

# A Resistive TCAM Accelerator for Data-Intensive Computing\*

Qing Guo<sup>2</sup> Xiaochen Guo<sup>1</sup> Yuxin Bai<sup>1</sup> Engin İpek<sup>1,2</sup>

<sup>1</sup>Department of Electrical and Computer Engineering

<sup>2</sup>Department of Computer Science

University of Rochester

Rochester, NY 14627 USA

{qing.guo, yuxin.bai, xiaochen.guo, engin.ipek}@rochester.edu

## ABSTRACT

Power dissipation and off-chip bandwidth restrictions are critical challenges that limit microprocessor performance. Ternary content addressable memories (TCAM) hold the potential to address both problems in the context of a wide range of data-intensive workloads that benefit from associative search capability. Power dissipation is reduced by eliminating instruction processing and data movement overheads present in a purely RAM based system. Bandwidth demand is lowered by processing data directly on the TCAM chip, thereby decreasing off-chip traffic. Unfortunately, CMOS-based TCAM implementations are severely power- and area-limited, which restricts the capacity of commercial products to a few megabytes, and confines their use to niche networking applications.

This paper explores a novel resistive TCAM cell and array architecture that has the potential to scale TCAM capacity from megabytes to gigabytes. High-density resistive TCAM chips are organized into a DDR3-compatible DIMM, and are accessed through a software library with zero modifications to the processor or the motherboard. On applications that do not benefit from associative search, the TCAM DIMM is configured to provide ordinary RAM functionality. By tightly integrating TCAM with conventional virtual memory, and by allowing a large fraction of the physical address space to be made content-addressable on demand, the proposed memory system improves average performance by 4× and average energy consumption by 10× on a set of evaluated data-intensive applications.

## Categories and Subject Descriptors

B.3 [Memory Structures]

## General Terms

Design, Performance

## Keywords

Resistive memory, TCAM, Accelerator

\*This work was supported in part by NSF CAREER award CCF-1054179 and gifts from Qualcomm.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MICRO'11 December 3-7, 2011, Porto Alegre, Brazil  
Copyright 2011 ACM 978-1-4503-1053-6/11/12 ...\$10.00.

## 1. INTRODUCTION

As CMOS scaling continues into the billion transistor era, power dissipation and off-chip bandwidth limitations are threatening to bring an end to microprocessor performance growth. Data intensive applications such as data mining, information retrieval, video processing, and image coding demand significant computational power and generate substantial memory traffic, which places a heavy strain on both off-chip bandwidth and overall system power. Device, circuit, and architecture innovations are needed to surmount this problem.

Ternary content addressable memories (TCAM) are an attractive solution to curb both power dissipation and off-chip bandwidth demand in a wide range of data-intensive applications. When associative lookups are implemented using TCAM, data is processed directly on the TCAM chip, which decreases off-chip traffic and lowers bandwidth demand. Often, a TCAM-based system also improves energy efficiency by eliminating instruction processing and data movement overheads that are present in a purely RAM based system. Unfortunately, even an area-optimized, CMOS-based TCAM cell is over 90× larger than a DRAM cell at the same technology node, which limits the capacity of commercially available TCAM parts to a few megabytes, and confines their use to niche networking applications.

This paper explores a new technique that aims at cost-effective, modular integration of a high-capacity TCAM system within a general-purpose computing platform. TCAM density is improved by more than 20× over existing, CMOS-based parts through a novel, resistive TCAM cell and array architecture. High-capacity resistive TCAM chips are placed on a DDR3-compatible DIMM, and are accessed through a user-level software library with zero modifications to the processor or the motherboard. The modularity of the resulting memory system allows TCAM to be selectively included in systems running workloads that are amenable to TCAM-based acceleration; moreover, when executing an application or a program phase that does not benefit from associative search capability, the TCAM DIMM can be configured to provide ordinary RAM functionality. By tightly integrating TCAM with conventional virtual memory, and by allowing a large fraction of the physical address space to be made content-addressable on demand, the proposed memory system improves average performance by 4× and average energy consumption by 10× on a set of evaluated data-intensive applications.

## 2. BACKGROUND AND MOTIVATION

We review CMOS-based ternary CAMs to motivate resistive TCAM systems, and provide background on resistive memories as applicable to memory system design.

### 2.1 Associative Computing and Ternary Content Addressable Memories

An effective way of addressing power and bandwidth limitations on many data intensive workloads is to use memories that are accessed based on content (also named association [28]), rather than index. Associative computing, which leverages memory systems that store and retrieve data by association, has been broadly applied to both software and hardware design. The best known software solution is a hash table, whereby data is located by an address computed through a hash function. A hash table has  $O(1)$  lookup time and is proven more efficient than other data structures in handling sparse data (i.e., when the number of keys in use is far less than the total number of possible keys). On the hardware side, a simple example of associative computing is the use of content addressable memory (CAM), which has seen wide use since 1970's [9]. Nowadays, CAMs are commonly used in highly associative caches, translation lookaside buffers (TLBs), and microarchitectural queues (e.g., issue queues). Compared to a hash table, a CAM has no collision problems and offers better storage utilization and shorter search time; moreover, a CAM avoids the software overhead of rehashing and chaining.

A ternary CAM (TCAM) is a special type of associative memory that allows for both storing and searching with a wildcard (X) in addition to a logic zero or one. A wildcard, when part of either the search key or the stored data word, matches against both binary states (as well as another wildcard). This flexibility can be exploited by a surprisingly large number of applications.

TCAM has been widely used in networking for decades. The primary commercial application of earlier generation TCAM was in routers [38], where it was used to store the routing table and to perform fast lookups through longest prefix matching [52]. With technology scaling over the last 20 years, TCAM capacity has increased from 64Kb [57] to 9Mb [26], while typical array width has grown from 72 bits to 576 bits. Numerous networking applications have emerged to leverage the benefits of TCAM, including packet classification [29], access control list filtering [39], and network intrusion detection [11].

Table 1 shows a comparison among CMOS-based TCAM, SRAM, and DRAM chips. State-of-the-art TCAM devices are as fast as SRAM, but the cost-per-bit is  $8\times$  higher due to the lower density of TCAM cells. Furthermore, TCAM is  $10\times$  more power hungry than SRAM and DRAM.

An example TCAM cell is shown in Figure 1: the two pairs of cross-coupled inverters act as bistable storage elements that hold the cell's value, and the two access transistors  $M1$  and  $M2$  are used to write a new value to the cell. On a search, the cross-coupled inverters supply the cell's contents, and the bottom four NMOS transistors ( $M3 - M6$ ) compare the search key to the data stored in the cell.

When searching for a 0 or 1, searchlines supply the complementary search values  $SL$  and  $\overline{SL}$ ; when searching with a wildcard (X), both  $SL$  and  $\overline{SL}$  are driven low. On a mismatch, one of the pull-down paths ( $M3 - M5$  or  $M4 - M6$ ) is activated; on a match, all pull-down paths are inac-

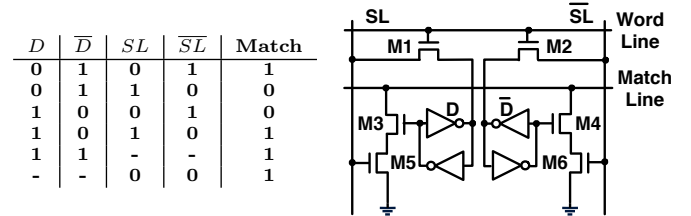


Figure 1: Illustrative example of a CMOS TCAM cell and its truth table.

tive. A CMOS TCAM cell is relatively large: even with an area-efficient implementation that uses half-latches instead of cross-coupled inverters [8], its area is  $541F^2$  (where  $F$  is the feature size), which is  $3.8\times$  as large as an SRAM cell, and over  $90\times$  as large as a DRAM cell.

### 2.2 Resistive Memory Technologies

As CMOS scales to 22nm and beyond, charge-based memory technologies such as DRAM, Flash, and SRAM are starting to experience scalability problems [24]. In response, the industry is exploring resistive memory technologies that can serve as scalable, long-term alternatives to charge-based memories. Resistive memories, which include phase-change memory (PCM) and spin-torque transfer magnetoresistive RAM (STT-MRAM), store information by modulating the resistance of nanoscale storage elements, and are expected to scale to much smaller geometries than charge-based memories. Moreover, resistive memories are non-volatile, which provides near-zero leakage power and immunity to radiation induced transient faults. However, because resistive memories need to change material states to update stored data, they generally suffer from high write energy, long write latency, and limited write endurance.

