An efficient fault tolerant mechanism to deal with permanent and transient failures in a network on chip

Muhammad Ali,* Michael Welzl and Sven Hessler

Institute of Computer Science,
University of Innsbruck, Austria
E-mail: Muhammad.Ali@uibk.ac.at
E-mail: Michael.Welzl@uibk.ac.at
E-mail: Sven.Hessler@uibk.ac.at
*Corresponding author

Sybille Hellebrand

Department of Electrical Engineering,
University of Paderborn, Germany
E-mail: Sybille.Hellebrand@date.upb.de

Abstract: Recent advances in the silicon technology is enabling the VLSI chips to accommodate billions of transistors; leading toward incorporating hundreds of heterogeneous components on a single chip. However, it has been observed that the scalability of chips is posing grave problems for the current interconnect architecture which is unable to cope with the growing number of components on a chip. To remedy the inefficiency of buses, researchers have explored the area of computer networks besides exploring parallel computing to come up with viable solutions for billion transistor chips. The outcome is a novel and scalable communication paradigm for future System on Chips (SoCs) called as Network on Chips (NoC). However, as the chip scales, the probability of both permanent and temporary faults is also increasing, making Fault Tolerance (FT) a key concern in scaling chips. Alpha particle emissions, Gaussian noise on channels are some of the reasons which introduce transient faults in the data. Besides that, due to electromigration of conductors, corrosion or aging factors, on-chip modules or links may suffer permanent damage. This paper proposes a comprehensive solution to deal with both permanent and transient errors affecting the VLSI chips. On the one hand we present an efficient packet retransmission mechanism to deal with packet corruption or loss due to transient faults. On the other hand, we propose a deterministic routing mechanism which routes packets on alternate paths when a communication link or a router suffers permanent failure.

Keywords: network on chips; NoC; fault-tolerance; self-healing; dynamic routing; reliable packet delivery.

Reference to this paper should be made as follows: Ali, M., Welzl, M., Hessler, S. and Hellebrand, S. (2007) 'An efficient fault tolerant mechanism to deal with permanent and transient failures in a network on chip', *Int. J. High Performance Systems Architecture*, Vol. 1, No. 2, pp.113–123.

Biographical notes: Muhammad Ali received an MSc in Computer Science from the University of Peshawar, Pakistan, in 1997. Since then he has been working at the Institute of Management Sciences (IMSciences), Peshawar, Pakistan. Currently, he is on study leave with a fellowship awarded by the Higher Education Commission (HEC) of Pakistan to pursue his PhD at the University of Innsbruck, Austria. His areas of interest are networks on chips, dependability and fault tolerance and routing protocols. He has already published over ten research papers in the international conferences and journals.

Michael Welzl works as a Postdoc at the University of Innsbruck, Austria. His PhD thesis was published by Springer in 2003, and he published a Wiley book on Congestion Control. In addition to being a Primary Investigator of several funded research projects and a Coordinator of a European project, he co-chairs the IRTF Internet Congestion Control Research Group.

Sven Hessler received an MSc from the University of Leipzig, Germany, in 2003. He works at the University of Innsbruck, Austria, while being enrolled as a PhD student at TU Darmstadt. He published several conference and journal papers, acted as Local Chair of WONS 2007, and also gave talks and taught courses on a diverse range of topics.

Sybille Hellebrand received a PhD in 1991 from the University of Karlsruhe, Germany. She was a Postdoctoral Fellow at the TIMA/IMAG-Computer Architecture Group, Grenoble, France. She served as a Full Professor and as the Head of the Department of Computer Science of the University of Innsbruck. Currently, she is a Professor and Head of the Department of Electrical Engineering at the University of Paderborn, Germany. Her main research interests include BIST for high-quality applications and synthesis of testable systems.

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 (Ali et al., 2006). 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 (SoCs) termed as Network on Chips (NoC) (Benini and De Michelli, 2002; Kumar et al., 2002). 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 (Dally and Towles, 2001). 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 (FT) a key concern. Shrinking silicon die 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 (Ming and Chung, 2005). 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 (Bushnell and Agarwal, 2000). This argument strengthens the notion that chips need to be designed with some level of built-in FT. 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 (Dumitras and Marculescu, 2003).

Although, already available standard diagnosis and FT 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. This paper fills this gap. It addresses two reliability issues, which we categorise 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'). For the first problem, we propose a protocol which ensures reliable delivery of data to the destination by retransmitting corrupt/missing packets. To cope with permanent faults, we propose a deterministic routing mechanism which allows the routers to recalculate new paths after topology changes.

2 Fault tolerance

Many applications of the interconnection networks require high level of reliability and availability. Reliability is becoming an issue of grave concern as of the chip scales. The chips used in today's embedded systems are typically more robust than their counterparts in desktop processors which utilise large, high-density memories making them more vulnerable to soft errors. However, SoCs of embedded systems are to be used in safety critical systems and consumer related products, making reliability a major issue. As the chip scalability reaches physical limits of devices and fabrication technology, designers increasingly have to consider qualitative changes. If the silicon scaling continues at its current pace, designers have to reconsider design methodologies resilient to the variations, defects and noise. Decreasing physical dimensions of silicon chips is posing a serious challenge to the chip manufacturers to maintain the overall chip yield as it is traditionally expected to be. Further, the International Technology Roadmap for Semiconductors (ITRS) states in its 2001 summary document,

> "Relaxing the requirement of 100% correctness for devices and interconnects may dramatically reduce costs of manufacturing, verification, and test. Such a paradigm shift is likely forced in any case by technology scaling, which leads to more transient and permanent failures of signals, logic values, devices, and interconnects."

Elsewhere the document says, "Silicon complexity places long-standing paradigms at risk …fabrication of die with 100% working transistors and interconnects becomes prohibitively expensive".

