International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

High Performance, Area Efficient Ternary Content Addressable Memory (TCAM) With Fast Mapping and Updating Algorithm

Amreen Begum
PG Scholar, Department of E&CE (VLSI & EMBEDDED SYSTEM), Sharanbasva University, Kalaburgi, India

Article Info
Volume 9, Issue 4
Page Number : 284-289

Publication Issue
July-August 2022

Article History
Accepted : 05 July 2022
Published : 20 July 2022

ABSTRACT

Because of their high-speed lookup, content-addressable memory (CAMs) are utilized in a wide range of applications, including IP filtering, data compression, as well as artificial neural networks. Fast mapping as well as update methods for a binary CAM (FMU-TCAM) were provided in this article, which effectively use lookup tables, slice registers, as well as block random access memory (RAMs) on the Xilinx FPGA to imitate faster mapping as well as updated CAMs. The suggested approach has the advantage of directly using the CAM key as a location, which aids in the updating of memory unit contents. In remapping the CAM words including the updating word, CAMs in the literature consume the whole CAM depth, resulting in increased update latency

Keywords : Content-addressable memory (CAM), fast mapping algorithm, fast updating algorithm, random access memory (RAM)-based CAM.

I. INTRODUCTION

CAM compares incoming input to cached data in real time and returns the position of the matching data. CAMs could be found in network routers, artificial intelligence, microprocessor translations look-aside buffers, cache memory, data compression, as well as radar signal monitoring, among other places. Many of the most current methods include real-time pattern finding in virus infection as well as intrusion detection systems, genetic patterns finding in bioinformatics, and picture processing.

C Although CAM is an offshoot of RAM, their search functions are diametrically opposed. CAM receives contents and a corresponding address at the output, whereas RAM requires a location to extract the appropriate data. The high-speed search function of CAM is well-known. The high-power consumption, low bit density, and high cost per bit of CAM, on the other hand, come at a cost. Ternary CAMs (TCAMs) are around 30 times more expensive than DDR SRAM per bit of storage. TCAM also uses 150 times the amount of electricity per bit as SRAM. High power consumption raises junction temperature, which raises fault currents, lowers chip speed, as well as lowers CAM device dependability. CAM systems have very low design capability. The evolution of CAM technology is slower than that of RAM. In contrast to CAM technology, RAM technique is driven by various uses, mainly computers as well as consumer electronic items, and as a result, cost per bit continues to decrease. RAMs are available in a larger range of sizes and flavours than ever before. It is much more generic and readily available, allowing you to avoid the high licence as well as royalty fees that some CAM providers impose.

As a result, a new TCAM design that takes use of dense SRAM and provides benefits such bit density, cheaper cost, as well as equivalent searching speed is required.

An design based on SRAM has recently been suggested. SRTCAM is the name we gave to this design (SRAM-based TCAM). We take it a step further in this study by implementing SR-TCAM on a Virtex-5 FPGA as well as obtaining equivalent power consumption as well as latency outcomes. To our knowledge, it is one of the first research papers on SRAM-based TCAMs and SRAM-based TCAM design on FPGA. Since conventional CAMs can't be done on FPGA right now, one of the benefits of FPGA implementation is that it's faster.

Utilizing mixed CMOS/nano-electronic circuitry, the efficiency of RCs could be considerably increased. The CMOS+MOLecular (CMOL) FPGA, wherein CMOL stands for CMOL scale hybrids circuitry, was created to combine the density benefits of upcoming technologies like nano-imprint lithography as well as monolithically integrated self-assembled nano-devices with flexibility of CMOS technology.

