Microprocessors and Microsystems 99 (2023) 104826

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

A parallel decision-making design for highly speedy packet classification

Midde Adiseshaiah [*], Maruvada Sailaja

Department of Electronics and Communication Engineering, Jawaharlal Nehru Technological University Kakinada, Kakinada, Andhra Pradesh, 533003, India

ARTICLE INFO

Keywords:
Dual port random access memory
Field programmable gate array
Packet classification
Packet scheduling
Software-defined networking

ABSTRACT

Packet classification is crucial in computer networks to increase network security because of developments in high-speed data communication. To support different network services including Quality of Service (QoS), security, and resource reservation, network packet classification is a crucial network kernel function. It became extremely challenging to classify arriving packets using the traditional packet classification algorithms at a decent pace due to the rapidly increasing size of rulesets and rule fields in current networks. In addition to hardware-based solutions, numerous contemporary software-based classification techniques have been put out to speed up packet classification. In general, it's critical to manage low latency, fast throughput, and higher energy efficiency with minimal memory needs while designing a packet classification method. In this paper, proposed the architecture makes use of the parallelism provided by current hardware technologies to classify numerous packets at once to improve throughput, reduce energy consumption, and significantly decrease the latency. The suggested high-speed parallel classification procedure uses two DPRAMs (DPRAM1 and DPRAM2) to support multiple read of the data. DPRAMs(Dual Port Random Access Memories) are preferred for stage memory because of their low latency performance, which enables quick rate packet lookup. The packet scheduling module receives the data from the DPRAM. The four successive packet splitter modules receive the packets in parallel. As a result, the packet classification throughput increased. To optimize throughput, and latency while maintaining low energy consumption, facilitating parallel packet classification architecture on FPGA is discussed in this work. Several parameters, including throughput, latency, energy efficiency, and memory requirements, will be evaluated and compared to traditional methods to assess performance. The classifier offers less latency of 51 ns and the lowest energy efficiency of 5.2 nJ, and its throughput may approach 802 million packets per second at the rate of 420 MHz clock frequency.

1. Introduction

Internet routers utilize packet classification to categorize packets into network flows. Packet classification depends on numerous packet headers fields that are provided in predefined filters. The term "flow" refers to a series of packets from a certain data source to a specific receiver. The term "traffic flow" or "packet flow" is more commonly used to describe a sequence of packets. A technique known as packet classification allows for the identification of flows by classifying arriving packets into various flows based on the contents of the header fields during a predetermined window of time [1]. Each incoming packet is examined by a set of rules to be recognized and classified into different flows [2]. An incoming packet is only accepted if it matches any rule in a rule set; otherwise, it is denied. Each flow can be handled differently when incoming packets are divided into various classes to separate the services based on user demand. Each program and service that a user requests require the same type of packets. Using a preset ruleset, the packet classification approach makes it possible to efficiently deliver the appropriate packets to the appropriate services. In order to recognize dangers and prevent unwanted access to the network, the packet classifier needs to be integrated into a number of services, such as firewalls, private virtual networks, cybersecurity, strategy navigation, congestion control, and quality of services [3,4]. Due to these technique's many advantages in contemporary communication, packet categorization is now a typical function among all types of security systems such as firewalls, internet gateways, and private virtual networks [5].

Packet classification plays a crucial role in selecting the appropriate course of action across all networking paradigms. The categorization of packets has long been considered a difficult offline task. Development of networking paradigms like software-defined networking (SDN), shown by Open Flow, and network function virtualization (NFV), in which the ability to characterize and change recommendations for inserting or

[*] Corresponding author.
E-mail address: adi.techster@gmail.com (M. Adiseshaiah).

https://doi.org/10.1016/j.micpro.2023.104826
Received 5 December 2022; Received in revised form 17 March 2023; Accepted 28 March 2023
Available online 5 April 2023

deleting packets has been a problem online [6,7].

A new enterprise networking system called SDN has been developed. SDN uses Open In order to manage network activity and through data plane as well as the control plane, flow is a flexible protocol and to separate the hardware-based data plane from the software-based network control. The flow table search is one of Open-kernel Flow's responsibilities. Significant issues in packet classification include handling large rule sets (up to 100 K rules), ensuring adequate outcomes, and allowing for dynamic updates.

Although there is software that can classify packets, it is inadequate for high-speed link functions [4]. Frequently, categorization is performed by software tools focuses just on protocol layers, IP addresses, or port numbers. For wire speed processing, it is not desirable for software solutions to provide inspection of many fields. For secure communication and wire-speed computing, hardware solutions are preferred, and categorizing packets can be done by looking at all packet header data. In a hardware packet classification system, several fields of an incoming packet are checked against each rule in a rule set. A ruleset may contain between one hundred and one thousand rules. The biggest issue with implementing a hardware solution is the need for a lot of memory to hold the rules. The on-chip memory of Field Programmable Gate Arrays (FPGA) is constrained, and using external memory is challenging. Yet, certain hardware solutions can achieve low latency and high throughput of 100/200/400 Gbps at the expense of memory.

For the prototype of packet classification systems, memory requirements to hold a large number of rules create a barrier and issue [2]. Every classifier rule is stored in descending order of priority, and actions are then done following that sequence.

Most network applications that require wired speed classification use hardware-based packet classification techniques like TCAM and hardware accelerators.

Ternary content addressable memories (TCAMs) are widely employed in modern hardware-based packet classifiers [10,11]. While having a very high throughput, TCAM-based solutions have a low energy efficiency because of the massively concurrent exhaustive search [8]. Recent studies [12,13], and [14] have suggested using FPGA technology as a more energy-efficient alternative platform for creating real-time network processing engines. Perhaps, FPGA-based packet classifiers could reach very high throughput. Yet there are still a lot of difficulties in building an FPGA-based packet classifier with such fast throughput and low energy consumption.

A packet sorting technique is a key component of many safety devices and functions used on the internet and in mainframe systems [9]. Due to unique computing methods and specific restrictions, a variety of packet classification methods have been developed to accomplish packet classification. The majority of strategies now in use might not be appropriate for hardware implementation. Below is a list of the main performance indicators that need to be taken into account while designing the algorithms and hardware architecture for packet classification systems [15].

The main performance metric used to assess packet classification algorithms is throughput. Fast processing is necessary for packet classification to support the Internet's current speeds of more than 40 Gbps [16]. To put it another way, an incoming IP datagram must be processed in 8 ns (assuming IP packets are at least 40 bytes in size). The hardware must be as simple as is practical to achieve this goal because complexity results in longer processing times [17]. For rule sets of average size, an FPGA-based packet classification engine can achieve very high throughput [2]. However, FPGA-based techniques frequently experience clock rate degradation as the number of packet header fields or the size of the rule set grows.

Another important parameter for a packet classification system is latency. Latency is the length of time a packet spends processing in a router. Latency and throughput are two different metrics. For instance, pipelined algorithms produce superior throughput results. The throughput value is determined by the processing time of one pipeline stage. On the other side, latency is the period between when a packet enters a pipeline and when it exits. In other words, latency is equal to the number of pipeline stages times the pipeline stage's processing time.

