Technique to Eliminate Sorting in IP Packet Forwarding Devices

Raymond W. Baldwin
Electrical and Computer Engineering
University of Illinois at Chicago
rbaldw3@uic.edu

Enrico Ng
Electrical and Computer Engineering
University of Illinois at Chicago
eng3@uic.edu

Abstract

This paper will present a solution to eliminate the requirements of sorting by prefix length in IP forwarding devices using Ternary Content Addressable Memories (TCAMs). This will do away with delays arising from inserting into a sorted list. To achieve this, the routing table entries in the TCAM are split by output port. This solution requires slight modifications to current TCAMs including the elimination of the built-in encoder. Overall, the solution presented reduces the insertion problem to lookup speed while maintaining similar clock rates and storage requirements of traditional TCAMs.

1. Introduction

Routers are growing more and more complex with each passing year. They must continually support new features such as Quality of Service and Service Level Agreement monitoring. To make this even more challenging they must perform these tasks at a faster and faster pace to match new Internet connection speeds. Many routers now include a co-processor to perform the crucial task of IP lookup. This paper focuses on a new design for an IP lookup co-processor.

This paper is organized as follows. In Section 2, we present background information about routing and the use of TCAMs in routers. Section 3 states our design including block diagrams and detailed logic circuits. In Section 4, we present our results. Section 5 presents possible future work. Section 6 concludes the paper.

2. Background

TCAMs are currently used to perform the task of Longest Prefix Match (LPM) in high-end routers. Figure 1 [8] illustrates a typical TCAM. In this example, the bits 01101 represent a destination IP address and is used to search the four-entry lookup table in parallel. Entries must be stored in order by length so the longest match may be found. The middle two rows result in a match using the "X" bits as don't care states. The matching line corresponding to the lowest address is chosen by the priority encoder shown on the right. The address of the LPM is the result from the TCAM. The address is then used to look up the corresponding output port in SRAM.

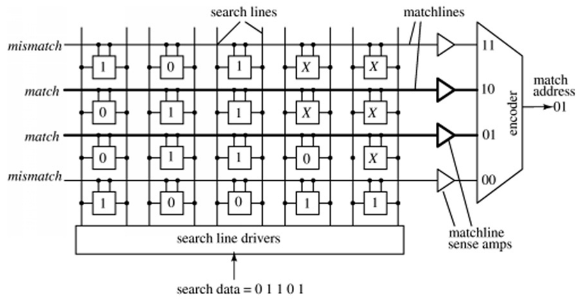

Figure 1. TCAM Diagram

There are several downsides to using TCAMs for implementation of LPM. First, TCAMs have a very high cost/density ratio [6]. This may make TCAMs less attractive than cheaper options. Second, TCAMs also have high power consumption. Having to search the entire table on each lookup causes this high power consumption [6]. These two issues can be overlooked for high-end routers since cost is not the main concern. The two most important issues are with the performance drawback when inserting a new entry into the table and the limitations imposed by the large encoder logic. We will first look at the problems from the need to maintain sorting in the table.

2.1. Insertion

Since the lookup table must remain sorted by prefix length, the insertion will take O(N) time to complete in the worst case where N represents the number of entries in the routing table. Internet core routers typically exchange three to six million updates per day, and these updates to the routing table appear to happen in bursts [7]. This means a router may get hit with several hundred prefix updates per second. On an average day, this same router will spend up to 10% of the time updating the table. Figure 2 shows a graph of the percentage of time that would be used for updates for various table sizes and update frequencies [7]. It is easy to see from this graph that insertion time has a dramatic effect on performance for routers with large tables.

