Efficient Fault-Tolerant Topology Reconfiguration Using a Maximum Flow Algorithm

YU REN, Tsinghua University
LEIBO LIU*, Tsinghua University
SHOUYI YIN, Tsinghua University
JIE HAN, University of Alberta
SHAOJUN WEI, Tsinghua University

With an increasing number of processing elements (PEs) integrated on a single chip, fault-tolerant techniques are critical to ensure the reliability of such complex systems. In current reconfigurable architectures, redundant PEs are utilized for fault tolerance. In the presence of faulty PEs, the physical topologies of various chips may be different, so the concept of virtual topology from network embedding problem has been used to alleviate the burden for the operating systems. With limited hardware resources, how to reconfigure a system into the most effective virtual topology such that the maximum repair rate can be reached, presents a significant challenge. In this paper, a new approach using a maximum flow (MF) algorithm is proposed for an efficient topology reconfiguration in reconfigurable architectures. In this approach, topology reconfiguration is converted into a network flow problem by constructing a directed graph; the solution is then found by using the MF algorithm. This approach optimizes the use of spare PEs with minimal impacts on area, throughput and delay, thus it significantly improves the repair rate of faulty PEs. In addition, it achieves a polynomial reconfiguration time. Experimental results show that compared with previous methods, the MF approach increases the probability to repair faulty PEs by up to 50% using the same redundant resources. Compared with a fault-free system, the throughput only decreases by less than 2.5% and latency increases by less than 4%. To consider various types of PEs in a practical application, a cost factor is introduced into the MF algorithm. An enhanced approach using a minimum-cost MF algorithm is further shown to be efficient in the fault-tolerant reconfiguration of heterogeneous reconfigurable architectures.

Categories and Subject Descriptors: **B.8.1 [Performance and Reliability]**: Reliability, Testing, and Faulty-Tolerance

General Terms: Reliability, Performance, Algorithms

Additional Key Words and Phrases: Fault tolerance, reconfigurable architecture, topology reconfiguration

1. INTRODUCTION

The advance in VLSI manufacturing technology has made it possible to integrate thousands of processing elements (PEs) on a single chip, which proves itself very useful in parallel processing applications [Borkar 2007]. In terms of communication infrastructure, Network-on-Chip (NoC) is generally considered as a promising interconnect scheme for manycore processors [Dally and Towles 2001]. The aggressive integration of computation and communication elements has caused great challenges on the reliability of such systems. Many solutions have been proposed to sustain the reliability of a system, including remapping [Derin et al. 2011], fault tolerant routing algorithms [Ebrahimi et al. 2012] and various topologies for implementing the communication infrastructure [Janidarmian et al. 2010, Kariniemi et al. 2006]. Improving the manufacturing process can help to increase the reliability, but it will become increasingly difficult in the future. A more practical solution is to construct a fault-free system by providing redundant hardware [Ren et al. 2013]. For example, built-in-self-repair has been widely used to cope with the increasing defects

This work is supported in part by the China National High Technologies Research Program (No. 2012AA012701).

Author's addresses: Y. Ren, L. Liu, S. Yin and S. Wei, Institute of Microelectronics and the National Lab for Information Science and Technology, Room 4-404, FIT Building, Tsinghua University, Beijing 100084, China; J. Han, Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Canada, T6G 2V4.

Corresponding author: liulb@tsinghua.edu.cn.

in memories using redundant circuits [Lee et al. 2011]. Such techniques have been extended to other types of VLSI circuits as well [Koren and Pradhan 1986].

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 processor or core level are considered, i.e. faulty PEs are replaced by spare ones. Because different chips may have different topologies and a faulty PE may change the underlying topology, it becomes a major challenge for the operating system (OS) to optimize the parallel programs for various topologies [Jan and Jozwiak 2012]. To address this issue, this paper reintroduces the concept of virtual topology originating from a network embedding problem [Gencata and Mukherjee 2003]. A virtual topology is isomorphic with the topology of the targeted design [Zhang et al. 2009]. Topology reconfiguration is implemented by mapping between the virtual topology and the physical topology. Using this technique, the OS and other programs always see a unified virtual topology regardless of the underlying physical topologies. With limited resources on a chip, an important question concerning topology reconfiguration is how to improve reliability by using the most of the spare resources with the least overhead. Such overhead includes the reconfiguration time, throughput, latency and area. Meanwhile, a functionally identical topology is also one of the considerations.

In this paper, a novel approach using a maximum flow (MF) algorithm [Goldberg and Tarjan 1988] in graph theory is first proposed for run-time topology reconfiguration to repair faults in PEs and to improve the reliability of a reconfigurable architecture. Our approach successfully converts the topology reconfiguration problem into a network flow problem, which can be solved mathematically using the MF algorithm. This approach maximizes the use of spare resources to improve the reliability with little overhead on area, throughput and delay. It also provides a unified virtual topology that helps the operating system to optimize parallel programs for various topologies. A polynomial computation time makes this approach suitable to run-time reconfiguration. Furthermore, to take various types of PEs in a practical application into consideration, a cost factor is used to model different PEs. An improved approach using a minimum-cost MF algorithm is proposed. Experiments show that the proposed approach outperforms other approaches in improving the probability to repair faulty PEs. It is also proved to be efficient in practical applications with heterogeneous reconfigurable architectures.

The rest of this paper is organized as follows. Section II introduces the design consideration and related work. In Section III, the problem is formulated and the proposed MF and minimum-cost MF topology reconfiguration algorithms are described in detail. In Section IV, experimental results are presented. Finally, Section V concludes this paper and future work is discussed.

2. DESIGN CONSIDERATION AND RELATED WORK

2.1 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, such as 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. Next, three additional evaluation metrics are introduced, i.e., the repair rate, reconfiguration time and topology.

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. Because the hardware resources on a chip are limited, our focus is to offer more repair options for each defective PE 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 reconfiguration time will have an effect on the overall performance of the entire system. 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. Because there is no prior knowledge about the faulty cores, when the faulty cores are replaced by spare ones, the topology of the target design may become irregular and would cause performance degradation for manycore processors. For example, Fig. 1 (a) shows a processor with 4×4 2D mesh topology. Suppose 4 spare cores in one column are provided to improve the reliability of the chip as shown in Fig. 1 (b). When faulty cores are replaced by spare ones, as shown in Fig. 1 (c) and (d), different chips may have different topologies, and the topologies may be also different from what we expect. It will be a big burden for the OS to optimize parallel programs on different topologies [Stallings 2011]. To address this problem, a unified virtual topology is introduced next. First, *Reference Topology* is defined as the topology of the target design [Zhang et al. 2009]. For example, Fig. 1 (a) shows the reference topology of a 4×4 2D mesh. Fig. 2 (a) shows a topology with four spare cores, in which the 2^{nd}, 7^{th}, 8^{th} and 19^{th} cores are assumed to be defective. *Physical Topology* is considered to be the topology of fault-free cores and their interconnections, as shown in Fig. 2 (b). Although this topology is different from the reference topology, the remaining cores can still construct a 4×4 processor. It should be noted that once a chip is taped out, the physical topology is determined and cannot be changed during lifetime. In a reconstructed chip, each core is considered to be virtually connected to its neighbors. *Virtual Topology* is then defined as the reconstructed topology. Fig. 2 (c) is an example of a virtual 4×4 2D mesh topology. In Fig. 2 (c), the 3^{rd}, 6^{th}, 9^{th} and 13^{th} PEs are four virtual neighbors of the 12^{th} PE. The 13^{th} PE is considered to be virtually located under the 12^{th} PE, although they are physically located side by side. Even though the 9^{th} PE is more than one hop away from the 12^{th} PE, they are considered to be adjacent in the virtual topology. A virtual topology appears as unified for the OS and other programs regardless of the underlying physical topology. The idea of constructing a virtual topology based on a physical topology originates from the network embedding problem. This makes it much easier for the OS to optimize parallel programs and dispatch tasks [Zhang et al. 2009].

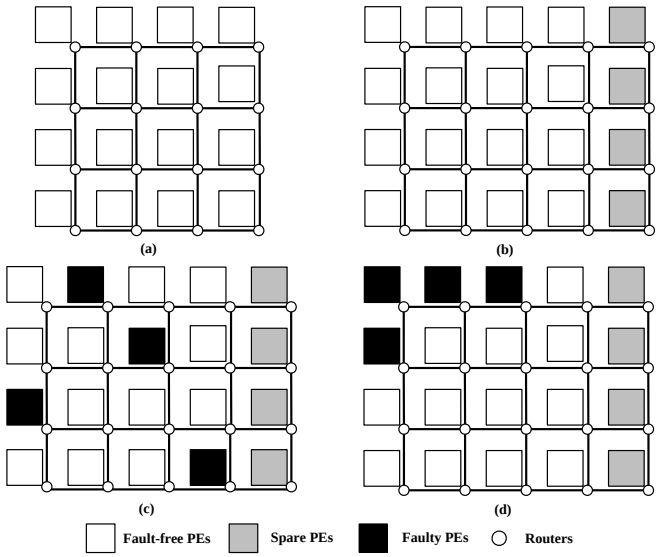


Fig. 1. Faulty PEs change the topology of a target design. (a) What we expect in a target design. (b) What we implement on a chip. (c) and (d) A chip with faulty PEs.

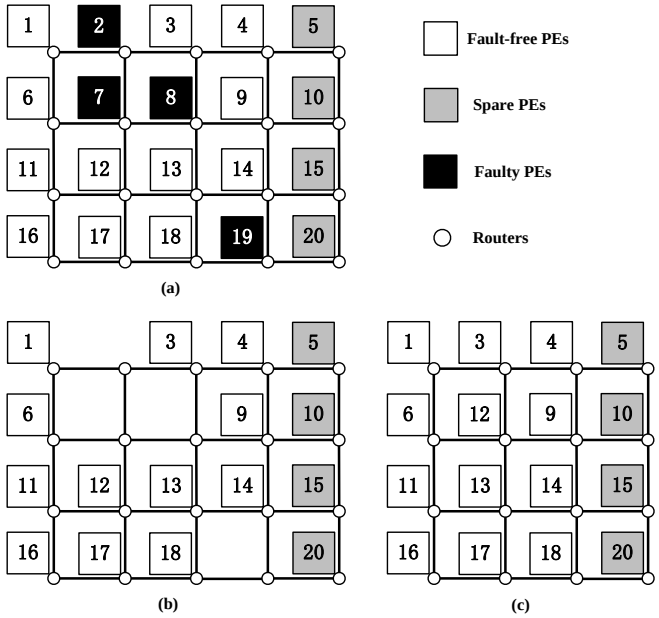


Fig. 2. (a) A chip with faulty PEs. (b) The physical topology. (c) A virtual topology.

