



High-speed routers design using data stream distributor unit

Ali El Kateeb*

Department of Electrical and Computer Engineering, University of Michigan

Received 14 April 2005; received in revised form 30 August 2005; accepted 23 September 2005

Abstract

As the line rates standards are changing frequently to provide higher bit rates, the routers design has become very challenging due to the need for new wire-speed router's network processor (NP) unit. Typically, designing new NPs could take a long time and is very costly. In this work, we are presenting a new approach in high-speed routers design. Our approach is to use a data stream distributor (or DSD) to split the high bit rate line to few lower rate lines. These low rate lines will be processed by existing NPs that are already in use with today routers that are designed to support such low line rates. Such approach will allow the developing of routers in a short time and at a low cost. Clearly, there are many design challenges associated with this approach of routers design such as load balancing, buffer managing, and traffic distribution.

This paper discusses the concept, advantages, and the architecture of the DSD approach. Also, we highlight the implementation of the DSD chip design using a Virtex Xilinx System-On-Chip (SOC) and specifically the Virtex XCV 150 chip. The cycle's accurate simulation has shown that the designed DSD chip is capable of splitting a 2.5 Gb/s line rate to four low bit rate lines of 622 Mb/s. The chip has 118,065 gates and runs at 70 MHz.

© 2005 Elsevier Ltd. All rights reserved.

Keywords: Router architecture; Network processor; FPGA; Cycle accurate simulation

*Tel.: +1 3135935115; fax: +1 3135836336.

E-mail address: elkateeb@umich.edu.

1. Introduction

The Internet traffic is increasing annually at a rate between 4 and 10 times. Such increase has occurred due to the rapid growth of the number of Internet users and to the support of different communications modes by today networks such as video, voice, and data that provide new applications such as entertainment, shopping, etc. Such spectacular increase with the annual traffic has required accelerating the packet forwarding capability of the network nodes, specifically the routers and switches. The design of routers/switches has become very challenging and has to be scalable, provide high-performance, flexible to protocol changes, and robust.

Typically, the routers design has many functional units such as line interfaces, network processors (NPs), switch fabric, and system processor (Fig. 1). The line interfaces to receive and transmit data and to provide framing functionality and attach the transmission channels to the router. The NP interfaces to the physical data links and provides the high-speed processing for lookup routing tables, analysis of packet headers, checking QoS rules, packet buffering and queuing, packets classifications based on their destination, analyses of source addresses, and control information. The switch fabric provides high-speed switching for the router's NPs. The system processor performs other router functions such as management and route computation.

The general structure of the NP is shown in Fig. 2 (O'Connor and Gomez, 2001). Typically, the operation principle of the NP is carried out as follows: the packets from the line interface or the switch fabric arrive at the NP and are processed by the NP's interface unit. This unit extracts the information needed to process the packet, such as the TCP source/destination port numbers, IP source/destination address. The extracted information is fed to the packet processor and the packet itself will be written to the packet memory. This packet processor could further process the packet header and submit part of the header to the search engine, which then performs the search of the IP address on the routing table. Also, the packet processor could process the virtual circuit/virtual path identifier lookup if the packet is recognized as an ATM cell. In addition, this processor handles the packet classification and is able to synchronize its operation with hardware accelerators, which are implemented to offload the packet processor from some performance-critical functions such as header checksum calculation, encryption/decryption, and so on. The packet processor has the control over the packet memory manager where the processor can instruct the manager to transmit any packet after some modification to its header. The packet processor is

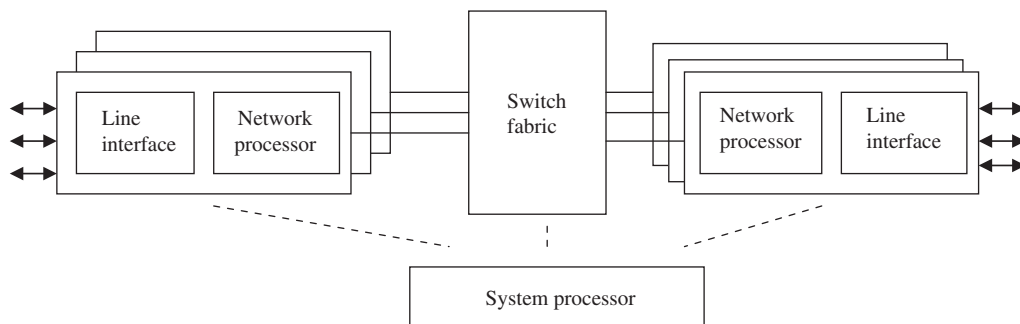


Fig. 1. Router architecture.

