400 Gbps Energy-Efficient Multi-Field Packet Classification on FPGA

Shijie Zhou, Sihan Zhao and Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089
Email: {shijiezh, sihanzha, prasanna}@usc.edu

Abstract—Packet classification is a network kernel function. It has been widely researched in the community over the past decade. However, most previous works only focus on achieving high-throughput, leaving energy-efficiency an untapped area for improvement. With a rapid growth of Internet, energy-efficiency has become an important metric for networks. In this paper, we design an energy-efficient packet classifier on Field-Programmable Gate Arrays (FPGA). The classifier is arranged as a 2-dimensional array of processing elements to sustain a high throughput. We develop a memory activation scheduling technique to selectively activate the memory blocks. Using this technique, the memory power dissipation is significantly reduced. In addition, to achieve a high clock rate, the architecture is arranged as a 2-dimensional array of processing elements. We conduct detailed experiments using real-life rule sets and packet traces. Experimental results show that with the memory activation scheduling technique, our design achieves 1.8x higher energy-efficiency. When there are 6 individual classifiers on a single chip and the rule set size is 1K, our design sustains a throughput of 400 Gbps for minimum size (40 bytes) packets and can process over 100 Gbps network traffic per Joule. Compared with the state-of-the-art solutions, we achieve at least 1.7x improvement in energy-efficiency.

Index Terms — FPGA, Energy-Efficient, Packet Classification

I. INTRODUCTION

Packet classification is a network kernel function performed by Internet routers. It enables various network services such as network security, Quality of Service (QoS) routing and resource reservation. Routers perform packet classification based on a predefined rule set and route the packet to a specific port according to the classification result. Packet classification requires the inspection of multiple fields of the packet header. In traditional packet classification [1], each predefined rule considers 5 header fields: source/destination IP addresses, source/destination port numbers and transport layer protocol. In Software Defined Networking (SDN) [2], which is proposed as the next generation of network, each rule considers up to 40 header fields [3].

Most of packet classification algorithms fall into three categories: decision-tree based [4], hash based [5] and decomposition based [6] algorithms. Decision-tree based algorithms employ several heuristics to cut the rule set space into smaller sub-regions in a multi-dimensional space. Each node in the decision-tree represents a sub-region. Hash based algorithms group rules into a set of tuple spaces and maintain each tuple space as a hash table. Decomposition based approaches first search each field individually and then merge the intermediate results of each field to obtain a final classification result.

As the Internet grows rapidly, the energy consumption of routers increases due to the larger amount of network traffic. Therefore, energy-efficiency becomes a critical concern in router design [7], [8], [9]. Many current hardware-based packet classifiers harness ternary content addressable memories (TCAMs) [10], [11]. Although TCAM-based solutions can achieve very high throughput, they suffer from poor energy-efficiency due to the massively parallel exhaustive search [8]. Recent researches have proposed to use FPGA technology as an energy-efficient alternate platform for implementing real-time network processing engines [12], [13], [14]. FPGA-based packet classifiers can achieve very high throughput. However, designing a high-throughput, yet energy-efficient packet classifier on FPGA still poses great challenges.

In this paper, we develop a memory activation scheduling technique which selectively activates memory blocks. Using this technique, the memory power dissipation can be significantly reduced. In addition, to achieve a high clock rate, the architecture is arranged as a 2-dimensional array of processing elements. We conduct detailed experiments using real-life rule sets and packet traces to evaluate the designs. The main contributions of our work are summarized below.

- We develop a memory activation scheduling technique which can significantly reduce memory power dissipation. Using this technique, we observe over 1.8x improvement in energy-efficiency compared with a baseline implementation.

- The clock rate of our design is scalable with respect to the size of the rule set. When the rule set size is 1K, our energy-efficient design sustains a throughput of 400 Gbps for minimum size (40 bytes) packets on a single chip.

- Compared with the state-of-the-art solutions, our design achieves over 1.7x improvement in energy-efficiency.

