An efficient parallel prefix matching architecture using Bloom filter for multi-bit trie IP Lookup algorithm in FPGA

K. SARAVANAN[a,*], A. SENTHILKUMAR
[a]*Department of electronics and communication, The Christian Institute for Technical education, Tamilnadu, India*
Professor, Department of Electrical and Electronics Engineering, Dr. Mahalingam College of Engineering and Technology, Pollachi, India

The major design challenge in Internet routers is the IP lookup, which is essential for packet forwarding. The Longest Prefix Matching (LPM) is used to obtain the best match for the incoming packets, which increases the time consumption of IP lookup. Current high speed link rate requires faster IP lookup. Due to the continuous increase in internet users, the routing table has to incorporate more routing entries for effective packet forwarding which results in memory usage increase. Various hardware based IP lookup mechanism using either TCAM or SRAM are available in use. TCAM–based solutions supports faster IP lookup with the high hardware cost and high power consumption compared to SRAM-based devices. The SRAM-based solutions require more memory access for IP lookup. This paper proposes parallel prefix matching using bloom filter for multi-bit trie based IP lookup, implemented in FPGA. The proposed method achieves faster IP lookup with single memory access. Comparison of implementation results shows that the proposed algorithm achieves 13.04 % increase in throughput and 27 times lesser memory usage.

(Received March 10, 2015; accepted May 7, 2015)

Keywords: Longest Prefix Matching (LPM), Multi-bit trie, SRAM based IP lookup, Packet Forwarding

1. Introduction

Continuous increase in the number of users, network and various domains connected to the Internet; has exponential growth of the Internet as an outcome. The network traffic flow is increased due to this large number of users with wide range of multimedia applications. In order to maintain the quality of the Internet without degradation, the router has to consider three main factors like: link speeds, router data throughput and packet forwarding rates [1].The high speed optical transmission rates of fiber optics provide faster transmission link rates and the current switching technology achieves high data throughput. Recently, the router has achieved link rates of 100Gbps with data throughput of 312.5 Mpps (million packets per second) for packet size of 40 bytes [2]. In order to cope up with this link rate and data throughput, the router has to adapt faster packet forwarding which requires efficient packet processing algorithm.

The IP lookup process is the main step in packet forwarding; in which the router performs lookup for the incoming packet's destination address in the routing database to find the appropriate outgoing link for the packet. The IPv4/ IPv6adapts Classless Inter-Domain Routing (CIDR) [3], which uses Longest Prefix Matching (LPM) to obtain the best matching prefix and it identifies corresponding next hop port for the matching prefix to forward the packet through that link.IPv4, IPv6consistof 32 and 128 bits packet address length respectively, the packet forwarding based on Longest Prefix Matching (LPM) use this address length for millions of packets. The vast number of internet user's result in routing table expansion which increases the search space for destination address in the routing table. The routing table expansion in terms of address length and number of prefixes requires proper IP lookup mechanism to achieve higher packet forwarding rate.

For IP lookup the hardware-based solutions are divided in to two main categories, they are Ternary Content Addressable Memory (TCAM) and Static Random Access Memory (SRAM). Due to the parallel hardware architecture in TCAM, it provides higher lookup rate than SRAM. The major disadvantage of TCAM based hardware solutions are high cost of devices and higher power consumption [4] [5].Further incremental updates in the routing table for TCAM-based solutions require number of memory operations as same as the prefix length. The SRAM based solutions has very less power consumption compared to TCAM devices [6]. Due to high memory access in SRAM based IP lookup, the power consumption is increased. Recently, SRAM based algorithmic IP lookup solutions are introduced to reduce the memory access in SRAM-based IP lookup solutions. In this paper SRAM based IP Lookup using Bloom filter is proposed for efficient IP lookup.

