Implementation of PNoC and Fault Detection on FPGA

Preethi T S[1], Nagaraj P[2], Siva Yellampalli[3]

Department of Electronics and Communication, VTU Extension Centre, UTL Technologies Ltd.

Abstract –**In this project, it describe the implementation of a new circuit-switched NoC designed specifically for FPGA-based systems. The flexible and lightweight advantages of this architecture are explained to quantify its area and performance benefits. The main motivation for choosing a circuit-switched network over packet-switched is its ability to maintain guaranteed throughput between nodes connected via virtual circuit. This technique is in direct contrast to packet-switched techniques, where significant variations in communication latency are often possible. This ability to give reliable high data rates between the nodes in a system that most need it was critical in this design decision. The circuit-switched networks are attractive option, which promises the high communication rates and predictable communication latencies. Here a new lightweight circuit-switched architecture called programmable NoC (PNoC) is described. In Fault detection, it identifies the faults in a router when detected, alternate routing path will be selected for successful transmission of data.**

Index Terms – PNoC, Circuit switching, router, floating buffer, fault detection.

I. INTRODUCTION

Multiprocessor architectures have been introduced to extend the applicability of Moore's law as they depend on concurrency and synchronization in both software and hardware to enhance the design productivity and system performance.

These platforms will also have to incorporate highly scalable, predictable, reusable, cost and also energy-efficient architectures. With the rapidly increasing billion transistors era, there are some problems in deep sub-micron technologies which are characterized by gate lengths that will arise from non-scalable wire delays, errors in signal integrity and also in unsynchronized communications. By using Network on Chip (NoC) architecture, these problems may be overcome. NoC can improve design productivity by supporting modularity and reuse of complex cores. Therefore NoC enables a higher level of abstraction in the architectural modeling of future systems.

II. LITERATURE SURVEY

Network on Chip (NoC) architectures provide a very efficient means for performance enhancement in digital circuits. In [1], the implementation of NoC that is specifically targeted towards FPGA based designs are described. Researchers have well addressed NoC architectures and hardware-related issues in [2]. In order to facilitate the process of porting a traditional hardware implementation to a NoC based implementation, a very simple circuit-switched architecture called programmable NoC (PNoC) was proposed in [3]. PNoC is a lightweight and flexible architecture for FPGA based systems. As in [4], a higher communication bandwidth and better scalability are the foremost merits of PNoC. However, the architecture that was developed in [4] was given in JHDL, which is not a commonly used hardware description language (HDL). In the paper [4], the authors had overcome this problem by presenting a Verilog (a more commonly used HDL) implementation of the PNoC architecture. Besides the implementation details, the paper also presents a detailed analysis of our model and the results have been found to be consistent with the JHDL based implementation proposed in [4]. In [5], it uses a modular design that facilities the usage of standard interfaces, IPs.

III. PROBLEM FORMULATION

Network-on-chip designs promise to offer considerable advantages over the traditional bus-based architecture. As continuing scaling of Moore's law enables ever greater transistor densities, design complexity, power limitations and application convergence networks have started to replace busses in much smaller systems and the enhancement of NoC. The disadvantage of busses is that they do not decouple the activities generally classified as transaction, transport and physical layer behaviours. This is the main reason that busses cannot adapt to changes in the system architecture or take advantage of the rapid advancement in the silicon process technology.

NoC technology is very rapidly being adopted in SoC designs of all levels of complexity because true NoC implementations of SoC interconnects are consistently proving to be significantly faster, smaller and also more power-efficient than traditional or decoupled busses and crossbars.Networks on chip (NoCs) promise to overcome the scalability problems found in bus-based interconnect. Nowadays, most work has focused on packet-switched NoCs. The circuit-switched networks are an interesting option, which promises high communication rates and predictable communication latencies.

A new lightweight circuit-switched architecture called programmable NoC (PNoC) is described which is a flexible architecture that is suitable for use in FPGA-based systems.In this paper, two problems are solved. One is synchronization problem if two communicating routers are of different speed then if size of the buffer is not sufficient it overflows and data is lost. The solution to this problem is to use a floating buffer and allocating this floating buffer to any output buffer when there is a overflow of data in that particular output buffer. This method achieves efficiency without increase in area of the design, since only one floating buffer is used. When there is multiple request arbiter is used to resolve the problem. Secondly, if any of the node goes faulty, then the packet is routed in alternate path to reach the destination successfully.

