Non-volatile Content Addressable Memory for Computing Acceleration

Mohsen Imani, Daniel Peroni, Abbas Rahimi†, and Tajana Rosing
CSE, UC San Diego, La Jolla, CA 92093, USA
†EECS, UC Berkeley, Berkeley, CA 94720, USA
{moimani, dperoni, tajana}@ucsd.edu; abbas@eecs.berkeley.edu

I. INTRODUCTION

Associative memory, as a form of computing-with-memory, reduces energy of the processing elements by eliminating redundant computations [2], [8], [11]–[15], [17]–[29]. An associative memory can quickly recall responses of a function for a subset of input patterns to save energy by avoiding the actual function execution on the processing element. An associative memory is typically composed of a ternary content-addressable memory (TCAM) to store input patterns and an output memory to return the pre-stored output. The operation of a TCAM goes beyond retrieving logic “0” and “1” and it has capability to store and search wildcard [5]. This feature opens the application of the TCAMs for approximate computing domain, and a wide range of applications in query processing [2], [14], text processing [26], search engine [9], [25], image processing [1], [7], pattern recognition and classification [10], [21], [23].

Associative memories can be implemented on both software and hardware. Software solutions are based on hashing where a frequent data can be stored and retrieved from hash function by keys [3], [22]. In hardware, the associative memory is being implemented by TCAMs. Conventional TCAM with two SRAM cells suffers from high power dissipation and low density [7], [6]. The cost per bit of CMOS-based TCAM is 8× more than SRAM [4]. These limit the application of TCAMs to network applications [24]. Low leakage power of non-volatile memory make them appropriate as a replacement for CMOS-based TCAMs [29].

II. RESISTIVE CONTENT ADDRESSABLE MEMORY

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. This motivates us to design a block which can learn the computation by storing the high frequency patterns and retrieving them at 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.

In this paper we propose a Resistive Content Addressable Memory (CAM) accelerator, called RCA, which performs faster and more energy efficient computations compared to general purpose CPUs/GPUs. In order to have computation capability, RCA has the capability to search for the closest distance row. However, conventional CAMs do not have the ability to search for the nearest row, as 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 stored values, (ii) to count the number of matches in each row, and (iii) finally to find a row with the minimum distance. To enable the nearest distance search capability, we design an Inverse 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.

We propose an inverse CAM functionality (InvCAM), which enables the capability of searching for a nearest distance row. Figure 1 shows the structure of crossbar memristive CAM in normal and proposed inverse (InvCAM) functionalities. InvCAM stores the opposite values on memristor devices such that in case of a match, a cell with low resistance discharges the ML, while during the mismatch, the ML stays charged. 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 words, 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. To find an appropriate nearest neighbor row, we consider hamming distance between input data and stored values on a CAM. Figure 2 shows the overview of proposed RCA. In our design, we split the N-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. We use three layers of peripheral circuitry.

![Fig. 1. Conventional and InvCAM cell in match and mismatch operations.](image1)

![Fig. 2. The overview of RCA structure using InvCAM blocks.](image2)
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 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 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 role 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.

III. 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 4 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 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 4 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. 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. 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 a large RCA block to provide enough computation accuracy, because the RCA stores most frequency data with large data locality, and ii) for each application we can tune computation accuracy by dynamically changing the THR value.

IV. EXPERIMENTAL RESULTS

We evaluate the impact of proposed RCA on x86 64-bit CPU. We use Multi2sim, a cycle accurate CPU-GPU simulator for architectural simulation [27]. We use McPAT [25] 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.

The two major parameters that impact computation accuracy are number of CAM rows and block size. Figure 3 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 3 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 ensuring 10% quality of loss. 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 cores using a threshold value which determines maximum acceptable distance of the input operand with the stored value in a CAM. Our experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 ensuring 10% quality of loss.

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 experiments show that using an RCA with 1024-buffer is a dominant term of CAM search energy in large sizes. In addition, as Figure 3 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 improve...