NOC 구조용 교착상태 없는 라우터 설계

Ankur Agarwal* · Mehmet Mustafa** · Ravi Shankar* · A.S. Pandya* · 노영욱***

A Deadlock Free Router Design for Network-on-Chip Architecture

Ankur Agarwal* · Mehmet Mustafa** · Ravi Shankar* · A.S. Pandya* · Young-Ugh Lho***

요 약

다중처리기 SoC(MPSoC) 플랫폼은 SoC 설계 분야에 새로운 여러 가지 혁신적인 트랜드를 가지고 있다. 급격히 십억 단위의 트랜지스터 집적이 가능한 시대에 게이트 길이가 60~90 nm 범위를 갖는 서브 마스크로 기술에서 주요 문제점들은 확장되지 않는 선 지연, 신호 무결성과 비동기화 통신에서의 오류로 인해 발생한다. 이러한 문제점들은 미래의 SoC을 위한 NOC 구조의 사용에 의해 해결될 수 있다. 대부분의 미래 SoC들은 칩 상에서 통신을 위해 네트워크 구조와 패킷 기반 통신 프로토콜을 사용할 것이다. 이 논문은 NOC 구조를 위한 칩 통신에서 교착상태가 발생되지 않는 것을 보장하기위해 적극적 turn prohibition을 갖는 적응적 wormhole 라우팅에 대해 기술한다. 또한 5개의 전이중, flit-wide 통신 채널을 갖는 간단한 라우팅 구조를 제시한다. 메시지 지연에 대한 시뮬레이션 결과를 나타내고 같은 연결비율에서 운영되는 다른 기술들의 결과와 비교한다.

ABSTRACT

Multiprocessor system on chip (MPSoC) platform has set a new innovative trend for the System on Chip (SoC) design. With the rapidly approaching billion transistors era, some of the main problem in deep sub-micron technologies characterized by gate lengths in the range of 60-90 nm will arise from non scalable wire delays, errors in signal integrity and un-synchronized communication. These problems may be addressed by the use of Network on Chip (NOC) architecture for future SoC. Most future SoCs will use network architecture and a packet based communication protocol for on chip communication. This paper presents an adaptive wormhole routing with proactive turn prohibition to guarantee deadlock free on chip communication for NOC architecture. It shows a simple routing architecture with five full-duplex, flit-wide communication channels. We provide simulation results for message latency and compare results with those of dimension ordered techniques operating at the same link rates.

키워드
Network on Chip, Modeling and Simulation, Quality of Service, System Level Design

I. Introduction

The System Level Design era where creativity, innovation, ingenuity and inspiration come to the fore, is approaching and pushing technology at every turn. Moore's law predicts that future chips will count more than four billion transistors operating in multi GHz range[1,2]. It is expected that the future SoCs will integrate from several

* Dept of Computer Science and Engineering Florida Atlantic University
** Electrical and Computer Engineering Department, Boston University.
*** 신라대학교 컴퓨터교육과

접수일자 : 2007. 1. 17

dozens to hundreds of cores in a single billion transistor chip. This is due to the exponential decrease in the transistor size enabling faster transistor switching times and more densely integrated circuits. Such computation power has posed some challenges which include the disparity in transistor and wire speed and increased power dissipation, leading to a decrease in the area of the chip which can be utilized with a single clock cycle[3,4]. Another dominant factor is to be able to design the system in an acceptable timeline known as time-to-market. Also, system level designers are constantly looking for ways to support a set of demanding Quality of Service (QoS) parameters and performance metrics, as customers became more savvy.

Technology scaling at the same time has unwanted side effects which include cross coupling, noise, and transient errors[5]. This has again led us to reuse of design blocks, referred as components, which have been carefully designed by expert designers. However it can never be guaranteed that those sub-micron effects will not pop-up again while reusing those components in a design of a sub-system or a system. Thus it can be concluded that components or the sub-system which perform as expected might not perform in the same way after system integration[6]. This has led to a new domain of research work for system level integration and verification viz, the NOC architecture[2,6].

