Development Of A Flexible Framework for A Network On Chip

Bedwal J, Dewangan D, Garga G, Goel A, Hadlar C, Jain A, Kumar S, Nandy S K, Narayani R, Negi A, Potineni S and Sahu D
Department of Electronic and Systems Engineering
Indian Institute of Science
Bangalore

***Abstract*—We propose the programmable framework for a Network-on-Chip(NoC) based architecture which can be synthesized as a part of SoC. This paper includes the design and implementation of an IDE based framework for realization of a NoC subsystem with many interconnected modules communicating with each other through message passing or the establishment of virtual circuits.**

I. Introduction

Object-based graphical user interfaces (GUIs) based on Linux environment uses an integrated development environment to design and implement the intelligent Design Assistant Agent which offers user inputs and suggestions not only in the course of the design but help them to avoid reinventing well known standard NoC components.

A NoC based architecture is better in terms of bandwidth, scalability, and design complexity with an increase in the number of cores in comparison to the traditional bus architecture. A NoC based architecture is conducive to both communication as well as computation. Thus the correct near-optimal design of the NoC guarantees crucial attributes such as correctness of communication, efficient and guaranteed routing of packets, freedom from dead-locks, and optimal use of silicon resources performing a complex design task by itself that must be repeated for every new silicon design.
In terms of regular topologies, NoC based architecture has many types with different dimensions. In terms of ease of physical mapping and routing patterns, the most common approach is mesh topology with two dimensions using wormhole switching technique.
The remainder of the paper is organized as follows. In Section 2, we describe some related work. In Section 3, we describe the different levels of implementation in the design of a NoC framework. In Section 4, we describe the implementation of a flow control mechanism technique. In Section 5, we describe the implementation of a routing mechanism technique. In Section 6, we describe switching technique implementation. In Section 7, we describe an arbitration implementation. In section we have provided a detailed description of the experiments performed with their results. In Section 8, we draw our conclusion. This is a citation [**?**] This is how you refer a section in another section Section **??**

II. Literature Survey

High performance with optimal power consumption to perform various applications required by the user is a major concern of research. Various proposals for a dynamic adaption of processors, memories and their interconnections are found to provide a more suitable configuration for the system[].With an emerging approach for a NoC based architecture there is a great enhancement on the performance in terms of power, throughput, latency and speed[1][4]. A NoC based architecture provides a reliable and a controlled structure interconnection of nodes[2]. The architecture includes various types of topologies like mesh, ring, triangular, honeycomb with regular as well as irregular interconnected networks. It provides a wide range of possible configurations and topologies on the space wire protocol[3], design automation [10][59][72][74] [45], performance evaluation [75][33][87], encoded signaling[56][43], error detection and correction [67][11], wire optimization [56] [22][70][65], and programming models [77]. Various approaches have been proposed to present a NoC framework with an improvement in accuracy and speed in an emulation environment platform[..]. The main difference in our approach is the application of wide range of combinations in terms of different topologies with different configurations in user development environment.

III. Design and Implementation

Top view of block diagram architecture. //MESH and Router description

A. Graphic User Interface (GUI)

The frontend of the NoC architecture is implemented on the Eclipse IDE framework to develop a Plug-in of type *Multi Page Wizard & Editor*. This allows each user to input settings for the NoC he/she desires. Stage 1 of the plug-in creates a four page wizard to allow the user the flexibility for choosing the required *Switching, Arbitration, Routing and Flow Control Mechanisms*. It also provides the freedom to the user to parameterize elements related to each flit/packet as per the previous settings. Refer to Figure 1.
Stage 2 of the plug-in makes use of the Open Source Project named *Graphiti* **??** to allow the user to view regular and build custom topologies of the NoC. If regular topology is selected, an auto generated graph for the select NoC is displayed else

the tool provides freedom to the user to make an irregular topology as per his needs. Refer to Figure 2.
Stage 3 of the plug-in provides a comprehensive amalgamation of the configuration settings and graph based topology of the network. To complete the generation of the required RTL synthesizable files, the 3rd page provides a *Generate NoC* button which is linked to the NoC backend generator scripts written in *Python* to dynamically generate the required files. Refer to Figure 3.

B. Flowcontrol

A flow control mechanism determines allocation of resources to messages as they traverse network including buffers and links. There is a significant impact on throughput and latency of a network through flow control.