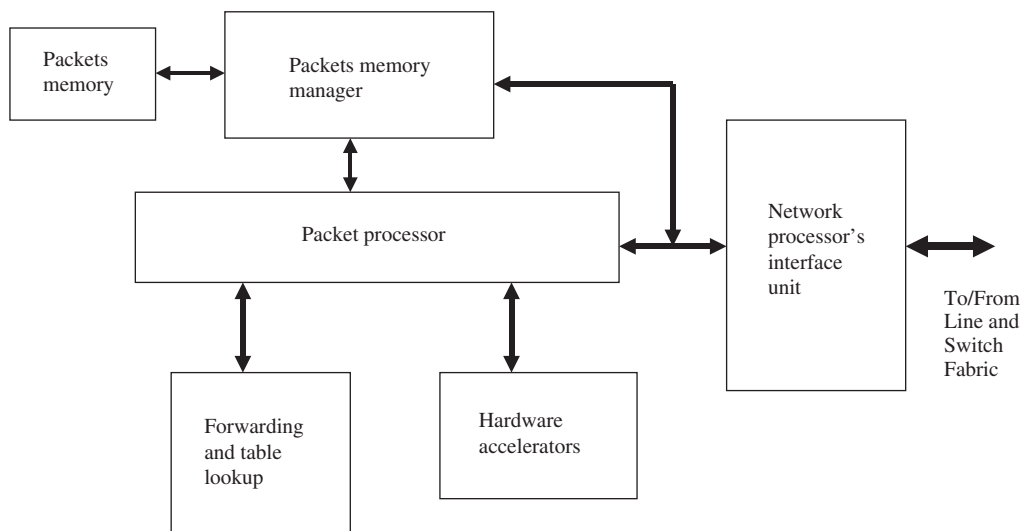


Fig. 2. A general single network processor structure.

designed to be programmable, which allows its function to be rapidly adapted to continuously changing communication protocols.

Today's routers have to be able to process multi-terabit per second data rates in order to handle the rapid growth of Internet traffic. Therefore, routers have to be designed to cope efficiently with new wire speed rates that reach 10 Gbps (OC 192) and beyond in the near future. Achieving such objective will require to design high-performance NPs, implementing advanced schemes to speed up certain router's functions such as the lookup function (O'Connor and Gomez, 2001; Shah and Gupta, 2001). For instance, the iFlow address processor has been designed based on the use of ultra-wide DRAM to perform fast forwarding-table lookups. The use of Ternary Content-Addressable Memory (TCAM) has also been reported (Shah and Gupta, 2001; Ruiz-Sánchez, 2001). The multiple NPs that run in parallel would be used to enhance the routers' performance (Bux et al., 2001).

In this paper, we evaluated the use of a novel scheme called Data Stream Distributor (DSD) that allows routers to support high line rates using those NPs that are designed for low line rates (Kateeb et al, 2003). The DSD-based routers shall distribute received messages from a high-speed line to many low-speed NPs that were designed previously to support lower line rate.

The next section discusses the main challenges that face the design of NPs. Section 3 will investigate the parallel NPs design and the reasons behind the needs for DSD approach. In Section 4, the general description of the DSD approach is highlighted. The DSD architecture and design is investigated in Section 5. The implementation of the DSD is discussed in Section 6 and a conclusion regarding this paper is stated in Section 7.

2. Processing challenges and related work

Today's NPs use multiple-dedicated processors to provide processing services required by routers. Each of these dedicated processors, shown in Fig. 2, can process one or more of

router's task(s). This form of parallelism scales nicely up to the point where the data rate for a single-line interface reaches the NP's performance. The breaking point from a complexity standpoint is when a single NP can no longer handle a single line interface. Therefore, adding more dedicated processors and more processing tasks run in parallel will enhance NP to handle faster single data stream. As more processors are added and tasks are redistributed between them, system complexity will be increased and changes in its functionality cause a further strain on the system.

