

On-line Fault Detection and Location for NoC Interconnects

Cristian Grecu¹, André Ivanov¹, Res Saleh¹, Egor S. Sogomonyan², Partha Pratim Pande³

¹ SoC Research Laboratory
Department of Electrical and Computer
Engineering
University of British Columbia
2356 Main Mall Vancouver, BC, V6T 1Z4 Canada

²University of Potsdam,
Fault Tolerant Computing Group, 14439
Potsdam, Germany

³School of Electrical Engineering &
Computer Science
Washington State University
PO Box 642752
Pullman, WA 99164-2752, USA

¹{grecuc, ivanov, res}@ece.ubc.ca; ²egor@cs.uni-potsdam.de; ³pande@eecs.wsu.edu

Abstract

A novel method for on-line fault detection and location in Network-on-chip (NoC) communication fabrics is introduced. This approach is able to distinguish between faults in the communication links and faults in the NoC switches. The idea is based on the use of code-disjoint routing elements, combined with parity check encoding for the inter-switch links. We analyze the effect of our method on relevant performance parameters – power, latency, and throughput. Experiments show that our approach is effective and requires minimal modifications of the existing design methods for NoC interconnects.

1. Introduction

Global on-chip interconnects are becoming a serious bottleneck in delivering performance and power consumption required by complex, multi-core SoCs. The problem of global interconnects is recognized as a challenging design constraint both in industry and academia. One of the solutions proposed by researchers is the use of packet-based on-chip interconnection networks, commonly referred to as networks-on-chip (NoCs) [1] [2], to address the issues of increasing interconnect complexity. The NoC design paradigm addresses the ever increasing delay and power dissipation of global on-chip interconnects by offering structured interconnect fabrics consisting of wire segments and intelligent routing blocks that ensure a certain level of Quality of Services (QoS) [3]. The QoS represents a collection of requirements on the NoC performance expressed by the achievable throughput (bandwidth), latency, power dissipation, and reliability.

At 65 nm and below, global wires are not able to deliver signals from one chip end to another in a single clock cycle. Pipelining and multi-cycle paths are *sine qua non* attributes of the deep-submicron interconnects. Thus, interconnect design is becoming a problem of circuit design, and the test (on- and off-line) must take these realities into account.

The major cause affecting the reliability of the VLSI global interconnects is the shrinking of the feature size, which exposes the interconnects to different faults of permanent, transient or intermittent nature. Among the

failure mechanisms we can enumerate factors such as crosstalk, electromigration, electromagnetic interference, alpha particle hit, and cosmic radiations. These faults can alter the behaviour of the NoC fabrics and degrade their QoS characteristics. Providing resilience from such faults is critical for the operation of NoC-based chips.

These factors motivated us to develop a method for concurrent error detection and location for NoC-based interconnects.

Traditionally, error detection and correction mechanisms are used to protect communication subsystems against the effects of transient malfunctions. Designers must carefully weigh the hardware cost of implementing such mechanisms for the on-chip data communication infrastructures against the potential benefits they can bring. Preliminary studies, e.g., [4] [5], show that complex error detection and correction (EDC) schemes may require unacceptably high energy dissipation and area overhead, and can adversely affect the performance of the NoC medium in terms of throughput and latency.

It is desirable that complex chips possess some form of self-checking mechanism. The self-checking property makes possible the on-line detection of faults in logic, without the need of externally applied test stimuli. EDC are widely employed for realizing self-checking systems, mainly due to design cost considerations and their flexibility. This is because they allow error recovery to be performed either in software or hardware.

In this paper we propose a scheme for on-line detection and location of faults in NoC interconnect infrastructures, based on the concept of code-disjoint error detecting circuits. Our detection scheme can differentiate between faults occurring in the network switches from faults in the inter-switch links. We evaluate the impact of our scheme on the overall performance of the NoC, and compare it with two recently proposed error detection (ED) mechanisms for NoCs, which we will briefly discuss in the following section.

