Hardware Based Routing Lookup for IPv4

Wen-Sheng Yang Jong-Jiann Shieh

Department of Computer Science and Information Engineering
Private Tatung University, Taipei, Taiwan, R.O.C.
Email: eric@amigo.ttu.edu.tw shieh@rv4.ttu.edu.tw

ABSTRACT

In this paper, consider the problem of organizing address tables for Internet routers to enable fast searching. Our proposal is to build an efficient, compact and easily searchable implementation of an IP routing table by using hardware.

One limitation of router performance is the route lookup mechanism. IP routing requires that a router perform a longest-prefix-match address lookup for each incoming datagram in order to determine the datagram's next hop.

We present a route lookup mechanism that when implemented in a pipelined fashion in hardware, can achieve one or more routes lookup every memory access. With 50ns DRAM, this corresponds to approximately 20 million packets per second; this corresponds to a rate of more than 10 Gb/s per port. We also present promotion mechanism that can be done in the early one stage of the pipeline for the forwarding table in the hardware. The advantage of the promotion mechanism is made clear from the simulation of the five-levels hierarchy routing table distribution.

1. INTRODUCTION

The growth of the Internet has lead to a situation where network capacity is becoming a scarce resource. It used to be that network capacity was mainly limited by speed of the links, but this situation is now changing. Thanks to the developments in transmission technology, with the introduction of fiber optics, the bottleneck is moving from the links to the routers. The current trend for dealing with this problem is to relieve routers from some of the burden of switching traffic, and instead use switches of different kinds, such as FDDI switches, ATM switches, and Ethernet switches. This turns out to be a more cost-effective solution, since the price for switching capacity is lower than the price for routing capacity.

One of the main limiting factors for performance in a router, compared to a switch, is often thought to be the processing of incoming packets: when an IP packet arrives at an input port of a router, the packet needs to be examined and classified, and based on the classification the packet is forwarded to an output port. The packet classification operation consists of analyzing information in the packet header (at least the destination address needs to be examined), and performing a lookup operation to obtain the information required forwarding the packet to its next hop. In principle, the same kind of classification needs to be performed by a switch, but the operation is generally thought to be more complicated for an IP packet than for an ATM cell or an Ethernet frame.

The classification operation performed by a switch can be done through a few and simple operations. This typically includes fetching a field from the header of the incoming data unit, performing some simple arithmetical-logical operations on this field, and using the result as a direct index into a table with forwarding information. Such a table can be stored in ordinary RAM, which means that in the ideal case, a switch can classify data units at the rate of one data unit per memory cycle in RAM.

On backbone routers there are very few routes with prefixes longer than 24-bits. This is verified by an examination of the MAE-EAST backbone routing tables [6]. A plot of prefix length distribution is shown in Figure 1; note the logarithmic scale on the y-axis. In this example, 99% of the prefixes are 24-bits or less.

IPv6 is still some way off; IPv4 is here to stay for the time being. Thus, a hardware scheme optimized for IPv4 routing lookups is useful today. There is a single general-purpose processor participating in routing table exchange protocols and constructing a full routing table (including protocol-specific information such as route lifetime, etc. for each route entry). The next hop entries from this routing table are down loaded by the general-purpose processor into each forwarding table, which are used to make per-packet forwarding decisions.

This thesis demonstrates that it is possible to classify IP packets at the rate of one and more packets per memory cycle. The purpose is to present two IP address lookup schemes and promotion mechanisms, which are suitable to implement in hardware. The address lookup mechanisms are for unicast IP addresses, and performs a longest prefix lookup. The data structure we use for this is a variant of a *trie* [5] with few particular levels. The promotion mechanism that can be done in the early one stage of the pipeline so that the next hop can be determined early.

1.1 Motivation

Our work is motivated by the need for faster route lookup; in particular, we are interested in fast, hardware-implementable lookup schemes. We desire a lookup mechanism that achieves four goals: The first is that the lookup procedure should be easily implementable in hardware using simple logic. The second is that each lookup takes at most one memory access time on average. The third is that if a lookup takes more than one memory access,

