Resistive Configurable Associative Memory for Approximate Computing

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

Abbas Rahimi
EECS, UC Berkeley
Berkeley, CA 94720, USA
abbas@eecs.berkeley.edu

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

***Abstract*—Modern computing machines are increasingly characterized by large scale parallelism in hardware (such as GP-GPUs) and advent of large scale and innovative memory blocks. Parallelism enables expanded performance tradeoffs whereas memories enable reuse of computational work. To be effective, however, one needs to ensure energy efficiency with minimal reuse overheads. In this paper, we describe a resistive configurable associative memory (ReCAM) that enables selective approximation and asymmetric voltage overscaling to manage delivered efficiency. The ReCAM structure matches an input pattern with pre-stored ones by applying an approximate search on selected bit indices (bitline-configurable) or selective pre-stored patterns (row-configurable). To further reduce energy, we explore proper ReCAM sizing, various configurable search operations with low overhead voltage overscaling, and different ReCAM update policies. Experimental result on the AMD Southern Islands GPUs for eight applications shows bitline-configurable and row-configurable ReCAM achieve on average to 43.6% and 44.5% energy savings with an acceptable quality loss of 10%.**

I. Introduction

L-CSC emerged as one of the energy-efficient supercomputer in the world which is powered by GPU accelerators, namely the AMD FirePro, to surpass 5 GFlops/Watt [1]. Recent advances in massively parallel integrated architectures offer over thousands processing elements in GP-GPUs, hence enforcing energy efficiency as a primary concern [2]. Associative memory, as a form of *computing-with-memory*, reduces energy of the processing elements by eliminating redundant computations [5], [12], [16], [17], [21], [28], [3]. 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 [4]. This feature opens the application of the TCAMs for approximate computing domain, and a wide range of applications in query processing [5], text processing [6], search engine [8], image processing [7], pattern recognition and classification [9].

Although high power consumption of the CMOS-based TCAMs limits their usage [12], non-volatile memory (NVM) opens new opportunities for realizing energy-efficient associative memories [10] [11] [13]. NVMs such as spin-transfer torque random access memory (STT-RAM) and resistive RAM (ReRAM) consume lower energy consumption compared to SRAM. STT-RAMs and ReRAMs store value based on ON/OFF resistance of the magnetic tunelling junction (MTJ) and memristor devices. Li *et al*. [10] design a 1Mb 2-transistor, 2-resistive (2T-2R) TCAM which is 10× smaller than SRAM-based TCAM. Another TCAM with 3T-1R structure is able to search in 64-row TCAM array less than 1ns with 0.51fJ energy/bit/search operation [14]. Hanyu *et al*. [15] introduce 5T-4MTJ and 6T-2MTJ efficient TCAMs based on STT-RAM device. To further decrease TCAM search energy the idea of associative memory with accepting mismatches suitable for approximate computing has been introduced [16].

We propose a resistive configurable associative memory (ReCAM) suitable for approximate computing. ReCAM performs approximate search by accepting up to 2-bit mismatches on selective bit indices (bitline-configurable), or selective rows (row-configurable) of the TCAM at very low energy cost. ReCAM applies selective voltage overscaling (VOS) from the TCAM least significant bits with granularity of 8-bit and continues the depth of approximation based on application requirements for quality. ReCAM also applies asymmetric VOS on selective rows with lower frequency of hit events, hence increasing energy saving with lower quality of loss (QoL). ReCAM architecture is described in Section III. We explore design space of ReCAM by varying the TCAM size, configuration of the search operations, and update policies for the TCAM contents in Section IV. We observe that TCAMs sizes of 256-rows and higher can closely follow hit rates of TCAMs using update policy with an Oracle knowledge. Our experimental result, in Section V, on the AMD Southern Islands GPUs shows bitline-configurable and row-configurable ReCAM achieve on average to 43.6% and 44.5% energy savings with an acceptable QoL of 10%.

II. Related Work

Associative memory in the form of look-up table exploits patterns similarity to eliminate redundant computations. 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 [17], [18]. In hardware, the associative memory is being implemented by TCAMs [3], [5], [12], [16], [21]. Conventional TCAM with two SRAM cells suffers from high power dissipation and low density [19]. The cost per-bit of CMOS-based TCAM is 8× more than SRAM [21]. These limit the application of TCAMs to network applications and classification [22]. Low leakage power of STT-RAM and ReRAM cells make them appropriate as a replacement for CMOS-based TCAMs [23] [10]. Although MTJ-based TCAMs show higher endurance ($> 10^{15}$) than ReRAM ($10^6 - 10^7$), however ReRAM-based TCAMs have better search speed and area efficiency [25]. Our ReCAM addresses ReRAM endurance issue by limiting the write stress only at the start of kernel execution.