Energy or Power consumption is a key factor in packet classification design. It becomes a crucial issue as routers reach trillions of bits per seconds data rate. The number of rules used to categorize the inbounded packets will affects power efficiency. This is one of the factors considered while assessing the power effectiveness of the packet classifiers. The energy needed by the router to dissipate the enormous amount of heat produced by its components greatly contributes to running costs [8]. Because each router port incorporates equipment for packet classification and router searching, power consumption is becoming a more crucial evaluation factor [4].

Another crucial aspect of packet classification is memory usage. Researchers are currently looking for answers to complex rule sets. The type of categorization and the number of rules kept in the classifier determine how much memory is needed. Memory has emerged as a crucial hardware issue for supporting a high number of rules due to the FPGA's resource limitations [15].

It is a very critical task to balance high throughput and latency at low energy efficiency. To overcome these problems, proposed an efficient parallel packet classification design with DPRAM (Dual Port Random Access Memory). The proposed system is used to trade off between the throughput and latency with minimum energy consumption for optimized packet classification with high clock frequency.

The remaining of the paper is structured as follows: Section 2 provides a literature review and explanation of the various packet classification and design architectural techniques. In Section 3, explain the proposed parallel packet classification architecture on FPGA design. To illustrate the performance and characteristics, Section 4 discusses the experimental findings. The conclusion and upcoming efforts are addressed in Section 5.

2. Literature survey

Before now, numerous studies have been put forth to achieve packet categorization in ASIC and FPGA devices. By improving the algorithm's architecture, these various techniques aim to lessen the complexity of the design. The works that have been previously proposed to carry out the hardware realization of packet classification are described here.

On the FPGA, Yun R and Viktor proposed a 2-dimensional pipelined design for packet categorization that offers a high throughput and supports non-static modifications. Such a 2D array of Programmable Processing Elements (PEs) is arranged. The entire array is pipelined in both the horizontal and vertical directions. The length of the rule list affects how much memory is available overall. Even when (1) each packet header's width or (2) the set of rules is increased, the system still operates at a fast clock rate. The number of distinct values in every field of the rule set had a negligible effect on the design's overall speed. Because of the PEs' self-reconfigurability, the dynamic ruleset modifications can be made during a move with little data loss [18]. Several bit vector algorithms execute individual lookups for each field and return the results' composition as bit vectors. Although concentrating on high-speed packet classification, the bit vector technique neglects to maintain proper memory utilization. The difficulty of design increases when utilizing big rule sets [19]. An Xnor-BV based packet sorting engine was proposed by Ausaf Umar Khan et al. In this protection tool and Internet service provider system. The Internet Protocol (IP) and Protocol layer address header fields of packets are sorted by established rules using the xnor gateway. Low latency and high throughput are supported [20].

The partition sort developed by Sorrachai et al. is a hybrid system which combines scalability, quick updates, and fast packet classification with decision tree properties. Rule set sorting is presented in Partition Sort, and it offers two key advantages. It produces a lot fewer divisions, to start with. The use of multidimensional interval trees [21] enables logarithmic classification and upgrading time for each split sorting guideline. An SRAM-based packet categorization hardware architecture that imitates the functionality of TCAM(Ternary Content Addressed Memory) was investigated by Weiwen Yu et al. The data in the packet's header is encoded using the prefix inclusion coding technique. Encoded rules are converted to SRAM-based matched units using a bit-selection technique. A portion of the incoming key's bits are used to access a comparison rule's SRAM address [22]. TCAMs (ternary content addressable memory), which K. Sakthi and P. Nirmal Kumar devised, are frequently utilized for system modules to perform packet classification. Among the uses are packet transfer, safety, and the construction of program based procedures. TCAMs are frequently used for systems administration as standalone devices or as a secured inventive block based on application-explicit integrated circuits. Field-programmable gate arrays do not support TCAM squares. Contrarily, due to their adaptability, FPGAs are preferred for SDN, and the majority of FPGA makers offer SDN expansion sets. While considering TCAM implementation in FPGA, it is essential to mimic TCAMs utilizing the interpretation blocks which were retrieved in the FPGA. Recently, several methods to simulate TCAMs with FPGAs have been put forth. The smallest of them execute TCAMs using the bulk of memory squares accessible in current FPGAs [23].

Range tree-linked list hierarchical search structure, which Oguzhan Erdem and Aydin Carus suggested for packet categorization, inhibits backpedaling by getting around memory restrictions in Stage 1 using a range tree and Stage 2 with an associated data structure. The value-coded trie prevents backtracking and solves the memory inefficiency issue by having a set-size bin at each node. Instead of placing a regular binary trie during a new unoccupied bin, a value-coded trie uses the -branch attribute. A rule classification technique is designed to separate an input guideline into groups based on field characteristics to conserve memory. To support the specified data formats, high-throughput parallel, and pipelined architectures using Field Programmable Gate Arrays (FPGAs) were built [24]. Rashid Hatami et al. proposed a tree-based approach to the Specified Range-Based Ternary Searching Trees (RTST) flow table search. With SDN switches, RTST focuses on flow tables to provide faster searching across flows. Provide RTST implementation a parallel multiprocessor design that has excellent efficiency and low latency [25].

A revolutionary design for a configurable packet processor that handles all three tasks needed by switches—packet data parsing, processing, and classification—was developed by Abbas Yazdinejad et al. at the network data plane. The Register Transfer Level (RTL) on the FPGA is defined using High-level P4 programming to carry out this procedure. Through pre-processing in the parsing graph, detecting traffic flow, and integrating the hybrid controlled flow structure to data processing, the suggested architecture employs a pipeline approach at the SDN data plane to enhance processing speed [26].

For the hash-based exact match categorization of several data packages within a single clock duration, Kekely et al. introduced a parallel hardware architecture. There is less requirement for memory replication with this architecture. The basic idea behind the proposed design is to make use of the fundamental memory organization structure present in all modern FPGAs, where hundreds of distinctly distributed memory pieces are accessible and may each be retrieved separately. Even in the lack of wholly duplicated memory resources in matching tables, the throughput of identical numerous packets per clock cycle is maintained [27].

The potential for improving policy-specific, built-in network processing circuits was examined by Hager et al. Use the example of router forwarding information bases to demonstrate that FIB-specific circuits need noticeably fewer logic resources than equivalent generic forwarding circuits (FIBs). Obtain reliable forwarding engines with low latency that can update their corresponding circuitry as needed. Together with FPGAs' limited reconfiguration capabilities [28].

A decomposition-based method of packet categorization proposed by Qu et al. can handle large rule sets composed of numerous different packet header data. Our approach simultaneously searches the fields of such input packet header using range-tree and hashing. The rule ID sets are used to represent the entire fields' restricted outcomes, then effectively combined to generate the match result [29].

Jamil et al. proposed an approach that, by developing an effective data structure for many rules sets with numerous fields, shows a scalable learning-based packet classification engine. This approach entails breaking down fields into smaller groups and utilizing deep reinforcement learning to construct distinct decision trees for those groups [30].

Ponnuswamy et al. proposed a method that, using a range bit-vector encoding technique, an effective range-enhanced packet classification (REPC) module is created. This design offers a special way to keep the precomputed values in memory. To match the packets to the appropriate header fields, the REPC also offers a range to prefix features [31].

