

A Hybrid Approach to CAM-Based Longest Prefix Matching for IP Route Lookup

Yan Sun and Min Sik Kim
 School of Electrical Engineering and Computer Science
 Washington State University
 Pullman, Washington 99164-2752, U.S.A.
 Email: {ysun,msk}@eecs.wsu.edu

Abstract—Ternary Content Addressable Memories (CAMs) are widely used by high-speed routers to find matching routes in a routing table, because they enable the longest prefix matching operation to complete in a single clock cycle. However, they are costly and their power consumption is very high. In this paper, we identify two kinds of redundancy in the usage of TCAMs in IP route lookup, and then propose a hybrid scheme which combines Binary CAMs and Ternary CAMs to reduce the total area and power consumption, exploiting the uneven distribution of IP prefix lengths in real-world IP routing tables. We also introduce shared memory blocks for further simplification of the lookup circuit. The simulation results show that our approach can save more than 50% of transistors in CAMs, compared with the traditional way in storing a set of real-world routing tables, and that it reduces the critical path in IP route lookup significantly.

I. INTRODUCTION

Rapid expansion of the Internet has led to the exponential growth of routing tables in routers, and the longest prefix matching in IP route lookup is often the bottleneck in today’s routers with such large routing tables. Thus, designing an efficient scheme to perform longest prefix matching operations is a critical problem in high-speed routers. A routing table in a router stores variable-length IP address prefixes and corresponding outgoing ports. Table I shows a simple routing table with four IP address prefixes.

TABLE I
 A SIMPLE ROUTING TABLE

Destination IP Address	Mask	Port
152.168.22.0	/24	5
152.168.30.0	/24	1
132.165.0.0	/16	8
122.128.0.0	/18	4

When a packet arrives, a router searches for the longest prefix match for the destination IP address of the packet, and then retrieve the corresponding outgoing port. Since this procedure has been the bottleneck of routers’ performance, many approaches have been proposed by researchers, both software-based and hardware-based ones.

Ternary Content Addressable Memories (TCAMs) have been widely adopted by routers to improve the speed of the longest prefix matching. They allow the “don’t care” state to be stored in each memory cell as well as binary states 0 and 1. A memory cell in a “don’t care” state matches both 0 and 1 in the corresponding input bit. A TCAM-based routing table is extremely fast because it allows the input key to compare

with all the prefixes stored in the TCAM simultaneously and retrieve the result in a single clock cycle. The architecture of a TCAM used in the longest prefix matching is shown in Fig. 1.

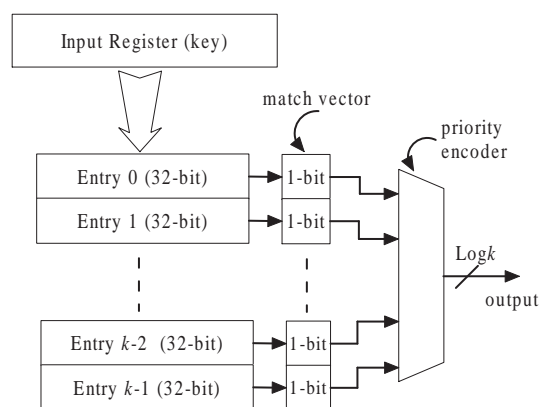


Fig. 1. TCAMs used in the Routing Tables

A key (destination IP address) is stored in the input register and each prefix is stored in a single entry. The key compares with all the prefixes in parallel and the results are stored in the match vector, where 1’s represent the corresponding entries match the key, and the priority encoder chooses the longest prefix match. At last, the output signal is used to find the corresponding outgoing port. While the TCAM-based search is very fast, TCAMs have two major disadvantages: high cost and high power consumption. In fact, both of them result from the circuit complexity of each TCAM cell. The high power consumption also affects the total cost and performance of routers, not only because it increases the power supply and cooling costs but also it reduces the port density since more space is needed between ports for cooling purpose. Therefore, how to use TCAMs efficiently becomes a critical issue, and many methods have been proposed to reduce TCAM requirements for a given set of prefixes. Compared with TCAMs, Binary CAMs (BCAMs) require many fewer transistors and less power because no mask is needed and the comparison circuit is simpler. However, they can store only 0 and 1; they don’t have the “don’t care” state.