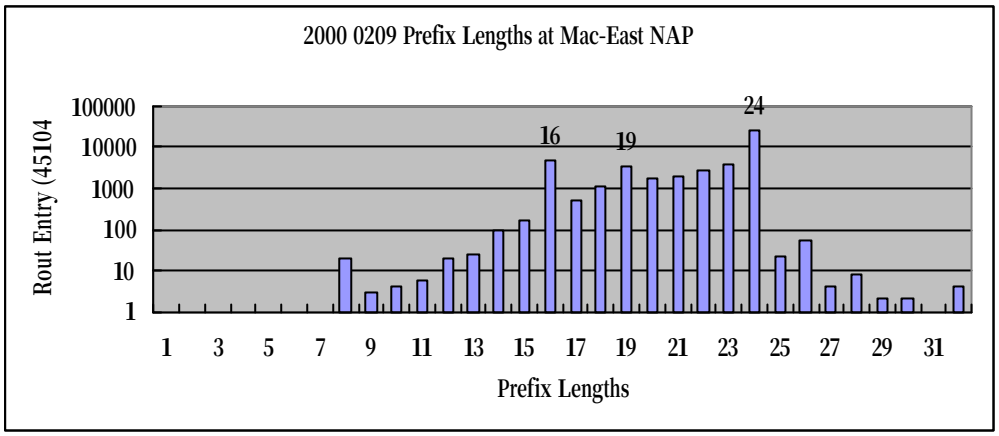

Figure 1: Prefix length distribution

then (a.) the number of accesses should be small, (b.) the number of accesses should be bounded by a small value in all cases, and (c.) the memory accesses should occur in different physical memories, enabling pipelined implementations. The fourth is the cost that is an important concern.

Both lookup scheme are based on cheap memory and standard logic, and are designed mainly with the goal to be scalable. The scalability is achieved in two ways: The first is that the lookup schemes are so cheap that it allows a distributed router design where each link interface card has its own lookup schemes. A router can then be built with any number of interface cards, since the lookup schemes put a limit only on the capacity of individual interfaces, and not on the total capacity of the router.

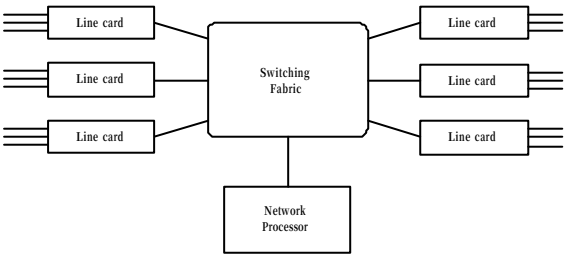

Figure 2: Router design with power on interfaces

In Figure 2, a number of network interfaces [1], and network processor are interconnected with a switching fabric. The forwarding engines use a local version of the routing table, a forwarding table, downloaded from the network processor to make their routing decisions. The GRF routers from Ascend communications, for instance, use this design. The second scalability property comes from the fact that the lookup schemes themselves are flexible. Using slow SRAM with a memory cycle time of 100 nanoseconds, it is possible to process 10 million packets per second. Assuming that an average IP packet is 1000 bits, this means that each lookup scheme can deal with 10 Gb/s. Higher capacities can easily be achieved by using faster memories. As for the promotion mechanism that can be done in the early one stage of the pipeline for the different physical memories resulting are lookup speeds increasing.

1.2 Background

Our design for performing longest prefix match is based on a tree representation of the forwording table, where the tree is searched from shorter to longer prefixes. Since the advent of the Classless InterDomain Routing (CIDR) in 1993 [16], IP routes have been identified by a <route prefix, prefix length> pair, where the prefix length is between 0 and 32 bits, inclusive, for every incoming packet, a search must be performed in the router's forwarding table to determine which next hop the packet is destined for.

1.2.1 Longest Prefix Matching

With CIDR, the search may be decomposed into two steps. First, we find the set of routes with prefixes that match the beginning of the incoming IP destination address. Then, among this set of routes, we select the one with the longest prefix. This is the route that we use to identify the next hop.

1.2.2 Classless InterDomain Routing (CIDR)

The objectives of CIDR are:

- Prolong the life of the IPv4 address space.
- Simplify the routing at the Internet's major traffic exchange points.
- Make more efficient use of the remaining IP address space.

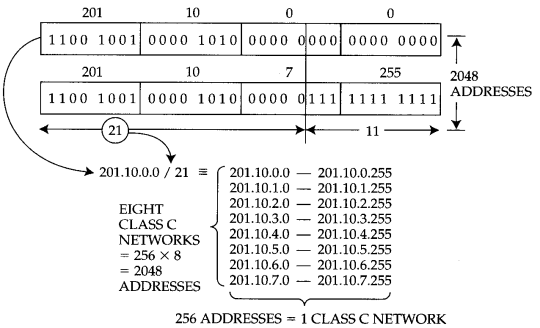

Figure 3 : CIDR example. With CIDR, a host can aggregate multiple Class C addresses to get an address space larger than that of a Class C network, but smaller than that of a Class B network. In this example, the 2048 addresses in the range 201.10.0.0 to 201.10.7.255, which belong to eight Class C networks, are aggregated to form a single CIDR class, described by the notation 201.10.0.0/21. This means that the first 21 bits of the address should be interpreted as the network number.

