Hindawi
Mobile Information Systems
Volume 2020, Article ID 7641073, 8 pages
https://doi.org/10.1155/2020/7641073

Research Article

A CRC-Based Classifier Micro-Engine for Efficient Flow Processing in SDN-Based Internet of Things

Mahdi Abbasi,[1] Navid Mousavi,[1] Milad Rafiee,[1] Mohammad R. Khosravi,[2,3] and Varun G. Menon[4]

[1]*Department of Computer Engineering, Engineering Faculty, Bu-Ali Sina University, Hamedan 65178-38695, Iran*
[2]*Department of Computer Engineering, Persian Gulf University, Bushehr, Iran*
[3]*Telecommunications Group, Department of Electrical and Electronic Engineering, Shiraz University of Technology, Shiraz, Iran*
[4]*Department of Computer Science and Engineering, SCMS School of Engineering and Technology, Ernakulam 683582, Kerala, India*

Correspondence should be addressed to Mahdi Abbasi; abbasi@basu.ac.ir

Received 14 December 2019; Accepted 3 April 2020; Published 18 May 2020

Academic Editor: Ali Kashif Bashir

In the Internet of things (IoT), network devices and mobile systems should exchange a considerable amount of data with negligible delays. For this purpose, the community has used the software-defined networking (SDN), which has provided high-speed flow-based communication mechanisms. To satisfy the requirements of SDN in the classification of communicated packets, high-throughput packet classification systems are needed. A hardware-based method of Internet packet classification that could be simultaneously high-speed and memory-aware has been proved to be able to fill the gap between the network speed and the processing speed of the systems on the network in traffics higher than 100 Gbps. The current architectures, however, have not been successful in achieving these two goals. This paper proposes the architecture of a processing micro-core for packet classification in high-speed, flow-based network systems. By using the hashing technique, this classifying micro-core fixes the length of the rules field. As a result, with a combination of SRAM and BRAM memory cells and implementation of two ports on Virtex®6 FPGAs, the memory usage of 14.5 bytes per rule and a throughput of 324 Mpps were achieved in our experiments. Also, the performance per memory of the proposed design is the highest as compared to its major counterparts and is able to simultaneously meet the speed and memory-usage criteria.

1. Introduction

Our world is connected by Internet of things (IoT). In the past few years, the considerable growth of network bandwidth and development of hardware technologies, especially in mobile communications, have led to a significant increase in the speed of communication lines of this worldwide network [1, 2]. That is, the speed of communication lines is reached to higher than "terabits per second." The SDN paradigm aims to achieve good performance in managing networks by accelerating routers and switches to process the packets with the rate of network links [3, 4]. Making SDN flexible enough to satisfy the different requirements of heterogeneous IoT applications is desirable in terms of software-defined IoT (SD-IoT) [5, 6]. For this purpose, network devices are equipped with a new mechanism, naming packet classification, which lets them to be flow-aware. That is, the network device, first classifies the incoming packets into predefined flows according to a set of filters, then any further processing is done accordingly. Therefore, a variety of packet processor devices including routers, firewalls, intrusion detection systems, account management systems, and network management systems use packet classification [1, 7–10]. That is, a number of important network management functions such as access control, quality of service provisioning, firewall, traffic policing, and policy-based switching make use of packet classification.

There are five fields in a typical classification rule including source and destination IP addresses (SA and DA), source and destination port numbers (SP and DP, respectively), and protocol (PT). SA and DA are address prefixes, SP and DP are number ranges, and the PT field may be either a specified value or a wildcard. The order of rules in a rule set determines their priority. The last rule is the default rule in which all the five fields are equal to the wildcard. If an incoming packet matches more than one rule, the action corresponding to the rule with the highest priority is performed. A description for packet classification algorithms is found in [1].

Classification of Internet packets in network devices is conducted through either software-based or hardware-based approaches. Considerable time overload of software-based methods makes them less popular among network equipment manufacturers [11]. On the other hand, there is a widespread tendency towards hardware-based methods and their higher throughput rate and lower delay [12]. Hardware-based implementation of packet classification algorithms may categorized into two groups. The first group consists of algorithms based on parallel search in the content addressable memory (CAM) chips Z-TCAM [13], E-TCAM [14], and ZI-CAM [15]. In spite of their relatively high speeds, the use of ternary memories in these algorithms leads to disadvantages such as excessive power consumption, lower speed than other memory cells, lack of scalability, undue consumption of chip resources, and high prices. The second group consists of algorithms such as decision tree, decomposition tree, geometric space, field encryption, and similar methods which are realized on programmable hardware devices like ASIC or FPGA.