In this paper, we identify two kinds of redundancy in the TCAM-based IP route lookup. Then we build a hybrid scheme that eliminates such redundancy by dividing all the input keys into seven groups, each of which is handled separately, taking advantages of both Binary and Ternary CAMs. We evaluate

our scheme by measuring savings in terms of the number of transistors when it is applied to real-world routing tables.

The remainder of this paper is organized as follows. Section II surveys related work on longest prefix matching algorithms. Section III details the design of the proposed hybrid scheme for longest prefix matching in IP route lookup. Section IV evaluates the savings in our hybrid scheme. Section V concludes the paper.

II. RELATED WORK

Many software-based longest prefix matching techniques have been proposed [1]–[4]. These techniques use algorithms such as hashing, trees, and tries to reduce computational complexity. While these approaches can reduce the number of memory accesses for a single input key, they usually cannot meet the requirement of today’s high-speed forwarding.

Among hardware-based longest prefix matching techniques, the CAM-based ones dominate the high-speed router market, especially of multi-gigabits per second and faster routers. A disadvantage is that the TCAM chips are expensive and power-hungry. Therefore, techniques have been proposed to use TCAM entries more efficiently or to reduce the required amount of TCAMs [5]–[8]. These approaches usually need to pre-process prefixes to compress them and then configure into TCAMs. However, these approaches should undergo a very complex update process, because they have to pre-process the set of prefixes again. Other approaches focus on the TCAM cell itself and try to reduce the number of transistors and power consumed by a single TCAM cell [9]–[11]. They usually need to modify the circuit within a TCAM cell or within a TCAM entry.

Our goal is to reduce the usage to TCAMs in IP route lookup without modifying the TCAM circuit itself. We achieve it by adding a small extra logic circuit, which consumes negligible area and power compared with TCAMs.

III. HYBRID APPROACH USING BCAMS AND TCAMS

A. Redundancy in Using of TCAMs for IP Route Lookup

Because IP route lookup is longest prefix matching, each 32-bit entry in a TCAM consists of the prefix part and the following “don’t care” bits. This implies two kinds of redundancy as follows:

- Each bit in the prefix part has two possible values only: 0 and 1; there is no “don’t care” bit in the prefix. Therefore, a Binary CAM should suffice to match the prefix.
- No comparison is needed for the “don’t care” suffix, since they always match the input.

Based on these observations, we propose a new approach to mitigate both kinds of waste.

B. Real-world IPv4 Prefix Distribution

In this section, we analyze the distribution of real-world prefixes. We collect the routing tables from the Route Views project [12]. In order to ensure that the characteristics of the distributions are not specific to some particular routers or time intervals, we inspect many routing tables and finally select four typical sets of routing table information from year 1997 to year 2009, where each set consists of all the data in a single typical

TABLE II
FOUR REAL-WORLD ROUTING TABLES

Name of Routing Tables	Date of Collection	Number of Prefixes
Oix-1997	11/08/1997	19,017 ($10^{4.28}$)
Oix-2001	08/01/2001	53,842 ($10^{4.73}$)
Oix-2005	08/01/2005	107,679 ($10^{5.03}$)
Oix-2009	08/19/2009	10,471,325 ($10^{7.02}$)

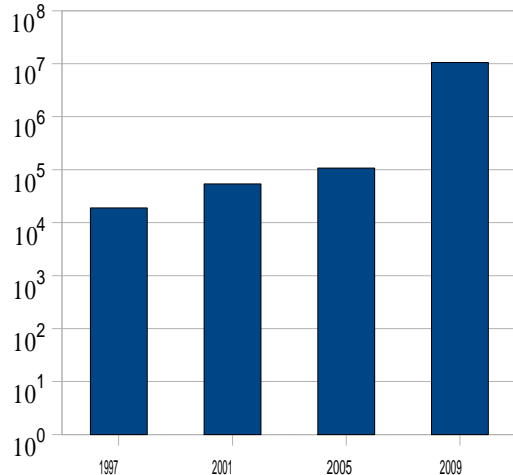


Fig. 2. The Number of Prefixes in a Real Typical Network

day. The information regarding the selected routing tables are summarized in Table II. The numbers of prefixes of the selected routing tables are plotted in Fig. 2. (Note the logarithmic scale on the vertical axis).