Finally, addressing error-tolerant design, the roadmap states,

> "The scaling of the design complexity and increasing transistor count will greatly reduce the potential for failures to occur. In this case, relaxing the requirement for 100% correctness in both transient and permanent failures of signals, logic values, devices, or interconnects may reduce the cost of manufacturing, verification, and testing."

2.1 Fault tolerance and defect tolerance

The concept of building useful computational systems with parts that might be initially defective, experience externally induced transient errors or eventually develop a permanent lifetime fault is not new. Researchers addressed these problems as far back as the 1940s with the work of Von Neumann (1995), continuing with emergence of FT computing in the 1960s and then recently, the field of Defect Tolerance (DT).

FT is the ability of a system to continue correct operation of its tasks after hardware or software faults occur (Johnson, 1993). Correct operation typically implies that no errors occur at any system output. Other FT definitions replace the word *correct* with *satisfactory* or *reliable*.

DT on the other hand, refers to any circuit implementation that provides higher yield than an implementation that is not DT, for given levels of defects and process variations. Enhancements in this category include redundancy (often in the form of spares) as well as defect avoidance, in the form of layout and circuit design techniques that reduce a circuit's sensitivity to fabrication defects and process variations.

3 Fault tolerance in NoCs

With regard to packet-based communication on-chip, there are issues that need to be considered while designing a FT system. Since the communication is carried in packets, the packets need some kind of protection in order to make sure that they reach their destination safe and in due time. Error detection or correction mechanism can protect packets from being corrupted or lost. In case of errors however, a packet needs to be retransmitted. The retransmission can be either end-to-end (system level) or switch-to-switch (link level). In a typical retransmission example, the sender adds error detection codes to the original data and sends it to the receiver. The receiver checks the data for correctness and, if found invalid, requests for a retransmission. Alternatively, the sender can add error correcting codes (e.g. Hamming codes) to the data and the receiver can correct the data on reception.[1] In some cases, Hybrid schemes combining both error correction codes and retransmission mechanism are also used. The choice of a particular scheme requires tradeoff among various factors like area-power overhead, on-chip storage and memory constraints, error detection and correction capabilities, error resilience threshold and the cost associated with it.

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

3.1 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 semi-conductor 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 (Shukla and Bahar, 2004). 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 *end-to-end level* (Dally and Towles, 2004). 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[2] before forwarding it to the next router in the path. Alternatively, error control can be implemented at the *end-to-end 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 modelled 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 FT literature however reveals that it may lie in the range of 10^{-9} and 10^{-20} BER (Bushnell and Agarwal, 2000).

Keeping in view the above described facts, we implement the reliable protocol delivery mechanism in the end systems – source and destination. In this case, a single *ACK* inorms 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 5.

3.2 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 behaviour does not change with time (Bushnell and Agarwal, 2000). Electromigration of a conductor, broken wires, dielectric breakdowns, etc. are a few examples of permanent failures on chip. Permanent faults, at one level, are modelled as stuck-at faults[3] or as fail-stop model where a complete module[4] malfunctions and informs its neighbours about its out-of-order status (Dally and Towles, 2004). Since we consider a behavioural design paradigm for NoCs, we consider a permanent failure as a fail-stop fault where a module after having permanent damage shuts itself down and inform the neighbours about it. Permanent failures are usually described in terms of MTBF[5] which is usually expressed in hours. Such failure rates are mostly expressed in failures in 10^9 hr (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 (Ali et al., 2005). The main idea is to keep the chip in functioning state with 'graceful degradation'[6] of performance in the presence to faults (Sigüenza-Tortosa and Nurmi, 2002).

We designed a mechanism inspired from the internet where reacting to link failures has always been a requirement. Our FT 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 7.

4 Related work

Dumitras and Marculescu (2003) present a communication paradigm for NoCs called as *stochastic communication*. More precisely, in a 2D mesh based NoC the IPs communicate using a probabilistic broadcast scheme, very similar to the randomised gossip protocol described by Demers et al. (1987) where a tile has a message that needs to be transmitted, this will be forwarded to a randomly chosen subset of the tiles in the neighbourhood. This way, the messages are diffused through the network to all the tiles. Every IP then selects, from the set of received messages, only those messages whose destination field equals the ID of the tile. The behaviour of this communication scheme is similar, but not identical, to the proliferation of an epidemic in a large population (Bailey, 1975). Although simple, flooding can add a huge amount of redundant traffic to the network especially when a large number of nodes are communicating simultaneously. This can eventually degrade the overall performance of the system.

Bertozzi and Benini (2004) present a NoC architecture termed as Xpipes. The Xpipes architecture is flit-based using wormhole switching mechanism. The reliability in this architecture is achieved by distributed error detection with link level retransmission as error recovery scheme. The retransmission policy implemented in Xpipes is *go-back-n*. Flits are transmitted continuously and the transmitter does not wait for an ACK after sending a flit. Such an ACK is received after a round-trip delay. When a NACK is received, the transmitter backs up to the flit that is negatively acknowledged and resends it in addition to the N succeeding flits that were transmitted during the round-trip delay. At the receiver, the $N - 1$ received flits following the corrupted one are discarded regardless of whether they were received error free or not. However, this scheme may also add extra burden on the network by ACKing every single flit keeping in view the relative stable condition of on-chip networks against traditional networks.