IV. BLOCK DIAGRAM OF PNOC

PNoC is a circuit switched technique that simplifies system design by providing flexible networking approach and is efficient in design and verification methodologies. Network consists of subnets and each subnet has a router and a bunch of network nodes shown in fig.1.

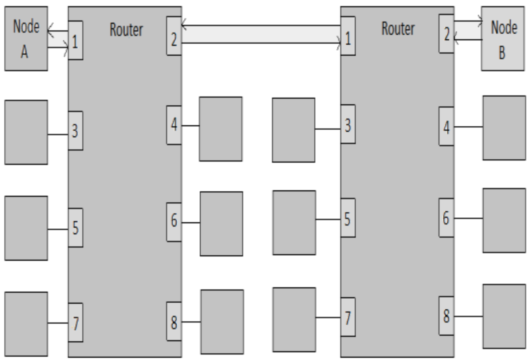


Fig. 1 PNoC Network Topology.

A dedicated connection path is used for data transfer so no acknowledge signal is required. Data transaction can occur on successive clock cycles if master receive is low, the read and write requests can be pipelined and a CPU is connected to the PNoC like any other module. The interfacing circuit constitutes FIFO's and FSM to communicate with the router. The router is the main component of the PNoC. The router includes the routing table, queue and a switch box. Another important part of the PNoC is the buffer which has a parameterizable feature. It is necessary in two cases- firstly, if nodes and routers are running at different clock rates, and secondly, when there is a difference between the transmitting and receiving rate.

V. ADAPTIVE ROUTER WITH BUFFER RESIZE

The router is the core of this network communication architecture. The major components that make up the PNoC router are shown in the block diagram of fig.2.
The purpose of each of these components is described as follows:

1. Arbiter: The arbiter receives connection requests and schedules access to the routing table in the case that multiple requests are received on the same clock cycle. This block is also responsible for managing the routing table update requests.
2. Crossbar switch: The crossbar switch forms the actual connections between modules by enabling tri-state buffers that allow the receive signals to drive the appropriate transmit signals.
3. Buffers: They are used to store the incoming data, until it gets the output port.
4. Floating Buffers: They are used to store the incoming data when the buffer is full. It is allotted to any one output based on logic of floating buffer controller during the time of congestion control.
5. Floating Buffer Controller: This selects floating buffer to one of the output during the time of congestion control.

For the dynamic distribution of buffering resources, we propose the use of floating buffers that can be assigned to any output port to increase their buffering size. With extra buffers it is possible to reduce the size of the fixed buffers to a minimum (e.g., 16 words) and then dynamically compensate for the lack of buffering by assigning the floating buffers to the output ports largely congested. The router will have a structure which is similar to the static router but the adaptive router includes a few extra modules to control the floating buffers and to dynamically put them in the data path of the router. Architecture shown in the fig.2 has a single floating buffer that can be associated with any output port, except with the local port. This port has not been considered since we assume the processing element connected to the local port is unable to collect data simultaneously from two inputs. The arbiter associated with an output port receives the requests from the input ports and grants access to its buffer. In this case the static buffer is full and the floating buffer is assigned to it then it grants access to the floating FIFO. It returns to the static FIFO if the floating FIFO gets full. The assignment of the floating buffer is made by the floating buffer controller. Several policies can be followed to assign the floating buffer. For a fair assignment, all eligible ports for assignment (those with full static buffers) are chosen in a round-robin manner.

