Considerations for fault-tolerant Network on Chips

Muhammad Ali[1], Michael Welzl[1], Martin Zwicknagl[1], Sybille Hellebrand[2]
[1] Institute of Computer Science, University of Innsbruck, Austria
[2] Institute of Electrical Engineering, University of Paderborn, Germany
{Muhammad.Ali, Michael.Welzl, Martin.Zwicknagl}@uibk.ac.at
Sybille.Hellebrand@date.upb.de

Abstract

According to International Technology Roadmap for Semiconductors (ITRS), before the end of this decade we will be entering the era of a billion transistors on a single chip. However, it has been observed that as the system grows, so does the complexity of integrating various components on a chip. The major threat toward the achievement of a billion transistor chip is poor scalability of current interconnect structure of today's SoCs[1]*. In order to cope with growing interconnect infrastructure, the "Network on chip (NoC)" concept was introduced. With network methodologies coming on-chip, various characteristics of traditional networks come into play. So far, failures that are common in regular networks were hardly considered on-chip; this paper introduces ideas of dynamic routing and congestion control in the context of NoCs and explains how they could be applied to cope with adverse physical effects of deep submicron technology.*

Keywords
Networks on Chip, fault-tolerant, self-healing, routing, congestion

1. Introduction

The development in silicon technology is moving at such a rapid pace that before the end of this decade we will witness chips capable of accommodating billions of transistors [1]. These advancements will surely help chip designers and engineers to build complex SoCs hence bringing a whole new revolution in chip technology. This revolution is, nevertheless, barricaded by buses – traditional interconnects of SoCs. Studies have revealed that buses cannot scale beyond a certain number of partners on-chip. Due to this very reason buses have become a bottleneck in the advancement and growth of future SoCs [1], [2]. To cope with the inefficiency of traditional interconnects, researchers have already explored traditional computer networks which has successfully dealt with issues like scalability (eg, Internet which scales very well). All these efforts have yielded a novel, efficient, flexible and scalable interconnect for future SoCs – *Network on chips (NoC)*.

A NoC is a structured set of routers or switches that are connected to each other on a point to point link in order to provide a communication backbone to the processing cores of a SoC [3]. Many design templates have been proposed for the future communication centric SoCs that include [1], [4], [5], [6], [7]. The most common of these templates is a 2D Mesh where each resource or set of resources is connected with a router. These routers are then connected to their immediate neighbor routers as shown in Figure 1.

The primary focus of this paper is to ensure the existence of a fault-tolerant chip which is reliable enough to demonstrate desirable performance even under adverse physical conditions. In this regard, our proposal is targeted toward dealing with two major sources of problems that may be inevitable to avoid on a scaling chip; firstly broken links or malfunctioning routers and secondly congestion occurring both in routers and links. To the best of our knowledge, the problem of broken links or malfunctioning routers on-chip is not yet addressed by the research community which is probably due to the assumption that NoCs are considered to be stable and, hence, no such failures are supposed to occur [8]. However, we believe that when these networks are implemented on a chip, they may face same kind of problems which are common on today's ICs [9]. For example, crosstalk faults can lead to permanent or transient failures of the communication links [10]. Similarly, routers can fail as any processor implemented on chip. Moreover, an increase in the number of transistors on-chip would result in more transient and permanent failures of signals, logic values and interconnects [11], [12], [13]. Although already available standard diagnosis and fault tolerance test may be applied to NoCs, yet they don't exploit any particular network properties. In such a scenario, we propose a dynamic routing

[1]System on a chip

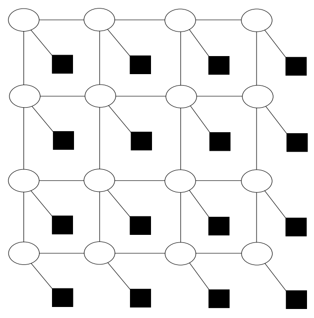

Figure 1. 2D mesh based NoC

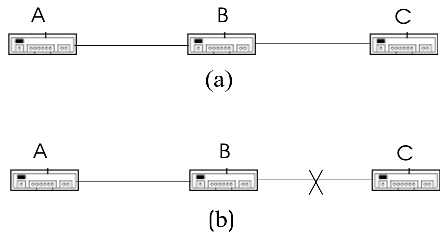

Figure 2. Counting-to-infinity problem

mechanism for NoCs which can automatically calculate alternate paths in case of link/router failures on a chip. Moreover, scalability of NoCs is limited if we (wrongly) assume that traffic does not grow with the number of communicating entities in the system. Thus, we are facing an increasing amount of traffic which might lead to congestion in routers, even keeping the assumption that the resources (end points) of the chip might have unlimited buffers due to which congestion might not occur at resource level [14]. This, in turn, leads to packet drops which are just as unwanted in case of NoCs as they are in traditional networks.

