMOS implementation. Section 5 concludes this paper.

2. THE GP-SIMD PROCESSOR

In this section, we briefly describe the microarchitecture of GP-SIMD and discuss CMOS SRAM-like implementation of the shared memory array. A comprehensive description of the GP-SIMD architecture and software model can be found in [29]. Code examples can be found in [28].

2.1. Top Level Architecture

Figure 1 shows the architecture of a GP-SIMD processor, comprising the sequential CPU, shared memory array, L1 data cache, SIMD coprocessor, SIMD sequencer, interconnection network and a reduction tree. The sequential processor schedules and operates the SIMD processor via the sequencer. In a sense, the sequential processor is the master controlling a slave SIMD co-processor. Any common sequential processor may be used, be it a simple RISC or a complicated processor. At the very least, the selected processor should execute normal sequential programs. The SIMD coprocessor contains r fine-grain bit-serial Processing Units (PUs), where r is the number of rows in the shared memory array. Each PU contains a single bit Full Adder ("+"), single bit Function Generator ("Logic") and a 4-bit register file, RA, RB, RC and RD, as depicted in Figure 2. A single PU is allocated per row of the shared memory array, and physically resides close to that row. The PUs are interconnected using an interconnection network [29]. The set of all row registers of the same name constitute a register slice. Note that the length of the memory row (e.g., 256 bits) may be longer than the word-length of the sequential processor (e.g., 32 bits), so that each memory row may contain several words.

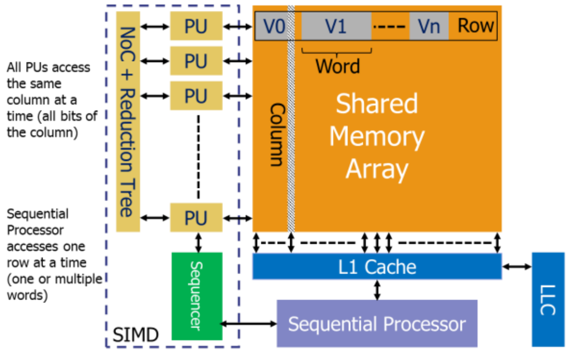

All PUs access the same column at a time (all bits of the column)
Sequential Processor accesses one row at a time (one or multiple words)
NoC + Reduction Tree
PU
PU
PU
PU
V0
V1
Vn
Row
Word
Column
Shared Memory Array
L1 Cache
LLC
Sequencer
Sequential Processor
SIMD

Figure 1. GP-SIMD top level architecture: Shared memory is accessed by rows from the PUs and by columns from the sequential processor

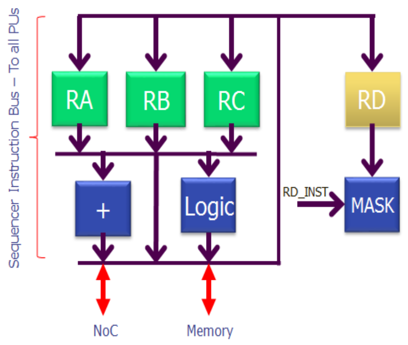
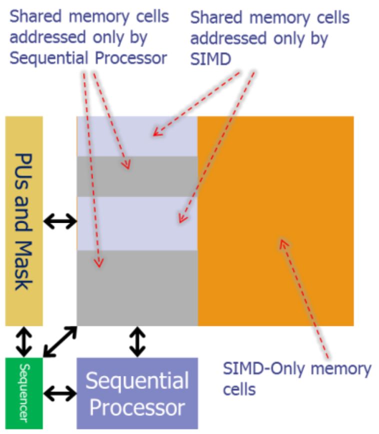


Figure 2. GP-SIMD Processing unit

2.2. Circuit Implementation

The GP-SIMD PU utilizes its memory row as its register file for storing variables and temporaries. The width of the memory array is much higher than the word width of the sequential processor (*e.g.*, 256 distinct columns vs. 32-bit word). A typical GP-SIMD memory array is depicted in Figure 3. To enable 2D access (word access by the sequential processor and column access by the SIMD co-processor), two types of cells are proposed. A shared-memory cell consisting of a 7-transistor CMOS SRAM bit is used in the shared columns (Figure 4), and a SIMD-Only cell using a 5-transistor cell is used in the SIMD-only columns (Figure 5).


Shared memory cells addressed only by Sequential Processor
Shared memory cells addressed only by SIMD
PUs and Mask
Sequencer
Sequential Processor
SIMD-Only memory cells
Figure 3. GP-SIMD memory segmentation into shared and SIMD-only cells

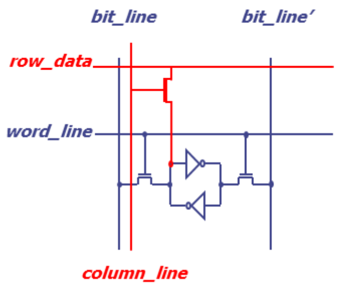


Figure 4. Shared cell

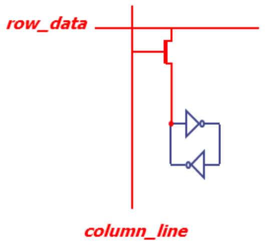


Figure 5. SIMD Only cell

The standard 6T CMOS SRAM bit cell (depicted by the blue-color drawings of Figure 4) is amended by a pass-gate to enable column read/write access (depicted by the red-color drawings). In the SIMD-only static bit cell (Figure 5), a 4T latch (blue) is connected by a pass-gate (red) to enable column read/write access. The *bit_line, bit_line_not*, *word_line* and associated pass-gates of Figure 4 are eliminated in Figure 5 as the sequential processor does not address these cells. When the width of the shared memory array is sufficiently small (e.g., 256 columns), a single pass-gate transistor is tied to the *row_data*. In wider arrays, a differential pair, namely, *row_data* and *row_data_not*, and an additional pass-gate transistor may be required.

Reading and writing to the two dimensional memory array is performed as follows:

— By sequential processor: to read data from memory, the *bit_line* and *bit_line-not* lines are pre-charged, the *word_line* is asserted, and the bit lines are sensed. To write data to the memory, the *bit_line*, *bit_line-not* and the *word_line* are asserted.

— By SIMD co-processor: to read data from memory, the *column_line* is asserted, and the *row_data* is pre-charged. To write data to the memory, the *column_line* and the *row_data* are asserted.

3. RESISTIVE GP-SIMD

The CMOS GP-SIMD is composed primarily (>96%) of leaky SRAM memory. We are thus compelled to search for a memory technology offering zero leakage power and much smaller footprint relative to the SRAM cell, as found in non-volatile memory (NVM) technology (NVM needs no power to retain its value – thus neither refresh power nor leakage power is consumed). In this section we outline alternative NVM memories, describe the novel ReRAM crossbar outlining its advantages for GP-SIMD, and present the resistive implementation of the GP-SIMD's shared memory array.

3.1. Alternative Non-Volatile Memory technologies

One of the limitations of commercially available flash non-volatile memory is its low endurance. Flash cells may withstand around 100,000 write cycles before the wear begins to deteriorate the integrity of the storage [38]. Further, endurance drops with scaling to smaller geometries, and modern multilevel 16nm NAND flash are limited to ~2000 write cycles. The size and power consumption of circuits such as charge pump further limits flash scalability. As VLSI geometries get smaller [16], the yield and reliability of flash cells decrease dramatically [32]. These limitations lead to the development of alternative nonvolatile memory technologies, such as Ferroelectric RAM (FeRAM), Magnetoresistive random access memory (MRAM), Spin-transfer torque random access memory (STT-RAM) potentially replacing MRAM in CMOS based designs, Phase-change memory (PRAM), Conductive-bridging RAM (CBRAM), and Resistive random access memory (ReRAM).

3.2. Resistive Memory (ReRAM)

ReRAM consists of a normally insulating dielectric material that can be made to conduct through a filament or conduction path formed after the application of a sufficiently high voltage. Once the filament is formed, it may be RESET (broken, resulting in high resistance state) or SET (re-formed, resulting in lower resistance state) by an appropriately applied voltage. Only when sufficient high-voltage current flows through the ReRAM cell in one direction, the electrical resistance increases; it decreases when current flows in the opposite direction. When the high-voltage current stops, the cell retains its resistance. The resistance does not change during read operation in which low voltage is utilized. Further promising features include low operating voltage, small cell area, and fast write time. To further increase ReRAM density, a "crossbar" structure is employed, as depicted in Figure 6. In the crossbar arrays, ReRAM cells are sandwiched between N wordlines and M bitlines forming $N \times M$ memory matrix, each having area as small as $4F^2$ (where F is the minimum size feature of the silicon process). The ReRAM cells are very similar to "via" (small openings in an insulating layer that allows a conductive connection between metal layers). The Resistive Crossbar is usually placed in the higher metal layers, above the CMOS logic and associated routing, limiting the impact on routing complexity. Since a crossbar array permits a multilayered structure, its effective cell area may be further reduced [1][11][18].