The rest of the paper is organized as follows: Section II introduces the background and related work; Section III presents the algorithms; Section IV discusses the architecture of our approach; Section V contains experimental results; Section VI concludes the paper.
<table>
<thead>
<tr>
<th>Rule ID</th>
<th>SA</th>
<th>DA</th>
<th>SP</th>
<th>DP</th>
<th>Protocol</th>
<th>Priority</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>175.77.88.155/32</td>
<td>192.0.0.0/8</td>
<td>0-65536</td>
<td>0-1000</td>
<td>0x06</td>
<td>1</td>
<td>Action 1</td>
</tr>
<tr>
<td>2</td>
<td>11.77.88.0/24</td>
<td>182.0.96.12/32</td>
<td>16-16</td>
<td>0-65536</td>
<td>0x11</td>
<td>2</td>
<td>Action 2</td>
</tr>
<tr>
<td>3</td>
<td>192.168.20.26/32</td>
<td>125.199.2.72/32</td>
<td>10-65536</td>
<td>200-300</td>
<td>0x11</td>
<td>3</td>
<td>Action 3</td>
</tr>
<tr>
<td>4</td>
<td>10.10.0.0/20</td>
<td>12.13.0.0/16</td>
<td>100-100</td>
<td>0-65536</td>
<td>0x2f</td>
<td>4</td>
<td>Action 4</td>
</tr>
<tr>
<td>5</td>
<td>12.1.11.256/32</td>
<td>137.135.0.0/16</td>
<td>0-1000</td>
<td>20-22</td>
<td>0x06</td>
<td>5</td>
<td>Action 5</td>
</tr>
</tbody>
</table>

### II. Background and Related Work

#### A. Multi-Field Packet Classification

In packet classification, an IP packet is classified based on 5 fields in the packet header: 32-bit source/destination IP addresses (denoted as SA/DA), 16-bit source/destination port numbers (denoted as SP/DP) and 8-bit transport layer protocol. Routers perform packet classification based on a predefined rule set. [1] studies the real-life firewall rule sets from several Internet Service Providers (ISPs) and reveals that 0.7% of the rule sets contain more than 1000 rules while nearly 99% of the rule sets contain less than 500 rules.

Each rule consists of the matching criteria of each field, a priority and an action to be taken if matched. Different fields require different types of matches: SA and DA require prefix match; SP and DP require range match; protocol field requires exact match. A packet matches a rule only when all the five header fields are matched. If a packet matches multiple rules, the action associated with the highest prioritized one will be taken. We show an example of rule set in TABLE I.

#### B. Prior Work on FPGA

State-of-the-art FPGA devices provide reconfigurability, abundant parallelism, dense logic units and vast amounts of on-chip memory. These features make it an ideal platform for networking applications. Packet classification has been extensively studied on FPGA.

[15] aims at high energy-efficiency and implements a simplified HyperCuts [16] algorithm on an Altera Cyclone 3 FPGA. But the clock rate is only 32 MHz, resulting in a throughput of less than 1 Gbps in the worst cases. [19] maps a decision-tree approach onto a pipelined architecture. The design achieves a throughput of 80 Gbps for a rule set containing 10K real-life rules. [13] presents an architecture called BV-TCAM which combines TCAMs and the BV algorithm [6]. TCAMs perform prefix and exact match, while a multi-bit trie is used for range match. However, the energy-efficiency is not discussed in [13], [19]. An Field-Split Bit Vector (FSBV) approach is proposed in [12]. FSBV examines a packet bit by bit against the rule set and generates an intermediate bit-vector (BV) for each bit of the packet. An intermediate BV implies which rules match the specific bit. A final BV is obtained by bitwise ANDing all the intermediate BVs. The non-zero bits in the final BV indicate the matching rules. StrideBV, which is an extension of FSBV, is proposed in [14]. Instead of examining packets bit by bit, StrideBV considers a stride of several bits as an examining unit. However, [12], [14] mainly focus on achieving high throughput, while the optimizations for energy-efficiency are not studied.

### III. Algorithm

In our design, we adopt the decomposition-based approach proposed in [6]. The idea of the algorithm is searching each field individually to produce an intermediate result (represented as a BV) and then merging all the intermediate results. In this section, we mainly discuss how we handle exact match, prefix match and range match. Throughout this paper, we use $N$ to denote the number of rules in the rule set.

#### A. Exact Match

The protocol field is 8-bit and requires exact match. The maximum number of unique protocols in a rule set is $2^8$, regardless of $N$. According to [20], only 6 protocols appear in the sample rule sets extracted from real-life rule sets. The 6 protocols are listed in TABLE II.