In rest of the paper we will discuss two famous dynamic routing mechanisms found in the Internet and will propose an algorithm suitable for NoCs. Furthermore, we propose to deal with the problem of congestion that might occur at the nodes and links in NoCs.

2. Dynamic Routing

Routing is the act of moving information from a source to a destination following the shortest possible path. In the Internet, routing can be static or dynamic. Static routing is managed by an administrator manually and is suitable for networks where network traffic is predictable and relatively simple, which is a rare case in the Internet. Dynamic routing, as the name shows, is used to dynamically discover routes in case of path changes.

Here we are going to discuss the two most popular dynamic routing algorithms in networks, *Distance Vector routing* and *Link State routing*.

2.1 Distance Vector Routing

This is quite a simple routing mechanism where each router maintains a table about the known destinations available to it along with the link to get there, and it sends this information to its neighbors. In this way distance vector routing keeps information about neighbors and destinations only and not of the whole network, which makes it a simple mechanism. Distance vector routing is also known as Bellman-Ford algorithm because it is based on a shortest path computation algorithm which was first described by R.E. Bellman, and the first description of its distributed variant was given by Ford and Fulkerson [15]. The router in this case knows the distance to each destination.

Distance vector routing, though simple, is not the mechanism of choice because it can lead to problems such as “count-to-infinity”. [15]. As an example, consider figure 2a where we assume the cost function to be the number of hops, so in this case the cost of each link is 1. B calculates its distance from C as 1 and A calculates it as 2. Now suppose the link between B and C goes down (figure 2b) and B does not distribute this new knowledge in time, so B recalculates its cost to C as 3 based on the information received from A, which is at a distance of 2 from C. This information is then transmitted to the the neighbors (A in this case) which recalculates its distance to C as 4 hops. This exchange of information between A and B will continue till infinity and the packets keep on bouncing between the two until they expire. To avoid this situation, Routing Information Protocol (RIP), a variant of distance vector defines a maximum count of 16 after which the bouncing back and forth of packets is discarded [16]. Besides defining maximum counts, various other improvements occurred in distance vector algorithms including split horizon with poisoned reverse, triggered updates etc. However, scalability of the network only enhanced the problems associated with distance vector. This paved the way for a more efficient routing mechanism – *link state*.

2.2 Link State Routing

Link state routing tells every router on the network about its neighbor. It is based upon “a distributed map”, where each router has a copy of the map and it is regularly updated. The goal of link state routing is for each peer to have an identical picture of the state of the entire network; link state protocols require each router to know whether a link is up or down and which cost it has, and then calculate the total cost to reach a destination.

The following are the significant phases of the algorithm:

1. When a router is booted, its first task is to gather information about its neighbors. This is achieved by sending a HELLO packet on each point-to-point link. Upon receiving this information, a router replies back with its unique identifier. Routers on a LAN may receive more than one replies since two or more routers may be connected to the same LAN.

2. The link state algorithm requires each router to know the delay or cost to each of its neighbors. This can be done by sending a special ECHO packet to the immediate neighbors, which should return it immediately such that the so-called "turn-around time" can be observed. The turn-around time can be divided by 2, hence yielding an estimate of the delay to the sender. To be more accurate, several ECHO packets can be sent to the receivers and an average of them can be estimated.

3. The gathered information is used to build a packet containing all this data by the routers. This packet is then broadcast to every router that can answer to this protocol, a process known as *flooding*, which means that it sends the information to all of its neighbors which in turn send it to all of their neighbors and so on. Soon, all routers on the network have this information.

4. The neighbor information is flooded periodically and whenever there is a routing-significant change in the network.

5. As every router knows everything about the network by structuring the information from other routers, it can calculate the best path to any host on any destination network at any time by using *Dijkstra's* "Shortest Path First" algorithm.

3 Routing in NoCs

Currently most of the proposals for NoCs are based upon static routing mechanisms [4], [8]. Although a simple mechanism, static routing is unable to handle communication when link or router fail. Also unlike computer networks, if a link fails, manual intervention is not possible to re-route the communication on-chip. Hence a mechanism is required to dynamically route packets in case of physical failures. From the above discussion of two famous dynamic routing mechanisms, its clear that link state is quite an efficient but very complex routing algorithm. The routing tables can grow endlessly with an increase in the number of routers in the network. However, in an NoC context, the complexity of link state routing can be simplified by implementing static routing tables. These tables will never grow based on the fact that no new nodes will be added once the chip is off the factory. In case when a link goes down, only relevant entries need to be removed from the tables. Also no periodic updates of neighbors is needed as there are no manual interventions to change the topology of the NoCs as in case of regular networks. Hence, this way we can counter the inherent problem of scalability of link state routing and can make it a feasible solution for NoCs.