Procedures for updating online trie-based algorithms should be improved. In contexts where the match fields of the rules are variable and the rules are dynamic, Li et al. suggested a SplitTrie is a packet classification technique that has been suggested for use with Software Defined Network (SDN) switches. This method's main flaw is that it struggles in rule sets with straightforward rules [32].

Cheng chi et al. proposed a Fast Packet Classification Method Based on Information Gain Ratio (PCMIgr), a heuristic technique that enhances The attribute selection process that differs from the conventional classification method using decision trees. The normal problem of "rule replication" that occurs during the construction of a regular decision tree is avoided since every single leaf node of the decision tree is guaranteed to be connected to just one rule. The requirement for computation and memory is effectively decreased [33].

Ponnusamy Vasaki et al. suggested IoT wireless intrusion detection, the goal of this research was to examine the properties of current datasets, including KDD-Cup and NSL-KDD, and determine whether they were appropriate for use with wireless IDS. In contrast to a wired network, the data packets recorded on a wireless network are the subject of this method's analysis of packet and flow properties [34].

To accommodate many-field rules, Lin et al. suggested a TCAM-based packet classification system. This method employs many TCAM access stages, each of which only compares a subset of the rules. Despite the possibility of more TCAM accesses, this technique allows for packet classification based on rules longer than TCAM words [35].

Nafis Irtija et al. design analyzes how well current System-on-Chip (SoC) designs perform concerning the control requirements related to running quantum gates using trapped-ion qubits, paying special attention to communication inside the SoC. This study primarily focuses on the data transmission latency and throughput of various high-speed on-chip techniques utilizing Xilinx multi-processor SoCs, including those that employ direct memory access. These are measured and assessed to establish a maximum estimate of the time needed to change a gate parameter [36].

The issue of balancing high throughput and latency with low energy efficiency is extremely important. It was suggested that an effective parallel packet classification design using DPRAM be used to address these issues (Dual Port Random Access Memory). The suggested solution uses a high clock frequency for better packet classification while minimizing energy consumption and balancing throughput and delay.

It is abundantly obvious from the discussion above that various works have previously proposed performing FPGA integration for packet classification. Many efforts are concentrated on lowering power consumption, increasing throughput, and reducing latency. The following is a statement of the work's primary goal:

1 To lower energy consumption and increase throughput on the chip, which uses less energy for high-speed operations with low latency.
2 To raise the maximum operating frequency with high-speed digital applications.

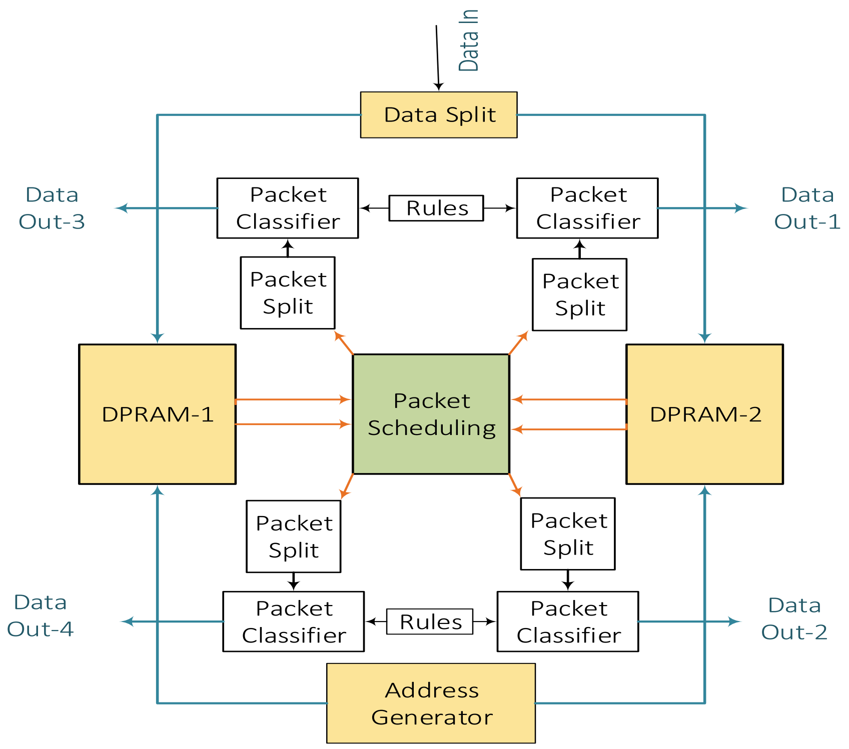

Fig. 1. Block diagram of the proposed technique for a dual-port RAM-equipped packet classification engine.

3 To develop a high-performance packet classification solution that balances throughput and latency while maintaining low energy consumption with considerable memory requirements.

3. Proposed method

The DPRAM (Dual Ported RAM) employed in this proposed method allows for multiple readings of the information. A information split unit is implemented to divide the information and send it to both DPRAM-1 and DPRAM-2 in turn. The packet scheduling device receives data from both DPRAMs, and the scheduling module is linked to a second four-packet split module. The four parallel successive packet splitter modules receive the packets last. The packet classifier unit is coupled to the specified rules. All 4 packet classifier modules work simultaneously to produce the final output that is classified, maximizing the packet classification's throughput.

3.1. Block diagram

The block diagram for the suggested technique, shown in Fig. 1, includes an address generator and a data split module. Data is sent to the packet scheduling module through two DPRAMs. To process the packets, four packet split modules and four packet classifiers are employed concurrently. This suggested module can classify four packets in one clock cycle. The classification process applies the rules, which are kept in the LUT tables. The classified packets are produced by the packet classifier.

3.2. Dual ported RAM(DPRAM)

DPRAM lets multiple reads or writes occur simultaneously, compared to single-ported RAM, which only allowed to retrieve in unit time. A sort of dual-ported dynamic RAM called VRAM (video RAM) enables the CPU to draw the image as its video circuitry follows it out of the display. The bulk of dual-ported RAM employs static RAM rather than VRAM. Most CPUs implement the processor registers as a modest

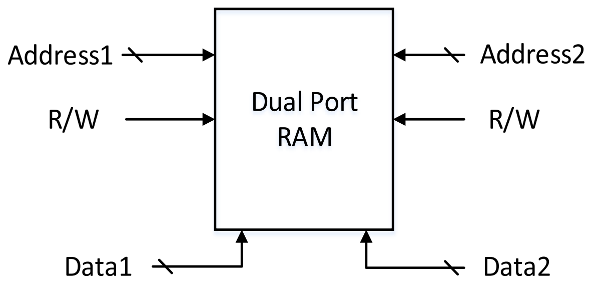

Fig. 2. Block diagram for the dual port RAM (DPRAM).

dual-ported or multi-ported RAM. Fig. 2 depicts the Dual-Ported RAM's internal structure.

3.3. Packet classifier

The important module that is utilized to carry out the packet classification process is the packet classifier unit (Fig. 3). The source address, a destination address, source port, and destination port will all be subject to different matching processes. Flip-flops, LUTs, and different logic gates make up this block.

