FPGA Implementation Of An Efficient Ip Routing Filter

M Suganthi[1] A.Albert Raj[2]

ABSTRACT : In this paper we propose a new mechanism for an efficient router design with focus to an IP address lookup algorithm, which uses a small amount of memory. Typically each filter is a destination address prefix and longest prefix matching is used to determine the next hop for each incoming packets By exploiting the low memory access latency and high bandwidth of on chip memory high speed packet forwarding can be achieved using this data structure. When designing a router three pertinent issues are to be addressed i.e., routing lookup, switching and scheduling. The main objective of this paper is to design an efficient router that uses a fast routing lookup algorithm and an efficient data compression algorithm to store the routing table in a tree, which uses a very little memory in the router. The route lookup mechanism proposed in this paper, when implemented in a pipeline fashion in hardware, can achieve one route lookup for every memory access. With the current 50 ns DRAM, this corresponds to approx 20 x 10^6 packets (lookups) per second. Analysis shows that this algorithm needs only 400kb memory for storing 20k entries thus achieving a high compression. This design can be easily scaled up from IPv4 to IPv6.

Index Terms: IP address lookup, Tree structure, Nine Coded Compression algorithm, router.

1. INTRODUCTION Routing of Internet protocol (IP) packets is the primary purpose of Internet routers. Simply stated, routing an IP packets involves forwarding each packet along a multi hop path from source to destination. For internet protocol version 4 (Ipv4) the forwarding decision is based on the 32 bit destination address carried in each packets header.. The main subject of this paper is to deal with the two other major issues, a high speed on chip IP address lookup algorithm for routing lookups in hardware at memory access speeds and an a nine coded compression technique for effectively compressing the storage space of the lookup table in the router.

The work is motivated by the need for fast route lookups in particular, we are interested in hardware implementable lookup algorithms. We desire a lookup algorithm that achieves the following goals.

1.The lookup must be easily implementable in hardware using simple logic.

2. Ideally, the route lookup procedure should exactly one memory access time.

3. If it takes more than one memory access, then the number of access should be small.

4. Practical considerations involved in real implementation such as cost are an important concern.

5. The overhead to update the forwarding table should be small.

The IP address lookup algorithm proposed in this paper uses a small amount of memory. With this merit, the algorithm can be implemented in one single chip. The main objective of this paper is the development of a novel

[1]Asst. Prof. Department of ECE, Thiagarajar College of Engineering, Madurai

[2]Asst. Prof. & Head, Department of EIE, N.I College Of Engineering, Kumarakoil

prefix compression algorithm for compactly storing the IP address lookup table.

The rest of the paper is organized as follows. In section 2 number of research work is reviewed. In section 3 lookup table algorithm is presented in section 4 we give the concepts of Nine coded compression algorithm. Section 5 deals with the Hardware design model, and finally In Section 6 we compared this paper with the previous works .

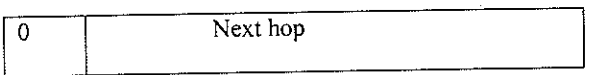
2 . Related Work

On the commercial front several companies have developed high speed lookup techniques using FPGAs and ASICs some current product claim throughputs of upto 100 million lookups per second and storage for 100 million entries[7] The most efficient lookup algorithm known from a theoretical perspective is the binary search over prefix lengths algorithm described in [8].The number of steps required by this algorithm grows logarithmically in the length of address making it particularly attractive for Ipv6 where address lengths increase to 128 bits. Waldvogel et al[14] have proposed a scheme that performs a binary search on hash tables organized by prefix length. This binary search scheme has an expected complexity of O(log W) for look ups An alternative adaptation of binary search to longest –prefix matching is developed in [2]using this adaptation , a look up in a table that has n prefix takes O(W+log n)= O(W) time. They are suited for static routing .

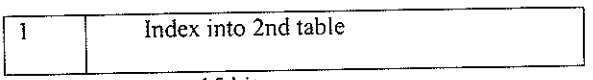
In [5]Lu and Sahni developed a data structure called BOB (binary tree on binary tree) for dynamic routing table in which the rule filters are non intersecting ranges and in which ties are broken by selecting the highest – priority rule that matches a destination address. Related structure PBOB(prefix BOB) is proposed for highest –priority prefix matching .