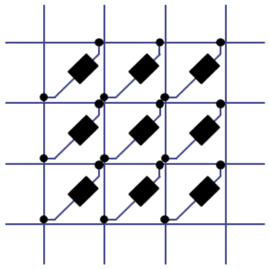


Figure 6. ReRAM cross-bar diagram

The resistance of ReRAM is changed by applying electrical current. The resistance is bounded by a minimum resistance R_{LRS} (low resistive state, logic '1') and a maximum resistance R_{HRS} (high resistive state, logic '0'). Write reliability is a serious concern in ReRAM crossbar arrays: Voltage applied across a crossbar varies based on the location of the cell as well as the data pattern stored in other ReRAM cells in the array, due to wire resistance and sneak paths.

Sneak paths are undesired paths for current, parallel to the intended path. The source of the sneak paths is the fact that the crossbar architecture is based on the ReRAM cell as the only memory element, without gating. Figure 7 shows an array with a simple voltage divider and its equivalent circuit. The figure shows the ideal case in which the current flows from the source to the ground passing through only the desired cell at the intersection between the activated column and row. Unfortunately this is not the real case as shown in Figure 8. In these figures, the dots represent ReRAM cells, the green lines show the desired path and the red ones show three possible sneak paths (note, many other sneak paths exist). The current flows through many sneak paths beside the desired one. These path act as an unknown parallel resistance to the desired cell resistance as shown in Figure 8. What makes the sneak paths problem harder to solve is the fact that the paths depend on the content of the memory. This is due to the fact that the current will sneak with more intensity through the paths with smaller resistance, which is memory content dependent.

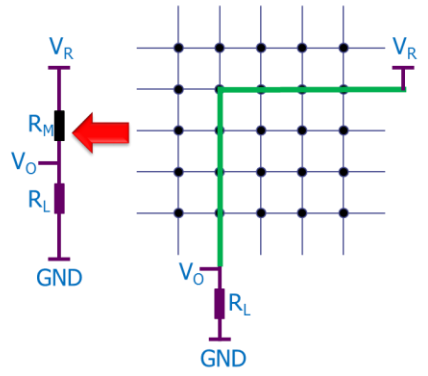


Figure 7. Current flow (green) during read / write, without 'sneak path'

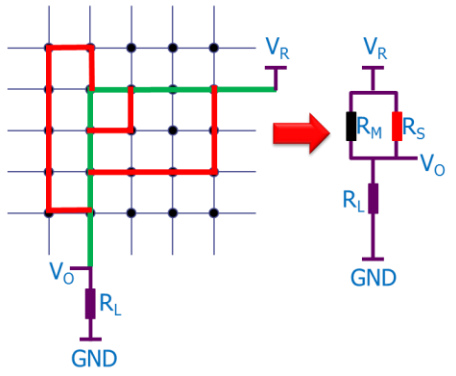


Figure 8. Three possible current 'sneak paths' (red)

Write is performed in two steps. First, SET is used to write all 1's along a row. Next, RESET writes the 0's in the remaining positions along the same row [31] (Figure 9).

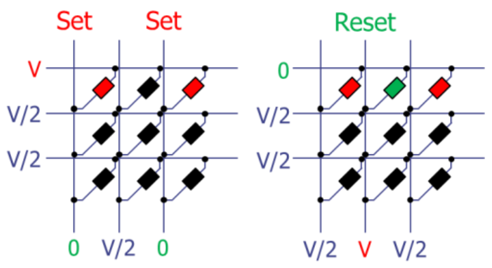


Figure 9. SET-before-RESET
Write **1**X**1** followed by Write X**0**X (*)

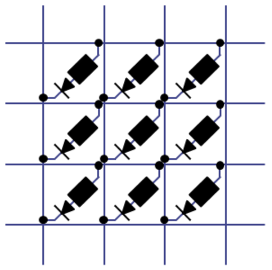


Figure 10. ReRAM having in-cell diode for sneak path mitigation

(*) Writing is disabled for the column(s) marked with X.

Read operation is also affected by sneak current, especially when the selected cell of a row is in high resistance state and the others are in low resistance state [5]. In order to alleviate read disturbance, parallel read can be employed to read all the cells in the same row. A per-cell diode [1][34] may be used to terminate read and write sneak paths, as depicted in Figure 10. If diodes are utilized, the ReRAM is driven with sufficient voltage *V* on write access, overriding the reverse biased diodes as well as the ReRAM write threshold.

A conventional resistive crossbar (Figure 12a) utilizes a single ReRAM device as its per-cell storage element. Access to the stored data is enabled by asserting a designated *word_line* and reading or writing from/to the crossbar columns [7]. Such a crossbar implementation may have a row driver for address select, column drivers for pre-charging the *bit_lines* and column sensors for inferring the data during crossbar read access.

3.3. Motivation for using ReRAM over alternative NVM technologies

The resistive GP-SIMD employs a resistive shared memory array instead of the CMOS memory of the original GP-SIMD design [29]. ReRAM is currently the most promising among all non-charge-based memory technologies intended to replace CMOS memory arrays, mainly due to its lowest footprint, CMOS compatibility, scalability, high endurance and low energy. TABLE 1 highlights the advantages and disadvantages of several leading NVM technologies: ReRAM, STT-MRAM and PCM [23][36].

TABLE 1
Leading NVM Technologies

	ReRAM	**STT-MRAM**	**PCM**
Endurance	High endurance (>10^{12})	Very high read/write endurance (but <10^{16})	Moderate endurance (>10^{7})
Low voltage switching	Fast low voltage switching (<1V): much faster than Flash	Fast low voltage switching (<1V): much faster than Flash	Reasonably fast at low voltage
Read	Non-destructive read	Non-destructive read	Non-destructive read
Maturity	Reasonable maturity	Reasonable maturity	Reasonable maturity
Cell Area	Small cell (4F^2)	Relatively large cell area (>> 6F^2).	Large cell (12-30F^2) due to heater circuitry
Process Scaling	Scales well to smaller VLSI geometries	Poor scalability of magnetic torque to smaller VLSI process geometries	Scales well to smaller VLSI geometries
CMOS Compatible	CMOS compatible (may be placed above logic)	CMOS compatible	Poor compatibility
Write power	High write power (large peripheral devices)	High write power (current driven → large peripheral devices)	Large write power because of heating (but current scales with cell dimensions)
Off-On resistance ratio	High	Small, ECC needed	High

ReRAM is a symmetric bit cell that requires no special per-bit pass gate to control access to its latch (unlike DRAM cell with one such transistor and SRAM cell with two). Reading and writing are simply enabled by passing current through the ReRAM cell. It is thus uniquely suited for GP-SIMD, since it enables access from either rows or columns without additional control devices per cell. A resistive crossbar array may thus facilitate row and column addressability.

Key CMOS SRAM, ReRAM and PU characteristics are contrasted in TABLE 2 [27][37][8][41][45][33][24]. ReRAM is CMOS compatible and hence the ReRAM crossbar may be placed on top of ReGP-SIMD PU logic, offering additional space savings. Further, ReRAM may be integrated in layers, reducing ReGP-SIMD storage cell to $4F^2/k$ footprint (where k is the number of vertically integrated ReRAM layers [1]).

TABLE 2
CMOS SRAM, ReRAM *vs.* PU

	CMOS SRAM	ReRAM	DRAM (For reference)
Memory area, cell (F^2)	190	4	6
Read latency, cell (ns)	0.1	0.25	0.3
Write latency, cell (ns)	0.1	0.25	0.2
Read energy, cell (fJ)	1	0.5(*)	5
Write energy, cell (fJ)	1	1(*)	5
PU area, cell (F^2)	1900		-
PU energy, cell (fJ)	5		-

(*) Extrapolated read and write energy values, based on consistent progress in scaling ReRAM speed and energy.

3.4. A Novel Symmetric 2-D access to Resistive Shared Memory Array

The conventional array (Figure 12a) is designed for access to columns of one row at a time. In the novel design of the two-dimensional symmetric access array (Figure 12b), row and column sense amplifiers and drivers are allocated to both the sequential processor and the SIMD. In a similar manner to [26][37], these circuits may be disconnected by pass-gates, allowing interleaved access either by the sequential processor or by the SIMD. Each cell consists of a single ReRAM device. The crossbar is thus designed to enable area efficient access to either the columns of one row (by the sequential processor) or to all rows of a single column (by the SIMD processor). In ReGP-SIMD, sneak currents affect both the read and write operations as in a standard ReRAM crossbar. To mitigate, a diode may be placed above ReRAM (hence consuming no additional area).

The top level architecture of the resistive GP-SIMD (ReGP-SIMD) closely follows the CMOS based GP-SIMD architecture of Figure 3, but (a) the storage element is ReRAM based; and (b) the storage is placed above of the PUs to conserve real-estate (Figure 11).