The architecture of the single NP has reached its limit and parallel NPs should be used instead. There are two main issues behind the need for parallel NPs: (i) the processing power of existing single NPs cannot meet the processing requirements to support the new line rate standard, and (ii) the limited speed of the available packets memory design.

The single NP's performance under a single-line interface can be easily estimated. With a minimum packet size in the range of 40 bytes, as in the case of TCP/IP ACK packet and with the current Internet backbone link rates of 2.5 Gb/s (OC-48), the current wire-speed forwarding rate has become 6 million packets/s. Typically, two to three instructions are required to process packet byte for a single routing table lookup using a general-purpose processor (Geppert, 2001). Furthermore, the layer 2/3 simple forwarding operations require multiple lookups per packet, such as MAC address resolution/learning and IP lookup. This implies that a NP performance of about 2.5 billion instructions/s is required. Clearly, we are not considering the processing for QoS and packet classification. Adding the processing required for QoS, a NP with much higher processing power than 2.5 billion instructions/s will be required to support 2.5 Gb/s line. Moreover, considering the use of new standards for line rates such as the 10 Gb/s (OC-192) and 40 Gb/s (OC-768), the NP should be capable of providing a minimum processing power of 10 and 40 billion instructions/s for 10 and 40 Gb/s, respectively (Bux et al., 2001).

Today's NPs use a different architecture to cope with the processing requirements of 2.5 and 10 Gb/s links. As mentioned during the description of the general NP structure, the hardware accelerators are used to offload computationally intensive functions from the packet processor. Therefore, it is important that the processor should be capable of dispatching an operation to these hardware accelerators, which are only coprocessors, while it continues to do other work. This will avoid stalling the processor core while waiting for the accelerator to complete its processing. The multi-threaded architecture is used widely in designing the packet processor within the single NP. For example, an 8-way multi-threaded processor is used in the packet processor to achieve the processing requirements for 10Gb/s line. Others use 16 RISC cores hooked up on a high-speed bus to run as a multi-processor that supports the same line rate (Geppert, 2001). Also, a single NP with 64 multi-threaded cores supported by very wide data paths of 512 bits has been reported (Building next generation network processors, 1999).

The other issue supporting the use of parallel NPs is the speed limits of the available packet memory design required for next generation terabit routers. The commercially available DRAMs have an access rate lower than that required by the high-speed NPs. The memory speed requirements can be estimated from the number of memory access required per packet and the bit rates required for the packets transfer per second. Four accesses to the memory buffer of NPs are needed for each arrived packet until that packet will be departed. The packet has to be stored at its arrival, then the packet processor reads the header where writes back to the processed header, and finally the packet processor instructs the packets memory manager to transmit the packet. For small packets, which

represent 40% of all Internet packets, the memory interface of 10 Gb/s is required for a line rate of 2.5 Gb/s. This memory access rate is exceeding the capacity of today's commercial DRAMs (Bux et al., 2001). However, the memory interleaving technique can be used to distribute the data over multiple parallel DRAM chips, which allows memory access rate acceptable for 2.5 Gb/s line rates. For line rates of higher speed and specifically the new standard of 40 Gb/s, the memory bus should be very wide and therefore, the number of DRAMs chips will be very large. An alternative technique can be achieved by using on-chip ultra-wide DRAM technology to support the memory capacity. Such technique has been used in the address processor design, which is one of the hardware accelerators coprocessor used in the single NP architecture (Wolf and Turner, 2000). The size of the on-chip DRAM design for a single NP is always limited by the chip capacity. Implementing a large buffer that will be fast enough to provide the wire-speed processing and be able to hold all packets will be a real challenge, especially for the new line rate standards.

3. The parallel network processors design

The support for the processing power and the memory rate requirements have already reached the limit of what single NPs can offer for a line rate of 10 Gb/s. The support for high rate lines (40 Gb/s or more) used in the next generation of multi-terabit routers requires the use of parallel NP architecture supported with ultra-wide DRAM memory. It is important that the design of such parallel NPs should not be started from scratch. In other words, the parallel NPs should be based on the use of a single NP that was developed previously. Generally, the available single NPs are designed to work as stand-alone processors and have no support for a parallel processing system environment. Any modifications to the existing single NP will require a major hardware re-design to mount many of these processors on a single system bus or communications network. Other issues such as tasks distribution, operating system design, memory and cache design should be taken into consideration for parallel processing system. The use of many single NPs to build parallel NPs using the conventional techniques that are already used in the field of computer architecture is an important area of research. However, we believe that using the DSD approach is very interesting to develop the parallel NPs in a short time and at minimum cost.