2.2 Related Work

There are many ways to implement the mapping between the virtual topology and the physical topology. 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 [Tan et al. 2006]. OS works on the virtual topology while the firmware is responsible for transformation. The idea of virtual topology is also applied in Cray T3E network [Scott and Thorson 1996]. 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.

Faults can be divided into two main categories: permanent and transient [Constantinescu 2003]. 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 [Stensgaard and Sparso 2008] or a few ports of a router [Fick et al. 2009]. Another method for tolerating permanent faults is to use spare components to replace defective elements [Chang et al. 2011, Han and Jonker 2003]. 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 memories. 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. The communication infrastructure is assumed to be fault-free. Generally, there are two main methods to deal with faulty cores on a chip [Zhang et al. 2009]. One is to obtain a degraded chip by disabling faulty cores. With the integration of on-chip PEs, however, the cost of a single PE becomes inexpensive compared with the entire chip, so the second method uses PEs as spares to improve reliability. When faults occur, the faulty PEs are replaced by the spare ones. If the spare PEs cannot replace all the faulty ones, the chip has to be discarded. We focus on the second method in this paper.

In [Chang 2011], for an $N \times N$ NoC system, a spare row of routers is added for defect-tolerance. Every router within each column shares the common spare router. If there is one defective router in a column, shifting is required to repair it with the spare. The redundancy level in terms of the ratio between the number of working routers and the number of spares is N:1 for this technique. Therefore, it is denoted as a shifting scheme, or "N:1" in the following sections. This architecture can tolerate one failure in each column in the grid. If a column contains more than one fault, spares in other columns cannot be utilized to repair the faults, therefore rendering a low repair rate. In [Kang et al. 2010], a repair strategy using two spares in one group is presented. The redundancy level of this technique is N:2. It is thus denoted as a switching scheme, or "N:2". It can tolerate two failures within one group. These two approaches are straightforward to implement, and topology reconfiguration is completed in a linear time. However, as to the repair rate, there is room for improvement.

In [Jiang et al. 2012], 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 in a TSV grid, so the repair of TSVs is similar to the problem of repairing faulty PEs. Thus this approach is applicable to a manycore system.

Simulated Annealing (SA) is a commonly used technique for optimization [Misevicius 2003, Wang 2012]. Since topology reconfiguration is an optimization problem, the SA algorithm can be utilized to solve it. Inspirited by the annealing process in metallurgy, SA solves the combinatorial optimization problem using a probabilistic local search technique. SA starts from an initial solution and successively moves the current solution. It is theoretically guaranteed to converge to the global optimum, so the optimal topology will be obtained for reconfiguration. However, it is often very time-consuming since many random solutions have to be

explored to achieve a satisfactory result. Since configuration time is an important evaluation metric of chip cost, SA is not suitable to be used for large scale manycore systems.

Previous research has made great contributions on improving the reliability of manycore systems [Vitkovskiy et al. 2012, Radetzki et al. 2013]. However, in terms of repair rate, previous studies did not achieve a satisfactory result. Particularly, they did not take comprehensive consideration of repair rate, reconfiguration time and performance. Therefore, this paper tries to meet the above design considerations and improve reliability while maintaining performance.

3. PROPOSED TOPOLOGY RECONFIGURATION APPROACH

In this section, the reconfiguration from physical to virtual topology is first introduced. Then two reconfiguration algorithms are detailed. The first one is the maximum flow (MF) approach. The data transmission between PEs is assumed to be identical. This approach does not change the underlying topology of the system. Besides, the overhead of reconfiguration time, throughput and latency is analyzed. The assumption that the data transmission is distributed uniformly may not work for a practical situation. So in the third part, an improved minimum-cost MF approach is introduced. Cost is added to the nodes in the graph to model the differences between PEs.

3.1 Topology Reconfiguration

Faulty PEs change the target design and the topology is reconfigured by mapping from various physical topologies to a unified virtual topology. In this paper, we focus on mesh, which is the most widely used topology in NoC systems. The reconfiguration controller is implemented by an ARM 7 processor. The connection between the controller and the NoC is shown in Fig. 3 [Rossi et al. 2010]. The controller is connected to the routers in the NoC via AHB (Advanced High performance Bus) [AMBA open specifications. 2014] and AHB to NoC Interface. The interface allows the AHB standard on the one side and the NoC standard on the other side. Fig. 4 shows a block diagram of a router [Scott and Thorson 1996]. The router contains a PE interface (i.e. interface between router and PE or router and AHB2NoC IF), four input/output ports and a crossbar. It also has a look-up table and two registers for storing its physical and virtual addresses. The PE interface performs the routing table lookup at the source node. Routing of data packets is accomplished by comparison of routing tags attached to the packets. A physical and a logical “who am I” registers are loaded on the router. The look-up table provides a mapping from the physical topology to a virtual topology. The index into the look-up table is the virtual address, while the entry in the table is the physical address. The failure information can be obtained through various testing strategies. IEEE P1500 [IEEE P1500] is a standardized core-based test approach in which cores are wrapped in a test wrapper. A test access mechanism (TAM) [Yuan et al. 2008] is usually used in combination with test wrappers to transport test data. Built-in self-test (BIST) [Wang 2007] is another popular approach to test regular logic blocks. No matter what kind of testing approach is adopted, when faults are captured, this information is sent to the controller. The reconfiguration procedure is activated according to the demands, e.g. once after power up, periodically, or at some specific time points at run-time programmed in advance. The controller calculates the virtual topology using the proposed algorithms (to be introduced in the next section) according to the failure information. Then the look-up table together with the virtual address register is

logically renamed. They are addressed as memory does. The refreshed data is sent from ARM via AHB bus and AHB2NoC Interface. OS works on the virtual topology, while routing of data packets is accomplished by comparing the physical addresses between the destination and current nodes. Fig. 5 shows an example of the relationship between a virtual topology and the mapping table. A packet sent from the virtual address #VI to #XVI is actually sent from the physical address of 12 to 20. If XY routing is used with X-axis first, the routing of the packet will be three hops to the right first and then one hop downward.

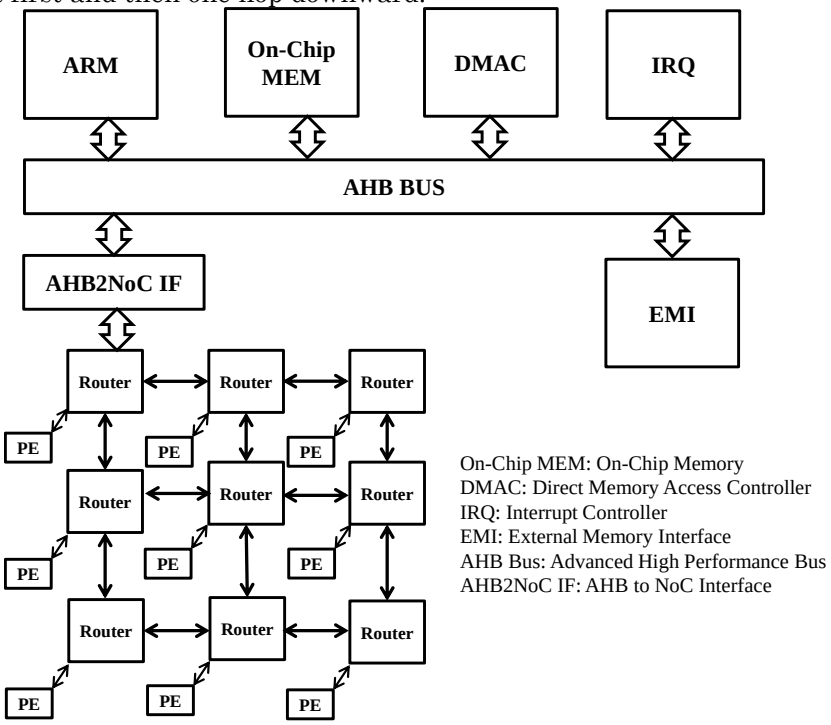


Fig. 3.The connection between the ARM controller and NoC.

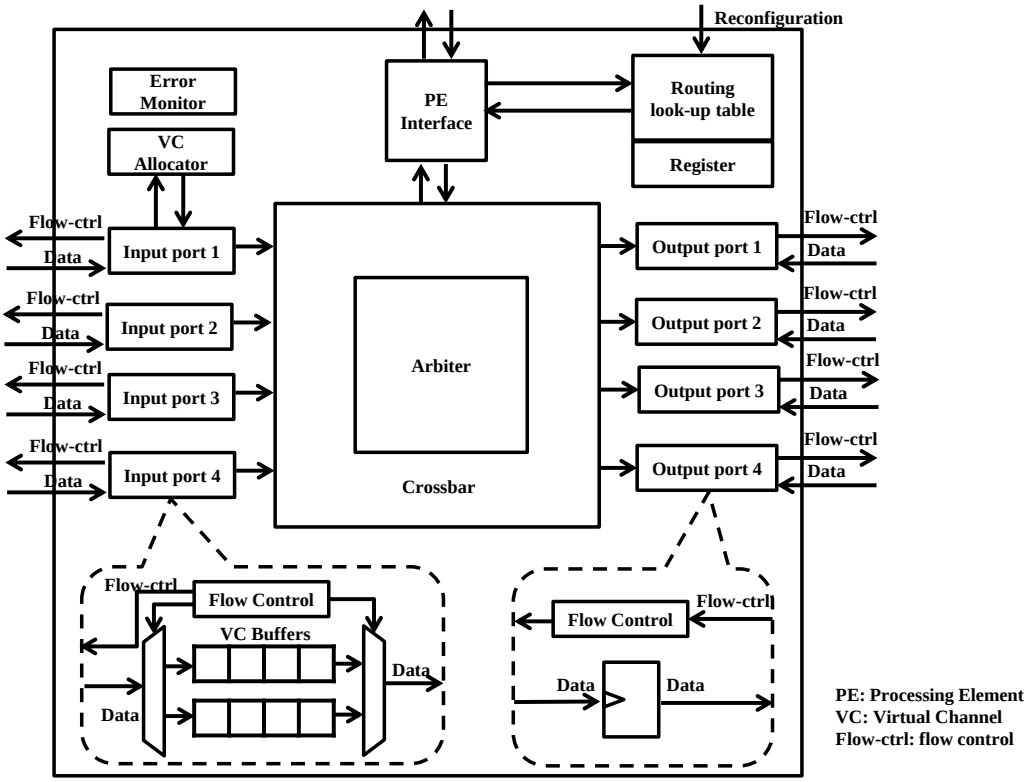
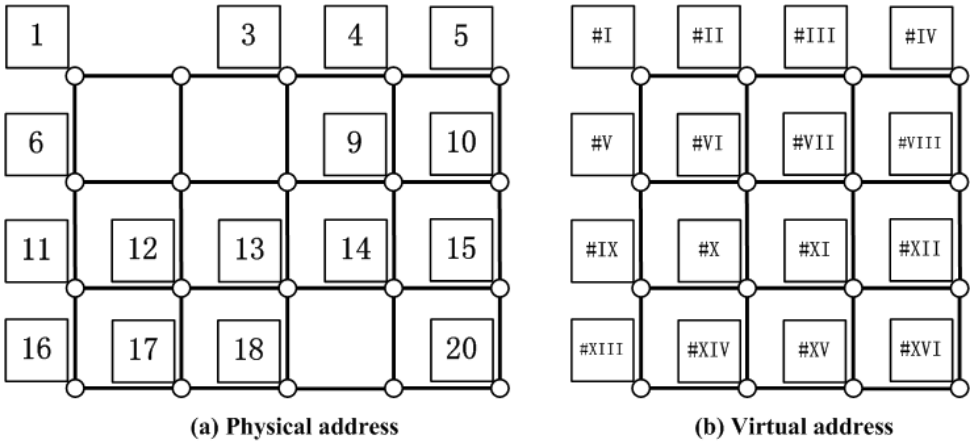


