TELKOMNIKA, Vol.17, No.4, August 2019, pp.1882~1889
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v17i4.11000

Ternary content addressable memory for longest prefix matching based on random access memory on field programmable gate array

Ng Shao Kay, M. N. Marsono*
School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia,
81300 Johor Bahru Malaysia
*Corresponding author, e-mail: mnadzir@utm.my

Abstract

Conventional ternary content addressable memory (TCAM) provides access to stored data, which consists of '0', '1' and 'don't care', and outputs the matched address. Content lookup in TCAM can be done in a single cycle, which makes it very important in applications such as address lookup and deep-packet inspection. This paper proposes an improved TCAM architecture with fast update functionality. To support longest prefix matching (LPM), LPM logic are needed to the proposed TCAM. The latency of the proposed LPM logic is dependent on the number of matching addresses in address prefix comparison. In order to improve the throughput, parallel LPM logic is added to improve the throughput by 10× compared to the one without. Although with resource overhead, the cost of throughput per bit is less as compared to the one without parallel LPM logic.

Keywords*: LPM, parallel processing, TCAM, update logic*

1. Introduction

Field-programmable gate array (FPGA) is beyond prototyping. FPGAs have been proposed in application-specific platform such as network FPGA (netFPGA) for high-throughput network processing [1, 2] such as for packet forwarding e.g. [3-5], deep packet inspection e.g. [6-8], network traffic classification e.g. [9-11], and network security e.g. [12-16]. However, conventional FPGAs do not have built-in ternary content addressable memory (TCAM). It is advantages for TCAM to be implemented in FPGAs especially for fast content look-up. TCAM on FPGA can be built based on either memory, or lookup table. UE-TCAM by Ullah [17] and Jiang [18] are examples of random-access memory (RAM) based TCAM. Both works proposed TCAM architectures with existing memory bits in FPGA. UE-TCAM [17] requires less latency compared to Jiang's [18], although the former is without update logic. On the other hand, Jiang's [18] consumes less memory as compared to UE-TCAM, which is suitable for RAM with narrow address width. TCAM may require frequent updates for LPM applications. Hence, UE-TCAM [17] and lookup table based TCAM such as [19] would require regeneration of partial bitfile when TCAM requires updates.

In order to perform the update operation, this paper proposes an update logic to the UE-TCAM [17] architecture. The proposed update logic performs TCAM add and delete operations. To extend the application of the proposed TCAM for LPM operations, a parallel LPM logic is also proposed. Section 2 discusses in details of related works. Section 3 explains the hardware architecture for proposed update logic and parallel LPM logic. Section 4 discusses and analyzes the result. Section 5 concludes the paper.

2. Related Works

Among the works that utilize RAM to implement TCAM are by Ullah et al. [17, 20, 21] where the RAM is being partitioned horizontally and vertically [17]. Divides conventional TCAM content into column and row order into $L \times N$ sub-tables, where $L$is the number of horizontal partitions and N is the number of vertical partitions. Each horizontal partition (a *layer*) is implemented as a RAM block of the same address range. Vertical partitioning divides W-bit

Received August 28, 2018; Revised February 23, 2019; Accepted March 12, 2019

words into *N* sub-words of *w*-bit wide. The vertical partitioning is aimed to reduce the memory requirement as much as possible. In [17], an input word of *C*-bit wide is divided into *Nw*-bit subwords in the vertical partitions. Hence, each RAM unit is $2^w \times K$, where *K* is the subset of original addresses. Each subword acts as an input to the RAM unit. The overall architecture of UE-TCAM [17] consists of RAM units, ANDing logic, and layer priority encoder.

Another RAM based TCAM is by Jiang [18], where narrow TCAM is used to build a wide TCAM. *P* narrow TCAMs are used to build $N \times W$ TCAM, and *P* is 1,2,..., *W*. The size of *i*-th TCAM is $N \times W_i$ where *i* is 1,2,..., *P* and *W* is the summation of W_i. The lookup time of *P* RAMs is $O(1)$ time. The update time will be determined by the longest time of update for RAM or $O(2^w)$ time [18]. Stated when the depth of each RAM is 2 (w=1) or 4 (w=2), the overall memory requirement is at its minimum. Simple dual-port RAM is being used to reduce update latency to 2^w+1 cc. Another work [22] focused on investigating different SRAM-based structure for IP look-up by the Jiang and Prasanna.