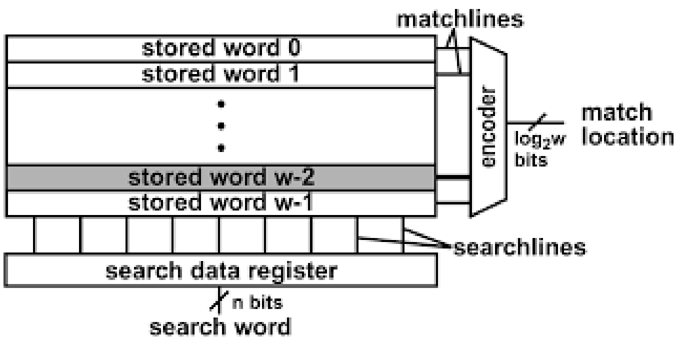


Fig. 1. Block diagram of CAM.

The second solution is based on TCAMs, that allow massively parallel bit-level comparison of data streams versus stored patterns. When compared to FPGA techniques, the relatively dense architecture of CAMs, which is around 2 sparser than standard static random-access memory (SRAMs), enables additional patterns to be recorded in the same piece of silicon.

Because of the TCAM's ternary nature, It's also useful in instances where you don't care. The method's drawbacks would be that the good memory bits used for matching should be loaded as well as released with each search cycle, even if no matches are detected.

II. EXISTING METHOD

In our existing mechanism, the update latency is independent of CAM depth. The low-power RAM-based CAM architectures are designed to reduce the power consumption for SRAM-based CAMs. Memory blocks are arranged in a hierarchy to have high and low priority blocks. If search word is found in the high priority block, low priority blocks remain deactivated. This power reduction in CAM is achieved by compromising latency and throughput, while reduction still not guaranteed in worst case scenario (e.g., activation of all priority blocks). Figure 2 depicts the suggested binary CAM rapid mapping and updating mechanisms/algorithms. Fast mapping as well as updating mechanisms that are more generalized using block RAMs are proposed for BiCAM, which is composed of L number of layers, each layer having cascaded SRAM blocks. The proposed mapping mechanism; however, uses only two clock cycles - t1 for generation of CAM sub words and t2 for mapping these sub words to the corresponding rows in SRAM blocks by using them as row address to set the corresponding bits of SRAM blocks. The beauty of the proposed mapping mechanism lies in directly mapping CAM words by using its sub words as row address to SRAM blocks without arranging it in ascending order unlike other designs. Instead of engaging the whole memory blocks, the suggested algorithm selects only one level of SRAM blocks to content updates at any position, consuming less energy in the process. Thus, the proposed mapping and updating algorithms speed up the table makeup and reduce energy consumption.

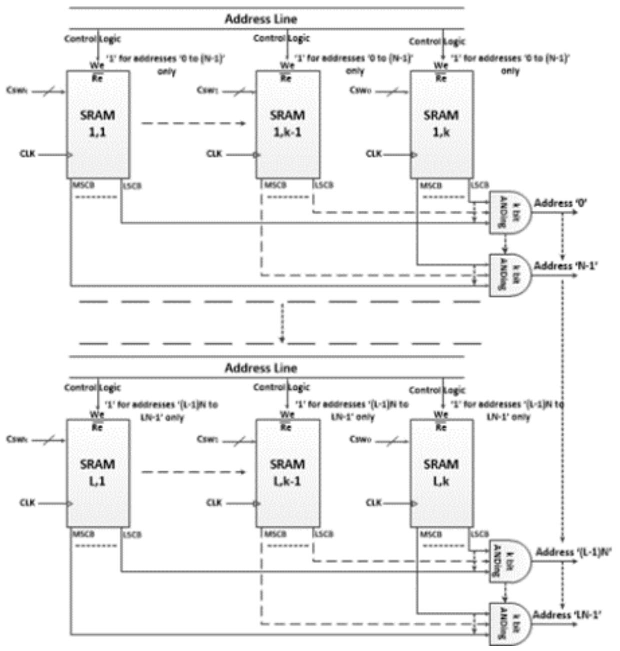


Fig.2 Architecture of Fast mapping and updating BI_CAM

Recent work shows that the comparison process is exhaustive in the worst-case scenario. This is to the fact that all SRAM-based CAM architectures use an indirect searching approach, which utilizes a lot of resources such as generation of temporary tables for storing interim data and generation of VM modules for checking and storing the presence of subworlds. Once the complicated circuitry is of no importance in the worst-case scenario, it can be removed and a direct searching approach may be incorporated. The removal of such complex circuitry significantly reduces the resource utilization and power consumption of SRAM-based CAMs and ultimately leads to optimization in other parameters as well. Updating of data contents discussed in the literature is solemnly dependent on the positioning of updated CAM word with existing stored contents.