**PCM.** PCM [30] is arguably the most mature among all resistive memory technologies, as evidenced by 128Mb parts that are currently in production [16], as well as gigabit array prototypes [51, 12]. A PCM cell is formed by sandwiching a chalcogenide phase-change material such as  $Ge_2Sb_2Te_5$  (GST) between two electrodes. PCM resistance is determined by the atomic ordering of this chalcogenide storage element, and can be changed from less than  $10K\Omega$  in crystalline state to greater than  $1M\Omega$  in amorphous state [51, 24]. On a write, a high amplitude current pulse is applied to the cell to induce Joule heating. A slow reduction in write current causes the device to undergo a fast annealing process, whereby the material reverts to a crystalline state. Conversely, an abrupt reduction in current causes the device to retain its amorphous state. On a read, a sensing current lower than the write current is passed through the cell, and the resulting voltage is sensed to infer the cell's content. Since the ratio of the high ( $R_{HI}$ ) and low ( $R_{LO}$ ) resistances is as high as 100, a large sensing margin is possible; however, the absolute resistance is in the mega-ohm range, which leads to large RC delays, and hence, slow reads [24]. PCM also suffers from finite write endurance; the typical number of writes that can be performed before a cell wears out is  $10^6 - 10^8$  [24]. Consequently, several architectural techniques have been proposed to alleviate PCM's wear-out problem [47, 23].

**STT-MRAM.** As a universal embedded memory candidate, STT-MRAM has a read speed as fast as SRAM [62], practically unlimited write endurance [4], and favorable de-

Memory	Single-chip Capacity	\$ / chip	\$ / MByte	Access Speed (ns)	Watts / chip	Watts / MByte
DRAM	128MB	\$10-\$20	\$0.08-\$0.16	40-80	1-2	0.008-0.016
SRAM	9MB	\$50-\$70	\$5.5-\$7.8	3-5	1.5-3	0.17-0.33
TCAM	4.5MB	\$200-\$300	\$44.5-\$66.7	4-5	15-20	3.33-4.44

Table 1: Comparison of high-speed memory technologies [18].

lay and energy scaling characteristics [24]. Multi-megabit array prototypes at competitive technology nodes (e.g., 45nm, 65nm) have already been demonstrated [56, 33], and the ITRS projects STT-MRAM to be in production by 2013 [24]. In STT-MRAM, information is stored by modulating the magnetoresistance of a thin film stack called a magnetic tunnel junction (MTJ). An MTJ is typically implemented using two ferromagnetic  $Co_{40}Fe_{40}B_{20}$  layers, and an MgO tunnel barrier that separates the two layers. One of the ferromagnetic layers, the *pinned layer*, has a fixed magnetic spin, while the magnetic spin of the *free layer* can be altered by applying a high-amplitude current pulse through the MTJ. Depending on the direction of the current, the magnetic polarity of the free layer can be made either parallel or antiparallel to that of the pinned layer. In the case of parallel alignment, the MTJ exhibits high resistance (12.5K $\Omega$  at 22nm [24]); in the case of antiparallel alignment, a low resistance (5K $\Omega$ ) is observed [24]. Although the typical  $\frac{R_{HI}}{R_{LO}}$  ratio (2.5) is lower than that of PCM (100), it is still relatively easy to sense the state of a single bit [54].

### 3. OVERVIEW

This paper proposes a novel resistive TCAM chip, which can be integrated on a DDR3-compatible DIMM and selectively placed on the memory bus. Figure 2 presents an example computer system with the proposed TCAM DIMM. A multicore processor connects to main memory through an on-chip memory controller. The TCAM DIMM sits side-by-side with DRAM on the DDR3 bus. An on-DIMM TCAM controller serves as the interface to DDR3, and is in charge of DIMM control. The processor communicates with the controller through a set of memory-mapped control registers (for configuring functionality) and a memory-mapped key store that resides with the controller (for buffering the search key). A 2KB result store on the controller die buffers search results for multiple processes. All TCAM chips share the on-DIMM command, address, and data buses; however, a search operation is restricted to be on a single chip due to power constraints. Each TCAM chip has 8 banks; a bank comprises a set of arrays that are searched against the query key, as well as a hierarchical reduction network for counting the number of matches and picking the highest-priority matching row.

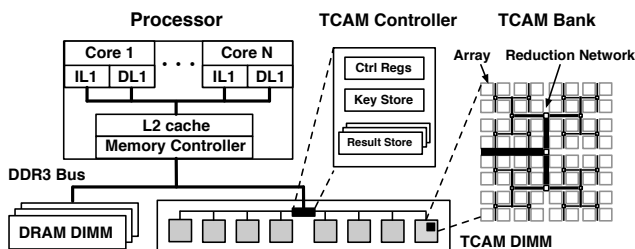


Figure 2: Illustrative example of a computer system with the proposed resistive TCAM DIMM.

## 4. CIRCUIT-LEVEL BUILDING BLOCKS

Building a high-performance, low-power TCAM system requires the development of a high-density resistive TCAM cell, as well as an appropriate row organization with attendant sensing and writing mechanisms. However, achieving the required cell density is complicated by the fact that cells need to be written as well as searched, which, in a naïve implementation, would require multiple access transistors per storage element to update cell contents. A TCAM design that leverages the matchlines to write the cells is proposed next to address this challenge.

### 4.1 Resistive TCAM Cell

The area of a TCAM cell not only affects the cost per bit in a multi-megabit array, but also has a profound effect on speed and energy since it determines the length (and thus, capacitance) of the matchlines and searchlines that need to be charged and discharged on every access. Figure 3 demonstrates the proposed area-efficient resistive TCAM cell. In the figure, a resistor represents a resistive storage element, which could be a GST stack or an MTJ (the impact of the exact technology on row organization, array architecture, and performance is explained later in Sections 4.2, 5, and 8). A TCAM cell consists of three pairs of resistors and access transistors. The first two resistors store the data bit and its complement; the third resistor is permanently programmed to  $R_{HI}$ . To store a logic 1 or 0, the leftmost resistor is programmed to store the data bit ( $D$ ), while the resistor in the middle is programmed to store the complement of the bit ( $\bar{D}$ ). For example, when storing a logic 1 (Figure 4-a), the resistor on the left is programmed to  $R_{HI}$ , and the resistor in the middle is programmed to  $R_{LO}$ . To store a wildcard ( $X$ ), the two leftmost resistors are both programmed to  $R_{HI}$ .

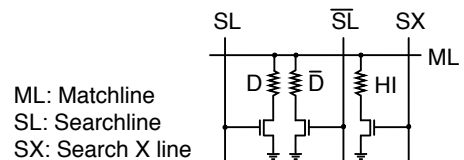
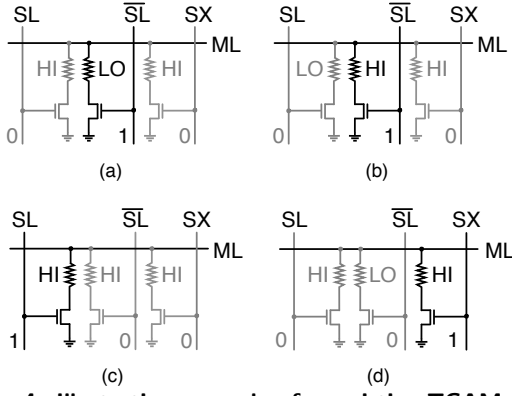


Figure 3: Illustrative example of a resistive TCAM cell.

To search for a logic 0 or 1,  $SL$  and  $\bar{SL}$  are driven with the search bit and its complement, respectively, turning one of the access transistors on, and the other off. A match is decided based on the effective resistance between the matchline and ground. If a resistor in its high-resistance state is in series with the on transistor—adding a resistance of  $R_{HI}$  between the matchline and ground—the search results in a match; conversely, a resistance of  $R_{LO}$  connected to the matchline indicates a mismatch. To search for a wildcard ( $X$ ),  $SL$  and  $\bar{SL}$  are disabled and  $SX$  is driven high; hence, a resistor in its  $R_{HI}$  state is connected to the matchline regardless of the value stored in the cell. Examples are shown in Figure 4: (a) demonstrates a mismatch case when the search bit is 0 and stored data is 1; (b) presents a match scenario which searches for a 0 when a 0 is stored; (c) shows

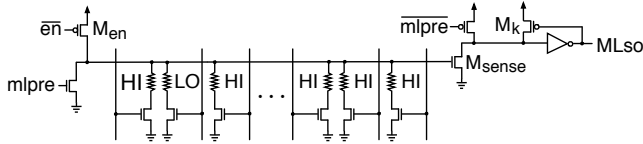


**Figure 4: Illustrative example of a resistive TCAM cell on a match (b, c, d) and on a mismatch (a). Inactive circuit paths are shown in light grey.**

a case where a wildcard (X) is stored in the cell, in which searching for either a 0 or a 1 will result in a match; and (d) illustrates a search for a wildcard (X). The proposed TCAM cell consists of 3 1T1R cells sharing matchline and GND contacts. At 22nm, this amounts to an area of  $27F^2$  ( $3 \times$  the 1T1R PCM cell size projected by ITRS [24]), which is  $\frac{1}{20}$  of a CMOS TCAM cell's area (Section 2.1).

