This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2016.2642057, IEEE Transactions on Emerging Topics in Computing

Resistive CAM Acceleration for Tunable Approximate Computing

Mohsen Imani, *Student Member, IEEE,* Daniel Peroni, *Student Member, IEEE,* Abbas Rahimi, *Member, IEEE,* and Tajana Rosing, *Senior Member, IEEE*

Abstract—The Internet of Things significantly increases the amount of data generated, straining the processing capability of current computing systems. Approximate computing is a promising solution to accelerate computation by trading off energy and accuracy. In this paper, we propose a resistive content addressable memory (CAM) accelerator, called RCA, which exploits data locality to have an approximate memory-based computation. RCA stores high frequency patterns and performs computation inside CAM without using processing cores. During execution time, RCA searches an input operand among all prestored values on a CAM and returns the row with the nearest distance. To manage accuracy, we use a distance metric which considers the impact of each bit indices on computation accuracy. We evaluate an application of proposed RCA on CPU approximation, where RCA can be used as a stand-alone or as a hybrid computing unit besides CPU cores for tunable CPU approximation. We evaluate the architecture of the proposed RCA using HSPICE and multi2sim by testing our results on x86 CPU processor. Our evaluation shows that RCA can accelerate CPU computation by 12.6× and improve the energy efficiency by 6.6× as compared to a traditional CPU architecture, while providing acceptable quality of service.

Index Terms—Approximate computing, Content addressable memory, Associative memory, Non-volatile memory

1 Introduction

Internet of Things (IoT) increases the number of smart devices around the world. This number is expected to double by 2020 [1]. The rate of data generated by IoT will quickly overtake the capabilities of current computing systems. Internet of Things applications (e.g. machine learning, multimedia etc.) cause large energy and performance inefficiencies when run on general purpose processors [2]. The need for computing systems that can efficiently handle such large volumes of streaming data is undeniable [3], [4]. Approximate computing is an effective way of accelerating computation by trading between performance and accuracy. Many applications, such as machine learning, signal processing, speech recognition, search, and graphics, can accept some inaccuracy in computation [5], [6], [7], [8], [9] to gain both energy and performance advantages. The capability of adaptively tuning the level of accuracy is important for many applications.

Memory-based computing is an efficient way of data processing without using processor cores. Associative memory, in the form of a lookup table, stores commonly seen patterns and retrieves them at runtime. 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 [10]. In CMOS technology, TCAMs are designed with SRAM cells and consume a lot of energy for both store and search operations. Non-volatile memories, such as Resistive RAMs (ReRAM), magnetic RAMs (MRAMs), Ferroelectric RAMs (FeRAMs), and Phase change RAMs (PCRAMs) provide a solution for area and energy efficient memory and logic [11], [12], [13], [14]. NVM-based TCAMs can use voltage overscaling (VoS) to inexactly match the input patterns with stored values while controlling for hamming distance error as a function of voltage [15]. Associative memories are typically used next to processing cores for error-free execution, approximate computing, or as a stand-alone memory for exact associative computing [15], [16]. However, large data locality in several IoT applications, alongside the low search energy consumption of NVM-based TCAMs, motivates us to use associative memories directly as efficient and approximate computation units.

In contrast to previous work, which used associative memory primarily for computational reuse, we propose a Resistive CAM Accelerator (RCA) which uses associative memories directly as approximate computing units. For each application RCA exploits data locality by storing high frequency patterns in content addressable memories. In computing mode, RCA searches for input operands among all prestored values on a CAM and returns a row with the closest distance. We use proper distance metric which: i) significantly reduces the search energy consumption of RCA by sequential search and selectively activates the rows of the CAM stages, and ii) ensures computation accuracy by considering the impact of each bit indices on computation accuracy. We enable RCA's nearest distance search capability using the analog characteristics of the memristive devices to find a row with the maximum number of matched bits for input operands. RCA can be implemented as a stand-alone commuting unit or as a hybrid computing unit beside CPU computing cores. In hybrid mode, the workload can run partially on RCA and CPU cores using a threshold value which determines maximum acceptable distance of the input operand with the stored value in a CAM. Our experimental evaluation on x86 64-bit CPU architecture shows that using small size RCAs can provide 12.6× energy

- *M.Imani, D. Peroni 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 Berkley, Berkley, CA, 94720.*
  E-mail: abbas@eecs.berkeley.edu

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2016.2642057, IEEE Transactions on Emerging Topics in Computing

saving and 6.6× speedup, while also providing sufficient accuracy.

2 Background and Related Work

Associative memories are known as a promising solution to reduce the energy consumption of parallel computation. Associative memory consists of two main blocks: a TCAM and a memory. A set of high frequency patterns and their corresponding output are prestored on the TCAM and memory respectively. In runtime, input operands are compared with the stored values in CAM to enable computation reuse. In hardware, these memories are implemented with TCAM blocks. TCAMs based on CMOS technology are very energy consuming, limiting their application [17]. The low leakage power and high density of NVMs make them appropriate candidates to replace CMOS-based memory [18], [19], [20] and logic [21], [22]. Resistive-based TCAMs suffer from low endurance, limiting the number of write operations on CAMs to $10^5 - 10^7$ [23], [24]. Instead, MTJ-based TCAMs show significantly higher endurance (10^{15}), high temprature stability [25], [26], but slower search operation than resistive TCAMs [27]. In this work, we use a resistive TCAM cell in our design and address the low endurance issue by only updating it sporadically. Instead of doing energy hungry processing, we design Resistive CAM Accelerator (RCA) which learn to perform computations very efficiently using selective row activation for low switching activity of content addressable memories.

In the approximate computing domain, previous work focused on using approximate processing units or enabling approximation on existing computing units [5], [16], [28], [29], [30]. Putting entire computing unit on approximation degrades computation accuracy and requires a complex programing language [30], [31]. Architectures with both approximate and precise instructions running can achieve higher benefits from approximation. Reducing the number of running instruction on the processor pipeline, using a neural network, is an effective way of applying approximation [28], [29]. A neural processing unit (NPU) accelerates both CPU and GPU computations by approximately running applications on NPUs. However, running a large size neural network on CPU/GPU or even CMOS-based ASIC design is inefficient in terms of energy, performance and area.

