Design Methodology of Fault-Tolerant Custom 3D Network-on-Chip

KATHERINE SHU-MIN LI, National Sun Yat-Sen University
SYING-JYAN WANG, National Chung Hsing University

A systematic design methodology is presented for custom Network-on-Chip (NoC) in three-dimensional integrated circuits (3D-ICs). In addition, fault tolerance is supported in the NoC if extra links are included in the NoC topology. In the proposed method, processors and the communication architecture are synthesized simultaneously in the 3D floorplanning process. 3D-IC technology enables ICs to be implemented in smaller size with higher performance; on the flip side, 3D-ICs suffer yield loss due to multiple dies in a 3D stack and lower manufacturing yield of through-silicon vias (TSVs). To alleviate this problem, a known-good-dies (KGD) test can be applied to ensure every die to be packaged into a 3D-IC is fault-free. However, faulty TSVs cannot be tested in the KGD test. In this article, the proposed method deals with the problem by providing fault tolerance in the NoC topology. The efficiency of the proposed method is evaluated using several benchmark circuits, and the experimental results show that the proposed method produces 3D NoCs with comparable performance than previous methods when fault-tolerant features are not realized. With fault tolerance in NoCs, higher yield can be achieved at the cost of performance penalty and elevated power level.

CCS Concepts: • **Hardware** → **Network on chip**; *3D integrated circuits*; *Fault tolerance*

Additional Key Words and Phrases: Fault tolerance, Three-dimensional integrated circuits (3D-IC), KGD test, Network-on-Chip (NoC),

ACM Reference Format:
Katherine Shu-Min Li and Sying-Jyan Wang. 2017. Design methodology of fault-tolerant custom 3d network-on-chip. ACM Trans. Des. Autom. Electron. Syst. 22, 4, Article 63 (May 2017), 20 pages.
DOI: http://dx.doi.org/10.1145/3054745

1. INTRODUCTION

The Network-on-Chip (NoC) [Dally et al. 2005; Feero et al. 2005; Numi 2005; Marculescu et al. 2009] is a packet-switched networking infrastructure for on-chip communication. In NoC, data are transferred by routers in the packet-switched network. NoC greatly improves the scalability of systems-on-a-chip (SoCs) and achieves higher power efficiency compared to other types of communication structures.

In a NoC-based system, processing elements (PEs) such as processors, application-specific integrated circuits (ASIC), intellectual properties, and memories exchange data through the NoC, which consists of network components including data links, network interfaces (NI), and routers. Links provide communication media on which data are transmitted. The NI is the interface between a PE and a router, and it is responsible

This work is supported by the Ministry of Science and Technology, Taiwan, under grant 104-2220-E-110-001 and grant 104-2220-E-110-007.
Authors' addresses: K. S.-M. Li, Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan; S.-J. Wang, Department of Computer Science, National Chung Hsing University, Taichung, Taiwan.

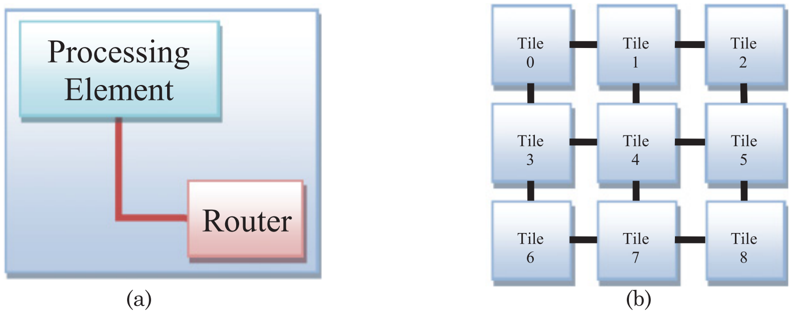

Fig. 1. Regular topology: (a) tile and (b) mesh.

for transforming data into packets or vice versa. The physical transmission path for a packet is set up by routers. The topology of NoC is determined by the physical connections of links and routers, while connection paths for packets are selected by the routing algorithm.

NoC topologies can be regular or irregular. If each PE is mapped to a router, then the PE and its assigned router are combined as a tile, as shown in Figure 1(a). In a regular topology, tiles are connected into a regular structure. Examples of regular structure include mesh (Figure 1(b)), torus, or hypercube. A router is usually connected to other routers in neighboring tiles in addition to the PE in the same tile.

An irregular topology can be a reduced regular structure constructed to deal with the placement of PEs with diversified sizes and aspect ratios. On the other hand, a custom topology is synthesized specially for given design goals. A custom NoC no longer consists of tiles, as routers are allocated according to the communication demand. In this article, we will focus on the problem of irregular topology generation for custom NoC.

Regular topologies are more appealing for applications in which communication demands are not known *a priori*, since in these applications the network architecture should be able to support all possible communication requests. On the other hand, regular topologies may lead to overdesigned systems with redundant network components as they cannot be optimized for specific communication demands. As a result, the overall system performance could be adversely affected by the larger circuit size and power consumption.

For specific applications, NoC topologies can be customized according to the communication demand to achieve optimal performance with reduced circuit size and power consumption [Jalabert et al. 2004; Benini 2006]. Many custom topology generation methods have been developed [Murali et al. 2006; Srinivasan et al. 2006a; Pinto et al. 2009].

As SoC designs become increasingly complex, three-dimensional integrated circuit (3D-IC) [Topol et al. 2006] technology provides a way to further improve SoC performance and integration density [Topol et al. 2006]. Specially, 3D stacked ICs based on through-silicon vias (TSVs) have attracted great attention. Advanced processing technology enables TSVs to be deployed with high density and short vertical distance. As a result, the overall routing lengths of a SoC implemented with 3D-IC can be much shorter than its 2D counterpart. A previous study showed that overall wire length in a 3D-IC can be 15% shorter than a 2D implementation, while 10% power reduction can be achieved due to the reduced wire length [Davis et al. 2001]. Besides, TSV-based 3D-IC technology offers other advantages, including providing higher communication bandwidth and making possible the integration of heterogeneous devices and systems.

Studies [Loi et al 2011; Miyakawa 2009] based on three real 3D-IC cases [Miyakawa 2009; Topol et al. 2005; Swinnen et al. 2006] indicate that the lower manufacturing yield of TSVs makes the overall yield of 3D-ICs worse than 2D ICs. Therefore, it may be difficult to popularize 3D-IC technology unless we can alleviate the yield problem. In particular, when only known-good-dies (KGD) are bonded, the overall chip yield will be solely decided by the TSV yield. One way to improve yield of 3D-ICs is to provide fault tolerance in the manufactured ICs, so the circuits can operate even if TSV faults exist. In this article, we present a methodology to design fault-tolerant NoC with custom topology; this method is called 3D-NoC-*FT* henceforth.

2. MOTIVATION AND PREVIOUS WORKS

In this article, we present a methodology to design low-power NoCs with custom topology targeted for 3D-IC environment. Besides, a fault-tolerant routing mechanism is provided to deal with faulty links among routers in the NoC.

Advances in processing technology allow multimillion transistor chips. SoC enables a wide range of applications using massive parallel processing in the real-time environment, which requires the support of underlying communication structure. There are many ways to implement communication structures in an SoC, including bus, point-to-point links, and on-chip networks. Although networks are more complicated, they are more scalable and achieve higher bandwidth. Besides, the modular router design and standard network interface actually make the design of a network easier than traditional designs. NoC is an on-chip packet-switched network infrastructure; a survey of NoC technology can be found in Bjerregaard et al. [2006].

Regular topologies are better choices for general-purpose systems where traffic conditions are not predictable, and they are also easy to design. However, they need more network components for ASICs with special communication demands. Therefore, the chip becomes larger and consumes more power with longer latency. These problems become more prominent as chips become larger. In some application-specific designs with specific communication demands, using regular topologies may lead to overdesigned systems [Murali et al. 2006; Srinivasan et al. 2006b] with redundant network components. As a result, the overall system performance will be adversely affected by the larger circuit size and power consumption.

Many custom NoC design methods have been developed [Benini 2006; Murali et al. 2006; Srinivasan et al. 2006a; Pinto et al. 2009; Jeang et al. 2005; Srinivasan et al. 2005; Srinivasan et al. 2006b; Jeang et al. 2008; Kim et al. 2008; Holsmark et al. 2008; Jovanovic et al. 2009; Kahng et al 2009; Li 2013].

