# High Performance IP Forwarding Engine for ATM based Gigabit Routers

Byeong-Cheol Choi*, Chang-Sik Choi*, Youn-Kwae Jeong*, and Jeong-Tae Lee**

*Electronics and Telecommunications Research Institute
#161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Korea
Tel : +82-42-860-6392, Fax : +82-42-860-5440
E-mail : bcchoi@etri.re.kr
** Department of Computer Engineering, Pusan National University
#30 Jangjeon-Dong, Kumjeong-Gu, Pusan, 609-735, Korea
E-mail: jtlee@hyowon.pusan.ac.kr

**Abstract :** In this paper, we proposed high performance packet forwarding engine for asynchronous transfer mode(ATM) based gigabit routers. The forwarding engine is based on ATM switch and accommodates four 622Mbps ports. The forwarding engine has been designed to be able to process the Internet protocol(IP) packet at 2.5Gbps using the pipelined IP header processing and lookup control mechanism. For high performance packet forwarding, we used content addressable memory(CAM) based routing coprocessor operating in hardware and implemented the pipelined lookup control function into a field programmable gate array(FPGA). The pipelined packet header processing mechanism enhanced the forwarding performance of the IP packets ingressed from four different 622Mbps ports. Moreover, the IP lookup controller designed to have the performance up to 12.5Mpps. The proposed forwarding engine is also designed to support differentiated services(DS) and multiprotocol label switching(MPLS).

## 1. Introduction

The transformation of the Internet into an important and ubiquitous commercial infrastructure has created rapidly rising bandwidth demand. In order to keep up this demand in services and bandwidth, all Internet service providers(ISPs) have scaled their networks severalfold in the size and bandwidth. The need to build fast forwarding engines is being addressed in a variety of ways[1]. A number of gigabit router, for example the Ascend GFR, the Lucent Cajun Switch and the Cisco 12000 router, are commercialized. All of the designs use the same functional components illustrated in Figure 1. The line card contains the physical layer components necessary to interface the external data link to the switch fabric. The switch fabric is used to interconnect the various components of the gigabit router. The forwarding engine inspects packet headers, determines to which outgoing line card they should be sent. The routing control processor runs the routing protocols and computes the routing tables that are copied into each of the forwarding engines. It handles network management and housekeeping functions and may also process unusual packets that require special handling. The proposed gigabit router follows the same architecture described above. The forwarding engine is based on ATM switch and accommodates four 622Mbps switch ports. The forwarding engine has been designed to be able to process the IP packet at 2.5Gbps using the pipelined IP header processing and lookup control mechanism. For high performance packet forwarding, we used CAM based routing coprocessor operating in hardware and implemented the pipelined lookup control function into FPGA. The pipelined packet header processing mechanism enhanced the forwarding performance of the IP packets ingressed from four different 622Mbps ports. The proposed forwarding engine also designed to support differentiated services and multiprotocol label switching[2,3].

This paper is organized as follows. In section 2, the architecture of the hardware based forwarding engine is introduced. In section 3, the pipelined IP lookup controller implemented into FPGA is proposed. In section 4, we examine the performance of the forwarding engine in each pipelined phase. Finally, conclusions are given.

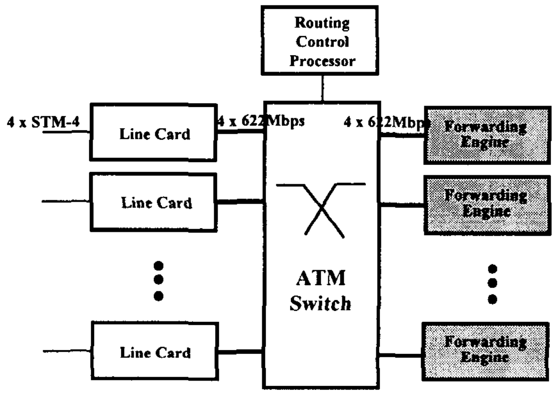


Figure 1. Architecture of ATM based gigabit routers

## 2. Architecture of Forwarding Engine

The architecture of the proposed forwarding engine consists of multiple switch interface modules and segmentation and reassembly(SAR) controller modules, forwarding engine manager module, IP lookup controller module, forwarding table, label information table, and routing table as shown in Figure 2.