3. Routing Lookup Algorithm

We call the basic scheme *DIR-24-8-BASIC* — it makes use of the two tables shown in Figure 2,3 both stored in DRAM. The first table (called *TBL24*) stores all possible route prefixes that are up to, and including, 24-bits long. This table has 224 entries, addressed from 0.0.0 to 255.255.255. Each entry in *TBL24* has the format shown in Figure 1. The second table (*TBLlong*) stores all route prefixes in the routing table that are longer than 24-bits.



longest route with this 24-bit prefix < 25 bits long



longest route with this 24-bit prefix > 25 bits long

Figure 1 TBL24 entry format

Assume for example that we wish to store a prefix, *X*, in an otherwise empty routing table. If *X* is less than or equal to 24 bits long, it need only be stored in *TBL24*: the first bit of the entry is set to zero to indicate that the remaining 15 bits designate the next-hop. If, on the other hand, the prefix *X* is longer than 24 bits, then we use the entry in *TBL24* addressed by the first 24 bits of *X*. We set the first bit of the entry to one to indicate that the remaining 15-bits contain a pointer to a set of entries in *TBLlong*.

In effect, route prefixes shorter than 24-bits are expanded; e.g. the route prefix 128.23/16 will have entries associated with it in *TBL24*, ranging from the memory address 128.23.0 through 128.23.255. All 256 entries will have exactly the same contents (the next hop corresponding to the routing prefix 128.23/16). By using memory inefficiently, we can find the next hop information within one memory access. *TBLlong* contains all route prefixes that are longer than 24 bits. When a destination address

is presented to the route lookup mechanism, the following steps are taken:

1) Using the first 24-bits of the address as an index into the first table *TBL24*, we perform a single memory read, yielding 2 bytes.

2) If the first bit equals zero, then the remaining 15 bits describe the next hop(fig 2).

3) Otherwise (if the first bit equals one), we multiply the remaining 15 bits by 256, add the product to the last 8 bits of the original destination address (achieved by shifting and concatenation), and use this value as a direct index into *TBLlong*, which contains the next-hop (fig 3).

* *		* *
10.53.255	-	--
10.54.0	0	A
10.54.1	0	A
* *		* *
10.54.33	0	A
10.54.34	1	123
10.54.35	0	A
* *		* *
10.54.255	0	A
10.55.0	-	--
* *		* *

Fig2 Route lookups for TBL 24

Assume that the following routes are already in the table: 10.54/16, 10.54.34/24, 10.54.34.192/26. The first route requires entries in *TBL24* that correspond to the 24-bit prefixes 10.54.0 through 10.54.255 (except for 10.54.34). The 2nd and 3rd routes require that the second table be used (because both of them have the same first 24-bits and one of them is more than 24-bits long). So, in *TBL24*, we insert a one followed by an index (in the example, the index equals 123) into the entry corresponding to the 10.54.34 prefix. In the second table, we allocate 256 entries starting with memory location. Most of these locations are filled in with the next hop corresponding to the 10.54.34 route, but 64 of them are filled in with the next hop corresponding to the 10.54.34.192 route. As a summary lets review some of advantages

Entry tbl *	content *
123*256	B
123*256+1	B
123*256+1	B
* *	* *
123*256+191	B
123*256+192	C
123*256+193	C
* *	* *
123*256+255	C
124*256	C
* *	* *

Figure 3 Route Lookups in TBL Long

1) Although (in general) two memory accesses are required, these accesses are in separate memories, allowing the scheme to be pipelined.

2) Except for the limit on the number of distinct 24-bit-pre-fixed routes with length greater than 24 bits, this infrastructure will support an unlimited number of routes.

4) The design is well suited to hardware implementation.

5) When pipelined, packets per second can be processed with currently available 50ns DRAM. The lookup time is equal to one memory access time.

