Journal of Information Systems and Communication
ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, 2012, pp.-174-178.
Available online at http://www.bioinfo.in/contents.php?id=45

A GENERAL PURPOSE SIMULATOR FOR NETWORK-ON-CHIPS

SUDEEP GHOSH[1] AND HAFIZUR RAHAMAN[2]

Department of Informational Technology, Bengal Engineering and Science University, Shibpur, Howrah, India.
*Corresponding Author: Email- [1]sudeep.ghosh123@gmail.com [2]rahaman_h@it.becs.in

Received: January 12, 2012; Accepted: February 15, 2012

Abstract- As a New paradigm for implementing communication among the system components embedded in a single chip, Network-on-Chip (NoC) has gained considerable attention over the last few years. Increasing integration produces a situation where bus structure, which is commonly used in System-on-chips (SoC), becomes blocked and increased capacitance poses physical problems. As such, there has been an increased need for defining and developing simulation software for carrying out simulation of NoC architectures. In this paper, we present a NoC simulator, a systemC based general-purpose network simulator for NoC, which is built upon the object-oriented modular design of the NoC architecture components. Here we demonstrate the use of our proposed simulator by simulating several existing well-known architectures. Finally, we have validated the outputs of our NoC simulator with the studies of existing NoC architectures.
Keywords- simulator, deadlock avoidance, NoC, switches, routers, network, communication.

Citation: Sudeep Ghosh and Hafizur Rahaman (2012) A General Purpose Simulator for Network-On-Chips. Journal of Information Systems and Communication, ISSN: 0976-8742 & E-ISSN: 0976-8750, Volume 3, Issue 1, pp.-174-178.

Introduction

A SoC consists of several system components, for example processors, programmable logic, or memories. These system components are known as Resources. Today's SoCs use a means of communication, which may become problematic in the future. This is why the NoC[1,2] concept is being developed. The NoC concept comprises many different ideas, something like IP, memories, and how to connect various parts of a SoC. Among these various NoC concepts distinction is made between Resources, which do the computation, and the Network, which handles the communication.

Simulations for NoC architectures are as useful as simulations for other computer network. Daniel Wiklund et. al. [8] shows the importance of network-on-chips simulation by giving examples which show how the simulation environment can be used to find the load bottleneck. They describe the design and simulation environment developed in the SoCBUS network-on-chip project. This environment is used as a basis to develop the bench marking procedures necessary to assess the performance of the networks. But this in-house built C++ based simulator only considers a 2-d mesh of size 8 by 8.

Evaluating network architectures requires both synthetic workloads and application oriented traffic. Zhonghai Lu et. al. [9] presents traffic configuration methods that can be used to configure uniform and locality traffic as synthetic workloads, and to configure channel-based traffic for specific application(s). Network messages (traffic) can typically be characterized and constructed by considering their distributions along the three dimensions: spatial distribution, temporal characteristics, and message size specification. The spatial distribution gives the communication partnership between sources and destinations. The temporal characteristics describe the message generation probability over time. The size specification defines the length of communicated messages.

Researchers have developed their own simulators to simulate specific NoC architectures, these simulators are not applicable for other architectures so only few simulator has been developed for simulating NoC architecture. For Example the simulator [3] performed Only evaluate Mesh and Butterfly Fat-tree architecture.

However Our NoC Simulator is a SystemC based general purpose simulator which supports different existing architectures for on-

chip networks: Mesh, Torus, Butterfly Fat Tree, etc. Moreover its modular design helps it to incorporate new architectures. Here we are providing full documentation on how and where to make changes in order to simulate new topology, new design of switch, new routing algorithm etc.

The rest of the paper is structured as follows. We have described NoC architecture in section 2, Modelling of NoC is discussed in section 3. We have described the Journey of a flit of the simulator in section 4, we have descried deadlock avoidance technique in section 5 and the implementation details in section 6. Section 7 delineates how to simulate a new architecture using our simulator. Then we describe the performance evaluation parameters and provide validation of our simulator in section 8. Finally we conclude the paper in section 9.

NoC Architecture

NoC is means of communications among different resources which is built up by following components:

- Switch
- Resource Block (called Intellectual Property (IP)).
- Resource Network Interface (RNI) and
- Dedicated wire for communications

The IP cooperate in performing one or several tasks. In the OSI model, IP implement the Application Layer. All IPs are connected to a network which consists of Network Interfaces and Switches. Above figure shows how the various parts are interconnected.

