Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 2, No. 1, February : 2023

AN ENHANCEMENT OF FAST MAPPING AND UPDATING ALGORITHMS FOR A BINARY CAM

Murali Dova, Associate Professor, Department of Electronics and Communication Engineering NRI Institute of Technology, Agiripalli, Krishna-Dist, AP, India.
S.SRINIVASA RAO, P. LAKSHMI SUDEEPTHI, MARY LAVANYA, V. HIRANMAYI, P. PAVANI, Department of Electronics and Communication Engineering NRI Institute of Technology, Agiripalli, Krishna-Dist, AP, India

ABSTRACT

Content addressable memory (CAM) is a data storage device that stores memory in cells. It is a high –speed technology. CAM is used in very high-speed searching applications. It is also called associative memory, associative storage, or associative array. It is suitable for parallel search. Content-addressable memories (CAMs) are used in a variety of applications, such as IP filtering, data compression, data analytics, digital signal processing and artificial neural networks due to their high-speed lookup. Fast field-programmable gate arrays (FPGAs) are nowadays used to emulate CAMs. These CAM emulations either make use of logical resources or use memory blocks on FPGAs to emulate CAMs. However, such CAM emulation suffers from slower mapping and updating mechanisms, which results in an unacceptable response in real-time applications.

The slower response in the update mechanism is proportionate to the CAM depth in the schemes. This article presents fast mapping and updating algorithms for a binary CAM (FMU-BiCAM) which efficiently utilizes lookup tables, slice registers, and block random access memories (RAMs) on Xilinx FPGA to emulate faster mapping and updating CAMs. The advantage of the proposed work lies in directly applying the CAM key as an address, which helps in updating contents in memory units. When we process the code in FPGA, it works like a sowing machine. But, currently work with Xilinx ise design tool to simulate and shows the processing.

Key Words: *CAM, SRAM, FPGA, Xilinx ISE Design Tool, Bi-CAM, LUT’s, Slice registers*

1. INTRODUCTION

Now-a-days FPGA plays a vital role. The implementation of CAM in FPGAs has attracted considerable research recently. Unlike the dedicated-hardware CAM made by the custom-designed cells, CAM in modern FPGAs is constructed from the embedded random access memory blocks with a special mapping technique applied in the input data and address. CONTENT-ADDRESSABLE memory (CAM) plays a pivotal role in the present era of information access. Search operations can be performed in random access memory (RAM) by iteratively comparing the entire RAM entries for every search request, but it leads to a significant delay because of the sequential searching. On contrary, CAM is designed to allow comparison of its entire entries with the searched contents in a deterministic time. This unique feature of high-speed comparison enables CAM a fetching device in applications, such as artificial intelligence, pattern recognition, file storage, and database management.

Static RAM (SRAM)-based CAM architecture implemented on Xilinx FPGA can be summarized by analyzing all 3-D progress:
1) Development in CAM designs for use in parallel search operations
2) Enhance the mapping technique for low-resource utilization
3) Reduce the power consumption. A number of architectures, such as, have been surfaced in this connection.

Some of them have significantly achieved scalability as well as throughput, but almost all of them lead to higher latency in updating their contents. The update mechanism involves two sub processes, i.e., erase the old contents and write the new contents.

The update of data contents is a crucial factor and the present SRAM-based CAMs possess limitations in high-performance applications. Like in related literature, it takes at least O (N) clock cycles for updating, where N is the CAM depth. This motivates us to develop a high-speed update strategy that is independent of the arrangement of CAM words before mapping them to the desired locations. The proposed scheme has been implemented to validate its efficiency in case of fast updating (add or remove entries) for use in high-speed and low-latency contents matching in networking appliances, including routers, switches, firewalls, security, storage, and high-performance computing. Time consuming and sequentially dependent updates would not be acceptable in most networking applications, such as packet classification and pattern recognition.

