Achieving Network on Chip Fault Tolerance by Adaptive Remapping

Cristinel Ababei
Electrical and Computer Engineering Dept.
North Dakota State University, Fargo ND, USA
cristinel.ababei@ndsu.edu

Rajendra Katti
Electrical and Computer Engineering Dept.
North Dakota State University, Fargo ND, USA
rajendra.katti@ndsu.edu

Abstract

This paper investigates achieving fault tolerance by adaptive remapping in the context of Networks on Chip. The problem of dynamic application remapping is formulated and an efficient algorithm is proposed to address single and multiple PE failures. The new algorithm can be used to dynamically react and recover from PE failures in order to maintain system functionality. The quality of results is similar to that achieved using simulated annealing but in significantly shorter runtimes.

1. Introduction

Fault tolerance of Networks on Chip (NoC) [1] is becoming an important design concern [2] - [10]. In the context of networks, fault tolerance means the capability to maintain the ability to route packets in the presence of faulty components, and usually can be facilitated by system adaptivity [11]- [13].

Typically, faults represent data errors or system malfunctioning, which can occur due to fluxes of neutron and alpha particles, power and interconnect noise, broken physical links and routers or due to electromigration, stress migration, time dependent dielectric breakdown, and thermal cycles. The main techniques for improving fault tolerance are the use of spare components and reconfigurable links [5], [6] and adaptive or dynamic routing [9]. Router node reconfiguration and the use of deterministic routing in faulty parts and of adaptive routing in fault-free parts is proposed in [7]. Backup or multiple paths can be utilized to achieve spatial redundancy, which helps achieving tolerance against faults or errors. When routers are the faulty components, NoC fault tolerance is typically addressed by fault-tolerant routing, which must employ non minimal routing techniques or completely different communication paradigms [14]. In situations when the PE becomes the faulty component [15], the application has to be remapped on a different set of PEs to maintain and guarantee correct functionality. In this paper, we focus specifically on this issue by formulating the problem of dynamic application remapping to address PE failures. We propose an efficient two-step algorithm to solve this problem. The proposed algorithm is validated against a simulated annealing based approach.

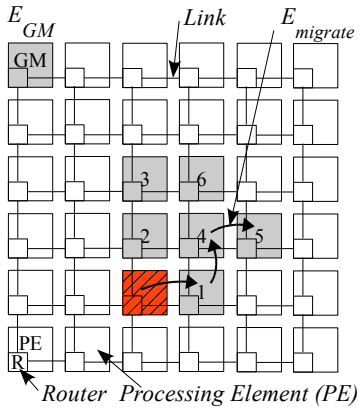

Figure 1. Assumed homogeneous 2D NoC architecture. Illustration of the main remapping energy consumption components.

2. Remapping for Fault Tolerance

2.1. Motivation Example

Let us consider a simple application running on the NoC shown in Fig. 1. Suppose that the PE located at the address $(2, 1)$ experiences a permanent failure, but the communication network (i.e., routers and physical links) remains functional. The application has to be remapped and possibly rescheduled in order to continue to work properly. The remapping has to be fast and with minimal task migration (i.e., displacement due to task relocation) in order to conserve energy. For example, the remapping in Fig. 1 requires three task relocations.

2.2. Network Architecture and Energy Model

We adopt the terminology introduced in [16] and use the bit energy metric proposed in [17] to model the total communication energy consumption of the network:

$$E_{comm} = E_{Lbit} \sum_{a_{ij} \in A} w(a_{ij}) dist(t_i, t_j) + E_{Rbit} \sum_{a_{ij} \in A} w(a_{ij})[dist(t_i, t_j) + 1] \quad (1)$$

where E_{Rbit} and E_{Lbit} represent the energy consumed by the router and link, respectively. $w(a_{ij})$ is the communication volume between two IP/cores and $dist(t_i, t_j)$ represents the Manhattan distance between tiles t_i and t_j. Eq. 1 states that the energy consumption is proportional to the total communication volume.