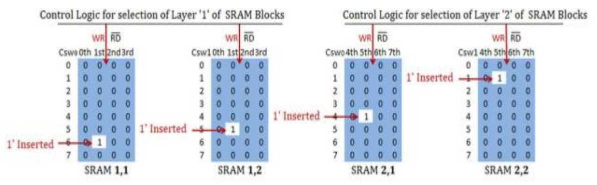


Fig 3 Mapping process in configured 8 × 4 architecture of a BiCAM with L and k both equal to 2.

Mapping the updated CAM word in the appropriate location involves iterative steps, which lead to higher latency. Such limitations in the update mechanism of SRAM-based CAMs motivate the development for a quicker mapping and updating techniques that aren't affected by the depths of the CAM .

The quick mapping and updates techniques have been generalized using block RAMs are proposed for BiCAM, which is composed of L number of layers, each layer having cascaded SRAM blocks, as shown in Fig. above. It's worth noting that we were using 18k block RAM rather than 36k block RAM, that was used in many of the previous research. This is used to cut down the number of block RAMs within every tier, for example, k. Although decreasing k increases the total number of levels in the design for correctly simulating the goal 512 36 CAM, testing results showed that increasing the number of lines and reducing the number of SRAM blocks in a layer serves to decrease energy consumption while updating. Hence, instead of using 32 block RAMs of 36k size, 64 block RAMs of 18k size are used. The first layer cascades blocks from (SRAM1,1) to (SRAM1,k), whereas the last layer cascades blocks from (SRAML,1) to (SRAML,k). Upon fetching the required address location on address line, the control logic on top of each layer selects the corresponding layer word 1 (shift_1) and mix Mapping of the CAM word starts from dividing CAM word into subworlds, whereas the number of CAM subwords depends on the number of SRAM blocks in each layer or vice versa. A number of addresses a layer can link indulges on the width N of SRAM block, whereas N refers to valid informative column bits of SRAM block. In every layer, the corresponding significant bit SB of each memory block verifies the presence of linked subwords. Concurrently, the correlated SBs of all SRAM blocks in a layer are ANDed together to check the presence of input word Cw. Equation (1) describes the size of CAM depth Cd.

Mapping process is complex with the use of temporary lookup tables. Divide the CAM word into subwords in time period t1, make bit position table (BPT) from the corresponding subwords in t2, build address position table address generator (APTAG) from corresponding BPT in t3, and finally make address position table (APT) from the corresponding APTAG table in t4. The process differs somewhat and by using VMs modules in place of BPT and original address table address generators (OATAGs) in place of APTAG, yet consume a number of clock cycles for mapping contents at the desired location. The proposed mapping mechanism; however, uses only two clock cycles - t1 for generation of CAM subwords and t2 for mapping these subwords to the corresponding rows in SRAM blocks by using them as row address to set the corresponding bits of SRAM blocks. The proposed mapping mechanism is independent of arranging the mapped words in any order. Algorithm 1 demonstrates the mapping process. To map the queued CAM word Cw into the desired location, Cw is divided into k same sized subworlds, Csw(0) to Csw(k). The beauty of the proposed work lies in incorporating control logic for mapping process unlike other architectures that peculiarly activate the addressed location layer only.

The proposed direct mapping and sequence-independent update procedure significantly reduce the energy consumption during both processes compared with all the reviewed SRAM-based CAM architectures.

