IP Prefix Matching with Binary and Ternary CAMs

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 in high-speed routers. They allow a longest-prefix matching operation to complete within a single clock cycle. However, TCAMs 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 propose a hybrid scheme which combines Binary CAMs and Ternary CAMs to reduce the total area and power consumption. We also introduce shared memory blocks for further simplification of the lookup circuit. The simulation result shows that our approach can save more than 50% of transistors in CAMs, compared with the traditional way, and that it reduces the critical path in IP route lookup significantly.

I. INTRODUCTION

Longest prefix matching is the main bottleneck in high-speed routers, and how to design an efficient scheme to perform longest prefix matching operations is a fundamental problem in improving router throughput. Ternary Content Addressable Memories (TCAMs) have been widely used to address the problem. TCAMs allow the “don’t care” state to be stored in each memory cell in addition to values 0 and 1. TCAM-based routing table lookup is extremely fast and can retrieve the result in a single clock cycle. However, TCAMs have two drawbacks: high cost and high power consumption caused by the complex circuit of a TCAM cell. Thus, the efficient use of TCAMs becomes crucial in router performance, and many methods have been proposed to reduce the TCAM storage requirement for a given set of prefixes [1], [2]. Compared with TCAM, Binary CAMs (BCAMs) use much fewer transistors and consume less power because no “don’t care” bit is needed and the comparison circuit is simpler. In this paper, we take advantage of the simplicity of BCAMs to build a high-performance hybrid scheme.

II. ALGORITHM

The typical use of TCAMs in IP prefix matching has the following inefficiencies. First, the prefix part (e.g., 192.168 in 192.168.255.255/16) is matched exactly; the “don’t care” bits in TCAM are unnecessary. Second, 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 that reduces the number of transistors for the matching logic.

For 32-bit IP prefixes, we first divide all prefixes into seven categories, $P_1$, $P_2$, …, $P_7$. Their prefix lengths are 8 bits, between 9 and 15, 16, between 17 and 23, 24, between 25 and 31 bits, and 32, respectively. For the categories $P_1$, $P_2$, $P_3$, and $P_4$, the prefix lengths are fixed, which means a BCAM for each prefix should suffice for prefix matching. And for the categories $P_5$, $P_6$, and $P_7$, we compare the prefix part and the remaining “don’t care” part separately, such that the prefix part can be stored in a BCAM while the remainder is stored in a TCAM.

The results from the two CAMs are combined using the AND operation because both parts must match in order to conclude that the given address matches the prefix. This process can be pipelined to improve the throughput further, as shown in Fig. 1.

In the first stage, the prefixes are grouped into seven categories. The second stage consists of BCAMs and TCAMs, and incoming packets are processed in this stage. All the TCAM entries are 7 bits long and BCAM entries are 8, 16, 24, and 32 bits long, respectively. Because each 32-bit destination IP address is divided into two parts, one for BCAMs and the other for TCAMs, we need to combine the results of two parts in the third stage using AND operations. As shown in Fig. 1, the seven categories are classified into four groups: $G_0$, $G_1$, $G_2$, and $G_3$, containing $P_1$, $P_2$, $P_3$, and $P_4$, $P_5$, $P_6$, and $P_7$, respectively. Based on the longest prefix matching policy, each group will choose the matching bit with the highest priority, $G_3$ being the highest and $G_0$ the lowest. One of major disadvantages of TCAM is that all the prefixes stored in TCAM must be ordered by prefix length because the longest prefix policy is implemented by priority encoders, which is a considerable task for a large routing table. In Fig. 1, although it appears that different groups use different TCAM blocks, we can use a single TCAM block and have dynamic boundaries between different groups, resulting in more efficient use of TCAMs.

For further flexibility, we introduce the shared BCAM mechanism. It lets some entries from $G_1$ and $G_3$ share a BCAM block, and some entries from other groups share another BCAM block. For the first BCAM block, each entry has 16 bits, and each
element in $P_3$ and $P_4$ only uses one 16-bit entry, while each element in $P_1$ uses two adjacent 16-bit entries. This is shown in Fig. 2(a). 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.

The two 8-bit entries in a pair have different search operations, as shown in Fig. 2(b). The value of “Sel” in Fig. 2(b) is based on whether this block is used by a 32-bit prefix. The 24-bit binary prefix and 8-bit binary prefix will share the same resources and the shared BCAM has the similar behavior as the previous one.

III. EVALUATION

A typical TCAM cell requires six transistors as a SRAM cell, the same number of transistors to store the mask bit, and four other transistors for the match logic. Thus, each TCAM cell requires 16 transistors, which is about 2.7 times the number of transistors in a typical SRAM cell. However, different techniques result in different numbers. For the evaluation of our scheme, we assume the number of transistors and the amount of power consumption of a TCAM are two times as large as those of a BCAM cell.

We collect routing tables from the Route Views Project [3]. In order to ensure that the characteristics of the distributions are not specific to some particular routers or time interval, we inspect many routing tables and finally select four representative sets of routing table information from year 1997 to year 2009, where each set consists of all the data in a single typical day.

The number of prefixes is growing exponentially, and is even growing faster in the recent four years. This indicates that the longest prefix matching remains to be a bottleneck in high-speed routers.

Fig. 3 shows the prefix length distribution since 1997. The horizontal axis is the prefix length, and the vertical axis is the number of prefixes. Note that the percentages of prefixes shorter than 16-bit and longer than 24-bit are decreasing, which makes our algorithm more efficient. Also note that in the recent routing tables, the 24-bit Class C prefixes dominate (about 50%), and more than 90% of the prefixes are between 18 bits and 24 bits long. Under this condition, a traditional way consumes 10,471,325 32-bit TCAM entries, while our algorithm can save 57.6% of transistors by reducing and replacing TCAMs. The full results are shown in Table I. The logic added by our algorithm such as AND operations is minimal, compared with TCAM sizes in terms of the number of transistors. Hence, we conclude that our algorithm decreases both the area and power consumption significantly.

Furthermore, pipelining and parallelization in our algorithm increase the throughput of the lookup process. Using the data collected in 2009, our algorithm needs about 51% of total clock cycles used by the traditional way in updating the prefixes. Finally, using narrower TCAMs can help increasing the clock speed in ASIC. In our algorithm, the critical path is the delay of the 32-bit-wide BCAM, which is about half of the delay of the 32-bit-wide TCAM in SMIC 0.13 μm CMOS technology.

IV. CONCLUSION

We present an IP prefix matching algorithm using both Binary and Ternary CAMs to save memory usage and improve throughput. It treats prefixes with different lengths separately and can operate in parallel. It also uses different types of CAMs to take advantage of their characteristics. The simulation results show that our algorithm saves 57.6% transistors of CAMs, decreasing the area and power consumption significantly. Under the same clock speed, our algorithm can double the throughput of the longest prefix matching. Besides, the clock speed can also be increased because our algorithm uses simpler, smaller CAMs.

REFERENCES

[1] H. Liu, “Reducing routing table size using ternary-cam,” in Proceedings of the 9th Annual IEEE Symposium on High-Performance Interconnects, Aug. 2001.
[2] S. Stergiou and J. Jain, “Optimizing routing tables on systems-on-chip with contentaddressable memories,” in Proceedings of International Symposium on System-on-Chip 2008, Nov. 2008, pp. 1–6.
[3] “University of Oregon Route Views Project.” [Online]. Available: http://www.routeviews.org/