VOL. 10, NO. 17, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

www.arpnjournals.com

# SRAM BASED ARCHITECTURE FOR TCAM FOR LOW AREA AND LESS POWER CONSUMPTION

Shaly Laurence and Anuros Thomas K.
ECE Department SCET, Kodakara, Thrissur, Kerala, India
E-Mail: shalylaurence1990@gmail.com

**ABSTRACT**

Ternary content addressable memories (TCAMs) perform high-speed search and network routing operations in a deterministic time. When compared with static random access memories (SRAMs), TCAMs suffer from certain limitations such as low storage density, relatively slow access time, low scalability, complex circuitry, and higher cost. This paper proposes an efficient memory architecture, which emulates the TCAM functionality with SRAM. This logically divides the classical TCAM table along columns and rows into hybrid TCAM sub tables and then maps them to their corresponding memory blocks. During search operation, the memory blocks are accessed by their corresponding sub words of the input word and a match address is produced.

**Keywords:** FPGA, TCAM, Hybrid partitioning, Application-specific integrated circuit, memory architecture, priority encoder.

## INTRODUCTION

Ternary content addressable memory (TCAM) is an outgrowth of random access memory (RAM) but unlike RAM, TCAM provides access to stored data by contents rather than by an address and outputs the match address. Since TCAM can store don't care state (*x*), which can be matched to both 0 and 1 during a comparison operation, multiple matches may occur. A typical CAM compares search key with all the stored words in parallel and returns the address of the best match. Since TCAM provides high speed parallel search operation, it has a wide use of applications such as it can find its applications in network routers, translation look-aside buffers in microprocessors, data compression, real-time pattern matching in virus-detection, intrusion-detection systems, gene pattern searching in bioinformatics, and image processing. The primary application of TCAM is in network systems where to compare the destination address of incoming packet against the stored addresses and forward the packet to the appropriate output port.

The proposed SRAM based TCAM may be used in networking systems where many data need to be compared in parallel at high speed. Currently, TCAMs are used in networking systems but they are expensive and not scalable with respect to clock rate or circuit area compared with SRAMs [1]. The throughput of TCAM is also limited by the relatively low speed of TCAMs [3]. Thus, SRAM- and FPGA-based TCAMs can be used in networking chips to achieve high speed and high throughput.

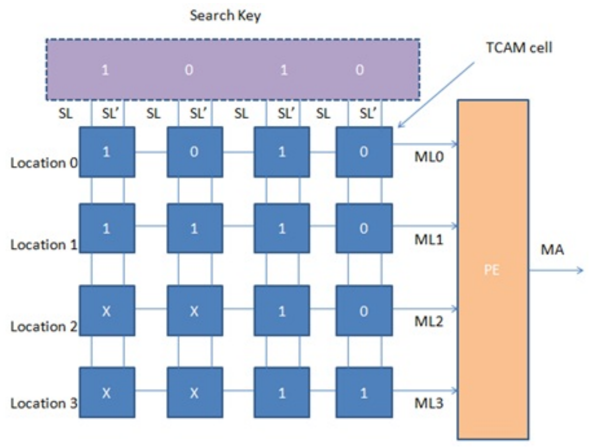


**Figure-1.** A simplified view of 4×4 TCAM.

In typical CAM architectures, before a search operation is performed, MLs are precharged and SLs are discharged. During a search operation, if comparison circuitry of its respective cell does not match the stored bit with the one on its corresponding SLs, the cell will pull down the ML to low. Even a mismatch of a single bit results in the mismatch of the ML. If we apply a search key of 1010 to TCAM in Fig.1, it finds matches at memory locations 0 and 3. PE then selects memory location 0 as the match address, considering that memory location 0 has the highest priority.

