ASIC Design of Fast IP Lookup for Next Generation IP Router

Yuan-Sun Chu, Po-Feng Lin, Jia-Huang Lin, Hui-Kai Su and Ming-Jen Chen

Computer Network Laboratory, Department of Electrical Engineering,

National Chung Cheng University, Taiwan, R.O.C

chu@ee.ccu.edu.tw

***Abstract*— One of the prime designs for the next generation IP routers is the IP-lookup mechanism. The IP lookup have become a major performance bottleneck for the routers. In this paper, we propose a complete hardware architecture which includes searching, updating, inserting, and deleting functions. A simple hash hardware design is used to reduce the lookup time, and a CAM is also used to solve the collision problems effectively. The ASIC includes search unit, memory controller, 1M-byte cache and 3.18 Kbytes CAM for a 32000 entries routing table. The searching, updating and deleting functions only need 1 cycle and it will be 96.88% chance to hit the correct next hop in the first cycle.**

I. Introduction

With the growth of Internet applications, the IP address has been exhausted. To solve this problem, the short term solution such as Classless Inter-Domain Routing (CIDR) is presented in 1993 [1,2]. With CIDR, each route is identified by a <prefix, prefix length> pair leading to IP address flexibly.

On the other hand, a long term solution is the IPv6 protocol which was proposed by IETF (Internet Engineering Task Force) [3,4]. The IPv6 protocol has 128-bit address which will replace 32-bit address in IPv4 that can avoid the exhausted IP address problems.

In recent years, a number of methods have been proposed for improving the performance of the longest matching prefix search [9, 10, 13-18]. Most of them can achieve high average search throughput for IPv4, but are slow in the updating speed, while some papers introduce the searching algorithm for IPv6 [5, 6, 7]. They can save the cost of memory, but slow in searching and lack of updating because of software implementation. In this paper, we propose a complete hardware architecture, which includes searching, updating, inserting, and deleting functions. Hash hardware is designed to hit the lookup table in a cycle, and a Content Addressable Memory (CAM) is also used to solve the collision problems which the collision numbers will be reduced by an improved function effectively.

In the paper, Section 2 describes the system architecture. Section 3 introduces the hash scheme and Section 4 is the function schemes. Section 5 is the performance analysis and the conclusion is given in Section 6.

II. System architecture

The IPv6 global unicast address format is shown in Figure 1. Network ID is a 64-bit address consisting of bits in positions 1 through 64. Interface ID is a 64-bit address consisting of bits in positions 65 through 128. The table 1

n bits	64-n bits	64 bits
global routing prefix	subnet ID	interface ID

Fig. 1 IPv6 Global Unicast Address Format

Table 1: Prefix length distribution in the router of 6Bone

	Prefix number	%
24^{th} bit	39	30.47%
28^{th} bit	39	30.47%
32^{th} bit	29	22.66%
35^{th} bit	1	0.78%
48^{th} bit	17	13.28%
64^{th} bit	3	2.34 %
Total	128	100 %

shows the IPv6 prefix length distribution in the router of 6Bone.

Obviously, we can observe two points from Table 1：

1) There are very few route prefixes longer than 64 bits, so the mechanism is designed to focus on Network ID.
2) The distributions of the prefix length equal to 24, 28, 32 and 48 are approximately 30.47%, 30.47%, 22.66% and 13.28%, so we deal with the four prefix lengths with four hash functions (caches TAB1, TAB3, TAB5 and TAB7) firstly, and that will hit the data in one cycle.

Figure 2 shows the system architecture which includes two major parts: routing lookup ASIC and DRAMs. In the ASIC, routing lookup engine sends the prefix data to each hash table. Each hash table which composed by a cache with a CAM for each address equals to the 24-bit, the 28-bit, the 32-bit and 48-bit. If it cannot get correct routing result (not the longest prefix route), DRAM control will access the off-chip DRAMs and get the correct routing result. The DRAMs contains 4 cache TABs which store the prefix of 25^{th} bit ~ 27^{th} bit, 29^{th} bit ~ 31^{th} bit, 33^{th} bit ~ 47^{th} bit and 49^{th} bit ~ 64^{th} bit separately.

