International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:7, 2008

Comparative analysis of transient-fault tolerant schemes for Network on Chips

Muhammad Ali (muhammad.ali73@gmail.com), Awais Adnan (owaisadnan@gmail.com)
Institute of Management Sciences, Peshawar, Pakistan

Abstract**—**Network on a chip (NoC) has been proposed as a viable solution to counter the inefficiency of buses in the current VLSI on-chip interconnects. However, as the silicon chip accommodates more transistors, the probability of transient faults is increasing, making fault tolerance a key concern in scaling chips. In packet based communication on a chip, transient failures can corrupt the data packet and hence, undermine the accuracy of data communication. In this paper, we present a comparative analysis of transient fault tolerant techniques including end-to-end, node-by-node, and stochastic communication based on flooding principle.

Keywords**—**NoC, fault-tolerance, transient faults

I. Introduction

Transient faults are a common source of errors in today's VLSI chips. It has been predicted that the soft error rate will increase once the gate length reaches 100nm or below [1]. Moreover, with new technologies being introduced on-chip in order to achieve low power and low cost, the problem of soft error becomes significant. Traditionally, the combinational circuits were relatively less prone to transient errors than the memory elements, however, decreasing feature size and increasing clock frequencies are exacerbating their prominence [2]. Some recent research indicates that by 2011 the Soft Error Rate (SER) in logic circuits per chip will become almost comparable to SER per chip of the memory elements [3]. This observation will consequently have a significant impact on the reliability of on-chip routers and links. Hence, it is imperative to provide some robust protective solutions against such upsets.

Network on Chips (NoC) have been proposed by researchers to cope with the inefficiency of on-chip buses in scaling chips [4], [5]. A NoC can be defined as 'a network of computational, storage and I/O resources that are interconnected by a network of switches, where resources communicate with each other using addressed data packets routed to their destinations by the switch fabric [6]". In contrast to long and shared buses, the communicating modules on-chip are connected via a network of switches/routers which are connected with each other through point-to-point links. In this organization, these point-to-point links represent the buses whereas switches/routers represent the bus-bridges. The NoC is a communication centric design paradigm where resources communicate via *packets*. A *packet* is the unit of data that is routed between an origin and a destination in the interconnection network. The on-chip communication is governed by a protocol stack irrespective of the computational infrastructure. The primary objective of communication-based designs is to separate communication from computation besides providing a re-usable interconnect architecture for ultra large scale System on Chips.

With shrinking die size, reliability of future chips is becoming an issue of grave concern. It has been observed that NoCs will face the same problems due to transient faults as faced by traditional VLSI chips. Although chips are traditionally designed with error detection and/or correction codes, such complex codes incur high energy and area overheads [7]. Since storage and logic resources are limited on-chip, its important to realize solutions which are low cost in terms of memory and energy without compromising on reliability and performance. Also bringing packet-based communication on-chip introduces new issues to deal with. A transient fault can scramble one or more bits in a packet either in the payload or in the header. If the payload data is corrupt, packet is invalid. Similarly, a bit flip in the header field can cause a packet to be misrouted. In either case, a retransmission of the corrupt/missing packet is required.

Keeping in view the increasing probability of transient faults and its growing impact on the on-chip interconnects, we proposed and implemented an efficient end-to-end reliable protocol for mesh based NoC [8]. The novelty of the protocol is that it uses a single ACK to acknowledge a predefined set of packets instead of ACKing each packet. There is no buffering in the intermediate routers as the packets are only buffered at the sender side. In this paper, we performed experiments to compare the performance of our end-to-end reliable protocol with some of its counterparts. Our experimental results clearly show that our protocol outperforms the link level and flooding protocols in terms of achieving high throughput, low packet drop-rate and little packet overhead.

II. Related work

Traditionally speaking, error detecting and correcting codes have been the most common means for handling on-chip errors. Researchers have explored various error detection and correction techniques that would apply to NoCs. State of the art coding schemes have been studied and their significance has already been analyzed with respect to NoCs [9], [10], [11], [12]. Error control codes can detect single and multi-bit errors. Moreover, most of the error correcting schemes can correct single-bit errors with an overhead of extra hardware. However, the increase in the complexity and size of VLSI chips increases the probability of multi-bit errors. Hence, in packet-based on-chip networks, multi-bit errors can corrupt the packets which need to be retransmitted. The retransmission

scheme can be either embedded in the end systems or in the intermediate level routers. In [13], Bertozzi et al. present a link level *flit-based*[1] 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. The main problem with this technique is that it generates a large amount of redundant packets in terms of ACKs/NACKs. Similarly, the authors of [14] present a link level retransmission technique with an exception that it only uses NACKs to inform the sending device for the non-reception of the packets. The key problem in link based schemes is that each intermediate router stores and checks the incoming packet for inconsistencies. If an error is detected, the receiving router drops the packet and requests for a retransmission from the previous router. Thus, this scheme bears storage and processing overhead at each router level.