## LITERATURE REVIEW

RAM-based CAM proposed in [5] uses hashing technique; thus, inheriting inborn disadvantages of hashing—collisions and bucket overflow. The number of stored elements has a great impact on its performance; with the increase in number of stored elements, the performance of the method becomes gracefully degradable. Furthermore, the method emulates BiCAM not the TCAM. The method in [6] also uses hashing

VOL. 10, NO. 17, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

technique to emulate the TCAM functionality with RAM. Being based on hashing technique, it also suffers from collisions and bucket overflow. The RAM-based CAM in [6] may have further limitations. First of all, the performance depends on the actual record distribution and how records are accessed. If many records have been placed n an overflow area due to collisions, a lookup may not finish until many buckets are examined. Furthermore, when stored keys contain don't care bits in the bit positions used for hashing; then, to maintain the semantics of don't care bits, such keys must be duplicated in multiple buckets, leading to an increased capacity requirement. On the other hand, if the search key contains don't care bits which are taken by the hash function, then multiple buckets must be accessed that results in performance degradation.

TCAM in U.S. patent is not completely RAM-based structure. It uses classical CAM as a part of the overall structure. Thus, the combination of small part of CAM and large part of RAM provides overall memory structure. Therefore, this scheme inherits the inborn disadvantages of CAM technology. The advantages of the proposed TCAM over the above mentioned related work are listed below:

- Compared with the hashed-based CAMs [5,6] and SRAM-based pipelined CAMs [1], our proposed TCAM provides deterministic search performance of one word comparison per clock cycle, has efficient memory utilization, and is independent of type of data. Furthermore, we emulate TCAM, not BiCAM.
- Compared with [9], the proposed TCAM has an appropriate partitioning scheme and efficiently supports arbitrarily large words. We have also successfully implemented our proposed TCAM on Xilinx FPGAs, which demonstrates its technical feasibility.
- Compared with , proposed TCAM has an appropriate partitioning scheme, considers the storage of original addresses, provides better memory utilization, and speed could be higher as proposed TCAM does not have adder and counter. We have also provided the FPGA implementation that demonstrates the technical feasibility of the TCAM.
- Compared with, our proposed TCAM has an appropriate partitioning scheme, uses purely RAM arrays, and has been successfully implemented on Xilinx FPGAs.

## HYBRID PARTITIONING

We divide conventional TCAM table along columns (vertical partitions) and rows (horizontal partitions) that result in TCAM sub-tables. Each TCAM sub-table is termed as hybrid partition and the collective partitioning scheme (vertical and horizontal) is called HP. Figure-2 illustrates conceptual view of HP. The function of vertical partitioning (VP) part of HP is to divide TCAM word of $C$ bits into $N$ sub-words, where each sub-word is of $w$ bits. Thus, VP divides entire TCAM table into $N$ vertical partitions. Horizontal partitioning (HrP) part of HP divides each vertical partition into $L$ horizontal partitions by using the original address range of conventional TCAM table. Thus, HP results in a total of $L \times N$ hybrid partitions. Dimension of each hybrid partition is $K \times w$, where $K$ is a subset of original addresses and $w$ is the number of bits in a sub-word.

Since increase in the number of bits in a TCAM word increases the memory usage exponentially to a prohibitive limit, VP is used to decrease the memory requirement as much as possible. HrP cannot be used alone because it needs a prohibitive size of memory. For instance, if a TCAM word is of 32 bits and if we divide TCAM table into four horizontal partitions, then the memory usage would be 4 × 232, which results in a huge memory. Thus, HrP is not feasible because it is not efficient in terms of area, power, and cost; however, it provides a very beautiful way to make layers. Thus, the integration of VP and HrP provides a very good solution to make hybrid partitions and to reduce memory usage and power consumption as much as possible. Hybrid partitions spanning the same address range are in the same layer. For example, HP21, HP22, HP23, . . ., and HP2$N$ span the same address range and are in the same layer.