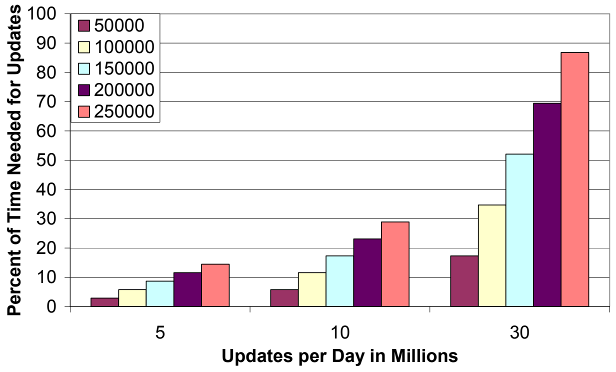

Figure 2. Effects of Updates

Figure 3 shows the percentage of time needed for updating the routing table using different table sizes [7]. With a constant insertion time of O(1), table size does not affect the graph. We can also see that O(N) insertion schemes on larger tables (>150k) basically saturate around 30 million updates since the time required to update the table would leave no additional time for IP lookup, but even with updates per day approaching 50 million, the effects of constant insertion are well below 1%. 30 million updates may seem large enough not to have to be concerned with the problem, but one study recorded a case where an Internet router received 30 million updates in one day [7]. It is also important to note that while an insertion is occurring, no lookups are possible. This means if 10% of the time is spent performing updates, the effective throughput of the router is now only 90% of its capabilities. This stresses the problems with insertion even more. This figure also shows an insertion time that takes 32 cycles. This is another option that some TCAMs use today. The improvement between the O(1) and O(32) is negligible in this graph, but it becomes a important improvement when a large burst of updates hits the router. These bursts will have no impact on single cycle insertion since it will take similar time as a lookup.

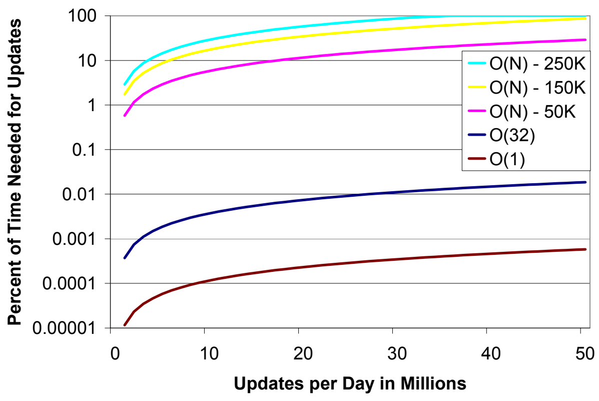

Figure 3. O(N) vs. O(1) Insertion Time

2.2. Priority Encoders

One major part of a TCAM's critical path is its priority encoder. The latency needed for this selection will result in decreased performance from the TCAM. Figure 4 shows how the encoder chooses the LPM from among all the entries in the TCAM [5]. Since the entries are stored by prefix length, the LPM is simply the lowest physical address from among the matching entries.

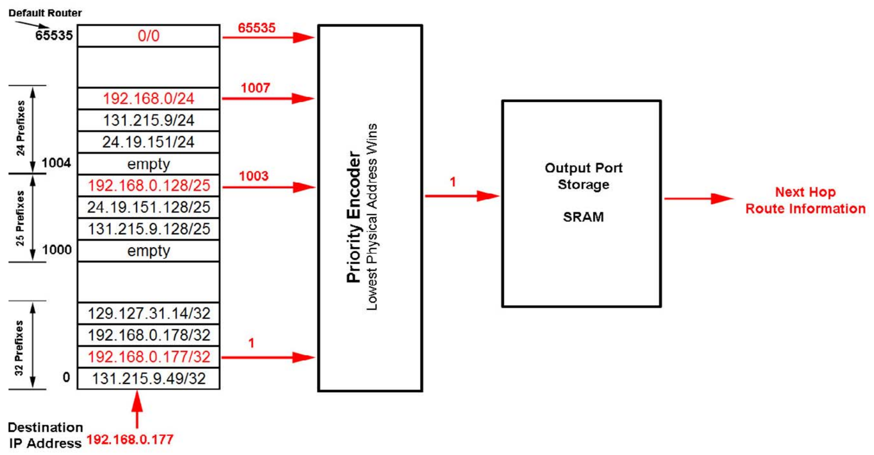

Figure 4. Longest Prefix Match Example Using TCAMs

A table can have many thousands of entries; this tends to make the delay of these large encoders very high. There are several designs that limit the time needed for the encoder, but they require large amounts of die area [4]. This introduces a tradeoff in encoder delay and size.

