This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

Scalable design methodology and online algorithm for TSV-cluster defects recovery in highly reliable 3D-NoC systems

Khanh N. Dang, *Student Member, IEEE,* Akram Ben Ahmed, *Member, IEEE,*
Yuichi Okuyama *Member, IEEE,* and Abderazek Ben Abdallah, *Senior Member, IEEE*

Abstract—3D-Network-on-Chips exploit the benefits of Network-on-Chips and 3D-Integrated Circuits allowing them to be considered as one of the most advanced and auspicious communication methodologies. On the other hand, the reliability of 3D-NoCs, due to the vulnerability of Through Silicon Vias, remains a major problem. Most of the existing techniques rely on correcting the TSV defects by using redundancies or employing routing algorithms. Nevertheless, they are not suitable for TSV-cluster defects as they can either lead to costly area and power consumption overheads, or they may result in non-minimal routing paths; thus, posing serious threats to the system reliability and overall performance. In this work, we present a scalable and low-overhead TSV usage and design method for 3D-NoC systems where the TSVs of a router can be utilized by its neighbors to deal with the cluster open defects. An adaptive online algorithm is also introduced to assist the proposed system to immediately work around the newly detected defects without using redundancies. The experimental results show the proposal ensure less than 2% of the routers being disabled, even with 50% of the TSV clusters defects. The performance evaluations also demonstrate unchanged performances for real applications under 5% of cluster defects.

Index Terms—3D-NoCs, Fault-tolerance, Reliability, Architecture and Design, TSV-cluster defects.

1 Introduction

In the past few years, the 3D-Network-on-Chip (3D-NoC) paradigm [1] is considered as one of the most promising architectures for IC design. It is a result of the fusion of 3D-Integrated Circuits (3D-ICs) [2] and the mesh-based Network-on-Chips (NoCs) [3]. In fact, the parallelism and scalability of NoCs can be further enhanced in the third dimension thanks to the short wire length and low power consumption of the Through-Silicon Vias (TSVs), that constitute one of the main inter-layer communication mediums. As a result, the 3D-NoC paradigm is considered to be one of the most advanced and auspicious architectures.

As depicted in Fig. 1, a TSV works as an inter-layer wire in 3D-NoCs, as well as in 3D-ICs. By creating vias, thinning the wafer and performing a thermo-compression [4], TSVs are established and the two wafers can connect through them. TSVs are usually fabricated regularly into a group or a cluster, or irregularly in random positions. While TSVs bring many advantages for 3D-NoCs, one of their major drawbacks is reliability.

The yield rates of 3D-ICs using TSVs have been considered as a critical factor due to the imperfection of the manufacturing process [5]. The TSV defect-rates have been reported as nearly 0.63% [6]. Moreover, 3D-ICs suffer from the stress issue due to the difference between thermal expansion coefficients of the implementation materials [7]. The temperature variation between two layers has been reported to reach up to 10°C [8] which negatively affects the *Time Dependent Dielectric Breakdown* and *Thermal Cycling* [9]. Not forget to mention that *Electromigration* [10] can also be a major concern. As a result, TSVs in 3D-ICs have become more fault sensitive, not only in the manufacturing phase; but, also during the operation time.

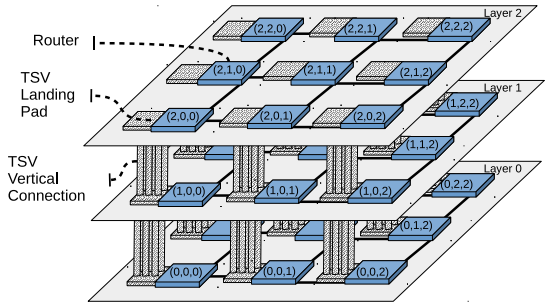

Fig. 1: Structure of a conventional 3D Network-on-Chip system.

The TSV defects can be classified into four main types: *Open* (or void), *Bridge*, *Stuck-at*, and *Crosstalk* [5], [9], [11]. Open defects occur when TSVs are broken or misaligned, and their terminals are electrically disconnected. Bridge defects manifest when two or more TSVs connect together. As a result, these TSVs are unable to transmit the different values. Stuck-at faults short the TSV to *ground* or to *Vdd* which makes the output to always remain at '0' or '1'. If the TSV is partially defected, an extra delay can occur which may violate the timing requirements [11]. Crosstalk is the interference inflicted by surrounding TSVs on a victim TSV which creates unexpected values. This paper mainly focuses on open TSV defects, and the other types are not addressed.

As explained in details in Section 2, existing works presented

- *Khanh N. Dang , Yuichi Okuyama, and Abderazek Ben Abdallah are with Adaptive Systems Laboratory, Graduate School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan.*
  E-mail: khanh.n.dang@ieee.org
- *Akram Ben Ahmed is with Department of Information and Computer Science, Keio University, Yokohama 223-8522, Japan*
- *Khanh N. Dang has moved to SISLAB, The University of Engineering and Technology, Vietnam National University Hanoi, Hanoi 123106, Vietnam since October 1st, 2017.*

Manuscript received -.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

so far have dealt with the high fault-rate of TSVs in different approaches: improving the manufacturing process to enhance the reliability of TSVs [12]; accounting the potential defects in the design stage [7]; correcting the defected TSVs by using supporting circuits [13], [14], redundancy [11], [15], [16], or *Error Correction Codes* [17]; and using an alternative channel to avoid the defected TSV channel (e.g. using fault-tolerant routing [1] in NoCs).

Although these works have impressively enhanced the reliability of TSV-based systems, there is still an existing issue in the fault distribution. Most of the first conducted works addressed the random distributions [16], [18]; however, the cluster defect distributions [5], [15], [19] are recently considered as the most realistic ones. To deal with the cluster TSV defect, most works aim to select a suitable grouping configuration [19] to distribute TSVs on different positions [5] or to enhance the redundancy correction rate [15]. Although these methods can improve the reliability of the system, adding extra redundancies and complex arbitration modules result in penalties on area cost, wire latency, and power consumption. Moreover, if the number of defective TSVs is larger than the number of assigned redundant ones, the vertical connection will be corrupted. Therefore, we observe that a better management solution can help to deal with this issue, especially for 3D-NoCs, where the low utilization rates of TSVs have been reported [20].

