A Hardware-Based Longest Prefix Matching Scheme for TCAMs

Soraya Kasnavi and Vincent C. Gaudet
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, AB, T6G 2V4
Email: kasnavi, vgaudet@ece.ualberta.ca

Paul Berube and José Nelson Amaral
Department of Computer Science
University of Alberta
Edmonton, AB, T6G 2E8
Email: berube, amaral@cs.ualberta.ca

***Abstract*—$Ternary\ \ Content\ \ Addressable\ \ Memory$ $(TCAM)$ is a popular device for hardware based lookup table solutions due to its high speed. However TCAM devices suffer from slow updates, high power consumption and low density. In this paper we present a novel Hardware-based Longest Prefix Matching ($HLPM$) technique for pipelined TCAMs to increase TCAM efficiency. Our HLPM provides very simple and fast table updates, with no TCAM management requirements, as well as potentially decreasing the power consumption and area requirements for a TCAM. Up to 30% power savings for matching entries, compared to previously designed TCAMs, is reported.**

I. Introduction

As a basic task of a Network Processor, Internet routing must employ a very fast routing mechanism to maintain required high throughputs. Internet routers forward incoming packets to their next hop by consulting their routing lookup tables (LUT). A routing lookup table stores the routing information for Internet Protocol (IP) prefixes rather than for exact IP addresses. In CIDR (Classless Inter Domain Routing) Prefixes can have any length and multiple prefixes might match with an IP address. A routing LUT requires performing a *Longest Prefix Matching (LPM)* and forwarding the corresponding information of the longest matching prefix of an IP address to the router.

Content Addressable Memory (CAM) is one of the hardware solutions to perform fast IP forwarding lookups. A CAM is a fully associative memory storing 0s and 1s and capable of searching a specific pattern of data in all its entries in parallel. A Ternary CAM (TCAM) is capable of storing *don't care* states in addition to 0s and 1s. Thus a TCAM is suitable for storing IP prefixes with different lengths, by filling the least significant bits with don't care values. A TCAM stores prefixes according to their lengths and a priority encoder resolves the longest matching prefix. TCAM-based tables have advantages over software-based methods in terms of the number of lookups needed and the control logic simplicity, but the need to maintain a sorted list makes updates slow. On the other hand, TCAM devices are expensive and power greedy due to charging and discharging long lines of memory with large capacitance.

Currently, TCAMs are being used for several different applications such as IP forwarding [1], packet classification [2], ATM switches [3], sorting and searching [4] and image processing [5]. All these increasing applications serve to increase the complexity of table lookup operations, driving further demand for low power-high throughput TCAMs.

We propose a pipelined TCAM with a novel Hardware-based Longest Prefix Matching (HLPM) solution to provide simple and fast updates, decrease the TCAM area requirements and save power through smart search operations. This paper is organized as follows. Section II gives a background on LPM solutions for CAM-based lookup tables. Section III describes the general features and functions of the proposed HLPM scheme. The performance of our TCAM is evaluated in Section IV. Finally, Section V concludes the paper.

II. LPM BackGround

Usually, finding the longest prefix match (LPM) during TCAM lookups requires maintaining the prefixes in a sorted length order which makes worst case updates very slow. For example, an insertion of a new entry in a TCAM storing N prefixes might result in moving (shifting) O(N) TCAM entries to create an empty space for the new insertion. This slow update is undesirable due to possibility of 100s to 1000s of updates per second in today's forwarding tables [6].

The routing table update delay is one of the key elements of routing lookup efficiency beside the lookup speed, power consumption and memory footprint. Several solutions have been proposed to decrease the routing table update delay such as reserving some empty entries between sets of different length prefixes. This leads to under-utilization of the TCAM space while the worst case complexity of updates remains the same. Since there is no need to sort the prefixes in a segment, the TCAM can reserve some empty space in the middle. Then an empty space can be provided anywhere with no more than $L/2$ shifts (L represents possible prefix lengths. *e.g.* $L = 32$ for IPv4 prefixes) [7]. However, the TCAM space management overhead and non-uniform update delays reduce the TCAM efficiency. These problems worsen with 128-bit IPv6 due to the much longer possible prefix lengths.