Recently, NVM-based associative memories have been used for enabling approximation of floating point units in GPU architecture. The idea of voltage overscaling has been applied on TCAM to enable approximate search operation [15], [32]. This technique reduces TCAM search energy and relaxes processor computation by accepting 1-2 bits hamming distance between input and prestored CAM values. However, for several applications voltage relaxation on entire TCAM block increases the computational inaccuracy below the acceptable range. Imani et al. in [15] applied voltage overscaling on the selected CAM rows/bits to enable tunable GPU approximation.

In contrast to previous work, we propose RCA, which can be used as stand-alone computing unit or as a hybrid design to provide tunable CPU approximation by partially running workloads on CPU and RCA blocks. RCA not only improves computation energy, but also significantly accelerates computation by avoiding redundant computations.

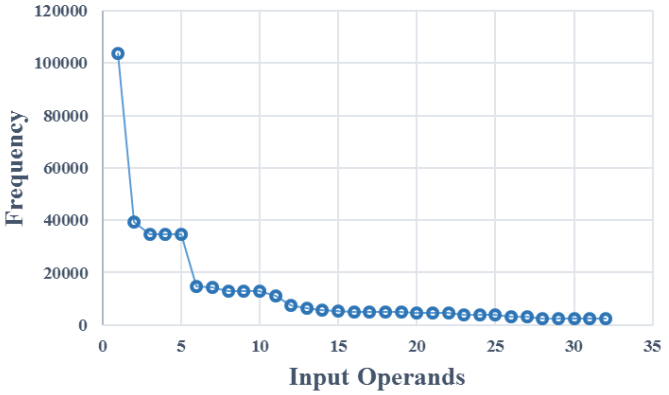


Fig. 1. The input data distribution of Black Scholes application running on CPU x86 core.

3 Proposed RCA Design

3.1 Motivation

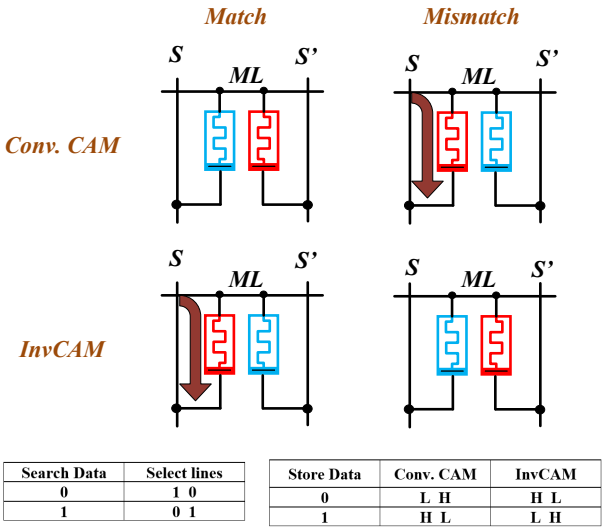
Approximate computing is one of the techniques to accelerate computation by trading energy and accuracy. We observed a large number of data similarities/locality in workloads. Figure 1 shows the data distribution of input operands when running Black Scholes application on x86 CPU. As graph shows, 15% of input operands have occurrence probability of more than 85%. This motivates us to design a block which can learn the computation by storing the high frequency patterns and retrieving them in runtime. This block should be a memory with the capability of computation. Although associative memories have this characteristic, they cannot be used directly as a computing unit. To enable computation capability, they need to return the nearest distance row at each search operation.

Resistive CAM accelerator can perform faster and more energy efficient computations compared to general purpose CPUs/GPUs. In order to have computation capability, RCA needs to have the capability of searching for closest distance row. However, conventional CAMs do not have the ability to search for the nearest row. They can only determine a row which exactly matches with the input pattern (if there is any). Implementing a fully digital CMOS-based design, which can search for nearest row, is very inefficient in term of power and area because it needs (i) bit level comparison of the input pattern and store values, (ii) to count the number of matches in each row, and (iii) finally finding a row with the minimum distance. To enable the nearest distance search capability, we design an Invert CAM (InvCAM) and then exploit analog characteristic of the NVM-based CAM to efficiently search for nearest data. It should be mentioned that, similar to a neural network, the RCA returns the best output based on the input's similarity to trained values. Therefore the RCA functionality for non-linear functions is not guaranteed.

3.2 InvCAM Cell

Figure 2 shows the structure of crossbar memristive CAM in normal and inverse (InvCAM) functionalities. In a conventional cell, the memristor devices and select lines are set such that the *ML* stays charged during the exact matching. In a match, there is no leakage current between the *ML* and ground, since the select

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2016.2642057, IEEE Transactions on Emerging Topics in Computing



Search Data	Select lines
0	1 0
1	0 1

Store Data	Conv. CAM	InvCAM
0	L H	H L
1	H L	L H

Fig. 2. Conventional and InvCAM cell in match and mismatch operations.

line which stored 0 is connected to high resistance (H). The *ML* current also cannot discharge from the cell with the select line of 1 because of same voltage across the memristor devices (even though the device is in low resistance mode). In the case of a mismatch in conventional cells, the select lines bias with inverse values, where the select line of a cell with low resistance (L) connects to the 0 and the high resistance to 1. Therefore, the *ML* discharges using the L resistance. We change functionality of CAM such that the *ML* counts the number of discharging cells in each row. InvCAM stores the opposite values on memristor devices. In case of a match, a cell with low resistance discharges the *ML*, while in case of mismatch the *ML* stays charged (as shown in Figure 2).

