Resource-Efficient SRAM-based Ternary Content Addressable Memory

Ali Ahmed, Kyungbae Park, and Sanghyeon Baeg

***Abstract*— Static random access memory (SRAM)-based ternary content addressable memory (TCAM) offers TCAM functionality by emulating it with SRAM. However, this emulation suffers from reduced memory efficiency while mapping the TCAM table on SRAM units. This is due to the limited capacity of the physical addresses in the SRAM unit. This brief offers a novel memory architecture called a resource-efficient SRAM-based TCAM (REST), which emulates TCAM functionality using optimal resources. The SRAM unit is divided into multiple virtual blocks to store the address information presented in the TCAM table. This approach virtually increases the overall address space of the SRAM unit, mapping a greater portion of the TCAM table in SRAM and increasing the overall emulated TCAM bits/SRAM at the cost of reduced throughput. A 72 × 28-bit REST consumes only one 36-kbit SRAM and a few distributed RAMs via implementation on a Xilinx Kintex-7 field-programmable gate array. It uses only 3.5% of the memory resources compared with a conventional SRAM-based TCAM (hybrid-partitioned TCAM).**

***Index Terms*— Field-programmable gate array (FPGA), memory architecture, memory–throughput tradeoff, SRAM-based ternary content addressable memory (TCAM), static random access memory (SRAM).**

I. Introduction

THE main task of a ternary content addressable memory (TCAM) is to search contents stored in its memory. This is done by accessing stored data by content rather than by address; this is unlike static random access memory (SRAM), which accesses content using the address. TCAM compares the search query with the contents that are preloaded in the entire memory array simultaneously and generates the result. TCAM can be used in one of the three states, as follows: two binary states (0 and 1) and a do not care state (X). Binary states are stored in data cells, and the do not care state is stored in mask cells [1].

A typical TCAM has the advantage of fast searching over SRAM-based searching solutions, but it also has drawbacks. A TCAM cell uses more transistors than an SRAM cell. Hence, it has high production cost per bit of memory storage and exhibits less storage efficiency than SRAM devices of comparable bit density and access time. This drawback is mainly because of emulating the do not care state, which requires additional hardware memory resources in terms of bits used for the emulation [2]. To overcome the drawbacks in a typical TCAM, such as relatively high energy consumption [4], complex memory structure, limited storage density, low scalability, heavy licensing, and royalty costs by some TCAM vendors [3], a research area called SRAM-based TCAM has been proposed in the latest research [5]–[7], [13].

Manuscript received May 26, 2016; revised September 26, 2016; accepted November 25, 2016. This work was supported in part by the Ministry of Trade, Industry & Energy under Grant 10052875 and in part by the Korea Semiconductor Research Consortium support program for the development of the future semiconductor device.

The authors are with the Department of Electronics and Communication Engineering, Hanyang University, Gyeonggi-do, South Korea (e-mail: aliahmed@hanyang.ac.kr; pkb1999@hanyang.ac.kr; bau@hanyang.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2016.2636294

The SRAM-based TCAM memory architectures generally require SRAMs to map TCAM bits [5]–[7], [13]. The search operation requires a fixed length binary query string (QS) as the address to access the SRAM. It generates the vector that contains information about matched address locations. The logical 1s and 0s in the vector are treated as matched and mismatched address locations, respectively. Later, a priority encoder is used to prioritize the locations of the vector when multiple matching bits are present in the vector.

SRAM has an inherent hardware structure suited to store data, which are accessed in a sequential manner. The SRAM-bit utilization in the SRAM-based TCAM can be negatively affected during the mapping of TCAM bits that are simultaneously accessed. The decrease in the memory utilization efficiency of SRAM can be even more significant for large TCAM emulations. This brief focuses on efficiently using resources, such as memory and throughput, and proposes a resource-efficient SRAM-based TCAM (REST) emulation architecture to make use of memory–throughput tradeoff in SRAM-based TCAM.

