Resistive Memory for Approximate Program Acceleration

Mohsen Imani
CSE, UC San Diego
La Jolla, CA 92093, USA
moimani@ucsd.edu

Yan Cheng
CSE, UC San Diego
La Jolla, CA 92093, USA
yacheng@ucsd.edu

Tajana S. Rosing
CSE, UC San Diego
La Jolla, CA 92093, USA
tajana@ucsd.edu

***Abstract*—The Internet of Things significantly increases the amount of data generated that strains the processing capability of current computing systems. Approximate computing can accelerate the computation and dramatically reduce the energy consumption with controllable accuracy loss. In this paper, we propose a Resistive Associative Unit, called RAU, which approximates computation alongside processing cores. RAU exploits the data locality with associative memory. It finds a row which has the closest distance to input patterns while considering the impact of each bit index on the computation accuracy. Our evaluation shows that RAU can accelerate the GPGPU computation by $1.15\times$ and improve the energy efficiency by 36% at only 10% accuracy loss.**

I. Introduction

In 2015, the number of smart devices around the world exceeded 25 billion. This number is expected to double by 2020 [4]. The rate of data generation by the Internet of things (IoT) will quickly overtake the capabilities of current computing systems. The need for systems that can efficiently handle such large volumes of streaming data is undeniable. This computational reduction is also essential in real time processes of varieties of fields including robotics [3], biology [8], etc. Running machine learning algorithms or multimedia applications on the general purpose processors, e.g. GPU, results in large energy and performance inefficiency. Many of these applications do not need highly accurate computation. Instead of doing all computation precisely, we can instead get the energy and performance advantages while accepting a slight loss in accuracy of computation [27], [35].

Computing in-memory can efficiently perform the computation without using processors. Associative memory, in a form of a lookup table, stores commonly seen patterns and retrieves them at runtime [2], [9], [10], [13]–[17], [19]–[21]. These memory blocks have application in a wide domain such as query processing [2], [16], text processing [25], search engine [11], [28], image processing [1], [7], [18], pattern recognition and classification [12], [23], [24]. In hardware these memories can be implemented using ternary content addressable memory (TCAM). TCAMs search for an input data among all rows in parallel within a single cycle. In CMOS technology, TCAMs are designed with SRAM cells and consume a lot of energy for both store and search operations [6]. Non-volatile memories, such as resistive randome access memory (ReRAMs), magnetic RAMs (MRAMs) and ferroelectric FETs provide a solution for area and energy efficient memory designs [26], [32]–[34]. ReRAMs with high density and performance have very low endurance, especially for write intensive applications. In contrast, MRAMs have high endurance ($\sim 10^{15}$) but have some issues with temperature stability and write latency. Prior work addressed the temperature issue [30], [31] and write latency of MRAMs [22]. NVM-based TCAMs can use voltage overscaling (VoS) to inexactly match the input patterns with prestored values while controlling for Hamming distance error as a function of voltage [14]. In most existing work, associative memories are used next to the processing cores for computational reuse to reduce the redundant computations or to enable error free execution. However, large data locality in multimedia and learning applications alongside with low search energy consumption of NVM-based TCAMs motivate us to use associative memories directly as a computation building blocks without processors.

An associative memory can quickly recall responses of a function for a subset of input patterns to save energy by avoiding the actual function execution on the processing element. An associative memory is typically composed of a ternary content-addressable memory (TCAM) to store input patterns and an output memory to return the pre-stored output. The operation of a TCAM goes beyond retrieving logic "0" and "1" and it has capability to store and search wildcard [5]. This feature opens the application of the TCAMs for approximate computing domain, and a wide range of applications in query processing [2], [16], text processing [25], search engine [11], [28], image processing [1], [7], [18], pattern recognition and classification [12], [23].

II. Resistive Associative unit

Resistive associative unit can perform faster and more energy efficient computation. However, in order to use RAU as a computing unit, it has to store all possible preprocessed patterns. For a simple floating point adder, RAU needs a TCAM with 264 rows. Clearly, energy and latency limitations make such large TCAM blocks infeasible. We instead use small size associative memories, shown in Fig. 1, that store a few high frequency patterns, to represent each processing unit. For the data that does not exactly match what is stored in TCAM, our design returns an output which has the lowest error. All bits do not have the same impact on the computation result. In floating point computation, bits in the position i-th of mantissa have two times higher impact on the computation accuracy than $i-1^{th}$ bit ($1 < i < 1 < 24$). Applying selective approximation (i.e. voltage overscaling) relaxes the computation on least significant bits with the minimal impact on the computation accuracy [9]. Our proposed limits the number of acceptable Hamming distances to the least significant block (LSB) where the mantissa bits (0-23 bits in floating point values) are stored. Indeed, at each search operation our design finds a row with the nearest weighted Hamming distance with the input pattern, considering bit impact.