For example, with CIDR, a network could be allocated eight Class C networks, spanning the 2048 addresses from 201.10.0.0 to 201.10.7.255, instead of a single Class B network, with 65,536 addresses. Since the network administrator is allocated eight Class C networks, which use three bits of the Class C space, the remaining 21 bits must be the network number. The address and prefix describing the network is, therefore, 201.10. 0.0 and 21, usually written as 201.10.0.0/21 shown in Figure 3.

The concepts of CIDR are relatively simple. Instead of filling routing tables —particularly those for the routers at the core of the Internet—with entries for individual network addresses, why not refer to a whole range of contiguous network addresses with one entry? CIDR does just that, by what is known as supernetworking. CIDR also eliminates the distinctions of Class A, B, and C addresses by subnetworking all IP address space into closely fitted "chunks" of address space. For example, let's say that all of the networks in the range from 190.100.1.0 through 190.100.255.0 have been assigned to a single ISP. That ISP is a customer of one of the top-level national ISPs, which advertises the existence of those networks. To other top-level ISPs, all those networks are reachable through the national ISP.

So instead of having the national ISP, as well as all the others at the Internet NAP, maintain routing table entries for 255 separate networks, why not just has one entry that represents them all? The CIDR entry for all 255 of those networks wouldbe 190.100.0.0/16. We will explain the /16 later, but, briefly here, it's a shorthand notation to indicate a block of 256 Class C networks, not just one network.

Using routers and routing protocols that can pass around network updates along with that /nn notation, network administrators can carve IP address space into appropriately sized chunks, instead of the large, medium, and tiny Class A, B and C network sizes. ARIN (American Registry for Internet Numbers) can assign address space that way, too, instead of having to dole out huge chunks of classical address space.

1.2.3 Routing and Forwarding Tables

A router design is shown in Figure 2 [1]. A number of network interfaces, and a network processor are interconnected with a switching fabric. The forwarding engines use a local version of the routing table, a forwarding table, downloaded from the network processor to make their routing decisions.

Inbound interfaces send packet headers to the forwarding engines through the switching fabric. The forwarding engines in turn determine which outgoing interface the packet should be sent to. It uses the destination address to determine the output port for the packet and its next-hop address (it also modifies the header or meta-data). This information is sent back to the outbound interface. The only task of a forwarding engine is to process packet headers. It is not necessary to download a new forwarding table for each routing update. Routing updates can be frequent but since routing protocols need time in the order of minutes to converge, This is because routing protocols, such as RIP (Routing Information Protocol) and OSPF (Open Shortest Path First). Forwarding tables can grow a little stale and need to be updated only once every 30-60 s [13]. The network processor needs a dynamic routing table designed for fast updates and fast generation of forwarding tables. The forwarding tables, on the other hand, can be optimized for lookup speed and need not be dynamic.

2. RELATED WORK AND DISCUSSION

Recently, several groups have proposed novel data structures to reduce the complexity of longest-prefix matching lookup [1][12], which are software schemes. And there are hardware schemes also, such as [2], [5] and [9].

2.1 Software Schemes

The software scheme in [1] requires the prefix tree is complete that each node in the tree has either two or no children. Nodes with a single child must be expanded to have two children; the children added in this way are always leaves, and their next-hop information is the same as the next-hop of the closest ancestor with next-hop information, or the "undefined" next-hop if no such ancestor exists.

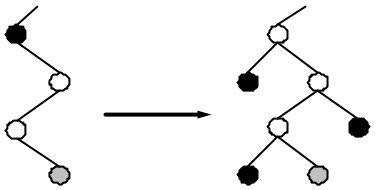

Figure 4: Expanding a prefix prefix tree to a complete tree

This procedure, illustrated in Figure 4, increases the number of nodes in the prefix tree, but allows building a small forwarding table. In [12], the author proposal is to build an efficient, compact and easily searchable implementation of an IP routing table by using an LC-trie, a trie structure with combined path and level compression. The depth of this structure increases very slowly as function of the number of entries in the table. These data structures and their accompanying algorithms are designed primarily for implementation in software, and cannot guarantee that a lookups will complete in one memory-access-time. Figure 5 shows compare table of their performance [1] & [12]. The distinct marks of the fourth column are the memory utilization and the whole total routing entry.

Techniques	Measured on	Lookup Speed	Memory Space
Level-Compressed	More Powerful SUN Sparc Ultra II Workstation 2 x 296MHZ Processors 512MB RAM	2 Million Packet per Seconds	0.8MB/40K
Small Forwarding	333MHZ Alpha 21164 Primary-Cache 8KB Secondary-Cache 96KB Tertiary-Cache 2MB	2.2 Million Packet per Seconds	0.16MB/40K

Figure 5: The compare of The Small Forwarding Scheme and The Level-Compressed Scheme

2.2 Hardware Schemes

The current techniques for performing longest matching prefix lookups in hardware, for example CAMs [2] and Tries [5]; do not seem to be able to meet the goals set forth above. CAMs are generally small, expensive and dissipate a lot of power when compared to DRAM. Tries, in general, have a worst case searching time of 32 memory accesses, leading to a wasteful 32-stage pipeline if we desire one lookup per memory access time. Furthermore, if we wish to fully pipeline the design, each layer of the trie needs to be implemented in a different physical memory. This leads to problems because the memory cannot be shared among layers; it could happen that a single layer of the trie exhausts its memory while other layers have free space.