2. Related work

The problem of error detection is thoroughly

studied in the context of traditional, off-chip communication networks [6] and VLSI stand-alone blocks such as memories [7], arithmetic-logic units [8], and other general-purpose digital blocks. Two frequently used codes for self-checking circuits are the parity check code and the dual-rail code. The former adds a parity bit to the information bits and detects all erroneous patterns with an odd number of faulty bits. The dual-rail code effectively doubles the amount of physical lines required to transmit the useful information, since it transmits both the original and complemented values for each bit. This overhead may not always be acceptable, in spite of better error detection efficiency compared to the parity-check code.

In [4], the error detection codes for NoC-specific application are classified in two broad categories: end-to-end (*e2e*) and switch-to-switch (*s2s*). In the end-to-end error detection schemes, ED codes are added to the data packets. At the sender and receiver interfaces, encoder and decoder circuits process the outgoing/incoming packets, respectively. The receiver interface sends a NACK or ACK signal back to the sender interface depending on whether the data is erroneous or not. The data is buffered at the sender interface and eventually retransmitted when a NACK signal is received; the sender buffers are cleared upon occurrence of a ACK signal.

In the switch-to-switch error detection scheme, error detection hardware is provided at each switch input and data is retransmitted between adjacent switches whenever an error is detected. The data is stored at each switch until the ACK signal is received from the subsequent switch on the data path. The amount of buffers required to implement the switch-to-switch error detection depends on whether the retransmission occurs at the packet or flit level.

Regardless of the particular implementation, we can identify two sources of extra power dissipation when an error recovery scheme is implemented based on detection and retransmission. The first one is the power required by the retransmission buffers, and the second is the power associated with the wires that carry the extra-bits needed to encode the data and the acknowledgment signal.

The potential problem associated with the above mentioned error detecting schemes is that the information they provide may not be sufficient to guarantee successful retransmission of corrupted data. Specifically, in the case of the end-to-end scheme, if the fault that causes the retransmission is of a non-transient nature (permanent or intermittent), then the retransmitted data cannot be guaranteed to be correct; on the contrary, it is very likely that it will be affected by the same fault. In such a case, the retransmission should take place on a different route than that was taken in the initial attempt. This naturally requires additional information on the location of the fault, in order to specify a new route for retransmission and avoid the NoC components affected by non-transient faults. In the end-to-end error detection, it is impossible to locate the position of the fault, since the checking takes place only at the destination interface.

The switch-to-switch scheme offers a better

potential for fault location: the data is checked for errors at each switch input, so if an error is detected, it must have occurred at the previous switch or on the link that connects the two switches. However, the exact location of the fault is very important when choosing a new route for retransmission. If the fault is located at a switch, then the switch needs to be marked as non-usable for all the subsequent messages; this in turn renders all the links connected to that switch unusable. If the non-transient fault appeared on an inter-switch link, then only that specific link has to be marked as invalid. From this example, it is evident that fault location is critical for any fault tolerant strategy. The reason behind this is that the location of non-transient faults affects in different ways the set of resources available to the fault tolerant mechanisms that eventually implement the retransmission of the erroneous packets.

Consequently, we propose a simple, effective method that allows detection of precise location of the faults.

3. Code-disjoint fault detection scheme

From the discussion in the previous section, we can infer that the data packets must be checked for errors both at the input and the output of the NoC switches, i.e., both the inputs and the outputs of the switches must be encoded.

Code-disjoint circuits form a special class of self-checking circuits that are characterized by this feature. Our objective is to design the switches such that they exhibit the code-disjoint property. In the following, we briefly present the background of code-disjoint circuits and a simple method to design code disjoint switches using parity codes, based on [9].

Code-disjoint circuits were originally developed for self-checking combinational circuits [9] [10]. Both their inputs and outputs are encoded. As long as no error occurs, input code-words produce output code-words, and non-code inputs produce non-code outputs.

3.1 Code-disjoint switches

Essentially, the switches of a NoC are not purely combinational circuits, but they can be divided into a control part (sequential) and a data path, which is pipelined [11] and unidirectional. The function of the switch is to direct messages from its input ports to its output ports according to the routing information stored in the header of the message. Therefore, a routing unit can be viewed as consisting of a data path (i.e., the circuitry that physically transports the messages from inputs to outputs), and a control part that decides to which output ports the incoming messages are to be directed.