It can be realized from the ITRS graph[7] that the manufacturing non recurring expense (NRE) of the chip with RTL design methodology alone would have been enormous had future improvements not come about. In the past few decades, it is due to such improvements that teams of engineers and managers were able to bring the cost of a product development down to an affordable price for the customers, while enhancing quality of service and customer support relative to previous releases. However, if such innovations and future trends are not brought into the early stage of the product development cycle, the NRE cost of the product can increase to an unaffordable amount of one billion dollars by 2010[7]. Thus it is expected that the future systems will have increasing roles of design automation, reusability and componentization, thus increasing the market share for the electronic design automation (EDA) Industry. For such scenario, System-level modeling environment should be developed that essentially supports the middle-out design philosophy to exploit reuse to the maximum in order to reduce the design effort[8,9]. The high volume of reuse should cut down the overall system design cycle[10,11]. Networks used in current SoCs are based on both dedicated channels and shared bus approaches. The buses give the best performance but have the poor reusability resulting in high time to market On the other hand, network based communication strategies provide a reusable, scalable and highly flexible solution to cope with the current technology trends. Thus, most of the future systems would have several SoCs that will use network architecture and a packet based communication protocol for on chip communication, referred as NOC. In NOC we divide the system into smaller, locally decoupled synchronous regions and then composing a local solution. These synchronous regions would be easier to integrate into a global solution and verify. At the same time there will be an asynchronous way in which all the local synchronous regions will cooperate at the system level; this is referred as the global solution. Thus these different synchronous regions need not have a single clock domain to work with. This approach would reduce the requirement for the clock tree designers as they need to worry about the local synchronous regions only which mainly consist of components and subsystems rather than the system as a whole. At the same time due to the flexibility to reduce the clock speed, the amount of power consumption in a system can be managed better and reduced. In this paper we present a deadlock free routing protocol for NOC.

Ⅱ. Background Work

As emphasized in the International Technology Roadmap for Semiconductor (ITRS) 2001 document[7], at system level, it is very important to separate the computation from communication aspects to enhance design productivity[14]. From the communication perspective, several researchers have suggested 2-D mesh architecture for NOC[12,13] consisting of resources and network elements. The network elements are switches, channels and resource-network interfaces (RNI). The resources for the NOC can be any general purpose processor core, memory, specified controller, FPGA, ASIC etc. This infrastructure will be ideal where QoS and performance parameters can be traded off based on the user requirements. New algorithms have been proposed in this domain to reduce the power consumption while securing cost optimization[15]. It has been a well established fact that such NOC architectures will be based on packet switched networks. This has led to new and efficient design of routers for the NOC architecture[16]. These routers will be responsible for routing the entire traffic across and have to be interfaced with switches and resources in the NOC architecture. The design of the RNI should again be highly scalable and re-usable to be able to be integrated with different types of resources without the need for change of the core RNI for different types of resources. Substantial research has been conducted to propose the right data formats needed for various layers in the protocol stack. A reusable switch is used for effectively routing the packet through the entire NOC; they buffer packets at both input and output[17].

Because of its simplicity, low channel setup times, high performance in delivering messages[18], wormhole routing appears to be uniquely suited for NOC applications. In wormhole routed networks, a message, also called a worm, is sent in flow control digits, called flits. Flits propagate from node to node in bit parallel fashion, with tight handshaking between adjacent nodes. This handshake is in the form of Ready/Acknowledge, R/A, to prevent overwriting a flit before it has been attended to. The sending router asserts the Ready line and the receiving router asserts the Acknowledge line. The input flit buffers being one flit deep, flits making up a message are therefore in the flit buffers of the intervening routers along the path between the source router and the destination router. In contrast to store-and-forward techniques in which message latency is linearly dependent on the path length, message latency in wormhole routed networks is insensitive to distance[19]. However, due to a number of channels being held up while requesting others, wormhole routing is susceptible to deadlocks. Turn prohibition was first reported in [20], where turn model was thoroughly investigated for multi-dimensional meshes, and some turns were prohibited to prevent all possible deadlocks. Authors considered only 90 degree turns which are sufficient for meshes to prevent deadlock formation. In [21-24] authors generalized the notion of a turn, and developed an algorithm to construct minimal sets of prohibited turns, to break all cycles and prevent deadlock formation. Authors also established that the fraction of prohibited turns could be used as one of the criteria of efficiency of a routing strategy. In [25] authors extended the use of turn prohibition as described in [21] to general topologies and applied the Network Calculus techniques, which, until then, were strictly for feed-forward routing networks.

III. Turn Prohibition

This section provides a brief and simplified overview of Turn Prohibition[21]. We assume that the network topology is represented by an undirected, connected graph G = (V, E) of N = |V| nodes and M = |E| edges. A turn is defined as a three-tuple of nodes (a, b, c), where a,b,c εV, are nodes in the network in which (a, b), (b,c) ε E are edges in G incident on node b. Turns are symmetric meaning that if turn (a, b, c) is prohibited then the turns (c, b, a) will also be prohibited. If a node ajhas degree dj then the total number of turns in the graph is given by $T(G) = \sum_{j=1}^{N} \binom{d_j}{2} = \sum_{j=1}^{N} \frac{d_j(d_j-1)}{2}$ where the summation is taken over all nodes. We denote by Z(G) the minimal number of turns breaking all cycles in G and by Z(G) = Z(G)/T(G) the fraction of prohibited turns.

