Computer Networks xxx (2012) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

# A high-throughput and high-capacity IPv6 routing lookup system

Yi-Mao Hsiao *, Yuan-Sun Chu, Jeng-Farn Lee, Jinn-Shyan Wang

*Department of Electrical Engineering, The Advanced Institute of Manufacturing with High-Tech Innovations (AIM-HI), National Chung Cheng University, Chia-Yi, Taiwan*

ARTICLE INFO

*Article history:*
Received 23 May 2012
Received in revised form 17 September 2012
Accepted 2 November 2012
Available online xxxx

*Keywords:*
IPv6
Routing lookup
ASIC

ABSTRACT

With the growing number of routing entries, IP routing lookup has become the major performance bottleneck in backbone routers. In this paper, a complete hardware-based routing lookup system is proposed to achieve high-throughput and high-capacity for IPv6. The proposed system is a cache-centric, hash-based architecture that contains a routing lookup application specific integrated circuit (ASIC) and a memory set. A hash function is used to reduce lookup time for the routing table and ternary content addressable memory (TCAM) effectively resolves the collision problem. The gate count of the ASIC, excluding the binary content addressable memory (BCAM), is about 5306 gates, using an in-house 0.18 μm CMOS single-poly six-metal standard cell library. The results of post-layout simulations show that the ASIC operates in 3.6 ns so that the routing lookup system approaches 260 Mega lookups per second (Mlps), which is sufficient for 100 Gbps networks. The memory density is good, with each routing entry requiring only 64 bits. Moreover, the routing table only needs 10.24 KB on-chip BCAM, 20.04 KB off-chip TCAM and 29.29 MB DRAM for 3.6 M routing entries in the proposed system.

Published by Elsevier B.V.

## 1. Introduction

With the rapid growth in the number of Internet users and services, the increasing volume of network traffic is a major challenge for backbone routers. A 32-bit address length Internet protocol (IP) has been the standard for classless inter-domain routing (CIDR) technology [1], where each routing entry uses a ⟨prefix, prefix length⟩ pair to increase the effective size of the IP address space. However, CIDR suffers from the longest prefix match (LPM) problem [2] and the available IP addresses will soon be exhausted. The IPv6 protocol with a 128-bit address length could provide a long-term solution to the problem of insufficient IP addresses, but the switch to longer addresses will make the design of routing tables more complex. Furthermore, the amount of Internet traffic is expected to double every few months. By analyzing the trend in network backbone routers, we found that as the link rate increased from 10 Gbps to 100 Gbps [3] and number of routing entries grew from 20 K to 400 K [4]. Internet routing table sizes are expected to exponentially double every 2 years and are expected to reach one million routes in the near future [5,6]. Thus, the routing lookup function has become the major performance bottleneck in backbone routers [7–11].

A number of algorithms have been proposed to resolve the routing lookup problem. One approach [12] creates a routing lookup table with a trie, which is a simple and suitable structure for the IP addresses. Specifically, a trie is an ordered tree data structure that is used to store an associative array in which the keys are usually strings. Most trie-based approaches can achieve high average search throughput for IPv4 [12–15], but their update speed is slow. For IPv6, Li et al. proposed a modified tree data structure and algorithms to resolve the routing lookup problem [16–19]. However, the time complexity for searching is $O(\log N)$, which is still too long for IPv6 core networks.

A different approach has been to use a hash scheme. The time complexity for searching under a hash scheme is only

* Corresponding author. Tel.: +886 5 2720411x23280; fax: +886 5 2720862.
*E-mail address:* 93mowmow@vlsi.ee.ccu.edu.tw (Y.-M. Hsiao).

1389-1286/$ - see front matter Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.comnet.2012.11.001

$O(1)$. In a hash scheme, a prefix entry with the next hop information generates a key using a hash function. The key is then used to access the routing table directly for the IP routing lookup [20–23]. The drawback of the hash scheme is that there is a risk of collisions when different keys have the same hash value. Gupta et al. [24] proposed a hierarchical hardware architecture using RAM for IPv4 routing lookup. Searching the RAM is sequential, so the search time becomes very long when the number of routing entries is large. A hash scheme can be used in a hardware-based architecture to increase the lookup speed [25–27]. Chang and Lim [25] designed a high speed ASIC with a trie architecture for IPv6. Fadishei et al. [26] implemented a fast IP routing lookup architecture with a hash scheme based on the Field-programmable Gate Array (FPGA), but the lookup speed is slow and the entry capacity is too small to meet even the current requirements of backbone routers.

Another proposal has been the use of Content Addressable Memory (CAM), memory that implements the lookup function using dedicated comparison circuitry. CAM is widely used for routing lookup systems [28–32]. Francis and McAuley [28] perform routing lookup with CAM, with all matching actions in one clock cycle. Tan and Gong [30] uses both CAM array and TCAM for high speed IP lookup; while Akhbarizadeh utilizes a parallel TCAM architecture for high speed packet forwarding [31]. The drawbacks of conventional CAM are high power consumption and that the cost of CAM is higher than RAM. To resolve these problems, Sahni couples TCAM and SRAM to overcome the power and size limitations of pure TCAM forwarding engines [32], but routing lookup performance depends on the access time of the off-chip memory (TCAM and SRAM) since the recently used entries are not cached in the on-chip memory.

In this paper, we propose a high-throughput and high-capacity routing lookup system. The system is a cache-centric, hash-based architecture with a memory set (RAMs) and a TCAM to resolve the problem of collisions caused by the hash function. We determine the optimal address length of the two-level hash table architecture by observing the prefix length distribution of real-world routing tables, so that most entries can be looked up in the first level of hash tables. The number of buckets and the size of the hash buckets in our proposed system are determined based on a numerical analysis of the hash table model and simulations of synthetic IPv6 routing entries. The system contains a routing lookup ASIC and a BCAM as cache memory to speed up the search time. The memory density is also good, with each routing entry requiring only 64 bits. The routing lookup system approaches 260 Mlps, which is sufficient for 100 Gbps networks, and the routing table only needs 10.24 KB on-chip BCAM, 20.04 KB off-chip TCAM and 29.29 MB DRAM. It can support 3.6 M routing entries, which is sufficient for routing tables in the near future.

The remainder of this paper is arranged as follows. The next section provides a preliminary description of the IPv6 routing lookup. In Section 3, we describe the proposed system architecture. In Section 4, we explain the implementation of the ASIC. Section 5 presents the experimental results and an analysis of performance. Section 6 contains some concluding remarks.

## 2. The IPv6 routing lookup

### 2.1. IPv6 global unicast address

The format of the IPv6 global unicast address contains the global routing prefix, subnet ID and interface ID. As shown in Fig. 1, the global routing prefix (typically hierarchically structured) is a value assigned to a site (a cluster of subnets/links), and the subnet ID is the identifier of a link within the site. The resulting format of the global unicast address under the 2000::/3 prefix is currently being managed by Internet Assigned Numbers Authority (IANA) in accordance with the recommendations in [33].

A prefix in IPv6 is represented by the notation: address/prefix length. For example, the prefix 21A9:C767:FFFC::/46 means the address of the IPv6 prefix is 21A9:C767:FFFC::, with a prefix length of 46. The "::" is used to compress leading or trailing zeros in an address.