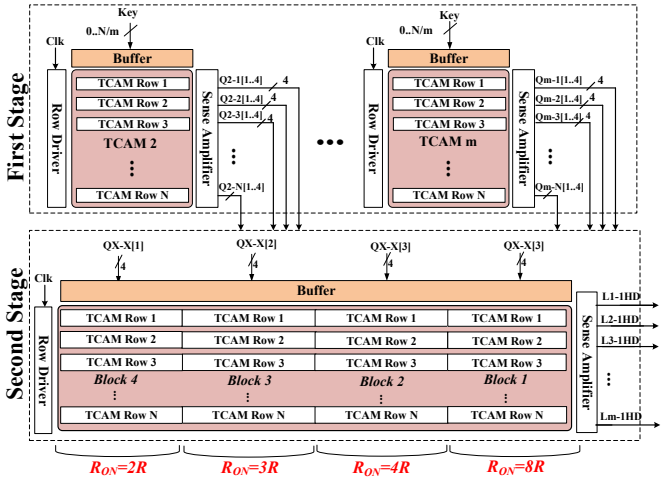

Fig. 1. Resistive associative memory with capability of weighted nearest Hamming distance search.

To provide this capability, our design splits TCAM to m partial stages where each TCAM can search for N/m bit in parallel. The output of each TCAM row is a $log(N/m)$ bits which represents the distance of the input key with the stored value. The difference between the mismatches on partial TCAM (¡8-bit) can be determined by having multiple sense amplifiers with different sampling periods. Data splitting allows us to implement selective approximation on the TCAM blocks in order to achieve maximum energy savings while delivering needed accuracy. The next stage starts counting the number of mismatches in all partial TCAMs in each row and finds a row that has the minimum distance to the input key. This stage can be implemented using another resistive CAM structure with different ON resistances in bits, to give weight to mismatches in the most significant blocks. The first block (Block1) takes the result of sense amplifier corresponding to 1-bit Hamming distance. Similarly, the next set of blocks (Block 2, 3 & 4) search for the mismatches corresponding to higher distances (2-bit, 3-bit and 4-bit distance respectively). This technique allows us to consider the real distance of the input pattern with the stored patterns, instead of just trusting the Hamming distance metric.

Fig. 2 shows the overview of the proposed Resistive GPU (ReGPU) architecture utilizing RAU blocks. ReGPU has two types of floating point units (FPUs): **i)** fast and energy efficient RAU for approximate computing and **ii)** conventional CMOS-based FPUs to process precise computations. In GPGPU, each SIMD lane contains four main floating point units: Adder (ADD), Multiplier (MUL), Multiply-accumulator (MAC) and SQRT. FPUs accept different number of input operands. The ADD and MUL accept two 32-bit, SQRT one 32-bit and MAC three 32-bit input operands, corresponding to 64-bit, 32-bit and 96-bit word sizes in TCAMs. We profile 10% of the input dataset offline to find the highest frequency patterns corresponding to each of the FPUs and then fill the RAUs accordingly. At runtime, RAUs update in parallel to applications running on cores and the computation runs completely on RAU (FPUs are clock gated). In order to reduce the area overhead of additional circuitry,

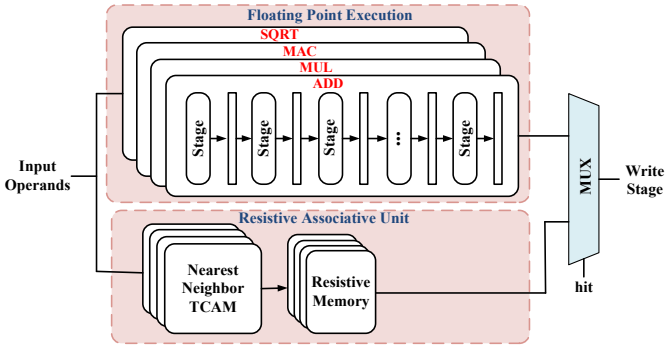

Fig. 2. Hybrid FPUs using resistive associative units.

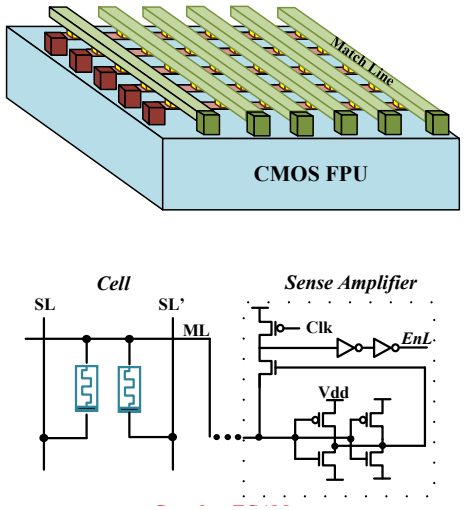

Fig. 3. Crossbar associative memory with FPUs..

both TCAM and resistive memories are implemented with crossbar memristive technology. Fig. 3 shows the structure of crossbar memories and how they can be implemented on top of the floating point units with negligible area overhead.