The NoC architecture (Fig. 1) is assumed to be similar to that suggested in [18]. Remapping is done by a general manager (GM), located on a selected tile of the network. The GM actions depend on the control signals that collect information on the status of every tile of the network. When the GM receives information about a possible PE failure, it executes the remapping algorithm and then sends back the remapping information. The remapping information can represent either task migration paths or complete reconfiguration information for PEs. The reconfiguration information could also be retrieved from selected local memories. The NoC architecture is assumed to contain shared memories distributed across the network that can store reconfiguration information and data [19].

Dynamic system reaction via remapping to PE failures will require additional energy consumed by the network monitoring/control and application remapping. The energy consumption due to remapping can be expressed as:

$$E_{remap} = E_{GM} + E_{migrate} \quad (2)$$

where E_{GM} is the energy consumed by the GM while running the remapping algorithm, and $E_{migrate}$ is the energy consumption required to relocate tasks to good PEs, which includes the energy consumption of the steps involved during reconfiguration. The first component in eq. 2 can be minimized by reducing the runtime of the remapping algorithm while the second component can be minimized by decreasing the amount of task migration (i.e., displacement) between the initial and new mappings. In this paper, our goal is to minimize both components by efficient remapping (hence shorter GM processing time) and minimization of the initial mapping displacement. The design of the actual hardware/software mechanisms to implement the actual system recovery (GM, PE failure detection, task migration, and control network) is outside the scope of this paper.

2.3. Application Remapping

2.3.1. Problem Formulation. The problem of *dynamic application remapping* can be formulated as follows.

Given the current application mapping of the application characterization graph $G(C, A)$ onto the PEs of the NoC architecture and a set of failures of PEs on which IP/cores of the current application are being executed;

Find a new region of good PEs and remapping function $M' : C \rightarrow R$ that maps the application IP/cores to routers (hence PEs/tiles) inside the new region, with the objective:

$$min\{E_{comm} + E_{remap}\} \quad (3)$$

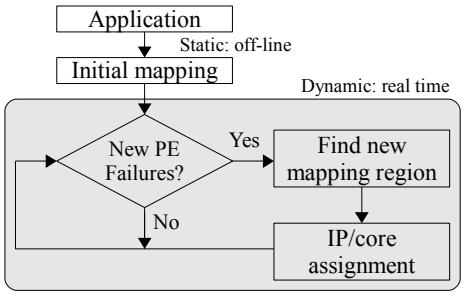

Figure 2. Block diagram of the proposed methodology.

This formulation is different from previous mapping problem formulations, such as [20]- [22], in that we start from a given initial mapping - obtained by using any mapping algorithm proposed in these works - which we assume to be already of high quality. Then, our goal is to efficiently find a new mapping region that uses only good PEs. The new mapping region should be as close as possible to the initial mapping region in order for the total application displacement to be minimized, hence the task migration component of E_{remap}. Moreover, E_{comm} minimization is equivalent to minimizing the total communication volume variation (eq. 1) between the initial and new mappings. In other words, the individual IP/cores remapping inside the new mapping region should be such that the new total communication volume should be as close as possible to that of the initial mapping.

2.3.2. Proposed Remapping Algorithm. Motivated by the above observations, we propose a two-step approach to solve this problem, as illustrated in Fig. 2.

• **Step 1:** *Find new mapping region*

We solve this subproblem by starting with the set of all the remaining good PEs on which the application was initially mapped. These PEs form the mapping region that needs to be augmented with additional good PEs to replace those that suffered failures. We iteratively grow this set by adding new PEs from its immediate vicinity. PEs closest to the mass center of the current mapping region are preferred first. This idea is similar to those presented in [18], [22], inspired from [23], based on the observation that the closer the new mapping region is to a convex shape, the closer it is to the optimal solution.

