TCAM-based High Speed Longest Prefix Matching with Fast Incremental Table Updates

A. Rasmussen, A. Kragelund, M. Berger, H. Wessing, S. Ruepp
DTU Fotonik, Technical University of Denmark
Ørsteds Plads 343, 2800 Kgs. Lyngby, Denmark
Email: anras@fotonik.dtu.dk

Abstract—This paper presents a new TCAM-based method for determining the Longest Prefix Match (LPM) in an IP routing table. The method is based on modifying the address encoder of the standard TCAM design to take the prefix lengths of the IP routing entries into account while performing multi-match resolution, thus allowing prefixes to be inserted in any random order. This enables full utilization of the TCAM address space while greatly simplifying the updating procedure as complex software sorting algorithms and extensive table modifications are avoided. The result is faster table updates and consequently a higher throughput of the network search engine, since the TCAM down time caused by incremental updates is eliminated. The LPM scheme is described in HDL for FPGA implementation and compared to an existing scheme for customized CAM circuits. The paper shows that the proposed scheme can process more packets per second, has less per-lookup power consumption and is easier to expand to larger routing tables than the existing implementation. The latency of the LPM operation is only $\log_2 N$ clock cycles, where $N$ is the maximum number of prefixes in the TCAM, and in a pipelined implementation the throughput of the system is one Longest Prefix Match lookup per clock cycle.

I. INTRODUCTION

The continuous growth of the Internet creates increasing requirements for central network elements, such as core routers running the Internet Protocol (IP). The increase in size of the networks and the lack of address space requires these routers to maintain large routing tables, possibly with several hundreds of thousands entries [1]. This puts dramatic pressure on implementations for managing these routing tables in the form of insertion of new entries, removal of old entries and searching the tables. The increase in network speed, and thereby the number of packets being received per second, sets further requirements on fast processing. At a bit rate of 40 Gbps over Ethernet, an IP router should be able to perform up to 60 million lookups per second when the minimum Ethernet frame size is used [2]. At 100 Gbps, the search rate increases to 150 million lookups per second. While there are several proposals on how to increase the processing speed of these routers from a software point of view [3], [4], the physical hardware architecture needs to be considered as well in order to ensure scalability and forward compatibility, especially with regards to IP version 6 (IPv6).

IP routers rely on routing tables to find the next hop of all the packets arriving on all incoming interfaces. The routing table stores information about all the known networks that can be reached in the form of network prefixes. The table is constructed by the exchange of information with connected peers by using a routing protocol such as Open Shortest Path First (OSPF), Border Gateway Protocol (BGP) or similar. Along with each network prefix, the next hop address is stored, which is the interface port or IP address used by the router to forward the packets. The entries in the routing table used for forwarding are stored in a fast local memory in order to achieve as many lookups per second as possible. Traditionally, Dynamic Random Access Memories (DRAMs) have been used due to their low cost and energy consumption, but the use of other types of memories, such as Static Random Access Memories (SRAMs) and Content Addressable Memories (CAMs), which have much higher performance, have gained more popularity in recent years with continuous increase in chip density and reduction in price [5]–[8].

A. Background

In 1993, McAuley et al. [9] investigated the use of CAMs to increase the speed of the routing table lookups. Since then, a lot of research on how to best utilize CAMs, and especially Ternary Content Addressable Memories (TCAMs), has taken place [10]–[18]. With respect to lookup speed, TCAMs have much higher performance than conventional RAM-based approaches. Even state of the art RAM-based search mechanisms involving hash tables, search tries, bloom filters etc. require several memory accesses for each lookup [19]. When the routing table is stored in a TCAM on the other hand, only one memory access is needed to search through the entire routing table. The drawback however is that the routing table must be arranged in a certain order in the TCAM for correct Longest Prefix Matching. As a consequence, a single route update can trigger a domino effect where large portions of the TCAM needs to be updated, during which the forwarding engine will be offline. With core routers receiving many updates per second this is a serious issue, causing latency jitter and packet loss, and a lot of research has therefore
focused on how to minimize the impact of having to store the routing table in this certain order [11]–[13], [15], [20]. At the same time, research has focused on how to make efficient memory structures that are most suitable for IP routing tables. Especially the power efficiency of TCAMs is an important topic, as this has traditionally been several times higher than for RAMs of equal size [11].