An alternate approach towards fault tolerance in NoCs is stochastic communication [15] which is based upon randomized rumor spreading [16]. 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. The method is quite simple but it has a very high packet overhead as each tile replicates and forwards the packet to the adjacent routers. Hence, with increasing number of senders, the packet overhead will increase exponentially.

We believe that on-chip errors are relatively scarce than off-chip networks. Moreover, on-chip storage and processing resources are limited, therefore, an end-to-end protocol is more effective in NoCs. We have already implemented a prototype of our protocol and performed various experiments to evaluate its performance [8]. The basic idea of the protocol is that the sender buffers and sends a predefined number of packets continuously to the receiver. The packets are sent in a sequence. After sending all the packets, the sender waits for a *positive acknowledgement (ACK)* from the receiver. The ack packet carries the sequence number of the next expected packet, i.e. the start of the next set of packets from the sender. The receiver receives the packets *in-order* and sends an ACK. The main aspect of our protocol is that is uses a single ACK to acknowledge a set of packets instead of acknowledging every single packet. Unlike traditional networks where communication wires are tremendously long and more unreliable, on-chip wires are physically stitched close to each other offering tight synchronization and stability. Thus, using a single ACK for a set of packets will enable the system to perform more efficiently. The primary purpose of this paper is to compare and evaluate the performance of above mentioned protocols.

III. NoC model

We visualize the NoC as a 2D mesh topology with packet level communication: The inter-switch wire is wide enough to transfer all bits of a packet simultaneously. As argued in [17], if we have 3 metal layers on the silicon die, then around 300 inter-switch wires can be implemented. This gives us 128-bit wide data bus in each direction. The packet consists of a payload and header. The header contains identification information like source and destination, routing information, number of nodes etc. The payload carries the actual data. We categorize packets as *data packets* and *ACK packets*. Data packets are long packets carrying the useful data. ACK packets are control packets including positive and negative ACKs. Routing is static — XY dimension based. Static routing is simple and well suitable for mesh based NoCs. We use a bufferless router which forwards the incoming packet immediately as it arrives instead of storing it. Hence, there are no queues used in the routers. In case of conflicts, packets are dropped based on priorities — the ACK/NACK has the higher priority than the data packets.

A. Simulation setup

We implemented the three fault tolerant protocols — end-to-end (E2EEC-G[2], link level (LL), and flooding (FL) — in network simulator ns-2 [18].

The ns-2 is an object-oriented, discrete event driven network simulator developed at UC Berkely and written in C++ and OTcl. C++ is used for detailed implementations of protocols like TCP or any customized ones. TCL scripting, on the other hand, is the front-end interpreter for ns-2 used for constructing commands and configuration interfaces.Moreover, a *Network AniMator* (NAM) is also provided with the ns-2 in order to visualize and interact with the system at run-time. Finally, graphs can be created from the produced results to evaluate and analyze the performance of the system. Many NoC designers have used ns-2 to simulate and evaluate the performance of their design at a higher abstraction level [19], [20], [21], [22].

We simulated a 10×10 prototype of a 2D mesh based NoC using the ns-2. The sender-receiver pairs were selected randomly. Since different arrangement of the pairs may affect the overall throughput, therefore, 50 simulations were performed for each pair. The pair selection process is explained with an example:

- In order to select 4 pairs of senders/receivers, for instance, first we generate 8 unique random numbers in the range of 1 to 100 (since we have 100 nodes in the mesh). These random numbers are divided into two equal sized arrays. The first array represent the set of senders and the second one the receivers. Then the first value of the array 1 (that is sender 1) is connected to the last value of array 2 (which is receiver 1). Similarly, the second value of array 1 is connected to the second last value of array 2, and so on. This organization is shown in figure 1. In this example, node 89 is the sender and node 91 becomes the receiver. Similarly, nodes 5, 75, and 23 are sender for nodes 49, 21, and 66 respectively. The same procedure is repeated 50 times for 4 senders and receivers. Every time different set of nodes are selected with varying number of hops between them. The purpose of these extensive simulations is to create a realistic NoC