The rest of the brief is organized as follows. In Section II, the previous works on SRAM-based TCAM and the motivations guiding this brief are explained. Section III discusses the proposed REST memory architecture. The use of virtual blocks (VBs) in the REST memory architecture is discussed in Section IV. Section V analyzes and compares various emulated TCAMs in terms of energy efficiency, memory efficiency, and the latency for both search and store operations. Section VI shows the hardware implementation for the field-programmable gate array (FPGA). Section VII concludes this brief.

II. Previous Works and Motivations

Various SRAM-based TCAMs have been implemented on the FPGA platform. A scalable TCAM is implemented on the Xilinx FPGA in [6]. It uses Xilinx primitive block RAMs (BRAMs) and distributed RAMs to emulate classical TCAM. In this brief, all the SRAMs are activated during a search operation, which increases the overall power consumption. To avoid the increase in power consumption in [6], a hierarchical low power search operation is proposed in [7], which activates RAMs hierarchically based on the match condition found in previous RAM blocks. A similar FPGA-based packet classification engine [5] and UE-TCAM [13] implemented TCAM using Altera and Xilinx primitive BRAMs, respectively.

Another large TCAM was previously handled by logically dividing the TCAM into relatively small TCAMs. Researchers implemented hybrid-portioned SRAM-based TCAM, HP-TCAM [8], Z-TCAM [9], and E-TCAM [12]. All used SRAMs and logic circuits to construct TCAM functionality. They partitioned the set of TCAM bits into groups of various sizes and mapped them to their SRAM-based TCAM architecture. The architecture uses two SRAMs. The first SRAM stores the information of the presence of a QS in TCAM bits. The second SRAM stores the address information (AI) for the corresponding QS. The HP-TCAM in [8] additionally stores the index

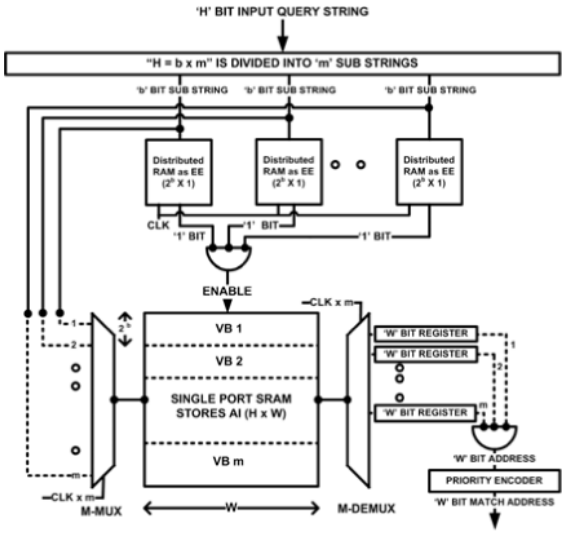

Fig. 1. Memory architecture of a single REST block of dimensions $W \times H$.

in the first SRAM that is later used to generate the address for the second SRAM.

Previous works [5]–[9], [12], [13] considered throughput as a primitive resource during emulation of TCAM and consumed large SRAM resources. To make the most of the available SRAM bits, it is essential to have the emulation scheme at an architecture level to exploit the SRAM bits available in different physical blocks. Thus, a resource-efficient approach is proposed, considering the tradeoff between resources like throughput and memory utilization.

III. REST Memory Architecture

REST is the emulation of classical TCAM in resource-efficient ways. It can search QS against a set of the same size strings (data string) in TCAM table and outputs the matched address for the string.

The REST memory architecture makes use of VBs in the SRAM units to achieve memory efficiency at the cost of reduced throughput. High-level organization of the memory architecture of a single REST block of dimension $W \times H - W$ addresses and H bits per address—is shown in Fig. 1. A unit REST block includes a single-port SRAM with m VBs for an AI table, m number of distributed RAMs for early elimination (EE) tables, a single multiplexer with m inputs, a single demultiplexer with m outputs, a priority encoder, and multiple AND gates to conduct the TCAM emulation.