A modular architecture which consists of arrays of small-size RAM-based TCAM was also shown in [18]. The modular architecture is mainly to improve throughput and resource usage especially for large-scale implementation that has complex routing from bitwise ANDing and priority encoding, thus reducing the throughput. Besides, the achievable clock rate is also affected. Resource sharing is achieved by decoupling update logic to be shared by multiple TCAM units. Each unit is pipelined to solve for the throughput degradation in ANDing large number of bit-vectors. Low power RAM-based hierarchical CAM [23] was also introduced to reduce power from the modular architecture which constructed from *P*-narrow TCAM. Several other worthy approaches can be found in [24, 25].

3. Proposed RAM-based TCAM for Longest Prefix Matching

In the UE-TCAM architecture, *C*-bit input data is sent to *L* number of layers which contain *N* vertical partitions in each layer (i.e., *Nw*-bit subwords). The result of the vertical partition is *K*-bit, which is the subset of original address. Each bit of *K*-bit represents the original matching address of the TCAM. The *K*-bit addresses is then subjected through *layer priority encoder* (LPE). The lowest matching address serves as the *possible match address* (PMA). The CAM *priority encoder* (CPE) selects the lowest PMA as the final *match address* (MA). Increasing data width of a TCAM can be done through increasing either the number of partitions in each layer or the depth of the RAM. On the other hand, increasing the number of words stored in TCAM can be done by increasing either the width of RAM in every partition or increasing the number of layers.

3.1. Proposed Update Logic

There are two types of operations in the proposed TCAM architecture; lookup operation and update operation. The lookup operation is where input data sent to the TCAM to produce the output address. On the other hand, the update operation consists of the add and delete operations. The add operation adds TCAM word to certain addresses, whereas the delete operation deletes TCAM word from certain addresses. The lookup operation takes 1 clock cycle (cc), whereas the update operations take 2^w+1 cc. To add the update logics to the existing UE-TCAM architecture, several blocks are added, which are *dataRegUnit*, *maskUnit*, and *updateLogic*. The overall architecture is shown in Figure 1. The *dataRegUnit* is to hold the input data for update operations that take many cycles. The *maskUnit* decodes the mask for the add operation. *MaskUnit* will have *N* number of *maskProcessUnit*. The *i*-th bit of mask is '1' if the *i*-th bit of ternary word is binary or '0' if the *i*-th bit of ternary word is 'don't care' or usually denoted as 'X', e.g., '10XX' is masked by '1100'. The *maskUnit* will take *C*-bit mask, data, and update address counter from *updateLogic* to produce the update address and *wren* flag for the RAM in each vertical partition. For LPM application, the 'X' or '0' are located in the lowest *i* significant bits. In networking, the netmask is divided into two portions; the higher network prefix and the lower host suffix. The network prefix bits contain only '1' in netmask and are used for IP address forwarding. In the *maskUnit*, the bit that is high or not 'X' in the mask will be ANDed with *C*-bit data, whereas the bit that is '0' or 'X' is inverted and ANDed with the update address counter from the *updateLogic*. Then the ANDed results are both ORed together to become a valid update address for each vertical partition. By doing so, data are translated to the update

address of the vertical partition. If there is an 'X' in the mask, one or more addresses of the vertical partition or the location inside RAM will be updated.

The *wr_addr* is the word address for writing to the TCAM. There is a decoder in *updateLogic*, which functions for decoding the write address into bit location *m* of *K*-bit data for RAM update. For example, when the write address is 111, the 7-th bit of *K*-bit output of the decoder will be '1'. There are two decoded data from the *updateLogic* to write to the RAM partition, one is for the add operation and another is for the delete operation. One-hot encoding output from *updateLogic* is sent to Control Unit (CU) to control the write enabling of the dedicated layer.