### 2.2. The prefix length distribution

The prefix length distribution is essential information for address routing lookup since if we design the correct address length for the first-level hash tables, most entries can be looked up in the first level of the hash. Thus, we examined this information in several real-word IPv6 routing tables: 6Net [34], BPG [4], Route Views [35] and RIPE RIS [36]. 6Net is a European project concerned with the continued growth of the Internet. BGP lists the number of unique routing entries in the Internet routing table. The Route Views project obtains real-time information about the global routing system from the perspective of several different backbones and locations around the Internet. RIPE is a forum open to all parties with an interest in the technical development of the Internet. The RIPE network coordination centre (NCC) is the Regional Internet Registry for Europe, the Middle East and parts of Central Asia, determining the allocation and registration of Internet number resources including IPv4 addresses, IPv6 addresses and Autonomous System numbers. The RIPE Routing Information Service (RIS) project collects and stores Internet routing data from several locations around the globe. Table 1 shows the distribution of prefix lengths for the 6Net router with 616 routing entries. The BGP tables for AS2.0 and AS6447 are analyzed in Fig. 2a. The table of AS2 was reported on May 1 05:45:02 2012 and AS6447 was on May 1 04:40:00 2012. The entry counts of the two BGP tables are 8435 and 8821, respectively. The Route Views tables for USA and Japan, with 651 and 595 entries, respectively, are shown in Fig. 2b. Four tables (rrc01, rrc03, rrc05 and

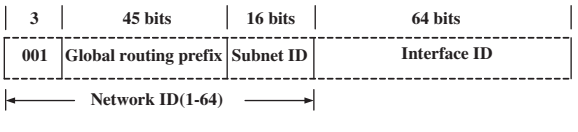



**Fig. 1.** IPv6 global unicast address format.

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001

**Table 1**
Prefix length distribution of 6 NET.

| Prefix length | Entry counts | % |
|---|---|---|
| 19–31 | 7 | 1.14 |
| 32 | 444 | 72.08 |
| 33–47 | 43 | 6.97 |
| 48 | 81 | 13.15 |
| 64 | 41 | 6.66 |
| Total | 616 | 100 |

rrc10) from RIPE RIS are analyzed in Fig. 2. All were reported at August 14 08:00 2012. According to our observations, the most common prefix lengths in the routing tables are 32 or 48. Based on the policy for assigning new Internet-wide IP addresses by the IANA organization, a local Internet register primarily assigns address space to the users of network services with a prefix length of 64. The total percentage of prefixes having a length of either 32, 48 or 64 in the nine data sets are 91.89%, 82.71%, 82.41%

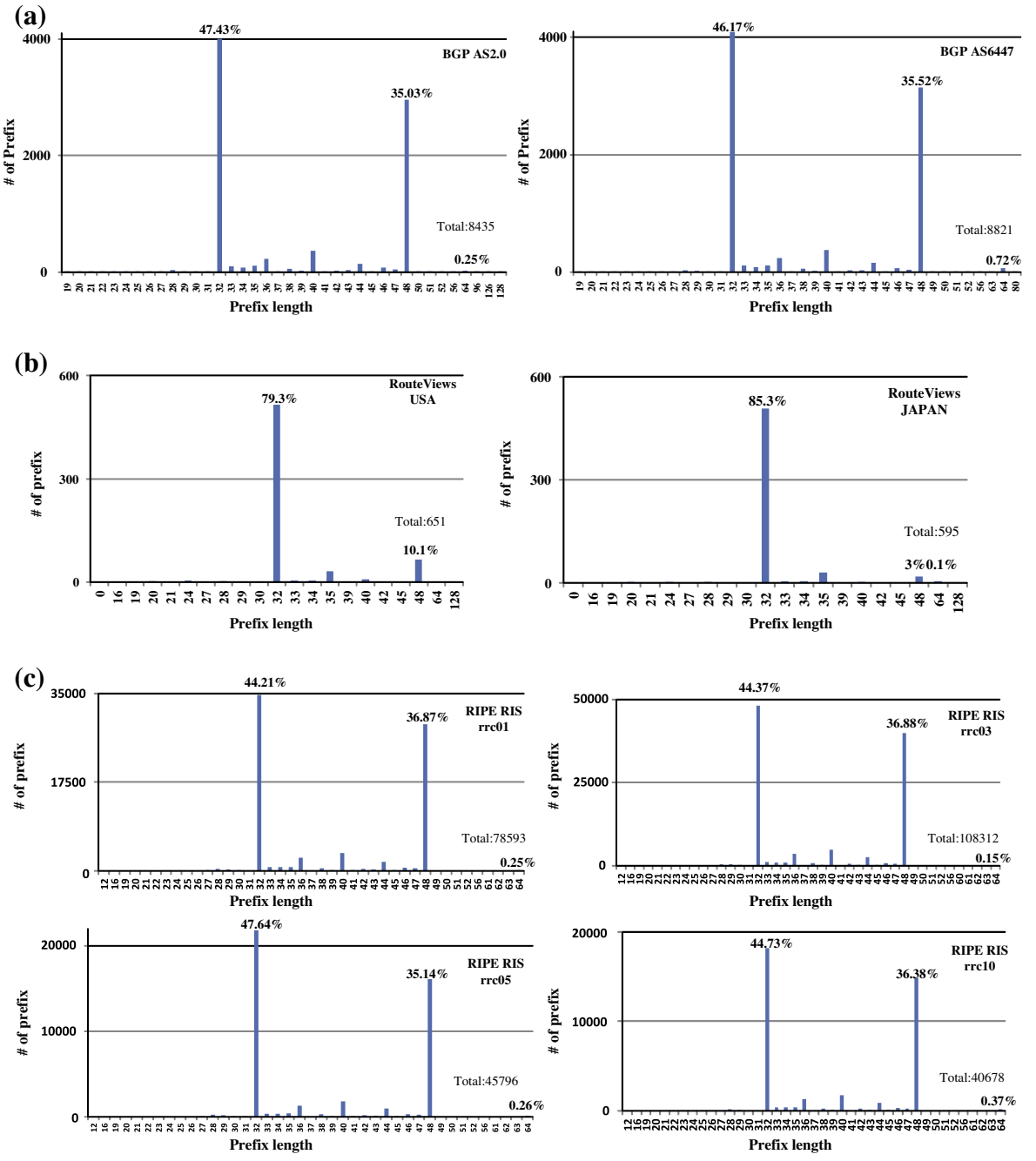



**Fig. 2.** Prefix length distribution: (a) the IPv6 BGP table; (b) the Route Views project; and (c) RIPE RIS project.

89.4%, 88.6%, 81.33%, 81.4%, 83.04% and 81.48%, respectively. Thus, the vast majority of routing entries use these three prefix lengths. Fig. 3 shows the historical data of prefix length distribution for rrc03 from 2003 to 2012. The prefix length of most routing entries is either 32 or 48. These represent over 80% of the prefix lengths assigned from 2003 to 2012. To design a fast routing lookup architecture, the first level of a hash table design will use a prefix length with these three lengths; the other prefix lengths, which are far less frequent, are put into the second level. This way, most of the routing entries (over 80%) can be searched in the first memory access.

## 3. System architecture

The proposed routing lookup system is a cache-centric and hash-based architecture, as shown in Fig. 4. Based on the network ID of the IPv6 global unicast address format, the system is designed to connect with the Internet backbone using 64 bit IPv6 addresses. The system provides insert, search, delete and update functions, and contains a routing lookup ASIC and memory sets. In ASIC, the cache memory has a hit ratio of up to 80% with a FIFO replacement algorithm, based on simulation results. Based on the prefix length distribution of existing IPv6 routing tables, we designed a hash architecture as shown in Section 3.2 to attain high performance routing lookup. Routing tables are stored in the memory set, which has a two-layer hierarchical memory architecture. The first layer of memory stores most of the routing entries, for prefix lengths of 32, 48 and 64. When hash collision happens, the routing entries are stored in a TCAM. The TCAM performs a parallel search to achieve high performance routing lookup. The second layer of the routing table contains two tables: TAB33_47, which stores entries with prefixes lengths between 33 and 47 bits; and TAB49_63, which stores entries with prefixes between 49 and 63 bits.

