A Deadlock-Free and Connectivity-Guaranteed Methodology for Achieving Fault-Tolerance in On-Chip Networks

Pengju Ren, *Member, IEEE*, Xiaowei Ren, Sudhanshu Sane, Michel A. Kinsy, *Member, IEEE*, and Nanning Zheng, *Fellow, IEEE*

Abstract—To improve the reliability of on-chip network based systems, we design a deadlock-free routing technique that is more resilient to component failures and guarantees a higher degree of node connectivity. The routing methodology consists of three key steps. First, we determine the *maximal connected subgraph* of the faulty network by checking whether the defective components happen to be the *cut vertices* and *bridges* of the network topology. A precise fault diagnosis mechanism is used to identify partial defective routers. Second, we construct an acyclic channel dependency graph that breaks all cycles and preserves connectivity of the *maximal connected subgraph*. This is done through the cycle-breaking and connectivity guaranteed (CBCG) algorithm. Finally, we introduce a fault-tolerant adaptive routing scheme that can be used with or without virtual channels for network congestion avoidance and high-throughput routing. The simulation results show both the effectiveness and robustness of the proposed approach. For an 8×8 2D-Mesh with 40 percent of link damage, full connectivity and deadlock freedom are still archived without disabling any faultless router in 98.18 percent of the simulations. In a 2D-Torus, the simulation percentage is even higher (99.93 percent). The hardware overhead for supporting the introduced features is minimal. An on-line implementation of CBCG using TSMC 65nm library has only 0.966 and 1.139 percent area overhead for the 8×8 and 16×16 2D-Meshes.

Index Terms—Fault-tolerance, Network-on-chip, Channel Dependency Graph, Reliability, Routing algorithm

1 Introduction

The ongoing miniaturization of semiconductor manufacturing technologies has enabled the integration of hundreds to thousands of processing cores on a single chip [1]. But with each successive node shrink, scaling of system-on-chip (SoCs) becomes more challenging. Gate widths are nearing the molecular scale and the need for more control over dopant distribution and voltage characteristics are running against the fundamental limits of physical laws [2]. Radiation, electromagnetic interference, electrostatic discharge, aging, process variability and dynamic temperature variation are the major causes of failure in MOSFET based circuits [2], [3], [4]. The combination of these factors in the near future will make long-term product reliability extremely difficult in many-core systems. Therefore, in order to maintain connectivity and correct operation, fault-tolerance issues must be taken into account when designing the communication fabric of these systems.

On-chip network (OCN) has emerged as an attractive solution to transmit messages through a distributed system of programmable routers connected by links. It enables a more efficient and flexible communication resources utilization and sharing than traditional point-to-point links and buses. At OCN level there are two main problems: (1) how to efficiently connect the increasing number of on-chip computation and storage resources, and (2) how to effectively manage decreasing transistor reliability. ONC can potentially achieve fault tolerance by providing alternative routes when messages or packets encounter faulty regions. Generally, fault control in OCN is a two-phase process: fault diagnosis and fault containment. For in-operation detection, different schemes like error-correcting-codes (ECC) have been explored. In this work, we use a dedicated *build-in self test (BIST)* module as described by Cota [5] and Kohler et al. [6] to pinpoint the location of faulty components. Our research mainly focuses on fault tolerance itself. It is worth mentioning that in practice a combination of techniques is required to provide a complete protection against different types of faults.

Circular route dependencies cause deadlock and are difficult to detect. They arise from unpredictable fault distribution that leads to irregular network topology and the use of alternative paths to avoid faults. Prohibiting certain turns and applying dedicated escape virtual channels are convenient and powerful approaches for deadlock prevention. However, the use of turn-models may not be adequate or even feasible if certain network connections are removed due to faults and heterogeneous IP blocks. Furthermore, prohibiting turns arbitrarily and

- *P. Ren, X. Ren and N. Zheng are with the Department of Electronic and Information Engineering, Xian Jiaotong University, Xian, Shaanxi 710049, P.R. China. E-mail: {pengjuren, renxiaowei66}@gmail.com, nnzheng@mail.xjtu.edu.cn.*
- *S. Sane and M. A. Kinsy are with the Department of Computer and Information, University of Oregon, Eugene, OR 97403. E-mail: {ssane, mkinsy}@cs.uoregon.edu.*

Manuscript received 24 Apr. 2014; revised 13 Apr. 2015; accepted 16 Apr. 2015. Date of publication 22 Apr. 2015; date of current version 15 Jan. 2016. Recommended for acceptance by E. Antelo.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TC.2015.2425887

independently from the faulty component or condition severely limits routing options. Dynamic routing and multiple virtual channels increase the complexity of routing decisions and router micro-architecture. More importantly, complex routers increase the power consumption and the probability of component failure.

In this paper, we develop a more general turn-prohibited methodology to address fault tolerance in OCNs by exploring cycle-breaking and connectivity guaranteed (CBCG) algorithm. First an adaptive fault-tolerant virtual channel-free routing algorithm is introduced, followed by a proposed heuristic rule to reduce network contention. We provide extensions to a previously developed fine-grained fault diagnosis method to make full utilization of the semi-defective router to improve the network performance. CBCG can be implemented by either on-line or off-line fashion and run right after chip fabrication or periodically for changing network topologies, where some of the links/routers are out-of-serive because of broken components or fine-grained ON/OFF power management of voltage-frequency islands.

Section 2 of this paper summarizes related work. Section 3 describes our generalized turn-prohibited fault-tolerant framework along with proofs for deadlock-freedom and connectivity guarantee. The adaptive fault-tolerant virtual channel free routing algorithm is presented in Section 4, followed by the proposed heuristic rule described in Section 5. Extensions to the methodology are the subject of Section 6. Section 7 provides experimental results and a comparison of CBCG to previous methods. Section 8 concludes the paper.

2 Related Work

The growing concern about reliability of OCN has prompted extensive research in recent years [7], [8], [9]. Glass and Ni [10] proposed *one-fault-tolerant* routing derived from negative-first routing algorithm without VCs with (n-1) fault-tolerant degree for n-dimensional meshes. Wu [11] presented a dimension-order and odd-even turn based algorithm, but his approach does not support failures of edge nodes. Recently, Fu et al. [12] proposed an abacus-turn-model based reconfigurable routing method named *AbTM*. It extended the Odd-Even turn-model, where a specified node is selected in each column and named *clockwise bead* or *counter-clockwise bead*. All East-South turns are prohibited above the *clockwise bead*, and all South-West turns are forbidden below it. As for the *counter-clockwise bead*, all North-West turns above it are prohibited and all East-North turns below it are forbidden. Fick presented a similar method [13].

A block based fault model [14] sacrifices system processing capability because some fault-free nodes are isolated and marked as faulty in order to form rectangular or convex regions. In addition, packets are routed around the fault regions, thus leading to significantly unbalanced link utilization and degraded network performance. Recently, Fukushima et al. [15] proposed a routing algorithm named *Overlapped-Ring-Chain-Route* to reduce the number of deactivated nodes in the rectangle region, while still satisfying the computational capability of faultless nodes.

Prohibiting certain turns is a convenient and powerful approach to deadlock prevention. Glass and Ni [16] proposed 12 different ways to break a link's dependence and three unique turn models, namely "West-first", "North-last" and "Negative-first"; and Chiu [17] introduced "Odd-even" turn. However, there are two obvious side effects when Glass and Chiu's turn prohibitions method is put into practice. The first is that turn-models are suitable for regular topologies such as meshes and tori, but they may not be feasible if certain connectivity is removed by the presence of heterogeneous IP blocks in MPSoCs or in customized On-chip network; the other is the fixed position of forbidden turns regardless of the realistic fault situation. Furthermore, the limited routing options could still be affected by faulty components, and cause flows inconsistency, thus strictly adhering to the turn-model may generate an inconsistent network [8], [18], [19], [20]. Turn-models are not feasible for irregular networks, but the proposed CBCG has no such restrictions on the network topology.

A well-known method named Up*/Down* delivers messages in irregular networks can be adopted to deal with faulty networks [21]. It first builds a BFS spanning tree, then assigns communication links either an "up" or "down" direction. Messages which are using a down link cannot make a turn to up links, eliminating the cyclic dependencies of links. However, this method might cause a bottleneck near the root of the tree. Dong proposed multiple spanning trees to improve the efficiency of the Up*/Down* routing scheme [22]. However, with high turn prohibitions the Up*/Down* routing disables too large a number of turns, resulting in lower adaptivity and degrading network performance. We propose an adaptive fault-tolerant routing methodology which prohibits a reasonable number of turns to avoid the formation of a cycle while preserving connectivity of the underlying faulty network.

Packets trapped in a hold-and-wait cycle can always be "drained" through an additional dedicated escape channel [23], [24], which provides routing flexibility without the constraint of an acyclic CDG. However, the route and virtual channel arbitration mechanism and their implementation become complicated due to the increased number of channels. Gomez [25] proposed an intermediate node based multi-phase routing using different escape channels for each phase, however this approach still suffers from some limitations because the underlying Duato's protocol may not make full use of escape channel's bandwidth and the required virtual channel number is proportional to the number of intermediate nodes. Recently, methods for full utilization of the semi-defective router have been introduced [26], [27], [28]. Other approaches with some tolerable addition of hardware including spare wires, backup paths, crossbar bypass bus and buffers would increase OCN components' complexity [18], [29], [30].

Inspired by a simple cycle-breaking (SCB) algorithm [31] we have improved our previous work [32] to be a more practical implementation on OCN and also propose a heuristic rule to select a relatively good performance solution among all the qualified candidates. Moreover, two extensions of CBCG are introduced to further reduce the negative influences of out-of-service links/routes and improve the performance of our proposed methodology.

