A Fault tolerant mechanism for handling Permanent and Transient Failures in a Network on Chip

Muhammad Ali, Michael Welzl, Sven Hessler
Institute of Computer Science, University of Innsbruck, Austria

Sybille Hellebrand
Institute of Electrical Engineering, University of Paderborn, Germany

Abstract

Network on chips (NoC) have emerged as a feasible solution to handle growing number of communicating components on a single chip. The scalability of chips however increases the probability of errors, hence making reliability a major issue in scaling chips. We hereby propose a comprehensive fault tolerant mechanism for packet based NoCs to deal with packet losses or corruption due to transient faults as well as a dynamic routing mechanism to deal with permanent link and/or router failure on-chip.

Keywords: NoC, fault-tolerance, self-healing, dynamic routing, reliable packet delivery

1. Introduction

Rapid development in silicon technology is enabling the chips to accommodate billions of transistors. It has been observed however that the current on-chip interconnects — *buses* — are becoming a bottleneck as they are unable to cope with growing number of participating cores on a chip [1]. This inability of buses has persuaded VLSI designers to look beyond their current domain and explore parallel architectures and computer networks. This has yielded a novel and scalable design for future interconnects for System on chips (SoC) termed as *Network on chips* (NoC) [1], [2]. This new communication paradigm for SoCs introduces the idea of creating a network of resources on a chip where communication takes place by routing packets between the resources instead of connecting them with dedicated wires [3]. Such a structure will be supported by a set of protocols which provides well defined interfaces in order to separate communication from computation.

As the size of a chip increases, so does the importance of error detection and recovery ("self-healing"); thus, it seems that the reliability of on-chip communication should be a primary issue. Application Specific Integrated Circuits (ASIC) used in today's embedded systems are an integral part of safety critical systems and consumer related products, making fault tolerance a key concern. Shrinking silicon size will lead to enhanced levels of cross talks, high field effects, and critical leakage currents which, in turn, will lead to more temporary and permanent errors on-chip [4]. Crash or permanent failures can occur due to electromigration of a conductor or a connection failure permanently halting the operation of some modules. On the other hand, faults like Gaussian noise on a channel and alpha particles strikes on memory and logic can cause one or more bits to be in error but do not cause permanent failures [5]. This argument strengthens the notion that chips need to be designed with some level of built-in fault tolerance. Furthermore, relaxing the requirement of 100% correctness in the operation of various components and on-chip channels profoundly reduces the manufacturing cost as well as cost incurred by test and verification [6].

Although already available standard diagnosis and fault tolerance tests may be applied to NoCs, they do not exploit any particular network properties like packets being forwarded over the network or links or routers failing. My idea is to fill this gap. In context to NoCs, my idea is to address two major reliability issues, categirized as "soft" and 'hard" errors based on the timescale of their occurrence: firstly, transient faults can corrupt individual packets causing them to be misrouted or invalid, in which case a retransmission is required. Secondly, due to electromigration, cracks, or dielectric breakdowns, links and/or routers become permanently unavailable causing them to stop functioning ("hard error"). I believe that unless there is a robust fault tolerant mechanism which stands feasible for resource-constrained NoCs, the future of SoCs based on billions of transistors would be precarious.

2 State of The Art

Based on the nature and time of occurrence of errors on-chip, we categorize them into two groups as described below;

2.1 Error recovery from transient faults

Transient failures can occur on a chip for many reasons: alpha particles emitted by trace uranium and thorium impurities in packages and high-energy neutrons from cosmic radiations can cause soft errors in semiconductor devices. Similarly, low energy cosmic neutrons interacting with isotope boron-10 can cause soft errors. These events, generally called single-event upsets, can affect the storage elements of a chip such as latches, memory and registers [7]. Furthermore, with shrinking dimension of silicon die, particle collision is more likely to impact the stored charge sufficiently enough to change its state. In addition to this, implementing packet-based communication on-chip brings new reliability related challenges along with it.