### 3.1. Hash table size

The collision problem causes the hash bucket to overflow. Selecting a sensible probability of overflow determines what is a reasonable size for the hash table. In [37], an exact probability model for finite hash tables is used to calculate the hash table size. In this hash table model, where $k$ is the total number of IP addresses, $b$ is the number of buckets in the hash table, and $s$ is the size of the bucket, there are $n$ IP addresses directed to the same bucket by a hash function. The probability $p(k,n,b)$ is defined as follows:

$$p(k,n,b) = C_n^k \left(\frac{1}{b}\right)^n \times \left(1-\frac{1}{b}\right)^{k-n} \tag{1}$$

The overflow probability is calculated by

$$\begin{aligned} P_{overflow} &= 1 - probability\ of\ nonOverflow \\ &= 1 - \left[ C_0^k \left(\frac{1}{b}\right)^0 \times \left(1-\frac{1}{b}\right)^k + C_1^k \left(\frac{1}{b}\right)^1 \times \left(1-\frac{1}{b}\right)^{k-1} \right. \\ &\quad \left. + \cdots + C_s^k \left(\frac{1}{b}\right)^s \times \left(1-\frac{1}{b}\right)^{k-s} \right] \\ &= 1 - \sum_{n=0}^{s} p(k,n,b) \end{aligned} \tag{2}$$

The expected number of IP addresses stored in one bucket is

$$\begin{aligned} ExpBucket(k,s,b) &= \sum_{n=0}^{s} p(k,n,b) \times n + \sum_{n=s+1}^{k} p(k,n,b) \times s \\ &= \sum_{n=0}^{s} p(k,n,b) \times n + \left[1 - \sum_{n=0}^{s} p(k,n,b)\right] \times s \\ &= s + \sum_{n=0}^{s} p(k,n,b) \times (n-s) \end{aligned}$$

The expected value of the overflow is $ExpOverflow(k,s,b)$:

$$\begin{aligned} &\because \frac{k - ExpOverflow(k,s,b)}{b} = ExpBucket(k,s,b) \\ &\therefore ExpOverflow(k,s,b) = k - b \times ExpBucket(k,s,b) \\ &\qquad = k - b\left[s + \sum_{n=0}^{s} p(k,n,b) \times (n-s)\right] \\ &\qquad = k - b\left[s + \sum_{n=0}^{s-1} p(k,n,b) \times (n-s)\right] \end{aligned} \tag{4}$$

The hash table size is determined by $b$ and $s$(i.e., $b^*s$). Two methods can be used to increase the size of the

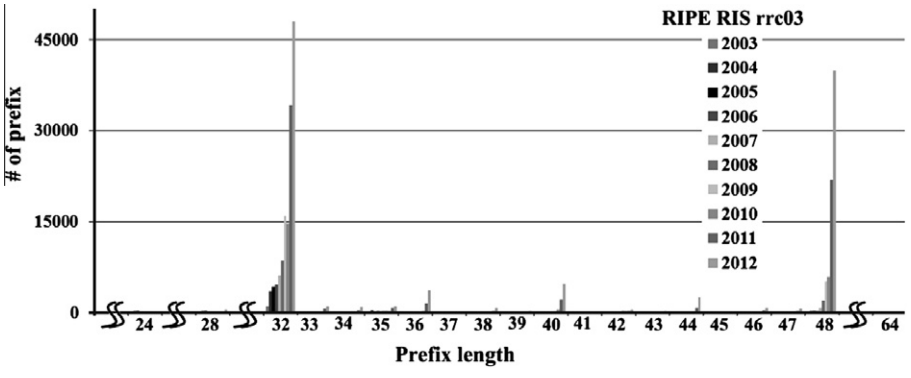



**Fig. 3.** The prefix length distribution of rrc03 RIPE RIS.

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001

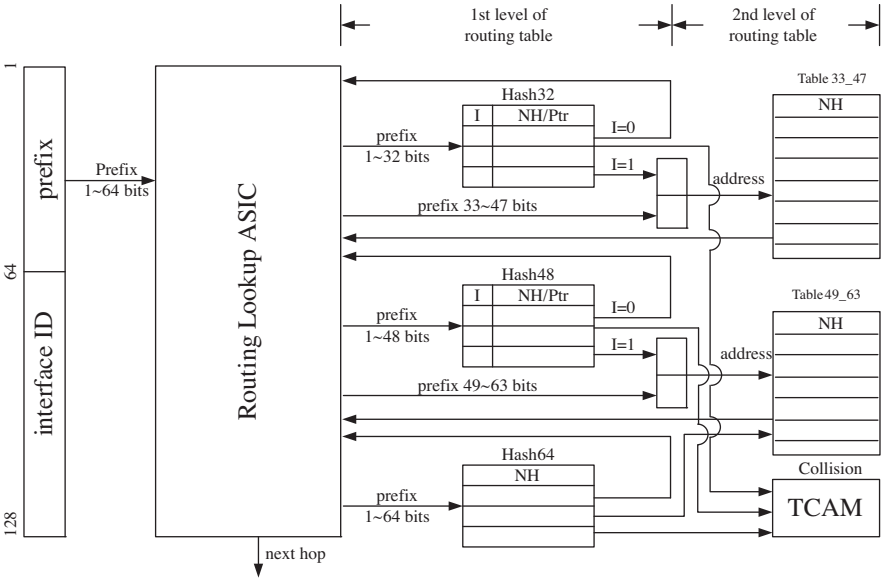
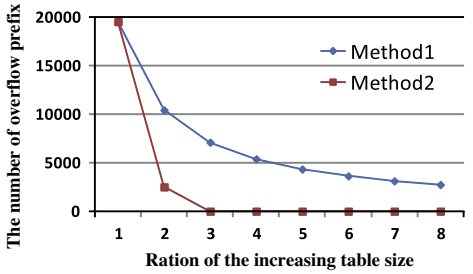



**Fig. 4.** The architecture of routing lookup system.

hash table. The first method increases the number of buckets ($b$), while the second increases bucket size ($s$). To compare the performance of the two methods, we use the value of $ExpOverflow(k,s,b)$ as an index. In method 1, increasing the number of buckets, the value of $(k,s,b)$ is $(108\,120,1,n\times 2^{18})$, and in method 2, increasing bucket size, the value of $(k,s,b)$ is $(108\,120,n,2^{18})$, where $n$ is defined as the ratio of increase in hash table size. Fig. 5 shows the results of increasing the hash table size for the two methods. If the hash table size is one, both methods experience the same number of overflows. As hash table size increases, the number of overflows with method 1 decreases slowly, but the number of overflows decreases quickly with method 2. Thus the second method is more suitable for increasing the size of the hash table. Therefore, we determined that the optimum number of buckets in a hash table is 2, since when the number of buckets exceeds 2, there are essentially no overflows.

The other essential issue is to decide optimal bucket size (i.e., the value of $b$) based on the number of buckets being 2. Since existing routing tables using IPv6 are small (108 K in RIPE RIS), we used synthetic routing entries in our simulations to find a suitable size for the bucket. We used V6Gen [38] to generate routing entries in IPv6 to simulate the relationship between overflow probability and bucket size, then determined ideal bucket size based on the simulation results. We set the prefix length distribution of 32, 48 and 64 as 72.08%, 13.15% and 6.66%, respectively, reflecting the values in the 6NET database. Based on CIDR and Cisco global IP traffic reports, we estimated the trend of routing table size as shown in Fig. 6. The routing table will exceed more than a million routes in 2020. To satisfy the expected trend in routing table size through 2035, 3.6 M prefixes were generated using V6Gen. Because the overflow routing entries are stored in TCAM, we set the target overflow ratio to be less than 5%, by adjusting the size of TCAM. The index sizes (i.e., the value of $b$) for Hash32, Hash48, and Hash64 tables are 18, 15, and 14, respectively, according to the simulation results shown in Table 2. This requires a DRAM size of 29.29 MB and a TCAM size of 20.04 KB for 3.6 M routing entries.