• **Step 2:** *Remapping of IP/cores within new mapping region*

We regard this subproblem as a linear assignment problem solved using the Kuhn-Munkres algorithm [24]. The bipartite graph contains two sets of nodes: nodes representing the application IP/cores and nodes representing the PEs that form the new mapping region. Edges connect every node from one set to all nodes in the other set (see Fig. 3.d). Edge weights are proportional to the distance between the initial tile location of an IP/core and the new PE locations

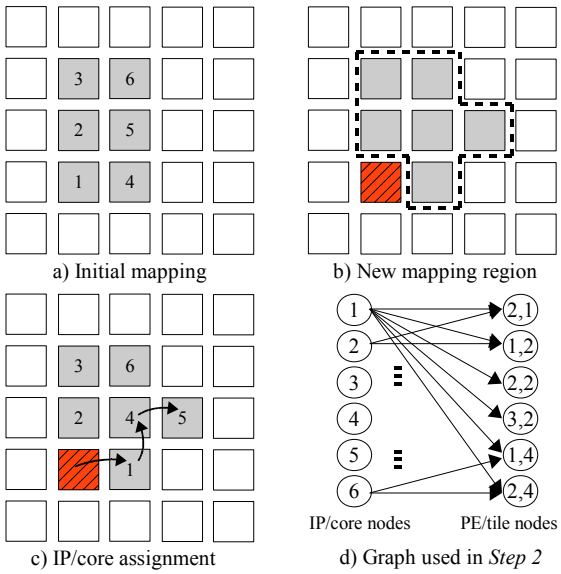

Figure 3. Illustration of finding the new near convex mapping region and assignment of IP/cores.

within the new mapping region. In this way we treat the assignment of all IP/cores *simultaneously* and achieve an overall minimal perturbation of each IP/core relative to the others, compared to the initial mapping. In other words, the total communication volume change between the initial and new mappings is minimized. For example, if the initial mapping of the application in Fig. 3.a experiences a PE failure, our algorithm will find the new mapping region shown in Fig. 3.b. After that, it will assign each IP/core to the PEs forming the new region as shown in Fig. 3.c.

The complexity of the first step is $O(m \log m)$ due to sorting, where m=$N \times N$ is the number of PEs of the NoC architecture. The second step has a complexity of $O(n^3)$, where n is the number of nodes of the bipartite graph.

3. Simulation Results

3.1. Multimedia Applications

We verified the proposed remapping algorithm using the *Video Object Plane Decoder (VOPD)* and *MPEG-4 decoder* from [25]. We also used the *DSP filter* from [20] and the *UseCase2* from [21]. The initial mappings of *VOPD* and *DSP filter* are from [20] and of *MPEG-4 decoder* and *UseCase2* are from [26] and [21].

3.1.1. Single Failures. First we consider single PE failures. Each of the four multi-media applications is simulated for single and two sequential single PE failure injection. The results for each application are averaged over all possible injections and are shown in Table 1.

In the first scenario, the average communication volume change - as percentage of the total communication volume of the initial mapping - between the initial and final mappings is small. This translates into reduced performance degradation. The average task migration (as number of hops) is also small, which translates into small energy consumption and short reconfiguration time during tasks migration.

Table 1. Single and two sequential single PE failures

		Single Failure		Two sequential Single Failures	
Testcase (Num. cores)	Arch. $N \times N$	Comm.vol. change [%]	Migr.	Comm.vol. change [%]	Migr.
VOPD (12)	4×4	8.01	3.5	19.14	6.77
	8×8	8.33	3.83	19.15	7.59
MPEG-4 (14)	5×5	13.63	6.43	27.44	9.41
	9×9	14.54	6.79	28.43	9.49
DSP filter (6)	3×3	21.11	2.33	32.44	4.33
	7×7	17.78	3	36	5.4
UseCase2 (7)	3×3	9.02	2	15.79	3.86
	7×7	9.02	2	14.79	3.95

Table 2. Simultaneous two and three PE failures

		Two Failures		Three Failures	
Testcase	Arch. $N \times N$	Comm.vol. change [%]	Migr.	Comm.vol. change [%]	Migr.
VOPD	4×4	17.25	6.73	28.77	9.85
	8×8	18.71	7.41	30.29	9.4
MPEG-4	5×5	28	8.99	39.93	12.03
	9×9	28.51	9.62	41.6	11.28
DSP filter	3×3	31.11	4.4	47.33	5.4
	7×7	31.11	5.67	40.7	6.9
UseCase2	3×3	16.54	3.95	-	-
	7×7	12.03	4.33	26.02	6.46

In the second scenario, we consider two PE failures that occur sequentially at different times. Remapping is done consecutively twice to address each failure. The second remapping uses as initial mapping the result of the first remapping. As expected, both the total communication volume change and average task migration deteriorate more than in the first scenario.