III. PROPOSED METHOD

Every CAM unit has its own store capacity and comparison circuitry. The two types of CAMs based on bit storage capacity are Binary CAM (BiCAM) and Ternary CAM (TCAM) (TCAM). TCAM can save three states (0, 1, and X, with X representing a don't care state), but BiCAM can only store two logical numbers (0 as well as 1). As a result, any of the two search values 1000 and 1001 will match a stored key of 100X in TCAM. TCAM is made up of two-dimensional arrays of cells. A match line (ML) is shared between cells in the same row (TCAM word), while search lines are shared between cells in the same column (SLs). A 4 x 3 TCAM array with 4 rows (TCAM words) and 3 columns, as seen in a standard TCAM array. Each TCAM phrase has its own ML, while cells in a same column have their own SLsIn a traditional TCAM, all the MLs should be pre-charged to VDD and all the SLs must be discharged to Gnd before the search key is pressed for a search operation. The search key is then simultaneously applied to all SLs. If there is a match, ML remains charged; otherwise, it discharged to Gnd via the TCAM cell's comparison circuitry. If all the cells that share the same ML have the same condition, the ML stays at VDD. Otherwise, every cell that shares the same ML will release it to Gnd if there is a mismatch. A priority encoder is fed with MLs (PE).
A basic encoder is used in BiCAM since a single battle can occur. In TCAM, on the other hand, a single input word can be matched over several memory locations. As a result, a priority encoder (PE) is being used to choose the memory location with the highest priority; traditionally, the physical address with the lowest priority has the most priority. The corresponding SRAM entry is then invoked using this address. In traditional TCAM, an instance of a search operation is shown in Fig 4 following. TCAM receives the input word 0111. PE selects the top entry and generates match location 1, which will then be utilised to index the linked RAM entry.

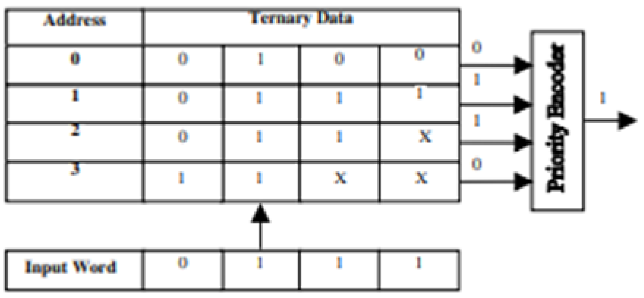
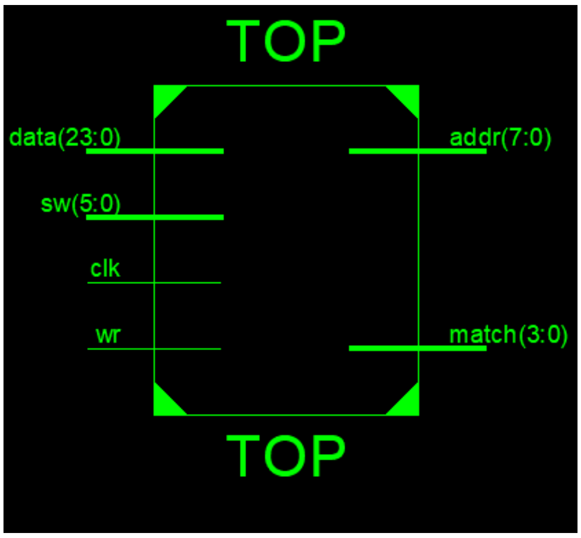


Fig 4: Example of search operation in classical TCAM

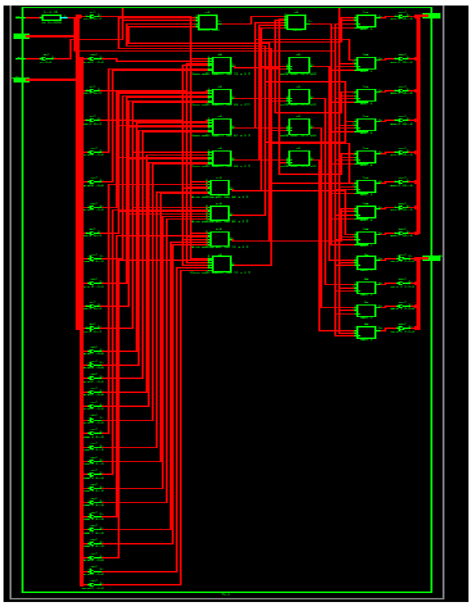
It is one of the first research papers on SRAM-based TCAMs and FPGA implementation that we are aware of. In regards of FPGA implementation and really measuring energy usage and delay, SR-TCAM is an addition, that has yet to be detailed.