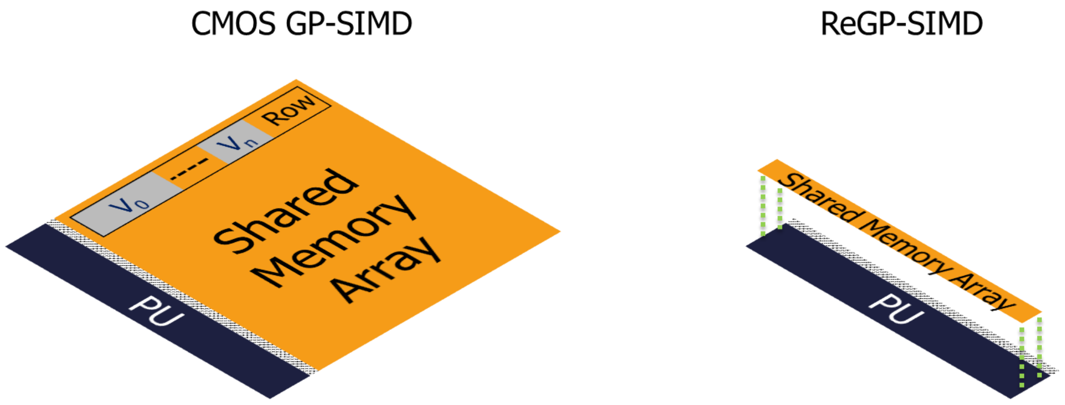
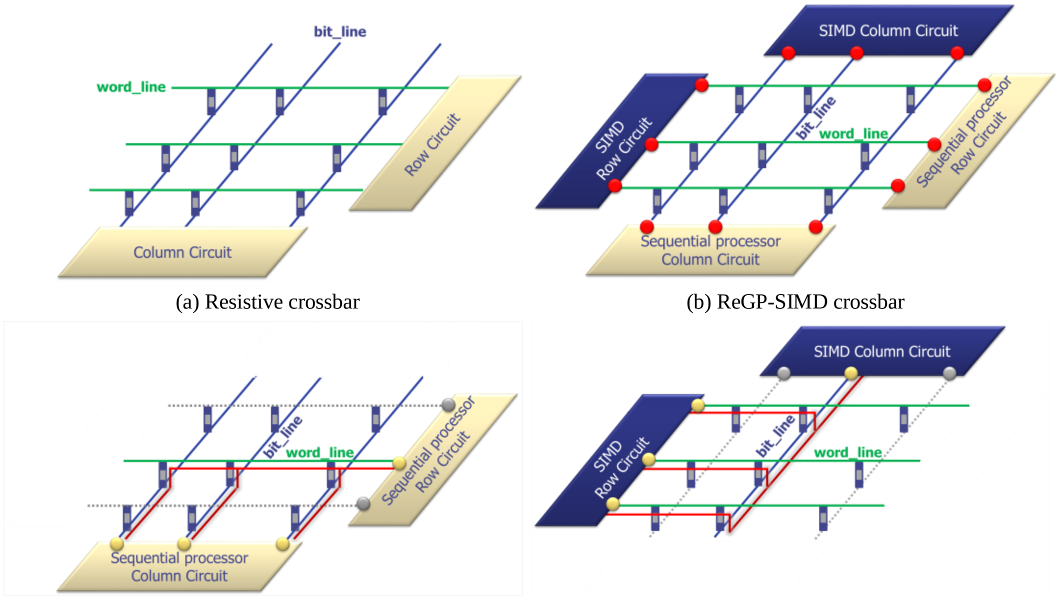


Figure 11. Shared Memory Array placement in CMOS (next to the PUs) and Resistive GP-SIMD (above the PUs)

Two modes of shared memory access are enabled. First, the sequential processor can access a row (see Figure 12c). Second, the SIMD processor can access a complete column, as in Figure 12d. While in the shared CMOS array (Figure 4) three transistors are needed to enable both modes of access (in addition to the storage 4-transistor latch itself), in ReRAM all that is needed is the storage node. The resistive storage element may be accessed from either of its two terminals, thus enabling symmetric two-dimensional access at much reduced area: 4F^2, in contrast with *190F*^{2} required for the CMOS shared memory cell. Further, partitioning the memory array to Shared and SIMD-only sections (as used in the CMOS GP-SIMD) is no longer required for area savings.



(a) Resistive crossbar (b) ReGP-SIMD crossbar

(c) R/W access by sequential processor (d) R/W access by SIMD co-processor

Figure 12. Crossbar array. Red lines denote current flow. Red dots denote pass-gates. Yellow dots denote connected pass-gates. Gray dots denote bias voltage for mitigating sneak paths.

Reading and writing to the memory crossbar array by the sequential processor and the SIMD are interleaved. The shared memory array is clocked four times faster than the sequential processor. Depending on the current sequential processor instruction and SIMD microinstruction (namely, if there is access to the array), up to four operations out of the following six possibilities may take place during a single access, requiring either one read or two writes for each processor:

Sequential processor access (Figure 12c):

1. *Read:* SIMD's row and column circuits are disconnected from the array. Sequential processor's column sensing circuits and row drivers are connected to the array. The *bit_lines* are precharged and the appropriate row driver corresponding to the sequential processors' addressed row grounds its *word_line*. Current flows from the *bit_lines* through all SET ReRAM devices connected to the grounded *word_line*. The columns sensing circuits sense changes on the *bit_line*, and output the corresponding bits to the sequential processor.
2. *Write ones, SET:* SIMD's row and column circuits are disconnected from the array. Sequential processor's row drivers are connected to the array. The *bit_lines*

designated to be SET are grounded, and the appropriate row driver corresponding to the sequential processors' addressed row asserts its *word_line*. Current flows from the *word_line* through all designated ReRAM devices connected to the asserted *bit_line*, and sets them to low resistance.

3. *Write zeros, RESET:* SIMD's row and column circuits are disconnected from the array. Sequential processor's row drivers are connected to the array. The *bit_lines* designated to be RESET are asserted, and the appropriate row driver corresponding to the sequential processors' addressed row grounds its *word_line*. Current flows from the *bit_lines* through all designated ReRAM devices connected to the grounded *word_line*, and resets them to high resistance.

SIMD co-processor access (Figure 12d):

4. *Read:* Sequential processor's row and column circuits are disconnected from the array. SIMD's Row sensing circuits and column drivers are connected to the array. The *word_lines* are precharged and the appropriate column driver corresponding to the SIMD co-processors' addressed column grounds its *bit_line*. Current flows from the *word_lines* through all SET ReRAM devices connected to that column. The rows sensing circuits sense changes on the *word_lines*, and outputs the corresponding bits to the SIMD PUs.
5. *Write ones, SET:* Sequential processor's row and column circuits are disconnected from the array. The SIMD's column drivers are connected to the array. The *word-lines* designated to be SET are grounded, and the appropriate column driver corresponding to the SIMD co-processors' addressed column asserts its *bit_line*. Current flows from the *bit_line* through all designated ReRAM devices connected to the asserted *bit_line*, and sets them to low resistance.
6. *Write zeros, RESET:* Sequential processor's row and column circuits are disconnected from the array. The SIMD's column drivers are connected to the array. The *word_lines* designated to be RESET are asserted, and the appropriate column driver corresponding to the SIMD co-processors' addressed column grounds its *bit_line*. Current flows from the *word_lines* through all designated ReRAM devices connected to the grounded *bit_line*, and resets them to high resistance.

Reading and writing by the SIMD or by the sequential processor are not concurrent. Thus, the SIMD sequencer may issue: (a) no access operation if the SIMD neither reads nor writes to the array; (b) a single access operation if the SIMD reads from the array; (c) two access operations if the SIMD writes to the array, and similarly for the sequential processor. Hence, zero to four shared memory operations are executed per a single clock cycle of the sequential processor (combining one SIMD memory operation and one memory operation of the sequential processor). The shared memory array is thus clocked with a frequency of four times the clock of the sequential and SIMD processor. That is, four distinct shared memory operations may be executed in a single sequential- or SIMD-processor cycle, and interleaving happens within that single clock cycle. Note that the interleaved addressing enables the sequential processor and the SIMD co-processor to read from and write to the shared memory without conflicts, as accesses are separated into distinct operation.

A more advanced implementation may allow the sequential processor to access the shared memory array on all four timeslots if the SIMD processor is not accessing it. However, our simulation and comparative analysis does not assume such implementation.

Utilizing the novel ReRAM crossbar reduces ReGP-SIMD storage cell to $4F^2/k$ footprint (where k is the number of vertically integrated ReRAM layers [1]). As shown in Section 4, with a single ReRAM layer, such level of integration allows placing 200M (k=1) 256-bit PU ReGP-SIMD on a single 200mm^2 silicon die. To compare, the CMOS GP-SIMD cell area is approximately $190F^2$ [6], limiting the GP-SIMD to 8.2M PU in the same silicon area. As detailed in Section 4, only the shared memory array scales with ReRAM. The PU however is implemented in CMOS, and thus limits the scalability of ReGP-SIMD. However, the architect may increase the number of bits per row to 475 (matching PU width) without changing the characteristics of ReGP-SIMD.