[1]a packet is further divided into small chunks called as flits

[2]since the protocol employs go-back-n scheme so we call it E2EEC-G)

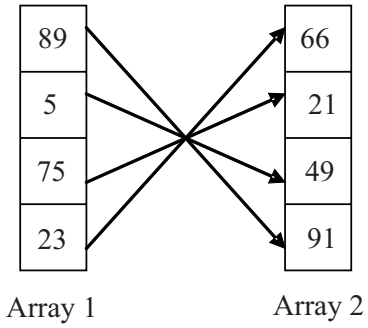

Fig. 1. Selection of sources and their corresponding destinations

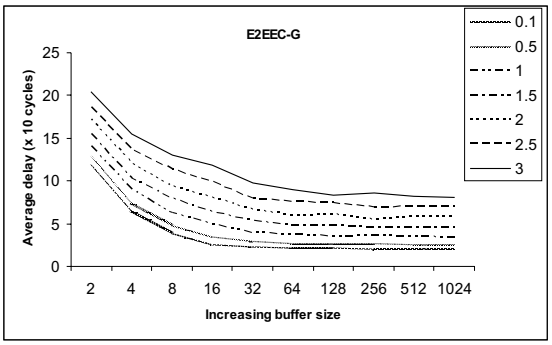

Fig. 2. Packet latency with different buffer sizes and varying error rates

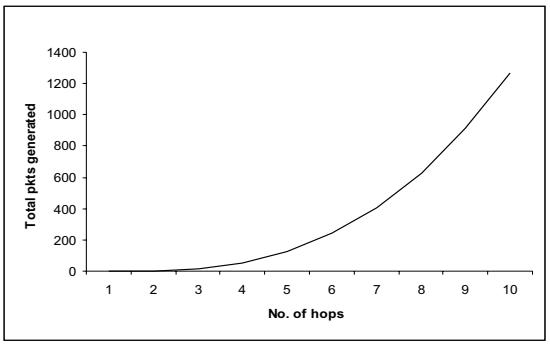

Fig. 3. Total packets generated after 10 hops

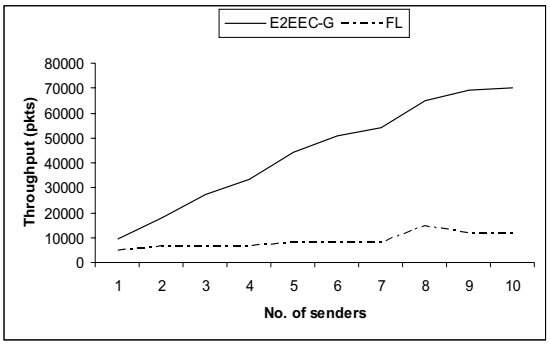

Fig. 4. Comparison of FL and E2EEC-G average throughput

environment where senders and receivers are arbitrarily distributed over the whole network.

We performed simulations to find out an optimal buffer size. Our simulations show that for a 10×10 NoC, a buffer size of 128 is most feasible with different error rates as shown in figure 2. The simulations were performed in zero-contention environment and the results are based on individual sender-receiver pairs. Since we use shortest path routing, the average latency is calculated to be 10 cycles where each cycle represents a link delay. It should be noted that we do not consider the router delay.

IV. Comparative analysis of E2EEC-G protocol with flooding protocol (FL)

We implemented a flooding variant of stochastic communication protocol based on flooding. The sender sends packets in all directions (except to one from where it receives the packet). Each node is equipped with CRC code to check the validity of packet. If the packet is corrupt, it is dropped. If the packet is valid, and it is destined for the current node, it is accepted, otherwise, it is replicated to all its output ports. In this case, the receiver may receive copies of the same packet from different neighbor nodes, in which case it drops all the packets which come later. Since a packet may reach earlier than the broadcast is complete, a time to live (TTL) value is associated with each packet. The TTL value of each packet is incremented at each hop and after reaching the maximum value, the packet is discarded. The flooding protocol works on the assumption that the packet will ultimately reach its destination, therefore, no acknowledgements are required.