This approach allows the usage of the DSD, which splits the high rate data stream, say 40 Gb/s, into multiple lower rate data streams such as 10 Gb/s or 2.5 Gb/s. Each lower rate streams will then forward to a separate single NP, which is designed to handle the lower rate stream. The main structure of the DSD-based parallel NPs is shown in Fig. 3.

The following points have highlighted the impact of the use of the DSD-based parallel NPs approach on the NPs and routers design:

1. The single NP developed for low line rates can still be used to support lines with higher rates without the need to design a new NP. Typically, when the single NP is required to support the new higher line rate standard, the existing NP has to be updated or completely designed. However, using the DSD-based parallel NPs will split the high-speed stream to many lower rate streams that could be processed by these single NPs that are already designed for lower rate streams (Fig. 3).
2. The DSD-based approach makes the router design scalable. Adopting a new standard with higher line rates can still be handled by this approach if more single NPs could be added to the parallel NPs. Such scalability will be feasible only if the stream DSD is fast

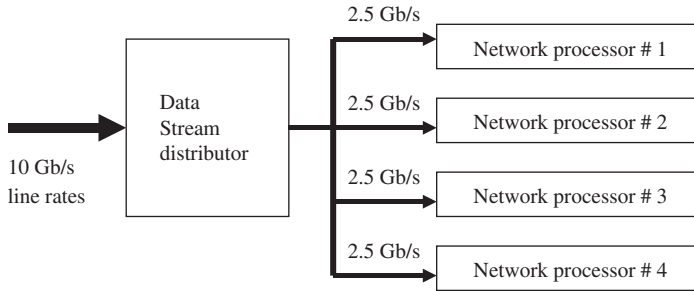


Fig. 3. Stream distributor approach (an example for 10 Gb/s line rates).

enough to handle the new line rates. However, if the DSD is not fast enough, then the redesign process will require some changes to enhance the DSD without changing the design of the NP itself. One of the challenges of the DSD design is to make it simple and very fast.

3. Using DSD will improve the robustness of routers. Since each processor becomes more related to certain traffic flows, the failure of any single NP inside the parallel NPs structure will affect only that traffic flow. Also, as each single NP is designed as a separate module, any module failure can be easily located and replaced.
4. The packets memory will be distributed over those single NPs located inside the parallel architecture. Since the packets memory will support low speed processors, the memory design will be simple under the use of DSD-approach parallel NPs.
5. As the DSD-based parallel NPs allow each single NP to handle a certain traffic flow (keeping packets with the same sequence to be processed in one specific NP inside the parallel structure), the lookup table can be distributed over many lookup engines. As routers might soon support a half million entry, allowing it to handle portion of these entries by each DSD-based parallel NPs will permit the routers to easily support more entries without increasing the router design complexity.
6. The use of DSD-based approach will have a significant impact on reducing the cost of development and the time-to-market for the new generation routers.
7. As different types of packets traffic could arrive to the router such as IP or ATM, the DSD could forward the traffic to different NPs that are optimized for handling that specific flow. Thus heterogeneous processors could be used instead of deploying the same processor for all types of data flow. Clearly, this approach of handling different packets flow will allow the usage of specialized and therefore efficient processors designed for specific data flow.

4. DSD design challenges

The DSD-based parallel NPs will face many design challenges. These challenges can be outlined as follows:

1. Newly arrived packets in a given data flow should always be assigned to the same NP inside the parallel processors structure. As a result, thousands of specific IP addresses will be allocated to each processor and any arrived packet belonging to the same

addresses group will be forwarded to and processed by the same NP. Such a packet distribution scheme can guarantee that packets will not be distributed randomly to different NPs. Failing to do so would increase processing complexity where packets for the same data flow will be distributed over different NPs.