The required ordering of the routing table comes from the fact that a TCAM will produce multiple results when queryed for an IP address which belongs to multiple network prefixes. In order to distinguish between these results, they must each be given a priority, which is traditionally given implicitly by their location in the TCAM such that entries with lower addresses have higher priority. Unfortunately, this makes it more difficult to update the routing table, as correct ordering must be constantly maintained. The ordering of the forwarding tables implies that the prefixes in the table must be sorted according to the length of the prefixes, such that the longer prefixes appear prior to the shorter prefixes in the TCAM and thus take precedence. This is due to the Longest Prefix Match operation requiring the longer prefix to be used in case of multiple matches. Some methods have been published on how to completely avoid the ordering of the forwarding tables in TCAMs. These ideas typically rely on either separating the forwarding table into different blocks of TCAM based on prefix length [11]–[15], on multiple memory accesses to the same TCAM [9]–[21], or on comparison of the lengths of the prefixes [18], [22]. There are advantages and disadvantages to all of these methods. Using several TCAMs may not be very efficient as some prefix lengths will have more entries than others, having to search the same TCAM multiple times reduces the throughput and increases the per packet power consumption while comparing the lengths of the prefixes requires additional logic.

This paper describes a novel method for completely avoiding the ordering of prefixes in a TCAM-based routing table by modifying the priority encoder used in the generic TCAM design [23] to perform position independent LPM as part of the match address encoding. Our scheme, the Comparator Network LPM (CN-LPM), adds a small area overhead for LPM, but also makes it possible to utilize the TCAM completely since no guard space or table separation is required to support incremental updates. Likewise, the CN-LPM only requires a single TCAM lookup to resolve the next-hop address which improves both the throughput and the energy efficiency of the forwarding engine compared to schemes relying on multiple memory accesses. As with the generic TCAM design, the area and the power consumption of the proposed scheme scales linearly the with number of prefixes \(O(N)\). The rest of the paper is organized as follows. Section II describes the typical design of a TCAM-based IP forwarding engine as well as the advantages and drawbacks of such a system. This forms the basis for Section III, which illustrates how the performance can be dramatically improved by modifying the TCAM to better suit the particular application of LPM. The section describes the CN-LPM solution proposed in this paper and compares it to an existing design by Gamache et al. [22], which uses the same basic concept of enhancing the priority encoder to perform LPM. The Gamache LPM (G-LPM) is used as a benchmark for our design in terms of resource utilization, power efficiency, performance, scalability and ease of implementation. This is described in Section IV, which also compares the two circuits to the standard priority encoder commonly used in TCAM designs to perform address encoding and multi-match resolution [23]. Section V summarizes the results and concludes the paper.

II. TCAM-BASED IP FORWARDING ENGINE

A TCAM based IP forwarding engine can be designed as depicted in Fig. 1. The main components are the TCAM which contains all the network prefixes known to the router, and a Random Access Memory (RAM) which contains the corresponding next hop addresses. Together these two components compose the routing table, which resolves the next hop addresses based on the IP address of incoming IP packets. A lookup operation is performed by scanning through the TCAM for the longest prefix which matches the incoming IP address. The resulting TCAM address is then used to access the RAM which returns the next hop address for that particular prefix. A TCAM is used as opposed to the simpler CAM because TCAMs support wildcards or \(don't \ care\) values in the stored prefixes. Hence, a 24-bit network prefix for IP version 4 (IPv4) can be stored as a 24-bit binary string followed by 8 \(don't \ care\) bits. This way, any IP address which matches the 24-bit prefix will cause a match in that particular memory location, regardless of the last 8 bits in the address. However, as mentioned in Section I-A, this property introduces the possibility of multiple matches to a single search of which only the one with the lowest address will be presented to the
forwarding engine. This introduces the issue of keeping the table sorted based on prefix length.

III. LPM WITH MODIFIED TCAM

The generic TCAM can be divided into the matching circuit and the address encoder. The matching circuit consists of \( N \) parallel comparators - one for each address in the memory. When a search is initiated, the input data is fed to all \( N \) comparators. The output from each comparator is a single match line, which indicates if the input matches the data pattern stored in this particular memory location. Based on these \( N \) matching results, the address of the match line with the highest priority is encoded using a simple priority encoder.