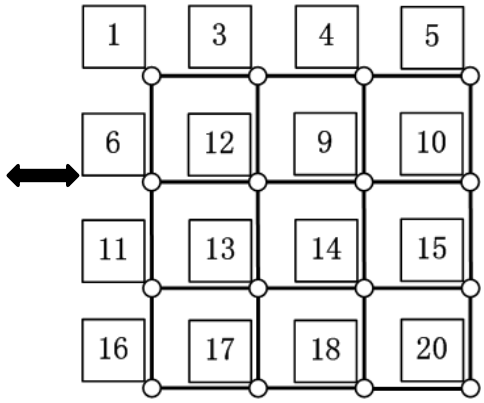
Fig. 4. Router block diagram.



(a) Physical address (b) Virtual address

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



(c) Virtual topology

Fig. 5.Example of the relationship between a virtual topology and the mapping table.

3.2 Maximum Flow (MF) Approach

3.2.1 Proposed Approach

Initially, the data transmission between PEs is assumed to be uniformly distributed. Assume a non-spare PE at location *(x, y)* 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. When the PE at location *(x, y)* is replaced by the PE at location *(x', y')*,the PE at location *(x', y')* goes on to 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. Inspired by the compensation path problems in [Varvarigou et al. 1993], a general methodology to reconfigure a mesh with faulty PEs is equivalent to determining the repair paths. When the repair paths are determined, the neighbors of each PE is determined, thus a virtual topology is obtained. Fig. 6 shows an example to illustrate the concept of a repair path and the reconfigured virtual topology.

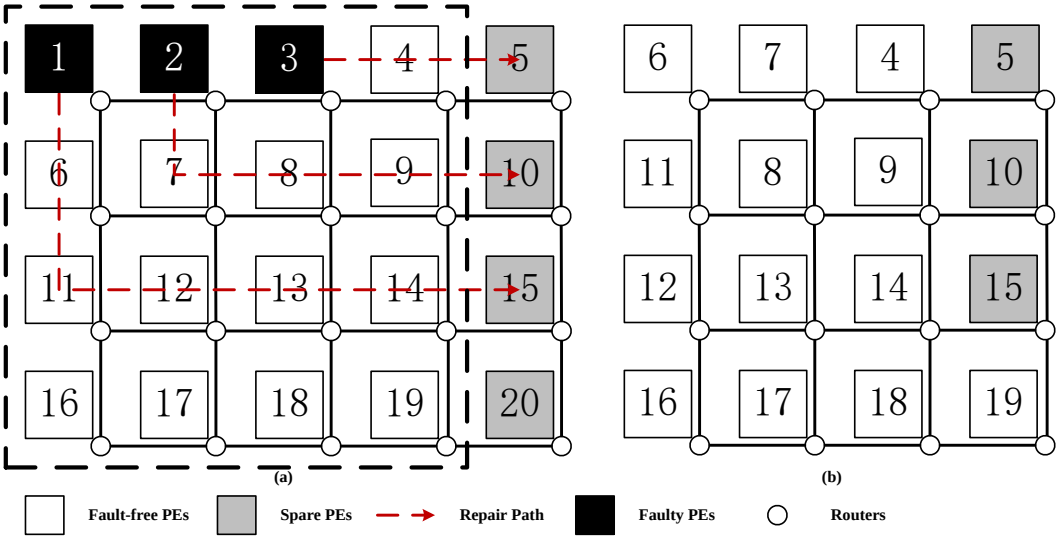


Fig. 6. (a) A 4×4 mesh with repair paths. (b) The reconfigured virtual topology.

In this figure, three faults are clustered on the top and they are connected to spare PEs by three disjoint repair paths. PE 3 is faulty and it is replaced by PE 4, which is in turn replaced by the spare PE 5. Packets that are sent to PE 3 will be sent to PE 4 instead. For example, if packets are sent from PE 9 (#VIII) to PE 1 (#I) in the original topology, the routing path is 9-8-7-6-1. In the reconfigured topology, because PE 9 is replaced by PE 10 and PE 1 is replaced by PE 6, packets are instead sent from PE 10 to PE 6. The routing path is therefore 10-9-8-7-6. Note that there is a difference between the repair path and the routing path. The repair path is virtual for 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. Each PE is assigned a new address in the virtual topology. The mapping is implemented by a look-up table. The reconfiguration controller calculates the repair paths and then assigns new indices to the PEs.

If faulty PEs are detected, a repair path must start from a faulty PE and end at a spare one, in order to repair the faulty PE. Because PE is assumed to be replaced by physical neighbors, each PE along the repair path must be physically next to each other, that is, a repair path is continuous. 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.

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

Next, a repair algorithm referred to as MF is proposed to analyze whether a mesh is repairable and if yes, how to generate a repair path set. If all faults can be repaired, MF returns a repair path set; if such a set does not exist, MF returns a set that covers the maximum possible number of faulty PEs. This problem of determining a set of non-intersecting continuous repair paths can be converted into an MF problem. It then becomes a classical combinatorial optimization problem, that is, 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 MF is stated as follows (see Fig. 7).

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. Because 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. Hence, the NoC system topology reconfiguration is converted into a maximum flow problem in graph theory.

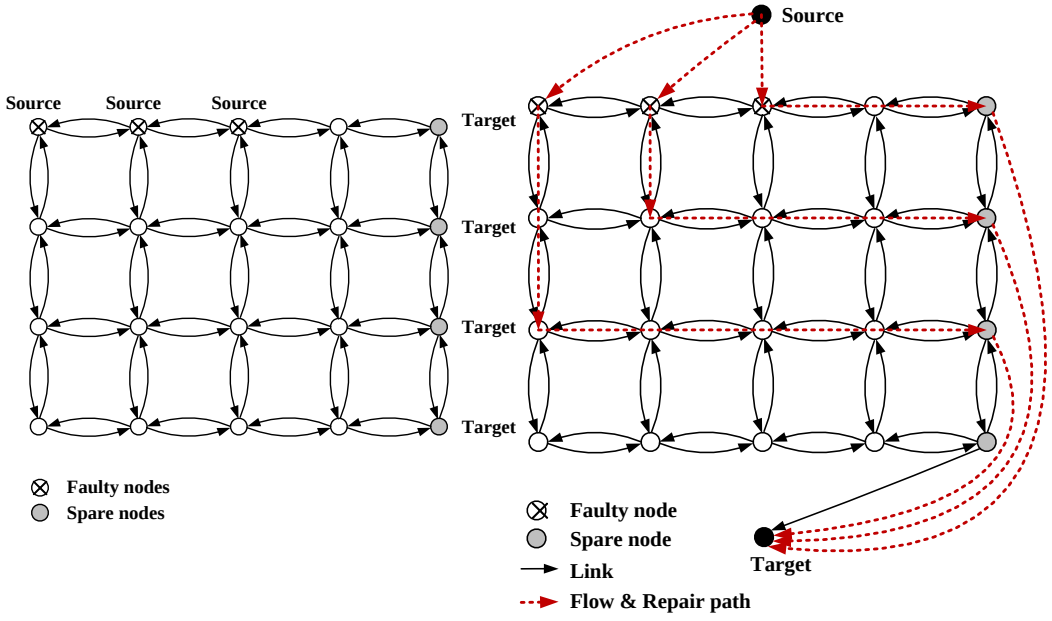


Fig. 7. Maximum flow algorithm for determining the repair paths. (a) Multiple-source multiple-target network. (b) Single-source single-target network with flow and repair path.

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. The problem can be described mathematically as follows.

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 between the nodes of V' is defined as follows:
   1) For every pair of nodes *(i, j)* that are adjacent in the grid, define two edges $i \rightarrow j$ and $j \rightarrow i$;
   2) For every spare node $v_s \in V$, define an edge $v_s \rightarrow T$;
   3) For every faulty node $v_f \in F$, define an edge $S \rightarrow v_f$.
3. Define the capacity of every edge to be 1.
4. Define the capacity of every node to be 1.
5. Solve the 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. 6 (b). On the other hand, if the maximum flow isn't equal to the number of faulty PEs, some faults are not repaired.

3.2.2 Reconfiguration Time Analysis

The feature of MF to find 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 maximum flow problem has polynomial-time solutions. In the general case, the time is bounded by $O(V^3)$ [Karzanov 1974], where V is the number of nodes. For spase graphs with integer edge capacities of moderate size, the time bound is improved to be $O(VE\log U)$, where E is the number of edges and U is the upper bound on the edge capacity [Edmonds and Karp 1972]. As to SA, it has a non-deterministic run time, but it cannot be optimally solved in polynomial time [Karzanov 1974]. The time becomes significantly long when the problem size grows very large. Because the proposed algorithm has polynomial-time solutions, it can be performed once after fabrication, or periodically at run-time, without introducing significant overhead in reconfiguration time.