## 4.2 Row Organization

A resistive TCAM row has an organization similar to that of a CMOS TCAM, where cells are cascaded to connect to a matchline in parallel. An example of the proposed row organization is shown in Figure 5. The key idea is that, on a match, each cell will connect the matchline to ground through an  $R_{HI}$  path, whereas at least one cell will provide an  $R_{LO}$  path to ground on a mismatch. Hence, the effective parallel resistance to ground (and hence, the matchline voltage) will always be higher in the case of a match.

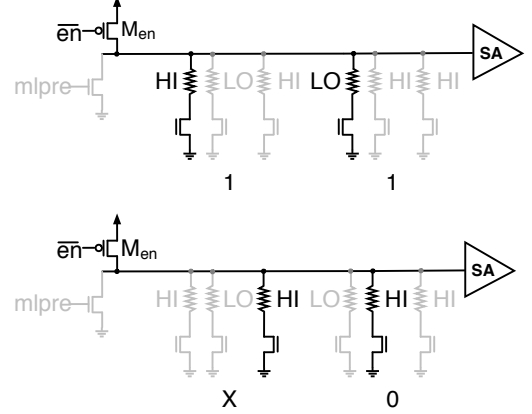


**Figure 5: Illustrative example of a resistive TCAM row.**

A precharge-low sensing scheme is employed on the matchline, where the matchline is discharged in the precharge phase and actively driven in the evaluation phase. The input of the inverter is also charged high in the precharge phase. On a match, the gate voltage of  $M_{sense}$  is higher than it is on a mismatch.  $M_{sense}$  and the keeper PMOS  $M_k$  are sized so that on a match, the gate-to-source voltage of  $M_{sense}$  is large enough to win against  $M_k$ , pulling the inverter input low and in turn driving the matchline output ( $MLso$ ) high. On a mismatch, the input to the inverter stays high and  $MLso$  outputs a zero.

**Searching.** Searching a TCAM row involves distinguishing the effective matchline-to-ground resistance on a word match, where all bits stored in a row match the corresponding search bits, from the effective resistance on a word mismatch, where one or more bits mismatch the search key. In a TCAM row, each matching bit contributes a parallel resistance of  $R_{HI} + R_{ON}$ , and each mismatching bit contributes

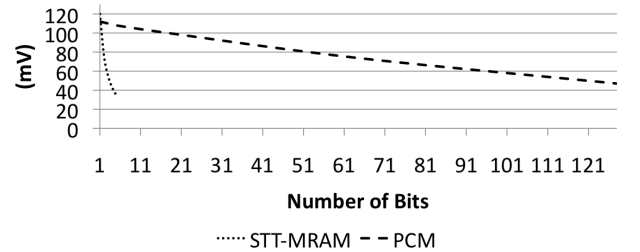
$R_{LO} + R_{ON}$  to the parallel resistance of the row, where  $R_{ON}$  represents the on-resistance of an access transistor. On a word match, the total resistance between the matchline and ground is  $R_{match} = \frac{R_{HI} + R_{ON}}{N}$ , where  $N$  is the number of bits in a word. On a worst-case mismatch ( $N - 1$  matching bits plus one mismatching bit), the matchline-to-ground resistance is  $R_{mismatch} = \frac{(R_{LO} + R_{ON})(R_{HI} + R_{ON})}{(N-1)(R_{LO} + R_{ON}) + (R_{HI} + R_{ON})}$ .



**Figure 6: Illustrative examples of a TCAM row on a word mismatch (top) and on a word match (bottom). Inactive circuit paths are shown in light grey.**

An example is shown in Figure 6: on the top is a mismatch case, which has one resistor in its low-resistance state connected to the matchline, and on the bottom is a match case, in which all resistors connected to the matchline are in their high-resistance state. The ratio between match and worst-case mismatch resistances is  $\frac{R_{match}}{R_{mismatch}} = 1 + \frac{R_{HI} - R_{LO}}{N(R_{LO} + R_{ON})}$ . This  $\frac{R_{match}}{R_{mismatch}}$  ratio must be high enough to tolerate process-voltage-temperature (PVT) variations, which affect both the transistors and the resistive devices. Therefore, a resistive memory technology with a greater difference between its high and low resistances (e.g., PCM) allows a larger number of bits to be sensed in parallel, whereas a technology with a lower  $\frac{R_{HI}}{R_{LO}}$  ratio (e.g., STT-MRAM) has a tighter constraint on the number of bits that can be searched at the same time.

At the evaluation stage, current is supplied through the enabling PMOS transistor ( $M_{en}$ ), and the voltage drop on the TCAM cells is sensed by a matchline sense amplifier.  $M_{en}$  is sized appropriately to limit the current through the resistors so that a search will not accidentally change cell states.



**Figure 7: Sensing margin as a function of key width.**

Figure 7 shows Cadence (Spectre) simulation results on PCM and STT-MRAM (details on the experimental setup

are presented in Section 7). As the number of simultaneously searched bits increases, both STT-MRAM and PCM encounter a significant drop on the voltage difference between a match and a mismatch, but the drop is much steeper in the case of STT-MRAM. Taking PVT variations into account, a smaller voltage difference (i.e., sensing margin) results in lower yield; hence, to tolerate variations and to improve yield, the number of bits that can be searched simultaneously must be limited. Given its superior sensing margin, the rest of this paper assumes a 22nm PCM technology to implement the resistive storage elements. We have confirmed that the TCAM array functions correctly in the face of extreme variations in  $R_{HI}$  (1M $\Omega$ -500K $\Omega$ ) and  $R_{LO}$  (15K $\Omega$ -30K $\Omega$ ). These resistance ranges cover 98% of all devices fabricated in a recent 45nm PCM prototype [51]; nevertheless, if the variation were to be higher, the search width could be reduced to improve the margin.

**Writing.** In order to maintain the density advantage of resistive memories in TCAM design, it is important not to unduly increase the number of transistors in a cell. Writing would normally require a write port, which would need additional access transistors and wires. Instead, we propose *bulk-sequential writes*, a new write mechanism that leverages the matchlines for writing in order to eliminate the need for extra write ports.

Figure 8 shows an example of how a TCAM row gets written under bulk-sequential writes. During a write, the sense amplifier and all search-X lines are disabled. Writing takes place in two phases. In the first phase, all resistors to be updated to  $R_{LO}$  are connected to the matchline by enabling the relevant searchlines, and are programmed by applying the appropriate switching current through the matchline. Next, resistors to be updated to  $R_{HI}$  are written in two steps. In the first step, the leftmost resistor in each cell to be updated with a 1 or X is programmed; in the second step, the middle resistor in every cell to be updated with a 0 or X is written. This two-step procedure when writing  $R_{HI}$  limits the maximum number of resistors to be programmed simultaneously to 128. Minimum-pitch local wires at 22nm can carry enough current to write these 128 bits in parallel, and a 200F<sup>2</sup> write driver supplies adequate switching current.

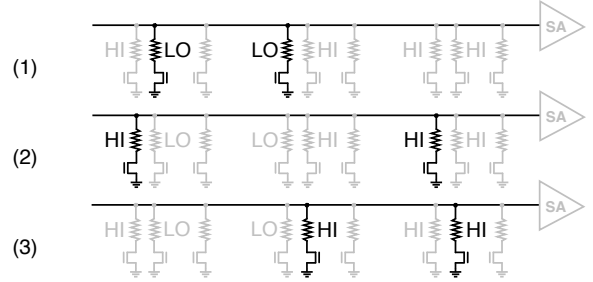
**Reading.** Since the proposed TCAM architecture is ultimately integrated on a discrete, DDR3-compatible DIMM, it can be selectively included in systems which run workloads that can leverage associative search. Nevertheless, even for such systems, it is desirable to use the TCAM address space as ordinary RAM when needed. Fortunately, this is straightforward to accomplish with the proposed TCAM array.

When the TCAM chip is configured as a RAM module, data is stored and retrieved column-wise in the TCAM array by the TCAM controller, and the searchlines  $SL$ ,  $\overline{SL}$ , and  $SX$  serve as word lines. The key observation that makes such configurability possible is that reading a cell is equivalent to searching the corresponding row with a 1 at the selected column position, and with wildcards at all other bit positions. An illustrative figure of the array and additional details on the required array-level support to enable configurability is outlined in Section 5.

**Detecting errors.** If the cells are implemented with a resistive memory technology that suffers from finite write endurance (e.g., PCM), it is necessary to verify the correctness of every write to the array. Error detection takes place in

two steps. In the first step, the enable bit (Section 5) of all rows in the array are flash-cleared, the enable bit of the newly written row is set high, and the array is searched twice—once with 1s in place of wildcards, and once with 0s—to ensure that all match cases function correctly. In the second step, each newly written  $R_{LO}$  value is checked one cell at a time (128 array searches maximum) to prevent an inadvertent wildcard from being stored to the row. Because a checker search accesses only one of 1024 arrays, the energy overhead of error detection is  $\frac{1}{8}$ th of a full search.