In this paper, we propose a scalable TSV utilization architecture and methodology to tackle the lack of reliability in inter-layer links. To reduce the TSV-cluster defects, a router corrects its defected TSV communication by choosing one of its four neighbor TSV-clusters located on the same layer. To avoid timing violation issues, we place the TSVs of two nearby routers in between them and a TSV-cluster is only shared between its two neighboring routers. Experimental results show that the solution can help 3D-NoCs to work around TSV-cluster defects without the need for redundancy. Therefore, reliability at reasonable overhead is guaranteed. Since this work only focuses on TSV fault recovery, detection is out of scope. Therefore, we assume there is an existing detection module which helps the system to detect the occurrence of TSV-cluster defects.

The paper is organized as follows. Section 2 presents the motivations and prior works. In Section 3, we describe the proposed TSV fault-tolerant architecture. The algorithms and optimizations are explained in Section 4. Section 5 shows our evaluations and comparison results. Finally, Section 6 concludes the paper.

2 Motivations and Prior Works

2.1 Reliability Issues of TSV-based 3D-ICs

TABLE 1: TSV Defect-rate Summary.

Work	TSV Pitch	Defect Rate	TSV Number	Yield w/o Spare
IBM'05 [21]	0.4μm	13.9E-6	1k-10k	95% 98%
IMEC'06 [22]	10μm	40.0E-6	10k	67%
HRI'07 [23]	-	9.75E-6	100k	68%
HRI'09 [24]	-	7.95E-6	100k	≥90%
SAMSUNG'09 [6]	-	0.63%	300	15%

Table 1 summarizes the defect-rates of TSV fabrication. The defect-rate of TSVs is considered high which negatively affects the final yield. In [6], 0.63% of the TSVs are reportedly defected, and the final yield without spares is only 15%. Besides the high defect-rates during the manufacturing stage, TSVs under operation also face several challenges with stress and thermal issues, as reported in [7]. As a result, TSVs are one of the most vulnerable components in 3D-ICs.

One of the matters that is still under investigation is the TSV failure distribution. In general, there are two main assumptions for the failure distribution: *Random* [16], [18] and *Clustering* distributions [5], [15], [19]. *Random* TSV defect is efficiently dealt by adding redundancies and recovery methods; but, *Clustering* defects still remain a considerable challenge. Moreover, TSV misalignment [16] also may occur and is classified as a cluster defect. Because of the stress and thermal issues, TSVs may also be defected after manufacturing. In [9], the authors presented several *Mean Time To Failure* equations of 3D-ICs affected by *Time Dependent Dielectric Breakdown*, *Thermal Cycling* and *Electro-migration* where the temperature values play an important role. Because of the clustering effects on hot-spot areas in 3D-ICs [4], the obvious result was found to be the TSV-cluster defect.

2.2 TSV Fault Tolerance

Numerous works have addressed the fault tolerances and reliability issues in 3D-NoCs. In this paper, we focus on TSV defect tolerances. The existing works have approached the TSV fault-tolerance in three layers: *Physical layer*, *Data-link layer* and *System layer*.

In *Physical layer*, the improvement of TSV manufacturing can help to reduce the defect-rate [12]. Designers can optimize the physical layout or use thermal-aware routing and placement methods to improve the reliability of 3D-ICs [7]. Even when a fabricated TSV has a short defect, a correction circuit, using a voltage comparator to gain the output voltage of the TSV, can be employed [13]. To enhance the reliability of TSVs, [14] proposed a method named *Double TSV* which uses two TSVs, instead of one, to maintain the vertical communication. If an open, short-to-substrate or bridge defect occurs in one TSV, the communication is still performed by the duplicate one.

In the *Data-link layer*, the most common method is adding redundant TSVs to correct the defected ones [11], [15], [16]. The major concern of this method is to efficiently route from a defected TSV to a spare one. There are four basic solutions: (a) signal switching [6], (b) single shifting [18], (c) crossbar [16] and (d) network routing [15]. Because of the cluster defect, adding redundancies becomes a costly technique with a high number of needed spare TSVs (up to 50% in [15], [19]). In [5], the authors propose a mapping method to reduce the impact of cluster defects. TSVs in the same group are mapped to a random position with the help of an optimization process. On the other hand, *Zhao et al.* [19] analyze the grouping method to achieve the best recovery. The work presented in [15] introduces an innovated method for TSV mapping by creating a network and implementing an algorithm for re-routing the defected TSVs. On the other hand, *Reddy et. al* [25] proposed a Time Division Multiplexing Access for TSVs which can help to correct defects with low area overheads. *Loi et al.* [16] proposed a crossbar redundancy structure for 3D-NoCs. A testing mechanism is also presented to help the system detecting the defected TSVs. Because TSVs manage the vertical connections in a 3D-NoC, *Error Correction Coding* [17] is also a prominent method for detecting and correcting the defected TSVs; however, this type of solutions requires extra bits, which significantly increases the area cost and power consumption.

In the *System layer*, which mainly focuses on 3D-NoCs, fault-tolerant routing algorithms [1], [26] are one of the most suitable

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

solutions. To reduce the risk of thermal and stress issues in 3D-NoCs, thermal aware management [27] is also a promising solution. On the other hand, the majority of works proposed off-line testing and recovery schemes which are not suitable for post-manufacturing. The system operation should stop to be tested and recovered. In [11], the authors presented an online testing solution. Because the reliability of TSVs is a critical issue, the need for online testing recovery is primordial.