Some applications avoid TCAM sorting requirements by manipulating the data before storing it. For example, in IP prefix caches implemented by TCAMs, the LUT is expanded to avoid multiple matches for correct cache results [8]. But not all applications have such a convenient solution to the problem. Many TCAM vendors employ a simple sorting technique and live with an $O(N)$ worst-case update time solution.

Storing prefix lengths is a simple and fast solution for LPM but it results in at least a 70% increase in the TCAM memory requirement [4]. On the other hand, binary CAMs, with no built-in mask circuits, can use this extra information to mask data as well as finding the LPM [9]. Faster updates are obtained at the cost of slower search times and lower memory density. Figure 1 depicts an example of searching *10100* in the table using the design in [9] for a 5-bit addressing scheme. A prefix is represented by a binary prefix entry and a binary mask entry. Mask entries are sequences of ones followed by zeros, such that the number of consecutive ones represent the length of the prefix and zeros stand for *don't cares*. Horizontal AND circuits mask each prefix entry and vertical *OR* circuits in the mask column find the longest length among all matching entries. A second search of the longest length in the mask column resolves the position of the LPM.

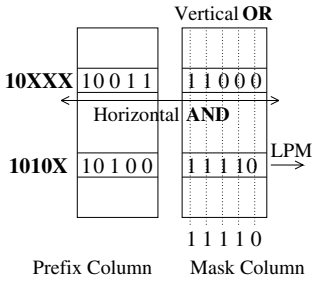

Fig. 1. Binary CAM with mask features presented in [9].

All existing LPM solutions: (1) have long worst-case update delays, (2) slow down the lookup speed, (3) need complicated table management and maintenance or (4) require a great amount of extra area. The Hardware-based Longest Prefix Matching (HLPM) technique proposed in this paper provides a simple, fast and scalable LPM solution with very small increase in area as well as potentially reducing the power consumption.

III. Architecture

This paper describes our proposed HLPM for a four stage pipelined TCAM applicable to IPv6, but the technique is scalable to any pipelined TCAM. Figure 2 shows our four-stage pipelined TCAM with an extra SRAM stage named: *Length Column*. Every entry in each stage is 32-bits wide to provide the 128 bits required by IPv6 prefixes. HLPM stores the binary coded lengths of prefixes in their ending stages in the *Length Column*. Thus 5 extra storage bits per entry are required for IPv6 prefixes (in a four-stage pipeline, 32 bits each). For example 00011 is stored in the corresponding *Length Column* entry of a prefix with size 35, because the prefix ends in the second stage, and there are only 3 bits in that stage. Same value will be stored for 3, 67 and 99 bit prefixes. However with no sorting requirement for the prefixes, new prefixes can be inserted in any TCAM entry, regardless of the prefix length.

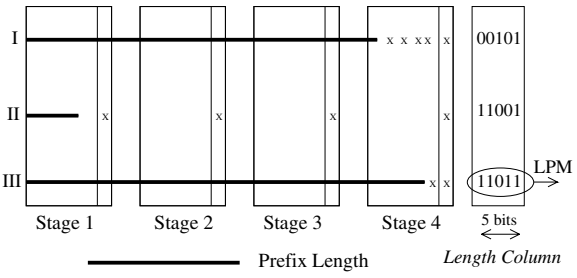

Fig. 2. The proposed Four-Stage Pipelined TCAM.

Pipelined TCAM devices save power by lowering the power consumption for non-matching entries. A 29% decrease in power consumption (compared to conventional TCAMs) is reported for a five stage TCAM, providing 144 ternary bits for IPv6, due to pipelining [10]. When a pattern is searched at each stage of the pipeline, only those entries that matched the key in the previous segments are searched, saving the power that would be required to search remaining bits of non-matching entries. The fact that very few entries in the TCAM actually match the data leads to dramatic decrease in power consumption. However, TCAM searches consume fixed power for different length matching prefixes. In CIDR prefixes can vary in size from 0 to 127 (IPv6) and many prefixes might match with an address. On the other hand, since short prefixes cover more addresses, there is higher probability of searching for a short prefix in a LUT than a long one. Our HLPM technique saves power not only on non-matching entries through pipelining, but also on short prefixes which are more likely to be searched.