Bounds on z(G)

Here we present two lower-bounds on the fraction of prohibited turns in the form of a theorem the proof of which has been eliminated from this paper but can be found in[21].

Theorem 1. If C={C1, C2⋯Cg} is a set of cycles in G

with N nodes and M edges and r the maximal number of cycles in C containing the same turn, then the fraction of prohibited turns Z(G) is

$$z(G) \ge (M - N + 1) / T(G), \quad (1)$$

$$Z(G) \ge \frac{R}{rT(G)}. \quad (2)$$

Bound (1) is general and is not as tight as the bound (2). For example in a $p \times p$ 2-dimensional mesh with $N = p^2$ and $M = 2p(p-1)$ from bound we have

$$z(G) \ge \frac{p^2 - 2p + 1}{6p^2 - 12p + 4}. \quad (3)$$

If r = 1, then bound becomes $z(G) \ge R / T(G)$ where R is the number of turn disjoint cycles in G. We note that for large meshes the fraction of prohibited turns approaches 1/6. Upper bounds are based on constructions using the Turn Prohibition or TP- algorithm. This algorithm can be used in many cases to construct minimal turn prohibition set of turns. In its simplest form, TP-algorithm selects a minimum degree node a with degree δ_a, and prohibits all turns (b, a, c) at the selected node. All turns that start with the selected node a, such as, (a, b, c) are permitted. Algorithm then deletes node a and all edges incident on it and then applies the algorithm to the sub-graph G-a. TP-algorithm, not only maintains the connectivity of the graph, but it also guarantees that the set of prohibited turns is irreducible and that the fraction of prohibited turns does not exceed 1/3. The only family of graphs that this limit is attained is the family of complete graphs, Kn[25, 26].

Routing table construction is performed based on all-pairs shortest paths between any two nodes in the network such that paths do not involve any of the turns from the set Z(G). Since the topology of the NOC is not likely to change during computations, we propose to construct the routing table off-line and download them into the respective nodes prior to use. Routing table is a 2-dimensional matrix where each row represents an input port. Head of the columns are the destination node numbers and a matrix entry is five bits wide vector, where each bit position is either 1 or zero. A bit position i implies that the worm coming in from the corresponding input port can be routed to output port i. If there are multiple non-zero entries, that implies that all of the corresponding output ports are equal distance from the destination node, and depending on the network state, either one could be used. In this sense, our routing approach is minimal and adaptive. For a pXp mesh or torus, the size of the routing table is only 25p2 bits. In Table 1, entries that are zero imply that no route exists for the destination from the corresponding input port. For example a message coming in on port 1 destined for node 1 is not routable. This however does not imply that messages cannot be delivered. It is an indication that a message that was meant tobe for node 1 should not be coming in from node 1. It is also interesting to note that, messages coming in on ports 2 and 3 have additional restrictions due to the fact that turn (9, 5, 6) is forbidden.

Table 1. Routing Table for node 5. Routing vector for worms coming in from South, input port 1, for destination node 10 can be routed adaptively on either port 2 (East) or port 3 (North), hence the table entry of 12 = < 01100>.

dest. / ip	0	1	···	5	···	10	11	···
0 (LP)	18	2		0		12	12	
1 (S)	16	0		1		12	12	
2 (E)	16	2		1		0	0	
3 (N)	18	2		1		0	0	
4 (W)	2	2		1		12	12	

Also note that messages coming in from ports 1...4 that are destined for node 5 are routed toport 0, where the local processor of node 5 is attached. Referring back to the prohibited turns for a 2-D Mesh, we note that, from prohibited turns perspective, widely used dimension ordered or XY routing algorithm[19] is not symmetrical. Using our convention for turns, as shown Figure 1 , for a core node of 2D Mesh, there are four prohibited turns and eight permitted turns. This means that for large meshes, where there are large numbers of core nodes, the fraction of prohibited turns for the XY routing algorithm is $z_{XY}\left(M_{p\times p}\right)\to 1/3,\ p\to\infty$. As was stated earlier, using our approach to turn prohibition would result in $z_{TP}\left(M_{p\times p}\right)\to 1/6,\ p\to\infty$. This implies that more resources would be available for communication with deadlock freedom guarantee using the TP approach.