There are two key advantages of the resistive hybrid GPUs. First, RAU can process operations in m cycles; $m-1$ search cycles and a cycle to read the output, where m is 3 or 4 based on the RAU size. While the floating point units require deep pipeline stage to process the data (23 cycles in AMD Southern Island GPU). This significantly accelerates FPU computation. Secondly, associative computing with 512 rows requires $4\times$ lower energy than FPUs for each computation. For error-tolerant applications processing the computation on RAU can result in significant GPGPU energy savings.

The energy and performance advantages are achieved by trading off accuracy. When RAU cannot find the exact result for a search operation, it returns a value that has the closest distance to the input. Accuracy is tuned by using a sufficient number of rows per TCAM block. Larger TCAM also has higher delay and energy consumption. Fortunately, for some machine learning algorithms (e.g. convolutional neural network) and multimedia applications (e.g. Robert, Shift), small TCAMs with ¡256 rows can grantee computation

TABLE I
NORMALIZED ENERGY CONSUMPTION, SPEED UP AND QUALITY OF SERVICE OF PROPOSED RESISTIVE GPGPU.

GP-GPU	GP-GPU	32-row	64-row	128-row	256-row	512-row	1024-row
Robert	Norm. Energy	0.11	0.17	0.29	0.49	0.72	0.89
	Speed up	3.8×	2.7×	1.9×	1.4×	1.2×	1.02×
	Quality of Loss	25.2%	25.1%	24.8%	16.2%	8.3%	7.6%
Shift	Norm. Energy	0.15	0.23	0.34	0.56	0.79	0.97
	Speed up	3.7×	2.3×	1.3×	1.1×	1.02×	0.94×
	Quality of Loss	31.2%	19.4%	14.9%	9.5%	7.4%	5.6%

accuracy of 10%.

III. EXPERIMENTAL RESULTS

We place RAU next to floating point units of AMD Southern Island GPUs. Multi2sim, a cycle accurate CPU-GPU simulator [29] is used for evaluation. TCAMs are modeled with HSPICE to search for input patterns and return a row which has the closest distance to the input key. This design limits the inaccuracy coming from the inexact match by applying weighted approximation. A key hits a row that has the closest real distance (not Hamming distance) to the input key. For example, 4-bit Hamming distance in LSB block has the same impact of having 2-bit Hamming distance in the second LSB block.

Table I shows the speed up and energy savings of hybrid GPGPU using associative memories of different sizes. Increasing the size of RAU improves computation accuracy at the cost of increasing energy consumption and search latency. However, the advantage of using an associative memory of any size is much higher than conventional GPU. Running OpenCL applications Robert and Shift on the hybrid GPGPU results in 1.2× and 1.1× speed up, and 28% and 44% energy savings as compared to the GPGPU computation, at less than 10% accuracy cost.

IV. ACKNOWLEDGMENTS

This work was supported by NSF grant 1527034 and Jacob of Engineering UCSD Powell Fellowship.

REFERENCES

[1] R. Agrawal, R. Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[2] N. Bandi, A. Metwally, D. Agrawal, and A. El Abbadi. Fast data stream algorithms using associative memories. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 247–256. ACM, 2007.

[3] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410. IEEE, 2003.

[4] J. Gantz and et al. Extracting value from chaos. *IDC iview*, 1142:1–12, 2011.

[5] Q. Guo, X. Guo, R. Patel, E. Ipek, and E. G. Friedman. Ac-dimm: associative computing with stt-mram. In *ACM SIGARCH Computer Architecture News*, volume 41, pages 189–200. ACM, 2013.

[6] M. Imani and H. Alimohamadi. A low power and reliable 12t sram cell considering process variation in 16nm cmos. *International journal of technology enhancements and merging engineering research.*, page 76, 2014.

[7] M. Imani, Y. Cheng, and T. Rosing. Processing acceleration with resistive memory-based computation. In *Proceedings of the Second International Symposium on Memory Systems*, pages 208–210. ACM, 2016.

[8] M. Imani and et al. Optimal gene regulatory network inference using the boolean kalman filter and multiple model adaptive estimation. In *2015 49th Asilomar Conference on Signals, Systems and Computers*, pages 423–427. IEEE, 2015.

[9] M. Imani and et al. Resistive configurable associative memory for approximate computing. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1327–1332. IEEE, 2016.

[10] M. Imani, Y. Kim, A. Rahimi, and T. Rosing. Acam: Approximate computing based on adaptive associative memory with online learning. In *ISLPED*, pages 162–167, 2016.