The main challenge in designing hardware-based methods is increasing the ratio of throughput to design cost. The throughput of a classifier is the number of packets that are classified in unit time. The required memory space is the main indicator of the system design cost. To reach this optimal point, we propose a micro-core that lowers memory consumption and simultaneously increases the classification throughput. The chief contributions of this paper are as follows:

(1) The proposed micro-core uses SRAM and BRAM cells, which allow for dual-port implementations.

(2) The proposed engine does not use any ternary content addressable memory. Instead, it encodes all of the prefixes by a cyclic redundancy check (CRC) code.

(3) Implementing the proposed classifier on Virtex®6 FPGA shows that the memory cost reduces to 14.5 bytes per rule, and simultaneously the throughput of the classifier reaches 324 Mpps. This result confirms the superiority of the proposed architecture to its counterparts.

The rest of this article is organized as follows. In Section 2, the related works on hardware-based packet classification systems are reviewed. The proposed microclassifier architecture is explained in Section 3. The performance evaluation of the proposed architecture is presented in Section 4 after introducing the metrics. Finally, conclusions and directions for future research are discussed in Section 5.

2. Related Work

So far, a wide variety of hardware-based classifier architectures have been proposed for packet classification. All of them attempt to increase the throughput and decrease the memory usage. The CAM-based classifier architectures benefit from the parallel search property of CAM modules but suffer high implementation costs and high levels of the consumption power. In [16, 17], two of the most recent architectures are proposed. They utilize a pipelined decision tree algorithm and a ternary memory, respectively. The architecture proposed in [16] has achieved a throughput of 103 Gbps, which is the highest among all the works mentioned here. However, depending on its hardware parameters like the number and length of pipelines on the distribution of the values of the classifier's rule fields, any updating requires reconfiguration of the architecture. On the other hand, the memory usage of this architecture is 63.5 bytes per packet. In [17], the researchers used ternary memories of the size $52*144$ and were able to reduced memory usage down to 18 bytes per rule; however, their maximum throughput was as low as 38 Gbps. The architectures proposed in [18, 19] could achieve a throughput of 100 Mpps while keeping the memory usage at 23.5 and 17.4 bytes, respectively. The focus of the architecture in [18] is on the rule search, and it does not address the issue of longest prefix matching (LPM). The architecture proposed in [19] adopts a TCAM-based approach to packet classification. Its major drawback is linear growth of TCAM usage is proportional to the number of rules that increases consumption of chip resources and power. Implementation of a merge algorithm based on decomposition tree in [20] achieves a throughput of 94 Gbps (amounting to 147 Mpps, given that each packet is 40 bytes). The study does neither provides the memory usage nor suggests any solution for updating rules.

Some classifiers like that presented in [21] use a special model to accelerate accessing the memory containing the rules, which in turn raises their memory consumption. Pipelined implementation of packet classification algorithms seeks appropriate solutions to reduce the number of pipeline stalls and the required memory space. For example, in the pipelined packet classifier of [22], the memory consumption varies from 16 to 24.5 bytes per rule.

To overcome the abovementioned disadvantages, we propose a packet-classifying micro-core with low memory consumption and high throughput. The proposed micro-core makes use of SRAM and BRAM cells, which allow for dual-port implementations. Processing based on cyclic redundancy check (CRC) codes in the internal structure of the micro-core without any need for ternary memories reduces the consumption of FPGA hardware resources and the time required for memory access in this classifier.

3. Proposed Architecture

This section explains the architecture of the proposed classifier which is aimed at increasing packet classification speed. Underlying the architecture are two principles: first, making use of BCAM memory in prefix matching and, second, using hash codes to reduce memory usage.

In this classifier, a set of processing micro-cores act like a CAM, each one storing the information about one rule field of the rule set. The architecture is shown in Figure 1. In this architecture, n micro-cores are defined for each of the fields used for packet classification, where n is the number of rules in the classifier representing the number of micro-cores per field.

