A Novel Approach Using a Minimum Cost Maximum Flow Algorithm for Fault-Tolerant Topology Reconfiguration in NoC Architectures

Leibo Liu, Yu Ren, *Chenchen Deng,
Shouyi Yin, Shaojun Wei

Institute of Microelectronics
Tsinghua University
Beijing 100084, China
Email: chenchendeng@tsinghua.edu.cn

Jie Han

Department of Electrical and
Computer Engineering
University of Alberta
Edmonton, AB, Canada T6G 2V4

Abstract - An approach using a minimum cost maximum flow algorithm is proposed for fault-tolerant topology reconfiguration in a Network-on-Chip system. Topology reconfiguration is converted into a network flow problem by constructing a directed graph with capacity constraints. A cost factor is considered to differentiate between processing elements. This approach maximizes the use of spare cores to repair faulty systems, with minimal impact on area, throughput and delay. It also provides a transparent virtual topology to alleviate the burden for operating systems.

I. Introduction

The advance in VLSI manufacturing technology has made it possible to integrate thousands of processing elements (PEs) on a single chip. In terms of communication infrastructure, Network-on-Chip (NoC) is considered as a promising interconnect scheme for manycore processors [1]. With the increasing circuit density, the reliability of a manycore system has become one of the most important challenges. Many solutions have been proposed to sustain the reliability of a system, including remapping [2], fault tolerant routing algorithms [3] and various topologies for implementing the communication infrastructure [4]. Improving the manufacturing process can help to increase the reliability, but this approach will become increasingly difficult in the future. A more practical solution is to provide redundant hardware to construct a fault-free system [5].

A reconfigurable system usually has many free resources. Due to its flexibility, these redundant resources can be utilized for improving reliability. In this paper, redundancies at the core level are considered, i.e. faulty PEs are replaced by spare ones. The concept of virtual topology [6] is also introduced because different chips may have different topologies and a faulty PE may change the underlying topology. A virtual topology is isomorphic to the topology of the target design. Topology reconfiguration is implemented by mapping between the virtual topology and the physical topology. With limited resources on a chip, an important question concerning topology reconfiguration is how to improve reliability by using spare resources with the least overhead.

In this paper, a novel approach using a minimum cost maximum flow (MCMF) algorithm [7] in graph theory is proposed for run-time topology reconfiguration. This method repairs faults in PEs and to improve the reliability of an NoC-based reconfigurable architecture at the cost of a minor performance reduction, compared to a fault-free system. The proposed approach successfully converts the topology reconfiguration problem into a network flow problem by constructing a directed graph based on the topology. A cost metric is introduced to model the overhead difference between PEs. Simulation results show that the success rate to repair all faulty PEs is increased by up to 40% compared with previous approaches [13] using the same redundant resources. The latency is 3.5% smaller and throughput is 4.7% higher than previous approaches [12]. Besides, the proposed approach has a polynomial computation time [7], which is suitable for run-time reconfiguration.

II. Design Consideration and Related Work

A. Design Consideration

Given a set of reliable and defective PEs, an objective is to obtain a system with the same functionality as the original one. The design consideration is how to improve the repair rate as much as possible with minimal impacts on the operational overhead, including the repair rate, the increase in reconfiguration time, the change of topology, and the increase in area, throughput and latency. As area, throughput and latency are common metrics for evaluating an NoC, they are not discussed in detail here. Three additional evaluation metrics including the repair rate, reconfiguration time and topology are introduced as follows.

Repair rate is an important metric to evaluate the effectiveness of a repair approach. It is defined as the probability that the faulty PEs in a topology can be successfully repaired by the spare ones. Different repair strategies result in different repair rates. Under the circumstances that the hardware resources on a chip are limited, more options are offered to each defective PE in this paper so that they can be efficiently repaired.

The reconfiguration time determines whether an approach can be performed at run-time. It depends on the required computation of the repair algorithm. If faults are detected at run-time and repaired by reconfiguration, the overall performance of the entire system will be improved. On the other hand, when chips are produced and tested massively, reconfiguration time is also an important parameter, because it is closely related to the cost of a chip. Therefore, a faster reconfiguration approach is preferred.