The LPM circuits described in this paper are based on a modified TCAM as depicted in Fig. 4, where the basic priority encoder has been replaced by an address encoder, which resolves multiple matches based on the lengths of the stored IP prefixes as opposed to just their relative position in the TCAM. Depending on the IP version, the prefix length can be from 0 to 32 for IPv4 or from 0 to 128 for IPv6. The length 0, which will correspond to any address, is the default next hop for packets with prefixes which cannot be found in the routing table. In order to reduce the overhead of storing the prefix lengths, this special case can be implemented as a special exception outside the TCAM by means of a simple comparator and a multiplexer. Hence, in order to support LPM for IPv6, the LPM circuit needs a 7-bit vector associated with each address in the TCAM, corresponding to prefix lengths from 1 to 128. Using this information together with the output of the matching circuit, the address of the match with the longest prefix can be encoded independently of its relative position to similar prefixes in the TCAM and without querying the matching circuit multiple times. This has a significant impact on the throughput and the power consumption of the forwarding engine as well as the memory efficiency of the TCAM compared to the approaches described in Section I-A. Also, since the relative position of prefixes in the TCAM becomes irrelevant, new prefixes can be easily added or removed at any position in the TCAM without resorting the table or even considering where the information is stored. In the following subsections, we will compare an existing architecture to our novel CN-LPM architecture, both of which implements this overall concept in a different way.

A. LPM circuit by Gamache et al.

An existing in-line comparison of prefix lengths in a TCAM structure is proposed by Gamache et al. [22]. In the G-LPM, each TCAM row is followed by special LPM cells which filters out all matches except the one with the longest prefix in a step by step manner as depicted in Fig. 2. For IPv6, seven LPM cells are used for storing and processing the length of a prefix corresponding to a binary encoding of the values 1-128. The enable signals are asserted one by one for each LPM cell (prefix bit). In each of the seven stages, the LPM circuit will only pass on TCAM matches with an active prefix bit, thus eliminating shorter prefix matches. In case none of the matches have an active prefix bit, all matches are passed on to the next level. After the seven steps corresponding to the binary length of an IPv6 prefix, there will only be a single match within these prefixes. In order to improve the clock speed of the circuit the entire TCAM is divided into 168 blocks of 128 entries each for a total of 21,504 prefixes and the process described above is performed independently within each individual block to produce 168 local winners. To find the longest prefix match among the 168 blocks, the seven step comparison is repeated in a “winner’s circle” block, as depicted in Fig. 3, which performs the same operation among the local winners to find the global longest prefix match. When a winner has been found, the single active match line feeding back to the originating row will trigger a readout of the next hop address stored in the corresponding SRAM cells.
B. Comparator Network LPM circuit

The basic concept of the Comparator Network LPM (CN-LPM) depicted in Fig. 5 is to expand the individual elements of the priority encoder to take the prefix length into account. Like the basic priority encoder, these elements are placed in a tree structure. At the first level, each element takes the match lines and the lengths from two TCAM addresses as an input. Based on this information, the Least Significant Bit (LSB) of the address with the longest prefix which reports a match is encoded and passed on to the next level along with the length of the prefix. As the active match lines ripple through the levels of the tree, one additional bit is added to the encoded address at each level, until the complete address of the match with highest priority is presented at the output of the last level. Using this address, the corresponding next hop can be read from an SRAM as described in Section II and depicted in Fig. 1. The CN-LPM is in many aspects a much simpler design than the G-LPM circuit described in the previous subsection. It features a homogeneous tree structure, which can be easily expanded to support larger or smaller routing tables without changing the basic design. The system only needs \( \log_2 N \) levels to support \( N \) prefixes and is easily pipelined by inserting registers between the levels at regular intervals. While the comparator network of the CN-LPM is far superior to the simple location based priority encoder, this functionality comes at the cost of additional complexity in the individual encoding elements. The cost is an additional 7-bit comparator, a 7-bit 2-to-1 multiplexer and a few extra logic gates as illustrated in Fig. 6 and Fig. 7. This extra cost must of course be taken into account when evaluating the new design in the following sections.