Designing code-disjoint sequential circuits is a difficult task, since the coded output depends on the internal states. Based on our previous works, the sequential control blocks of NoC switches are small compared to their data paths [12]. Consequently, it is reasonable to modify only the data paths of the switches such that they have the code-disjoint property.

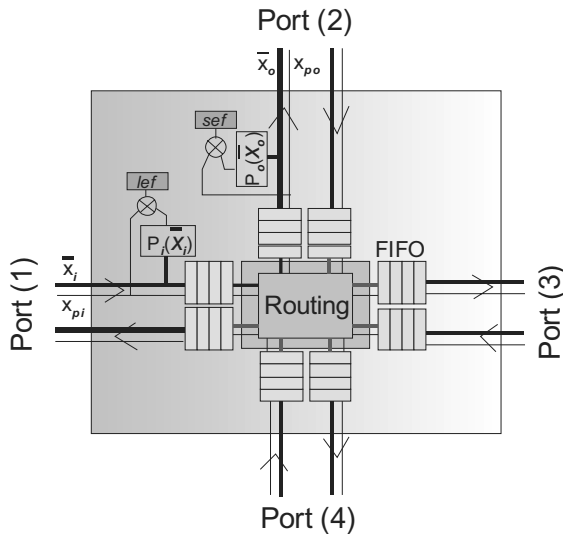


Figure 1: Code-disjoint switch (for clarity, the extra hardware blocks are only shown on the path from port (1) to port (2))

Our implementation is shown conceptually in Fig. 1. It is similar to the double output code-disjoint design proposed in [9]. As shown in Fig. 1, the incoming flit \bar{X}_i at port (1) has the incoming parity bit X_{pi} . The actual parity of the incoming flit is computed by the input parity prediction block $P_i(\bar{X}_i)$. If an error is detected, then the output of the $P_i(\bar{X}_i)$ block is different from X_{pi} . If no error is detected, the incoming flit \bar{X}_i is directed through the routing block towards the output of port (2) and becomes the outgoing flit \bar{X}_o . The parity of \bar{X}_o is computed again before the flit leaves the switch by the output parity prediction block $P_o(\bar{X}_o)$. If there is a mismatch between the output of the $P_o(\bar{X}_o)$ block and the incoming parity bit, then an error occurred on the data path from the input to the output of the switch. If there is no mismatch (no error was detected), the flit is forwarded towards the next hop on its route. The difference from the approach in [9] is that the incoming parity bit (X_{pi}) and the input-predicted parity bit (calculated by the $P_i(\bar{X}_i)$ block), are not both transmitted towards the output ports of the switches; instead, they are used locally to generate the link error flag (*lef*) which signals an error on the inter-switch link. If no error is detected at the switch input, the incoming parity bit is compared with the output parity bit (calculated by the $P_o(\bar{X}_o)$ block) and finally the switch error flag (*sef*) is generated. When an error is detected at the input of the switch, the transmission can be interrupted immediately without storing the flit and routing it to an output. Thus, latency and power can be significantly reduced, since no further processing is required when an erroneous incoming flit is detected.

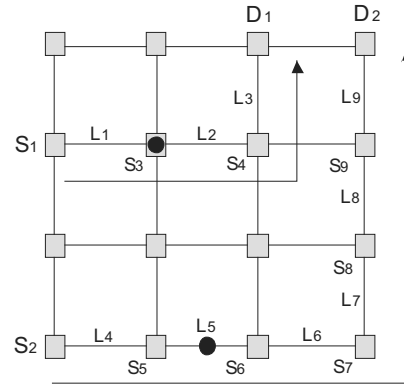


Figure 2: 4x4 mesh NoC with two fault sites: S3 and L5 (IP cores not shown).