Zhang, *et al*. [27] use imprecise floating-point units (FPUs) for approximate computing in GPUs. However, CMOS-based imprecise blocks can suffer from massive number of errors under VOS [24] [27] [28]. The usage of memristor to increase computational reuse in GP-GPUs has been proposed in [3]. However this technique does not exploit applicable potential of energy saving through approximation. Rahimi *et al*. [16] propose approximate associative memristive memory based on VOS to decrease the search energy consumption of the associative memories. This technique accepts mismatch within hamming distance of 0–2 bits on entire TCAM bitline. How-

978-3-9815370-7-9/DATE16/

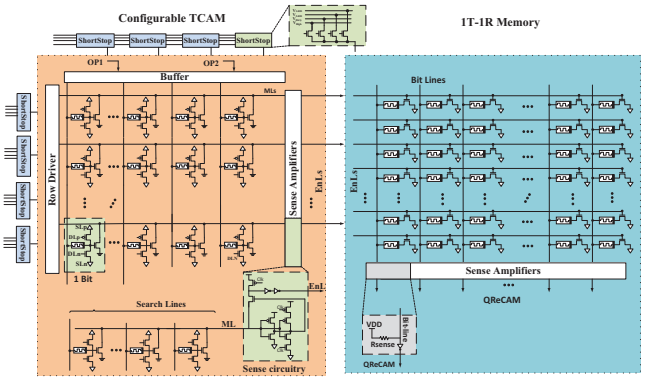


Fig. 1. ReCAM module architecture: 3T-1R TCAM plus 1T-1R memory.

ever, its scope is limited to image processing applications with tiny associative memory.

We design ReCAM as a configurable approximate associative memory module which applies asymmetric voltage relaxation on selective TCAM bit indices or rows. ReCAM balances the depth of approximation and QoL by starting voltage relaxation from least significant bits and lowest frequent TCAM rows at very low cost.

III. ReCAM Architecture

Our proposed ReCAM consists of a TCAM and a memory. For each processing element based on profiling, the frequent input patterns and their corresponding outputs are pre-stored on the TCAM and memory respectively. Fig. 1 shows our proposed ReCAM structure. During runtime, the input pattern are compared with TCAM pre-stored patterns in single cycle. TCAM works concurrently with the fist stage of the processing element. In a case of a hit event, the TCAM EnL signal halts the rest of stages of the processing element by the clock-gating technique. In this condition, the processing element consumes a small leakage power, and ReCAM returns the output value by reading it from the memory at a very low power cost instead of performing full computation through the rest of stages in the processing element.

3T-1R TCAM: NVM-based TCAM cells save the value based on the state of NVM resistance on each cell. The small NVM resistance ratio between ON and OFF modes degrades the cell stability by reducing the margin between match and mismatch currents. The small current ratio along with large match line (ML) capacitance increases both search energy and delay. To have an efficient and reliable search in TCAM, there should be a significant difference between match and mismatch currents. The worst case condition is when all cells are matched (no discharge) and when solely one of the cells is not matched ($\Delta V_{Mismatch} = \Delta V_{Match} - \Delta V_{1b-Miss}$). The best condition is to have high and distinct $\Delta V_{Miss} - \Delta V_{Match}$ value. Indeed, we expect to have a zero match current which means having a very high (infinite) ReRAM resistance ($\Delta V_{Match} \approx 0$). We use 3T-1R TCAM structure [14] which uses ReRAM with high resistance ratio and multiple-level cell capability. This cell has some advantages over the previous TCAM cells [10] [29]. In 3T-1R TCAM, cell is connected to ML with only one junction to decrease the effective load/capacitance of ML which results in higher search speed and lower energy consumption. In addition, in the cell structure the value of resistance has indirect effect on MLs ON/OFF current. This significantly improves the cell stability and makes the cell more robust against resistors variations. In addition, lower ML load (capacitance) of 3T-1R cell allows TCAM to have more cells per TCAM row.

1T-1R memory: As Fig. 1 shows, memory cell consists of a memristor and an access transistor. The data is saved on memristor based on its high or low resistance. In a case of a TCAM hit, the EnL signal activates the corresponding row of 1T-1R memory to select the result of the computation for the output.

To decrease the energy consumption of ReCAM on approximate mode, we apply VOS on 1T-1R memory and partially on TCAM. The illustrated robustness of TCAM allows us to decrease the nominal supply voltage from 1V to 0.85V for exact matching with acceptable stability and 1.5ns delay at 64-row, 96-bits TCAM (the worst-case TCAM delay with 64-rows). This supply voltage reduction improves the power efficiency of ReCAM more than 1.9× respect to a baseline with supply voltage of 1V. In approximate search mode, TCAM VOS to 0.73V/0.67V matches the input pattern with any of the stored pattern on TCAM with one/two Hamming distance (i.e., 1-HD/2-HD). We limit the scope of applying VOS to designated parts of ReCAM since all bit indices and TCAM rows do not have the same impact on the result of computation. Hence, our proposed ReCAM implements VOS on the selective TCAM bitline and rows to decreases the energy consumption with an acceptable QoL.