The major problem with the flooding protocol is that it generates a huge amount of redundant packets in the whole network. The situation gets worst when there are multiple number of senders and receivers. Excessive packets not only affect the overall performance but also affect the power consumption of the system which is rather scarce on-chip. In order to show the packet overhead, we took sample data from the simulations for 1 sender and receiver, which is depicted in figure 3. The graph clearly shows that after the first 10 hops, more than 1200 packets have already been generated in the network. This trend would generally mean that the packet overhead will drastically increase with increasing number of senders.

Figure 4 shows the comparison of the average throughput of FL and E2EEC-G. As we can see that the overall throughput remains quite low in FL even when the number of senders increase. On the other hand, the E2EEC-G throughput gradually increases with increasing senders with relative stability.

Another important aspect is the packet drop rate. Figure 5 gives the comparison of packet drops in FL and E2EEC-G protocols. In E2EEC-G, the drop rate is pretty low as compared to FL which lies in the range of 50% to 90%. Analyzing the graph shows that until 3 senders, there is hardly any drop of packets in E2EEC-G protocol, however, when

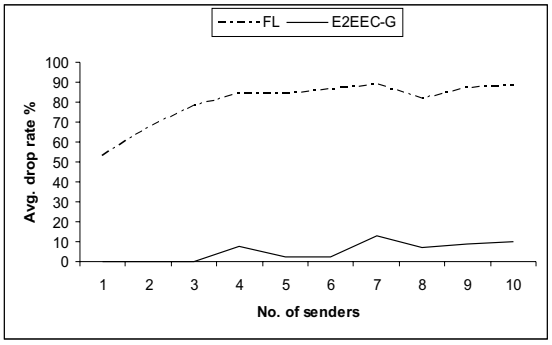

Fig. 5. Comparison of FL and E2EEC-G drop rate

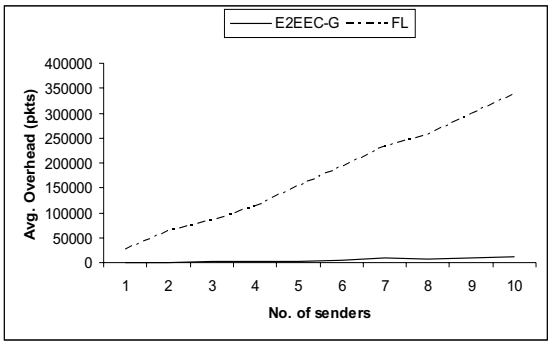

Fig. 6. Comparison of FL and E2EEC-G packet overhead

sender increase, packet drops are inevitable, mainly due to congestion in the routers. Routing strategy is static, and the routers are bufferless, so when two packets are contesting for the same output port, one has to be dropped. Nevertheless, as it can be seen in the graph that even in the worst of the situation, the E2EEC-G drop rate remains under 15% which is much lower than FL.

The overhead (O) is defined as the goodput, that is, packets with an increasing sequence number, divided by the number of packets transferred in total. ACKs and NACKs, and retransmitted packets are counted as the overhead. 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 acknowledgement packet (an **ACK**) has to be transmitted. Thus, for an error rate of 0, the overhead could be expressed as:

$O_{bs} = \frac{1}{(bs+1)}$ and $\lim_{bs\to\infty} O_{bs} \to 0$.

Figure 6 shows a comparison of the packet overhead in terms of increasing number of senders.

Again we can see that the packet overhead is drastically increasing with increasing number of sender devices in FL protocol. In E2EEC-G, on the contrary, it remains very low. It is important to note that these results reflect an error free NoC. Further, we introduce transient faults in the links in order to observe the performance of the E2EEC-G protocol. Although the on-chip error probability is much less than off-chip networks, lying in the range of 10^{-9} and 10^{-20} BER, it is feasible to introduce a much higher error rate in order to observe some significant results. Based on this fact, we defined an error model which introduces 1%, 5%, 10%, 15%, and 20% error rate gradually in the interconnects. The important observation is the comparison of error induced E2EEC-G protocol with the FL without any error as shown in figure 7.

It is clearly noticeable that even with increasing error rate, the average throughput of E2EEC-G remains better than the FL protocol. Even at the highest error rate of 20%, E2EEC-G fairly performs better with large number of senders.