NoC topology generation for 3D NoC designs has been studied only recently [Loi et al. 2011; Murali et al. 2009; Chen et al. 2009; Matsutani et al. 2013; Yeh et al. 2013; Zhou et al. 2010; Ghiribaldi et al. 2014]. An optimal mesh-based NoC topology is presented in Murali et al. [2009], and it is shown that the 3D design achieves lower power consumption and delay than its conventional counterparts. A de Bruijn graph–based 3D NoC architecture was presented in Chen et al. [2009]. Application-specific 3D NoC synthesis methods have been developed [Zhou et al. 2010; Ghiribaldi et al. 2014]. It was reported that further improvement of delay and power consumption could be achieved through a custom 3D topology called 3D-SAL-FP [Zhou et al. 2010].

The correct operations of NoCs will be affected when faults occur. In this article, we will focus on permanent faults caused by physical defects in the circuit. Defects may be introduced in the manufacturing process, or they may appear after a period of time of use. In particular, the lower yield of TSVs can be a major source of defects in the 3D-ICs [Loi et al. 2011]. Both routers and links can be affected by faults, and it is assumed that the faulty router/link can no longer be used.

Since 3D packaging is much more expensive than its 2D counterpart, it is important to ensure the high yield of manufactured 3D-ICs. To achieve this goal, one needs to ensure that only KGD are included in a 3D stack, and this can be done if the dies are tested thoroughly. On the other hand, TSVs are not tested in the KGD test, so faulty TSVs will make manufacturing yield lower unless the faulty TSVs can be tolerated.

There are several ways to improve the yield loss due to faulty TSVs. One possible way is to employ more robust TSVs without changing the network structure. A fault-tolerant implementation of 3D NoCs that employs multiple TSVs for each signal net is presented in Loi et al. [2011] to improve yield and reliability of 3D-ICs.

Faulty TSVs can also be tolerated by providing fault-tolerant routing in 3D NoC with regular topology. Fault-tolerant routing algorithms for NoCs try to establish paths in the presence of faults [Jovanovic et al. 2009; Fick et al. 2009]. Many fault-tolerant schemes for mesh-based 2D NoCs have been developed to tolerate faults in links and routers [Seyrafi et al. 2010; Park et al. 2006; Bogdan et al. 2007; Bogdan et al. 2011]. Fault tolerance in regular 3D NoC structures has also been studied [Rad et al. 2010; Pasca et al. 2010; Rahmani et al. 2011]. Since mesh-based NoCs support multiple routing paths between a source and a destination, in previous methods [Fick et al. 2009; Seyrafi et al. 2010; Park et al. 2006; Bogdan et al. 2007; Bogdan et al. 2011; Rad et al. 2010; Pasca et al. 2010; Rahmani et al. 2011] faults in routers and links can be tolerated through fault-tolerant routing schemes.

Unfortunately, fault-tolerant routing cannot be carried out in custom 3D NoCs. Custom NoCs provide better performance with lower cost than regular NoC structures when communication demands are known. However, since the communication fabrics are tailor-made according the demands, normally no alternative paths exist from a source to a destination. As a result, fault tolerance in custom 3D NoCs can only be achieved through topology modification.

In this article, we propose to achieve fault tolerance in custom 3D NoCs by providing redundant paths in the network, as normally only one path is available in a non-fault-tolerant custom NoC. A possible way to achieve fault tolerance is to provide multiple NIs to each PE; this approach, however, requires a higher hardware cost. Instead, in this article, we will focus on providing extra routing paths to achieve fault tolerance. We will present a custom NoC topology generation method that synthesizes fault-tolerant 3D NoCs. The proposed design method provides the following distinct features.

—A new design flow for fault-tolerant custom 3D NoCs is presented. With known communication demands, custom NoCs achieve better performance/cost than NoCs with regular topologies. Employing 3D-IC topology in the proposed fault-tolerant NoC architecture further improves performance with reduced power consumption.
—Fault tolerance in the NoCs is achieved by providing multiple paths between each source-destination pair. As a result, all single faults, including link and router faults, can be tolerated if the faulty TSVs are located through 3D-IC interconnect test. Multiple faults can also be tolerated unless a PE is isolated from one of its source/destination due to the faults.
—The lower manufacturing yield of TSVs may significantly increase the cost of 3D-ICs. The proposed fault-tolerant architecture can tolerate faulty TSVs so the overall yield of 3D-ICs is improved.

3. PROBLEM DESCRIPTION

3.1. Problem Formulation

A system is usually represented by a block diagram. For example, Figure 2(a) is a block diagram of benchmark "vopd." An arc (i.e., directed edge) indicates the direction of data transmission. The edge weight on an arc represents the communication requirement of

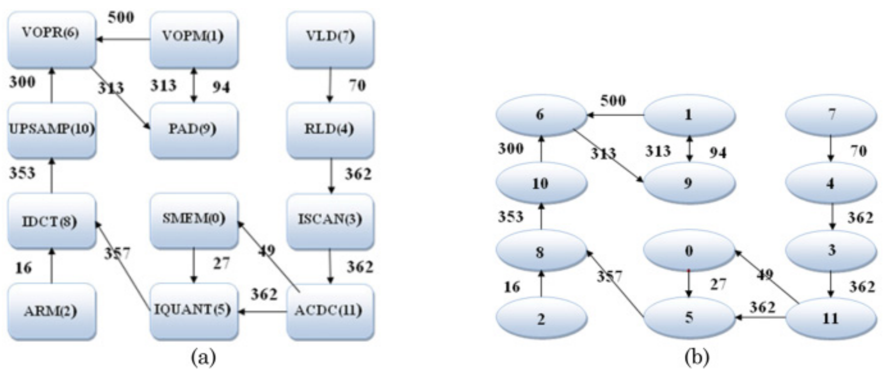

Fig. 2. Benchmark example "vopd": (a) block diagram and (b) communication trace graph (CTG).

this arc with the unit of MB/s. A block diagram is usually represented by a simplified Communication Trace Graph (*CTG*). The CTG of the block diagram given in Figure 2(a) is shown in Figure 2(b).

Definition 3.1 (*CTG*). The CTG of a block diagram is a graph $CTG = (V_G, E_G)$, where each $v \in V_G$ represents a block and each $e \in E_G$ is an edge in the block diagram. In the CTG, $w(e)$ stands for the communication demand of edge e.

Given a CTG and the input/output port limits of routers, the proposed design method, called 3D-NoC-FT, generates a custom 3D NoC topology with the corresponding system-level floorplan. A three-phase procedure is used in 3D-NoC-FT, with Phases I and II generating the NoC topology and Phase III taking both processing elements and routers for floorplanning. The inputs and outputs of the proposed scheme are described as follows, while the design procedure will be described later.

Input:

(1) CTG.
(2) Limitation of the number of input (r_i) and output ports (r_o) for a router. The number of input (output) ports must be less than or equal to r_i (r_o). A router with m input ports and n output ports is denoted as $R_{m\times n}$.
(3) The number of tiers in the ASIC design.

Output:

A fault-tolerant 3D NoC topology is generated in Phases I through II, and Phase III transforms the network topology into a system-level floorplan of the custom NoC.

3.2. Notations and Definitions

Definition 3.2 (*Router Bipartite*). A *Router Bipartite* $m\times n$ ($RB_{m\times n}$) is a complete bipartite graph representation of $R_{m\times n}$, which is a router providing crossbar connections between m input ports and n output ports. Source and destination nodes in $RB_{m\times n}$ represent input and output ports in $R_{m\times n}$. $RB_{m\times n} = (S_R, T_R, A_R)$, where S_R and T_R are the input port set and output port set and A_R is the set of arcs between S_R and T_R.

Definition 3.3 (*Communication Trace Bipartite Graph (CTB)*). A $CTB = (S_B, T_B, A_B)$ is a bipartite graph representation of a $CTG = (V_G, E_G)$, where $S_B = \{s_i \,|\, (v_i, v_j) \in E_G\}$, $T_B = \{t_j \,|\, (v_i, v_j) \in E_G\}$, $A_B = \{(s_i, t_j) \,|\, s_i \in S_B, t_i \in T_B, (v_i, v_j) \in E_G\}$.

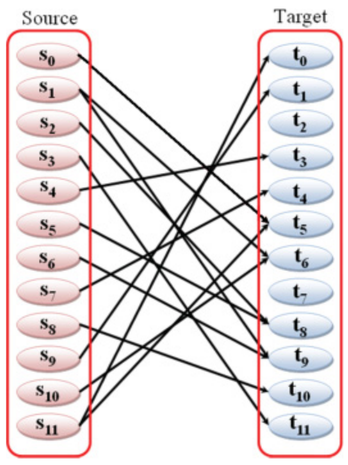

Fig. 3. The communication trace bipartite graph (CTB) of “vopd.”