3.2.3 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 [Zhang et al. 2009]. Obviously, the average hop count between nodes in an irregular topology is larger than in the reference topology, so the latency becomes longer. DF 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 k virtual neighbors, as shown in Eq. (1).

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

The DF of a topology is defined as the average DF_n of all the N nodes in the topology, as shown in Eq. (2).

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

It is clear that the reference topology has the minimum DF because virtual neighbors are usually located next to each other physically. For example, 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. In other words, virtual neighbors are located closer to each other physically.

Next, experiments are performed to evaluate the DF. Two repair schemes are used as baseline solutions for comparison, i.e., the shifting [Chang et al. 2011] and

switching schemes [Kang et al. 2010]. They are denoted as "N:1" and "N:2" respectively. Figs. 6 and 7 show examples of the N:1 and N:2 approaches. The shifting 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, so this scheme can tolerate one failure in each row. The switching scheme has two spare PE columns for defect-tolerance, one on the left and one on the right border of the grid. This scheme can tolerate at most two failures in a row. Therefore, there are two types of topologies in the experiments, i.e., *N×(N+1)* and *N×(N+2)* meshes. For a fair comparison, MF is considered to have the same spare resources as the baseline schemes. In the comparison between MF and N:1, both approaches are conducted on an *N×(N+1)* topology. For a comparison between MF and N:2, both approaches are conducted on an *N×(N+2)* topology. Given the same physical topology and fault pattern, each approach may return a different virtual topology, therefore leading to different performances. Fig. 9 (c) and (d) are examples of different virtual topologies obtained by N:2 and MF respectively.

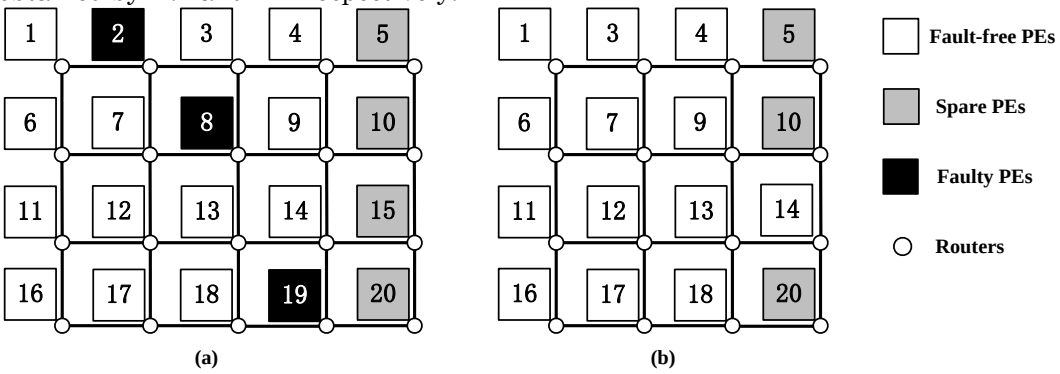


Fig. 8. Examples of N:1. (a) A 4×5 mesh with faults; (b) Reconfigured virtual topology using N:1.

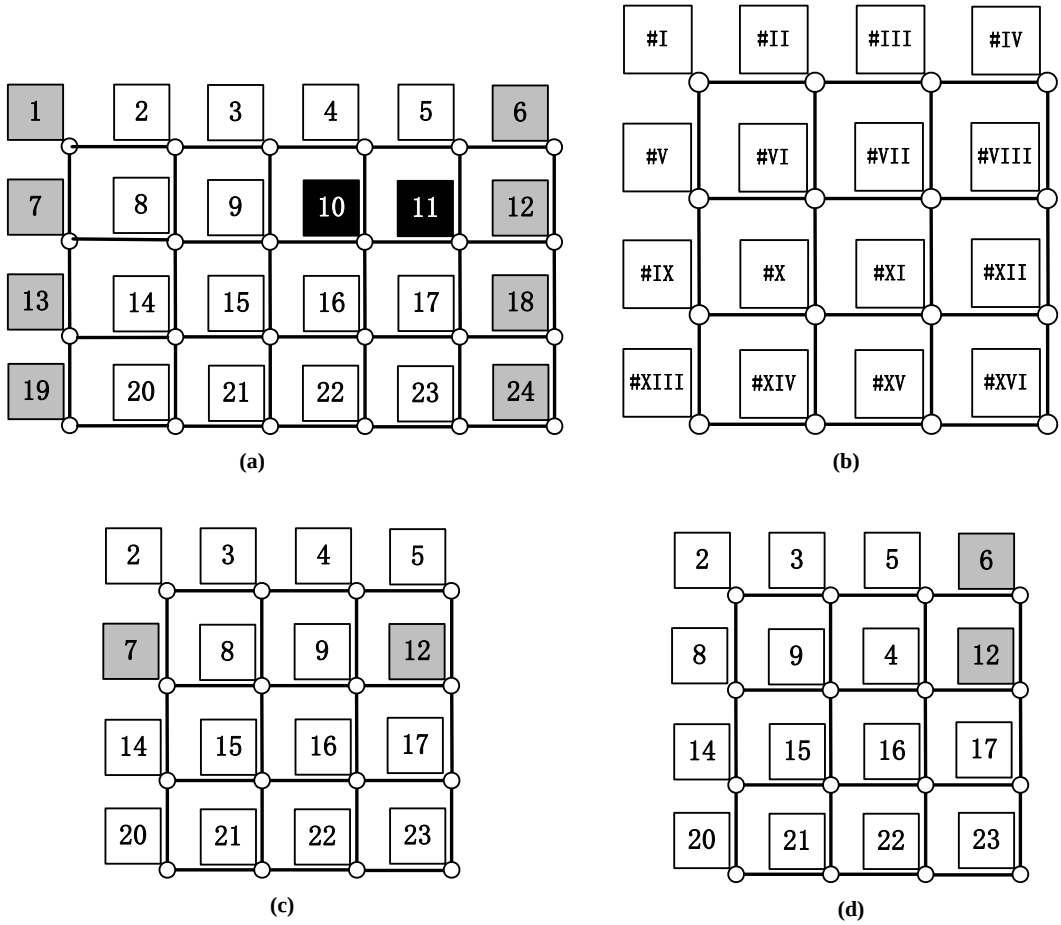


Fig. 9. Examples of N:2. (a) A 4×6 mesh with faults; (b) Virtual address;
(c) Reconfigured virtual topology using N:2; (d) Reconfigured virtual topology using MF.

Two different mesh sizes are simulated, i.e. $N = 4$ and $N = 8$. The number of faulty PEs ranges from 1 to the maximum number of spare PEs. For each mesh size with a number of faults, 3000 random fault patterns are simulated. In some cases, MF can repair all the faults while N:1 or N:2 cannot, such as a row with more than two faults. In this experiment, however, these cases are not considered, because DF cannot be calculated for a network with faults. DF is calculated only when all faults are repaired by both approaches in a network. Fig. 10 shows the DF comparison results. As shown in the figure, DF increases with the increase of failure number. 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, thereby leading to the same DF. In other words, N:1 scheme is a subset of MF. Compared with N:2, MF has a smaller DF. The N:2 approach tries to replace faults using the spare PEs in the same row. It does not take the characteristics of a repair path into consideration. Fig. 9 (c) and (d) shows an example of a faulty 4×6 mesh, which is repaired using N:2 and MF respectively. As shown in Fig. 9 (a), PE 10 is faulty. Using N:2, the repair path can only be 9-8-7. While MF has a better flexibility: the repair path is 4-5-6. The DF for N:2 is 1.3958, while it is 1.2187 for MF. Assume that a packet is sent from #V to #VIII in the virtual topology in Fig. 9(b). In the N:2 approach, it is physically from PE

7 to PE 12 and takes 5 hops. In the MF approach, it is from PE 8 to PE 12 and only takes 4 hops. Based on a breadth-first search, MF manages to find the shortest repair path. This advantage becomes more significant when the mesh size increases. As in the comparison between N:2 and MF, a solution found by N:2 for a given fault pattern may also be found by MF. This indicates that MF find the same or better solutions than N:2. According to the definition of DF, the average hop between virtual neighbors using MF is the same as N:1, and smaller than N:2. Considering that in a practical situation, a system with faults will continue to work and the DF experiments do not include all the cases, thus it is expected that the throughput and latency using MF are better than the baseline schemes. The throughput and latency will be measured by simulation in the following section.