As we previously mentioned, the cluster defect is predicted to frequently occur. The most efficient solution for correcting random defects is grouping and adding redundancies. However, they are still inefficient for the cluster defect and require a costly extra area for the redundancies. Therefore, fault-tolerance for cluster defect is the main interest of this paper. On the other hand, several works [20] have been reporting the low utilization of the vertical connections using TSVs in 3D-NoCs. Motivated by the cluster defect issue and the low utilization of the TSVs in 3D-NoC, we propose in this paper a low-cost method for TSV fault-tolerance in 3D-NoCs.

3 Proposed TSV Fault Tolerance Architecture

To handle the TSV-cluster defects in 3D-NoCs, our solution is to share TSVs between neighboring routers. Therefore, when a TSV-cluster fails, its router can borrow a healthy cluster from one of its neighbors to maintain the connection. Moreover, we also present several design optimization methods to improve the reliability of the system (Section 4.3).

3.1 Fault assumptions

Before we present the system structure, this subsection clarifies the fault assumptions taken in this proposal. Because the cluster defect [5], [15], [19] is the primary obstacle to be dealt in this paper, we assume there are no random defects. Here, we consider an occurred fault makes the whole TSVs in the cluster defected. For those who might be concerned about random defects, using redundancy [6], [14], [16], [18] can be easily integrated into our TSV-cluster design. For controlling signals using TSVs, they are considered as a part of the TSV cluster instead of separated TSVs, which are better dealt as random defects (e.g., [11] uses Double TSV [14]). The detection process, which can be handled in an online fashion using thermal-aware testing [28] or in an offline approach using a Built-In-Self-Test module [29], is assumed to be existing and connected to the fault-tolerance module. To synchronize the configuration, the existing NoC infrastructure is used instead of adding TSVs. Therefore, no redundancy is required in the proposed architecture.

3.2 System structure

A simplified layout example of $3 \times 3 \times 3$ 3D-NoC system using the proposed TSV usage is depicted in Fig. 2. For each vertical connection, a router needs a set of TSVs. Instead of grouping all TSVs together, as shown in Fig. 1, they are divided into four groups. As a result, a router owns four TSV-clusters and has a maximum of four nearby TSV-clusters. If a TSV-cluster of a router defects, the router can choose one of its four neighboring clusters as a replacement without the need for redundancy. To satisfy the timing constraints, the router chooses the closest TSV-cluster among its neighbor clusters. Taking into account further TSV-clusters is not considered in order to avoid long wires that are needed to establish the connection. By structuring the TSVs into four clusters for each router, we can maintain the scalability of 3D-NoCs and avoid long wire delay.

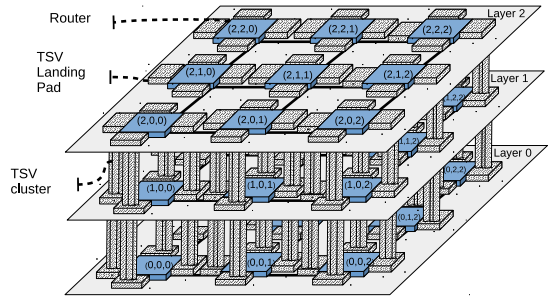

Fig. 2: Simplified block diagram illustrating the proposed system structure.

Figure 3 shows the placement and connection of the TSV sharing area between *R(1,1,1)* and *R(1,0,1)*. Because each router has two ports (up and down) and two directions (in and out), the number of TSV clusters is eight. Each TSV cluster handles a quarter of the vertical connection. By using tri-state gates, the system can control which router has access to the TSV clusters.

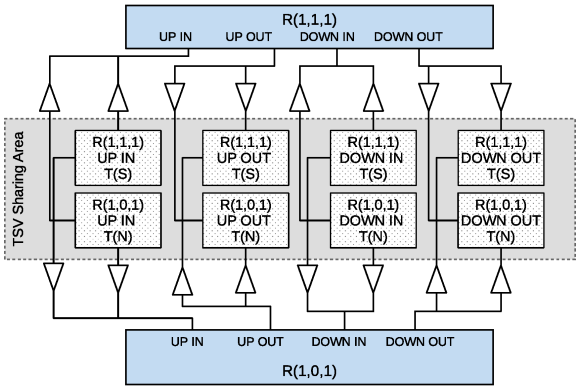

Fig. 3: TSV sharing area placement and connectivity between two neighboring routers.

3.3 Sharing Circuit Design

To borrow a TSV-cluster from a neighbor, the router needs supporting modules. Figure 4 (a) shows the wrapper of a 3D-Router with the additional supporting modules that perform the sharing algorithm, later explained in Section 4. There are two identical sharing modules (S-UP and S-DOWN) for the two vertical up and down connections and each connection has two configuration registers (CR) for the input and output ports. As previously depicted in Fig. 2, *R(1,1,1)* shares the TSV-clusters with its four neighbors: *R(1,1,0)*, *R(1,1,2)*, *R(1,0,1)*, and *R(1,2,1)*. Figure 4 (b) shows the sharing circuit for one TSV-cluster. The input of this TSV-cluster is shared between *R(2,1,0)* and *R(2,1,1)* on layer2. The output of this TSV-cluster is shared between *R(1,1,1)* and *R(1,1,0)* on layer1. In the case where this TSV-cluster is defected, or borrowed, the data can be sent by using one of the four neighboring clusters.

Based on the value of the 6-bit CR, the input, and output ports can select the data from: (1) its original TSV-cluster (first bit), (2) one of its four neighboring clusters (second bit) or (3) being disconnected (a replacement cluster is indicated in one of last four bits). As shown in Figure 4 (b), the least significant bit decides

