Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 12, December 2013, pg.303 – 307

RESEARCH ARTICLE

IMPLEMENTATION OF HARDWARE IP ROUTER BASED ON VLSI

B. Tejasree Yadav
M. Tech (VLSI)
GRIET
ytejasree54@gmail.com
India

Abstract: Through this paper our attempt is to give a onetime networking solution by the means of merging the VLSI field with the networking field as now a days the router is the key player in networking domain so the focus remains on that itself to get a good control over the network, Networking router today are with minimum pins and to enhance the network we go for the bridging loops which effect the latency and security concerns. The other is of multiple protocols being used in the industry today. Through this paper the attempt is to overcome the security and latency issues with protocol switching technique embedded in the router engine itself. This paper is based on the hardware coding which will give a great impact on the latency issue as the hardware itself will be designed according to the need. In this paper our attempt is to provide a multipurpose networking router by means of Verilog code, by this we can maintain the same switching speed with more secured way of approach we have even the packet storage buffer on chip being generated by code in our design in the so we call this as the self-independent router called as the VLSI Based router. This paper has the main focus on the implementation of hardware IP router. The approach here is that router will process multiple incoming IP packets with different versions of protocols simultaneously and even it is going to hold true for the IPv4 as well as for IPv6. With the approach of increasing switching speed of a routing per packet for both the current trend protocols. This paper thus is going to be a revolutionary enhancement in the domain of networking.

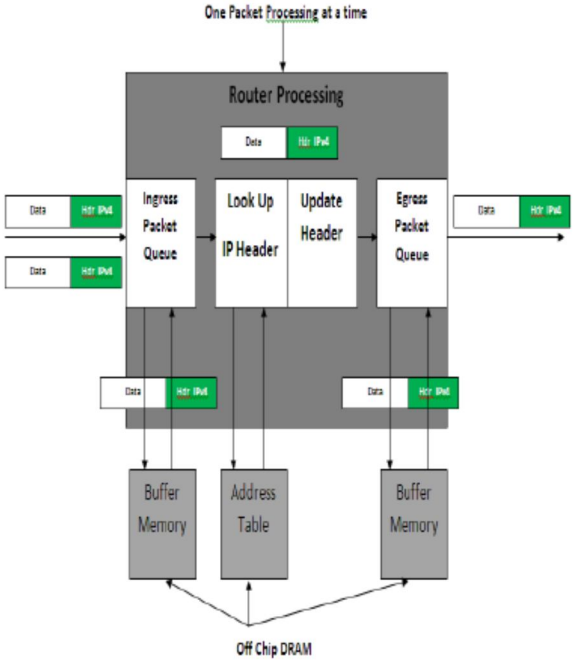
Keywords: IPv4; VLSI; VLSI Based router

I. INTRODUCTION

The popularity of the Internet has caused the traffic on the Internet to grow drastically every year for the last several years. It has also spurred the emergence of many Internet Service Providers (ISPs).Our approach here is to design a variable hardware router code by using Verilog and the same to be implemented for the SOC (System On Chip) level router. In this paper we are making a VLSI design for the implementation at the synthesizable level the same can be further enhanced to SOC level, but our main aim is limited to the NetList generation level which would give

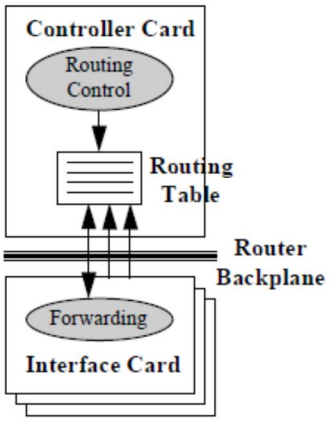
the result prediction and workable module vision. Our focus being in this is to make this router as much variable as we can which will give the robustness for the design to be called even as a Robust Router in which we can make the same router to not only go for N number of connections but also to detect all variety of packets and route the same.