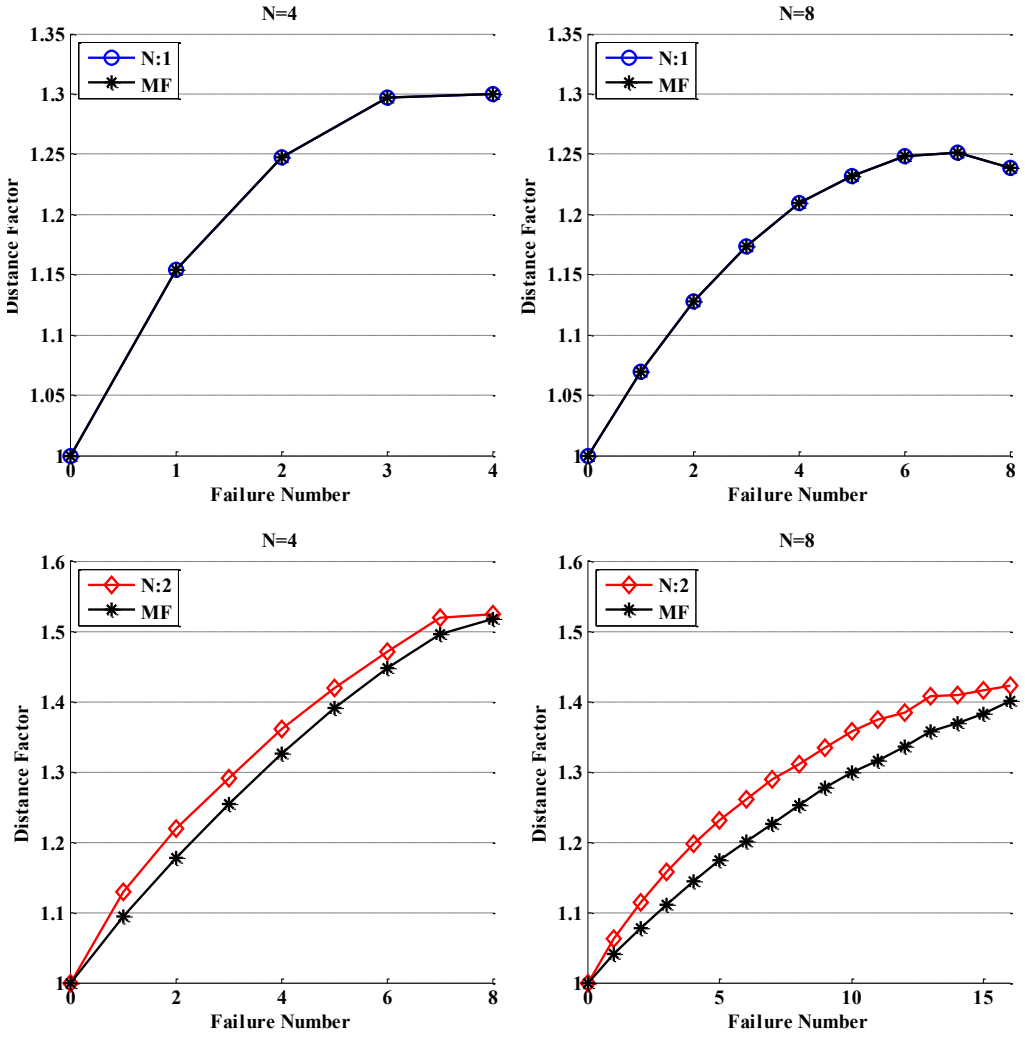


Fig. 10. DF comparison between MF, N:1 and N:2 for different mesh sizes and failure number.

3.3 Minimum-cost MF Approach

In the above discussion, all the elements in the system are considered to be identical. In a practical application, however, the data transmission of PEs and the traffic load on each link may be different. To be more accurate, these variations need to be modeled with different weight. To address this problem, a minimum-cost MF is

proposed to achieve the maximum flow of the directed graph while simultaneously minimizing the cost under predetermined cost constraints.

A directed flow graph is established in a way similar to MF. A super source and a target node are added. The capacity of each node and edge is set to be 1. A new variant, cost, is introduced in the graph. The cost can be defined on a node or an edge. It can be of any metric to model the differences of the new network, e.g., an edge delay, hardware consumption and a financial cost. Then the problem is solved using the minimum cost maximum flow algorithm, which has polynomial-time solutions [Edmonds and Karp 1972].

4. EXPERIMENTAL RESULTS

According to the design consideration, many metrics, such as repair rate, reconfiguration time, topology, area, throughput and latency, have to be evaluated. As mentioned above, MF is based on the fact that the underlying physical topology does not change, therefore alleviating the burden on OS and other programs. The proposed approach requires the same area and topology as the comparative approach, as shown in the following experiments. The theoretical analysis of reconfiguration time has been given above, so next, simulations are performed on the repair rate, reconfiguration time, throughput and latency. To model the practical applications, experiments on minimum-cost MF are also conducted.

4.1 Experimental Setup

The baseline schemes are the same as mentioned in the experimental setup in DF calculation, i.e. N:1 and N:2. Two different mesh sizes are simulated, i.e. $N = 4$ and $N = 8$. The repair rate is obtained from simulation using Matlab. Next, throughput and latency are measured in a C++ NoC platform using Carbon SoC Designer. A cycle-accurate modeling technique is used in the simulator design. It is based on a two-phase iteration technique. During the first phase, i.e. the communication phase, the simulator performs the communication between PEs. During the second phase, i.e. the update phase, the resources within the internal component are updated. Wormhole switching is used as the switching technique of the router. The packet length is set to be 16 flits, including one header flit. Each input port has three virtual channels and each channel has a FIFO buffer to store four flits. Each PE can inject packets independently. The destination of a packet is randomly determined, resulting in a uniform traffic pattern. The XY routing is used for this network, 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, which is stored in the routers. Thus the difference between the NoC models lies in the look-up tables, which are imported into the simulator when generated by the repair schemes. The other experimental settings are the same for different models.

4.2 Repair Rate

As mentioned before, repair rate is defined as the probability that all the faulty PEs in a topology can be successfully repaired by spare ones. To investigate the repair rate, PEs are assumed to work independently. All PEs including the spare ones are subject to failures. In the next experiment, 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 and repaired using the different approaches. Repair rate is the probability that a fault pattern can be fully repaired. Fig. 11 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 with different slopes while MF attains a much higher repair rate. 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. Especially when the number of faults is nearly close the number of spares, the repair rate is less than 15%. In other words, the utilization efficiency of the spare hardware is low. A lot of resources are wasted, while they could have been used to improve the reliability of the system by a more efficient approach. Because the resources on a chip are limited, we focus on a better utilization of the spare resources. The PEs used for repair in MF are not restricted to the faulty row. It is more capable of using spare PEs to repair faults. If a fault pattern can be repaired by N:1 or N:2, it can also be repaired by MF as well, but not vice versa. Even though MF has a much higher repair rate, it does not always achieve a repair rate of 100%. As discussed in previous sections, the set of repair paths has to meet three requirements. MF is used to return a set that covers the maximum possible number of faulty PEs. If a fault pattern cannot be fully repaired, it means that a set of non-intersecting continuous repair paths cannot be obtained to cover all faults within this fault pattern. In these cases, a solution to repair all faults is not found, but MF manages to repair a maximum number of faults. Fig. 12 shows an example when a fault pattern cannot be fully repaired. There are six faulty PEs and six spare PEs. One of the spares is faulty. In this pattern, only three faults can be repaired. The repair paths are shown in Fig. 12.

It can also be observed that the repair rate of MF in an 8×9 mesh is higher than that of N:2 in a 4×6 mesh. 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 MF 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.

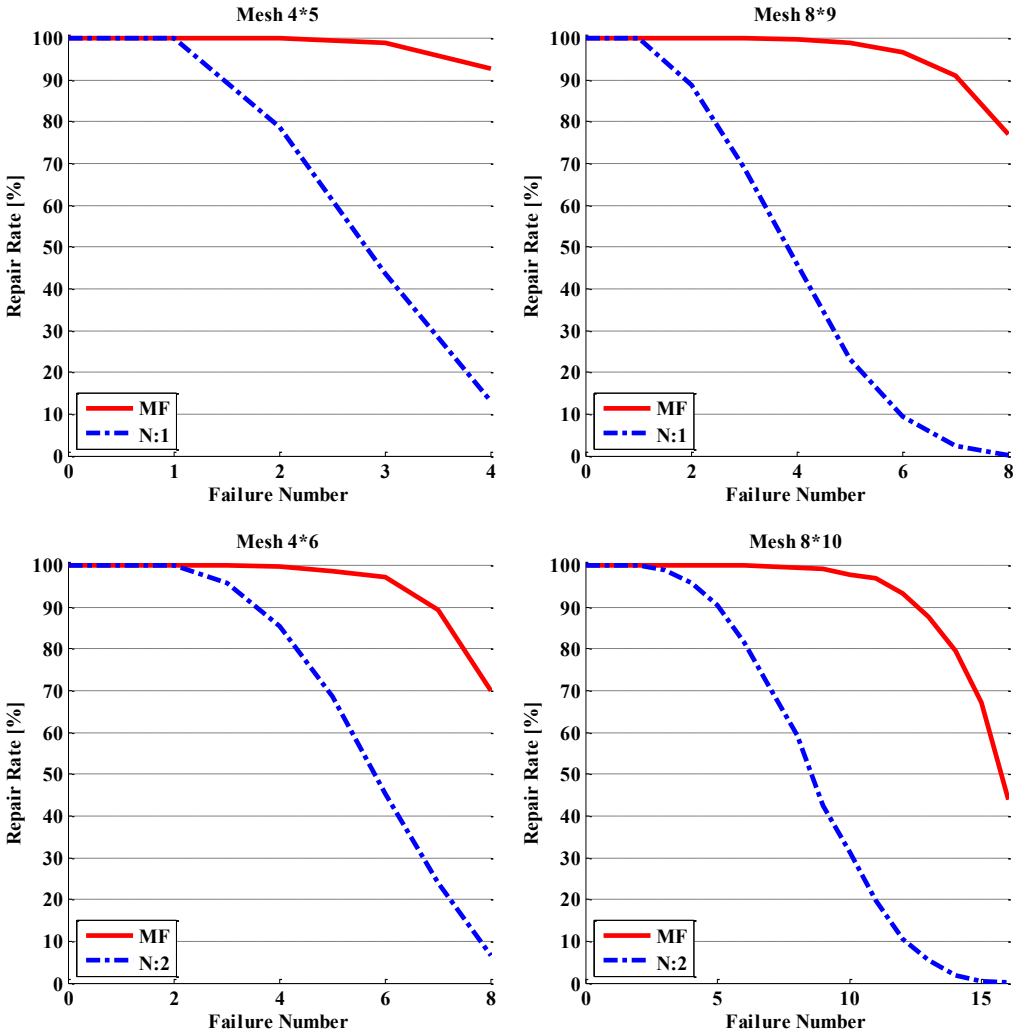


Fig. 11. Repair rate comparison for different mesh sizes and failure number.

