Multi-stage Tunable Approximate Search in Resistive Associative Memory

Mohsen Imani, *Student Member, IEEE,* Abbas Rahimi, *Member, IEEE,* Pietro Mercati, *Student Member, IEEE,* and Tajana Rosing, *Senior Member, IEEE*

Abstract—General-purpose graphics processing units (GPGPUs), as programmable accelerators, improve energy efficiency by integrating a large number of relatively small cores. In this paper, we focus on improving energy efficiency of such processing core by integrating an associative memory where function responses are prestored. Associative memories can search and recall function responses for a subset of input values therefore avoiding the actual function execution on the processing core that leads to energy saving. We propose a novel low-energy Resistive Multi-stage Associative Memory (ReMAM) architecture to significantly reduce energy of a search operation by employing selective row activation and in-advance precharging techniques. ReMAM splits the search operations in a ternary content addressable memory (TCAM) to a number of shorter searches in consecutive stages. Then, it selectively activates TCAM rows at each stage based on the hits of previous stages, thus enabling energy savings. The proposed in-advance precharging technique mitigates the delay of the sequential TCAM search and limits the number of precharges to two low-cost steps. ReMAM further implements approximation on the selective TCAM blocks to reduce the search energy that relaxes the function output in a fine-grained granularity with very low impact on accuracy of the results. Its multi-stage search operation makes ReMAM applicable to many applications such as search engines, sorting, image coding, pattern recognition, query processing, machine learning. In this work, we show an application of proposed ReMAM on AMD Southern Island GPUs. Our experimental evaluation shows that ReMAM reduces on average GPGPU energy consumption by 35% in the exact mode, and 58% in approximate mode with average relative error lower than 10%. These energy savings are 1.8× and 1.5× higher than state-of-the-art associative memories used in GPGPUs in exact and approximate modes.

Index Terms—Associative memory, Approximate computing, Resistive Memory, GPUs, Ternary content addressable memory (TCAM), Non-volatile memory

1 Introduction

Big data computation demands for massive parallel processing with extremely high energy efficiency [1], [2], [3]. The idea of associative memory has been introduced to reduce the energy consumption of a processing core by recalling a function result and avoiding the actual function execution on the core [4], [5], [6]. An associative memory compares the function inputs with a set of prestored input patterns and readily returns the related result if any. When a match is found during the search operation, the associative memory clock-gates the function execution on the core, enabling energy saving. Associative memories can be implemented in both software and hardware. Software solutions are based on a hashing where frequent patterns can be stored and retrieved from a table using a set of keys [7]. In hardware, they are implemented with Ternary Content Addressable Memory (TCAM) blocks [5], [6], [8]. Associative memories today are directly used in a broad spectrum of applications including query processing [9], search engines, text and image processing, pattern recognition and data mining [10], [11].

CMOS-based TCAM designs have high energy consumption which limits their usage to network and classification applications [12]. Smaller versions of such TCAMs have been used for memoization memoization [13] facilitating instruction reuse in GPGPUs. On the other hand, non-volatile memory (NVM)-based TCAMs offer an opportunity for building more energy-efficient associative memories by providing higher density and lower leakage power [14], [15], [16]. The use of NVM-based TCAMs with approximate search can further reduce energy consumption by voltage overscaling (VOS) [6]. Such VOS produces controllable approximation by accepting a Hamming distance of 1 or 2 between the inputs and the prestored patterns in NVM-based TCAMs that simply cannot be realized in CMOS-based TCAMs due to rapid timing errors [6]. However, during every search operation all TCAM lines are precharged and discharged draining large amount of current.

In this paper we propose Resistive Multistage Associative Memory (ReMAM), a new associative memory architecture that significantly decreases search energy consumption by employing two novel techniques: selective row activation and in-advance precharging. With selective row activation, ReMAM splits the TCAM search into a set of shorter searches and selectively activates rows based on the hit of the previous stages, thus reducing the overall energy consumption. At the same time, in-advance precharging mitigates the delay of such sequential TCAM access. This limits the number of precharges to only two for ReMAM with arbitrary number of stages. Our experimental evaluation on AMD Southern Island GPU archi-tecture running five OpenCL applications shows that the proposed ReMAM architecture reduces the energy consumption of the GPGPU by 35% on average. We also

- *M.Imani, P. Mercati and T. Rosing are with the department of computer science and engineering, University of California San Diego, La Jolla, CA, 92093.*
  E-mail: moimani, dperoni, tajana@ucsd.edu
- *A. Rahimi is with the department of electrical engineering and computer science, University of California Berkeley, Berkley, CA, 94720.*
  E-mail: abbas@eecs.berkeley.edu

implement selective fine-grained approximation on TCAM blocks to reduce both ReMAM and processing energy consumption while ensuring low impact on accuracy. Our evaluation shows that GPGPU using approximate ReMAM achieves 58% energy savings on average with an acceptable average relative error lower than 10%. Energy savings are 1.5× and 1.8× higher for exact and approximate ReMAM compared to GPGPU using conventional single-stage associative memory.

The reminder of the paper is organized as follows. Section 2 reviews the related work. Section 3 describes the architecture of NVM-based associative memories and related challenges. Section 4 presents the proposed ReMAM architecture. Section 5 describes two techniques to enable ReMAM approximation. The experimental results are presented in Section 6. Finally, Section 6 concludes this paper.

2 Related Work

Associative memories in the form of TCAM blocks can exploit pattern similarities of parallel workloads to decrease the amount of redundant computations in GPGPUs or CPU [6], [17], [18], [19], [20], [21]. TCAMs in CMOS technology suffer from low density and high energy consumption [4], [22].

Recent research work uses NVMs as a replacement for CMOS-based TCAMs due to their low search energy, low leakage power and high density [23], [24], [25]. Resistive RAM (ReRAM) and Spin-transfer Torque RAM (STT-RAM) are two types of NVMs based on memristive devices and magnetic tunneling junctions (MTJ) respectively [23], [26], [27]. The endurance of the ReRAMs is limited to 106-107 write operations while STT-RAMs show an endurance of 10^{15} [28], [29]. Li, et al. designed a 1Mb energy efficient 2T-2R (2-Transistor/2-Memristor) TCAM, which is 10× smaller than SRAM-based TCAM [14]. Chang, et al. [30] proposed 3T-1R TCAM cell which performs a search in less than 1-ns using 0.61fJ/search/bit. An efficient 2Kb 4T-2MTJ cell is proposed in [31]. This cell is for standby-power-free TCAM and has 86% area reduction with respect to SRAM-based design. Hanyu, et al. in [32] introduced a 5T-4MTJ TCAM cell with very low energy and high sense merging. Although MTJ-based TCAMs have higher endurance, the ReRAM-based TCAMs have better search speed (ON/OFF resistive ratio) and area efficiency, which makes them more suitable for low energy associative memories. Hence, in this paper we focus on ReRAM technology rather than STT-RAMs. We solely need to program our ReMAM design before executing a new kernel that maximizes device endurance.

Although TCAMs are used in various architectures, we focus on its application for GPGPUs. Various memoization techniques reduce the energy overhead of error recovery by exploiting data locality. For instance, in [13], a single cycle look-up table has been implemented alongside each Floating Point Unit (FPU) to maintain error-free execution. Zhang, et al. [17] leverage VOS for imprecise FPUs in GPU computation suffering from high error rate under VOS. Similarly, many prior work used associative memory for enabling approximate computing on GPU [6], [18], [19], [20], [21], [33]. An approximate associative memristive memory architecture is introduced in [34] to reduce the TCAM energy consumption by applying VOS. Work in [6] applied uniform voltage overscaling on the selected least significant bits/blocks to have low-est accuracy degradation. However, the granularity of approximation is fixed and precise bits still consume huge energy. The above publications show that approximation techniques may severely degrade the computation quality of service (QoS). Work in [35] propose an online learning technique, which considers both data and temporal locality to fill associative memories approximately on runtime.

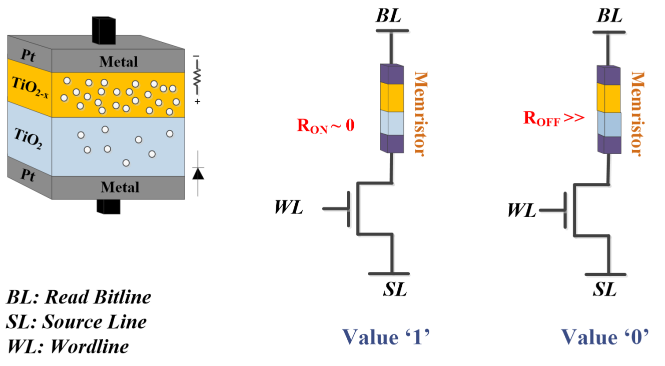


Fig. 1. Working mechanism of ReRAM structure.