The forwarding engine is designed with multiple 622Mbps switch interfaces and SAR controllers for accommodating gigabit Internet traffic. The lookup controller processes packet header analysis, header validation, lookup control, and header manipulation in pipelined scheme for multiple ports. The search engine is designed with the CAM based routing coprocessor which has a fully deterministic search time. It provides the function of the longest prefix match and the 64 bit exact match. The label information table stores the information which associated with L3/L4 lookup, differentiated service and multiprotocol label switching. The forwarding engine manager receives a forwarding information from the routing control processor and stores it into forwarding table and label information table.

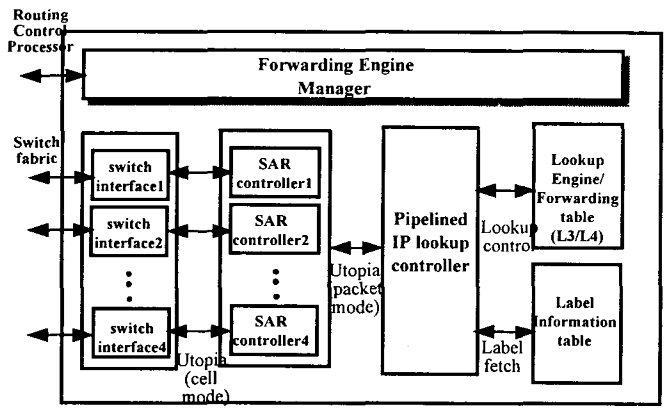


Figure 2. Functional block of forwarding engine

The architecture between lookup controller and multiple SAR controllers is designed with packet-extended universal test and operations physical interface for ATM(UTOPIA) interface. The packet transmission path is shared with multiple SAR controllers. The path between receiving SAR controller and lookup controller is divided into two paths for a packet header and a payload transmission. This architecture provides the higher transmission utilization because a packet header can be transmitted to the lookup controller while a payload of another SAR controller is transmitted. This architecture is shown in Figure 3. In this architecture, each SAR controller operates in master mode and the lookup controller in slave mode. The slave can control the packet fetch operation by asserting and negating the transmitting available signal. When the available signal is asserted, the receiving SAR controller can send the packet data to the IP lookup controller. If the slave negates the signal before the end of packet is transmitted, the master stops the sending operation until the signal is asserted. If the slave asserts the signal again, the master starts the transmission from the next remaining data packet. For IP lookup, only the packet header data is used and the payload is not.

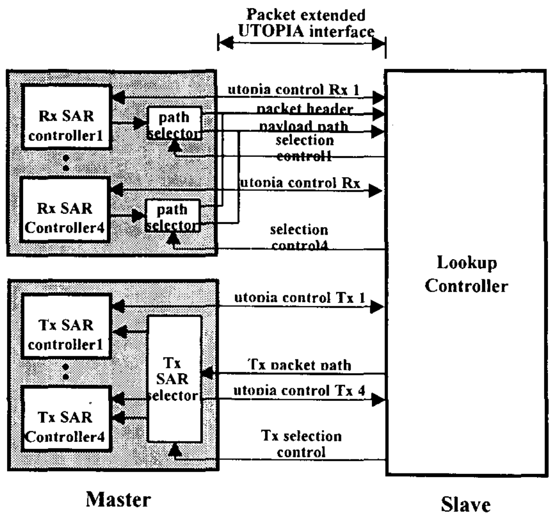


Figure 3. Interface between SAR controllers and lookup controller

## 3. IP Lookup Controller

The proposed lookup controller is designed to process IP header in a pipelined scheme for high performance packet forwarding. The functional phase is divided into header analysis and verification, L3 and label lookup, label information fetch, outgoing header modification, and classification. The IP lookup controller has been implemented into FPGA for fast header processing in parallel. The structure of IP lookup controller is shown in Figure 4.