During configuration, topology is also one of the considerations. When the faulty cores are replaced by spare ones, the topology of the target design may become irregular and would cause performance degradation due to the lack of prior knowledge of the faulty cores. For example, Fig. 1 (a) shows a processor with 4×4 2D mesh topology. Suppose 4 spare cores are provided as shown in Fig. 1 (b). When faulty cores are present, as shown in Fig. 1 (c) and (d), different chips may have different topologies, and the topologies also may not be the same as expected. It will be a big burden for the operating system (OS) to optimize parallel programs on different topologies. To address this problem, a unified virtual topology is introduced. *Reference Topology* is defined as the topology of the target design, e.g. Fig. 1 (a). Fig. 1 (d) shows a topology with four spare cores. *Physical Topology* is the topology of fault-free cores and their interconnections, as shown in Fig. 1 (e). A fault-free 4×4 processor can still be obtained. Its topology is different but isomorphic to the reference topology. In a reconstructed chip, each core is considered to be virtually connected to its neighbors. *Virtual Topology* is defined as the reconstructed topology. Fig. 1 (f) is an example of a virtual 4×4 2D mesh topology. The 9th, 12th, 15th and 19th PEs are four virtual neighbors of the 13th PE. The 9th PE is considered to be virtually located under the 8th PE, although they are physically located side by side. A virtual topology appears as unified for the OS and other programs regardless of the underlying physical topology.

B. Related Work

There are many ways to implement the mapping between the virtual and physical topologies. One possible solution is to add a firmware layer to record the mapping information, similar to the CORE_AVAILABLE_REG used in UltraSPARC T1 processor [8]. OS works on the virtual topology, and the firmware is responsible for transformation. The idea of virtual topology is also applied in Cray T3E network [9]. The mapping from physical to virtual numbers is implemented by changing the routing table in each node and logically renaming the logical "who am I" register.

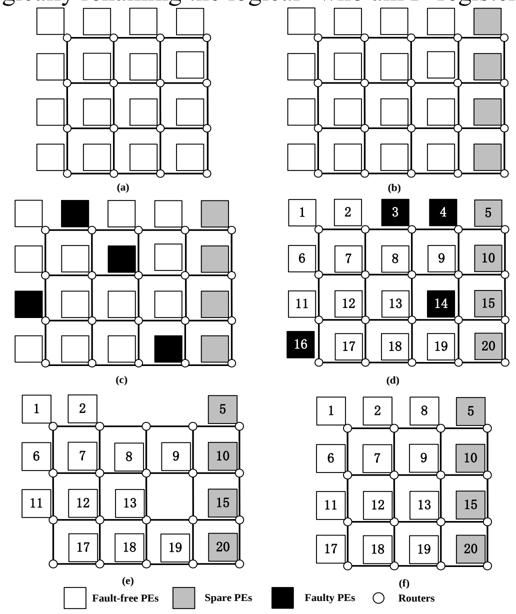


Fig. 1. (a) The expected target design. (b) The implementation on a chip. (c) and (d) A chip with faulty PEs. (e) The physical topology. (f) A virtual topology.

Faults can be divided into two main categories: permanent faults and transient ones. Permanent faults are usually caused by manufacturing defects, aging effects, and/or physical damages to the resources that generate or transport data. One approach to dealing with permanent faults is fault tolerant routing, which involves isolating the entire router [10] or a few ports of a router [11]. Another method for tolerating permanent faults is to use spare components to replace defective elements [12]. Transient faults are usually caused by neutron and alpha particles, power supply and interconnect noise, electromagnetic inference and electrostatic discharge. Error detecting/correcting codes are pervasively used to handle these errors. In this paper, only permanent faults are considered. Faults can occur at links, network interface, router and processing element levels. A VLSI processor integrates a large number of PEs on a single chip. As the size of a system increases and the cost of a single PE becomes relatively inexpensive compared with the entire system, PE-level redundancy is considered efficient. In this paper, only faults in PEs are considered while the communication infrastructure is assumed to be fault-free.

In [12], for an $N \times N$ NoC system, a spare row of routers is added. Every router within each column shares the common spare router. However, if a column contains more than one fault, spares in other columns cannot be utilized to repair the faults which renders a low repair rate. In [13], a repair strategy using two spares in one group is presented. It can tolerate two failures within one group. These two approaches are straightforward to implement. However, there is room for improvement in the repair rate. In [14], a novel repair technique is proposed to improve the yield of through-silicon vias (TSVs). This technique enables faulty TSVs to be repaired by redundant TSVs that are far apart. An NoC is used as the communication infrastructure, so the repair of TSVs is similar to the problem of repairing faulty PEs. Thus this approach is applicable to a manycore system.