Our proposed multi-stage associative memory architecture complements to other TCAM energy reduction techniques, such as VOS. In contrast to the previous designs, ReMAM reduces the energy consumption by decreasing the number of active lines. The proposed architecture gradually reduces the number of active rows with selective row activation. It also mitigates the delay of the multi-stage TCAM block using in-advance precharging technique.

3 Background

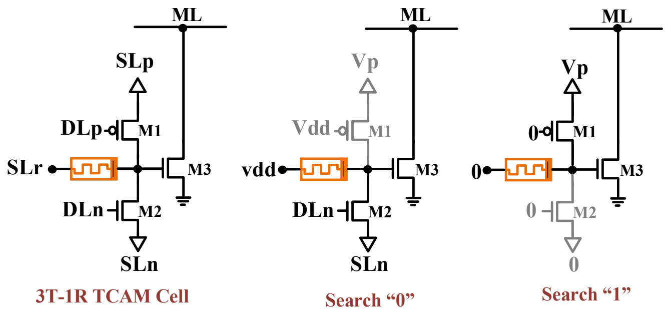
3.1 Memristive Devices and TCAM

Resistive memory technology (ReRAM) has shown a great potential to build high performance NVMs [36]. Figure 1 shows the structure of memristor device and a 1T-1R memory cell designed by memristor. To enable fast switching, ReRAMs use CMOS-based access transistors. The memristor device has alternate layers of metal/oxide/metal. Two metal layers (e.g. Pt) sandwich an oxide layer based on Ta, Ti or HF [37], [38]. The metal TiO_{2-x} connection on the top behaves like a resistance, while the $TiO_2/metal$ connection on the bottom behaves like a diode. It is possible to improve endurance and reduce switching delay to as low as 1-ns by using different combinations of materials, such as $Pd/Ta_2O_{5-x}/TaO_y/Pd$ [38].

Data is pre-stored on cells based on the memristor resistance state. The device can be switched ON by applying negative bias and OFF by applying positive voltage across the device. The search operation is performed by applying a small voltage across the BL and SL and reading the output using a sense amplifier.

3.2 Resistive Associative Memory

Resistive Associative Memory (ReAM) consists of two main blocks: TCAM and resistive 1T-1R memory. A set of frequent input patterns and their corresponding results are pre-stored on



Signals in search operation

Data	*DLn*	*DLp*	*SLr*	*SLp*	*SLn*
0	Vdd	Vdd	Vdd	Vp	0
1	0	0	0	Vp	0

Fig. 2. 3T-1R TCAM cell and search control signals.

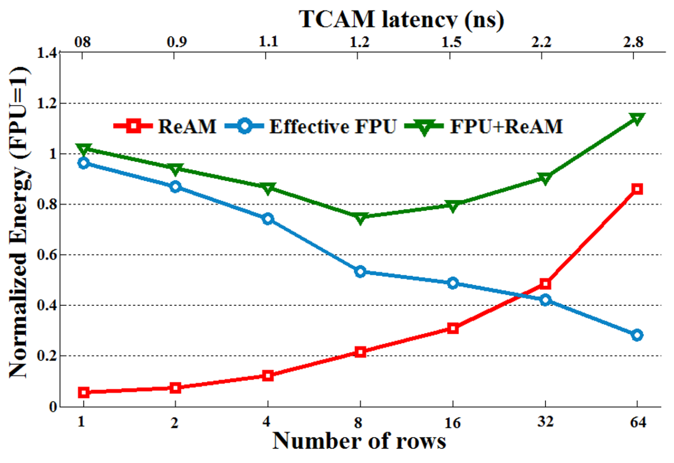


Fig. 3. Normalized FPU+ReAM energy consumption vs. number of TCAM rows.

TCAM and 1T-1R memory respectively. Resistive 1T-1R memory has cells with one memristor and one access transistor. When a computation is issued, the operands are searched in parallel on the TCAM. If there is match, the computing unit is clock-gated, enabling energy saving, and the corresponding result stored in the 1T-1R memory is returned.

TCAMs are the main components of associative memories. In NVM-based TCAMs, values are pre-stored on the cells based on the NVMs resistance state (Low or High). During the search operation, an input operand is compared with all the pre-stored TCAM patterns at the same time. If the data is found, a charged Match Line (ML) interrupts the processor computation by clock-gating. The small ON/OFF current ratio along with large match line capacitance increase both search energy and delay. There should be a significant difference between match and mismatch currents to have an efficient and reliable search in TCAM. Thus, in this work we use 3T-1R TCAM structure [30] which has advantages over the previous TCAM cells [14], [39]. Figure 2 shows the resistive 3T-1R TCAM cell structure and its control signals during the search operation. In 3T-1R TCAM, cells are connected to ML with only one junction to decrease the effective capacitance of ML. Connecting the resistance indirectly to ML (using M_3 transistor) has positive impact on increasing the MLs ON/OFF current ratio. This significantly improves the cell stability and makes it more robust against resistance variations. Figure 2 shows the state of the M_1 and M_2 transistors during searching 0 and 1. To search for 0 value, TCAM control signals deactivate M_1 transistor. Then based on the resistance value, the gate of M_3 will bias with zero or vdd voltages. Similarly, to search for 1 value, M_2 transistor becomes deactivated and the ON or OFF current path through M_3 based on the competition between M_1 and resistance.

3.3 Challenges of Resistive Associative Memory

Several associative memory applications require a TCAM with both large word size and high number of rows to store long keys and improve the hit rate. However, having a large TCAM exacerbates the following issues:

(i) Large word size decreases the stability of the TCAM because of the high leakage currents of cells connected to the match-line (ML). These leakage currents can result in wrong search operations [30], [32], [39]. One solution is to split the TCAM into multiple shorter searches which can be either sequential or parallel. Sequential access increases the search delay and degrades the average time that the processor can be clock-gated. A parallel search requires an `OR` gate to find the line corresponding to all partial TCAM matches. This results in energy and area overhead.

(ii) A TCAM with a large number of rows requires big and power hungry input buffers to distribute the input signals to all rows. The delay introduced by big buffers prevents searching the entire block in a single cycle and degrades the energy efficiency.

(iii) The search operation is energy hungry because of the large ML switching activity which limits efficiency and applications of using associative memories [4], [5]. For example, in GPGPUs there is a tradeoff between the energy consumption of associative memory and FPUs depending on the size of the TCAM. Figure 3 shows the energy consumption of ReAM, effective FPU and FPU+ReAM for Sobel OpenCL benchmark with different TCAM sizes. The effective FPU energy consumption is calculated based on TCAM hit rate and delay for each TCAM size. Graph lines are normalized based on the FPU energy consumption. The result show that in small TCAMs FPU energy dominates GPGPU energy consumption. Doubling TCAM size improves ReAM hit rate, which results in higher effective FPU computation due to clock gating. However, the high energy consumption of ReAM degrades total FPU+ReAM energy for TCAMs with more than 8 rows. The results show that the minimum energy consumption of the ReAM is obtained for an 8-row TCAM, with 24% of energy saving. Any technique to reduce ReRAM energy improves the GPGPU energy consumption and shifts the minimum GPGPU energy point to the larger TCAM size.

Our proposed ReMAM architecture addresses the main limitations of associative memory by adopting a multi-stage structure with sequential access. ReMAM allows us to have a larger number of shorter rows arranged in consecutive stages. In the following section we explain the details of our proposed ReMAM and how our design lowers the energy consumption and mitigates the delay associated with sequential access with selective row activation and in-advance precharging.

4 Proposed ReMAM

4.1 ReMAM Architecture

Figure 4 shows that ReMAM consists of a multi-stage TCAM and a 1T-1R memory block. While standard associative memory has a single TCAM block, ReMAM splits it into m shorter stages. When the computing unit is running, the first N/m bits of each operand are searched in the first stage, where N is the size of the input. In case of a hit, the architecture employs selective row activation by sending EnL_1 signal to the next stage to enable (precharge) only the corresponding lines. Then, it searches the following N/m bits of the input data in the second stage just on the selected rows. This procedure is repeated until the last stage. This technique gradually reduces the number of active rows from 2^{nd} to m^{th} stage. Reducing the number of precharge on active rows saves energy. After traversing all TCAM stages, if the pattern exists in the TCAM, one of the MLs in the m^{th} stage stays high and activates the Enl_m signal. A hit in the m^{th} stage stops the computation by clock gating. At the same time, the hit rows of m^{th} stage activate the corresponding row of 1T-1R memory in order to read the pre-stored results of computation.