In InvCAM rows (consisting of several InvCAM cells), each matched cell adds a new discharging current component to the *ML*. So, during the search operation in InvCAM, all rows start discharging, except a row where all bits are mismatched with input operands. However, all InvCAM rows do not discharge at the same speed. In other word, in rows with more bit matches, the cells will discharge *ML* faster. We exploit this analog characteristic of memristor devices to design a CAM which has the capability of searching for the nearest distance row. For robustly detecting the closet row, InvCAM needs to have a limited number of cells in the bitline, because a *ML* discharging current/time does not change linearly with the number of matches in a line. For example, in a 16-bit InCAM, rows with 15 or 16 matches have very similar discharging characteristics. In other words, to distinguish a row with the fastest discharging time, we require an ultra-fast sense circuitry (˜ps delay). To address this issue, we limit the bitline size in each CAM to $<$8-bits and use memristive devices with large ON resistances for search operations. Short bitlines of InvCAMs help us to identify the difference between the number of mismatches with reasonable detector circuitry delay. Table 3.2 shows the *ML* discharging current for a 4-bit InvCAM with different numbers of matching bits and different InvCAM sizes. When all four cells match the input operand, the *ML* has the maximum discharging current, which means it has fastest discharging speed. Having a fewer number of matches results in slower *ML* discharging current. In

TABLE 1
ML hit/sampling time in 4-bit InvCAM having different number of mismatches/Hamming distances (HD) and different CAM sizes

InvCAM Mode	Number of Matches	128 rows	256 rows	512 rows	1024 rows	2048 rows
Exact	4	0.7ns	0.9ns	1.5ns	2.1ns	2.8ns
1-HD	3	0.9ns	1.1ns	1.8ns	2.4ns	3.4ns
2-HD	2	1.1ns	1.4ns	2.2ns	2.8ns	4.1ns
3-HD	1	1.4ns	1.7ns	2.6ns	3.3ns	4.6ns
4-HD	0	1.8ns	2.3ns	3.2ns	3.9ns	5.2ns

our design, rows with different number of matches have obvious discharging time distances. We exploit this characteristic to design a CAM with the capability of finding nearest hamming distance. This functionality is not easily implementable on conventional CAMs where *ML* stay charged in the case of a match. However, here we could provide this functionality by designing a circuity which can detect a row with the fastest discharging current.

3.3 InvCAM Architecture

To find an appropriate nearest neighbor row, we consider hamming distance between input data and stored values on a CAM. However, hamming distance metric is an aggressive metric for finding the nearest row, since hamming distance does not consider the impact of each bit indices in calculating the distance. In real computation, the most significant bits (MSBs) usually have higher impact on computation when compared to the least significant bits (LSBs). We exploit this feature to design a RCA with lower power consumption and better accuracy. Figure 3 shows the overview of proposed RCA. In our design, we split the R-row*N-bit CAM block to m small size stages (i.e. B_1, B_2, B_m) where each stage contains N/m-bits. In RCA, partial blocks search for input data in serial stages. The search operation starts from the block with the most significant bit. Each block has the capability of configuring as a CAM or memory block. In memory configuration, it uses sense amplifiers at the tail of vertical bitline to read the memory rows. In CAM mode, each stage can find the nearest hamming distance row(s) using the sense circuitry in the horizontal MLs. The sense amplifier finds the row with the fastest ML discharging current. The nearest row corresponds to data with the maximum matching bits with the input operand. After the sense amplifier, we have analog detector circuitry which can sense the number of active rows in each CAM. As soon as the detector block senses an active row, it stops the search on the current CAM stage and selectively activates the rows of the next stage CAM. During the next cycle, a similar search on the second stage starts, but only in the selected rows. This search operation continues until our design reaches the last stage with a row with nearest distance to input operand.

For applications with multiple input operands, the first RCA stage stores the first m-bits corresponding to all input operands and then uniformly searches for the stored data with the closest distance to the input key. All existing bits in the first RCA stage have similar weight in our nearest distance search. Based on the rows activated by this first stage, the search on the next RCA stages continues selectively.

Figure 4 shows peripheral circuitry which supports the nearest distance search. For each CAM stage, we use three layers of peripheral circuitry to support nearest distance search. First, the sense amplifier reads the value of match lines (MLs) to find a row which matches with the input data. Based on our cell design, a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2016.2642057, IEEE Transactions on Emerging Topics in Computing

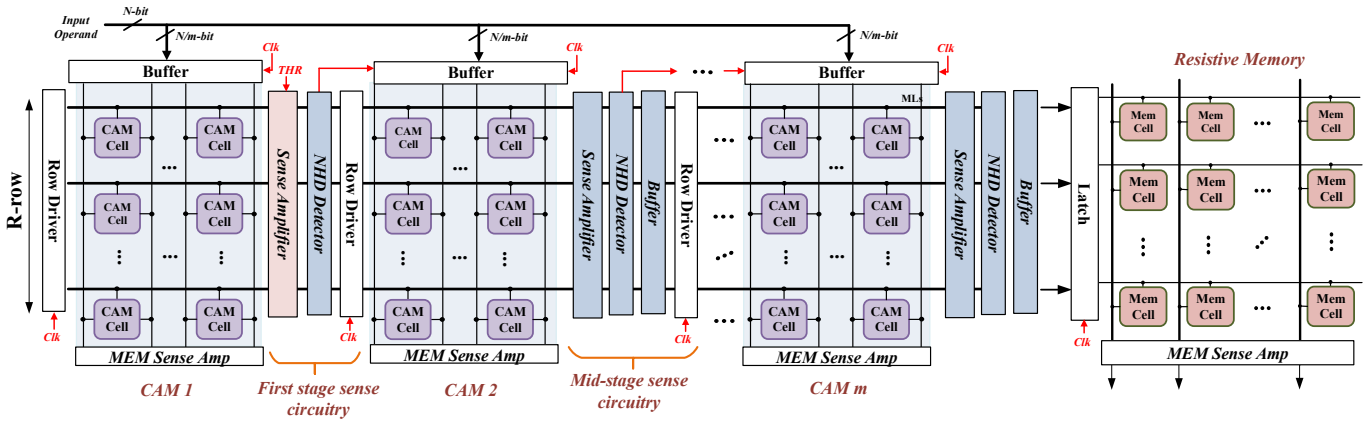


Fig. 3. The overview of RCA structure using InvCAM blocks.