The update address counter in *updateLogic* counts from 0 to 2^w-1. As the count is *w*-bit, concatenating *N* number of *w*-bit counts become *C*-bit update address counter. The *C*-bit update address is sent to *maskUnit* for processing, which returns the write address for RAM. During the add operation, only certain location of the RAM will need to be updated, whereas during delete operation, all RAM locations will be updated. Every vertical partition is made of a simple dual-port RAM. Each bit stored in RAM represents a TCAM location. As every RAM is *K*-bit wide, each RAM stores *K*-bit TCAM location. As the update operation will only update *m*-th bit without overwriting the remaining *K-1* bits, the RAM performs read for one cycle and performs write (of a new *K*-bit data) in another cycle. By using dual-port RAM, read and write operations can be done simultaneously in one cc. Hence, the total cycle for updating the RAM is 2^w+1 cc. When no update operation is performed, the data input of TCAM will be the input to TCAM for the lookup operation.

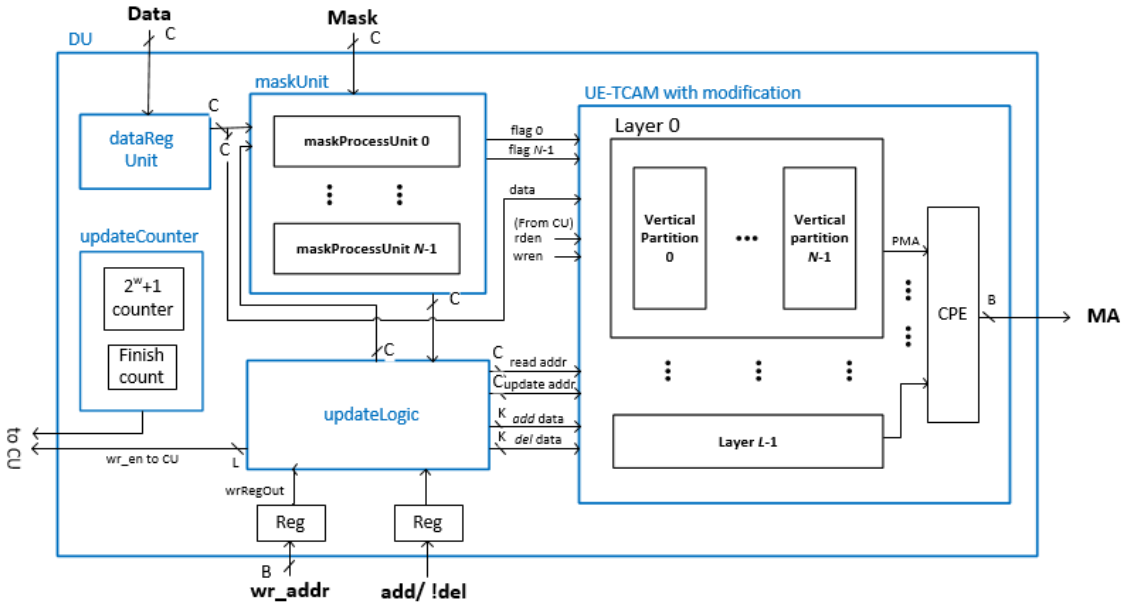

Figure 1. Modified UE-TCAM [17] with added update logic

3.2. Proposed LPM Logic

The proposed LPM logic for LPM application is shown in Figure 2. The modified TCAM with added update logic is as in Figure 1, except the LPE and CPE are removed. The *K*-bit output from each layer is concatenated to become the total match vector of *T* bits. The total match vector is then sent to *prefix_block* for extracting the prefix length for each match address. The *prefix_location_search* functions to obtain one match address starting from a lower address at each cycle. The match address from *prefix_location_search* is sent to *prefix_table* to obtain the prefix length of the matched address. After extracting the prefix length for each matched address, the *scan_longest_prefix* determines the longest prefix hit as the LPM result.

- The *mask_prefix_quantity* and the *write_prefix_data* are for the prefix length update. The *mask_prefix_quantity* calculates the total number of prefix length, while

the *write_prefix_data* processes the prefix length before sending it to *prefix_table* for an update operation.