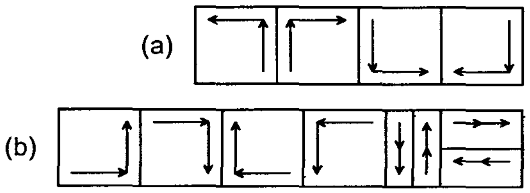


Fig. 1 Prohibited (a) and permitted (b) turns in XY routing algorithm

Ⅳ. Router Architecture

In this section we describe the architecture of a simple router capable of wormhole routing in the 2-D mesh. In Figure 2, we show the architecture of such a simple wormhole router. Router is a five port device where each port has an input and an output flit bus as shown. Incoming flit buses are each connected to their respective flit buffers. Flit buffers are the only storage components in the data path in the router. Each of the five output buses is driven by a 5-to-1 multiplexer which is further detailed in Figure 3. When a flit is strobed in with the RDY_IN signal by the adjacent router, the RDY signal is asserted tothe local router. Router controller detects the arrival of a flit by the rising edge of the RDY signal. Since the flit is now available on the output of the flit buffer, its type can be inspected and the necessary actions taken by the router controller. Multiplexers form the switching component in the router. Under the direction of the router controller, each output port would be connected to the selected input flit bus. Once an input has been bound to an output port, the three-bit selection latch, SL, would hold the selection information during the entire lifetime of a worm through the router.

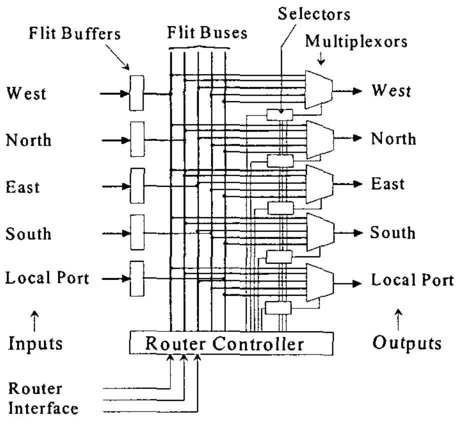


Fig. 2 Wormhole Router Architecture

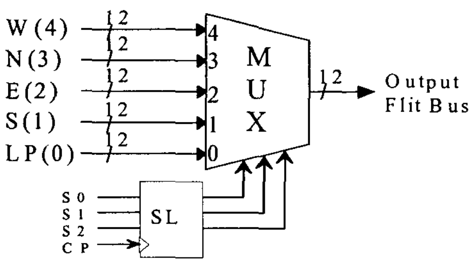


Fig. 3 Flit Bus Multiplexer and Selection Latch

After the tail of a worm, identified by the type field of the flit, leaves the router and an acknowledgement has been received from the adjacent router, the input/output binding is dissolved and the router controller can use the freed up output channel for another worm. Structure of the flit buffer and the handshaking logic is depicted in Figure 4.

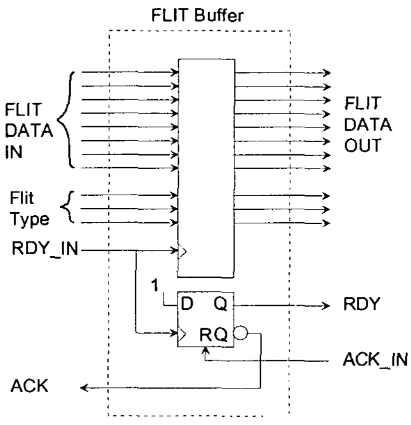


Fig. 4 Flit buffer and the Handshaking Control Logic

For example if there are multiple header flits that had been waiting for an output port which has just been freed up, the controller inspects all awaiting header flits and establishes which flit should acquire ownership of the output port. Based on the arrival time stamps, captured with the rising edge of the RDY signal, router controller can invoke a number of selection algorithms to arbitrate among the waiting header flits. Once the decision is made and the input-output pair is identified, the appropriate 3-bit selection value is written into the SL latch of the output causing the flit and the control signal RDY to propagate to the flit buffer of the adjacent router on the output port. When the destination is reached, the flit is taken off and fed to the local processor. As a flit is taken care of by a router, the latter asserts the ACK_IN signal back to the adjacent source router. This is detected by the router controller, which then is interpreted tomean that next flit could be sent. This tight handshaking continues until the entire worm is delivered toits destination. Process of transmitting a worm is initiated by a processor at the injection channel or port 0. Local processor owns port 0 and knows if it is free or busy. Therefore without any ambiguity initiates the transfer by writing the header flit into port 0 flit buffer. This action then is detected by the local router controller and the sequence of events discussed earlier takes place to move the flits and the worm across the network.

V. Wormhole Routing Simulation

We developed simulation models in Opnet and simulated message delivery in 2-D Meshes. In this section we describe the simulation models. In Figure 5, we show a typical 16-node Mesh that we simulated using Opnet. As will be clearer in the following detailed discussions of the model, Opnet's finite state machine approach for describing the behavior of modules, simplified our design considerably.