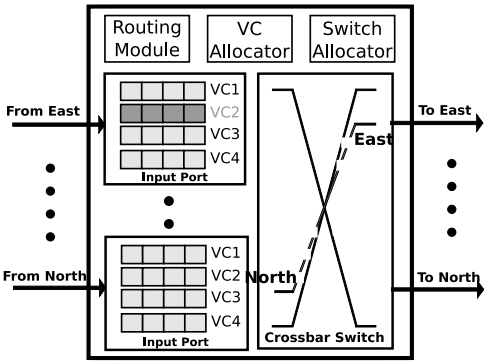
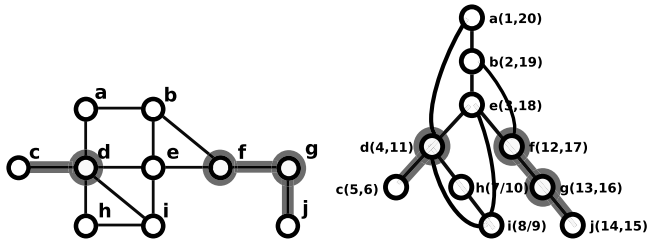


Fig. 1. Structure of a generic wormhole VC router.



(a) Network topology example (b) Depth-first tree

Fig. 2. The *cut vertices* and *bridges* are heavily shaded in (a) The network topology which is a connected graph and (b) The corresponding depth-first search spanning tree. Timestamps within nodes indicate discovery time/finishing times.

3 METHODOLOGY

3.1 Network Model

The structure of a generic wormhole virtual-channel router is shown in Fig. 1. Each port is associated with a pair of input and output links, which contains a single or several input buffers (also called VCs). As shown in the figure, the data path of the router consists of buffers and a crossbar switch. The routing module and VC allocator determine the next hop and the next virtual channel, and switch allocator is responsible for determining which flits are selected to traverse the crossbar. When a message is blocked because there is no available buffer space in the downstream router, it will hold the buffer resources that are along its path. Therefore, message routing in wormhole switch based networks is prone to deadlock.

3.2 Definitions and Preliminaries

We first give standard definitions of architecture characterization graph (ARCG) and directed channel dependency graph (DCDG).

Definition 1. *Given an OCN architecture characterization graph $G = G(R, L)$, where the routers and links in the network are given by the sets R and L, each $r_i \in R$ represents one router that is associated with each processor element, while each arc $l_{i,j} \in L$ represents a bidirectional link from r_i to r_j.*

Definition 2. *A directed channel dependency graph $DCDG = DG(V, E)$ is derived from the ARCG, where nodes $v_{i,j}, v_{j,i} \in V$ corresponds to a bidirectional edge $l_{i,j}$ in ARCG. There is a directed arc from $v_{i,x}$ to $v_{x,j}$ if $l_{i,x}$ and $l_{x,j}$ are input and output links of the same node x (ignoring 180 degree turns).*

For a given $ARCG$, two nodes i and j are called *connected* if $ARCG$ contains a path from i to j; graph $ARCG$ is said to be *connected* if every pair of nodes in $ARCG$ is connected. For a given connected graph $ARCG$, a *cut vertex* of $ARCG$ is a node whose removal results in a disconnected $ARCG$; a *bridge* of $ARCG$ is an arc whose removal disconnects $ARCG$ (see Fig. 2a for an example: nodes d, f, g are *cut vertices* and arcs (c, d), (f, g) and (g, j) are bridges). *Strongly connected nodes* of the $ARCG$ is a set of nodes $C \subset V$ such that for every pair of nodes x and y in C, they are reachable from each other; a *connected component* ($ARCG^{cc}$) is a subgraph of $ARCG$ containing all the nodes belonging to C; a *turn* in $ARCG$ is a triple of nodes(i, j, k) if $l_{i,j}$ and $l_{j,k}$ are edges in $ARCG$, whereas in the corresponding $DCDG$, it is the edge between the nodes $v_{i,j}$ and $v_{j,k}$; a path $P = (r_i, r_j, \ldots r_i, r_j)$ in $ARCG$ or $P = ((v_i, v_j), (v_m, v_n), \ \ldots, (v_i, v_j))$ in $DCDG$ is called a **cycle**.

3.3 Identify *Cut Vertices* and *Bridges*

It is important to emphasize that the unpredicted position of faults can possibly result in a disconnected network even though it was initially designed to be connected. If one of the fault routers (links) happens to be the *cut vertex* (*bridge*) of the underlying $ARCG$, the failure will disconnect all node pairs belonging to disconnected subgraphs. Fault tolerance without disabling any flawless routers, in a defective network can be beyond the capacity of fault-tolerant routing. Therefore, checking whether the defective components are the *cut-vertices (bridges)* of the network is a prerequisite of our methodology.

Let $ARCG_\pi = (R, L_\pi)$ be the depth-first tree of $ARCG$. The root of $ARCG_\pi$ is a *cut vertex iff* it has at least two children in $ARCG_\pi$; if v is a non-root node of $ARCG_\pi$, v is a *cut vertex iff* v has a child u such that there is no back edge from u or any descendant of u to a proper ancestor of v. We can see that in Fig. 2b, root a has only one child b, thus a is not a *cut vertex*, non-root d has a child node c, where there is no back edge from c to any ancestor of d, thus node d is a *cut vertex*. Similarly, node f and g are identified as *cut vertices*. An edge of $ARCG_\pi$ is a *bridge iff* it does not lie on any simple cycle of $ARCG$. Therefore, edges (c, d), (f, g) and (g, j) are marked as *bridges* in Fig. 2b.

Note that although different root node selection, and the order of visiting neighbors of a node, might generate different Depth-first search spanning tree, the set of *cut vertices* and *bridges* are equivalent as described in Chapter 22 [33].

3.4 Cycle-Breaking and Connectivity Guaranteed

Deadlock occurs when in-flight packets are holding onto a set of network resources in a cyclic manner, thereby inhibiting routing progress indefinitely. A network's deadlocking properties can be depicted using $DCDG$ by noting the existence of cycles, in other words, the potential for deadlock exists if cycles are present in the graph. Therefore, a widely used deadlock avoidance approach is to disallow the appearance of cycles in the network's $DCDG$. According to Dally and Seitz [34], a routing algorithm is deadlock free if the links can be numbered and every message can only traverses links in a strictly increasing (or decreasing) order.

Therefore, to meet our dual objectives of deadlock avoidance and preserve the connectivity, we need to find a way to label each link and generate a SET_{pturns} of prohibited turns. The proper ordering of available links can be used for routing, while guaranteeing at least one path could connect every pair of nodes for a connected graph.

We present an algorithm called CBCG as described in Algorithm 1, followed by properties and an analysis of the CBCG.

- Line 8: We can determine the set of *cut vertices* of $ARCG$ using the Depth-first search algorithm (as described in Section 3.3).
- Line 9-10: It is reasonable to assume that a node with larger degree produces more traffic congestion, so we prefer to select nodes with minimal degree as the constrained turns. There is possibly more than one node with the minimal degree in SET_{ncut}, and different ordering of the selected nodes will produce different SET_{pturns} and SET_{aturns}. The influence on network performance and a proposed heuristic rule for node selection is discussed in Sections 4 and 5.
- Line 11-16: Note that at this stage of the algorithm when node x is selected, all other undeleted nodes are yet unlabeled, therefore turn (i, x, j) is prohibited *iff* $label(x) < label(i)$ and $label(x) < label(j)$; but turn (x, i, j) and (i, j, x) are allowed, such that $label(x) < label(i)$ and $label(x) < label(j)$;
- Line 18-19: The labeling assignment is in an increasing order and guarantees each link has a unique number.

Algorithm 1. Pseudocode of CBCG Algorithm

Input: An undirected graph $ARCG(R, L)$
Result: Sets of prohibited and allowed turns at each node in the $ARCG$.

1: **Initialization**;
2: SET_{ncut} is empty; /* Set of *non-cut vertices* */
3: $SET_{lvertex}$ is empty ; /* Set of labeled nodes */
4: SET_{pturns} is empty ; /* Set of prohibited turns */
5: SET_{aturns} is empty; /* Set of allowed turns */
6: $label == 1$;
7: **while** $|R|! = 2$ **do**
8: Generate $ARCG_{\pi}(R, L_{\pi})$; /* *Depth-first tree of ARCG* */
9: Determine the SET_{ncut} of $ARCG(R, L)$;
10: Select a *non-cut vertex* x from SET_{ncut} with
11: the minimal degree;
12: **for** *turns* (i, x, j) in $ARCG(R, L)$ **do**
13: Add (i, x, j) in SET_{pturn}
14: **end**
15: **for** *turns* (x, i, j) *and* (i, j, x) *in* $ARCG(R, L)$ **do**
16: Add (x, i, j) and (i, j, x) in SET_{aturn}
17: **end**
18: Remove node x from $ARCG(R, L)$;
19: Label node x equal to $label$;
20: $label++$
21: **end**
22: Label the remained two nodes $|R| - 1$ and $|R|$;
23: **retrun** SET_{pturns} and SET_{aturns}

Example. Fig. 3 demonstrates the operation of the CBCG algorithm. The original defective connected network is shown in Fig. 3a, where dashed lines indicate faulty router D and faulty links. It requires seven stages to complete the algorithm because there are eight available nodes in the graph. SET_{ncut}, $SET_{lvertex}$, SET_{pturns} and SET_{aturns} are $\emptyset$ after initialization. At the beginning, the number of unlabeled nodes in $ARCG$ is 8. By using DFS algorithm, the number of *non-cut vertex* is determined to be 6 (the *cut vertices* B and H are heavily shaded in Fig. 3a). At stage one, there are two *non-cut vertices* A and G, both with the minimal degree of 1. We randomly select A and label it '1', as shown in Fig. 3b. None of the turns are prohibited, since the degree of node A is 1; after that, node A is removed and CBCG algorithm proceeds. At stage two, node G is selected and labeled '2', as is shown in Fig. 3c; in stage 3, node C with degree 2 is selected (among all the four qualified candidates B, C, H and I) and is labeled '3'. Now the two prohibited turns are denoted by dotted arcs in Fig. 3d, i.e., (B, C, F) and (F, C, B). During the last stage, nodes I, F are labeled '7', '8' and algorithm CBCG is finished. All the prohibited turns and labeled nodes are shown in Fig. 3g.