- The *prefix_location_search* contains *Lget_location* that process*K* number of match vectors to obtain one matched address in each cycle. The CPE selects the matched address.*prefix_location_search* requires 1 cc to load the *T* match vectors into the *get_location* for initialization and several cc more to obtain match addressed from the match vector.
- The *prefix_table* contains several *prefix_table_partition*, each can store 1024 prefix data. The *prefix_table_partition* can be increased to store more prefix data. The *prefix_table_partitio*n also performs prefix add and delete operations. Reading the prefix length for a matched address takes only 1 cc. To add or delete the prefix data from the *prefix_table_partition*, 1 cc is required to read from RAM and 1 cc to write back to the RAM. Since the update latency is not critical for *prefix_table_partition*, normal RAM structure is sufficient.
- The *scan_longest_prefix* stores the match address with the longest prefix as the LPM result that takes 3 cc. It takes 1 cc to load the matched address and selects the 6-bit prefix length, another 1 cc loads the matched address with the longest prefix into the register, and another 1 cc to load the matched address as an output.

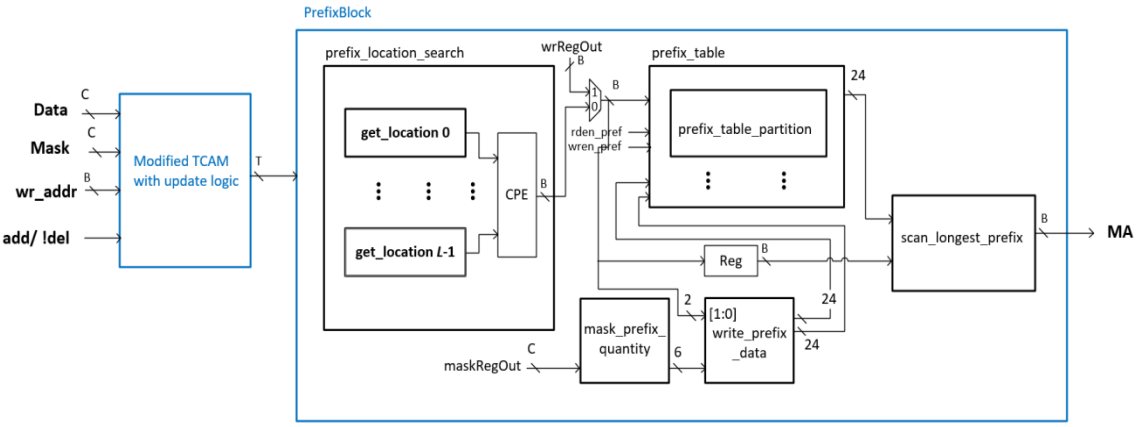

Figure 2. Modified TCAM with LPM logic for obtaining longest prefix address

The total clock cycle for obtaining the LPM depends on the number of matched addresses for prefix comparison. The total latency until the first result is obtained is 7 cc (1 cc to obtain the *T*-bit match vector from the proposed TCAM, 1 cc to load *T*-bit match vector into the register in *prefix_location_search*, 1 cc to start processing in *prefix_location_search*, 1 cc for *prefix_table*, and 3 cc for the *scan_longest_prefix*). In order to compare prefix length for next matching address, the process continues in the pipeline starting from processing *prefix_location_search* until the end of *scan_longest_prefix*. Let say the number of matching address for prefix comparing is *y*, thus the total cycle to perform LPM is 7+(*y*-1) cc. The update operation for LPM logic does not add additional cycles to the existing architecture as the prefix update operation is performed simultaneously with the TCAM word update.

3.3. Parallel LPM Logic

To increase the throughput of TCAM for LPM application, many LPM logic blocks are added. While waiting for the current LPM block to compare for prefix length of match addresses, another LPM block can start processing new set of output from the TCAM. The additional logic elements added for the parallel LPM logic are the demux, mux and counter. The counter counts *H* parallel stages of LPM logic. In each cc, *T*-bit match vector is loaded to different parallel LPM logic for processing. Figure 3 shows the parallel LPM logic. To reduce the latency of LPM logic, the *scan_longest_prefix* has been changed to 1 cc processing instead of 3 cc as in Section 3.2. The total cc for the improved LPM logic in one stage is 5 cc. Hence, to compare *y* match addresses for prefix length requires 5+(*y*-1) cc.