The H-bit input QS is partitioned into the m number of b-bit substrings. The partitioned substrings are simultaneously used as the addresses of m number of distributed RAMs with a dimension of $2^b \times 1$ bit.

In the REST memory architecture, distributed RAMs are used as the EE table, as shown in Fig. 1. Precomputed lookup table (LUT) data in distributed RAM are built to find the presence of b-bit substring at an early stage. The output of each distributed memory is single bit. The m number of bits from m distributed RAMs are logically ANDed together to produce the signal used as the ENABLE signal to the AI table. If the enable signal is logically low, the rest of REST block operations are inactivated, and this corresponds to the EE operation.

The SRAM unit for the AI table is logically divided into VBs. The purpose of the AI table is to produce the original address in the TCAM table. Each VB has a size of $2^b \times W$ bits and stores the

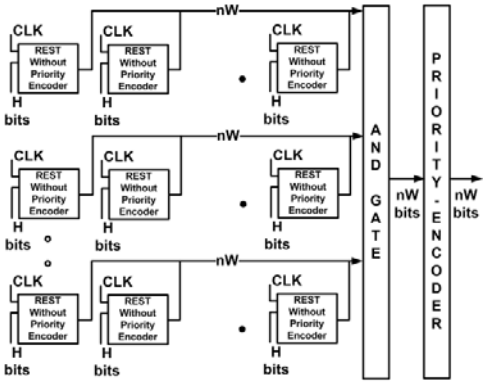

Fig. 2. Organization of the REST blocks to create larger TCAM.

W-bit original address that had the same data substring as the b-bit substring used.

The VBs of the SRAM hold AI, and each VB is uniquely associated with one substring. The association is achieved using a multiplexer. When EE operation is not active, all the b-bit substrings are sequentially used with their associated VBs in the AI table. The sequential access occurs because the AI table is made of a single-port SRAM.

The accessed and associated VB produces W-bit AI that is associated with its corresponding W-bit register through a demultiplexer. When all mW-bit registers are processed, the registers are bitwise-ANDed to produce the W-bit matching address. The m times faster clock (CLK $\times$ m) than the system clock (CLK) is used for the multiplexer, demultiplexer, and the SRAM unit. When there are multiple matching addresses, the priority encoder selects one prioritized address to complete a single search operation.

To emulate large TCAMs, it is indispensable to logically partition TCAM into smaller sizes of TCAM, and the partitioned TCAM is then mapped to one REST block. Fig. 2 shows an example of the array structure of REST blocks without priority encoders, bitwise AND gate, and a priority encoder.

All n-REST blocks are executed in parallel and produce nW-bit AI for the bitwise AND operation. The priority encoder selects 1 bit from nW bits when multiple matching addresses exist.

To update a data string stored in any of the multiple rows of REST blocks, it is first updated in the corresponding part of TCAM table, and then the updated portion is mapped to the conventional row by the memory write operations to the REST blocks (LUTs and the associated AI table in respected REST blocks). All the REST blocks in a row are updated in parallel. The update in a row is independent, as it does not affect the other data strings stored in other rows.

IV. Virtual Blocks of SRAM

Typically, a single-port SRAM unit with 2^n addresses and W bits per address—SRAM ($2^n \times W$)—can be used to store the W addresses and n-bit data strings in the TCAM table for the purpose of emulating TCAM with the dimension of $W \times n$. These W bits correspond to the AI in the AI table. Each bit in the W bits represents the original address position that stores the QS. The correlation of SRAM and TCAM sizes is shown in the following mapping relation:

$$\text{SRAM unit}(2^n \times W) \overset{\text{mapping}}{\Longleftrightarrow} \text{TCAM}(W \times n). \tag{1}$$

Equation (1) shows that SRAM-based TCAM can emulate TCAM with a size up to $W \times n$ bits in the SRAM with $2^n \times W$ bits. All previously published SRAM-based TCAMs used (1) to store AI in