The switching time of ReRAM may reach the range of a hundred picoseconds [8][39], allowing memory access rates at frequencies of more than 1 GHz. The energy consumption of the resistive cell during read is less than 1fJ per bit. Unfortunately, the ReRAM write energy is in the range 1fJ per bit, and in addition to the energy required by the PU CMOS logic [29], prohibits scaling ReGP-SIMD to 200M rows, as shown in Section 4. However, write energy depends on the ReRAM material, and fortunately research of efficient ReRAM materials is likely to result in lower write energy consumption in the future. Further, resistive implementation of logic [21] may result in a power- and area-efficient ReRAM PU.

4. SIMULATIONS AND COMPARATIVE ANALYSIS

In this section, we evaluate power and performance of ReGP-SIMD and compare it to CMOS GP-SIMD.

4.1. Cycle Accurate Simulation, DMM

We use the previously reported GP-SIMD cycle-accurate simulator [29]. Simulations are performed on Intel® XEON™ C5549 processor with 32GB RAM, and simulation times vary from few minutes to few hours depending on the size of the matrix. The simulator employs the ReGP-SIMD performance and power figures obtained by SPICE simulations. We use CMOS 22nm process with operating frequency of 1GHz, and single precision floating point arithmetic. To evaluate ReRAM characteristics, we use the TEAM model [20][22][43]. Access time to the shared memory by the SIMD PU is significantly shorter than in CMOS GP-SIMD, due to the elimination of long wires (all word bits are laid above the corresponding PU). Access time for the sequential processor is the same in ReGP-SIMD as in the CMOS version; the memory is partitioned for this purpose, as described in [29]. The CMOS SRAM and ReRAM simulation parameters are outlined in TABLE 2.

Following [28] and [29], Dense Matrix Multiplication of a two $\sqrt{N}\times\sqrt{N}$ matrices (DMM) workload has been selected for our simulations, where N is the data set size, scaled for simplicity to the processor size (following the methodology suggested in [28]), *i.e.*, $N = n_{GPSIMD}$. Note that simulations do not cover the cases where the data size exceeds the size of the processor (requiring off-chip data synchronization). For $\sqrt{N} \times \sqrt{N}$ dense matrix multiplication (DMM), the sequential execution time is $O(N^{3/2})$. The GP-SIMD DMM algorithm utilizes N PUs and yields parallel execution time of $O(N)$, with $\sqrt{N}$ data elements being broadcast every step [28]. The GP-SIMD simulator is cycle based, recording the state of each register of each PU, and of the memory row assigned to it. Each command (for example, floating point multiply) is broken down to a series of fine-grain single bit PU operations. In a similar manner to SimpleScalar [4], the simulator also keeps track of the registers, buses and memory cells that switch during execution. With the switching activity and area power models of each baseline operation detailed in [29], the simulator tracks the total energy consumed during workload execution.

We follow the assumptions detailed in [29] and in section 3, and simulate speedup and power for growing datasets corresponding to the growing number of PUs and hence growing total area. The die area as a function of the integrated number of PUs for the CMOS and resistive designs are depicted in Figure 13(a). In all simulations, each PU is allocated 256 memory bits. The matrix dimension $\sqrt{N}$ is simply the square root of the number of PUs. The area of both designs was limited to 250mm². The simulated performance as functions of the number of integrated PUs is depicted in Figure 13(b). The power consumption and the power efficiency results are shown in Figure 13(c) and (d), respectively. Performance and power in both versions depict power law relationships to area and data set size (performance at an approximate exponent of 1/3, and power having exponent slightly larger than 1). Power efficiency thus depicts a power law relationship having an exponent of approximately -2/3.

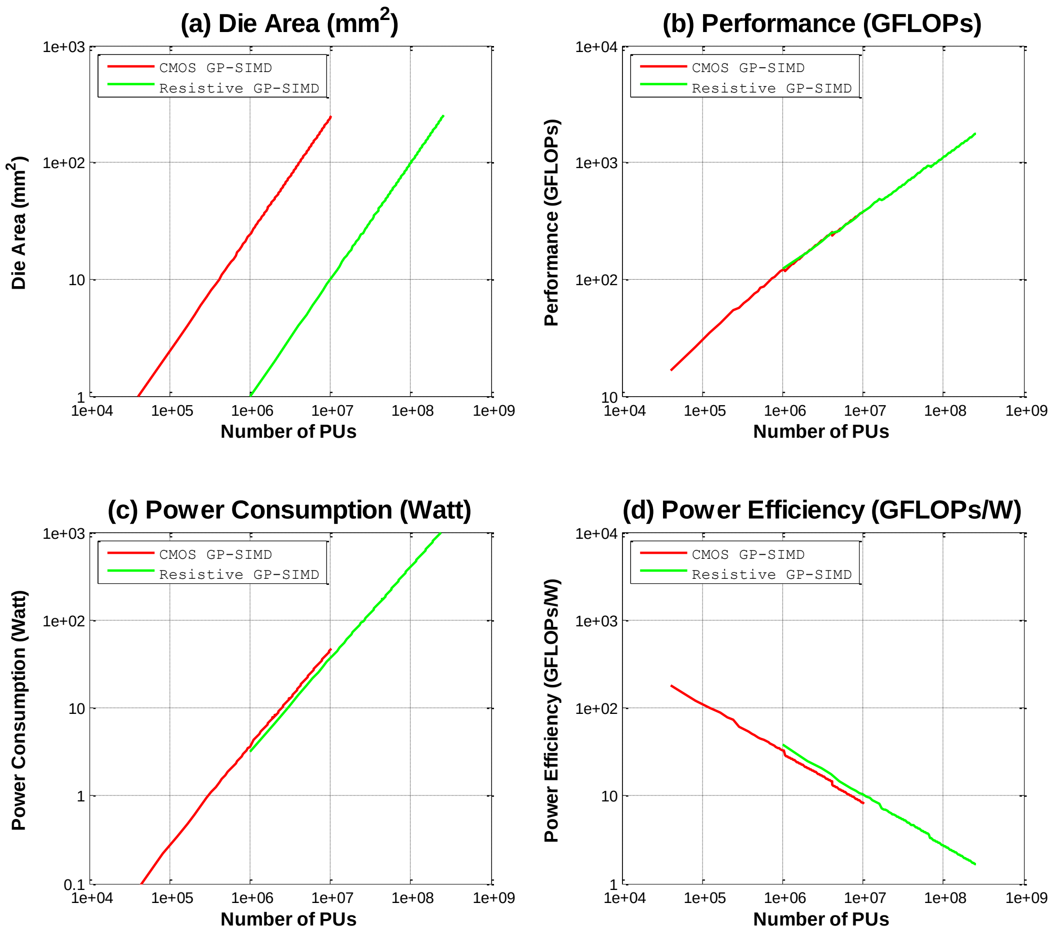


Figure 13. Cycle accurate simulations. CMOS GP-SIMD vs. ReGP-SIMD: (a) Die area; (b) Performance (GFLOPs); (c) Power (Watt); and (d) Power efficiency (GFLOPs/W).

4.1.1. GP-SIMD vs. ReGP-SIMD having identical PU count

At area of 196mm², the CMOS GP-SIMD integrates 8.1M PUs and achieves maximum DMM performance of 337 GFLOPS while consuming only 35.9W. The

Resistive GP-SIMD having the same number of PUs occupies only 8mm^2 and achieves the same DMM performance of 337 GFLOPS while consuming only 29.4W (TABLE 3).

TABLE 3
GP-SIMD vs. ReGP-SIMD having identical number of PUs, executing DMM

	Resistive GP-SIMD	CMOS GP-SIMD	Ratio (Resistive / CMOS)
Die Area (mm^2)	8	196	0.04
Number of PUs ($\times 10^6$)	8.1	8.1	1
Matrix Dimension ($\sqrt{N}$)	2844	2844	1
GFLOPS	337	337	1
Power (W)	29.4	35.9	0.8
Power Density (W/mm^2)	3.67	0.18	20.1
Power Efficiency (GFLOPs/W)	11.5	9.4	1.2

Given the same number of PUs, the CMOS GP-SIMD occupies area 24.5 times larger than the resistive version and achieves only about four-fifths of the power efficiency. This is also evident in Figure 13. The significant area savings of ReGP-SIMD may outweigh the additional cost of the resistive layer making ReGP-SIMD the preferred solution for cost effective implementation.

4.1.2. Area Constrained ReGP-SIMD

At area of 200mm^2, the CMOS GP-SIMD contains 8.2M PUs and achieves maximum DMM performance of 340 GFLOPS while consuming only 36.6W (TABLE 4). Given the same area, the ReGP-SIMD would consume unrealistic power of ~830W. At these figures, ReGP-SIMD achieves about 5 times the performance of the CMOS version, but only about one-fifth of its power efficiency.