A. ReCAM with Bitline Configurable

In approximate TCAM, any mismatches in the most significant bits (MSBs) has high impact on QoL while computation is more robust to mismatches on least significant bits. In several applications, implementing 1-HD and 2-HD approximation on entire word size of associative memory causes large QoL at the output beyond the acceptable range. This fact motivates us to implement VOS on the selective bit indices to achieve better energy efficiency with acceptable QoL. We design a configurable approximate TCAM which is able to adaptively relax different TCAM bitlines based on application requirement. We consider the approximation on different TCAM bitline segmentation with 8-bit granularity. To design a configurable approximate TCAM, we need to have a mechanism that can implement relaxation on selective TCAM bit indices. The change in the relaxed bit indices should be done adaptively based on application type. For this purpose, we use an efficient Shortstop technique [31]. Shortstop is a fast boosting supply voltage technique with very low supply drop. This techniques uses a boost capacitor and a dirty supply rail to rapidly boost the voltage without changing the clean supply. The technique isolates the supply voltage of boost part from other supplies to avoid supply drop in clean part. Implementing this technique adds an extra energy overhead which depends on the number of relaxed bits and the VOS levels (1-HD or 2-HD). To reduce the energy overhead, we implement this technique on 8-bit granularity of TCAM. Therefore, ReCAM implements Shortstop on first 8, 16, 24 or 32 bits.

B. ReCAM with Row Configurable

As another method of approximation, we apply VOS on selective TCAM rows storing different patterns. In TCAM, all rows do not have equal probability of hit, hence our technique connects supply voltage of a row with low frequency of a hit event to Vdd related to 1-HD or 2-HD approximation. In such row-configurable ReCAM, the number of approximate rows can dynamically change based on running applications. We use the ShortStop technique for fast and efficient changes of Vdd of the TCAM rows. This VOS is implemented on 4-row granularity to decrease the overhead of voltage boosting. This configurable TCAM has more impact on energy saving than bitline segmentation. Let us consider a TCAM that half of the TCAM bitline are in 1-HD approximate mode. In this case, all TCAM rows accept one Hamming distance on selected bits

resulting in huge QoL. On the other hand, if we implement approximation on 50% of TCAM rows, the TCAM accept 1-HD on only 50% of the low frequent TCAM rows and the rest of the TCAM rows are in the exact matching. This reduction on the number of approximate rows: (i) allows us to use TCAM with higher number of rows since row-configurable ReCAM deceases the number of wrong matches; (ii) for the same error-rate, row-configurable ReCAM increases the number of cells in the approximate mode and hence improves energy efficiency.

IV. ReCAM Integration with GP-GPUs

A. GP-GPUs Architecture

We focus on the AMD Southern Islands GPUs which is a RISC single instruction, multiple data (SIMD) architecture. We target Radeon HD 7970 device which has 32 compute units. Every compute unit contains a scheduler and a set of four SIMD execution units, aka vector units. Each SIMD execution unit has 16 stream cores, or parallel lanes, constituting a total number of 64 stream cores per compute unit.

An OpenCL application is formed of a host program and one or more device kernels that can be run on a GPU device. An instance of the OpenCL kernel is called a work-item. Each stream core is devoted to the execution of one work-item using the integer or FP units. Most arithmetic operations on a GP-GPU are performed by vector instructions. A vector instruction is fetched once and executed in a SIMD fashion by all its comprising work-items. After the fetch and decode stages, the source operands for each instruction are read from vector registers or local memory. The core stage of a GP-GPU is the execute stage, where arithmetic instructions are carried out in each stream core. When the source operands are ready in the vector unit, the execution stage starts to issue the operations into the integer units or FPUs. The execution stage of every FPU has a latency of six cycles and a throughput of one instruction per cycle [32]. Finally, the result of the computation is written back to the destination operands. In the following, we briefly describe our approach to programming ReCAM and design space exploration for ReCAM in improving energy efficiency of GP-GPUs.

B. ReCAM Execution Flow

Execution flow using ReCAM has two main stages: (i) profiling, and (ii) runtime computational reuse. The goal of profiling stage is to identify redundant computations with a high frequency of occurrence. This profiling stage is a one-off activity whose cost is amortized across all future usage of the kernel. In profiling, we pre-store the frequent patterns corresponding the each application and then write them on TCAM and 1T-1R memory on the runtime. This update can be done very fast, since the write operation on both TCAM and 1T-1R memory can be done on less than 5ns. In the profiling stage, we have an OpenCL kernel, a host code with a training input dataset. We focus on the individual FPUs to observe the dispersion of the input operands at the finest granularity. To expose highly frequent set of operands for each FP operation, we individually profile every type of FP operation and keep the distinct sets of the input operands with the related output result. The output of this stage for every FP operation is highly frequent computations (HFC): a sorted list of sets of values, each set has the input operand(s) and the related result, and the sets are sorted based on their frequency of occurrence. After extracting HFC, we need to determine how much approximation can be tolerated during the reuse of these key computations. To do so, we leverage the Southern Islands functional simulator to determine the degree of approximation applicable to each ReCAM module. In bitline-configurable ReCAM, each TCAM bitline is partitioned to four segments with 8-bit. The voltage relaxation can be implemented on each segment separately. Row-configurable ReCAM relaxes the TCAM rows in 4-row and 8-row granularity. The flow of finding the number of approximate TCAM segments/rows for each application is illustrated on Fig. 2.