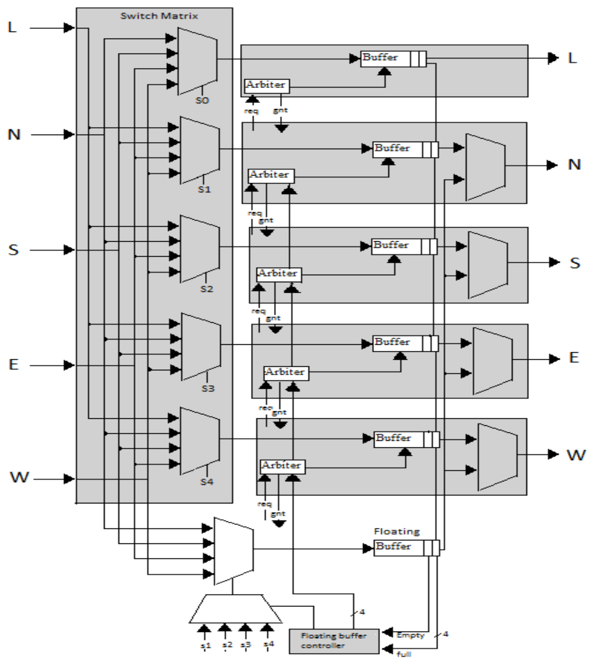


Fig. 2 Architecture of an adaptive router with buffer resize

VI. FAULT DETECTION IN 4*4 MESH

In this technique, it identifies the faults in a router while detect any fault in the path the alternate router will be selected for data transmission to ensure that the data is successfully transmitted. Fault-tolerance or graceful degradation is the property that enables a system (often computer based) to continue operating properly in the event of the failure of (or one or more faults within) some of its component. If the operating quality is reduced, then the reduction is proportional to the severity of the act of failing a breakdown, as compared to a natively designed system in which even a small failure can cause total breakdown. Fault-tolerance is particularly required in high-availability systems.

A. *Schematic of Node of a Mesh*

A NoC router is composed of a number of input ports (connected to shared NoC channels), a number of output ports (connected to possibly other shared channels), a switching matrix connecting the input ports to the output ports, and a local port to access the IP core connected to this router. Here in, we use the terms router and switch as one and the same, but the term switch can also imply the internal switch matrix that actually connects the router inputs to its outputs. In addition to this physical connection infrastructure, the NoC router also contains a logic block that implements the flow control policies (routing, arbiter, etc.) and defines the overall strategy for moving the data through the NoC.

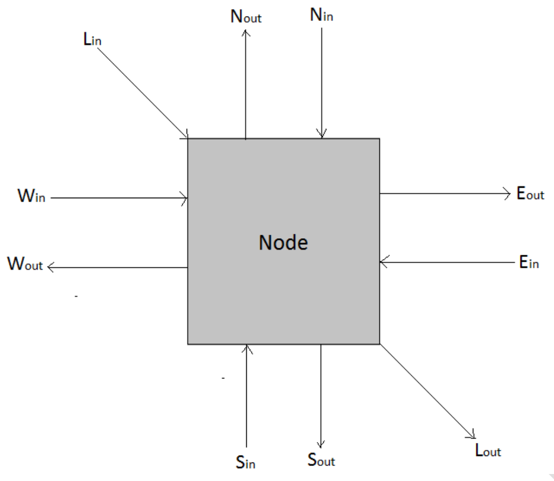


Fig. 3 Schematic of a Node.

B. *A 4*4 Mesh*

2D Mesh is a very popular topology in Network on Chip due to its facilitated implementation, simplicity of the XY routing technique and the network scalability. Due to the fact that every connected device to mesh topology is a node, a mesh topology can resist high amounts of traffic. Also interconnection of many devices at once also means that simultaneous transfers do not hinder or affect the network in anyway. Mesh topologies aren't affected by size or a shortage of users. The internet is constantly growing with more and more devices connecting and information being flooded into the network. The creation of large hub of data and useful information that many users connected to the network can avail. There are 16 nodes in the mesh as shown in figure 4.If any one the node has fault data can be routed to destination from the alternated path using modified XY routing technique. The route of data from node3 to destination node14 is shown in dotted lines in fig.4 when node6 and node11 has fault. Node3 data is routed through E_{out} of node3 to E_{out} of node7 to S_{out} of node11 to E_{out} of node12 to N_{out} of node16 to N_{out} of node15 to d_{out} of node14.