3.1.2. Multiple Simultaneous Failures. Here we test the proposed algorithm when two or three PE failures occur concurrently. The expected task migration is larger due to the increased number of PE failures which lead to larger disturbance. The results are shown in Table 2. Data are not available for *UseCase2* for a 3×3 NoC because there are only two good PEs available in this case. The variation of the total communication volume, normalized with respect to the failure-free initial mapping, for single, simultaneous two and three failures is shown in Fig. 4, which demonstrates graceful energy (proportional to communication volume) consumption degradation.

3.2. Comparison to Simulated Annealing

In order to further validate our algorithm we compare it against an in-house simulated annealing (SA) based

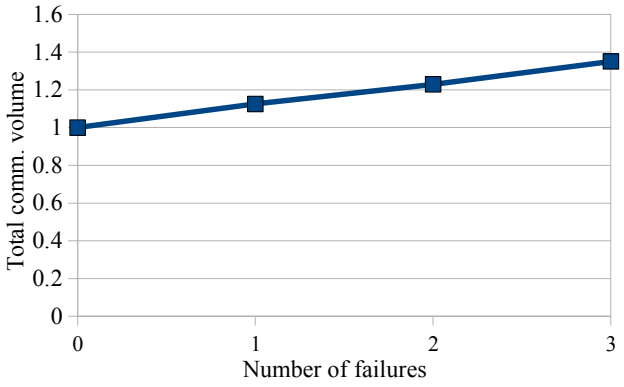

Figure 4. Variation of the communication volume after remapping for up to three simultaneous injected failures. Every data point represents the average of all testcases.

Table 3. Proposed algorithm vs. SA based algorithm

		Proposed		SA based	
Testcase	Arch. $N \times N$	Comm.vol. change [%]	Migr.	Comm.vol. change [%]	Migr.
VOPD	8×8	8.33	3.83	6.19	2.77
MPEG-4	9×9	14.54	6.79	15.09	3.43
DSP filter	7×7	17.78	3	12.12	1.83
UseCase2	7×7	9.02	2	6.77	1.86

remapping algorithm (see Table 3). The goal of the annealing process is to minimize both the application total communication volume and displacement from the initial mapping. The communication volume changes are similar, while the SA based algorithm achieves slightly smaller task migration amounts. The communication volume change achieved using the SA based algorithm can be geared toward smaller values by tuning the weight put on the displacement penalty. This is a trade-off, which depends on how the energy consumption due to task migration compares to the long-run extra energy savings achieved by a smaller communication volume change. The runtime of the proposed algorithm is in the range $30 - 60\mu s$ (on Intel Core Duo 2.8GHz and 2GB memory), which is at least $1000\times$ shorter than the runtime of the SA based algorithm.

4. Conclusion

This paper proposed the use of application remapping to achieve fault tolerance in the context of Network on Chips. An efficient remapping algorithm was introduced to address single and multiple PE failures. The proposed algorithm can be used to dynamically react and recover from PE failures in order to maintain system functionality.

References

[1] Giovanni De Micheli, Luca Benini, Networks on Chips: Technology and Tools, Morgan Kaufmann, 2006.

[2] A. Pullini, F. Angiolini, D. Bertozzi, L. Benini, "Fault Tolerance Overhead in Network-on-Chip Flow Control Schemes," *Symposium on ICSD*, 2005.

[3] M. Hubner, M. Ullmann, L. Braun, A. Klausmann, J. Becker, "Scalable Application-Dependent Network on Chip Adaptivity for Dynamical Reconfigurable Real-Time Systems," *International Conference on FPL*, 2004.

[4] S. Murali, D. Atienza, L. Benini, G. De Micheli, "A Method for Routing Packets Across Multiple Paths in NoCs with In-Order Delivery and Fault-Tolerance Gaurantees," *Hindawi VLSI Design*, 2007.