A. TCAM Search Operation

IP prefixes are formed as sequences of data bits (either 0s or 1s) followed by *don't care* bits (the number of 1s and 0s represents the length of each prefix). Since don't cares match with both zeros and ones, a TCAM search may result in multiple matching entries with an address. Figure 2 shows an example of multiple matching entries. Entries I,II and III are three matching prefixes with different lengths for a given address. Searching the IP address in the first stage results in matches in all those entries. These matches lead to further searching of the IP address in the following stages of the pipeline in those three entries. There is no need to search the rest of entry II, simply because the rest of the bits of that entry are *don't cares* and will always match. Thus, if the last bit of an entry in one stage of the pipeline stores a *don't care* value, there is no need to search the rest of the entry in the following stages. In our example, second stage searches are necessary only for entries I and III. On the other hand, after the fourth stage of the pipeline, it is clear that entry II

is not the longest matching prefix due to the fact that prefix II ended in the first stage of the pipeline. These observations lead us to simplify the LPM in our TCAM. Since several matching prefixes might end in the same stage, a second level search is necessary to find the LPM from those entries (e.g. entry I and III in Figure 2), by using the information stored in the Length Column. Avoiding unnecessary searches for short matching prefixes reduces power consumption in comparison with previously reported pipelined TCAMs [10]. Meanwhile, since not all matching prefixes require the second level search (e.g. entry II), we achieve further power saving compared to previous designs such as [9]. However, in order to find out if a prefix ends in one stage of the pipeline or not, the last cell of the TCAM in each stage should be modified. This modification is described in Section III-B. Section III-C describes the Second Level Search which resolves the LPM of matching entries ending in the same stage.

B. TCAM Entry Modification

Figure 3 shows a TCAM entry in one stage of the pipeline. The last cell is modified by adding two extra transistors (M1 and M2) which are controlled by the complementary bits of the data stored in the cell. A normal search operation of the entry includes searching for a $don't\ care$ in the last cell in parallel with the conventional searching for a match or a miss-match between the data stored in the entry and the input pattern. The extra circuits for the last cell are similar to the normal search circuits. A very short match line, shown by MLx in Figure 3, is precharged to logic high in the precharge state. In the evaluation phase, the last cell searches for a *don't care* which corresponds to storing 00 in the last TCAM cell. If the cell stores a don't care, the paths from ML to ground are closed but the path from MLx to ground is open. MLx discharges, and the output senses a logic high. Although search operations of ML and MLx are similar, MLx evaluation is much faster, consumes less power, and does not require complicated sensing circuits due to very short length of MLx. The general evaluation of the entry is described in Table I. However, smart sensing and precharging circuits such as [11], precharge the ML/MLx only for matching entries, resulting in further power savings.

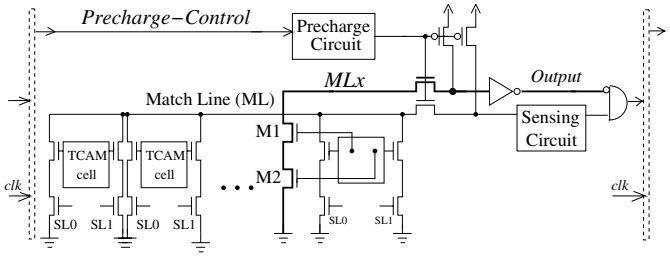

Fig. 3. A Modified TCAM Entry.