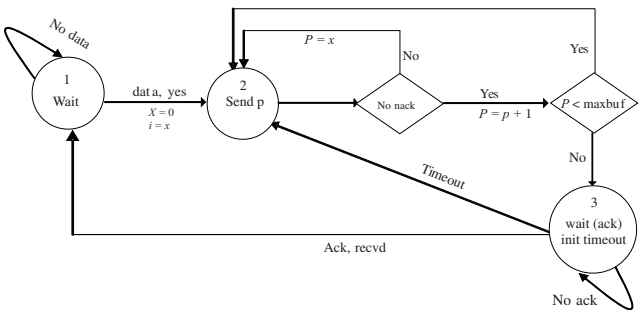
Park et al. (2006) present a Hop By Hop (HBH) error control scheme for mesh-based NoCs. The routers at each stage monitor each incoming flit and if found corrupt discard it and request for its retransmission from the previous level. The proposed HBH retransmission scheme requires a 3-flit-deep retransmission buffer per virtual channel, since a flit should be kept for 3 cycles after it leaves the current node. This 3 cycle delay corresponds to the sum of the link traversal delay (1 cycle), error checking delay at the adjacent receiving node (1 cycle) and the Negative Acknowledgment (NACK) propagation delay (1 cycle). The retransmission buffer is implemented as a barrel-shift register. This way, a flit is stored at the back of the buffer upon transmission on the link and it moves to the front by the time a possible NACK signal arrives from the receiving node. This scheme is relatively better in terms of its counterparts since it only involves NACKs for missing or corrupt packets, thus, reducing the packet overhead in terms of ACKs. However, the authors did not mention what will happen if a NACK is corrupt or even missing. In the absence of an explicit ACK it is very difficult to realise what is the correct sequence of sending packets.

Keeping in view the above mentioned schemes besides the fact that on-chip networks are rather much stable than the off-chip networks, this paper argues to provide a rather simpler mechanism – end-to-end – where the intermediate nodes are relieved of any storing or error checking logic capabilities as this is embedded in the end systems.

5 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 are given in Figures 1 and 2, respectively. 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 the specified timeout, it resends the same set of data. The process continues until it gets the expected ACK from the receiver.

Figure 1 State diagram for sender protocol

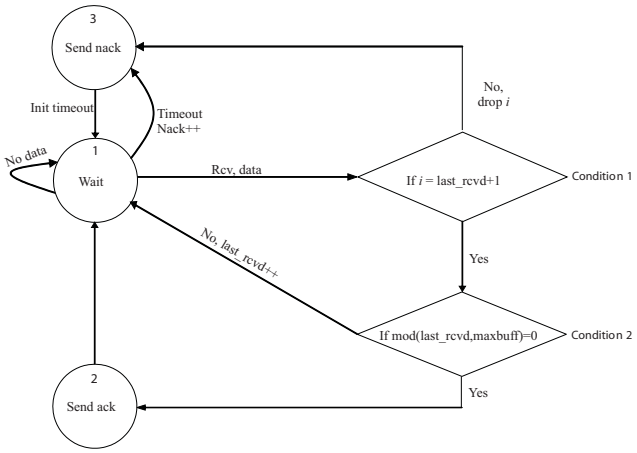


The receiver side receives the packets in sequence and issues an ACK telling the sender of successful reception. In case the receiver does not receive a packet at all, or the CRC code detects data corruption in the payload, a NACK is generated asking the sender to resend the missing/corrupt packet. Since 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 $L_{\text{delay}\times\text{hops}}(\text{bs} + 1)$, where

$L_{\text{delay}\times\text{hops}}$ 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 $2 \times L_{\text{delay}\times\text{hops}}$ after which it resends the nack signal for the missing/corrupt packet.

Figure 2 State diagram for receiver protocol



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, 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.

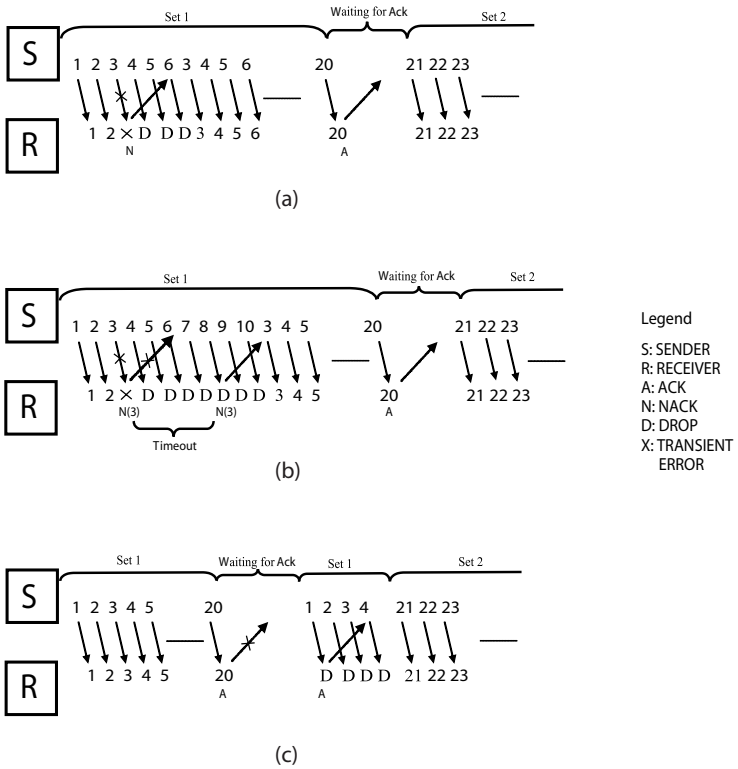
One of the significance of this protocol is that no buffering is required at the receiver side as we employ *go-back-n* retransmission policy where the receiver receives all the incoming packets in order. If a packet is corrupt or missing, the receiver requests for a retransmission and discards all the 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 reordering algorithms besides maintaining buffers at the receiver side. Considering the scarcity of storage capacity on-chip inaddition to simplicity, we have employed *go-back-n* mechanism in our protocol.