IV. RESOURCE UTILIZATION AND PERFORMANCE ANALYSIS

In this section, the cost of the CN-LPM will be evaluated in terms of area, power consumption and processing speed compared to the G-LPM design proposed by Gamache et al. [22] as well as the priority encoder, which is part of the standard TCAM design. In order to do so, the three systems have been implemented in the Hardware Description Language VHDL and synthesized for a Stratix IV Field Programmable Gate Array (FPGA) from Altera. The three solutions have been compared for 256, 1,024 and 4,096 match lines with the following configurations: All comparators elements are 2x1 (thus comparing only two elements per clock cycle, though still in parallel for all input match lines) in order to achieve the highest possible operation frequency. For the Gamache implementation, the number of inputs per LPM block in both the first and second level is 32 for the results with 256 or 1,024 match lines, and 64 for the results with 4,096 match lines.

A. Logic Cells

The two designs have been compared with a traditional priority encoder in order to determine the ratio between the number of logic elements required for the LPM circuits compared to the priority encoder in an FPGA implementation. As expected, the increased functionality of the LPM circuits does not come for free. In fact, based on 1,024 match lines, the results in Table I show an increase of 771% and 789% in the number of Adaptive Look-up Tables (ALUTs) for the two LPM implementations, respectively, compared to the priority encoder for a similar number of match lines. The results
also show an increase of 334% and 370%, respectively, in the number of registers used. This also includes the registers occupied by the prefix lengths, which are stored in a total of 1,024 × 7 = 7,168 registers. Both the priority encoder and the LPM implementations scale linearly with the number of match lines, so the increase of approximately 770% ALUTs and 350% registers can be expected to remain fixed for larger TCAMs.

### Table I
**Resource utilization compared to priority encoder (PE) for 1,024 match lines.**

<table>
<thead>
<tr>
<th></th>
<th>ALUTs</th>
<th>Dedic. Logic Registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority encoder</td>
<td>2,013</td>
<td>3,061</td>
</tr>
<tr>
<td>CN-LPM</td>
<td>15,513</td>
<td>10,223</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>771%</td>
<td>334%</td>
</tr>
<tr>
<td>G-LPM</td>
<td>15,887</td>
<td>11,318</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>789%</td>
<td>370%</td>
</tr>
</tbody>
</table>

While the area increase may seem very significant compared to a standard priority encoder, they are in fact quite modest compared to the size of the TCAM’s matching circuitry, which for each prefix needs a total of 128 × 2 = 256 SRAM cells just to store the prefix and the don’t care indicators as well as a 128-bit comparator [23]. To ascertain the relation between the resource requirements for the matching circuit and the LPM circuits, the TCAM matching circuit has also been synthesized for FPGA implementation. Compared to the total lookup circuit, the resource requirements of the two multi-match resolution circuits only represent approximately 16% of the total ALUs and 4% of the total registers.

### B. Power Consumption

The power measurements were made with 256 match lines, and with a clock period of 2.776 ns (∼360 MHz). Using the same clock frequency ensures fair comparison of the power consumption of the different implementations, as the toggle rate of the signals depend on the clock frequency.

### Table II
**Dynamic thermal power dissipation with 256 match lines.**

<table>
<thead>
<tr>
<th></th>
<th>Prio. Enc.</th>
<th>CN-LPM</th>
<th>G-LPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic TPD</td>
<td>37.57mW</td>
<td>112.43mW</td>
<td>157.43mW</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>299%</td>
<td>419%</td>
<td></td>
</tr>
</tbody>
</table>

Table II shows the dynamic power consumption of the two different implementations for 256 match lines compared to the priority encoder. The static thermal power dissipation (TPD) has not included since it varies very little for FPGA implementations using the same physical device. The table shows that the CN-LPM has a dynamic TPD which is three times higher than that of the priority encoder. For the G-LPM design from [22], the dynamic TPD is over four times higher. Hence, from a power consumption perspective the CN-LPM is significantly more efficient than the G-LPM implementation by Gamache et al. [22].