Figure 3 gives the CTB of the example in Figure 2(b). Since each core can be a source in one communication and a target in another, it appears in both sides of the bipartite graph. For example, both s_1 and t_1 correspond to VOPM (core 1) in Figure 2.

4. TOPOLOGY GENERATION ALGORITHM: 3D-NOC-FT

Previous studies show that custom NoC can achieve better performance with lower circuit size and power consumption for specific application with known communication demand. However, the lack of alternate routes in custom NoC creates problems when faults occur. This is especially true in the case of 3D-ICs where the yield of TSVs may not be satisfactory. To solve the problem, we propose a design flow targeted for 3D custom NoC with (1) a minimum number of TSVs and (2) fault tolerance capability. The process consists of three phases as follows:

(I) 3D Partition and Tier Assignment
(II) Generating Fault-Tolerant Topology
(III) Wire-Length Driven Floorplanning

The design procedure in Phase I is targeted to achieve the first goal, a 3D custom NoC with a minimized number of TSVs. Based on the optimized 3D partition, the second goal (i.e., fault-tolerant topology generation) is achieved in Phase II. Phase III is employed to produce a floorplan so circuit parameters (size, power, etc.) can be evaluated.

4.1. 3D Partition and Tier Assignment

In Phase I, a circuit specification is partitioned and transformed into a 3D structure consisting of multiple tiers. The goal in this step is to reduce the total number of TSVs that connect components in different tiers. To achieve this goal, cores with communication demands from/to each other will likely be placed in the same tier or adjacent tiers; as a result, power can be minimized as well. The restriction is that all Input/Output (I/O) ports are placed in the bottom tier. In order to achieve this goal, the first step is to partition a circuit so the number of inter-partition communication links is minimized. The task is carried out in three steps: (1) Constructing CTB from a given CTG, (2) Min-cut based CTB partitioning, and (3) Tier assignment.

In the proposed approach, the goal of Phase I is to minimize the number of TSVs. We can always employ another partition scheme in which other goals, including power, temperature, area, and so on, are considered in the optimization process. Since Phase I and Phase II are independently executed tasks, employing a different algorithm in Phase I will not affect the execution of Phase II, although the results might differ.

4.1.1. Constructing CTB. The communication requirement of a system is specified by a CTG. The first step of custom NoC design is to transform the CTG to a CTB, from which routers are constructed. An example of the CTG-to-CTB transformation process is given in Li [2013].

4.1.2. CTB Partitioning. A 3D-IC consists of multiple tiers, where each tier contains some modules of the system. The number of TSVs can be reduced if the amount of inter-tier communication is minimized, and this goal can be achieved by partitioning the CTB such that inter-partition communication is minimized. Each partition will later be assigned to a tier in the 3D-IC.

In this article, we apply the max-flow min-cut partitioning algorithm [Cormen et al. 2001] to carry out CTB partition. The algorithm is outlined in Algorithm 1 below. The goal is to partition a CTB into n partitions. In each iteration, a set of vertices with min-cut with respect to the other vertices in the graph is selected as a new partition. This set of vertices is removed from the graph and the process is repeated until n partitions are formed. We impose a constraint that the sizes of all partitions are roughly balanced in the process. The complexity of the max-flow min-cut algorithm is O(|V|·|A|2), where |V| and |A| are the number of vertices and edges in CTB.

ALGORITHM 1: Max-Flow Min-Cut Based Partition

Input: *CTB (S, T, A).*
Output: Partition 0 to Partition n–1.

```
set $f \leftarrow 0$; //flow 0 on all edges
set opt ← false;
while NOT Opt do
  construct the residual graph $G_f$;
  find a directed path $P$ from $S$ to $T$ in $G_f$;
  //$P$: an augmenting path
  if $P$ exists then
    update flow $f$ along $P$;
  else
    set Opt ← TRUE;
    $X \leftarrow$ the set of vertices in $G_f$ reachable from $S$;
  end if
end while
return $f$, $(X, V-X)$;
// $f$ as the max flow, $(X,V-X)$ as the min-cut
```

4.1.3. Tier Assignment. Once the partitions are known, the next step is to determine the order to partition in the 3D structure, with the constraint that the partition containing I/O pins must be placed in the bottom tier.

Assume that a circuit is partitioned into a set of n partitions $P = \{P_0, P_1, \ldots, P_{n-1}\}$. These partitions will be organized as an n-tier 3D structure $T_0, T_1, \ldots, T_{n-1}$, where T_0 is the tier at the bottom and T_{n-1} is on the top. The task in this step is to find a one-to-one mapping from $\{P_0, P_1, \ldots, P_{n-1}\}$ to $\{T_0, T_1, \ldots, T_{n-1}\}$ such that the overall TSVs will be minimized. The tier assignment can be treated as a permutation of the n–1 partitions, as the partition containing I/O pins has to be assigned to the bottom

Table I. Connectivity Matrix

	P_0	P_1	P_2	P_3
P_0		20	30	40
P_1	10		40	30
P_2				
P_3	20	40	50	

tier. For a large n, a heuristic tier assignment scheme should be used due to the large number of permutations.

The total number of inter-partition links is not equal to the number of TSVs, as a link may consist of multiple TSVs if the two partitions are not placed in adjacent tiers. For example, when two partitions are placed at T_0 and T_{n-1}, any link between the two partitions must be realized by n–1 TSVs. On the other hand, if the two partitions are placed at adjacent tiers, each inter-partition link is realized by one TSV. In general, if two partitions are placed at P_i and P_j, then an inter-partition link must be implemented by connecting $|i–j|$ TSVs in series. In other words, the weight of a link depends on the relative positions of these two partitions.

Once a partition is placed at a certain tier, partition P_i is placed at tier T_j, and T_j is essentially equivalent to P_i.

Definition 4.1 (*Weighted TSV*). Assume that a partition P_k is placed at tier T_i. The Weighted TSV, denoted as $WTSV(P_k, i)$, is the total number of TSVs required to connect a partition P_k to those i partitions placed in tiers $T_0, T_1, \dots, T_{i-1}$.

$WTSV(P_k, i)$ can be expressed by the following equation:

$$WTSV(P_k, i) = \sum_{j=0}^{i-1} Conn(P_k, T_j) \times (i - j), \tag{1}$$

where $Conn(P_k, T_j)$ is the number of connection links between partition P_k and the partition in tier T_j.

The proposed tier assignment scheme is preceded as follows. First, the partition that contains I/O pins is assigned to T_0. Assume that the number of assigned partitions is i, and let the set of unassigned partitions be UP. Partition $P_k \in UP$ will be assigned to tier T_i if the following condition is satisfied:

$$\forall P_j \in UP \wedge j \neq k \Rightarrow WTSV(P_k, i) > WTSV(P_j, i). \tag{2}$$

In other words, the partition in UP with the maximum weighted TSV will be assigned to the tier to be processed.

An example of the tier assignment is given as follows. A system is partitioned into four parts $\{P_0, P_1, P_2, P_3\}$, in which P_2 contains I/O pins. The connections among partitions are summarized in Table I. Partition P_2 is first assigned to T_0, so $WTSV(P_0, 0) = 30 \times (1 - 0) = 30$, $WTSV(P_1, 0) = 40$, and $WTSV(P_3, 0) = 50$. Therefore, P_3 is assigned to T_1. In the next step, P_1 is assigned to T_2, and thus P_0 is left on top of the 3D stack.

The complexity of the above procedure is a $\mathbf{O}(n^2)$, as the WTSV of each unassigned partition has to be computed in each iteration. On the other hand, if exhaustive search is carried out, $(n–1)!$ different configurations have to be evaluated. The heuristic approach works well in practice, as in our experiment both methods achieve the same result.

4.2. Generating Fault-Tolerant Topology

A fault-tolerant 3D NoC topology will be generated in Phase II. The synthesized NoC should satisfy the following two goals:

—An upper limit on communication bandwidth is assigned to each router to reduce routing congestion.
—The NoC architecture can tolerate a single-link fault.

The above two major tasks are carried in this phase. In the first step, routers in each tier are established by dividing communication requirements of all PEs in a tier, where the communication requirement in single a group is supported by a router. Next, a single-link fault-tolerant NoC topology based on a de Bruijn graph model is generated. Details of these steps are discussed in this section.