Sequential row activation on the TCAM causes a large search delay, because each stage needs to wait for the following one to precharge before moving on. To considerably reduce the delay, we propose in-advance precharging. This technique precharges each row per stage (row driver activation) based on the hit of the second previous stage. For example, in a m-stage TCAM, when data is searched in the k^{th} TCAM stage, the $(k+1)^{th}$ stage precharges the rows based on $(k-1)^{th}$ stage hit $(2 < k < m+1)$. ReMAM still has some delay due to the precharge of the first two stages. The delay reduction achieved by in-advance precharging starts with the third stage. However, these two initial precharging steps can be done quickly for short word size TCAM, with negligible impact.

The application of the proposed low energy associative memory is not limited to GPU processing. ReMAMs multistage operation makes it ideal for a wide variety of applications, such as search engines, searching and sorting, image coding, pattern recognition, query processing and machine learning based processing, as well as other applications of associative memories [10], [11], [20], [40], [41]. In most of these applications, the goal is to search a long key in TCAM. If the first digit is not available, there is no need to go further. For this reason, we consider ReMAM as a promising solution for a broad range of applications.

Figure 5 shows a comparison of an 8-row, 4-stage ReMAM in which selective row activation and in-advance precharging are disabled (Figure 5.a) and enabled (Figure 5.b). The goal is to search for A B C D string in TCAM. Each digit can be stored in one stage. In the first case, the TCAM row activation does not depend on previous TCAM hits, and all stages are precharged at the same time. The proposed ReMAM, instead, activates the rows based on the hit of previous stages. The row activation of the 3^{rd}, 4^{th} and 1T-1R memory is done based on the hits of 1^{st}, 2^{nd} and 3^{rd} stages respectively. The selective row activation significantly reduces the number of active rows, and consequently ReMAM energy consumption. At the same time, in-advance precharging guarantees a consistent delay reduction. The graph shows that the proposed ReMAM has 16 active rows while in conventional TCAM all 32 rows are active and consuming energy. Therefore, we expect about 50% TCAM search energy savings.

4.2 ReMAM & Number of Stages

The number of TCAM stages has a major impact on the associative memory energy consumption. Splitting the TCAM increases the number of partial hits on the first stages and progressively reduces the number of active rows on the following stages and resistive memory. Moreover, it reduces the energy consumption of the first TCAM stages by shortening the word size. However, our evolution shows that to achieve noticeable advantage of selective row activation, we should not split the first TCAM stage to less than 4-bit. Having short block size in first stage, results in too many active rows on next TCAM stages and thus large energy inefficiency. We consider the average normalized switch activity on the TCAM energy when the first TCAM stage is split to different number of bits. Our evaluation shows than in 16-stage ReMAM, our design consumes 23%, 14% and 19% energy consumption when the first stage is 2-bit, 4-bit and 8-bit widths. Indeed, increasing the first TCAM bitline at first improves computation accuracy. However, for TCAM with larger bitline than 4-bit, the TCAM energy starts degrading. In this work, we set a lower bound on the size of bitline of the first stage TCAM to 4-bit for any partitioned TCAM.

A TCAM with a high number of rows improves the hit rate and the average time that FPU can be clock-gated. In the proposed ReMAM architecture, the energy consumption has been decreased by utilizing selective row activation and in-advance precharging techniques. Figure 6 compares the TCAM delay and search energy consumption for a single-stage and for the proposed multi-stage TCAM in different sizes for the Sobel application. The search energy consumption of a conventional TCAM is application independent, because all lines are activated at each search. On the other hand, in the proposed multi-stage TCAM the number of activated rows, and thus the energy consumption, depends on hit rate and application type. In ReMAM, the energy consumption is lower than ReAM since it just consumes maximum energy on the first TCAM stage and the rest of the stages and resistive memory have fewer active rows. Such energy consumption decreases further if we split TCAM into more stages.
TCAM delay consists of two terms, precharging and evaluation (search) latency. Precharging the ML is usually long portion of TCAM search delay. Although our design searches sequentially, in-advance precharging deletes the precharging term from all stages. Because when the k^{th} stage is in evaluating mode, the $k+1^{th}$ stage is in precharging state (based on the hit in $k-1^{th}$ stage). In summary the total search latency of TCAM can be written as follows:

$$Delay = T_{Pre} + N * T_{Eval}$$

Where T_{Pre} and T_{Eval} are precharging and evaluation delays respectively. Going from 8 to 15 stages increases the TCAM delay dramatically with only a small energy improvement. This happens because our design sets a lower bound on the first TCAM stage to 4-bit word size. At 64-row, TCAM splitting to 2-stage, 4-stage and 8-stage achieves respectively $1.8\times, 2.7\times$ and $5.1\times$ energy savings

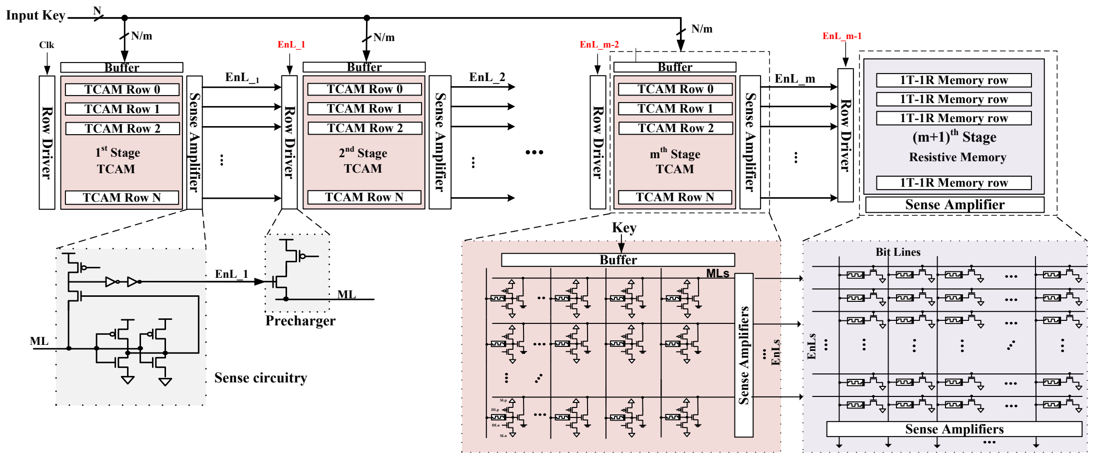


Fig. 4. Overview of the proposed ReMAM architecture.

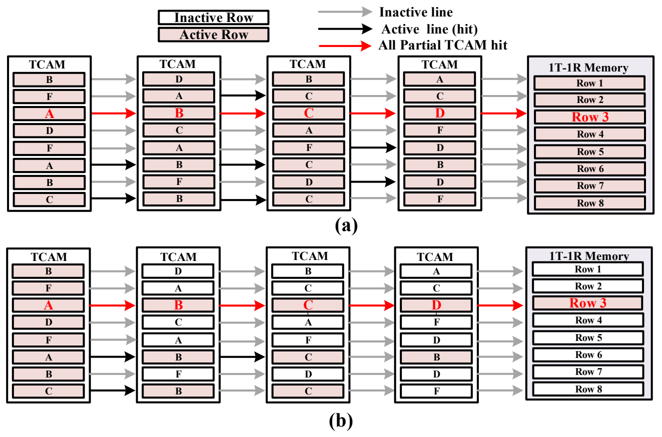


Fig. 5. Example of searching ABCD string on 4-stage ReMAM (a) without and (b) with selective row activation and in-advance precharging techniques.

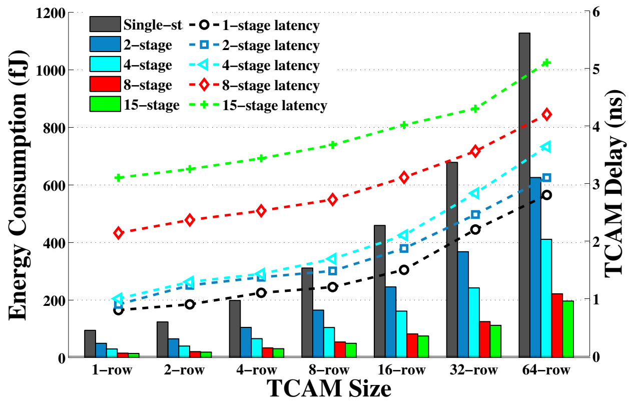


Fig. 6. Energy consumption of the proposed multi-stage and conventional single-stage TCAMs in different size.

compared to single-stage TCAM. The delay overhead less than 0.1ns, 0.3ns and 0.5ns.

Our evaluation also shows that the energy ratio of the ReMAM to single-stage TCAM increases for large TCAM sizes. This increases the hit rate by including a large number of undesired activations. Thus, the proposed ReMAM is well-suited for large associative memories. In addition, in ReMAM the TCAM and resistive memory rows are activated based on the hits in the previous stages. Therefore, as Figure 6 shows, a large ReMAM delay changes relatively small compared to conventional ReAM.