I. Key Contributions of the Research Work

II. The key contributions are listed as follows.

1) The proposed fast mapping and updating algorithms for a binary CAM (FMU-BiCAM) are independent of the CAM depth and consume only two clock cycles unlike related prior work in which the update mechanism depends on the entire CAM depth, hence resulting in faster update mechanism compared with the prior work.

2) The development of the research work enables the user to configure the hardware implementation as per CAM depth requirements.

3) Since mapping/updating implies write operations, the proposed strategies may result in a significant reduction in power consumption compared with the state-of-the-art.

II. RELATED WORKS

Modern FPGAs having an unprecedented logic density along with the digital signal processing blocks, clocking, embedded processors, and high-speed searching with reasonable prices are preferred to be used in digital system designs. These advantages along with its reconfigurability motivated designers to implement CAM architecture on the FPGA platform.

The subsequent columns are named sub tables of CAM with n number of entries and processed to be stored in their corresponding SRAM blocks. Effectively incorporated parallelism in the matching process of large CAM words, but its mapping and updating process depend on O (N) (where N is the number of CAM words) and this raises the update latency to a very high value.

The approach adopted in [7] maximizes the parallelism in mapping and searching of CAM words by incorporating horizontal partitioning along with vertical partitioning, whereas the approach adopted in [5] removes the entire process of sequentially calculating the last index used in [3] and [7]. Designs in [3], [5], and [7], are focused to reduce the latency of mapping and lookup process, but at the cost of using redundant memory resources, which can be avoided. In the proposed work, we eliminated all such redundant memory resources and directly used the CAM word as an address to SRAM blocks, which results in efficient mapping and updating mechanisms.

The SRAM-based CAM designs presented in [4] and [8] provide deterministic searching in contents matching and are independent of data type and support search of arbitrarily large words, whereas lag in update mechanism as the update process is sequentially dependent on the entire CAM depth. In our proposed mechanism, the update latency is independent of CAM depth. The low-power RAM-based CAM architectures in [9] – [11] 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). The dependence on entire CAM depth for contents update in [12] – [17] also leads to high update latency.

Scalable CAM [18] is based on dividing the input CAM word into P sub words and deploys SRAM for matching by Adding the output of all corresponding memory blocks, connected to the encoder. The entire process is based on the trade-off between throughput and resource utilization. On one side, high throughput is ensured, but excessive use of RAMs has increased memory resources and power consumption. Despite all this, this update mechanism is still dependent on arranging its contents before mapping it to appropriate locations. The architecture in [19] uses a single memory block to emulate CAM on Kintex-7 FPGA. The method though achieves optimization of memory resources but at the cost of degraded scalability. In addition, virtual partitioning in a single memory block and use of validation memory (VM) trigger successive processes in single memory block and reduce speed significantly. The update mechanism mentioned in [19] again depends on O (N). Lee et al. [20] adopted the technique of bundle updates and the update mechanism in the architecture is proportionate to the width size of configured CAM rather than the depth. The design in [20] has compromised speed and scalability to achieve optimization in memory resources but still depends on the width of the configured CAM.

The SRAM-based CAM architectures mentioned in [12], [13], [21], and [22] address the issue of update mechanism through dedicated circuitry, yet the update latency is dependent on CAM depth. The logic-based schemes mentioned in [6], [23], and [24] simplify the update procedure at the cost of degradation in scalability and speed. Update time is reduced up to some extent in [24] and [25] but compromising either speed or resource utilization. The dependence of update mechanism on CAM depth in the literature motivated us to resolve the issue of slower updating and mapping for SRAM-based CAM. We effectively develop an algorithm that supports fast run time updating in its mapped contents. The design is having the beauty that the user can configure it as per CAM size requirement to avoid problems in run time updating. For example, to implement the CAM to handle a 36-bit key, the configured CAM could be of size (512×36) or (1024×36), or to handle 72-bit key, the configured CAM could be of size (512 × 72) or (1024 × 72).