2. Related works

Recently several algorithm based approaches are proposed for IP lookup solutions. Binary tries is the basic algorithmic solutions for the IP lookup. Binary tries provides simple and trouble-free implementations for IP lookup with good support for incremental updates and scalability [6]. A Binary trie adopts tree-based data structure; the branching is based on the prefix bits, where each branch leads to two child nodes [7]. In uni-bit trie, only one bit is used for branch decisions, the maximum number of lookup required is same as the prefix length. In order to overcome it, multiple bits are used to direct the branching decisions. The number of bits used in each branch decisions is called as stride length. The stride length is varied to build an effective multi-bit trie. There are various multi-bit algorithm available [8-11]; such as tree bitmap [12] [13], shape-shifting trie [14], priority tries [15-16], binary decision diagram [17] and trie partitioning algorithm [18]. Various bloom filter based implementations for these algorithmic IP lookup mechanisms are proposed [19-22]. In [19], parallel bloom filter based architecture is proposed for the IP lookup mechanism. In [23], fixed stride tries based longest prefix matching is used for faster IP lookup. In [25-26], SRAM based pipeline architecture is used for reducing the memory access.

3. Bloom filter

The bloom filter is bit vector of size m to represent a set of n data elements X = {x_1,x_2, -- x_n}. The bloom filter has k hash functions {h_1,h_2, --- h_k} to represent the bloom filter such that the hashed value are used as index for bit vector. For entry of an element, the bit position pointed by the k hashed value is set to one. To process the membership queries, the bit locations pointed by all the hashed values are checked. If all the bit locations value is set to one, then it will have possibilities for False Positive Error. If anyone bit locations are not set, then it is definitely a False Negative. The False Positive probability is calculated by

$$f = \left(1 - e^{\wedge} nk/m\right)^{\wedge} k \tag{1}$$

The optimal value of k is given as

$$k = (m/n) ln2 \tag{2}$$

Let x, y are two elements represented in bloom filter, with three hash functions. Fig. 1 shows the bloom filter representation for the two elements. The hashed value for the two elements has collisions at the sixth bit location.

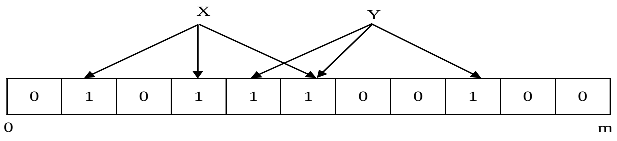

Fig. 1. An Example for bloom filter.

The false positives of a given n elements can be reduced by the appropriate selection of values for m and k, the value m is quite large compared to n. Further for given m/n ratio value, the False Probability is reduced with increase in value of k. Since the different elements bits are overlapped in the bloom filter. For a dynamic environment, it is not possible to delete an entry in the bitmap. In order to overcome it, the counting bloom filter came to existence. The counting bloom filter has a small b bit counter for each location in the bloom filter; to track the number of entries corresponded to that particular bit location. These counters will affect the space efficiency achieved by the bloom filter.

4. Proposed method

In CIDR, the routing information called routing update is exchanged between routers; which was entered in to the routing table. The routing information exchange takes place often, in order to route the packets effectively. In this paper, scalable bloom filter architecture is proposed for multi-bit trie based IP lookup algorithm. The proposed bloom filter based architecture supports faster IP lookup and dynamic insertion and deletion of prefixes. It uses fixed-stride multi-bit trie without leaf pushing. The nodes at each level have same stride length. Thus, a single bloom filter is employed for each node and the hashing is performed in parallel for all the bloom filters. A small counter is employed for each bit location of bloom filter at all levels. The counter supports dynamic routing update in the routing table through insertion and deletion of prefixes. The proposed algorithm employs an efficient update manager to select the LPM for the input IP address from all the bloom filter output and to include the routing update.

Fig. 2 shows the FPGA implementation of proposed IP lookup algorithm. Since the number of prefixes at each level is different, the proposed algorithm varies the bloom filter size correspondingly. In normal bloom filter based IP lookup algorithm, the bloom filter is included for all possible length of prefix. Here, the prefix at higher length includes prefixes from previous length. Thus, each bloom filter has to represent 0 – 2^n entries, where n denotes the prefix length of input IP address.