III. Hash Scheme

Hash algorithms are usually used in searching a large database. The proposed hash-base forwarding table is simple and can be easily implemented by hardware. This section describes an approach of good hash functions and shows how to reduce the number of hash collisions.

A. Hash Function

Since different keys map to the same hash index by hash function, the major design consideration is to minimize hash collisions. The hash collision problems can

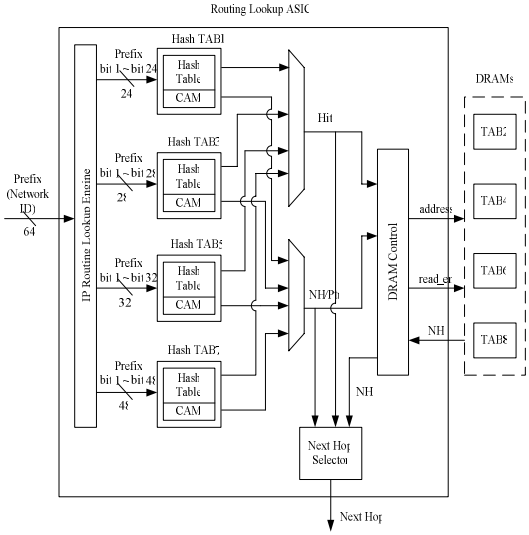

Fig. 2 System architecture

Table 2: The entropy of various hash functions

Hash Function / Hash Index	Bit extraction	Fletch Checksum	XOR Folding	CRC
12	11.9694	11.9714	11.9753	11.977

be solved by using good hash function and by grouping hashed keys into different subtables.

To compare various hash functions, a trace of destination addresses are observed on a backbone router. The trace consists of 7.66157 million entries in a period of one hour, but there are 43867 distinct destination addresses. We compute the entropy of various hash functions, which are shown in Table 2.

In Table 2, a cyclic redundancy checking (CRC) polynomial is the best hashing function. However, CRC polynomial requires extensive computation, and it is more complex than exclusive-OR folding function in hardware. The Fletcher checksum and the bit extraction cannot offer good performance if the patterns of extracted bits are randomly distributed. The exclusive-OR folding does not need computation and can offer good performance[8] while it is also simple to be implemented in hardware, so it is used in our scheme.

B. Hash Table Size

In this part, the hash table size will be decided. According to Table 3, if the hash index size is shorter than 13-bit, the hash collision will exceed 41.3% in hash TAB1 and hash TAB3. Considering the tradeoff between memory size and hash collision numbers, we choose the hash index size as 15 in TAB1 and TAB3. In the same case, the hash index size of hash TAB5 is equal to 14 according to Table 4, and the hash TAB7 is equal to 14 according to Table 5.

C. Performance improvement

In Fig. 2, we use the destination address's bits from 1 to 24, 1 to 28, 1 to 32 and 1 to 48 through hash function to map to the caches TAB1, TAB3, TAB5 and TAB7 separately.

TABLE 3: Maximum number of collision in hash TAB1 and hash TAB3

Hash table size	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
Maximum number of collision	4050	2402	1317	691	354
%	41.5%	24.6%	13.5%	7.1%	3.6%

TABLE 4: Maximum number of collision in hash TAB5

Hash table size	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
Maximum number of collision	2428	1385	742	385	196
%	33.6%	19.1%	10.3%	5.3%	2.7%

TABLE 5: Maximum number of collision in hash TAB7

Hash table size	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
Maximum number of collision	1605	934	506	264	135
%	37.8%	22%	11.9%	6.2%	3.2%