III. EXISTING SYSTEM

RAM-based CAM implemented on FPGA is vertically partitioned CAM, which logically divides the conventional CAM table column-wise. The subsequent columns are named sub-tables of CAM with n number of entries and processed to be stored in their corresponding SRAM blocks. Effectively incorporated parallelism in the matching process of large CAM words, but its mapping and updating process depend on O (N) (where N is the number of CAM words) and this raises the update latency to a very high value.

Due to uncertainty conditions in existing method we can upgrade the features and goo for new approach called fast mapping.

DISADVANTAGES

- The cost of using redundant memory resources can’t be avoided.
- High update latency.
- Mapping positions are vary in slowly due to disturb in performance.
- Update status is slow due to increase the time and power consumption.

IV PROPOSED METHOD

4.1 INTRODUCTION

The proposed fast mapping and updating algorithms for a binary CAM (FMU-BiCAM) are independent of the CAM depth and consume only two clock cycles unlike related prior work in which the update mechanism depends on the entire CAM depth, hence resulting in faster update mechanism compared with the prior work.

Since mapping/updating implies write operations, the proposed strategies may result in a significant reduction in power consumption compared with the state-of-the-art CAMs. Our main Moto about this project is to effective utilization of RAM and fast mapping.

4.2 ADVANTAGES

- Resolve the issue of slower updating and mapping for SRAM-based CAM.
- Fast run time updating in its mapped contents.
- The design is having the beauty that the user can configure it as per CAM size requirement to avoid problems in run time updating.

4.3 ALGORITHMS

ALGORITHM 1: PROPOSED MAPPING ALGORITHM

Procedure: THIS ALGORITHM IS USED TO MAP THE CAM WORDS INTO DESIRED LOCATIONS OF SRAM BLOCKS

```
[DIVIDE THE CAM WORD INTO SUBWORDS]:
For i=0 to k do
Csw (i) =cw (i)/k
Next i
end for
[Find the address and insert CAM word]:
for i=1 to L do
for j =1 to K do
If SRAM (i, j) = Csw (i, j) then
SB of SRAM (i, j) = Cw
Exit
end if
next i, j
end for
end for
end procedure
```

ALGORITHM 2: LOOKUP OPERATION

Procedure: THIS ALGORITHM IS USED TO GET THE ADDRESS OF MATCHED CONTENTS FROM SRAM BLOCKS

```
[DIVIDE THE CAM WORDS INTO SUBWORDS]:
For i=0 to k do
Csw (i) =Cw (i)/k
Next i
end for
[Comparison]:
for i=1 to L do
for j=1 to K do
SRAM (i, j) =Csw (i, j)
Next i, j
end for
end for
[Address Making]:
for  i= 1 to L do
for j= 1 to K do
Temp = Temp + SB (i, j)
 if Temp == i then
Address = Temp
end if
Next i, j
end for
end for
[Get Address]:
Display Address
end procedure
```

ALGORITHM 3: PROPOSED UPDATING ALGORITHM

Procedure: THIS ALGORITHM IS USED TO UPDATE THE CONTENTS MAPPED TO ADDRESSED LOCATIONS AT S-RAM BLOCKS

[DIVIDE THE CAM WORDS INTO SUB WORDS]:

```
for i = 0 to K do
Csw (i) = Cw (i)/k
 end for
 Next i
[Find the Address and delete contents];
for i = 1 to L do
for j = 1 to K do
if SRAM (i, j) = Csw (i, j) then
Del (SBs of SRAM (i, j))
Exit
end if
Next i, j
end for
end for
[Insert new contents];
SB of SRAM (i, j) = new-word
Exit
end procedure
```

System Architecture:

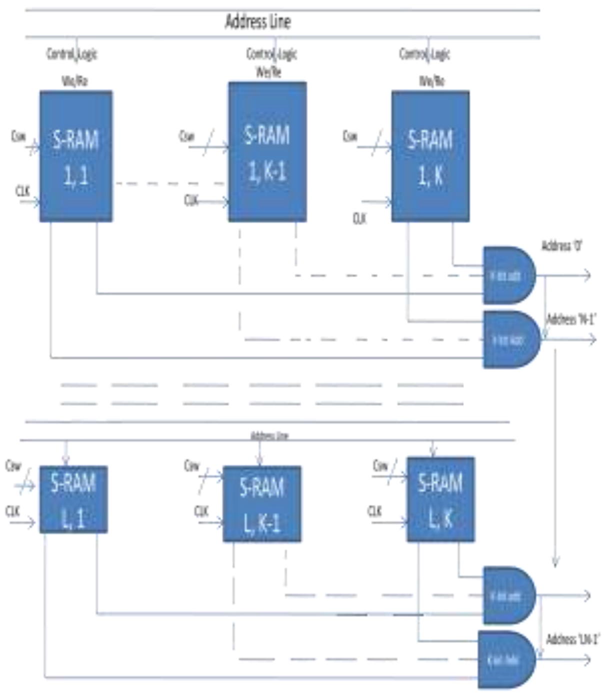


Fig-1 S-RAM Architecture

SYSTEM MODULES:

- MODULE 1- Mapping Mechanism
- MODULE 2- Lookup Process
- MODULE 3- Update Mechanism

MODULE 1- Mapping Mechanism

The proposed mapping mechanism is independent of arranging the mapped words in any order. Algorithm 1 demonstrates the mapping process.

MODULE 2- Lookup Process

The lookup process is initiated concurrently in all the layers. Algorithm 2 demonstrates the lookup process.

MODULE 3- Update Mechanism

Our proposed update mechanism by using the control logic, the CAM words are applied directly as row addresses to activate the corresponding SRAM blocks in the selected layer and avoid activation and rewriting contents of all layers. Algorithm 3 elaborates on the proposed update mechanism.

RESULT

RTL Schematic:

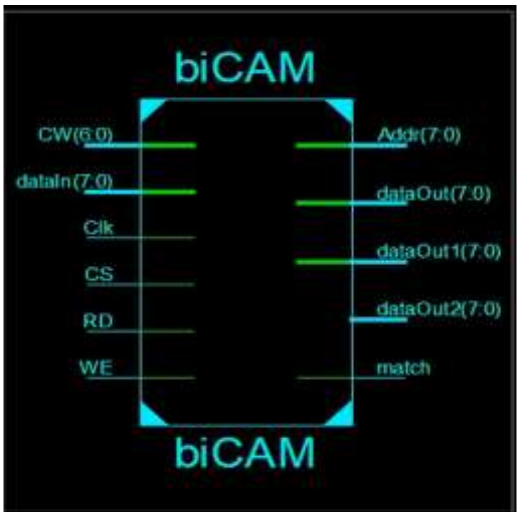


Fig-2 RTL SCHEMATIC OF 8x4 Bi-CAM architecture

We configure 8 × 4 Bi-CAM architecture. For this particular example, we select L = 2 layers of cascaded SRAM blocks having depth Rd = 8, and each block links N/L addresses, where N refers to informative column bits in SRAM block. Mapping CAM words 101110 to address location “1” and 010100 to address location “5” is the aim.

After partitioning first CAM word into k = 2 number of sub words, the least significant sub word Csw (0) = 110 is mapped to (SRAM1,1) by setting the corresponding significant bit of located address “1,” while the most significant sub word Csw (1) = 101 is mapped to the corresponding location of (SRAM1,2). Similarly, for second CAM word, the least significant sub word Csw (0) = 100 is mapped to (SRAM2, 1), while the most significant sub word Csw (1) = 010 is mapped to (SRAM2, 2) at address “5.” It is important to mention that mapping control logic singularly activates layer 1 for address “1” as it is linked to layer 1 and activates layer 2 for address “5” as it is linked to layer 2. 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.