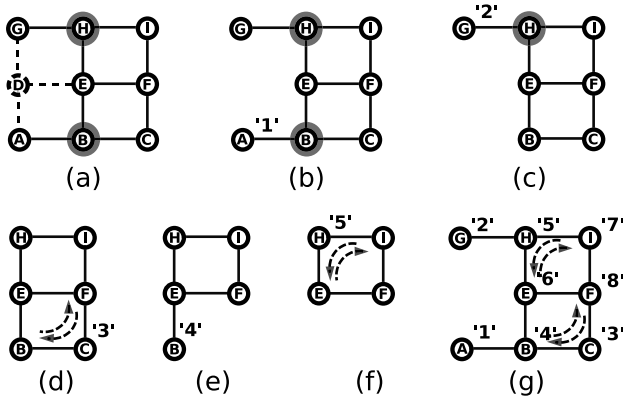


Fig. 3. Example demonstrating the CBCG algorithm. The fraction of prohibited turns is 20 percent.

Two properties of the CBCG algorithm are :

- CBCG is deadlock-free guaranteed.
- CBCG is connectivity guaranteed.

Proof of Property 1. Assuming there is a cycle C in $ARCG$, node i is with the minimum label $label(i)$ in C. Then there exists a turn (m, i, n), both $label(m)$ and $label(n)$ are greater than $label(i)$, $m, i, n \in C$. Obviously, based on the labeling scheme turn $(m, i, n) \in SET_{pturns}$, which is forbidden and brings a contradiction, so cycle C is non-existent.

Proof of Property 2. The property can be expressed as: for any two nodes $x, y \in ARCG$, there exists a path $P = (x, \ldots, y)$ that does not include turns from SET_{pturns}. In the first stage, node V_1 is selected, labeled '1' and deleted afterwards, therefore, $SET_{lvertex}$ has one element V_1, and SET_{aturns} has turns of the form (V_1, i, j) and (i, j, V_1). Because node V_1 is a *non-cut vertex* in $ARCG$, according to line 9 in Algorithm 1, there still exists a path from any node x to any node y if $x, y \in ARCG \setminus SET_{lvertex}$; Likewise, if $x = V_1$ or $y = V_1$, all the turns of the form $(i, j, x$ or $y)$ and $(x$ or $y, i, j)$ are allowed. Thus there exists at least one path from x, $x \in ARCG \setminus SET_{lvertex}$ to $y = V_1$ or from y,

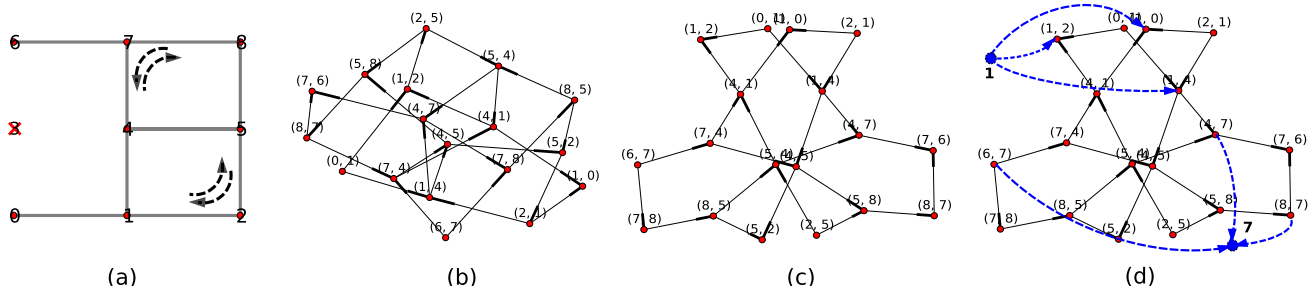


Fig. 4. Examples of acyclic channel dependency graphs generated using CBCG algorithm. (a) The defect network, fault node is 3 and fault link is $(0,3)$; (b) DCDG of (a); (c) The acyclic DCDG of Fig. 3 using CBCG algorithm, the order of selected nodes is 0-6-2-1-7-4-8-5, and forbidden turns are ((1,2), (2,5)) and ((5,2),(2,1)) at node 2, ((4,7),(7,8)) and ((8,7),(7,4)) at node 7; (d) is (c) with added dummy nodes and arcs to determine available routes for flow given source and destination nodes 1 and 7. They are highlighted with blue dashed circles and lines.

$y \in ARCG \setminus SET_{lvertex}$ to $x = V_1$. Now, assuming after stage n, there are total of four kinds of circumstances of source-destination pair (x, y) as discussed below:

Case1. $x, y \notin SET_{lvertex}$.
At stage n, node V_n is the selected *non-cut vertex* and labeled 'n', according to the definition of *cut vertex*, there still exists at least one path between unlabeled nodes $x, y \in ARCG \setminus SET_{lvertex}$.

Case2. $x \in SET_{lvertex}$ and $y \notin SET_{lvertex}$.
Assuming $x \in SET_{lvertex}$ and is labeled 'X', $y \in ARCG \setminus SET_{lvertex}$ without a label. Taking a step back to stage X-1, based on the CBCG labeling scheme, at this moment both x and y are unlabeled, thus according to **Case1** x and y are connected; Then at stage X node x is selected, however all turns of the form (x, i, j) are permitted, so x and y are still connected.

Case3. $x \notin SET_{lvertex}$ and $y \in SET_{lvertex}$.
At this moment $y \in SET_{lvertex}$ and is labeled 'Y', $x \in ARCG \setminus SET_{lvertex}$ without a label. Consider stage Y-1, we have the same situation as **Case 2**, thus x and y are connected because $x, y \in ARCG \setminus SET_{lvertex}$; at the next stage Y, *non-cut vertex* y is selected and all turns of the form (i, j, y) are still allowed, thus x and y are connected.

Case4. $x, y \in SET_{lvertex}$.
Considering now x and y are labeled 'X' and 'Y', assuming 'X'<'Y', then go back to stage X, at this moment we have the same situation as **Case 2**; whereas 'Y'<'X', moving back to stage Y, we have the same situation as **Case 3**, therefore x and y are connected.

Finally for both cases, the proposed CBCG is connectivity guaranteed. In summary, by imposing the turn prohibitions in this manner, any possible cycle contains at least one constrained turn, so all cyclic link dependencies in the network are eliminated, while the network remains connected.

3.5 Complexity Analysis

A Depth-first tree $ARCG_\pi = (R, L_\pi)$ is generated from $ARCG$, which is needed to calculate the set of *cut vertices* (line 8 of Algorithm 1) and DFS runs in time $O(|R| + |L_\pi|)$. It takes $O(1)$ to check whether the root of $ARCG_\pi$ is a *cut vertex*. Any non-root node is a *cut vertex iff* it has a child u in $ARCG_\pi$ with no back edge to a proper ancestor of itself, time for this procedure is proportional to the number of children of v in $ARCG_\pi$. Thus to check over all non-root nodes is $O(|R|)$. Selecting a node x with the minimal degree from the set of *non-cut vertices* (line 9 of Algorithm 1) runs in $O(|L_\pi|)$. Lines 11-12 and 14-15 both are $O(|R|)$, and lines 17-19 is $O(1)$. After that, selected *non-cut vertex* is removed from the $ARCG_\pi$ and we need to repeat the procedure once again until the number of remaining nodes is 2. The removed node only affects the cut property of its parents, so the worst-case running time is $O(|R|)$. In all, the complexity of CBCG algorithm is $O(|R| + |L_\pi|)$.

4 Fault-Tolerant Routing

An irregular underlying topology further compounds the difficulty of message routing in a defective network. In general, fault tolerant routing framework needs to determine whether any router needs to be disabled and it needs to assign a feasible routing guideline.

4.1 Determining the Maximal Connected Subgraph of *ARCG*

Before calculating all the prohibited turns to eliminate cycles using CBCG we need to verify whether the defective network is fault-tolerable without disabling any healthy nodes, as mentioned in Section 3.3. Node pairs can only communicate with each other if they belong to the same *connected subgraph*. Suppose the node set $V^{cg} = \{G_1, G_2, \ldots, G_k\}$ contains all the *connected subgraph* G_i of $ARCG(R, L)$. If there is only one element G_1 in V^{cg} ($V^{cg} = \{G_1\}$) then $G_1 = R$, then every node pair in $ARCG$ is connected. Otherwise we only keep the G_{max} with the maximize number of verties in V^{cg} and disable all the other vertices. $ARCG^{cg}_{max}$ is the corresponding subgraph of $ARCG$ driven from G_{max}. The V^{cg} is totally determined by the fault situation of the defective network, which can be calculated by calling the DFS($ARCG$). Each tree formed by DFS algorithm is a separate G_i of $ARCG$ and accordingly the one with the maximal number of nodes is the G_{max}.

4.2 Adaptive Routing without Virtual Channel

After determining the $ARCG^{cg}_{max}$ we can generate the acyclic $DCDG$ by deleting edges $((i, j), (j, k))$ according to CBCG ($ARCG^{cg}_{max}$) and forbidding all the turns in the form (i, j, k) in SET_{pturns}. Fig. 4 demonstrates an example, here the $ARCG^{cg}_{max} = ARCG$ and $G_{max} = \{0, 1, 2, 4, 5, 6, 7, 8\}$ in this