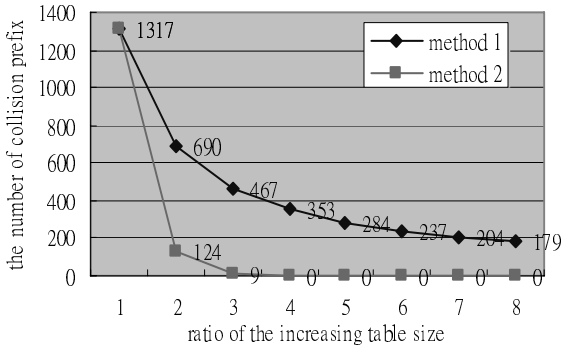

Fig. 3 Simulation result

By using CAM, a sufficient number of entries in hash table are given to accommodate hash collision keys while heeding the design tradeoff. Two methods are improved and they can reduce the hash collision number. The first method is to increase the size of the hash table, and the second one is to have a sufficient number of entries in a subtable to accommodate hash collision. The simulation result of these two methods is shown in Fig. 3.

In the first method (diamond line) of Fig. 3, if we increase the size of hash table, the collisions will decrease slowly. In the second method (square line) of Fig. 3, if the number of entries is increased in a subtable, the hash collisions decrease quickly. Thus, the second method will be chosen. According to the simulation result, when per entry saves two routes in a hash table, the number of hash collisions is reduced to 124. In other words, the collision occurrence is reduced to 1.27% in hash TAB1 and TAB3. In the same case, the collision occurrence chances in hash TAB5 and TAB 7 is reduced to 2.63% and 0.99% respectively.

Fig. 4 shows the architecture of a hash table with two routes and a CAM. Both the hash table and the CAM execute simultaneously, and only one cycle is needed for

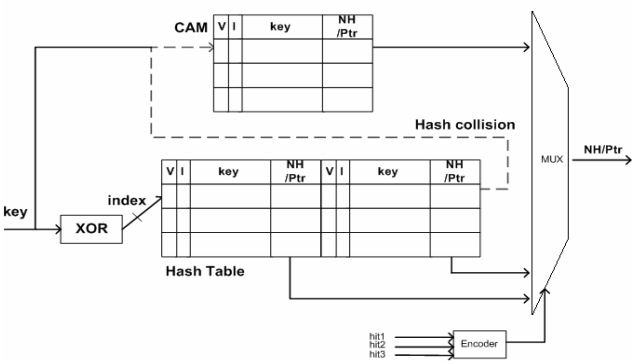

Fig. 4 Hash table with a CAM

v	I	Key	NH/Ptr
1	0	FFFFFE	A
1	1	FFFFFC	4

TAB 1

Address	Length	NH/Ptr
4*8	2(	B
4*8+1	2(	B

TAB 2

Fig. 5 Example of two routes in TAB1 and TAB2

searching.

IV. Function Schemes

Searching

In Fig. 2, the architecture is divided into two levels. The first level is hash TAB1 which stores all the routes equal to the 24 bit. The second level is TAB2, which stores all possible routes whose prefix length is between 24 bits and 28 bits. This table has 2^3 entries. Each 24-bit prefix that has at least one route longer than 24 bits and shorter than 28 bits is allocated 2^3 entries in TAB2. For example, assume that there are two routes (<FFFF:FE00/24/A>、<FFFF:FC00/26/B>) in the table already. Forwarding table distributions are shown in Fig. 5.

The first route requires one entry addressed from 001 in TAB1. The second route requires TAB2 to be used. In TAB1, the index (4) is entered into the corresponding to the FFFFFC prefix. In the TAB2, 8 entries starting with memory location 4*8 are allocated. The second route requires two entries addressed from (4*8~4*8+1) in TAB2.

In Fig. 2, Hash TAB1, TAB3, TAB5 and TAB7 execute simultaneously. In this system, the longest delay time for IP-lookup is 2 cycles (TAB1→ TAB2 or TAB3 → TAB4 or TAB5→ TAB6 or TAB7 → TAB8). Because the number of prefix length being equal to 24, 28, 32 and 48 is approximately 96.88% in the 6-Bone, so we deal with the prefix length equal to 24, 28, 32 and 48 with hash function firstly.

