

# High-Speed IP Routing With Binary Decision Diagrams Based Hardware Address Lookup Engine

Rama Sangireddy, *Student Member, IEEE*, and Arun K. Somani, *Fellow, IEEE*

**Abstract**—With a rapid increase in the data transmission link rates and an immense continuous growth in the Internet traffic, the demand for routers that perform Internet protocol packet forwarding at high speed and throughput is ever increasing. The key issue in the router performance is the IP address lookup mechanism based on the longest prefix matching scheme. Earlier work on fast Internet Protocol Version 4 (IPv4) routing table lookup includes, software mechanisms based on tree traversal or binary search methods, and hardware schemes based on content addressable memory (CAM), memory lookups and the CPU caching. These schemes depend on the memory access technology which limits their performance. The paper presents a binary decision diagrams (BDDs) based optimized combinational logic for an efficient implementation of fast address lookup scheme in reconfigurable hardware. The results show that the BDD hardware engine gives a throughput of up to 175.7 million lookups per second (Ml/s) for a large AADS routing table with 33 796 prefixes, a throughput of up to 168.6 Ml/s for an MAE–West routing table with 29 487 prefixes, and a throughput of up to 229.3 Ml/s for the Pacbell routing table with 6822 prefixes. Besides the performance of the scheme, routing table update and the scalability to Internet Protocol Version 6 (IPv6) issues are discussed.<sup>1</sup>

**Index Terms**—Address lookups, binary decision diagrams (BDDs), IP routing, longest prefix matching.

## I. INTRODUCTION

THE effective handling of the tremendous amount of Internet traffic and its continuous doubling every few months depends on the efficacy of the routers. The key issue in router performance is the Internet protocol (IP) address lookup mechanism used for the ferrying of the large number of incoming communication packets to respective outgoing links. The router uses the destination IP address encoded in the incoming packet to lookup the next-hop router to which the packet has to be forwarded. Since the introduction of classless interdomain routing (CIDR) in 1993, the IP address lookup mechanism has been designed based on the longest prefix matching (LPM) algorithm. The problem involves two steps, first to search the routing database to obtain the longest matching prefix from all the possible prefixes that match the particular destination IP address, and second to retrieve the next-hop port for the longest matched

prefix. With the advent of optical medium for data transmission, the link rates have rapidly increased from 10 Mb/s Ethernet to 40 Gb/s OC768c and there is every possibility of the line rates increasing well beyond. The primary concern in the design of next-generation routers is to obtain maximum possible packet throughput to meet the demand from the high-speed transmission links. The continuous increase in the number of users on the Internet causes the creation of some explicit routes for certain users and constrains the router from aggregating the routing table effectively. This results in the expansion of routing tables and, thus, the search space of prefixes against which the destination address of each packet needs to be matched. Further, when the Internet Protocol Version 6 (IPv6) routing protocol is introduced where the address length is 128 bits, the problem of routing millions of communication packets every second, based on longest prefix matching, becomes a labyrinth. In these circumstances, where IP routing tables are expanding in both the dimensions, i.e., address length and number of prefixes, the routing mechanisms developed should be capable of providing the throughput demand from high-speed transmission links.

In this paper, we propose a reconfigurable hardware solution, using the well received concept of binary decision diagrams (BDDs), that provides a high-speed IP address lookup and enables a data throughput of 200 Gb/s (average packet size of 1000 bits) in the current day large routers [1]. BDDs are one of the biggest breakthroughs in CAD in the last decade. BDDs are a canonical and efficient way to represent and manipulate Boolean functions and have been successfully used in numerous CAD applications. Although the basic idea has been around for more than 30 years [2], it was Bryant who described a canonical BDD representation [3] and efficient implementation algorithms [4].

The rest of the paper is organized as follows. Section II presents an overview of the longest prefix matching problem. In Section III, we review the related work done followed by a brief overview on BDDs and the motivation for the proposed scheme. Section IV gives the details of the proposed scheme and the implementation issues. In Section V, we present the results obtained from the implementation and analyze the performance of the scheme. We also discuss in detail, the routing table update scheme and the scalability of the scheme to IPv6. Finally, Section VI concludes the discussion.

## II. LONGEST PREFIX MATCHING

The routing of the communication packets in the IP domain is done on the next-hop basis, i.e., the router takes the responsibility of sending any incoming packet till the next hop only.

Manuscript received December 18, 2002; revised January 16, 2003. This work was supported in part by the National Science Foundation under Grant ANI9973102 and Grant CCR9900601.

The authors are with the Dependable Computing and Networking Laboratory Department of Electrical and Computer Engineering Iowa State University, Ames, IA 50011 USA (email: sangired@iastate.edu; arun@iastate.edu).

Digital Object Identifier 10.1109/JSAC.2003.810516

<sup>1</sup>This paper is based on earlier work presented at IEEE International Conference on Computer Communications and Networks, ICCCN2001, as referred in [32].

TABLE I  
A SAMPLE ROUTING TABLE

<i>Prefix</i>	<i>length</i>	<i>NHP</i>
*	0	0
0*	1	1
01*	2	3
10*	2	2
001*	3	1
101*	3	2

Consequently, the packet reaches its final destination in multiple hops. The next hop for a packet is determined by the router based on its destination IP address. Each router has a database, in the form of a routing table, of the prefixes of varying length and the corresponding next-hop port (NHP) for each prefix. A typical routing table is shown in Table I.

The length of the prefixes can vary from 0 to 32 bits. For an incoming packet, its destination address is compared with all the current prefixes in the routing table and the NHP associated with the longest matching prefix is determined to be the output port for the packet. For an example shown in Fig. 1, a destination IP address 129.186.200.205 matches three prefixes 129/8 (prefix/length), 129.186/16 and 129.186.192/20 in which case, the longest matched prefix 129.186.192/20 is considered to be the best match and the packet is routed to the output port associated with that particular prefix. In other words, routing based on the longest prefix matching is equivalent to routing the packet to the nearest possible IP address. If none of the prefixes match with the destination IP address, the packet is sent to a default port, which is associated with a prefix of length zero.