[11] M. Imani, Y. Kim, and T. Rosing. Nngine: Ultra-efficient nearest neighbor accelerator based on in-memory computing.

[12] M. Imani, Y. Kim, and T. Rosing. Mpim: Multi-purpose in-memory processing using configurable resistive memory. In *Design Automation Conference (ASP-DAC), 2017 22nd Asia and South Pacific*, pages 757–763. IEEE, 2017.

[13] M. Imani, P. Mercati, and T. Rosing. Remam: low energy resistive multi-stage associative memory for energy efficient computing. In *Quality Electronic Design (ISQED), 2016 17th International Symposium on*, pages 101–106. IEEE, 2016.

[14] M. Imani, S. Patil, and T. Rosing. Approximate computing using multiple-access single-charge associative memory. 2016.

[15] M. Imani, S. Patil, and T. S. Rosing. Masc: Ultra-low energy multiple-access single-charge tcam for approximate computing. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 373–378. IEEE, 2016.

[16] M. Imani, D. Peroni, Y. Kim, A. Rahimi, and T. Rosing. Efficient neural network acceleration on gpgpu using content addressable memory. In *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1026–1031. IEEE, 2017.

[17] M. Imani, D. Peroni, A. Rahimi, and T. Rosing. Resistive cam acceleration for tunable approximate computing. *IEEE Transactions on Emerging Topics in Computing*, 2016.

[18] M. Imani, D. Peroni, and T. Rosing. Nvalt: Non-volatile approximate lookup table for gpu acceleration. *IEEE Embedded Systems Letters*, 2017.

[19] M. Imani, A. Rahimi, D. Kong, T. Rosing, and J. M. Rabaey. Exploring hyperdimensional associative memory. In *High Performance Computer Architecture (HPCA), 2017 IEEE International Symposium on*, pages 445–456. IEEE, 2017.

[20] M. Imani, A. Rahimi, P. Mercati, and T. Rosing. Multi-stage tunable approximate search in resistive associative memory. *IEEE Transactions on Multi-Scale Computing Systems*, 2017.

[21] M. Imani and T. Rosing. Cap: Configurable resistive associative processor for near-data computing. In *Quality Electronic Design (ISQED), 2017 18th International Symposium on*, pages 346–352. IEEE, 2017.

[22] N. Khoshavi and et al. Read-tuned stt-ram and edram cache hierarchies for throughput and energy enhancement. *arXiv preprint arXiv:1607.08086*, 2016.

[23] Y. Kim, M. Imani, and T. Rosing. Orchard: Visual object recognition accelerator based on approximate in-memory processing. In *International Conference on Computer-Aided Design (ICCAD). ACM*, 2017.

[24] T. Kohonen. *Content-addressable memories*, volume 1. Springer Science & Business Media, 2012.

[25] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, pages 13–24. Ieee, 2007.

[26] M. Saremi. A physical-based simulation for the dynamic behavior of photodoping mechanism in chalcogenide materials used in the lateral programmable metallization cells. *Solid State Ionics*, 290:1–5, 2016.

[27] F. S. Snigdha and et al. Optimal design of jpeg hardware under the approximate computing paradigm. In *Proceedings of the 53rd Annual Design Automation Conference*, page 106. ACM, 2016.

[28] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli. Multi2sim: a simulation framework for cpu-gpu computing. In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, pages 335–344. ACM, 2012.

[29] R. Ubal, B. Jang, P. Mistry, D. Schaa, and D. Kaeli. Multi2sim: a simulation framework for cpu-gpu computing. In *Proceedings of the 21st international conference on Parallel architectures and compilation techniques*, pages 335–344. ACM, 2012.

[30] M. Valad Beigi and et al. Tapas: Temperature-aware adaptive placement for 3d stacked hybrid caches. In *In international Symposium on Memory Systems (MEMSYS), 2016*.

[31] M. Valad Beigi and et al. Tesla: Using microfluidics to thermally stabilize 3d stacked stt-ram caches. In *34th International Conference on Computer Design (ICCD), 2016*.

[32] C. J. Xue and et al. Emerging non-volatile memories: opportunities and challenges. In *Proceedings of the seventh IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 325–334, 2011.

[33] X. Yin and et al. Exploiting ferroelectric fets for low-power non-volatile logic-in-memory circuits. In *IEEE/ACM International Conference On Computer Aided Design, 2016*.

[34] X. Yin and et al. Design of latches and flip-flops using emerging tunneling devices. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 367–372. IEEE, 2016.

[35] R. Zhou and et al. A general sign bit error correction scheme for approximate adders. In *Proceedings of the 26th edition on Great Lakes Symposium on VLSI*, pages 221–226. ACM, 2016.