The packet classifier's block diagram is shown in Fig. 3. It is a crucial module that is used in the classification process for packets. The source address, a destination address, source port, and destination port will all be subject to different matching processes. Flip-flops, LUTs, and different logic gates are included in the classifier block. This is one of the key blocks used in the classification of packets. The matching process is carried out independently for each of the incoming data's source address, a destination address, source port, and destination port. The XNOR logical gates were used for the 2-input matching process, which produces an output '1′ if the two inputs are the same or else it produces '0′. Internal similarity is achieved in this stage and overall similarity conditional operation will be performed. Resetting of address generators is done based on the output, which helps to avoid unnecessary clocking.

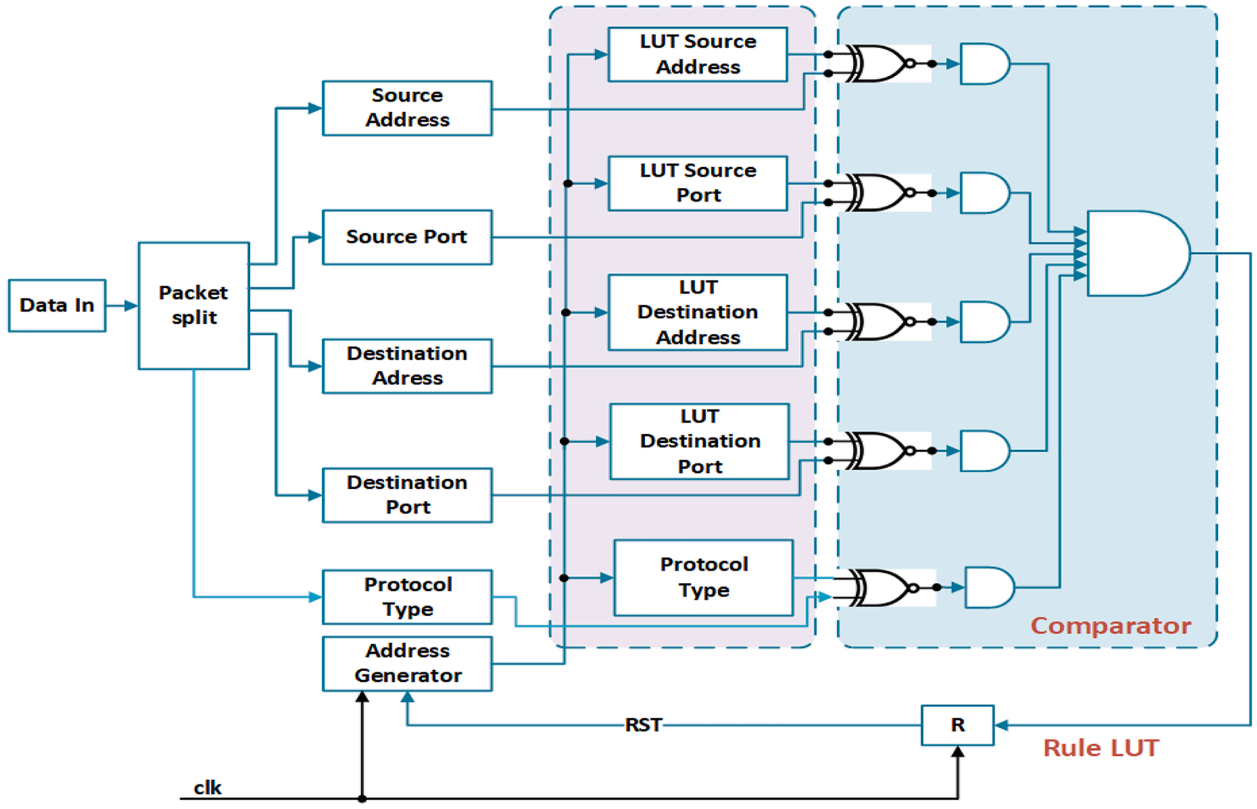

Fig. 3. Design of the module for packet classification.

Algorithm 1
Packet classification using dual RAM based classification technique.

```
   Input: Packets($d_{in}$) of 104-Bit length and N- Samples
   Rule LUT for Source Port (LUT_Sport),
   Destination Port (LUT_Dport),
   Source Address (LUT_Saddr),
   Destination Address (LUT_Daddr).
   Protocol Type (LUT_Ptype).
1  Read packet ($d_{in11}$ and $d_{in12,}$ $d_{in21}$ and $d_{in22}$) from $RAM_1$ and $RAM_2$
2  for n 1 to M do
3  [$Ptype_{m\ x\ n}$, $Saddr_{m\ x\ n}$, $Daddr_{m\ x\ n}$, $Sport_{m\ x\ n}$, Dport] = SplitPacket($d_{in\ m\ x\ n}$)
4  t1 = Match($Ptype_{m\ x\ n,}$ LUT_Ptype)
5  t2 = Match($Daddr_{m\ x\ n,}$ LUT_Daddr)
6  t3 = Match($Saddr_{m\ x\ n,}$ LUT_Saddr)
7  t4 = Match($Sport_{m\ x\ n,}$ LUT_Sport)
8  t5 = Match($Dport_{m\ x\ n,}$ LUT_Dport)
9
10 $t$ = t1 & t2 & t3 & t4
11 if($t$==1)
12 $y_{out} = d_{in}$
13 else
14 $y_{out} = 0$
15 endif
16 end for
```

The new matching process is initiated by resetting the LUT address generator. Then the RAM address is incremented by 1 and a new value is read into the RAM. This helps to reduce the dynamic power consumption of the entire FPGA operation in real time.

The source address, source port, destination address, and destination port of incoming data packets are divided using the splitter, as was detailed in section A. The XNOR gate is utilized to carry out the matching procedure, as shown in Fig. 2. As an illustration, two identical inputs will result in an output of 1, but two distinct inputs will result in an outcome of 0 (Algorithm 1).

3.4. Packet scheduler module

For packet scheduling, many queueing procedures can be applied. The procedure for utilizing a packet scheduler to schedule the packets is shown in Fig. 4. The bounded delay constraint is crucial to take into account in the discipline of queuing. whenever the router transmits the first packet that it receives. Note that FIFO queuing is often referred to as FCFS queuing. The most straightforward software-based router is FIFO, which is easy to construct with minimal impact on system overhead. The benefit of a FIFO queue is that it gives packets passing through the router a known latency. When c is its link speed and B is its extreme buffer size, the D may be calculated as follows.

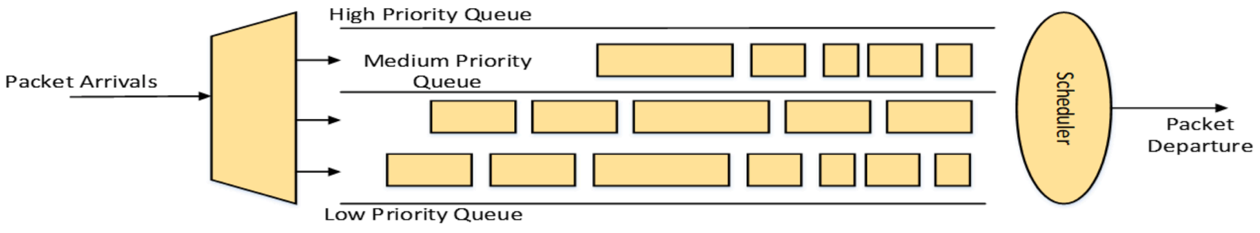

Fig. 4. Making use of a packet scheduler to schedule the packets.