The incoming packets are classified as follows: First, the packets are transmitted through a shared bus to the micro-core unit. As soon as a packet enters a micro-core, the fields of source and destination IP addresses of the header are read by Parallel Hash Calculator. Next, in a parallel manner and proportional to the length of the value of the Prefix register, CRC-16 generation process is performed on the input address and the result is stored in the Temp register. Each micro-core has a control unit that, in addition to controlling the function of the micro-core, manages the generation process as well as the process of matching the hash code generated and stored in the Temp register against the hash code of the prefix field of the corresponding rule of the micro-core in the Hash-of-rule register. If the hash code of the incoming packet header matches the hash code in the micro-core, the Adder will add one to the variable stored in the Rank register. Moreover, in the case of correct matching, the one-bit flag Match and the corresponding bit of the micro-core in the n-bit register Packet Matching will be set. If matching fails, these bits will be reset by default. The Packet Matching register is used to record matches or mismatches in other micro-cores. In this register, any bit with a value of one denotes a match and any bit with a value of zero denotes a mismatch in the micro-core corresponding to a field that is being searched. After matching all fields, the results are written to the bits corresponding to each field in the Packet Matching register. Next, the result of logical AND operation on all Packet Matching registers is stored in Matched Vector. Finally, a prioritized decoder selects the matching rule with the highest priority.

As seen in Figure 1, the classifier consists of a set of processing micro-cores. In the following, we shall discuss the internal architecture of the micro-cores that is illustrated in Figure 2. Also, the length of each register of the micro-cores is shown in the bottom of Figure 2.

The main body of the micro-core is composed of two modules, i.e., CRC Calculator and Controller. The Controller module is responsible for management of all control lines, inputs, and outputs. Operations in this unit include management of selection lines, injection, and updating of registers. In fact, this unit is the decision-maker of the micro-core, which is separately controlled by the main controller of the classifier. In other words, the functioning of the micro-cores is not disrupted by updating and changing one of the micro-cores.

The second important module of the micro-core that bears a major part of its processing load is CRC Calculator. Figure 3 shows the function of this module. It receives the incoming packets and calculates their hash code in parallel (Line 1 of Algorithm 1). For this purpose, each IP address along with the corresponding prefix which has been already stored in the Prefix register enters the module. Next, a hash code is computed using them and sent to the Controller for the purpose of matching. Implementation of this module consumes 40 out of 204000 LUTs on Virtex-6.

The micro-core has 7 input pins and 2 output pins. Table 1 lists the input and output ports of the micro-core along with their length in bits as well as their description. A micro-core is composed of registers, hash generator modules (CRC Calculator), and a controller.

One set of the input pins of the micro-core is named "Select," which determines the operation mode of the micro-core (Line 2 of Algorithm 1). In fact, this pin is responsible for management of the micro-core's functioning. It is connected to a two-bit bus which is used to address different modes of the micro-core function as described below (also, see Table 2):

Mode 0: the micro-core performs its main task, which is the packet classification (Lines 3–7 of Algorithm 1).

Mode 1: when the address from the Address port is identical with that in the Address register, the information in the selected micro-core is updated by means of the information from the Prefix and Rules ports (Lines 9–11 of Algorithm 1).

Mode 2: each micro-core sends the information related to its rank into a shared output bus. This operation is aimed at selecting the best candidate for being removed by the classifying controller. The central controller stores the number of the classifier with the lowest rank to update the rules (Lines 12-13 of Algorithm 1).

Mode 3: the majority of the existing classifiers do not offer a dynamic solution for updating rules and perform this task by reconfiguration of the chipset. However, given its low-rank feature, our proposed architecture enables us to update the rules under certain conditions. It is easy to inject new rules without changing the order of the existing rules. The last mode is Select(1, 1), which is used to initialize the micro-cores by injecting rules into each micro-core (Lines 14–17 of Algorithm 1).

Before injecting the packets into a processing micro-core, the rules are injected. In this step, the rules that have been converted to hash codes are stored in the Hash Code register. Prefix and Address registers keep the prefix length and the address of the processing core of the rules. Flag belongs to the internal controller which manages the input/output operation inside the micro-core so that the acts of processing the input packets would not overlap. Rank register has a length of 32 bits and holds the correct matches between the incoming packets and the matched field in the micro-core. For each correct match in the micro-core, one is added to the value of this register. This is a criterion used for updating and removing rules in other processing micro-cores. Thus, the micro-core with the lowest rank is selected for removing and updating. In fact, in this classifier, a lower rank is indicative of decreased use of the rule in the rule set.