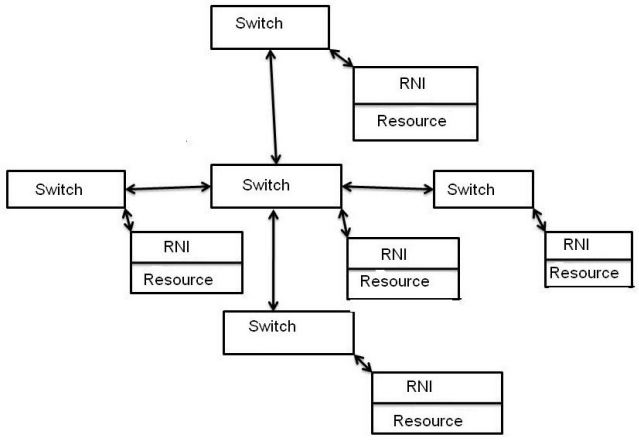

Fig. 1- Block diagram of Noc

The Network Interface provides networking services to an IP. The layers that are implemented in the Network Interface are the presentation, session, and transport layers. Some of the responsibilities of the network layer may be taken cared of here as well.

Each Network Interface is con-nected to a Switch, which itself is connected to a number of other Switches. The responsibility of the Switches is to deliver packets from the source to its destination. One of the problems with future silicon technologies is that it becomes hard to distribute one clock throughout the entire chip to work all the components synchronously.

Modelling Of NoC

1. Interconnect Architectures

In the forthcoming era Network-on-chip become as a most promising design for System-on-chip (SoC) to communicate numerous heterogeneous semiconductor intellectual property (IP) blocks which communicate each other through the switches. There is exits various interconnect routing topology for interconnect routing for example, Mesh, Butterfly Fat Tree, Torus, etc—are applicable for the SoC design. Our simulator provides the designers with the flexibility to simulate different network configurations of NoC.

2. Switch architecture

In this simulator, each switch has number of port to connect with IP block and adjacent switches. Each includes input buffer, output buffer, a router and a controller and the each controller has divided in to two parts- input link controller and out put link controller. The arbiter is implemented in round-robin fashion.

3. Switching Technique

We have used wormhole switching technique [6].It also use virtual channel in output and input port. In wormhole switching technique the packet divided into equal size flits and the first flit i.e. header flit of the packet contains the routing information. Header flit decoding enables the switches to establish a path and this part will be reserved for the rest of flits of that packet. As a result, each incoming data flit of a message packet is simply forwarded along the same output channel, as the preceding data flit and no packet reordering is required at destination.

4. Network Traffic

The Network traffic in NoC architecture is divided in to two types, Guaranteed Throughput (GT) and Best Effort (BE) traffic [7].An arbiter of GT traffic guarantees that some portion of sent data overtakes the receiver in some time slot.GT work best with the routing algorithm that acts like switched network. Best effort packet are arbitrated as trustworthy as posile.Still there are no guarantees that BE packets will ever reach the receiver .Latencies can be vary and most worst case packet can be lost. Traffic in a basic packet switched network is mostly BE-traffic.

Journey of a Flit

With the switch structure specified in the section 3.2 we have implemented following packet communication model (Fig. 2). Controller has not been shown.

When a flit arrived at an input port,it is received at input port of the corresponding Input controller (IC).And the flits moves to parents switch.

The IC read the header of the flits which contains the routing information and other information of the flits like destination, size of the flit and makes a request to controller to allocate a appropriate virtual channel (VC) to store the flit in its corresponding buffer. After getting a VC the flit transfers to either the adjacent node or adjacent switch depending on the destination. If the flits moves to another switch then above technique will repeat again until it will reach to the destination node.As we have used wormhole algorithm this path will be reserved for the rest of the flits of the packet.

Deadlock avoidance

Deadlock occur in NoC architecture when a group of flits unable to make progress because they are waiting for some resources which is currently hold by some other filt.Wormhole switching technique reduce the buffer requirement, but despite of this wormhole switching technique is susceptible to deadlock. In wormhole switching technique the main resources is virtual channel and their

corresponding buffer. Here, cyclic dependency on flit buffers occurs among several packets. As a result, the network is blocked with packets without making any progress. To prevent this problem, we adopt deadlock-avoidance policy in our NoC simulator. Here we used distance class approach [7] to eliminate the cycles in the re-source dependence.