III. Proposed Topology Reconfiguration Approach

In this section, the reconfiguration from physical to virtual topology is first introduced. Then reconfiguration algorithm is detailed. Besides, the overhead of reconfiguration time, throughput and latency is analyzed.

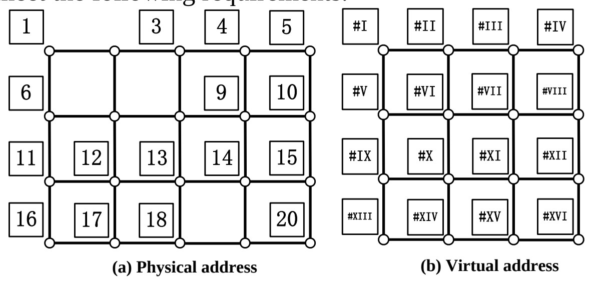
A. Topology Reconfiguration

Faulty PEs change the target design and the topology is reconfigured by mapping from various physical topologies to a unified virtual topology. Each router has a look-up table and two registers for storing its physical and virtual numbers. The index into the look-up table is the virtual address, while the entry in the table is the physical address. The look-up table provides a mapping from the physical topology to a virtual topology. The failure information can be obtained through various testing strategies. No matter what kind of testing approach is adopted, when faults are captured, the failure information is sent to the controller. The controller calculates the virtual topology using the proposed algorithms. Then the look-up table together with the virtual address register is logically renamed. OS works on the virtual topology. Fig. 2 shows an example of a virtual topology and the mapping table. A packet sent from the virtual address #II to #XVI is actually sent from the physical address 3 to 20. If XY routing is used with X-axis first, the routing of the packet will be 2 hops to the right and 4 hops downward.

B. *Minimum Cost Maximum Flow (MCMF) Approach*

A non-spare PE at location *(x, y)* is assumed to be faulty. In a valid repair solution, it is logically replaced by a healthy PE at location *(x', y')*. To be more specific, the PE at location *(x', y')* will be re-indexed as *(x, y)* in the reconfigured mesh. The PE at location *(x', y')* will then be replaced by a healthy PE at location *(x'', y'')* until the replacement ends at a spare PE. The ordered sequence of nodes *(x, y)*, *(x', y')*, *(x'', y'')*… involved in the replacement chain is defined as a repair path. It is a sequence of substitutions that logically replaces a faulty PE using a spare one. A general methodology to reconfigure a mesh with faulty PEs is equivalent to determining the repair paths. The repair paths determine the neighbors of each PE, and then a virtual topology is obtained. Fig. 3 shows an example to illustrate the concept of a repair path and the reconfigured virtual topology. The repair path is a virtual path indicating the replacement of PEs, and it does not physically exist. In contrast, the routing path is physically implemented by the NoC, and it is determined by the source and destination addresses.

If faulty PEs are detected, a repair path will start from a faulty PE and end at a spare one. Each PE along the repair path must be physically next to each other because PEs are assumed to be replaced by physical neighbors. If multiple repair paths are present, intersections are not allowed. Because each PE can only be mapped to one index in the virtual topology, an intersection means that the PE is mapped to two locations. In summary, the set of repair paths must meet the following requirements.



Mapping Table

Virtual address	Physical address	Virtual address	Physical address
#I	1	#IX	11
#II	3	#X	13
#III	4	#XI	14
#IV	5	#XII	15
#V	6	#XIII	16
#VI	12	#XIV	17
#VII	9	#XV	18
#VIII	10	#XVI	20

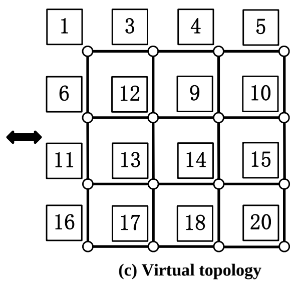


Fig. 2. Example of a virtual topology and the mapping table.