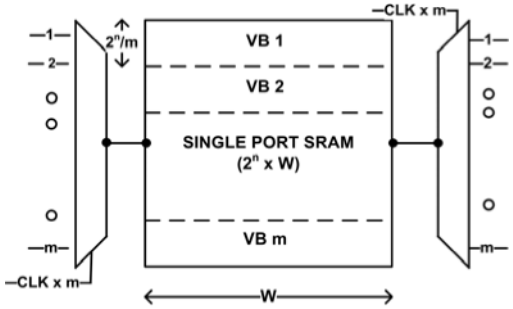

Fig. 3. VBs of single-port SRAM unit.

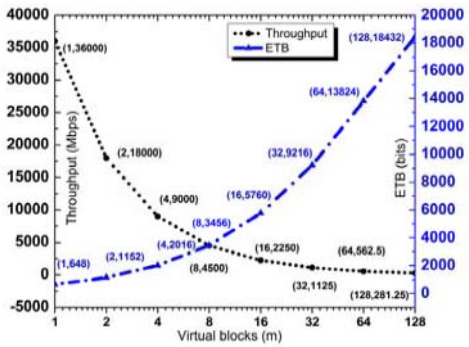

Fig. 4. Throughput versus ETBs/SRAM for different VBs (m) in an SRAM unit (512 × 72) running on a 500-MHz clock.

the SRAM unit. This allows only limited TCAM bits to be emulated in the SRAM. To add more TCAM bits, more SRAM blocks are required, which eventually reduces the memory efficiency. To mitigate this effect and increase the emulated TCAM bits (ETBs) per SRAM, VBs of the SRAM unit were created in REST. This was done by halving the SRAM address space until it reached the point when there were no more VBs available.

The size of each VB that determines the dimension of the TCAM table is partitioned. Fig. 3 shows a generalized example of the $2^n \times W$ SRAM used in REST memory architecture with m VBs; each VB has the dimension of $(2^n \div m) \times W$. In totality, $m \times [(2^n \div m) \times W]$-bit TCAM can be emulated.

As the number of VBs increases, ETB also increases in REST by exhaustively consuming all bits available in the SRAM—this is known as maximal memory efficiency. Note that the previous works [5]–[9], [12], [13] did not consider the memory efficiency in terms of the number of bits and correspond to the case of that in (1), as shown when m is 1 in Fig. 3.

Equation (2) shows ETB with a mathematical formula that has the three parameters of m, W, and n. The formations of AI tables are governed by the following formula:

$$\text{ETB} = m \times \{W \times (n - \log_2 m)\}. \tag{2}$$

Throughput–ETB Tradeoff in REST

Fig. 4 compares ETB in bits and throughput in megabits/s with varying numbers of VBs (m). The comparison was performed for a 512×72 SRAM that can operate at up to a 500-MHz clock frequency. As the number of VBs in SRAM increases from 1 to 128, the ETB increases from 648 to 18 432 bits. At the same time, the throughput decreases from 36 Gbits/s (72 × 500 MHz) to 281 Mbits/s (72×500 MHz divided by 128), respectively. REST resources can be

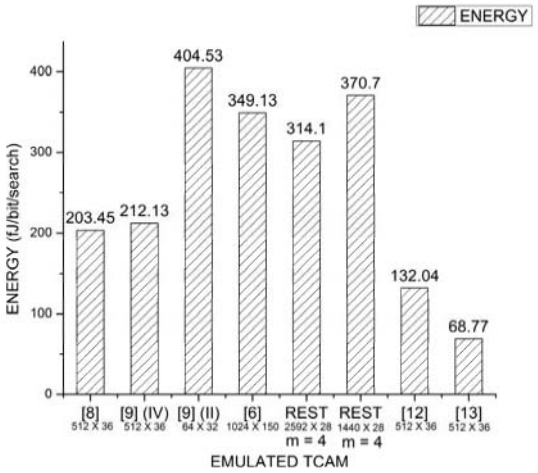

Fig. 5. Analysis of energy/bit/Search of emulated CAM Architectures.

adjusted by selecting either $m = 1$ (throughput efficient) or $m > 1$ (memory efficient).