What follows is a simplified version of our proposed dynamic routing mechanism for NoCs;

1. The network configuration can be set in the beginning as the topology, number of nodes and the distance between immediate neighbors is known in advance. Thus, the cost of each neighbor can be calculated in the beginning. Based on this information, routing tables are built and stored in every router.

2. Whenever there is any significant change in the network – a link is down or a router malfunctions – ECHO packets can be sent to calculate the cost toward the alternate paths and the network configuration can be changed accordingly; otherwise, routing tables will be built using the last known configuration. There is no need to frequently send the updates over the network.

3. In case of a topology change, best paths can be calculated using Dijkstra's Shortest Path First algorithm.

We believe that without the provision of a dynamic routing protocol in NoCs, it would be unrealistic to produce a reliable chip which would continue to exhibit uniform performance over its lifetime. Furthermore, it may also be helpful in case of congestion at the links or routers (which will be discussed in the next section).

4 Congestion control

Congestion occurs in computer networks when resource demands exceed the capacity: queues grow until packets are lost. If a connection oriented service is desired, these packets need to be resubmitted by the transport layer. The resubmitted packets fill the queues again until, eventually, the network is blocked. As the bandwidth of network links increases, one might be tempted to believe that congestion will eventually go away[2], whereas in fact, the problem becomes more severe as link bandwidths grow more and more diverse [17].

In NoCs, IP blocks that may generate an arbitrary amount of traffic are connected. Depending on the path choice (routing again) and the number of such blocks, congestion may be inevitable. Consider fig. 3: here, we have a

[2]This is the myth of overprovisioning; a common phrase in this context is to "throw bandwidth at the problem".

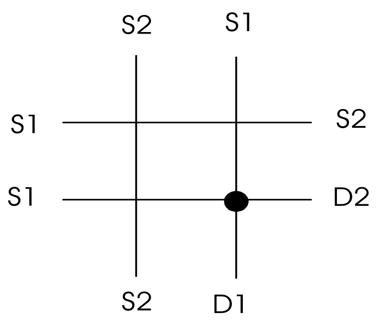

Figure 3. Congestion in a NoC

simple grid-like topology with six senders and two receivers at its ends; S1 and S2 represent sources which transmit data to destinations D1 and D2, respectively. From the way these resources are combined in the network, we can tell that congestion might occur at the marked router if sources generate too much traffic. Common solutions to overcome this issue are contention based (e.g., hot potato) routing, where packets are simply sent onto another link if the link of choice is occupied.

Quite obviously, if there is so much traffic that *all* links that a router is connected to are congested and it does not have a buffer, packets can only be dropped. Buffers can compensate for short bursts, but, as the Internet research community knows very well, they cannot accommodate each and every possible situation, no matter how large they are [18] (and it is also reasonable to assume that on-chip buffers are small [19]). This means two things:

1. mechanisms that cope with congestion must be deployed
2. packets must be retransmitted if they are dropped in order to realize a reliable communication service

In the Internet, it was decided to realize both these functions in the endpoints; this is due to a well known rule called the "end-to-end" argument, which helps ensure scalability [20]. Clearly, such an approach would also be recommendable in NoCs, which share this goal with the Internet.

While the applicability of related mechanisms found in the Internet (store-and-forward routing with a variety of queuing mechanisms[3] and highly complex end-to-end protocols such as TCP [21]) is limited, it may in fact be possible to transfer protocols such as "Congestion Avoidance with Distributed Proportional Control" (CADPC) [22] to NoCs: CADPC manages to be both efficient and much simpler than TCP by dropping the requirement of TCP-friendliness, i.e. the mechanism is not compatible with TCP and cannot be used in the public Internet without ensuring proper isolation.

[3]Note that these mechanisms would also work without queues — e.g., TCP mainly relies on loss and not queuing delay as a congestion indicator.

Since congestion control would be an entirely novel function in NoCs, it is up to NoC designers to set the standard at this point – downward compatibility is not necessary, and suitable mechanisms can be built from scratch. However, care must be taken: once a mechanism is designed and it becomes the de facto standard, Internet experience clearly shows that it may in fact *never* be possible to replace it with radically different technology.

5 Conclusion

In this paper we introduced the idea of dynamic routing and congestion control in NoCs. We have argued that, in order to produce fault-tolerant SoCs, it would be inevitable to incorporate dynamic routing and congestion control on a chip. This fact is underlined by the increasing physical effects of deep sub-micron technology as a chip scales in size.

In the first part of the paper, we have discussed two popular dynamic routing algorithm classes; distance vector routing and link state routing with their pros and cons. We have proposed a dynamic routing algorithm for NoCs that is based on link state routing. The second part of our work emphasized a different aspect: the growing amount of NoC traffic can ultimately lead to congestion and thus packet loss, which ought to be avoided. Furthermore, the heterogeneity of NoCs comprised of IP blocks with varying link speeds will necessitate performing congestion control even in these networks.