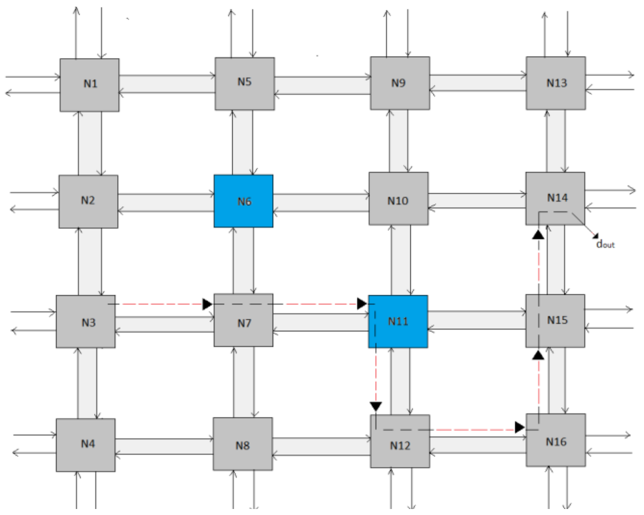
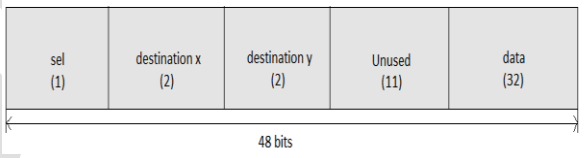


Fig. 4 Schematic of a 4*4 Mesh with fault in Node6 and Node11.

C. *Format of a Packet*

The 32-bit data value can be sent between source and destination Processing Element. For the router packet control, the address of destination PE is represented by relative distance of horizontal (X-direction) and vertical (Y-direction) direction in magnitude values. The 11 bit unused field is reserved for future purpose, in case of the extension of the network. The 1st bit is reserved for select input for forwarding data from one router to other.



sel (1)	destination x (2)	destination y (2)	Unused (11)	data (32)

Fig. 5 Packet Format.

D. *Modified XY Routing Algorithm*

The classic XY routing has the disadvantage of traffic is loaded at the centre of the network, but cannot distribute the traffic all over the network. In this modified XY routing is proposed along with priority encoder is used to route the packet from source to destination. The select(sel) bit as shown in fig.5 is set to 1, in order to route the packet from one router to another. In XY routing it routes the packets in x-direction first i.e., in horizontal direction then in y direction i.e., in vertical direction. For routing the packet, each router has got (x,y) co-ordinates. The current address of the router has (c_x,c_y) and destination address of the router has (d_x,d_y). In order to route the packet, it compares the current address of the router with destination address of the router as follows:

1. If (d_x = = c_x) and (d_y = = c_y) then output of router is from local out port.
2. If (d_y > c_y) then output of router is from south out port.
3. If (d_y < c_y) then output of router is from north out port.
4. If (d_x > c_x) then output of router is from east out port.

5. If (d_x < c_x) then output of router is from west out port.

VII. Results

The simulation results of Router, PNOC network, fault detection with fault in node 6 and node 11 and fault detection without fault is as shown below:

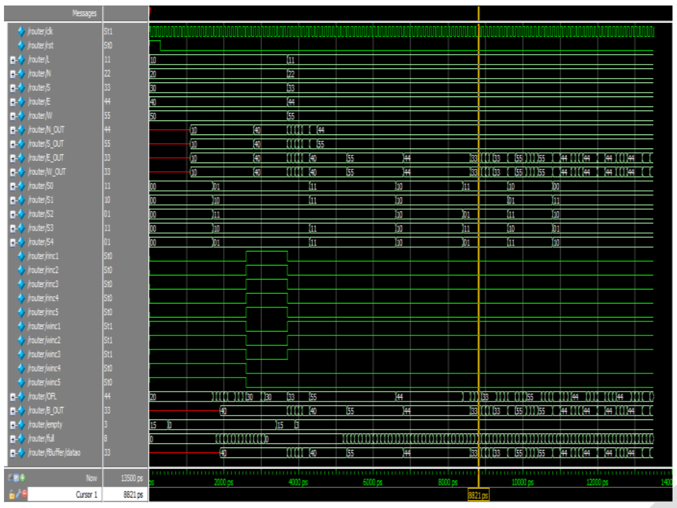


Fig. 6 Simulation result of Router.

In fig.6, based on the select line input (s0,s1,s2,s3,s4) the input from Local, North, South, East and West (L,N,S,E,W) are selected to output N_OUT , S_OUT , E_OUT , W_OUT .When respective read increment (rinc) is high the data is read from respective buffers associated with each input . When respective write increment (winc) is high the data from the input is stored on to the buffer latter it is transferred to the output of the node. If any of the input buffers is full then a request is made to access the floating buffer. If two or more make the request for floating buffer simultaneously then arbiter will assign the floating buffer to any one input based on round robin scheme.