To illustrate the fault location capabilities of the code-disjoint detection (*cdd*) scheme, let us consider the example of a 4x4 NoC as shown in Fig. 2, with two fault sites: one at switch S_3 , and the other at link L_5 . Two messages are routed, the first from the switch S_1 (source switch) to the switch D_1 (destination switch), and the second from switch S_2 to switch D_2 . If a fault appears at site S_3 , then the sets of possible fault sites returned by the three ED schemes discussed will be:

- *e2e*: $\{S_1, L_1, S_3, L_2, S_4, L_3, D_1\}$;
- *s2s*: $\{S_3, L_2\}$
- *cdd*: $\{S_3\}$

Similarly, if a fault appears at site L_5 , then the sets of possible fault sites returned by the ED schemes will be:

- *e2e*: $\{S_2, L_4, S_5, L_5, S_6, L_6, S_7, L_7, S_8, L_8, S_9, D_2\}$;
- *s2s*: $\{S_5, L_5\}$
- *cdd*: $\{L_5\}$

This example outlines the main advantage of the *cdd* scheme: it can detect *precisely* on which link/switch the fault appeared, so that only that element will be eventually marked as faulty.

In view of validating our approach, we need to estimate the effect of using code-disjoint switches on the performance of the NoC fabric. In [11], we analyzed the performance of different NoC fabrics with respect to a set of parameters such as power, latency and throughput. Here, we use the same parameters to estimate the performance of the code-disjoint detection scheme. We further augment our comparison by comparing our scheme with two previously proposed error detection methods, namely the end-to-end (*e2e*) and flit-level switch-to-switch (*s2sf*) [4].

3.2 Energy estimation

A generic energy estimation model describes the relation between the energy consumed by each unit of information (a flit in our case), the number of hop traversals and the energy consumed at each hop. To perform a consistent comparison with previous work [4], we characterized our designs (code-disjoint switches) using the **Predictive Technology Model** (PTM) developed at ASU [14] (<http://www.eas.asu.edu/~ptm/>), formerly known as BPTM (Berkeley Predictive Technology Model), for a 65 nm CMOS technology.

We define the following terms that help us formulate quantitative expressions for the power overhead required by error detecting schemes:

- IR : the rate at which the IP cores inject traffic (injection rate), measured in flits/cycle/IP;
- ST : amount of traffic at each switch (switch traffic), measured in flits/cycle/switch;
- STI : traffic increase due to retransmission at each switch (switch traffic increase);
- P_{PO} : power overhead caused in inter-switch links due to adding extra-bits through coding;
- N_{e2e} : size of buffers required at each end-switch to implement the end-to-end scheme;
- P_{PAR} : power consumption of the parity check circuits;
- P_b : power consumption of the retransmission buffers in the end-to-end scheme;
- P_{sw} : power consumption of a switch without error detection features.

With these notations, we can express the power overhead of the end-to-end scheme as:

$$P_{e2e} = \sum_{all\ IP_s} [IR \times (2 \times P_{PAR} + N_{e2e} \times P_b)] + \sum_{all\ switches} (STI \times P_{sw}) + \sum_{all\ links} P_{PO}$$

Eq. (1)

We can identify two major sources of power consumption in the end-to-end scheme: the first is due to the buffers necessary to store the packets (first term of Eq. 1), and the second is due to increased traffic generated by retransmitted messages (the second and third terms of Eq. 1).

Similarly, the power consumption of the flit-level switch-to-switch scheme (P_{s2sf}) can be expressed as:

$$P_{s2sf} = \sum_{all\ IP_s} [IR \times P_{PAR}] + \sum_{all\ switches} [ST \times (P_{PAR} + P_{RB})] + \sum_{all\ links} P_{PO}$$

Eq. (2)

where P_{RB} is the power consumption of each switch retransmission buffer. The major component of P_{s2sf} is caused by the power dissipation at each switch retransmission buffer.

Finally, the power dissipation of the code-disjoint detection scheme (P_{cdd}) can be expressed analytically as:

$$P_{cdd} = \sum_{all\ IP_s} [IR \times P_{PAR}] + \sum_{all\ switches} [ST \times ((1+a) \cdot P_{PAR} + b \cdot P_{RB})] + \sum_{all\ links} P_{PO}$$

Eq. (3)

where a and b denote the occurrence (or absence) of a fault at an inter-switch link ($a = 0$, $b = 1$) or within a switch ($a = 1$, $b = 1$).