5 ReMAM Approximation

Although ReMAM reduces the search energy of associative memories, the hit rate of ReMAM has not changed as compared to conventional associative memories. The same size single stage associative memory has a similar hit rate as ReMAM. The minimum GPGPU energy point using ReMAM happens for small ReMAM size, e.g. 16-row, where the FPU is a dominant energy term. Therefore, to improve GPGPU energy consumption, we need to increases the ReMAM hit rate.

To address the low hit rate of associative memories, we proposed segmented and weighted approximation techniques. Segmented approximation relaxes the FPU computation of the selective bits in order to increases the associative memory hit rate. Bit relaxation happens by applying voltage overscaling (VOS) on selective TCAM blocks. The level of supply voltage determines the Hamming distance error that ReMAM accepts.

Weighted approximation considers the impact of each block on approximate mode. This technique addresses the search delay issue of ReMAM by reducing the number of serial stages. In weighted approximate ReMAM, the search operation on the LSB blocks performs in a single stage, since these stages do not have large switching activity. This accelerates the search operation by reducing the number of serial stages. In the following subsections, we explain the details of the proposed designs.

5.1 Segmented Approximation

When the workload changes, our TCAM must be able to efficiently change the TCAM configuration. Based on our previous work [6], we use ShortStop [42], [43] to apply voltage overscaling on partial TCAM blocks. Short-Stop transitions selected blocks into approximate mode in less than 5ns. Figure 7 shows the details of its implementation for multiple TCAM stages. This technique uses a dirty line to switch between high and low supply voltage

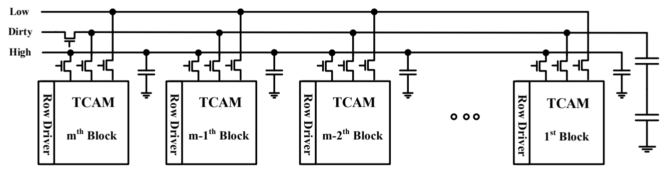


Fig. 7. ShortStop technique used to support voltage overscaling.

TABLE 1
Supply voltage (V) for putting different TCAM size in exact and approximate mode.

Configuration	2-bit	4-bit	8-bit	16-bit	32-bit
Exact	0.78V	0.80V	0.83V	0.85V	0.90V
1-HD	0.51V	0.54V	0.59V	0.63V	0.67V
2-HD	0.47V	0.49V	0.55V	0.58V	0.62V

at very small latency penalty. The functionality of voltage in ShortStop can be summarized in three steps. Before boosting, all blocks are connected to the high Vdd voltage. During boosting, a block which is going to be boosted disconnects from the low voltage line and connect to the dirty bitline. The dirty line uses boost capacitors to prevent disturbances on the supply voltages. Finally, the control signals disconnect the selected block from low dirty bitline and connect it to high Vdd. In each iteration multiple blocks can change from low to high Vdd or vice versa. In our evaluation, we account for the switching energy overhead of ShortStop on the total energy consumption.

Table 5.1 shows the level of supply voltage at which the TCAM can work in exact, 1-bit Hamming distance (1-HD) and 2-HD approximation modes. However, this voltage is different for TCAMs with different word-sizes. Our evaluation shows that a TCAM with a short word size can be approximated at significantly lower supply voltage compared to a large word size TCAM. For example, 4-bit TCAM achieves 1-HD and 2-HD approximation at 540mV and 490mV while these values are 630mV and 580mV for the 16-bit block. Using small size blocks can result in larger energy savings in partial blocks. However, since each block can separately add 1-2 mismatches to computation, the number of relaxed blocks should be limited. A TCAM with wide word size (e.g. 2-stage ReMAM) cannot reduce the energy consumption significantly due to (i) a higher supply voltage for VOS and (ii) a less tunable structure which does not put large portions of TCAM blocks into approximate mode.

To obtain significant energy reduction, we need to implement VOS on TCAM stages that have a low impact on computation accuracy. Error sensitive applications need a high percentage of exact matches to satisfy QoS, while other applications such as Robert and Sharpen image processing applications accept multiple mismatches on entire TCAM bitline. Take into account two key issues:

- Stored bits do not have the same impact on approximation. The most significant bits (MSBs) have a higher impact on the result of computation as compared to the least significant bits (LSBs). To control the computation error rate, we implement approximation on the TCAM blocks starting from the least significant bits. and put the LSB stages on approximation.
- Different configurations of ReMAM provide approximation at varying granularity. For example, ReMAM with 15-stages has the opportunity to put several partial blocks in approximate mode, while a 2-stage ReMAM can apply voltage relaxation only to two 8-bit blocks. Therefore, the ReMAM with 15 stages can achieve a higher energy efficiency than 2-stage, 4-stage and 8-stage ReMAM.

In order to achieve higher energy savings in approximate mode, proposed design starts the search operation from the least significant blocks so that the last search operation is on MSB blocks. Because a miss on the LSBs has lower impact on the arithmetic value compared to a miss on MSBs, we can relax a few of least significant blocks while delivering acceptable QoS. For each application the number of blocks in approximate mode is based on the ReMAM configuration. The number of blocks in approximate mode should not be too high, since in ReMAM each partial block adds 1 or 2 bits Hamming distance (HD).

5.2 Weighted Approximation

Although ReMAM improves GPGPU energy consumption, there are still two issues to resolve:

- ReMAM with many stages requires peripheral circuity (row driver and sense amplifier) for each partial stage which makes ReMAM inefficient in case of cost and area. In addition, in last ReMAM stages we do not require serial search, since ReMAM has active rows in the first stages.
- The search operation in ReMAM is slower than single stage associative memory. It slows down the FPU computation, since it needs to be searched in a single cycle of the FPU. Thus FPUs have to finish in the same clock cycle as ReMAM search.

To address these issues, our ReMAM uses multistage search TCAMs in the first ReMAM stages where partial TCAMs have many active rows. However, the rest of TCAMs have low number of active rows and can search only through a single stage, thus speeding up the overall process and enabling better energy savings. The number of serial stages in ReMAM is determined as a function of the acceptable ReMAM search delay for different configurations. To ensure ReMAM works at the same speed as FPU clock cycle, we merge the last stages of ReMAM into a single weighted TCAM stage. The number of stages is determined based on ReMAM configurations. For example, in 4-stage, 8-stage and 15-stage ReMAM our design merges last 2, 3 and 10 stages respectively to guarantee the ReMAM search performance.

Weighted TCAM is partitioned into m stages, (i.e. $B_1, B_2, , B_m$), where each block can accept different number of Hamming distances based on the position of bits on computation. In weighted TCAM each block is put on approximate mode $2\times$ less than the adjacent MSB blocks. For example, when B_i block accepts S-bit mismatches, the B_{i-1} and B_{i-2} blocks accept $S/2$, and $S/4$ bit Hamming distances (HD) respectively. In general, for two stages i and j, the Hamming distances accepted are defined as:

$$HD_i = HD_j/2^{i-j}$$

where, HDi is the maximum acceptable Hamming distance of block Bi.

TABLE 2
Discharging current and time of weighted TCAM during the search operation.

Single Mismatch	B_1	B_2	B_3	B_4
ML Discharging Current (mA)	6.5	3.2	1.7	0.9
Discharging time(ns)	0.3	0.4	0.5	0.6

TABLE 3
The ratio of running CNN on in different sizes, ensuring less than 2% quality loss.

Approximate Mode	**Voltage**	B_1	B_2	B_3	B_4
Exact	0.85V	Exact	Exact	Exact	Exact
Approx1	0.78V	Exact	Exact	Exact	1-HD
Approx2	0.68V	Exact	Exact	1-HD	2-HD
Approx3	0.78V	Exact	1-HD	2-HD	4-HD
Approx4	0.59V	1-HD	2-HD	4-HD	8-HD

We implement the idea of weighted TCAM using cells with larger access transistors in MSB bits. The access transistors size needs to be set based on the impact of that bit on accuracy. Our design works based on the timing characteristic of the ML discharging current. Using CAM with high small access transistor shows that any mismatches discharge TCAM ML very slowly. However, small R_{ON} or large access transistor of most significant blocks immediately discharge the ML. Table 2 shows the discharging current and times of ML when a single mismatch happens in each partial block (B_1, B_2, B_3, B_4). ML discharging current is not sensitive to mismatch on the first TCAM block. A mismatch of the cells with large access transistors (e.g. second, third and fourth) has a higher impact on ML discharge. Selecting a more aggressive approximation level (e.g. Approx4) improves the computational energy at the cost of lower accuracy, as expected. Table 2 shows that a mismatch in the MSB block, B_1, has ˜7× higher discharge current as compared to a mismatch in the LSB block, B_4. This large current increases the discharging speed, so that a mismatch on B1 discharges the ML in 0.3ns, while for less significant blocks it is slower. We exploit this timing differential during search to prioritize mismatches of different bit indices.