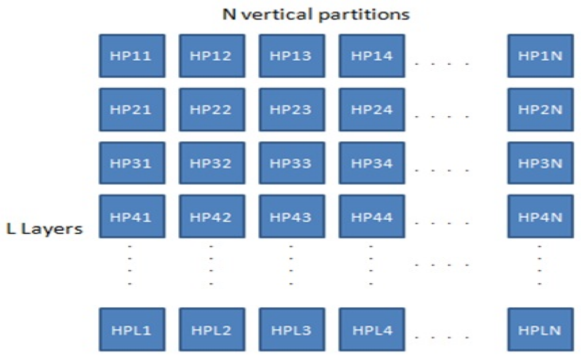


**Figure-2.** Conceptual view of hybrid partitioning (HP).

## ARCHITECTURE OF TCAM

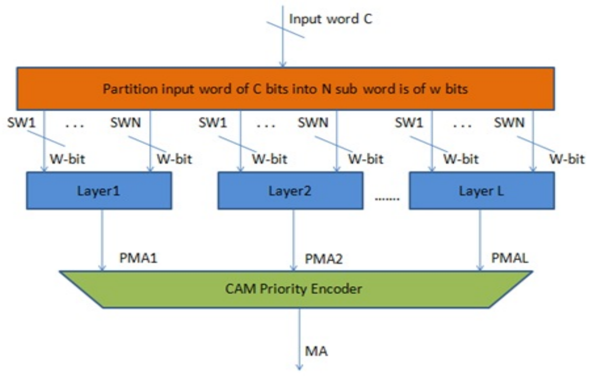


**Figure-3.** Architecture of proposed TCAM.

Figure-3 illustrates the overall architecture of E-TCAM, where each layer represents the architecture depicted in Figure-3. E-TCAM has $L$ layers and a CAM priority encoder (CPE). As can be seen from Figure-4, all $L$ layers receive $N$ sub-words simultaneously during the lookup operation. Each layer outputs a potential match address (PMA). The PMAs are fed to CPE, which selects match address (MA) among PMAs. For instance, if the proposed TCAM produces PMAs from layers 3, 4, 7, and 10 for an arbitrary input word, CPE selects PMA from layer 3 as MA, considering it has the highest priority.

**Layer Architecture**

Figure-4 depicts the layer architecture of E-TCAM. Since the total number of vertical partitions is *N* in a layer, to accommodate data in *N* vertical partitions, *N* validation memories (VMs) and *N* original address tables (OATs) are required. To achieve the TCAM functionality, the layer architecture is further equipped with 1-bit AND operation, *K*-bit AND operation, and a layer priority encoder (LPE). Each hybrid partitioning a layer has its corresponding VM and OAT. For instance, HP11 has its corresponding VM11 and OAT11. The VM and OAT of its corresponding partition constitute a pair. Thus, for *N* partitions, *N* pairs are required; each for a hybrid partition in a layer. We use SRAM units along with the additional logic of 1-bit ANDing, *K*-bit ANDing, and priority encoding to develop the architecture of E-TCAM. Thus, the collective functionality of 1-bit ANDing, *K*-bit ANDing, and priority encoding constitutes additional logic.

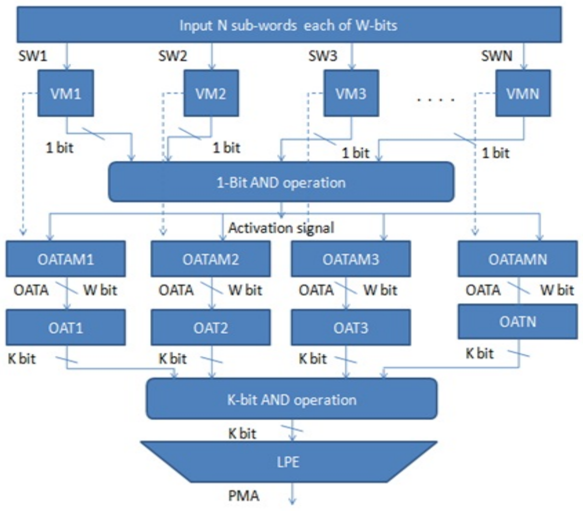


**Figure-4.** Architecture of a layer of SRAM based TCAM.

**Validation Memory**

Size of each VM is 2*w* × 1 bits, where *w* represents number of bits in each sub-word and 2*w* shows number of rows in each VM. A sub-word of *w* bits implies that it has total combinations of 2*w*, where each combination represents a sub-word. For example, if *w* is of four bits, then it means that there are total of 24 = 16 combinations. This explanation is also related to OAT. The sub-word validated by its corresponding VM is used as an address to OAT to read out a particular row, provided the search is permitted by 1-bit AND operation.