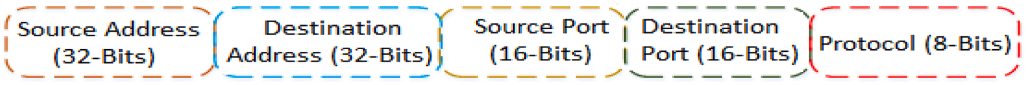

Fig. 5. Structure of packet in the network.

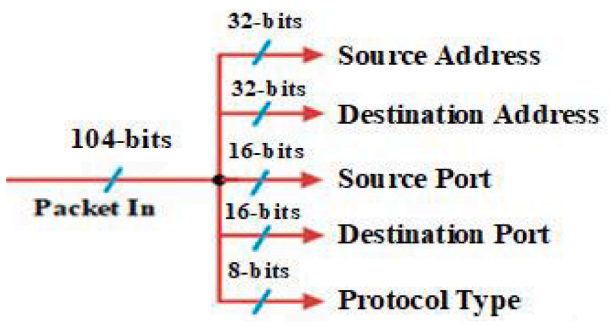

Fig. 6. Divestiture of the data packet's structure.

$$D \leq \frac{B}{C} \tag{1}$$

A single packet movement can be completely monopolize the queue's storage area, depriving all other flows of service unless the burst is handled. As a result, this type of queuing is used by a standard for a router's output link if no other queuing discipline is specified.

The packet scheduler is a vital component of the networking field:

- It decides the packets delivery order;
- It gives the foundations for QoS.

The input flow represents all traffic packets sent to the router. The classifier identifies a class of packets in the input flow and moves them to the appropriate queue. The priority queues have to hold packets from different traffic classes, each traffic class has its queue. The last element of proposed system is the scheduler. Its task is to choose the queue which will be serviced. Fig. 4 presents the system model.

Classifier moves packets from input flow to appropriate queues. We assume perfect recognition of packet classes. Each traffic categorization has its own priority queue, denoted by qi, where $I = 1,.., N$. Each queue has a limited capacity and can only store N_i packets. If q_i holds exactly N_i packets, and the classifier finds in the input flow the next packet from this traffic class, this packet will be rejected. All traffic classes have the same arrival distribution described by the Bernoulli process, with the mean value λ_i for the i^{th} traffic class. This means that in each time slot, we can observe a new packet in queue qi with probability λ_i. The mean delay requirement R_{ti} is the maximum acceptable mean delay per packet in queue qi and time slot t. We assume that this requirement is defined for the queues q_1, .., q_{N-1} and it can change in time. The best-effort queue is the last queue q_N, which has no delay requirements and should be handled as quickly as feasible.

The scheduler has to choose exactly one packet which will be transmitted in each time slot. The scheduler must formally select one action (a_i) from the action set $A = \{a_1, a_2,.., a_N\}$. The server will receive one message from queue q_i as a consequence of action ai. First Come, First Served (FCFS) is the order in which all queues are filled.

3.5. Packet splitter

The information obtained from the input RAM is used by the packet splitter modules to split packets. The network packet involved in a

Table 1
Performance assessment table for FPGA.

		Rules	LUTs	Flip-Flops	Slices	Frequency (MHz)
This Work	xc4vfx12–12sf363	16	388/10,944	95/10,944	211/5472	259
		32	526/10,944	82/10,944	304/5472	260
		64	598/10,944	90/10,944	420/5472	259
		128	991/10,944	104/10,944		259
		256	1812/10,944	98/10,944	1294/5472	230
		512	2985/10,944	101/10,944	1892/5472	255
		1024	5432/10,944	194/10,944	1922/5472	259
		2048	10,651/10,944	302/10,944	1209/5472	259
	xc6vlx760–2ff1760	16	209/474,240	86/948,480	69/118,560	230
		32	181/474,240	89/948,480	81/118,560	225
		64	192/474,240	87/948,480	72/118,560	225
		128	294/474,240	91/948,480	139/118,560	225
		256	395/474,240	85/948,480	148/118,560	230
		512	722/474,240	99/948,480	169/118,560	230
		1024	1385/474,240	105/948,480	181/118,560	250
		2048	2468/474,240	112/948,480	193/118,560	230
[20]	xc6vlx760–2ff1760	16	–	–	–	–
		32	566/474,240	34/948,480	144/118,560	–
		64	1070/474,240	65/948,480	185/118,560	–
		128	2144/474,240	127/948,480	231/118,560	–
		256	4125/474,240	255/948,480	358/118,560	–
		512	7704/474,240	508/948,480	613/118,560	–
		1024	–	–	–	–
		2048	–	–	–	–
[18]	XC6VLX760 FFG1760–2	16	–	–	–	257
		32				259
		64				201
		128	2144/474,240	48,704/948,480	14,773/118,560	–
		256	4125/474,240	97,502/948,480	29,056/118,560	–
		512	7704/474,240	195,164/948,480	57,209/118,560	–
		1024	–	329,690/948,480	112,812/118,560	–
		2048	–	–	–	–

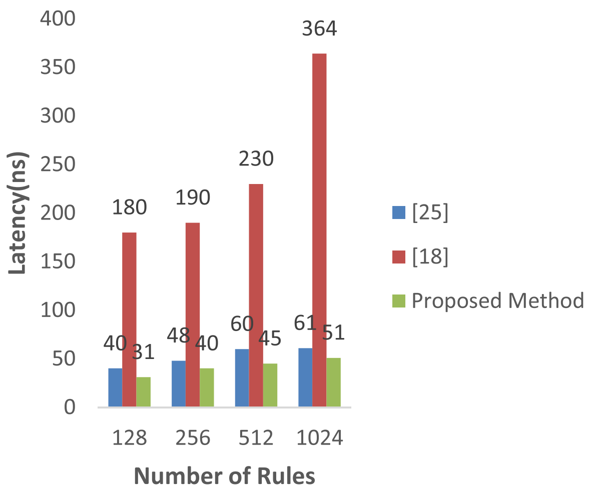

Fig. 7. Comparison of latency for 128, 256, 512, and 1024 Rules with the methods currently in use.

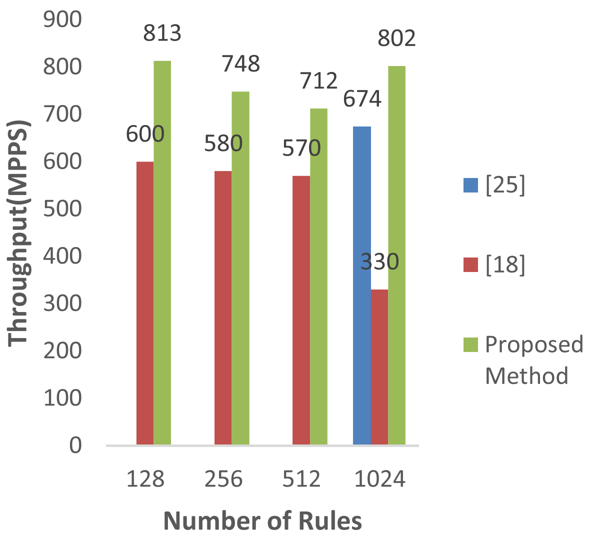

Fig. 8. Comparing throughput for 128, 256, 512, and 1024 Rules with the methods currently in use.

typical TCP communication procedure is depicted in Fig. 5. A 16-bit source and destination port will often have a 32-bit transceiver address and an 8-bit protocol type field. Each objective's location is identified in the Fig. 6. Based on the schematic details provided below, the resulting packet will be separated.

