A Dynamic Routing Mechanism for Network on Chip

Muhammad Ali, Michael Welzl
Institute of Computer Science
University of Innsbruck, Austria
{Muhammad.Ali, Michael.Welzl}@uibk.ac.at

Sybille Hellebrand
Institute of Electrical Engineering
University of Paderborn, Germany
Sybille.Hellebrand@date.upb.de

Abstract

With an increase in the number of transistors on-chip, the complexity of the system also increases. In order to cope with the 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 in the context of NoCs and explains how they could be applied to cope with adverse physical effects of deep sub-micron technology.

Keywords: NoC, self-healing, dynamic routing

1. Introduction

Recent advances in technology is driving the current silicon die to accommodate billions of transistor in the near future leading to design of very complex Systems-on-Chip (SoCs)[1]. However, a study of current SoC infrastructure reveals that buses – existing interconnect medium for SoCs – are posing a serious threat toward achieving the era of billion transistors because of their poor scalability beyond a certain number of partners [1], [2]. To counter such a situation, researchers from the chip design area have explored not only large-scale multiprocessors systems but also traditional computer networks which has already dealt with issues such as scalability (e.g, Internet technology scales very well). This research has yielded a novel interconnect architecture – Network on Chip (NoC) [1], [3], [4], [5] .

This paper focuses on a major area regarding the communication on-chip; that is, how to deal with links or/and routers that become unavailable either temporarily or permanenetly. To the best of our knowledge, this area is not yet addressed by the research community which is probably due to the assumption that networks on chip are considered to be stable and, hence, no such failures are supposed to occur [6]. 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. For example crosstalk faults can lead to permanent or transient failures of the communication links [7]. 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 [8]. 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 mechanism for NoCs which can automatically calculate alternate paths in case of link/router failures on a chip.

2. Routing in NoCs

Routing is the act of moving information from a source to destination following the shortest possible path available (see figure 1). In the Internet, packet based communication was introduced to replace the circuit switching in order to bring reliability to communication. However, time proved that it played a vital role in the scalability of the Internet. Packet based communication has been brought to NoCs but loses the original advantage of reliability in the absence of dynamic routing as is being implemented in the Internet [9]. Currently, most of the proposals for routing in NoCs are based upon static routing mechanisms – XY-coordinate discipline [3], [6]. This is a simple mechanism but in case of link failures, a mechanism is needed to route packets dynamically. Dynamic routing, as the name shows, is used to dynamically discover routes in case

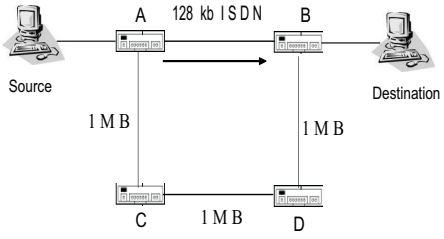

Figure 1. Shortest Path Routing

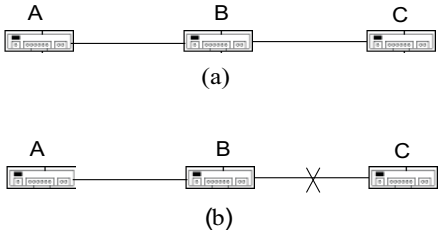

Figure 2. Counting-to-infinity problem

of path changes.

In the following we describe the two most popular dynamic routing algorithms in networks and discuss how they can be utilized in case of NoCs.

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 (e.g., node A in figure 1 has no knowledge about the link between C and D). 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 [9]. The router in this case knows the distance to each destination. Such a technique is quite feasible where the links are of similar speed, as it calculates only the hop count to destinations and chooses best routes according to the hop count metric.[1]

However, distance vector mechanism can lead to a very serious problem called as "count-to.infinity". To explain this phenomenon, lets consider figure 2a. We assume 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. Although various solutions to the problems associated with distance vector have been proposed in the internet standards like split horizon with poisoned reverse, defining the maximum count (e.g, the "Routing Information Protocol" (RIP) defines the maximum count as 16), and triggered updates [9], [10] etc, the inadequacy of this mechanism lead the designers to find a more efficient method – link state routing.