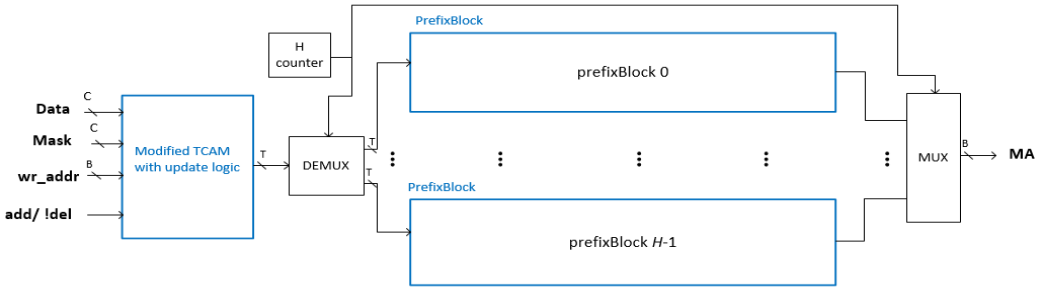

Figure 3. TCAM with parallel processing LPM logic

4. Results and Analysis

The TCAM architecture for LPM application is been configured for IPv4 (32-bit addressing). Besides, the proposed TCAM is evaluated for N=4, w=8, and K=128. The TCAM architecture is built on Intel Cyclone IV E Family. The memory unit inside every vertical partition is built with the on-chip dual-port RAM using M9K memory block type. The M9K memory can be configured to 256×128 block size. The value of L is varied as it determines the total number of TCAM words that can be stored. Also, the number of matching address for prefix length comparison is y=6. Therefore, the maximum parallelism stage that can be defined for increasing the throughput is H=10, i.e., as the total cycle of one LPM stage for six matching addresses is 10 cc.

4.1. Evaluation Results

As the input data and mask are 32-bit, the content stored in every address of TCAM shall be 32-bit as well. Since there are four vertical partitions in each layer, each partition has RAM block with 8-bit address width. The analysis results obtained from the proposed TCAM architecture are shown in Figures 4 to 7. As the number of TCAM word that can be stored increases, the number of logic elements and memory resource usage also increase. When the size of TCAM doubles, the memory usage also doubles as illustrated in Figure 4. The memory resource usage with parallel LPM logic increases slightly as compared to without parallel LPM logic as the additional memory is for the *prefix_table* only. Intuitively, more logic elements are required for the parallel LPM logic. However, the logic elements over throughputper TCAM bit is actually less compared to without parallel LPM logic Figure 5. The throughput improvement is about 10× as shown in Table 1.

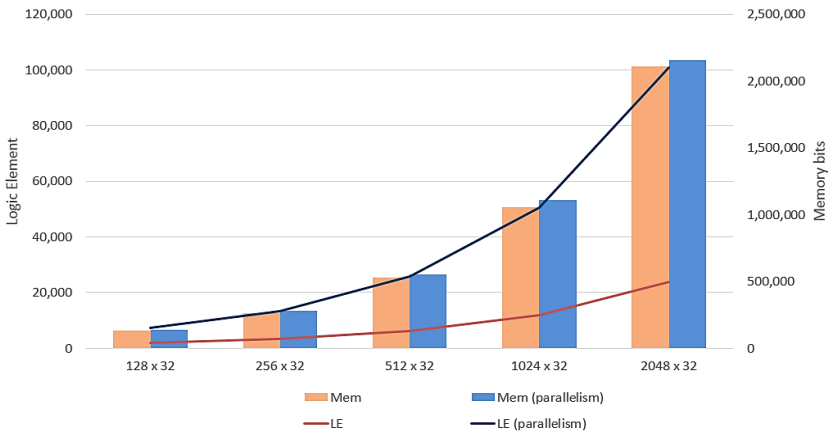

Figure 4. Effect of TCAM depth (N) on LE and Mem bits

The effect of different TCAM configuration on the maximum frequency is also evaluated. Figure 6 shows the frequency of the TCAM architecture as obtained from the TimeQuest Timing

Analyzer. The frequency decreases as TCAM size increases due to the fact that the critical path is affected by the PEs and CPEs in the TCAM architecture. This applies to TCAM with parallel LPM logic as well. In addition, the frequency of TCAM with parallel LPM logic is less than without parallel LPM logic. This is resulted from the addition of the demux, mux and counter for parallel LPM logic architecture. Figure 7 shows the effect of the number parallel stages of LPM logic to the TCAM architecture. The Figure shows that as the number of parallelism stage increases, the number of logic elements and memory bits increases almost linearly. This means that by enabling a different number of stages for parallelism, the number of logic element and memory bits can be estimated.