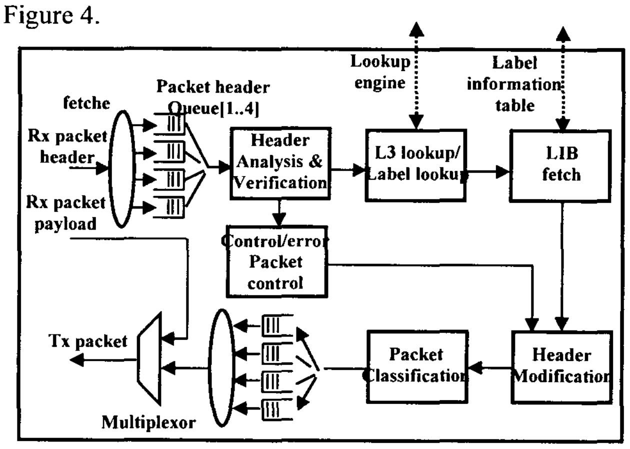


Figure 4. Structure of IP lookup controller

### 3.1 Header Fetcher

The packet header is basically fetched in round robin scheme among multiple Rx SARs. The lookup controller operating in slave mode checks the service states of the fetched header, the packet arrival in each Rx SAR and the sending of payload. If the following conditions of { service of the fetched header is finished AND any packet is arrived AND the payload is in sending state} are satisfied, the packet header in that SAR is fetched. Otherwise, the operation of check for the next SAR is performed. State transition of arbitration for fetching is shown in Figure 5.

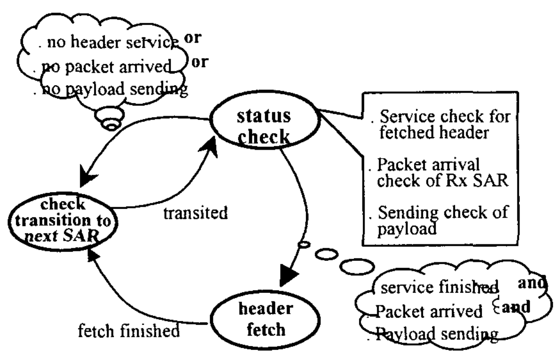


Figure 5. State diagram of arbitration for fetching

### 3.2 Header analysis and verification

The IP lookup controller analyzes the packet type such as IP over ATM(IPOA), point-to-point protocol(PPP), MPLS, null encapsulation. After that analysis, it starts the verification for time-to-live(TTL), IP version and header checksum check in parallel. In this phase, this kind of packet such as ATMARP, PPP control and the invalidated header are excluded out of lookup procedures and directly transferred to the header modification phase.

### 3.3 Lookup and label fetch

In this proposed forwarding engine, we designed with CAM based routing coprocessor operating in hardware instead of processor based software search algorithm. The routing coprocessor can accommodate up to 128K forwarding entries. Routing entries are stores in forwarding table which consists of a hierarchical structure of address partitioned by IP masks. A technique can be used that will encode a ternary value and store it in the forwarding table[4,5]. The IP address and IP mask can be combined and stored as 64 bits. These 64-bit values are stored in the forwarding table in IP mask order[6]. That means the IP mask values that have the higher number of contiguous 1's are stored in the higher priority location. To store an address and mask in a 64-bit word, the bit wise logical AND of the address and mask are store in the upper 32 bits while the bit wise logical AND of the mask and the address's one's complement are stored in the lower 32 bits. Routing coprocessor has a fully deterministic search time, independent of the size of the list and the position of the data in the list. Address fields from the packet header are transmitted with compare instruction and control signal. This operation indicates the comparison cycle in hardware. Figure 6 shows the state diagram of hardware based lookup procedure. The state is classified into the arrival check of the validated IP header, a address extraction, comparison, and index fetch. The comparison state is only CAM based memory access which takes just 50nsec. Others are operated by 100MHz FPGA implementation reference clock. In Figure 6, the transition time in each state is shown. As a result of the comparison against a list of entries stored in the array, the routing coprocessor generates an index that is used to access an external label information table where outgoing label and other associated information is stored. The label information table is consists of L3 lookup and label exact match associated information.