Table 1 presents the individual power consumption of different components used in Eqs. (1), (2), and (3), consistent with the data available in [4] for the $e2e$ and $s2sf$ cases.

Table 1: Component power consumption

Component	Dynamic power [mW]
Routing	5
Control	2
Buffers	15
Total (P_{sw})	22
Parity check P_{PAR}	0.15
Retransmission buffers – $e2e$ scheme P_b - 1 packet ^[4]	2.5
Retransmission buffers – $s2sf$ ^[4] and cdd scheme P_{RB} - 1 flit	0.53
Coding overhead power P_{PO}	0.2

We obtained the values in Table 1 by implementing the corresponding blocks in a 0.18 μ m CMOS standard-cell based technology and using Synopsys' Prime Power tool to estimate power dissipation. The power numbers were then scaled to reflect the device parameters (in terms of gate capacitance, interconnect capacitance, leakage current) and voltage characteristic to the PTM 65 nm technology [14].

3.3 Latency estimation

Latency is defined as the time (in clock cycles) that elapses between the occurrence of a message injection at a source IP and the occurrence of a message reception at a destination IP. Retransmitted messages may block other messages on their way through the network, and consequently cause additional delays. Latency is influenced by both the message injection rate and the error rate, since both tend to increase the number of messages that compete for the same resources.

3.4 Throughput estimation

The occurrence of detected errors will be reflected by retransmission of data through the communication fabric, which will adversely impact the flow of useful data, and consequently the effective throughput of the NoC will decrease. Eventually, the effective throughput will degrade to a level below that deemed acceptable for normal operation of the NoC medium. We define the effective throughput of the NoC as the maximum number of successfully transmitted messages, in the presence of retransmission caused by detected errors.

4. Experimental results

Our objective was to estimate the efficiency of the code-disjoint detection scheme, and to compare it with

other error detection methods. In our experiments, we used a 4 x 4 mesh-connected network with 16 IP cores and 16 switches. The message length was kept constant at 4 flits, with a flit size of 64 bits. Messages were injected with a uniform traffic pattern (in each cycle, all IP cores can generate messages with the same probability). Though the self-similar (bursty) traffic appears to be a better approximation of real-life applications [15], for a consistent comparison with [4], we preferred here the use of uniform traffic. The routing mechanism used for all simulations was the *e-cube* (dimension order) routing [13].

We define the flit error rate as the probability of a flit having one or more erroneous bits, and study the efficiency of the three above mentioned error detection/correction schemes on the performance of the NoC.

Fig. 3 shows the power consumption of the three error detection schemes considered in this work, for variable flit-error rates. In all the cases, the *s2sf* scheme consumes more power, mainly due to the fact that more retransmission buffers are required. The *e2e* scheme is more power-effective (less retransmission buffers involved). The *cdd* approach offers a good compromise. This is because though it theoretically requires the same amount of retransmission buffers, not all of them are used effectively when an error is detected (if an error is detected at a switch input, the buffers of that switch are not used).

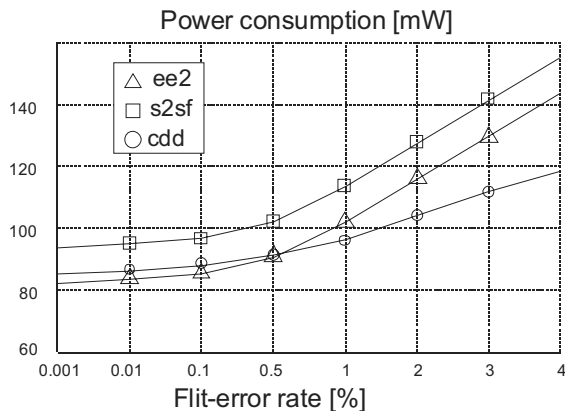


Figure 3: Power consumption of error detection schemes (injection rate = 0.1 flits/cycle/IP): end-to-end (e2e), flit-level switch-to-switch (s2sf), and code-disjoint detection (cdd).