A packet usually consists of a *header* and a *payload*. the header of the packet mainly contains a unique packet ID, source and destination addresses, routing information, error recovery codes etc. The payload of a packet contains the actual data. A transient fault can either corrupt the header or payload of a packet. In former case, a bitflip in the destination address, for instance, can misroute the packet to a wrong destination. In latter case, a bit scramble can make the packet invalid. In both situation, a retransmission is requested by the receiver. Error control can be implemented at either *link level* or *system level* (end-to-end level) [8]. At *link level* error control, routers at the edges of a link work together to deal with transient faults. Each router stores and checks the incoming flit[1] before forwarding it to the next router in the path. Alternatively, error control can be implemented at the *system level*, that is, in the end systems. In this case, intermediate routers do not need to store and check packets for inconsistency and are only required to route them according to the routing mechanism implemented.

Transient faults are usually modeled with a bit-error rate (BER). Although it is quite difficult to get the precise frequency of soft errors as it mainly depends upon the physical location in which the system is present; a study of fault-tolernace literature however revels that it may lie in the range of 10^{-9} and 10^{-20} BER [5].

Due to its simplicity and need for limited processing and storage requirements, along with low transient fault rate as described above. we implement the reliable protocol delivery mechanism in the end systems — source and destination. In this case, a single ack informs the sender about the correct reception of a predefined number of packets. In addition, we use a nack to inform the sender of an error as soon as possible such that the problem can be repaired immediately and the sender side buffer can be kept small. A detailed description of the protocol is given in section 3.

[1] each packet is sub-divided into equal sized smaller chunks called as flits

2.2 Permanent errors in a NoC

Permanent faults, as the name shows, cause permanent damage to the circuit. These faults cause physical changes in the circuits whose behavior does not change with time [5]. Electromigration of a conductor, broken wires, dielectric breakdowns etc. are a few examples of permanent failures on chip. Permanent faults, at one level, are modeled as stuck-at faults[2], or as fail-stop model where a complete module[3] malfunctions and informs its neighbors about its out-of-order status [8]. Since we consider a behavioral design paradigm for NoCs, hence we consider a permanent failure as a fail-stop fault where a module after having permanent damage shuts itself down and inform the neighbors about it. Permanent failures are usually described in terms of MTBF[4] which is usually expressed in hours. Such failure rates are mostly expressed in Failures in 10^9 hours (FIT).

Although physical faults are not as common and frequent as transient faults on-chip, in case a component fails however it is impossible to repair or replace it on-chip. In such a case, it is important to reroute the packets on alternate paths so that the communication infrastructure remains intact [9]. The main idea is to keep the chip in functioning state with "graceful degradation"[5] of performance in the presence to faults [10].

We designed a dynamic routing mechanism which is inspired from the Internet, where reacting to link failures has always been a requirement. Our NoC model uses a deterministic routing mechanism where packets are routed on the shortest path available. However, when a link or router fails, the routing tables are updated and routing takes place according to the newly calculated paths. The protocol is described in detail in section 4.

2.3 Simulation Environment

We have used *Network Simulator ns-2* for implementing our reliability protocols [11]. ns-2 is an open source, object-oriented and discrete event driven network simulator written in C++ and OTcl. Its a very common and widely used tool to simulate small and large area networks. Due to similarities between NoCs and networks, ns-2 has been a choice of many NoC researchers to simulate and observe the behavior of a NoC at a higher abstraction level of design [12], [13], [14]. It has a huge variety of protocols and various topologies can be created with little effort. Moreover, customized protocols for NoCs can easily be incorporated into ns-2. The parameters for routers and links can easily be scaled down to reflect the real situation on a chip.