Method 1: Increasing number of buckets
Method 2: Increasing bucket size

**Fig. 5.** Number of overflows with increasing hash size.

### 3.2. Hash table design

In our hash table architecture, each hash table has two hash buckets in DRAM and one in TCAM. The major objectives of this design are to minimize the number of hash collisions and to speed up search time when a collision occurs. As shown in Fig. 7, using two hash buckets in DRAM reduces the hash collision probability significantly, as described in Section 3.1, and TCAM can match data in one clock cycle to resolve the hash collision problem. The hash function is XOR, which is a simple architecture that yields a good performance. Our simulation results demonstrate that the designed hash function with XOR is

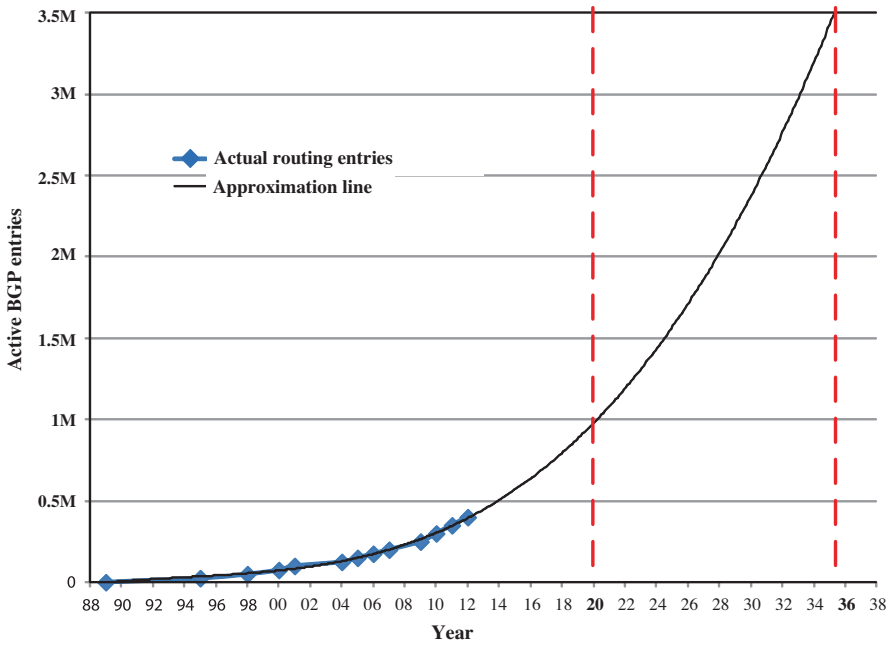



**Fig. 6.** IPv4 routing table size with estimated approximation line.

**Table 2**
Simulation results for 3.6 M entries

| Hash table | Prefix length (%) | Number of entries | Hash table size | Overflow (number) | Overflow (%) |
|---|---|---|---|---|---|
| Hash32 | 72.08 | 2594880 | $2^{17}$ | 274680 | 7.63 |
| | | | $2^{18}$ | 83520 | 2.32 |
| | | | $2^{19}$ | 23040 | 0.64 |
| Hash48 | 13.15 | 473400 | $2^{15}$ | 162360 | 4.51 |
| | | | $2^{16}$ | 46800 | 1.30 |
| | | | $2^{17}$ | 12600 | 0.35 |
| Hash64 | 6.66 | 239760 | $2^{14}$ | 165960 | 4.61 |
| | | | $2^{15}$ | 47880 | 1.33 |
| | | | $2^{16}$ | 12600 | 0.35 |

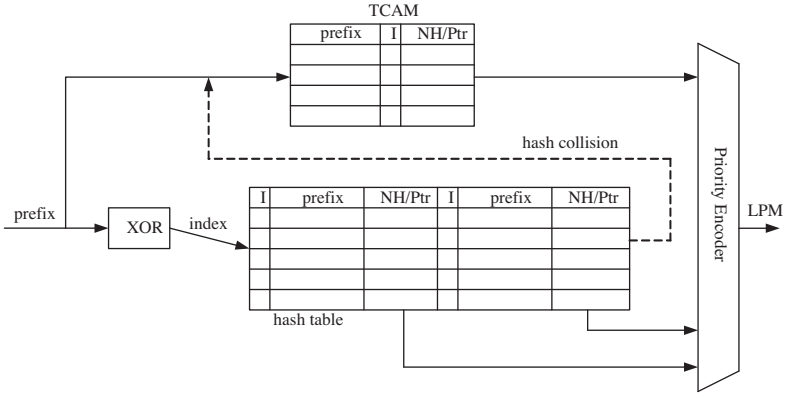



**Fig. 7.** Hash table architecture.

effective and efficient. Moreover, it is easy to implement. Detailed performance evaluations via simulations of the hash function are described in the next section.

The routing lookup performs search operations on the routing tables. After inputting the routing entries, the hash table stores these entries in a dedicated address designated

by the hash function. If there are hash collisions, these entries are stored in the TCAM. In the search operation, the next hop information can be looked up by the hash function from the hash table. Moreover, the TCAM searches all the entries in parallel at the same time. Then, the priority encoder selects the LPM given the next hop information. The next section describes our design of a *next hop (NH)* detector module to select the LPM entry.

## 4. Routing lookup ASIC

### 4.1. Routing lookup ASIC

The routing lookup ASIC contains the control unit, the datapath and the cache memory implemented in BCAM, as shown in Fig. 8. The control unit is responsible for the insert, search, update and delete functions. It also instructs the cache controller to execute the cache data search and cache replacement operations. The datapath in ASIC is comprised of BCAM, the *NH detector* and the *Hash Table Block*. The cache memory utilizes BCAM instead of the traditional SRAM since BCAM performs parallel searching in one clock cycle. Furthermore, we customize the BCAM to support insert, update and delete functions. A detailed description of BCAM is given in the next Section 4.2. The cache memory stores recently used routing entries. If a cache hit occurs, the ASIC can send the NH information to the router immediately. The on-chip BCAM performs high throughput routing lookup because of the ASIC operation speed. All the routing entries are stored as hash tables in DRAMs and TCAM. The tables are managed by the *Hash Table Block*, which contains the *hash table generator*, *hash data generator* and *2nd addr generator* blocks. We explain the details of the datapath in the next paragraph.

### 4.2. Datapath

#### 4.2.1. Cache replacement algorithm

The cache memory is an essential component of a routing lookup ASIC. To find a suitable cache replacement algorithm, we conducted simulations to compare the performance of five replacement algorithms: FIFO [39], Least Recently Used (LRU) [39], multi-segment Least Recently Used (mLRU) [40], Second Chance-Frequency Least Recently Used (SF-LRU) [41], and Least Frequently Used (LFU) [42]. Table 3 shows the results derived by the algorithms on the NLNAR traffic trace [43]. Overall, SF-LRU and LFU achieve the best performance, but they require counters to record the last reference time and their sorting action requires a more complex hardware design. FIFO yields the third best performance, and it needs only one register to record the next address for replacement. In the simulations, the cache size ranged from 16 to 1024. The cache hit ratio of SF-LRU, LFU and FIFO was over 80% on a cache size of 1024. However, as the hardware design of SF-LRU and LFU is more complex than FIFO, we select FIFO as the cache replacement algorithm for routing lookup in our system.

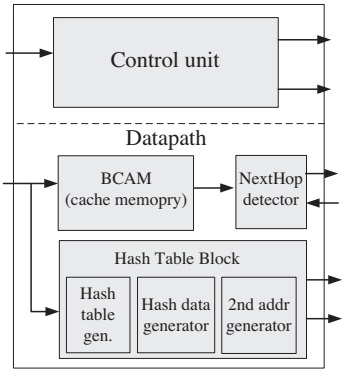



**Fig. 8.** ASIC routing lookup.

#### 4.2.2. BCAM design

The most frequently used routing entries are stored in the cache memory. In the proposed routing lookup ASIC, the BCAM is the cache memory that contains the cell structure and match line structure. Fig. 9 shows the BCAM architecture in our high-speed IPv6 routing lookup system. The architecture incorporates both circuit-level and gate-level techniques [44]. We use the pseudo-footless clock-and-data pre-charged dynamic circuit (PF-CDPD) technique to design the cell structure of BCAM. When the circuit is in the pre-charge phase, the search data can be transferred to the search line without influencing the circuit correction. The most critical time is when all the bits match and the bottle part of NMOS are logic 1. When the circuit changes to the evaluation phase, the PF-CDPD is like an inverter chain that transfers the data to the last stage to reduce the search time in the circuit. Besides reducing the search time, the advantages of PF-CDPD are the following: (1) low switching activity; (2) evaluation of a stage depends on the result of the preceding stage; (3) no concerns about the race condition or the DC current; and (4) reduction of the charge/discharge capacitance.

To speed up the search time of the lookup system, we designed a word structure so that the critical path is reduced to four stages. As Fig. 10a shows, a quadruple input AND gate design is simple and reduces the search time. The layout implementation as shown in Fig. 10b can reduce the resistance and capacitance of the metal.