[1] This is the most common case, where all link costs are set to 1.

2.2. Link State Routing

In contrast to distance vector routing which shares its routing table with immediate neighbours only, the link state routing algorithm tells every router on the network about its neighbour routers. The mechanism 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.

Following is a brief description of the significant steps that a re carried out by the link state algorithm;

1. On startup, a router gathers information about its peers by sending a HELLO packet on each point-to-point link. Upon receiving this packet, a router replies to the sender with its unique identifier.

2. A router needs to calculate the delay or cost of each of its neighbors by sending an ECHO packet which should return it immediately in order to observe the turn-around time.

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.

Link state routing is an efficient but a very complex protocol. 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.

Following is a simplified form of link state algorithm for NoCs which we call as NoC-LS (NoC link state);

1. The network configuration can be set in the beginning as the topology, distance between the nodes and their number are known in advance. 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.

3. Simulation Environment

In order to support our proposal, we have simulated a 4x4 mesh based NoC using Network Simulator (ns-2). We have considered a simple scenario where the nodes *1, 2, 3 & 4* are the senders and nodes *13, 14, 15 & 16* act as receivers (see figure 3a). The senders take the shortest possible path to reach the destination (*e.g 1 - 5 - 9 - 13*). At any time t, the link between node 5 and 9 fails after which the NoC-LS routing algorithm recalculates the alternative paths and resumes the communication (*in this case it will be 1 - 5 - 6 - 10 - 14 - 13*) as shown in figure 3b.

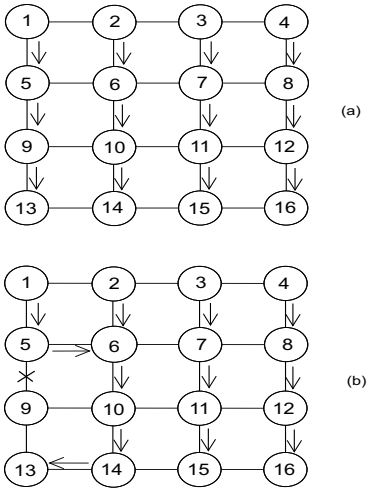

Figure 3. Mesh topology (a) communication path between sender and receiver before link failure (b) communication path after failure

The bandwidth for an on-chip communication link is expected to be in a range of 2.2 GB to 3.2 GB with a packet size of 64 bytes [11]. However, to simplify the simulation process, we have scaled down the bandwidth to 200MB with a packet size of 12 bytes. The data rate is set to 190 MB/s. However, it has been observed that keeping the data rate half of the available bandwidth reduces packet drop to zero with a buffer size of four at each node [4]. However, in real situations where a link may be shared among various communicating partners, it would be wise to assign a much smaller data rate which is easily accommodated in the available bandwidth without incurring any data loss.

3.1. Simulation Results

A comparison of communication with and without NoC-LS routing mechanism is shown in figure 4. In the absence of the dynamic routing mechanism, the throughput of the receiver (*node 13*) is reduced to zero (shown by line), which means that when a link goes down no communication occurrs at that point. The dashed line, on the other hand, shows that the communication between node 1 and node 13 is still active even

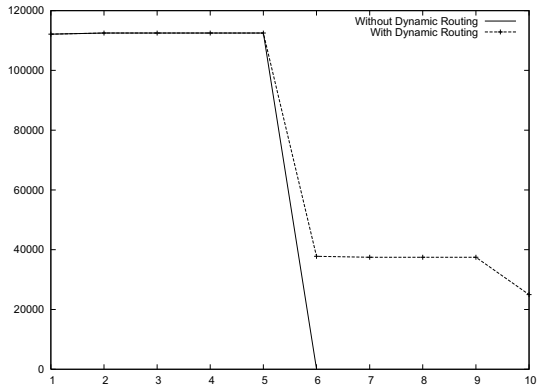

Figure 4. Graph showing throughput with and without NoC-LS (190MB/s)