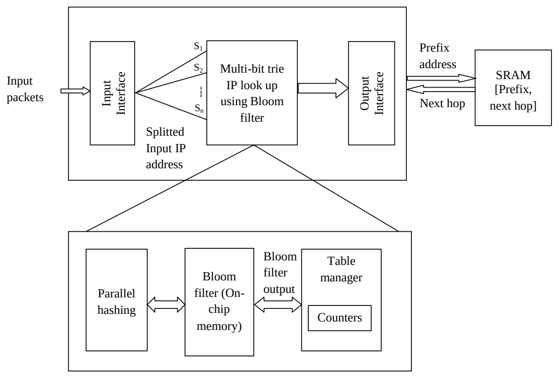

Fig. 2. FPGA implementation of proposed IP lookup algorithm.

In case of multi-bit trie, the prefix range at each stride level does not include the bits from the previous range. The bloom filter represents only the prefix entries in the range of prefix $0 - 2^s$, where s denotes the stride length. Due to minimum entries in each bloom filter, the proposed algorithm uses small bloom filter size with reduced False Positive Probability FPR at each level. Table 1 shows an example for prefix set. Fig. 3 shows the detailed implementation of proposed algorithm for Table 1.

Table 1. Prefix set.

Prefix	Value
P1	0*
P2	1*
P3	10*
P4	110*
P5	1000*
P6	11001*
P7	100000*
P8	1000000*

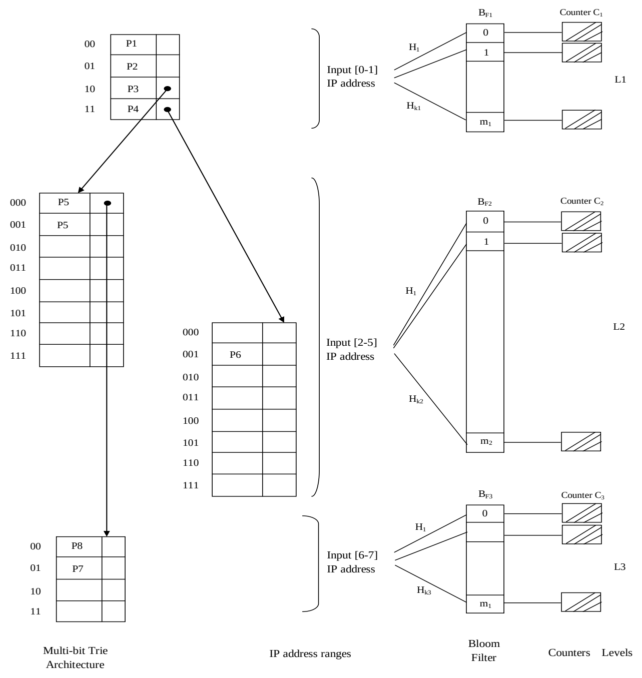

Fig. 3. Bloom filter representation for the prefix set.

5. Insertion and deletion of prefixes

The update manager performs the update operation from all the bloom filter output for the input IP address. The update operation includes deals with updating the counter. The update operation is performed in any particular level when there is no matching prefix at the next level. For prefix matching the child nodes in i^{th} level, if next level has no matching prefix then the update operation is performed at the same i^{th} level. Otherwise, the update operation is performed at the new level (i + j), where j is the new matching level with no matching prefix at the next level. In insertion, the update manager increments the counter for the highest matching prefix level. In deletion, the update manager decrements the counter for the highest matching prefix level. In case of incrementing, if the counter value changes from 0 to 1, the corresponding bit in the bloom filter are changed to one. In case of decrementing, if the counter value changes from 1 to 0, the corresponding bit in the bloom filter are changed to zero.