C. Second Level Search

The second level search resolves the LPM for matching prefixes ending in one stage. Figure 4 depicts the length column in detail. The $Second\ Search\ Signal(0)$ is the result of the last stage of the pipelined TCAM. SSS(0) is set only for matching prefixes requiring a second level search. In the example given in Figure 2, the second search signal is set only for entries I and III. Since the length column stores the binary lengths of prefixes in their ending stages (5 bits long), the entry storing the max value is the longest matching prefix. We adopted a 5-stage pipelined $Bit-Serial$ approach to find the max length. At each stage if a second search is required ($SSS(i) = 1$), one bit of the data in the corresponding entry of the Length Column is evaluated. If there is only one data equal to '1' among all entries, that entry is the max of all. But if no entry has a '1' or more than one entries store 1s, those entries should be searched in their next stages as well. Thus the SSS signal for the next stage of those entries will be set ($SSS(i+1) = 1$). However, after the fourth stage, if the max is not yet resolved (two entries similar in 4 MSBs), the one whose last bit is a '1' is definitely the max. Thus the pipeline is actually a four stage pipeline.

TABLE I
ENTRY EVALUATION.

ML	MLx	Search the Next Stage?
$High$	$High$	Yes
Low	$High$	No
$High$	Low	No
Low	Low	No

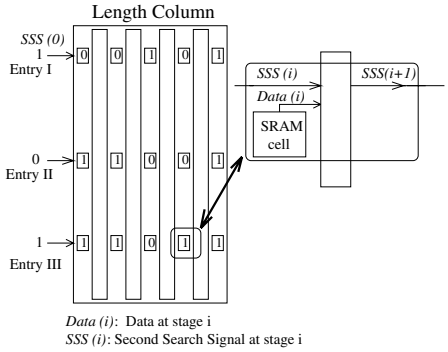

Fig. 4. Length Column.

However, the simple length-column pipeline can be clocked faster (*e.g.*it can be sensitive to rising and falling edges of the clock) than the TCAM pipeline or a digit serial approach can be used to provide short latency.

IV. SIMULATIONS AND PERFORMANCE EVALUATION

To evaluate the power savings obtained by HLPM, we simulated the HLPM using real lookup tables and IP traces of three distributing routers (not edge nor core). Because IPv6 is not yet broadly used, traces from real IPv6 traffic are not available. Instead we simulated the HLPM performance for a two-stage pipelined TCAM storing 32 bit IPv4 prefixes. Since IPv4 prefixes are mostly 16 to 24 bits wide, we stored prefixes with less than 20 bits in the first stage.

TABLE II

SIMULATION RESULTS FOR REAL TRACES.

	ISP1	ISP2	ISP3
LUT Short Prefixes %	28	27	28
Referenced Short Prefixes %	80	72	75
Second Level Search %	25	40	28
Power Savings %	30	27	28

TABLE III

SIMULATION RESULTS FOR MERGED LUTS.

	ISP1	ISP2	ISP3
LUT Short Prefixes %	31	28	30
Referenced Short Prefixes %	79	92	75
Prefix Compaction %	92	91	93
Second Level Search %	24	55	19
Power Savings %	30	34	28

Table II shows the results for the three traces (ISP1, ISP2 and ISP3). We found that less than 30% of measured prefixes stored in the LUT are short (have less than 20 bits). But more than 70% of the incoming IP addresses match these short prefixes. Second level searches are only required for a portion of the matching prefixes. This is achieved through the new search mechanism described in Section III. Since most of the IP lookups match with a short prefix, the pipelined design saves power by avoiding unnecessary searches in the second stage. The power savings estimates reported in Table II are in comparison with a standard full length TCAM processing the same traces.

Since low power TCAM designs use most of the power for matching entries [11], this power reduction directly affects the total TCAM power consumption. The HLPM architecture should be even more power-effective for IPv6 prefixes that have wider prefix length variation.

To evaluate the HLPM performance for a *worst case* situation, we compacted the real LUTs by eliminating all redundant prefixes. This computation reduced the tables to almost 10% of their original ones. Table III shows the results for the compacted tables. As expected, more second level searches are required, but the HLPM architecture is still very effective to conserve power.

