ReMAM: Low Energy Resistive Multi-Stage Associative Memory for Energy Efficient Computing

Mohsen Imani, Pietro Mercati, Tajana Rosing
Computer Science and Engineering Department
University of California San Diego, La Jolla, CA 92093, USA
E-mail: {moimani, pimercat, tajana}@ucsd.edu

Abstract

The Internet of things (IoT) significantly increases the volume of computations and the number of running applications on processors, from mobiles to servers. Big data computation requires massive parallel processing and acceleration. In parallel processing, associative memories represent a promising solution to improve energy efficiency by eliminating redundant computations. However, the tradeoff between memory size and search energy consumption limits their applications. In this paper, we propose a novel low energy Resistive Multi-stage Associative Memory (ReMAM) architecture, which significantly reduces the search energy consumption by employing s*elective row activation* and *in-advance precharging* techniques. ReMAM splits the search in the 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 saving. 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. Our experimental evaluation on AMD Southern Island GPUs shows that ReMAM reduces energy consumption by 38.2% on average, which is 1.62X larger than using GPGPU with conventional single-stage associative memory.

Keywords

Associative memory, Ternary content addressable memory, Non-volatile memory, GPU

1. Introduction

The *Internet of Things* (IoT) increases the number of computations and running applications on processors [1, 2], from mobiles to servers. Big data computation demands for efficient massive parallel processing [1, 3]. However, parallel processing is extremely energy hungry. The idea of associative memory has been introduced to reduce the energy consumption by eliminating redundant computations [4-9]. Associative memory compares input data with a set of pre-stored data and searches for the corresponding result. If a matching is found, computation is clock-gated, enabling energy saving. Associative memories can be implemented in both software and hardware. Software solutions are based on a hashing where frequent data can be stored and retrieved from table using a set of keys [10]. In hardware, they are implemented with Ternary Content Addressable Memory (TCAM) blocks [5, 6].

Existing designs include both CMOS-based and non-volatile memory (NVM)-based TCAMs. CMOS-based TCAMs have a higher energy consumption, which limits their usage to network applications and classification [11]. In this scenario, NVMs offer an opportunity for building more energy efficient associative memories, as they have high density and low leakage power consumption [12-14]. Among NVMs, Resistive RAMs are a promising technology for associative memories [15, 16].

The use of "approximate" NVM-based TCAMs can help reducing energy consumption further, through the implementation of voltage overscaling (VOS) [17] [7]. While in CMOS-based TCAM the VOS significantly degrades the computation accuracy due to timing error and process variations[17, 18], in non-volatile memories the VOS trades search energy by accepting an arbitrary hamming distance between the input and the pre-stored patterns [7].

In every search cycle, all TCAM lines are precharged and discharged. This is the main reason of TCAM energy consumption. In this paper we propose Resistive Multi-stage Associative Memory (ReMAM), a new hardware 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 number of shorter searches and selectively activates rows based on the hit of the previous stages, thus reducing energy consumption. At the same time, *in-advance precharging* allows mitigating the delay of sequential TCAM access. This limits the number of precharges to only two steps for ReMAM with arbitrary number of stages. Our experimental evaluation on AMD Southern Island GPU architecture running five OpenCL applications shows that the proposed ReMAM architecture reduces the energy consumption of the GPGPU by 38.2% on average. Results also indicates that ReMAM achieves very high energy saving on systems with large associative memory. Such energy saving is 1.62X higher than the case of GPGPU using conventional resistive associative memory.

The remaining 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 shows the experimental setup and results. Finally, section 6 concludes this paper.

2. Related Work

Associative memories exploit the high number of pattern similarities of parallel processing to decrease the amount of computations [4-7, 19]. Associative memories are implemented using TCAM blocks. TCAMs in CMOS technology have low density and high energy consumption, which limits their application [11].