<table>
<thead>
<tr>
<th>Hex</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>HOPOPT IPv6 Hop-by-Hop Option</td>
</tr>
<tr>
<td>0x01</td>
<td>ICMP Internet Control Message Protocol</td>
</tr>
<tr>
<td>0x06</td>
<td>TCP Transmission Control Protocol</td>
</tr>
<tr>
<td>0x11</td>
<td>UDP User Datagram Protocol</td>
</tr>
<tr>
<td>0x2F</td>
<td>GRE Generic Routing Encapsulation</td>
</tr>
<tr>
<td>0x32</td>
<td>ESP Encapsulating Security Payload</td>
</tr>
</tbody>
</table>

Since the number of unique protocols in a rule set is usually small, it is very likely to find a perfect hash function or minimal perfect hash function which maps distinct protocols to a set of integers with no collisions [21]. We use such a perfect hash function to construct a hash table and perform classification for the protocol field. Protocol values and the associated BVs are stored as hash values in the hash table. Based on the protocol field of TABLE I, TABLE III shows the corresponding hash table; Algorithm IV illustrates the hash-based classification.

<table>
<thead>
<tr>
<th>Index</th>
<th>Protocol</th>
<th>BV</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000 0110</td>
<td>10001</td>
</tr>
<tr>
<td>1</td>
<td>0001 0001</td>
<td>01100</td>
</tr>
<tr>
<td>2</td>
<td>0010 1111</td>
<td>00010</td>
</tr>
</tbody>
</table>
Algorithm 1 Hash-based classification

Classify(input)
1: Index = input >> 4;
2: if Hash_Table[Index].protocol = input then
3:    return Hash_Table[Index].BV
4: else
5:    return 0
6: end if

B. Prefix Match

SA and DA are two 32-bit fields requiring prefix match which involves ternary bits. A ternary bit can be “0”, “1”, or “don’t care (*”). We adopt the same approach of [14] to handle prefix match. A 32-bit input IP is divided into several strides. Each stride is examined by a processing element to produce a BV. For a k-bit stride, we store 2^k BVs in random access memory (RAM). Each BV corresponds to one possible combination of the k-bit stride and implies which rules match the combination. Processing elements employ the input stride as the address to access RAM and obtain the BV. The output BVs from all the processing elements are bitwise ANDed.

Fig. 1: Example of prefix match

Fig. 1 shows an example of BVs stored in the RAM for a 4-bit field which is divided into two 2-bit strides (N = 3). When the input “1110” comes, we obtain “001” based on the first stride and “111” based on the second stride. By ANDing the two BVs we obtain “001”, indicating that the input matches the third rule.

C. Range Match

SP and DP are two 16-bit fields which require range match. A 16-bit range can be expanded to 30 prefixes or 28 ternary strings [17]. Such a problem is called “rule expansion”. In order to avoid the range-to-prefix conversion, we use 2 x N 16-bit registers to explicitly store the upper boundaries and lower boundaries of the N ranges. The input value is compared against all the boundaries in parallel, and each pair of the lower and upper boundaries determines the match result of a rule. Fig. 2 depicts the range match process.

IV. Architecture

A. 2-dimensional Architecture

As N increases, BVs become larger, resulting in more complex routing and long wire connections. Thus, the clock rate is negatively affected. To address this problem, we divide the classifier into a set of small processing elements laying in two dimensions (R rows x C columns).

Horizontally, each processing element is responsible for classifying a field (or a stride of IP address) as discussed in Section III. The local result BV of each processing element is bitwise ANDed with the BV imported from the previous horizontal neighbour; the ANDed result is output to the next horizontal neighbour.

Vertically, each row of processing elements store the BVs for a subset (\( \frac{N}{R} \) rules) of the rule sets. Therefore, the size of input and output BVs is \( \frac{N}{R} \) bit for each processing element. Since the rules of different rows have different priorities, all the final BVs of different rows need to be fed into a priority encoder to determine the global result. The priority encoder is implemented in a pipeline fashion with \( \log N \) stages [14].

The architecture is implemented in a pipelined fashion to provide high clock rate and be scalable to support larger rule sets.