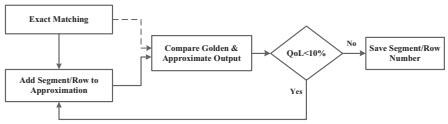


Fig. 2. Flow of ReCAM to determine approximate segments/rows.

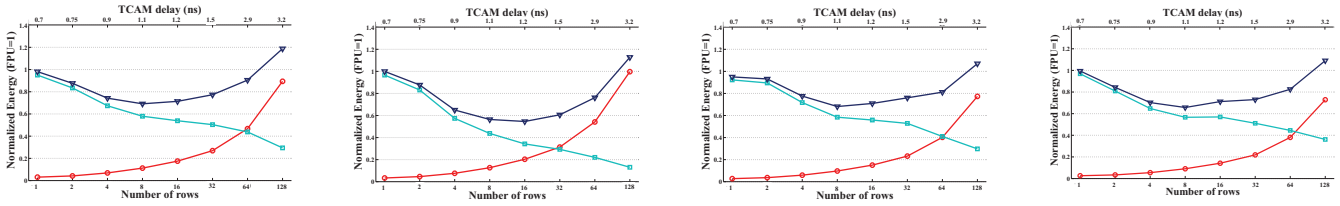
The algorithm starts selecting all TCAM bitline (rows) in the exact matching mode to find a golden error-free output. Then, it gradually applies 1-HD and 2-HD approximations on the first TCAM segment (or low-frequent TCAM rows) of each TCAM. The output result is compared with the golden output to measure the QoL. If the QoL is less than 10%, our algorithm selects more number of segments (rows) for the approximation mode. We set the QoL to a maximum of 10% which is commensurate with other work on quality trade-offs [20]. Finally, the maximum degree of approximation is determined for each ReCAM module such that the QoL is lower than the desired value (e.g., 10% [20]).

C. Design Space Exploration for ReCAM

1) TCAM Sizing: Fig. 3 compares energy consumption of FPU, ReCAM, and integrated (FPU+ReCAM) for four different studied applications. For ReCAM, we increase the size of TCAM that imposes larger delay; we accordingly measure the energy of FPU for each corresponding delay point. All energy values are normalized to the FPU energy with typical delay of a 96-bit TCAM. For ReCAM, the TCAM search energy is always an available term on total energy consumption, independent of TCAM hit rate. However, in integrated FPU+ReCAM, when an input pattern hits in the TCAM, a hit signal clock-gates the FPU computation and limits FPU energy consumption to only leakage energy. The hit rate improvement is not linear with the number of TCAM rows. For instance, moving from 4-row to 8-row TCAM has more impact on TCAM hit rate improvement compared to moving from 64-row to 128-row TCAM.

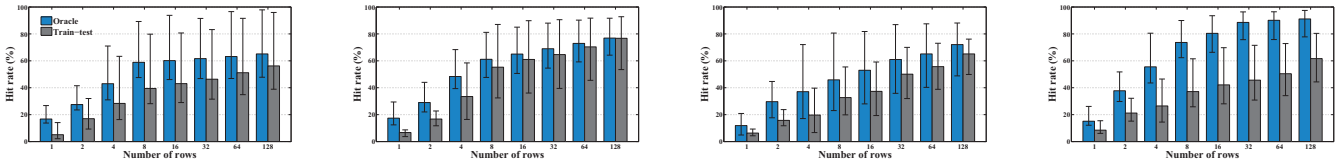
In most of the studied applications using 32-row or 64-row TCAM, covers most highly frequent patterns and TCAM with high number of rows, only adds a few low frequent patterns to TCAM which do not have major impact on hit rate improvement. This results in a tradeoff between ReCAM and FPU energy on total computational energy. TCAM in small ReCAM consumes very low search energy, so doubling the number of TCAM rows reduces the FPU energy consumption by improving TCAM hit rate. While in large ReCAM size, TCAM search energy is a dominant term of total energy consumption and doubling TCAM size does not have significant impact on hit rate improvement. Thus, the minimum energy point occurs on TCAM size with 8-row or 16-row where energy is balanced between ReCAM and FPU energy.

2) TCAM Content Update Policy: One possible way to improve energy efficiency of computation using associative memory is to have run-time profiling. Profiling in run-time is able to fill the TCAM with more realtime data representative of local values of the current application. The goal of this section is to answer the following question: *How much is the ReCAM energy efficiency using run-time profiling?*. This can be a guideline to observe the feasibility of using associative memory with realtime profiling capability.