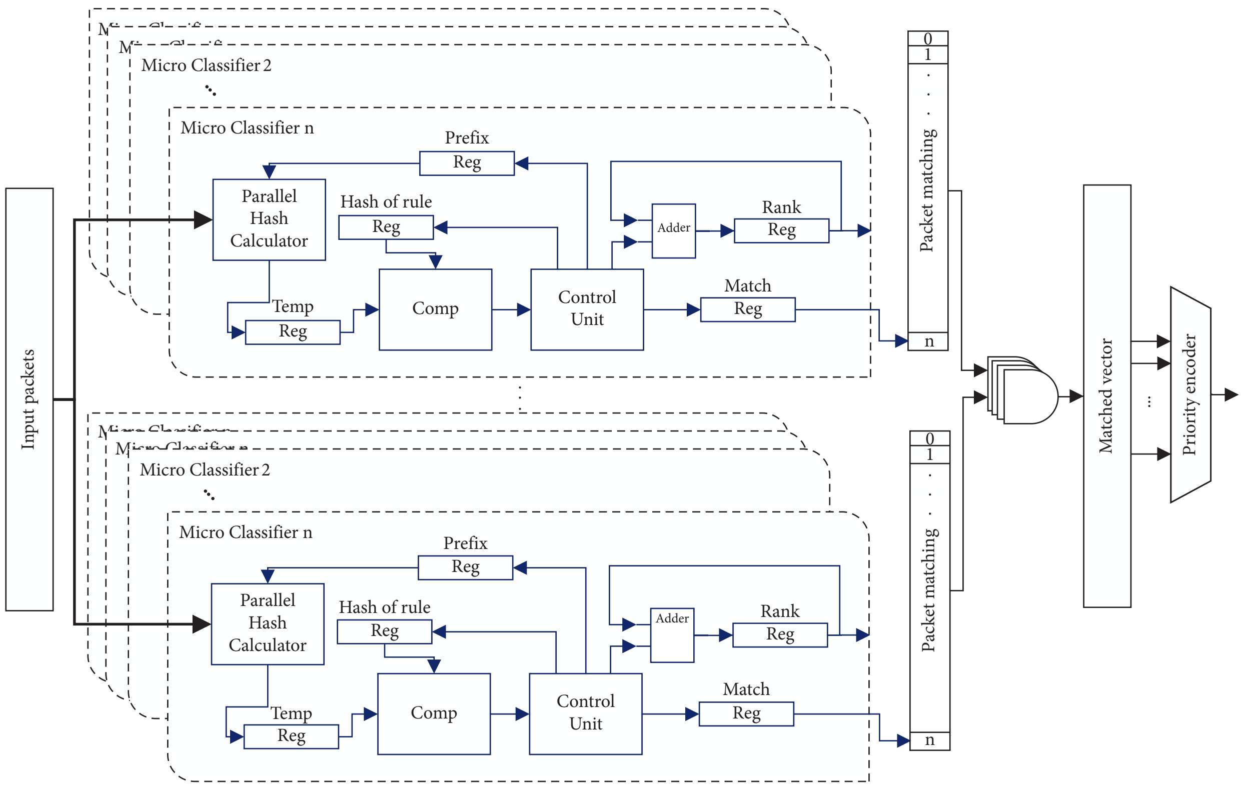

FIGURE 1: Proposed architecture for a packet classifier.