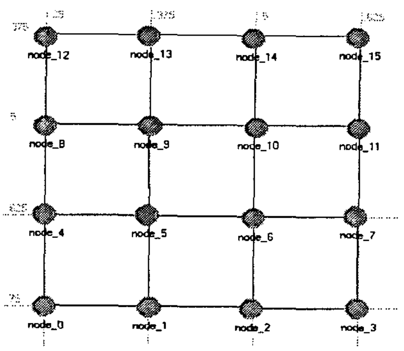
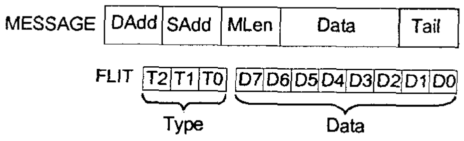


Fig. 5 Mesh Network in Opnet

Structure of a wormhole message is depicted in Figure 6. A typical unicast message has three header flits, where the first flit is Destination Address Flit, the second flit is Source Address Flit and the third flit is the Message Length Flit. Header is followed by zero or more data/payload flits. Last flit in a wormhole message is a tail flit, which contains the last payload data for the message. We used an 11-bit long packet with two fields as our packet format simulating one flit. Three bits wide Type field identifies the type of the flit and the eight bits wide Data field is the flit payload. As described below, wormhole handshaking is implemented using the built-in statwire and remote interrupt mechanisms in Opnet.



T2	T1	T0	Flit Type
0	0	0	Data Flit
0	0	1	Dest Address Flit
0	1	0	Src Address Flit
1	1	0	Message Length Flit
1	1	1	Tail Flit

Fig. 6 Wormhole message and flit formats and flit types used in simulation models

5.1 Wormhole Node Model

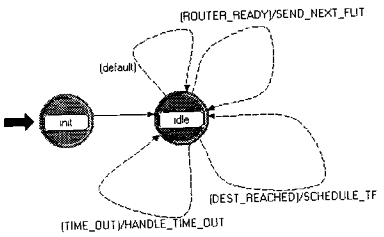
A wormhole node consists of a router and a processor, the latter being modeled by two queues; PGQueue for generation of messages at the source node, and PSQueue, for consumption of messages at the destination node. Router module is responsible for forwarding all flits arriving at the input ports to the appropriate output ports. Wormhole handshaking[19] is implemented using the statwires within the node, and remote interrupts with adjacent nodes. Reception of a stream interrupt is interpreted as RDY, indicating the arrival of a flit. If a flit arrives from the local injection channel from the PGQueue module, router uses the statwire to acknowledge the receipt and completion of handling of the flit. When PGQueue module receives a statwire event, it retrieves the next flit of the message from the queue and sends it to the router. When the tail flit arrives from an adjacent router, a remote interrupt is used to acknowledge that router is ready for the next flit.

When the Destination Address Flit arrives at a router, it is stored temporarily in the flit buffer. Reference to the routing table identifies which output port to use. If the output port is busy, the header flit is blocked and no acknowledgement is sent to the sender. When the output port is finally freed up, it is associated with the waiting header flit. If multiple header flits are waiting for the same output port that has just been freed up, selection and binding is done on a FIFO basis.

In this environment, sending a 200 flit long message would involve sending 200 packets: a very inefficient use of simulation resources. Because of this we incorporated an optimization mechanism, in which, source node processor generates only as many flits as necessary for the Source Address flit toarrive at its destination. Once the destination is reached, the destination node, knowing the sending node address, sends a remote interrupt tothe source node. When this remote interrupt is received, source node generates a tail flit, stores it in the queue and schedules a self interrupt. Scheduling of the self interrupt is based on number of flits already generated, message length, and the flit transmission time. When this tail self interrupt arrives, the tail flit is transmitted to the router from the PGQueue. This optimization significantly improved the simulation run time since for each message of 200 flits; only six to seven flits per message are generated and processed.

5.2 PGQueue Module

In Figure 7 we show the process model of the PGQueue module of the wormhole node. The forced init state performs the one-time setup necessary at the module, reads in the values for the run-time attributes, schedules a self interrupt for generating a message, and transitions into the only blocking state labeled idle. In the idle state, if ROUTER_READY event is triggered, it implies that the local router has just sent an acknowledgement via the statwire (used for status signals). In this case the PGQueue process sends either a flit already in the queue, or if necessary generates one and transmits it tothe router. In our implementation, we use the stream interrupt as the READY signal, and the statwire as the ACK signal of the wormhole handshake. When the DEST_REACHED event fires, it implies that we just received a remote interrupt from the destination, and we schedule a tail flit tobe transmitted when self interrupt expires. When this self interrupt expires, the TIME_OUT macro is triggered, and the HANDLE_TIME_OUT is executed. In this function, the tail flit is transmitted if the interrupt code indicates so. Otherwise, time-out implies that time for generating another message has arrived and a new worm is generated.