(a) Sobel (b) Robert (c) HwtHaar1D (d) MatrixMul

Fig. 3. FPU, ReCAM, and FPU+ReCAM energy consumption with different TCAM sizes (or corresponding delay) in the exact matching mode.



(a) Sobel (b) Robert (c) HwtHaar1D (d) MatrixMul

Fig. 4. Applications hit rate (min, max, and average) for various TCAM sizes using Oracle and train-test policies.

To answer this question, at first we should calculate the maximum hit rate improvement that can be achieved using such profiling; then, we can evaluate the possible potential energy saving on run-time profiling. We use an Oracle as the best-case estimation of the run-time profiling where the system can see all possible input patterns. To implement Oracle in our design, we train and test on 100% of the input samples. For each application, the train and test are done separately for 100 different inputs. In this paper we compare Oracle with a simple training (train-test) technique where the train and test are done on 10% and 100% of the input dataset respectively. For train-test profiling, we use the same size dataset containing 100 different inputs. For image processing applications we use Caltech 101 [34] as dataset, while for other applications we generate the dataset using random stream data. In this training, we rank all input patterns and then choose the top frequent ones for each FPU and finally fill those top patterns on the corresponding ReCAM based on its available size. Fig. 4 shows the hit rate using Oracle and train-test update policies for TCAM with different sizes.

The results show that Oracle can achieve up to 18% higher hit rate with respect to the traintest. As Fig. 4 shows, TCAM with low number of rows (1-row or 2-rows) has very low hit rate. Doubling TCAM rows from 1-row at first increases the TCAM hit rate severely but then the trend of the hit rate improvement saturates. The change of hit rate depending on application type can be fast or slow in different TCAM sizes. In the train-test, this hit rate improvement saturates in larger TCAM size since train-test consists more variety of input patterns and needs larger TCAM to cover all high frequent ones. But in the case of Oracle, a TCAM with a few number of rows can cover the major frequent patterns.

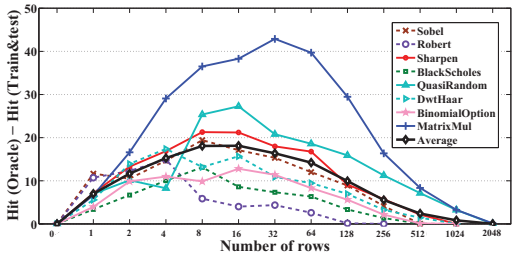


Fig. 5. Oracle and train-test hit rate difference for different applications type.

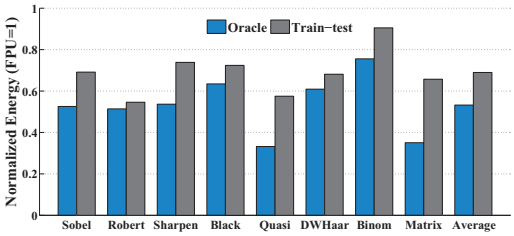


Fig. 6. The best GP-GPU minimum energy points using ReCAM in Oracle and train-test.

We define $\Delta Hitrate$ as the hit rate difference of Oracle and train-test profiling. As Fig. 5 shows, $\Delta Hitrate$ is low in small TCAM size because the absolute hit rate values of both profiling techniques are small. In middle size TCAMs (8-row and 16-row), $\Delta Hitrate$ is maximized because of filling the TCAM spaces with more frequent patterns by Oracle with respect to the train-test technique. In very large TCAM sizes (>256 rows), the $\Delta Hitrate$ drops again ($\Delta Hitrate$ <0.1%) because the train-test hit rate is high while Oracle hit rate is saturated. This fact is also shown in Fig. 4 in which, an Oracle TCAM with 64-rows and 128-rows has very close hit rate, meaning that larger TCAMs will not have positive impact on the overall energy efficiency. Fig. 6 shows the minimum energy point considering the best TCAM size (from 1-row to 128-row) with Oracle and train-test. These values are potential energy saving since they are achievable using different TCAM sizes. Our evaluations show Oracle improves the GP-GPU energy efficiency just by 15.7% compared to the train-test technique. However, implementing such run-time profiling may cause significant energy overhead to keep track and rank of the incoming patterns.

V. Experimental Results

A. Experimental Setup

We implement our proposed ReCAM on the AMD Southern Island GPU, Radeon HD 7970 device, which is one of the most recent GP-GPU architectures. We use Multi2sim [32], a cycle accurate CPU-GPU simulator and modified the code to do profiling and run-time simulation. Eight GP-GPU applications *Sobel, Roberts, HwtHaar1D, MatrixMul, BlackScholes,*

Sharpen, QuasiRandom and *BinomialOption* from AMD APP SDK v2.5 [30] are used to set the efficiency of the proposed ReCAM. We extracted the frequent patterns for various FPUs including adder (ADD), multiplier (MUL), multiply-accumulator (MAC) and SQRT Operations. The FPUs are balanced for 6-stage using FloPoCo [33] and are synthesized by *Synopsys Design Compiler* in 45-nm ASIC flow. FPUs are optimized for power based on different TCAM delays.