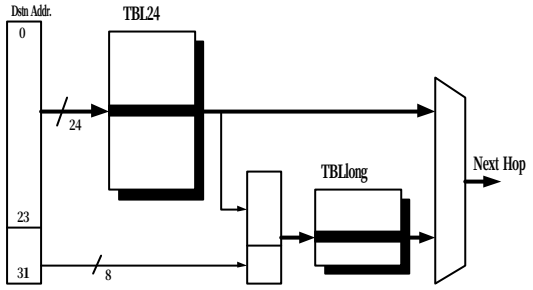

Figure 6 :DIR 24-8-BASIC architecture. The next hop result comes from either TBL24 or TBLlong

We have learned that the lookup technique outline here is a paper by P. Gupta, S. Lin, and N. McKeown, described in [9]. They call the basic scheme DIR-24-8-BASIC—it makes use of the two tables shown in Figure 6, 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 16 million entries, addressed from 0.0.0 to 255.255.255. Each entry in TBL24 has the format shown in Figure 7. The second table (TBLlong) stores all route prefixes in the routing table that are longer than 24-bits.

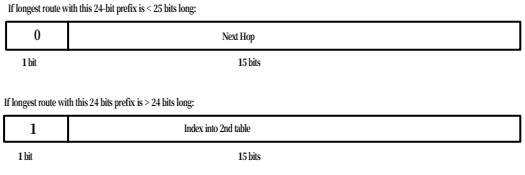

Figure 7 : TBL24 entry format

As a summary, let's review some of the pros and cons associated with the basic DIR-24-8-BASIC scheme.

Pros:

- Although (in general) two memory accesses are required, these accesses are in separate memories, allowing the scheme to be pipelined.
- Except for the limit on the number of distinct 24-bit-prefixed routes with length greater than 24 bits, this infrastructure will support an unlimited number of routes.
- The total cost of memory in this scheme is the cost of 33MB of DRAM. No exotic memory architectures are required.
- The design is well suited to hardware implementation.
- When pipelined, 20 million packets per second can be processed with currently available 50ns DRAM. The lookup time is equal to one memory access time.

Cons:

- Memory is used inefficiently
- Only one lookup valid per cycle in the multiple stages pipeline

3. PROPOSED SCHEME

In the paper, we present a route lookup mechanism that when implemented in a pipelined fashion in hardware, can achieve one or more routes lookup every memory access. We also present promotion mechanism that can be done in the early one stage of the pipeline for the forwarding table in the hardware.

3.1 The Data Structure for Forwarding Table

The address space can be though of as a tree, where the nodes represent prefixes. Each level in the tree represents a specific prefix length, which is the same for all nodes on that level. In our schemes we limit the tree to a few levels. Figure 8 depicts a prefix tree with 6-bit addresses using three prefix lengths (2 and 4). Prefixes with other lengths than the ones used in the tree have to be expanded into several longer prefixes. For example, for the tree in Figure 8, a prefix with length 3 has to be expanded into two prefixes with length 4, so the binary prefix 010/3 would have to be expanded into 0100/4 and 0101/4. This is due to prefix expansion. This will generate more nodes when there are few levels in the tree.

The tree has three types of nodes: valid, index and invalid nodes. A valid node represents an entry in the forwarding table. An index node corresponds to a prefix that matches an entry in the forwarding table, but is shorter than that entry ("prefix of a prefix"). An invalid node represents a prefix that does not appear in the forwarding table.

To simplify the processing of the tree, we introduce two restrictions: First, all possible children of an index node must be present in the tree (this is called a prefix group). The unused prefixes in a prefix group are marked as invalid. The second restriction is that we do not allow a node to be both valid and index at the same time. So if there is a prefix in the routing table which is both a route in itself and a prefix of other routes, it will appear as several nodes in the tree: The prefix itself is inserted as an index node and the prefix is expanded into a prefix group where all entries are valid.

To find a matching route, the tree is searched from the shortest prefix until the first valid or invalid node that matches the route is encountered. In this way, the longest matching entry is guaranteed to be found. For example, the following is the procedure to find the address 0101 in a forwarding table represented by the tree in Figure 8. First the shortest prefix 01/2 is looked up. The matching entry is an index, resulting in a second lookup, 0101/4. A last lookup 0101 is then performed, resulting in a valid entry. This entry points into a forwarding table where the forwarding information is stored. If only simple forwarding is needed for the entry, an output port

identifier could be stored instead of the pointer.