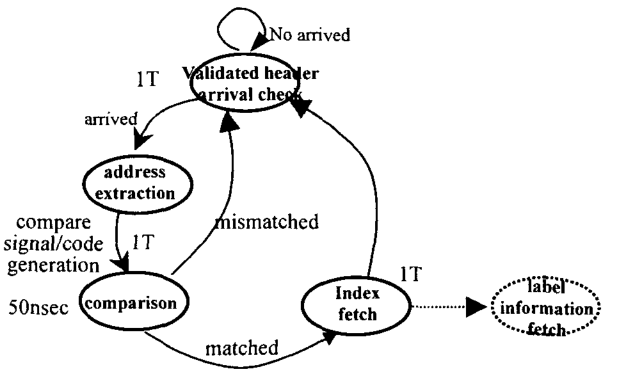


Figure 6. State diagram of hardware based lookup

### 3.4 Header modification and classification

From the fetched label information, we obtain outgoing DS value corresponding to the incoming IP packet. In this phase, the header format is modified according to the outgoing packet type such as IPOA, PPP, MPLS and null encapsulation. At the same time, the modification or replacement of DS code is also accomplished. The DS code is used for packet classification. The packet classifier stores the modified packet header into the outgoing queue according to DS code. In this implementation, The scheduling for transmission is performed from the higher priority packet. When a higher priority arrives while a lower priority packet is being serviced, the scheduler examines the remaining packet size of the transmitting packet. If the size is larger than the predefined threshold value, the higher priority preempts the transmission of the lower priority and sends its packet. At the same time, the priority of the preempted packet is raised by one priority. When the transmission is finished, the scheduler selects the next packet.

### 3.5 Packet Multiplexor

In this paper, we used a multiplexor for sending a outgoing packet from the modified packet header and the remaining payload. Just before the scheduler sends the last several data of the packet header, it requests the receiving SAR to fetch the remaining payload. It is possible to transmit the payload in cut through using multiplexor. This scheme does not require any buffer for payload in the lookup controller. Figure 7 shows the timing diagram of cut through transmission of outgoing packet.

## 4. Forwarding Performance

The IP lookup controller is implemented into a FPGA. The lookup control is processed in pipelined and hardware. The processing is divided into 5 phases. We used 100MHz global clock in implementing FPGA. Therefore, we can

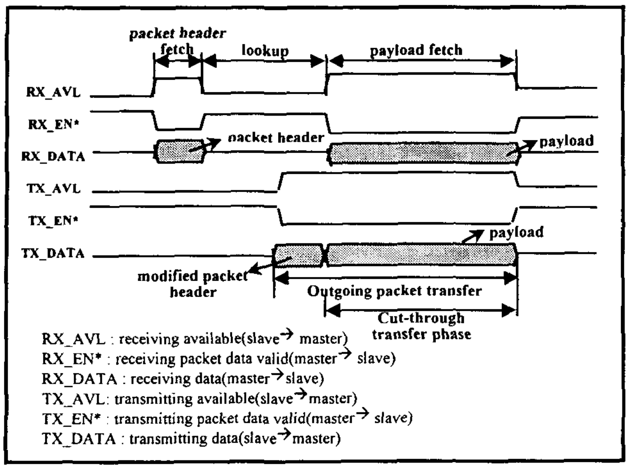

packet header fetch
lookup
payload fetch
RX_AVL
RX_EN*
RX_DATA
packet header
payload
TX_AVL
TX_EN*
TX_DATA
payload
modified packet header
Outgoing packet transfer
Cut-through transfer phase
RX_AVL : receiving available(slave→ master)
RX_EN* : receiving packet data valid(master→ slave)
RX_DATA : receiving data(master→slave)
TX_AVL: transmitting available(slave→master)
TX_EN* : transmitting packet data valid(master→ slave)
TX_DATA : transmitting data(slave→master)