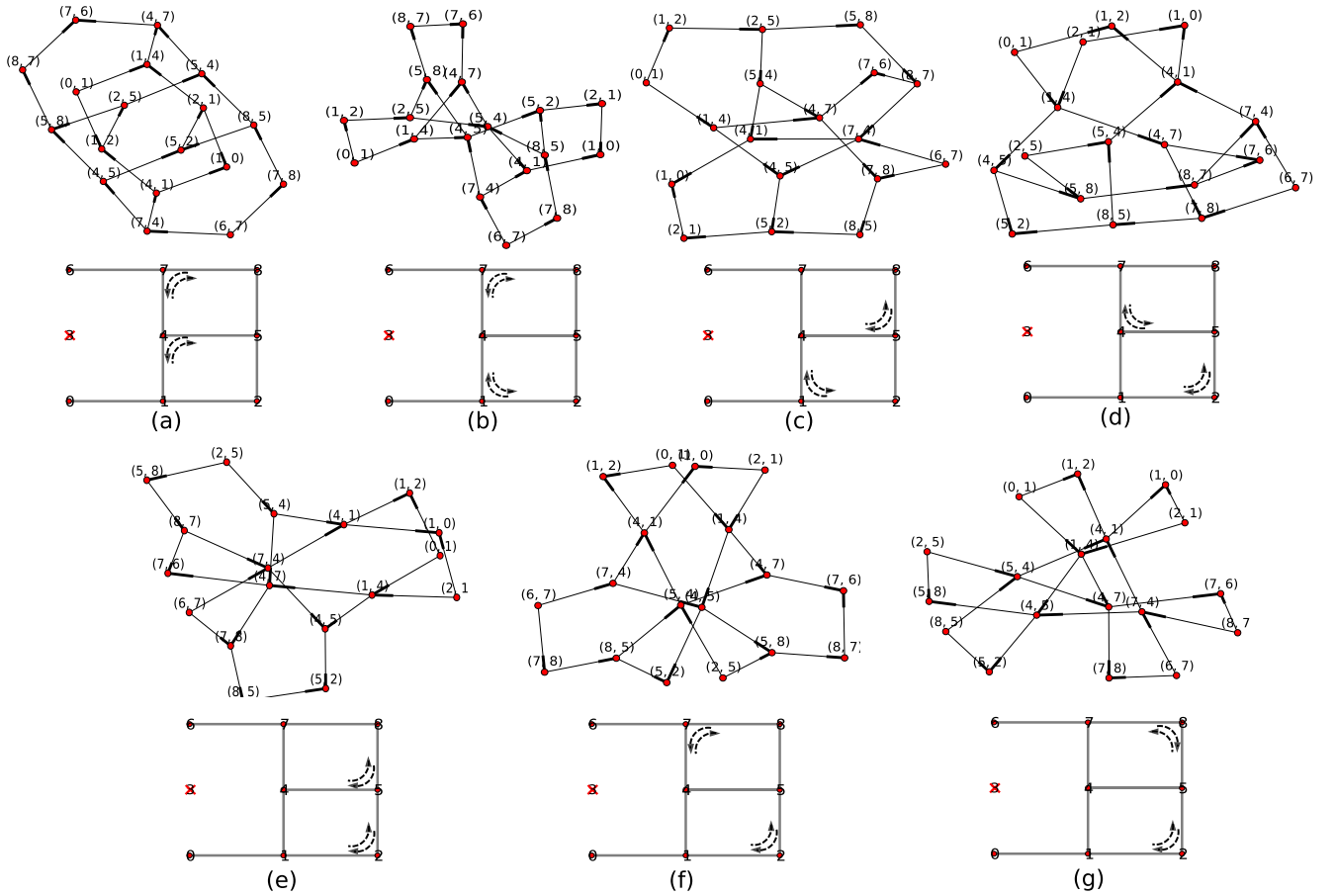


Fig. 5. All the seven DCDGs generated using CBCG algorithm with different ordering of selected *non-cut vertices*, the sequence of selected nodes are 6-0-7-8-4-5-1-2, 6-0-1-2-7-4-5-8, 0-6-7-8-5-4-2-1, 0-6-8-7-4-1-5-2, 6-0-8-7-5-2-4-1, 0-6-2-1-7-4-5-8 and 6-0-2-1-8-7-5-4; the dashed arrows in each figure indicates forbidden turns.

case. Fig. 4a is the same with Fig. 3a, except that nodes are annotated with natural numbers rather than alphabets for convenience. Fig. 4b is the corresponding DCDG of Fig. 3a. According to the previous analysis, SET_{pturns} = {$(1,2,5)$, $(5,2,1)$, $(4,7,8)$, $(8,7,4)$}. Fig. 4c is generated by removing corresponding edges $((1,2),(2,5))$, $((5,2),(2,1))$, $((4,7),(7,8))$ and $((8,7),(7,4))$ from the original $DCDG$. The properties of CBCG can be verified by checking for the existence of cycles and at least one path between all node pairs in the constructed $DCDG$.

As mentioned earlier, there might be multiple *non-cut vertices* with the minimal degree in every stage of the CBCG. Different ordering of selected nodes would in turn construct different acyclic $DCDGs$ as shown in Fig. 5, thus generating different SET_{pturns}. Obviously, it is worth noting that the qualities of network performance might be directly affected by different underlying topology of $DCDGs$. Further observations and discussions are introduced in Section 5.

Before making the routing decision for flow i with source S_i and destination D_i, we need to make a temporary modification to the acyclic $DCDG$ by adding dummy nodes S_i and D_i. Then we add directed arcs from S_i to all nodes of the form (S_i, x), and to D_i from nodes of the form (x, D_i). Fig. 4d shows an example of traffic flow from node 1 to 7 (Dummy vertices are 1 and 7, added arcs are $((1,(1,2))$, $(1,(1,0))$,$(1,(1,4))$,$((6,7),7)$,$((4,7),7)$ and $((8,7),7)$, they are highlighted using blue dashed circles and lines). Then adaptive routing is applied and the route taken by a packet is determined dynamically based on available paths. For example, if node 1 wants to send a packet to node 7, it can choose either path $(1,(1,4),(4,7),7)$ or $(1,(1,4),(4,5),(5,8),(8,7),7)$ as is shown in Fig. 4e. Dummy nodes 1, 7 and associated arcs are removed afterwards. Repeat the procedure until all the traffic flows are assigned with feasible routes.

Proposed adaptive routing is implemented with node table-based routing. Tables are stored at each router, which consist of all the possible outgoing channels corresponding to its flow identifier. Adaptive route decisions are made by leveraging the local congestion information at intermediate hops along the path at runtime to improve network performance.

5 Observations

The CBCG algorithm prohibits turns to avoid deadlock while preserving connectivity. The different orders of selected *non-cut vertices* generate different positions and types of forbidden turns, which will influence the DCDG topology directly. In Fig. 5, we depict seven different acyclic DCDGs obtained by using different ordering of *non-cut vertices* selection. Since route options are determined by the available channels of the underlying DCDG, different orders of selected *non-cut vertices* will influence the network performance indirectly.

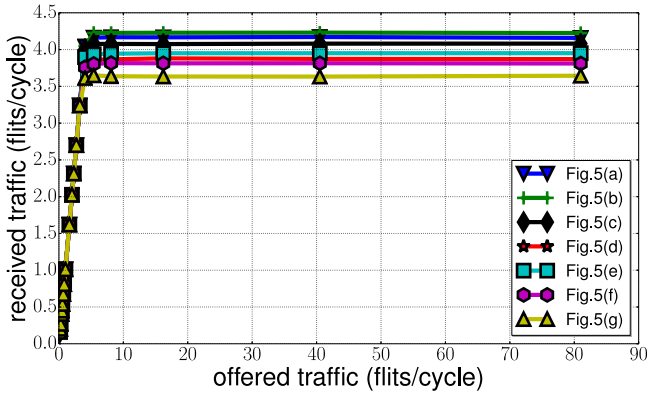


Fig. 6. Throughput results for all the seven DCDGs as shown in Fig. 5 under UNIFORM-RANDOM traffic pattern.

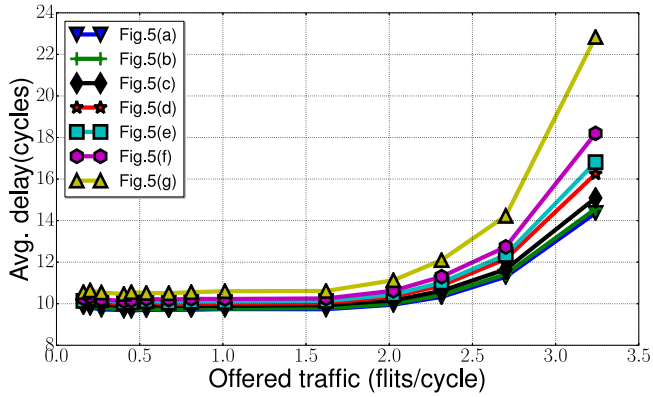


Fig. 7. Packet latency for all the seven DCDGs as shown in Fig. 5 under UNIFORM-RANDOM traffic pattern.

Using Fig. 4a as an example, there are a total of 64 different ways to prevent deadlock while guaranteeing connectivity. As per the procedure of CBCG, at stage 1 nodes 0 and 6 both have the minimal degree 1; then at stage 3, there are four nodes $1, 2, 7, 8$ with the minimal degree 2, notice that at this moment nodes 0 and 6 are removed, thus 1 and 7 also have degree 2 (see Fig. 3d). Similarly, there are four options at stage 5 and 2 options at stage 6. Therefore, there are a total of $C_2^1 * C_4^1 * C_4^1 * C_2^1 = 64$ different sequence orders for selecting *non-cut vertices*. However, there are only seven unique patterns when symmetries are taken into account. In the following sections, we put forth some observations made through our extensive experiments.

5.1 What Is the Performance Impact of Different Ordering of *Non-Cut Vertices* Selection or Different DCDGs?

We use HORNET, a highly configurable, cycle-accurate on-chip network simulator [35] for all the simulations. Figs. 6 and 7 display throughput and average packet latency under UNIFORM-RANDOM synthetic traffic pattern for all of the seven DCDGs in Fig. 5. The CBCG achieved a large variety of saturation throughput, with a maximum of 4.225 flits/cycle for Fig. 5b and a minimum of 3.644 filts/cycle for Fig. 5g. Similarly, average packet latency was a maximum of 22.838 cycles for Fig. 5g and a minimum of 14.366 cycles for Fig. 5a.

Note that all the seven DCDGs prohibit four turns. Since the number of disallowed turns are equal, we turn our investigation to properties of the generated acyclic DCDGs. We compare different DCDGs in terms of the degree of nodes. The degree of a node is defined as the number of channels entering or leaving the node. In Fig. 5f for example, there are four nodes $(4,1)$, $(1,4)$, $(5,4)$ and $(4,5)$ with a degree of 4, and four nodes $(7,4)$, $(4,7)$, $(8,5)$ and $(5,8)$ with a degree of 3 and the other eight nodes have a degree of 2. Analysis results and corresponding network performance of all the seven DCDGs are categorized into four classes in Table 1. Fig. 5a with 12 3-degree nodes and six 2-degree nodes achieved the best network performance. In contrast, Fig. 5g with six 4-degree nodes, 12 2-degree nodes performed the worst. Observing the noticeable pattern in the Table 1, we can make an assumption that network performance is inversely proportional to the number of maximal degree nodes. The conclusion is reasonable, because local congestion could take place around the nodes with large degree. The higher degree results in higher possibility of contention, thus becoming susceptible to turning into a bottleneck. The random selection of the *non-cut vertex* cannot guarantee that the generated DCDG achieves the best behavior. Moreover, it is not practical to performs a large number of runs to get the best result. Hence, we propose a heuristic rule for proper selection of the *non-cut vertices* to obtain the DCDG that achieves near-optimal performance and state it as follows:

- Heuristic Rule to select the sequence of *non-cut vertices*: Select a *non-cut vertex* with the minimal degree, and in the event of multiple candidates, choose the one whose $Sum_d(i)$ has the largest value.