IV. RESULTS AND DISCUSSION

The proposed mapping and updating algorithm and existing mapping and updating algorithm both are functionally verified by using Xilinx 14.7 version. The proposed design has better area and less delay when compared to existing design without degrading the other factors. In the proposed method while we are using parallel searching it will decrease the delay, improve the performance of the architecture.

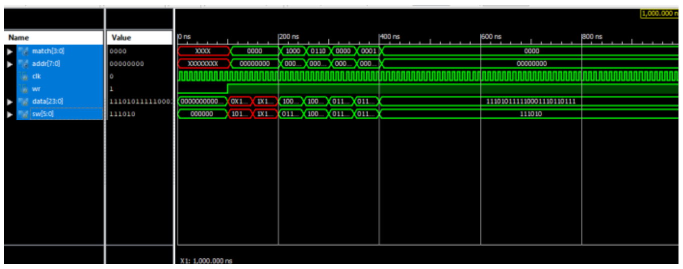


RTL Schematic



Technology Schematic

Simulation Results:



Evaluation of Area, Delay report:

	Area	Delay
Existing	38	3.948ns
Proposed	14	3.147ns

V. CONCLUSION

The designers have proposed several architectures on reconfigurable hardware, i.e., FPGAs. The dependence on the entire TCAM depth during update stage also leads to significant power consumption in the update process. SRAM-based TCAM architecture plays a pivotal role in artificial intelligence, pattern recognition, file storage, and networking router. This research work presents a different direction of sequence-independent update mechanism, which does not depend on TCAM depth. The proposed algorithm selects at most one layer of SRAM blocks for contents updating at any location rather than activating the entire memory blocks and ultimately consumes less energy during the update process. Thus, the proposed mapping and updating algorithms speed up the table makeup and reduce energy consumption.

VI. REFERENCES

[1]. A. Madhavan, T. Sherwood, and D. B. Strukov, "High-throughput pattern matching with CMOL FPGA circuits: Case for logic-in-memory computing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 12, pp. 2759–2772, Dec. 2018.

[2]. R. Govindaraj and S. Ghosh, "Design and analysis of sttram-based ternary content addressable memory cell," ACM J. Emerg. Technol. Comput. Syst., vol. 13, no. 4, p. 52, 2017.

[3]. Z. Ullah, M. K. Jaiswal, Y. C. Chan, and R. C. C. Cheung, "FPGA implementation of SRAM-based ternary content addressable memory," in Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. Workshops PhD Forum, May 2012, pp. 383–389.

[4]. 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, Oct. 2014.

[5]. 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.

[6]. M. Irfan, Z. Ullah, and R. C. C. Cheung, "Zi-CAM: A power and resource efficient binary content-addressable memory on FPGAs," Electronics, vol. 8, no. 5, p. 584, May 2019.

[7]. 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.

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

[9]. S.-H. Yang, Y.-J. Huang, and J.-F. Li, "A low-power ternary content addressable memory with pai-sigma matchlines," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 10, pp. 1909–1913, Oct. 2012.

[10]. B.-D. Yang, Y.-K. Lee, S.-W. Sung, J.-J. Min, J.-M. Oh, and H.-J. Kang, "A low power content addressable memory using low swing search lines," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 58, no. 12, pp. 2849–2858, Dec. 2011

Cite this article as :

Amreen Begum, "High Performance, Area Efficient Ternary Content Addressable Memory (TCAM) With Fast Mapping and Updating Algorithm", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 4, pp. 284-289, July-August 2022. Journal URL : https://ijsrst.com/IJSRST229445