3.6. Rule-database

Depending on the source address, a destination address, source port, and destination port, packet classification information is recorded. The source and destination details of the rule are stored in an Nx32-bit-sized LUT. Similar metadata regarding both source and destination addresses are kept in NX16-bit storage. The Protocol Type information is stored in NX8-bit storage. N is also the number of rules employed in the grouping of packets. The address generator will read the rule information from the LUT to carry out the matching operation. The number of rules utilized in the design has a direct correlation with the length of the address generator. The address generator would be reset after the input packet matches have been made.

4. Results and Discussion

Using the MATLAB tool 2020a, the computational evaluation of the packet classification is performed. The proposed packet classification framework is created using the Xilinx XCV1000–6 BG560 on FPGA. The proposed design was simulated with VERILOG. (n/2) only provides the number of clock cycles required to accurately calculate 32-bit packets with 32 output bits. There is an n/2 clock cycle delay in the architecture. A high throughput rate of four legitimate results per clock cycle is achieved by returning four valid results each clock cycle. The overall power utilization of the created architecture is measured using the program X-Power Analyzer Version 9.2.04i, Version: J.40. The actual design uses 380 mW of power and operates at a clock frequency of 56.96 MHz. The performance of the design can be enhanced by employing a target device.

4.1. Throughput and latency

Table 1 displays an evaluation of latency performance considering several techniques. The suggested technique can classify packets in a single clock cycle. Assume that the suggested method makes use of fwe pipeline stages. A single pipeline operating at 105 MHz is used in the proposed method to estimate latency. So, four pipeline stages operate at 420 MHz of clock frequency. This method introduces the most delay due to the heavily pipelined design. However, it eliminates long cables and raises the clock rate, which improves latency performance. The comparison of latency with the methods currently in use is shown in Fig. 7. The system's latency is evaluated here for a range of rule numbers. Comparing this work to traditional packet classification techniques, the latency is reduced.

Using 128, 256, 512, and 1024 rules, respectively, Hatami & Bahramgiri's proposed architecture maintains latency at 40 ns, 18 ns, 60 ns, and 61 ns [25]. The architecture that Qu and Prasanna proposed has a latency of 180 ns, 190 ns, 230 ns, and 364 ns, with 128, 256, 512, and 1024 rules, respectively [18]. For 128, 256, 512, and 1024 rules, the proposed parallel architecture offers latency at 31 ns, 40 ns, 45 ns, and 51 ns. The design of Qu and Prasanna has more delay and the current state of the art had maintained lower latency for many regulations compared to the methods now in use.

Fig. 8 compares throughput using the currently used methods. In this case, compared to traditional procedures, throughput is enhanced. Constant throughput is achieved for various regulations, as demonstrated in the comparison chart.

Qu and Prasanna develop a two-dimensional pipeline architecture that enables N flows including an L-bit packet header in contrast to the prior techniques. For 128, 256, 512, and 1024 rules, this technique has a consistent throughput of 600, 580, 570, and 330 MPPS. The RTST architecture developed by Hatami and Bahramgiri supported 1024 rules and had a maximum throughput of around 674 MPPS [25]. Using 128, 256,512 and 1024 rules, the suggested method produces throughput of 813,748,712 and 802 MPPS respectively. High throughput is achieved by the suggested strategy.

The analysis that follows provides a succinct overview of packet categorization techniques that support 1 K rules (see Fig. 9). With a clock rate of 360 MHz, TCAM enables packet classification in such a single clock cycle. According to Zane et al. [23], it performs with a low latency of 10 ns and a throughput of 383 MPPS. Jiang and Prasanna used a mono pipeline framework with 89 stages that was controlled at a 105 MHz clock rate to maintain a latency of 428 ns while achieving a throughput of 391 MPPS [2]. Jiang and Prasanna used seven pipeline stages with a throughput of 376 MPPS and a latency of 25 ns at a 167 MHz supported clock rate [19].

Fig. 9 compares the throughput and latency of the various existing designs. This work demonstrates great throughput with minimal latency at 1k rules.

Jiang and Prasanna [12] created a 16-stage pipeline premised on the decision tree technique that functioned at 125 MHz with 82 ns of latency and 384 MPPS of throughput. An L-bit packet header allows N flows in

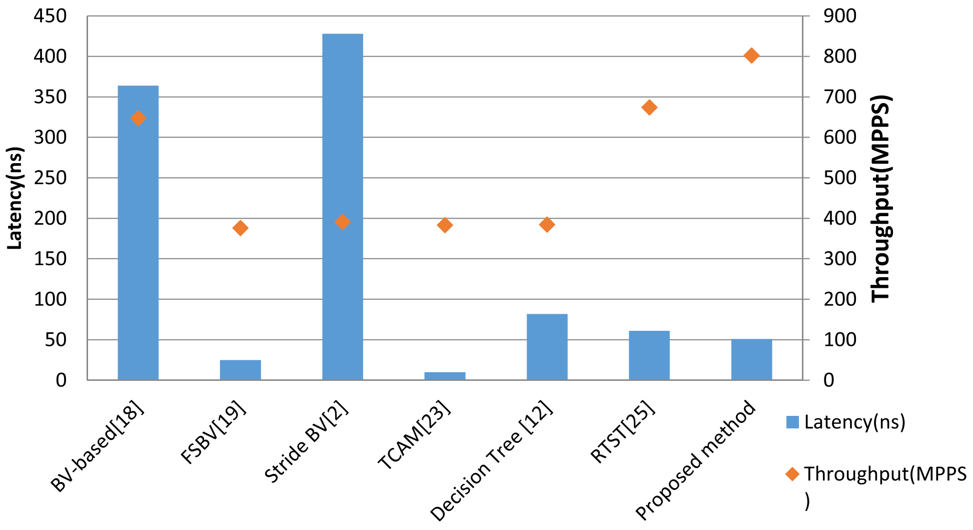

Fig. 9. Comparison of throughput and latency with the previous designs with the support of 1k rules.

Table 2
Performance of ASIC TSMC 180 nm and 90 nm.

	Rule	Area (mm^2)	Power (nW)	Delay (pS)	APP ($x10^{16}$)	ADP ($X10^3$)
TSMC180nm	16	280	405	2224	1.1787	646
	32	570	768	2430	4.3076	1362
	64	1187	803	2506	8.7366	2724
	128	2037	987	2622	1.9929	5289
	256	4087	106	2736	4.3674	1118
	512	8034	117	2809	9.4236	2257
	1024	12,089	127	2934	1.5469	3547
	2048	15,482	134	3081	2.0853	4770
TSMC90nm	16	145	205	1325	1.2987	1925
	32	283	380	1489	1.0765	4215
	64	562	471	1570	2.6511	8832
	128	1089	548	1650	5.9808	1797
	256	2145	589	1734	1.2643	3720
	512	4378	689	1841	3.0178	8060
	1024	8613	713	1963	6.1452	1690
	2048	17,265	727	2074	1.2566	3580

Table. 3
Comparison between latency and throughput.