4.2.1. Creating Routers. In the first step, routers connected to all PEs in each tier are established by applying Bin-Packing algorithm [Vazirani 2003] to partition the communication requirements of all PEs into groups, where communication in each group is supported by a router.

PEs are connected to routers in a NoC-based system. Let the communication bandwidth requirement of PE$_i$ be denoted as BW_i. To restrict the communication bandwidth associated with a router, an upper bound on router bandwidth MB is set on each router. A set S of PEs may share a router as long as overall communication requirement in S is less than MB:

$$\sum\nolimits_{PE_i \in S} BW_i \leq MB. \tag{3}$$

In case the communication bandwidth requirement of PE_i is greater than MB, at least m_i routers should be provided for PE_i, where

$$m_i = \left\lceil {}^{BW_i}/_{MB} \right\rceil. \tag{4}$$

This problem can be solved by applying Bin-Packing [Vazirani 2003] algorithm as follows. Let the bin capacity (i.e., the upper bound of router bandwidth) be MB. The goal is thus to support PEs' communication bandwidths with the minimum number of routers, which is analogous to pack all the communication bandwidths into the minimum number of bins with fixed size.

Assume that there are n PEs in a given tier. Let the PEs be labeled as PE_i, $0 \leq i < n$, and their communication bandwidths are BW_i, $0 \leq i < n$, respectively. First, for each PE_i, BW_i is partitioned into m_i bandwidth elements according to Equation (4). Note that $m_i = 1$ *if* $BW_i < MB$. In this case, the bandwidth is not partitioned. Otherwise BW_i is partitioned into m_i–1 elements of size MB and one element of size $BW_i - (m_i - 1) \times MB$ (i.e., the remaining bandwidth).

Once the above procedure is done, there will be $M = \sum_{i=0}^{n-1} m_i$ bandwidth elements. The Bin-Packing algorithm is applied to partition the M bandwidth elements into groups, where condition (3) is true for all bandwidth elements in the same group. PEs associated with bandwidth elements in the same group share the same router, and communications among these PEs are processed by the router. An example is shown in Figure 4.

Note that if a PE is connected to multiple routers due to the partitioned communication demand, essentially the NI of PE has to carry out a router's function, which will somewhat complicate the NI design.

The Bin-Packing problem is known to be NP-hard, so a heuristic approach is used to solve the problem. The bandwidth elements are first sorted in descending order of their sizes, and then the first-fit algorithm is applied. Each element is placed in the first bin in which it fits. The time complexity is O(M·log$M$), where M is the number of bandwidth elements.

4.2.2. Establishing Fault-Tolerant NoC Topology. In the second step, links among routers are established. To make the custom NoC topology fault tolerant, extra intra-tier links

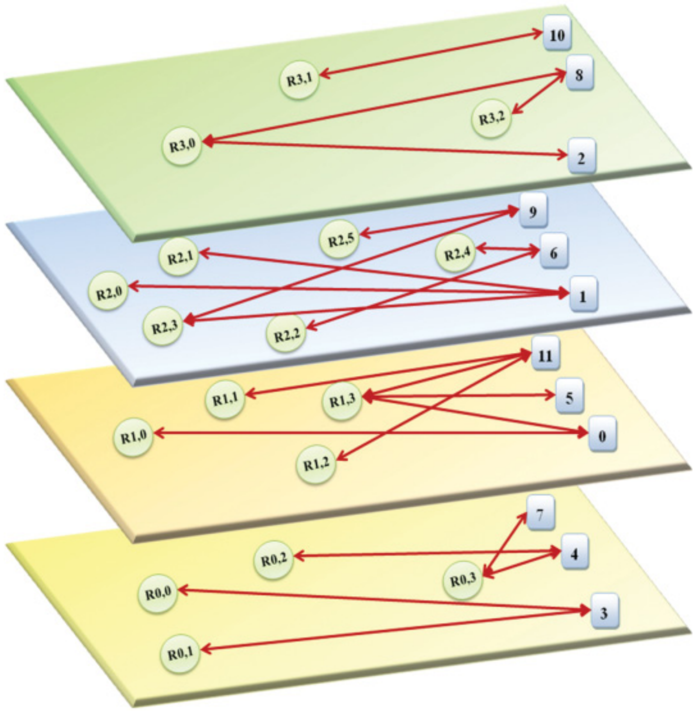

Fig. 4. Establish routers in each tier.

are provided among routers so an alternative route can be found if a single link is faulty. The detailed synthesis procedure is discussed as follows.

First, a fault-tolerant topology is derived from generalized de Bruijn graph (DBDG) [Imase et al. 1981; Imase et al. 1983]. The reason is that a generalized DBDG provides multiple paths between any each pair of vertices and its diameter is less than or equal to that of any previously proposed graph. As a result, we can apply generalized DBDG to construct redundant links in the proposed fault-tolerant NoC topology so the lengths of the alternative routes are minimized. In other words, this approach ensures that a fault-tolerant topology is achieved while the number of hops required for routing is minimized.

Definition 4.2 (*Generalized de Bruijn Digraph*). A generalized de Bruijn digraph $G_I(d, n)$ is a directed graph with n nodes and d input/output ports. Label the n nodes of the generalized de Bruijn digraph $G_I(d, n)$ as $0, 1, 2, \ldots, n-1$. Node i has a directed link to node j if and only if

$$j = d \times (n - 1 - i) + (\text{mod } n), 0 \le r \le d - 1 \tag{5}$$

The diameter m of $G_I(d, n)$ satisfies [Imase et al. 1983]:

$$m \le \lceil \log \mathrm{dn} \rceil \tag{6}$$

The optimal routing algorithm for generalized de Bruijn digraphs has been developed in Liu et al. [1993]. In this method, a control tag t based on the source node u and the destination node v is generated using depth-first search, and the tag will lead to a shortest path from u to v in $G_I(d, n)$.

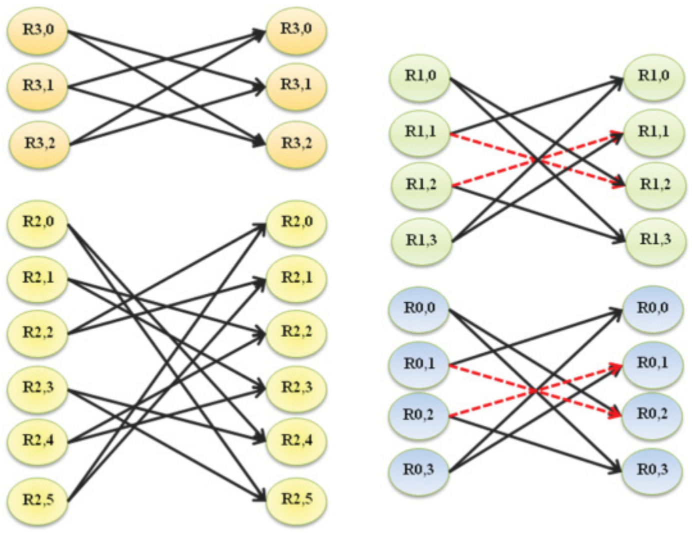

Fig. 5. Intra-die connections with a generalized de Bruijn digraph.

To tolerate single-link fault, connections among routers in the same tier are constructed with a generalized de Bruijn digraph $G_I(2, n)$. In this way, each router is equipped with two incoming links and two outgoing links. According to Equation (6), all routers in the same tier can be reached in three hops for $n \leq 8$. The intra-tier inter-router connections corresponding to the router structure in Figure 4 are given in Figure 5.

Unfortunately, a generalized de Bruijn digraph may contain a self-loop, which cannot provide an alternative route. To deal with this problem, we select a pair of routers with self-loops and swap their destinations. For example, four routers exist in both tier 0 and tier 1 in Figure 5; thus, a $G_I(2, 4)$ is constructed in each tier. There are self-loops associated with node 1 and node 2 in $G_I(2, 4)$ according to Equation (5); therefore, routers $R_{1,1}$ and $R_{1,2}$ will have self-loops (i.e., links $R_{1,1} \rightarrow R_{1,1}$ *and* $R_{1,2} \rightarrow R_{1,2}$) if a generalized de Bruijn digraph $G_I(2, 4)$ is deployed. In order to provide alternate routes, the destinations of this pair of links are swapped (i.e., links $R_{1,1} \rightarrow R_{1,2}$ *and* $R_{1,2} \rightarrow R_{1,1}$). In Figure 5, intra-tier links shown in broken red lines are the swapped connections. Similarly, two links in tier 0 are modified in the same way. Note that the diameter of the modified de Bruijn digraph is smaller than or equal to that of the original graph (DBDG), as in this case more routers can be reached in the same number of hops.