[2] where a logical node is stuck at either logical zero or one

[3] a module may be a link or a router

[4] Mean Time between failures

[5] after a fault has been detected, the system is reconfigured and continues its work with possibly reduced performance of performance in the presence of faults

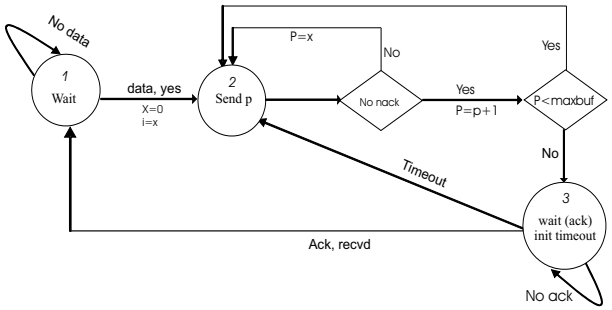

Figure 1. State diagram for sender protocol

We have simulated a 4x4 2D mesh NoC using ns-2. The chip is assumed to be of size 22mm x 22mm with link size of 4.5mm each. The capacity of each link is 2 Gbits/sec with a link delay of 10 nanosecs. We use packet based communication where the whole packet is generated and received by a router. We carried out various simulations and the performance evaluations for both protocols (packet delivery and dynamic routing) are discussed in their respective sections.

3 Reliable packet delivery protocol

Since our protocol involves a distinct sender and receiver, we have divided the protocol description into two parts. The state diagrams of both the sender side and receiver side is given in the figures 1 and 2. The sender buffers and sends a set of data packets continuously at a constant data rate after which it waits for a single acknowledgment (ack) from the receiver. Once the ack is received the sender relinquishes the buffers and fills it up with next set of data packets. If the sender does not receive the expected ack within a certain time limit, it resends the same set of data. The process continues until it gets an ack from the receiver.

The receiver side receives the packets in sequence and issues an ack telling the sender of successful reception. In case the receiver does not receive a packet at all, or the CRC code detects data corruption in the payload, a negative acknowledgment (nack) is generated asking the sender to resend the missing/corrupt packet. Since we employ *go-back-N* retransmission mechanism at the receiving end, the receiver discards the incoming packets until it receives the expected packet.

Timeouts are implemented on both sender and receiver sides. The sender after sending the set of packets, waits for an ack until a certain timeout occurs. The timeout depends upon the path the set of packets traverse plus one ack packet. Hence, it can be expressed as $L_{delay \times hops}(bs + 1)$, where $L_{delay \times hops}$ represents the product of link delay of each link and number of switches the packet traverses until reaching the destination, and *bs* is the buffer size. Similarly, the receiver, after generating a nack, waits for timeout $2 \times L_{delay \times hops}$ after which it resends the nack signal for the missing / corrupt packet.

This mechanism is quite simple and yields less latency as a set of packets are acknowledged with a single ack. It is possible that the number of packets to be sent is less than the prescribed size of buffers, that is, if the predefined size of buffer is 10 and there are only 5 packets to be sent. In such a case empty packets with zero payload can be added to the list to fill the buffers. This however may add extra data traffic to the network. Alternatively, the last packet may carry a special flag indicating the receiver about the end of the current packet stream.

One of the significance of this protocol is that no buffering is required at the receiver side as we employ *go-back-n* retransmission policy where the receiver receives all the incoming packets in order. If a packet is corrupt or missing, the receiver requests for a retransmission and discards all the incoming packets until it receives the required packet. The *go-back-n* mechanism is in contrast to *selective repeat* where the receiver keeps all the incoming packets in its buffers even if they are out of order. This mechanism requires complex re-ordering algorithms besides buffers at the receiver side. Considering the scarcity of storage capacity on-chip besides simplicity, we have employed *go-back-N* mechanism in our protocol.

3.1 Performance evaluation of reliable packet delivery protocol

Consider figure 3 which shows the total throughput achieved with increasing error rate and different buffer sizes. Its quite clear from the figure that the overall throughput increases with increasing buffer sizes besides increasing error rate. However, it has been observed that in the presence of errors, the packet overhead also increases with increasing buffer size. The overhead (O) is defined as the goodput, e.g. packets with an increasing sequence number, divided by the number of packets transferred in total. For example, let the error rate be 0, and the buffer size (bs) be 1. Then, the overhead of our protocol is 50% as for every data packet an acknowledgment packet (an **ack**) has to be transmitted.