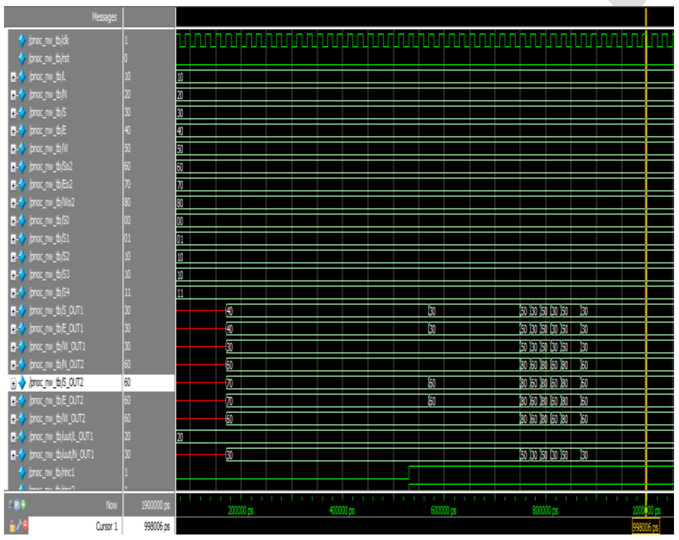


Fig. 7 Simulation result of PNoC.

In fig.7, it shows the simulation result of communication of two routers. The input to the router 1 is shown by L, N, S, E and W. The output L_out, N_out of router1 is fed as input to router 2. The remaining input of router 2 is shown by S02, E02 and w02. Based on the select line input, the output of the router is selected.

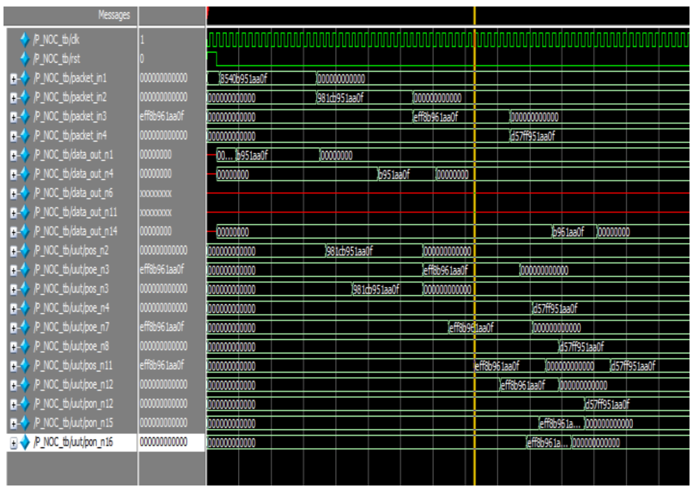


Fig. 8 Simulation result showing fault in node6 and node11.

Fig.8 shows simulation result when node6 and node11 has fault and how data is transferred to destination using modified xy routing. Four packets are sent from node1, node2, node3 and node4 represented by packet_ in1, packet_ in2, packet_ in3, and packet_ in4 respectively. The node1 data has destination as node1 itself. Similarly node2 data has destination as node4, node3 data has destination as node14 and node4 data has destination as node11. Node2 data is routed through S_{out} of node2 to S_{out} of node3 to d_{out} of node4. Node3 data is routed through E_{out} of node3 to E_{out} of node7 to S_{out} of node11 to E_{out} of node12 to N_{out} of node16 to N_{out} of node15 to d_{out} of node14. Node4 data is routed through E_{out} of node4 to E_{out} of node8 to N_{out} of node12 to S_{out} of node11.Since node11 has fault data from node11 is not transferred to d_{out} of node11.The route of data from node3 to destination node14 is shown in dotted lines in fig.4 when node6 and node11 has fault.