3. Technical Approach

Having to maintain sorting means an update to the table may result in O(N) moves. The proposed design removes this requirement, improving insertion time to O(1).

Since entries will not be stored in sorted order, a different method for locating the LPM will be used. Figure 5 shows our proposed design. The TCAM will be divided by output port into several smaller TCAMS. The number of TCAMs will now be equal to the number of output ports on the router. Each TCAM will hold a collection of all the entries that map to the output port it represents. Since all entries in a partitioned table map to the same output port, there is no longer a need to keep the entries sorted. When a search occurs each TCAM looks up the IP address in parallel. The TCAMs output all their length matches to a selection logic. After the selection logic chooses the LPM, the packet is forwarded to the output port based on which table had the longest match. When an insertion occurs, the entry is first analyzed to see which output port it matches. Then, the entry is then inserted into any open location in the corresponding TCAM.

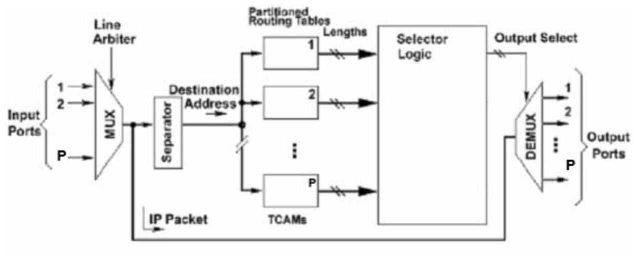

Figure 5. Proposed Architecture

The Figure 6 shows the modified design for the partition table. Typical TCAM cells are shown on the left for storing the network address prefix. The priority encoder is removed, and SRAM cells are shown on the right for storing the length. There is also no need for SRAM memory to store the output ports since this information is known after the logic selects which table contains the LPM. The encoder and decoders are not needed because the TCAM cells can be connected to the corresponding rows since they are directly related. In addition, more than one SRAM row can be asserted at once. This is possible because there can only be one match per length. If a lookup results in multiple matches of the same length, this would mean duplicate entries have been stored. Therefore, the maximum number of matches is the length of an IP address, 32 bits. Since lengths are being stored in SRAM cells instead of output ports, the number of SRAM cells needed is more than in a typical router.

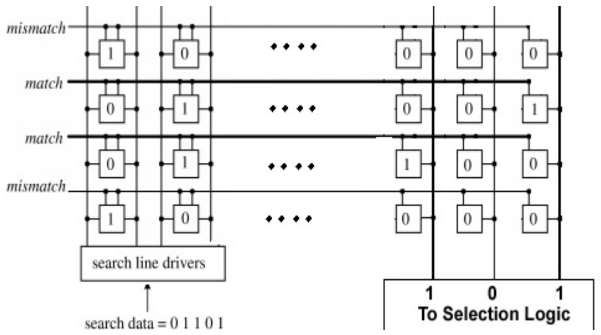

Figure 6. Modified TCAM

Figure 7 shows the Length Selection Logic (LSL). The LSL finds the length of the longest matching prefix. The logic is constructed entirely from 2-input logic gates. The first level of logic will check all the lengths for a match by ORing together the outputs from each partitioned table corresponding to each length. After identifying which lengths contain matches, the next level of logic determines the highest bit match. It can be observed from the diagram that the logic complexity depends on the number of possible lengths and the number of output ports. This logic takes (32*P) inputs and outputs 32 lines where P is the number of output ports. The outputs from the LSL connect to the next level of logic. This is shown in Figure 8.