**Table-1.** An example of validation memory.

| Address | 1-bit data |
|---|---|
| 000 | 1 |
| 001 | 0 |
| 010 | 1 |
| ....... | .... |
| 111 | 1 |

**Original Address Table Address Memory**

Each OATAM is of 2*w* × *w* bits where 2*w* is the number of rows and each row has *w* bits. In OATAM, an address is stored at the memory location indexed by a sub-word and that address is then used to invoke a row from its corresponding OAT [10]. If a sub-word in VM is mapped, then a corresponding address is also stored in OATAM at a memory location accessed by the sub-word.

**Original Address Table**

Dimensions of OAT are 2*w*× *K* where *w* is the number of bits in a sub-word, 2*w* represents number of rows, and *K* is the number of bits in each row where each bit represents an original address. Here *K* is a subset of original addresses from conventional TCAM table. It is OAT, which considers the storage of original addresses. An example of OAT is given in Table 2, where 1 shows the presence of a sub-word at an original address.

**K-bit AND Operation**

It ANDs bit-by-bit the output of all OATs. *K* bits rows from all OATs are read out by using their corresponding sub-words, which are then ANDed and the result is then forwarded to LPE for further processing. Possible PMA is present in the result of *K*-bit AND operation.

**a) Layer Priority Encoder**

Since we emulate TCAM and as in TCAM multiple matches may occur in a layer, LPE is used to select PMA from the output of *K*-bit AND operation.

**Table-2.** Proposed TCAM data mapping.

| ADDRESS | VM21 VM22 | | OATAM21 OATAM22 | | ORIGINAL ADDRESS | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | OAT21 | | OAT22 | |
| | | | | | 2 | 3 | 2 | 3 |
| 0 | 1 | 0 | 0 | - | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | - | 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | - | 0 | 0 | 1 | 0 | 0 |
| 3 | 1 | 1 | 2 | 1 | 0 | 0 | 0 | 0 |

VOL. 10, NO. 17, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

## MEMORY OPERATIONS

### Data Mapping

A conventional TCAM table is logically partitioned into hybrid partitions. Since we emulate TCAM, a hybrid partition may contain an *x* bit. Since SRAM cannot store *x* bit, we first expand *x* into binary bits (0 and 1). For example, if a ternary word of 010*x* is present in an arbitrary hybrid partition, then it is first expanded into 0100 and 0101. These binary words are then stored in SRAM. Each sub-word acts as an address and accesses a particular memory location in the corresponding memory blocks (VM and OAT). Each sub-word is applied to its Corresponding VM and a logic _1_ is written at that memory location. The same sub-word is also applied to its corresponding OAT and *K* bits are written at that memory location. Thus, in this way, each sub-word in all hybrid partitions is mapped/programmed to its corresponding memory location in VM and original address of the same sub-word are mapped to its/their corresponding bit(s) in the corresponding OAT, respectively. Conceptual view of data mapping is illustrated in Figure-5.

A sub-word in a hybrid partition may be present at multiple locations. So, it is programmed in VM and its original addresses are mapped to their corresponding bits in OAT. A single bit in OAT represents an original address. Only those memory locations in VMs and address positions/original addresses in OATs are high, which are mapped, while remaining memory locations and address positions are set low in VMs and OATs, respectively.

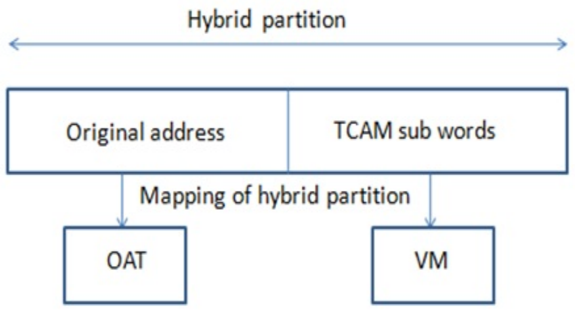


**Figure-5.** Mapping of a hybrid partition to its corresponding memory blocks.

### Search Operation

*N* sub-words are concurrently applied to a layer. The sub-words then read out their corresponding memory locations from their respective VMs. If all VMs validate their corresponding sub-words, then searching will continue, otherwise mismatch occurs in the layer. Upon validation of all sub-words the sub-words read out their respective memory locations from their corresponding OATAMs concurrently and output their corresponding OATAs. All OATAs then read out *K*-bit rows from their corresponding OATs simultaneously, which are then bitwise ANDed. LPE selects PMA from the result of the *K*-bit AND operation. Search operation in the proposed TCAM occurs concurrently in all layers. Search key is applied to TCAM, which is then divided into *N* sub-words. After searching, PMAs are available from all layers. CPE selects MA among PMAs; otherwise a mismatch of the input word occurs.