The metrics taken into consideration, in general, while designing the IP lookup algorithms are *preprocessing time*, *storage requirements*, *lookup rate* and *update time*. Lookup rate is the most significant parameter that needs to be addressed. With the latest advancements in the network technology, the communication speed is leaping from Ethernet of 10 Mb/s to fiber distributed-data interface (FDDI) of 100 Mb/s to gigabit Ethernet. With the OC192c Line (Line-rate 10 Gb/s), 31.25 million packets (average size of 40 Bytes) have to be processed each second, while for the OC768c (Line-rate 40 Gb/s), the processing rate required is 125 million packets per second. The data throughput rates of various transmission links and the corresponding time budget for packet processing in a network processor is shown in Table II. It is significant to note that, apart from the lookup and forwarding operation the packet processing in a network processor includes various other functions like *protocol recognition and classification*, *segmentation assembly and reassembly (SAR)*, *queueing and access control*, and *quality-of-service (QoS)*. The time budget shown for packet processing includes the execution of all these functions, and the lookup operation is required to consume a portion of that budget. Hence, the significance of designing a mechanism for high-speed IP address lookups cannot be overemphasized.

The large IPv4 routing tables known today typically contain around 50 000 prefixes and a large IPv6 routing table is expected to contain around 500 000 prefixes. Consequently, the need for enormous amount of data processing at phenomenal speeds, based on longest prefix matching in a large database of prefixes, makes the problem more complicated. Further, the

IP address, which is 32 bits long in Internet Protocol Version 4 (IPv4), would be 128 bits long, when IPv6 is introduced, making the problem of IP forwarding even more complex.

### III. RELATED RESEARCH

The IP address lookup schemes introduced so far can be broadly classified into two categories, viz., *software* and *hardware* approaches [5]. In amelioration to the classical binary trie traversal approach, several software solutions have been proposed. One of the first was the prefix matching algorithm using *path-compressed tries* [6] based on the practical algorithm to retrieve information coded in alphanumeric (PATRICIA) trie introduced in 1968 [7]. The other schemes subsequently proposed include the various trie based approaches [8]–[10] and other binary search methods like *binary search on prefix lengths* [11], [12] and *multiway and multicolumn search* [13]. Besides, other schemes based on *prefix expansion* [14], *string matching* [15] and *level compression tries* [16] have been proposed. The software address lookup schemes mostly are based on the tree traversal approach and, hence, perform according to the computing environment used for the implementation of the algorithm. The key elements that play a pivotal role in the performance of the software mechanism are the processor speed and the memory characteristics (capacity and access time) of the computing environment in which the algorithm is implemented. The best known algorithm for IP address lookups is the *binary search on prefix lengths* with the complexity of the lookup operation being logarithmic in the prefix length ( $W$ ), i.e.,  $\mathcal{O}(\log W)$ , independent of the number of prefixes ( $N$ ). Even in such case, the lookup operation involves five memory accesses in the worst case for IPv4 and, thus, is limited by the capacity of the computing environment. The scheme with  $\mathcal{O}(\log W)$  lookup complexity gives a throughput up to 10 million lookups per second (ML/s), when implemented in a computing environment consisting of a Pentium-Pro-based computer with a clock speed of 200 MHz and a 512 kB L2 cache memory [5]. The throughput, even when scaled for faster processors and larger memory capacity, will not be able to meet the current day requirements in packet processing rates.

Apart from the above software schemes, attempts have been made to design hardware mechanisms for prefix matching to enable high-speed routing. Various hardware schemes like content addressable memories (CAMs) [17], memory lookup-based schemes [18]–[20], CPU caching [21], and circuit logic implementation in field programmable gate arrays (FPGAs) [22] have been proposed. Recently, Pao *et al.* proposed a hardware architecture [23] implemented with the partition of binary trie into multiple levels, and Taylor *et al.* proposed a reconfigurable device based fast Internet protocol lookup (FIPL) engine [24] for high-speed routing.

CAM can search all of its entries in parallel given a destination IP address. The scheme based on CAMs uses a separate CAM for each possible prefix length and, hence, will require 32 CAMs for IPv4 and 128 CAMs for IPv6 resulting in an expensive solution. Besides, CAM might not be able to keep pace with the fast developing high-speed networks as it depends on and is limited by the IC process technology. Memory lookup