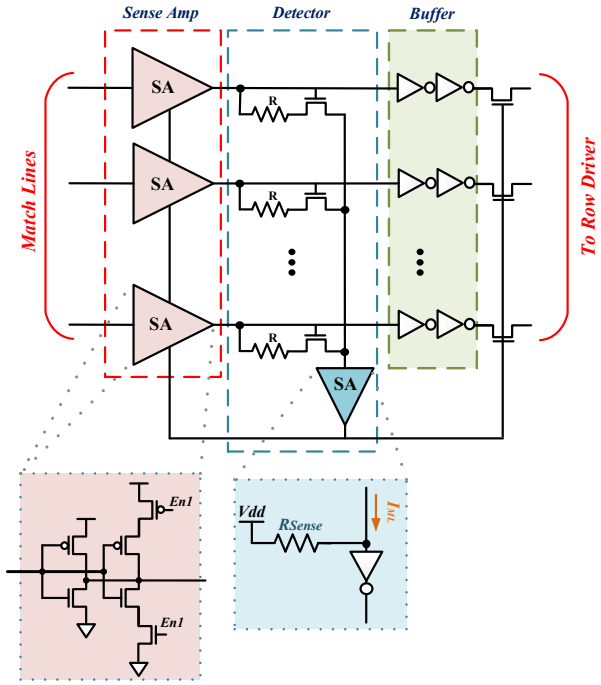


Fig. 4. Details of the sense amplifier, detector and buffer circuitries of RCA block.

row with more matched cells will start discharging the ML first. The matching results in a hit on the output of the sense amplifier. A detector circuitry, shown in Figure 4, checks for hits in CAM rows. In the case of any active rows, (i) it activates the access transistors to selectively activate the rows of the next CAM stage, and (ii) sends signals to the sense amplifier of the current CAM stage to stop the search operation from hitting additional rows. As our sense amplifier circuitry does not use a clock in its structure, so the delay of the buffer stage has an important rule on the correct functionality of our design. Because the detector circuitry is not an ideal circuit, and has a long delay, it may not be able to immediately sample the active rows. This delay can result in several matched rows and missing the rows which do not have minimum distance.

To better clarify the functionality of proposed multistage RCA, Figure 5 shows an example of search operation on a 4 stage, 8 row RCA. The search operation starts from the first stage, which is the most significant block, by searching for nearest Hamming distance row. The hit rows on the first stage selectively activate rows of the second RCA stage. The second TCAM searches for

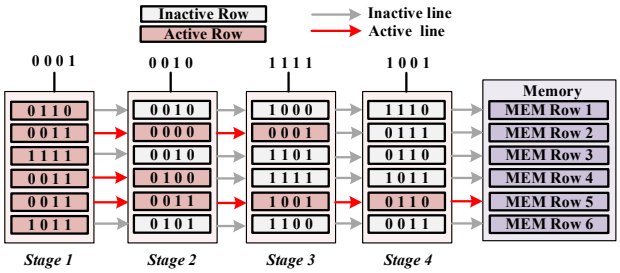


Fig. 5. Example to show RCA search functionality in 4 stage search.

the row nearest to 0010 on the three activated rows. The rows of the next TCAM stage activate serially, such that the design has a single active row in the final stage.

To guarantee the functionality of the proposed design, we add a buffer stage which has two main rules: **(i)** The block needs to delay the sense amplifier hits while the detector senses any active rows. In order to have stable detection, the delay of the buffer stage should be higher than the delay of the detector circuitry. To set the size of the buffer and detector circuitry, we consider 10% process variation on the transistors size and threshold voltage [33]. We designed our circuity for the worst-case scenario where there is only a single hit row on the CAM rows (maximum detector latency) and when CAM has a row with all matched cells and a row with a single mismatch. As we explained before, the ML discharging current starts saturating with an increase of the matched cells in each row. A row with all matched cells and a row with a single mismatch have fastest ML discharging speed. **(ii)** The second rule of buffer is to sample the activated rows (output of buffer stage) in case of a hit in detector circuitry.

The number of TCAM rows in the InvCAM structure depends on the precision of the detector circuitry. The detector circuitry should be able to identify a single activated row in CAM architecture. TCAMs with many rows suffer from large amounts of leakage currents through the sense circuitry output, resulting in a wrongly identified matching row by the detector. To have enough ratio between the leakage and matching currents, we need to have $I_{Matching} >> N * I_{Leakage}$, where the matching and leakage currents are the ON and OFF currents of sense amplifier output respectively. We use a diode connected transistor beside the detector resistance to control the OFF current ($I_{Leakage}$) of different rows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2016.2642057, IEEE Transactions on Emerging Topics in Computing

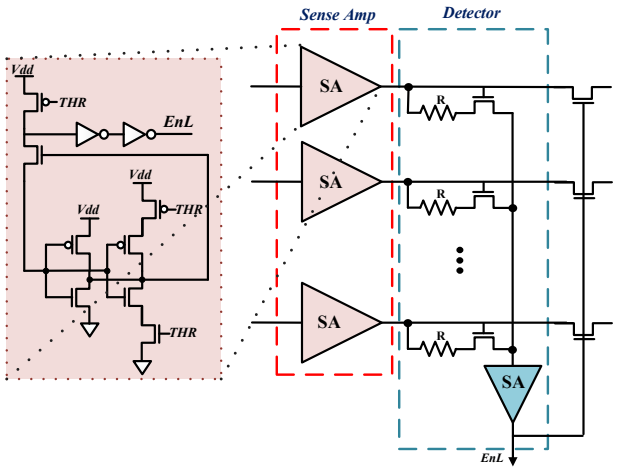


Fig. 6. The sense circuitry of the first stage InvCAM in RCA structure

3.4 Tunable RCA Approximation

RCA, as a stand-alone processing unit, can provide enough accuracy for several applications. As we explained, there are large amounts of redundant input operands in computation. For example, in images, a large portion of background pixels are usually similar. Processing these high frequency patterns provides enough computation accuracy even using small sized and fast RCA. However, uncommon/infrequent part of the workloads (with low data locality) cannot run accurately using a small RCA. To provide high computation accuracy for general workloads, we need to store a large set of data on the RCA, which results in high energy and performance overhead. To address this issue, we use both the existing processor and the RCA partially for each workload's computation. For the majority of data, which has close distance to stored values, our design uses RCA to perform computation fast and approximately, while the other part of input data with large distance to the stored values, can run on a precise core. Our goal is to have a hybrid computation which can set the level of accuracy by partially running the data on an exact processor and an RCA. When the input data is far from the stored values on RCA, the data is sent to CPU cores to process. Meanwhile, the RCA stops computation until the CPU cores process the assigned data. After getting the final result from the CPU, the RCA starts processing the next input.