Inserting

When a new route whose prefix length is equal to the 24 bit is inserted, it must enter TAB1 firstly. Because each entry in TAB1 saves two routes, all the hash collision entry will be saved in and index to the CAM. The insertions in

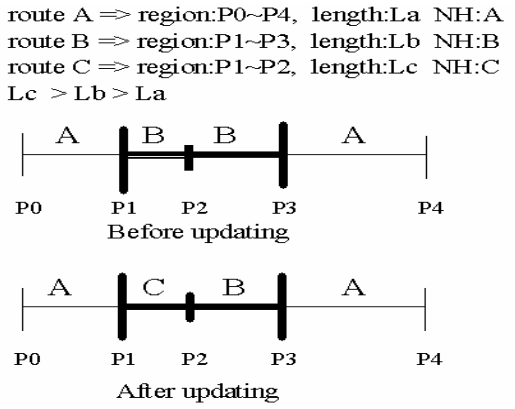

Fig. 6 Example for explaining the updating process

TAB3, TAB5 and TAB7 have the same operations.

Updating

The updating in the hash TAB1, TAB3, TAB5 and TAB7 is the same as the insertion. In TAB2, TAB4, TAB6 and TAB8, when a new route with the prefix length L and the updating region R is inserted, the following steps are to conform to the longest matching prefix.

1) For each entry in updating region R whose length filed is less than or equal to L, the new next hop is entered.
2) Otherwise, those entries with the length filed greater than L will be unchanged.

Fig. 6 explains this case. There are two routes (route A and route B) in TAB2, and a new route (route C) is updated. We must follow the above two rules for updating.

Firstly, because the prefix length of the new route (route C) is longer than the prefix length of the existing route (route B) in TAB2, we must update prefix length field and next hop field between P1~P2. Then, replace the next hop filed and the length field of an existing route with the next hop C and the prefix length of a new route.

Deleting

The process of deletion is almost the same as the updating.

V. Performance analysis

A. Memory Requirements

Note that the current backbone routers have a routing table with about 32000 entries[11]. Our system architecture consists of two kinds of memory tables:

1) Hash table:
   (a) TAB1:
   - Cache:(2^{15}×42(Vaild:1 bit, Index:1 bit, Key:24 Bites, Length:6 bits, NH:10bits)×2)/8= 336Kbytes.
   - CAM= (136×42)/8= 714 Bytes.

   (b) TAB3:
   - Cache: (2^{15}×46(Vaild:1 bit, Index:1 bit, Key:28 Bites, Length:6 bits, NH:10bits)×2)/8= 368Kbytes.
   - CAM= (136×43)/8= 731 Bytes.

Table6: Comparison

	My Scheme	Scalable High Speed [5]	Prefix Expansion [6]	Multiway and Multicolum Search[7]	NP for gigabit router[18]
Operation	Hardware	Software	Software	Software	Software
Function	Complete	Complete	Complete	Search	Search
Lookup speed (worst case)	1 h	6 m	2 h+4 m	W/M+log $_2$k	IXP2400 (600MHZ)
Update scheme	Easy to implement	Rebulit	Rebulit	Null	Null
Memory size(32000)	Caches: 1 MB CAM: 3.18 KB	DRAM: 2.4MB	DRAM: 4.37MB	DRAM: 2 MB	Null

h: hashing access cycle m: memory access cycle

(c) TAB5:

- Cache: (2^{14}×50(Vaild:1 bit, Index:1 bit, Key:32 Bites, Length:6 bits, NH:10bits)×2)/8= 200Kbytes.
- CAM= (192×50)/8= 1200 Bytes.

(d) TAB7:

- Cache: (2^{14}×102(Vaild:1 bit, Index:1 bit, Key:48 Bites, Length:6 bits, NH:10bits)×2)/8= 408Kbytes.
- CAM= (48×102)/8= 612 Bytes.