All partial TCAM blocks can do exact matching with sampling time of T_{Exact} (1.5ns for 32-row TCAM). If we instead enable ML voltage sampling at T_4 time, TCAM accepts a single bit Hamming distance in the B_4 block. Similarly, sampling at T_3 time is equivalent of accepting a single Hamming bit distance in the B_3 or similarly a 2-bit Hamming distance in the B_4. This weighted approximation encourages mismatches to be in the least significant blocks. Table 5.2 shows five configurations of 4-stage TCAM at different supply voltages. Instead of changing the sampling time, we can apply voltage overscaling to slow down search. As Table 5.2 shows, if we reduce the supply voltage of entire TCAM from 0.85V to 0.78V, the dropping voltage for 1-bit Hamming distance shifts to T_4 time. Therefore, TCAM matches rows with 1-bit Hamming distance in the first block. Further reducing the supply voltage shifts sampling time from T_3 to T_{Exact}, which means accepting 2-bit Hamming distance in the first block or a single bit Hamming distance in the second block.

Each application satisfies accuracy at potentially different approximation levels. Therefore, ReMAM should be able to change the level of approximation quickly and efficiently. In contrast to segmented approximation, which uses complex peripheral circuitry for approximation, weighted ReMAM reduces switching activity and requires small, fast and scalable peripheral circuitry for approximation. In our design a sense amplifier is aware of both number and place of mismatches in bitline. For example, when TCAM is in Approx2 mode, it can accept 1-HD on B_3 or 2-HD on the B4. However, having 1-bit Hamming distance on B4 and B3 is not acceptable, since it creates ML discharging current larger than Approx2 mode. This significantly improves the ReMAM energy efficiency, since i) it allows TCAM to work with deep VOS while minimizing the computation inaccuracy and ii) we can put more portion of TCAM bitline on approximate mode.

6 Experimental Results

6.1 Experimental Setup

We implemented the proposed ReMAM architecture on the AMD Southern Island GPU, Radeon HD 7970 device, which is one of the most recent GPU architectures. The benchmark applications have been adopted from AMD APP SDK v2.5 in OpenCL, to make it suitable for stream processing [44]. We run five popular OpenCL applications, to test the efficiency of ReMAM: Sobel, Robert, Sharpen, BlackScholes (Black), DwtHaar1D (DwtH), and MatrixMul (Matrix). The first three are image processing benchmarks, while the last two are general purpose applications. We use Multi2sim to simulate the described device [45]. This is a cycle accurate CPU-GPU simulator of which we modified the kernel code to enable profiling and runtime simulation. We extracted the most frequent patterns for adder (ADD), multiplier (MUL), multiplier-accumulator (MAC) and SQRT FPU computations. To obtain energy and delay, the 6-stage balanced FPUs are designed using Synopsys Design Compiler in 45-nm ASIC ow [46]. FPUs are optimized for power, based on measured delay of the TCAM in different sizes. We use memristor model in [47] for our memory design simulation with R_{ON} and R_{OFF} of $10k\Omega$ and $10M\Omega$ respectively.

In GPGPUs, the FPUs have a different number of input operands. The ADD and MUL accept two 32-bit, SQRT a 32-bit and MAD three 32-bit input operands. Therefore, their related TCAMs need to have 64-bit, 32-bit and 96-bit word sizes respectively. The circuit level simulation of TCAM design has been performed with HSPICE simulator for 45nm technology. For sizing, capacitors and resistors we used data from [33].

The execution flow of ReMAM has two main steps: design time profiling and runtime reuse. In profiling, we use an OpenCL kernel and host code to train the associative memory values based on an input dataset. We used 100 random images from Caltech 101 computer vision [48] as input for image processing applications (Sobel, Robert and Sharpen). For the remaining two applications, we test them on a sequence of input numbers with 100 different size. The training is done on 10% of the input dataset, randomly extracted. After that, the host code starts to save and rank the input patterns for each FPU based on their frequency of occurrence. In this state, the AMD compute abstraction layer provides a runtime device driver library and allows a host program to work with the stream cores at lowest level. The programming of TCAMs is done with software by using the host code. All

TCAMs associated with instances of the same kind of FPUs are programmed concurrently with the same data. To evaluate the computation accuracy of the proposed ReMAM in approximate mode, our framework compares the output file of each application with the golden output obtained from exact matching. We use 10% average relative error as an acceptable quality of service similar to other state of the art [49].

6.2 ReMAM and TCAM size

Figure 8 compares the normalized energy consumption of GPGPU using the proposed ReMAM and the conventional ReAM. The FPU energy is calculated based on the measured delay obtained for each TCAM size and the power consumption. The GPGPU energy is normalized to the FPU energy consumption at each point. The results in Figure 8 indicates that there is a tradeoff between TCAM and FPU energy consumption as a function of different TCAM sizes. There are a few reasons for this as explained below.

Figure 8 compares the normalized energy consumption of GPGPU using the proposed ReMAM and the conventional ReAM. The FPU energy is calculated based on the measured delay obtained for each TCAM size and the power consumption. The GPGPU energy is normalized to the FPU energy consumption at each point. The results in Figure 8 indicates that there is a tradeoff between TCAM and FPU energy consumption as a function of different TCAM sizes. Such tradeoff can be explained as follows:

FPU energy: large TCAMs have a higher hit rate. The hit rate improvement is not linear with the TCAM size. For example, going from 2-row to 4-row TCAM has more impact on the hit rate improvement than going from 64-row to 128-row. Figure 8 shows the impact of higher hit rate on the effective FPU energy. Higher hit rate increases the amount of time that FPU is in clock gated mode. However, longer delay of larger TCAMs increases the FPU clock cycle and reduces the energy advantage that we can achieve with FPU clock gating.

TCAM energy: a large size TCAM is a dominant contributor to the total energy consumption. As a result, GPGPU using conventional ReAM has a minimum energy point with 8-row TCAM. Decreasing TCAM energy is an effective way to improve the total GPGPU energy consumption. This not only affects the TCAM energy, but also allows the system to use larger TCAMs with a higher hit rate to decrease the effective FPU energy consumption. Average GPGPU energy savings with respect to FPU are 37.5% and 22.9% with ReMAM and ReAM respectively.

Figure 9 shows the normalized GPGPU energy consumption for TCAMs with different number of stages. Each line in the graph is normalized to the FPU energy using a single stage TCAM. At larger sizes, splitting the TCAM increases the number of undesired hits, and thus limits energy savings. At smaller sizes GPGPU energy is not sensitive to TCAM partitioning. Therefore, as it is shown in Figure 9, it is better to split the small size TCAMs into more stages. However, too much partitioning increases the number of undesired active rows and results in higher GPGPU energy consumption. Our evaluation indicates that in all applications GPGPU with 64-row ReMAM has a better energy improvement with respect to 4 and 16-row ReMAM.

6.3 ReMAM Approximation

6.3.1 Segmented ReMAM Approximation

For error sensitive applications using ReMAM with highly tunable granularity can result in higher energy savings since this configuration puts maximum number of non-critical blocks in approximate mode (see explanation in Section 5.1). For applications less sensitive to approximation, using ReMAM with coarse tuning granularity results in the best energy savings. Table 4 shows the impact of ReMAM on the overall GPGPU computation energy considering both floating point and integer units. The results show that for all tested applications, the FPUs are the main source of GPGPU energy consumption, where they consume about 92% of overall energy. As Table 4 shows, for exact matching (0-HD) the applications have the minimum energy point with 8-stage TCAM. For 1-HD and 2-HD approximation, the less sensitive applications, such as Robert, Sharpe and MatrixMult, prefer 2-stage and 4-stage ReMAM to achieve minimum energy point at coarse granularity.

For small TCAM sizes, our framework puts more partial blocks in approximate mode due to a few inexact matching in small TCAM. In contrast, in large TCAM size the number of inexact matching increases significantly which results low computational accuracy. The same tradeoff between energy can be observed in 2-HD approximation. Figure 10 shows the GPGPU energy savings and ReMAM hit rate improvements for different TCAM sizes. Our evaluation shows that, in most cases the GPGPU using 8-row or 16-row ReRAM can achieve the best minimum energy point. ReMAM hit rate improvement due to approximation is proportional to the number of TCAM rows. This fact is more observable on 2-HD ReMAM where the input can pre-store values with 2-HD. Finally, our results show that GPGPU using 4-stage, 16-row ReMAM can achieve minimum energy point of 48% and 51% in 1-HD and 2-HD approximation with an acceptable QoS. This large energy gain that ReMAM achieve in approximate mode is with the penalty of reducing GPU performance. Because, FPUs work with the same speed as their alongside TCAMs. This performance reduction is related to TCAM size and number of stages that determine the clock cycle of FPUs. In 16-row TCAM, using 4-stage, 8-stage and 15-stage increases the average computation speed over six running application by 0.5%, 3.5% and 6% respectively.