Equation (3) is the formula used to calculate the throughput in a single REST block. The throughput is defined as the number of address bits per second produced from a REST block, which operates on CLK. However, the maximum frequency of CLK is dependent on m. An increase in m decreases the maximum frequency of CLK and thus reduces the throughput

$$\text{Throughput} = \text{CLK} \times W. \tag{3}$$

It should be noted that VBs in REST ($m > 1$) always bring larger ETB values compared with the case without VBs ($m = 1$). This makes SRAM-based TCAM a practical design choice in implementing large TCAMs when limited memory resources are available.

It is also true that SRAM-based TCAM may not bring the throughput required with a reasonable ETB because there are limited numbers of SRAM sizes in the application-specific integrated circuit or FPGA environments.

In comparison with the other emulated TCAM structures [5]–[9], [12], [13], REST has a built-in option of adjusting $m$to obtain the desired ETB, throughput, and latency. To the best of our knowledge, this is the first work to show relationships among the performance parameters embedded as a part of the TCAM emulation architecture.

V. Theoretical Analysis of REST

A. Energy Efficiency Analysis of Various TCAMs

The energy efficiencies of various emulated TCAMs are compared in Fig. 5. The comparison includes emulated TCAMs [6], [8], [9], [12], and [13] and the proposed REST. Emulated TCAM architectures along with their sizes and cases for respective architectures are shown on the X-axis, while associated energy/bit/search metrics are shown on the Y-axis.

The technology differences in the compared samples of CAM have been factored out by normalizing the energy consumption against the 65 nm using the following as seen in [11]:

$$E_{\substack{\text{norm. to} \\ \text{65 nm}}} = E \times \frac{65\ (\text{nm})}{\text{tech. node (nm)}} \times \frac{1.00\ (\text{V})}{V_{\text{DD}}(\text{V})}. \tag{4}$$

The large size 1024 × 150 TCAM in [6] consumed worst case energy of 349.13 fJ/bit. REST, when m is 4, also consumes comparable worst case energy, 314.1 and 370.7 fJ/bit for large sizes

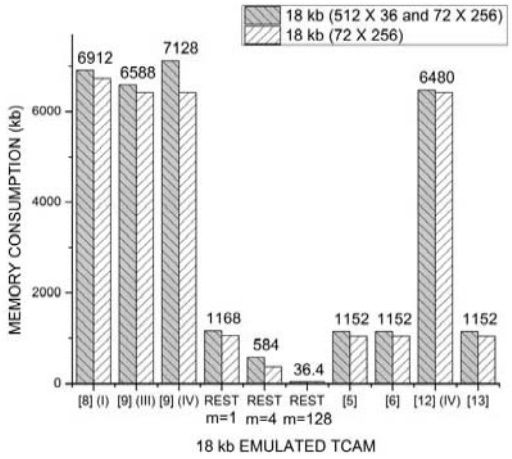

Fig. 6. Memory consumption comparison between 18-kbit emulated TCAM architectures and REST.

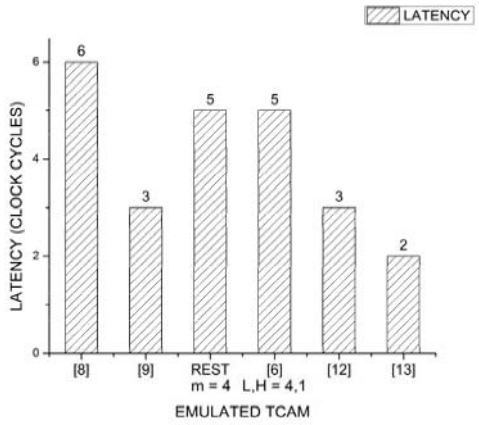

Fig. 7. Comparison of search latencies among various emulated TCAM architectures.

of 2592 × 28 and 1440 × 28 RESTs, respectively. The emulated TCAMs [12] and [13] consume average energy of 132.04 and 68.77 fJ/bit, respectively. All the energy results in Fig. 5 are based on the assumption in their respective previous works. REST calculated energies using the worst case scenario in which all SRAMs are activated during search operations.