The $Sum_d(i)$ for node i can be calculated as:

$$Sum_d(i) = \sum_{j}^{Set_n(i)} ((d_i - 1) + (d_j - 1)) = d_i(d_i - 1) + \sum_{j}^{Set_n(i)} (d_j - 1),$$

TABLE 1
Property and Network Performance of DCDGs in Fig. 5

Different Acyclic DCDGs	Number of nodes with different degrees				Saturation Throughput (flits/cycle)	Latency (cycles)	Avg. Throughput (flits/cycle)	Avg. Latency (cycles)
	4-degree	3-degree	2-degree	1-degree				
Fig. 5a	0	12	6	0	4.159	14.366	4.159	14.366
Fig. 5b	2	8	2	0	4.225	14.566		
Fig. 5c	2	8	2	0	4.081	15.089	4.058	15.301
Fig. 5d	2	8	2	0	3.870	16.248		
Fig. 5e	4	4	10	0	3.950	16.813	3.880	17.505
Fig. 5f	4	4	10	0	3.811	18.198		
Fig. 5g	6	0	12	0	3.644	22.838	3.644	22.838

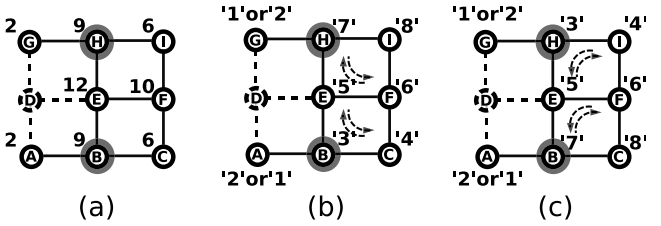


Fig. 8. Example illustrating the heuristic CBCG algorithm, dashed arrows indicate the forbidden turns.

where $Set_n(i)$ is the set containing immediate neighbors of i and d_i is the degree of node i in the ARCG. For a selected *non-cut vertex* x in an undirected ARCG, a forbidden turn comes from one of its neighbors and goes to another, thus the number of disallowed turns is proportional to square of the number of its neighbors. In addition, the forbidden turns will influence all the nodes in DCDG with the form of (x, y) and (y, x). Consequently, selection of a node with the largest value of $Sum_d(i)$, will decrease the maximal degree of nodes in the constructed acyclic DCDG. This results in a higher possibility of generating minimal number of nodes with the maximum degree. The heuristic method cannot always guarantee to get the best DCDG, however in most cases it enables us to obtain a near optimal result. To implement the proposed heuristic method we need to replace lines 9-10 in Algorithm 1 with lines in Algorithm 2.

Algorithm 2. Pseudocode of the Heuristic Method

1 SET_{min_c} is the set contains *non-cut vertex* i in SET_{ncut} with the minimal degree;
2 **if** *There is one element in* SET_{min_c} **then**
3: x = the single element in SET_{min_c};
4: **else**
5: **for** i *in* SET_{min_c} **do** Calculate $Sum_d(i)$
6: x is the one with the maximal $Sum_d(x)$
7: **end**

5.2 What Is the Complexity of the Heuristic Algorithm?

The computational cost of the heuristic method is $O(|R|)$. The degree of each node in ARCG has to be computed and added up in the initial phase, this step runs in $O(|R|)$. Time complexity to calculate $Sum_d(i)$ for every node i is $O(|R|)$. Therefore, the overall computational cost of the CBCG algorithm remains $O(|R| + |L_\pi|)$.

Example. Fig. 8 illustrates the operation of the improved CBCG algorithm. The $Sum_d(i)$ of every node i is shown in the upper-left corner of nodes in Fig. 8a. According to our proposed heuristic algorithm, at the third stage of CBCG, node B, C, H and I have the same degree of 2, thus SET_{min_c} contains 4 nodes. Then B or H is selected, because $Sum_d(B) = Sum_d(H) = 9 > Sum_d(C) = Sum_d(I) = 6$. Similarly, at the fifth stage node E with the largest $Sum_d(E) = 12$ is selected among nodes E, F, H and I or among nodes E, F, B and C depending on whether node B or H is selected at the third stage. Therefore, two circumstances are generated as shown in Figs. 8b and 8c, they are equivalent when considering the symmetry of network topology, and the corresponding DCDG is shown in Fig. 5a. The last two columns of Table 1 indicate the heuristic solution outperform others as expected.

6 Complementary Mechanisms

In this section, we consider two possible extensions to the proposed methodology : (1) improving resource utilization by fine-grained fault diagnosis and (2) increasing routing options by adding more virtual channels.

6.1 Precise Fault Diagnosis

Fault diagnosis is the ability to pinpoint the location of faults. An entire router could be out-of-service even when only a single input buffer or switch of the crossbar is defective [6]. By precisely narrowing down on the fault location we can operate semi-faulty routers or links in partial-usage modes by making full use of remaining available network resources, such as healthy input buffers and crossbar switches and re-routing packets around faulty components. Therefore, more fine-grained fault diagnosis models can be explored for better resource utilization and network performance. Grecu et al. [36] presented an on-line fault detection and location method and Kohler et al. [6] proposed a fault location approach which enables users to locate fault in the links, input buffers and in parts of the crossbar.

For example, assuming that the East to North switch link of the crossbar and the south input port buffer at router 3 are broken for network in Fig. 4a. For the generated DCDG, all the healthy buffers and available crossbar switches of the defective router are kept, and only the broken components are removed. Consequently, vertex (0,3) (the broken South input port buffer) and the edge between (4,3) and (3,6) (East to North switch link of the crossbar) are removed from the DCDG, Fig. 9a illustrates this. It is easy to come to the conclusion that the fine-grained diagnosis has some advantages over the coarse-grained technique because the former has more vertices and channels for message routing.

6.2 Multiple Virtual Channels

Virtual channels can be very expensive to implement, requiring additional memory resources and associated allocation and arbitration logic. Nevertheless, for better fault tolerance and increased link utilization multiple virtual channels can be employed as they provide logically independent communication paths to packets multiplexed across each network link, thereby reducing head-of-line blocking and improving network performance.

In the fine-grained scheme, if a link is not physically broken or all its related crossbar switches are not damaged, then the link is deemed functional as long as one of its VCs is still operational. As a result, a single broken input buffer at a port would not affect the correct functionality of that input port. Therefore, using more VCs also decreases the possibility of out-of-service links caused by unavailable downstream VCs. Consequently, there is a positive effect on the maximal connected subgraph $ARCG^{cg}_{max}$, as well as the corresponding acyclic DCDGs.

7 Performance Evaluation

This section presents the performance of CBCG algorithms under course and fine grained schemes, single and multiple

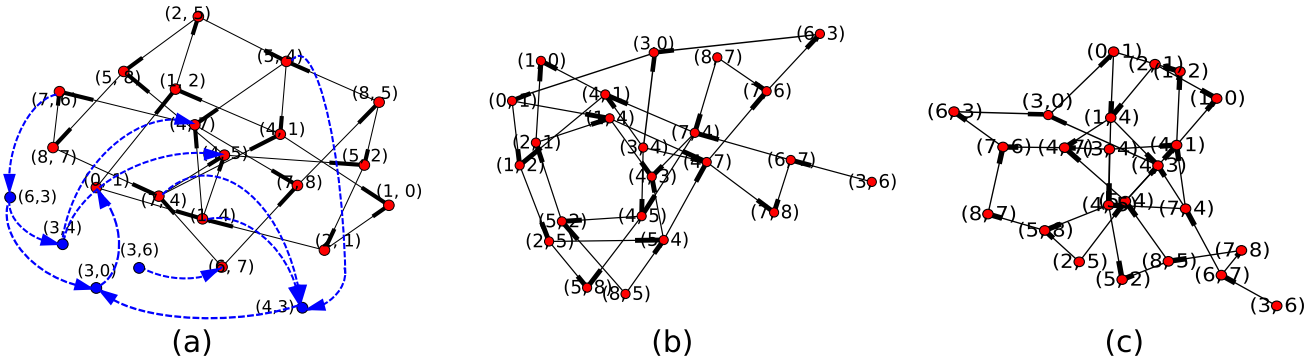


Fig. 9. (a) DCDG of Fig. 4a using precise fault diagnosis method, input buffer associated with link (0,3) and switch link from East to North of the crossbar at router 3 are broken, compared with Fig. 4a, the additional available resources are highlighted in blue dashed lines; (b) A generated acyclic DCDG by CBCG algorithm, the order of selected nodes is 3-6-0-8-7-4-5-1 and forbidden turns are ((4,3),(3,6)), ((6,3),(3,4)), ((1,4),(4,5)), ((5,4),(4,1)), ((5,8),(8,7)) and ((7,8),(8,5)); (c) Another constructed acyclic DCDG the order of selected nodes is 3-6-0-7-8-2-5-1-4 and forbidden turns are ((4,3),(3,6)), ((6.3),(3,4)), ((1,2),(2,5)), ((5,2),(2,1)), ((4,7),(7,8)) and ((8,7),(7,4)).

virtual channel configuration. Through simulation experiments, we compare CBCG with other fault-tolerant routing methods like *Finter* [25], DATE09 [13] and uDIREC [28].

7.1 Simulation Details and Traffic Patterns

We use HORNET, a highly configurable, cycle-accurate on-chip network simulator [35] for all the simulations. In our experiments, HORNET as working under network-only mode. Synthetic benchmarks and traffic profiles obtained from a parallel implementation of a H.264 decoder are used to investigate network performance. For the coarse-grained scheme, flows with faulty nodes as source or destination are removed. In the fine-grained scheme, a node is unavailable as a source node when all the virtual channels of its input port from the local processing element are broken, or if all the switch links of the crossbar connected with local input port are dead. Similarly, a node cannot be a destination node if all the switch links of the crossbar leading from other input ports to the local output port are out-of-service. The traffic has a uniform random distribution with packet length set to 8 flits. We implemented an 8×8 2D-Mesh and Torus with 5, 10, 15, 20, 30 and 40 percent fault-rates. The positions of unavailable nodes and links were randomly generated. For the fault situations, we made the assumption that the number of out-of-service nodes and links ratio is approximately 1:2 like Boppana presented [14]. Under the fine grained scheme, virtual channels and switch links have the same probability of failure in a faulty router. In addition, we gave the same total amount of buffer resources for all the experiments. In order to get performance results independent of relative distribution of faults, we performed 10,000 simulations with different fault distribution for each fault-rate case. All experiments have 200,000 warm up cycles and a total of 1,200,000 analyzed cycles.