5.1 Data transmission example

Consider the example data transmission process shown in Figure 3 which illustrates different cases of packet losses that can occur due to a transient fault on-chip. We assume that 20 packets are sent by the sender before it expects an ACK from the receiver. Three different cases are considered here:

- In Figure 3(a), packet 3 from the first set is lost so the receiver generates a NACK(3) and discards all the subsequent packets until it receives the desired one. The sender keeps on sending the packets until it receives the NACK, after which it resends the requested packet followed by the subsequent packets.
- In Figure 3(b), a NACK is generated by the receiver when it does not get packet 3. This NACK does not make it to the sender which keeps on sending the packets. After a certain time, the receiver resends the NACK and meanwhile keeps on dropping all the packets until it receives the packet 3.

Figure 3 Sender, receiver behaviour



- In Figure 3(c), the receiver successfully receives all the packets and generates an ACK. The ACK is lost and after a certain time, the sender resends the same set of packets again. The receiver detects that its the same set, so it discards the packets and generates an ACK again. Upon reception of the ACK, the sender stops and relinquishes its buffer, gets the next set of packets and sends it to the receiver.

6 Performance evaluation

We have used *Network Simulator ns-2* for implementing our reliability protocols (www.isi.edu/nsnam/ns/). ns-2 is an open source, object-oriented and discrete event driven network simulator written in C++ and OTcl. It is 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 behaviour of a NoC at a higher abstraction level of design, for instance, Ngo and Choi (2005), Sun et al. (2002), and Vahdatpour et al. (2005) have used ns-2 to simulate the behaviour of their NoC designs. ns-2 is already equipped with a wide variety of protocols and various topologies can be created with little effort. Moreover, customised 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.

We have simulated a 4×4 2D mesh NoC using ns-2. The chip is assumed to be of size 22 mm $\times$ 22 mm with link size of 4.5 mm 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 performed simulations the results of which are furnished below.

6.1 Reliable packet delivery protocol

As discussed earlier, the probability of soft errors in NoCs may lie in a range of 10^{-9} and 10^{-20} BER. However, this value is too small to obtain substantial results. We therefore use an error model which arbitrarily introduces errors in the data packets. The error model affects the links connecting the nodes causing bits to scramble. The error rate ranges from 1 to 5% where 1% means one packet in every hundred packets is faulty. We also used different buffer sizes to observe the performance of the protocol.

In order to show that the protocol is retransmitting the missing packets, we took sample data from our simulations for two nodes. To make a fair comparison, we performed similar tests using the unreliable internet protocol UDP (Postel, 1980) with a constant bit rate. Figure 4 shows the representative of the results of these tests where the sequence numbers of transferred packets are plotted against time. The straight line shows the constant behaviour of UDP which sends packets with a constant sending rate without considering any packet loss. The packet loss, caused by the underlying error-model, is represented by the small gaps in the line, for example, packet drops can be seen at times 2.003, 2.009, 2.010, etc. which are never retransmitted. Contrarily, in Figure 5, with a buffer size of 10, gaps as seen in UDP protocol are missing in our NoC protocol due to its reliable nature where lost packet get retransmitted. It can be seen that after every 10 packets the protocol waits for an ACK, after which it moves higher. When packets are lost, the graph retraces the path and resends them, like at time 2.006, packets are resend. Hence we do not see any gap which shows that all the packets are ultimately successfully transmitted.

Figure 4 UDP protocol showing packet drops never retransmitted

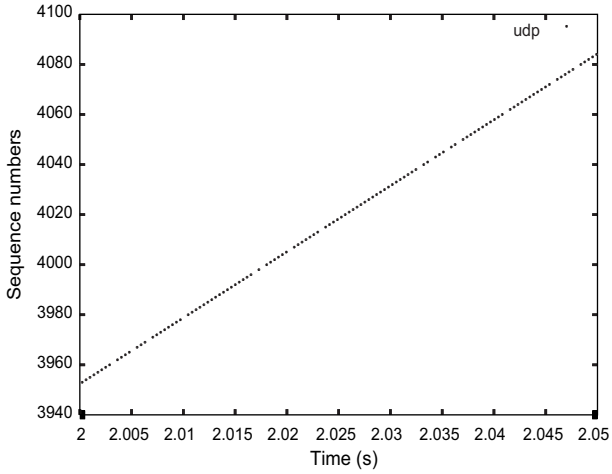
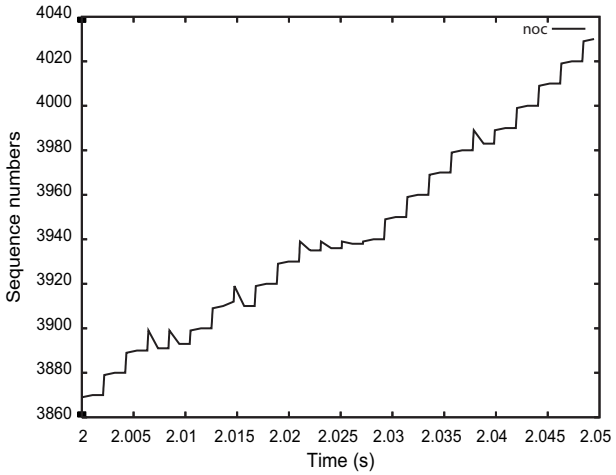


Figure 5 NoC protocol showing retransmission of dropped packets



6.2 Throughput vs packet overhead

Consider Figure 6 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, for example, 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: $O_{\text{bs}} = 1/(bs + 1)$ and $\lim_{\text{bs}\to\infty} O_{\text{bs}} \to 0$. This is underlined by our experimental results shown in Figure 7. The reason behind this behaviour 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 7 shows that using a buffer size of 10 minimises the overhead for most of the examined error rates.