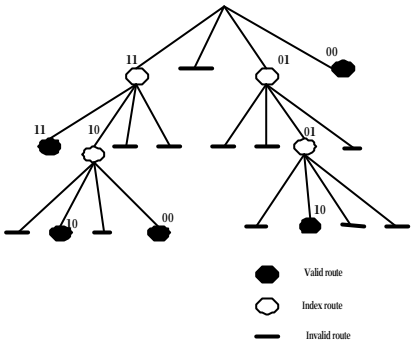

Figure 8: Example routes in a prefix tree

3.2 Five-levels Hierarchy Searching Structure

The simplest addressing structure is a flat address space, where we simply assign each destination a unique address chosen anywhere from the address space. This is seen, for example, in Ethernet addresses and for local connection identifiers. This method has the advantage of simplicity; but is limited to small networks, where routing table size is manageable.

It is inefficient to store a prefix length in one memory blank that is comparatively considerable due to the pipeline system. Therefore, every memory blank need to be independent and that means 32 individual memories blank are required for the worst case. How many individual memory blanks is required? On backbone routers there are very few routes with prefixes longer than 24-bits that shown in Figure 1. There are three prefix lengths (16, 19 and 24) with majority of routing entry in Figure 1. When we would like to construct a prefix tree as Figure 8, it is important to avoid expansion that can be fulfilled by using three individual memory blanks for the prefix lengths mentioned above (16, 19 and 24). Adds the prefix length of 8 and the prefix length of 32, there are five levels of memory tables. The prefix tree is partitioned into five levels, each level mapped to a memory table, as shown figure 10. We called this a 5-levels hierarchy searching structure. An entry in a table either represents a valid route, (contains an index points to a table defining the next hop), or represents an index of a route, (to the next level).

On a lookup, the IP address is divided into sub fields, one for each level in the prefix tree. These sub fields are used as indices into the tables at the corresponding levels in the trie. So the first sub field from the IP address is used as an index into the first level table. This gives an offset to a table at level two, indexed by the second sub field from the IP address; the rest can be done in the same way.

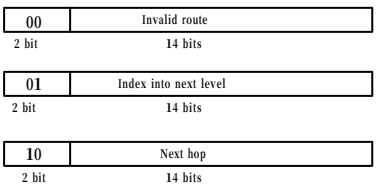

Figure 11 : 5-levels hierarchy searching structure entry format

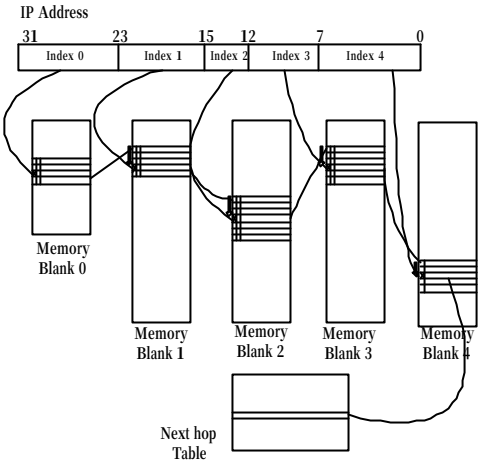

Figure 10 : 5- Levels hierarchy searching structure

Besides the pointer field, each table entry also includes two bits indicating whether the node is index or valid or invalid, so there are 16 bits in total in each entry. In contrast with other tables, each entry of the fifth memory only has a bit indicating the node whether is valid or invalid; therefore, there are 8 bits in total. Figure 11 shows the entry formats of the memory table. The memory consumption depends on the size of the table entries. The pointer with 14 bits can yield a maximum of the 16k next hops in the forwarding table; each entry occupies 16 bits.

We can classify routing lookup engines in two schemes according to the times of valid lookup per memory cycle. One is called single output scheme, which has one valid lookup per memory cycle. The other is called multiple outputs scheme, which has one or more valid lookup per memory cycle. The performance of the two designs are mainly limited by the speed of the memory, hence the fast lookup is correspondent to the fast DRAM.

3.3 Promotion Mechanism

We observe the distribution of forwarding tables that has two features in Figure 16. One is that there is small number of prefixes longer than 24-bits. The other is that the prefixes are not distributed evenly. The latter feature is the motivation of promotion mechanism comes from. The not evenly distributed feature can be explained easily by the data structure of tree. The father's node has fixed number of child's nodes in the data structure of tree. The fixed number of child's nodes are called group. When there is one only child used in the group, it is defined as the not even distributed condition in Figure 12, which are the same as the condition of the 5-levels hierarchy searching structure, which is not even distributed in the certain memory level. By examination the backbone collected data, there are 19 percent of the groups, whose condition is not evenly distributed in the 4th memory blank.