We can see that the number of prefixes is growing exponentially, and is even growing faster in recent four years. This indicates that the longest prefix matching will remain to be the bottleneck in high-speed routers.

The distributions of prefixes of these four sets of routing tables are shown in Fig. 3. The horizontal axis is the length of prefixes and the vertical axis is the number of prefixes. We can see that the distributions in the four different years are similar, and the trend is that the percentages of prefixes shorter than 16-bit and longer than 24-bit are decreasing. It is this trend that provides room to design a more efficient algorithm. We can see that in recent routing tables, the 24-bit Class C Prefixes dominate the number of prefixes (about 50% alone), and over 90% of the prefixes are between 18-bit and 24-bit long. Based on these observations, we propose a scheme to utilize CAMs more efficiently.

C. Proposed Scheme

Based on the observations of TCAM redundancy and real-world IPv4 prefix distribution discussed above, we propose our approach to reducing the usage of TCAMs in IP route lookup. Given IP address prefixes with the maximum length of 32 bits, we divide them into seven categories, P_1 , P_2 , P_3 , P_4 , P_5 , P_6 , and P_7 , according to the prefix lengths. All 8-bit long prefixes are added to P_1 , 9 to 15-bit long prefixes to P_2 , 16-bit to P_3 , 17 to 23-bit to P_4 , 24-bit to P_5 , 25 to 31-bit to P_6 , and 32-bit to P_7 . For the categories P_1 , P_3 , P_5 , and P_7 , the length

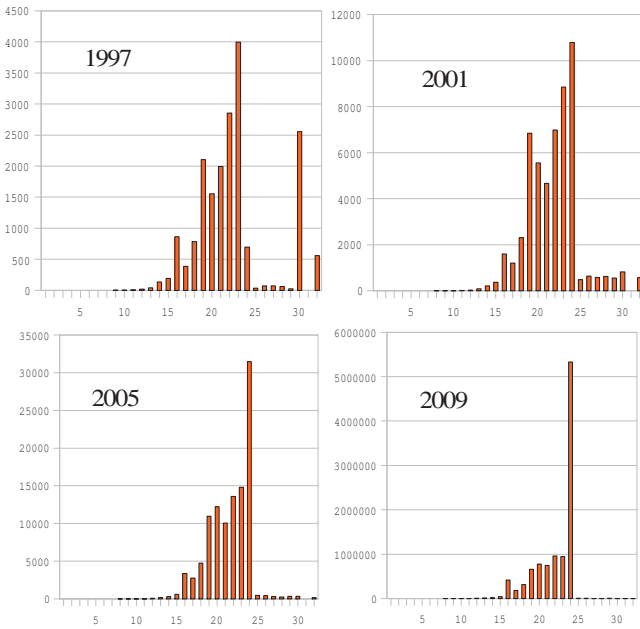


Fig. 3. The Distribution of Prefix Lengths in the Past Twelve Years

of prefixes is unique, and thus Binary CAMs, one for each category, should suffice to perform prefix matching. On the other hand, for the categories P_2 , P_4 , and P_6 , we need to split each prefix into the fixed-length prefix part and the remainder part, the former containing binary values which can be stored in a Binary CAM, and the latter containing “don’t care” bits which should be stored in a Ternary CAM. For example, since P_2 has prefixes of length between 9 and 15, the first 9 bits are stored in a Binary CAM while the remaining bits are stored in a Ternary CAM. To combine the results from two CAMs, we need to perform an AND operation between the two to see whether an entry matches the destination IP address of a given packet. For further improvement, we pipeline our scheme as shown in Fig. 4.