### Data Mapping Example

Proposed architecture for TCAM performs search operation in all *L* layers in parallel. It uses Algorithm 2 to accomplish the search operation. A search key is applied to the structure, which is then divided into *N* subwords. The subwords are then searched in their corresponding pairs in all layers in parallel. Algorithm 2 uses Algorithm 1 at step 3. PMAs from all layers are then received by CPE, which selects MA among PMAs; otherwise, mismatch of the search will be shown.

## MODIFICATION OF THE EXISTING ARCHITECTURE

### SRAM cell for the TCAM

The basic SRAM cell is shown in figure 6, it is constituted using a gate, latch and a tri-state buffer. Inputs to the gate are select line (sel_bar) ans Read/Write_bar(R_WB). When both sel_bar and R_WB are high then write operation takes place, i.e G becomes 1.If G=1, then the latch is enabled and the input D goes to Q (output of latch). When TB_EN=0, then OUTPUT =Q else 'Z'. This cell becomes a part of vertical memory .Assume a 4-bit search word is given, then we divide the search word in to two. So now we have 2 bit search word, thus require $2^2$ memory locations. So we implement the code for VM memory in the Xilinx ISE software and verified the working.

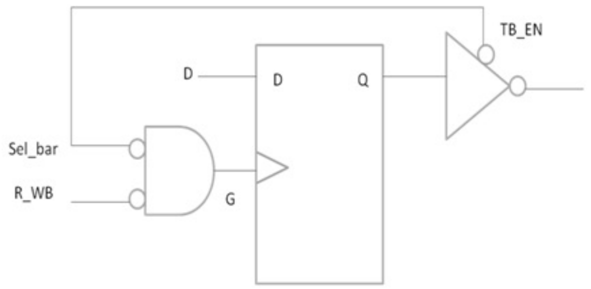


**Figure-6.** Basic SRAM cell for TCAM.

### Architecture Modification

Modification to existing TCAM architecture using SRAM is done by bypassing OATAM. The modified architecture connects the VMs directly to corresponding OATs and thus achieve the TCAM operations through correct data mapping. The main advantages of this modified structure is that less resources thus less memory usage and resource utilization. Speed of operation is increased.

VOL. 10, NO. 17, SEPTEMBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences
©2006-2015 Asian Research Publishing Network (ARPN). All rights reserved.

In the modified architecture the OATAM table is eliminated and direct mapping from VMs to OATs is done. This helps in reducing much resource utilization to be reduced and by reducing memory usage the performance and speed is improved. The OATAM tables are very difficult to design and map, and as the number of bits in the input search word increases the OATAM table size also increase enormously. So by eliminating the OATAM table we mapped the VMS directly with OATs and thus achieved the TCAM functionality.

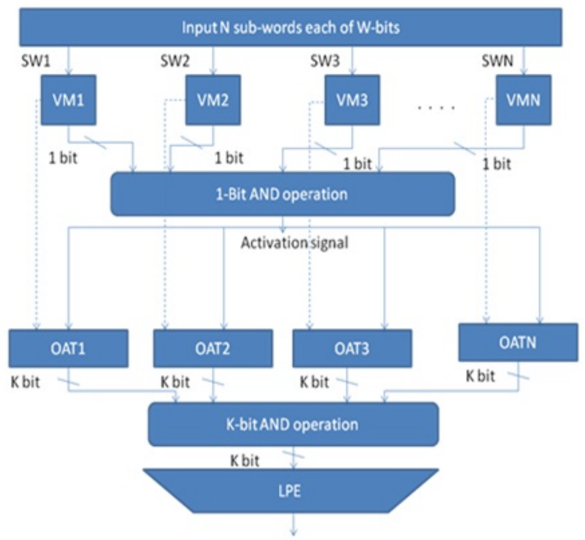


**Figure-7.** Modified architecture for TCAM.

## IMPLEMENTATION AND RESULTS

The Xilinx ISE design suite is used for the coding of the architecture. The ISE Design Suite: Embedded Edition includes Xilinx Platform Studio (XPS), Software Development Kit (SDK), large repository of plug and play IP including MicroBlaze Soft Processor and peripherals, and a complete RTL to bit stream design flow. Embedded Edition provides the fundamental tools, technologies and familiar design flow to achieve optimal design results. These include intelligent clock gating for dynamic power reduction, team design for multi-site design teams, design preservation for timing repeatability, and a partial reconfiguration option for greater system flexibility, size, power, and cost reduction.