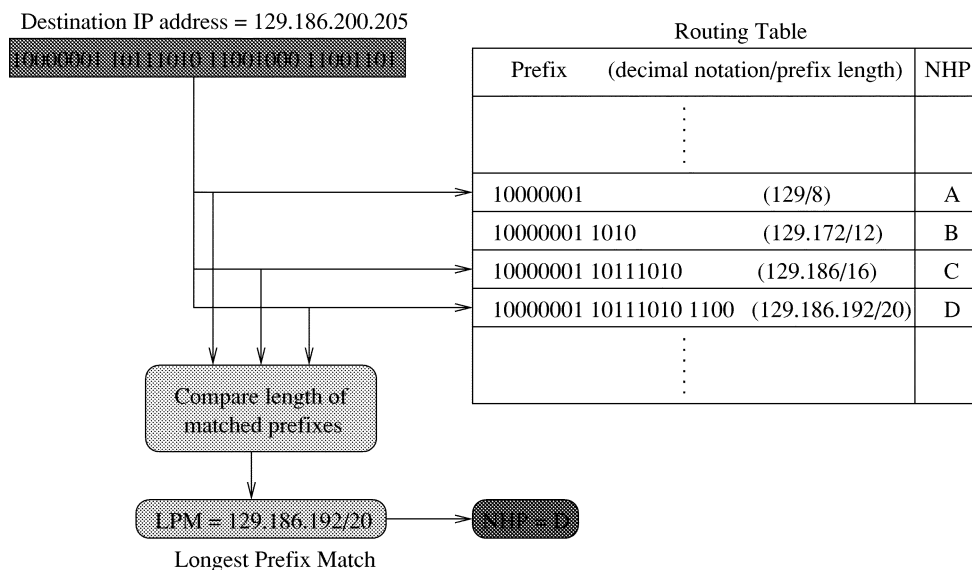
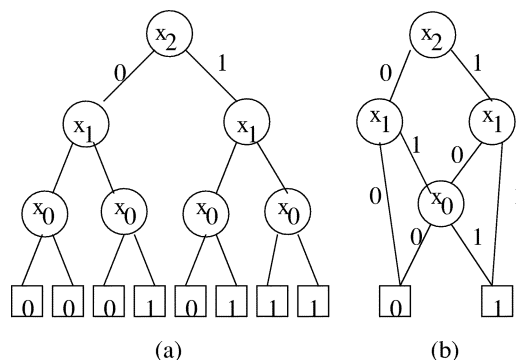


Fig. 1. Packet routing based on longest prefix matching mechanism.

TABLE II  
 DATA THROUGHPUT AND PACKET PROCESSING TIME BUDGETS  
 FOR ATM OVER SONET

Media	link rate	packets/sec (Million)	time per packet (ns)
OC3	~ 150 Mbps	~ 0.491	~ 2034.5
OC12	625 Mbps	~ 2	~ 488.2
OC48	2.5 Gbps	~ 8.38	~ 119.2
OC192	10 Gbps	~ 33.5	~ 29.8
OC768	40 Gbps	~ 134.2	~ 7.45

Fig. 2. Function  $f = x_0x_1 + x_1x_2 + x_2x_0$  represented as (a) binary decision tree and (b) BDD.

schemes are based on SRAM indirect indexing and, hence, require an additional ASIC to incorporate with the memory. The basic scheme in [18] uses a two-level multibit trie with a fixed stride of 24 bits and 8 bits for the first and second level, respectively. This scheme is developed based on the important observation that in a typical backbone router, most of the prefixes are of length 24 bits or less. Thus, a prefix expansion methodology is used wherein all the prefixes with length less than 24 bits are expanded accordingly. The memory access speed might not be able to cope up with the advent of new optical link rates and, hence, limits its performance. The architecture with hardware indexing implementation [23] of binary trie promises a high throughput, but requires a large memory for the implementation. Consequently, the practicality of the implementation of the scheme for the next-generation Internet routing with IPv6 protocol remains to be seen. The FIPL engine [24] gives an average throughput of 10 MI/s using eight FIPL engines for routing in the MAE–West router with 16 564 prefixes. Besides, the scheme does not discuss its scalability to the future trends in Internet routing. In this paper, we have shown a higher throughput of up to 168.6 MI/s for MAE–West router with a larger number of prefixes (29 487), with an added advantage that our scheme is easily scalable for the rapidly expanding routing tables.

In the past, caching has not worked well in backbone routers because of the need to cache full addresses. This potentially dilutes the cache with hundreds of addresses that map to the same prefix. Besides, typical backbone routers may expect to

have hundreds of thousands of flows to different addresses. The Wilder study [25] reports up to 240 000 concurrent flows with less than 20 packets per flow. Short web transfers are a likely reason for this behavior. Some studies have shown cache hit ratios of around 50%–70% [26]. Thus caching can help, but does not avoid the need for fast lookups. Most important, the above schemes become impractical at the advent of IPv6 due to the requirement of larger storage capacity.

#### A. BDDs

As is well-known, a Boolean function  $f : B^n \rightarrow B$  can be represented by a BDD, a directed acyclic graph obtained by applying an ordering constraint over the input variables and reduction operations on a binary decision tree, as proposed by Bryant [3]. For example, the binary decision tree and the diagram for the function  $f = x_0x_1 + x_1x_2 + x_2x_0$  are as shown in Fig. 2.

Furthermore, the complexity of the BDD is dependent on its size, measured in the number of nodes. Hence, since a long time, one of the main research focuses has been to reduce the number of nodes in BDD representation. Reduction operations consist in eliminating redundant nodes from the binary tree. A node can be eliminated if: 1) both the child nodes are equivalent, which means that the binary logic extracted from both the nodes leads

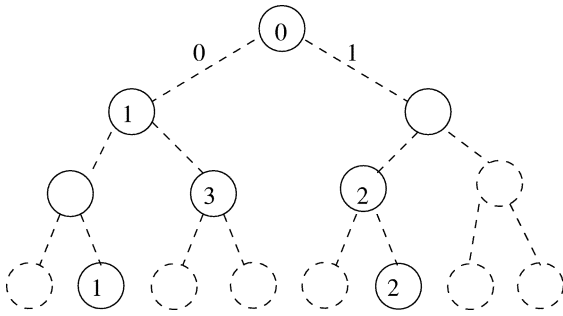


Fig. 3. Binary decision tree for the sample routing table.

to a same output and 2) there exists another node at the same level in the decision tree and with equivalent high and low child nodes, respectively.