We hope that our ideas will not only benefit the NoC designers to come up with fault tolerant designs but may also inspire the network research community to actively contribute their ideas.

References

[1] Luca Benini and Giovanni De Michelli "Networks on Chips: A New SoC Paradigm", *DATE 2002,* March 3 - 7

[2] Ahmed Jerraya, Hannu Tenhunen and Wayne Wolf, "Multiprocessor System-on-chips, *IEEE Computer magazine*, July 2005, Vol. 38, No. 7

[3] Érika Cota, Luigi Carro, Flávio Wagner, Marcelo Lubaszewski, "Power Aware NoC reuse on the testing of core based systems", *International Test Conference ITC 2003,* September 30 - Oct 02, 2003, Charlotte NC USA

[4] William J. Dally and Brian Towles, "Route packets, not wires: on-chip interconnection networks", *Pro-*

ceedings of the Design Automation Conference (DAC), pp. 684-689, Las Vegas, NV, June 2001

[5] Shashi Kumar, Axel Jantsch, Juha-Pekka Soininen, Martti Forsell, Mikael Millberg Johny Oberg, Kari Tiensyrja and Ahmed Himani, "A network on chip Architecture and Design Methodology", P*roceedings of the IEEE computer society annual symposium on VLSI*, 2002

[6] Pierre Guerrier, Allen Greiner, "A generic architecture for on-chip packet switched interconnections", *Proceedings of Design, Automation and Test in Europe conference and Exhibition*, 2000

[7] Ahmed Hemani, Axel Jantsch, Shashi Kumar, Adam Postula, Johny Oberg, Mikael Millberg, Dan Lindqvist, "Networks on a chip: An Architecture for billion transistor era", *Proceedings of the IEEE NorChip Conference*, November 2000

[8] Bart Vermeulen, John Dielissen, Kees Goosens, Kalin Ciordas, "Bringing Communication Networks on chips: Test and Verification Implications", *IEEE Communications Magazine*, 41(9):74-81, September 2003

[9] S.K. Shukla and R.I. Bahar, "Nano, Quantum amd Molecular Computing, Implicaitons to High Level Design and Validation", *Boston, Dordrecht, London: Kluwer Academic Publishers*, 2004

[10] Ming Shae Wu and Chung Len Lee, "Using a Periodic Square Wave Test Signal to Detect Cross Talk Faults", *IEEE Design and Test of Computers*, Volume 22, Issue 2, March-April 2005 Page(s):160 - 169

[11] *http://public.itrs.net/Files/2001ITRS/Home.html*

[12] M. A. Breuer, S. K. Gupta and T. M. Mak, "Defect and Error Tolerance in the Presence of Massive Numbers of Defects", *IEEE Design and Test*, Vol. 21, No. 3, pp. 216-227, May-June 2004

[13] S. Sirisantana, B. C. Paul and K. Roy, "Enhancing Yield at the End of the Technology Roadmap", *IEEE Design and Test of Computers*, Vol. 21, No. 6, pp. 563-571, Nov-Dec 2004

[14] Yi-Ran Sun, "Simulation and Performance Evaluation for Networks on Chips", M.Sc. Thesis, *Department of Microelectronics and Information Technology*, Royal Institute of Technology, Stockholm, Sweden, Dec., 2001.

[15] Christian Huitema, "Routing in the Internet", *Second Edition, Prentice Hall*, NewJersey, 2000.

[16] C. Hendrick. "Routing Information Protocol. RFC 1058", *IETF Network Working Group*, June 1988. Category: Historical.

[17] Sally Floyd and Kevin Fall, "Promoting the Use of End-to-End Congestion Control in the Internet", *IEEE/ACM Transactions on Networking*, August 1999.

[18] , K. G. Ramakrishnan, S. Floyd and D. L. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", *RFC 3168*, September 2001.

[19] Andrei Radulescu and Kees Goossens, "Communication Services for Networks on Silicon", in Shuvra Bhattacharyya and Ed Deprettere and Juergen Teich, editors,"Domain-Specific Processors: Systems, Architectures, Modeling, and Simulation", Marcel Dekker, December 2002

[20] J. H. Saltzer, D. P. Reed and D. D. Clark, "End-to-End Arguments in System Design", *Proceedings of the Second International Conference on Distributed Computing Systems*, April 1981, pp. 509-512.

[21] Van Jacobson, "Congestion Avoidance and Control", *Proceedings of SIGCOMM* 1988, pp. 314-329

[22] Michael Welzl, "Scalable Performance Signalling and Congestion Avoidance", *Kluwer Academic Publishers*, 2003