Beside being power efficient, the HLPM architecture has the following advantages:

1) HLPM resolves the LPM without requiring LUT management or maintenance or sorting of the entries in the TCAM. Table updates require a single update operation.
2) HLPM is simple and scalable with TCAM width. The second level search does not change if the TCAM has more or less 32-bit stages. The first level search, as described in III-A, is also independent of the TCAM size. Thus the extra area and the complexity of HLPM remains the same if the TCAM size is scaled.
3) HLPM is efficient in extra storage area requirements. For example, a conventional TCAM needs 100x128 bits of extra storage to reserve only one empty entry for only 100 different IPv6 prefixes. This area is equal to the storage area of the 5th stage (5 bit entries) for a 5K entry TCAM with the proposed HLPM. Since the Length Column requires only 5 SRAM bit per entry SRAM rather than 32 CAM bit per entry, the total area of our TCAM is approximately 20% less than similar designs [9].

V. CONCLUSION

A novel Hardware-based Longest Prefix Matching (HLPM) for TCAM-based lookup tables is proposed in this paper. The technique is applied to pipelined TCAMs and aims at further decrease in power consumption compared to previously reported pipelined TCAM designs, by saving power for matching short prefixes.

The HLPM is a two level search. The first level resolves the LPM of prefixes ending in different stages by searching for a *don't care* in the last bit of each stage. A very simple cell modification is presented in this paper to perform the first level search. The second level resolves the LPM of multiple prefixes ending in one common stage of the pipeline by finding the max value of the coded lengths of prefixes in the last stage of the pipeline.

Our HLPM provides very fast table updates (no worst-case delays) with no table maintenance/management requirements. It also saved area compared to other fast table update solutions. Up to 30% power reduction for IPv4 matching prefixes is reported.

REFERENCES

[1] M. A. Ruiz-Sanchez, E. Biersack, and W. Dabbous, "Survey and taxonomy of ip address lookup algorithms," *IEEE Network*, vol. 15, pp. 8–23, Mar./Apr. 2001.

[2] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE/ACM Trans. Networking*, vol. 15, pp. 24–32, Mar./Apr. 2001.

[3] K. J. Schultz and P. G. Gulak, "CAM-based single-chip shared buffer ATM switch," in *IEEE International Conference on Communications (ICC'94)*, vol. 2, New Orleans, LA, May 1994, pp. 1190–1195.

[4] S. Sharma and R. Panigrahy, "Sorting and searching using ternary CAMs," in *10th Symposium on High Performance Interconnects*, Stanford, CA, Aug. 2002, pp. 101–106.

[5] T. Ogura, M. Nakanishi, T. Baba, Y. Nakabayashi, and R. Kasai, "A 336-kbit content addressable memory for highly parallel image processing," in *IEEE 1996 Custom Integrated Circuit Conference*, San Diego, CA, May 1996, pp. 273–276.

[6] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," *IEEE/ACM Trans. Networking*, vol. 6, pp. 515–528, Oct. 1999.

[7] D. Shah and P. Gupta, "Fast updating algorithms for tcams," *IEEE Micro*, vol. 21, pp. 36–47, Jan./Feb. 2001.

[8] H. Liu, "Routing prefix caching in network processor design," in *Tenth International Conference on Computer Communications and Networks*, Scottsdale, AZ, Oct. 2001, pp. 18–23.

[9] M. Kobayashi, T. Murase, and A. Kuriyama, "A longest prefix match search engine for multi-gigabit ip processing," in *International Conference on Communications (ICC 2000)*, vol. 3, New Orleans, LA, June 2000, pp. 1360–1364.

[10] K. Pagiamtzis and A. Sheikholeslami, "Pipelined match-lines and hierarchical search-lines for low-power content addressable memories," in *IEEE Custom Integrated Circuit Conference*, San Jose, CA, Sept. 2003, pp. 383–386.

[11] A. Sheikholeslami and I. Arsovski, "A mismatch-dependent power allocation technique for match line sensing in content-addressable memories," *IEEE J. Solid-State Circuits*, vol. 38, pp. 1958–1966, Nov. 2003.