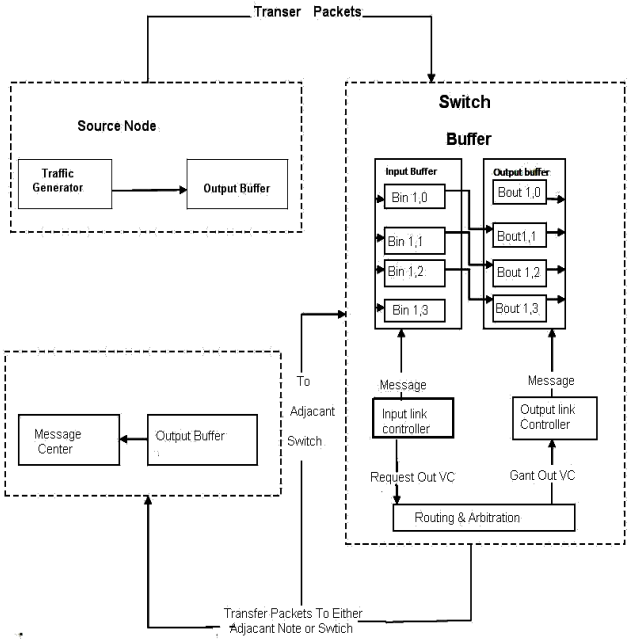

Fig. 2- Journey of a flit

Distance class is an approach to group the virtual channel and the associated buffers into num-bered classes and restrict allocation of resources so that packets acquire resources in ascending order. In this approach, a packet at distance i from its source is allocated virtual channel from class i. Initially from the source node to the parent switch, the packet takes virtual channel 0. Then to transfer to the next switch, it takes virtual channel 1. With this system, a packet holding an input buffer from an input virtual channel waits on an output buffer on higher numbered out-put virtual channel. Since, there is no dependency on the lower or same numbered virtual chan-nels, no cycle in the resource dependence graph can occur; hence deadlock cannot occur.

Fig. 3 gives a concrete example of distance classes, where a network using buffer classes based on distance. From switch i, a flit in an input buffer Bin-i,k moves to an output buffer Bout-i,k+1 to reach the input buffer Bin-i+1,k+1 of the switch i+1. In this figure, none of the packets flowing between switches is generated in the children nodes of these four switches, oth-erwise there would have been flow between Bin-i,k to Bout-i,k.

Implementation Details

A brief description of the component (class) Of or simulator is given below:

- **Controller-** The Controller is the most important class which performs the total flow of execution of the simulator.
- **Network Manager-** Network is the object through which all the nodes (resource blocks) and all the communication switches are connected to perform the desired goal into one unit.
- **Switch-** Main objects of the network where switching of flits from various input ports to output ports are performed (routing).
- **Router-** Routing is the process that determines the path (output link) through which an in-coming header flit in an input link passes through
- **Node-** The Node class checks for free virtual channels, assigns the virtual channels to a message and forwards incoming messages to message center of the node.
- **InputLinkController-** Input Link Controller controls the incoming flits and helps in determin-ing the route of the flits in a switch
- **Output Link Controller-** Output Link Controller class controls the outgoing flits.It helps in sending the flits to the input link controllers of adjacent switches and nodes by using round robin algorithm among the virtual channels.
- **Input Virtual channel buffer-** Input Virtual channel buffer define the data structure of input buffers and functionalities for storing and managing flits of a particular port.
- **Out put Virtual channel buffer-** out put VC buffer define the data structure of out put buffer and fnctionalitiesfor storing and managing flits of a particular port.
- **Flit-** The Flit class encapsulates the variables and methods related to the encoded information of header and data flits.