The other important parameter under study is latency. There are two different factors that affect the latency of messages: the first is the injection of messages from the source IPs, and the second is the retransmission of messages when error recovery is attempted upon error detection. For low injection rates, the effect of retransmitted messages is not enough to saturate the network and provide useful information about the behaviour of the error detection/recovery schemes. Hence, we increased the injection rate close to the saturation so that the added effect of retransmissions can be made significant. The simulations showed that an injection rate of 0.25 flits/cycle/IP is close enough to

cause saturation [11] due to the added latency arising out of retransmission to be observable, for the particular architecture and traffic model under consideration. For different NoCs and/or traffic patterns, simulations have to be run in the error-free case to determine the appropriate injection rate at which the effect of retransmission needs to be studied.

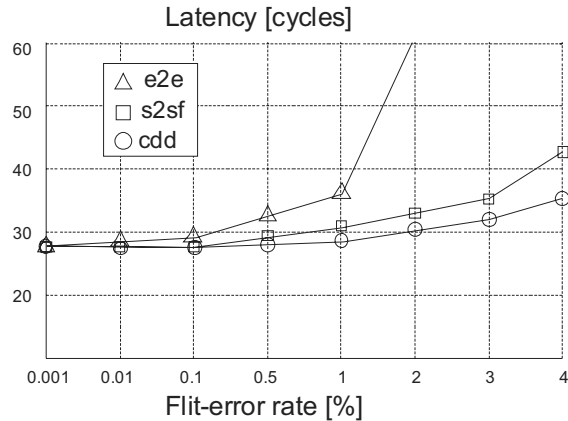


Figure 4: Average message latencies of error detection schemes (injection rate = 0.25 flits/cycle/IP): end-to-end (e2e), flit-level switch-to-switch (s2sf), and code-disjoint detection (cdd).

Our experiments reveal, as shown in Fig. 4, that the *e2e* scheme is adding the largest latency penalty, due to the fact that messages must traverse the network entirely from source to destination, taking up the largest amount of resources and blocking other messages on their path. The *s2sf* scheme behaves better, since error detection occurs at the flit level at each individual switch on the message path. Thus, only individual flits need to be retransmitted, and fewer messages are blocked by those retransmitted flits. The *cdd* scheme improves the *s2sf* case and provides a better latency, due to the fact that if errors are detected at the inputs of the switches, the corresponding flits are stopped and retransmission can begin immediately.

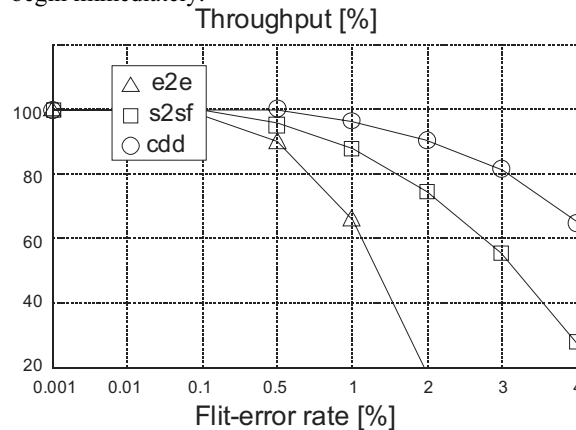


Figure 5: Relative throughput degradation for error detection schemes: end-to-end (e2e), flit-level switch-to-switch (s2sf), and code-disjoint detection (cdd).

Fig. 5 shows the effect of increasing flit-error rates on the throughput of the NoC under consideration. Again, simulations were run at a level approaching the saturating injection rate of the network under uniform traffic, such that the effect of the retransmitted messages be readily visible. Similar to the case of latency plots, the *e2e* scheme is the most prone to throughput degradation with increased flit error rates. Our code-disjoint scheme implies the smallest throughput penalty due to early detection of erroneous flits at the inputs of the NoC switches.

An additional advantage of the *cdd* scheme is the better flexibility that it offers with respect to subsequent recovery mechanisms. Regardless of the nature of the fault tolerant mechanism that designers may wish to implement, detailed knowledge of the error/fault location is crucial for its effectiveness. The *cdd* scheme can discriminate between errors on inter-switch links and errors on NoC switches, thus offering a more detailed information on the available fault-free resources that can be used for retransmitting erroneous data or rerouting faulty paths. We believe this to be a compelling argument for the *cdd* scheme to be used when developing fault tolerant mechanisms for NoCs.