#### IV. BDD BASED IP ADDRESS LOOKUPS

##### A. Motivation

The proposed scheme is motivated from two observations, first being that even at the largest Network Access Point, the number of NHPs is generally not greater than 256. Hence, a NHP associated with any prefix in the routing table can be encoded using a 8-bit binary code. For example, any NHP in the MAE-East [1] routing table can be safely represented by a 6-bit binary code. Every bit of the output port can be computed by a combinational logic circuit whose optimal minimization is obtained with the help of BDDs. The second observation is that the number of *effective nodes*, defined as the minimal number of nodes required to construct a binary decision tree in order to cover all the prefixes in the routing table, is significantly smaller as compared with the upper bound on the theoretically required number of nodes. It is shown in the following section that, for the 32-bit IP address with the biggest available routing table of MAE-East [1], the number of redundant nodes is more than 99.99%. Thus constructing the binary decision tree with a fewer nodes and without any redundant nodes makes it very attractive for the application of BDDs to optimize the logic. Besides, it is shown in the next sections that, while the upper bound on nodes increases exponentially with the IP address size, the number of *effective nodes* do not, making it an advantageous fact in view of the future implementation of IPv6 with the 128-bit IP address.

##### B. Details of the Scheme

For further understanding, consider the routing table given in Table I. The binary decision tree representation for the routing table is shown in Fig. 3. A node is assigned with the associated NHP if the path taken till that node from the root node forms a valid prefix. Note that the root node is assigned with a default output port (in this case zero) as the length of the prefix at that node is zero. However, as mentioned earlier, the partial construction of the binary decision tree is sufficient to cover all the prefixes in the routing table resulting in only eight *effective nodes*. The redundant nodes, which can be conveniently ignored in the binary decision tree representation, are shown in dotted lines.

Now, the number of distinct next-hop ports in this case being four, each NHP is encoded with a 2-bit binary code, NHP<sub>1</sub> and

NHP<sub>0</sub> being the most significant (MSB) and the least significant (LSB) bits, respectively. When the ports are identified with the binary code, the binary decision tree representations for the NHP<sub>0</sub> and NHP<sub>1</sub> bits are as shown in Fig. 4. It can be observed that a further reduction in the number of *effective nodes* is obtained, the process of which is explained in detail in the subsequent subsection. Note that any effective node without an output bit assigned to it, would inherit the output of its parent node.

For the sake of convenience, let us assume that the n-bit IP address is represented by the binary variables  $x_{n-1}, x_{n-2}, \dots, x_0$  where  $x_{n-1}$  represents the MSB of the IP address. Now, applying the BDD algorithm on the binary decision trees of each bit of the output port, the BDDs for the functions are obtained to be as shown in Fig. 5.

##### C. Reducing Effective Nodes

When the output port is assigned to each of the node on the binary decision tree, with the further analysis, it is observed that the binary encoding of the output ports has given a further scope for the reduction in the number of nodes. For example, consider a situation where two leaf nodes, with a common parent, are assigned with output ports of 3 and 11, respectively. Suppose the parent node is assigned with an output port of two. When the next-hop port is encoded with a 4-bit binary code, it can be observed, as shown in Fig. 6, that a child node with the same output bit as its parent becomes redundant. The redundant nodes are shown in dotted lines in the figure.

With the above procedure, it is shown that in each of the output bit representation, for the biggest available routing table of MAE-East [1] with 32-bit IP address, an additional 36% reduction is obtained in the number of effective nodes. This significant reduction in the number of effective nodes makes the application of BDD approach, for obtaining the optimized logic, even more effective.

The number of effective nodes are obtained during the construction of the binary decision tree for a few sample routing tables with IP address lengths of 3, 5, 8, and 16 and for the real-time 32-bit IP MAE-East routing table. The prefix distribution of a real-time routing table available at [1], has enabled to generate the prefixes in similar lines for the sample routing tables with IP address lengths of 3, 5, 8, and 16. It is observed that the construction of the binary decision tree for the MAE-East routing table required only 91 925 effective nodes, which is largely insignificant as compared with the theoretical upper bound of more than eight billion nodes. Further, when the output port is encoded with a 6-bit binary code and the reduction procedure is applied on each of the trees for individual binary output bits, the number of effective nodes obtained were only around 64% of the actual effective nodes. The summary of results is shown in Table III.

##### D. Implementation Issues

As mentioned earlier, the output interface ports at any router can be identified by at most an 8-bit binary code. Hence, for the above proposed scheme, the combinational logic design has to be done for eight output bits and, hence, that would give eight BDDs to be processed. Each of the synthesized logic can be mapped into one or more configurable logic blocks (CLBs) in

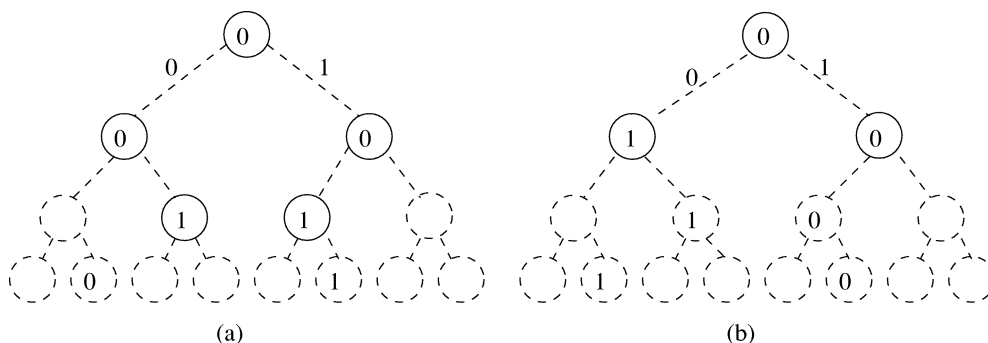


Fig. 4. Binary decision tree for (a)  $NHP_1$  (b)  $NHP_0$ , with all effective nodes assigned with output.

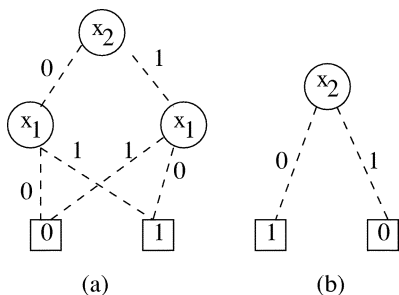


Fig. 5. BDDs for (a)  $NHP_1$  and (b)  $NHP_0$ .