Figure 6 Total throughput achieved with NoC protocol

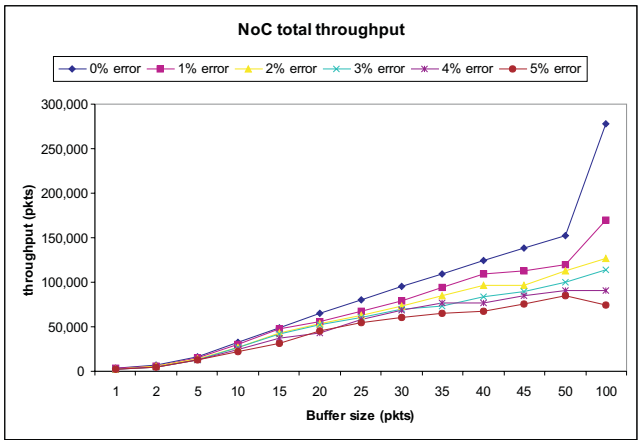
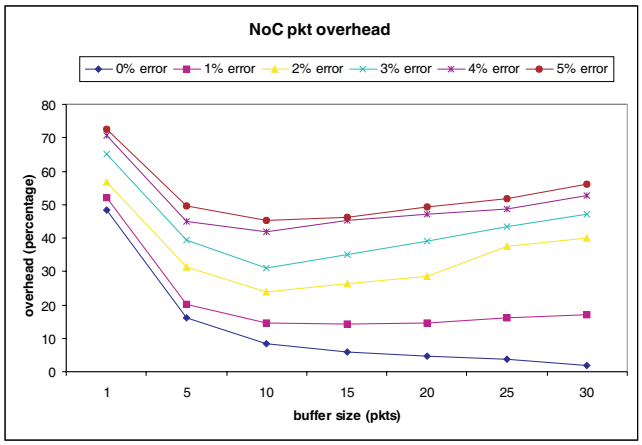


Figure 7 Effect on packet overhead with varying buffer sizes and error rates



6.3 NoC protocol versus TCP

TCP is the most widely used protocol in today's traditional networks. It is quite complex and is therefore not suitable to be implemented in a NoC due to storage and logic constraints. However, since its a reliable protocol which is based on the same idea of retransmissions, we performed comparative experiments to observe the behaviour of our protocol against TCP. The results are shown in Figure 8. It can be seen that in stable condition, that is, when the channel is error free, TCP seems to perform better with greater buffer size. It is because TCP doubles the sending rate once it receives positive ACK for the last packet sent. Unlike our protocol, which after sending the predefined number of packets, waits for a positive ACK from the receiver, before it can send the next set, TCP continuously sends the packets at the maximum rate defined. However, with the introduction of errors in the links, the performance of TCP drastically decreases and keeps on decreasing with higher error rates. It can be seen that our protocol, on the other hand, behaves in relatively stable way maintaining its overall throughput to a considerable amount. This clearly shows that our reliability protocol, in comparison to TCP, provides considerably better performance at higher error rates.

7 Fault tolerant routing

There are two major classes of dynamic routing protocols in the Internet; *Distance Vector* and *Link State* (Huitema, 2000). In *distance vector* routing, each router shares its information only with its direct neighbours. Though simple, distance vector leads to well-known problems like counting-to-infinity, bouncing effect etc.[7] In contrast, *link state* mechanism tells every router on the network about its neighbours 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, the topology is frequently changing as 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.

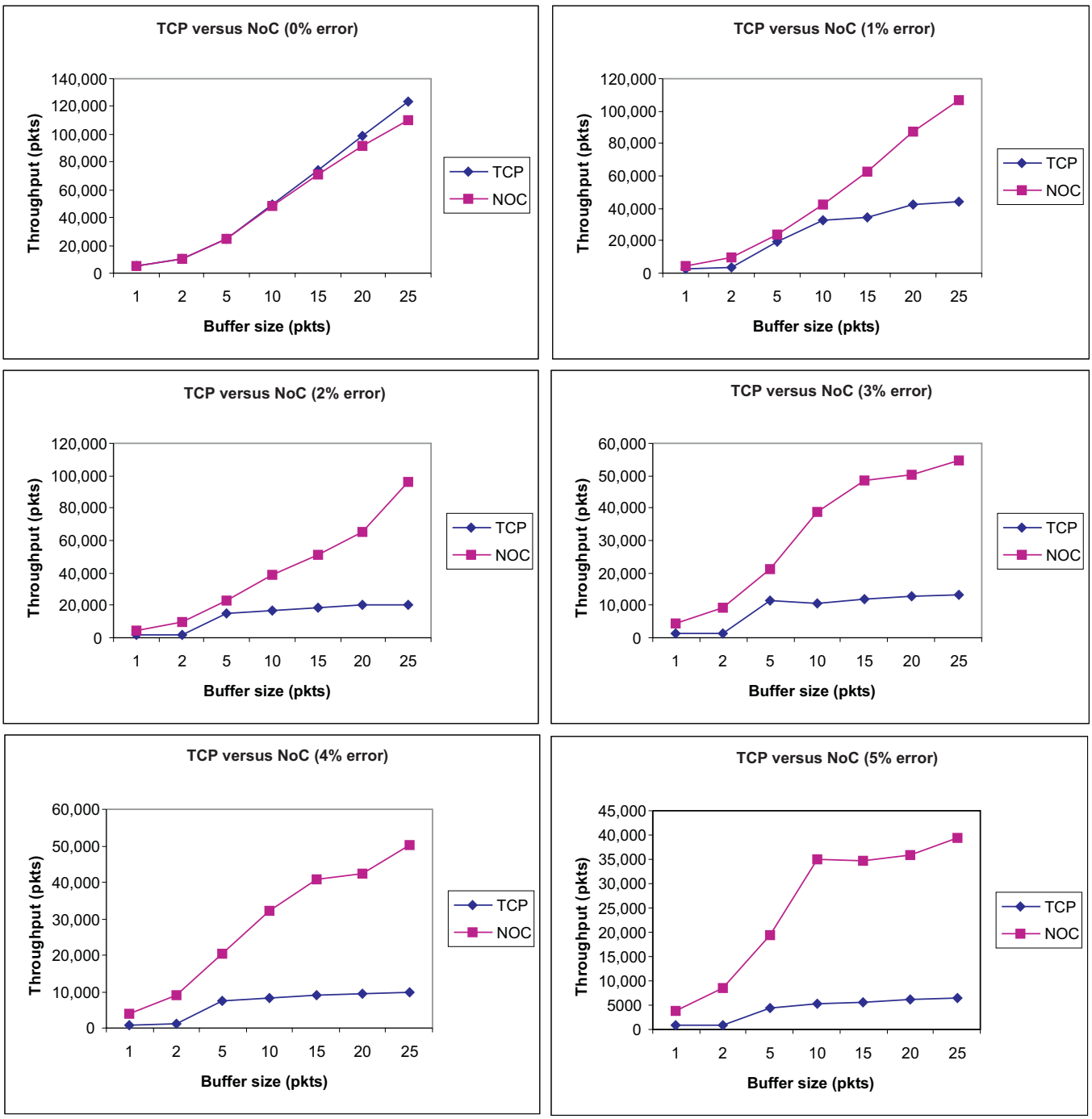
Neither of these mechanisms are suitable in their current form for NoCs because of excessive complexity and huge logic resources. As mentioned earlier that permanent faults are extremely low in NoCs as compared to the internet. The network is relatively stable on-chip as compared to off-chip networks. 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 FT routing scheme where routing tables are initialised in advance and hard-coded in the system. When a module fails it informs its neighbours 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, the shortest path is calculated by a shortest path algorithm. Hence, the overall communication continues with graceful degradation of service.