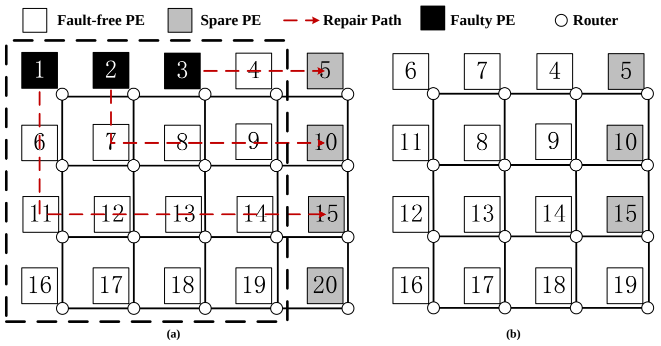


Fig. 3. (a) A 4×4 mesh with repair paths. (b) The reconfigured virtual topology.

1) Each repair path is continuous;
2) The set of repair paths covers all faulty non-spare PEs;
3) There is no intersection between any repair paths.

Next, a repair algorithm referred to as MCMF is proposed to analyze whether a mesh is repairable and how to generate a repair path set. The problem of determining a set of non-intersecting continuous repair paths can be converted into an MCMF problem. It is a classical combinatorial optimization problem, i.e., how to find the maximum flow between a source and a target in a network with capacity constraints (on nodes and edges). The relationship between repair paths and the MCMF is stated as follows (see Fig. 4).

Consider the mesh as a directed graph. Each continuous repair path can be seen as a unit flow starting from a faulty PE and ending at a spare PE. The grid then becomes a multi-source multi-target network. A unit capacity "1" on each edge and node ensures that an edge or a node can only be utilized once in the repair paths. By adding a super source node that points to all the faulty PEs and merging all the spare PEs into a target node, the grid is converted into a single-source single-target network. Since each repair path is defined by a unit flow from a source to a target in the network, the weight of the maximum flow is equal to the number of faulty PEs that can be repaired by spare PEs. When all the faulty PEs find their repair paths, i.e., all the faults can be repaired, the weight of the maximum flow is equal to the number of faulty PEs.

In a practical application, PEs differ from each other. Replacing a PE with another one will cause changes in the system, so PEs should not be treated equally. As a result, another variable, *cost*, is introduced to model these differences. Cost is used to describe the overhead of replacing a PE with another one. It can be any metric to model the differences of the network. For example, in a network with high throughput, edge delay is critical to guarantee the quality of communication, so cost is defined as the edge delay. In an area sensitive chip, cost is defined as the hardware consumption. Cost can be defined on an edge or a node, according to the problem requirement. As an example in this paper, the amount of data transmission on each PE is taken as cost. An H. 264 video-decoding application [15] is chosen to be the benchmark. The H. 264 decoding algorithm is computation-intensive. Subtasks are partitioned clearly, and they work independently of each other. So they can be mapped onto different PEs and be executed in parallel. Besides, the various amounts of computation and communication in each subtask meet the verification requirements of the proposed approach. H. 264 decoding algorithm is so widely used that resource codes and verification data can be found easily, so it is used as benchmark in this paper. Many other applications can be used as benchmarks, too. In the example of H. 264 video-decoding algorithm, cost is defined as the volume of transmitted data on each PE, because communication cost is an important factor in this application. An H.264 decoder is mainly composed of Inverse Discrete Cosine Transform – Inverse Quantization (IDCT-IQ), Motion Compensation (MC) and Deblocking blocks. The distribution of data transmission is calculated, as shown in Table I. Each block is randomly mapped onto one PE. Fig. 5 (a) shows an example of the weighted graph with three faulty PEs.

Now, the NoC system topology reconfiguration is converted into a minimum cost maximum flow problem in graph theory. The MCMF algorithm is performed in achieving the maximum flow of the directed graph while simultaneously minimizing the cost under predetermined cost constraints. A mesh is represented as a directed graph *G (V, E)*, where *V* is the set of nodes in the mesh, and *E* is the set of edges between nodes. *F* is the set of faulty nodes. Each node represents a PE and its corresponding router, while the directed edge connecting two nodes is the wire between two routers. Each edge and each node has a unit capacity. This is described mathematically as follows.