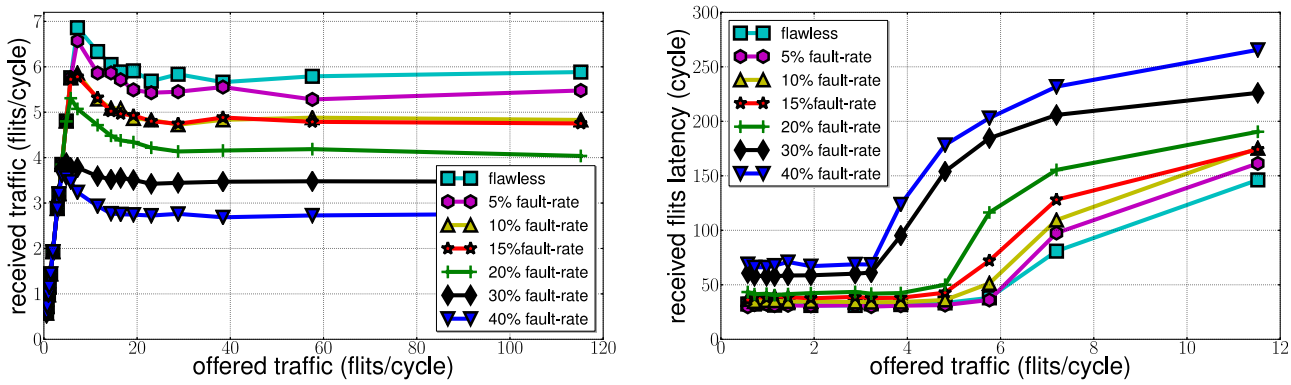
7.2 Reliability Analysis

The CBCG methodology is capable of tolerating a significant percentage of faults without disabling any healthy nodes, subject to the underlying defect topology being connected. The statistical results can be viewed in Table 2. We observe that even under 40 percent fault-rate, CBCG-*fine* exhibits very good reliability. Only 0.07 percent of the simulations are not fully achievable for CBCG-*fine* with two VCs in a 2D-Torus network. For lower fault-rates, reliability improves slightly in the 2D-Mesh topology and it is 100 percent reliable for a 5 percent fault-rate. For 2D-Torus, reliability is a 100 percent for fault rates of 20 percent and below. In the 2D-Mesh network for CBCG-*coarse*, reliability drops rapidly as the number of faults in the network increase. Interestingly, 2D-Torus suffers only 2.49 percent of non fully achievable simulations even under 40 percent fault-rate for CBCG-*coarse*. It is worth noting that applying precise fault diagnosis scheme can dramatically improve system reliability, however adopting multiple virtual channels only increases the reliability slightly. The probability of disjoint subgraphs increases when the number of faults rises, at the same time the rare

TABLE 2
Reliability with Increasing Faults in a 8×8 2D-Mesh and 2D-Torus Networks without Disabling Any Healthy Nodes

F_rate	2D-MESH						2D-TORUS					
	coarse (VC1/2)		fine(VC1)		fine(VC2)		coarse (VC1/2)		fine(VC1)		fine(VC2)	
	Reliability	T_rate	Reliability	T_rate	Reliability	T_rate	Reliability	T_rate	Reliability	T_rate	Reliability	T_rate
5%	99.98%	24.74%	100%	26.15%	100%	25.93%	100%	29.68%	100%	30.87%	100%	30.63%
10%	99.89%	24.38%	99.98%	25.99%	99.98%	25.75%	100%	28.43%	100%	30.24%	100%	29.99%
15%	99.20%	23.94%	99.91%	27.00%	99.93%	26.53%	100%	27.15%	100%	30.28%	100%	29.79%
20%	97.63%	23.39%	99.73%	26.80%	99.80%	26.30%	100%	26.00%	100%	29.68%	100%	29.17%
30%	90.08%	21.94%	98.98%	27.37%	99.02%	26.60%	99.85%	24.67%	100%	29.53%	99.99%	28.78%
40%	83.66%	21.01%	98.12%	28.11%	98.18%	27.10%	97.51%	22.93%	99.95%	31.08%	99.93%	29.70%

F_rate and T_rate are the fault rate and percentage of forbidden turns.



(a) Throughput (b) Latency

Fig. 10. Throughput and latency results of different fault-rates. Each point is the average results of 10,000 simulations for an 8 × 8 2D-Mesh. The results of 4 × 4 2D-Mesh exhibited the same feature and we omit them here for brevity. The average fraction of prohibited turns are 25.12, 24.47, 24.38, 23.94, 23.39, 21.94 and 21.01 percent for flawless, 5, 10, 15, 20, 30 and 40 percent fault-rates respectively.

situation of a small number of faults in a large network resulting in a disjoint network has been encountered. In disjoint networks node pairs belonging to different subgraphs can no longer communicate. In these cases, faults cannot be tolerated without disabling any faultless nodes and this is an inherent limitation for all kinds of fault-tolerant routing.

7.3 Performance in the Presence of Faults

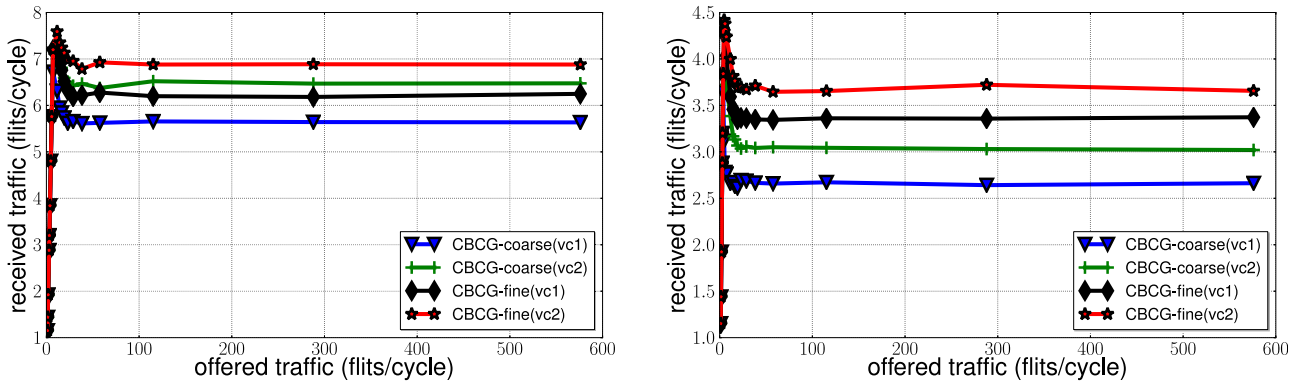
7.3.1 Coarse-Grained CBCG without VCs

Using a flawless network as the baseline for our discussion, we first measure network performance by means of saturation throughput and packet latency obtained in the presence of faults under uniform traffic. Fig. 10a shows the network saturation throughput. The achieved throughput is decreased by 5.15 percent when 5 percent faults are present in the network, but it drops by 46.5 percent when fault rate increases to 40 percent. It is noticeable that saturation throughput drops rapidly as the number of injected faults increases. This behavior is a consequence of fewer available links and routers for communication. The fact that average distance of paths increases when routes around faults use non-minimal routing contributes to this behavior. Fig. 12 shows the results of packet latency. We see that packet latency is significantly increased when the number of failure links rises. At packet injection rates below saturation, average flit latency is increased by 8.24 percent when 5 percent faults are present and the latency is increased by 131.36 percent when the fault rate is 40 percent.

7.3.2 Fine-Grained CBCG with 2VCs

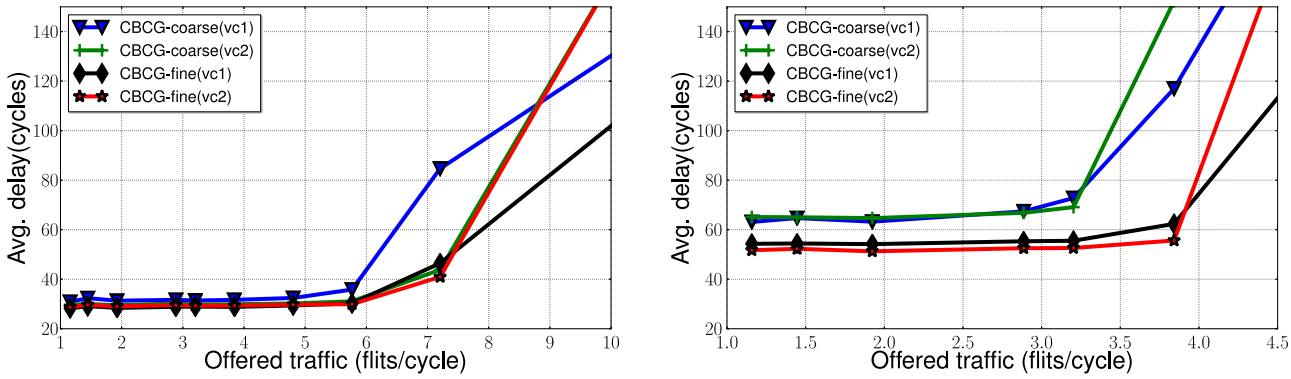
In this section, we compare the network performance when precise fault diagnosis and multiple VCs are employed. Fig. 11 shows the throughput results when fault rates are 5 and 40 percent. As expected, 2VCs along with a fine-grained fault diagnosis scheme exhibits the best performance. CBCG-*coarse* with two VCs gains 14.92 percent improvement over CBCG-*coarse* using only a single VC under 5 percent fault-rate. When the fault-rate increases to 40 percent, the improvement is reduced to 14.15 percent. As for CBCG-*fine*, the performance of two VCs compared to single VC is improved by 10.11 and 10.38 percent under 5 and 40 percent fault-rates respectively.

