Architecture design tradeoffs in SRAM-based TCAMs

Ali Ahmed\textsuperscript{1,3}, Kyungbae Park\textsuperscript{2}, Saqib Ali Khan\textsuperscript{3}, Naeem Maroof\textsuperscript{3,4}, and Sanghyeon Baeg\textsuperscript{2}

Abstract An SRAM-based TCAM (SbT) memory architecture is proposed which exploits the tradeoffs among critical design parameters – such as throughput (T), latency (L), SRAM utilization (U), and power dissipation (P). An 18 kb TCAM is implemented on FPGA that can be adapted as latency & throughput efficient (LTE), Mid-efficient (ME), or a power & memory efficient (PME). Our implementation results show that LTE utilizes 79.3% and 96.5% more SRAM bit resources, consumes 45% and 55% more dynamic power than the ME and the PME, respectively. However, the LTE architecture shows an efficient single clock cycle latency and higher throughput than ME and PME, respectively.

Keywords: TCAM, SRAM, FPGA, emulation, memory architecture

Classification: Integrated circuits

1. Introduction

The classical ternary content-addressable memory (TCAM) compares input binary vectors with entire list of registered ternary vectors and outputs the address of matched one [1, 2, 3]. Despite being famous for fast searching capability, classical TCAM has high power dissipation, limited and fixed storage capacity [1], complex memory structure and high cost per bit. Researchers have attempted to emulate classical TCAM using static random-access memory (SRAM), which has relatively low power dissipation, better access time, reduced cost per bit and a simple structure [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15].

The SRAM-based TCAM (SbT) mimics the search capabilities by exploiting SRAM read operation. For a binary vector, applied as an input address, it generates a vector holding information of matched and mismatched locations. A priority encoder generates a single address in case of multiple matches. SbT is a suitable design choice in applications like information detection system [16, 17], forwarding engine on routers [18, 19, 20], IP address look-up tables [21, 22, 23], and high-speed data searching solutions [24, 26, 25]. Many works has been done on SbT architectures. Partitioning of large TCAM tables and mapping them to SRAM units have been employed in [8, 9, 10]. However, these techniques choose arbitrary table size and use larger SRAM units. Another technique [11] maps TCAM table on all block RAMs (BRAMs) and distributes on-chip memory of FPGA. Previously, throughput was considered as the only design parameter to emulate TCAM [8, 9, 11], which results in poor resource utilization. There is also a related valuable work [27] that emulate TCAM using eDRAM at chip-level to reduce power consumption and cost for FIB application in routers. DRAMs usually exhibit high access time in comparison to SRAMs which may affect latency issues for search operations. The tradeoffs among power and search latency remain static in [27] and cannot be changed.

Therefore, a methodology needs to be devised that capitalizes on the tradeoffs among all design parameters like throughput, latency, SRAM utilization and power dissipation. This paper aims to provide such methodology as a part of memory architecture. We use the concept of virtual blocks from our previously published work [28] to make a flexible and efficient SbT. Besides, we propose a memory architecture that prevents the early elimination unit from our previous architecture [28] and also exploits inter-dependencies of design tradeoffs for emulation of TCAM.

2. Proposed memory architecture: SbT

The memory architecture of the proposed SbT primitive block of dimension $k \times n$, $k$ addresses and $n$ bits per address is presented in Fig. 1. It is composed of a $m : 1$ multiplexer (MUX) and a $1 : m$ demultiplexer (DeMUX), $k$-bit AND gates, a priority encoder, and a single-port SRAM unit of array size $2^p \times k$, which is dissected virtually in $m$-layers. An $n$-bit input is partitioned into $m$ sub-keys. The SRAM unit is divided into virtual layers. The partitioned sub-keys are sequentially used as addresses to virtual layers of SRAM unit, which produces potential match addresses (PMA) in TCAM. Each virtual layer stores a $k$-bit PMA that is associated with the sub-key used. The 1 and 0 of $k$-bit PMA shows matched and mismatched addresses, respectively.

![Basic block of proposed SbT memory architecture](image-url)
The virtual layers of SRAM hold address. Each layer is exclusively associated with one sub-string, which is achieved using a MUX. The accessed virtual layers produce k-bit PMA that registered through a DeMUX. When all k-bit PMA registers are processed, they are bitwise-ANDed to generate the k-bit matching address. Priority encoder produces a single address in case of multiple matches occur.

To emulate a large TCAM, it is necessary to logically partition it into smaller units, which is mapped to a primitive block. Fig. 2(a) shows an example of a large TCAM array of size $N \times K$ which is divided into smaller units of $n \times k$. The $N$-bit and $K$-bit vectors are divided into $S$ and $L$ equal parts of size $n$ and $k$, respectively. Fig. 2(b) shows an arrangement of multiple primitive blocks to emulate a large TCAM of Fig. 2(a). The primitive blocks are executed in parallel and bitwise ANDed to produce $k$-bit PMA for priority encoder which selects the prioritized address.