In the paper, we record the offset and next hop fields of the only one entry (child node) of the group into the entry of previous memory blank (father node) as shown in Figure 13. It will decrease one memory access cycle when lookup the only one entry of the group in the 4th memory blank.

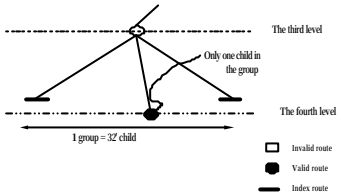

Figure 12 : Not evenly distribution in the certain memory level

We implement the five memories blank by pipeline. The lookup process can be ended in an earlier stage with the aid of promotion mechanism.

	Index	Next-next hop	Offset
2 bit 11	14 bits	11 bits	5 bits

Figure 13 : 5-levels hierarchy searching structure entry format with promotion mechanism

4. IMPLEMENTATION

Figure 14,15 shows very simple hardware designs of two 5-levels lookup engines. One is of single output scheme, as shown in Figure 14; the other one is the multiple outputs scheme, as shown in Figure 15.

The tables are stored in different memory blanks —that is to say each level stored in one blank. This results in a pipelined design, with one stage per level. The lookup is completed when a stage either has the valid bit set, or has both the valid bit and the index bit cleared. When a match is found (i.e., a stage has the valid bit set), the resulting next hop pointer flows through the following stages, without being changed.

When packet comes in the 5-levels single output lookup, the IP address will be divided into five subfieldes. The IP address of IP version 4 has 32 bits. The IP address of the five subfieldes are 31 ~ 24 bits, 23 ~ 16 bits, 15 ~ 13 bits, 12 ~ 8 bits and 7 ~ 0 bits.

Figure 14 illustrated five pipeline address registers in strips on the top of the drawing, there is a stage (level 3) omitted due to the limited figure space. The 31 ~ 24 bits of the first pipeline address register are used as an index of the memory blank of level 1. There are a ptr in every entry of every memory blank. The ptr, that it' s the content of one entry of the memory blank of level 1, which is indexed by the 31 ~ 24 bits of address. The ptr used as segment and the 23 ~ 16 bits of the second IP address pipeline register is an offset. This combination of the segment and offset is used as indices into the memory blank of level 2. The outcomes of memory are latched into the pipeline register for the next stage to use as shown in the Figure. All other levels use the same principle.

The ptr address of the front memory blank with latch connects to the ' 0' end of the input of the multiplex and other ptr address of the present memory blank connect to the other end of the input of the multiplex shown as in Figure 14. Therefore, the out of multiplex will by way of the ptr address of the front memory blank with latch when the valid of the front memory blank is set. Besides a ptr, there are valid and index fields in the content of every entry of every memory blank. The two fields mentioned above, are used as control signals in the 5-levels single output lookup engine. The valid and index fields cannot be set as ' 1' in the same entry of every memory blank. In addition, we used the longest prefix-matching scheme. Hence, if the valid field of the entry of the front memory blank is set, the ptr will be selected and outputted from the multiplex in the corresponding memory level. In other words, the valid control signal outputs Hi from OR gate, which indicates a match is found and the next hop index is outputted.

As soon as packet comes into the 5-levels multiple outputs lookup engine, the packet head is stored into IP address buffer. There is a pointer, which is used as indices into the memory blank of level 1 by the 31 ~ 24 bits of the IP address buffer, in the IP address buffer. There are five stages in the pipeline for the 5-levels multiple outputs lookup engine. These are indicated by the five strips on the Figure 15, which are named as the IP address' s pipeline registers. The content of the first IP address' s pipeline register (the left) is passed from the IP address buffer, when clock come in. At the same time, the content of the second IP address' s pipeline register will be also passed from the first IP address' s pipeline register' s content. This movement is simultaneously from left to right.

In general, the lookup action of the IP address data moves from left to right through each IP address' s pipeline register of the five stages. There is a dropped bit in every IP address pipeline registers. The dropped bit is used to indicate whether the IP address' s next hop index is found. If this bit is set to Hi, the next hop index is correctly found and the IP address data is drained from the pipeline, as shown in Figure 15' s lower pipeline register.

Each target IP address is divided by 5 fields, according to bit position 31~24, 23~16, 15~13, 12~8 and 7~0. Each field is used as an offset into the corresponding hierarchy memory level. The base of this segment is the latched pointer (or index) of previous level' s output, shown in Figure 11.

In order to accomplish the promotion mechanism, the memory' s content of level 3 is set differently as shown in Figure 13. If bit 12~8 of the IP pipeline register is matched with the 3th level' s output' s offset field, then the promotion mechanism is activated. The next hop is found and outputted.

There is an OR gate in the third stage of pipeline. Which is used in the case that the valid bit of the latcher is Lo and the dropped bit of IP address' s pipeline register is Hi. In other words, the input end of the D type FIFO must be set as Hi. When next read clock come in, the latcher will be disabled in the stage of the pipeline. This will make sure the promotion mechanism is finished.