Methods	Latency (ns)	Throughput (MPPS)
Stride BV [2]	428	391
Decision-tree [12]	82	384
BV-based pipe [18]	364	647
FSBV [19]	25	376
TCAM [23]	10	383
RTST [25]	61	674
Proposed Method	51	802

Table 4
Memory requirement.

Approach	Memory req.
Stride BV [2]	53.4 bytes/flow
FSBV [19]	99.23 bytes/flow
TCAM [23]	44.5 bytes/flow
RTST [25]	44.5 bytes/flow
Proposed Method	90.6 bytes/flow

the 2D pipeline design that Qu and Prasanna develop. With a clock rate of 324 MHz, this method has a maintained throughput of 647 MPPS and just a latency of 364 ns [18]. With the higher clock rate of 337 MHz and low latency of 61 ns, the suggested RTST architecture by Hatami and Bahramgiri was able to reach a maximum throughput of 674 MPPS [25]. The proposed approach employs four pipeline phases. It uses a single pipeline running at 105 MHz. The 420 MHz is used in the suggested method to determine the latency of 4 pipeline stages. At a high clock rate of 420 MHz, the suggested technique produces a throughput of over 802 MPPS and a latency of 51 ns. With less reasonable latency, a proposed-based technique achieves the best throughput (802 MPPS) with low latency (51 ns).

4.2. Performance evaluation

By contrasting it with current packet classification methods, performance can be evaluated of the proposed technique. This suggested algorithm is intended for use with enterprise-level security hardware. Given that such equipment frequently uses Intel Xeon processors with several cores and big amounts of memory.

4.2.1. FPGA implementation

To assess performance relying on resource consumption and frequency, the suggested design was implemented in a Xilinx FPGA. For the best performance, the design process should use the fewest resources possible. Compared to the conventional design, the proposed architecture in this work requires fewer hardware resources. Compared to the proposed architecture, the traditional design required more resources. The FPGA Performance assessment for various rules with area and frequency usage is shown in Table. 1.

4.2.2. ASIC implementation

The utilized area, energy, latency, clock frequency, and other metrics can be used to assess the suggested architecture. The suggested ASIC performance metric evaluation design is written in the Verilog program and executed on a 180 nm processor. Area, power, and latency should all be decreased in comparison to conventional designs for optimum performance. The ASIC on TSMC 180 nm and 90 nm Productivity is displayed in Table 2.

4.3. Design parameters / performance metrics

Table 3 compares latency and throughput with earlier methods. When compared to other approaches, this work provided a superior throughput of around 802 MPPS (Million Packets Per Second), which is high. A 51 ns delay is also attained.

The memory requirements for various approaches are shown in

Table 5
Energy efficiency in existing methods Vs proposed method.

Approach	Energy used(nJ)
Stride-BV [2]	12.7
Decision-tree [12]	75.1
BV-based pipe [18]	15.9
FSBV [19]	33.8
TCAM [23]	280
Proposed Method	5.2

Table 4. The proposed technique uses 90.6 bytes/flow of memory.

Table 5 compares energy usage across various architectural types. When compared to other ways, the suggested method used 5.2 nJ less energy (nano Joules).

The TCAM provided by Zane et al. consumed more energy in each packet, i.e. 280 nJ, in comparison to the other approaches, while the proposed methodology uses much less energy per packet of 5.2 nJ in comparison to the other approaches. In comparison to other systems currently in use, the suggested methodology is more energy efficient. The Stride-BV approach, which required 12.7 nJ of energy for every packet, was suggested by Kennedy et al. Prasanna and Ganegedara used 33.8 nJ of energy. Qu and Prasanna use 15.9nJ of energy in every packet while Jiang and Prasanna use 75.1nJ of energy efficiently. The proposed technique made use of an energy or packet of 5.2 nJ. By comparing it to other methods already in use, it has greater energy efficiency.

5. Conclusion

This paper presents a novel method for packet classification that integrates the parallel processing of packets. Due to their low latency performance, DPRAMs are preferred for stage memory to achieve fast rate packet lookup. The proposed method makes use of two DPRAMs (DPRAM1 and DPRAM2), which enable multiple read of the data. The parallel packet classification architecture is implemented on a Xilinx Virtex XCV1000–6 BG560 FPGA. The proposed design may achieve an 802 MPPS throughput, resulting in a 51 ns latency at 420 MHz clock frequency, according to experimental results. Moreover, the power study shows that the total packet classification engine uses 5.2 nJ (nano Joules) of energy with the maximal rule set. Also, it had able to reach a memory efficiency of 90.6 bytes/flow. Designing a packet classification method generally requires managing low latency, high throughput, and improved energy efficiency with high clock frequency maintained. Existing techniques are unable to simultaneously meet all these demands. Contrary to previous algorithms, our suggested method can support both fast packet classification and very large rule sets simultaneously. The proposed technique may be able to maintain low energy consumption while making a trade-off between throughput and delay. It is explored how this suggested packet classification technique balances energy efficiency, latency, and throughput. This suggested approach is intended to be a key component of the effective network and network security solutions.