[5] T. Lehtonen, P. Liljeberg, J. Plosila, "Online Reconfigurable Self-Timed Links for Fault Tolerant NoC," *Hindawi VLSI Design*, 2007.

[6] D. Koch, C. Haubelt, J. Teich, "Efficient Reconfigurable On-Chip Buses for FPGAs," *IEEE Symposium on FCCM*, 2008.

[7] K. Kariniemi, J. Nurmi, "Fault tolerant XGFT network on chip for multi processor system on chip circuits," *Intl. Conf. on FPL*, 2005.

[8] M. Koibuchi, H. Matsutani, H. Amano, T. Mark Pinkston, "A Lightweight Fault-Tolerant Mechanism for Network-on-Chip," *International Symposium on Networks-on-Chip (NoCS)*, 2008.

[9] M. Ali, M. Welzl, M. Zwicknagl, S. Hellebrand, "Considerations for fault-tolerant network on chips," *International Conference on Microelectronics*, 2005.

[10] Z. Gu, C. Zhu, L. Shang, R.P. Dick, "Application-Specific MPSoC Reliability," *IEEE Tran. VLSI Systems*, 2008.

[11] Jose Duato, Sudhakar Yalamanchili, Lionel Ni, Interconnection Networks, Morgan Kaufmann Ed., 2003.

[12] T. Streichert, D. Koch, C. Haubelt, J. Teich, "Modeling and Design of Fault-Tolerant and Self-Adaptive Reconfigurable Networked Embedded Systems," *EURASIP Journal on Embedded Systems*, 2006.

[13] M. Hubner, L. Braun, D. Gohringer, J. Becker, "Run-time reconfigurable adaptive multilayer network-on-chip for FPGA-based systems," *International Symposium on IPDPS*, 2008.

[14] P. Bogdan, T. Dumitras, R. Marculescu, "Stochastic Communication: A New Paradigm for Fault-Tolerant Networks-on-Chip," *Hindawi VLSI Design*, 2007.

[15] A.K. Coskun, T. Simunic, K. Mihic, G.D. Micheli, Y. Leblebici, "Analysis and Optimization of MPSoC Reliability," *Jounal of Low Power Electronics*, 2006.

[16] R. Marculescu, U.Y. Ogras, L.-S. Peh, N.E. Jerger, Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," *IEEE Tran. VLSI Sys.*, 2009.

[17] T.T. Ye, L. Benini, G. De Micheli, "Analysis of Power Consumption on Switch Fabrics in Network Routers," *Proc. DAC*, 2002.

[18] C.-L. Chou, U.Y. Ogras, R. Marculescu, "Energy- and Performance-Aware Incremental Mapping for Networks on Chip With Multiple Voltage Levels," *IEEE Tran. CAD*, 2008.

[19] S. Bertozzi, A. Acquaviva, D. Bertozzi, A. Poggiali, "Supporting Task Migration in Multi-Processor Systems-on-Chip: A Feasibility Study," *Proc. DATE*, 2006.

[20] S. Murali, G. De Micheli, "SUNMAP: a tool for automatic topology selection and generation for NoCs," *Proc. DAC*, 2004.

[21] S. Murali, M. Coenen, A. Radulescu, K. Goossens, G. De Micheli, "A methodology for mapping multiple use-cases onto networks on chips," *Proc. DATE*, 2006.

[22] E. Carvalho, N. Calazans, F. Moraes, "Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs," *IEEE International Workshop on Rapid System Prototyping*, 2007.

[23] C.M. Bender, M.A. Bender, E.D. Demaine, S.P. Fekete, "What is the optimal shape of a city?," *Journal of Physics A: Mathematical and General*, 2004.

[24] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *Journal of the Society of Industrial and Applied Mathematics*, 1957.

[25] E.B. van der Tol, E.G.T. Jaspers, "Mapping of MPEG-4 decoding on a flexible architecture platform," *Proceedings of the SPIE, Media Processors*, 2002.

[26] D. Kim, K. Lee, S.-J. Lee, H.-J. Yoo "A reconfigurable crossbar switch with adaptive bandwidth control for networks-on-chip," *IEEE ISCAS*, 2005.