V. Comparative analysis of E2EEC-G protocol with Link Level (LL) retransmission protocol

We implemented a packet based link level error control protocol (LL) where each router sends a packet to the next router and waits for the ACK. When it receives the ACK, then it sends the next packet. Each router follows the same pattern. In order to compare the performance of LL with our E2EEC-G protocol, we performed simulations using the same configuration as described in section **??**. Figure **??** plots comparative throughput of LL and E2EEC-G against increasing number of senders. The network is stable as there are no errors in the links. Since each node in the LL protocol processes the packet, the overall throughput is much lower than the E2EEC-G which transfers packets uninterruptedly until the buffer is emptied. Due to node delay, the router remains blocked until the packet is released and ACK is generated. In this case, each packet is delayed by 3 times as compared to E2EEC-G protocol.

Figure 8 shows the packet overhead comparison of E2EEC-G with LL. This is an interesting graph as we can see the overhead of LL at some points even reaches above 700% primarily because of ACKs generated at every router. As compared to LL, the E2EEC-G is showing minimal overhead as there is only one ACK after transfer of the entire buffer, that is, 100 packets. Similarly, figure 9 shows average throughput variation with increasing error rates. This shows that at extreme error rates of 40% or higher, the throughput of both protocols will become equal. The trend shows that above 60% error rate, LL protocol may perform better especially in a large interconnection network. However, we believe that such error levels are unrealistic and until that happens, our E2EEC-G protocol still prove to be a viable solution. Furthermore, besides the fact that the LL protocol behaves rather in a stable fashion, it is important to realize that, irrespective of the number of senders, it bears a huge packet overhead. Since every packet is ACKed at every intermediate stage router, so in a large network, LL would generate a huge packet overhead.

VI. Conclusion and Future Work

This paper gives a detailed account of comparative analysis of various fault tolerant schemes to deal with transient faults in Network on Chips. We first gave a brief outline of our end-to-end based reliability protocol and discussed its main features. Further, we compared our protocol with two of its