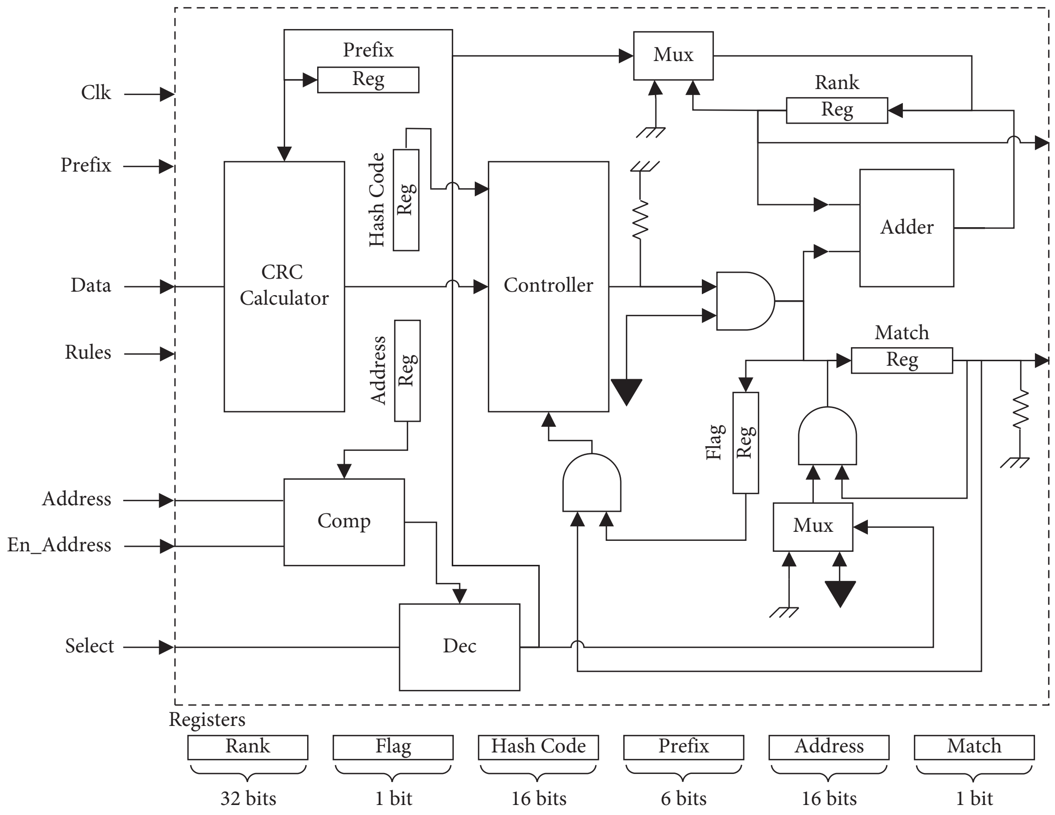

FIGURE 2: Proposed micro-core architecture.

```
    Input: Data, $\text{Input}_{\text{Prefix}}$, Rules, Select, $\text{Input}_{\text{Address}}$
    Output: rank_out, match
    Registers: Rank, Flag, $\text{Hash}_{\text{code}}$, Prefix, Address, Match
    Data: Packet P, CRC of Packet $P_{\text{CRC}}$
(1)   $P_{\text{CRC}} \longleftarrow$ Calculate CRC (P, $\text{Hash}_{\text{Code}}$)
(2)   Switch  Select
(3)     Case 00:  //classify Operation
(4)       if  $P_{\text{CRC}} == \text{Hash}_{\text{code}}$ then
(5)         Match $\longleftarrow$ 1,   Rank $\leftarrow$ Rank + 1, Flag $\leftarrow$ 1
(6)       Else
(7)         Match $\longleftarrow$ 0
(8)       end if
(9)     Case 01:  //update
(10)      Rank $\longleftarrow$ 0, Match $\leftarrow$ 0, Flag $\leftarrow$ 0
(11)      Prefix $\longleftarrow$ Input_prefix, $\text{Hash}_{\text{code}} \leftarrow$ Rules
(12)    Case 10:  //rank
(13)      Rank_out $\longleftarrow$ Rank
(14)    Case 11:  //set address
(15)      Address $\longleftarrow$ $\text{Input}_{\text{Address}}$, Prefix $\leftarrow$ Input_prefix
(16)      $\text{Hash}_{\text{code}} \longleftarrow$ Rules
(17)      Match $\longleftarrow$ 0, Flag $\longleftarrow$ 0
(18)  End Switch
(19)  If flag == 1 then
(20)    Flag $\longleftarrow$ 0, Match $\longleftarrow$ 0
(21)  End if
```

Algorithm 1: Implementation of the packet classifier micro-core.

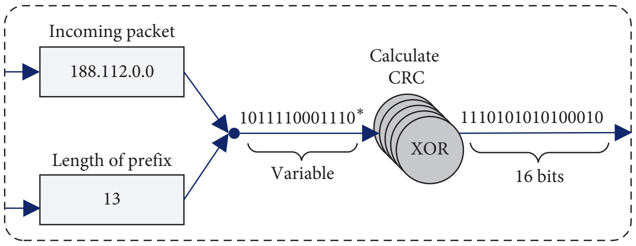

Figure 3: CRC Calculator module.