Recent research work uses NVMs as a replacement for CMOS-based TCAMs due to their low search energy, low leakage power and high density [12-14, 20]. Resistive RAM (ReRAM) and Spin-transfer Torque RAM (STT-RAM) are two kinds of high speed and reliable NVMs based respectively on memristive devices and magnetic tunneling junctions (MTJ) [21]. The endurance of the Resistive RAMs sets a limit to 10^6-10^7 write operations while for STT-RAMs this value is much higher (more than 10^{15}). Li, *et al.* designed a 1Mb energy efficient 2T-2R (2-Transistor/2-Memristor) TCAM, which is 10X smaller than SRAM-based TCAM [12]. Chang, *et al.* in [22] 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 [23]. This cell is for standby-power-free TCAM and has 86% area reduction with respect to SRAM-based design. Hanyu, *et al.* in [24] 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. For this reason, in this paper we adopt resistive technology rather than STT-RAMs. Also, to mitigate the effect of a lower endurance, our ReMAM design limits to only one write in kernel level.

Although associative memories are used in multiple contexts, in this section we review recent work targeting GPUs. In GPUs, memoization reduces the energy overhead of error recovery by exploiting data locality, as reported in reference [25]. In this work, 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. However, imprecise blocks like the SRAM-based LUTs suffer from high error-rate under VOS [17]. Similarly, an approximate associative memristive memory architecture is introduced in [7] to reduce the TCAM energy consumption by applying VOS. Rahimi, *et al.* in [19] used a memristor-based look-up table to increase the computational reuse in GPGPUs. The above publications show that approximation techniques may severely degrade the computation accuracy. In this work, instead, we show that ReMAM is robust when VOS is applied.

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

3. Resistive Associative Memory

Previous work introduced Resistive Associative Memory (*ReAM*), consisting of two main blocks, **TCAM** and **resistive 1T-1R memory**. A set of frequent input patterns and their corresponding results are pre-stored on TCAM and 1T-1R memory respectively. When a computation is issued, the operands are searched in parallel on the TCAM. If there is matching, 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 stored on cells based on the NVMs resistance state (Low or High). During search mode, the input operands are compared with all pre-stored TCAM patterns and if the data is found, a charged Match Line (ML) interrupts the processor computation by using clock-gating technique. In this paper we use 3T-1R cells as in [22] for TCAMs, which have been demonstrated to have advantages over previous cells [12, 26]. To reduce the effective load/capacitance of the ML, each cell is connected to it with a single junction, which results in a higher search speed and a lower energy consumption. Resistive 1T-1R memory consists of cells made by one memristor and one access transistor. Such cell is used to store the results of computations. The hit on the TCAM activates the corresponding line of the 1T-1R memory to retrieve the result of computation.

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 [9, 27]. However, having a TCAM with large size exacerbates some problems, described in the following.

(i) Word size: A 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 [8, 9]. One solution is to split the TCAM to 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. On the other hand, 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) Number of rows: A TCAM with a large number of rows requires big and power hungry input buffers to distribute the input signal to all rows. The delay introduced by big buffers prevents searching in the entire block on a single cycle and degrades energy efficiency.

(iii) Search energy: For each search operation, resistive associative memory requires a complete precharging among all TCAM rows. The search operation is energy hungry because of the high number of charges and discharges. The search operation on a large TCAM limits the energy efficiency of using associative memory and their application.