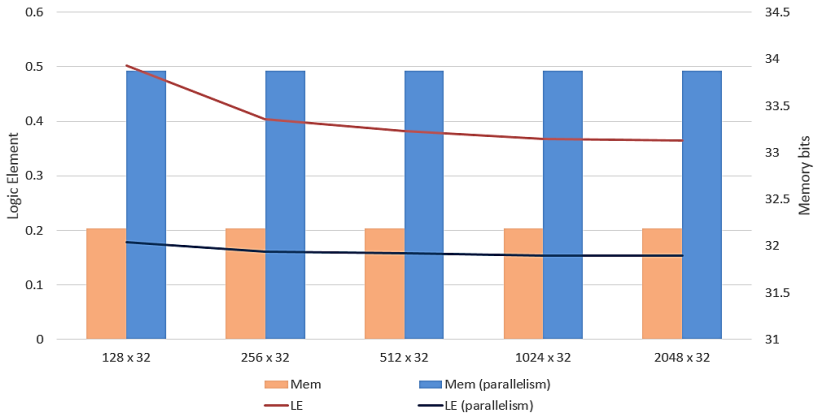

Figure 5. LE/throughput and Mem bits per TCAM bit

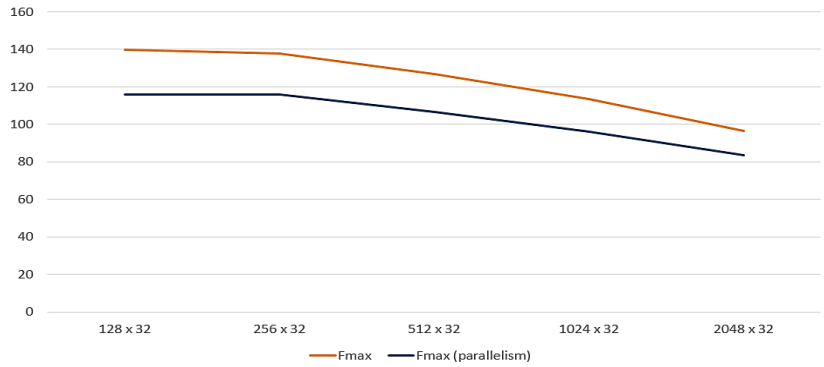

Figure 6. Effect of TCAM depth (*N*) on maximum frequency

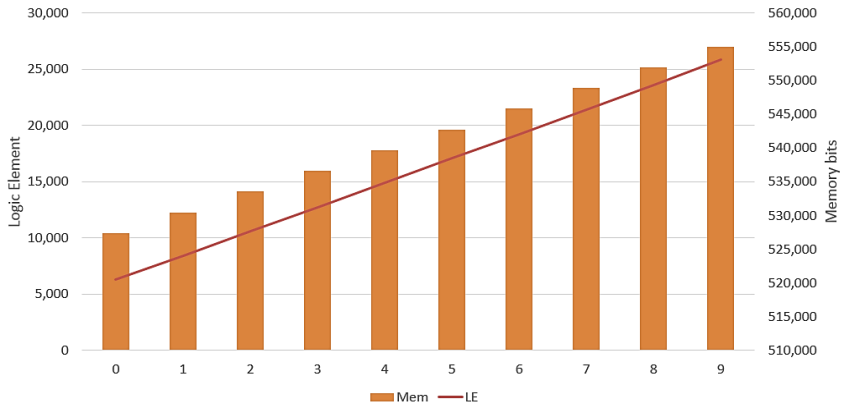

Figure 7. Effect of #parallel stages on LE/Mem bits

4.2. Comparison with Previous Works

There are two major works on RAM-based TCAM built on FPGA being compared to this work, which are [17,18]. The difference is that [18] was built with narrow TCAMs whereas UE-TCAM was built with large RAM blocks. The proposed TCAM for LPM application cannot be directly compared with the existing works [17,18] because their works were implemented on different FPGA families. Thus, UE-TCAM is built on the targeted Cyclone IV FPGA in this work. Table 1 shows the comparison with previous works. The UE-TCAM built has a maximum frequency of 179.28 MHz and latency of 3 cc. When update logic and LPM application are added, the frequency becomes 126.66 MHz with 12 cycle latency. When parallel LPM logic is added to the TCAM, the frequency decreases to 106.52 MHz at 10 cc latency due to additional logic added for controlling parallel LPM logic.