FPUs accept different number of input operands. The ADD and MUL accept two 32-bit, SQRT a 32-bit and MAC three 32-bit input operands. Therefore, their related TCAMs need to have 64-bit, 32-bit and 96-bit word-size respectively. The circuit level simulation of TCAM design has been done on *HSPICE* simulator with 45-nm technology. For 3T-1R TCAM design we extract parameters (sizing, resistors and capacitors, etc.) used in [14]

B. Energy Saving and QoL

Based on our explanations in section IV, implementing approximation on entire TCAM bitline significantly degrades the computation accuracy. Having mismatch on LSBs and MSBs of input pattern does not have the same effect on the computation accuracy. In addition, all TCAM rows do not have the same impact on the result of computation such that any mismatches in the first TCAM rows, with high probability of hit, degrades the QoL more than other rows. Our bitline-configurable and row-configurable ReCAM keep the non-sensitive segments/rows of TCAM in the approximation mode at 8-bit and 8-row (and also 4-row) granularity. We consider the energy overhead of Shortstop technique during approximation of different TCAM segments/rows. We use extensive simulations on Multi2Sim to calculate the QoL for each application. This constraint is to have less than 10% QoL, commensurate with [20], on the output data for all inputs. Our results show that each application satisfies the QoL with different bit indices for 1-HD and 2-HD approximation.

Fig. 8 shows the normalized energy consumption and QoL, implementing voltage relaxation on different TCAM segments. The overall energy efficiency depends on two parameters. (i) Approximate TCAM consumes lower search energy depending on the depth of approximation (1-HD or 2-HD) and number of segments on approximate mode. (ii) Higher TCAM hit rate increases the average time that FPU is clock-gated and saves energy. Thus, increasing the coverage of approximation from 0-bit toward 32-bit reduces the GP-GPU energy consumption. However, approximation on the more than half of the TCAM bitline ($>$16-bits) degrades the accuracy of results severely which indicates the sensitivity of computation to the place of mismatches. Fig. 8 indicates that moving from 24-bits (3-segment) approximation to 32-bit approximation (4-segment) degrades computational accuracy without major impact on GP-GPU energy efficiency (due to high ShortStop energy overhead). Implementing 1-HD (2-HD) approximation on the first 8-bits of TCAM results on average 35.3% (38.8%) energy savings among all applications with the acceptable QoL ($<$10%). Dynamically changing the approximate bit index for each application improves 1-HD (2-HD) energy saving to 39.1% (43.6%) with the same QoL.

Fig. 7 shows normalized GP-GPU computation energy and QoL using row-configurable ReCAM running eight applications. Row-configurable ReCAM implements voltage relaxation on entire bitline of the selective TCAM rows. We start row approximation from the last TCAM rows with the lowest probability of hit. As the number of rows in approximate mode increases, GP-GPU energy efficiency improves proportionally at the expense of computational inaccuracy. The comparison of Fig. 8 and Fig. 7 in 8-bit and 8-row approximate granularity shows in same percentage of approximation, GP-GPU with bitline-configurable ReCAM achieves lower energy consumption and higher QoL compared to the row-configurable ReCAM. This is due to the fact that bitline-configurable ReCAM selects all TCAM rows on approximate matching while the row-configurable ReCAM limits the number of approximate rows to a few selective ones. For example in 25% approximation, the bitline-configurable and row-configurable ReCAM put respectively 100% and 25% of the TCAM rows on approximation. This feature improves the computational accuracy of the row-configurable ReCAM on partial approximation with respect to bitline-configurable ReCAM. However, higher hit rate of the bitline-configurable ReCAM in partial approximation improves the GP-GPU energy efficiency as compared to row-configurable one. On the other hand, implementing approximation on a high portion of ReCAM ($>$24-row & $>$24-bit), results in same energy savings and QoL in both row and bitline-configurable ReCAM because of their similar approximate rows. The results show that GP-GPU using row-configurable ReCAM can meet QoL with enabling 75% and 53% of TCAM cells in 1-HD and 2-HD approximate mode which results in 44.5% and 42.9% energy savings. The number of relaxed cells in 1-HD and 2-HD bitline-configurable ReCAM is about 65% and 46% respectively.

Low QoL of row-configurable TCAM in partial approximation motivates us to implement it in lower granularity. Table I shows the maximum number of relaxed rows, normalized GP-GPU computational energy and QoL using row-configurable ReCAM with 4-row granularity and acceptable quality of loss. In this row-configurable ReCAM, GP-GPU can achieve 44.7% and 48.6% energy savings on average implementing 1-HD and 2-HD approximation.

Our evaluation shows that the bitline-configurable ReCAM achieves high energy efficiency for applications with low sensitivity to QoL while row-configurable ReCAM is more appropriate architecture for highly sensitive applications to QoL such as *Sobel*, *BlackScholes* and *BinomialOption*.