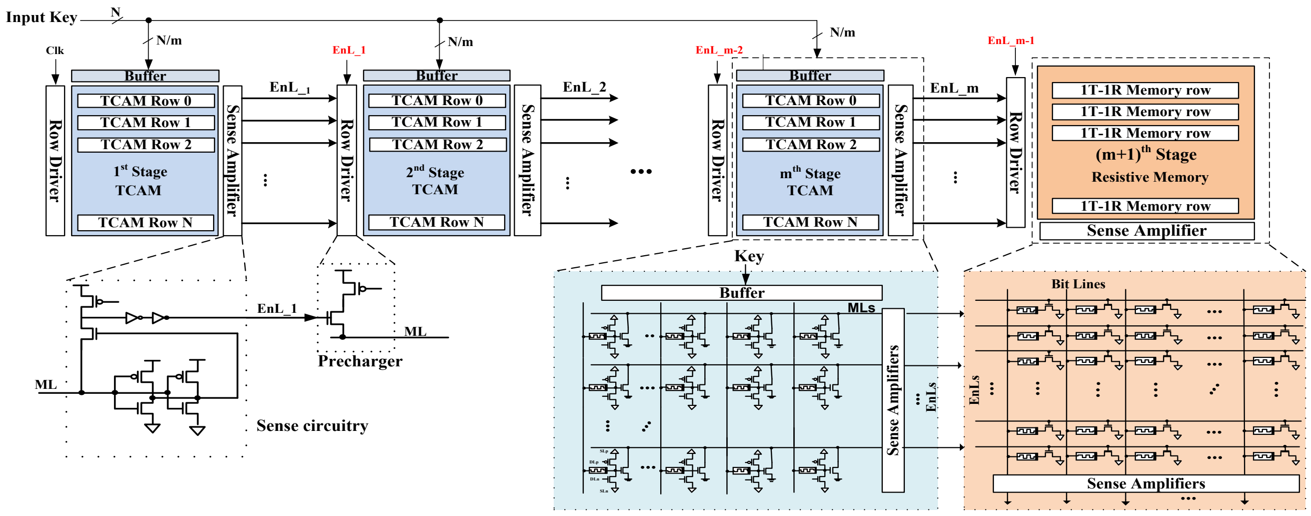

Figure 1. Proposed ReMAM structure.

Our ReMAM architecture addresses the first two problems by adopting a multi-stage structure with sequential access. This allows to have shorter rows in consecutive stages, which enables having a higher number of rows on each stage. Also, we address the third problem by introducing selective row activation technique. Finally, we mitigate the delay associated with sequential access by adopting in-advance precharging. The following section will explain the details of ReMAM architecture and proposed technique.

4. Proposed ReMAM

4.1. ReMAM architecture

This section describes the proposed ReMAM architecture and the details of the two techniques: *selective row activation* and *in-advance precharging.* Figure 1 shows the structure of ReMAM. It is composed by 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 performing a computation, the first N/m bits of each operand are searched on the first stage, where N is the size of the input. In case of a hit on some lines, the architecture employs selective row activation by sending *EnL_1* signal to the next stage to enable (precharge) only the corresponding lines. Then, the architecture searches the following N/m bits of the input data on 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 the 2^{nd} to the m^{th} stage. Reducing the number of precharges on active rows enables energy saving. After traversing all TCAM stages, if the pattern exists on the TCAM, one of the MLs in the m^{th} stage will stay in high voltage state (VDD) and activate the *Enl_m* signal. A hit in the m^{th} stage stops the computation by clock-gating technique. At the same time, this signal activates the corresponding row of 1T-1R memory in order to read the pre-stored result of computation.

Sequential row activation on the TCAM determines a large search delay, because each stage needs to wait for the following one to precharge before moving on. To considerably reduce such delay, we propose in-advance precharging technique. This technique precharges each row in a 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 hits ($2<k<m+1$). Note that a ReMAM with any number of stages still has some delay due to the precharge of the first two stages. Indeed, the delay reduction given by in-advance precharging takes place from the third stage. However, these two initial precharging steps can be done quickly on a short word size TCAM, with negligible impact.

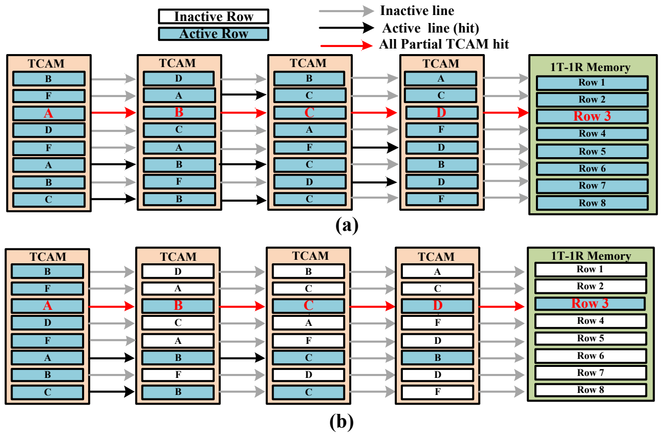

Figure 2. Example of searching “ABCD” string on 4-stage ReMAM (a) without and (b) with *selective row activation* and *in-advance precharging* technques.