The next step is to construct inter-tier connections among routers. The custom communication fabrics are constructed according to the communication requirement among PEs. For example, assume that there is a communication link from PE_i to PE_j, where PE_i and PE_j locate in different tiers. In this case, a communication link is established from the router associated with PE_i to the router associated with PE_j.

The network topology synthesized by the above procedure is a fault-tolerant 3D NoC architecture; hence, it is referred to as 3D-NoC-FT. The routing paths in 3D-NoC-FT are determined by finding the shortest paths among all pairs of communication links using the Floyd-Warshall algorithm [Lian et al. 2009]. The fault-tolerant routing is carried out by employing look-up tables in the routers, which increases the latency of the router. However, the overall performance impact due to the fault-tolerant architecture should scale well as the number of cores increases. First, a fault-tolerant topology is developed individually in each tier rather than globally. Since the size of a tier should be limited due to the 3D-stacking technology, the overhead in each tier should not be affected by the overall circuit size. Besides, a core tends to communicate with only a small number of other cores, as shown in Figure 2, regardless of the total number of cores. In a custom NoC topology, only a limited number of routers are involved for any subset of cores. For these reasons, the performance and area overhead due to the fault-tolerant architecture is limited, as we will see from the experimental results later.

The main difference between 3D-NoC-FT and the non-fault-tolerant version, 3D-NoC, is the intra-die connections that realize the generalized de Bruijn digraphs. In case there is a faulty TSV connecting router $R_{i,j}$ to $R_{i+1,k}$, the intra-die connections provide an alternative path from $R_{i,j}$ to $R_{i,l}$, from which a fault-free TSV connecting $R_{i+1,m}$ is used to establish an inter-die connection. The router connections in tier $i{+}1$ ensure that a correct path can be constructed starting from $R_{i+1,m}$ to the destination.

The generation of redundant links for fault tolerance creates alternate routes from a source to a destination. A problem in the fault-tolerant topology is that deadlocks may occur. To prevent deadlock formation, a deadlock-free router architecture using retransmission buffers has been proposed [Zhou et al. 2010]. Alternatively, deadlock-free routing algorithms [Holsmark et al. 2008; Jovanovic et al. 2009] can be employed to deal with the problem.

4.3. Wirelength-Aware Floorplanning

The goal of this article is to generate fault-tolerant custom NoC topology for 3D-ICs, and the major tasks have been completed in the previous two phases. However, to evaluate the efficiency of the proposed method, it is necessary to generate a floorplan so the circuit performance can be estimated. We apply a fast simulated annealing algorithm [Chen et al. 2008] to get the floorplan of a custom NoC. To balance the chip area and wirelength, the cost function is set as follows:

$$Cost = \alpha \times Area + \beta \times WL \tag{7}$$

Area and WL in Equation (7) are the estimated chip area and wirelength of a floorplan, while α and β are their respective weights. The selection of α and β provides a tradeoff between chip size and total wire length. Since we want to optimize both goals, we set $\alpha = \beta = 0.5$ in our experiment.

In our approach, the floorplan is carried out after the topology is known. A recent study [Ghiribaldi et al. 2014] shows that it is possible to move the floorplanning step to the early phase of the design methodology and achieves good results in 2D ICs.

5. EXPERIMENTAL RESULTS

We conduct the experiments on Ubuntu 8.0.4, with Xeon 2.0GHz processor, 3GB memory. The benchmark circuits and their statistics are summarized in Table II. The benchmark circuits are application-specific SoC designs for video applications and multimedia applications [Murali et al. 2005]. Column 2 gives names of the benchmarks, while columns 3 and 4 give the number of PEs and communications in each circuit. The last column respresents the average number of communications per PE; a larger number implies more complicated communication in a circuit. The designs are synthesized with the 70nm technology file provided by CosiNoC [Pinto et al. 2009], where the area and power consumption of routers are estimated by Orion 2 [Kahng et al. 2009]. Recently, it was reported that the area and power parameters extracted using Orion 2 were too pessimistic [Hayenga et al. 2012]. Nevertheless, since our goal is to compare the relative performance of various custom NoC implementations, the results should still be acceptable.

Table II. Characteristics of Benchmark Circuits

SoC	Description	# Core	# Comm.	Avg. Comm. (# Comm. / # Core)
G1	PIP	8	8	1.00
G2	263 enc mp3 dec	12	12	1.00
G3	MWD	12	12	1.00
G4	mp3 enc mp3 dec	13	13	1.00
G5	263 dec mp3 dec	14	15	1.07
G6	VOPD	12	14	1.17
G7	VOPD + MPEG-4 + MWD	36	52	1.44
G8	VOPD + MPEG-4	24	40	1.67
G9	MPEG-4 + PIP	20	34	1.70
G10	MPEG-4	12	26	2.17
G11	IMP	27	96	3.56

Table III. 2D NoC without Fault Tolerance (3D-NoC with #tier = 1)

Graph	Size (mm^2)	#Hops	Network				Performance		
			#Router	A_R (mm^2)	#Link	WL (mm)	Power (mW)	Latency (ns)	Throughput (packet/s)
G1	2.43	2	5	0.0192	23	7.85	4.40	2.74	3764.12
G2	2.86	2	9	0.0239	31	9.43	4.39	2.66	5585.44
G3	2.64	2	10	0.0289	34	10.61	5.08	2.56	5691.40
G4	2.27	2	11	0.0278	35	9.43	4.07	2.51	6358.70
G5	4.44	2	12	0.0366	43	15.82	5.91	2.82	7016.63
G6	7.66	3	20	0.0594	60	32.26	10.19	2.90	10069.16
G7	13.71	2	62	0.2491	230	149.77	32.97	3.83	34165.83
G8	10.67	3	48	0.2141	194	122.58	28.52	4.04	28742.51
G9	4.98	2	39	0.1627	151	71.38	19.16	3.26	22450.61
G10	2.49	3	36	0.1503	134	45.12	13.94	2.66	18945.47
G11	6.86	3	69	0.3999	357	240.80	50.31	3.38	53304.80
Average	**5.55**	**2.36**	**29.18**	**0.12**	**117.45**	**65.00**	**16.27**	**3.03**	**17826.79**
Norm.	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**

5.1. Analysis of the Fault-Tolerant 3D NoC Architecture

To analyze the effect of fault-tolerant architecture, two versions of 3D NoC architectures have been synthesized for each circuit. One of them (3D-NoC-FT) provides de Bruijn links among routers in the same die for fault tolerance, while no such links are provided in the other version called3D-NoC. In other words, 3D-NoC-FT and 3D-NoC are synthesized in the same procedure, except that the step to construct de Bruijn links in Phase II is not carried out in 3D-NoC.

Table III shows the results of 2D custom NoCs synthesized by 3D-NoC, and Table III gives the results of 2D custom NoCs synthesized by 3D-NoC-FT. In other words, these two tables give the special case of 3D NoCs in which there is only one tier (#tier = 1). In each version of NoC, columns 2 and 3 give general synthesis results, including chip area (Size) and number of hops (#Hops), which is equal to the number of routers along a path plus one for the destination. The next four columns summarize results related to the network structure, including number of routers (#Router), router area (A_R), number of links (#Link), and wirelength of links (WL). The last three columns give power, average latency, and throughput of the NoC. The latency of a packet is the number of clock cycles required to send the whole packet to its destination. The last

Table IV. 2D NoC with Fault Tolerance (3D-NoC-FT with #tier = 1)

Graph	Size (mm^2)	#Hops	Network				Performance		
			#Router	A_R (mm^2)	#Link	WL (mm)	Power (mW)	Latency (ns)	Throughput (packet/s)
G1	2.41	3	10	0.0351	26	9.89	9.68	4.31	3446.30
G2	2.86	3	13	0.0568	42	13.60	9.77	5.16	5052.40
G3	2.66	3	14	0.0592	44	15.73	10.09	5.19	4766.95
G4	2.28	4	15	0.0680	50	17.06	10.20	6.16	5011.14
G5	4.47	3	18	0.0762	56	24.87	12.70	5.73	7068.06
G6	7.65	4	21	0.1132	84	53.29	16.74	7.86	6896.07
G7	13.47	6	68	0.3444	250	212.85	48.43	12.01	21030.62
G8	10.57	6	56	0.2889	212	144.82	35.64	10.38	17812.96
G9	5.02	5	43	0.2303	168	81.60	22.95	8.48	14950.17
G10	2.56	5	37	0.1958	144	58.22	18.37	7.58	12753.04
G11	6.97	7	98	0.4156	310	212.03	48.40	10.96	29381.10
Average	**5.54**	**4.45**	**35.73**	**0.17**	**126.00**	**76.72**	**22.09**	**7.62**	**11651.71**
Norm.	**1.00**	**1.89**	**1.22**	**1.42**	**1.07**	**1.18**	**1.36**	**2.31**	**0.65**

row of Table III and that of Table IV compares normalized results between the two versions of NoCs.