#### 4.2.3. Hash function

A hash scheme is a suitable solution for designing a fast routing lookup system. Utilizing the hash function, the NH information is looked up by a key value. However, the hash scheme is affected by collisions that occur when too many entries are matched to the same key. To minimize the number of hash collisions, we have to choose a good hash function. Jain [45] defined entropy as the average reduction in search times, and used the entropy value to compare various hash functions. He proposed a hash model in which a mathematical function maps the given key to a hash cell $I$, which points to the sub-table of size $n\_i$. There are $R$ frames with $N$ distinct addresses and a hash table of $M$ cells. Given an address that hashes to the $i$th cell, searching through a sub-table of $n\_i$ entries requires only $\log_2(2n\_i)$ lookups. The total number of lookups saved is

**Table 3**
The hit ratio of cache replacement algorithms.

| Cache size | LRU (%) | mLRU (%) | mLRU (%) | SF-LRU (%) | LFU (%) | FIFO (%) |
|---|---|---|---|---|---|---|
| 16 | 44.41 | 36.20 | 20.68 | 44.81 | 38.10 | 37.86 |
| 32 | 49.37 | 42.21 | 27.07 | 49.66 | 42.86 | 43.28 |
| 64 | 54.72 | 45.84 | 36.20 | 57.15 | 48.65 | 49.21 |
| 128 | 61.50 | 49.91 | 42.21 | 64.10 | 55.71 | 56.07 |
| 256 | 67.27 | 57.33 | 45.84 | 71.50 | 63.50 | 63.46 |
| 512 | 73.52 | 64.10 | 49.91 | 79.66 | 71.97 | 72.08 |
| 1024 | 79.78 | 71.50 | 57.33 | 87.32 | 85.76 | 81.19 |

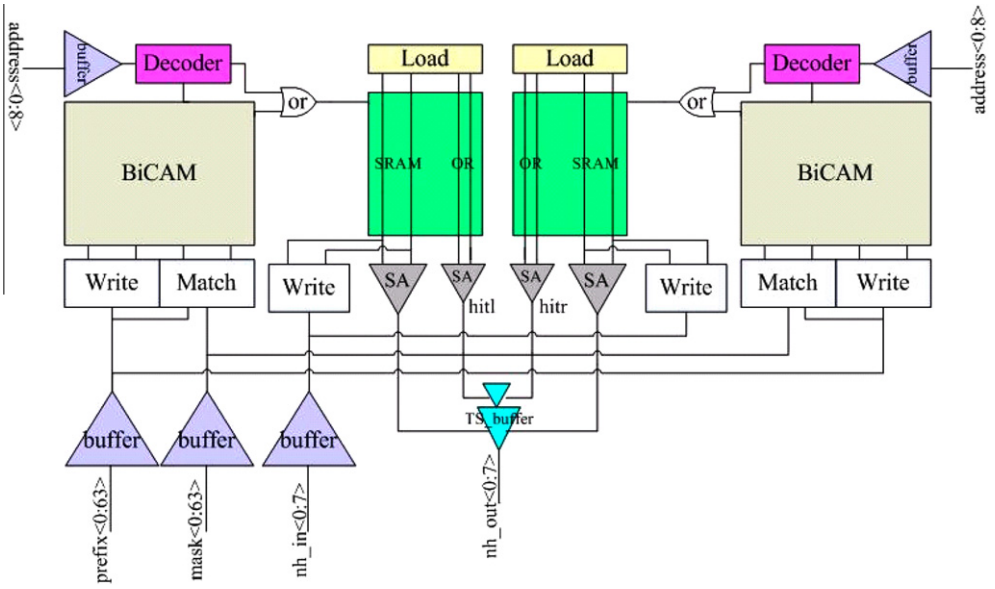



**Fig. 9.** The BCAM architecture for the high-speed IPv6 routing lookup system.

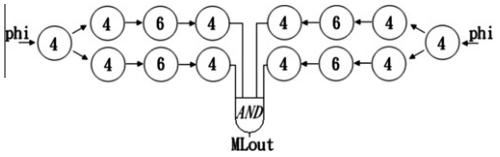



**Fig. 10a.** Two level tree type BCAM architecture.

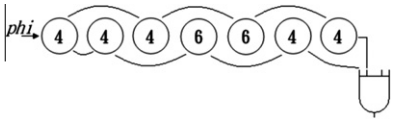



**Fig. 10b.** Layout implementation.

$$\sum_i r\_i[\log_2(2N) - \log_2(2n\_i)] \tag{5}$$

where $r\_i$ is the number of frames that hash to the $i$th cell $\sum_{r_i} = R$. Thus, the entropy is defined as

$$\sum_i -\frac{r\_i}{R}\log_2\left(\frac{n\_i}{N}\right) = \sum_i -q_i \log_2 p_i \tag{6}$$

To evaluate the performance of hash functions, we utilized a real trace of destination IP addresses on the TANET backbone router [46] in Taiwan. The trace recorded 7.6 million entries with 43 867 distinct destination addresses in a 1-h period. We simulated the following hash functions on the data: Bit Extraction, Fletch Checksum, exclusive-OR (XOR) Folding, and cyclic redundancy checking (CRC). Bit Extraction, the simplest function, uses the last 12 bits of an IP address as the hash index. The first 24 bits of an IP address are used to perform the CRC16 operation and the first to twelfth bits of the CRC16 result make up the hash index. Since each bit of CRC contains rich information, this hash function has a high computation cost. In XOR Folding, bits 1–24 of an IP address are divided equally into two units, i.e., 12 bits per unit, and the XOR operation uses them with the hash index. The entropy values of above hash functions are shown in Table 4. Although CRC performs slightly better than XOR Folding, it requires more complex computation in the hardware design. The Fletcher checksum and bit extraction functions do not perform well if the patterns of extracted bits are randomly distributed. Since XOR Folding achieves a good performance and does not need complex computation, we utilize it in our proposed routing lookup system.

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001

**Table 4**
The entropy of various hash functions.

| Hash function | Bit extraction | Fletch checksum | XOR folding | CRC |
|---|---|---|---|---|
| Entropy | 11.9694 | 11.9714 | 11.9753 | 11.9770 |

*4.2.4. Hash Table Block and NH detector*

The Hash Table Block contains the *hash table generator*, *hash data generator* and *2nd addr generator* blocks. We use XOR with a 64-bit index as the hash function, as described above. The hash table generator module generates the addresses of the hash tables, i.e., Hash32, Hash48 and Hash64. Then the *hash data generator* converts the prefix data into hash tables. The formats of the hash tables are shown in Table 5. In the first level, the prefix lengths are 32/48/64, the number of index bits is one and the next hop is eight bits. In the second level, the bit length is five and the length for next hop information is eight bits. The *2nd addr generator* generates the addresses of the second layer tables (i.e., TAB33_47 and TAB49_63).

The *NH detector* decides the next hop bit information based on the cache hit from the cache memory or hash table and outputs from the NH to the router, as shown in Fig. 11. CIDR has a drawback, namely LPM, that a router has to choose the longest prefix in the routing table. When searching the NH from the hash table, there may be more than one NH result matched. Thus, in our proposed system the NH from hash64 has the highest priority and the NH from hash32 has the lowest priority for the priority encoder in order to resolve the LPM problem. Next, the NH of LPM is sent to a multiplexer. If the cache memory hits, the NH is from the cache. Otherwise, the NH is the LPM from the hash tables.

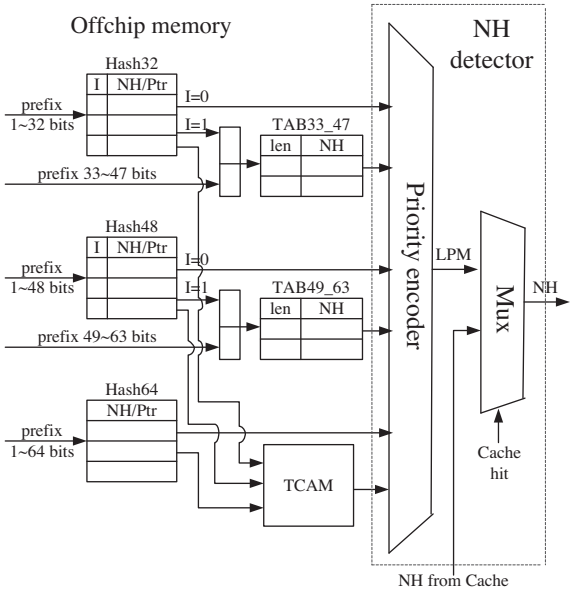



**Fig. 11.** The architecture of the NH detector.

*4.3. Control unit*

The control unit contains the main finite-state machine (FSM), RAM controller, Binary CAM controller and TCAM controller, which control all the components on the datapath. With the control unit design, the system supports insert, search, update and delete functions for the router.