2. The load balancing is required for evenly distributing packet loads between each processor inside the DSD-based parallel NPs. The volume of the packets delivered to each processor has to be continuously evaluated. The results of the evaluation are used by the load-balancing scheme to determine the actual processor load. The next new flow session should be forwarded to the processor with a minimum load.
3. The DSD design should be simple and scalable to guarantee its support for high line rates. It should be easily redesigned in the case of the use of new high line rate interface standard, which were not anticipated at the time that such distributor developed. In addition, the DSD will increase the silicon area of the router implementation and that will increase the power consumption of the DSD chip. However, the use of low power technology could elevate this power consumption problem.

5. General description of the DSD

The architecture of the DSD unit has four main components: serial to parallel converter, processing core, load-balancing unit, and buffers (Fig. 4). The wire-speed serial to parallel converter is designed to get the serial data from the communication network and convert it to parallel format. The parallel data will be forwarded to other components of the DSD, specifically to the input buffer and the processing core.

The function of the processing unit is to extract the header of the received packets to get the required information that is needed for packets processing such as the destination IP address and packet type and size. The destination IP address is necessary to decide which NP that a particular packet is to be delivered to. This can be achieved by using the Destination IP address to reference one of the entries of the lookup, which has the values of the NP identifier that shall decide the NP that will be used to process the packet.

The DSD has an input buffer to store the received packets from the serial to parallel converter while the packet header is delivered to the processing unit to conduct all processing required by the header. When the processing unit completes the header processing in addition to the information provided by the load-balancing unit, the packet will then be forward from the input buffer to one of the output buffers that are associated with the NPs. The load-balancing unit will keep the load that is associated with every output buffers as equal as possible by continuously checking the status of the output buffers. The operation of the load-balancing unit is based on an appropriate algorithm that will be discussed later.

6. DSD architecture and load balancing

The architecture of the DSD is shown in Fig. 5. Although this architecture supports four NPs, extending the DSD to handle more processors is also possible; however, we limited the number of processors to simplify the architecture. The parallel data provided by the serial to parallel converter is 160-bit wide. This size of data format provides reasonable space to hold a minimum packet size without increasing the complexity of the design. For

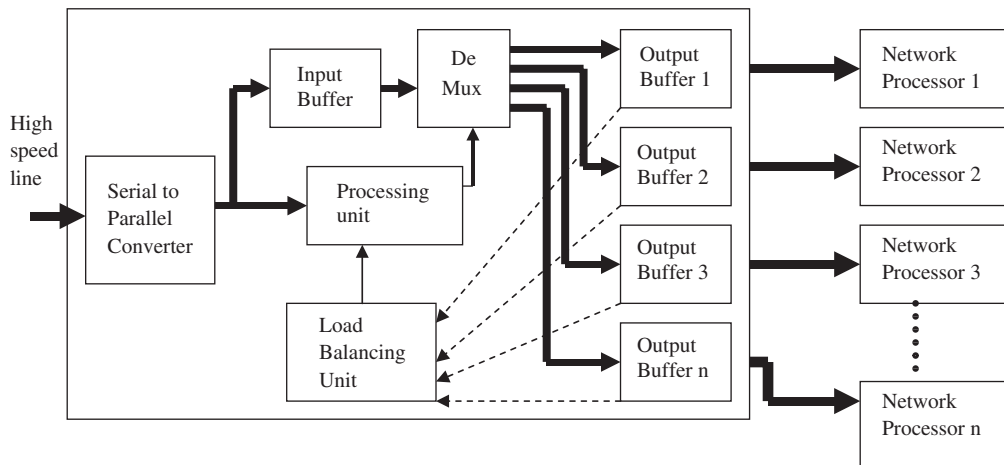


Fig. 4. Overview of the DSD.

example the TCP/IP ACK, which is in the range of 40 bytes can be delivered in two consecutive data format of 160-bit. Longer packets can be supported with multiple 160-bit data format. Also, the 160-bit parallel data format will provide about 16ns to process a packet by different units of the DSD, which provide reasonable time for DSD's units to perform their processing. Increasing/decreasing such time can be achieved by making the parallel data format wider/shorter than 160-bit.

As the 32-bit IP address is used in this work, the lookup table of 2^{32} entries will result with a large lookup table that is not practical to implement. Hence we proposed to pass the Destination IP address through a hashing function, which would provide a hashed output of less than 32 bits. That is, the lookup table will be smaller and therefore, easy to implement. The DSD design uses simple hashing scheme consisting of EX-ORs logic (Tanenbaum, 2001).