Similarly, for an error rate of 0 and a buffer size of 10 the expected overhead will be 9%. Thus, for a error rate of 0, the overhead could be expressed as: $O_{bs} = \frac{1}{(bs+1)}$ and $\lim_{bs \to \infty} O_{bs} \to 0$ This is underlined by our experimental results shown in figure 4. The reason behind this behavior is due to the inherent nature of the protocol: if at buffer size 20, packet 3 is corrupt, then sender after receiving the nack resends the packet 3 along with all following it. This leads us to choose a suitable buffer size which incurs minimum overhead. The figure 4 shows that using a buffer size of 10 minimizes the overhead for most of the examined error rates.

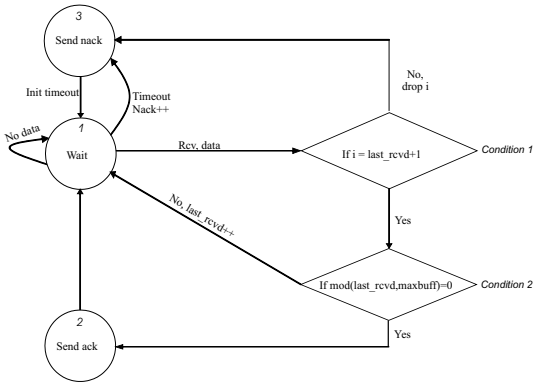

Figure 2. State diagram for receiver protocol

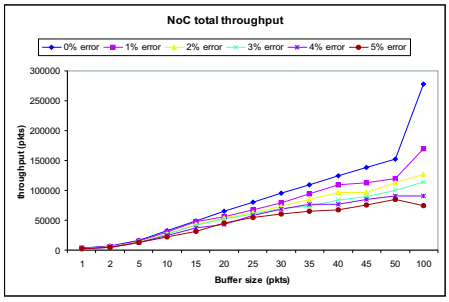

Figure 3. Total throughput achieved with NoC protocol

4 Dynamic routing protocol for NoCs

There are two major classes of dynamic routing protocols in the Internet; *Distance Vector* and *Link State* [15]. In *distance vector* routing, each router shares its information only with its direct neighbors. Though simple, distance vector leads to well-known problems like counting-to-infinity, bouncing effect etc.[6]. In contrast, *link state* mechanism tells every router on the network about its neighbors with periodic updates. Each router sends the copy of its routing table to every adjacent router which updates its tables and shares it with others. This way, in the end, the network converges to a stable state where each router has an identical copy of the routing table. In the internet, nodes are very frequently going down due to which link state routing makes periodic updates. This makes link state an efficient but complex mechanism; routing tables can grow significantly when new nodes are added to the network, or periodic updates can flood the network limiting the scalability of the protocol.

Neither of these mechanisms are suitable in their current form for NoCs because of excessive complexity and huge logic resources. As mentioned earlier that permanent faults are extremely low in NoCs as compared to the internet. The network is relatively stable on-chip as compared to off-chip networks. Also most components monitor their current state and inform the connected resources about their state if they malfunction, making periodic updates unnecessary. We therefore, propose a dynamic fault tolerant routing scheme where routing tables are initialized in advance and hard-coded in the system. When a module fails it informs its neighbors which send a special packet to all the routers in the network on receiving which they update their routing tables. After updating the routing tables, shortest path is calculated a shortest path algorithm. Hence, the overall communication continues with graceful degradation of service.

[6] A detailed reference to the problems in distance vector routing is given in [15]

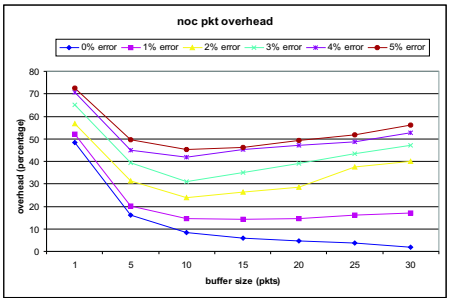

Figure 4. Effect on packet overhead with varying buffer sizes and error rates

4.1 Dynamic routing protocol for NoCs (DR-NoC

In what follows, we will explain the major steps that are carried out by the proposed dynamic routing protocol for NoCs;

1. The routing tables are appropriately initialized in advance as the topology, link "costs"[7] (typically 1 for shortest path routing) and the number of nodes and links are known in advance when the chip is built. Unlike the Internet, a faulty component in a NoC will not be replaced neither a new component is added, hence no new components will be added to the network.