*4.3.1. Search*

The search operation is the function of the routing lookup system that initiates the lookup operation. On-chip BCAM, off-chip DRAMs and TCAM are searched in parallel. If a cache hit happens, so that the prefix is found in the on-chip BCAM, the NH information is passed to the router immediately. In the off-chip DRAMs, the search operation starts at Hash32, Hash48, and Hash64. If the hash tables do not have an entry whose prefix information is the same as the incoming routing lookup data, the search operation is finished, and the function unit sends a hit signal of 0 to the router. However, if a matching entry exists, the function unit checks the *I* bit. If *I* bit is 0, the search operation is finished, and the *NH/Ptr* field in this matching entry becomes the next hop information for the router. An example of why the prefix 21A9:C767:FFFC::/46 uses a value of 1 for the *I* bit in the second level of the routing table is shown in Fig. 12. When the *I* bit of the matching entry in Hash32 is 1, the function unit gets the NH information from TAB33_47 with the following address:

"$NH/Ptr \times 2^{15} + \quad (33rd - 47th\,bits\,of\,destination\,IP)$". In this example, the address is $3 \times 2^{15} + 65532$. For the prefix 21A9:C767:FFFC::/46, the NH is A. The NH information in this entry is the routing lookup result. The NH of another prefix entry 18EA:4CB5:3A6B::/33 is D. The search operations are the same in TAB49_63 and TAB33_47.

**Table 5**
The data format of the off-chip routing tables.

| Component | Entry field | Field length | Function | Size |
|---|---|---|---|---|
| Hash32/48/64 | Prefix | 32/48/64 bits | Store prefix data | 21.11/3.85/0.19 MB |
| | *I* | 1 bits | *I* = 1: NH/Ptr is next hop information | |
| | NH/Ptr | 8 bits | *I* = 0: NH/Ptr is index for 2nd level routing table | |
| Table33_47/49_63 | Length | 5 bits | Used to create the address range | 2.06/2.06 MB |
| | NH | 8 bits | Next hop information | |

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001

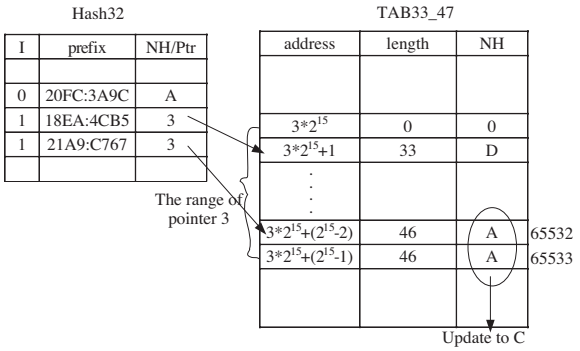



**Fig. 12.** An example of the relationship between Hash32 and TAB33 47.

*4.3.2. Update*

The update function changes the information about routing entries. When a routing path is changed, the entry in the routing table must be modified. The updating steps in the routing lookup system are as follows:

(1) The function unit searches for matching entries whose prefix is the same as the prefix of updated data in the hash table and the on-chip BCAM.
(2) The matching entries' next hops are updated with the new next hop information in the on-chip BCAM.
(3) The matching entries' next hops in the hash table are amended with the new next hop information.

As shown in Fig. 12, the original prefix is 21A9:C767:FFFC::/46 and the NH is A. If there is an update request to update its NH as C, the system calculates the second level of the routing table and finds the range is two. Finally, two addresses of NH information are changed from A to C. Because we designed the routing table as a two level memory set, the worst case memory access time for the update operation is generally twice the shortest. Although there are several NHs updated in TAB49_63 and TAB33_47 depending on the dedicated prefix and the length, we designed the *hash data generator* and *2nd addr generator* in our ASIC to update the NH of these entries. The memory controller in ASIC uses direct memory access; the update of the second level memory is operated in a micro way using the pipeline technique.

*4.4. ASIC implementation*

The ASIC VLSI design was partially written in Verilog code and synthesized by Synopsys using the CCU 0.18 μm CMOS single-poly six-metal standard cell library. The CCU standard cell library [47] is an in-house design in our school. The gate count of the ASIC, excluding Binary CAM, is about 5306 gates; and the gate count of the Binary CAM in the ASIC is nearly 138 074 NAND gates. The ASIC design was converted to the physical layout shown in Fig. 13, using placement and routing tools. The core area of the layout is 1.93 mm × 1.99 mm. The layout was double-checked with the design rule check (DRC) and layout versus schematic rules (LVS). It operates at 277 MHz, using

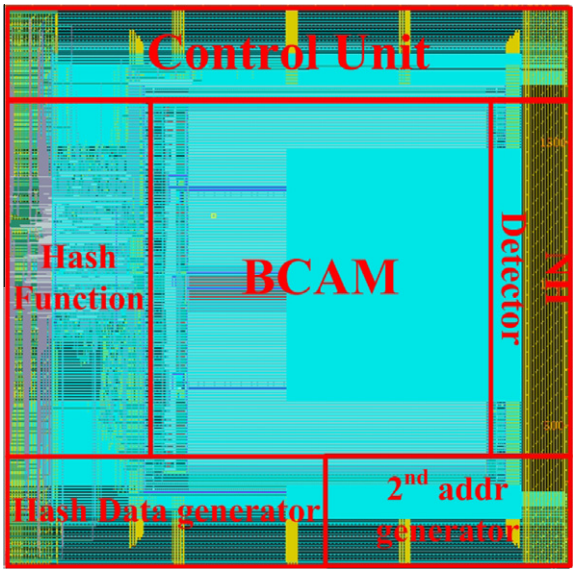



**Fig. 13.** ASIC layout of the routing lookup system.

Nanosim for post-layout simulation. The supply voltage is 1.8 V and the power dissipation is about 109.18 mW. The chip information is summarized in Table 6.

## 5. Results and analysis

Currently, the fastest line rate is 100 Gbps, and the smallest Ethernet frame is 64 bytes. It is assumed that the traffic flow rate in the backbone router is as follows:

$$\begin{aligned} \textit{Traffic Flow Rate} &= \textit{Line Rate} \div \textit{Frame Size} \\ &= (100\ \text{Gbs/s})/(64\ \text{bytes}) \\ &= 195(\textit{Mega lookups per second}) \end{aligned}$$

$$\begin{aligned} &\textit{Routing lookup speed of the system}: \\ &\textit{Average routing lookup time} = \textit{hit ratio} \times \textit{cache search time} \\ &\quad + \textit{miss ratio} \times \textit{non} - \textit{collision accesstime}(\textit{RAM}) \\ &\quad + \textit{miss ratio} \times \textit{collision} \times \textit{TCAM access time} \\ &\textit{Throughput} = 1 \div \textit{average routing lookup time} \end{aligned}$$

The clock period of the implemented ASIC is 3.6 ns, so the routing lookup speed is 277 MHz. The clock periods for commercial TCAM [48] and DRAM [49] are 10 ns and 5 ns, respectively. The above information is used for the off-chip and on-chip parameters to calculate the

**Table 6**
Specifications of the implemented ASIC.

| Technology | 0.18 μm 1P6M mix-signal CMOS |
|---|---|
| Core supply voltage | 1.8 V |
| Core area | 1.93 × 1.99 mm$^2$ |
| Logic gate count | 5306 gates |
| BCAM size | 10.24 KB |
| Total gate count | 143 K gates |
| Clock frequency | 277 MHz |
| Power consumption | 109.18 mW |