7.1 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 initialised in advance as the topology, link 'costs'[8] (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.

Figure 8 TCP, NOC performance comparison



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 tables and exchange this information with their neighbours.

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

7.2 Performance evaluation of DR-NoC

Consider Figure 9 which shows basic implementation of our dynamic routing protocol. At time 5 ms, 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 10 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.

7.3 Example

Consider Figure 11(a) which represents a 2 × 3 mesh NoC. The routing table of these routers after convergence is represented as shown in Figure 12. Each node is labelled as A, B, C, D, E, F whereas links are labeled from 1 through 7. Since it is a regular mesh topology, the distance between adjacent nodes is assumed to be equal (e.g. 1). Each row in the table represents a two-way link with the link number. With this look up table, each node is capable of calculating the shortest path to the destination using *Dijkstra's shortest path algorithm*. If there are alternate paths with the same number of hops to the destination, then the algorithm follows the *XY* dimensions, that is, the packets first follows the *X* direction and when it reaches the column where the destination node lies, it forwards the packet in *Y* direction. For example, if node transports packets to node F, it follows the path A – B – C – F. This way, the routes remain coherent and loops do not occur.

Figure 9 Throughput w/without dynamic routing protocol

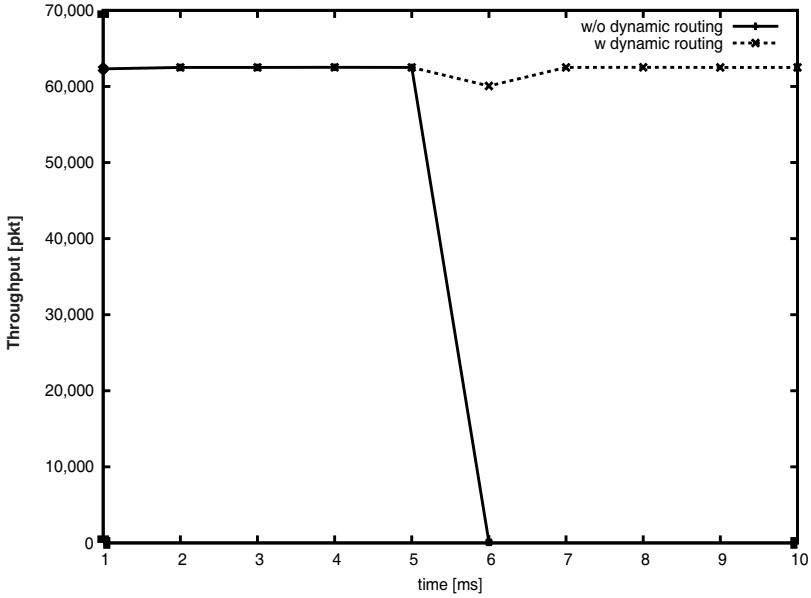


Figure 10 Throughput against many links failing

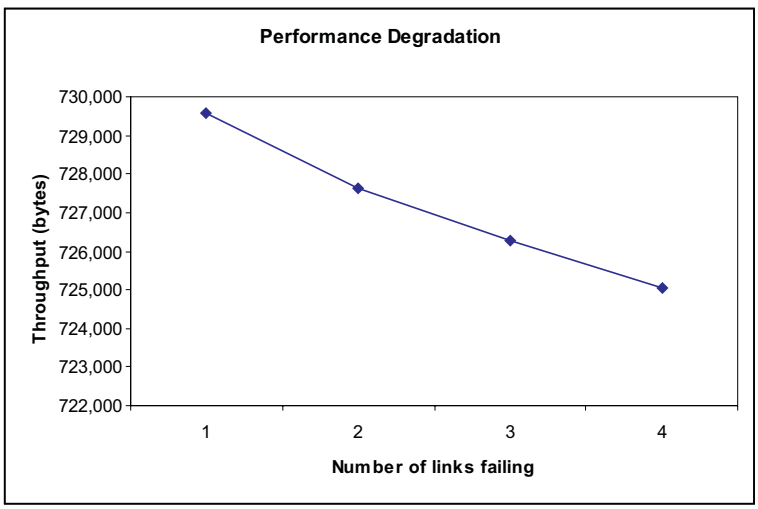


Figure 11 (a) Shows a fully working topology, (b) shows a link failed and (c) shows a router failed