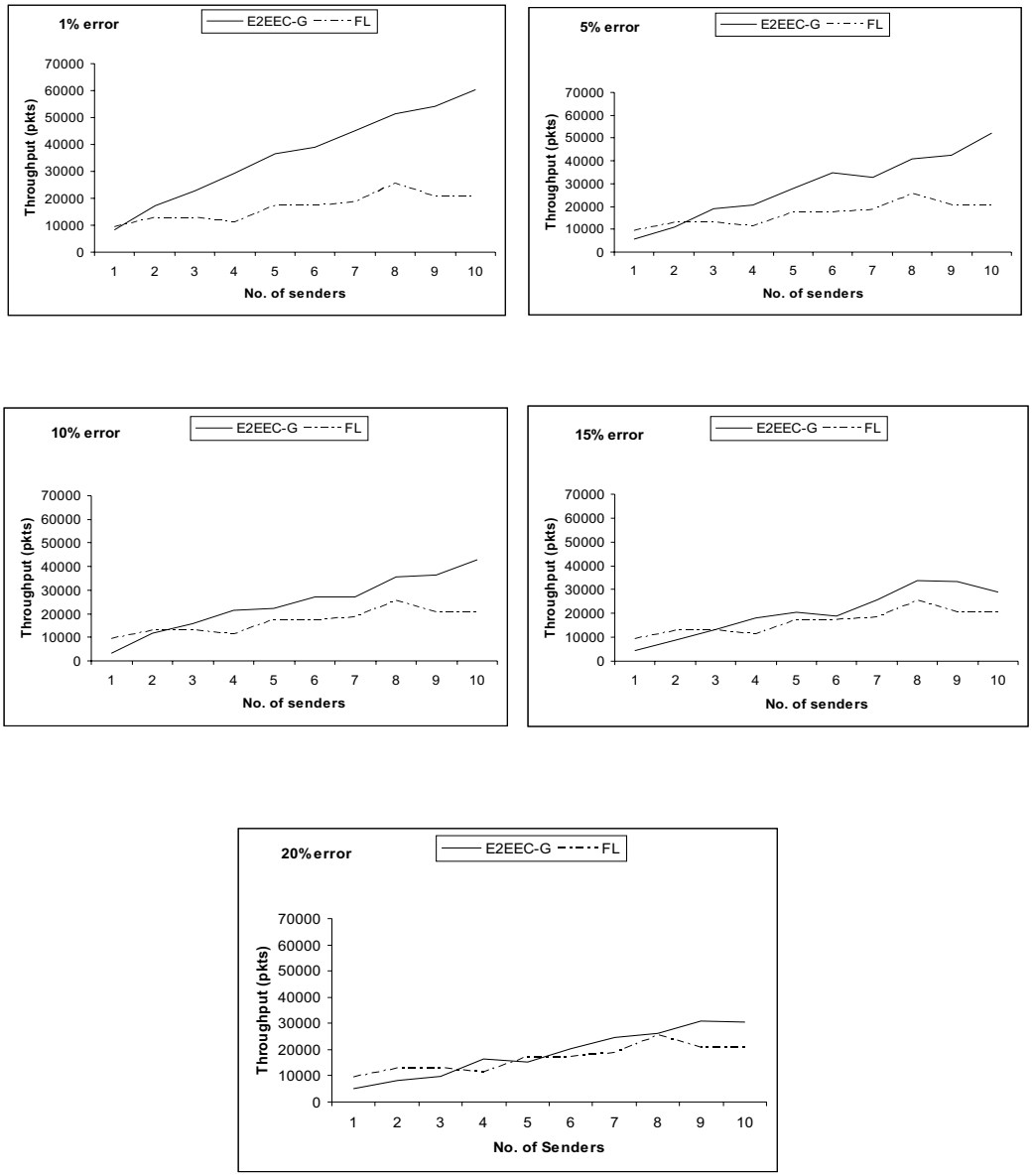

Fig. 7. Comparison of error-free FL protocol with (A) E2EEC-G with 1% error rate, (B) E2EEC-G with 5%, (C) E2EEC-G with 10%, (D) E2EEC-G with 15%, and (E) E2EEC-G with 20%

counterparts: flooding based protocol and link level error control protocol for NoCs. We performed extensive simulations comparing the performance of all the protocols. Our simulation results show that the overall performance of our E2EEC-G protocol is much better than its two counterparts. Also our protocol bears minimal overhead besides requiring limited storage and processing capabilities.

In future, we plan to extend our protocol to support multicasting[3] as it currently supports unicasting[4].

[3]one source sends packets to many receivers
[4]one source sending packets to one receiver

REFERENCES

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

[2] Dongkook Park, Chrysostomos Nicopoulos, Jongman Kim, N. Vijaykrishnan, Chita R. Das, "Exploring Fault-Tolerant Network-on-Chip Architectures", *Proceedings of the 2006 International Conference on Dependable Systems and Networks (DSN06)*.

[3] P. Shivakumar, M. Kistler, S. W. Keckler, D. Burger, and L. Alvisi, "Modeling the effect of technology trends on the soft error rate of combinational logic", *In Proceedings of the Dependable Systems and Networks (DSN)*, pp: 389-398, 2002.

[4] L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm", *Magazine, IEEE Computer*, Jan 2002 Vol.35, No.1, pp.70-78

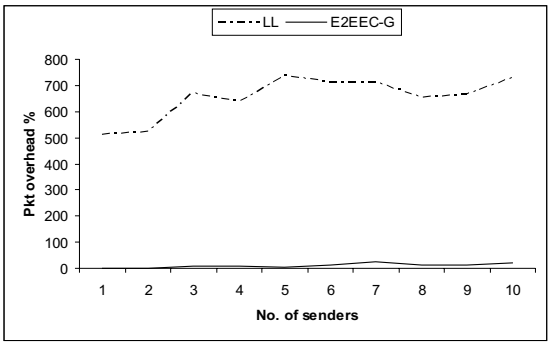

Fig. 8. Avg. Packet overhead comparison of E2EEC-G and LL

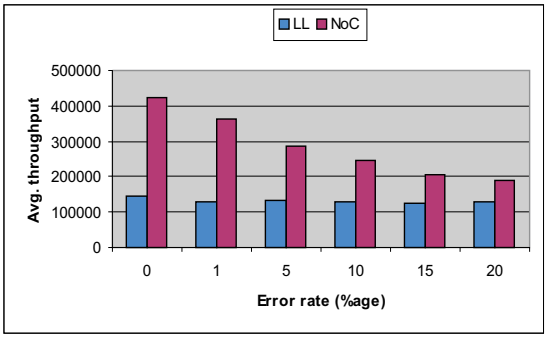

Fig. 9. Overall Avg. throughput comparison of NoC and LL protocols with varying error rates