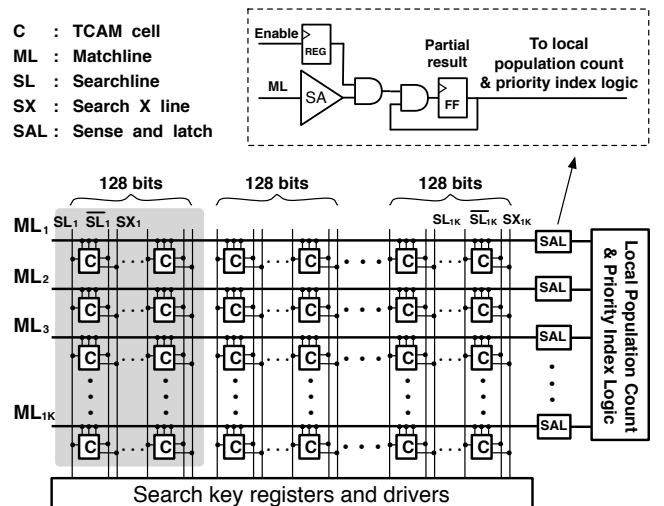
The proposed error detection technique allows for detecting permanent write failures. If, on the other hand, the technology suffers from transient faults, the TCAM controller periodically puts the TCAM DIMM into RAM mode, and refreshes the contents (similar to DRAM scrubbing).



**Figure 8:** Illustrative example of writing the three bit pattern “10X” to a row by (1) programming all low resistors, (2) programming all leftmost high resistors, and (3) programming all middle high resistors. Inactive circuit paths are shown in light grey.

## 5. ARRAY ARCHITECTURE

Figure 9 shows an example 1K $\times$ 1K TCAM array. Cells (C) are arranged in a 2D organization with horizontal matchlines (ML) and vertical searchlines (SL). Searchlines are driven by search key registers and drivers, while matchlines connect to sense amplifiers (SA). A hierarchical reduction network facilitates counting the number of matching lines and selecting one of multiple matching rows (Figure 10).



**Figure 9:** Illustrative example of a 1K $\times$ 1K TCAM array with 128b wide matchline segmentation.

An important design decision that affects area efficiency, power, speed, and reliability is the size of an array. Larger arrays improve area efficiency by amortizing the area overhead of the peripheral circuitry over more cells; however, sensing margin deteriorates with the number of bits that are searched in parallel (Figure 7). Table 2 compares absolute area and area efficiency for 1Gb TCAM chips constructed from arrays of different sizes (See Section 7 for the experimental setup). As array size increases from  $128 \times 128$  to  $1K \times 1K$ , overall chip area reduces by more than  $2\times$ .

Array Size	Total Chip Area (mm <sup>2</sup> )	Area Efficiency
1K×1K	36.00	38.98%
512×512	41.04	34.19%
256×256	51.49	27.25%
128×128	77.51	18.10%

**Table 2: Total chip area and area efficiency with different array sizes.**

Interestingly, it is possible to enjoy the high area efficiency of a large array while also delivering the large sensing margin of a smaller array through *matchline segmentation*. The key idea is to build a wide row, only a part (segment) of which can be searched simultaneously, providing additional peripheral circuitry to support iterative searching for a key larger than the size of a segment. Figure 9 shows an example  $1K \times 1K$  array, where each row is partitioned into 128b segments. To perform a full 1024b wide search across a row, the array is accessed eight times, once per each segment. On each access, searchlines connected to the relevant segment are driven with the corresponding bits from the search key, while all other searchlines are driven low. Each row is augmented with a helper flip-flop that stores the partial result of the ongoing search operation as different segments are accessed. This helper flop is initially set high; at the end of each cycle, its content is ANDed with the sense amplifier’s output, and the result overwrites the old content of the helper flop. Hence, at the end of eight cycles, the helper flop contains a logic 1 if and only if all segments within the row match. To permanently disable a row, an additional “enable” flop is added to the design.

The proposed resistive TCAM is capable of providing two different results on a search operation: (1) a population count indicating the total number of matching rows in the system, and (2) a priority index indicating the address of the matching row with the lowest index. Once the search is complete, the array’s local population count and priority logic start operating on the results (Figure 10).

**Priority index logic.** The highest priority row among all matching rows is selected in a hierarchical fashion. 32 priority encoders are used in the first level of the hierarchy, where each encoder selects one valid matchline out of 32 helper flops; also selected are the 5b addresses of the corresponding matchlines. Then, 32 valid bits with corresponding addresses enter the second level priority encoder. Finally, an 11b result, containing 1 valid bit and a 10b address, is selected and forwarded to the reduction network.

**Population count logic.** The local population count for the array is computed iteratively, accumulating the result of a 64b population count each cycle (itself computed using 16 four-input lookup tables and a carry save adder tree). It takes a total of 16 cycles to walk the array and accumulate the population count, after which the result is sent to the reduction network for further processing.

## 5.1 Reduction Network

The reduction network is a quad tree, whose leaves are the results of different arrays’ population count and priority logic. Each internal node of the quad tree takes the outputs of its four children, processes them by counting or priority encoding, and then forwards the result to its parent (Figure 10). Once the results of a search propagate to the root of the quad tree, final results are obtained and placed in an SRAM-based result store that resides with the TCAM controller. For a fixed-capacity chip, the size of an array affects the size, and hence, the latency and energy of the reduction network. Table 3 shows the energy and delay when searching for a 128b key in an eighth of a 1Gbit chip constructed from different size arrays. As array size increases, delay and energy within an array increase due to longer searchlines and matchlines; however, delay and energy consumed in the reduction network decrease because there are fewer levels of priority and population count computation in the hierarchy. Since search energy dominates total energy and reduction network delay dominates total delay, enlarging the array size results in higher energy and lower delay. Considering the area efficiency is highest with the largest array size (Table 2), the rest of this paper focuses on  $1K \times 1K$  arrays.

## 5.2 Result Store

Search throughput can be improved significantly by pipelining the reduction network, the local population count, and priority computations; however, this requires the ability to buffer the results of in-flight search operations and to retrieve them at a later time.

To facilitate such buffering, the on-DIMM TCAM controller provides a 2KB SRAM mapped to the system’s physical address space. As part of the search operation, the application sets up a number of memory-mapped control registers, one of which indicates the location where the search results should be placed. In this way, the application overlaps multiple search operations in a pipelined fashion, and retrieves the results through indexed reads from the result store.

## 6. SYSTEM ARCHITECTURE

The level of the memory hierarchy at which the TCAM is placed can have a significant impact on overall system cost, performance, and flexibility. On the one hand, an integrated solution that places the TCAM on the processor die would result in the shortest possible communication latency, at the expense of much desired modularity. On the other hand, treating the TCAM as an I/O device and placing it on the PCI or PCI Express bus would result in a modular system architecture (in which the TCAM can be selectively included), but the high latency of I/O busses would limit performance. Instead, this paper explores an intermediate solution that places the TCAM on a DDR3-compatible DIMM with an on-DIMM TCAM controller. The resulting design requires no changes to existing processors, memory controllers, or motherboards, while providing modularity to enable selective inclusion of TCAM in systems that benefit from it. Moreover, with the ability to configure the TCAM as a regular RAM, users can also leverage the TCAM DIMM as a byte-addressable, persistent memory module.

### 6.1 Processor Interface

Software is given access to a TCAM accelerator by mapping the TCAM address range to the system’s physical ad-

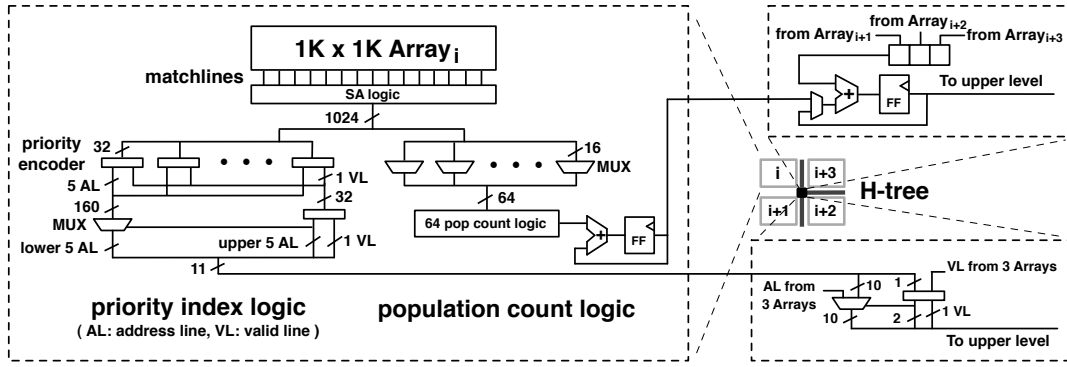


Figure 10: Architecture of the population count logic, priority index logic, and the reduction network.

