FPGA based low-latency IPv6 Route Lookup Using Dynamic XOR Table

Takeshi Matsuya*, Hajime Tazaki†, Yohei Kuga*, Rodney Van Meter*, Osamu Nakamura*
*Keio University †University of Tokyo
{macchan, tazaki, sora, rdv, osamu}@sfc.wide.ad.jp

IPv6 full-route lookup in current high-end routers employs Ternary Content-Addressable Memory (TCAM) or DRAM to ensure the required lookup performance for packet forwarding. TCAM has the benefit of low-latency access but consumes a lot of power, while DRAM has a high-delay access as well as frequent memory access to implement a Trie structure. In this poster, we propose a method for lookups on the IPv6 routing table by using small-scale block RAMs embedded in an FPGA.

Various prior work has improved the speed of routing table lookup, such as DIR-24-8-BASIC [1], but is not applicable since an IPv6 address is 128 bits. Our idea is to configure hash tables dynamically by using XOR.

Figure 1 illustrates the hashing method that we used for storing network prefixes in block RAMs. In this example, an IPv6 routing entry with 64-bit network address and 8-bit prefix length is hashed into a 4-bit value used as a memory address.

The XOR Hash Table (XHT) defines the hash function used, with one XHT entry for each bit of the resulting hash. In the figure1, since the third XHT entry has the value C0000000, the upper two bits of the input IP address are extracted, XORed, and assigned to the third bit of the memory address. When we receive a routing table update, this hash function can be modified to reduce the probability of hash collisions.

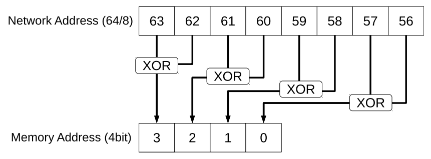
Based on this idea, we designed a block diagram for the FPGA implementation, depicted in figure 2. The implementation uses a pair of an XOR Hash Table and a Block RAM: this pair is used by each different network prefix (e.g., 2001:db8:1:0::/64) and uses multiple pairs when the number of entries in routing table is large. In this example, since the address width of block RAM is 9 bits, we can register 512 routing entries per 1 block RAM, at maximum.

The following describes the method to lookup an IP routing entry when the prefix length is 64-bit long as shown in the top of the figure 2.

(1) Output the upper 64 bits of IPv6 address to the Hash Module and Compare Module

(2) Hash Module translates into a 9-bit memory address based on pre-registered entry in XOR Hash Table

(3) Block RAM generates and stores the upper 64-bit of IPv6 address before applying a hash function and a gateway ID in the lower 8-bit.



	Value
XOR Hash Table (3)	C0000000
XOR Hash Table (2)	30000000
XOR Hash Table (1)	0C000000
XOR Hash Table (0)	03000000

Figure 1: XOR Hash table

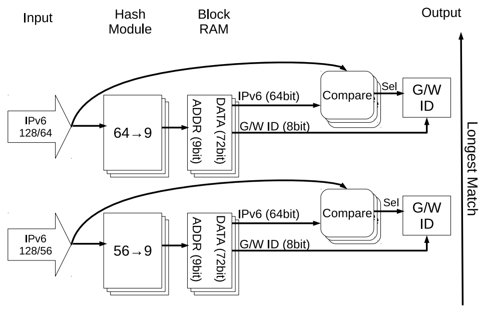


Figure 2: Block Diagram

(4) Compare Module assigns 1 as a Select signal if the address generated by (1) and the IPv6 before applying a hash function are identical.

(5) By applying the longest match, the most preferred Select signal is chosen and outputs the gateway ID.

(6) If the gateway ID is not zero, the FPGA replies IP address with a MAC address.

We are implementing this logic on a Xilinx FPGA board. The whole process of lookup costs about 5 cycles, which will give us a 17*ns* delay with a 300 MHz processor. Since the delay of simple DRAM access is about 40*ns*, our proposed method has a benefit on the latency.

References

[1] P. Gupta, S. Lin, and N. McKeown. Routing lookups in hardware at memory access speeds. In *Proceedings of InfoCom '98*, 1998.