(ROUTER_READY)/SEND_NEXT_FLIT
(default)
init
idle
(DEST_REACHED)/SCHEDULE_TF
(TIME_OUT)/HANDLE_TIME_OUT
Fig. 7 Process domain description of the generator

5.3 PSQueue Module

As shown in Figure 8, the process model of the PSQueue of the wormhole node processor is very simple. Again, all of the one-time, module level processing is done in the forcing state called init. In the blocking state labeled idle, flits that arrive from the local router are handled. Arrival of a flit is triggered by the ARRIVAL event, which causes the RECEIVE_FLIT exec to run.

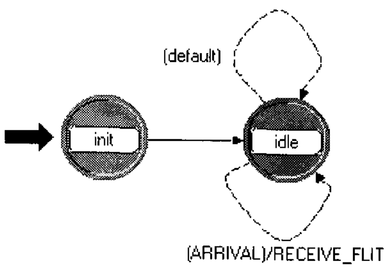


Fig. 8 Sink model of a node

When run, it stores the flit in the receive queue using the wormhole handshaking with the local router. If the incoming flit is a Source Address Flit, then a remote interrupt is sent to the PGQueue of the source node. If the incoming flit is a tail flit, then end-to-end delay is computed and saved and message in the PSQueue is discarded. One other activity that takes place in RECEIVE_FLIT is monitoring the attainment of network stability. PSQueue process, computes the cumulative average with every message, and if the stability criterion is reached the simulation is terminated. Since we are interested in running many simulations on many different to pologies, unattended by the user, we opted to determine the stability condition in this process. We compute the percent difference between the current and the previous values for the cumulative running average. If this difference is less than $\pm 0.1\%$ for 300 consecutive messages, we assume stability is attained and terminate the simulation. Figure 9 shows the time evolution and attainment of network stability at low message injection rates, requiring about 1000 messages.

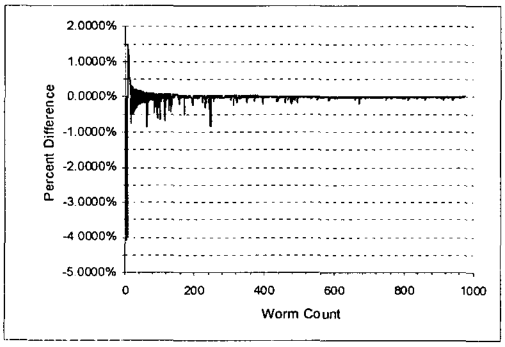


Fig. 9 Attainment of network stability

5.4 Router Module

Router module process model is depicted in Figure 10, where we have two blocking states and a forcing state. In Init state, router identifies its node number from its name, reads in the run-time attribute values, identifies the attached PGQueue and PSQueue objects, initializes all state variables, dynamically allocates memory for routing tables, and reads in the node specific routing table. It then transitions tothe Identify blocking state. In this state, each node identifies its neighbors by sending just a Source Node Address Flit to each of its active ports. When the router receives all responses from its active ports, it schedules a self interrupt with no delay and transitions to the next blocking state called Listening.

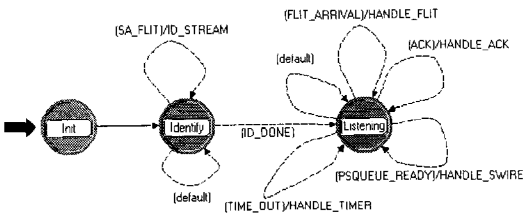


Fig. 10 Process model for the router

In Listening state, FLIT_ARRIVAL is defined to be a stream interrupt and all flit types are processed by a function represented by the HANDLE_FLIT macro. This function calls other procedures that handle individual flit types. For example, if the flit type is a Destination Node Address flit, then routing table is referenced to identify the output port that flit needs to be transmitted out of. If the output port is busy then process of handling the waiting header flit is blocked.

When a tail flit is transmitted out on any busy output port, then list of waiting headers is examined. If there is a header flit waiting for the output port that has just been freed up, then the input port that it came from is associated with the, now free, output port and binding takes place. When the HANDLE_FLIT macro, identifies that the destination node address is equal to its own node address, then the local output port to the PSQueue is bound to the port delivering the flit. From now on, all incoming flits and the tail flit from the associated input port are sent out to this output port. With each transmitted flit, router process will send a stat wire to the PGQueue object and will be awaiting a remote interrupt from the adjacent router. As discussed earlier, wormhole handshaking takes place at the router at three interfaces; first between the two adjacent routers, second between the PGQueue and the local router, and third between the PSQueue and the local router. The transition event called ACK is defined to be a remote interrupt from an adjacent node. Router process identifies the adjacent node by the interrupt code and interprets it to mean that remote node is ready for the next flit. All remote interrupts are processed by the function defined in the HANDLE_ACK macro. PSQUEUE_READY transition event is defined to trigger when a statwire interrupt is received from the PSQueue module, indicating that it is ready for the next flit. This interrupt is managed by the function represented by the HANDLE_SWIRE macro. The TIME_OUT macro is for debugging purposes and is not used.