In the following we show an example to clarify the advantages of the proposed design. Figure 2 shows a comparison of a 8-row, 4-stage ReMAM in which selective row activation and in-advance precharging are respectively disabled (Figure 2.a) and enabled (Figure 2.b). The goal is to search for “A B C D” string in TCAM. Each digit can be stored on 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. Based on our explanation, the row activation on the *3rd, 4th* and 1T-1R memory is done based on the hits of *1st, 2nd* and *3rd* 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.

4.2. ReMAM stages

The number of TCAM stages has a major impact on the associative memory energy consumption. Indeed, there is a tradeoff between the number of stages and the energy consumption of the ReMAM. Splitting the TCAM increases the number of hit rows 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, with a preliminary study we observed that in case of a 2-bit line in the first stage, the number of hits is high, which results in a large number of active rows in the following stages, thus a reduced opportunity for energy saving. To counteract this problem, we set a lower bound on the size of bitline on the first stage TCAM to 4-bit for any partitioned TCAM. Our evaluation revealed that having at least 4-bit lines in the first stage provides best results among other possible sizes, so we report only this case in the paper.

Note that the application of the proposed low energy associative memory is not limited to GPU processing. Thanks to multi-stage search operation, ReMAM can be used for search engines, searching and sorting, image coding, pattern recognition, query processing and several machine learning based processing, as well as previous work on associative memory [28, 29]. In most of these applications, the goal is to search a long key containing several digits on 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.

5. Experimental Results

5.1. Experimental setup and support framework

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 [30]. We run five popular OpenCL applications, to test the efficiency of ReMAM: *Sobel, Robert, Sharpen, BlackScholes* and *DwtHaar1D*. The first three are image processing benchmarks, while the last two are general purpose applications. This makes the evaluation more robust in the context of GPGPUs. We use Multi2sim to simulate the described device [31]. This is a cycle accurate CPU-GPU simulator of which we modified the kernel code to enable profiling and run-time simulation. We extracted the most frequent patterns for adder (ADD), multiplier (MUL), multiplier-accumulator (MAD) and SQRT FPU computations. To obtain energy and delay, the 6-stage balanced FPUs are designed using Synopsys Design Compiler in 45-nm ASIC flow [32]. FPUs are optimized for power, based on measured delay of the TCAM in different sizes.

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 on *HSPICE* simulator considering 45-nm technology. For sizing, capacitors and resistors we used the experimental details of reference [22].

We also present a framework which is compatible with OpenCL as a standard for parallel programming of heterogeneous systems. The execution flow of ReMAM has two main steps: design time profiling and run-time reuse. In profiling, we use an *OpenCL* kernel and a host code to train the associative memory values based on an input dataset. We used 100 random images from Caltech 101 computer vision [33] as input dataset 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, extracted randomly. After that, the host code starts to save and rank the input patterns for each FPUs based on their frequency of occurrence. In this state, the AMD compute abstraction layer provides a runtime device driver library and allows host program to work with the stream cores in lowest level. The programming of the TCAMs is done at the software level by using the host code. Note that all TCAMs associated with instances of the same kind of FPUs are programmed concurrently with the same data.

5.2. ReMAM and TCAM sizing

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 4 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 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. Going from 8 to 15 stages increases the TCAM delay severely 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 1.8X, 2.7X and 5.1X energy savings compared to single-stage TCAM. The delay overheads are respectively less than 0.1ns, 0.3ns and 0.5ns.