Therefore, the RCA should have the ability of detecting the distance of input data to store values. As shown in Figure 6, the first CAM stage uses different sense circuitry, compared to the other stages, which can detect the ML discharging voltage corresponding to h-bit matching. Based on the h value, we set the sampling pulse of the sense circuitry (*THR*) to find the rows which have less than h-bit distance with pre-stored value. A detector circuitry checks the sense amplifier output of all rows and activates the row driver of next CAM stage accordingly. However, in case of missing data in detector circuitry, the EnL signal sends a signal to start running this data on exact processor instead. Based on desired accuracy, changing the clock time of the sense circuitry gives us different *THR* values. Using a late *THR* period, corresponds to deep approximation, while fast sampling means precise computation on the existing processor.

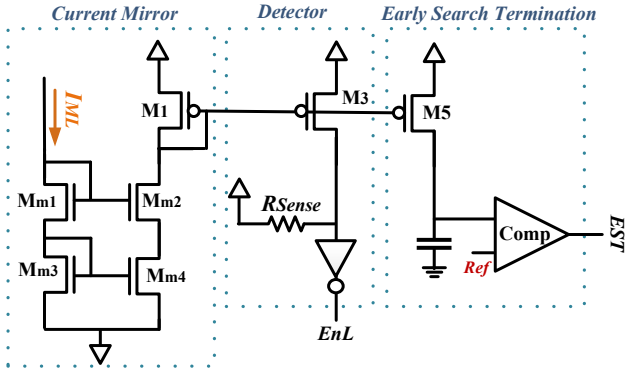


Fig. 7. Early search termination technique using analog comparator block

3.5 Early search termination

Our evaluation shows that in several cases we do not need to go through all CAM stages to find a nearest row. Instead, we can stop the search operation when the number of active rows in a block becomes one. Going further to InvCAM stages is unnecessary when we already found the row with the minimum distance, thus resulting in improved energy efficiency for the proposed design. We also observed that this condition occurs frequently. Even in a CAM with many rows, searching a few stages of CAMs is enough to find the closest row. Therefore, our design exploits Early Search Termination (EST) detector circuitry (shown in Figure 7) which can identify a case that single CAM row is activated. EST is designed using an analog comparator circuitry [34], which samples the same current that the detector is sampling and can identify the case that a single row of a CAM is matched. Obviously, using ETS in more stages can accelerate the search operation, however, it increases the energy and area of the CAM. Therefore, for all tested applications in this paper we use EST circuitry in one of the middle stages to check the number of row activations once and then stop further search operations. The best stage can be found based on the RCA configuration using profiling results from training mode.

In summary, our design considers the impact of each bit indices on the computation accuracy by searching for a closest input data starting from most significant blocks. In addition, serial search operation reduces the number of active rows in TCAM block, since the rows of each can be activated by the hit of the previous stage. This reduces the number of active rows stage by stage, until we achieve a single row at the last stage. The energy savings achieved by selective row activation depends on the InvCAM block size. InvCAM with smaller blocks reduces the overall number of active rows through CAM stages. However, this requires a larger number of sense amplifier and detector circuitries. The tradeoff between the size and energy consumption also impacts the computation performance.

4 RCA for CPU Approximation

RCA can have application in different processing platforms such as CPU, GPU, DSPs, or can be used as a stand-alone accelerator. In this work we consider the application of RCA for tunable CPU approximation. Figure 8 shows the overview of the CPU using RCA blocks beside each core. The proposed enhanced CPU has two types of execution units: **i)** fast and energy efficient RCA

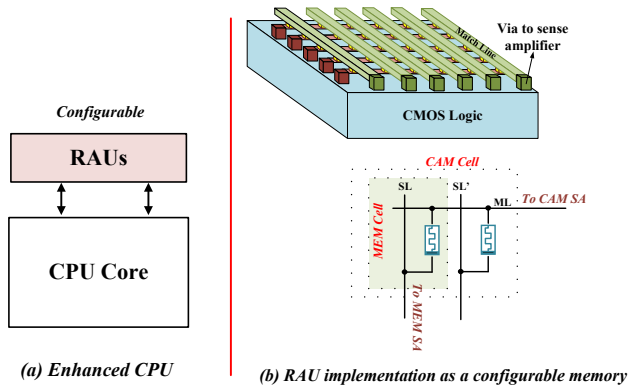


Fig. 8. Enhanced approximate CPU architecture and RCA implementation using crossbar memory

for approximate computing, and **ii)** conventional CMOS-based cores to process precise computations. Based on the running application and desired accuracy, the compiler can decide to run the application on RCAs or CPU cores or partially on both. Figure 8b shows how RCA using crossbar memories can be implemented at the top of the CMOS logics and configured as memory or CAM structure.

RCA is a configurable associative memory with the capability of learning. In training/profiling mode, RCA learns to perform specific memory-based computation and, as it trained, it can perform approximate computation continually. The RCA computation accuracy depends on training, nearest distance metric, and RCA blocks size. In our design based on training, we set the RCA size in order to provide less than 10% average relative error as verified by [28]. At runtime, instead of running applications on CPU, the RCA can perform the computation very fast, energy efficiently and approximately. There are two key advantages of the resistive hybrid CPU. First, RCA can process the search operation much faster than CPU computation. Second, energy and performance advantages are achieved by trading with accuracy. Accuracy is tuned by using a sufficient number of rows per TCAM block. Larger TCAM also has higher delay and energy consumption. RCA as a stand-alone processing unit can provide high energy savings for many CPU applications. However, it suffers from a lack of accuracy tuning capability at runtime.

In order to generalize our design, we need to have an architecture which can tune CPU approximation with fine-grained granularity. This motivates us to use both CPU and RCA partially for each workload computation. As we explained in section III.D, for the majority of data which has close distance to stored values, our design uses RCA to perform computation fast and approximately, while the other part of input data with large distance to the stored values, can run on a CPU core. To enable this functionality, the input data first searches the RCA and in the case of a hit within a threshold distance (*THR*) of RCA rows, it continues running that on RCA. Otherwise, this input data is sent to the CPU to process. This technique has several advantages: i) we do not need to have large RCA block to provide enough computation accuracy, because the RCA stores most frequency data with large data locality. ii) For each application we can tune computation accuracy by dynamically changing the *THR* value.