Please note that the chip area (column 2 in Tables III and IV) include PEs and NoC architecture. Since the chip area is dominated by the PEs, the difference between the sizes of 3D-NoC and 3D-NoC-FT is very small, even though the area of network components in 3D-NoC-FT is about 42% larger than the area of network components in 3D-NoC. In principle, the chip area should be positively correlated to the area of network components. However, in many cases the chips synthesized by 3D-NoC are larger than those synthesized by 3D-NoC-FT. The abnormity can be attributed to the fact that 3D-NoC-FT employs a larger number of smaller routers. It is possible that some smaller routers can actually be placed in the unused space so the overall chip area will not increase. In contrast, it is more difficult to place the larger routers used in 3D-NoC in the unused space in a given floorplan.

Since the fault-tolerant architecture is achieved by providing redundant paths, it surely will encounter area and performance penalty, which is evident in Tables III and IV. The numbers of routers are differ, because routers may be further merged in the 3D-NoC. In contrast, routers in 3D-NoC-FT cannot be merged due to the extra inter-router links added for fault tolerance. Besides, the extra links will affect the bandwidth of a router.

Compared with 3D-NoC, the fault-tolerant version 3D-NoC-FT requires 42% more router area and 18% more wirelength. However, since the majority part of the chip area is occupied by the PEs, the impact on the overall chip size in not significant. It can be seen that, for each circuit, overall chips sizes corresponding to the two NoC implementations are very close. On the other hand, the performance impact is more significant, as the power consumption in the network structure increases 36%, while the latency and throughput of NoCs synthesized by 3D-NoC-FT are 2.31× and 0.65× of the NoCs produced by 3D-NoC.

An interesting observation of the performance penalty due to the fault-tolerant topology is that it is relatively independent of the number of cores. This validates our discussion in Section 4.2 that the proposed fault-tolerant topology scales well with increasing number of cores.

Next we analyze how the number of tiers used to implement the 3D-IC affects the NoC. In this experiment, one to five tiers are synthesized in each case, and the average results of all the benchmarks are taken for evaluation. The results are shown

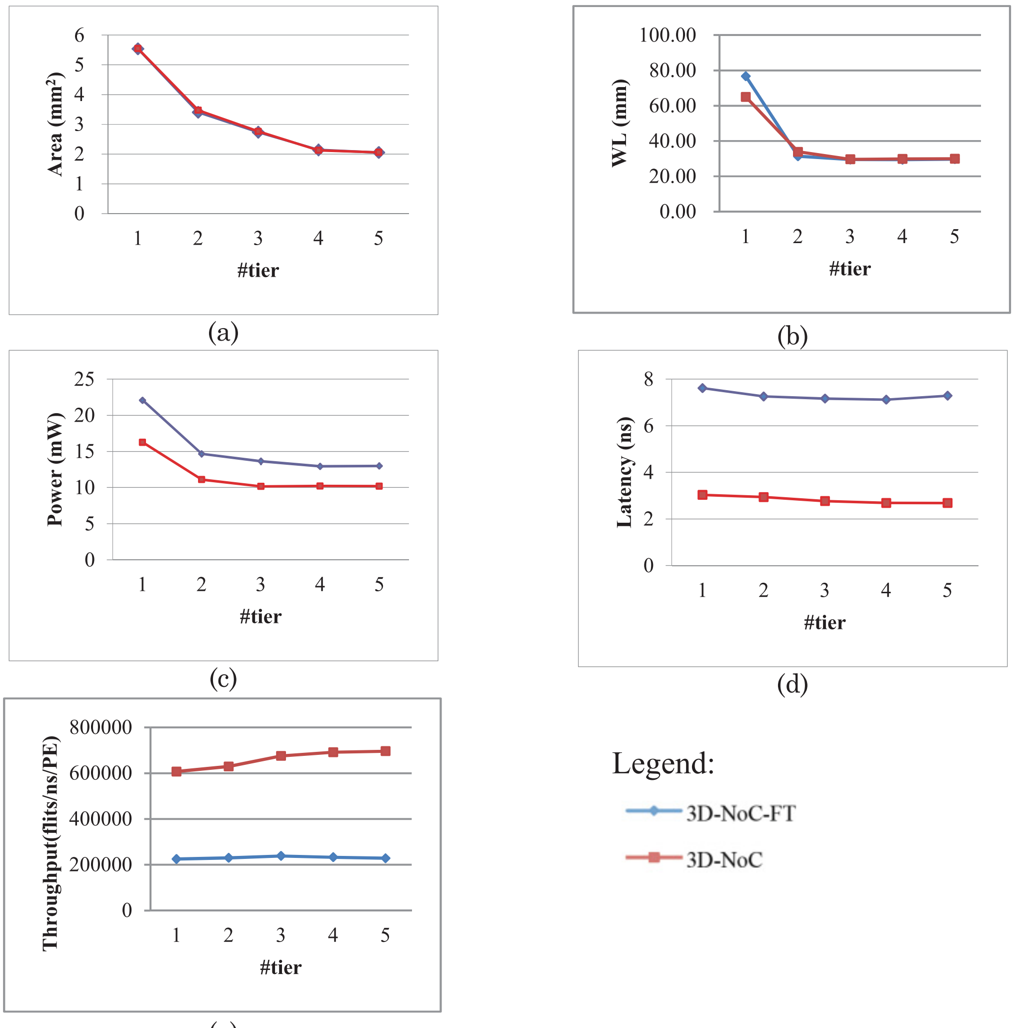

Fig. 6. Simulation results of 3D NoCs: (a) area, (b) wirelength, (c) power, (d) latency, and (e) throughput.

in Figure 6, where the relationships among five parameters (area, wirelength, power, latency, and throughput) vs. the number of tiers are illustrated. The other four parameters (#Hops, #Router, A_R, #Link) are not shown, because they are not significantly affected by the number of tiers used.

The chip size decreases as the number of tiers increases, as shown in Figure 6(a). The reason is that the same PEs are distributed over multiple tiers, so the size of each tier becomes smaller. The reduction in area becomes less significant as the number tiers increases. The sizes of circuits synthesized by 3D-NoC and 3D-NoC-FT are roughly the same, as the circuit size is dominated the PEs instead of the NoC architecture.

The overall wirelength of the NoC decreases rapidly from 2D (i.e., #tier = 1) to the two-tier 3D structure. The reason is that most of the global connections in 2D NoC are replaced by much shorter vertical links. As shown in Figure 6(b), the reduction of wirelength from one-tier NoC to two-tier NoC is more than 50%. However, for 3D NoCs

Table V. Comparison: 3D-NoC, 3D-NoC-FT, and 3D-SAL-FP with Four Tiers and 45nm Technology

Graph	3D-SAL-FP	3D-NoC						3D-NoC-FT					
	#TSVs	Power (mW)			Lat. (ns)	#TSVs	Throu. (packet/s)	Power (mW)			Lat. (ns)	#TSVs	Throu. (packet/s)
		Router	Link	Total				Router	Link	Total			
G1	6	42.36	1.02	43.38	2.71	6	1205.22	58.62	1.04	59.66	3.66	14	1219.51
G3	9	51.34	1.10	52.44	2.50	6	1874.02	53.84	1.33	55.18	4.97	14	1870.13
G6	9	49.88	3.87	53.75	2.79	5	3413.79	52.62	3.75	56.37	7.35	26	3074.06
G10	14	50.53	4.44	54.97	2.64	32	6158.36	54.90	3.84	58.74	7.63	40	4723.82
G11	40	199.37	10.21	209.59	2.99	22	17891.75	317.57	10.45	328.02	5.79	26	15843.20
Avg.	**15.60**	**78.70**	**4.13**	**82.83**	**2.73**	**14.20**	**6108.63**	**107.51**	**4.08**	**111.59**	**5.88**	**24.00**	**5346.14**
Norm.	**1.10**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.37**	**0.99**	**1.35**	**2.15**	**1.69**	**0.88**

with more than two tiers, the reduction of wirelength is insignificant. NoCs synthesized by 3D-NoC-FT have larger wirelengths than their counterparts by 3D-NoC; however, the difference is limited for true 3D NoC structures (i.e., with two or more tiers).