6.4 Weighted ReMAM Approximation

To address the performance issue of ReMAM, we merge the ReMAM stages to a single weighted TCAM stage. In 16-row TCAM, we set the size of last TCAM stage so that ReMAM works at the same clock frequency as FPUs. Table **??** lists the minimum GPGPU energy point and number of approximated blocks for 2, 4 and 8-block TCAM. Using 2-block TCAM gives coarse-grained approximation granularity and results in large energy savings for non-sensitive applications that can accept multiple mismatches on an entire TCAM bitline. Therefore, TCAM applies voltage relaxation on entire blocks. However, error sensitive applications do better with fine-grained approximation (4-block 8-block) with the capability of ensuring that mismatches occur on the least significant blocks. Our results show that GPGPU using 2, 4 and 8-block TCAM can achieve 55%, 58% and 54% energy savings on average, while delivering acceptable quality of service. Figure 11 shows the impact of approximation for the Sobel, Robert and Sharpen applications. The differences between the image processed with approximate and exact computation are visually negligible.

Table 6 compares the energy consumption of proposed ReMAM with the approximate associative memristive memory

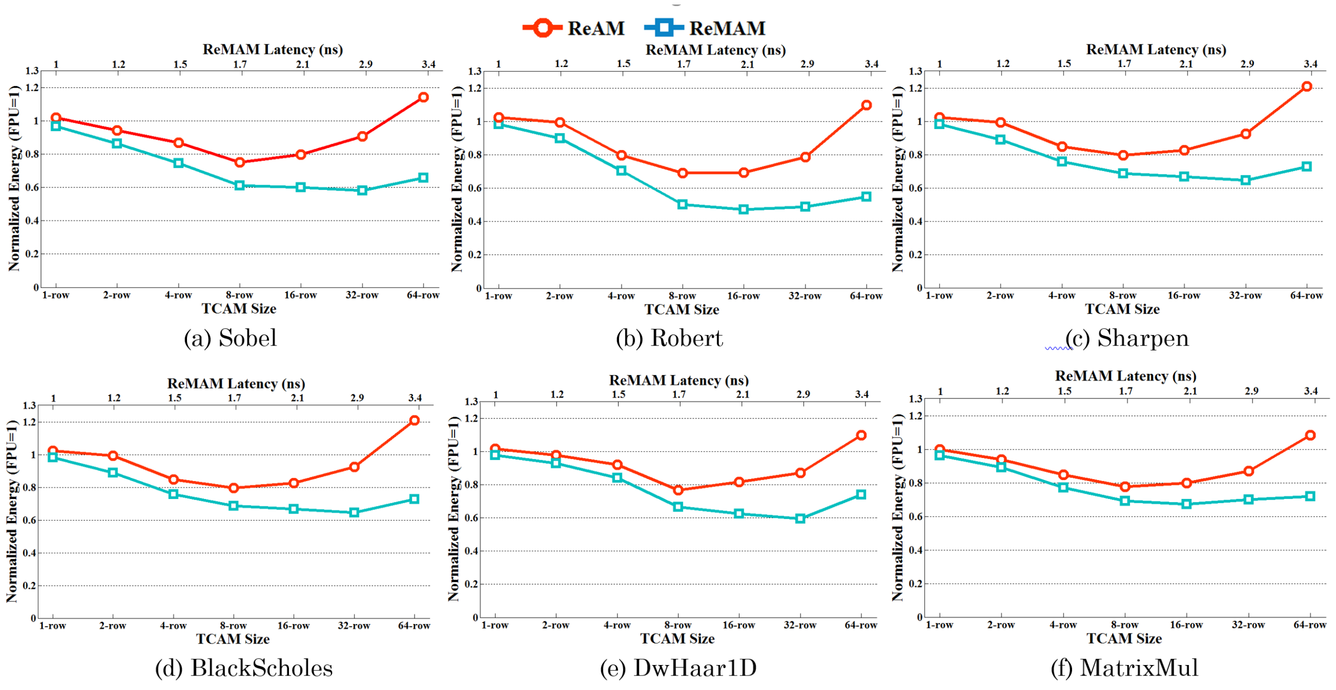


Fig. 8. Normalized FPUs energy consumption using ReMAM and conventional ReAM.

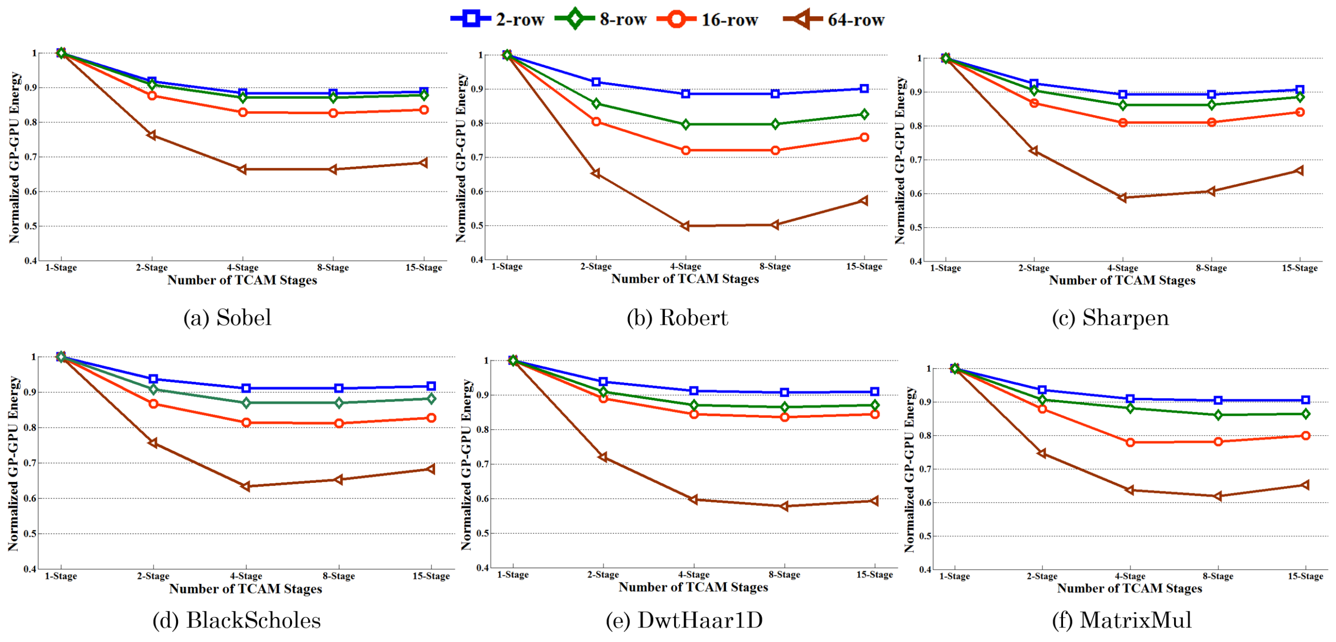


Fig. 9. Normalized FPUs energy consumption using ReMAM with different number of stages.

TABLE 4
GPGPU energy savings and performance reduction that ensures QoS in 16 row ReMAM.