TABLE 2
CPU and RCA detailed parameters and tested benchmarks.

CPU Core		**RCA**	
CPU Architecture	x86 64-bit	Number of RCA	8
L1 Cache	32k I/32KB D	RCA block size	1-6 bits
L2 Cache	2MB	Output Register	8×32-bit
Benchmarks			
Name	**InvCAM input/output**	**Input Dataset**	
Black Scholes	6 inputs, 1 output	200000 stock variables	
FFT	1 input, 2 outputs	250000 random floating point values	
Sobel	9 inputs 1 output	512*512-pixel color images	
inversek2j	2 inputs 2 outputs	200000 random 2D coordinates	

5 Experimental Results

5.1 Experimental Setup

We evaluate the impact of proposed RCA on x86 64-bit CPU. The detailed parameters of the processor are shown in Table 2. We use Multi2sim, a cycle accurate CPU-GPU simulator for architectural simulation [35]. We use McPAT [36] to estimate the energy consumption of CPU. Four general applications, listed in Table 2, are used to show the efficiency of the proposed design. Each benchmark consists of small size building blocks with a few number of input/outputs. The circuit level simulation of TCAM design performs using HSPICE simulator on 45nm technology. We use the Vdd=0.85V for RCA block without accepting any computation error. To guarantee the impact of variation on circuit level design, we consider 10% process variation on the size and threshold voltage of transistors by running 10,000 Monte Carlo simulations.

5.2 RCA Framework

Our framework executes in two modes: training mode (design time) and execution (runtime). In training mode, we profile 10% of the input data for each application. Table 2 shows the dataset corresponding to each application. For non-image processing applications, the inputs use random stream data. During profiling, the system counts and ranks all operations based on the frequency of their occurrence for each application. Then, the host code chooses the top frequency patterns for each application to fill RCA. To evaluate the computation accuracy in approximate mode, our framework compares the output file of each application with the golden output for exact matching. First, approximation starts from the maximum level at each size, then it decreases until it can accept accuracy corresponding to 10% of average relative error.

5.3 RCA CAM and block size

The two major parameters that impact computation accuracy are number of CAM rows and block size. Figure 9 shows the energy improvement, speedup, and computation accuracy of a CPU using RCA with different CAM sizes and a 1-bit block size. Large RCA improves computation accuracy by storing more high frequency patterns in the CAM block. However, CAM with many rows requires a large input buffer to distribute input data among all rows simultaneously. This buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 9 shows, the computation speedup decreases in large CAM which is the result of the slow search delay of RCA at these large sizes. The result shows that using an RCA with 1024-rows, each application can achieve 6.5x energy improvement and 3.9x speedup, while

TABLE 3
RCA computation accuracy in different block sizes

Applications	BlackScholes	FFT	Sobel	Inversek2j
Number of rows	1024-row	256-row	512-row	1024-row
1-bit	3.4%	6.4%	9.3%	7.3%
2-bit	3.8%	6.6%	9.7%	8.3%
3-bit	4.2%	7.4%	10.0%	9.3%
4-bit	5.6%	9.2%	11.3%	10.4%
6-bit	7.2%	10.1%	12.5%	11.8%

TABLE 4
The best RCA stage to implement ETS in each configuration

	1-bit	2-bit	3-bit	4-bit	6-bit
Number of stages	32	16	11	8	6
ETS Stage	10	7	5	4	4

ensuring 10% quality of loss.

Block size is another important aspect to consider CAMs when ensuring computation accuracy. To accelerate computation, RCA prefers to use larger block sizes. Hamming distance metric does not consider the impact of each bit indices on the computation, so it is not an adequate metric to find the best and closet CAM row. In CAM with small blocks the hamming distance has less impact on computation accuracy compared to CAM with larger blocks. Table 5.3 shows the applications computation accuracy using different block sizes. CAMs with large block size accelerate RCA computation at the expense of lower computation accuracy. Because each block searches for nearest row based on hamming distance metric and the large CAM increases the probability of inaccurate matching.

In this paper, we used ETS on different RCA stage depending on RCA configuration. To find this RCA stage, in profiling we find the best RCA stage using different RCA block size which results in maximum energy saving. Table 4 shows the best number of RCA stage has been chosen to provide maximum average energy saving over all tested applications.

Figure 10 shows the energy improvement and speedup of different applications using RCA in different block sizes. Small blocks degrade the computation performance by increasing the number of serial stages. On the other hand, using short bitlines in RCA stages allows us to find more accuracy output when searching for nearest distance row. As a result, RCA can process most of input data. However, using very short block size reduces computation tune-ability by decreasing the number of bits in first RCA stage. From other side, RCA with large block size can better assign inputs between RCA and CPU, but requires higher switching activity and provides lower computation accuracy (due to Hamming distance metric). Our evaluation shows that using a middle size block (3-bit) is the best configuration to achieve maximum CPU+RCA energy saving. The 3-bit block CAM is able to provide 4.8× energy improvement and 8.3× speedup compare to CPU architecture, while providing acceptable quality of service.

5.4 Tunable CPU Approximation

Although RCA as stand-alone computing unit could accelerate the CPU computation, there is no way to tune the computation accuracy at runtime, because the CAM and block sizes are fixed. In order to have tunable CPU approximation, we use a design which can partially run workloads on CPU and RCA based on the distance of input data from the stored value in CAM. As we explained in Section IV, the threshold value (*THR*) determines the level of accuracy that the RCA will accept in the first block. In the case of an input value with higher distance than h-bit (can be tuned at runtime), our design runs the program on CPU. Otherwise the RCA will handle all computations. Table 5 shows the computation accuracy and percentage of time the CPU needs to run the computation in different application types. For each application, we use the CAM size obtained in section IV.D for our evaluation. Then, we adjust block size and if there is enough error in the first block it is counted as a miss and calculated on CPU. As we expected, using a larger block size results in more misses, which are run on CPU, increasing the computation accuracy. The improved accuracy comes at the expense of lower energy improvement and speedup.