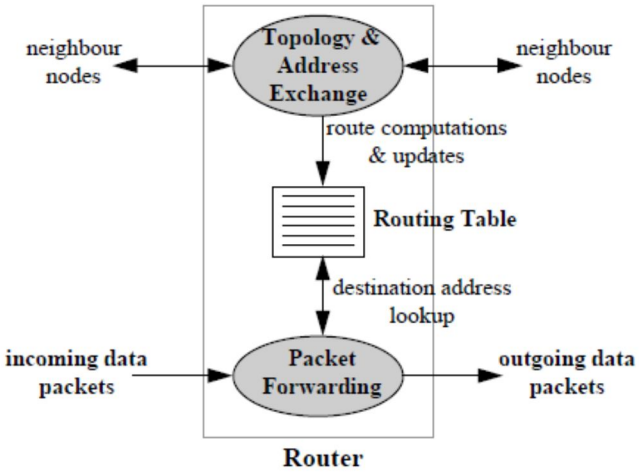
A. Generic Router Architecture



II. BASIC IP ROUTER FUNCTIONALITIES

Generally, routers consist of the following basic components: several network interfaces to the attached networks, processing module(s), buffering module(s), and an internal interconnection unit (or switch fabric). Typically, packets are received at an inbound network interface, processed by the processing module and, possibly, stored in the buffering module. Then, they are forwarded through the internal interconnection unit to the outbound interface that transmits them on the next hop on the journey to their final destination. The aggregate packet rate of all attached network interfaces needs to be processed, buffered and relayed. Therefore, the processing and memory modules may be replicated either fully or partially on the network interfaces to allow for concurrent operations. A generic architecture of an IP router is given in Figure 1. Figure 1a shows the basic architecture of a typical router: the controller card (which holds the CPU), the router backplane, and interface cards. The CPU in the router typically performs such functions as path computations, routing table maintenance, and reach ability propagation. It runs which ever routing protocols are needed in the router. The interface cards consist of adapters that perform inbound and outbound packet forwarding (and may even cache routing table entries or have extensive packet processing capabilities). The router backplane is responsible for transferring packets between the cards. The basic functionalities in an IP 4 router can be categorized as: route processing, packet forwarding, and router special services. The two key functionalities are route processing (i.e., path computation, routing table maintenance, and reachability propagation) and packet forwarding (see Figure 1b). We discuss the three functionalities in more detail below.





a) Basic architecture **b) Router Components**

III. IP PACKET VALIDATION

The router must check that the received packet is properly formed for the protocol before it proceeds with protocol processing. This involves checking the version number, checking the header length field (also needed to determine whether any options are present in the packet), and calculating the header checksum

A Destination IP Address Parsing and Table Lookup:

The router performs a table lookup to determine the output port onto which to direct the packet and the next hop to which to send the packet along this route. This is based on the destination IP address in the received packet and the subnet mask(s) of the associated table entries. The result of this lookup could imply: A local delivery (that is, the destination address is one of the router's local addresses and the packet is locally delivered). A unicast delivery to a single output port, either to a next-hop router or to the ultimate destination station (in the case of a direct connection to the destination network).A multicast delivery to a set of output ports that depends on the router's knowledge of multicast group membership. The router must also determine the mapping of the destination network address to the data link address for the output port (address resolution or ARP). This can be done either as a separate step or integrated as part of the routing lookup.

B Packet Lifetime Control

The router adjusts the time-to-live (TTL) field in the packet used to prevent packets from circulating endlessly throughout the internetwork. A packet being delivered to a local address within the router is acceptable if it has any positive value of TTL. A packet being routed to output ports has its TTL value decremented as appropriate and then is rechecked to determine if it has any life before it is actually forwarded. A packet whose lifetime is exceeded is discarded by the router (and may cause an error message to be generated to the original sender).

C Checksum Calculation

The IP header checksum must be recalculated due to the change in the TTL field. Fortunately, the checksum algorithm employed (a 16-bit one's complement addition of the header fields) is both commutative and associative, thereby allowing simple, differential recomputation. RFC 1071 [8] 6 contains implementation techniques for computing the IP checksum. Since a router often changes only the TTL field (decrementing it by 1), a router can

incrementally update the checksum when it forwards a received packet, instead of calculating the checksum over the entire IP header again. RFC 1141 [9] describes an efficient way to do this. IP packets might also have to be fragmented to fit within the Maximum Transmission Unit (MTU) specified for the outgoing network interface. Fragmentation, however, can affect performance adversely [10] but now that IP MTU discovery is prevalent [11], fragmentation should be rare.

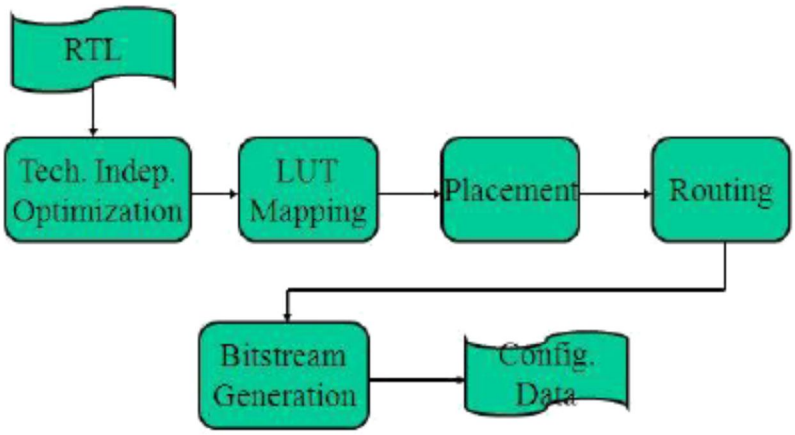
IV. SYSTEM FLOW DIAGRAM

The system flow diagram is as shown below which makes us to understand the flow of the signals through the system from each block by block and transaction carried between the blocks to accomplish the task of the robust router. The flow diagram described here is a brief one, which helps us to understand the flow of every block. Every block have the state machine cycle included in them to enhance the system logical transaction to the level of parallelism. First the packet is received from the ingress channel ring to the input interface block the packet is parsed to data packet and header packet, the data packet is stored in the parser queue and the header is sent to the filer block. The flier block then checks weather the packet is IPv4 or IPv6 and accordingly send the request to the filer table to router the packet to required destination. The filer table cross verifies the egress ring channel with it Dest-IP address and send the egress ring ID to the filer block. The filer block send and enables the particular egress ring in egress blocks and gives the command to the particular egress ring in egress block. Then in egress block the stored data packet in the parser queue is added back with header and is sent out with the specified egress ring channel. In this way the every packet is processed and routed in robust router.