VI. Conclusion

We propose a resistive configurable associative memory which implements approximation on selective TCAM bitline and rows to balance the depth of approximation based on running applications. The proposed ReCAM is based on emerging NVMs: (i) it decreases the QoL by applying voltage relaxation on selective bits and TCAM rows with lower sensitivity. (ii) It has capability to adaptively change the depth of approximation based on running application to guarantee the desired accuracy. The experimental results on the AMD Southern Islands GPUs show that bitline-configurable and row-configurable ReCAM reduce the GP-GPU energy computation by 43.6% and 44.5% on average with delivering the acceptable QoL.

VII. Acknowledgments

This work was supported by NSF grant#1527034.

References

[1] "Green5000. Available:http://www.green500.org/"

[2] S. Huang, et al., "On the energy efficiency of graphics processing units for scientific computing," *IEEE IPDPS*, 2009.

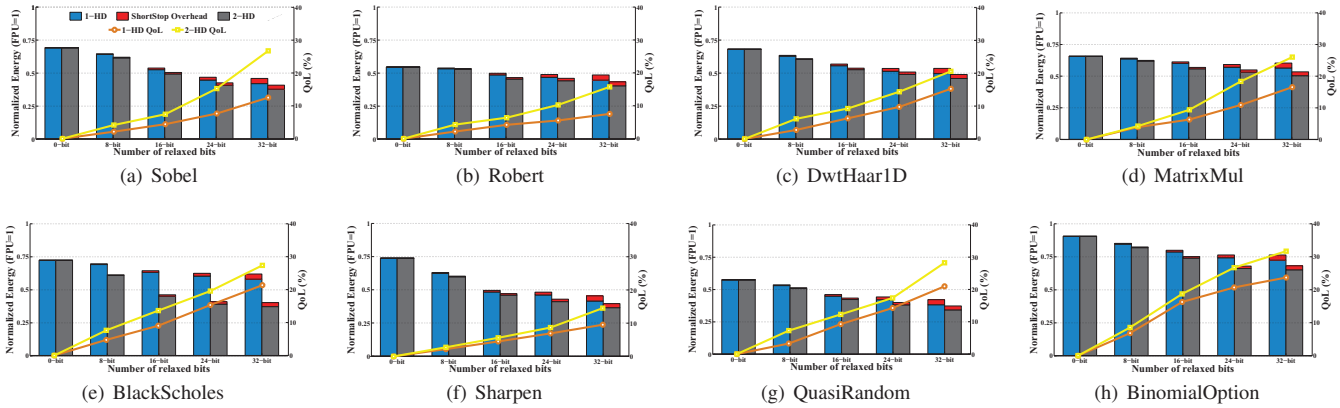
[3] A. Rahimi, et al., "Energy-efficient gpgpu architectures via collaborative compilation and memristive memory-based computing," *IEEE DAC*, pp. 1–6, 2014.

[4] Q. Guo, et al., "AC-DIMM: associative computing with STT-MRAM," *ACM SIGARCH*, Vol. 41. No. 3, 2013.

[5] N. Bandi, et al., "Fast data stream algorithms using associative memories," *ACM SIGMOD*, 2007.

[6] C. Ranger, et al., "Evaluating mapreduce for multi-core and multiprocessor systems," *IEEE HPCA*, 2007.

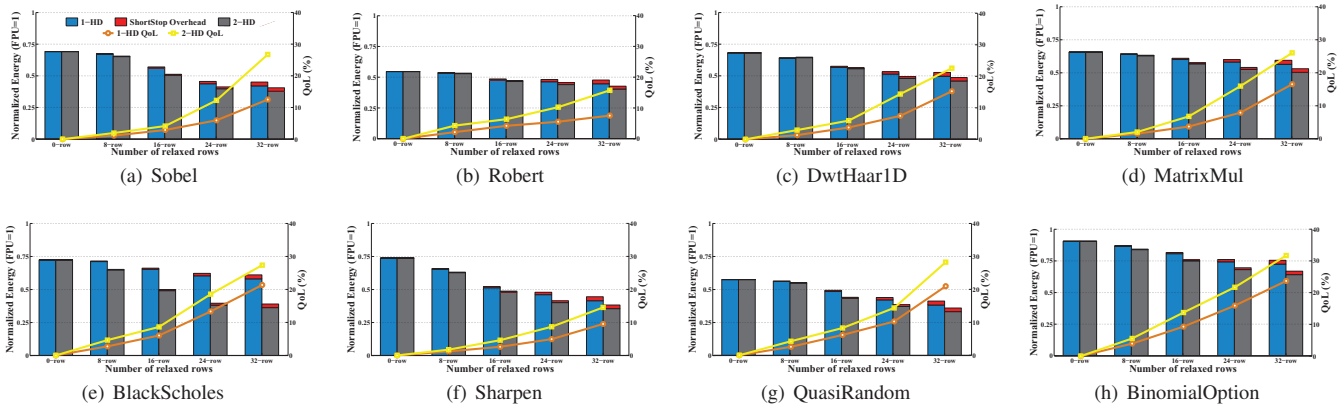
[7] R. Agrawal, et al., "Fast algorithms for mining association rules," *IEEE VLDB*, Vol. 1215. 1994.