TABLE I
Proportion of data transmission of each block in an H.264 decoder

IDCT-IQ	MC	Deblocking	Others
52%	21%	17%	10%

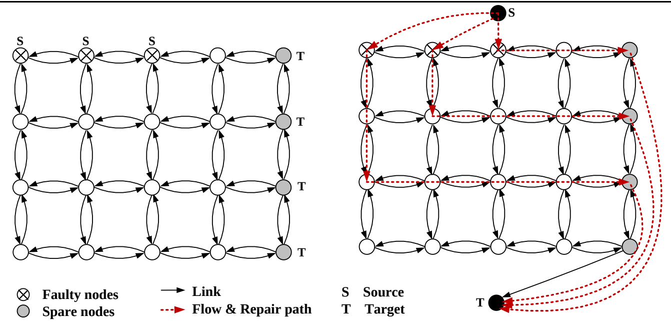


Fig. 4. MCMF algorithm for determining the repair paths. (a) Multiple-source multiple-target network. (b) Single-source single-target network with flow and repair path.

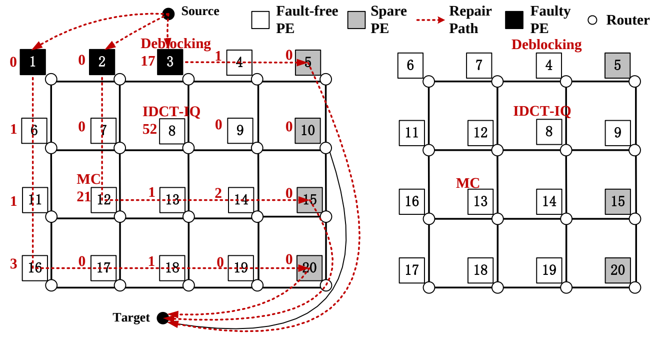


Fig. 5. (a) An example for H. 264 video-decoding application.
(b) Reconfigured virtual topology.

1. The set of nodes is defined as $V' = V \cup \{S, T\}$, where *S* is the source node, and *T* is the target node.

2. The set of edges *E* is defined as follows:

1) For every pair of nodes *(i, j)* that are adjacent in the grid, define two edges $i \to j$ and $j \to i$;

2) For every spare node $v_s \in V$, define an edge $v_s \to T$

3) For every faulty node $v_f \in F$, define an edge $S \to V_f$

3. Define the capacity of every edge to be 1.

4. Define the capacity of every node to be 1.

5. Solve the minimum cost maximum flow problem for the graph constructed above.

A solution to the above problem will return the maximum flow of the constructed graph, as well as individual flows. The maximum flow indicates how many faulty PEs can be repaired, and every flow represents a repair path. According to the repair path set, a virtual topology can be obtained, as shown in Fig. 5 (b). If the maximum flow is not equal to the number of faulty PEs, some faults are not repaired.

C. *Reconfiguration Time Analysis*

The feature of the maximum flow between source and target ensures that the faulty PEs are replaced by spare PEs as much as possible, therefore improving the reliability of the network. The MCMF problem has polynomial-time solutions, which runs in *O(ElogV(E+VlogV))* [7], where *E* is the number of edges and *V* is the number of nodes. If cost is not considered, i.e., every PE is considered to be identical, then MCMF is degraded to a maximum flow (MF) problem. In this way, the proposed algorithm has polynomial-time solutions. Apart from the execution of the algorithm, it will also take some time to reconfigure the look-up tables and the virtual-address registers. Taking all the reconfiguration bits into consideration, the refresh time is less than 3% of the algorithm execution time. Furthermore, in practical situations not all the routers and all the values in the look-up tables in one router need refreshing. Only partial reconfiguration is needed.

D. *Throughput and Latency Overhead Analysis*

From the viewpoint of the NoC, it is necessary to model the performance degradation of different virtual topologies. A metric named Distance Factor (DF) is introduced in [16]. It is used to describe the average hop count between virtual neighbors, so it reflects the average delay and throughput of a network. The distance factor between two nodes *m* and *n* is defined as the physical hops between them ($DF_{mn}=Hops_{mn}$). The distance factor of node *n* (DF_n) is defined as the average distance factor between node *n* and all its virtual neighbors.

$$DF_n = \frac{1}{k}\sum_{m=1}^{k} DF_{mn} \tag{1}$$

The DF of a topology is defined as the average DF_n of all its *N* nodes in the topology.

$$DF = \frac{1}{N}\sum_{n=1}^{N} DF_n \tag{2}$$

It is clear that the reference topology has the minimum DF. DF=1 in a mesh, which means that each pair of virtual neighbors is exactly one hop away from each other. Smaller DF indicates shorter communication delay among virtual neighbors.