V. SIMULATION AND DISCUSSION

A Net List of the Robust Router

The Net List is RTL level of the robust router system, which is syntasizable and can be extracted on the Xilinx tool. By which we can get preface look of the system and a transition from the frontend of the VLSI designing to backend of the VLSI designing. Which means the same can run on FPGA kit and test its robustness and errors of the system can be debugged before it is taken to SOC Level and to Fab-Labs. The snap below is the Pin configuration of the proposed Robust Router.

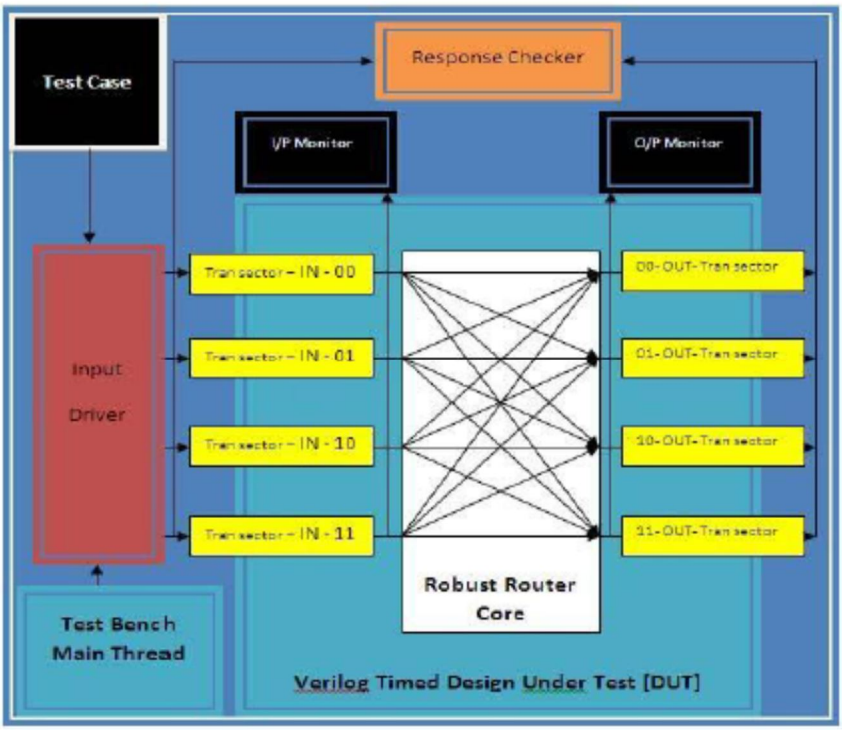


The RTL level of design which we get from the Net List of the system will have gate delay, propagation delay and wire delays included in them. These are all calculated and made into an optimization level. Then the design is fixed into LUT'S and the mapped between the LUT'S further the placement of the LUT'S are prissily done keeping mind the power utilization and the delay calculated earlier. Then the routing is done between the CLB'S. Further the bit-stream is generated to test the system and verification done across the Net List output to get the exact design. Then the system design is masked and made to the GDSSI Level further to be sent on to the Fab-Labs for fabrication.

B Design under Test

The design under test [DUT] is made to test the system robustness under different cases. The DUT architecture includes the Test case which will define the test. The input driver block will generate the test input signals for the system testing. The input and output transacted will make the system get the input and output according to the

system core requirement. The input and out monitor are placed to compare the system testing. At last the Response checker is to give the system testing pass or fail. The DUT is as shown below.



VI. CONCLUSIONS

In this paper the code given in the output for IPv4 and IPv6 packets is put in the router at the same time. The Robust Router will route both packets at the same time at the same speed. The Robustness is simulated with Model-Sim Tool with different Test Cases and the same code's NetList is extracted with Xilinx Tool for the synthesizable code. The same can be taken to the SOC (System on chip) level with Cadences Encounter Tool. The same Verilog code design can be taken to the implementation of MPLS (Multi-Protocol Label Switching). The some code design can be taken to the SOC (System on chip) level and can be implemented as the Ethernet Standalone System Router. The same code can be made variable with TCP and UDP Protocols.

REFERENCES

1. Abromovici, M. Breuer Digital System Testing and Testable Design.

2. Charles, H. Roth Jr Digital System Design Using VHDL

3. James, Balfour and William, J. Dally. (2006). Design tradeoffs for tiled cmp on-chip networks. In ICS '06: Proceedings of the 20th annual international conference on Supercomputing, pages 187–198.

4. Jenkins, Jesse H. Designing with FPGAs and CPLDs

5. Moy, J. (1998). OSPF: Anatomy of an Internet Routing Protocol.

6. Tobias, Bjerregaard and Shankar, Mahadevan (2006). A survey of research and practices of network-on-chip. ACM Comput. Surv., 38(1):1, 2006.