4. Implementation and Evaluation

In this section, the results of implementation of the proposed micro-core architecture are discussed. This architecture is implemented on a XC6VLX75T chip from Virtex-6 FPGA family with a Xilinx ISE 14.7 simulator using VHDL language. Experiments are done on a system with the characteristics mentioned in Table 3.

The evaluation criteria in this experiment are throughput and memory. Throughput refers to the number of packets classified in a second. Assuming a minimum of 40 bytes for each packet [20], we use Gbps instead of Mpps for measuring the throughput. Memory usage is measured in bytes for each rule in the classifier.

We use Classbench tool to generate rules and experimental packets in our experiments. Classbench runs on Linux platform and is used for generating rulesets with desirable distributions in the model of the geometric space of rules. It generates rules and corresponding headers by using a set of input distribution parameters [23].

The highest throughput achieved so far belongs to Chang [16], which is 103.53 Gbps. However, it is 0.150 Gbps less than our throughput rate. Also, the memory usage for storing the classifier's rules in that architecture is four times greater than in our method. In fact, Chang's architecture resembles traditional TCAM-based architectures in terms of memory usage.

Table 4 compares the proposed micro-core architecture with the existing architectures. With a clock frequency of 170 MHz, the processing time of each micro-core is in the worst case 6.2 nanoseconds per packet and power consumption is 118 mW. With a dual-port memory which can process two packets simultaneously, the proposed micro-core is able to process 324 million packets of at least 40 bytes in a second which amounts to a throughput of more than 100 Gbps. In this simulation, our architecture used 137 Slice registers and 182 search tables.

In Table 4, the proposed micro-core architecture is compared with major recently proposed counterparts in terms of throughput and memory usage per rule. From among these architectures, Jiang and Prasanna [19] and Irfan et al. [17] require the least amount of memory, i.e., 17.4 and 18 bytes per rule, respectively. With a required memory of 14.5 bytes per rule, our proposed micro-core outperforms these two architectures. The major reason behind the low memory usage in our method is that, in contrast to the two mentioned architectures, our classifier does not rely on TCAM and mask for matching operation.

In a more fare approach, Table 5 compares the proposed design with other designs with regard to the performance per memory [19]:

Table 1: The function of input pins.

Name	Length (bit)	Description
Clk	1	Clock for all micro-cores
Prefix	6	Length of Prefix
Data	32	Width of input line for incoming packets
Rule	16	Maximum length of the hash code of rules
Address	16	Address of the selected processing core
En-address	1	Activation of updating and configuration operations
Select	2	Mode of processing in selected core
Rank-out	32	Width of the bus which is shared with all micro-cores and is used for updating
Match	1	A flag for signalling match/mismatch in a micro-core

Table 2: Modes provided by select pin.

Select	Modes	Corresponding lines of Algorithm 1
00	Classifying packets	3–7
01	Updating rules and propertyof micro-cores	9–11
10	Outputting rank of micro-cores on BUS	12–13
11	Initializing the selected micro-core	14–17

Table 3: System specification.

Specifications	Processor
Name	Intel Core i7-3720QM
Clock speed	2600 MHz
L3 cache	6 MB
Main memory	16 GB DDR3
Operation system	Windows 10 enterprise 18.03, 64 bit

Table 4: Comparison of the performance of the proposed method with different architectures.

Reference	Throughput (Gbit/s)	Frequency (MHz)	Memory (byte)	Seri	Chip
Pus and Korenek [18]	100	125	23.5	Virtex5	LX110T
Chang and Chen [16]	103.53	161.76	63.5	Virtex-6	XC5VFX200T
Fiessler et al. [24]	92.16	180	NA	Virtex-7	XC7VX690T
Orosz et al. [25]	100	312	NA	Virtex-6	XC6VHX255T
Zhou et al. [20]	147 mil	NA	NA	Virtex-7	XC7VX690T
Irfan et al. [17]	37.3	259	18	Virtex-6	XC6VLX760
Jiang and Prasanna [19]	100	167	17.4	Virtex-5	XC5VFX200T
Our design	**103.680**	**170**	**14.5**	**Virtex-6**	**XC6VLX75T**

Table 5: Comparison of performance per memory for various systems.