Array Size	Energy of Search (nJ)	Energy with Priority Index (nJ)	Energy with Pop. Count (nJ)	Delay of Local Search (ns)	Delay with Priority Index (ns)	Delay with Pop. Count (ns)
1K×1K	244.97	248.59	249.64	2.50	21.57	60.28
512×512	176.16	180.22	181.30	1.75	21.91	61.48
256×256	96.21	100.85	102.36	1.00	23.78	64.63
128×128	55.57	61.78	63.33	0.50	28.29	67.67

Table 3: Energy and delay comparison of different array sizes.

dress space. In addition, the on-DIMM TCAM controller maintains a 2KB RAM array (Section 5.2), to implement memory-mapped control registers, search key buffers, and the result store. All accesses issued to TCAM are uncacheable, and subject to strong ordering, which prevents all of the requests to the memory-mapped TCAM address range from being reordered. (This can be accomplished, for example, by marking the corresponding physical pages strong-uncacheable in the page attribute table of any Intel X86 processor since the Intel 386 [22].)

Communication between the processor and TCAM takes place in four ways:

1. **Device configuration.** The processor configures the TCAM system by writing to memory-mapped control registers. Configurable system attributes include key length, required result type (population count, priority index, or both), and whether the module should operate in RAM or TCAM mode.

2. **Content update.** The processor stores data into TCAM control registers, after which the TCAM controller updates the TCAM array.

3. **Search.** The processor stores the query key into the memory-mapped TCAM key buffer, which resides with the TCAM controller. As soon as the last word of the key is received, the TCAM controller initiates a search operation whose results are written to the appropriate words within the memory-mapped result store.

4. **Read.** After a search, the processor loads the outcome from the result store.

## 6.2 TCAM Controller

To plug into an existing DIMM socket with no modifications to the memory controller, the TCAM’s memory bus interface should be fully compatible with DDRx and its timing constraints. Here we discuss the TCAM interface in the context of a modern DDR3 protocol and an FR-FCFS [50] based memory controller, but the memory controller’s scheduling policy is orthogonal to the TCAM DIMM, since a DDR3 compatible TCAM DIMM is compatible with any DDR3-compatible memory controller by definition. In DDR3, a

complete read (write) transaction includes a row activation (activate a row in a bank and move data to a row buffer), a column read (write) involving data transfer, and a precharge (write data back to the row and precharge the bitlines). When consecutive requests hit an open row, no activate or precharge operation is needed. The minimum number of cycles between a write and a consecutive read in DDR3 is less than the time needed to perform a search; hence, without additional support, it is possible for the memory controller to issue a read from the result store before the corresponding search operation is complete.

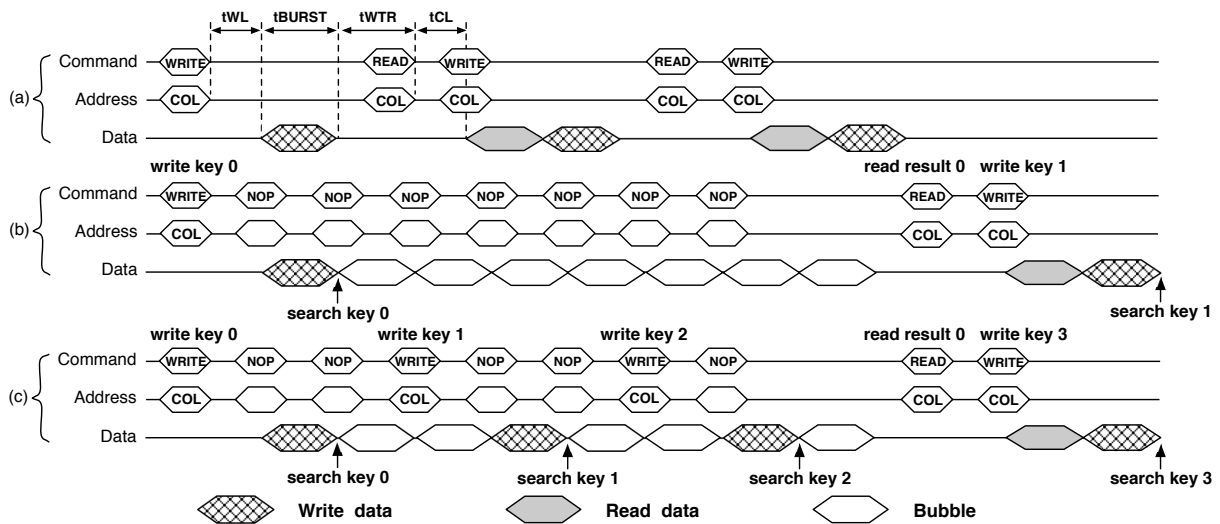
To avert this problem, the software library used for accessing TCAM inserts an appropriate number of dummy writes between the search and the subsequent read from the result store, generating the required number of idle cycles on the DDR3 bus for the search operation to complete. (Recall that all requests to TCAM control registers are processed in-order, since the corresponding physical pages are marked strong-uncacheable.) This is accomplished by issuing writes to a known location in physical memory, which allows the TCAM controller to detect the dummy write and silently drop it upon receipt.

Although injecting dummy writes (i.e., bubbles) into the DDR3 bus guarantees correctness, it also lowers the utilization of the bus and wastes precious bandwidth. To improve search throughput, the TCAM chip is pipelined (Section 5) so that shortly after an array search is complete, the next search is allowed to begin, largely overlapping the array search operation of a later request with the reduction network and result store update of an earlier request. Consequently, when performing multiple search operations, it becomes possible to pipeline search and read operations, thereby reducing the number of required dummy writes. Figure 11 shows an example of pipelined search with a 128b key, where pipelining improves performance by almost 3×.

## 6.3 Software Support

A user-level library serves as the API to the programmer, and implements four functions.

**Create.** This function is called at the beginning of an ap-



**Figure 11: A consecutive write-read-write command sequence in DDR3 (a); a consecutive search-read command sequence in TCAM without pipelining (b); and a consecutive search-read command sequence in TCAM with pipelining (c). Dummy writes are labeled “NOP”.**

plication to map a virtual address range to a portion of the TCAM’s physical address space. Physical pages in TCAM are marked strong-uncacheable (Section 6.1), and the OS is signaled to allocate content addressable physical pages by leveraging one of the reserved flags of the mmap system call [55]. Upon successful termination, create returns a pointer to the newly allocated TCAM space, which helps distinguish independent TCAM regions.

**Destroy.** This function releases the allocated TCAM space.

**Store.** This function updates the TCAM array. The function accepts a mask indicating the bit positions that should be set to wildcards (X), as well as the data to be stored in the remaining positions. Internally, the function communicates to the TCAM controller by writing two memory-mapped control registers, and the TCAM controller writes the data into the TCAM array. To hide the long latency of writing to TCAM, the library function distributes consecutive writes to different TCAM banks. In the case of a bank conflict, an appropriate number of dummy writes are inserted by the library to cover the busy period. (In the evaluated benchmarks, the database is stored in a sequential order; thus, no bank conflicts were observed.)

**Search.** This function performs a search operation in three steps, by (1) storing the search key into the memory-mapped TCAM query key register, (2) issuing enough dummy writes to pad the waiting period between the search operation and the subsequent read from the result store, and (3) reading the results from the memory-mapped result store. Two different flavors of the search function are provided: a “single” search call performs a search with a single key, whereas a “batch” search call searches for multiple independent keys in a pipelined fashion to improve throughput (keys are stored in memory, and are passed to the batch search function through a pointer). A pointer to the TCAM region to be searched is given as an argument to the search call, and the library ensures that only results from the specified region are returned by storing a (searchable) unique region ID with each TCAM row.

### 6.3.1 Multiprogramming

Supporting multiprogramming with the proposed TCAM system requires mechanisms to prevent the search results of one application from being affected by data that belongs to another application. Hence, although conventional virtual memory solves protection problems in the case of TCAM reads and writes, search operations need additional support.

To enable multiprogramming, each process is assigned an address space identifier (ASID); when the process allocates space in the TCAM address range, the OS records the ASID in a memory-mapped control register within the same physical page that contains the key buffer and the result store for that process. On a write, the TCAM controller stores the corresponding ASID along with each word in a given row. On a search, after the process writes the key buffer, the TCAM controller appends the ASID to the search key; as a result, all rows that belong to other processes result in mismatches, and do not affect search results.

### 6.3.2 Handling Misfits

Although the OS naturally supports virtualization of the physical address space, searching a data structure larger than the allocated TCAM space requires extra care. Specifically, two types of misfit are possible: (1) a horizontal misfit, in which the key is larger than the width of a TCAM array (e.g., 2Kb key for a 1K×1K array), and (2) a vertical misfit, in which the number of rows to be searched exceeds the capacity of the allocated TCAM region.