ISE supports the following devices families and their previous generations: Spartan-6, Virtex-6, and Coolrunner. The coding is done in the VHDL language. Behavioral modeling is done for the coding part. Codes for the basic architecture and modified version are developed using Xilinx ISE design suite.

The basic architecture has VM, OATAM, OAT, AND, CPE (Priority Encoder), Clock modules. The modified structure has VM, OAT, AND, CPE, Clock and the application window code as well.

ATLYS development board for Spartan 6 is used as the hardware. The Atlys circuit board is a complete, ready-to-use digital circuit development platform based on a Xilinx Spartan 6 LX45 FPGA. The on-board collection of high-end peripherals, including Gbit Ethernet, HDMI Video, 128Mbyte DDR2 memory array, audio and USB ports make the Atlys board an ideal host for complete digital systems built around embedded processors like Xilinx's MicroBlaze. Atlys is fully compatible with all Xilinx CAD tools, including ChipScope, EDK, and the free WebPack, so designs can be completed with no extra costs.

### Simulation Results

Simulation windows of TCAM base architecture and TCAM modified architecture are shown below in Figure-8 and Figure-9 respectively.

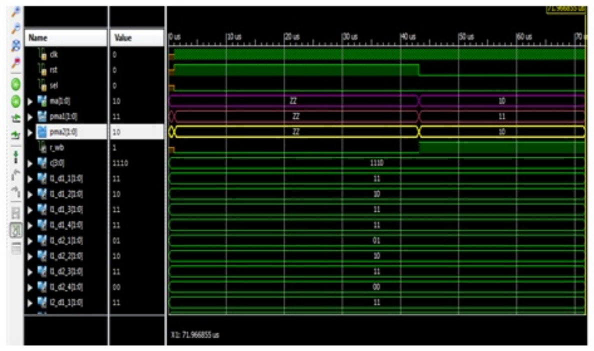


**Figure-8.** Simulation of basic TCAM architecture using SRAM.

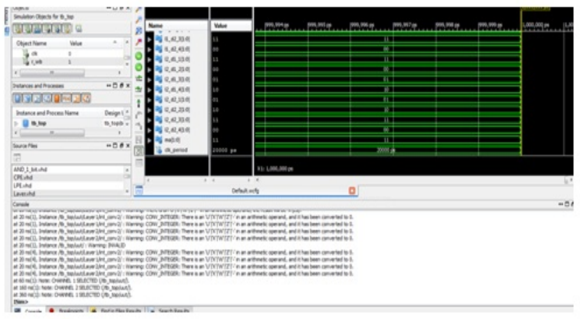
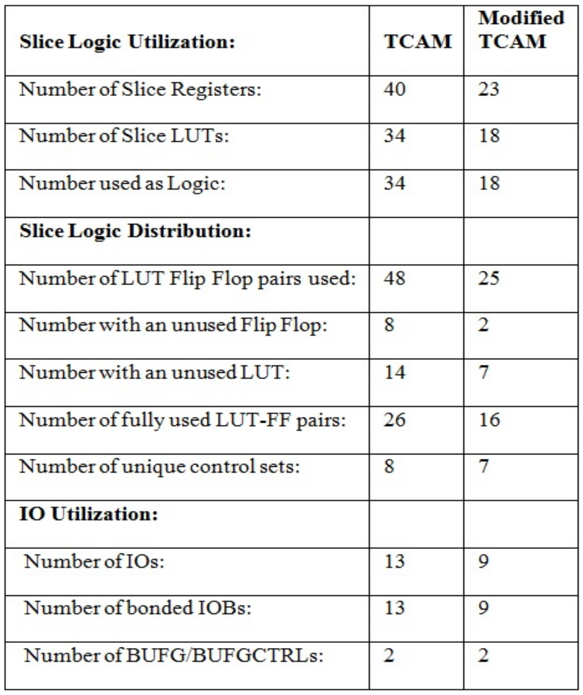


**Figure-9.** Simulation of modified TCAM structure.

The console window of modified TCAM architecture says which channel has been selected during the search operation. This application is useful while using in network routers. In network routers the addresses are stored in different locations and when we get an indication about the channel in which these addresses are stored, and then need to focus on those channels only.