Next, experiments are performed to evaluate the DF. Two repair schemes are used as baseline solutions for comparison, i.e., "N:1" [12] and "N:2" [13]. The N:1 scheme has one column of spare PEs on the right border. If there is one defective PE in a row, shifting is conducted to repair it with the spare one. This scheme can tolerate one failure in each row. The N:2 scheme has two spare PE columns, one on the left and one on the right border. This scheme can tolerate at most two failures in a row. For a fair comparison, MCMF is considered to have the same spare resources as the baseline schemes. In the comparison between MCMF and N:1, both approaches are conducted on a 4×5 mesh. The MF approach is also performed for comparison, in which the data transmission of each PE is the same. These two approaches are also compared with N:2 on 4×6 mesh. For each topology, 10,000 different fault patterns are considered, with PE failure rate ranging from 1% to 10%. DF is calculated only when all the faulty patterns in the network could be repaired. Fig. 6 shows the DF comparison results. As shown in the figure, DF increases with the increase of failure rate. In Fig. 6 (a), the N:1 scheme has the same DF as MF. This is because under the circumstances that all the test fault patterns can be repaired by both approaches, the fault patterns that can be repaired by N:1 can also be repaired using MF in the same way. In other words, N:1 scheme is a subset of MF. The DF of MCMF is a little higher than MF and N:1. By taking cost into consideration, MCMF tries to find the maximum flow with the minimum cost. Because the cost varies among PEs, the reconfigured topology may become unbalanced resulting in a larger DF than MF. In Fig. 6 (b), compared with N:2, both MCMF and MF have smaller DF. Because DF is used to describe the average hop between virtual neighbors, it is predicted that the throughput and latency using MCMF and MF is better than the baseline schemes.

IV. Experimental Results

While reconfiguration time is analyzed in the previous section, the performance of repair rate, throughput and latency are evaluated on a cycle-accurate simulator. The proposed approach is also implemented using Verilog and verified on FPGA. In the following experiments, the proposed approach uses the same topology and area as those in the baseline approach.

A. Experimental Setup

The baseline schemes are the same as in DF calculation, i.e. N:1 and N:2. The repair rate is obtained from simulation using Matlab. Throughput and latency are measured in a C++ NoC platform using Carbon SoC Designer. The packet length is set to be 16 flits. Each input port has 3 virtual channels and each channel has a FIFO buffer to store 4 flits. Each PE injects packets independently and the destination of a packet is randomly determined. XY routing is used, which routes packets along the X-axis first, and then Y-axis. The performance measures include system throughput and latency. Average delay is the time required for a packet to traverse the network from source to destination. Network throughput is the packets delivering rate for a particular traffic pattern. Each scheme may generate a different virtual topology for a given fault pattern. The mapping between the virtual topology and physical topology is given by a look-up table. Thus the difference between the NoC models lies in the look-up tables.

B. Repair Rate

Repair rate is the probability that faulty PEs can be successfully repaired by spare ones. PEs are assumed to work independently. All PEs including the spare ones are subject to failures. The number of faulty PEs is varied from 1 to the maximum number of spare PEs. For each number of faults, 3000 fault patterns are randomly generated. Fig. 7 shows the repair rate comparison between different spare topologies and mesh sizes. With the increase of faults, the repair rate using the two baseline schemes drops significantly while MCMF maintains a much higher repair rate. The N:2 scheme performs well for small meshes with a repair rate of over 90%, but in the case of large meshes, the repair rate drops by nearly 40% at the failure rate of 10%. The N:1 scheme can only tolerate one fault at each row and the N:2 scheme can tolerate at most two faults at each row. When the number of faults increases, even if the faulty PEs are fewer than the spare ones, they cannot be fully repaired. In other words, the utilization efficiency of the spare hardware is low. The PEs used for repair in MCMF are not restricted to the faulty row, and it is more capable of using spare PEs to repair faults.