6. IP Lookup operation

During the IP lookup operation, the hashing is performed in parallel for all the prefixes of input IP address based on the stride length of the levels. The update manager selects the matching prefix at higher level using the novel selection mechanism. The update manager selects the highest matching prefix level from the all bloom filter output and traces the table entry from this node to the root node. If a link exists between all the previous levels then it consider a next hop address exist for the input IP address. The next hop address is retrieved from the SRAM. In order to fetch the next-hop address from SRAM, new address value is generated from input IP address. The new address value includes input IP address value up to the highest matching prefix and appending zero for remaining address value.

7. Performance evaluation

The proposed algorithm is implemented in Altera Cyclone IV FPGA. The bloom filter is implemented in on-chip memory to support faster lookup and the corresponding next-hop address is accessed from SRAM memory. For implementation, the proposed algorithm use operating frequency of 150MHz. The external SRAM works in operating frequency of 300MHz. To evaluate the proposed algorithm, the routing entries of rrc11 are used [29]. From [24], the average number of prefixes for prefix length in the range of 21 to 24 is approximately 70%. In the multi-stride trie, for higher prefix length ‘l’ the change in bits are evenly distributed among the lower stride level corresponds to (l-1) bits. Thus, even for higher distribution of prefixes in particular prefix length, the number of entries in the bloom filter increases linearly at lower levels. In the proposed algorithm, stride length of 2, 3are used for the levels (1 to 24) and stride length of 4is used for the levels (25 to 32).

The SRAM based IP lookup solutions has faster memory access with less power consumption. In order to obtain the LPM for input IP address, it requires more number of memory accesses which increases the power consumption enormously. The proposed bloom filter based IP lookup supports parallel prefix matching for all the entries of routing table. Due to this, a single memory access is enough to fetch the proper next-hop address with reduced on-chip memory usage. As the bloom filter based implementations has False Positive Probability (FPR), the reduced memory access overcomes it. In [27], the SRAM-based linear pipeline architecture for IP lookup is used for low power consumption. This architecture adapts Binary Tree Search (BST) based IP lookup mechanism implemented in pipeline architecture. In this method, the ranges of prefixes having less number of prefixes are converted to consecutive higher length. This architecture achieves higher throughput at the cost of large on-chip memory usage.

In [28],a hash table based reconfigurable architecture is proposed for the IP lookup mechanism. This architecture implements the hash table as multi-column main table with sub table; the prefixes corresponding to collisions are implemented in the on-chip memory with remaining prefixes in the SRAM. It has high on-chip memory usage, even for less number of prefixes due to Hash table implementation. Table 1 shows the memory usage of proposed algorithm for different number of prefixes.

Table 2. Memory usage for various prefixes.

No of Prefixes	Memory Usage
100M	162
150M	203
200M	247
250M	293
300M	311
350M	354
400M	397
450M	456
500M	513

Table 2 shows the comparison of implementation results of proposed algorithm with other existing SRAM-based IP lookup implementation. Compared to both IP lookup implementation, the proposed algorithm achieves the high throughput for more number of prefixes with less on-chip memory usage. The hardware utilization of the proposed algorithm is more due to the complexity in IP lookup. Compared to [27], the proposed algorithm achieves 27 times less memory usage and 13.04% increase in throughput. Compared to [28], the proposed algorithm achieves 14 times less memory usage and 32.74% increase in throughput.

Table 3. Implementation results of proposed algorithm.

	On chip memory	SRAM	LUT	Throughout (MLPS)	Prefixs
Proposed	314 Kb	4.1MB	37715	391	293K
[27]	473Blocks (BRAM)	-	16617	340	249K
[28]	254 Blocks (BRAM)	-	14274	263	80K

The proposed algorithm uses less on-chip memory with high throughput even for 290k prefixes. Thus the proposed algorithm suits well for routing table expansion of 500 million prefixes. Since the proposed algorithm achieves higher throughput in packet forwarding of 391 MLPS, it supports higher link rate of optical carriers such as OC-768, OC-1536.