Each router has five ports, which can be used for 2-D tori and meshes. In each port one or two virtual channels are used. In the first configuration the buffer size can be optimized according to one's requirement so occurs the transfer of packets from one router to another which results in the reduction of traffic congestion and eliminates the loss of packets. There is a consistent check on the buffer overflow by the use of a counter. Message sources injected in the form of packets which consist of transfer units known as flits. These flits in a buffer can be accessed by a read and a write pointer for deque and enque operation respectively, similar to a FIFO mechanism. This can be achieved by two major algorithms named as Go and stop[..] and Credit based flow control algorithm[..]
The second configuration deals with two virtual channels which is selected by the routing mechanism. The output packet is then demultiplexed through the arbitration.
In order to achieve the specified goals, we introduced the implementation of a parameterized Verilog models of flow control in NoC-Routers. The depth of input buffers, located at the input ports, and the flit size (channel width), are parameterized by means of verilog generics.

C. Routing

Routing mechanism is targeted to k-ary n-cube topology categorized into deterministic and adaptive algorithms. Deterministic routing algorithms always use the same fixed path for communication between a source-destination pair. Adaptive routing algorithms, on the other hand, can take various conditions into account when determining the path to the destination.Unlike deterministic, adaptability is constrained to area, cost and power consumption. XY Routing and YX Routing are classified under deterministic routing Approach whereas turn Around Routing and virtual channel transition routing comes under Adaptive Routing. West-first, North-last and Negative-first are among the few examples of turn around routing which prohibits the turns that can cause deadlock. The cyclic dependencies can be eliminated through the use of virtual channel mechanism
In the algorithm of routing, an offset is defined which is calculated by the difference between the destination address of the flit and address of the router where the flit is currently present. xOffset and yOffset are the offset for x, y coordinates respectively.

D. Arbitration

In Arbitration, three modules namely *Fixed, Least Recently Used (LRU), Round Robin* have been implemented as a hybrid priority encoder, with the priority list getting updated after each clock cycle by a feedback mechanism. The basic architecture is shown in Figure **??**. Each *Request Array* is a multi bit array with the convention shown in Table I. Each bit when enabled denotes that the corresponding port has polled a request and its service would be attended to as per the priority list logic, based on the arbitration protocol being used. Each combinational logic block is dependent on the protocol being serviced to. Refer to Table II for allocation of grant per algorithm.

L	**S**	**N**	**W**	**E**
1	**1**	**0**	**0**	**0**
[4]	**[3]**	**[2]**	**[1]**	**[0]**

TABLE I
REQUEST ARRAY CONVENTION

Current priority	**Next priority for Fixed**	**Next priority for LRU**	**Next priority for Classical RRB**	**Next priority for Advanced RRB**	**Grant allocation**
West	West	West	East	South	0
East	East	East	Local	North	0
Local	Local	South	South	West	0
South	South	North	North	East	0
North	North	Local	West	Local	1

TABLE II
COMPARISON OF MULTIPLE PROTOCOLS FOR ALLOCATION OF THE GRANT

The classical round robin (RRB) version **??** of static allocation has been upgraded to avoid wastage of clock cycles by allocating the slot if only and if it polls a request.

E. Switching

The switching mechanism is implemented using a multiplexed crossbar network. The inputs to the each port of the router comprises of an input flit and a flit validation signal from the flow control mechanism followed by a validation signal corresponding to the coordinates computed by the routing mechanism and a grant signal generated on the basis of the direction request logic from the arbitration mechanism. Each port works in parallel to achieve a crossbar implementation of communication between the native and the immediate neighbouring routers. Based on the grant signals and the input validation signals from flow control & routing modules, the selection logic to a multiplexed network is enabled to generate the respective En-queue and De-queue signals for each port of the router. These are accepted as control signals by the buffers present at each port of the immediate neighbouring routers.

1) Single Flit Packet Switching: For implementing single flit behaviour, each flit carries a unique destination address which is utilized by routing mechanism to compute its immediate next coordinates and are made available as inputs to the switching module. Since 1 packet is equivalent to 1 flit, *Wormhole Switching* [**?**] is implemented in our design.

2) Multi Flit Packet Switching: For implementing multi flit behaviour, a sequence of 3 types of flits primarily known as *Header*, *Body* and *Tail* are utilized. In the above set sequence, the destination coordinates reside in the *Header* flit. As the *Header* hops across each router of the mesh architecture, it disables a set of signals named *allowHeader* for each port along its path to which it has hopped from. This prevents the entry of another *Header* flit along the same path and makes sure that only *Body* flits and *Tail* flits can traverse along the path used by its corresponding *Header*.