2. When a link or a router goes down, the failing component informs the adjacent modules about its failure after which the adjacent nodes send a special packet to all the routers informing about the failed component.

3. The routers after receiving the information remove that particular link or router from their routing

[7] distance value between two nodes

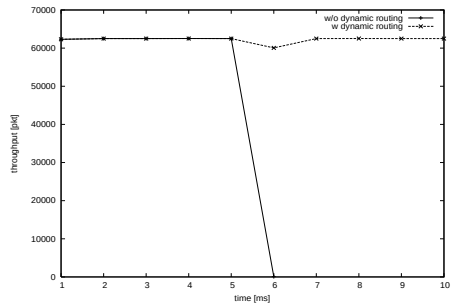

Figure 5. Throughput w/without dynamic routing protocol

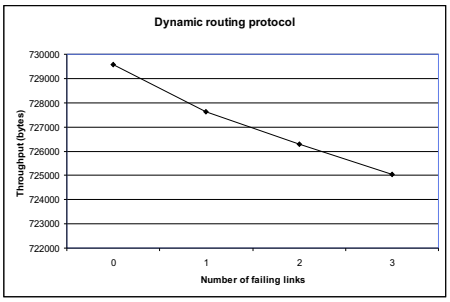

Figure 6. Throughput against many links failing

tables and exchange this information with their neighbors.

4. After all the routers have the identical copy of the routig table, each router calculates the shortest paths using Dijkstra's "Shortest Path First" algorithm.

4.2 Performance evaluation of our dynamic routing protocol

Consider figure 5 which shows basic implementation of our dynamic routing protocol. At time 5ms, a link goes down and in the absence of *DR-NoC*, the communication drops to zero. On the other hand, we can see that with *DR-NoC* protocol, the communication continues showing the alternate routes have been calculated. A slight drop here is due to the recalculation of the new routes.

Figure 6 shows a comparison of total throughput against number of nodes randomly failing in a NoC. It can be observed that although the overall throughput decreases, yet the communication is still alive clinging to the notion of graceful degradation of service. Since some packets may be lost due to link failures, however, in the presence of our reliable packet delivery protocol, all those packets will be retransmitted.

5 Related Work

The authors of [16] have already argued about the significance of retransmission mechanisms for NoCs in order to ensure reliability. In [17], Bertozzi et al. present a link level flit-based error control model. Every intermediate router checks the incoming flit for accuracy and generates acks for successful reception of the flit. Besides acks, nacks are generated for missing flits. Similarly, the authors of [18] present a flit-based hop-by-hop retransmission mechanism with retransmission architecture. Each router maintains a retransmission buffer in addition to the regular transmission buffers. The flits, after being injected into the network, are moved into the retransmission buffer where they are kept for three clock cycles — which is the minimal latency overhead in the event of an error. If an error is detected, the flit is resent from the retransmission buffer otherwise, after three clock cycles, next coming flits are moved into it. Although effective, these mechanisms require fault tolerant logic at each router besides maintaining retransmission buffers.

The authors of [6], [19] present a stochastic communication model for NoCs based upon randomized rumor spreading. Such a method employs a probabilistic flooding algorithm where, if a node has a data packet to send, it will be forwarded to a randomly chosen set of tiles in the neighborhood and this way the packet will be flooded to the whole network of nodes and finally make it to the destination. However, these models are not deterministic and can become a bottleneck when network load is heavy.