Mode		Sobel	Robert	Sharpen	BlackScholes	DwtHaar1D	MatrixMul
0-HD	Energy Saving	34%	51%	32%	29%	26%	37%
	# of Stages	8	8	4	8	4	4
1-HD	Energy Saving	44%	63%	43%	39%	51%	45%
	# of Stages	8	4	4	8	8	4
2-HD	Energy Saving	46%	61%	45%	42%	53%	51%
	# of Stages	4	2	2	8	4	4

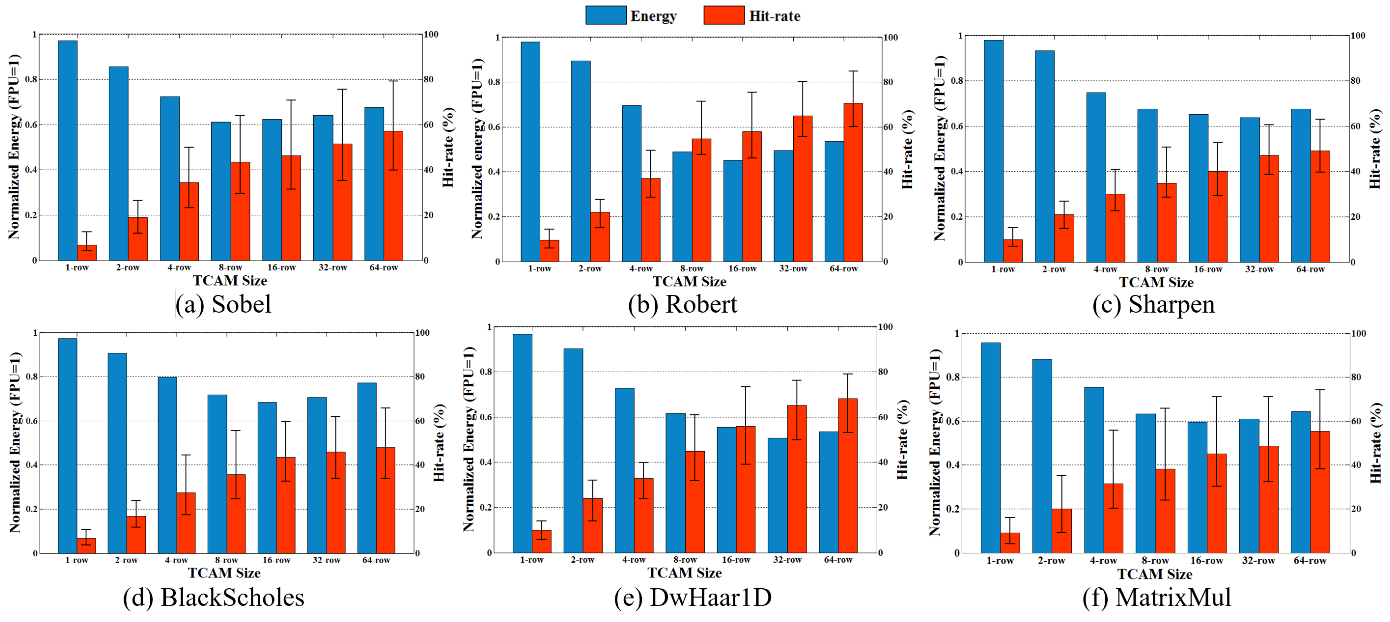


Fig. 10. The GPGPU energy savings and ReRAM hit improvements for 1-HD approximation.

TABLE 5
GPGPU energy saving and number of TCAM blocks in weighted approximation using different configurations.

Mode		Sobel	Robert	Sharpen	BlackScholes	DwtHaar1D	MatrixMul
2-block	Norm. Energy	56%	66%	70%	38%	62%	40%
	Block under VOS	2	2	2	1	2	2
4-block	Norm. Energy	53%	60%	67%	47%	67%	54%
	Block under VOS	2	3	3	3	2	2
8-block	Norm. Energy	41%	48%	56%	49%	69%	55%
	Block under VOS	4	5	5	4	3	3

TABLE 6
GPGPU Energy Saving using different associative memory design.

Different Designs	Mode	Sobel	Robert	Sharpen	Black Scholes	DwtHaar1D	MatrixMul
A^2M^2 [34]	Exact	25%	31%	20%	16%	23%	22%
	Approximate	30%	38%	24%	27%	34%	38%
ReCAM [6]	Exact	25%	31%	20%	16%	23%	22%
	Approximate	55%	53%	56%	36%	47%	41%
MASC [20]	Exact	28%	34%	27%	21%	25%	30%
	Approximate	43%	49%	47%	38%	49%	44%
ReMAM	Exact	34%	51%	32%	29%	26%	37%
	Approximate	53%	60%	67%	47%	68%	54%

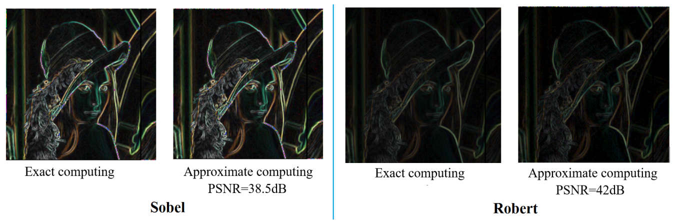


Fig. 11. Output quality comparison for Sobel, Robert, Sharpen applications.

(A^2M^2) [34], resistive configurable associative memory (ReCAM) [6], multi-stage single charge associative memory [20]. In the CAM level the ReMAM achieves 2.4× lower energy consumption as compare to the ReCAM and A^2M^2 design, which having an acceptable quality of service. Since the ReMAM not only exploits the voltage overscaling, but also uses selective row activation to significantly reduce the search energy. We also started the search operation from least significant bits where the blocks have large number of active rows. This provides an opportunity for implementing voltage overscaling on these stages with low impact on accuracy. In addition, in contrast to ReCAM, our design applies voltage overscaling in different block granularities on ReMAM. This opens an opportunity for having ReMAM with different block granularities, with low impact on approximation. Our evaluation shows that GPU+ReMAM energy consumption is 1.8× higher than GPU with ReCAM and A^2M^2 in exact matching mode. Our evaluation shows that in approximate mode ReMAM can achieve 1.3× and 1.8× energy saving in average compared to A^2M^2 and

TABLE 7
Energy-inaccuracy tradeoff of enhanced GPGPU over all applications.

QoS	**0%**	**2%**	**4%**	**6%**	**8%**	**10%**
Energy Saving	35%	38%	42%	46%	54%	58%

ReCAM using six different applications.

The main advantage of ReMAM compare to other associative memories (e.g. ReCAM and MASC) is its scalability in a term of bitline size. Most of previous designs work for processors with limited input operand word-size (¡128-bit operands), while the ReMAM address this limitation by having multistage pipeline search operation. Indeed, the application of ReMAM goes over the GPU, since it can be an appropriate design for DSPs such as text processing, search engine, image coding, etc.

To show the tradeoff between energy saving and accuracy, table 6.4 shows the average energy saving of enhanced GPGPU using ReMAM when we accept different QoL in applications computation. Having more accurate computation reduces the average energy that enhanced GPGPU can save. For instance, accepting 6% QoL provides larger energy saving, just 12% less than design with 10% QoL.

7 Conclusion

We propose a low-energy Resistive Multi-stage Associative Memory architecture, named ReMAM, which splits the TCAM search into a sequence of shorter stages. The proposed architecture employs selective row activation and in-advance precharging techniques to reduce the energy consumption and to mitigate the delay of sequential access. Our experimental results on AMD Southern Island GPU show that ReMAM decreases the system energy consumption of GPGPU more than 35% with error-free computation. We also show that ReMAM is particularly beneficial for systems with large size associative memories. Implementing weighted approximation on the proposed ReMAM further improves GPGPU energy savings by 58% on average with less than 10% average relative error as compared to GPGPU without associative memory.

Acknowledgment

This work was supported by NSF grant #1527034 and Jacobs School of Engineering UCSD Powell Fellowship.

References

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, "Big data: The next frontier for innovation, competition, and productivity," 2011.

[2] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context aware computing for the internet of things: A survey," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[3] B. D. Rouhani, E. M. Songhori, A. Mirhoseini, and F. Koushanfar, "Ssketch: An automated framework for streaming sketch-based analysis of big data on fpga," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, pp. 187–194, IEEE, 2015.

[4] T. Kohonen, *Associative memory: A system-theoretical approach*, vol. 17. Springer Science & Business Media, 2012.

[5] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (cam) circuits and architectures: A tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, 2006.

[6] M. Imani, A. Rahimi, and T. S. Rosing, "Resistive configurable associative memory for approximate computing," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1327–1332, IEEE, 2016.

[7] W. Eatherton, G. Varghese, and Z. Dittia, "Tree bitmap: hardware/software ip lookups with incremental updates," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 97–122, 2004.

[8] S. Li, L. Liu, P. Gu, C. Xu, and Y. Xie, "Nvsim-cam: a circuit-level simulator for emerging nonvolatile memory based content-addressable memory," in *Proceedings of the 35th International Conference on Computer-Aided Design*, p. 2, ACM, 2016.

[9] 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*, pp. 247–256, ACM, 2007.

[10] D. S. Vijayasarathi, M. Nourani, M. J. Akhbarizadeh, and P. T. Balsara, "Ripple-precharge tcam: a low-power solution for network search engines," in *2005 International Conference on Computer Design*, pp. 243–248, IEEE, 2005.

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

[12] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary, "Algorithms for advanced packet classification with ternary cams," in *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 193–204, ACM, 2005.

[13] A. Rahimi, L. Benini, and R. K. Gupta, "Spatial memoization: Concurrent instruction reuse to correct timing errors in simd architectures," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, pp. 847–851, 2013.