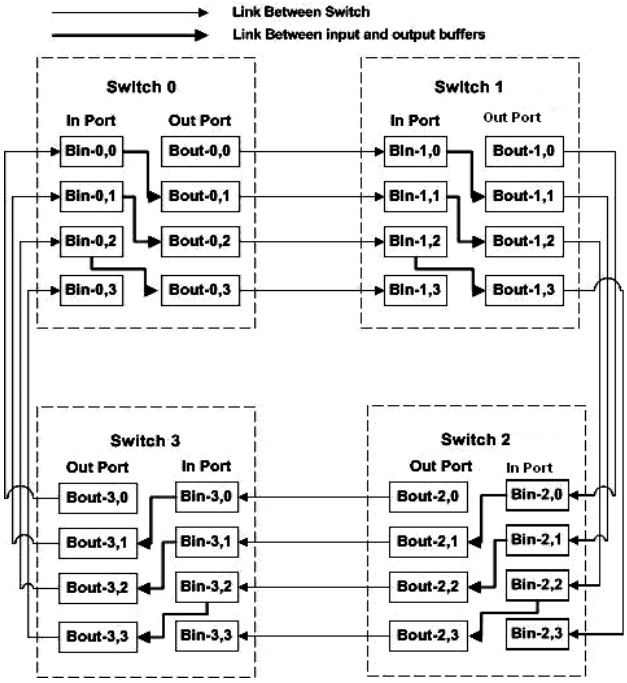

Fig. 3- Implementation of distance class

StatisticalData- The StatisticalData class calculates various performance parameters such as throughput, average packet delay, average hop count, average number of packets sent, average number of packets produced, average number of packets not produced and utilization of input and output buffers, input and output links of nodes and switches.

Helping Utility -This class defines methods for initializing random seed, generating random numbers and reading data from the input file.

Simulator setup

- To create a new architecture for simulation following change has to be made:
- To create new architecture of switch,some one have to create new class name newarchswitch [for Ex-fatswitch]which implements the Switch interface and overrides all the methods necessary to build the switch.
- To change the routing algorithm, one has to create a new class, e.g. NewArchRouter, that implements the getDestination method of the router interface .
- To change the distribution of packet length, one has to modify the getMessageSize method in ConcreteNodeTraffic class.
- To change traffic pattern, one has to modify the getDestination method in ConcreteNodeTraffic class.
- To add new parameter statistical data class has to be maintained.

Experimental Result

Our proposed simulator is able to model and evaluate both the existing and feature architecture It gives output in terms of a number of performance metrics: throughput, latency (average packet delay), link utilization, buffer utilization.

Latency Measurement

Latency is defined as the number of clock cycles required for complete transfer of a packet from the source node to the destination node on average. Here all parameter of mesh topology and 64 switches are kept fixed.

Table 1- Performance measurements for Mesh Topology

Number Of Flits	Average Latency (Clock Cycle)	Number Of Flits in GT Traffic	Latency in GT Traffic	Number Of Flits in BE traffic	Latency in BE Traffic
220	14	75	12	135	15
1434	20	819	23	615	19
1703	16	970	16	633	15
1954	21	994	25	960	17
2267	19	1140	21	1127	16
2649	24	1351	22	1298	27
2786	25	1348	26	1438	24
3199	34	1579	35	1620	33
3544	44	1764	31	1780	26
3702	31	1903	32	1799	29
4502	48	2248	50	2254	45
5004	77	2529	60	2475	95
5897	88	3029	76	2868	102
4405	187	2709	163	1696	224
5898	134	3542	118	2356	158
6184	141	4018	124	2166	173
5858	171	4027	156	1831	205

First we have send some filts and took the average latency then we have divided those flits number into two part and one part we have send one part using Guarantee Throughput traffic and rest of part through Best ef¬fort traffic and took the average latency for both.

From the Fig. 4, we find that buffer size has a strong effect on buffer utilization. With the increase of buffer size the buffer utilization decreases exponentially. This can be verified theoretically. As the network load is fixed and network is not saturated with the data the increment of buffer size simply will keep the new buffer unused and so utilization will decline with the increase of buffer size.

Buffer Utilization

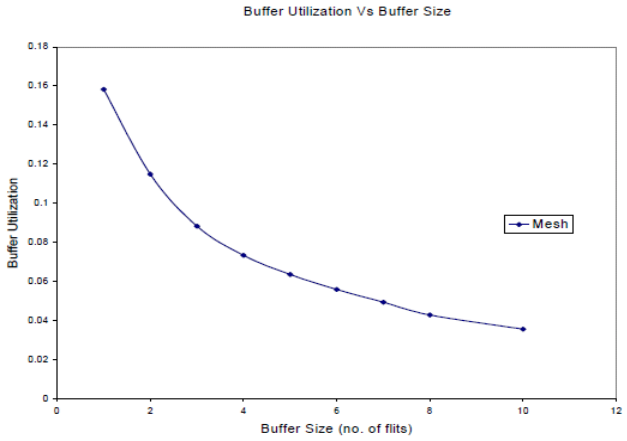

Fig. 4- Buffer Utilization Vs Buffer Size comparison for Mesh topology

Throughput

Throughput is defined as follows:-

$$\frac{\text{Total number of flits received at their destinations}}{\text{(Number of IP blocks) x (Total Time in Cycles)}}$$