6 Conclusion and future work

We have presented a comprehensive fault tolerant mechanism to deal with permanent and transient faults in a NoC. We have introduced the idea of dynamic routing in NoCs for coping with hard errors. We argued that in order to produce fault-tolerant NoCs, it would be inevitable to incorporate such mechanisms on a chip. This proposal is underlined by the fact that, as the chip scales, the number of defects are likely to increase. Furthermore, we have designed a protocol for reliable delivery of packets from source to a destination. We have defended our mechanism with simulations and showed that on-chip communication survives in the presence of transient and permanent faults.

In future we intend to extend our scheme with congestion control functionality so as to maintain the highest possible sending rates at all times while avoiding packet drops. We expect that, with such a mechanism in place, our scheme will be able to cope with high traffic loads even when a significant number of links fail. Since congestion control mechanisms require a relatively stable routing behavior, we consider it unlikely that this problem can be solved in conjunction with hot potato

routing, where each individual packet of a single data stream can take a different route.

References

[1] Luca Benini and Giovanni De Micheli, "Networks on Chips: A New SoC Paradigm", *Proceedings, Design, Automation and Test in Europe*, Paris, March 3 - 7, 2002

[2] 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, IEEE computer society annual symposium on VLSI*, pp. 117 - 124, April 25 - 26, 2002

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

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

[5] Michael L. Bushnell, Vishwani D. Agarwal, "Essentials of Electronic Testing for Digital, Memory and Mixed-Signal Circuits", *Kluwer Academic Publishers*, USA, 2000

[6] T. Dumitras, R. Marculescu, "On-Chip Stochastic Communication", *Proceedings, Design, Automation and Test in Europe*, pp. 790-795, 3-7 March, 2003, Munich, Germany

[7] S.K. Shukla and R.I. Bahar, "Nano, Quantum and Molecular Computing, Implications to High Level Design and Validation", *Kluwer Academic Publishers*, Boston, 2004

[8] W. Dally and B. Towles, "Principles and practices of Interconnection Networks", *Morgan Kaufmann Publishers*, USA 2004.

[9] Muhammad Ali, Michael Welzl, Martin Zwicknagl, Sybille Hellebrand, "Considerations for fault-tolerant Network on chips, *Proceedings, 17th International Conference on Microelectronics (ICM)*, Islamabad, Dec. 13 15, 2005

[10] D. Sigenza-Tortosa and J. Nurmi, " Proteo: A New Approach to Network-On-Chip", *Proceedings, IASTED-Communication Systems and Networks (CSN 2002)*, Malaga, Spain, 2002

[11] http://www.isi.edu/nsnam/ns/

[12] Vu-Duc Ngo, Hae-Wook Choi, "On Chip Network: Topology design and evaluation using NS2 ", *Proceedings, 7th International Conference on Advanced Communication Technology (ICACT 2005)*, Phoenix Park, Korea, Feb. 21-23, 2005

[13] Y.-R. Sun, S. Kumar, and A. Jantsch, "Simulation and evaluation of a network on chip architecture using ns-2", *Proceedings, 23rd IEEE NorChip Conference*, Finland, November 2002

[14] Alireza Vahdatpour, Ahmadreza Tavakoli, Mohammad Hossein Falaki, 'Hierarchical Graph: A New Cost Effective Architecture for Network on Chip", *Proceedings, International Conference on Embedded And Ubiquitous Computing*, Nagasaki, Japan, December 2005

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

[16] S. Murali, L. Benini et al. "Analysis of Error Recovery Schemes for Networks on Chips", *IEEE Design and Test*, Vol.22 ,Issue 5, Sep.2005

[17] D. Bertozzi et al., "Xpipes: A Network-on-chip Architecture for Gigascale System-on-chip", *IEEE Circuits and Systems*, 2nd quarter, 2004, pp. 18-21

[18] D.Park, C.Nicopoulos et al. "Exploring Fault Tolerant Network-on-Chip Architectures", *Proceedings, DSN-2006*, pp. 93-102

[19] M. Pirretti, et al. "Fault Tolerant Algorithms for Network-On-Chip Interconnect, *Proceedings, ISVLSI 2004*, pp. 46-51

[20] Michael Welzl, "Network Congestion Control: Managing Internet Traffic", *John Wiley & Sons*, August 2005