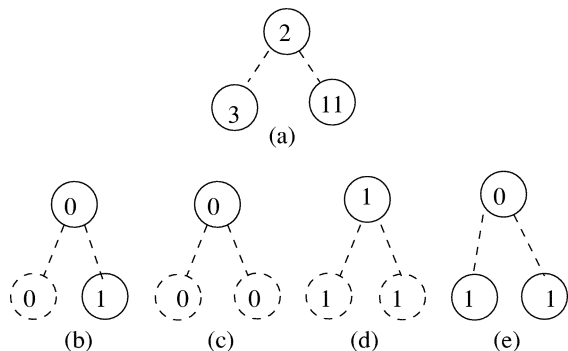


Fig. 6. (a) Nodes with assigned NHP ports. (b), (c), (d), (e) Output bits assigned to each of the nodes in 4-bit binary encoding of NHP.

an FPGA as shown in Fig. 7. The processing of the BDDs is performed with the SIS package [27]. The combinational logic subsequently obtained is implemented using Verilog coding and the logic synthesis is performed using the design analyzer tools from Synopsys [28].

1) *Timing Optimization*: Since the logic design obtained for the IP routing table is a combinational circuit, the timing optimization can be achieved using the *pipelining* and *retiming* techniques. Pipelining involves the insertion of delay elements at specific points of a circuit and retiming is the process of moving delays around a circuit such that the overall computation is unaltered. It aims to move a computation in an attempt to reduce the critical path, the path with the longest computation time without delays. By pipelining the computational data path, the throughput in terms of number of address lookups per unit time can be increased with a little or no additional cost in the overall area and latency.

## V. RESULTS AND ANALYSIS

The recent research in logic optimization [29], [30] using BDDs has proved that the logic implementation, with a binary decision tree size of more than 100 000 nodes is done in less than a second. Subsequently, it is an encouraging factor when the routing table, with only around 50 000 effective nodes on average, is implemented as a combinational logic optimized using BDDs. While the complexity metrics are important for assessing the feasibility of the implementation, it is equally important to measure the performance of the schemes for real-time routing tables. We measured the performance of our scheme with prefix database of real-time snapshots of various routing tables [1]. The implementation of the routing mechanism is performed as discussed earlier. The lookup time is measured as the propagation delay between the input and output ports of the combinational logic. This is same as the time taken for signal propagation along the critical path between the input and outputs of the logic. The critical path exists between one of 32 input signals and one of eight outputs. Thus, this measurement of propagation delay gives the worst case lookup time. The worst case lookup time and the corresponding packet throughput for the proposed scheme, for different routers, are shown in Table IV.

A main advantage with the proposed scheme is that, for an  $n$ -bit binary encoding  $2^n$  number of output ports can be represented and, hence, with an increase by one bit in the binary code twice the current number of output ports can be represented. Thus, the proposed scheme proves to be more beneficial in the scenario that the number of physical ports in a router would increase continuously. Besides, when the routing table is implemented in an FPGA, we can conclude that the IP address lookup rate is bounded only by the CLB delay. The maximum clock period bound for processing the IP address lookup would be the sum of one CLB delay and the maximum net delay. Previous hardware schemes have the lookup rate bounded by the RAM access speed. Further, in this scheme the resources required are utmost one FPGA while the other schemes require an ASIC and three or four-bank RAM.

### A. Routing Table Update

As discussed in the earlier sections, the routing table update time is one of the important metrics to be considered for a scheme attempting the IP address lookup problem. In earlier hardware schemes for IP address lookups, the update scheme

TABLE III  
EFFECTIVE NODES FOR SAMPLE ROUTING TABLES AND THE REAL-TIME  
32-BIT IP MAE-EAST ROUTING TABLE WITH 24792 PREFIXES

IP address length	Effective nodes before reduction	Effective nodes after reduction					
		$NHP_5$	$NHP_4$	$NHP_3$	$NHP_2$	$NHP_1$	$NHP_0$
3	8	-	-	-	-	5	3
5	31	-	-	-	20	18	20
8	116	-	-	85	87	81	85
16	3408	-	2098	2172	2181	2219	2164
32 (MAE-east)	91925	57955	58781	58577	58409	58387	58712

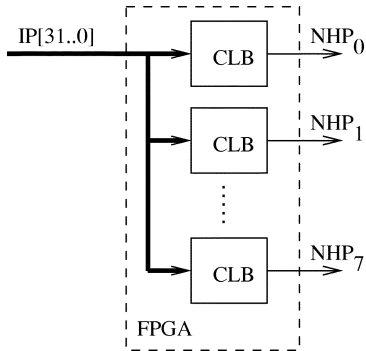


Fig. 7. CLB mapping in FPGA.

TABLE V  
MODIFIED ROUTING TABLE

<i>Prefix</i>	<i>length</i>	<i>NHP</i>
*	0	0
0*	1	1
01*	2	3
10*	2	2
11*	2	1
001*	3	1
101*	3	2

TABLE IV  
LOOKUP TIME PERFORMANCE ANALYSIS OF BDD BASED ROUTING ENGINE

<i>Router</i>	<i>Prefix count</i>	<i>lookup time (ns)</i>	<i>throughput (Million)</i>
MAE-west	29487	5.93	168.63
MAE-east	24792	5.81	172.11
AADS	33796	5.69	175.74
PacBell	6822	4.36	229.35

is based on the assumption of availability of redundant hardware resources, which can be a duplicate memory bank [18] or CAMs. When one unit is actively involved in the routing of packets, the redundant unit is used by the backbone router to update the routing table offline. The two units are switched alternatively for routing mechanism in a periodical fashion. In our scheme too, we assume a similar mechanism. In this scheme, we show that when there is an update in the routing table, then in most cases, not all of the logic blocks have to be recomputed, thus reducing the computational complexity. For example, if a prefix 11\* is inserted with an associated next-hop port to be 1 into the routing table shown in Table I, then the new routing table would be as shown in Table V.