Table 1. Comparison with Existing Works

Work	FPGA Family	Max Clock Rate (MHz)	Latency (cycles)	Memory Bit	Resource Usage	Update Cycle	Throughput (Mega packet per second, Mpps)	Application
Jiang [18] (1024×150)	Xilinx Virtex-7	150	4	272 BRAM (~36 kbit)	-	33	150	No
Jiang [18] (1024×150)	Xilinx Virtex-7	-	-	272 BRAM (~36 kbit)	37.556 reg	33	199	Access Control List (ACL)
Ullah et al. [17] (512×36)	Xilinx Virtex-6	201.78	2	32×36K (~1 Mbit)	1.583 LUT	-	100.89	No
Rework of UE-TCAM [17] (512×32)	Cyclone IV E	179.28	3	524.288	1.644	-	59.76	No
Proposed TCAM (512×32)	Cyclone IV E	126.66	12	527.360	6.233	257	10.555	LPM
Proposed TCAM with parallelism (512×32)	Cyclone IV E	106.52	10	555.008	25.714	257	106.52	LPM

5. Conclusion

In this paper, a RAM-based TCAM is proposed on targeted FPGA based on the UE-TCAM architecture [17]. An update logic has been added for performing the update operations. Parallel LPM logic has been proposed with improved throughput for LPM application. In the proposed TCAM, the lookup latency depends the number of matching addresses for prefix comparison. Apart from the proposed TCAM architecture for LPM application, there is still some rooms for improvement. The TCAM architecture can be divided into different blocks to store for contents according to prefix length. Doing so could reduce the processing time of the LPM logic. A smaller number of matching vectors will be processed by LPM logic. Data of different prefix length could be stored with RAM block of different sizes to reduce the overall memory size for building TCAM. This concept is about similar to the DR-TCAM in [26]. Lastly, depending on the TCAM application, the input data could be compressed before being stored in RAM block to further reduce TCAM memory usage.

References

[1] Lockwood JW, McKeown N, Watson G, Gibb G, Hartke P, Naous J, Raghuraman R, Luo J. *NetFPGA--an open platform for gigabit-rate network switching and routing*. Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education (MSE'07), San Diego, 2007: 160-161.

[2] Zilberman N, Audzevich Y, Kalogeridou G, Manihatty-Bojan N, Zhang J, Moore A. NetFPGA: Rapid prototyping of networking devices in open source. *ACM SIGCOMM Computer Communication Review*. 2015; 45(4): 363--364.

[3] Srinivasan V and Varghese G. Fast address lookups using controlled prefix expansion. *ACM Transactions on Computer Systems (TOCS)*, 1999; 17(1): 1–40.

[4] Kobayashi M, Murase T. and Kuriyama A. *A longest prefix match search engine for multi-gigabit IP processing*. Proceedings of the 2000 IEEE International Conference on Communications (ICC 2000). New Orleans. 2000: 1360–1364.

[5] Li W, Li X, Li H. *MEET-IP: Memory and energy efficient TCAM-based IP lookup*. Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN). Vancouver. 2017: 1-8.