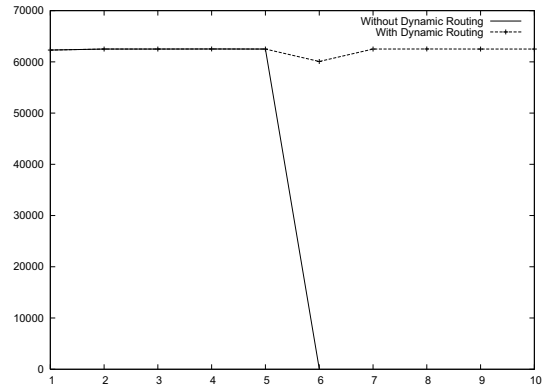

Figure 5. Graph showing throughput with and without NoC-LS (90MB/s)

after the failure of the specified link. However, it can be observed that in case of NoC-LS the throughput is reduced to less that half of the expected bandwidth. It is due to the fact that after the link failure at nodes 5 and 9, the link between nodes 6 and 14 are being shared by two senders simultaneously. Since the bandwidth of the link is almost equal to the rate at which data is being sent (190MB/sec), we see a huge amount of packets being lost. Hence we reduced the data rate to 45 percent of the available bandwidth of the link and the effects can be well seen in figure 5. In this case, when the link goes down there is a slight drop in the throughput but it soon recovers from it. It is during this time that the NoC-LS recalculates the paths and updates its routing tables and then continues with the communication. Also it can be observed that no change in the throughput has occurred before and after the link failure.

4. Conclusion

In this paper we introduced the idea of dynamic routing in NoCs. We argued that in order to produce fault-tolerant NoCs, it would be inevitable to incorporate dynamic routing mechanisms on a chip. This proposal is underlined by the fact that as the chip scales, number of defects may increase.

We have discussed two popular dynamic routing algorithm classes; distance vector routing and link state routing with their pros and cons. We have presented a simplified form of link state routing (NoC-LS) which well suits a NoC. We carried out simulations using ns-2 to show that in case of a link/router failure, the communication can continue by implementing NoC-LS, in contrast to static routing mechanism. Moreover, we also have shown that by reducing the data rate, the packet drop ratio can be reduced to zero.

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 Micheli, “Networks on Chips: A New SoC Paradigm”, *DATE, Design, Automation and Test in Europe, Paris, March 3 - 7, 2002*

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

[3] William J. Dally and Brian Towles, “Route packets, not wires: on-chip interconnection networks”, in Proceedings of the Design Automation Conference (DAC), pp. 684-689, Las Vegas, NV, June 2001.

[4] 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”, Proceedings of the IEEE computer society annual symposium on VLSI 2002.

[5] P.Guerrier and A.Greiner “A generic architecture for on-chip packet switched interconnections, *In Proc. of DATE 2000*, March 2000.

[6] Bart Vermeulen, John Dielissen, Kees Goossens, Kalin Ciordas, “Bringing Communication Networks On Chip: Test and Verification Implications”, *IEEE Communications Magazine, Vol. 41, No. 9, September 2003, pp. 74-81*

[7] 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

[8] http://public.itrs.net/Files/2001ITRS/Home.htm.

[9] Christian Huitema, “Routing in the Internet”, *Second Edition, 2000, Prentice Hall, New Jersey*

[10] C. Hendrick. “Routing Information Protocol”, *RFC 1058, IETF Network Working Group, June 1988. Cat-*

egory: Historical

[11] David Bertozzi and Luca Benini, “Xpipes: A Network-on-chip Architecture for Gigascale System-on-chip”. *IEEE Circuits and Systems magazine*, Second Quarter 2004.