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001

**Table 7**
Summary of routing lookup speeds.

| | On-chip (ns) | Off-chip (ns) | Worst case (cycles) | Mlps |
|---|---|---|---|---|
| ICC 04 [30] | 10 | 10 | 2 | 97 |
| TCS 04 [25] | 33 | – | 1 | 30 |
| TComputer 07 [31] | 3.9 | 10 | 1 | 72 |
| Micro 08 [26] | 5.1 | 5 | 1 | 164 |
| Infocom 09 [8] | 6.6 | 4 | 1 | 149 |
| TNetworking 12 [9] | 5 | 5 | 6 | 200 |
| Infocom 12 [11] | 3.4 | 4 | 1 | 250 |
| This work | 3.6 | 5 | 2 | 260 |

**Table 8**
Comparison with other systems.

| Design | [30] ICC 2004 | [25] TCS 2004 | [31] TComputer 2007 | [26] Microp. & Micros. 2008 | [8] Infocom 2009 | [9] TNetworking 2012 | [11] Infocom 2012 | This work |
|---|---|---|---|---|---|---|---|---|
| Strategy | Hash | Tree | Multi-selector Multi-block | Hash + Update Tree | Hash | Hash + Compressed Tree | Tree | Cache + Hash |
| Table storage | CAM | RAM | CAM | RAM | RAM | RAM | RAM + CAM | CAM + RAM |
| Lookup speed (ASIC) | 100 Mlps | 30 Mlps | 100 Mlps | 193 Mlps | 180 Mlps | 200 Mlps | 297 Mlps | 277 Mlps |
| Update (worst case) | 4 | N/A | N/A | 2 | 1 | 5 | 1 | 2 |
| On-chip memory | CAM: 1.55 MB TCAM: 0.094 MB | SRAM: 0.59 MB | N/A | SRAM: 0.002 MB | SRAM: 21.6 MB | SRAM: 1.17 MB | SRAM: N/A | BCAM: 0.01 MB |
| Off-chip memory | N/A | N/A | TCAM: 18.4 MB | SRAM: 8.5 MB | SRAM: 36 MB | DRAM: 19.5 MB | TCAM: N/A | TCAM: 0.02 MB DRAM: 29.29 MB |
| Bits per entry | 12.13 | 14.75 | 72 | 38.7 | 320 | 83 | N/A | 64 |
| Capacity (# of entries) | 128 K | 40 K | 256 K | 220 K | 180 K | 2558 K | N/A | 3665 K |
| System throughput | 97 Mlps | 30 Mlps | 72 Mlps | 164 Mlps | 149 Mlps | 200 Mlps | 250 Mlps (Theoretical) 5.95 Mlps (Measured) | 260 Mlps |

Microp. & Micros.: Microprocessors and Microsystems.

throughput of the routing lookup system. Thus, in terms of system throughput, the design can provide approximately 260 Mega lookups per second (Mlps), which is sufficient for 100 Gbps. Table 7 shows the estimated system lookup speed of various approaches.

In Table 8, we compare the proposed cache-centric hash-based architecture with existing approaches. All architectures of the approaches compared in Table 8 use tree-based or hash-based strategies, with routing table data stored in the RAM or CAM. The memory size on-chip and off-chip are also shown in the comparison table. To compare the performance of routing lookup hardware design, we analyzed system throughput and ASIC lookup speed. We also estimated the system throughput of all related works. The results show that [26,9,11] and our system can all achieve 100 Gbps. The system throughput of [30,25,31] are slow and their capacities are small: 128 K, 40 K and 256 K, respectively, capacity being the number of entries in the hardware routing table. On the other hand, their memory densities, the bits per entry, are good: 12.13 and 14.75 for [30,25], respectively. [26] can achieve 100 Gbps and the memory density is good, 38.7, but it only supports 220 K routing entries. The system throughput of [9] is 200 Mlps and supports 2558 K entries. This is the only architecture apart from ours that can support more than 2 M entries. We anticipate that our design will be able to support more than 3 M entries with advances in the future. The system described in [11] achieves a throughput of 297 Mlps in ASIC lookup speed and 250 Mlps of system throughput based on the theoretical and the maximum speed of FPGA and TCAM. [11] is theoretically the fastest lookup hardware design, but the measured speed is only 5.95 Mlps because of a performance bottleneck in the Gigabit Ethernet interface card. The memory size and capacity are not mentioned [11]. We estimate memory density as a way of determining the memory efficiency of the routing lookup system. As shown in Table 8, [30] has the lowest density at 12.13 bits per entry and [8] has the highest density at 320 bits per entry. Our memory density is 64 bits. Although this work used 0.03 MB CAM (on-chip and off-chip) at a little higher cost than RAM, it is cheaper than the cost of [30,31]. Our system supports full functions, including update. The systems described in [25,31] do not support the update function. The update function of [30]

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001

requires 4 cycles of memory access and [9] needs 5 cycles. Because they use a compressed tree, in the worst case the update requires that the memory be accessed many times, slowing the process.

The limitation of an ASIC is that it is not as flexible as software. If the prefix length distribution changes so that the length of most routing entries is no longer 32, 48 and 64, most of the search operations would require second level memory so that the search operation in the hash table would have to access memory twice. The cache-centric architecture of our system stores the most frequently used routing entries in the cache memory (on-chip BCAM), producing a high throughput routing lookup system.

## 6. Conclusion

We have proposed a high-throughput, low-cost, high-capacity IPv6 routing lookup system, and designed a hash-based architecture with a cache memory. The system is comprised of a routing lookup ASIC and memory set. The ASIC contains a function unit and a BCAM. The function unit performs insert, search, update, and delete operations in the routing lookup system. The BCAM, which is used as a cache memory, can guarantee an 80% hit ratio. FIFO is used as cache replacement algorithm in the proposed architecture. In the hierarchal memory set design, most routing entries can be found in the first memory access, in the worst case requiring two memory accesses. The routing lookup speed of proposed system is 260 Mlps, satisfying the requirement of 100 Gbps. The high speed IPv6 routing lookup system requires only 20.04 KB TCAM, 10.24 KB BCAM, and 29.29 MB DRAM for 3.6 M routing entries.