Approaches	Throughput (Gb/s)	Memory (byte)	Efficiency (throughput/memory)	Chip
Orosz et al. [25]	100	156	0.64	XC6VHX255T
Our approach	**101.7**	**14.5**	**7.01**	
Irfan et al. [17]	37.3	18	2.072	XC6VLX760
Our approach	**75.83**	**14.5**	**5.229**	
Ganegedara and Prasanna [21]	407	156	2.660	XC6VLX760
Our approach	**75.83**	**14.5**	**5.229**	
Qi et al. [26]	73.9	46.4	1.592	XC6VSX475T
Our approach	**86.83**	**14.5**	**5.988**	
Pao and Lu [22]	108.8	18	6.04	XC6VLX75T
Our approach	**103.680**	**14.5**	**7.150**	

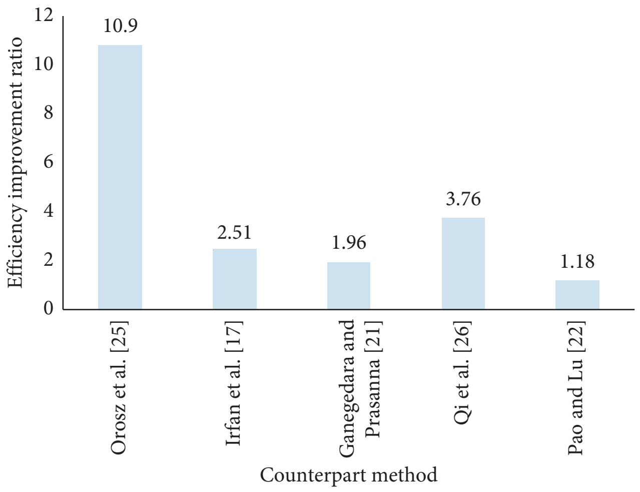

Figure 4: The ratio of efficiency improvement.

$$\text{efficiency} = \frac{\text{throughput (Gb/s)}}{\text{normalized memory (B/rule)}}. \tag{1}$$

For this purpose, the throughput as well as the memory consumption of the proposed architecture is measured on the chipsets that are used in the evaluation of the competitor designs. The memory usage per rule is always constant in the proposed design. Therefore, the performance per memory of the proposed design is the highest as compared to its major counterparts.

The ratio of superiority of the efficiency of the proposed method with respect to each counterpart is illustrated in Figure 4. Our comparisons suggest that the proposed architecture has considerably improved the throughput rate and memory usage of the Internet packet classification systems. The performance per memory of the proposed design is at least 18% and at most 990% better than the best and the worst designs, namely, Pao and Lu [22] and Orosz et al. [25].

5. Conclusion

In this paper, we proposed a new micro-core architecture for classification of Internet packets that is capable of being updated. The proposed architecture allows for adding or removing rules during processing and enjoys lower memory usage as well as higher throughput rate in comparison with other architectures. Our evaluations suggest that this micro-core can classify packets with a throughput of more than 103 Gbps, which amounts to about 324 Mpps. Another advantage of this architecture is that memory usage per rule is always constant. Therefore, the performance per memory of the proposed design is the highest as compared to its major counterparts. This achievement helps the proposed micro-core to avoid the problem of resource requirement in the longest prefix matching (LPM). A fruitful topic for future research would be to apply this inherent feature of the micro-core to the implementation of pipeline classifiers in which LPM is a great problem in pipeline processing.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computing Surveys (CSUR)*, vol. 37, no. 3, pp. 238–275, 2005.

[2] N. T. Le, M. A. Hossain, A. Islam, D.-Y. Kim, Y.-J. Choi, and Y. M. Jang, "Survey of promising technologies for 5G networks," *Mobile Information Systems*, vol. 2016, Article ID 2676589, 25 pages, 2016.

[3] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," *Future Generation Computer Systems*, 2019.

[4] M. Shakil, A. Fuad Yousif Mohammed, R. Arul, A. K. Bashir, and J. K. Choi, "A novel dynamic framework to detect DDoS in SDN using metaheuristic clustering," *Transactions on Emerging Telecommunications Technologies*, p. e3622, 2019.

[5] A. R. Bevi, P. Shakthipriya, and S. Malarvizhi, "Design of software defined networking gateway for the internet-of-things," *Wireless Personal Communications*, vol. 107, pp. 1273–1287, 2019.

[6] A. K. Bashir, R. Arul, S. Basheer, G. Raja, R. Jayaraman, and N. M. F. Qureshi, "An optimal multitier resource allocation of cloud RAN in 5G using machine learning," *Transactions on Emerging Telecommunications Technologies*, vol. 30, p. e3627, 2019.