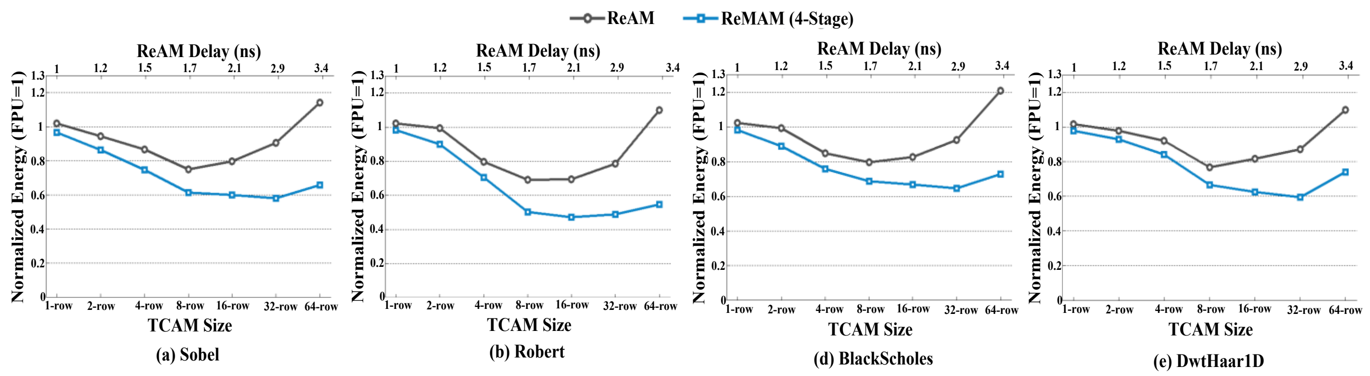

Figure 3. Normalized GPGPU energy consumption using ReMAM and conventional ReAM.

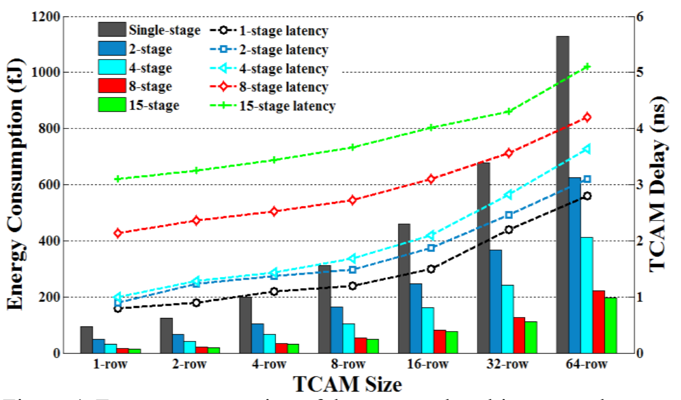

Figure 4. Energy consumption of the proposed multi-stage and conventional single-stage TCAMs in different size.

Our evaluation also shows that the energy ratio of the ReMAM to single-stage TCAM increases in large TCAM sizes. Indeed, this increases the hit rate by including a large number of undesired activations. This observation suggests that the proposed ReMAM is well-suited for implementation on large associative memories. In addition, in ReMAM the TCAM and resistive memory rows are activated based on the hit of the previous stages. Therefore, as Figure 4 shows, the ReMAM delay characteristic have a low difference with respect to conventional ReAM in large sizes.

Figure 3 compares the normalized energy consumptions of GPGPU using the proposed ReMAM and the conventional ReAM. The FPU energy is calculated based on the measured delay obtained with each TCAM size. The GPGPU energy is normalized to the FPU energy consumption in each point. The results in and Figure 3 indicate that there is a tradeoff between TCAM and FPU energy consumptions related to different TCAM size. Such tradeoff can be explained as follows.

(i) FPU energy: large TCAMs result in higher hit-rate. Moreover, 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. This shows the impact of higher hit-rate on the effective FPU energy. The reason is that a higher hit-rate increases the amount of time that FPU is in clock-gate mode. In addition, the FPU energy depends on the TCAM delay. The longer delay of larger TCAMs allows the Design Compiler to optimize the FPU energy consumption. Our results show that the FPU energy optimization continues until a TCAM with 1024-row (6.2ns TCAM delay).

(ii) TCAM energy: a large size TCAM is a dominant contributor to the total energy consumption. As a result, the 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. As Figure 3 shows, using efficient ReMAM results in minimum GPGPU energy point taking place with larger TCAM size (16 or 32 rows) compared to conventional ReAM. The GPGPU energy saving are 38.2% and 23.1% on average employing ReMAM and ReAM respectively with respect to FPU.