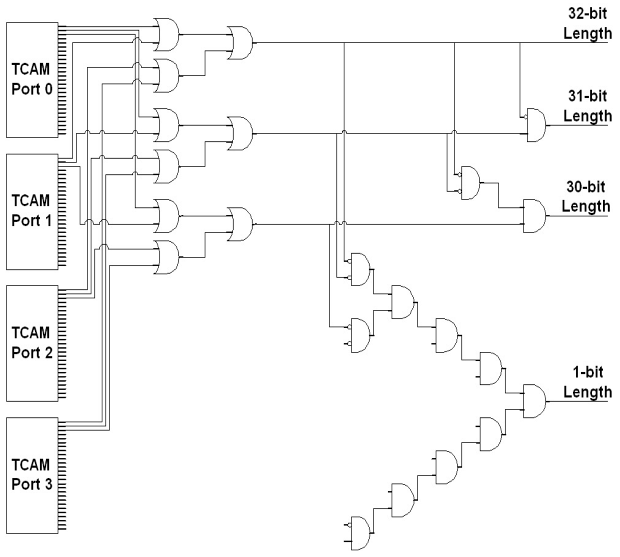

Figure 7. Length Selection Logic

Figure 8 shows the Port Selection Logic (PSL). After the highest length is found, we then need to identify the table that had this match to determine the output port. The PSL simply checks to see which port the highest length is associated with by taking 32 inputs from the LSL and 32*P inputs from the individual TCAMs. Only 1-bit of the 32 will be high to indicate the longest prefix length. The logic contains P output lines. The logic will set one of these lines high to indicate the direction the IP packet should take. This level of logic also depends on the IP address length and the number of output ports. After the port is found, the packet can be forwarded to the correct port.

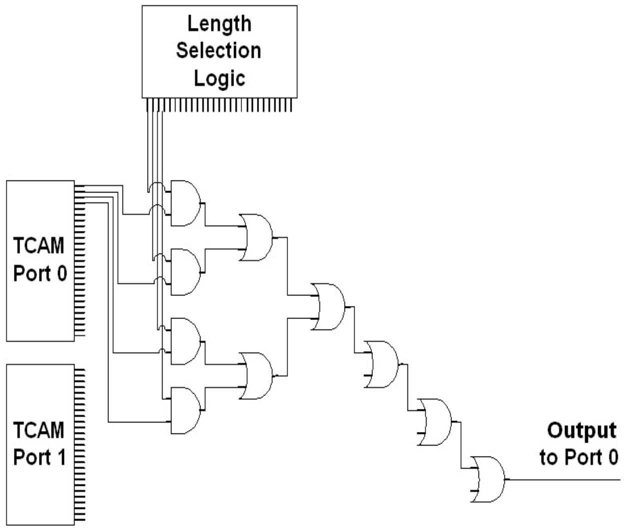

Figure 8. Port Selection Logic

Partitioning the table will decrease the storage efficiency unless additional steps are taken. This would occur since each TCAM must contain enough space to hold all the entries for that output port. The router would effectively "run-out" of memory if any partition filled. This improvement would obviously work best when the routing table is balanced, or the routing table grows in a predictable manor. This balancing would also grow more complex for each output port added. For this reason, routers with relatively few output ports would work best with this design.

4. Results

The main improvement of our design is with insertion time. The entries in the table no longer need to be sorted, so worst-case insertion time is reduced to a constant. As shown in the Figures 2 and 3, update time can have a severe impact. Since updates tend to come in groups, a large number of updates in a short amount of time can severely slow the router's performance [7]. Table size is expected to grow in the future at an exponential rate, as shown in Figure 9, and the update time will grow linear with table size [1]. The impact is examined more closely in this section.

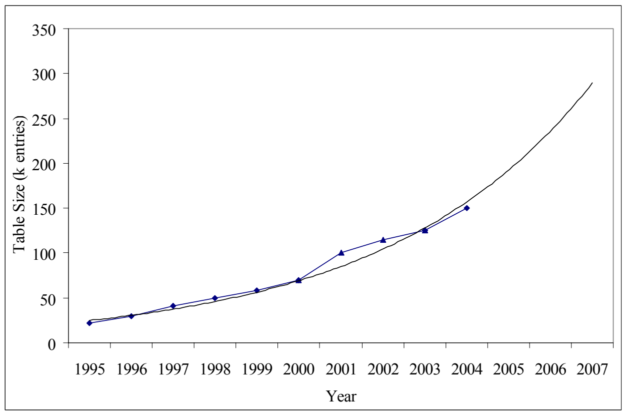

Figure 9. Table Growth Projection

Qualitative analysis of the proposed design is compared to a typical co-processor design. The typical design for this comparison contains a standard TCAM capable of holding up to 256 thousand entries [10]. Typical core routers contain between four and sixteen output ports. For this analysis, sixteen output ports are used. Another assumption is that packets are distributed evenly over output ports. This avoids problems that could occur if a partitioned TCAM filled.