Ⅵ. Simulation Results

In this section we provide the results of our simulation experiments for dimension ordered or XY routing, deterministic Turn Prohibition or TP based routing, and Adaptive TP or TPA routing. We simulated a 8x8 2D Mesh using all three techniques and collected average end-to-end delay statistics for message delivery. We determine the message latency as the time difference between when the tail of the message is received by the destination and the message launch time into the network. Communication channels were operating at 109Mfps or 1.2Gbps rates. All messages were 200 flits long and traffic distribution was uniform. We chose to have the routers with no overhead during channel setup. Simulations progressed until network stability was attained and subsequently data were collected. It can be seen that adaptive TPprovided the best saturation characteristics. Deterministic TP and XY routing performance were similar with marginally better saturation characteristic demonstrated by the TP approach.

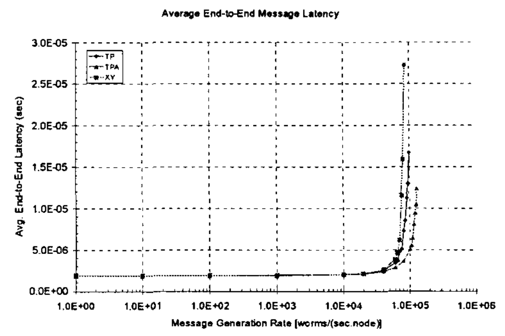


Fig. 11 Experimental results for end-to-end message delivery in a 64 node 8x8 2D Mesh

Ⅶ. Conclusions

In this paper we quantified the deadlock free communication architecture for NOC. All the links among the nodes are considered tobe reliable for on chip communication and are operating at 1.2Gbps. Wormhole node and router simulation models were developed using Opnet based tools. We further investigated two competing algorithms in 2D meshes. We provided simulation results, which indicate that adaptive TP provided the best saturation characteristics. XY and deterministic TP approaches demonstrated similar characteristics with deterministic TP having marginal advantage over XY routing.

REFERENCES

[1] L. Benini and G. De Micheli. Networks on chip: a new SOC paradigm, IEEE Computer, Volume 35, No. 1, January, 2002, 70-78.

[2] Xu, Jiang, W. Wolf, J. Hankel, S. Charkdhar, A Methodology for design, modeling and analysis for networks-on-Chip, IEEE International Symposium on Circuits and Systems, May 2005, 1778-1781

[3] Hemani, Axel Jantsch, Shashi Kumar Adam Postula, Johnny Öberg, MikaelMillberg, Dan Lindqvist, Network on Chip: an architecture for billion transistor era, Proc. of IEEE NorChip Conference, November 2000.

[4] Paul Wielage, Kees Goossens. Network on silicon:

blessing or nightmare? In Euromicro Symposium on Digital System Design, Dortmund, Genmany, September 2003. Keynote Speech.

[5] Tejasvi Das, Clyde Washburn, P. R. Mukund, Steve Howard, Ken Paradis, Jung-Geau Jang, Jan Kolnik, Effects of technology and dimensional scaling on input loss prediction of RF MOSFETs, International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design, 2005, pp. 295-300.

[6] Alexandre M. Amory, Érika Cota, Marcelo Lubaszewski, Fernando G. Moraes, Reducing test time with processor reuse in network-on-chip based systems, Proceedings of the 17th ACM symposium on Integrated circuits and system design, 2004, pp. 111-116.

[7] Semiconductor Industry Association, The international Technology Roadmap for Semiconductors. 2001. http://public.itrs.net/Files/2001ITRS/Home.htm

[8] Edward A. Lee, Yuhing Xiong, System level types for component-based design, Workshop on Embedded Software, California, October 2001.

[9] Y. Xiong and E. A. Lee, "An extensible type system for component-based design", 6th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Berlin, Germany, April 2000.

[10] Cota, E.; Kreutz, M.; Zeferino, C.A.; Carro, L.; Lubaszewski, M.; Susin, A., The impact of NoC reuse on the testing of core-based systems, 21st Proceedings of VLSI Test Symposium, 2003, April 2003, 128 - 133

[11] A. Jantsch and H. Tenhunen. Networks on Chip Kluwer Academic Publisher, 2003.