The lookup table contains 3 bits at every location; 2 bits are used to identify the NP that will be used to process the received packet. The 3rd bit is used as a flag that is required for each lookup table entry, which shall be used to identify whether that particular entry of the lookup table has been accessed before or not. When the reference to the lookup table indicates that the entry is not used before, the load-balancing unit, upon checking the status of the NPs' FIFOs, will set the identity of the NP that will process the received packet in the look up table. At the same time the load-balancing unit delivers the identity of the NP to the control FIFO and saves it in the next available location in that FIFO. That is, when a portion of the data packet is saved in the 160-bit entry of the input FIFO, the NP identity corresponds to that input FIFO entry will be saved in a location available at the control FIFO. The control FIFO is 2-bit wide that stores the identity of the NP that will control the De-multi-plexer unit (Demux) in delivering each 160-bit to the appropriate NP.

If the destination IP address of the newly arrived packet is referenced to an entry in the lookup table in which the flag is set to '0', it would mean that IP address never allocated a particular NP. Then the load-balancing unit would assign a NP identity to that destination IP address and also write that NP's Identity into the lookup table. The selection of the NP

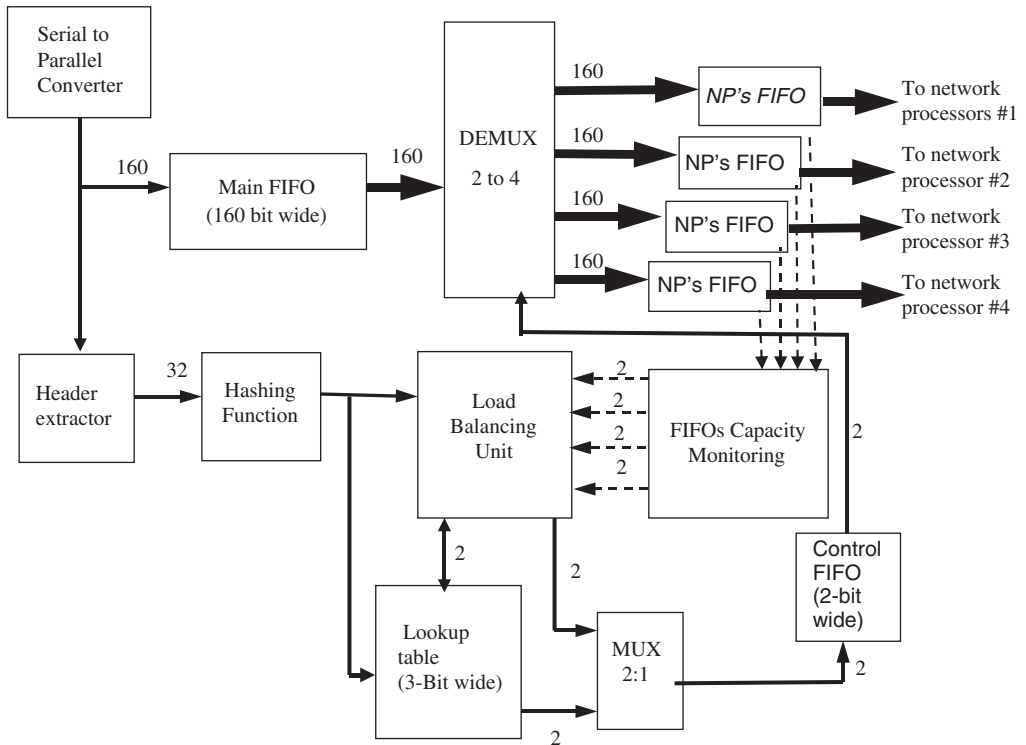


Fig. 5. DSD architecture.

will process that specific destination IP address is based on the load on each NPs' FIFOs at that point in time. The next lookups of that IP address would result in the packet going to the same NP.