[6] Yang J, Jiang L, Bai X, Peng H, Dai Q. *A High-Performance Round-Robin Regular Expression Matching Architecture Based on FPGA*. Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC). Natal. 2018: 1--7.
[7] Jayashree S and Shivashankarappa N. *Deep packet inspection using ternary content addressable memory*. Proceedings of the 2014 International Conference on Circuits, Communication, Control and Computing (I4C), Bangalore. 2014: 441–447.
[8] Yang J, Jiang L, Bai X, Dai Q. *High Performance Regular Expression Matching on FPGA*. Proceedings of the 2017 International Conference on Collaborative Computing: Networking, Applications and Worksharing. Edinburgh. 2017: 541--553.
[9] Wang K. and Hengkui W. *TCAM-PC: Space-efficient TCAM-based packet classification with packet-forwarding-rate constraints*. Proceedings of the 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI). Qingdao. 2015. 1:260–264.
[10] Qu YR, Prasanna VK. High-performance and dynamically updatable packet classification engine on FPGA. *IEEE Transactions on Parallel and Distributed Systems.* 2016; 27(1): 197-- 209.
[11] Zerbini CA, Finochietto JM. *Performance evaluation of packet classification on FPGA-based TCAM emulation architectures.* Proceedings of the 2012 IEEE Global Communications Conference (GLOBECOM). Anaheim. 2012: 2766—2771.
[12] Nakahara H, Sasao T, Matsuura M, Kawamura Y. *The parallel sieve method for a virus scanning engine*. Proceedings of the 2009 12th IEEE Euromicro Conference on Digital System Design, Architectures, Methods and Tools (DSD 2009). Patras. 2009: 809--816.
[13] Aldwairi M, Flaifel Y, Mhaidat K. *Efficient Wu-Manber pattern matching hardware for intrusion and malware detection*. Proceedings of the 2018 IEEE International Conference on Electrical, Electronics, Computers, Communication, Mechanical and Computing (EECCMC). Vellore. 2018.
[14] Onizawa N, Gross WJ, Hanyu T. *A low-energy variation-tolerant asynchronous TCAM for network intrusion detection systems*. Proceedings of the IEEE 19th International Symposium on Asynchronous Circuits and Systems (ASYNC), Santa Monica. 2013: 8–15.
[15] Ricart-Sanchez R, Malagon P, Alcaraz-Calero JM, Wang Q. *Hardware-Accelerated Firewall for 5G Mobile Networks*. Proceedings of the IEEE 26th International Conference on Network Protocols (ICNP). Cambridge. 2018: 446--447.
[16] Nakamura Y, Sawaguchi S, Nishi H. Implementation and evaluation of an FPGA-based network data anonymizer. *IEEJ Transactions on Electrical and Electronic Engineering*. 2017; 12: S134--40.
[17] Ullah Z, Jaiswal MK, Cheung RC, So HK. *UE-TCAM:An ultra efficient SRAM-based TCAM*. Proceedings of the 2015 IEEE Region 10 Conference (TENCON 2015). Macau. 2015: 1--6.
[18] Jiang W. *Scalable ternary content addressable memory implementation using FPGAs*. Proceedings of the 2013 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). San Jose. 2013; 71–82.
[19] Hager S, Bendyk D, Scheuermann B. *Partial reconfiguration and specialized circuitry for flexible FPGA-based packet processing.* Proceedings of the 2015 International Conference on ReConFigurable Computing and FPGAs (ReConFig). Riviera Maya. 2015; 1--6.
[20] Ullah Z, Jaiswal MK, Cheung RC. E-TCAM: an efficient SRAMbased architecture for TCAM. Circuits, *Systems, and Signal Processing.* 2014; 33(10): 3123–3144
[21] Ullah Z, Ilgon K, Baeg S. Hybrid Partitioned SRAM-Based Ternary Content Addressable Memory. *IEEE Transactions on Circuits and Systems I*. 2012; 59(12): 2969–2979.
[22] Jiang W, Prasanna VK. Data structure optimization for power-efficient IP lookup architectures. *IEEE Transactions on Computers*. 2013; 62(11): 2169--2182.
[23] Qian Z, Margala M. *Low power RAM-based hierarchical CAM on FPGA*. Proceedings of the 2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig). Cancun. 2014; 1–4.
[24] Ahmed A, Park K, Baeg S. Resource-efficient SRAM-based ternary content addressable memory. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. 2017; 25(4):1583--7.
[25] Huang K, Chen X. *A power-efficient approach to TCAM-based regular expression matching*. Proceedings of the 27th IEEE International Conference on Computer Communication and Networks (ICCCN). Hangzhou. 2018: 1-9.
[26] Yang BD. Low-Power Effective Memory-Size Expanded TCAM Using Data-Relocation Scheme. *IEEE Journal of Solid-State Circuits.* 2015; 50(10): 2441–2450.