### C. Performance

This subsection looks into the actual performance of the three circuits in terms of address searches per second. Since they all have a throughput of one address lookup per clock cycle, the individual performance comes down to the maximum clock frequency at which the circuits can operate. Hence, the maximum clock frequency and the address lookup rate in packets per second (pps) are interchangeable. The lookup rates are specified in Table III for both 1,024 and 4,096 match lines in order to assess how well the system performance scales with increasing table size. The results in Table III show that there is an increase of over 110 Mpps between the two LPM implementations in favour of the CN-LPM for 1,024 match lines.

### Table III
**Processing rate in million packets per second compared to PE for 1,024 and 4,096 match lines.**

<table>
<thead>
<tr>
<th></th>
<th>1,024</th>
<th>4,096</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority encoder</td>
<td>734.21Mpps</td>
<td>550.96Mpps</td>
</tr>
<tr>
<td>CN-LPM</td>
<td>411.52Mpps</td>
<td>311.24Mpps</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>56.0%</td>
<td>56.5%</td>
</tr>
<tr>
<td>G-LPM</td>
<td>300.30Mpps</td>
<td>245.76Mpps</td>
</tr>
<tr>
<td><strong>Difference</strong></td>
<td>40.9%</td>
<td>44.6%</td>
</tr>
</tbody>
</table>

For 4,096 match lines, the difference is 65.48 Mpps in favour of the CN-LPM. It can be seen that the G-LPM is generally the slower of the two implementations. A likely cause is the inherent dependence between the individual components in the first levels of the G-LPM, which makes it necessary to implement long and slow interconnection paths between the elements at the same level. Also, the design of the first levels have to be changed depending on the number of match lines which must be supported. The CN-LPM on the other hand does not suffer from any of these issues. It is a homogeneous tree structure of similar elements, each of which only depends on the aggregated results from the previous levels. This allows for a simpler design with shorter interconnection paths, which is easily expanded to support larger routing tables. As expected, the priority encoder is by far the fastest of the three designs, given that the contents of the TCAM has already been sorted based on prefix length.

### V. Conclusion

In this paper, we propose a novel method for performing Longest Prefix Matching in hardware and compare it to the G-LPM design proposal by Gamache et al. [22]. Both designs are build on a TCAM-based search engine which enables search operations in a single clock cycle. These implementations allow for pure hardware based IP lookup, and perform the LPM operation by comparing the lengths of the matching prefixes in the TCAM. The sorting of the prefixes in TCAM-based routing tables is thereby eliminated completely and updates to the TCAM can be performed in a single clock
cycle for every update. The CN-LPM scheme presented in this paper compares the prefix lengths two by two in a comparator network and is able to determine the LPM in \( \log_2 N \) clock cycles, where \( N \) is the number of match lines in the TCAM. To evaluate this method, a VHDL implementation has been made and synthesised for an FPGA. This has been compared with another VHDL implementation of the G-LPM scheme [22]. For an FPGA application, the CN-LPM method proposed in this paper is shown to perform better than the implementation of the G-LPM scheme with less power and area consumption. Compared with the standard priority encoder usually used for multi-match resolving and address encoding in a TCAM, the number of ALUTs and registers required for an FPGA implementation of the comparator network is 773% and 334% higher, respectively. However, when the resource requirements of the matching circuit of the TCAM are taken into account, the ALUT and register requirements of the CN-LPM multi-match resolver remain below 16% and 4% of the total TCAM circuit, respectively. Furthermore, our technique completely eliminates the need for guard space, which would otherwise waste valuable TCAM resources. It has been shown that the CN-LPM is both significantly faster than the G-LPM scheme, with a maximum packet forwarding rate of 411Mpps and 300Mpps, respectively, for 1,024 match lines, and uses approximately 29% less power.

The proposed CN-LPM is a simple and scalable design, which can be modified to support routing tables of different sizes and pipelined to support higher or lower clock frequencies without changing the basic design of the system. By tailoring the TCAM design specifically to the purpose of LPM IP lookup, complex solutions involving advanced sorting and placement schemes, multiple lookups and multiple TCAMs are avoided, thus resulting in a simpler and faster forwarding engine with a lower energy consumption and a lower area cost.

REFERENCES