VI. Conclusions

In this paper, a fast route mechanism for IPv6 with a complete function is proposed. The ASIC is used to speed up the lookup time which is implemented by the exclusive-OR folding function, and the collision problems are solved by a CAM. It is easily implemented in hardware. Table 6 shows the comparison of the existing schemes with the proposed system.

For IP lookup table in IPv6, the IP route only needs 1 cycle in the operations for the searching, updating and deleting functions. It will hit the correct next hop in the first cycle with 96.88% chance for 32000 entries. Moreover, the worst case is two memory cycles (1 hash + 1 memory). The result shows our architecture will be faster than the existing schemes in the search time. It only requires 1 Mbytes cache and 3.18 Kbytes CAM in the chip.

VII. References

[1] Y. Rekhter and T. Li., “An Architecture for IP Address Allocation with CIDR,” RFC 1518, Sep 1993.

[2] V. Fuller et al., “Classless Inter-Domain Routing,” RFC 1519, Jun 1993.

[3] R. Hinden and S. Deering, “Internet Protocol Version 6 (IPv6) Addressing Architecture,” RFC 3513, Apr 2003.

[4] R. Hinden, S. Deering and E. Nordmark, “IPv6 Global Unicast Address Format,” RFC 3587, Aug 2003.

[5] M. Waldvogel, G. Varghese, J. Turner and B. Plattner, “Scalable High Speed IP Routing Lookups,” in Proc. ACM SIGCOMM’ 97, pp.26-36, Cannes, France, Sep 1997.

[6] V. Srinivsan and G. Varghese, “Fast Address Lookups using Controlled Prefix Expansion”, in the ACM Tocs’99.

[7] B. Lampson, V. Srinivasan, and G. Varghese, “IP lookups using Multiway and Multicolumn Search”, IEEE INFOCOM'98, San Francisco, Apr 1998.

[8] R. Jain, “A Comparison of Hashing Schemes for Address Lookup in computer a Networks,” IEEE Transactions on Communications, vol. 40, no. 10, pp. 1570-1573, 1992.

[9] D. Yu, B.C. Smith and B. Wei, “Forwarding engine for fast routing lookups and updates,” Global Telecommunications Conference, vol. 2, pp.1556-1564, 1999,

[10] J. Xu, “A novel cache architecture to support layer-four packet classification at memory access speeds,” Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1445-1454, 2000.

[11] Merit Networks Inc., “Internet Performance Measurement and Analysis(IPMA) Statistics and Daily Reports,” see http://www.merit.edu/ipma/routing_table.

[12] Cheng-Chi You and Yuan-San Chu, “Fast IP Lookups and Update in Hardware,” ISCOM 2001.

[13] S. Nilsson and G. Karlsson, “IP-Address Lookup Using LC-Tries,” IEEE Journal on Selected Areas in Communications, vol. 17, no. 6, pp. 1083-1092, Jun 1999.

[14] W. Doeringer, G. Karjoth and M. Nassehi, “Routing on Longest-Matching Prefixes”, IEEE/ACM Trans. Networking, vol. 4, pp. 86-97, Feb 1996.

[15] B. Lampson, V. Srinivason, and G. Varghese, “IP lookups using Multiway and Multicolumn Search”, IEEE INFOCOM’98, San Francisco, April 1998, Session 10B-2.

[16] P. Gupta, S. Lin and N. McKeown, “Routing Lookups in Hardware at Memory Access Speeds”, IEEE INFOCOM’98, San Francisco, April 1998, Session 10B-1.

[17] M. Degemark, A. Brodnik, S. Carlsson, and S. Pink, “Small Forwarding Table for Fast Routing Lookups”, in Proc. ACM SIGCOMM’97, France, p. 34.

[18] E.K. Karuppiah and R. Abdullah, "Global IPv6 anycast address lookup with NP," Communications, 2003. APCC 2003. The 9th Asia-Pacific Conference on, vol. 3, pp. 21-24, Sep 2003.