TABLE 4
GP-SIMD vs. ReGP-SIMD (constrained area 200mm^2) executing DMM

	Resistive GP-SIMD	CMOS GP-SIMD	Ratio (Resistive / CMOS)
Die Area (mm^2)	200	200	1
Number of PUs ($\times 10^6$)	202	8.3	24.5
Matrix Dimension ($\sqrt{N}$)	14213	2873	4.9
GFLOPS	1564	340	4.6
Power (W)	830.3	36.6	22.7
Power Density (W/mm^2)	4.15	0.18	22.7
Power Efficiency (GFLOPs/W)	1.9	9.3	0.2

4.1.3. Power Ratio Analysis

CMOS GP-SIMD and the ReGP-SIMD software models are identical, and for the purpose of simulation, the very same code is executed on both, making the number of read/write operations to the shared memory array identical. TABLE 4 states that while the number of ReGP-SIMD PUs is 24.5× larger than the number of CMOS GP-SIMD for the same silicon area, power dissipation of ReGP-SIMD is only 22.7× higher than that of CMOS GP-SIMD. This effect is explained by observing DMM implementation on both architectures. For large matrices, broadcast consumes the largest part of compute time. Broadcast consists of many reads and a single write [28]. Since the energy for reading

ReRAM cells is significantly lower than reading CMOS memory, power increases less than the increase in the number of PUs.

4.1.4. Power Constrained ReGP-SIMD

To address the high power requirement of ReGP-SIMD, we consider a version constrained to dissipating 150W in TABLE 5. It requires only 39mm^2. A same area CMOS version achieves only ~1/25 of its density and ~1/5 of its performance. Power efficiency remains ~1:5 in favor of the CMOS GP-SIMD, but both versions exhibit similar or better DMM power efficiency (23.6 and 4.8 GFLOPS/W) than NVidia GTX480 GPU (5 GFLOPS/W according to [9] scaled to 22nm CMOS).

TABLE 5
ReGP-SIMD (constrained power 150W) and CMOS GP-SIMD (same area) executing DMM

	Resistive GP-SIMD	CMOS GP-SIMD	Ratio (Resistive / CMOS)
Die Area (mm^2)	39	39	1
Number of PUs (×10^6)	39.4	1.6	24.5
Matrix Dimension ($\sqrt{N}$)	6276	1268	4.9
GFLOPS	721	149	4.8
Power (W)	151.4	6.3	24
Power Density (W/mm^2)	3.88	0.16	24
Power Efficiency (GFLOPs/W)	4.8	23.6	0.2
GTX480 Performance (GFLOPs/W)		5	

4.1.5. Area and Power Constrained ReGP-SIMD

When both area and power constraints are applied, CMOS and resistive GP-SIMD exhibit the relative ratios shown in TABLE 6 (the highlighted cells shows the limiting factor of each architecture).

TABLE 6
ReGP-SIMD (constrained power 150W) and CMOS GP-SIMD (constrained area 200mm^2) executing DMM

	Resistive GP-SIMD	CMOS GP-SIMD	Ratio (Resistive / CMOS)
Die Area (mm^2)	39	200	0.2
Number of PUs (×10^6)	39.4	8.3	4.8
Matrix Dimension ($\sqrt{N}$)	6276	2873	2.2
GFLOPS	721	340	2.1
Power (W)	151.4	36.6	4.1
Power Density (W/mm^2)	3.88	0.18	21.2
Power Efficiency (GFLOPs/W)	4.8	9.3	0.5

4.2. Cycle Accurate Simulation, SpMM

In the previous section, ReGP-SIMD and CMOS GP-SIMD were simulated using the compute intensive Dense Matrix Multiplication (DMM) benchmark. In this section, ReGP-SIMD and CMOS GP-SIMD are compared using a Sparse Matrix Multiplication (SpMM) benchmark, leading to irregular read and write patterns. Our cycle accurate simulation results are compared with those of nVidea K20 [35], Intel XEON PHI [35] and the Associative Processor (AP) [42]. We followed the methodology outlined in [28] and simulated 1,000 floating-point square matrices with the number of nonzero elements spanning from one hundred thousand to eight million, randomly selected from the collection of sparse matrices from the University of Florida [12]. Figure 14(a), (b) present the selected test-set.

As detailed in [28], the sparse N×M matrix multiplication algorithm has a computational complexity of $O(N_{NNZ}M_{NNZ}logM)$, where N_{NNZ} is the number of nonzero rows of the multiplier matrix A and M_{NNZ} is the average number of nonzero elements per row (equal to number of nonzero elements of the multiplier matrix A, denoted by A_{NNZ}, divided by N_{NNZ}), and the efficiency grows with the number of multiplier matrix's average nonzero elements per row, and with the number of multiplicand matrix's columns. Further, GP-SIMD performance depends on data word-length rather than on data set size. If matrix elements are presented in a floating-point format, the word-length is 32 bit (IEEE754 single precision). Data set size in SpMM typically equals the number of nonzero elements in the sparse matrix.

Although the efficiency of the GP-SIMD SpMM algorithm [28] grows with the number of columns of the multiplicand matrix, for fair comparison we limit our analysis to multiplicand matrices of 16 columns, as used in [35]. Further, we assume that both ReGP-SIMD and CMOS GP-SIMD integrate 8 million PUs. We consider test-case sparse multiplier matrices having 1M columns or less and up to 8M nonzero elements, and dense multiplicand matrices having 16 columns. Figure 14(c) presents the sparse by dense matrix multiplication execution times for these matrices.

The spread in execution times depicted in Figure 14(c) stems from the sensitivity of the GP-SIMD SpMM algorithm to the average number of non-zero elements per row. This sensitivity is shared, although possibly to a lesser extent, by conventional SpMV and SpMM implementations (on GPU or multi-core) [19][25]. Since the average of nonzero elements per row in our test-set is somewhat capped (see Figure 14(b)), as the number of nonzero elements grows, so does the average number of rows and columns of matrices. The execution time of Broadcast command (see [28]) depends on both the number of columns of matrix A and the number of rows. As the matrix gets larger (large number of rows each having a small number of nonzero elements), the execution time hinders the power efficiency of GP-SIMD. For two matrices with a similar number of nonzero elements, the difference of two orders of magnitude in the average number of nonzero elements per row results in a similar difference in the execution time.

The performance of the GP-SIMD SpMM algorithm as a function of the average number of nonzero elements per row is presented in Figure 14(d). The figure demonstrates a close to logarithmic dependency of the GP-SIMD SpMM algorithm performance on the average number of nonzero elements per row. Hence, if the average number of nonzero elements per row is small (which is consistently the case in the University of Florida collection matrices), the effectiveness of the ReGP-SIMD and CMOS GP-SIMD SpMM algorithm is limited. Also shown in Figure 14(d) are performance data for AP and two commercial processors (Intel XEON-PHI and K20 [35]). The simulated power consumption of the GP-SIMD SpMM and AP (as well as reported power of NVidia K20 [13]) is presented in Figure 14(e).

The ReGP-SIMD and GP-SIMD SpMM power efficiency is in the range of 0.1 to 100 GFLOPS/W (Figure 14(f)) with ReGP-SIMD taking the lead thanks to power efficient resistive implementation. The power efficiency declines with the number of nonzero elements (requiring higher power consumption). The SpMM/SpMV power efficiency of advanced contemporary GPUs such as NVidia's K20 and GTX660 is in the 0.1-0.5 GFLOPS/W range [13]. A wide variety of multicore processors such as quad-core AMD Opteron 2214, quad-core Intel Xeon E5345, eight-core Sun UltraSparc T2+ T5140 and eight-SPE IBM QS20 Cell reportedly reach the SpMM power efficiency of up to 0.03 GFLOPS/W [40]. Several FPGA SpMV and SpMM implementation were proposed (for example [28][10]), however these studies focused on optimization of performance or energy-delay, and power dissipation figures were not reported. The ReGP-SIMD, GP-SIMD and AP power efficiency advantage stem from in-memory computing (there are no data transfers between processing units and memory hierarchies) and from low-power design made possible by the very small size of each processing unit.