The NPs' FIFO sends status signals to the capacity monitor unit. This unit will check the size of the space inside the NPs' FIFOs that became full. The capacity monitor unit will then inform the load-balancing unit whether each FIFO is 50% full, 50% to 75% full, or above 75% full. Such information will help the load-balancing unit to take an appropriate action to redistribute packets as evenly as possible over the NPs' FIFOs. We have used the Round-Robin algorithm with multiple priority levels (Kateeb et al., 2003) to balance loads over processors' FIFOs. In a multi-level Round Robin algorithm, there are different levels of availability (what we shall call eligibility) that are associated with the operation of every NP. If one of the NPs is experiencing a heavier load than the others, then it is less available. If any of them has a lighter load, then it is more available (eligible to receive new incoming packets). Two Flags signals delivered from the capacity monitor unit to load-balancing unit are used to indicate the availability of each NP FIFO. These flags will be raised to indicate when the FIFO has been filled to a particular level of its capacity. We have created 3 levels. "Level 0" (lower level—no flags set) is when that particular NP is experiencing very little load (NP's FIFO filled with less than 50%). "Level 3" (higher level—FIFO filled with above 75%) is when that NP has experienced a heavy load. The

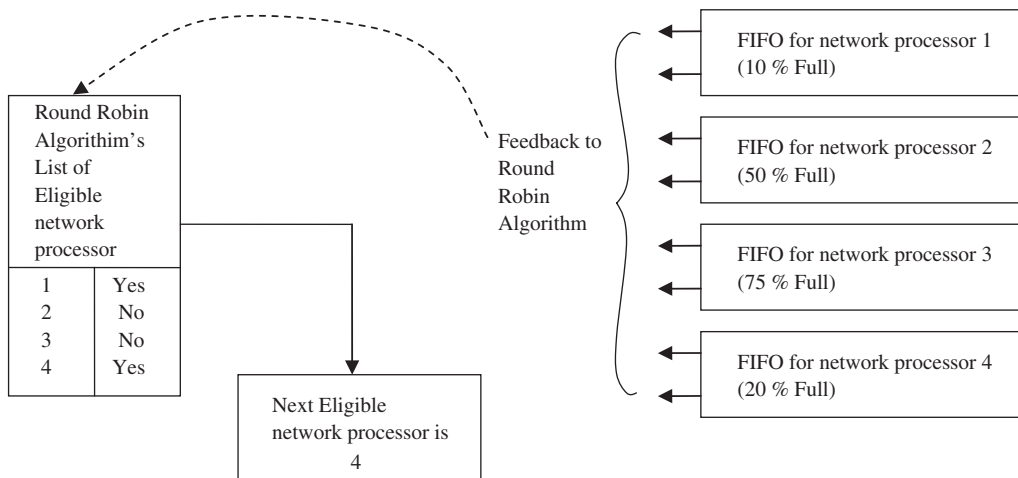


Fig. 6. Round robin algorithm with multiple priority levels.

new incoming packets are to be distributed amongst the NPs which are at the lowest “level” of load in a cyclic manner.

The Round Robin algorithm with multiple priority levels distribute the incoming packets with ‘new’ destination IP addresses evenly among all ‘eligible’ NPs. Suppose at one point the FIFO of any particular NP raises one of its feedback flags indicating that ‘level 1’ has been reached (and at least one of the remaining NPs does not have its flag raised). This makes that NP ‘ineligible’, that is no ‘new’ entries will be assigned to this ‘ineligible’ NP till it becomes ‘eligible’ again.

That particular NP would become ‘eligible’ again if either that queue reduces below ‘level 1’ or all the other NPs’ FIFO queues’ also raise ‘level 1’ flags. The same procedure is followed for ‘levels 2 and 3’ flags. Thus by using feedback from the FIFO queues we introduce priority levels in the round Robin algorithm such that the NPs with the lowest number of flags set are given the highest priority for assignment of a ‘new’ destination IP address. Fig. 6 shows an example of the multi-level round robin scheme operation.

7. DSD Implementation

The DSD design has been implemented over a reconfigurable Field Programmable Gate Array (FPGA) using Xilinx Virtex System-On-Chip (SOC) device. Although the FPGA-based implementations cannot provide the required high performance that is usually produced with ASIC or custom-made chip, we decided to use FPGA-SOC devices due to their flexibility and short development cycle at a reasonable cost. We have used VHDL in implementing the DSD on FPGA devices and such code could be easily synthesized for ASIC or other technology later on to get a very high-performance DSD chip.