In the first stage, the routing table update prefixes are classified into seven categories. The second stage consists of BCAMs and TCAMs, and both updating the routing table and handling incoming packets are performed in this stage. All the TCAM blocks are 7 bits wide and the four BCAM blocks are 8 bits, 16 bits, 24 bits, and 32 bits wide, respectively. Because a 32-bit destination IP address is divided into two parts if the length of prefix is not a multiple of 8, we need to combine the results of these two parts together using AND logic in the third stage to get the matching result. As shown in Fig. 4, the seven categories are divided into four groups, G_0 , G_1 , G_2 , and G_3 , containing P_1 and P_2 , P_3 and P_4 , P_5 and P_6 , and P_7 , respectively. Based on the longest prefix matching policy, each group chooses the matching bit with the highest priority, G_3 being the highest and G_0 the lowest. The three Priority Encoders used by the three groups can operate in parallel to reduce the latency of stage 3.

For each routing lookup entry, we need at most a 7-bit-wide TCAM. When the length of the prefixes is a multiple of 8, no TCAM is needed and only BCAM entries are used. When

the length of entries is shorter or the TCAM is replaced by a BCAM, the memory access latency is reduced, especially for write operations, which result in higher clock speed.

One of major disadvantages of TCAM-based approaches is that all the prefixes stored in TCAM must in the order of increasing prefix length because the longest prefix policy is implemented using a priority encoder. In our approach, sorting overhead is significantly lower because four smaller sorting operations are performed in parallel.

In Fig. 4, different groups use different TCAM blocks. For further improvement, we can use a single TCAM block with dynamic boundaries between different groups.

D. Shared CAM

The distribution of prefix lengths varies. Spatially, core routers have more short prefixes than edge routers. Temporally, the percentages of entries in the four groups change over time as shown in Fig. 3. To make our approach more flexible and efficient in utilizing CAM resources, we introduce the shared CAM mechanism, where four groups share the same memory. In this mechanism, we let entries from P_3 , P_4 , and P_7 share a BCAM block, and entries from P_1 , P_2 , P_5 , and P_6 share another BCAM block. For the first share group, each BCAM entry has 16 bits, and each element in P_3 and P_4 only uses one 16-bit entry while each element in P_7 uses two adjacent 16-bit entries. They are depicted in Fig. 5. A 32-bit IP address is stored in a pair of 16-bit BCAM entries, starting from the pair with the highest address. A 16-bit prefix is stored in a single 16-bit BCAM entry, starting from the entry with the lowest address.

Similarly, the 24-bit binary prefixes and 8-bit binary prefixes share the same resources. We use a 8-bit-wide BCAM block to locate both prefixes as shown in Fig. 6. A 24-bit prefix is stored in a triple of three BCAM entries, starting from