5. EXPERIMENTAL RESULTS

The behavior of memory distribution for 5-levels hierarchy searching structure has been simulated. The experimental routing backbone is the Mac East [6]. The routing table data of the backbone files is downloaded from the Merit Networks.

Figure 16 shows the simulated results. The sixth column, "16len' level one child," explains the group percentage of the group with only one entry in the memory bank of 16 prefix length. The "(88) and (89)" are the total groups of this hind. There are the candidates for promotion. But as the data shown, there are only 88 (or 89) group for the second level and 4163 (or 4048) group for the third level, we choice the fourth level for promotion.

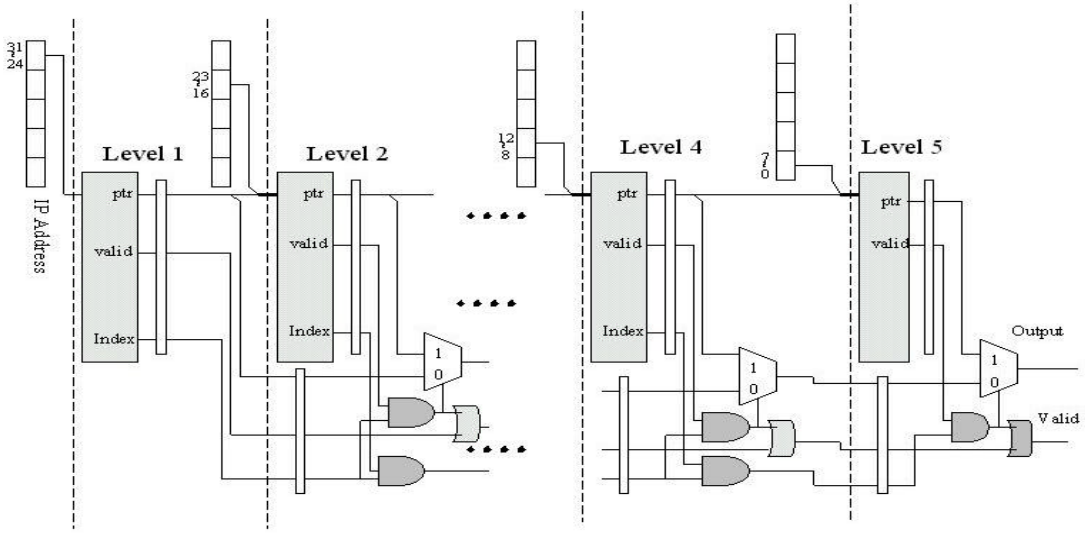

Figure 14: 5-levels single output lookup engine

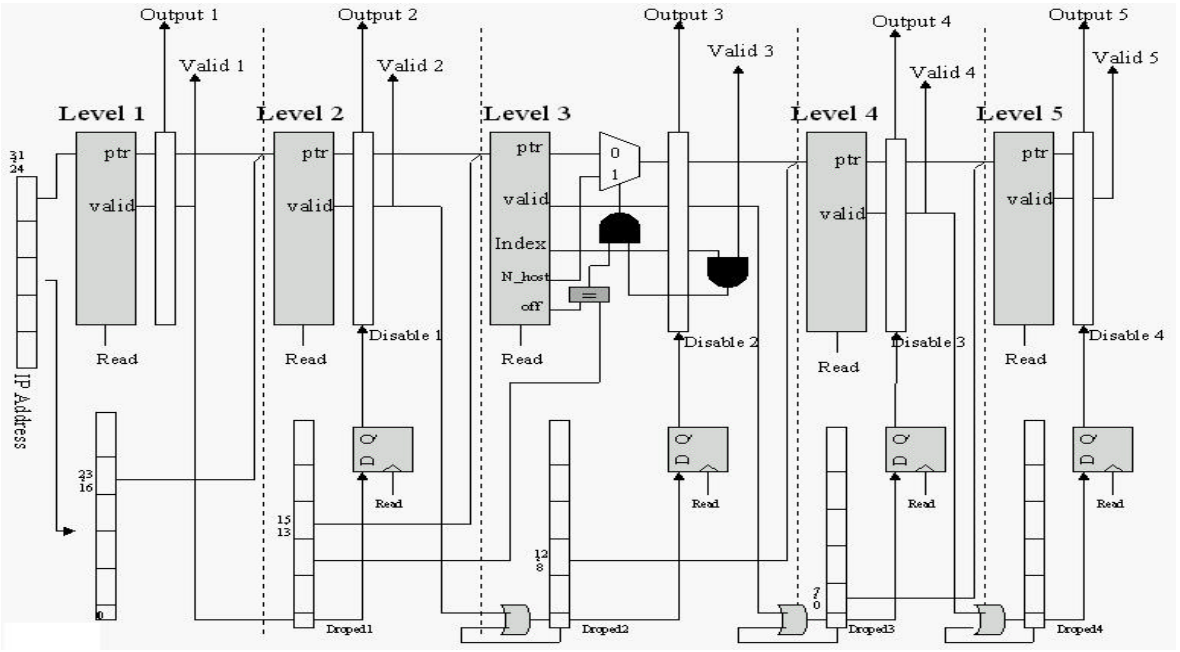

Figure 15: 5-levels multiple outputs lookup engine

The memory consumption of five-levels searching structure lookup engine depends on the size of the table entries. Every entry of the above consists of one valid bit and index bit plus a pointer field, which are 16 bits in total. There is a cleared valid bit indicates an invalid route, which is 8 bits in total, on the last level entry. The usage of the memory in the case of the large routing table such as the table from MAE-EAST (routing backbone) is less than 900KB (0.89MB) size. However, the total size of memory blank of the five level is approximately 14MB and therefore, the utilization of memory can be inefficient.

6. CONCLUSIONS AND FUTURE WORK

We have demonstrated that hardware fast IP address lookup engines can be built from a small amount of inexpensive, ready-made components such as slow static or dynamic RAM (SRAM, DRAM) and programmable logic devices (PLDs). We have presented two designs: One for one output per memory cycle -- single output lookup engine and one for one or more outputs per memory cycle --multiple outputs lookup engine, and one mechanism: can be done in the early one stage of the pipeline.

Site	Date	Year	Routing entries	Size (MB)	16len'level one child	19len'level one child	24len'level one child	32len'level one child
Mac East	MAR 10	2000	56442	0.89	11% (88)	18% (4163)	18.3%(11537)	5% (80)
Mac East	MAR 20	2000	51640	0.86	12% (89)	18.3%(4084)	18.8%(11100)	0%

Figure 16 : Experimental data in the Mac East