Table 1 shows estimates for transistor counts and gate delays. These estimates are based off similar designs and our own calculations. TCAM cells require 16 transistors, and SRAM cells require six transistors. Each prefix stored in the table is of length L. This is 32 for IPv4 and 128 for IPv6. N represents the total number of entries stored in the routing table.

The number of transistors needed for the TCAM in each design is the same. In the typical design, only the output port value is stored in SRAM. The new design will require more SRAM since log P is less than L. Assuming 16 output ports, the storage requirement will be 32 / (log 16) = 8 times greater for the new approach. The number of rows of SRAM still remains the same, only the width increases in size. The equations for the priority encoder can be found in the reference article [3]. An address decoder requires 2^N AND gates of 2^N inputs each. Since we use two input gates, the gate delay becomes log (2^N) or N. In the case of the address decoder, the number of inputs is (log N) so the gate delay becomes (log N). Using the logic developed from Figures 7 and 8 we calculated the number of gates needed and the logic gate delay for the port selection logic and length selection logic.

Table 1. Estimates for Transistor Count and Gate Delay: N=Number of Entries, L=IP Length, P=Number of Output Ports

	Transistors	Gate Delay
TCAM Prefix	16*N*L	UNCHANGED
SRAM	6*N*logP : Typical 6*N*L : New Design	UNCHANGED
Priority Encoder	16*N+6*(N/4)	logN
Address Decoder	N*logN	loglogN
LSL	$L*(P-1) + (L^2-L)/2$	logP + logL
PSL	P*(2*L-1)	logL + 1

Table 2. Typical Design

	Transistors	Gate Delay
TCAM Prefix	134M	UNCHANGED
Priority Encoder	4.6M	18
Address Decoder	< 1M	5
SRAM	6.3M	UNCHANGED
Total	146M	23

Table 3. Proposed Design

	Transistors	Gate Delay
TCAM Prefix	134M	UNCHANGED
SRAM	50M	UNCHANGED
LSL	0.0006M	~18
PSL	0.001M	~12
Total	200M	30

Tables 2 and 3 compare logic complexity between the two designs. The design of the TCAM and SRAM cells are the same for each design. The shaded rows show the differences between the two logic designs with respect to size and speed.

We will look at differences in speed between the designs. The speed of the first TCAM lookup, to find matching rows, remains the same since the design for the TCAM cells have not been changed. The time it takes to read a SRAM cell also has not changed since only the block width has changed. The main difference in speed comes from the difference in encoder/decoder and LSL/PSL. The logic for the encoder/decoder is using the latest high-speed technology that sacrifices space for speed [4]. Our estimates were taken from the logic developed in Figures 7 and 8. To get a rough estimate for CMOS gate delays we multiply the number of logic gates along the longest path by two. This gives us a general estimate of the speed. Further calculations will be needed to get an exact number. We expect that the CMOS gates will perform better than this. We can see from the above table that our speed is slower but comparable to the typical design with future work. We must stress that these numbers are estimates, and numbers that are more accurate can be found with further research.

By comparing the size and speed of each design, we see our design trades some size for better insertion speed. A comparison between these two approaches in presented in Figure 10. Note that the major change is the replacement of the encoder with the selection logic and the placement of the SRAM lookup stage.