The power consumption in NoC consists of router power and link power. The power consumption in links is roughly proportional to wirelength, which reduces as the number of tiers increases. On the other hand, the router areas are more or less the same even though the number of tiers changes. Therefore, power consumption indeed reduces from one-tier NoC to two-tier NoC, as shown in Figure 6(c); however, the reduction is not as significant as the reduction in wirelength (Figure 6(b)). Power consumption in a NoC synthesized by 3D-NoC-FT is higher than its counterpart by 3D-NoC, as the fault-tolerant architectures need more complicated routers.

With 3D-NoC, the NoC performance (i.e., latency and throughput) improves slightly as the number of tiers increases, as shown in Figure 6(d) and Figure 6(e). On the other hand, for 3D-NoC-FT, the best performance occurs when the number of tiers ranges from two to four. The performance of 3D-NoC-FT is always inferior to 3D-NoC, and the reason is mainly due to the fault-tolerant routing topology. As shown in Table IV, the number of hops in the fault-tolerant topology is 1.89× longer than the topology with no fault tolerance (Table III), which increases the latency.

5.2. Comparison of Custom 3D NoCs

In this part, we compare the custom 3D NoCs synthesized by 3D-NoC, 3D-NoC-FT, and 3D-SAL-FP [Zhou et al. 2010], the only design method for custom 3D NoCs that has been published at this moment. To achieve fair comparison, all circuits are estimated under the same design parameters as 3D-SAL-FP [Zhou et al. 2010]: 900MHz clock frequency, 512-bit packets, 4-flit buffers, and 32-bit flits. The number of tiers in each case is four (#tiers = 4), and the router ports limit is set to 4 (i.e., a router has at most four ports). All switches and links are evaluated under a 45nm technology. Table V provides a comparison among 3D-SAL-FP, 3D-NoC, and 3D-NoC-FT with the five benchmarks used in Zhou et al. [2010]. The power and performance figures of 3D-SAL-FP are not listed in Table V for comparison, since they are estimated by a different version of Orion. The results show that the proposed design method for custom 3D NoCs produces fewer TSVs. On the other hand, to support fault-tolerant routing, 3D-NoC-FT produces networks with more TSVs while both power consumption and latency are higher. While the penalty is non-trivial, it should be noted that impact on the overall power consumption and delay is limited as the NoC part is relatively small compared with the whole system.

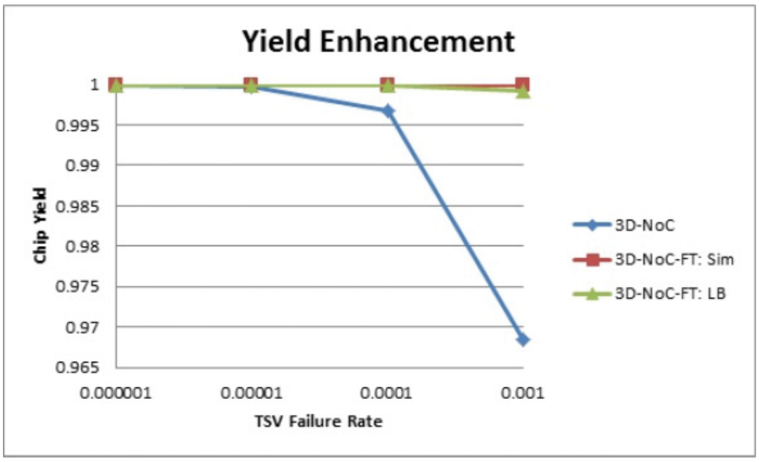

Fig. 7. Yield enhancement due to the fault-tolerant topology for G10 with four tiers.

5.3. Yield Analysis

Let the yield of TSV (i.e., the probability of a good TSV is manufactured) be y in a given 3D technology. Therefore, the failure rate of a TSV is 1–y. Assume that all dies are KGD, so the chip yield is solely decided by the TSV yield y. In a 3D-IC synthesized by 3D-NoC (i.e., non-fault-tolerant), all TSVs have to be fault-free to make the chip is functionally correct. In a chip with n TSVs, the chip yield $Y_{3D\text{-}NoC}$ is

$$Y_{\text{3D-NoC}} = y^n. \tag{8}$$

For a chip synthesized by 3D-NoC-FT, the chip can tolerate at least one faulty TSV. Therefore, in a 3D-NoC-FT chip with n TSVs, the chip can be treated as good if there is no more than one faulty TSV (i.e., 0 or 1 faulty TSV in all n TSVs). Thus, the lower bound on chip yield $Y_{3D\text{-}NoC\text{-}FT}$ is

$$Y_{\text{3D-NoC-FT}} \geq y^n + n(1-y)y^{n-1}. \tag{9}$$

The analytical model of the yield of 3D-NoC-FT is hard to derive, as two or more faulty TSVs can still be tolerated as long as at least one routing path exists for each communication link. In this case, we can use simulation to estimate $Y_{3D\text{-}NoC\text{-}FT}$. Consider the four-tier implementations of benchmark G10 given in Table V. For 3D-NoC, 32 TSVs are required, while 40 TSVs are used in 3D-NoC-FT. For 3D-NoC, a single faulty TSV in the 32 TSVs will cause system failure, while in 3D-NoC-FT a single TSV fault will surely to be tolerated. The chip yields $Y_{3D\text{-}NoC\text{-}FT}$, the lower bound on $Y_{3D\text{-}NoC\text{-}FT}$ (by Equation (9)) and yield $Y_{3D\text{-}NoC\text{-}FT}$ obtained by simulation are all shown in Figure 7. It can be seen that the lower bound is reasonably close to the results from simulation.

6. CONCLUSIONS

A three-phase design method targeted for 3D NoCs with custom topology is presented. In the simplest form (3D-NoC), the synthesized 3D NoCs are optimized according to the given communication requirement. To tolerate link faults in the NoC, extra links may be included using a generalized de Bruijn digraph in the topology generation phase (Phase II). As a result, 3D NoCs synthesized by 3D-NoC-FT are single-link faults tolerant.

Experimental results show that, in addition to reduced chip size, 3D NoC technology also provides improved performance with lower power consumption. On the flip side, 3D-ICs may suffer yield loss due to stacking of multiple dies as well as lower yield of TSVs. One way to deal with this problem is to support fault tolerance in the communication infrastructure. The proposed 3D-NoC-FT achieved this goal by introducing redundant links in NoC topology at the cost of performance penalty and elevated power level. Still, 3D-NoC-FT achieves comparable performance with previous methods, while alleviating the yield loss caused by 3D stacking.

REFERENCES

Luca Benini. 2006. Application specific NoC design. In *Proceedings of the IEEE/ACM Design, Automation, and Test in Europe Conference (DATE)*. 491–495.

Tobias Bjerregaard and Shankar Mahadevan. 2006. A survey of research and practices of network-on-chip. *ACM Computing Surveys* 38, (March 2006), Article 1, 51 pages.

Paul Bogdan, Tudor Dumitraş, and Radu Marculescu. 2007. Stochastic communication: a new paradigm for fault-tolerant networks-on-chip. *VLSI Des.* 2007, Article ID 95348, 17 pages.

Paul Bogdan and Radu Marculescu. 2011. Hitting time analysis for fault-tolerant communication at nanoscale in future multiprocessor platforms. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 30, 8 (Aug. 2011), 1197–1210.

Tung-Chieh Chen, Yao-Wen Chang, and Shyh-Chang Lin. 2008. A new multilevel framework for large-scale interconnection-driven floorplanning. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 27, 2 (Feb. 2008), 286–294.

Yiou Chen, Jianhao Hu, and Xiang Ling. 2009. De Bruijn graph based 3D network on chip architecture design. In *Proceedings of the International Conference on Communications, Circuits and Systems (ICCCAS 2009)*. 986–990.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms* (3rd ed.). MIT Press.

William James Dally and Brian Patrick Towles. 2005. Principles and practices of interconnection networks. Morgan Kaufmann.

Jeffrey A. Davis. 2001. Interconnect limits on gigascale integration (GSI) in the 21st century. In *Proc. IEEE* 89, 3 (Mar. 2001), 305–324.