[5] William J. Dally, Brian Towles, "Route packets, not wires: on-chip interconnection networks", *Proceeding, DAC 2001*, pp.684-689
[6] Érika Cota, Luigi Carro, Flávio Wagner, Marcelo Lubaszewski, "Power Aware NoC reuse on the testing of core based systems", International Test Conference ITC 2003, September 30 - Oct 02, 2003, Charlotte NC USA.
[7] B.W. Johnson, Fault Tolerance, "The Electrical Engineering Handbook", R.C. Dorf, ed., *CRC Press*, 1993.
[8] Muhammad Ali, Michael Welzl, Sven Hessler, Sybille Hellebrand: "A Fault Tolerant Mechanism for handling Permanent and Transient Failures in a Network on Chip", *In Proceedings of the 4th International Conference on Information Technology : New Generations (IEEE ITNG 2007)*, Las Vegas, USA, 2-4 April 2007.
[9] Srinivasa R. Sridhara, and Naresh R. Shanbhag, "Coding for System-on-Chip Networks: A Unified Framework", *IEEE Transactions on very large scale integration (VLSI) Systems*, VOL. 13, NO. 6, JUNE 2005.
[10] D. Bertozzi, L. Benini, Giovanni de Micheli, "Low Power Error Resilient Encoding for On-Chip Data Buses", *Proceedings of the conference on Design, automation and test in Europe (DATE 2002)*, 2002, pp: 102, ISBN:1530-1591.
[11] P. Vellanki, N. Banerjee, and K.S. Chatha, "Quality-of- Service and Error Control Techniques for Network-on-Chip Architectures", *Proceeding of 14th GLSVLSI*, Boston, MA, ACM Press, 2004, pp: 45-50.
[12] H. Zimmer and A. Jantsch, "A Fault Model Notation and Error-Control Scheme for Switch-to-Switch Buses in a Network-on-Chip", *Proceedings of 1st International Conference on Hardware/Software Codesign and System Synthesis (CODES 03)*, IEEE Press, 2003, pp: 188-193.
[13] David Bertozzi and Luca Benini, "Xpipes: A Network-on-chip Architecture for Gigascale System-on-chip". *Magazine, IEEE Circuits and Systems*, Second Quarter 2004, pp: 18-31.
[14] Srinivasan Murali, Theocharis Theocharides, N. Vijaykrishnan, Mary Jane Irwin, Luca Benini, Giovanni De Micheli, "Analysis of Error Recovery Schemes for Networks on Chips", *IEEE Design and Test*, September/October 2005 Vol. 22, No. 5, pp: 434-442.ISSN: 0740-7475.
[15] Dumitras, T.; Marculescu, R. "On-chip stochastic communication", Proceeding, Design, Automation and Test in Europe Conference and Exhibition, 2003 Volume , Issue , 2003, pp: 790 - 795.
[16] Karp R. et. al. "Randomized rumor spreading", *In Proceedings of the IEEE Symposium on Foundations of Computer Science*, 2000.
[17] M. Millberg, E. Nilsson, R. Thid, S. Kumar, A. Jantsch, "The Nostrum backbonea communication protocol stack for networks on chip", *In VLSI Design Conference*, Mumbai, India, January 2004.
[18] http://www.isi.edu/nsnam/ns/
[19] Vu-Duc Ngo, Hae-Wook Choi, "On Chip Network: Topology design and evaluation using NS2", *In Proceedings of the 7th International Conference on Advanced Communication Technology (ICACT 2005)*, Phoenix Park, Korea, Feb. 21-23, 2005.
[20] Y.-R. Sun, S. Kumar, and A. Jantsch, "Simulation and evaluation of a network on chip architecture using ns-2", *In Proceedings of the IEEE NorChip Conference*, November 2002.
[21] Alireza Vahdatpour, Ahmadreza Tavakoli, Mohammad Hossein Falaki, "Hierarchical Graph: A New Cost Effective Architecture for Network on Chip", *In Proceedings of the International Conference on Embedded And Ubiquitous Computing*, Nagasaki, Japan, December 2005.
[22] Franco Fummi, Giovanni Perbellini, Paolo Gallo, Massimo Poncino, Stefano Martini and Fabio Ricciato, "A Timing-Accurate Modeling and Simulation Environment for Networked Embedded Systems", *In Proceedings of the 40th Design Automation Conference (DAC)*, USA, June 2-6, 2003.