Figure 11 shows the best energy improvement and speedup that hybrid CPU can achieve for different block sizes, when it satisfies the roughly 10% quality of service, while accepting deterministic error in the first block. For each application, the error is controlled by *THR* (h-bit) value resulting in the minimum energy-delay point for a given block and CAM size using a hybrid approach. Increasing the block size accelerates RCA computation by reducing the number of sequential search cycles. In addition, it allows us to use smaller CAMs for search operation resulting in faster searches. However, the computation speedup saturates in blocks larger than 4-bit, because our design requires a larger portion of the data to run on CPU cores rather than RCA. Using middle block sizes (˜3 bits) can provide maximum computation speedup in hybrid design. Inversek2j and FFT benefit greatly from the tunable system, as the error decreases sharply while still running a majority of the application on CAM.

Block size also affects energy reduction by trading between the portion of running workload on CPU and RCA.

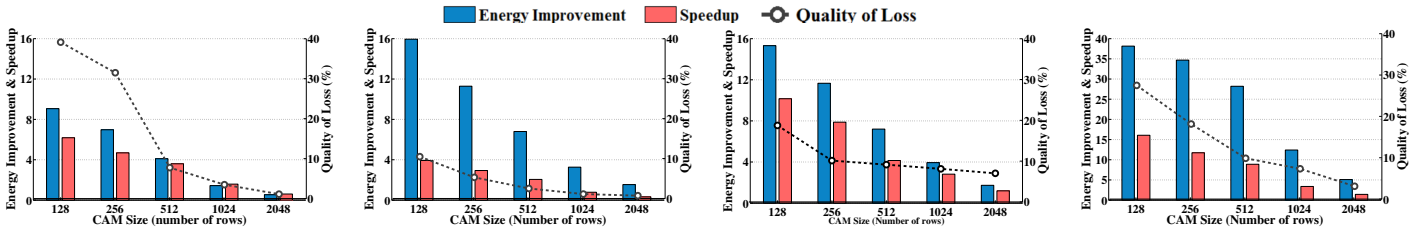
RCA is usually faster than CPU, but to provide enough accuracy, we need to increase the CAM size which degrades the energy and speedup advantages of RCA. Our experimental evaluation shows that using hybrid CPU can achieve 12.6× energy saving and 6.6x computation acceleration while ensuring computation accuracy. Table 6 compares the energy reduction and speedup of RCA design with the result of state-of-the-art NPU [28] which uses neural processing units for CPU approximation. The result shows that the fast and energy efficient RCA can achieve 3.2x higher energy reduction and 1.7× speedup compare to NPU in average while they ensure the required accuracy.

Figure 12 shows the visual results of Sobel application using the original computation (i.e., the golden image case) and approximate computation, resulting in no perceivable change. Our design can provide enough accuracy by selecting which portion of application runs on CPU or RCA.

6 Conclusion

In this paper we propose Resistive CAM Accelerator (RCA), which learns to accelerate computation approximately. RCA exploits the data locality by storing high frequency patterns inside content addressable memory. In computation mode, RCA finds a row which has the closest distance to input patterns while

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2016.2642057, IEEE Transactions on Emerging Topics in Computing



(a) Black Scholes (b) FFT (c) Sobel (d) inversek2j

Fig. 9. Energy improvement and speedup of approximate CPU.

TABLE 5
Hybrid ratio of approximate CPU for different block sizes

Block size	Blackscholes			FFT			Sobel			Inversek2j		
	QoL	*RCA Activation*	*RCA rows*	*QoL*	*RCA Activation*	*RCA rows*	*QoL*	*RCA Activation*	*RCA rows*	*QoL*	*RCA Activation*	*RCA rows*
1-bit	8.9%	100%	512	9.8%	97%	128	9.8%	97%	256	9.9%	100%	512
2-bit	9.8%	69%	256	9.4%	95%	128	9.6%	95%	256	9.4%	95%	512
3-bit	8.8%	67%	256	9.6%	95%	128	7.0%	80%	128	9.6%	92%	512
4-bit	2.5%	48%	256	9.4%	91%	128	4.8%	73%	128	9.6%	65%	256
6-bit	1.8%	43%	256	9.3%	89%	128	2.3%	64%	128	2.9%	25%	256

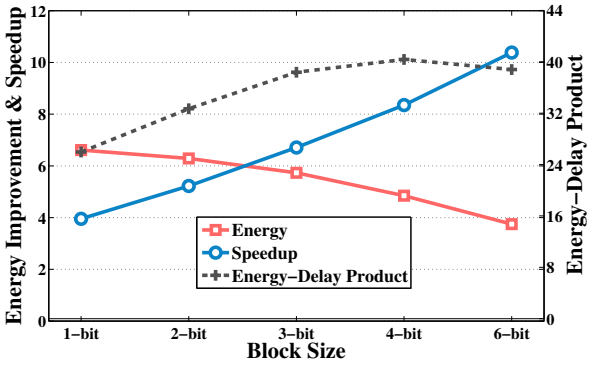


Fig. 10. Energy improvement speedup of approximate CPU compare to traditional CPU

TABLE 6
Energy and speedup comparison of RCA and NPU for CPU approximation

Applications Improvement		Black Scholes	FFT	Sobel	Inversek2j	Geomean
RCA	*Energy*	5.2×	13.5×	13.1×	27.7×	12.6×
	Performance	4.2×	4.4×	7.9×	13.2×	6.6×
NPU [28]	*Energy*	1.7×	3.1×	2.2×	21.1×	3.9×
	Performance	2.4×	3.6×	2.2×	11.1×	3.8×

considering the impact of each bit index on the computation accuracy. We show the application of RCA on CPU approximation where our framework partially runs workloads on precise CPU cores and RCA to achieve maximum energy-delay point. We evaluate the architecture using HSPICE and multi2sim simulator which shows that proposed RCA can accelerate CPU computation by 12.6× and improve the energy efficiency by 6.6x compare to CPU architecture while providing acceptable 10% quality loss.

ACKNOWLEDGMENT

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

REFERENCES

[1] J. Gantz and D. Reinsel, "Extracting value from chaos," *IDC iview*, vol. 1142, pp. 1–12, 2011.

[2] A. Mirhoseini, B. D. Rouhani, E. M. Songhori, and F. Koushanfar, "Perform-ml: performance optimized machine learning by platform and content aware customization," in *Proceedings of the 53rd Annual Design Automation Conference*, p. 20, ACM, 2016.