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

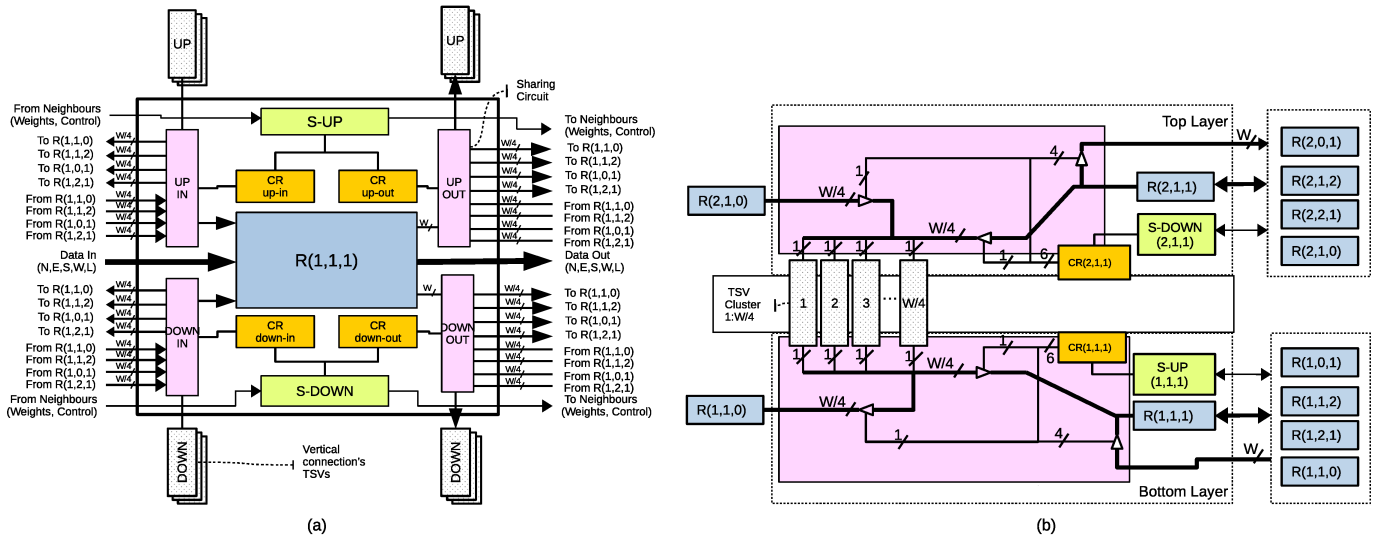

Fig. 4: The TSV fault-tolerance architecture: (a) Router wrapper; (b) Connection between two layers. Red rectangles represent TSVs. *S-UP* and *S-DOWN* are the sharing arbitrators which manage the proposed mechanism. *CR* stands for configuration register and *W* is the flit width.

whether *R(2,1,1)* can access its own TSV-cluster. The second least significant bit allows the neighboring router (*R(2,1,0)*) takes the access to this cluster. The last one-hot 4-bit of CR helps *R(2,1,1)* accessing one of its four nearby clusters. At the receiving router (*R(1,1,1)*), a similar CR with a synchronized value is used to establish the connection.

Because the CR only manages the connectivity, its value has to be set carefully to avoid the possible conflict of TSV-cluster usage and to optimize the performance. To this aim, an adaptive sharing algorithm is needed.

4 Adaptive Online Sharing Algorithm

Algorithm 1: TSV Sharing Algorithm.

```
// Weight values of the current router and its N neighbors
Input: Weight_current, Weight_neighbor[1 : N]
// Status of current and neighboring TSV-clusters
Input: TSV_Status_current[1 : N], TSV_Status_neighbor[1 : N]
// Request to link TSV-clusters to neighbors
Output: RQ_link[1 : N]
// Current router status
Output: Router_Status
1  foreach TSV_Status_current[i] do
2      if TSV_Status_current[i] == "NORMAL" then
           // It is a healthy TSV-cluster
3          RQ_link[i] = "NULL"
4      else
           // It is a faulty or borrowed TSV-cluster
5          find c in 1:N with:
6          Weight_neighbor[c] < Weight_current
7          Weight_neighbor[c] is minimal
8          and TSV_Status_neighbor[c] == "NORMAL";
9          if (c==NULL) then
10             return RQ_link[i] = "NULL"
11             return Router_Status = "DISABLE"
12         else
13             return RQ_link[i] = c
14             return Router_Status = "NORMAL"
```

In the previous section, we presented how a router can use its nearby TSV-clusters to maintain the connection and the operation on a layer. The CR values need to be configured to deal with the TSV defects. The simplest way for this process is to perform it offline, and the configuration fuses the TSV group [15]. However, fixing the connections has two main drawbacks: (1) recovering a newly defected TSV needs to halt the system and re-perform the mapping, and (2) each application has a different distribution in the vertical connections and variations depending on the running task which is not optimized by offline mappings. Consequently, we aim to perform the mapping online so that the system can react immediately to the newly defected TSV-clusters and can consider the connectivity of the 3D-NoC system. Thus, this subsection provides an online algorithm for sharing TSVs which can be implemented into the system.

Algorithm 1 shows the proposed algorithm for our sharing mechanism. Each router is assigned a weight for each of the vertical connections. This weight decides its priority in sharing/borrowing. The weights can be assigned at the design process or can be updated by a dedicated module. Changing the weights of routers can create different mappings. At the initial stage, all routers in the network exchange their weights and their TSV-clusters status with their neighbors. In the next step, the algorithm performs the mapping process. If a TSV-cluster is defected, its corresponding router should find from its neighbors a possible candidate by relying on the following conditions:

- The weight of the candidate has to be smaller than the current router.
- The candidate TSV-cluster has to be healthy and not borrowed.
- The weight of the final candidate is the least among all the possible candidates.

At the end of the algorithm, the router finds out a possible candidate for borrowing. If no candidates were found, the router's vertical connection is disabled. If there is a candidate, the router sends a request to the borrowing router to use its TSV-cluster as a replacement for the defected one. The routers having borrowed TSV-clusters also look for replacements among their neighbors. By using a weighted system, the disabled TSV-clusters focus on smaller weight routers.