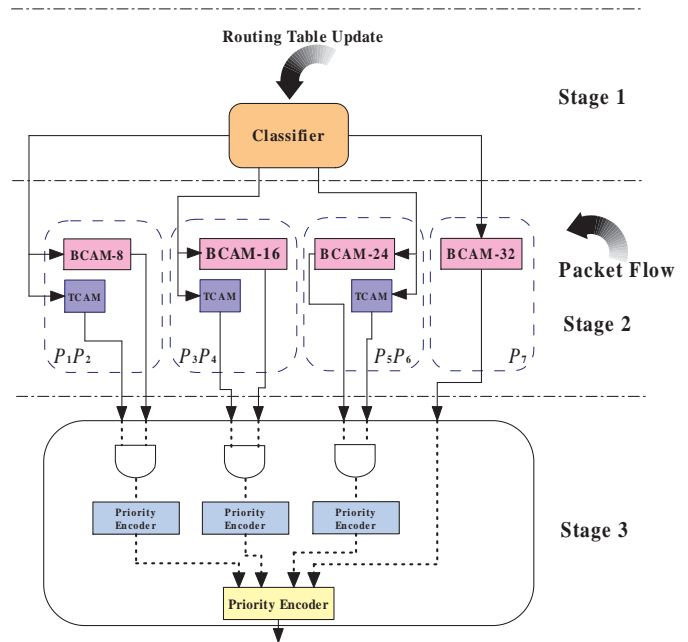


Fig. 4. The Architecture of the Hybrid Approach

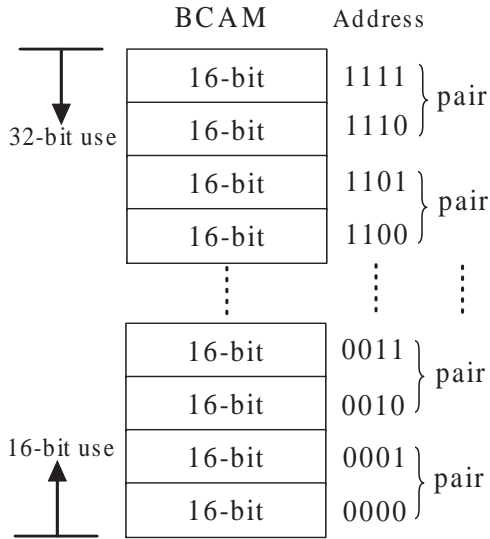


Fig. 5. BCAM Shared by 32-bit and 16-bit Prefixes

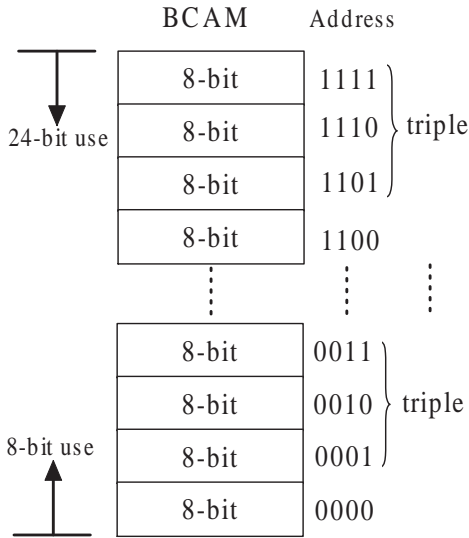


Fig. 6. BCAM Shared by 24-bit and 8-bit Prefixes

the triple with the highest address. A 8-bit prefix is stored in a single 8-bit BCAM entry, starting from the entry with the lowest address.

The two 8-bit entries in a single *pair* have different search operations, which are shown in Fig. 7. The value of “Sel” in Fig. 7 is based on whether this block is used by a 32-bit prefix.

Similarly, the three 8-bit entries in a *triple* have different search operations, which are shown in Fig. 8. The value of “Sel” in Fig. 8 is based on whether this block is used by a 24-bit prefix.

IV. EVALUATION

A. Transistors Savings

In a Binary CAM entry cell, the content can be made up of binary bits, each of which has either 0 or 1. In a Ternary CAM cell, however, a third “don’t care” state can be used as

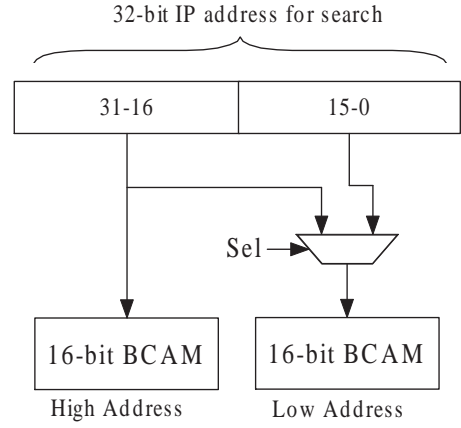


Fig. 7. IP address selected by the shared BCAM for P_3 , P_4 and P_7 in the Search Operation