SIMULATION:

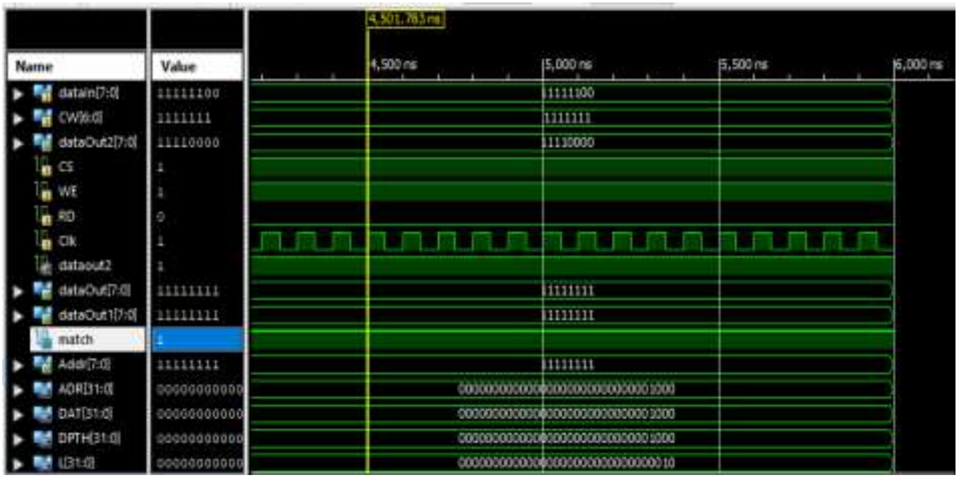
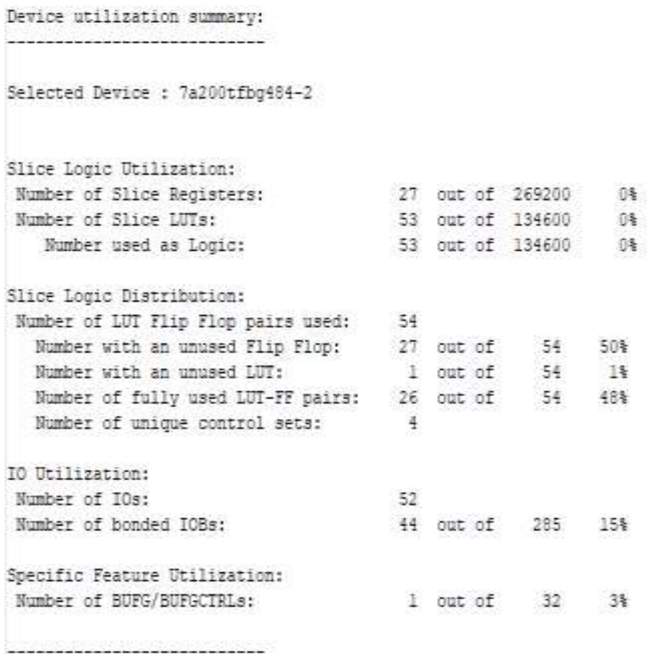