## References

[1] J. Yu, V. Fuller, T. Li, K. Varadhan, RFC1519: Classless Inter-Domain Routing (CIDR): an Address Assignment and Aggregation Strategy, Internet Engineering Task Force (IETF), September 1993.
[2] W. Doeringer, G. Karjoth, M. Nassehi, Routing on longest-matching prefixes, IEEE/ACM Trans. Netw. 4 (1) (1996) 86–97.
[3] Cisco Router T1600. <http://www.cisco.com>.
[4] BGP Routing Table. <http://bgp.potaroo.net/>.
[5] CIDR Report. <http://www.cidr-report.org/>.
[6] Cisco Global IP Traffic Report. <http://www.cisco.com/en/US/netsol/ns827/networking_solutions_sub_solution.html>.
[7] A. Singhal, R. Jain, Terabit switching: a survey of techniques and current products, Comput. Commun. 25 (2002) 547–556.
[8] H. Song, F. Hao, M. Kodialam, T.V. Lakshman, IPv6 lookups using distributed and load balanced bloom filters for 100 Gbps core router line cards, in: Proc. IEEE INFOCOM, 2009.
[9] M. Bando, Y.-L. Lin, H.J. Chao, FlashTrie: beyond 100-Gb/s IP route lookup using hash-based prefix-compressed trie, IEEE/ACM Trans. Netw. 20 (4) (2012) 1262–1275.
[10] K. Huang, G. Xie, Y. Li, Alex X. Liu, Offset addressing approach to memory-efficient IP address lookup, in: Proc. IEEE INFOCOM, 2011.
[11] L. Luo, G. Xie, Y. Li, L. Mathy, K. Salamatian, A hybrid IP lookup architecture with fast updates, in: Proc. IEEE INFOCOM, 2012.
[12] G. Varghese, B. Lampson, V. Srinivasan, IP lookups using multiway and multicolumn search, IEEE/ACM Trans. Netw. 7 (3) (1999) 324–334.
[13] J. Turner M. Waldvogel, G. Varghese, B. Plattner, Scalable high speed IP routing lookups, in: Proc. ACM SIGCOMM'97, 1997.
[14] A.K. Somani, R. Sangireddy, High-speed IP routing with binary decision diagrams based hardware address lookup engine, IEEE J. Sel. Areas Commun. 21 (4) (2003) 513–521.
[15] L.-C. Wuu, T.-J. Liu, K.-M. Chen, A longest prefix first search tree for IP lookup, Comput. Netw. 51 (12) (2007) 3354–3367.
[16] Z. Li, D. Zheng, Y. Ma, Tree, Segment table, and route bucket: a multi-stage algorithm for IPv6 routing table, in: Proc. IEEE INFOCOM, 2007.
[17] Y.-K. Chang, Fast binary and multiway prefix searches for packet forwarding, Comput. Netw. 51 (3) (2007) 588–605.
[18] X. Huang, X. Zhao, G. Zhao, W. Jiang, D. Zheng, Q. Sun, Y. Ma, A novel level-based IPv6 routing lookup algorithm, in: Proc. IEEE GLOBECOM, 2008.
[19] H. Park, H. Hong, S. Kang, An efficient IP address lookup algorithm based on a small balanced tree using entry reduction, Comput. Netw. 56 (1) (2012) 231–243.
[20] H. Lim, J.-H. Seo, Y.-J. Hung, High speed IP address lookup architecture using hashing, IEEE Commun. Lett. 7 (10) (2003) 502–504.
[21] Y.-K. Chang, A small and fast IP forwarding table using hashing, IEICE Trans. Commun. E88-B (1) (2005).
[22] Q. Sun, X.Huang, X. Zhou, Y. Ma, A dynamic binary hash scheme for IPv6 lookup, in: Proc. IEEE GLOBECOM, 2008.
[23] Z. Huang, J.-K. Peir, S. Chen, Approximately-perfect hashing: improving network throughput through efficient off-chip routing table lookup, in: Proc. IEEE INFOCOM, 2011.
[24] S. Lin, P. Gupta, N. McKeown, Routing lookups in hardware at memory access speeds, in Proc. IEEE INFOCOM, 1998.
[25] R.C. Chang, B.-H. Lim, Efficient IP routing table VLSI design for multigigabit routers, IEEE Trans. Circ. Syst. I: Fundam. Theor. Appl. 51 (4) (2004).
[26] H. Fadishei, M.S. Zamani, M. Sabaei, A fast IP routing lookup architecture for multi-gigabit switching routers based on reconfigurable systems, Microprocess. Microsyst. 32 (4) (2008) 223–233.
[27] W. Jiang, Q. Wang, V.K. Prasanna, Beyond TCAMs: an SRAM-based parallel multi-pipeline architecture for terabit IP lookup, in: Proc. IEEE INFOCOM, 2008.
[28] P. Francis, A.J. McAuley, Fast routing table lookup using CAMs, in: Proc. IEEE INFOCOM, 1993.
[29] K. Pagiamtzis, A. Sheikholeslami, Content-addressable memory (CAM) circuits and architectures: a tutorial and survey, IEEE J. Solid-State Circ. 41 (3) (2006) 712–727.
[30] M. Tan, Z. Gong, High speed IP lookup algorithm with scalability and parallelism based on CAM array and TCAM, in: Proc. IEEE ICC, 2004.
[31] M.J. Akhbarizadeh, M. Nourani, R. Panigrahy, S. Sharma, A TCAM-based parallel architecture for high-speed packet forwarding, IEEE Trans. Comput. 56 (1) (2007) 58–72.
[32] W. Lu, S. Sahni, Low power TCAMs for very large forwarding tables, in: Proc. IEEE INFOCOM, 2008.
[33] S. Deering, R. Hinden, E. Nordmark, RFC3587: IPv6 Global Unicast Address Format, Internet Engineering Task Force (IETF), August. 2003.
[34] 6NET, Large-Scale International IPv6 Pilot Network. <http://www.6net.org/>.
[35] University of Oregon Route Views Project. <http://www.routeviews.org/>.
[36] RIPERIS. <http://www.ripe.net/data-tools/stats/ris/routing-information-service>.
[37] M.V. Ramakrishna, An exact probability model for finite hash table, in: Proc. IEEE Fourth International Conference on Data Engineering, February 1991.
[38] K. Zheng, B. Liu, A Scalable IPv6 prefix generator for route lookup algorithm, in: Proc. IEEE Advanced Information Networking and Applications, 2006.
[39] A.S. Tanenbaum, A.S. Woodhull, Operating Systems: Design and Implementation, Second ed., Prentice Hall, 1996.
[40] H. Liu, Reducing cache miss ratio for routing prefix cache, in: Proc. IEEE GLOBECOM, 2002.
[41] A. Akaaboune, J. Alghazo, N. Botros, Sf-lru cache replacement algorithm, in: Proc. International Workshop on Memory Technology Design and Testing, 2004.
[42] C.-S. Wu, W.-L. Shyu, T.-C. Hou, Efficiency analyses on routing cache replacement algorithms, in Proc. IEEE ICC, 2002.
[43] Nlnar Measurement and Network Analysis. <http://pma.nlanr.net/>.
[44] C.-C. Wang, J.-S. Wang, C. Yeh, High-speed and low-power design techniques for TCAM macros, IEEE J. Solid-State Circ. 43 (2) (2008) 530–540.
[45] R. Jain, A comparison of hashing schemes for address lookup in computer networks, IEEE Trans. Commun. 40 (3) (1992) 1570–1573.
[46] The Computer Center of the Ministry of Education. <http://www.edu.tw/tanet/introduction.html>.

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001

[47] The SoC group of Electrical Engineering department in National Chung Cheng University. <http://www.soc.ccu.edu.tw>.
[48] Renesas TCAM. <http://tw.renesas.com/media/products/memory/TCAM/p20_tcam_products.pdf>.
[49] Micron DRAM, 256Mb DDR SDRAM, MT46V32M8P-5B. <http://www.micron.com>.

**Yi-Mao Hsiao** is currently a Ph.D. student in the Department of Electrical Engineering at National Chung-Cheng University. He received the B.S. degree in Information Engineering from Feng Chia University in 2004 and the M.S. degree in Electrical Engineering from National Chung-Cheng University in 2006. His research interests are digital VLSI system design, high speed routing lookup system design, computer network, and Quality of Service system design.

**Yuan-Sun Chu** received the B.S. degree in electrical engineering from Feng-Chia University, Taichung, Taiwan in 1978, and the M.S. and Ph.D. degrees in electrical engineering from Katholieke Universiteit Leuven, Leuven, Belgium, in 1986 and 1991, respectively. He is a Professor with the Department of Electrical Engineering, and the director of SOC Research Center, National Chung Cheng University, Chiayi, Taiwan. His research interest includes computer architecture, VLSI low power design, multimedia network IC design.

**Jeng-Farn Lee** received the B.S., M.S., and Ph.D. degrees from National Taiwan University, Taipei, Taiwan, in 1998, 2000, and 2007, respectively. He was a Postdoctoral Fellow with the Institute of Information Science, Academia Sinica, Taipei, until July 2007. In August 2007, he joined the Department of Computer Science and Information Engineering, National Chung-Cheng University, Chia-yi, Taiwan, as an Assistant Professor. His current research interests include quality-of-service networking, scheduling, and wireless access networks.

**Jinn-Shyan Wang** (S'85–M'88) received the B.S. degree in electrical engineering from the National Cheng-Kung University, Tainan, Taiwan, in 1982 and the M.S. and Ph.D. degrees from the Institute of Electronics, National Chiao-Tung University, Hsinchu, Taiwan, in 1984 and 1988, respectively. He was with the Industrial Technology Research Institute (ITRI) from 1988 to 1995, engaged in ASIC circuit and system design, and became the Manager of the Department of VLSI Design. He joined the Department of Electrical Engineering, National Chung-Cheng University, Chia-Yi, Taiwan, in 1995, where currently he is a full Professor. He funded the SOC Research Center of National Chung-Cheng University. His research interests are in low-power and high-speed digital integrated circuits and systems, analog integrated circuits, IP and SOC design, and CMOS image sensors. He has published over 35 journal papers and 50 conference papers and holds over 30 patents on VLSI circuits and architectures. Dr. Wang has served as an ITPC member of IEEE ISSCC since 2007.

Please cite this article in press as: Y.-M. Hsiao et al., A high-throughput and high-capacity IPv6 routing lookup system, Comput. Netw. (2012), http://dx.doi.org/10.1016/j.comnet.2012.11.001