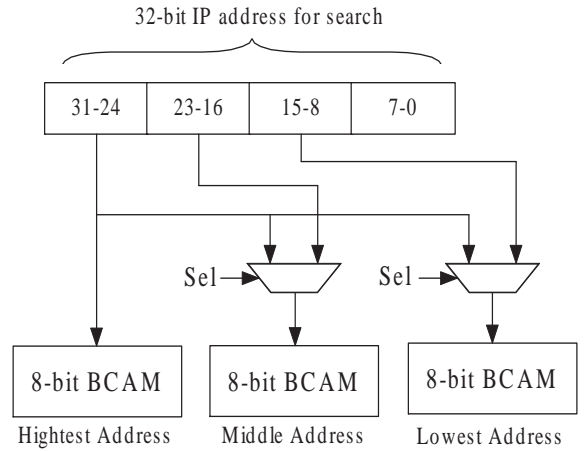


Fig. 8. IP address selected by the shared BCAM for P_1 , P_2 , P_5 and P_6 in the Search Operation

a bit value. An entry of a Ternary CAM stores content as a (value, mask) pair, where value and mask are W -bit numbers, requiring W storage cells for the value and additional W storage cells for the mask. Moreover, the matching circuitry is more complicated than that of a Binary CAM.

A typical TCAM cell requires six transistors as a SRAM cell. The same number of transistors are required to store the mask bit, and four transistors for the match logic. Thus, each TCAM cell requires 16 transistors, which is about 2.7 times of a typical SRAM cell. However, different techniques used by CAM manufactures result in different numbers [13]. For the evaluation of our scheme in this paper, we assume that the number of transistors and power consumption of a TCAM are two times as large as those of a BCAM cell on average.

Table III shows the transistors saved under different lengths of prefixes, compared with ordinary TCAM-based routing tables. On average, it can save about 60.7% of transistors.

B. Simulation with Real-World IPv4 Prefix Distribution

We use the data collected in year 2009 shown in Fig. 3 to evaluate our approach with real-world IP address prefixes. We see that the 24-bit Class C Prefixes dominate the number of

TABLE III
TRANSISTORS SAVED

Prefix	BCAM used by each entry (bits)	TCAM used by each entry (bits)	Transistors Saved (%)
8-bit	8	0	87.5
9-bit to 15-bit	8	7	62.5
16-bit	16	0	75.0
17-bit to 23-bit	16	7	50.0
24-bit	24	0	62.5
25-bit to 31-bit	24	7	37.5
32-bit	32	0	50.0

TABLE IV
TRANSISTORS SAVED BY OUR APPROACH

Length of Prefix(bits)	Number of Prefix	Percentage(%)	Transistors Saved (%)
8	768	0.2368	87.5
9 to 15	93423	0.8922	62.5
16	419800	4.0090	75.0
17 to 23	4595478	43.8863	50.0
24	5326651	50.8689	62.5
25 to 31	24795	0.2368	37.5
32	258	0.0025	50.0

prefixes (about 50% alone), and over 90% of the prefixes are between 18-bit and 24-bit. Under this condition, the traditional way consumes 10,471,325 32-bit TCAM entries. The BCAM and TCAM usage by our approach is shown in Table IV.

We calculate the total transistors saved by our approach using following equation:

$$\text{TotalTransistorsSaved} = \sum_{i=1}^7 \text{TransistorsSaved}(i) \times \text{Percentage}(i) \quad (1)$$

where i represents each prefix group shown in Table IV.

According to data shown in Table IV and the equation above, we can save 57.6% transistors by reducing and replacing the TCAMs in IP route lookup, and the additional logic introduced by our approach is much smaller than typical logic in TCAMs. Thus we conclude that our approach can reduce both the area and power consumption significantly.

Our approach can be used together with existing algorithms that solve other problems of CAM-based IP route lookup, such as dealing with more routing table entries than TCAM entries. For example, our approach can be combined with software-based approaches to reduce the number of required CAM entries, because our approach is designed to be a drop-in replacement of TCAM-based longest prefix matching.