In order to achieve this implementation in short period of time and within the allocated budget, we decided to optimize this implementation by reducing the size of the FIFOs, and the lookup tables to 256 entries. Clearly, the DSD would use larger FIFOs and lookup tables when it works in real networks. However, the optimized implementation will

Table 1
Implementation of various components

| Name of component | Number of gates | Speed of component (MHz) |
|--------------------------|-----------------|--------------------------|
| FIFO (160-bit wide) | 20,674 | 71 |
| FIFO (2-bit wide) | 450 | 183 |
| Header extractor | 1579 | 86 |
| Hashing function | 48 | 112 |
| Lookup table | 6804 | 87 |
| Load balancing algorithm | 294 | 120 |

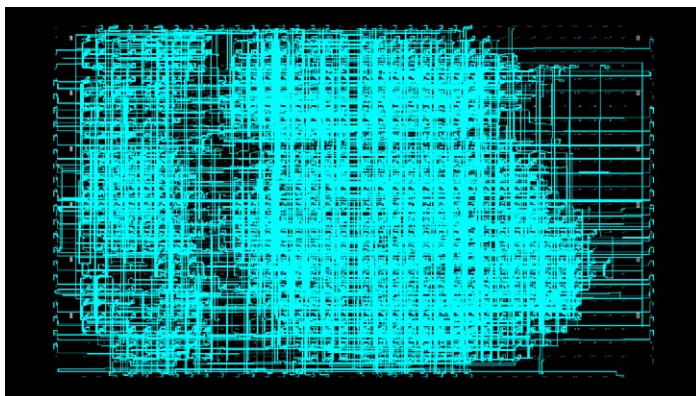


Fig. 7. DSD implementation on FPGA Xilinx XCV 150.

provide accurate results to test the functionality of the DSD architecture and its performance.

Table 1 shows the speed of the main components that are used in the implementation of DSD. The components are integrated together to implement the final DSD architecture. The cycle-accurate timing simulation has shown that DSD could support a wire speed of 2.5 Gb/s. Also, it supports 622 Mb/s data rate for the NPs' FIFOs. Such performance was achieved when the chips were running at 70 MHz. The implementation of the final chip is shown in Fig. 7.

8. Conclusions

The DSD approach is significant in designing and implementing high-speed routers. This approach could be used to develop routers in a short time and at low cost. The challenges associated with this approach of routers design such as the load balancing, buffer managing, and traffic distribution has been investigated in this paper. The DSD architecture and design has shown that this approach is feasible and simple. The DSD implementation is achieved on Xilinx SOC using Virtex XCV 150 chip. The cycle-accurate simulation showed that a 2.5 Gbps line rate can be split to four 622 Mbps low rate lines. The NPs that are used for 622 Mbps lines can be still used with 2.5 Gbps lines without designing a new processor for the 2.5 Gbps line rate. The DSD approach can also be used

for higher and newer line rate standards such as 10 and 40 Gbps without developing new NPs, which is time consuming and costly.

References

- Building next generation network processors, Agere Inc, White paper, September 1999.
- Bux W, Denzel WE, Engbersen T, Herkersdorf A, Luijten RP. Technologies and building blocks for fast packet forwarding. *IEEE Commun Mag*, January 2001.
- Geppert L. The new chips on the block. *IEEE Spectr* 2001;January:66–8.
- Kateeb AE, Richardson P, Gadde M. A data stream distributor chip for high-speed routers design: a reconfigurable approach. *Cool Chips*. Japan, April 2003.
- O'Connor M, Gomez CA. The iFlow address processor. *IEEE Micro* 2001;21(2):16–23.
- Ruiz-Sánchez MA. Survey and taxonomy of IP address lookup algorithms. In: Ruiz-Sánchez MA, Biersack EW, Dabbous W, editors. *IEEE network*, vol. 15(2); March/April 2001. p. 8–23.
- Shah D, Gupta P. Fast updating algorithms for TCAMs. *IEEE Micro* 2001;21(1):36–47.
- Tanenbaum AS, editor. *Modern operating systems*. 2nd ed. Englewood Cliffs, NJ: Prentice Hall; 2001.
- Wolf T, Turner J. Design issues for high performance active routers. *Proceedings of the 2000 international Zurich seminar*. Zurich, Switzerland: Broadband Communications; February 2000. p. 199–205.