Both designs are pipelined and can perform lookups at the rate of one or more packets per memory cycle. Depending on the memory technology used, this corresponds to rates up to 50 million packets per second. How will the memory be utilized efficiently? From the experiment, 14MB are required but only 900KB are used. A hash function may be used to map the 14MB to 900KB memory. This is still another worth to be done. The improvement can be used with hash function, which is to contract memory table and to achieve good memory utilization. As to the selection of the hash function and the process of the contention with routing lookup table, which are the critical issues of future work.

7. REFERENCES

[1] A. Brodnik, S. Carlsson, M. Degermark, S. Pink. "Small Forwarding Tables for Fast Routing Lookups." *Proc. ACM SIGCOMM 1997*, PP. 3-14, Cannes, France.

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

[3] B. Dutcher. Managing IP Addresses, How to Number Your Network for Growth and Change. *Wiley Computer Publishing, John Wiley & Sons, Inc* 2000.

[4] C. Labovitz, G. R. Malan, F. Jahanian. " Internet Routing Instability." *Proc. ACM SIGCOMM 1997*, pp. 115-126, Cannes, France.

[5] E. Fredlkin, "Trie Memory," *Communications of the ACM*, vol. 3, no. 9, pp. 490-499, Sept. 1960

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

[7] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner. Scalable high speed IP routing lookups. *ACM Computer Communication Review*, 27(4): 25-36, October 1997.

[8] M. Waldvogel, G. Varghese, J. Turner, B. Plattner. "Scalable High-speed IP Routing Lookups." *Proc. ACM SIGCOMM 1997*, PP. 25-36, Cannes, France.

[9] P. Gupta, S. Lin, and N. Mckeown. Routing Lookup in Hardware at Memory Access Speeds, *INFOCOM'98, 17th Annual Joint Conference of the IEEE Computer and Communications Societies.*

[10] P. Newman, G. Minshall, T. Lyon, and L. Huston. IP switching and gigabit routers. *IEEE Communications Magazine*, 35(1): 64-69, January 1997.

[11] R. J. Walsh and C. M. Ozveren. The gigaswitch control processor. *IEEE Network*, 9(1): 36-43, January/February 1995

[12] S. Nilsson, G. Karlsson. "Fast address lookup for Internet routers." In *Proc. IFIP 4th International Conference on Broadband Communications*, pp. 11-22, 1998

[13] Stanford University Workshop on Fast Routing and witching, December 1996. http://tiny-tera.stanford.edu/Workshop_Dec96

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

[15] W. Doeringer, G. Karjoth, and M. Nassehi. Routing on longest-matching prefixes. *IEEE/ACM Transactions on Networking*, 4(1): 86-97, February 1996.

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

[17] Gosling, J and McGilton, H. *The Java Language Environment: A White Paper*, JAVASOFT, 1996