Figure 5 shows an example of how the sharing algorithm works on a 4×4 layer with ten defected TSV-clusters. Initially, the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

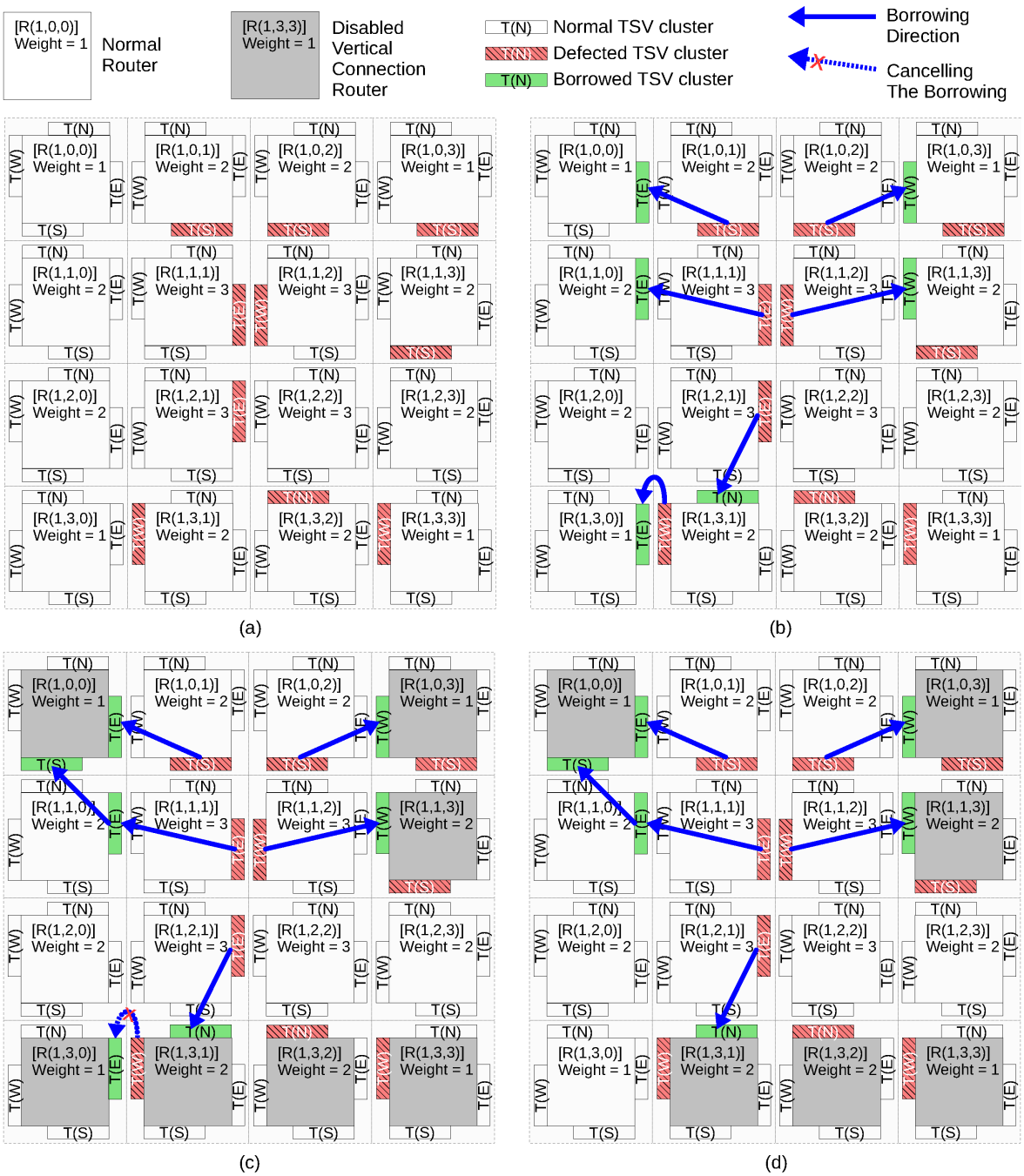

Fig. 5: An example of the sharing algorithm on a 4×4 layer: (a) Initial state with ten defected TSV-clusters; (b) Best candidates selection; (c) Borrowing chain creating and selection refining. (d) Final result with six disabled routers.

routers in the center, which are predefined to have higher TSV utilization rates, have higher weights than those at the edges of the network, as depicted in Fig. 5 (a). The sharing algorithm selects the best candidates, shown in Fig. 5 (b), by following the rules previously explained in Algorithm 1. Fig. 5 (c) shows that this selection must be further refined by disabling the router having less than four functional (or not borrowed) TSV-clusters and canceling their borrowing. Moreover, we also observe the case in Fig. 5 (d) where two routers *R(1,3,2)* and *R(1,3,3)* are disabled; but, *R(1,3,3)* can borrow a TSV-cluster from *R(1,3,2)* to obtain full four TSV clusters. However, the borrowing is prohibited due to the higher weight of router *R(1,3,2)*. In order to optimize this case, we use a technique named *Weight adjustment* in Section 4.2.

As shown in the above example, the chains of sharing lead to disabling the routers on the edges. Instead of having ten defected TSV-clusters, the algorithm only disables six routers having the lowest weights (40% of reduction). Consequently, maintaining the connections of the center routers, which have higher weights and utilize more vertical communications, can reduce the impact of TSV defects regarding overall performance.

4.1 Weight Generation

One of the most important parameters in the sharing algorithm is the weight values of the routers. The weights help the algorithm deciding what router is suitable to be borrowed. As shown in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

Fig. 5, the routers having smaller weights are disabled after the chains of sharing are established.

Because the weights of routers play important roles in the sharing process, they need to be optimized to obtain optimal system performance and defect-resiliency. One possible solution is using a statistic-based solution where the priority of the vertical connections depends on the communication traffic [30]. In other words, the vertical connections having more data transmissions are assigned higher weights; otherwise, smaller weights are assigned.

Because the application mapping is out of the scope of this paper, we adopt a simple method where the routers in the middle of the layer have the highest weights. This priority rule is based on the observations made on network traffic during our evaluations where the middle routers usually have to handle more data than the ones located on the layer edges. Moreover, the middle routers have more diversity in the routing path which can enhance the ability to route packets inside the network. This assignment also helps to shift the disablement of vertical connections and the serialization processes to be more likely executed on the edges of the network. In such locations, the proposed algorithm can avoid the congestion by serializing or using virtual TSVs (Section 4.3). The router's weights are decreased and become the lowest at the edges of the layer. Equation 1 shows the used weight value assignment.

$$\text{Weight}_{\text{router}}(x, y) = \min(x, \text{cols}-x-1)+\min(y, \text{rows}-y-1)+1 \quad (1)$$

Where *cols* and *rows* are the number of columns and rows within a layer, respectively.