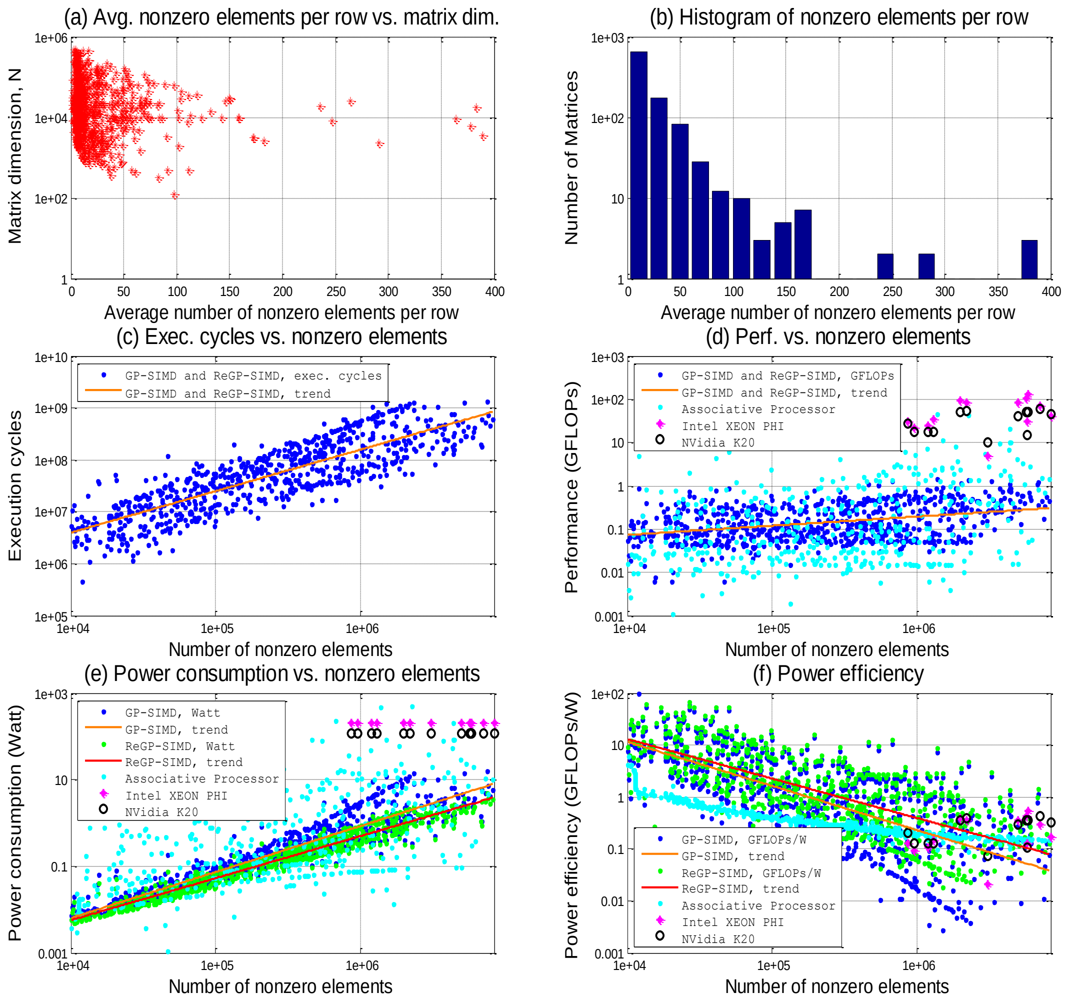


Figure 14. (a) University of Florida Sparse Matrix Collection (1000 matrices) matrix dimension *vs.* average number of nonzero elements per row, (b) Histogram of the average number of nonzero elements per row, (c) Execution cycles *vs.* number of nonzero elements, (d) Performance (GFLOPs) *vs.* number of nonzero elements, (e) Power consumption (Watt) *vs.* average number of nonzero elements, (f) Power efficiency (GFLOPs/W) *vs.* average number of nonzero elements

4.3. Additional Considerations

While the shared memory array benefits from ReRAM scalability, the PU however is CMOS based and thus limits the scalability of ReGP-SIMD. A notable advantage of ReGP-SIMD is the ability to place the resistive crossbar on top of the PU logic [24], thus leading to material area savings. The PU consumes an equivalent area to 10 SRAM cells [29], each of $190F^2$. On the other hand, a row consisting of 256 resistive cells each of $4F^2$, consumes only 5.4 equivalent SRAM cell area. Thus, the total row area consisting of both the PU and the resistive storage is only 10 SRAM cells. Due to the near-zero leakage of resistive RAM cell, the number of resistive bits in a single row could have been extended to 475 without having an impact on GP-SIMD overall area and power consumption, as the majority of the ReGP-SIMD area would consists of PUs with shared memory overlaid on top. In addition to saving area, access time by the PU is significantly reduced because no long wires are needed. A hybrid CMOS-memristor PU is possible [21] to further scale the ReGP-SIMD.

The benefit of placing ReRAM cells over the logic is much more pronounced in processing-in-memory architectures than in other multicores. For instance, if CMOS memory occupies 50% of the chip area in a multicore, replacing memory by resistive cells laid on top the logic might cut chip area by two. In contrast, CMOS GP-SIMD memory occupies more than 96% of chip area and converting to ReGP-SIMD (where the entire memory area is eliminated) may result in area reduction by a factor of 25.

Another noticeable power advantage of ReGP-SIMD is its near-zero leakage power of the crossbar resistive array, while static power consumption in GPUs or CMOS GP-SIMD cannot be neglected [29]. If a way to reduce the write energy of ReRAM, and a more efficient implementation of the PU logic are found, so that the ReGP-SIMD performance is no longer limited by its power density, it can reach above 3.9 TFLOPs/s while processing data sets of more than 200M elements in a $200mm^2$ die, as shown in TABLE 4.

Another factor potentially limiting the use of ReGP-SIMD is the resistive memory endurance, which is in the range of 10^{12} [30]. Given that during arithmetic operations (*cf.* [29]), with write operation typically occurring once every third cycle (taking vector add as an example, and considering only SIMD accesses to the array), the probability of a single bit to be written is $\left(1/_{256}\right)\cdot\left(1/_{3}\right) \approx 1\cdot 10^{-3}$ per cycle. At 1GHz, it limits the endurance-driven MTBF of a ReGP-SIMD to $\sim 1\cdot 10^{6}$ sec, less than two weeks. However, recent studies predict that the endurance of resistive memories is likely to grow to the $10^{14} - 10^{15}$ range [30][14], which may extend the endurance-driven MTBF of a ReGP-SIMD to a number of years.

Lastly, we note that the GP-SIMD retains the entire data structure (two large dense matrices, temporaries and program variables) on chip, and thus neither time nor power are spent for data exchange with off-chip memory. Note further that a 200M rows GP-SIMD array having 256 bits per row translate to 6.4GByte on-chip memory. However, the architect may increase the number of bits per row to 475 (matching PU width) without changing the characteristics of ReGP-SIMD. Additional layers may be used to increase the overall ReGP-SIMD on-chip storage, further reducing off-chip communication bandwidth (and hence power) in cases of very large work-load. For example, a 200mm^2 die may have 11.8 GByte (475 bits per row) on-chip storage, 23.6 GBytes with double ReRAM layer, and so on.

5. FUTURE RESEARCH DIRECTION: RERAM MULTI-VALUED ARITHMETIC

Multi-valued resistive memory may become available in the future [27][37][8][41][45][33]. Certain such devices may portray a linear range of operation: the resistivity may be set to a value k by applying a certain fixed voltage for time kt_0 (a ‘pulse’), where t_0 is a constant time unit. Moreover, the device is additive: If a first pulse lasting k_1 time units is followed by a second pulse lasting k_2 time units, the resulting stored value is k_1+k_2 (as long as the result lies within the range of allowed values for the multi-valued device). Reading a multi-valued cell requires converting the voltage to a digital number, e.g. by means of an analog-to-digital converter at the periphery of the ReRAM array. Writing, on the other hand, calls for converting a digital number into pulse duration.

Addition may be performed by passing a timed voltage pulse corresponding to the first operand, to the resistive cell storing the second operand. The voltage is applied in the direction that decreases the resistivity of the cell. Subtraction is performed similarly, by applying the voltage in the opposite direction. To exemplify the accumulation process, assume a resistive cell capable of storing four bits or 16 levels. If the cell stores 9, and we wish to subtract 6, we subject the cell to a $6t_0$ voltage pulse in the direction that increases the resistivity of the cell. At the end of the operation, the cell contains 3. Since no carry operation is implemented, this is saturation arithmetic (note that we assume power-of-2 multi-valued cells).

The single bit ReRAM cell dissipates about 1fJ when driven from R_{ON} to R_{OFF} and vice versa. Multi-valued arithmetic requires driving the cells from and to levels within the ReRAM's dynamic range limited by R_{ON} and R_{OFF}. Thus, a single cell arithmetic operation may dissipate up to 1fJ. However, the peripheral circuitry incorporates A/D converter for reading out the contents of the multi-valued cell, taking larger area and consuming more power. These parameters have been simulated and validated using the TEAM model [20][22][43] and the SPICE simulator.