(a) Sobel (b) Robert (c) DwtHaar1D (d) MatrixMul

(e) BlackScholes (f) Sharpen (g) QuasiRandom (h) BinomialOption

Fig. 7. Normalized GP-GPU energy consumption and QoL of bitline-configurable ReCAM.



(a) Sobel (b) Robert (c) DwtHaar1D (d) MatrixMul

(e) BlackScholes (f) Sharpen (g) QuasiRandom (h) BinomialOption

Fig. 8. Normalized GP-GPU energy consumption and QoL of row-configurable ReCAM.

TABLE I. BEST GP-GPU ENERGY POINT WITH 4-ROW GRANULARITY ROW-CONFIGURABLE RECAM.

GP-GPU	Sobel		Robert		DwtHaar1D		MatixMul		BlackScholes		Sharpen		QuasiRandom		BinomialOption	
	1-HD	*2-HD*	*1-HD*	*2-HD*	*1-HD*	*2-HD*	*1-HD*	*2-HD*	*1-HD*	*2-HD*	*1-HD*	*2-HD*	*1-HD*	*2-HD*	*1-HD*	*2-HD*
Relaxed rows	28	20	32	24	24	20	24	16	20	16	32	28	20	16	16	12
Norm. Energy	0.45	0.41	0.47	0.45	0.53	0.52	0.59	0.57	0.64	0.50	0.44	0.41	0.47	0.46	0.83	0.79
QoL (%)	8.7	7.4	7.5	8.4	7.4	9.2	7.9	6.8	9.3	8.6	9.5	9.7	9.1	8.2	9.2	7.4

[8] R. Ubal, et al., "Multi2Sim: a simulation framework for CPU-GPU computing," *ACM PACT*, 2012.

[9] T. Kohonen, "Content-addressable memories," *Springer Science and Business Media*, Vol. 1, 2012.

[10] J. Li, et al., "1 mb 0.41 μm^2 2t-2r cell nonvolatile tcam with two-bit encoding and clocked self-referenced sensing," *IEEE JSSC*, pp. 896–907, April 2014.

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

[12] K. Lakshminarayanan, et al., "Algorithms for advanced packet classification with ternary CAMs," *ACM IGCOMM*, Vol. 35. No. 4, 2005.

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

[14] M. -F. Chang, et al., "A 3T1R Nonvolatile TCAM Using MLC ReRAM with Sub-1ns Search Time," *IEEE ISSCC*, Vol. 58, 2015.

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

[16] A. Rahimi, et al., "Approximate associative memristive memory for energy-efficient GPUs," *IEEE DATE*, 2015.

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

[18] W. Eatherton, et al., "Tree bitmap: hardware/software IP lookups with incremental updates," *ACM SIGCOMM*, pp. 97–122, 2004.

[19] T. Kohonen, "Low-leakage storage cells for ternary content addressable memories," *IEEE TVLSI*, Vol. 17, pp. 604-612, 2009.

[20] H. Esmaeilzadeh, et al., "Neural acceleration for general-purpose approximate programs," *IEEE Micro*, 2012.

[21] A. Goel, et al., "Small subset queries and bloom filters using ternary associative memories, with applications," *ACM SIGMETRICS*, Vol. 38. No. 1, 2010.

[22] K. Lakshminarayanan, et al., "Algorithms for advanced packet classification with ternary CAMs," *ACM SIGCOMM*, Vol. 35. No. 4, 2005.

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

[24] W. Xu, et al., "Design of spin-torque transfer magnetoresistive RAM and CAM/TCAM with high sensing and search speed," *IEEE TVLSI*, Vol. 18, pp. 66-74, 2010).

[25] Y. Kim, et al., "CAUSE: Critical application usage-aware memory system using non-volatile memory for mobile devices," *ACM ICCAD*, pp. 690-696, 2015.

[26] A. Rahimi, et al., "Temporal memoization for energy-efficient timing error recovery in gpgpus," *IEEE DATE*, pp. 1–6, March 2014.

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

[28] D. Mohapatra, et al., "Design of voltage-scalable meta-functions for approximate computing," *IEEE DATE*, pp. 1–6, March 2011.

[29] 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*, June 2014.

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

[31] N. Pinckney, et al. "Shortstop: An on-chip fast supply boosting technique," *IEEE VLSIC*, 2013.

[32] "Multi2sim [online]. Available: https://www.multi2sim.org/"

[33] "Flopoco [online]. Available:http://flopoco.gforge.inria.fr/"

[34] "Caltech 101 [online]. http://www.vision.caltech.edu/Image_Datasets/Caltech101/"