B. Memory Efficiency Analysis of Various TCAMs

The bar chart in Fig. 6 illustrates the memory consumption of 18-kbit emulated TCAMs [5], [6], [8], [9], [12], and [13] and REST, when organized in variable array sizes and in the fixed array size of 72 × 256. The variable array of sizes 72 × 256 and 512 × 36 are organized for REST when m is 128 and for remaining emulated TCAMs, respectively.

Fig. 6 shows that REST using variable dimensions is the most memory efficient when m is 128. REST uses 36.4 kbits of memory resources when m is 128, representing 0.5% of memory consumption compared with case IV of Z-TCAM. The no-VB case in [5], [6], and [13] and REST consume comparable memory resources. However, REST is highly memory efficient with its inbuilt VB methodology that allows the designer to use almost 50% and 97% fewer memory resources than [5], [6], and [13], as shown in cases when m values are 4 and 128, respectively.

The fixed array of the 72 × 256 size also consumes similar memory except REST when m is 4, which further reduces memory consumption from 584 to 368 kbits in comparison with the array size of 512 × 36. Memory consumptions for 72 × 256 TCAM are estimated based on the memory architectures provided by the respective previous works, and all emulated TCAMs are organized in a single layer.

C. Search and Store Latency Analysis of Various TCAMs

The search latency in a REST block is defined as the SRAM read operations required from the time a QS is available at the input ports of the REST block to the time the outcome is ready at output ports. The latency is defined as the number of read operations in clock cycles.

The bar chart in Fig. 7 compares the search latency in clock cycles of emulated TCAMs [6], [8], [9], [12], and [13] and REST blocks with an m of 4. The TCAMs in [8], [9], [12], and [13] had the fixed latencies of six, three, three, and two cycles, respectively. However, the latencies of REST and [6] are dependent on the value of m and

TABLE I
BREAKUP OF DYNAMIC POWER CONSUMPTION OF A SINGLE 72 × 28 REST BLOCK USING THE XILINX X-POWER TOOL (WORST CASE SCENARIO)

RESOURCES	POWER CONSUMPTION	POWER CONSUMPTION (%)
CLOCK/PLL	7.71+ 93.28= 101	89.52 %
LOGIC	0.52	0.46%
SIGNAL	3.18	2.82%
BRAMS	8.13	7.21%
TOTAL	112.83	100%

architectural parameters L and H, respectively. The latency of REST with respect to the fast clock, CLK × m, is calculated using (5). It takes m cycles to read AI and 1 CLK × m cycle to take AI to the output port. The latency of [6] shows five clock cycles, when $L = 4$ and $H = 1$ are selected.

The store latency is defined as the maximum number of clock cycles required for write operations to store the worst case word, all Xs in the SRAM of REST block. The store latency of SRAM-based TCAMs [5]–[9], [12], and [13] and REST for an SRAM with a size of $2n \times W$ in their respective memory architecture is calculated using (6). For example, SRAM-based TCAMs of array size 72 × 9, using SRAM of array size 512 × 72, require 513 cycles to store all Xs as a word

$$\text{Latency}_{\text{search}} = m + 1 \tag{5}$$

$$\text{Latency}_{\text{store}} = 2^n + 1. \tag{6}$$

VI. HARDWARE IMPLEMENTATION IN FPGA

The 72 ×28 REST block has been implemented in the Xilinx Kintex-7 FPGA [10]. The REST block has four VBs and a system clock, CLK, of 50 MHz; it uses 200 MHz for the AI block because there are four VBs. The design uses one BRAM of 36 K and eight LUTs of 64 ×1.