Comparing the performance of CBCG-*fine* and CBCG-*coarse*, at 5 percent fault-rate we see that precise fault diagnosis with a single VC achieves an improvement of 10.90 percent over CBCG-*coarse*. When two VCs are used, the improvement is 6.25 percent. As the fault-rate increases to 40 percent, CBCG-*fine* with a single VC has a



(a) Fault rate 5% (b) Fault rate 40%

Fig. 11. Throughput results of 5 and 40 percent fault-rates for an 8 × 8 2D-Mesh under UNIFORM-RANDOM traffic pattern, faults position are randomly distributed. The results of 10, 15, 20 and 30 percent exhibited the same feature. The average fraction of prohibited turns are 24.74 and 21.01 percent for CBCG-coarse(vc1) and CBCG-coarse(vc2), meanwhile, 26.15 and 25.93 percent (28.11 and 27.10 percent) for CBCG-fine(vc1) and CBCG-fine(vc2) under 5 percent(40 percent) fault-rate, respectively.



(a) Fault rate 5% (b) Fault rate 40%

Fig. 12. Latency results of 5 and 40 percent fault-rates for an 8 × 8 2D-Mesh.

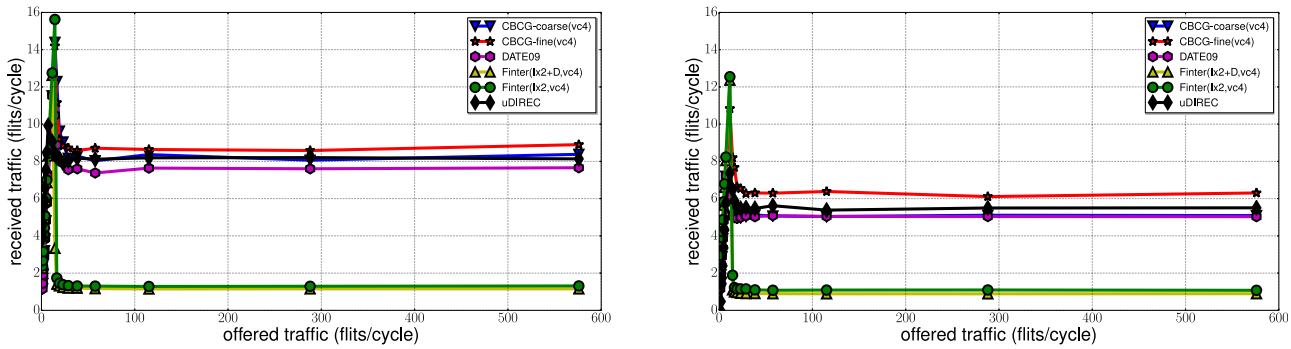
throughput of 3.050 flits/cycle as compared to 2.672 flits/cycle for CBCG-*coarse* with one VC, an improvement of 26.20 percent. CBCG-*fine* has a throughput of 3.722 flits/cycle if we use two VCs. This is a 22.03 percent improvement over CBCG-*coarse* with 2 VCs, which has a throughput of 2.672 flits/cycle. Therefore, we conclude that precise fault diagnosis is necessary especially for severe fault cases, because it utilizes the communication resources more efficiently. It is interesting to observe that while comparing Fig. 11a with Fig. 11b, under 5 percent fault-rate CBCG-*coarse* using 2VCs performs slightly better (3.63 percent) than CBCG-*fine* with single VC. However, when the fault-rate is 40 percent CBCG-*fine* using single VC gains 10.56 percent more delivery than CBCG-*coarse* with two VCs. This observation shows that precise fault-diagnosis is more effective way to improve network performance than simply adding more virtual channels for severe fault cases.

Fig. 12 demonstrates the packet latency results for 5 and 40 percent fault-rates. As expected, adopting multiple VCs and a precise fault-diagnosis scheme decreases the average latency by alleviating the HoL blocking and providing more available operational paths for transmission. Comparing Fig. 12a with Fig. 12b, we see that the average packet latency for low traffic densities more than doubles for CBCG-*coarse* as the fault-rate increases from 5 to 40 percent. For example, CBCG-*coarse* with one VC has an average packet latency of 31.327 cycles at 5 percent fault-rate and this increases to 67.411 cycles at 40 percent fault-rate. As the fault-rate increases from 5 to 40 percent the average packet latency for CBCG-*fine* employed with single or two VCs increases by 88.11 and 80.07 percent respectively. Note that both CBCG-*coarse* and CBCG-*fine* have comparable average packet latency for low fault-rate, but CBCG-*fine* degrades better as the fault-rate increases.

7.3.3 CBCG versus others

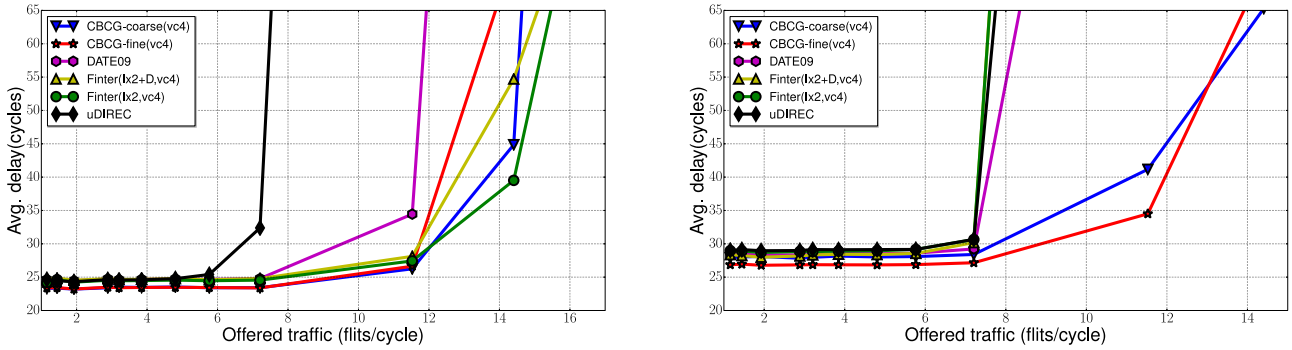
In this section, we further investigate the network performance of CBCG by comparing it with previous methods. References for comparison are the schemes proposed by Gomez et al. in [25] named *Finter*, which is an intermediate node based multi-phase routing using a different escape channel for each phase. A flag transmission and routing entry update mechanism presented by Fick et al. named DATE09 [13]. A fine-resolution detection and reconfiguration strategy named uDIREC proposed by Parikh and Bertacco [28], that makes detection decisions and stores the topology information in a software-maintained scoreboard at the "supervisor node" before appling Up*/Down* routing to avoid deadlock. As mentioned previously, 10,000 simulations are performed with uniform-randomly injected faults for different fault-rates. Fig. 13 shows the performance results for an 8 × 8 2D-Mesh network under 5 and 15 percent fault-rates. Fig. 14 shows the packet latency results for 5 and 15 percent fault-rates.

As shown in Fig. 13, CBCG-*fine(vc4)* achieves the best performance with 8.899 flits/cycle delivery rate and an average latency of 23.207 cycles under 5 percent fault-rate.



(a) Fault rate 5% (b) Fault rate 15%

Fig. 13. Throughput results of 5 and 15 percent fault-rates for an 8×8 2D-Mesh under UNIFORM-RANDOM traffic pattern, faults position are randomly distributed. The average fraction of prohibited turns are 24.74, 26.01 and 27.32 percent (23.94, 26.31 and 26.89 percent) for CBCG-coarse(vc4), CBCG-fine(vc4) and DATE09 under 5 percent (15 percent) fault-rate.



(a) Fault rate 5% (b) Fault rate 15%

Fig. 14. Latency results of 5 and 15 percent fault-rates for an 8 × 8 2D-Mesh.

CBCG-*fine(vc4)* gains 16.2 percent in throughput and decreases average packet latency by 4.29 percent when compared with DATE09. The advantages are more notable when the fault-rate increases with the throughput gain increasing to 25.3 percent and average latency improving to 4.68 percent over DATE09, under 15 percent fault-rate. DATE09 determines which links and turns are disallowed depending on the defective network topology in order to safely route packets around failures, however it overlooks the traffic load balance. CBCG applies a heuristic rule to alleviate network congestion and hence shows improved network performance. *Finter* avoids deadlock by using different escape channels along each phase, however its performance suffers from the under-utilization of escape channel's bandwidth. It shows comparable average latency for 5 percent fault-rate, but does not degrade well as fault-rate increases. CBCG has significantly better throughput at both 5 and 15 percent fault-rate. uDIREC applies fine-grain fault-diagnosis and Up*/Down* routing after exhaustively searching for the best connectivity solution among all the different *breath-first searching* results. A large percentage of turns are forbidden, because there are only two types of turns ("up-to-down" and "down-to-up"). Therefore, the routing options are limited and performance is sacrificed. uDIREC has similar throughput performance as CBCG, but poor average packet latency. In conclusion, according to our simulations results CBCG outperforms other schemes when taking both throughput and latency into consideration.

7.4 Implementation

CBCG can be implemented "off-line" or by using a "supervisor node" to maintain the network topology information allowing forbidden turns to be realized by look-up table based schemes. Meanwhile, CBCG can also be realized in a distributed fashion using node-table based routing. Specifically, the *cut-vertices* can be annotated by a distributed Depth-first search using embedded computation logic at the router, while each node's table consists of the row corresponding to the input ports and possible output ports for each direction. The disabled turns can be masked at the corresponding position of the routing table. We synthesized our "on-line" implementation using Synopsys Design Complier with TSMC 65nm library at 1.38 GHz, the area is 2209.32 and 2612.88 μm^2, occupied 0.966 and 1.139 percent for the baseline wormhole based router of 8 × 8 and 16 × 16 2D-Meshes. There are four VCs at each port, each VC contains eight flits and flit size is 64-bits [37].