Assume *N* bit multivalued ReRAM Cell storing a 2's complement binary number in which the last (most significant) bit represents the overflow ("O"), and the second to last bit represents sign ("S"). The addition and subtraction flow diagram is depicted in Figure 15. Initially, the first operand (A) is stored in the ReRAM cell. The cell is then subjected to a timed pulse corresponding to the second operand (B). The cell accumulates both operands and retains the result in-place. The result is read back to the PU, and subsequently overflow condition is tested as follows:

The PU retains the sign bit of the operands (OPA_SIGN, OPB_SIGN) and result sign bit (RES_SIGN). Following arithmetic operation,

```
If (OPA_SIGN == OPB_SIGN == 0) && (RES_SIGN ==1)
    //both operands are positive and the result is negative
    Overflow detected
elseif (OPA_SIGN == OPB_SIGN == 1) && (RES_SIGN ==0)
    //both operands are negative and the result is positive
    Overflow detected

If(O==1)
    Reset "O" bit
```

The overflow detection protects from overflowing the cell's dynamic range. If overflow is found, saturation is applied followed by write-back. When the overflow ("O" bit) becomes "1", the ReRAM is driven with sufficient interval to reset it back to zero, preserving the result that has not over-flown. Thus, fixed point *m* bit addition/subtraction consumes, worst case, four cycles, regardless of the ReGP-SIMD vector size. Note that we assume that it takes a single cycle to switch the resistive cell from low to high resistivity (i.e., from the value 0 to *N*-1 in *N* bit multivalued cell).

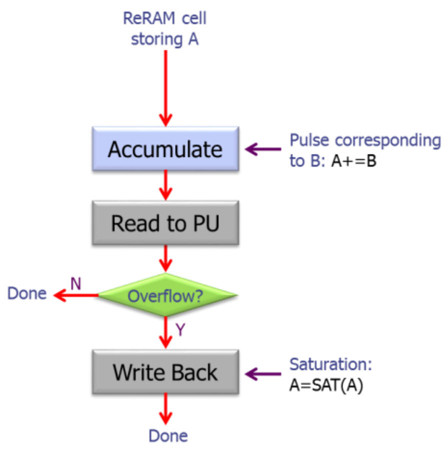


Figure 15. Addition and Subtraction take four cycles

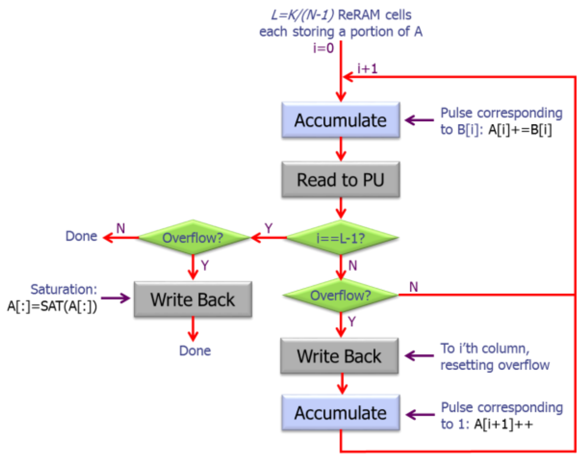

L=K/(N-1) ReRAM cells each storing a portion of A
i=0
i+1
Accumulate
Pulse corresponding to B[i]: A[i]+=B[i]
Read to PU
i==L-1?
Y
Overflow?
N
Done
Y
Write Back
Saturation: A[:]=SAT(A[:])
Done
N
Overflow?
N
Y
Write Back
To i'th column, resetting overflow
Accumulate
Pulse corresponding to 1: A[i+1]++

Figure 16. Long word Addition and Subtraction; each loop iteration takes six cycles.

A long word containing K bits (including the sign bit), where K>N-1, may be partitioned into L=[K/(N-1)] ReRAM cells. The long addition and subtraction flow diagram is depicted in Figure 16. Initially, the first operand (A) is partitioned into l ReRAM cells (the sign bit is allocated only in the most significant cell). Addition of the first cell is performed in the same manner as above. If overflow is found, the overflow bit is reset and carried to the next cell. First the carry is added to that cell, followed by the second operand's N-1 bits. The process repeats until all L ReRAM cells are addressed. The last step is testing for saturation condition; if the last cell is saturated, all previous L-1 cells are set to saturation. Thus, fixed point K bit vector addition, where K>N-1, requires 6×[K/(N-1)]+L+5 ∈O(K/(N-1)) cycles, regardless of the ReGP-SIMD vector size.

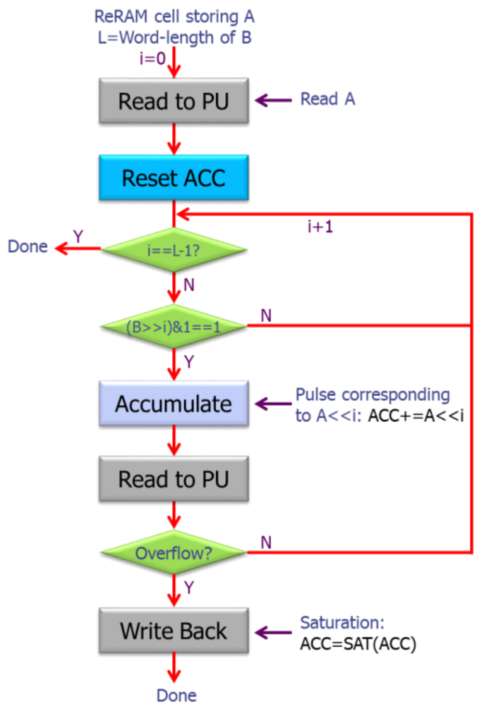


Figure 17. In-cell (m×m) Multiplication takes 1+5m cycles

Fixed point single-cell multiplication and division in multi-valued ReGP-SIMD are computed serially, performing a series of add-shift and subtract-shift vector operations (see flow diagram in Figure 17). When multiplying A×B, and assuming the product may be stored in a single ReRAM cell (that is, $\log_2 A \times B < N - 1$), both operands are fetched to the PU. If the i^{th} bit of B is one, the operand A (left-shifted by i places) is added to the accumulator which resides in the ReRAM. For each i, the loop takes five cycles. Thus, fixed point m×m bit vector multiplication requires $1+5m \in O(m)$ cycles. The above may be extended to long word operands following the flow-diagram of Figure 16. Cell reset is accomplished by reading the content of the multi-valued cell and adding to it the negated 2's complement content. The fixed point single-cell multiplication may be extended to fixed point long word multiplication in a similar manner to long word addition.

The potential energy and area savings of multi-valued ReRAM computation is a promise worth investigating. In a standard 45nm CMOS process, a register file access consume 6pJ (12pJ for read and write) while an 8bit integer addition consume only 0.03pJ [15]. The total 8bit load/add/store operation thus entails 12.03pJ. With an 8bit multi-bit ReRAM cell, the energy for the same operation is expected to be less than 1fJ, four orders of magnitude lower energy than typical CMOS. More so, the addition is performed on a tiny $4F^2$ cell, with processing time similar to SRAM cell access time.

This ReRAM multi-valued arithmetic capability can potentially transform the resistive crossbar to a massively parallel multi-cell processor grid, capable of executing many calculations in parallel, in a small footprint. Further, multi-valued arithmetic may significantly improve ReGP-SIMD power efficiency as the operations are performed right on the cell, concurrently for the entire cell-column, rather than moving information in bit-wise manner back and forth between the cell-column and the PUs.

6. CONCLUSIONS

GP-SIMD architecture combines a general purpose sequential processor and a massive parallel SIMD co-processor with a large shared memory array. Since the shared memory is accessible by both processors, data is not transferred (synchronized) back and forth between separate memories of the two processors. Unlike conventional processors or SIMD accelerators, GP-SIMD's core component (and by far, the largest area-wise) is the shared memory array, thus it stands to benefit the most from a reduced bit-cell architecture offered by a resistive crossbar.

This paper explores a resistive design of the GP-SIMD (ReGP-SIMD), which has the potential to scale the GP-SIMD from a few millions of PUs to a few hundred millions of PUs on a single silicon die. In resistive GP-SIMD, a novel resistive row and column addressable $4F^2$ crossbar is utilized, replacing the modified CMOS $190F^2$ SRAM storage previously proposed for GP-SIMD architecture. We find that PiM architectures and specifically, GP-SIMD, benefit more than other many-core architectures from using resistive memory.

Our simulations show that although high power density and finite endurance of the ReRAM and the CMOS PU limit the potential of ReGP-SIMD, it allows significantly better scalability and performance as compared to a CMOS GP-SIMD as well as other conventional SIMD accelerators. Future progress in ReRAM technologies is likely to offer continuous scalability, improved power efficiency, and higher endurance for ReRAM and thus further enhance ReGP-SIMD. A hybrid CMOS-memristor PUs are possible to further scale the ReGP-SIMD. Additional notable advantage is the placement of the PUs under the resistive shared memory crossbar leading to a significant area savings over the CMOS GP-SIMD. Finally, we noted that the large ReGP-SIMD on-chip memory helps to reduce the latency and power consumption of addressing an off-chip memory.