Device utilization summary for both the architectures are shown in the Table-3. From the table it is clear that the modified architecture for TCAM is more optimized in terms of the code as well as the device utilization. Thus area is reduced, and since device utilization is reduced, the static power loss can be reduced.

**Table-3.** Device utilization summary.



| **Slice Logic Utilization:** | **TCAM** | **Modified TCAM** |
|---|---|---|
| Number of Slice Registers: | 40 | 23 |
| Number of Slice LUTs: | 34 | 18 |
| Number used as Logic: | 34 | 18 |
| **Slice Logic Distribution:** | | |
| Number of LUT Flip Flop pairs used: | 48 | 25 |
| Number with an unused Flip Flop: | 8 | 2 |
| Number with an unused LUT: | 14 | 7 |
| Number of fully used LUT-FF pairs: | 26 | 16 |
| Number of unique control sets: | 8 | 7 |
| **IO Utilization:** | | |
| Number of IOs: | 13 | 9 |
| Number of bonded IOBs: | 13 | 9 |
| Number of BUFG/BUFGCTRLs: | 2 | 2 |

## CONCLUSIONS

We have designed both TCAM architecture basic using SRAM and a modified version of it. After the simulation in ISE design suite and implementation on the Spartan6 ATLYS board, we could conclude that modified architecture works better than the current one in terms of resource utilization, power consumption and latency. As resource utilization is reduced in latches, Tristate buffers etc, the space utilization as well as power consumption is being reduced. From the timing analysis it is evident that delay is reduced in the modified architecture for TCAM. The console window is designed to show the channel selection, which helps in understanding which channel being selected for the search operations. So upon considering the above facts we conclude that an efficient structure for TCAM using SRAM is being implemented with certain improvement factors.

## REFERENCES

[1] M. Becchi and P. Crowley. 2008. Efficient regular expression evaluation: theory to practice, in Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS. ACM, New York, NY, USA, pp. 50–59.

[2] W. Jiang and V. Prasanna. 2012. Scalable packet classification on FPGA. IEEE Trans. Very Large Scale Integr. Syst. Vol. 20, No. 9, 1668–1680.

[3] W. Jiang and V.K. Prasanna. 2009. Large-scale wire-speed packet classification on FPGAs. In Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays, FPGA ’09, pp. 219–228.

[4] W. Jiang, V.K. Prasanna and N. Yamagaki. 2010. Decision forest: a scalable architecture for flexible flow matching on FPGA. In Proceedings of the International Conference on Field Programmable Logic and Applications, FPL’10, pp. 394–399.

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

[6] Z. Ullah, K. Ilgon and S. Baeg. 2012. Hybrid partitioned SRAM-based ternary content addressable memory. Circuits Syst. I, Vol. 59, No. 12, pp. 2969–2979.

[7] W. Jiang and V. Prasanna. 2008. Parallel IP lookup using multiple SRAM-based pipelines. In IEEE International Symposium on Parallel and Distributed Processing. IPDPS 2008, pp. 1–14.

[8] S. Cho, J. Martin, R. Xu, M. Hammoud and R. Melhem. 2007. CA-RAM: a high-performance memory substrate for search-intensive applications, in IEEE International Symposium on Performance Analysis of Systems Software, 2007. ISPASS, pp. 230–241.

[9] S. Dharmapurikar and P. Krishnamurthy. 2006. D. Taylor, Longest prefix matching using bloom filters. IEEE/ACM Trans. Netw. Vol. 14, No. 2, pp. 397–409.

[10] Zahid Ullah, Manish K. Jaiswal and Ray C. C. Cheung. 2014. “Z-TCAM: An SRAM-based Architecture for TCAM”, IEEE Trans. Very Large Scale Integr. (VLSI) Syst No. 99.