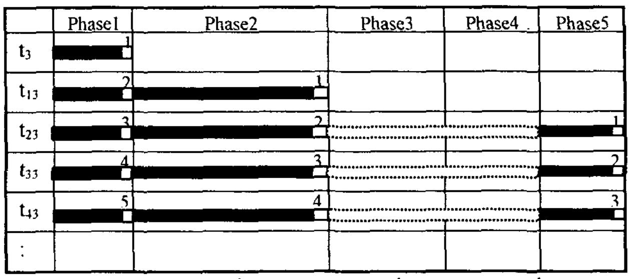
Figure 7. Timing diagram for the packet transmission

show processing time in the unit of global clock. In first phase, the type analysis of IP packet and IP header verification is performed. The check of TTL, IP version, option, IP header checksum is performed in parallel. The longest processing time is 3T of checksum check. In second phase, CAM based lookup which is longest prefix match(LPM) for L3 lookup or label exact match for MPLS are performed. The O(1) CAM based LPM lookup time is 50nsec. Additional 3T is taken for address extraction and index fetching as mentioned in Figure 6. In third phase, with the resulting index of phase 2, the corresponding information is fetched. When we use 32 bit wide SRAM, it requires 2 accesses for fetching the 8 byte information. It takes 3T for control and fetching. In forth, TTL decrement, DS field manipulation, checksum generation, and packet header modification is performed according to label information including packet type, hop count, DS information. In fifth, the modified header is classified into output queue, which takes 2T. Table 1 shows the processing time in each phase.

Table 1. Processing time in each phase.

| phase | function | Time |
|---|---|---|
| Phase 1 | Header analysis & verification | 3T(30nsec) |
| Phase 2 | L3 lookup/L4 lookup | 50nsec + 3T (80nsec) |
| Phase 3 | Label Information fetch | 4T(40nsec) |
| Phase 4 | Header modification | 3T(30nsec) |
| Phase 5 | classification | 2T(20nsec) |
|  | Total processing time | 50nsec + 15T = 200nsec |

The flow of pipelined lookup control for time versus phase is conceptually shown in Figure 8. In this case, we assumed that the packet header is serviced continuously from the receiving header queues. We can see the service time is dependent on the phase2 which has the longest deterministic processing time among all phases. As shown in Figure 8, the first service is finished at $t_3$ in phase 1 and it goes into phase 2 directly. The second service waits in the phase 1 until the first service is finished in the phase 2. Next, while the second service is in phase 2 the third service waits in phase 1 until the second service is finished in phase 2. However, the first service goes through the phase 3, phase 4 and phase5. The next services follow the above described scheme.



1: 1st service, 2 : 2nd service, 3 : 3rd service 4: 4th service
▬ : processing state

Figure 8. Pipelined processing in IP lookup control

## 5. Conclusions

We have implemented the pipelined IP lookup mechanism into FPGA. The pipelined design improves the utilization of lookup operation so that the forwarding performance is enhanced. We found that the total processing time for one packet takes 200nsec when we used 100 MHz clock(1T=10nsec) in FPGA implementation. The proposed forwarding engine provides the performance of 2.5Gbps. We can maximize the packet forwarding performance up to 12.5 million packets/second when we consider only the lookup processing time. Moreover, the proposed gigabit router can provide the new advanced functionality for differentiated services and multiprotocol label switching.

## References

[1] S. Keshav and Rosen Sharma, " Issues and Trends in Router Design," *IEEE Communications Magazine*, pp. 144 – 151, May 1998.

[2] R. Callon *et al.*, "A Framework for Mutiprotocol Label Switching," Internet Draft draft-ietf-mpls-framework-02.txt, Nov. 1997.

[3] V. J. Kumar, T. V. Lakshman, and D. Stiladia, "Beyond Best Effort : Router Architectures for the Differentiated Services of Tomorrow's Internet," *IEEE Communications Magazine*, pp. 152 – 164, May 1998.

[4] T. B. Pei and C. Zukowski, "Putting Routing Tables in Silicon," *IEEE Network Magazine*, pp. 42 – 50, Jan. 1992.

[5] A. McAuley and P. Francis, "Fast Routing Table lookup using CAMs," *Proc. of IEEE INFOCOM'93*, pp. 1382-1391, Mar. 1993.

[6] V. Fuller, T. Li, J. Y, and K. Varadhan, "Classless InterDomain Routing(CIDR)," RFC 1519, IETF, Sep. 1993.