B. Memory Activation Scheduling

We develop the memory activation scheduling technique to reduce the amount of memory power. The idea of this technique is to deactivate unneeded memory blocks during classification. To achieve this goal, each processing element first checks the input BV which is from the previous pipeline stage. If the input BV is “0”, it indicates the final BV of this row must be “0”, no matter what the local result BV is. Hence, access to the memory is not needed and the corresponding memory block can be deactivated; the processing element can directly pass “0” to the next stage. When the input BV contains non-zero bit, the memory block is reactivated.

In order to maximize the benefit from the memory activation scheduling, we arrange the processing elements for the protocol field at the beginning of each row. The reason is that...
because the number of unique protocols is small, this results in many “0” BVs at the first stage, enabling the memory power for the following processing elements in the same row to be saved. Fig. 3 depicts the top-level architecture.

V. PERFORMANCE EVALUATION

A. Experimental Setup

The experiments are conducted on the state-of-the-art Xilinx Virtex 7 XC7VX980 with -2L speed grade. The target platform has 303,600 logic slices, 800 I/O pins, 36 Mb BRAMs and up to 16 Mb distributed RAMs (distRAM). The performance are evaluated using Xilinx Vivado 2014.2 development tools. The packet traces and rule sets are provided in [20], a widely used benchmark for packet classification in the community. We use the VCD (value change dump) file as input to the Xilinx Vivado Power Analyzer to produce accurate power dissipation estimations. All the reported results are based on post-place-and-route simulations. We use clock rate and energy-efficiency as the performance metrics for evaluating. We define energy-efficiency as the amount of network traffic (Gb) classified per Joule (assuming 40-byte packets). While looking at power consumption in our experiments, we only consider dynamic power since static power is outside of our control.

Throughout this section, we use \( m \) to denote the number of rules that each row of processing elements is responsible for, and \( k \) to denote the size of stride which the 32-bit IP address is divided into.

B. Distributed vs. Block RAMs

The “RAM” in Fig. 3 can be implemented using either BRAM or distRAM. BRAMs are provided as standalone RAMs. DistRAMs are built by logic resources and local to processing elements. In this section, we explore the energy-efficiency of the BRAM-based design and the distRAM-based design. TABLE IV shows the performance under various configurations of \( m \) and \( k \) (\( N=1024 \)).

![Fig. 3: Top-level architecture](image)

<table>
<thead>
<tr>
<th>( m )</th>
<th>( k )</th>
<th>Clock Rate (MHz)</th>
<th>Energy Efficiency (Gb/J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRAM</td>
<td>distRAM</td>
<td>BRAM</td>
<td>distRAM</td>
</tr>
<tr>
<td>64</td>
<td>8</td>
<td>194</td>
<td>201</td>
</tr>
<tr>
<td>64</td>
<td>4</td>
<td>198</td>
<td>203</td>
</tr>
<tr>
<td>64</td>
<td>2</td>
<td>195</td>
<td>208</td>
</tr>
<tr>
<td>32</td>
<td>8</td>
<td>212</td>
<td>202</td>
</tr>
<tr>
<td>32</td>
<td>4</td>
<td>213</td>
<td>217</td>
</tr>
<tr>
<td>32</td>
<td>2</td>
<td>208</td>
<td>225</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>213</td>
<td>203</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
<td>208</td>
<td>212</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>158</td>
<td>217</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>205</td>
<td>207</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>162</td>
<td>204</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Not Supported</td>
<td>207</td>
</tr>
</tbody>
</table>

We observe that the highest energy-efficiency is achieved when \( m=32 \) for both BRAM-based design and distRAM-based design. If \( m \) decreases, the number of processing elements increases, resulting in more complex routing. If \( m \) increases, the data width increases, which deteriorates the clock rate and increases the signal power. Moreover, the possibility of outputting “0” BV from each processing element is lower for larger \( m \), reducing the effect of memory activation scheduling.

For BRAM-based design, we achieve the highest energy-

---

1 When \( m=8 \) and \( k=2 \), the BRAMs on the target platform are not enough
efficiency when \( k = 8 \). Since the minimum BRAM on the target platform is 18 Kb, if \( k \) decreases, the number of needed BRAMs increases, resulting in more memory power. For distRAM-based design, the highest energy efficiency is achieved when \( k = 2 \). This is due to that the design with a smaller \( s \) has lower distRAM consumption [14].