[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[4] 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.

[5] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman, "Enerj: Approximate data types for safe and general low-power computation," in *ACM SIGPLAN Notices*, vol. 46, pp. 164–174, ACM, 2011.

[6] 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.

[7] C. Alvarez, J. Corbal, and M. Valero, "Fuzzy memoization for floating-point multimedia applications," *IEEE Transactions on Computers*, vol. 54, no. 7, pp. 922–927, 2005.

[8] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, pp. 1–6, IEEE, 2013.

[9] M. Imani and U. Braga-Neto, "Optimal state estimation for boolean dynamical systems using a boolean kalman smoother," in *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 972–976, IEEE, 2015.

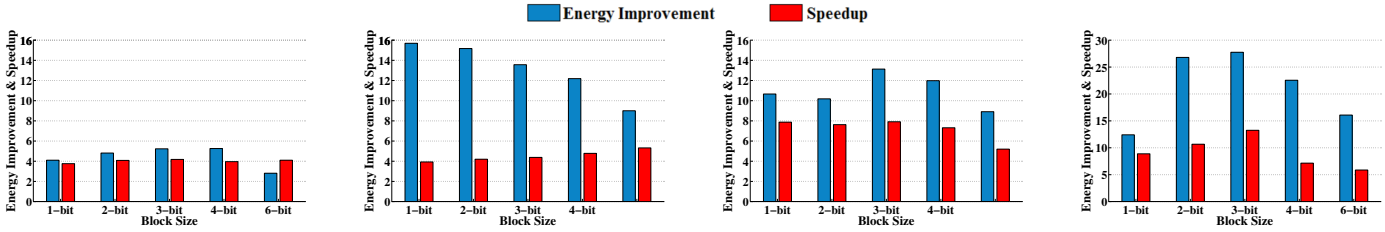
[10] M. Imani, S. Patil, and T. S. Rosing, "Masc: Ultra-low energy multiple-access single-charge tcam for approximate computing," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 373–378, IEEE, 2016.

[11] B. Pourshirazi and Z. Zhu, "Refree: A refresh-free hybrid dram/pcm main memory system," in *Parallel and Distributed Processing Symposium, 2016 IEEE International*, pp. 566–575, IEEE, 2016.

[12] 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*.

[13] 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.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2016.2642057, IEEE Transactions on Emerging Topics in Computing



(a) Black Scholes (b) FFT (c) Sobel (d) inversek2j

Fig. 11. Energy improvement and speedup of approximate CPU.

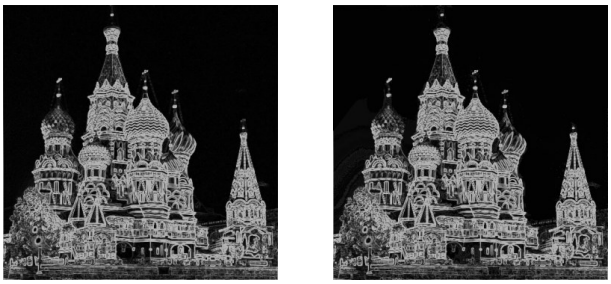


Fig. 12. Output quality for Sobel application

[14] M. Imani, A. Rahimi, Y. Kim, and T. Rosing, "A low-power hybrid magnetic cache architecture exploiting narrow-width values," in *2016 5th Non-Volatile Memory Systems and Applications Symposium (NVMSA)*, pp. 1–6, IEEE, 2016.

[15] 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.

[16] M. Imani, S. Patil, and T. Rosing, "Approximate computing using multiple-access single-charge associative memory," *Transactions on Emerging Topics in Computing*, 2016.

[17] J.-M. Arnau, J.-M. Parcerisa, and P. Xekalakis, "Eliminating redundant fragment shader executions on a mobile gpu via hardware memoization," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 529–540, IEEE, 2014.

[18] A. Goel and P. Gupta, "Small subset queries and bloom filters using ternary associative memories, with applications," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 1, pp. 143–154, 2010.

[19] 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.

[20] B. Yan, Z. Li, Y. Zhang, J. Yang, H. Li, W. Zhao, and P. C.-F. Chia, "A high-speed robust nvm-tcam design using body bias feedback," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pp. 69–74, ACM, 2015.

[21] S. Li and et al, "Pinatubo: a processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories," in *Proceedings of the 53rd Annual Design Automation Conference*, p. 173, ACM, 2016.

[22] 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.

[23] Y. Xie, "Modeling, architecture, and applications for emerging memory technologies," *IEEE Design & Test of Computers*, no. 1, pp. 44–51, 2011.

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

[25] 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*.

[26] 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*.

[27] N. Khoshavi and et al, "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.

[28] 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.

[29] A. Yazdanbakhsh, J. Park, H. Sharma, P. Lotfi-Kamran, and H. Esmaeilzadeh, "Neural acceleration for gpu throughput processors," in *Proceedings of the 48th International Symposium on Microarchitecture*, pp. 482–493, ACM, 2015.

[30] 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.

[31] L. Leem, H. Cho, J. Bau, Q. A. Jacobson, and S. Mitra, "Ersa: Error resilient system architecture for probabilistic applications," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, pp. 1560–1565, IEEE, 2010.

[32] 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.

[33] S. N. Mozaffari and A. Afzali-Kusha, "Statistical model for subthreshold current considering process variations," in *Quality Electronic Design (ASQED), 2010 2nd Asia Symposium on*, pp. 356–360, IEEE, 2010.

[34] C. Liu, B. Yan, C. Yang, L. Song, Z. Li, B. Liu, Y. Chen, H. Li, Q. Wu, and H. Jiang, "A spiking neuromorphic design with resistive crossbar," in *Proceedings of the 52nd Annual Design Automation Conference*, p. 14, ACM, 2015.

[35] 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.

[36] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 469–480, ACM, 2009.

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 Sci-ence 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 archi-tecture 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.

Daniel Peroni completed his B.S. in Computer Engineering at California Polytechnic State University in 2015. As of 2015, he is a Masters student in the Department of Computer Science and Engineering at the University of California San Diego. He is a member of the System Energy Efficient Laboratory (SeeLab), where he is investigating approximate computing using non-volatile memory solutions.

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.

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.