To solve the horizontal misfit problem, the word is broken into a series of 1Kb-wide subwords, each of which is stored in a consecutive row of an array (if the final subword is narrower than 1Kb, it is padded with enough wildcards (X) to cover the row). On a search, the TCAM controller partitions the key in units of 1Kb subwords, and searches the array one row after another. At the end of a search operation, the content of the helper flip-flop that connects to a row is shifted into the helper flip-flop of the next row, and is ANDed with the outcome of the next row’s search operation. Hence, the helper flop of the final row contains the complete search result.

<b>Core</b> <b>Functional units</b> <b>IQ, LSQ, ROB size</b> <b>Physical registers</b> <b>Branch predictor</b> <b>IL1 cache (per core)</b> <b>DL1 cache (per core)</b> <b>L2 cache (shared)</b> <b>Memory controller</b> <b>DRAM Subsystem</b> <b>Timing (DRAM cycles) [41]</b>	8 cores, 4.0 GHz, 4-issue Int/FP/Ld/St/Br units 2/2/2/2/2, Int/FP Mult 1/1 IssueQ 32, LoadQ/StoreQ 24/24, ROB 96 Int/FP 96/96 Hybrid, local/global/meta 2K/2K/8K, 512-entry direct-mapped BTB, 32-entry RAS 32KB, direct-mapped, 32B block, 2-cycle hit time 32KB, 4-way, LRU, 32B block, hit/miss delay 3/3, MESI protocol 4MB, 8-way, LRU, 64B block, 24-cycle hit time 4-channel, 64-entry queue, FR-FCFS, page interleaving DDR3-1066 MHz tRCD: 7, tCL: 7, tWL: 6, tCCD: 4, tWTR: 4, tWR: 8, tRTP: 4, tRP: 7, tRRD: 4, tRAS: 20, tRC: 27, tBURST: 4, tFAW: 20
---	--

**Table 4: Core parameters.**

Benchmarks	Input	Description	TCAM Content	TCAM Search Keys
<b>Apriori</b> [5]	95,554 transactions 1000 items, 2.7MB	association rule mining	transaction database	candidate itemsets
<b>BitCount</b> [20]	75,000 array size 64 bits per element	non-recursive bit count by bytes	integer data array	N-bit vectors with a single one and N-1 wildcards
<b>Histogram</b> [60]	34,843,392 pixels 104MB	pixel value distribution in bitmap image	pixel values	distinct RGB values
<b>ReverseIndex</b>	78,355 files 14,025 folders, 1.01GB	extract links and compile an index from links to files	URLs	URLs
<b>StringMatch</b>	50MB non-encrypted file and non-encrypted keys	string search on encrypted file	encrypted strings	encrypted keys
<b>Vortex</b> [27]	3 inter-related databases, 200MB	insert, delete and lookup operations	unique item ID and valid bit	unique item ID
<b>WordCount</b>	10MB text file	count frequencies of distinct words	English words	distinct keywords

**Table 5: Evaluated applications.**

On a vertical misfit, the search is staged over multiple local search operations, in which the missing pages are transferred from DRAM to TCAM. This process is transparent to the user and is handled by the TCAM library. Since data transfer between TCAM and DRAM can be expensive, the search is optimized for minimizing data movement. For example, if the search region is larger than the capacity of the TCAM, the library function partitions the search region to fit the TCAM space, and does batch search (Section 6.3) in each subregion. The final results are calculated by merging all of the partial results. This is obviously cheaper than doing each single search in the entire region, which would generate constant data movement.

## 7. EXPERIMENTAL SETUP

We evaluate the proposed TCAM accelerator on seven data-intensive applications from existing benchmark suites, running on a model of an eight-core processor with a DDR3-1066 memory system.

**Circuits.** We model the TCAM array and sensing circuitry using BSIM-4 predictive technology models (PTM) [61] of NMOS and PMOS transistors at 22nm, and conduct circuit simulations using Cadence (Spectre). Resistances and capacitances on searchlines, matchlines, and the H-tree are modeled based on interconnect projections from ITRS [24]. All peripheral circuits (decoders, population count logic, priority index logic, and reduction network nodes) are synthesized using Cadence Encounter RTL Compiler [1] with FreePDK [2] at 45nm, and results are scaled to 22nm (relevant parameters are shown in Table 7). Resistive memory parameters based on ITRS projections are listed in Table 6.

**Architecture.** We use a heavily modified version of the SESC simulator [49] to model an eight-core system with a 1GB TCAM DIMM. Energy results for the cores and the DRAM subsystem are evaluated using MCPAT [32]. Details of the experiments are shown in Table 4.

Technology	$R_{HI}$	$R_{LO}$	Cell Size	Write Current
PCM	1M $\Omega$	15K $\Omega$	8.4F <sup>2</sup>	48 $\mu$ A
STT-MRAM	12.5K $\Omega$	5K $\Omega$	8F <sup>2</sup>	35 $\mu$ A

**Table 6: Resistive memory parameters [24]**

Technology	Voltage	FO4 Delay
45nm	1.1 V	20.25ps
22nm	0.83 V	11.75ps

**Table 7: Technology parameters [24, 61]**

**Applications.** We modify seven data-intensive applications from MiBench [20], NU-MineBench [43], SPEC2000 [27], and Phoenix [60] benchmark suites to take advantage of the proposed TCAM system. Table 5 presents a summary of each benchmark, as well as a description of how it is adapted to TCAM. Aside from BitCount and Vortex—which are sequential applications—all baselines are simulated with eight threads. TCAM-accelerated versions of all codes use one thread.

We found many other application domains amenable to TCAM acceleration, albeit with no readily available, representative benchmarks in most cases. These applications include sorting and searching [52], similarity search [53], subset and superset queries [18], decision tree training [25], search engines [13], spell checking [20], sequential pattern mining [31], packet classification [29], IP routing [38], parametric curve extraction [40], Hough transformation [42], Huffman encoding [35], Lempel-Ziv compression [58], image coding [45], and logic minimization [6]. Evaluation of these and other applications is left for future work.

## 8. EVALUATION

We first evaluate the contribution of different hardware structures to search energy, search delay, and overall area. We then present performance and energy improvements of a single-threaded, TCAM-accelerated version of each application over a baseline parallel execution with eight threads.

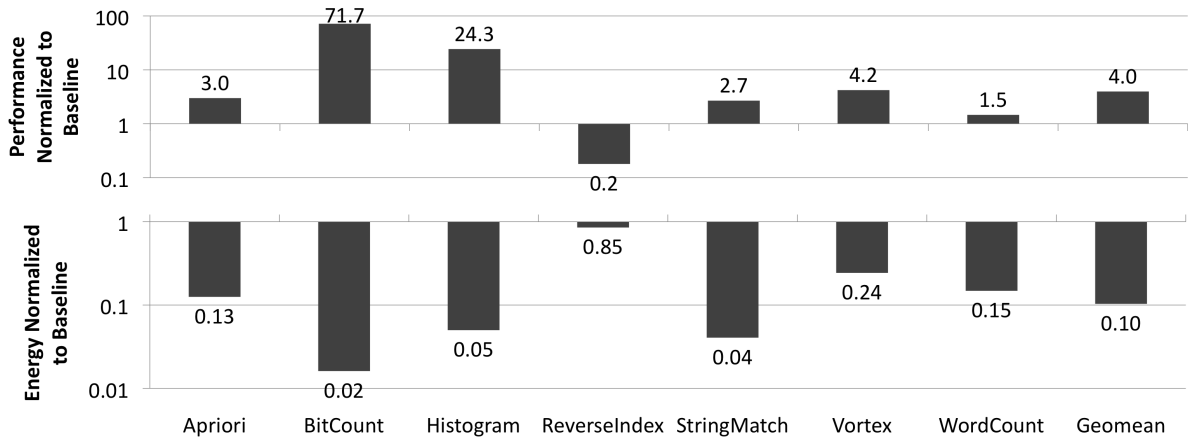


Figure 12: TCAM performance (top) and energy (bottom) normalized to the RAM-based baseline. All baseline applications except BitCount and Vortex (sequential codes) execute with eight threads.

## 8.1 TCAM Delay, Energy, and Area: Where are the Bottlenecks?

Figure 13 shows the breakdown of search delay, search energy, and die area over the reduction network, local population count/priority logic, and array search operations for a 1Gb TCAM chip. Delay and energy results correspond to a chip-wide search with a 128b key; in each case, results are reported for both population count and priority index configurations of the system.