The output of this weight assignment on a layer of 4×4 can be seen in Fig. 5 where, for instance, the weights of routers *R(1,0,0), R(1,1,0)*, and *R(1,1,1)* are 1, 2, and 3, respectively.

4.2 Weight adjustment

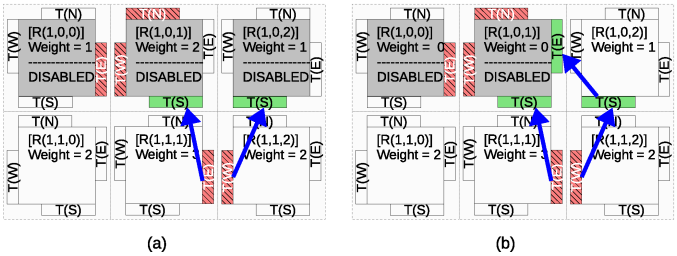

Fig. 6: Example of the weight adjustment performed to disable routers' sharing: (a) Before weight update; (b) After weight update.

After applying the sharing mechanism, the disabled TSV-clusters are shifted to the regions which consist of low weighted routers. Figure 6 (a) shows a case of three routers (*R(1,0,0), R(1,0,1)* and *R(1,0,2)*) which are disabled after the sharing process. However, there still are chances of optimizing these routers to obtain a better mapping. In fact, *R(1,0,2)* can borrow a TSV-cluster from *R(1,0,1)*. Therefore, the number of TSV-clusters of *R(1,0,2)* can be maintained to four.

To perform this optimization, the disabled router, after the sharing process by Algorithm 1, is brought to a new process. First, the router counts the number of possible TSV-clusters that it can borrow. Since three routers (*R(1,0,0), R(1,0,1)* and *R(1,0,2)*) are disabled, their TSV-clusters are free to be taken. At the end of this stage, *R(1,0,0), R(1,0,1)* and *R(1,0,2)* have 1, 3, and 1 borrowed/defected TSV-clusters and are able to take 0, 1 and 1 TSV-cluster from their disabled neighbors, respectively. At the second stage, the router checks whether it can take the disabled router's cluster to obtain a full connection. Because *R(1,0,2)* has one borrowed cluster and is able to borrow another one from *R(1,0,1)*, its weight is kept. The weights of other routers (*R(1,0,1)* and *R(1,0,0)*) are reduced to zero. As a result, *R(1,0,2)* can borrow a TSV cluster from *R(1,0,1)* despite the fact that it originally has a lower weight. The result is shown in Fig. 6 (b) where *R(1,0,2)* vertical connection is re-enabled. If the system wants to restart the sharing mechanism, the weights of all routers need to be reinitialized.

Algorithm 2: Weight Adjustment Algorithm.

```
// Status of current and neighboring TSV-clusters
Input: TSV_Status_current[1:N], TSV_Status_neighbor[1:N]
// Current and neighboring routers status
Input: Curr_Status, Neighbor_Status[1:N]
// Request to link TSV-clusters to neighbors
Output: Weight_current
1  Curr_TSVs = 0;
2  foreach TSV_Status_current[i] do
3      if TSV_Status_current[i] == "NORMAL" then
4          Curr_TSVs + +;
5  Neighbor_TSVs = 0;
6  foreach TSV_Status_neighbor[i] do
7      if TSV_Status_neighbor[i] == "NORMAL" and
         Neighbor_Status[i] == "DISABLED" then
8          Neighbor_TSVs + +;
   // If there is at least 4 cluster, run the sharing algorithm
9  if Neighbor_TSVs + Curr_TSVs >= 4 then
10     call TSV_Sharing()
11 else
       // Reduce the current weight to allow the neighbors borrow
12     Weight_current = 0;
```

Algorithm 2 shows the weight adjustment algorithm. It first calculates the total number of healthy TSVs that are possible for use. If the total number of healthy TSV-clusters is larger or equal than four, which is enough to maintain the vertical connection, the neighboring routers' weights are reduced. After that, the TSV sharing algorithm (Algorithm 1) is performed, where the router now can take TSV-clusters from the routers having higher weights, but is disabled.

4.3 Design optimization

Without adding redundancy, borrowing TSV-clusters to work around the defected ones makes some routers to have less than four accessible clusters (e.g., R(1,0,0) in Fig. 5 (d)). As a result, the communications of these routers have been disabled. To tackle this problem, the naive solution is using a fault-tolerant routing algorithm to re-route the packets to a neighboring router. As we mentioned in Section 2, this solution may lead to non-minimal routing and congestion in the network. Therefore, we propose *Virtual TSV* to help these routers maintaining the connection without using any fault-tolerant routing algorithm. In the case where the *Virtual TSV* is unable to be performed, we also implement the *Serialization* technique which helps the vertical connection establishing only one or two TSV-clusters.