Here all the parameters of Mesh and Butterfly Fat Tree topologies with 64 switches were kept fixed to some standard values except the number of virtual channels. Buffer size was set four. Messages were generating at the rate of 300 cycles per message per node. Each switch has only one IP node connected to it. The average message length was 300 bytes and each flit had 64 bits. Number of virtual channels was varied from 1 to 16. Each buffer can hold 4 flits. The measured throughputs are plotted against the Number of virtual channels for Mesh, and Butterfly Fat Tree NoC topologies. The plotted result is shown in Fig. 5.

Comparison with others result

We have compared our simulation result with the work[4]. Here we are providing the comparison graph of avg_throughput vs num_Buffer with work[4].

The Fig. 6 depicts the comparison graph between work[4] and our proposed scheme .We can state the fact that at a given number of buffer our scheme gives enhanced average throughput than work[4].

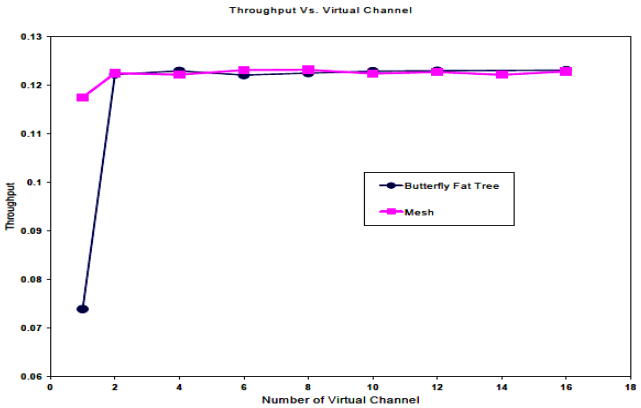

Fig.5- Comparison graph between [4] and ours proposed scheme

Conclusion

In this paper, we have presented a general-purpose simulation software for the NoC architectures and described its modular software architecture. We have considered only uniform traffic distribution in the present version of the simulator. We have facilitated application specific pattern by separating the traffic generating class (NodeTraf-fic) from the Node class. So, in future application specific traffic patterns can be implemented easily. We have implemented flit level simulation. Although we have not considered the power consumed by the hardware, some performance parameters, e.g. buffer utilization, link utilization, average hop counts can be used to derive power related information. But still, more specific power consumption equations can be considered for further study. We designed and developed the general-purpose simulator for NoC considering no faults exits in the connections among ad-jacent switches. Although in chip level the faults will be towards zero if not equal to zero, this faulty links can be considered in routing decision. As we have not considered the gate level de-sign of the topologies, comparison of number of gates required to implement the networks is not made. The measurement of number of gates required can be incorporated in future.

References

[1] Benini L. and De Micheli G. (2002) *IEEE Computer*, 35, 70-78.
[2] Kumar S., Jantsch A., Soininen J.P., Forsell M., Millberg M., Oberg J., Tiensyrja K. and Hemani A. (2002) *IEEE Computer*, 117-124.
[3] Pande P., Grecu C., Jones M., Ivanov A. and Saleh R. (2005) *IEEE Transactions on Computers*, 54(8), 1025-1040.
[4] Wang Zhang, Wuchen Wu, Lei Zuo, Xiaohong Peng (2009) *Global Congress on Intelligent Systems.*
[5] Pande P., Grecu C., Ivanov A. and Saleh R. (2003) *SPIE, VLSI Circuits and Systems*, 5117, 228-237.
[6] Mohapatra P. (1998) ACM Computing Surveys, 30(3), 374-410.
[7] Dally W.J. and Towels B. (2004) *Principles and Practices of Interconnection Networks*.
[8] Wiklund D., Sathe S., Liu D. (2004) 4*th IEEE International Workshop,* 269-274.
[9] Lu Z., Jantsch A. (2005) *Fifth International Workshop on System-on-Chip for Real-Time Applications,* 535-540.