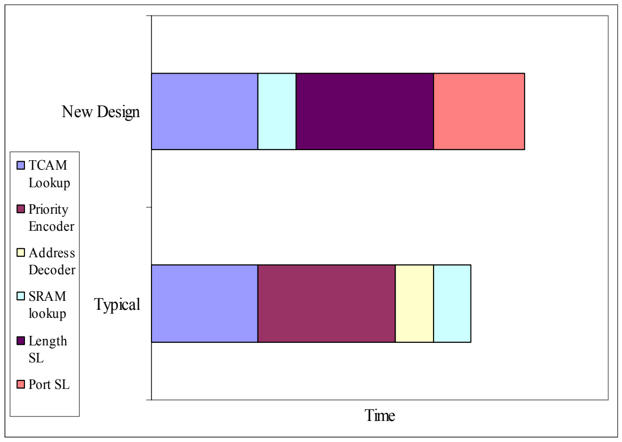

Figure 10. Factors Effecting Timing

5. Future Work

Several possible improvements can be made to our design. Currently, 32 bits are being used for each entry to store the prefix lengths. This is because the maximum prefix length is 32. In reality, 24-bits would be enough to store all lengths in the table since lengths less than 8-bits never occur. Several other routing techniques use this same approach [6]. Using only 24-bits would decrease storage requirements in SRAM and increase the speed of the selection logic.

The selection logic has some similarities to the priority encoder logic. Current priority encoders take advantage of techniques like priority lookahead to improve gate delay [4]. Similar techniques can be applied the selection logic design to improve total gate delay.

The design is naturally broken down into four parts. The initial TCAM prefix lookup, the length output, and the two levels of selection logic. A pipeline can be implemented since each of these sections could be made independent. The pipeline implementation should greatly improve the performance and allow improved latency over the original design.

The quantitative values given above are estimates based on current designs and calculations. Future work will require simulating this design to extract more accurate results for an improved quantitative comparison.

6. Conclusion

A main purpose of IP routers is to forward packets correctly by performing routing table lookups. As the wire speed increases and more routes are added, it grows increasingly more difficult to scale up based on current designs. The main contribution of this design is to improve update time for TCAMs by allowing the table to be stored in an unsorted order. This reduces the worst-case update time from O(N) to O(1) time. This reduction forced a change in the design for lookups. Lengths are now stored via SRAM in the table. The logic for the priority encoder and other logic have been replaced with simpler selection logic. The current values discussed in Section 4 are only simple estimates for size and speed. Exact results were not available since they have not been simulated yet. Overall, our design shows a large improvement in insertion time without greatly increasing lookup time.

7. Acknowledgments

We would like to acknowledge Prof. Gyungho Lee, at University of Illinois at Chicago, for his help with this paper.

References

[1] Geoff Huston, “AS1221 – BGP Table Statistics”, http://bgp.potaroo.net/as1221/bgp-active.html

[2] Mohammad J. Akhbarizadeh and Mehrdad Nourani, “An IP Packet Forwarding Technique Based on Partitioned Lookup Table”, ICC 2002 - IEEE International Conference on Communications, no. 1, April 2002 pp. 2263-2267.

[3] JG Delgado-Frias and J. Nyathi, “A high-performance encoder with priority lookahead”, IEEE Trans. Circuit Sys. I, vol.47, pp.1390-1393, Sept. 2000.

[4] C.H. Huang, J.S. Wang, and Y.C. Huang, “Design of high-performance CMOS priority encoders and incrementer/decrementers using multilevel lookahead and multilevel folding techniques”, IEEE Journal of Solid-State Circuits, 37(1):63-76, January 2002.

[5] Mike Ichiriu, “High Performance Layer 3 Forwarding, The Need for Dedicated Hardware Solutions”, 2000.

[6] Stefanos Kaxiras and Georgios Keramidas, “IPStash: a Power-Efficient Memory Architecture for IP-lookup”, IEEE/ACM International Symposium on Microarchitecture, no 36, 2003.

[7] C. Labovitz, R. Malan, and F. Jahanian, "Internet routing instability", IEEE/ACM Trans. Networking, vol. 6, no. 5, pp. 515-558, 1998.

[8] Kostas Pagiamtzis, “CAM Primer”, http://www.eecg.toronto.edu/~pagiamt/cam/camintro.html

[9] D. Shah and P. Gupta, "Fast incremental updates on Ternary-CAMs for routing lookups and packet classification", Proc. of Hot Interconnects-8, Stanford, CA, USA, Aug. 2000.

[10] SiberCore Technologies, “Ultra-18M SCT1842 Product Brief”, http://www.sibercore.com/products_siberCAM.htm#3

[11] Francis Zane, Girija Narlikar, and Anindya Basu, “CoolCAMs: Power-Efficient TCAMs for Forwarding Engines”, IEEE Infocom, 2003.