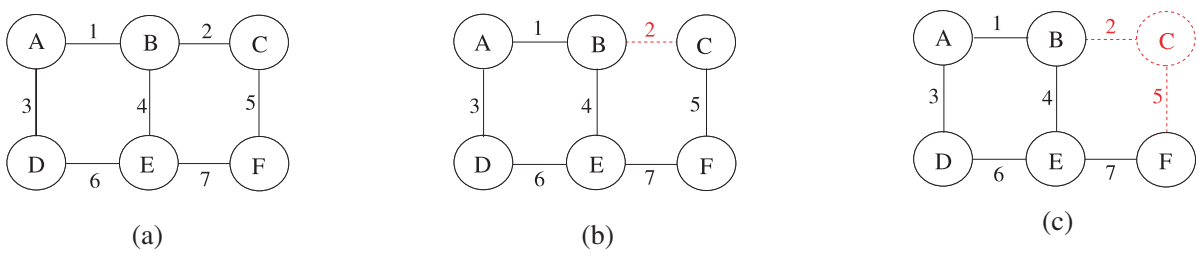


Figure 12 Lookup table showing 1-1 links

From / To	Link
A <-----> B	1
A <-----> D	3
B <-----> C	2
B <-----> E	4
C <-----> F	5
D <-----> E	6
E <-----> F	7

At any instance t it is assumed that the link between nodes B and C suffered a permanent damage as shown in Figure 11(b). This information is immediately shared among all the routers in the network which remove the corresponding link from their routing tables as shown in Figure 13. After the change, new routes have to be calculated using the shortest path algorithm. Similarly, in case a routing element F fails, the links associated with it also become inactive as shown in Figure 14. The routing table shown in Figure 11(c) consequently marks links 5 and 7 as unreachable.

Figure 13 Link 2 failed and marked unreachable

From / To	Link
A <-----> B	1
A <-----> D	3
B <-----> C	2
B <-----> E	4
C <-----> F	5
D <-----> E	6
E <-----> F	7

Figure 14 Node C failed and connected links marked unreachable

From / To	Link
A <-----> B	1
A <-----> D	3
B <-----> C	2
B <-----> E	4
C <-----> F	5
D <-----> E	6
E <-----> F	7

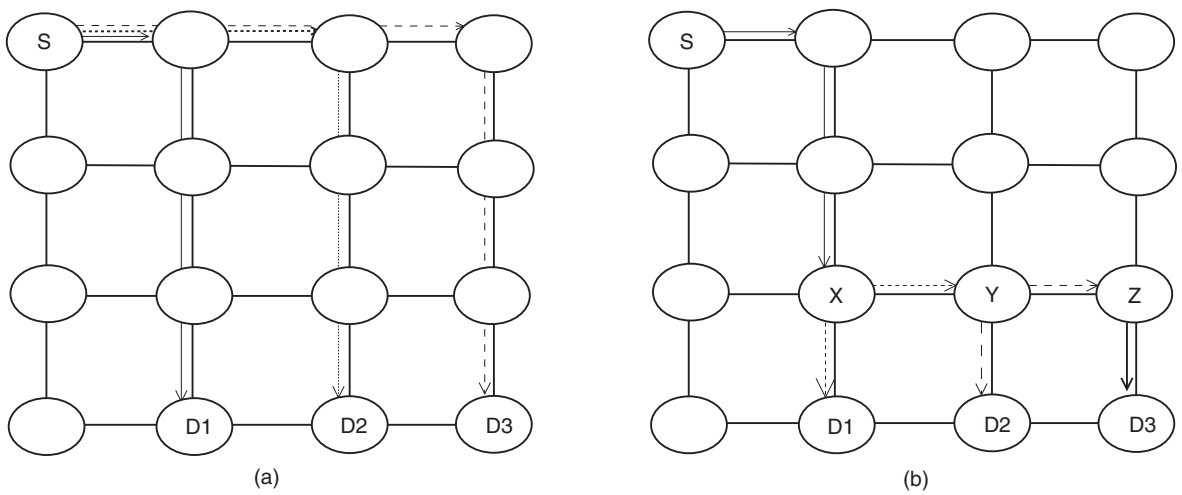
8 Conclusion and future work

This paper presents a comprehensive solution against transient and permanent faults affecting a NoC. In order to handle transient faults on-chip, we propose an end-to-end reliability protocol in which a set of packets are positively acknowledged by a single ACK instead of an ACK for every packet. This greatly reduces the packet latency and redundant traffic in the network. We performed simulations to show the performance of our protocol and showed that it performs very well in comparison to TCP especially at high error rates.

Furthermore, we also presented a robust, efficient and FT mechanism for dealing with permanent faults which may affect one or more modules on a chip. We have provided a simple mechanism where in case of module failures, the routers inform the adjacent routers about the failure and update their tables. New routes can then be calculated and communication remains intact in a deterministic fashion.