### 3. Design tradeoffs in SbT

The proposed SbT involves numerous tradeoffs among important design parameters–like $T$, $L$, $U$, and $P$–to obtain a customized design, that best meets the requirements. The proposed work uses the concept of virtual layers in a single-port-SbT. The proposed methodology aims to evaluate the pivotal tradeoffs in designing SbT, which can emulate a TCAM of size $k \times n$ using an SRAM of size $2^p \times k$ with a single virtual layer. Previous SbT designs allow only a limited bits to be emulated; to increase TCAM size, more SRAM blocks are required. To mitigate this effect and increase emulated TCAM bits per SRAM (ETB), virtual layers are created by bisecting the address space until no more virtual layers can be created. The size of virtual layers is determined such that the capacity of holding TCAM bits of related entries is larger than that of a SRAM unit without using them. However, increasing virtual layers requires multiple accesses to the SRAM during search operation—which decreases $T$. Parameter $T$ in proposed SbT is defined as the number of address bits generated per second from a primitive block, which operates at system clock (CLK). The $T$ is halved, whenever virtual layers are doubled. For example, when virtual layers are increased from one to two in SRAM unit of $2^p \times k$; the ETB increases from $k \times p$ to $2 \times k \times (p - 1)$, but at the cost of two times reduced $T$. Eq. (1) and (2) show the mathematical expressions to compute ETB and $T$ by selecting appropriate number of virtual layers in SRAM units ($m$), respectively.

$$ETB = m \times (k \times (p - \log_2 m))$$  \hspace{1cm} (1)

$$Throughput = CLK \times k/m$$  \hspace{1cm} (2)

The latency is defined as SRAM read operations required from the arrival of input at primitive block till the output is generated. The selection of $m$ also affects the latency: it increases as $m$ is increased. Eq. (1) and (2) show the interdependencies of design parameters on $m$. The ETB increases with $m$, which contribute to efficient utilization of SRAM resources, however at the cost of reduced $T$ and increased $L$. Increasing $m$ also reduces $P$ as it requires less number of SRAM resources to emulate a TCAM. Therefore, $m$ in the proposed architecture can be determined to allow designers select the tradeoffs in relation to parameters such as $T$, $L$, $U$ and $P$. The proposed memory architecture can be adapted as the latency and throughput efficient (LTE) TCAM by selecting one virtual layer in the SRAM unit. The selection of $m = 1$ allows one access to SRAM for search operation to achieve higher throughput & single cycle latency. However, these benefits come at the cost of poor SRAM utilization and a higher power dissipation. Similarly, the architecture can be adapted as the power & memory efficient (PME) TCAM by creating maximum possible virtual layers in an SRAM unit. This allows the SRAM unit to hold large number of ETB and achieve minimum power dissipation, due to efficient SRAM utilization. However, these benefits come at the cost of high latency and reduced throughput because the PME configuration requires maximum number of sequential accesses to complete a search operation. Selecting $m$ in the middle of its maximum and minimum limit results in mid-efficient (ME) configuration of SbT. It produces intermediate latency, power dissipation and SRAM utilization in comparison to LTE and PME TCAM configurations.

### 4. Hardware implementation

We implemented the 18 kb LTE, ME and PME TCAMs using Xilinx Kintex-7 to exploit the tradeoffs among different parameters. The primitive block uses system clock of 100 MHz, and the input vector bit width (N) 256 is divided into smaller units ($n = 9$ for LTE, $n = 6$ for ME and $n = 2$ for PME) when 18 kb TCAM is realized. Table I shows the resource utilization and dynamic power dissipation for the Kintex-7 implementation of 18 kb TCAM configurations. The LTE, ME and PME designs utilize 29, 6 and 1 unit(s) of 36 kb RAM and consume 236, 129 and 105 mW, respectively, whereas number of layers (m) used are 1, 8 and 128, respectively. Increasing the number of virtual layers (m) from 1 to 8 and then from 8 to 128 decreases throughput by 87% and 94%, reduces the power dissipation by 45% and 19%, respectively. Required BRAM resources decrease by 81% and 84% when comparing LTE with ME and ME with PME, respectively. Nevertheless, latency of LTE mode is too low compared with the PME, while the latency of ME is medium-low.

### 5. Scalability of memory architecture

Here we discuss the scalability of proposed architecture for...
For the LTE configuration does not allow to be implemented on an FPGA. At the cost of increased latency & reduced throughput (see Table I), 18 kb SbT on Xilinx FPGA show that the architecture is flexible to implement LTE TCAM at the cost of 79.3% and 96.5% more SRAM resource usage and 45% and 55% higher power dissipation than ME and PME configurations, respectively. Likewise, PME configuration implements a power and memory (resource) efficient configuration at the cost of 129 clock cycles latency and 128 times reduced throughput than that of LTE TCAM.

### References

[1] S. Baeg: “Low power configuration strategy of TCAM lookup table,” IEICE Trans. Commun. E91-B (2008) 915 (DOI: 10.1109/ietcom/e91-b.3.915).

[2] B. S. Nataraj, et al.: U.S. Patent 6154384A (1999).