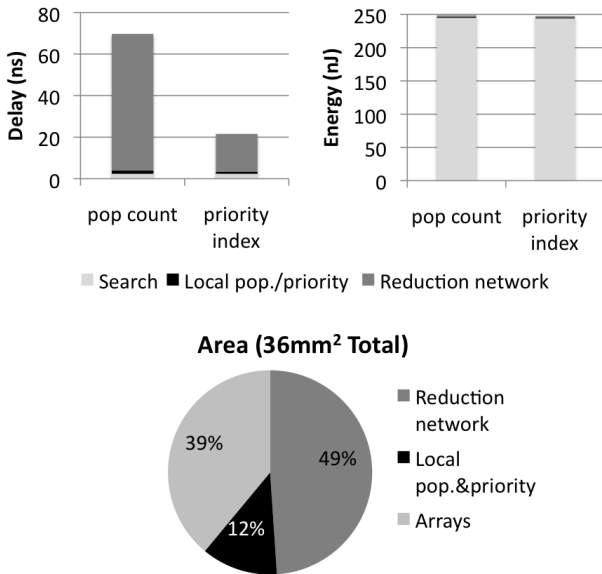


Figure 13: Delay, energy, and area breakdown.

Overall delay is dominated by the delay of the reduction network in both priority index and population count configurations. This is because each node of the reduction network, which is implemented as a quad tree, depends on its children. To improve area efficiency, a reduction network node implements population count using an accumulator that iteratively sums the counts sent from four lower-level nodes;

this adds three clock cycles per network node. As a result, obtaining the global population count takes over  $3\times$  longer than the priority index.

Although searching the array contributes a small fraction of the overall delay, it almost entirely dominates energy consumption. This is because all matchlines are activated with every search operation. Since the array search—which is needed for priority and population count configurations—dominates energy, both configurations of the system are equally energy-hungry.

The area efficiency of TCAM is 39%, which is competitive but less than the projected DRAM area efficiency (50% [24]) at 22nm. Nearly half the area is devoted to the reduction network, which is responsible for distributing the 128b key to all arrays, and aggregating results.

## 8.2 System Performance and Energy

Figure 12 shows performance and energy evaluations on seven data-intensive benchmarks. The TCAM accelerator achieves significant performance improvement over the baseline multicore system on six of the seven benchmarks (except ReverseIndex), with an average speedup of  $4\times$ . Highest speedups are obtained on BitCount ( $71.7\times$ ) and Histogram ( $24.3\times$ ), where over 99% of the baseline runtime is in search and comparisons—operations amenable to TCAM acceleration. ReverseIndex performs considerable preprocessing, whereby a set of HTML files are parsed to extract links to others files. This portion of the application cannot leverage the TCAM accelerator; we have measured the maximum theoretical (i.e., Amdahl’s Law) speedup on a single-threaded version of this benchmark, finding it to be only  $1.06\times$ . The TCAM accelerator achieves a  $1.04\times$  speedup over this sequential version, but is nearly  $5\times$  slower than the parallel version of the benchmark which uses eight threads (recall that the TCAM-enabled version of the benchmarks are single-threaded).

Figure 12 also shows overall system energy with the TCAM accelerator, normalized to the eight-core baseline. Six of the seven benchmarks achieve significant energy reduction, with an overall average of  $10\times$ . The efficiency gains are due to two factors. First, TCAM eliminates off-chip data movement and instruction processing overheads by processing data di-

rectly on the chip; second, the faster execution time leads to lower leakage energy. As one would expect, energy savings on ReverseIndex are lower than other applications due to the limited applicability of TCAM in this benchmark.

## 9. RELATED WORK

This paper builds upon existing work in resistive memories, TCAM, and processing in memory.

**Resistive Memories.** Recent research on TCAM is seeking opportunities to utilize the low leakage power and high scalability of resistive memories to improve power efficiency. PCM-based 2T2R (two pairs of transistors and resistors in parallel) TCAM cells have been demonstrated in prior work [14, 48]. In a patent [3], a 2T2R PCM TCAM is proposed with two additional bit lines. Matsunaga et al. [36] propose a bit-serial 2T2R TCAM design using MTJ devices. Xu et al. [59] propose an STT-MRAM CAM design with high sensing and search speed. Matsunaga et al. [37] later present a 6T2R cell, which adds assist circuitry to each cell to increase the sensing margin and search width. Alibart et al. [7] propose a TCAM cell with a pair of memristors and demonstrate how layout can be done in such a way that it takes full advantage of the potential for memristor densities. Eshraghian et al. [17] evaluate four types of memristor-based CAM designs. Existing resistive TCAM proposals focus on circuit design, whereas this paper explores a 3T3R TCAM cell, its array organization, and the system architecture for an on-DIMM TCAM accelerator. Moreover, prior work does not explore bulk-sequential writes—a parallel writing mechanism with no area overhead added to the cell (Section 4.2). Other proposals for improving TCAM power efficiency include a stacked 3D-TCAM design [34] and a scheme for virtualizing TCAM [10].

**TCAM.** TCAMs are commonly used in networking [46, 29]. Recent work applies TCAM to a wider range of applications. Panigrahy et al. use TCAM in sorting and searching [52], Goel and Gupta solve set query problems [18], Shinde et al. study similarity search and locality sensitive hashing [53]. Hashimi and Lipasti [21] propose a TCAM accelerator as a functional unit. Other applications of TCAM include decision tree training [25], search engines [13], spell checking [20], sequential pattern mining [31], packet classification [29], IP routing [46], parametric curve extraction [40], Hough transformation [42], Huffman encoding [35], Lempel-Ziv compression [58], image coding [45], and logic minimization [6].

**Processing in memory.** Processing in memory has been proposed to reduce memory bandwidth demand in prior work. Elliott et al. [15] build a computational RAM, which adds processing elements (PE) of a SIMD machine directly to the sense amplifiers of a 4Mb DRAM chip. Gokhale et al. [19] propose the processor-in-memory (PIM) chip. PIM can be configured in conventional memory mode or in SIMD mode to speedup massively parallel SIMD applications. Oskin et al. [44] propose Active Pages, which adds reconfigurable logic elements to each DRAM subarray to process data. None of the previous work has proposed the use of a resistive TCAM chip on a DRAM channel to accelerate search operations and to reduce memory bandwidth demand.

## 10. CONCLUSIONS

We have presented a new resistive TCAM cell and array architecture that deliver a 20× density improvement over

existing CMOS-based solutions. We have explored a modular memory system that places resistive TCAM chips on a DDR3-compatible DIMM, and accesses the DIMM through a user-level software library with zero modifications to the processor or the motherboard. By tightly integrating TCAM with conventional virtual memory, we have observed an average performance improvement of 4× and an average energy reduction of 10× on a set of data-intensive applications that can exploit associative search capability. We believe this work is part of a larger trend toward leveraging resistive memory technologies in designing memory systems with qualitatively new capabilities. With technology scaling, power and bandwidth restrictions will become more stringent while resistive memories move to mainstream, making such systems increasingly appealing and viable.

## 11. ACKNOWLEDGMENTS

The authors would like to thank Eby Friedman, Seung Kang, and anonymous reviewers for useful feedback.

## 12. REFERENCES

- [1] Encounter RTL compiler. [http://www.cadence.com/products/ld/rtl\\_compiler/](http://www.cadence.com/products/ld/rtl_compiler/).
- [2] Free PDK 45nm open-access based PDK for the 45nm technology node. <http://www.eda.ncsu.edu/wiki/FreePDK>.
- [3] Phase-change based TCAM architectures and cell. <http://ip.com/IPC0M/000187746>.
- [4] A. Driskill-Smith (Grandis, Inc). Latest advances and future prospects of STT-MRAM. In *Non-Volatile Memories Workshop*, University of California, San Diego, Apr. 2010.
- [5] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of the 20th Very Large Databases Conference*, Santiago de Chile, Chile, Sept. 1994.
- [6] S. Ahmad and R. Mahapatra. An efficient approach to on-chip logic minimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 15(9):1040–1050, Sept. 2007.
- [7] F. Alibart, T. Sherwood, and D. Strukov. Hybrid CMOS/nanodevice circuits for high throughput pattern matching applications. In *NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, June 2011.
- [8] I. Arsovski, T. Chandler, and A. Sheikholeslami. A ternary content-addressable memory TCAM based on 4T static storage and including a current-race sensing scheme. *Journal of Solid-State Circuits*, 38(1):155–158, Jan. 2003.
- [9] K. E. Batcher. Staran parallel processor system hardware. In *Proceedings of the national computer conference and exposition*, AFIPS, 1974.
- [10] S. Bhattacharya and K. Gopinath. Virtually cool ternary content addressable memory. In *Proceedings of the 13th USENIX conference on hot topics in OS*, May 2011.
- [11] Y.-K. Chang, M.-L. Tsai, and C.-C. Su. Improved TCAM-based pre-filtering for network intrusion detection systems. In *22nd International Conference on Advanced Information Networking and Applications*, Mar. 2008.
- [12] H. Chung et al. A 58nm 1.8V 1Gb PRAM with 6.4MB/s program BW. In *IEEE International Solid-State Circuits Conference Digest of Technical Papers*, Feb. 2011.
- [13] O. de Kretser and A. Moffat. Needles and Haystacks: a search engine for personal information collections. In *23rd Australasian Computer Science Conference*, 2000.
- [14] N. Derharcobian and C. N. Murphy. Phase-change memory (PCM) based universal content-addressable memory (CAM) configured as binary/ternary CAM. United States Patent US 7,675,765 B2, Agate Logic, Inc., Mar. 2010.
- [15] D. Elliott, W. Snelgrove, and M. Stumm. Computational Ram: A memory-SIMD hybrid and its application to DSP. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 30.6.1–30.6.4, May 1992.
- [16] Micron Technology, Inc. <http://www.micron.com/products/pcm>.
- [17] K. Eshraghian et al. Memristor MOS content addressable memory (MCAM): Hybrid architecture for future high performance search engines. *IEEE Transactions on Very Large*