4. Nine Coded Compression Algorithm

The 9C technique is used for reducing the memory size in the entry table. This technique is based on fixed-length blocks of input prefix. Assume that K is the size of the

input blocks. The input test vectors are partitioned into groups of K bits, and encoding is performed on each K-bit block. Table I shows the proposed coding for K=8 . As shown in the second column of the table, each K-bit block is divided into K/2 two halves. There are overall nine fixed code words. For cases 1–4, each half K-bit matches all zeros or ones. Cases 5–8 show the mismatched bits, which have to be sent along with the codeword. xxxx denotes mix of 0 and 1 and perhaps don't-care x. Case 9 shows that each half is a mismatch and the entire K-bit block has to be sent along with the codeword.. In this technique, regardless of K, we use only nine unique codewords (C*i*) shown in the fifth column in Table I. Hence, we called this method nine-coded compression, or 9C for short. which can be only a codeword

Table 1 nine coded compression table

Case	Input block	Descrption	Codewrd Ci
1	0000 0000	All zeros	0
2	1111 1111	All ones	10
3	0000 1111	Left half 0 right half1	11000
4	1111 0000	Left half 1 right half0	11001
5	1111 xxxx	Left half 1 right half mismatch	11010
6	xxxx 1111	Left half mismatch, right half1	11011
7	0000 xxxx	Left half0, right half mismatch	11100
8	xxxx 0000	Left half mismatch, right half0	11101
9	xxxx xxxx	All mismatch	1111

Prefix T1: 00xx1001xx11111xxxxx0xxxxx00xx0x

Prefix T2:xxxxxx0xxx0xxxxx0x0xxx0xxxx00xxx

Total length is 32+32 =64

The compressed format

Prefix T1:00xx1001 xx11111x xxxx0xxx xx00xx0x

Prefix T2:xxxxxx0x xx0xxxxx 0x0xxx0x xxx00xxx

T1codewordsize 9+2+1+1=13

T2 codeword size 1+1+1+1=4

Total code size is 17

Fig 3 :Example of formatting the codeword

As mentioned earlier, in the 9C technique, the input test vectors are divided into K-bit blocks and each K-bit block is divided into two halves. Various cases that can arise match one of the above symbols. Various cases that can arise match one of the above symbol

5. Hardware Design Reference Model

fig 4 is a reference model for the hardware design , the left part is a chip and the right part is external DRAM memory. The Ip destination address enters the chip as a key for looking up the next hop information. The output of the chip is an index to the external DRAM where the port information can be found. The chip is an FPGA chip. the memory assess time is very low. Assuming there are 144 bits in each row the chip has less than 213 rows altogether. This means 13 bits are enough to index into any row of the memory. The ALU may access the memory system and perform some simple logic and arithmetic operation . According to different design goal the chip can be configured or programmed. In fact this reference model can be modified to be tailored to different situation. The entire program was done in Xilinx.

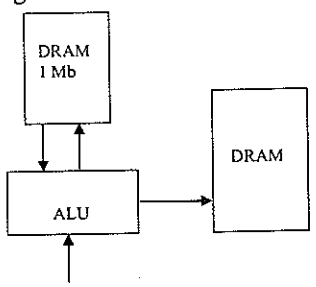


Fig 4: hardware design model

6. Comparision

We only compare with those that are similar to our approach . We also list papers that use a small amount of memory and omit those that use a large amount of memory. Performance measurements for some of the algorithms are highlighted and compared as follows

Multiway search [4] is the most similar approach to our method . How ever due to the lack of techniques to compress and optimize the data structures , larger memory requirements than ours are reported and the number of memory accesses is comparable to ours. For a routing table with 32 thousand entries 5.6 M bits memory is needed .In the worst case four memory accesses are needed.

If 256 bit memory width is used ,we can report much better results than those in the previous section .

Table 3:Number of memory accesses for IPv4 as a function of Table size And memory width

Memory width	Table size (number of prefix)				
	20000	31284	48210	144124	223112
168	6	7	8	8	8
296	5	7	7	7	7
552	5	5	6	6	6

Even with 144 bit memory width , we reported more favorable results than those from [4] reference [14] proposed an algorithm which require a worst case of log W hash lookups where W is the length of the address in bits. Hence utmost 5 lookups for Ipv4 and atmost seven hash lookups for Ipv6 are needed. However data from [4] showed that this algorithm needs 1600 K bytes for an Ipv4 routing table with roughly 40k entries. In addition to that the building of the data structure is not fast. The routing table is meant to put in off-chip memory for they need 450-470k bytes memory for routing table with 40000 entries.