Brett Stanley Feero and Partha Pratim Pande. 2005. Networks-on-chip in a three-dimensional environment: A performance evaluation. *IEEE Trans. Comput.* 58, 1 (Jan. 2005), 32–45.

David Fick. 2009. A highly resilient routing algorithm for fault-tolerant NoCs. In *Proceedings of* the *Design, Automation & Test in Europe Conference (DATE)*. 21–26.

Alberto Ghiribaldi. 2014. A vertically integrated and interoperable multi-vendor synthesis flow for predictable NoC design in nanoscale technologies. In *Proceedings of the 19th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 337–342.

Mitchell Hayenga, Daniel R. Johnson, and Mikko Lipasti. 2012. Pitfalls of Orion-based simulation. In *Proceedings of the 10th Annual Workshop on Duplicating, Deconstructing, and Debunking (WDDD-10)*.

Rickard Holsmarka, Maurizio Palesib, and Shashi Kumara. 2008. Deadlock free routing algorithms for irregular mesh topology NoC systems with rectangular regions. *J. Syst. Arch.* 54, 3–4, (Mar.-Apr. 2008), 427–440.

Makoto Imase and Masaki Itoh. 1981. Design to minimize diameter on building-block network. *IEEE Trans. Comput.* C-30, 6 (Jun. 1981), 439–442.

Makoto Imase and Masaki Itoh. 1983. A design for directed graphs with minimum diameter. *IEEE Trans. Comput.* C-32, 8 (Aug. 1983), 782–784.

Antoine Jalabert, Srinivasan Murali, Luca Benini, and Giovanni De Micheli. 2004. ×pipesCompiler: A tool for instantiating application specific networks on chip. In *Proceedings of the IEEE/ACM Design, Automation, and Test in Europe Conference (DATE)*. 884–889.

Yuan-Long Jeang, Jer-Min Jou, and Win-Hsien Huang. 2005. A binary tree based methodology for designing an application specific network-on-chip (ASNOC). *IEICE Trans. Fundam. Electron.Commun. Comput. Sci.* E88-A, 12 (Dec. 2005), 3531–3538.

Yuan-Long Jeang, Tzuu-Shaang Wey, Hung-Yu Wang, Chung-Wei Hung, and Ji-Hong Liu. 2008. An adaptive routing algorithm for mesh-tree architecture in network- on-chip designs. In *Proceedings of the 3rd International Conference on Innovative Computing Information and Control*. 18–20.

Slavissa Jovanovic, Camel Tanougast, Serge Weber, and Christophe Bobda. 2009. A new deadlock-free fault-tolerant routing algorithm for NoC interconnections. In *Proceedings of the 2009 International Conference on Field Programmable Logic and Applications*. 326–331.

Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. 2009. Orion 2.0: a fast and accurate NoC power and area model for early-stage design space exploration. In *Proceedings of the IEEE/ACM Design, Automation, and Test in Europe Conference (DATE)*. 423–428.

Woo Joo Kim and Sun Young Hwang. 2008. Design of an area-efficient and low-power NoC architecture using a hybrid network topology. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* E91-A, 11 (Nov. 2008), 3297–3303.

Katherine Shu-Min Li. 2013. CusNoC: Fast full-chip custom NoC generation. *IEEE Trans. VLSI) Syst*. 21, 4 (Apr. 2013), 692–705.

Huai-En Lian, Chien Chen, Je-Wei Chang, Chien-Chung Shen, and Rong-Hong Jan. 2009. Shortest path routing with reliability requirement in delay tolerant networks. In *Proceedings of the International Conference on Future Information Networks (ICFIN)*. 92–297.

Guoping Liu and Kyungsook Lee. 1993. Optimal routing algorithms for generalized De Bruijn digraphs. In *Proceedings of the International Conference on Parallel Processing*. 67–174.

Igor Loi, Federico Angiolini, Shinobu Fujita, Subhasish Mitra, and Luca Benini. 2011. Characterization and implementation of fault-tolerant vertical links for 3-D networks-on-chip. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 30, 1 (Jan. 2011), 124–134.

Radu Marculescu, Umit Y. Ogras, Li-Shiuan Peh, Natalie Enright Jerger, and Yatin Hoskote. 2009. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 28, 1 (Jan. 2009), 3–21.

Hiroki Matsutani. 2013. A case for wireless 3D NoCs for CMPs. In *Proceedings of the 18th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 23–28.

Nobuaki Miyakawa. 2009. A 3D prototyping chip based on a wafer-level stacking technology. In *Proceedings of the 14th IEEE/ACM Asia and South Pacific Design Automation Conference*. 416–420.

Srinivasan Murali, Luca Benini, and Giovanni Micheli. 2005. Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees. In *Proceedings of the 10th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 27–32.

Srinivasan Murali. 2006. Designing application-specific networks on chips with floorplan information. In *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*. 355–362.

Srinivasan Murali, Ciprian Seiculescu, Luca Benini, and Giovanni De Micheli. 2009. Synthesis of networks on chips for 3D systems on chip. In *Proceedings of the 14th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 242–247.

Jari Nurmi. 2005. Network-on-chip: A new paradigm for system-on-chip design. In *Proceedings of the 2005 International Symposium on System-on-Chip*. 2–6.

Dongkook Park, C. Nicopoulos, Jongman Kim, N. Vijaykrishnan, and Chita R. Das. 2006. Exploring fault-tolerant network-on-chip architectures. In *Proceedings of the International Conference on Dependable Systems and Networks (DSN'06)*. 93–104.

Vladimir Pasca, Lorena Anghel, Claudia Rusu, and Mounir Benabdenbi. 2010. Configurable serial fault-tolerant link for communication in 3D integrated systems. In *Proceedings of the 16th IEEE International On-Line Testing Symposium*. 115–120.

Alessandro Pinto, Luca P. Carloni, and Alberto L. Sangiovanni-Vincentelli. 2009. A methodology for constraint-driven synthesis of on-chip communication. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 28, 3 (Mar. 2009), 364–377.

Amir-Mohammad Rahmani. 2011. Congestion aware, fault tolerant, and thermally efficient inter-layer communication scheme for hybrid NoC-bus 3D architectures. In *Proceedings of the IEEE/ACM International Symposium on Networks on Chip (NoCS)*. 65–72.

Mohammad Reza Nouri Rad, Reza Kourdy, Majid Rahimi Nasab, and Mohammad Poyan. 2010. Improvement the NOC bandwidth and fault tolerant by multipath routing in three-dimensional topologies for multimedia applications. In *Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE)*. 497–501.

Mehrdad Seyrafi. 2010. A new low cost fault tolerant solution for mesh based NoC. In *Proceedings of International Conference on Electronics and Information Engineering (ICEIE)*. 207–213.

Krishnan Srinivasan, Karam S. Chatha, and Goran Konjevod. 2005. An automated technique for topology and route generation of application specific on-chip interconnection networks. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 231–237.

Krishnan Srinivasan, Karam S. Chatha, and Goran Konjevod. 2006a. Linear-programming-based techniques for synthesis of networks-on-chip architectures. *IEEE Trans. VLSI Syst.* 14, 4 (Apr. 2006), 407–420.

Krishnan Srinivasan and Karam S. Chatha. 2006b. A methodology for layout aware design and optimization of custom network-on-chip architectures. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED'06)*. 352–357.

B. Swinnen. 2006. 3-D integration by cu-cu thermo-compression bonding of extremely thinned bulk-Si die containing 10μm pitch through-Si vias. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*. 1–4.

Anna W. Topol. 2005. Enabling SOI based assembly technology for three dimensional (3D) integrated circuits (ICs). In *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*. 352–355.

Anna W. Topol. 2006. Three-dimensional integrated circuits. *IBM J. Res. Dev.* 50, 4/5, (Jul./Sep. 2006), 491–506.

Vijay V. Vazirani. 2003. *Approximation Algorithms*. Springer.

Yaoyao Ye. 2013. 3D mesh-based optical network-on-chip for multiprocessor system-on-chip. *IEEE Trans. Comput.-Aid. Des. Integr. Circ. Syst.* 32, 4 (Apr. 2013), 584–596.

Pingqiang Zhou, Ping-Hung Yuh, and Sachin S. Sapatnekar. 2010. Application-specific 3D network-on-chip design using simulated allocation. In *Proceedings of the 15th IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 517–522.

Received June 2016; revised January 2017; accepted January 2017