The binary decision tree for the modified routing table would be as shown in Fig. 8(a). It is obvious that there is no change in the BDD representation for the output bit  $NHP_1$  while the slightly modified BDD representation for output bit  $NHP_0$  is as shown in Fig. 8(b).

However, this update of the logic may not be that simple for the 32-bit IP MAE-East routing table, but is also not as complex as assumed in general. To demonstrate this simplicity in updating the routing table, we have considered two consecutive snapshots of the MAE-East routing tables from [1], with the number of prefixes 19477 and 19525, respectively. The analysis for the updating of the table is done in terms of the number

of nodes at each level, in the binary decision tree for the latter routing table that differ in the output as compared with the corresponding nodes in the binary decision tree for the former routing table. Encouraging results have been obtained during this analysis and the results are shown in Table VI.

It is interesting to note that there is none or a significantly smaller variation in the output between the consecutive routing tables at the higher levels (level 0 to 15) and the lower levels (level 25 to 31). As is commonly known, the change in the logic would be minimal when the changes are minimum at higher levels in the binary decision tree, and we can observe that the same is the case in the current scenario. Furthermore, it can be observed that the number of nodes, in the levels 16–24, that differ in their outputs, are significantly smaller. Based on the observations, it can be safely concluded that, when the routing table is updated, there would only be a partial change in the combinational logic for each of the output bit. Thus, the reconfiguration of only those logic segments, that need to have the updated logic, can be done. The commercial availability of partially reconfigurable FPGAs makes this update scheme even more attractive, where in, only those CLBs that have a modified design can be reconfigured leaving the remaining CLBs unaltered.

### B. Scalability to IPv6

To demonstrate that an optimized combinational logic can be obtained for a mapping between 128-bit long IP address of IPv6 and the binary encoded next-hop port, it would be appropriate to show that the number of effective nodes that need to be processed by the BDD reduction techniques is significantly smaller than the theoretical upper bound. The performance analysis of our scheme in IPv4 was feasible since real-time 32-bit routing table were readily available [1]. However, a similar analysis was not possible in IPv6 due to the nonavailability of the routing table and, hence, to start with, we had to construct a routing table in line with the specifications of IPv6 protocol.

The building of the IPv6 routing table is done by incorporating the best-effort unicast address called *aggregatable global*

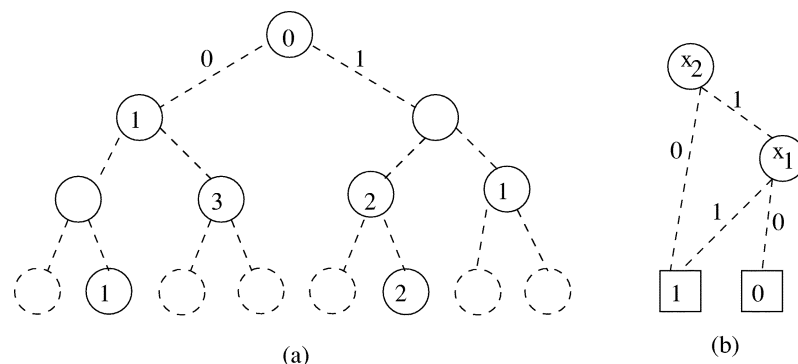
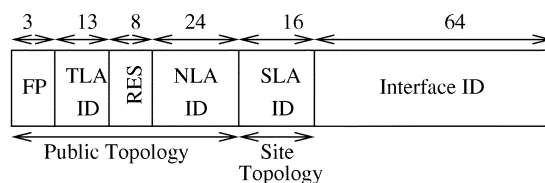


Fig. 8. (a) Binary decision tree representation of the modified routing table. Dotted nodes are redundant. (b) Modified BDD for  $NHP_0$ .

TABLE VI  
NUMBER OF CORRESPONDING NODES IN EACH LEVEL OF BINARY DECISION TREES THAT DIFFER IN THEIR OUTPUT. THE TWO BINARY DECISION TREES COMPARED ARE FOR ADJACENT SNAPSHOTS OF REAL-TIME MAE-EAST ROUTING TABLE

Level	Number of nodes					
	$NHP_5$	$NHP_4$	$NHP_3$	$NHP_2$	$NHP_1$	$NHP_0$
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	1	1	0	2	2	0
9	0	0	2	1	1	1
10	0	0	0	1	1	0
11	0	0	0	1	1	0
12	0	1	1	3	3	1
13	0	2	2	4	4	2
14	0	4	4	6	6	4
15	0	8	7	8	8	8
16	1	16	14	14	15	17
17	2	24	23	25	23	24
18	6	35	32	33	33	37
19	20	36	37	40	38	35
20	9	8	10	6	11	5
21	5	4	7	5	6	3
22	7	3	5	4	4	6
23	8	11	10	10	11	7
24	31	42	34	34	39	31
25	0	0	0	0	0	0
26	0	0	0	0	0	0
27	0	0	0	0	0	0
28	0	0	0	0	0	0
29	0	0	0	0	0	0
30	0	0	0	0	0	0
31	0	0	0	0	0	0

unicast address [31]. This address format was designed to facilitate scalable Internet routing, by providing an address hierarchy flow aggregation. The address format has a fixed structure as shown in Fig. 9 and is organized into a three level hierarchy: public topology; site topology; and interface identifier. The public topology consists of a two level hierarchy of service providers with a top-level aggregation identifier (TLA ID) and a next-level aggregation identifier (NLA ID). The TLA ID is initially to be restricted to 13 bits which translates to 8192 routers in the core IPv6 network. This was done to constrain



FP: Format Prefix, TLA ID: Top-Level Aggregation Identifier  
RES: Reserved for future use, NLA ID: Next-Level Aggregation ID  
SLA ID: Site-Level Aggregation ID

Fig. 9. Aggregatable global unicast address for IPv6.

core routing table sizes. The NLA ID is 24-bits long and allows for a flat or hierarchical allocation of the NLA address space. The site-level aggregation identifier (SLA ID) is 16-bits long. It is used by an individual organization to define its local address hierarchy and subnets.