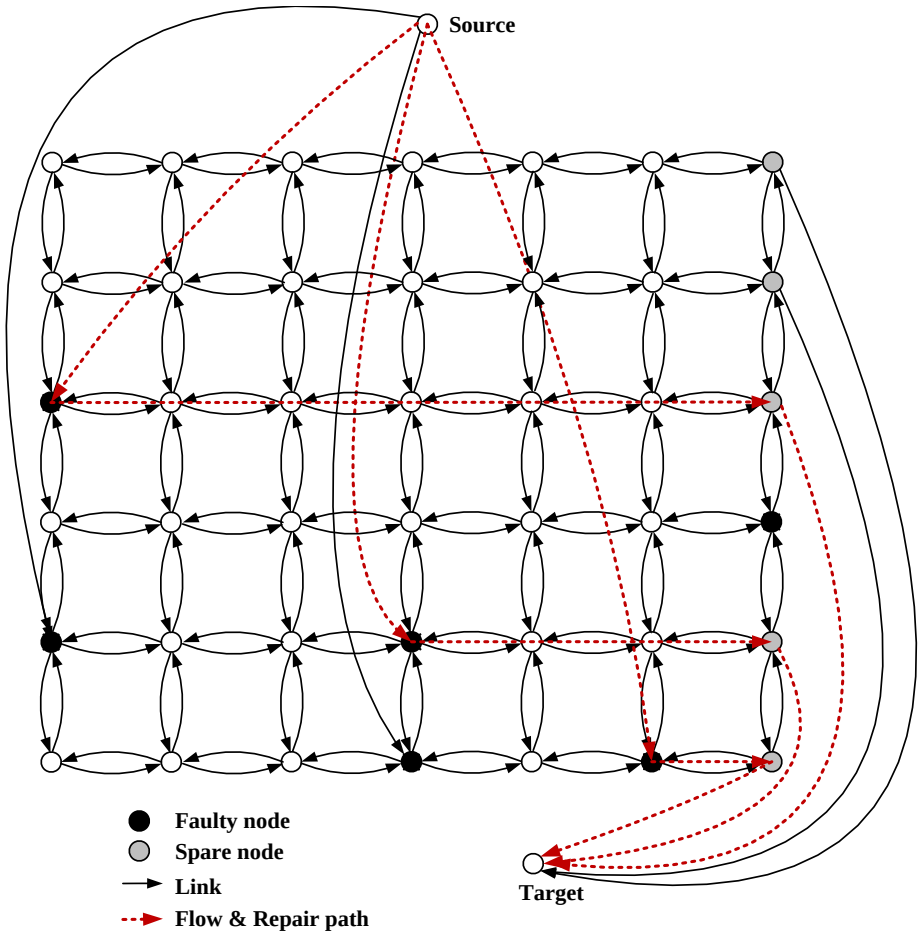


Fig. 12. An example that a fault pattern cannot be fully repaired.

4.3 Reconfiguration Time and Hardware Overhead

As discussed in Section 3.1.2, the proposed algorithm has a time complexity of $O(V^3)$. Since the time complexity is not very high, there is no special requirement on the computing capability of the reconfiguration controller. The reconfiguration controller can be implemented by any component with basic computing capability, such as GPP (general purpose processor), ASIP (application-specific instruction-set processors), or ASIC (application-specific integrated circuit). In this paper, the reconfiguration controller is implemented by an ARM 7 processor, which is a relatively low-performance embedded processor launched by ARM Holdings in 1994. It uses approximately 20,000 gates. Reconfiguration time of the three repair methods is measured in this section. There are many algorithms for maximum flow problem. An algorithm in [Karzanov 1974] is used in this experiment due to its simplicity in implementation. 30,000 random fault patterns are generated and the repair techniques are applied. The average clock cycles of 30,000 cases are shown in Table I. The reconfiguration time is shown in Table II for a clock frequency of 200MHz.It can be seen that the reconfiguration time of MF is longer than N:1 and N:2. The difference is not significant for small meshes. However, for large meshes the computation time of MF increases significantly while for N:1 and N:2 it does not change so much. Because the baseline approaches are simple, the complexity is

simple as well. Apart from the execution of the algorithms, it will also take some time to reconfigure the look-up tables and the virtual-address registers. Assume $N \times M$ PEs are implemented and an $N \times N$ virtual topology is obtained. The physical address is $\lceil \log_2 NM \rceil$ bits and the size of the look-up table is $(N \times N) \times \lceil \log_2 NM \rceil$ bits. Taking the virtual-address register into consideration, $(N \times N \times \lceil \log_2 NM \rceil + \lceil \log_2 NM \rceil) \times (N \times N)$ bits have to be refreshed in total in the worst case. Therefore the refresh time in terms of clock cycles could be expressed as Eq. (3) and Eq. (4).

$$AHB\ transmission\ time = (N \times N \times \lceil \log_2 NM \rceil + \lceil \log_2 NM \rceil) \times (N \times N) / (Buswidth \times \alpha) \quad (3)$$

$$NoC\ transmission\ time = (N \times N \times \lceil \log_2 NM \rceil + \lceil \log_2 NM \rceil) \times \beta \times (N + M) / (NoC\ bandwidth) \quad (4)$$

The first equation stands for the transmission time on AHB bus and the second equation stands for the maximum transmission time in the NoC. *Bus-width* is the width of AHB bus, ranging between 8 and 1024 bits [AMBA open specifications. 2014], and α is a parameter ranging between 0.5 and 1, dependent upon the transmission type. β is the routing latency in a router, which is about 2~4 clock cycles, and *(M+N)* is the number of nodes in the longest routing path in an NoC. The size of an NoC depends on the number of PEs. So far thousands of PEs have been integrated on a single chip [Mahanta et al. 2014], in which *N* is about 32. In practical applications, *N* usually ranges between 4 and 10. In this paper, it is assumed that *N*=8, *M*=9, *Bus-width*=32bits, α=16/18 (the transmission type of AHB bus is selected to be Burst 16), and β=3, *NoC bandwidth*=32bits. In this case, *AHB transmission time* is 1023.75 clock cycles, and *NoC transmission time* is 725.16 clock cycles. The transmission in NoC could be in parallel with AHB transmission, so the refresh time is the longer one, i.e. 1023.75 clock cycles. Under 200MHz working frequency, it is 5.12us. The average execution time of the reconfiguration algorithm for 8×9 mesh using N:1 is 218.2us, so the refresh time is only 2.35% of the algorithm execution time. If a more complicated reconfiguration approach is used, the proportion would be smaller. Furthermore, in practical situations not all the routers need refreshing, and even in one router not all the values in the look-up tables and virtual-address registers need refreshing. Partial reconfiguration is needed under these circumstances. The refreshing time could therefore be reduced by up to 80%. Besides, the width of AHB bus could be extended to 1024 bits, so the AHB transmission time in Eq. (3) could be further reduced. The bandwidth of NoC could be extended to match the AHB bus-width accordingly, leading to a reduction in the NoC transmission time.

Table I. Average clock cycles for different approaches

Mesh Size	MF	N:1	Mesh Size	MF	N:2
4×5	80779	40665	4×6	78262	41949
8×9	1399843	43641	8×10	1137092	48726

Table II. Average reconfiguration time for different approaches at 200MHz

Mesh Size	MF(us)	N:1(us)	Mesh Size	MF(us)	N:2(us)
4×5	403.9	203.3	4×6	391.3	209.7
8×9	6999	218.2	8×10	5686	243.6

4.4 Throughput and Latency Overhead

The distance factor (DF) is used to describe the average hop count between virtual neighbors. DF reflects the average delay and throughput of a network. It is predicted in Section 3.2.3 that the throughput and latency using MF is better than the baseline schemes. Next the throughput and latency are measured by simulation. In the first experiment, MF and N:1 are implemented on an 8×8 mesh, with 8 spare PEs in a column on the right border, similar to Fig. 1 (b). 8 out of the 72 PEs are randomly chosen to be faulty and 100 fault patterns are randomly generated. A mesh with no faults is also simulated to be the baseline. Different approaches are performed on these patterns. The average latency and throughput is shown in Fig. 13 (a) and (b). In the second experiment, an 8×8 mesh is implemented with one column of spare PEs on the left and right borders. There are 80 PEs in this topology, and 10 out of them are randomly chosen to be faulty. MF and N:2 are implemented on this topology. The comparison results are shown in Fig. 13 (c) and (d).

It can be seen from Fig. 13 (a) and (c) that the average latency increases with the traffic and the network becomes saturated when the traffic is large enough. The latency of MF is smaller than the other two approaches. As discussed in the previous section, when all the test fault patterns can be repaired by N:1 and MF, MF has the same DF as N:1. Hence, DF is measured under the circumstance that a fault pattern can be repaired by both approaches. Because for a fault pattern that cannot be repaired by N:1, the virtual topology cannot be obtained and the DF cannot be calculated. In practice, however, there are many cases that some faults cannot be fully corrected by N:1. The remaining cores will still work, but with a degraded performance. These cases are considered in the throughput and latency simulations. So the latency of MF is smaller than N:1, as shown in Fig. 13 (a). Compared with the non-faulty topology, the latency of MF is increased by less than 4%. The latency of MF is 4.5% smaller than N:1 and 5.3% smaller than N:2.

It is also observed that the throughput of MF is higher than the other two baseline approaches. Compared with the non-faulty topology, the throughput of MF is decreased by less than 2.5%. It is 11.3% higher than N:1 and 6.3% higher than N:2. Hence, the performance overhead introduced by MF is not significant.