We observe that distRAM-based design can achieve a higher energy-efficiency than BRAM-based design. This is because (1) the wire length between distRAMs and processing elements is much shorter and (2) distRAM is more energy-efficient for small size memories (< 4 Kb) [22].

C. Baseline vs. Optimized Design

In this section, we compare our optimized designs with the baseline designs which do not apply the memory activation scheduling (for \( N = 1024 \)). For BRAM-based designs, we fix \( m \) at 32 and \( k \) at 8; for distRAM-based designs, we fix \( m \) at 32 and \( k \) at 2. Fig. 4 shows the comparison result.

It can be observed that the clock rate drops slightly due to applying memory activity scheduling, but the energy-efficiency is improved by over 1.8x for both BRAM-based design and distRAM-based design. Fig. 5 depicts the power dissipation of each type of dynamic power when the designs run at 200 MHz. For the BRAM-based designs, the memory power from BRAMs dominates. Due to the memory activation scheduling, the BRAM power for the optimized design is significantly reduced. For the distRAM-based designs, the memory power for distRAMs is dissipated in terms of signal power and logic power. As shown in Fig. 5, the optimized distRAM-based design dissipates much less signal power than the baseline design.

D. Throughput and Resource Utilization

The target platform allows to implement 6 individual classifiers. The main constraint comes from the number of I/O pins. For the BRAM-based design, each classifier can run at 205 MHz, resulting in a throughput of 393 Gbps for minimum size (40 bytes) packets; the energy-efficiency is 122 Gb/J. TABLE V and TABLE VI show the resource utilization of the BRAM-based design and the distRAM-based design when there are 6 classifiers on a single chip, respectively.

<table>
<thead>
<tr>
<th>TABLE V: Resource Utilization for BRAM-based design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice LUTs</td>
</tr>
<tr>
<td>20%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE VI: Resource Utilization for distRAM-based design</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slice LUTs</td>
</tr>
<tr>
<td>83%</td>
</tr>
</tbody>
</table>

E. Scalability

To study the scalability of our approach, we conduct experiments using various sizes of rule sets. Fig. 6 and Fig. 7 show the performance of the BRAM-based design and distRAM-based design respectively.

It can be observed that for both designs, as the rule set size increases, the clock rate degrades. This is due to the longer wire connections and more complex routing. The energy-efficiency degrades for larger rule sets as well. This is because larger rule sets require more processing elements. However, due to the memory activation scheduling, the energy-efficiency does not degrade in a linear fashion.
Fig. 7: Scalability of distRAM-based design

F. Comparison with Existing Approaches

In this section, we compare our design with several existing approaches \[^{[10], [12], [14]}\]. For fair comparisons, we implement the FPGA-based approaches \[^{[12], [14]}\] on the same target platform. The TCAM-based approach \[^{[10]}\] is also scaled to the state-of-the-art technology based on an 18 Mb TCAM running at 360 MHz and consuming 15 W \[^{[23]}\]. We use the rule set with 512 rules and only consider the dynamic power. The comparison results are depicted in Fig. 8. Our distRAM-based design and BRAM-based design achieve over 1.9x and 1.7x higher energy-efficiency, respectively.

Fig. 8: Comparison result

VI. CONCLUSION

In this paper, we developed an energy-efficient and high-throughput packet classifier on FPGA. We conducted comprehensive experiments using real-life rule sets and packet traces. We handled exact match by perfect hashing, prefix match by StrideBV approach and range match by explicitly storing the ranges. We devised the memory activation scheduling technique to deactivate unused memory blocks. By applying this technique, the energy-efficiency was improved by over 1.8x and the clock rate was not affected much. We had such observation for both BRAM-based design and distRAM-based design. For a rule set with 1K rules, with 6 classifiers on a single chip, our design sustained over 400 Gbps throughput and achieved the energy-efficiency over 100 Gb/J. Compared with the state-of-the-art approaches, our design demonstrated at least 1.7x improvement.

In the future, we will explore more techniques to reduce the signal power as well. We also plan to extend our work to design an energy-efficient OpenFlow packet classifier.

REFERENCES