4.3.1 Virtual TSV

When a router is not granted the access to four TSV-clusters, it is disabled. However, if the number of nearby TSVs is larger or equal than four they can be utilized to establish a connection. A possible connection, which requires four TSV-clusters, may need

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

Algorithm 3: Virtual TSV.

```
// Status of current and neighboring TSV-clusters
Input: TSV_Status_current[1:N], TSV_Status_neighbor[1:N]
// Allowing signals from the neighbors
Input: Allow2Borrow[1:N], Allow2Return[1:N]
// Current and neighboring routers status
Input: Curr_Status, Neighbor_Status[1:N]
// Request to link TSV-clusters to neighbors
Output: Req2Borrow, Req2Return, RunMode
1  Curr_TSVs = number of healthy and owning TSV clusters;
2  Borrowed_TSVs = number of borrowed TSV clusters;
3  if Curr_TSVs + Borrowed_TSVs == 4 then
       // Request to return the borrowed TSV clusters
4      Req2Return = True;
5  else
       // Perform the sharing algorithm to find suitable clusters
6      find (4 − Curr_TSVs + Borrowed_TSVs) TSV clusters.
       // Request to return the borrowed TSV clusters
7      Req2Return = True;
       // Request to borrow new TSV clusters
8      Req2Borrow = True;
9  Allow_TSVs = number of allowed to return/borrow TSV clusters;
10 if Curr_TSVs + Allow_TSVs == 4 then
11     RunMode = "VIRTUAL"
12 else
13     Req2Return = False;
14     Req2Borrow = False;
15     if Curr_TSVs ≥ 1 then
16         RunMode = "SERIALIZATION"
17     else
18         RunMode = "FAULT-TOLERANT ROUTING"
```

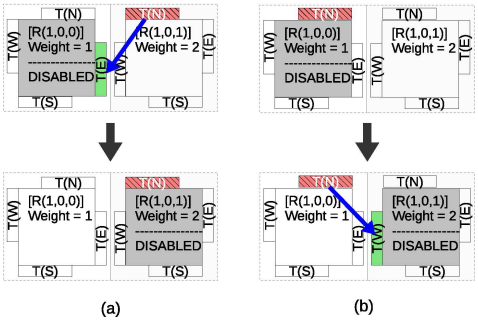

Fig. 7: Examples of Virtual TSV: (a) return the TSV-cluster to the original router; (b) borrow a cluster from a higher weight router.

clusters belonging to the neighboring routers. If these routers do not use these clusters, the disabled router can borrow them for a short period to establish communication. The process of *Virtual TSV* can be seen in Algorithm 3.

Figure 7 (a) shows an example of how *Virtual TSV* works where *R(1,0,1)* has a defective cluster (T(N)) and borrows a cluster from disabled *R(1,0,0)*. When *R(1,0,0)* needs to establish an inter-layer communication, it requests to return the borrowed cluster T(E). When the packet is completely transmitted, the borrowing cluster is taken back by *R(1,0,1)* again. On the other hand, Fig. 7 (b) shows the case where a disabled router *R(1,0,0)* temporarily borrows a TSV-cluster from a higher weight router *R(1,0,1)* to establish an inter-layer connection. For selecting a suitable candidate to temporarily borrow, Algorithm 1 is utilized.

Because there is a case where *R(1,0,1)*, which has the higher priority, occupies the TSV for a long transmission time, *R(1,0,0)* is unable to access the TSV to establish a connection. Moreover, at a high defect-rates, *R(1,0,0)* may not find any suitable candidate for *Virtual TSV*. In order to address these cases, we adopt the *Serialization* [31] technique to maintain the connection.

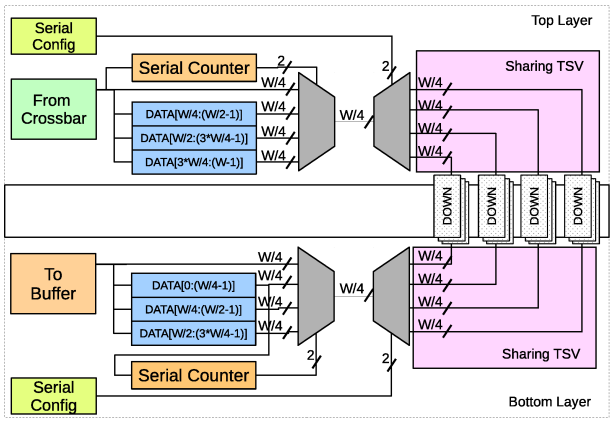

Fig. 8: Circuit of 1:4 Serialization.

Algorithm 3 shows how a router can request clusters for *Virtual TSV*. It first checks its original TSV clusters. If it has clusters lending to its neighbors, it requests to take them back. If the total number of available clusters is less than four, it must find suitable nearby clusters to have the necessary four clusters. This is achieved by using a similar process to the sharing algorithm (Algorithm 1). After the returning and borrowing requests are granted, the router can transmit its data. If the router fails to have four clusters, it enables *Serialization* mode (if it has a least one current cluster) or use fault-tolerant routing (if it has no cluster).

For a TSV-owning router receiving a "borrow-request" signal, it first checks whether it can temporarily let the requesting router use its clusters. If the clusters are not used, it sends a "borrow-grant" signal to the requesting router. At the same time, the TSV-owner router stops sending any grant signals to its routing units to prevent them from using the borrowed clusters. As a result, the TSV-owner router is unable to route its packets in the vertical connection, and the requesting router can use the clusters. If the requesting router succeeds to find the necessary amount of four clusters, it transmits data and returns the borrowed TSV clusters back to their owners once the communication is over. However, if the requesting router fails to find the necessary four clusters, it turns off the "borrow-request" signal which cancels the borrowing process and returns the borrowed clusters to their owners.

4.3.2 *Serialization Technique*

Although the *Virtual TSV* can help the disabled router maintaining its vertical connection, there are still two situations where *Virtual TSV* cannot be performed: (a) there are less than four healthy TSV-clusters, (b) the candidate TSV-cluster is occupied continuously by a higher priority router. In order to solve these cases, we use the *Serialization* technique [31] to keep the connectivity. If a cluster in this router is defected, *Serialization* is utilized to maintain the connection. Besides the serialization technique, Time Division Multiplexing Access for TSVs [25] is also promising solution to maintain the connection with a limited number of TSVs. Since the *Serialization* technique demands extra buffer slots and multiple cycles to handle a single flit, it may result in a considerable area cost overhead and performance degradation. However, the *Serialization* technique ensures the availability of a connection in high defect rates. Depending on the reliability requirements, designers can switch between the *Serialization* and fault-tolerant routing in the design phase or during the system operation.