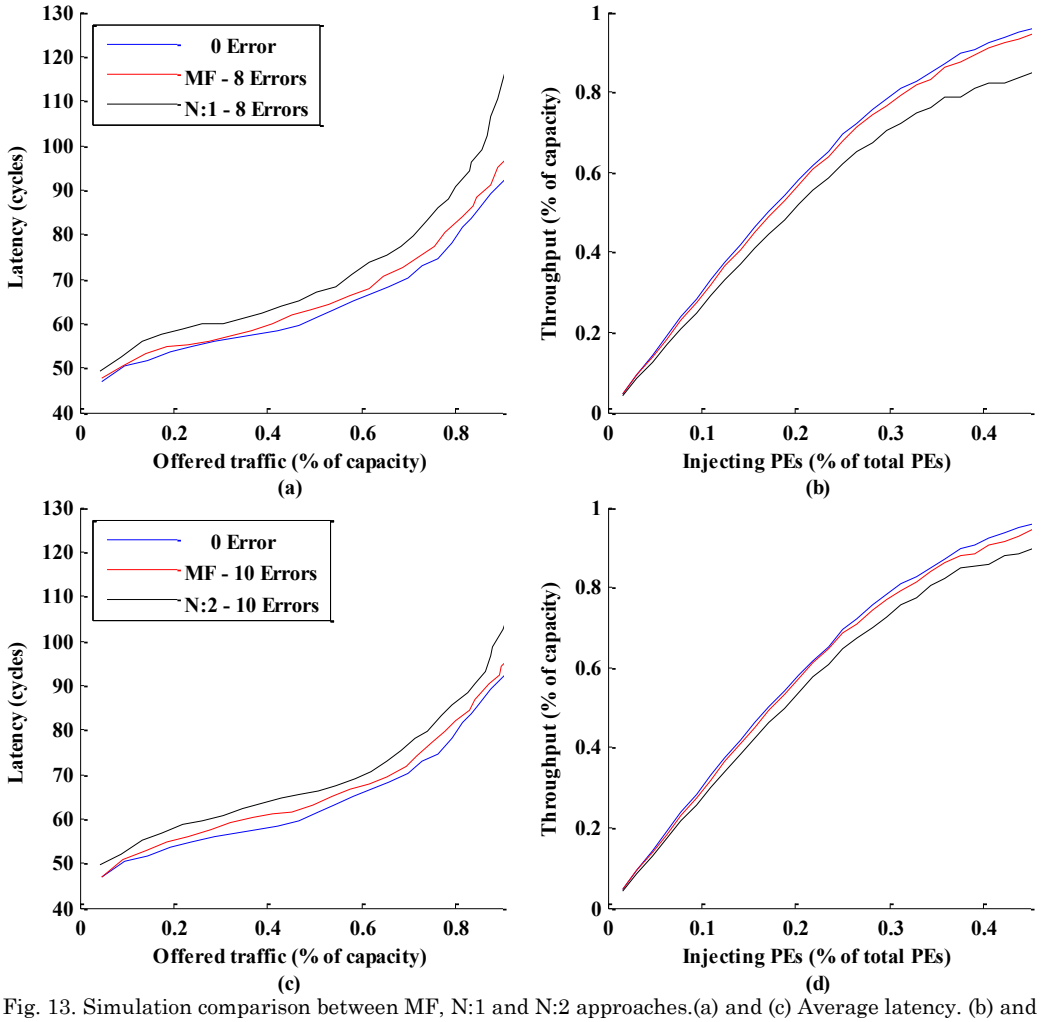


Fig. 13. Simulation comparison between MF, N:1 and N:2 approaches.(a) and (c) Average latency. (b) and (d) Throughput.

4.5 Performance of Minimum-cost MF

A real-time H. 264 video-decoding application [Sair and Kim 2005] is taken as an example to illustrate minimum-cost MF, which is implemented on a 4×4 NoC with one column of spare routers. The communication cost is an important factor in this application, so the cost is defined as the volume of transmitted data on each PE. An H.264 decoder is mainly composed of IDCT-IQ (Inverse Discrete Cosine Transform – Inverse Quantization), MC (Motion Compensation) and Deblocking blocks. The diagram of H.264 is shown in Fig. 14. The proportion of data transmission is calculated, as shown in Table III.

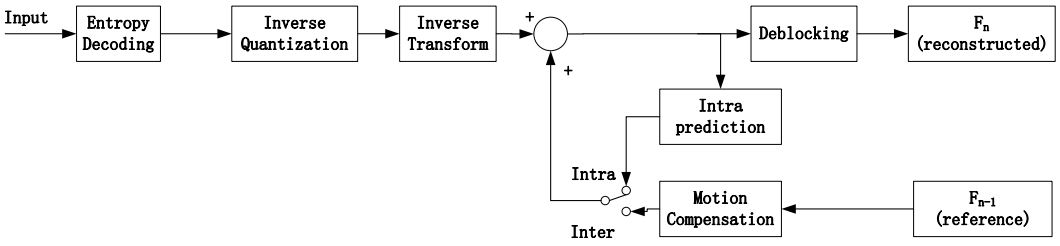


Fig. 14. Decoder diagram of H.264

Table III. Proportion of data transmission of each block in an H.264 decoder

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

The throughput and latency of this application are simulated for one example. Each block is randomly mapped onto one PE, and three faulty PEs are chosen. Fig. 15 (a) shows an example of the mapping of each block. The cost is defined as the proportion of data transmission, as shown in the figure. A weighted graph is obtained by including a super source and a super target. The following experiment is based on this topology. The minimum-cost MF algorithm is performed and the obtained virtual topology is shown in Fig. 15 (b). The functionality of each PE is performed by the corresponding PE in the virtual topology. Reconfiguration is performed by refreshing the look-up tables and the virtual-address registers in the routers. When reconfiguration is completed, a unified virtual topology is obtained for the OS to work on.

The performance measurement of the minimum-cost MF is the same as MF. Because MF is a special case of minimum-cost MF with the cost to be zero, the minimum-cost MF has the same repair rate as MF if the same hardware resources are used. Next, the throughput and latency are measured in the NoC platform for different injection rates. For an application in a stable state, the data transmits between PEs according to the diagram and the offered traffic is not varied. However, in this experiment, a general faulty scenario is simulated, i.e. the number of injected PEs is varied. When the application starts, or there is traffic block in the system, it is possible that not all the PEs are working. So it is reasonable to measure the performance under different injection-rate. The throughput and latency simulation results are shown in Fig. 16. Compared with the fault-free topology, the latency of MF is increased by less than 10% and it is 3.7% smaller than N:1. Compared with the fault-free topology, the throughput of MF is decreased by less than 5.3%, while 6.6% higher than N:1. As shown in these results, the minimum-cost MF works efficiently for practical applications.

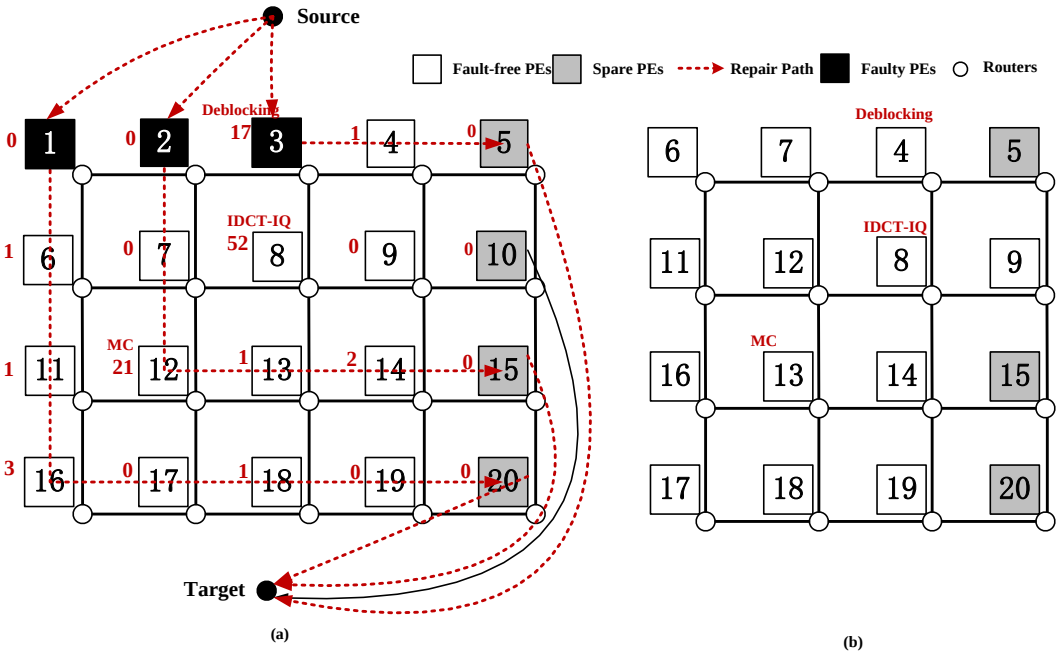


Fig. 15. (a) An example for H. 264 video-decoding application. (b) Reconfigured virtual topology.

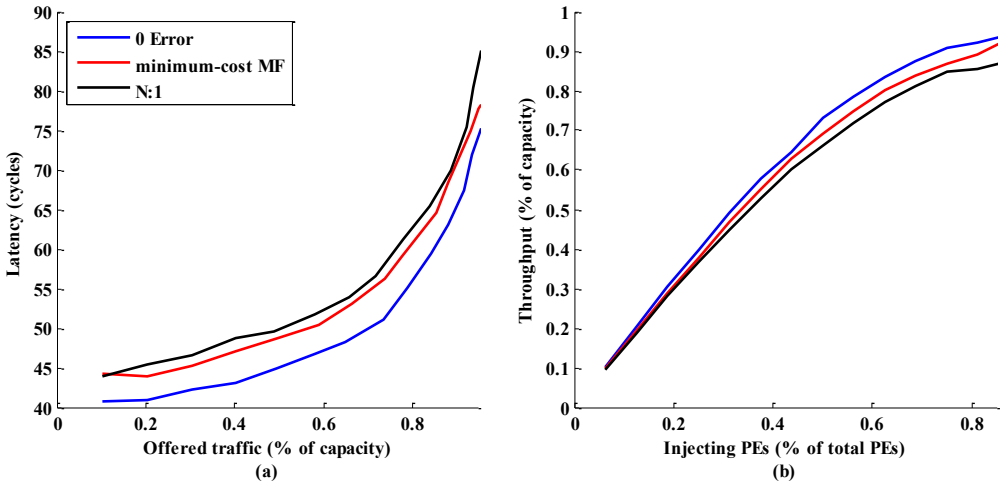


Fig. 16. Latency and throughput comparison between minimum-cost MF and N:1 approaches.

5. CONCLUSION

Effective fault-tolerant techniques are critical to ensure the reliability of integrated circuits. In this paper, an efficient approach using a maximum flow algorithm is proposed for topology reconfiguration to improve the fault-tolerance of reconfigurable architectures. An enhanced approach using a minimum-cost MF algorithm is further presented by considering various PEs in practical applications. These approaches are shown to be efficient to improve the reliability of both homogeneous and heterogeneous reconfigurable architectures. Experiment results show that the MF approaches maximize the use of redundant resources and achieve higher reliability than previous techniques. It also achieves a polynomial reconfiguration time, higher

throughput and shorter delay than other approaches with the same topology. Besides, it provides a unified topology for the OS and other programs.

The reliability of reconfigurable architectures is improved by converting topology reconfiguration into a network flow problem in graph theory. The idea of using maximum flow to maximize the utilization of redundant resources is not restricted to the 2D-mesh topology. It can also be used with other types of topologies and the locations of faulty and spare PEs can be arbitrary. The basic idea is the same, i.e. merging faults into a super source and merging spares into a super target to construct a graph for finding the maximum flow. Our current work is investigating a 3D topology. Hence, the proposed fault tolerant topology reconfiguration is expected to be useful in future reconfigurable designs.