3) Circuit Switching with VC Support: For implementing *Virtual Circuit* [**?**] behaviour, a set of configuration files are generated per port per router. These configurations contain a sequence of cycle based allocation for each port of a router to emulate **Time Division Multiplexing** (TDM)**??** at each node of the mesh architecture. The allocation takes place in synchronization with the *Weighted Round Robin*. Refer section III-Dfor the algorithm .

IV. Experiments

Here is another section for experiments

V. Conclusion and Future Works

Here is the penultimate section

References

[1] Partha Pratim Pande,Cristian Grecu, Michael Jones, Andre Ivanov,Resve Saleh"Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect," *IEEE transaction on Computers*, vol. 54, No. 8, August 2005.

[2] Goossens K., Dielissen J., Radulescu A."thereal network on chip: Concepts, architectures and implementations," *IEEE Design and Test of Computers*, vol 22, 2005.

[3] Ying-Cherng Lan, Michael C. Chen, Alan P. Su, Yu-Hen Hu, Sao-Jie Chen"Flow Maximization for NoC Routing Algorithms"*Proceedings of the 2008 IEEE Computer Society Annual Symposium on VLSI*

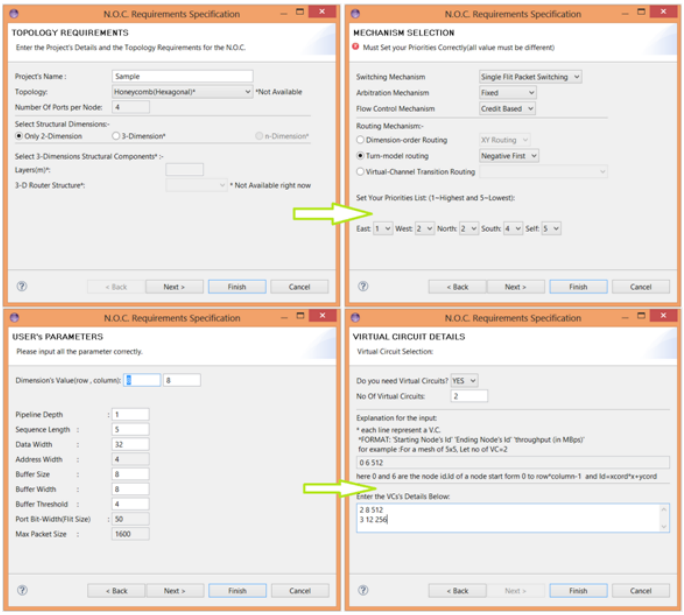

Fig. 1. Stage 1

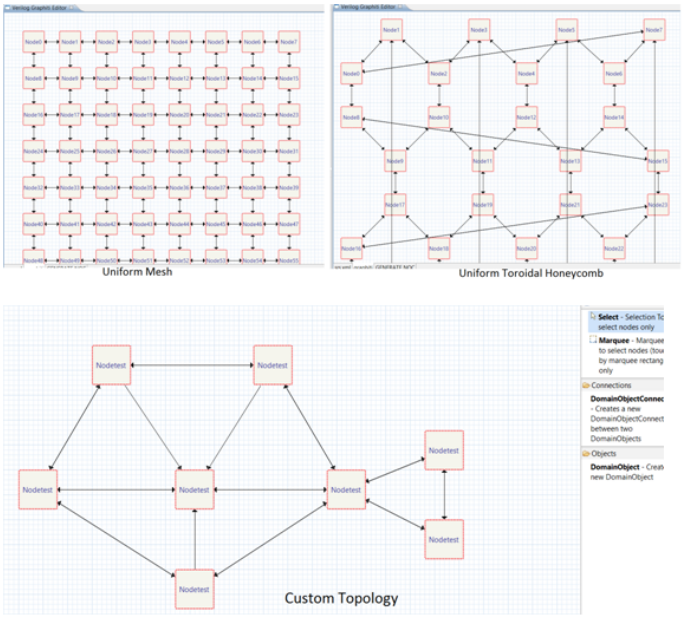

Fig. 2. Stage 2

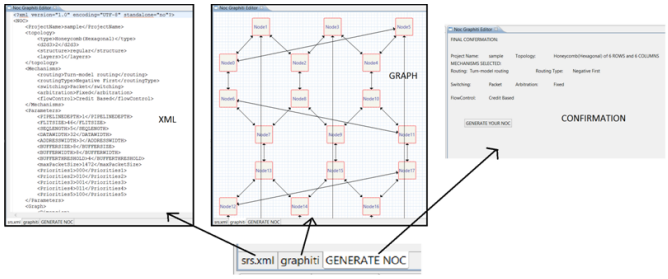

Fig. 3. Stage 3