[3] B. S. Nataraj, et al.: U.S. Patent 6108227A (1999).

[4] S. Baeg and Z. Ullah: Korea Patent 101074495S1 (2009).

[5] W. Jiang: U.S. Patent US9111651B1 (2013).

[6] A. Ahmed and S. Baeg: PCT Patent WO2017052125 (2017).

[7] A. Ahmed and S. Baeg: Korea Patent 1017205A (2017).

[8] Z. Ullah, et al.: “Hybrid partitioned SRAM-based ternary content addressable memory,” IEEE Trans. Circuits Syst. I, Reg. Papers 59 (2012) 2969 (DOI: 10.1109/TCSI.2012.2215736).

[9] Z. Ullah, et al.: “Z-TCAM: An SRAM-based architecture for TCAM,” IEEE Trans. Very Large Scale Integ. (VLSI) Syst. 23 (2015) 402 (DOI: 10.1109/TVLSI.2014.2309150).

[10] Z. Ullah and S. Baeg: “Vertically partitioned SRAM-based TCAM,” JET 4 (2012) 760 (DOI: 10.7763/JET.2012.V4.479).

[11] W. Jiang: “Scalable ternary content addressable memory implementation using FPGAs,” IEE/ACM Symp. on ANCS (2013) 71 (DOI: 10.1109/ANCS.2013.6656177).

[12] T. Sasao: U.S. Patent US8352727B2 (2006).

[13] G. Levi, et al.: U.S. Patent 8052731A1 (2017).

[14] P. Maiocchi, et al.: U.S. Patent 9653165B2 (2017).

[15] T. Sasao, et al.: PCT Patent 7119533A1 (2007).

[16] D.-H. Le, et al.: “An FPGA-based information detection hardware system employing multi-match content addressable memory,” IEICE Trans. Fundamentals E95-A (2012) 1708 (DOI: 10.1587/.../E95.A.1708).

[17] D. R. Cheriton, et al.: U.S. Patent 7002965B1 (2001).

[18] K. Zaisu, et al.: “FPS-RAAM: Fast prefix search RAM-based hardware for forwarding engine,” IEICE Trans. Commun. E95-B (2012) 2306 (DOI: 10.1587/transcom.E95.B.2306).

[19] S. Sahni and W. Lu: U.S. Patent 2009015063A1 (2007).

[20] A. Basu, et al.: U.S. Patent 7356033B2 (2002).

[21] H. Park, et al.: “An efficient IP address lookup scheme using balanced binary search with minimal entry and optimal prefix vector,” IEICE Trans. Commun. E94-B (2011) 3128 (DOI: 10.1587/transcom.E94.B.3128).

[22] H. Lim: U.S. Patent US20050083937A1 (2003).

[23] V. Chandrasekaran: U.S. Patent 6862281B1 (2001).

[24] N. Onizawa, et al.: “High-throughput CAM based on a synchronous overlapped search scheme,” IEICE Electron. Express 10 (2013) 20130148 (DOI: 10.1587/elex.20130148).

[25] G. Kaldani and K. Izadi: U.S. Patent 6741591B1 (1999).

[26] H. Lim: China Patent CN1441579A (2003).

[27] Y. Kuroda, et al.: “A 200 Mbps, 0.6 W eDRAM-based search engine applying full-route capacity dedicated FIB application,” IEEE Custom Integrated Circuits Conference (2012) 1 (DOI: 10.1109/CICC.2012.6330671).

[28] A. Ahmed, et al.: “Resource-efficient SRAM-based ternary content addressable memory,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 25 (2017) 1583 (DOI: 10.1109/TVLSI.2016.2636294).

## Table I

| Parameters | LTE | ME | PME |
|------------|-----|----|-----|
| Dimensions (N × K) | 256 × 72 | 256 × 72 | 256 × 72 |
| LUT (logic + f) | 1188 | 795 | 686 |
| RAMB36 | 29 | 6 | 1 |
| No. of virtual layers (m) | 1 | 8 | 128 |
| Latency (cycles) | 2 | 9 | 129 |
| Throughput (Mbps) | 72000 | 900 | 56.25 |
| Power_L (mW) | 0.1 | 2 | 0.1 |
| Power_S (mW) | 6 | 3 | 1 |
| Power_R (mW) | 131 | 25 | 4 |
| Power_total (mW) | 95 | 95 | 95 |
| Power_total (mW) | 236 | 129 | 105 |

Where:

- N = S × n (used/total) = 29 × 9 = 261
- L = K × k (256/261) = 1 × 72 = 72

### Table II

| TCAM Size Mb | LTE (256 × 72) | ME (256 × 72) | PME (256 × 72) |
|--------------|----------------|---------------|---------------|
| 1 Mb | 58 | 2 | 12 | 2 | 129 | 0.56 |
| 2 Mb | 116 | 2 | 24 | 9 | 9 | 129 | 0.56 |
| 4 Mb | 232 | 2 | 48 | 9 | 9 | 129 | 0.56 |
| 8 Mb | 464 | 2 | 96 | 9 | 9 | 129 | 0.56 |