A conceptual multi-valued arithmetic in a single resistive storage device is explored. Such a device can potentially transform the resistive crossbar to massively parallel multi-cell processor grid, capable of executing a large amount of calculations in parallel, in a small die. Future research is needed to optimize such devices and to devise PiM architectures that realize and extend these density and performance potentials.

ACKNOWLEDGMENT

This research was funded in part by the Viterbi Fellowship and by Hasso-Plattner Institute (HPI).

REFERENCES

[1] Alibart F., *et al*. "Hybrid CMOS/nanodevice circuits for high throughput pattern matching applications," *IEEE Conference on Adaptive Hardware and Systems*, 2011

[2] Alibart F., *et al*. "High precision tuning of state for memristive de-vices by adaptable variation-tolerant algorithm." Nanotechnology 23.7 (2012): 075201

[3] Banerjee K., *et al*., "A self-consistent junction temperature estimation methodology for nanometer scale ICs with implications for performance and thermal management," *IEEE IEDM*, 2003, pp. 887-890.

[4] Burger D., T. Austin. "The SimpleScalar tool set, version 2.0," *ACM SIGARCH Computer Architecture News* 25.3 (1997): 13-25.

[5] Cassuto, S., *et al*., "Sneak-Path Constraints in Memristor Crossbar Arrays," *Proceedings of the IEEE International Symposium on Information Theory*, pp. 156-160, July 2013.

[6] Chang, M. T,, *et al*. "Technology comparison for large last-level caches (L 3 Cs): Low-leakage SRAM, low write-energy STT-RAM, and refresh-optimized eDRAM." *High Performance Computer Architecture (HPCA2013)*, 2013 IEEE 19th International Symposium on. IEEE, 2013.

[7] Chen Y., *et al*., "An Access-Transistor-Free (0T/1R) Non-Volatile Resistance Random Access Memory (RRAM) Using a Novel Threshold Switching, Self-Rectifying Chalcogenide Device," *IEEE IEDM*, pp. 37.4.1-37.4.4, 2003.

[8] Chang M.-F., *et al*., "A 3T1R Non-volatile TCAM Using MLC ReRAM with Sub-1ns Search Time," IEEE International Solid-State Circuits Conference (ISSCC) Dig. Tech. Feb. 2015

[9] Chung E., *et al*. "Single-chip heterogeneous computing: Does the future include custom logic, FPGAs, and GPGPUs?" *43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010.

[10] Colin L., *et al*., "Design space exploration for sparse matrix-matrix multiplication on FPGAs." *International Journal of Circuit Theory and Applications* 41.2 (2013): 205-219.

[11] Cong Xu., *et al*., "Design Implications of Memristor-Based RRAM Cross-Point Structures," *DATE*, pp. 1-6, 2011.

[12] Davis T., *et al*., "The University of Florida sparse matrix collection," *ACM Transactions on Mathematical Software (TOMS)*, 38, no. 1 (2011): 1.

[13] Dorrance R., *et al*., "A scalable sparse matrix-vector multiplication kernel for energy-efficient sparse-BLAS on FPGAs", 2014 ACM/SIGDA *international symposium on Field-programmable gate arrays*.

[14] Eshraghian K., et al. "Memristor MOS content addressable memory (MCAM): Hybrid architecture for future high performance search engines", *IEEE Transactions on VLSI Systems*, 19.8 (, 2011): 1407-1417.

[15] Horowitz, M., "1.1 Computing's energy problem (and what we can do about it)." *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*. IEEE, 2014.

[16] ITRS Roadmap (http://www.itri.net).

[17] Kamil S., *et al*., "An Auto-Tuning Framework for Parallel Multicore Stencil Computations," *IEEE International Symposium on Parallel & Distributed Processing* 2010, pages 1-12.

[18] Kawahara A., et al., "An 8Mb Multi-Layered Cross-Point ReRAM Macro With 443MB/s Write Throughput," *IEEE Journal of Solid-State Circuits*, Vol. 48, No. 1, January 2013.

[19] Kurzak J., *et al*., "Scientific Computing with Multicore and Accelerators", CRC Press, Inc., 2010.

[20] Kvatinsky S., *et al*. "Models of Memristors for SPICE Simulations," *Proceedings of the IEEE Convention of Electrical and Electronics Engineers in Israel*, pp. 1-5, November 2012.

[21] Kvatinsky S., *et al*. "MRL - Memristor Ratioed Logic," *Cellular Nanoscale Networks and Their Applications (CNNA), 2012 13th International Workshop on*, vol., no., pp.1,6, 29-31 Aug. 2012.

[22] Kvatinsky S., *et al*. "TEAM: threshold adaptive memristor model", *IEEE Transactions on Circuits and Systems* I, 2013.

[23] Lauwereins R., "New Memory Technologies and their Impact on Computer Architectures." HiPeac'15

keynote, 2015.
[24] Liu T-Y., *et al.*, "A 130.7 mm^2 2-Layer 32 Gb ReRAM Memory Device in 24 nm Technology," *Proceedings of the IEEE International Solid-State Circuits Conference*, pp. 210-211, February 2013.
[25] Liu X., *et al.*, "Efficient sparse matrix-vector multiplication on x86-based many-core processors", *International conference on supercomputing, ACM*, 2013.
[26] Meng-Fan C., *et. al.,* "A 0.5V 4Mb logic-process compatible embedded resistive RAM (ReRAM) in 65nm CMOS using low-voltage current-mode sensing scheme with 45ns random read time," *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International* pp.434,436, 19-23 Feb. 2012.
[27] Ming-Chi W., *et. al.*, "Low-Power and Highly Reliable Multilevel Operation in ZrO2 1T1R RRAM," Electron Device Letters, IEEE, vol.32, no.8, pp.1026, 1028, Aug. 2011.
[28] Morad A., *et. al.*, "Efficient Dense And Sparse Matrix Multiplication On GP-SIMD." *Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Sept. 2014.
[29] Morad A., *et. al.*, "GP-SIMD Processing-in-Memory." *ACM Transactions on Architecture and Code Optimization (TACO)*, Dec. 2014.
[30] Nickel K., "Memristor Materials Engineering: From Flash Replacement Towards a Universal Memory," *Proceedings of* the IEEE International Electron Devices Meeting, December 2011.
[31] Niu D., *et al.*, "Design Trade-Offs for High Density Cross-Point Resistive Memory," *ISLPED*, 2012, pp. 209-214.
[32] Ou E., *et. al.*, "Array Architecture for a Nonvolatile 3-Dimensional Cross-Point Resistance-Change Memory," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2158-2170, Sep. 2011.
[33] Patel, R., *et al.*, "Arithmetic encoding for memristive multi-bit storage," VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on, vol., no., pp.99,104, 7-10 Oct. 2012.
[34] Patel R., *et al.*, "Multistate Register Based on Resistive RAM", *IEEE Transactions on VLSI*, 2014.
[35] Saule E., *et al.*, "Performance Evaluation of Sparse Matrix Multiplication Kernels on Intel Xeon Phi." arXiv preprint arXiv:1302.1078 (2013).
[36] Seungbum H., *et al.*, "Emerging Non-Volatile Memories". Springer, 2014.
[37] Shyh-Shyuan S., *et. al.*, "A 5ns fast write multi-level non-volatile 1 K bits RRAM memory with advance write scheme," VLSI Circuits, 2009 Symposium on , vol., no., pp.82,83, 16-18 June 2009.
[38] Thatcher J., *et al.*, "NAND Flash Solid State Storage for the Enterprise, An in-depth Look at Reliability," *Solid State Storage Initiative (SSSI) of the Storage Network Industry Association (SNIA)*, April 2009.
[39] Torrezan A., *et al.*, "Sub-nanosecond switching of a tantalum oxide memristor." *Nanotechnology* 22.48 (2011): 485203.
[40] Williams S., *et al.*, "Optimization of sparse matrix–vector multiplication on emerging multicore platforms." *Parallel Computing* 35, no. 3 (2009): 178-194.
[41] Wong, H.-S.P., *et al.*, "Metal–Oxide RRAM," Proceedings of the IEEE, vol.100, no.6, pp.1951,1970, June 2012.
[42] Yavits L., *et al.*, "Computer Architecture with Associative Processor Replacing Last Level Cache and SIMD Accelerator," *IEEE Trans. On Computers*, 2014
[43] Yavits L., *et al.*, "Resistive Associative Processor", *Computer Architecture Letters*, 2014.
[44] Yavits L., *et al.*, "The effect of communication and synchronization on Amdahl's law in multicore systems", *Parallel Computing Journal*, 2014.
[45] Zangeneh, M., *et al.*, "Design and Optimization of Nonvolatile Multibit 1T1R Resistive RAM," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* , vol.22, no.8, pp.1815,1828, Aug. 2014.
[46] Zhuo L., *et al.*, "Sparse matrix-vector multiplication on FPGAs." *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, pp. 63-74. ACM, 2005.