It can also be observed that the repair rate of MCMF in an 8×9 mesh is higher than that of N:2 in a 4×6 mesh although they both have 8 spare PEs. This indicates that redundancy is not the only dominating factor for determining the final repair rate. It implies that by using the MCMF algorithm, a higher repair rate with less redundant resources can be achieved. Hence, this algorithm can reduce the redundant hardware required to obtain a high repair rate.

C. Throughput and Latency Overhead

The H. 264 video-decoding application is taken as an example to measure the performance of MCMF. MCMF and N:1 approaches are implemented on a 4×4 mesh, with 4 spare PEs in a column on the right border, the same as Fig. 1 (b). 4 out of the 20 PEs are randomly chosen to be faulty. 100 different faulty patterns are generated. A mesh with no faults is also simulated for comparison. The average latency and throughput are shown in Fig. 8. It can be seen that the average latency increases with the increase of traffic volume. Compared to N:1, the latency of MCMF is decreased by 3.5%. Compared with the fault-free system, the latency of MCMF is increased by 7.0%~10.1%. The throughput of MCMF is 4.7% higher than N:1. Compared to the fault-free system, the throughput of MCMF is decreased by 4.1%~5.3%. If the differences of PEs are not considered, MCMF is degraded to a MF algorithm. The data transmission is the same between PEs. As an ideal case of MCMF, the latency of MF is 4.5% smaller than N:1 and 5.3% smaller than N:2. The throughput is 11.3% higher than N:1 and 6.3% higher than N:2.

V. Conclusions and Future Work

Effective fault-tolerant techniques are critical to ensure the reliability of integrated circuits. In this paper, an approach using an MCMF algorithm is proposed for topology reconfiguration to improve fault-tolerance. Cost is used to model different PEs in practical applications. Experiment results show that the proposed approach achieves a higher repair rate, higher throughput and shorter delay than other approaches with the same topology. In addition to that, a polynomial reconfiguration time is achieved.

The proposed approaches are not restricted to use in NoC-based manycore systems; they are also applicable to many other highly integrated systems. For example, in a reconfigurable system with many free PEs, how to organize and utilize these PEs for communication and computation during run-time reconfiguration presents a significant challenge. The approaches presented in this paper may be useful for addressing this issue and at the same time, providing a unified topology for the OS and other programs.

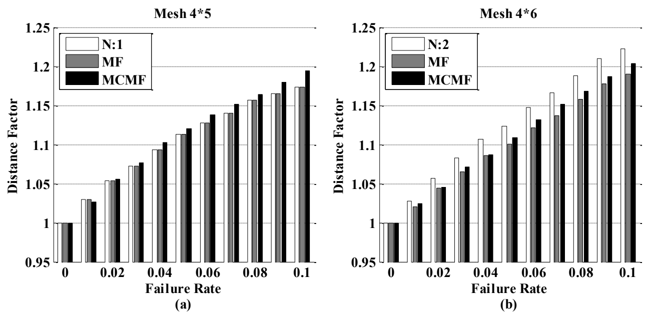


Fig. 6. DF comparison between MCMF, MF, N:1 and N:2.

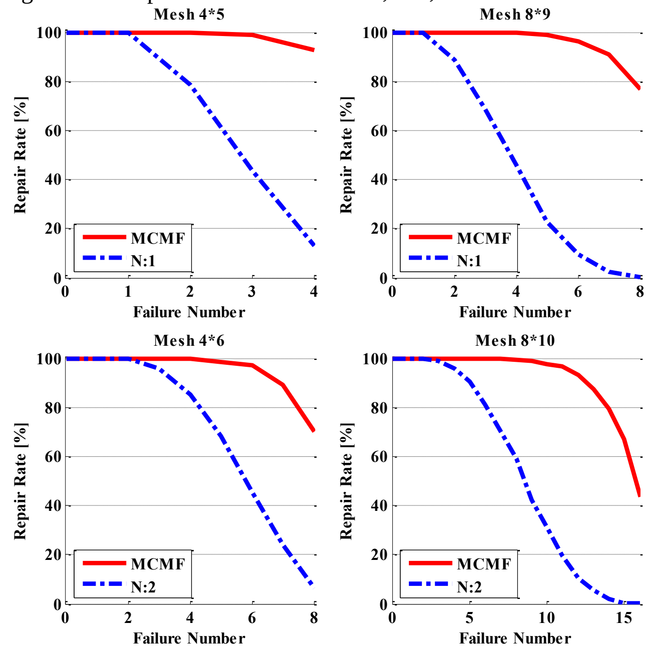


Fig. 7. Repair rate comparison.