Table 4: Number of memory accesses for Ipv6 as a Function of table size and memory width

Mem width	Table size(number of prefix)				
	50499	66004	128479	258784	591050
68	9	9	10	10	10
100	6	7	7	8	8
164	5	5	6	6	6

6. Conclusion

We have proposed an algorithm, which can be tailored to hardware technology. The distinguishing merit of our algorithm is that it has a very small memory requirement. With this merit, a routing table can be put up in a single chip, thus the memory latency to access the routing table can be reduced. With the pipeline implementation of our algorithm, the IP address lookup speed can only be limited by the memory access technology.

7.References

[1] A Demers, S.Keshav, and S. Shenker, *"Analysis and simulation of a fair queing algorithm"* in Proc SIGCOMM, Sep 1989, pp1-12

[2] W. Doeringer, G. Karjoth, M. Nassehi. *"Routing on Longest-Matching Prefixes". IEEE/ACM Trans. Networking*, Vol. 4, No. 1. Feb. 1996.

[3] B Lampson ,V Srinivasan and G Varghese *" IP look up using multi-way and multicolumn search"*, Proc IEEE INFOCOM,'98,1998

[4] B. Lampson, V. Srinivasan, and G. Varghese, *"IP Lookups Using Multiway and Multicolumn Search"*, IEEE/ACM Trans. Networking, vol. 7, pp. 324-334, 1999.

[5] H.Lu and S.Sahni and D.Mehta, *"Fundamentals of data structure in C++ new york"* W.H.Freeman 1995

[6] Merit Networks, Inc. See http://www.merit.edu

[7] A. McAuley, P. Francis. *"Fast Routing Table Lookup Using CAMs". Proc. IEEE INFOCOM 1993*, Vol. 3, pp 1382-1391, San Francisco, USA.

[8] P. Newman, T. Lyon, G. Minshall. *"Flow Labelled IP: A Connectionless Approach to ATM". Proc. IEEE INFOCOM 1996*, pp. 1251-1260, USA.

[9] Sibercore Technologies Inc ., *"SiberCAM Ultra-2M SCT2000"*, Product brief 2000

[10]Marcel Waldvogel, George Varghese, *"Jon Tuner and Benhard Plattner, Scalable Ip routing table lookups"*, in proceedings of ACM SIGCOMM '97 September 97,pp 25-36

[11]Mwaldvogel G.Varghese J.Tuner and plattner *"Scalable High Speed IP routing lookups"* proc ACM SIGCOMM pp 25-36 1997

[12] Y. Rekhter, T. Li. *"An Architecture for IP Address Allocation with CIDR". RFC 1518*, Sept. 1993.

[13]K. Seppanen, *"Novel IP Address Lookup Algorithmfor Inexpensive Hardware Implementation"*, WSEAS Trans. Comm, vol. 1,no. 1, pp. 76-84, 2002.

[14] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, *"Scalable High Speed IP Routing Lookups"*, Proc. ACM SIGCOMM, pp. 25-36, Sept. 1997.

[15] *"Wire-Speed IP Routing", White Paper*, Extreme Networks, 1997.

[16] Y.-S. Yeh, M. Hluchyj and A. S. Acampora, *"The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching", IEEE J. on Selected Areas in Commun.* Vol. SAC-5, No. 8, Oct. 1987, pp. 1274 - 1282.

[17] Zhongchao Yu, Jianping Wu, Ke Xu, and Mingwei Xu. A fast IP classification algorithm applying to multiple fields. In *Proceedings of IEEE ICC 2001*, 2001.

[18]. V. P. Kumar, T. V. Lakshman, and D. Stiliadis, *"Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet", IEEE Commun. Mag.*, May 1998, pp. 152 - 164.

Author's Biography

M.Suganthy was obtained her B.E degree from Thiagarajar College of Engineering Madurai in 1983 and M.E degree from PSG College of Technology, Coimbatore in 1985. She got her PhD degree in the area of Mobile and wireless communication in 2000 She had published more than 30 papers including Journals , National and International Conferences.

A.Albert Raj was born in Tamil Nadu, India on September 21 1971.He received the B.E Degree in Electronics and Communication Engineering at Madras University, and his M.E Degree in Alagappa Chettiar College of Engineering and Technology ,karikudi, TamilNadu. Currently he is doing the PhD in Anna University. He is a Asst.Professor and Head of Dept of Electronics and Instrumentation Engineering, Noorul Islam College of Engineering. Specific assignment involved with include VLSI Design and Digital System Design.. He is the author of the Book entitled "VLSI Circuit Design ." Anuradha Agencies Publishers, Chennai.