5.3. Energy and TCAM stages

Figure 5 shows the normalized GPGPU energy consumption in TCAMs with different number of stages. Each line in the graph is normalized to the FPU energy using single-stage TCAM. In large size, splitting the TCAM increases the number of undesired hits on stages, and limits the opportunity for energy saving. However with a smaller size, the GPGPU energy is not sensitive to the TCAM partitioning. Therefore, as it is shown in Figure 5, it is preferable to split small size TCAMs into a high number of stages. However, high partitioning of a large TCAM, increases the number of undesired active rows and results in GPGPU energy degradation. In addition our evaluation indicates that in all applications the GPGPU using 64-row ReMAM has higher energy improvement respect to 4–row and 16-row ReMAM. This fact indicates that ReMAM is highly desirable for systems with large TCAM sizes. The results show that the 64-row ReMAM decreases the energy consumption of GPGPU computation by 27.7%, 42.2% and 41.2% on 2-stage, 4-stage and 8-stage respectively, compared to utilizing conventional ReAM.

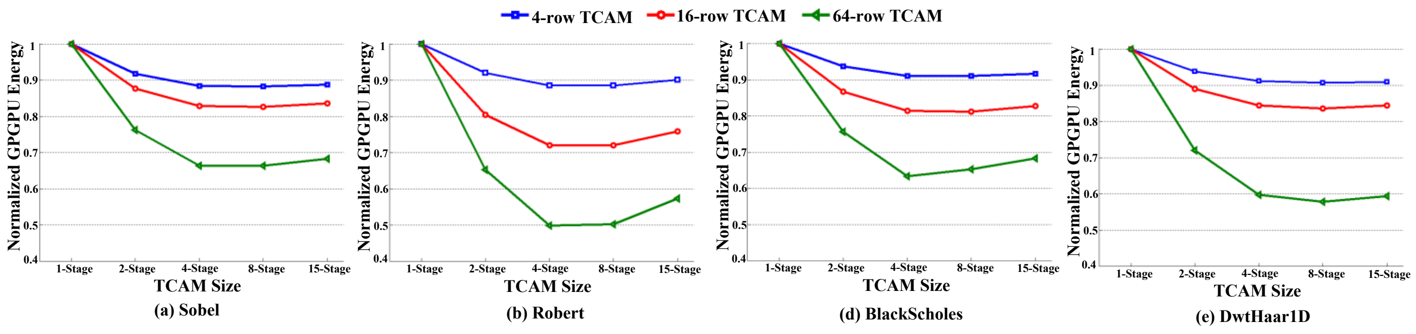

Figure 5. Normalized GPGPU energy consumption using ReMAM with different number of stages.

6. Conclusion

We proposed a novel low energy Resistive Multi-stage Associative Memory architecture named ReMAM, which splits the TCAM search to a sequence of shorter stages. The proposed architecture employs *selective row activation* and *in-advance precharging* techniques to reduce energy consumption and mitigate the delay of sequential access. The search operation in the proposed ReMAM can be done with very low energy consumption depending on the number of TCAM stage and application type. Our experimental results on AMD Southern Island GPU show that ReMAM decreases the system energy consumption of GPGPU more than 38.2% with error-free computation. Finally, we show that ReMAM is particularly beneficial for systems with large size associative memory.

Acknowledgement

This work was sponsored by NSF grant #1527034.

7. References

[1] C. Perera, et al., "Context aware computing for the internet of things: A survey," IEEE, Communications Surveys & Tutorials, vol. 16, pp. 414-454, 2014.

[2] J. Gubbi, et al., "Internet of Things (IoT): A vision, architectural elements, and future directions," Elsevier, Future Generation Computer Systems, vol. 29, pp. 1645-1660, 2013.

[3] J. Manyika, et al., "Big data: The next frontier for innovation, competition, and productivity," 2011.

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

[5] K. Pagiamtzis, et al., "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," IEEE ISSC, vol. 41, pp. 712-727, 2006.

[6] L. Chisvin, et al., "Content-addressable and associative memory: Alternatives to the ubiquitous RAM," IEEE Computer, pp. 51-64, 1989.