The routing table constructed using the described IPv6 address format constituted a large database of 400 000 prefixes with prefix length ranging from 0 to 128 bits. The number of output ports were 512 and, hence, a 9-bit binary code is used to encode the NHP. The database is built such that the prefix length distribution in the IPv6 routing table should ratify the hierarchical topology of aggregatable global unicast address. The number of effective nodes are obtained during the construction of the binary decision tree for the IPv6 routing table with 128-bit long address. It is observed that the construction of the binary decision tree for the IPv6 routing table required only  $13.5 \times 10^6$  effective which is enormously insignificant as compared with the theoretical upper bound of  $6.8 \times 10^{38}$ . Further, when the output port is encoded with the binary code and the reduction procedure is applied on each of the trees for individual binary output bits, the number of effective nodes obtained were only  $7 \times 10^6$ , around 51% of the actual effective nodes. Thus, the significantly smaller number of nodes that need to be processed in the BDD solving of the logic shows that the scheme is scalable for the forthcoming IPv6. The implementation of the routing table and the measurement of the lookup time in IPv6 routing is reserved for our future study.

## VI. CONCLUSION

With the advancements in the communication link technologies the IP address lookup is becoming a major bottleneck in

router technologies. We propose a reconfigurable hardware solution, using the well received concept of BDDs, that provides an efficient IP address lookup along with providing a better scheme for updating the routing table. The argument, to support the adoption of BDD techniques for obtaining an optimized combinational logic, has been put forward by showing the fact that the number of effective nodes required to represent a 32-bit IP address routing table is significantly smaller than the theoretically required number of nodes. Besides, it has been shown that this number of effective nodes can be further reduced when the next-hop port is represented with a binary code and a tree representation is obtained for each of the output bits. The implementation of the routing scheme shows that the BDD hardware engine gives a throughput of up to 172.1 Ml/s for a large MAE-East routing table with 24 792 prefixes, a throughput of up to 168.6 Ml/s for an MAE-West routing table with 29 487 prefixes, and a throughput of up to 229.3 Ml/s for the Pacbell routing table with 6,822 prefixes. Thus, a data throughput of 200 Gb/s (with an average packet size of 1000 bits) can be obtained in the router implemented with the BDD based hardware address lookup engine.

Following the implementation of the scheme with the analysis of its performance in terms of the lookup time and packet throughput in IPv4 routing, and the proof that the processing time for the logic optimization in IPv6 routing tables is well under limit due to the emphatically smaller number of effective nodes, the next step is to obtain an implementation of the scheme for IPv6 and measure the lookup time.

#### ACKNOWLEDGMENT

The authors would like to thank Y. Ong, Department of Electrical and Computer Engineering, Iowa State University, for her simulation work as a support to our analytical study on IP routing tables. They would also like to thank S. Aluru for his valuable suggestions regarding the hardware synthesis of the designs.

#### REFERENCES

- [1] Routing-Analysis, Internet Performance Measurement and Analysis (IPMA) Project [Online]. Available: <http://www.merit.edu/ipma>
- [2] S. B. Akers, "Binary decision diagrams," *IEEE Trans. Comput.*, vol. C-27, pp. 509–516, June 1978.
- [3] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, no. 8, pp. 677–691, Aug. 1986.
- [4] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," in *Proc. 27th IEEE/ACM Design Automation Conf.*, 1990, pp. 40–45.
- [5] M. A. Ruiz-Sanchez, E. W. Biersack, and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, vol. 15, pp. 8–23, Mar.-Apr. 2001.
- [6] K. Sklower, "A tree-based packet routing table for Berkeley Unix," in *Proc. Winter Usenix Conf.*, 1991, pp. 93–99.
- [7] D. Morrison, "PATRICIA—Practical Algorithm to Retrieve Information Coded in Alphanumeric," *J. ACM*, vol. 15, no. 4, pp. 514–534, Oct. 1968.
- [8] T. Kijkanjanarat and H. J. Chao, "Fast IP lookups using a two-trie data structure," in *Proc. Global Telecommunications Conf. GLOBECOM '99*, vol. 2, 1999, pp. 1570–1575.
- [9] H. H.-Y. Henry Hong-Yi Tzeng and T. Tony Przygienda, "On fast address-lookup algorithms," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1067–1082, June 1999.
- [10] N. Yazdani and P. S. Min, "Fast and scalable schemes for the IP address lookup problem," in *Proc. IEEE Conf. High Performance Switching and Routing*, 2000, pp. 83–92.