[12] S. Kumar, A. Jantsch, J-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A Network on Chip Architecture and Design Methodology. In IEEE Computer Society Annual symposium on VLSI, April 2002, 117-124

[13] K. Keutzer, S. Malik, A. Richard Newton, Jan M. Rabaey, A. Sangiovanni-Vincentelli, System level design: orthogolanlzation of concerns and platform based design, IEEE Transaction on CAD of Integrated Circuits and Systems, 19(12): 2000, 1523-1543

[14] P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, Performance evaluation and design tradeoffs for network on chip interconnect architecture, IEEE Transaction on Computers, vol. 54, Issue 8, August 2005, 1025-1040

[15] D. Rostilav, V. Vishnyakov, E. Friedman, R. Ginosar, An asynchronous router for multiple service levels networks on chip, 11th IEEE international symposium on asynchronous circuits and systems, March 2005, 44-53.

[16] Yi Ran Sun, S. Kumar, A. Jain, Simulation and evaluation for network on chip architecture using NS-2, 20th IEEE International Conference preceding for NorChip vol. 5, May 2003.

[17] F. Silla and J. Duato"High-Performance Routing in Networks of Workstations with Irregular Topology," IEEE Trans. on Parallel and Distributed Systems vol. 11, no. 7, pp. 699-719, 2000.

[18] L. Ni, M. and P. McKinley, K. "A Survey of Wormhole Routing Techniques in Directed Networks," Computer vol. 26, pp. 62-76, 1993.

[19] C. Glass and L. Ni "The Turn Model for Adaptive Routing," Journal of ACM vol. 5, pp. 874-902, 1994.

[20] L. Zakrevski "PhD Thesis: Fault-Tolerant Wormhole Message Routiing in Computer Communication Networks," College of Engineering pp. 21-27, 2000.

[21] L. Zakrevski, S. Jaiswal, L. Levitin and M. Karpovsky "A New Method for Deadlock Elimination in Computer Networks With Irregular Toplologies," Pro. of the IASTED Conf. PDCS-99, vol.1, pp.396-402, 1999.

[22] L. Zakrevski, S. Jaiswaland M. Karpovsky "Unicast Message Routing in Communication Networks With Irregular Topologies," Proc. of CAD-99 1999.

[23] L. Zakrevski, M. Mustafa and M. Karpovsky "Turn Prohibition Based Routing in Irregular Computer Networks," Proc. of the IASTED International Conference on Parallel and Distributed Computing and Systems pp. 175-179, 2000.

[24] D. Starobinski, M. Karpovsky and L. Zakrevski "Application of Network Calculus to General Topologies Using Turn Prohibition," IEEE/ACM Transactions on Networking vol. 11, no. 3, pp. 411-421, 2003.

저자소개

Ankur Agarwal

is an assistant professor at the Computer Science and Engineering Department, Florida Atlantic University. He received his MS and Ph.D. in computer engineering from FAU. He also holds two post graduate diplomas in VLSI design and real-time embedded system design. He has earned his bachelor of engineering from Pune University, India in year 2000.

※Research Areas : concurrency modeling, system level design, network-on-chip, real-time-operating system and VLSI design.

Mehmet Mustafa

is research associate at Electrical and Computer Engineering department, Boston University. He obtained his MS. degree in Electrical and Systems Engineering in 1978 and worked at GTE Laboratories during 26 years. He received his second MS. degree from Boston University in Computer Science in 1997 and subsequently the Ph.D degree in 2006, in Computer Engineering.

Research Areas : network-on-chip, Wormhole Routing, concurrency modeling, system level design

Rabi Shankar

is a professor at the Computer Science and Engineering Department, Florida Atlantic University. He received his Ph.D. in Computer Science from University of Wisconsin-Madison.

※Research Areas : Engineering Productivity, Concurrency, Software Decomposition

A. S. Pandya

is a professor at the Computer Science and Engineering Department, Florida Atlantic University. He received his undergraduate education at the Indian Institute of Technology, Bombay. He earned his M.S. and Ph.D. in Computer Science from the Syracuse University, New York. He has worked as a visiting Professor in various countries including Japan, Korea, India, etc.

※Research Areas : VLSI implementable algorithms, Applications of AI and Image analysis in Medicine, Financial Forecasting using Neural Networks.

노영욱(Young-Uhg Lho):교신저자

1985년 2월 부산대학교 학사
1989년 2월 부산대학교 석사
1995년 2월 부산대학교 박사

1989년 ~ 1996년 한국전자통신연구원(ETRI) 연구원
1996년 ~ 현재 신라대학교 교수

※관심분야: 내장형시스템, 멀티미디어 시스템, 병렬분산 시스템, 지능형시스템, 실시간운영체제, 컴퓨터교육