[7] A. Rahimi, et al., "Approximate associative memristive memory for energy-efficient GPUs," IEEE DATE, pp. 1497-1502, 2015.

[8] M. Imani, et al., "Resistive Configurable Associative Memory for Approximate Computing," IEEE DATE, 2016.

[9] M. Imani, et al., "MASC: Ultra-Low Energy Multiple-Access Single-Charge TCAM for Approximate Computing," IEEE DATE, 2016.

[10] W. Eatherton, et al., "Tree bitmap: hardware/software IP lookups with incremental updates," ACM SIGCOMM Computer Communication Review, vol. 34, pp. 97-122, 2004.

[11] K. Lakshminarayanan, et al., "Algorithms for advanced packet classification with ternary CAMs," ACM SIGCOMM Computer Communication Review, pp. 193-204, 2005.

[12] J. Li, et al., "1 Mb 0.41 μm² 2T-2R Cell Nonvolatile TCAM With Two-Bit Encoding and Clocked Self-Referenced Sensing," IEEE JSSC, vol. 49, pp. 896-907, 2014.

[13] S. Paul, et al., "Nanoscale reconfigurable computing using non-volatile 2-d sttram array," IEEE Nanotechnology, pp. 880-883, 2009.

[14] J. Cong, et al., "Energy-efficient computing using adaptive table lookup based on nonvolatile memories," IEEE ISLPED, , pp. 280-285, 2013.

[15] Y. Kim, et al., "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, 2015, pp. 690-696.

[16] S. N. Mozaffari, et al., "Fast march tests for defects in resistive memory," IEEE/ACM Nanoarch, pp. 88-93, 2015.

[17] H. Zhang, et al., "Low power gpgpu computation with imprecise hardware," IEEE DAC, pp. 1-6, 2014.

[18] M. Imani, et al., "Hierarchical design of robust and low data dependent FinFET based SRAM array," IEEE/ACM Nanoarch, pp. 63-68, 2015.

[19] A. Rahimi, et al., "Energy-efficient gpgpu architectures via collaborative compilation and memristive memory-based computing," IEEE DAC, pp. 1-6, 2014.

[20] B. Yan, et al., "A High-Speed Robust NVM-TCAM Design Using Body Bias Feedback," ACM GLSVLSI, pp. 69-74, 2015.

[21] Y. Xie, “Emerging Memory Technologies: Design, Architecture, and Applications,” Springer Science & Business Media, 2013.

[22] M.-F. Chang, et al., "A 3T1R Nonvolatile TCAM Using MLC ReRAM with Sub-1ns Search Time," IEEE ISSCC, 2015.

[23] S. Matsunaga, et al., "A 3.14 um 2 4T-2MTJ cell fully parallel TCAM based on nonvolatile logic-in-memory architecture," IEEE VLSIC, pp. 44-45, 2012.

[24] T. Hanyu, et al., "Spintronics-based nonvolatile logic-in-memory architecture towards an ultra-low-power and highly reliable VLSI computing paradigm," IEEE DATE, pp. 1006-1011, 2015.

[25] A. Rahimi, et al., "Temporal memoization for energy-efficient timing error recovery in gpgpus," IEEE DATE, 2014.

[26] L.Y. Huang, 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," IEEE VLSIC, pp. 1-2, 2014.

[27] Q. Guo, et al., "AC-DIMM: associative computing with STT-MRAM," ACM SIGARCH, pp. 189-200, 2013.

[28] N. Bandi, et al., "Fast data stream algorithms using associative memories," ACM SIGMOD, pp. 247-256, 2007.

[29] T. Kohonen, “Content-addressable memories,” Springer Science & Business Media, vol. 1, 2012.

[30] "AMD APP SDK v2.5" “http://www.amd.com/stream”.

[31] R. Ubal, et al., "Multi2Sim: a simulation framework for CPU-GPU computing," in ACM PACT, pp. 335-344, 2012.

[32] Design Compiler, "Synopsys Inc," ed, 2000.

[33] "http://www.vision.caltech.edu/Image_Datasets/Caltech1”