- [11] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, "Scalable high speed IP routing lookups," *Proc. ACM SIGCOMM '97*, vol. 27, no. 4, pp. 25–36, Oct. 1997.
- [12] —, "Scalable high-speed prefix matching," *ACM Trans. Comput. Syst.*, vol. 19, no. 4, pp. 440–482, Nov. 2001.
- [13] B. Lampson, V. Srinivasan, and G. Varghese, "IP lookups using multiway and multicolumn search," in *Proc. IEEE INFOCOM '98*, San Francisco, CA, 1998, pp. 1248–1256.
- [14] V. Srinivasan and G. Varghese, "Fast address lookups using controlled prefix expansion," in *Proc. ACM Sigmetrics '98*, June 1998, pp. 1–11.
- [15] A. Donnelly and T. Deegan, "IP route lookups as string matching," in *Proc. 25th Annual IEEE Conf. Local Computer Networks, LCN*, 2000, pp. 589–595.
- [16] S. Nilsson and G. Karlsson, "IP address lookup using LC-Tries," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 1083–1092, June 1999.
- [17] A. McAuley, P. Tsuchiya, and D. Wilson, "Fast Multilevel Hierarchical Routing Table Using Content-Addressable Memory," U. S. Patent serial number 034444, 1995.
- [18] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," in *Proc. IEEE INFOCOM '98*, vol. 3, San Francisco, USA, 1998, pp. 1240–1247.
- [19] N. Huang, S. Zhao, J. Pan, and C. Su, "A fast IP routing lookup scheme for gigabit switch routers," in *Proc. IEEE INFOCOM '99*, vol. 3, New York, 1999, pp. 1429–1436.
- [20] P.-C. Pi-Chung Wang, C.-T. Chia-Tai Chan, and Y.-C. Yaw-Chung Chen, "A fast IP routing lookup scheme," in *Proc. IEEE Int. Conf. Communications, ICC 2000*, vol. 2, 2000, pp. 1140–1144.
- [21] T. Chiueh and P. Pradhan, "High-performance IP routing table lookup using CPU caching," in *Proc. IEEE INFOCOM '99*, vol. 3, New York, 1999, pp. 1421–1428.
- [22] I. Y.-L. Ilion Yi-Liang Hsiao and C.-W. Chein-Wei Jen, "A new hardware design and FPGA implementation for internet routing toward IP over WDM and terabit routers," in *Proc. IEEE Int. Symp. Circuits and Systems, ISCAS 2000*, vol. 1, Geneva, 2000, pp. 387–390.
- [23] D. Pao, C. Liu, A. Wu, L. Yeung, and K. S. Chan, "Efficient hardware architecture for fast IP address lookup," in *Proc. IEEE INFOCOM '2002*, vol. 2, New York, 2002, pp. 555–561.
- [24] D. E. Taylor, J. W. Lockwood, T. S. Sproull, J. S. Turner, and D. B. Parlour, "Scalable IP lookup for programmable routers," in *Proc. IEEE INFOCOM '2002*, vol. 2, New York, 2002, pp. 562–571.
- [25] K. Thompson, G. J. Miller, and R. Wilder, "Wide-area internet traffic patterns and characteristics," *IEEE Network*, vol. 11, no. 6, pp. 10–23, Nov.-Dec. 1997.
- [26] P. Newman, G. Minshall, and L. Huston, "IP switching and gigabit routers," *IEEE Commun. Mag.*, vol. 35, no. 1, pp. 64–69, Jan. 1997.
- [27] E. M. Sentovich *et al.*, "SIS: A System for Sequential Circuit Synthesis," Electronics Research Laboratory, Univ. California, Berkeley, CA, Memorandum UCB/ERL M92/41.
- [28] [Online]. Available: <http://www.synopsys.com/>
- [29] R. Drechsler and W. Gunther, "Optimization of sequential verification by history-based dynamic minimization of BDDs," in *Proc. IEEE Int. Symp. Circuits and Systems, ISCAS 2000*, vol. 4, Geneva, 2000, pp. 737–740.
- [30] V. Bertacco and M. Damiani, "The disjunctive decomposition of logic functions," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design, ICCAD '97*, 1997, pp. 78–82.
- [31] P. Boustead and J. Chicharo, "Label switching using the IPv6 address hierarchy," in *Proc. IEEE Global Telecommunications Conf. GLOBECOM 2000*, vol. 1, 2000, pp. 500–504.
- [32] R. Rama Sangireddy and A. K. Arun K. Somani, "Binary decision diagrams for efficient hardware implementation of fast IP routing lookups," in *Proc. IEEE Int. Conf. Computer Communications and Networks, ICCCN*, Oct. 2001, pp. 12–17.



**Rama Sangireddy** (S'98) received the B.Tech. degree with distinction in electrical and electronics engineering from the Regional Engineering College, Warangal, India, and the M.S. degree in electrical engineering from the University of Missouri, Rolla, in 1996 and 1999, respectively, and is currently working toward the Ph.D. degree in computer engineering at Iowa State University, Ames, IA.

His research interests include computer architecture, reconfigurable computing, computer communication networks, and fault tolerant computing.



**Arun K. Somani** (S'83–M'83–SM'88–F'99) received the B.E. (Honors) degree in electrical and electronics engineering from the Birla Institute of Technology and Science (BITS), Pilani, India, in 1973, the M.Tech. degree in computer engineering from the Indian Institute of Technology, Delhi, India, in 1979, the M.S.E.E. and Ph.D. degrees in electrical engineering from the McGill University, Montreal, Canada, in 1983 and 1985, respectively.

He is currently the Jerry R. Junkins Endowed Professor of Electrical and Computer Engineering at Iowa State University. He worked as Scientific Officer for the Government of India, New Delhi, from 1974 to 1982. From 1985 to 1997, he was a Faculty Member at the University of Washington, Seattle, WA, where he was a Professor of Electrical Engineering and Computer Science Engineering, since 1995. His research interests are in the areas of fault tolerant computing, computer communication and networks, optical networking, computer architecture, and parallel computer systems. He has taught courses in these areas and published more than 150 technical papers.

Prof. Somani has served on several program committees of various conferences in his research areas, was the General Chair of IEEE Fault Tolerant Computing Symposium and Technical Program Chair of IEEE Conference on Computer Communications and Networks. He is currently serving as an Associate Editor of IEEE TRANSACTIONS ON COMPUTERS and an Editor of *Microprocessors and Microsystems*. He was also a Distinguished Lecturer and Distinguished Tutorial Speaker of the IEEE.