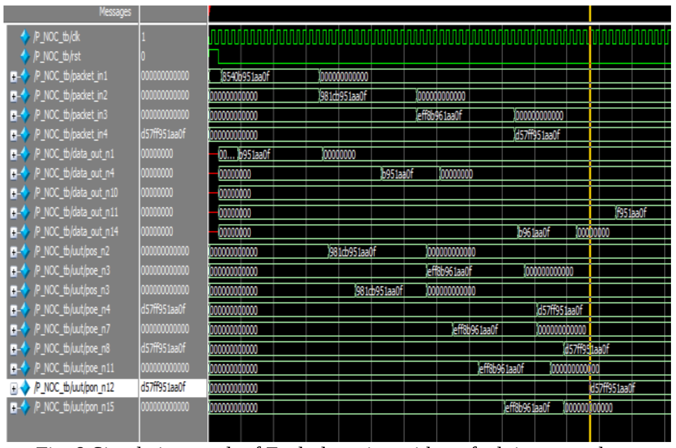


Fig. 9 Simulation result of Fault detection without fault in any node

. Fig.9 shows the result when none of node has any fault. Packets are sent from node1, node2, node3 and node4 represented by packet_ in1, packet_ in2, packet_ in3, and packet_ in4 respectively. The node1 data has destination as node1 itself. Similarly node2 data has destination as node4, node3 data has destination as node14 and node4 data has destination as node11. Node2 data is routed through S_{out} of node2 to S_{out} of node3 to d_{out} of node4. Node3 data is routed through E_{out} of node3 to E_{out} of node7 to E_{out} of node11 to N_{out} of node15 to d_{out} of node14. Node4 data is routed through E_{out} of node4 to E_{out} of node8 to N_{out} of node12 to d_{out} of node11.

VIII CONCLUSION

In this project programmable NOC is implemented by adding a floating buffer to the router without increasing the area of the circuit. The size of buffer can be increased dynamically when overflow occurs using the floating buffer. This technique involves simple logic and effectively increases the performance of the router. The fault detection is used to effectively route the packets in alternate path if any node goes faulty.

REFERENCES

[1] U.Mushtaq, O.Hasan, and F.Awwad, "PNOC: Implementation on Verilog for FPGA",Proc. IEEE Innovations in Information Technology (IIT), 9th International Conference, pp. 148-151, 2013.

[2] A.Agarwal and R. Shankar, "A layered architecture for NOC design methodology", IASTED International Conference on Parallel and Distribute Computing and Systems pp. 659-666, 2005.

[3] L. Benini and G. De Micheli, "Networks on Chip: a New SOC Paradigm", IEEEComputer, volume1, pp. 70-78, 2002.

[4] C.Hilton and B. Nelson, "PNOC: A flexible circuit switched NOC for FPGA-based systems," IEEE proceedings computers and digital techniques, volume 153 Issue 3, 2006.

[5] C.Hilton and B. Nelson, "PNOC: A flexible circuit switched NOC for FPGA-based systems," IEEE proceedings computers and digital techniques, volume 153 Issue 3, 2006.

[6] E. Salminen, V. Lahtinen, K. Kuusilinna, and T. Hamalainen, "Overview of bus-based system-on-chip interconnections," in Proceedings of the IEEE International Symposium on Circuits and Systems. ISCAS 02, pp. 372–375 vol.2, 2002.

[7] Hutchings, B., Bellows, P., Hawkins, J., Hemmert, S., Nelson, B., and Rytting, M., "A CAD suite for highperformance FPGA design", in Pocek, K.L., and Arnold, J.M. (Eds.). Proc. IEEE Workshop on FPGAs for Custom Computing Machines, Napa, CA, USA, pp. 175-184, 1999, (IEEE Computer Society).

[8] J. Dally and B. Towles Principles and Practices of Interconnection Networks, MorganKaufmann, 2004.

[9] A. Ivanov, C. Grecu, M. Jones, P. Pratim Pande, and R. Saleh, P. Pratim Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design tradeoffs for network-on-chip interconnect architectures",IEEE Transactions on Computers, volume54, no. 8, pp. 1025-1040, 2005.

[10] Hosseinabady, M.; Kakoee, M.R.; Mathew, J.; Pradhan, D.K.; , "Low Latency and Energy Efficient Scalable Architecture for Massive NOCs Using Generalized de Bruijn Graph,"IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol.19, no.8, pp.1469-1480, Aug. 2011.

[11] J. W. Dally and B. Towles, "Route packets, not wires:"On-Chip interconnection networks", Proc. IEEE International Conference on Design and Automation, pp. 684- 689, 2001.