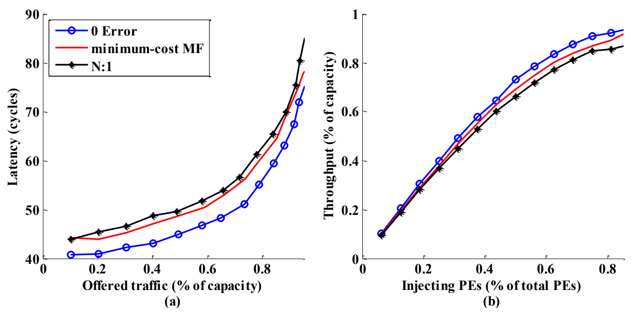


Fig. 8. Latency and throughput comparison between MCMF and N:1 approaches.

References

[1] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," *Proc. DAC*, pp. 684-689, 2001.

[2] O. Derin, D. Kabakci, and L. Fiorin, "Online task remapping strategies for fault-tolerant Network-on-Chip multiprocessors," *Symp. NoCS*, pp. 129-136, 2011.

[3] M. Ebrahimi, M. Daneshtalab, F. Farahnakian, et al., "HARAQ congestion-aware learning model for highly adaptive routing algorithm in on-chip networks," *Symp. NoCS*, pp. 19-26, 2012.

[4] M. Janidarmian, V. S. Bokharaie, A. Khademzadeh, et al., "Sorena: new on chip network topology featuring efficient mapping and simple deadlock free routing algorithm," *Conf. Computer and Information Technology*, pp. 2290-2299, 2010.

[5] Y. Ren, L. Liu, S. Yin, et al., "A fault tolerant NoC architecture using quad-spare mesh topology and dynamic reconfiguration," *J. Systems Architecture*, vol. 59, no. 7, pp. 482-491, Aug. 2013.

[6] A. Gencata and B. Mukherjee, "Virtual-topology adaptation for WDM mesh networks under dynamic traffic," *J. IEEE/ACM Trans. on Networking (TON)*, vol. 11, no. 2, pp. 236-247, Apr. 2003.

[7] J. B. Orlin, "A faster strongly polynomial minimum cost flow algorithm," *J. Operations research*, vol.41, no.2, pp.338-350, 1993.

[8] P. J. Tan, T. Le, K.-H. Ng, et al., "Testing of UltraSPARC T1 Microprocessor and its Challenges," *IEEE International Test Conference*, pp. 1-10, 2006.

[9] S. L. Scott and G. M. Thorson, "The Cray T3E network: adaptive routing in a high performance 3D torus," *Proc. Hot Interconnects IV Symposium*, pp. 147-156, Aug. 1996.

[10] M. B. Stensgaard and J. Sparso, "ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology", *Symp. NoCS*, pp. 55-64, 2008

[11] D. Fick, A. DeOrio, J. Hu, et al., "Vicis: A reliable network for unreliable silicon," *Proc. DAC*, pp. 812-817, 2009.

[12] Y.-C. Chang, C.-T. Chiu, S.-Y. Lin, et al., "On the design and analysis of fault tolerant NoC architecture using spare routers," *Proc. ASP-DAC*, pp. 431-436, 2011.

[13] U. Kang, H.-J. Chung, S. Heo et al., "8 Gb 3-D DDR3 DRAM using through-silicon-via technology," *J. Solid-State Circuits (JSSC)*, vol. 45, no. 1, pp. 111-119, 2010.

[14] L. Jiang, Q. Xu and B. Eklow. "On effective TSV repair for 3D-stacked ICs," *Conf. DATE*, pp. 793-798, 2012.

[15] Y. Kim and S. Sair, "Designing real-time H.264 decoders with dataflow architectures," *Conf. Hardware/Software Codesign and System Synthesis*, pp. 291-296, 2005.

[16] L. Zhang, Y. Han, Q. Xu, X. Li and H. Li, "On topology reconfiguration for defect-tolerant NoC-based homogeneous manycore systems," *IEEE Trans. VLSI*, vol. 17, no. 9, pp. 1173-1186, Sep. 2009.