The current form of our reliability protocol supports unicasting.[9] Next our aim is to incorporate multicasting[10] into our mechanism. Although, the stochastic communication model for NoCs presented by Dumitras and Marculescu (2003) resolves the issue of multicasting, it might also

Figure 15 Multicasting: (a) sender sends same packets individually to three receivers through different paths, (b) sender sends packet to intermediate nodes which further replicate it and send to other destinations



add considerable amount of redundant traffic to the network. The problem of multicasting is quite complex as many issues are involved in it. Firstly, if a source is communicating with many destinations, then it needs to keep track of packet corruption or losses for each individual receiver. Then comes the question of communication paths; does the same packet be sent via different paths or via same path especially when the destinations are situated at the same level. Another possibility is to send a single copy of packet until it reaches one router before the first destination. This router then replicates the packet and sends to other attached destinations. Both of the described situations are depicted in Figure 15. In Figure 15(a), a single sender *S* takes three different paths to send same set of packets to destinations D1, D2, D3. Alternatively, in Figure 15(b), the sender sends packets to intermediate node X, which besides delivering it to D1, replicates the packets and sends to node Y, which does the same. This mechanism on the one hand, reduces redundant traffic in the network however at the cost of complexity at the router side. We intend to perform experiments to realise an optimal multicast solution for mesh based NoCs.

References

Ali, M., Welzl, M., Zwicknagl, M. and Hellebrand, S. (2005) 'Considerations for fault-tolerant Network on chips', *Proceedings, 17th International Conference on Microelectronics (ICM)*, Islamabad, 13–15, December, pp.177–181.

Ali, M., Welzl, M. and Zwicknagl, M. (2006) 'Networks on chips: scalable interconnects for future systems on chips', *Proceedings of the 3rd IEEE International Conference on Circuits and Systems for Communications (ICCSC'06),* 6–7 July, Bucharest, Romania.

Bailey, N. (1975) 'The mathematical theory of infectious diseases', London: 2nd edition, *Charles Griffin and Company*.

Bertozzi, D. and Benini, L. (2004) 'Xpipes: a network-on-chip architecture for gigascale system-on-chip', *IEEE Circuits and Systems Magazine*, Vol. 4, No. 2, pp.18–21.

Benini, L. and De Michelli, G. (2002) 'Networks on chips: a new SoC paradigm', *IEEE Computer*, Vol. 35, No. 1, pp.70–78.

Bushnell, M.L. and Agarwal, V.D. (2000) 'Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits', *Kluwer Academic Publishers*, USA.

Dally, W.J. and Towles, B. (2001) 'Route packets, not wires: on-chip interconnection networks', *Proceedings, Design Automation Conference (DAC)*, pp.684–689, Las Vegas, USA.

Dally, W and Towles, B. (2004) *Principles and Practices of Interconnection Networks*, USA: Morgan Kaufmann Publishers.

Demers, A., et al. (1987) 'Epidemic algorithms for replicated database maintenance', *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pp.1–12, Vancouver, British Columbia, Canada.

Dumitras, T. and Marculescu, R. (2003) 'On-chip stochastic communication', *Proceedings, Design, Automation and Test in Europe (DATE)*, pp.790–795, 3–7 March, Munich, Germany.

Huitema, C. (2000) *Routing in the Internet*, *2nd edition, 2000*, New Jersey: Prentice Hall.

Johnson, B.W. (1993) 'Fault tolerance', in *The Electrical Engineering Handbook*, R.C. Dorf (Ed). CRC Press, p.2020.

Kumar, S., et al. (2002) 'A network on chip architecture and design methodology', *Proceedings, IEEE Computer Society Annual Symposium on VLSI*, pp.117–124, 25–26, April.

Ngo, V. and Choi, H. (2005) 'On chip network: topology design and evaluation using NS2', *Proceedings, 7th International Conference on Advanced Communication Technology (ICACT 2005)*, pp.1292–1295, Phoenix Park, Korea, 21–23, February.

Park, D., et al. (2006) 'Exploring fault tolerant network-on-chip architectures', *Proceedings, The International Conference on Dependable Systems and Networks (DSN-2006)*, pp.93–102, Philadelphia, PA, USA.

Postel, J. (1980) 'User datagram protocol', *RFC 768*.

Shukla, S.K. and Bahar, R.I. (2004) Nano, Quantum and Molecular Computing, Implications to High Level Design and Validation, Boston: Kluwer Academic Publishers.

Sigüenza-Tortosa, D. and Nurmi, J. (2002) 'Proteo: a new approach to network-on-chip', *Proceedings, IASTED-Communication Systems and Networks (CSN 2002)*, Malaga, Spain.

Sun, Y.R., Kumar, S. and Jain, A. (2002) 'Simulation and evaluation of a network on chip architecture using ns-2', *Proceedings 20th NORCHIP Conference*, Copenhagen, November.

Vahdatpour, A., Tavakoli, A. and Falaki, M.H. (2005) 'Hierarchical graph: a new cost effective architecture for network on chip', *Proceedings, International Conference on Embedded And Ubiquitous Computing, Nagasaki, Japan, Lecture Notes in Computer Science* Vols. 3824/2005, ISBN 3-540-30807-5, Berlin/Heidelberg: Springer, pp.311–320.

Von Neumann, J. (1955) 'Probabilistic logic and synthesis of reliable organisms from unreliable components, automata studies', in C.E. Shannon and J. McCarthy (Eds). *Princeton University Press (1956)*, pp.43–98.

Wu, M.S. and Lee, C.L. (2005) 'Using a periodic square wave test signal to detect cross talk faults', *IEEE Design and Test of Computers*, Vol. 22, No. 2, pp.160–169.

Notes

[1]Error correction codes are much more complex in terms of implementation.

[2]Each packet is subdivided into equal sized smaller chunks called as flits.

[3]Where a logical node is stuck at either logical zero or one.

[4]A module may be a link or a router.

[5]Mean time between failures.

[6]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.

[7]A detailed reference to the problems in distance vector routing is given in Huitema (2000).

[8]Distance value between two nodes.

[9]One source sending packets to one receiver.

[10]Single source sending packets to many receivers simultaneously.