[7] P. Gupta and N. McKeown, "Packet classification on multiple fields," *ACM SIGCOMM Computer Communication Review*, vol. 29, no. 4, pp. 147–160, 1999.

[8] K. Kang and Y. S. Deng, "Scalable packet classification via GPU metaprogramming," in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, March 2011.

[9] S. Shieh, F.-Y. Lee, and Y.-W. Lin, "Accelerating network security services with fast packet classification," *Computer Communications*, vol. 27, no. 16, pp. 1637–1646, 2004.

[10] G. Varghese, "Chapter 12-packet classification," in *Network Algorithmics*, G. Varghese, Ed., pp. 270–301, Morgan Kaufmann, San Francisco, CA, USA, 2005.

[11] J. Wee, J.-G. Choi, and W. Pak, "Wildcard fields-based partitioning for fast and scalable packet classification in vehicle-to-everything," *Sensors (Basel, Switzerland)*, vol. 19, no. 11, p. 2563, 2019.

[12] Y. R. Qu, S. Zhou, and V. K. Prasanna, "High-performance architecture for dynamically updatable packet classification on FPGA," in Proceedings of the Ninth ACM/IEEE Symposiumon Architectures for Networking and Communications Systems, pp. 125–136,IEEE, San Francisco, CA, USA, October 2013.

[13] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "Z-TCAM: an SRAM-based architecture for TCAM," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 1, pp. 402–406, 2014.

[14] Z. Ullah, M. K. Jaiswal, and R. C. C. Cheung, "E-TCAM: an efficient SRAM-based architecture for TCAM," *Circuits, Systems, and Signal Processing*, vol. 33, no. 10, pp. 3123–3144, 2014.

[15] 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, 2019.

[16] Y.-K. Chang and H.-C. Chen, "Fast packet classification using recursive endpoint-cutting and bucket compression on FPGA," *The Computer Journal*, vol. 62, no. 2, pp. 198–214, 2018.

[17] M. Irfan, Z. Ullah, and R. C. C. Cheung, "D-TCAM: a high-performance distributed ram based TCAM architecture on FPGAs," *IEEE Access*, vol. 7, pp. 96060–96069, 2019.

[18] V. Puš and J. Korenek, "Fast and scalable packet classification using perfect hash functions," in Proceedings of the ACM/SIGDA International Symposium on Field Programmable GateArrays, ACM, pp. 229–236, Monterey, CA, USA, February 2009.

[19] W. Jiang and V. K. Prasanna, "Field-split parallel architecture for high performance multi-match packet classification using FPGAs," in Proceedings of the Twenty-irst Annual Symposiumon Parallelism in Algorithms and Architectures, ACM, pp. 188–196,Calgary, Alberta, Canada, August 2009.

[20] S. Zhou, Y. R. Qu, and V. K. Prasanna, "Large-scale packet classification on FPGA," in Proceedings of the IEEE 26thInternational Conference on Application-pecific Systems, Architectures and Processors (ASAP), pp. 226–233, IEEE, Toronto, Ontario July 2015.

[21] T. Ganegedara and V. K. Prasanna, "StrideBV: single chip 400G+ packet classification," in Proceedings of the IEEE 13thInternational Conference on High Performance Switching andRouting,pp. 1–6, IEEE, Belgrade, Serbia, June 2012.

[22] D. Pao and Z. Lu, "A multi-pipeline architecture for high-speed packet classification," *Computer Communications*, vol. 54, pp. 84–96, 2014.

[23] D. E. Taylor and J. S. Turner, "Classbench: a packet classification benchmark," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 499–511, 2007.

[24] A. Fiessler, C. Lorenz, S. Hager, B. Scheuermann, and A. W. Moore, "HyPaFilter+: enhanced hybrid packet filtering using hardware assisted classification and header space analysis," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3655–3669, 2017.

[25] P. Orosz, T. Tóthfalusi, and P. Varga, "C-GEP: adaptive network management with reconfigurable hardware," in *Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 954–959, IEEE, Toronto, Ontario, Canada, May 2015.

[26] Y. Qi, J. Fong, W. Jiang, B. Xu, J. Li, and V. Prasanna, "Multi-dimensional packet classification on FPGA: 100 Gbps and beyond," in *Proceedings of the International Conference on Field-Programmable Technology*, pp. 241–248, IEEE, Beijing, China, December 2010.