Furthermore, our approach enables different groups of prefixes to update operate in parallel, which leads to further increase of the throughput of packet processing. Using the same data collected in year 2009, our approach needs about 51% of total clock cycles used by the traditional way to update these prefixes.

Finally, using narrower TCAMs can help increase the clock speed in ASIC or FPGA. In our algorithm, the critical path

is the delay of the 32-bit-wide BCAM, which is about half of the delay of a 32-bit-wide TCAM in the SMIC 0.13 μm CMOS technology; we get the similar results using FPGAs. Therefore, the critical path of the ACL subsystem can be shorter compared with the traditional approach.

V. CONCLUSION

In this paper, we present a hybrid approach to IP route lookup using Binary CAMs to save memory usage and improve the throughput of longest prefix matching process. We treat prefixes with different lengths separately in parallel, and use different types of CAMs to take advantage of their characteristics. The simulation results show that our approach saves 57.6% of transistors, reducing the area and power consumption significantly. Our approach can double the throughput of the longest prefix matching under the same clock speed. Furthermore, the clock speed can also be increased because the paths traversing through CAMs are shorter. Because our approach provides the same interface as that of the traditional TCAM-based approach, it can be used with other preprocessing-based approaches together to increase performance further. Our approach can be easily extended to be used in IPV6 applications based on the properties of routing tables in IPV6.

REFERENCES

- [1] N. F. Tzeng, "Routing table partitioning for speedy packet lookups in scalable routers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 5, pp. 481–494, May 2006.
- [2] Y. Chang and Y. Lin, "A fast and memory efficient dynamic IP lookup algorithm based on B-Tree," in *Proceedings of International Conference on Advanced Information Networking and Applications*, vol. 0, May 2009, pp. 278–284.
- [3] L. C. Wu, T. J. Liu, and K. M. Chen, "A longest prefix first search tree for IP lookup," *Computer Networks*, vol. 51, no. 12, pp. 3354–3367, 2007.
- [4] X. H. Sun and Y. Q. Zhao, "An on-chip IP address lookup algorithm," *IEEE Transaction on Computers*, vol. 54, no. 7, pp. 873–885, 2005.
- [5] H. Liu, "Reducing routing table size using Ternary-CAM," in *Proceedings of the Ninth Symposium on High Performance Interconnects*, 2001, p. 69.
- [6] S. Stergiou and J. Jain, "Optimizing routing tables on systems-on-chip with content-addressable memories," in *Proceedings of International Symposium on System-on-Chip*, 2008, pp. 1–6.
- [7] Y. Tang, W. Lin, and B. Liu, "A TCAM index scheme for IP address lookup," in *Proceedings of First International Conference on Communications and Networking in China*, vol. 0, 2006, pp. 1–5.
- [8] K. Zheng, Z. Liu, and B. Liu, "High performance embedded route lookup coprocessor for network processors," vol. 3619, pp. 188–197, 2005.
- [9] H. Noda, K. Inoue, M. Kuroiwa, A. Amo, A. Hachisuka, H. J. Mattausch, T. Koide, S. Soeda, K. Dosaka, and K. Arimoto, "A 143mhz 1.1w 4.5mb dynamic TCAM with hierarchical searching and shift redundancy architecture," in *Proceedings of International Conference on Solid-State Circuits*, 2004, pp. 208–209.
- [10] S. Choi, K. Sohn, M. W. Lee, S. Kim, H. M. Choi, D. Kim, U. R. Cho, H. G. Byun, Y. S. Shin, and H. Yoo, "A 0.7fj/bit/search, 2.2ns search time hybrid type TCAM architecture," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 1, pp. 254–260, 2005.
- [11] J. S. Wang, H. Y. Li, C. C. Chen, and C. W. Yeh, "An AND-type match-line scheme for energy-efficient content addressable memories," in *Proceedings of International Conference on Solid-State Circuits*, 2005, pp. 464–610.
- [12] "University of Oregon Route Views Project," <http://www.routeviews.org/>.
- [13] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: a tutorial and survey," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, 2006.