A. Power Consumption

Table I shows the dynamic power consumptions in megawatts of various resources in the 72 × 28 REST block.

Power consumption was computed using the Xilinx X-Power tool for the worst case scenario, in which case BRAMs and logic blocks are activated and there are no power reduction cases applicable, such as no-EE case.

TABLE II
RESOURCE UTILIZATION OF 72×28 REST BLOCK

RESOURCES	QUANTITY
SLICE LUTS	130
SLICE REGISTERS	390
F7 MUX	4
BRAM (36 Kb)	1

B. Resource Utilization

Table II shows the resource utilization of the 72×28 REST block from the Xilinx Vivado tool. The first and second columns show the resources and their quantity, respectively.

VII. CONCLUSION

In this brief, we presented the novel memory architecture REST, which aims to adjust the resource efficiency in SRAM-based TCAM by selecting numbers of VBs in SRAM. It also investigated the tradeoff among resources like throughput and memory efficiency in constructing SRAM-based TCAM. The experimental results from the implementation of a 72×28-bit REST on FPGA showed a dramatic increase in memory efficiency using four VBs at the cost of a reduced throughput. The system uses one 36-kbit SRAM and eight 64×1 distributed RAMs, which is equivalent to 3.5% and 25.3% of memory resources compared with HP-TCAM and Z-TCAM, respectively.

To the best of our knowledge, this is the first brief to show the relationship among the performance parameters embedded in the TCAM emulation architecture.

ACKNOWLEDGMENT

The authors would like to thank S. J. Wen and Cisco Systems, Inc., for functional validation of REST, using Netlogic TCAM.

REFERENCES

[1] S. Baeg, "Low-power ternary content-addressable memory design using a segmented match line," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 6, pp. 1485–1494, Jul. 2008.

[2] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, Mar. 2006.

[3] P. Mahoney, Y. Savaria, G. Bois, and P. Plante, "Parallel hashing memories: An alternative to content addressable memories," in *Proc. 3rd Int. IEEE-NEWCAS Conf.*, Jun. 2005, pp. 223–226.

[4] B. Agrawal and T. Sherwood, "Ternary CAM power and delay model: Extensions and uses," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 5, pp. 554–564, May 2008.

[5] C. A. Zerbini and J. M. Finochietto, "Performance evaluation of packet classification on FPGA-based TCAM emulation architectures," in *Proc. Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 2766–2771.

[6] W. Jiang, "Scalable ternary content addressable memory implementation using FPGAs," in *Proc. ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS)*, Oct. 2013, pp. 71–82.

[7] Z. Qian and M. Margala, "Low power RAM-based hierarchical CAM on FPGA," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Dec. 2014, pp. 1–4.

[8] Z. Ullah, K. Ilgon, and S. Baeg, "Hybrid partitioned SRAM-based ternary content addressable memory," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 12, pp. 2969–2979, Dec. 2012.

[9] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "Z-TCAM: An SRAM-based Architecture for TCAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 2, pp. 402–406, Feb. 2015.

[10] Xilinx, San Jose, CA, USA. *Xilinx FPGAs*, accessed on Apr. 20, 2016. [Online]. Available: http://www.xilinx.com.

[11] P.-T. Huang and W. Hwang, "A 65 nm 0.165 fJ/bit/search 256×144 TCAM macro design for IPv6 lookup tables," *IEEE J. Solid-State Circuits*, vol. 46, no. 2, pp. 507–519, Feb. 2011.

[12] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "E-TCAM: An efficient SRAM-based architecture for TCAM," *Circuits, Syst. Signal Process.*, vol. 33, no. 10, pp. 3123–3144, 2014.

[13] Z. Ullah, M. K. Jaiswal, R. C. C. Cheung, and H. K. H. So, "UE-TCAM: An ultra efficient SRAM-based TCAM," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2015, pp. 1–6.