8 Conclusion

Routing methods to enhance message transmission include maintaining connectivity and avoiding deadlocks, both crucially important in the presence of failures. Three main contributions for reliable network-on-chip have been proposed in this paper:

- First, deadlock freedom and preservation of connectivity are achieved by our proposed CBCG methodology, which generates a connected acyclic channel dependency graph. Our method needs to disable healthy nodes only when there are disjoint subgraphs caused by faults, which is infrequent for a network with few faults.
- Second, a heuristic method is presented to improve the effectiveness of the CDCG algorithm in order to reduce the probability of congestion and achieve near optimal performance.
- Finally, improving resource utilization by applying a precise fault diagnosis scheme, and increasing link utilization by adopting multiple virtual channels are discussed. Furthermore, these complementary mechanisms yield benefits by maximizing the size of the *maximal connected subgraph* of the network.

Finding the right balance between performance gain and resource overhead is the fundamental engineering challenge. The goal is to obtain a relatively good performance, while keeping the overhead under practical limits. CBCG works well for OCN without virtual channels, thus allowing lesser design complexity and relatively simple architecture. Precise fault diagnosis and adding virtual channels are not required in our routing algorithm, but they can be used to improve performance at the cost of increased complexity of the router implementation. To conclude, CBCG lets the user to make the decision of whether or not to use more VCs based on different application requirements.

Acknowledgments

The authors want to thank Mieszko Lis at the University of British Columbia and Srinivas Devadas at MIT, for interesting discussions throughout the course of this work. This research was partially funded by Key Project of of NSFC

(61231018) and NSFC (61303036), China Postdoctoral Science Foundation (2012M521777) and National High Technology Research and Development Program of China (2014AA01A301).

REFERENCES

[1] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. 44th ACM/IEEE Design Autom. Conf.*, 2007, pp. 746–749.

[2] S. Borkar, "Designing reliable systems from unreliable components: The challenges of transistor variability and degradation," *IEEE Micro*, vol. 25, no. 6, pp. 10–16, Nov./Dec. 2005.

[3] J. Keane and C. H. Kim, "Transistor aging," *IEEE Spectrum*, vol. 48, no. 5, pp. 28–33, 2011.

[4] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, Jul./Aug. 2003.

[5] É. Cota, F. L. Kastensmidt, M. Cassel, M. Herve, P. Almeida, P. Meirelles, A. Amory, and M. Lubaszewski, "A high-fault-coverage approach for the test of data, control and handshake interconnects in mesh networks-on-chip," *IEEE Trans. Comput.*, vol. 57, no. 9, pp. 1202–1215, Sep. 2008.

[6] A. Kohler, G. Schley, and M. Radetzki, "Fault tolerant network on chip switching with graceful performance degradation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 6, pp. 883–896, Jun. 2010.

[7] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, no. 1, pp. 1–51, 2006.

[8] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Mateo, CA, USA: Morgan Kaufmann, 2004.

[9] R. Marculescu, U. Y. Ogras, L. shiuan Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NOC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, pp. 3–21, Jan. 2009.

[10] C. J. Glass and L. M. Ni, "Fault-tolerant wormhole routing in meshes without virtual channels," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 6, pp. 620–636, Jun. 1996.

[11] J. Wu, "A fault-tolerant and deadlock-free routing protocol in 2d meshes based on odd-even turn model," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1154–1169, Sep. 2003.

[12] B. Fu, Y. Han, J. Ma, H. Li, and X. Li, "An abacus turn model for time/space-efficient reconfigurable routing," in *Proc. 38th Annu. Int. Symp. Comput. Archit.*, 2011, pp. 259–270.

[13] D. Fick, A. DeOrio, G. Chen, V. Bertacco, D. Sylvester, and D. Blaauw, "A highly resilient routing algorithm for fault-tolerant NOCs," in *Proc. Conf. Design, Autom. Test Europe*, 2009, pp. 21–26.

[14] R. Boppana and S. Chalasani, "Fault-tolerant wormhole routing algorithms for mesh networks," *IEEE Trans. Comput.*, vol. 44, no. 7, pp. 848–864, Jul. 1995.

[15] Y. Fukushima, M. Fukushi, and S. Horiguchi, "Fault-tolerant routing algorithm for network on chip without virtual channels," in *Proc. 24th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2009, pp. 313–321.

[16] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proc. Int. Symp. Comput. Archit.*, 1992, pp. 278–287.

[17] G. Chiu, "The odd-even turn model for adaptive routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 729–738, Jul. 2000.

[18] A. DeOrio, D. Fick, V. Bertacco, D. Sylvester, D. Blaauw, J. Hu, and G. Chen, "A reliable routing architecture and algorithm for NOCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 5, pp. 726–739, May 2012.

[19] N. E. Jerger and L.-S. Peh, "On-chip networks," *Synthesis Lectures Comput. Archit.*, vol. 4, no. 1, pp. 1–141, 2009.

[20] Z. Zhang, A. Greiner, and S. Taktak, "A reconfigurable routing algorithm for a fault-tolerant 2d-mesh network-on-chip," in *Proc. 45th ACM/IEEE Design Autom. Conf.*, 2008, pp. 441–446.

[21] M. D. Schroeder, A. D. Birrell, M. Burrows, H. Murray, R. M. Needham, T. L. Rodeheffer, E. H. Satterthwaite, and C. P. Thacker, "Autonet: A high-speed, self-configuring local area network using point-to-point links," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 8, pp. 1318–1335, Oct. 1991.

[22] D. Xiang and J. Han,"Multiple spanning tree construction for deadlock-free adaptive routing in irregular networks," in *Proc. 10th Int. Symp. Parallel Distrib. Process. Appl.*, 2012, pp. 9–16.

[23] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 4, no. 12, pp. 1320–1331, Dec. 1993.

[24] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 6, no. 10, pp. 1055–1067, Oct. 1995.

[25] M. Gomez, N. Nordbotten, J. Flich, P. Lopez, A. Robles, J. Duato, T. Skeie, and O. Lysne, "A routing methodology for achieving fault tolerance in direct networks," *IEEE Trans. Comput.*, vol. 55, no. 4, pp. 400–415, Apr. 2006.

[26] V. Puente and J. A. Á, "Immucube: Scalable fault-tolerant routing for k-ary n-cube networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 776–788, Jun. 2007.

[27] M. H. Neishaburi and Z. Zilic, "Eravc: Enhanced reliability aware NOC router," in *Proc. 12th Int. Symp. Quality Electron. Design*, 2011, pp. 1–6.

[28] R. Parikh and V. Bertacco, "udirec: Unified diagnosis and reconfiguration for frugal bypass of NOC faults," in *Proc. 46th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2013, pp. 148–159.

[29] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: A reliable network for unreliable silicon," in *Proc. 46th Annu. Design Autom. Conf.*, 2009, pp. 812–817.

[30] S. Murali, D. Atienza, L. Benini, and G. De Michel, "A multi-path routing strategy with guaranteed in-order packet delivery and fault-tolerance for networks on chip," in *Proc. 43rd Annu. Design Autom. Conf.*, 2006, pp. 845–848.

[31] L. Levitin, M. Karpovsky, and M. Mustafa, "Minimal sets of turns for breaking cycles in graphs modeling networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 9, pp. 1342–1353, Sep. 2010.

[32] P. Ren, Q. Meng, X. Ren, and N. Zheng, "Fault-tolerant routing for on-chip network without using virtual channels," in *Proc. 51st Annu. Design Autom. Conf. Design Autom. Conf.*, 2014, pp. 1–6.

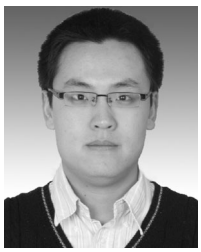
[33] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2009.

[34] W. Dally and C. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 100, no. 5, pp. 547–553, May 1987.

[35] P. Ren, M. Lis, M. H. Cho, K. S. Shim, C. W. Fletcher, O. Khan, N. Zheng, and S. Devadas, "Hornet: A cycle-level multicore simulator," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 31, no. 6, pp. 890–903, Jun. 2012.

[36] C. Grecu, A. Ivanov, R. Saleh, E. S. Sogomonyan, and P. P. Pande, "On-line fault detection and location for NOC interconnects," in *Proc. 12th IEEE Int. On-Line Testing Symp.*, 2006, pp. 145–150.

[37] D. U. U, "Efficient microarchitecture for network-on-chip routers," Ph.D. dissertation, Stanford Univ., Stanford, CA, USA, 2012.



Pengju Ren received the PhD degree in electrical engineering from Xi'an Jiaotong University in 2012. During 2009 to 2011, he was a visiting PhD student in Computer Science and Artifical Intelligence Laboratory, Massachusetts Institute of Technology (MIT). He is an assistant professor in the School of Electronic and Information Engineering, Xi'an Jiaotong University. His current research interests include On-Chip network, scalable many-core designs, and VLSI architecture for digital video processing. He is a member of the IEEE.

Xiaowei Ren is a graduate student in the School of Electronic and Information Engineering, Xi'an Jiaotong University. His current research interests include scalable many-core designs and advanced signal processing.

Sudhanshu Sane received the undergraduation degree from the University of Pune, India. He is a graduate student in the Department of Computer and Information Science, University of Oregon and is a member of the Computer Architecture and Embedded Systems (CAES) Laboratory. His research interests include intelligent network-on-chip designs, routing algorithms, and computer systems security.

Michel A. Kinsy received the BSE degree in computer systems engineering, the BS degree in computer science, both from Arizona State University and the MS degree in electrical engineering and computer science from MIT. He received the PhD degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT) in 2013. He is an assistant professor in the Department of Computer and Information, University of Oregon and the director of the Computer Architecture and Embedded Systems (CAES) Laboratory. He is an MIT Presidential fellow. His research interests include cognitive high-performance many-core architectures, self-aware adaptive multicore systems, intelligent network-on-chip designs, hardware and embedded systems security, and cyber-physical systems. He is a member of the IEEE.

Nanning Zheng received the graduation degree from the Department of Electrical Engineering, Xi'an Jiaotong University, Xi'an, China, in 1975, and the PhD degree in electrical engineering from Keio University, Yokohama, Japan, in 1985. He is currently a professor and the director of the Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University. His research interests include computer vision, pattern recognition, machine vision and image processing, neural networks, and hardware implementation of intelligent systems. He became a member of the Chinese Academy of Engineering in 1999, and he is the Chinese Representative on the Governing Board of the International Association for Pattern Recognition. He also serves as an executive deputy editor of the Chinese Science Bulletin. He is a fellow of the IEEE.