Funding

There was no funding for the work presented in this paper.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Andrea Sanny, Thilan Ganegedara, Viktor K. Prasanna; "A comparison of ruleset feature independent packet classification engines on FPGA," in 27th International Symposium on Parallel & Distributed Processing Workshops and Ph.D. Forum, 978-0-7695-4979-8/13 $26.00 © 2013 IEEE.
[2] T. Ganegedara, V.K. Prasanna, StrideBV: single chip 400G+ Packet classification, in: Proc. of IEEE International Conference on High Performance Switching and Routing (HPSR), 2012, pp. 1–6.
[3] Mahmood Ahmadi, S.Arash Ostadzadeh, Stephan Wong, An analysis of rule-set databases in packet classification, in: 18th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC 2007), Veldhoven, The Netherlands, 2007, 29-30 November.
[4] Nekoo Rafiei Karkvandi, Hassan Asgharian, Amir Kusedghi, Ahmad Akbari, Hardware network packet classifier for high speed intrusion systems, Int. J. Eng. Technol. 4 (3) (2014). March.
[5] W. Pak, Y.J Choi, High performance and high scal- able packet classification algorithm for network security systems, IEEE Trans. Depend. Sec. Comput. 14 (2015), https://doi.org/10.1109/TDSC.2015.2443773, 1–1.
[6] M. Akkoc, B Canberk, Interval partitioning for packet classification in open flow vSwitch, IEEE Netw. Lett. 2 (2020) 128–131, https://doi.org/10.1109/LNET.2020.3007570.
[7] D.E. Taylor, Survey and taxonomy of packet classification techniques, ACM Comput. Surv. 37 (3) (2005) 238–275.
[8] Balasaheb S. Agarkar, Uday V. Kulkarni, A novel technique for fast parallel packet classification, Int J Comput. Appl. 76 (4) (2013), 0975 –8887–August_.
[9] Aladdin Abdulhassan, Mahmood Ahmadi, Parallel many fields packet classification technique using R-tree, in: Annual Conference on New Trends in Information & Communications Technology Applications-(NTICT'2017), 2017, 7-9 March.
[10] C.R. Meiners, A.X. Liu, E. Torng, Hardware Based Packet Classification for High Speed Internet Routers, SpringerVerlag, Berlin, Germany, 2010.
[11] K. Lakshminarayanan, A. Rangarajan, S. Venkatachary, Algorithms for advanced packet classification with ternary CAMs, in: Proc. of SIGCOMM, 2005, pp. 193–204.
[12] Weirong Jiang, Viktor K. Prasanna, Scalable packet classification on FPGA, IEEE Trans. Very Large Scale Integration (VLSI) Syst. 20 (9) (2011) 1668–1680.
[13] Yao Xin, et al., FPGA-based updatable packet classification using TSS-combined bit-selecting tree, IEEE/ACM Trans. Networking 30 (6) (2022) 2760–2775.
[14] D. Yin, D. Unnikrishnan, Y. Liao, L. Gao, R. Tessier, Customizing virtual networks with partial FPGA reconfiguration, SIGCOMM Comput. Commun. Rev. 41 (1) (2011) 125–132.
[15] Andreas Fiessler, Sven Hager, Björn Scheuermann, Flexible line speed network packet classification using hybrid on-chip matching circuits, in: IEEE 18th International Conference on High Performance Switching and Routing (HPSR), 2017, 18-21 June.
[16] A. Sudarsanam, R. Barnes, J. Carver, R. Kallam, A. Dasu, Dynamically reconfigurable systolic array accelerators: a case study with extended kalman filter and discrete wavelet transform algorithms, Compu. Digital Techniques, IET 4 (2) (2010) 126–142.
[17] W. Jiang, V.K. Prasanna, Large-scale wire-speed packet classification on FPGAs, in: Proc. FPGA, Feb. 2009, pp. 219–228.
[18] Y.R. Qu, V.K Prasanna, High-performance and dynamically updatable packet classification engine on FPGA, IEEE Trans. Parallel Distrib. Syst. 27 (2015) 197–209, https://doi.org/10.1109/TPDS.2015.2389239.
[19] W. Jiang, V.K. Prasanna, Field-split parallel architecture for high performance multi match packet classification using FPGAs, in: Proceedings of the of the 21st Annual symposium on Parallelism in Algorithms and Arch. (SPAA), 2009, pp. 188–196, https://doi.org/10.1145/1583991.1584044.
[20] Design of high performance packet classification architecture for communication networks, in: A.U. Khan, M. Chawhan, Yogesh Suryawanshi, Sandeep Kakde (Eds.), Design of high performance packet classification architecture for communication networks, J. Telecommun. (Electron. Comput. Eng. 9 (4) (2017) 109–115 (JTEC).
[21] S. Yingchareonthawornchai, J. Daly, A.X. Liu, E Torng, A sorted-partitioning approach to fast and scalable dynamic packet clas- sification, IEEE ACM Trans. Netw. 26 (2018) 1907–1920, https://doi.org/10.1109/TNET.2018.2852710.
[22] W. Yu, S. Sivakumar, D. Pao, Pseudo-TCAM: sRAM-based architecture for packet classification in one memory access, IEEE Netw. Lett. 1 (2019) 89–92, https://doi.org/10.1109/LNET.2019.2897934.
[23] K. Sakthi, P. Nirmal Kumar, Efficient soft error resiliency bymulti-match packet classification using scalable TCAM implementation in FPGA, Microprocess. Microsyst. 74 (2020), https://doi.org/10.1016/j.micpro.2019.102985. PubMed: 102985.
[24] O. Erdem, A Carus, Multi-pipelined and memory-efficient packet classification engines on FPGAs, Comput. Commun. 67 (2015) 75–91, https://doi.org/10.1016/j.comcom.2015.05.017.
[25] R. Hatami, H Bahramgiri, High-performance architecture for flow-table lookup in SDN on FPGA, J. Supercomput. 75 (2019) 384–399, https://doi.org/10.1007/s11227-018-02732-2.
[26] A. Yazdinejad, R.M. Parizi, A. Bohlooli, A. Dehghantanha, K.R. Choo, A high-performance framework for a network programmable packet processor using P4 and FPGA, J. Netw. Comput. Appl. 156 (2020), https://doi.org/10.1016/j.jnca.2020.102564. PubMed: 102564.
[27] M. Kekely, L. Kekely, J. Kořenek, General memory efficient packet matching FPGA architecture for future high-speednetworks, Microprocess. Microsyst. 73 (2020), https://doi.org/10.1016/j.micpro.2019.102950. PubMed: 102950.
[28] S. Hager, D. Bendyk, B. Scheuermann, Matching circuits can be small: partial evaluation and reconfiguration for FPGA-based packet processing, J. Parallel Distrib. Comput. 109 (2017) 42–49, https://doi.org/10.1016/j.jpdc.2017.05.004.

[29] Y.R. Qu, S. Zhou, V.K Prasanna, A decomposition-based approach for scalable many-field packet classification on multi-core pro- cessors, Int. J. Parallel Program. 43 (2015) 965–987, https://doi.org/10.1007/s10766-014-0325-6.
[30] H. Jamil, N. Yang, N. Weng, Many-field packet classification with decomposition and reinforcement learning, IET Networks 11 (3–4) (2022) 112–127.
[31] A. Ponnuswamy, M Devi, Range-enhanced packet classification to improve computational performance on field programmable gate array, Int. J. Electric. Comput. Eng. 12 (6) (2022) 2088–8708.
[32] Y. Li, J. Wang, X. Chen, J. Wu, SplitTrie: a fast update packet classification algorithm with trie splitting, Electronics (Basel) 11 (2) (2022) 199.
[33] Y. Cheng, Q. Shi, PCMIgr: a fast packet classification method based on information gain ratio, J. Supercomput. (2022) 1–24.
[34] Vasaki Ponnusamy, et al., IoT wireless intrusion detection and network traffic analysis, Comput. Syst. Sci. Eng. 40 (3) (2022) 865–879.
[35] Hsin-Tsung Lin, Pi-Chung Wang, TCAM-based packet classification for many-field rules of SDNs, Comput. Commun. (2023).
[36] Nafis Irtija, et al., Design and analysis of digital communication within an SoC-based control system for trapped-ion quantum computing, IEEE Trans. Quantum Eng. 4 (2023) 1–24.

Author1 Adiseshaiah Midde received his M.Tech in VLSI System Design form Siddhartha Institute of Technology, Puttur, Affliated to JNTU Anantapur, India in 2011. He is currently pursuing Ph.D Degree in Electronics and Communication Engineering Department, JNTU Kakinada, India. His-research interests includes VLSI Design.

Author2 Sailaja Maruvada obtained her Ph.D. in ATM NETWORKS from JNTU Kakinada, India, in 2009. She is currently serving as Professor, ECE Department, JNTU Kakinada. Her research interests include computer networks and adaptive signal processing.