6. Conclusions

Networks-on-chip are emerging as a solution for managing the complexity of nanometer-scale multi-core VLSI chips. Reduced feature size and DSM effects expose the data transport medium of these chips to high error rates, and special caution has to be taken to ensure their reliable, fault-free operation. In this paper, we presented a fault detection scheme based on a code-disjoint design, and evaluated and compared the performance of our scheme and other error detection/recovery mechanisms. The code-disjoint scheme compares favorably in terms of relevant NoC performance parameters such as power consumption, throughput, and latency. To the best of our knowledge, this is the first work that proposes the use of code-disjoint switches for error detection in NoC communication fabrics. This work is ongoing and we are looking actively into developing a quantitative analysis method to capture the fault location property of the code-disjoint scheme when applied to distributed environments such as networks-on-chip.

7. Acknowledgements

We thank Micronet, PMC-Sierra, Gennum, and NSERC for their financial support and the CMC for providing access to CAD tools. The authors also thank Prof. Michael Goessel of University of Potsdam for his invaluable comments.

8. References

- [1] L. Benini, G. De Micheli, "Networks on Chips: A New SoC Paradigm" *Computer*, Volume: 35 Issue: 1, Jan. 2002, pp.: 70-78.
- [2] P. Pande, C. Grecu, A. Ivanov, R. Saleh, G. De

- Micheli, "Design, Synthesis and Test of Networks on Chip", *IEEE Design and Test of Computers*, Vol. 22, No. 5, 2005, pp. 404-413.
- [3] E. Rijpkema, K. G. W. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, and E. Waterlander, "Trade Offs in the Design of a Router with Both Guaranteed and Best-Effort Services for Networks on Chip", *Proceedings of Design, Automation and Test Conference in Europe*, March 2003.
- [4] S. Murali, G. De Micheli, L. Benini, T. Theocharides, N. Vijaykrishnan, and M. Irwin, "Analysis of Error Recovery Schemes for Networks on Chips," *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 434-442, 2005.
- [5] D. Bertozzi, L. Benini and G. De Micheli, "Low-Power Error-Resilient Encoding for On-chip Data Busses," *Proceedings of Design, Automation and Test Conference in Europe*, 2000, pp. 102-109.
- [6] D. Bertsekas, R. Gallager, *Data Networks*, 2nd Edition. Prentice-Hall, 1992.
- [7] E. Dupont, M. Nicolaidis, P. Rohr, "Embedded Robustness IPs for Transient-Error-Free ICs", *IEEE Design & Test of Computers*, vol. 19, no. 3, pp. 54-68, 2002.
- [8] M. Nicolaidis, "Carry checking/parity prediction adders and ALUs", *IEEE Transactions on VLSI*, vol. 11, no. 1, pp. 121-128, 2003.
- [9] H. Hartje, M. Goessel, E. S. Sogomonyan. "Code-Disjoint Circuits for Parity Circuits," *Proceedings of the Sixth Asian Test Symposium (ATS'97)*, p. 100, 1997.
- [10] M. Goessel, S. Graf, *Error Detection Circuits*, McGraw-Hill, 1991.
- [11] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh. "Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025-1040, August, 2005.
- [12] P. P. Pande, C. Grecu, A. Ivanov, R. Saleh, "Design of a Switch for Network on Chip Applications", *IEEE International Symposium on Circuits and Systems, ISCAS 2003, Vol.V*, pp. 217-220.
- [13] J. Duato, S. Yalamanchili, L. Ni, *Interconnection Networks – An Engineering Approach*, Morgan Kaufmann, 2002.
- [14] Predictive Technology Model, <http://www.eas.asu.edu/~ptm/>.
- [15] G. Varatkar, R. Marculescu, "Traffic Analysis for On-chip Networks Design of Multimedia Applications", *Proceedings of DAC*, pp: 510-517, June 10-14, 2002