[14] J. Li, R. K. Montoye, M. Ishii, and L. Chang, "1 mb 0.41 μm^2 2t-2r cell nonvolatile tcam with two-bit encoding and clocked self-referenced sensing," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 4, pp. 896–907, 2014.

[15] S. Paul, S. Chatterjee, S. Mukhopadhyay, and S. Bhunia, "Nanoscale reconfigurable computing using non-volatile 2-d sttram array," in *Nanotechnology, 2009. IEEE-NANO 2009. 9th IEEE Conference on*, pp. 880–883, IEEE, 2009.

[16] 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)*, pp. 367–372, IEEE, 2016.

[17] H. Zhang, M. Putic, and J. Lach, "Low power gpgpu computation with imprecise hardware," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2014.

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

[19] 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)*, pp. 373–378, IEEE, 2016.

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

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

[22] X. Yin, A. Aziz, J. Nahas, S. Datta, S. Gupta, M. Niemier, and X. S. Hu, "Exploiting ferroelectric fets for low-power non-volatile logic-in-memory circuits," in *Proceedings of the 35th International Conference on Computer-Aided Design*, p. 121, ACM, 2016.

[23] N. Khoshavi, X. Chen, J. Wang, and R. F. DeMara, "Bit-upset vulnerability factor for edram last level cache immunity analysis," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, pp. 6–11, IEEE, 2016.

[24] 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*, vol. 290, pp. 1–5, 2016.

[25] S. Rajabi, M. Saremi, H. Barnaby, A. Edwards, M. Kozicki, M. Mitkova, D. Mahalanabis, Y. Gonzalez-Velo, and A. Mahmud, "Static impedance behavior of programmable metallization cells," *Solid-State Electronics*, vol. 106, pp. 27–33, 2015.

[26] 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*.

[27] 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*.

[28] Y. Kim, M. Imani, S. Patil, and T. S. Rosing, "Cause: critical application usage-aware memory system using non-volatile memory for mobile devices," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 690–696, IEEE Press, 2015.

[29] N. Khoshavi, X. Chen, J. Wang, and R. F. DeMara, "Read-tuned stt-ram and edram cache hierarchies for throughput and energy enhancement," *arXiv preprint arXiv:1607.08086*, 2016.

[30] M.-F. Chang, C.-C. Lin, A. Lee, C.-C. Kuo, G.-H. Yang, H.-J. Tsai, T.-F. Chen, S.-S. Sheu, P.-L. Tseng, H.-Y. Lee, *et al.*, "A 3t1r nonvolatile tcam using mlc reram with sub-1ns search time," in *2015 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 58, pp. 318–U449, 2015.

[31] S. Matsunaga, S. Miura, H. Honjou, K. Kinoshita, S. Ikeda, T. Endoh, H. Ohno, and T. Hanyu, "A 3.14 um 2 4t-2mtj-cell fully parallel tcam based on nonvolatile logic-in-memory architecture," in *2012 Symposium on VLSI Circuits (VLSIC)*, pp. 44–45, IEEE, 2012.

[32] T. Hanyu, D. Suzuki, N. Onizawa, S. Matsunaga, M. Natsui, and A. Mochizuki, "Spintronics-based nonvolatile logic-in-memory architecture towards an ultra-low-power and highly reliable vlsi computing paradigm," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1006–1011, IEEE, 2015.

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

[34] A. Rahimi, A. Ghofrani, K.-T. Cheng, L. Benini, and R. K. Gupta, "Approximate associative memristive memory for energy-efficient gpus," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1497–1502, IEEE, 2015.

[35] M. Imani, Y. Kim, A. Rahimi, and T. Rosing, "Acam: Approximate computing based on adaptive associative memory with online learning," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2016.

[36] R. Waser and M. Aono, "Nanoionics-based resistive switching memories," *Nature materials*, vol. 6, no. 11, pp. 833–840, 2007.

[37] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature nanotechnology*, vol. 3, no. 7, pp. 429–433, 2008.

[38] Y. Yang, P. Sheridan, and W. Lu, "Complementary resistive switching in tantalum oxide-based resistive memory devices," *Applied Physics Letters*, vol. 100, no. 20, p. 203112, 2012.

[39] L.-Y. Huang, M.-F. Chang, C.-H. Chuang, C.-C. Kuo, C.-F. Chen, G.-H. Yang, H.-J. Tsai, T.-F. Chen, S.-S. Sheu, K.-L. Su, *et al.*, "Reram-based 4t2r nonvolatile tcam with 7x nvm-stress reduction, and 4x improvement in speed-wordlength-capacity for normally-off instant-on filter-based search engines used in big-data processing," in *2014 Symposium on VLSI Circuits Digest of Technical Papers*, pp. 1–2, IEEE, 2014.

[40] M. Imani, D. Kong, A. Rahimi, T. Rosing, and J. Rabaey, "Exploring hyperdimensional associative memory," in *International Symposium on High-Performance Computer Architecture (HPCA)*, IEEE, 2017.

[41] 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*, vol. 41, pp. 189–200, ACM, 2013.

[42] C.-H. Hsu, Y. Zhang, M. A. Laurenzano, D. Meisner, T. Wenisch, J. Mars, L. Tang, and R. G. Dreslinski, "Adrenaline: Pinpointing and reining in tail queries with quick voltage boosting," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 271–282, IEEE, 2015.

[43] N. Pinckney, M. Fojtik, B. Giridhar, D. Sylvester, and D. Blaauw, "Shortstop: An on-chip fast supply boosting technique," in *2013 Symposium on VLSI Circuits*, pp. C290–C291, IEEE, 2013.

[44] "AMD APP SDK v2.5 [online]." http://www.amd.com/stream.

[45] 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*, pp. 335–344, ACM, 2012.

[46] "Synopsys design compiler,"

[47] S. Kvatinsky *et al.*, "Vteam: a general model for voltage-controlled memristors," *TCAS II*, vol. 62, no. 8, pp. 786–790, 2015.

[48] "Available at." http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[49] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 449–460, IEEE Computer Society, 2012.

Mohsen Imani received his M.S. and BCs degrees from the School of Electrical and Computer Engineering at the University of Tehran in March 2014 and September 2011 respectively. From September 2014, he is a Ph.D. student in the Department of Computer Science and Engineering at the University of California San Diego, CA, USA. He is a member of the System Energy Efficient Laboratory (SeeLab), where he is searching for alternative computer architecture to address memory bottleneck and computation cost. Mr. Imani is a Powell Fellow student at UC San Diego. His research interests include approximate computing, neuromorphic computing and memory centric computing.

Abbas Rahimi Abbas Rahimi is currently a Postdoctoral Scholar at the Department of Electrical Engineering and Computer Sciences, University of California Berkeley, Berkeley, CA, USA. He is a Member of the Berkeley Wireless Research Center and collaborating with the Berkeley Redwood Center for Theoretical Neuroscience. Rahimi has a BS in computer engineering from the University of Tehran, Tehran, Iran (2010) and an MS and a PhD in computer science and engineering from the University of California San Diego, La Jolla, CA, USA (2015). His research interests include brain-inspired computing, massively parallel memory-centric architectures, embedded systems and software with an emphasis on improving energy-efficiency and robustness in the presence of variability-induced errors and approximation opportunities. His doctoral dissertation has been selected to receive the 2015 Outstanding Dissertation Award in the area of new directions in embedded system design and embedded software from the European Design and Automation Association. He received the Best Paper Candidate at 50th IEEE/ACM Design Automation Conference.

Pietro Mercati Pietro Mercati received his M.S. and BCs degrees in Electronics Engineering from the University of Bologna, Italy in March 2013 and July 2010 respectively. From September 2013, he is a PhD student in the Department of Computer Science and Engineering at the University of California at San Diego, CA, USA, under the supervision of Professor Tajana Simunic Rosing. From April 2015 he is a PhD Candidate in Computer Science. He is a member of the System Energy Efficient Laboratory (SEELAB), University of California at San Diego. His current research interests include embedded systems, computer architecture, system energy efficiency and reliability. He is the author of several publications in peer-reviewed international journals and conferences.

Tajana Simunic Rosing is a Professor, a holder of the Fratamico Endowed Chair, and a director of System Energy Efficiency Lab at UCSD. She is currently heading the effort in SmartCities as a part of DARPA and industry funded TerraSwarm center. During 2009-2012 she led the energy efficient datacenters theme as a part of the MuSyC center. Her research interests are energy efficient computing, embedded and distributed systems. Prior to this she was a full time researcher at HP Labs while being leading research part-time at Stanford University. She finished her PhD in 2001 at Stanford University, concurrently with finishing her Masters in Engineering Management. Her PhD topic was Dynamic Management of Power Consumption. Prior to pursuing the PhD, she worked as a Senior Design Engineer at Altera Corporation.