- Scale Integration (VLSI) Systems*, 19(8):1407–1417, Aug. 2011.
- [18] A. Goel and P. Gupta. Small subset queries and bloom filters using ternary associative memories, with applications. In *Proceedings of the ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS, 2010.
- [19] M. Gokhale and *et al.* Processing in memory: the terasys massively parallel PIM array. *Computer*, 28(4):23–31, Apr. 1995.
- [20] M. R. Guthaus *et al.* MiBench: A free, commercially representative embedded benchmark suite. In *IEEE 4th Annual Workshop on Workload Characterization*, Austin, TX, Dec. 2001.
- [21] A. Hashmi and M. Lipasti. Accelerating search and recognition with a TCAM functional unit. In *IEEE International Conference on Computer Design*, Oct. 2008.
- [22] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3A: System Programming Guide, Part 1*, May 2011. <http://www.intel.com/products/processor/manuals>.
- [23] E. Ipek *et al.* Dynamically replicated memory: Building reliable systems from nanoscale resistive memories. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar. 2010.
- [24] ITRS. *International Technology Roadmap for Semiconductors: 2010 Update*. <http://www.itrs.net/links/2010itrs/home2010.htm>.
- [25] M. Joshi, G. Karypis, and V. Kumar. ScalParC: A new scalable and efficient parallel classification algorithm for mining large datasets. In *Proceedings of the International Parallel Processing Symposium*, pages 573–579, 1998.
- [26] G. Kasai *et al.* 200MHz/200MSPS 3.2W at 1.5V V<sub>dd</sub>, 9.4Mbits ternary CAM with new charge injection match detect circuits and bank selection scheme. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pages 387–390, Sept. 2003.
- [27] A. KleinOowski and D. J. Lilja. MinneSPEC: A new SPEC benchmark workload for simulation-based computer architecture research. *Computer Architecture Letters*, 1, 2002.
- [28] T. Kohonen. *Content-Addressable Memories*. Springer-Verlag, 1980.
- [29] K. Lakshminarayanan, A. Rangarajan, and S. Venkatachary. Algorithms for advanced packet classification with ternary CAMs. In *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM, pages 193–204, 2005.
- [30] B. Lee *et al.* Architecting phase-change memory as a scalable DRAM alternative. In *International Symposium on Computer Architecture*, Austin, TX, June 2009.
- [31] M. Leleu, C. Rigotti, and J. François Boulicaut. GO-SPADE: Mining sequential patterns over datasets with consecutive repetitions. In *Proceedings of International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM)*, pages 293–306, 2003.
- [32] S. Li *et al.* McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *International Symposium on Computer Architecture*, pages 469–480, 2009.
- [33] C. J. Lin *et al.* 45nm low power CMOS logic compatible embedded STT-MRAM utilizing a reverse-connection 1T/1MTJ cell. In *IEEE Electron Devices Meeting*, Dec. 2009.
- [34] M. Lin, J. Luo, and Y. Ma. A low-power monolithically stacked 3D-TCAM. In *IEEE International Symposium on Circuits and Systems*, pages 3318–3321, May 2008.
- [35] L.-Y. Liu *et al.* CAM-based VLSI architectures for dynamic huffman coding. In *Consumer Electronics, Digest of Technical Papers, IEEE International Conference on*, Jun. 1994.
- [36] S. Matsunaga *et al.* Standby-power-free compact ternary content-addressable memory cell chip using magnetic tunnel junction devices. *Applied Physics Express*, 2(2):23004, 2009.
- [37] S. Matsunaga *et al.* Fully parallel 6T-2MTJ nonvolatile TCAM with single-transistor-based self match-line discharge control. In *2011 Symposium on VLSI Circuits (VLSIC)*, June 2011.
- [38] A. J. McAuley and P. Francis. Fast routing table lookup using CAMs. In *Proceedings, Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies*, 1993.
- [39] R. McGeer and P. Yalagandula. Minimizing rulesets for TCAM implementation. In *Proceedings, The 28th Conference on Computer Communications*, Apr. 2009.
- [40] M. Meribout, T. Ogura, and M. Nakanishi. On using the CAM concept for parametric curve extraction. *IEEE Transactions on Image Processing*, 9(12):2126–2130, Dec. 2000.
- [41] Micron Technology, Inc., <http://www.micron.com//get-document/?documentId=425>. 1Gb DDR3 SDRAM, 2006.
- [42] M. Nakanishi and T. Ogura. A real-time CAM-based hough transform algorithm and its performance evaluation. In *Proceedings of the 13th International Conference on Pattern Recognition*, Aug. 1996.
- [43] R. Narayanan *et al.* Minebench: A benchmark suite for data mining workloads. In *IEEE International Symposium on Workload Characterization*, Oct. 2006.
- [44] M. Oskin, F. Chong, and T. Sherwood. Active pages: a computation model for intelligent memory. In *The 25th Annual International Symposium on Computer Architecture*, 1998.
- [45] S. Panchanathan and M. Goldberg. A content-addressable memory architecture for image coding using vector quantization. *IEEE Transactions on Signal Processing*, 39(9):2066–2078, Sept. 1991.
- [46] T.-B. Pei and C. Zukowski. VLSI implementation of routing tables: tries and CAMs. In *Proceedings, Tenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Apr. 1991.
- [47] M. K. Qureshi *et al.* Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, 2009.
- [48] B. Rajendran *et al.* Demonstration of CAM and TCAM using phase change devices. In *3rd IEEE International Memory Workshop*, May 2011.
- [49] J. Renau *et al.* SESC simulator, Jan. 2005. <http://sesc.sourceforge.net>.
- [50] S. Rixner *et al.* Memory access scheduling. In *Proceedings of the 27th annual international symposium on Computer architecture*, May 2000.
- [51] G. Servalli. A 45nm generation phase change memory technology. In *2009 IEEE International Electron Devices Meeting*.
- [52] S. Sharma and R. Panigrahy. Sorting and searching using ternary CAMs. In *Proceedings, 10th Symposium on High Performance Interconnects*, Dec. 2002.
- [53] R. Shiinde *et al.* Similarity search and locality sensitive hashing using ternary content addressable memories. In *Proceedings of the 2010 international conference on Management of data*, Jun. 2010.
- [54] J. M. Slaughter. Materials for magnetoresistive random access memory. *Annual Review of Materials Research*, 39:277–296, 2009.
- [55] R. W. Stevens and S. A. Rago. *Advanced Programming in the UNIX(R) Environment (2nd Edition)*. Addison-Wesley Professional, 2005.
- [56] K. Tsuchida *et al.* A 64Mb MRAM with clamped-reference and adequate-reference schemes. In *Proceedings of the IEEE International Solid-State Circuits Conference*, 2010.
- [57] J. Wade and C. Sodini. Dynamic cross-coupled bitline content addressable memory cell for high density arrays. In *International Electron Devices Meeting*, 1985.
- [58] B. Wei *et al.* A single chip lempel-ziv data compressor. In *IEEE International Symposium on Circuits and Systems*, 1993.
- [59] W. Xu, T. Zhang, and Y. Chen. Design of spin-torque transfer magnetoresistive RAM and CAM/TCAM with high sensing and search speed. *IEEE Transactions on Very Large Scale Integration Systems*, 18(1):66–74, Jan. 2010.
- [60] R. M. Yoo, A. Romano, and C. Kozyrakis. Phoenix rebirth: Scalable MapReduce on a large-scale shared-memory system. In *Proceedings of the 2009 IEEE International Symposium on Workload Characterization*, 2009.
- [61] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm design exploration. In *International Symposium on Quality Electronic Design*, 2006.
- [62] W. Zhao, C. Chappert, and P. Mazoyer. Spin transfer torque (STT) MRAM-based runtime reconfiguration FPGA circuit. *ACM Transactions on Embedded Computing Systems*, 9(2):14:1–14:16, Oct. 2009.