REFERENCES

AMBA Open Specifications. 2014. [Online].http://www.arm.com/products/system-ip/amba/amba-open-specifications.php

ShekharBorkar. 2007. Thousand core chips-A technology perspective. In *Proceedings of the 44th annual Design Automation Conference (DAC '07)*. ACM, NewYork, NY, 746-749. DOI:http://dx.doi.org/10.1145/1278480.1278667

Yung-Chang Chang, Ching-Te Chiu, Shih-Yin Lin and Chung-Kai Liu. 2011. On the design and analysis of fault tolerant NoC architecture using spare routers. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference (ASPDAC '11*), IEEE, NJ, 431-436. DOI:http://dx.doi.org/10.1109/ASPDAC.2011.5722228

Cristian Constantinescu. 2003. Trends and challenges in VLSI circuit reliability. In *IEEE Micro* 23, 4 (July 2003), 14-19. DOI:http://dx.doi.org/10.1109/MM.2003.1225959

William J. Dally and Brain Towles. 2001. Route packets, not wires: on-chip interconnection networks. In *Proceedings of the 38th annual Design Automation Conference (DAC '01)*, ACM, New York, NY, 684-689. DOI:http://dx.doi.org/10.1145/378239.379048

OnurDerin, DenizKabakci, and Leandro Fiorin. 2011. Online task remapping strategies for fault-tolerant Network-on-Chip multiprocessors. In *Proceedings of 5th ACM/IEEE International Symposium on Networks-on-Chip (NOCS '11)*, 129-136. DOI:http://dx.doi.org/10.1145/1999946.1999967

MasoumehEbrahimi, MasoudDaneshtalab, FahimehFarahnakian, JuhaPlosila, PasiLiljeberg, Maurizio Palesi, and HannuTenhunen. 2012. HARAQ congestion-aware learning model for highly adaptive routing algorithm in on-chip networks. In *Proceedings of 6th IEEE/ACM International Symposium on Networks-on-Chips (NOCS '12*), 19-26. DOI:http://dx.doi.org/10.1109/NOCS.2012.10

Jack Edmonds and Richard M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. In *Journal of the ACM (JACM)* 19, 2 (Apr. 1972), 248-264. DOI:http://dx.doi.org/10.1145/321694.321699

David Fick, Andrew DeOrio, Jin Hu, Valeria Bertacco, David Blaauw and Dennis Sylvester."Vicis: A reliable network for unreliable silicon,"In *Proceedings of the 46th annual Design Automation Conference (DAC '09)*, ACM, New York, NY, 812-817.DOI:http://dx.doi.org/10.1145/1629911.1630119.

AysegulGencata and Biswanath Mukherjee. 2003. Virtual-topology adaptation for WDM mesh networks under dynamic traffic. *J. IEEE/ACM Trans. on Networking (TON)* 11, 2 (Apr. 2003), 236-247. DOI:http://dx.doi.org/10.1109/TNET.2003.810319

Andrew V. Goldberg and Robert E. Tarjan. 1988. A new approach to the maximum-flow problem. *Journal of the ACM (JACM)* 35, 4 (Oct. 1988), 921-940. DOI:http://dx.doi.org/10.1145/48014.61051

Jie Han and Pieter Jonker. 2003. A defect and fault-tolerant architecture for nanocomputers. *Nanotechnology*, 14(2), 224-230. DOI:http://dx.doi.org/10.1088/0957-4484/14/2/324

IEEE P1500: http://grouper.ieee.org/groups/1500/

Yahya Jan and Lech Jóźwiak. 2012. Scalable communication architectures for massively parallel hardware multi-processors. *J. Parallel Distrib. Comput.* 72, 11 (Nov. 2012), 1450-1463. DOI:http://dx.doi.org/10.1016/j.jpdc.2012.01.017

MajidJanidarmian, Vahhab S. Bokharaie, Ahmad Khademzadeh, and MisaghTavanpour. 2010. Sorena: new on chip network topology featuring efficient mapping and simple deadlock free routing algorithm. In *Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology (CIT '10)*, 2290-2299.DOI:http://dx.doi.org/10.1109/CIT.2010.395

Li Jiang, QiangXu and Bill Eklow. 2012. On effective TSV repair for 3D-stacked ICs. *Design, Automation & Test in Europe Conference & Exhibition (DATE '12)*, 793-798. DOI:http://dx.doi.org/10.1109/DATE.2012.6176602

Uksong Kang, Hoe-Ju Chung, SeongmooHeo et al. 2010. 8 Gb 3-D DDR3 DRAM using through-silicon-via technology. In *IEEE Journal of Solid-State Circuits (JSSC 2010)*, 111-119.DOI:http://dx.doi.org/10.1109/JSSC.2009.2034408

HeikkiKariniemi and JariNurmi. 2006. Fault-tolerant XGFT network-on-chip for multi-processor system-on-chip circuits. In *Proceedings of Field Programmable Logic and Applications*, Finland, 2006, 186-190.DOI:http://dx.doi.org/10.1109/FPL.2005.1515723

A. V.Karzanov. 1974. Determining the maximal flow in a network by the approach of pre-flows. *Souviet Mathematics Doklady*15 (1974), 434-437.

Israel Koren and Dhiraj K. Pradhan. 1986. Yield and performance enhancement through redundancy in VLSI and WSI multiprocessor systems. In *Proceedings of the IEEE* 74, 5 (May 1986), 699-711. DOI:http://dx.doi.org/10.1109/PROC.1986.13532

Mincent Lee, Li-Ming Deng and Cheng-Wen Wu. 2011. A memory built-in self-repair scheme based on configurable spares. *IEEE Trans. On CAD of Integrated Circuits and Systems* 30, 6 (June 2011), 919-929. DOI:http://dx.doi.org/10.1109/TCAD.2011.2106812

HridoyJyotiMahanta, AbhijitBiswas, and Md. Anwar Hussai. 2014. Networks on chip: the new trend of on chip interconnection. Fourth international conference on communication systems and network technologies, (Apr. 2014), 1050-1053.DOI:http://dx.doi.org/10.1109/CSNT.2014.214

AlfonsasMisevicius. 2003. A modified simulated annealing algorithm for the quadratic assignment problem. *Informatica* 14, 4 (Dec. 2003), 497-514.

Martin Radetzki, ChaochaoFeng, Xueqian Zhao and Axel Jantsch. 2013. Methods for fault tolerance in networks-on-chip. In *ACM Computing Surveys (CSUR)* 46, 1, article 8 (Oct. 2013). DOI:http://dx.doi.org/10.1145/2522968.2522976

Yu Ren, Leibo Liu, Shouyi Yin, Jie Han, Qinghua Wu and Shaojun Wei. 2013. A fault tolerant NoC architecture using quad-spare mesh topology and dynamic reconfiguration. In *Journal of Systems Architecture* 59, 7 (Aug. 2013), 482-491.DOI:http://dx.doi.org/10.1016/j.sysarc.2013.03.010

Davide Rossi, Fabio Campi, Simone Spolzino, Stefano Pucillo and Roberto Gueerieri. 2010. A heterogeneous digital signal processor for dynamically reconfigurable computing. In *IEEE Journal of Solid-State Circuits (JSSC 2010)*, 45, 8 (Aug. 2010), 1615-1626. DOI:http://dx.doi.org/10.1109/JSSC.2010.2048149.

SuleymanSair and Youngsoo Kim. 2005. Designing real-time H. 264 decoders with dataflow architectures.

In the *3rd IEEE/ACM/IFIP International Conf. on Hardware/Software Codesign and System Synthesis(2005)*, 291-296. DOI:http://dx.doi.org/10.1145/1084834.1084906

Steven L. Scott and Gregory M. Thorson. 1996. The Cray T3E network: adaptive routing in a high performance 3D torus. In *Proc. Hot Interconnects IV Symposium*, (Aug. 1996), 147-156.

William Stallings. 2011. Operating Systems: Internals and Design Principles (7th. Ed.). Prentice Hall.

Mikkel B. Stensgaard and JensSparso. 2008. ReNoC: A Network-on-Chip Architecture with Reconfigurable Topology. In*Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS 2008)*, 55-64.DOI:http://dx.doi.org/10.1109/NOCS.2008.4492725

P. J. Tan, Tung Le, Keng-Hian Ng, Prasad Mantri, James Westfall. 2006. Testing of UltraSPARC T1 Microprocessor and its Challenges. In *IEEE International Test Conference (ITC 2006)*, 1-10.DOI:http://dx.doi.org/10.1109/TEST.2006.297637

Theodora A. Varvarigou, Vwani P. Roychowdhury and Thomas Kailath. 1993. Reconfiguring processor arrays using multiple-track models: The 3-track-1-spare-approach. In *IEEE Trans. on Computers* 42, 11 (Nov. 1993), 1281-1293. DOI:http://dx.doi.org/10.1109/12.247834

ArseniyVitkovskiy, VassosSoteriou, and ChrysostomosNicopoulos. 2012. A dynamically adjusting gracefully degrading link-level fault-tolerant mechanism for NoCs. In I*EEE Trans.On CAD of Integrated Circuits and Systems* 31(8), 1235-1248, 2012. DOI:http://dx.doi.org/10.1109/TCAD.2012.2188801

Jiunn-Chin Wang. 2012. A multistart simulated annealing algorithm for the quadratic assignment problem. In *Proceedings of the 3rd International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA '12)*. 19-23, Sep. 2012. DOI:http://dx.doi.org/10.1109/IBICA.2012.56

Seongmoon Wang. 2007. A BIST TPG for low power dissipation and high fault coverage. In *IEEE Trans. on VLSI Systems (TVLSI)*. 777-789, Jul. 2007. DOI:http://dx.doi.org/10.1109/TVLSI.2007.899234

Feng Yuan, Lin Huang, and QiangXu. 2008. Re-examing the use of network-on-chip as test access mechanism. In *IEEE Design, Automation and Test in Europe (DATE '08)*. 808-811, Mar. 10-14, 2008. DOI:http://dx.doi.org/10.1109/DATE.2008.4484917

Lei Zhang, Yinhe Han, QiangXu, Xiaowei Li and Huawei Li. 2009.On topology reconfiguration for defect-tolerant NoC-based homogeneous manycore systems.*J. IEEE Trans. on Very Large Scale Integration (VLSI) Systems* 17, 9 (Sep. 2009), 1173-1186. DOI:http://dx.doi.org/10.1109/TVLSI.2008.2002108