8. Conclusion

In this work, a bloom filter based parallel prefix matching is proposed for faster and efficient IP lookup. Due to the single bit representation for prefix representation in bloom filter, the proposed algorithm achieves less memory usage. In the proposed algorithm, the bloom filter occupies the on-chip memory whereas the actual routing prefixes with next-hop addresses are stored in the SRAM. Since the parallel prefix matching was done using bloom filter, a single memory access is required to access the external SRAM memory. This overcomes the drawback in SRAM-based IP lookup of higher memory access. Compared to other existing SRAM-based IP lookup solutions, the proposed bloom filter based multi-bit trie architecture achieves high packet forwarding rate with reduced memory usage.

References

[1] M. Waldvogel, G. Varghese, J. Turner, B. Plattner, Scalable high speed IP routing lookups, in Proc.SIGCOMM’97, Cannes, France.
[2] Verizon offers U.S. 100-Gbps deployment details.

http://www.lightwaveonline.com/articles/2011/09/verizon offers-us-100-gbps-deployment-details-129650943.html, 2011.
[3] Y. Rekhter, T. Li. An Architecture for IP Address Allocation with CIDR, RFC 1518, 1993.
[4] IDT Generic Part: 5K72100http://www.idt.com/?catID=58523&genID=75K7 2100.
[5] David E. Taylor, Washington University Technical Report, WUCSE-2004, 2004.
[6] M. A. Ruiz-Sanchex, E. W. Biersack, W. Dabbous, IEEE Network, **15**(2), 8 (2001).
[7] D. Mehta, S. Sahni, Handbook of Data Structures and Applications, Chapman and HALL/CRC, 2005.
[8] T. Kijkanjanarat, H. J. Chao, Proc. IEEE GLOBECOM, 2, 1999, pp-1570–1575.
[9] H. H.-Y. Tzeng, T. Przygienda, IEEE J. Sel. Areas Commun, **17**(6), 1067 (1999).
[10] W. Eatherton, G. Varghese, Z. Dittia, ACM SIGCOMM Computer Communications Review, **34**(2), 97 (2004).
[11] D. E. Taylor, J. S. Turner, J. W. Lockwood, T. S. Sproull, IEEE J. Sel. Areas Commun., **21**(4), 522 (2003).
[12] H. Song, J. Turner, J. Lockwood, IEEE ICNP, 358 (2005).
[13] K. Kim, S. Sahni, IEEE Trans. Comput, **56**(1), 32 (2007).
[14] R. Sangireddy, A. K. Somani, IEEE J.Sel. Areas Commun, **21**(4), 513 (2003).
[15] H. Lim, J. Mun, Proc. IEEE Globecom, 2006, pp-1–5.
[16] H. Lim, C. Yim, E. E. Swartzlander, IEEE Trans. Comput, **59**(6), 784 (2010).
[17] W. Lu, S. Sahni, IEEE Trans. Comput, **59**(12), 1683 (2010).
[18] S. Dharmapurikar, P. Krishnamurthy, D. Taylor, In ACM SIGCOMM, 2003.
[19] T. S. Sarang Dharmapurikar, Praveen Krishnamurthy, J. Lockwood, in MICRO 37.
[20] H. Song, S. Dharmapurikar, J. Turner, J. Lockwood, SIGCOMM ’05.
[21] F. Bonomi, M. Mitzenmacher, R. Panigrah, S. Singh, G. Varghese, SIGCOMM ’06.
[22] V. Srinivasan, G. Varghese, ACM Trans. Comput. Syst., **17**, 1 (1999).
[23] chttp://bgp.potaroo.net. BGP Routing Table Analysis Reports, 2008.
[24] W. Jian, V. K. Prasanna, J. Parallel Distrib. Comput, **69**(9), 778 (2009).
[25] L. D. Carli, Y. Pan, A. Kumar, C. Estan, K. Sankaralingam, Proc. SIGCOMM, 2009.
[26] Scalable High Throughput and Power Efficient IP-Lookup on FPGA.
[27] H. Fadishei, M. S. Zamani, M. Sabaei, Proc. ANCS ’05, 81 (2005).
[28] RIS RAW DATA [Online]. [http://data.ris.ripe.net].

*Corresponding author: saravanantlf@gmail.com