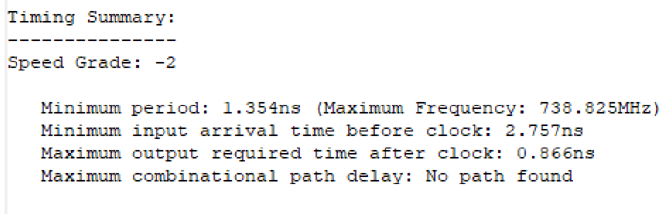
Fig-3 simulation of 8x4 Bi-CAM

Area:



```
Device utilization summary:
---------------------------

Selected Device : 7a200tfbg484-2


Slice Logic Utilization:
 Number of Slice Registers:             27  out of  269200     0%
 Number of Slice LUTs:                  53  out of  134600     0%
    Number used as Logic:               53  out of  134600     0%

Slice Logic Distribution:
 Number of LUT Flip Flop pairs used:    54
   Number with an unused Flip Flop:     27  out of     54    50%
   Number with an unused LUT:            1  out of     54     1%
   Number of fully used LUT-FF pairs:   26  out of     54    48%
   Number of unique control sets:        4

IO Utilization:
 Number of IOs:                         52
 Number of bonded IOBs:                 44  out of    285    15%

Specific Feature Utilization:
 Number of BUFG/BUFGCTRLs:               1  out of     32     3%

---------------------------
```

Timing:



```
Timing Summary:
---------------
Speed Grade: -2

   Minimum period: 1.354ns (Maximum Frequency: 738.825MHz)
   Minimum input arrival time before clock: 2.757ns
   Maximum output required time after clock: 0.866ns
   Maximum combinational path delay: No path found
```

V. CONCLUSION

SRAM-based CAM architecture plays a pivotal role in artificial intelligence, pattern recognition, file storage, and networking router. The designers have proposed several architectures on reconfigurable hardware, i.e., FPGAs. The state of-the-art CAMs suffer from higher update latency during contents updating as their update mechanism is dependent on arranging the entire contents with the updated contents. The dependence on the entire CAM depth during the update stage also leads to significant power consumption in the update process.

Future Work:

At present we do not work with FPGA hardware board. But, in future we can work with FPGA hardware board and definitely introduce a machine called "FPGA based sowing machine". It completely automated with Xilinx software.

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.
[11] V. S. Satti and S. Sriadibhatla, "Hybrid self-controlled precharge-free CAM design for low power and high performance," Turkish J. Electr. Eng. Comput. Sci., vol. 27, no. 2, pp. 1132–1146, 2019.
[12] I. Ullah, Z. Ullah, U. Afzaal, and J.-A. Lee, "DURE: An energy- and resource-efficient TCAM architecture for FPGAs with dynamic updates," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 6, pp. 1298–1307, Jun. 2019.
[13] I. Ullah, Z. Ullah, and J.-A. Lee, "EE-TCAM: An energy-efficient SRAM-based TCAM on FPGA," Electronics, vol. 7, no. 9, p. 186, Sep. 2018.
[14] Y.-J. Chang and Y.-H. Liao, "Hybrid-type CAM design for both power and performance efficiency," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 8, pp. 965–974, Aug. 2008.
[15] D. B. Grover, R. J. Stephani, and C. D. Browning, "Low power content addressable memory hitlineprecharge and sensing circuit," U.S. Patent 13 456 419, Oct. 31, 2013.
[16] D. Jothi and R. Sivakumar, "Design and analysis of power efficient binary content addressable memory (PEBCAM) core cells," Circuits, Syst., Signal Process., vol. 37, no. 4, pp. 1422–1451, Apr. 2018.
[17] Y.-J. Chang, K.-L. Tsai, and H.-J. Tsai, "Low leakage TCAM for IP lookup using two-side self-gating," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 60, no. 6, pp. 1478–1486, Jun. 2013.
[18] W. Jiang, "Scalable ternary content addressable memory implementation was using FPGAs," in Proc. 9th ACM/IEEE Symp. Archit. Netw. Commun. Syst. (ANCS), Piscataway, NJ, USA, Oct. 2013, pp. 71–82.
[19] A. Ahmed, K. Park, and S. Baeg, "Resource-efficient SRAM-based ternary content addressable memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 25, no. 4, pp. 1583–1587, Apr. 2017.
[20] D.-Y. Lee, C.-C. Wang and A.-Y. Wu, "Bundle-updatable SRAM-based TCAM design for Open Flow-compliant packet processor," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 27, no. 6, pp. 1450–1454, Jun. 2019.