For the serialization, the router needs at least one TSV-cluster to maintain its connection. If there is one available cluster, the 1:4

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

serialization is used, if there are two available clusters, the 1:2 serialization is established. The up and down directions' output of the crossbar is stored in a register, and the serialization module transmits flits over the remained clusters. Figure 8 shows the vertical interface between two routers using 1:4 serialization. Two serial counters are used to synchronize the order of data. A flit is divided into four segments and transmitted in four clock cycles. For 1:2 serialization, the similar principle is used for two TSV clusters instead of one in 1:4.

4.4 Partially connected 3D-NoCs

Besides the uniform TSV distributed NoCs, shown in Fig. 2, there is a case where TSVs are not found in every router which creates a partially connected 3D-NoC. Such systems are sometimes preferred for custom 3D-NoC designs, due to their low area cost and power overhead. Therefore, the efficiency of the proposed methodology with such 3D-NoC systems should be clarified. In partially connected 3D-NoCs, the TSV placing process should favor the case where routers with TSVs are placed close to each other to maintain the timing constraints. By placing these routers in a region, the sharing algorithm, presented in Section 4, can be performed in the same way as fully-connected networks without any modifications. The TSV clusters status of the router without TSVs is considered as defected and its corresponded vertical connection is disabled. In the case where routers with TSVs are placed distantly, the Serialization technique explained in Section 4.3.2, or a fault-tolerant routing algorithm can be utilized to maintain the connectivity of the standalone routers.

5 Evaluation Results

5.1 Evaluation Methodology

The proposed system was designed in Verilog-HDL, synthesized and prototyped with commercial CAD tools. We use NANGATE 45nm library [32] and NCSU FreePDK TSV [33]. The TSV size, pitch and Keep-out Zone are $4.06\mu m \times 4.06\mu m$, 10 μm, and 15 μm, respectively. The proposed technique is implemented into a 3D Mesh NoC system having four as the input buffers depth and 44-bit flit size. The flow-control is Stall-Go and the forwarding mechanism is Wormhole. First, we evalute the defect-rate by inserting faults (defects) into TSV-clusters and assess the reliability of the proposed 3D-NoC system. Second, we use both synthetic and realistic traffic patterns as benchmarks to study the performance of the proposed system in comparison to the baseline model [34]. Third, we evaluate the hardware complexity of a single 3D router and compare our system with other proposed approaches [15], [19].

5.2 Defect-rate evaluation

In this section, we demonstrate the efficiency of the proposed technique under different defect-rates, as shown in Figure 9. To prove the scalability of our proposal, we evaluated several layer sizes: 2×2, 4×4, 8×8, 16×16, 32×32, and 64×64. TSVs are grouped into clusters, as presented in Section 3 and the defect-rates vary from 5% to 50%. We perform the Monte-Carlo simulation for the proposed algorithms with 100,000 different samples and calculate the average results. We measure the ratio of four types routers in the layer: *Normal* (healthy or corrected), *Virtual* (router with virtual TSV), *Serial* (router using serialization) and *Disabled* (routers with disabled vertical connections). We also compare the obtained results with "Normal w/o FT" (Normal without Fault-Tolerance), where no fault-tolerant methods are used and routers with defected vertical connections are disabled.

As shown in Figure 9, the system mostly operates without disabling any vertical connections with fault-rates under 50%. Thanks to the *Virtual TSV* and *Serialization* techniques, the routers having less than four clusters are still able to work. Even at less than 20% of defect-rate, there are less than 10% of serialization connections in all simulated layer sizes. With 50% of defect-rate and a 2×2 layer size, the disabled router rate is negligible with about 1.565%. This can be easily dealt using a light-weight fault-tolerant routing algorithm. When the layer size increases to be larger than 8×8, the number of disabled connections is mostly insubstantial. At 50% defect-rate, the disabled router ratios are nearly 0.63%, 0.50%, 0.44% and 0.42% with 8×8, 16×16, 32×32, and 64×64 layer sizes, respectively. However, these defect-rates are extremely high; thus, our proposed mechanism can be considered as highly reliable.

In comparison to the system without fault-tolerant methods, there are significant improvements concerning healthy connections, especially at large layer sizes. In Figure 9, the percentage of routers having four healthy TSV-clusters is represented by the "Normal w/o FT" curve. At 50% defect-rate, the average ratio of normal routers has been improved by 29.83%, 186.26%, 280.76%, 324.42%, 346.74%, and 257.79% for 2×2, 4×4, 8×8, 16×16, 32×32, and 64×64 layer sizes, respectively. The improvements are lesser with the small layer sizes such as 2×2 or 4×4. However, thanks to the *Virtual TSV* and *Serialization*, the workable connection rates have nearly reached 100%. As shown in Fig. 9, the *Serialization* technique is utilized under 10% in the low defect rate cases ($\leq$ 20%). Therefore, if the defect rates are expectedly low, designers can remove the *Serialization* technique from the proposed approach in order to save the overall area cost and power consumption without any significant impact on the system reliability.

In summary, this evaluation has shown a significant improvement in terms of reliability provided by our proposed mechanism. Thanks to the efficiency of the proposed architecture and algorithms, the system can mostly maintain all vertical connections, even at an extremely high defect-rate (50%). Although the defect rates are extremely high in comparison to the other works [15], [19] (maximum 1%), this evaluation aims to show the limitations of the proposed work when employing a significant amount of TSV defects. This evaluation also shows the proposed mechanism ability to remain efficiently scalable. The proposal can be applied from a small layer size (e.g., 2×2) to a larger one (e.g., 64×64). The evaluation is also performed with a solid number of tests (100,000) which powerfully demonstrates the efficiency of the proposed approach. There were some cases where some routers were disabled; however, they can be recovered by simple and light-weight fault-tolerant routing algorithms.

5.3 Performance Evaluation

The previous section has proved the reliability of the proposed solution. In this section, we evaluate the system performance under TSV-cluster defects. As we previously mentioned, works in [20] have demonstrated the low utilization rates of the vertical connections; nevertheless, the performance degradation on highly stressed networks has to be investigated. To evaluate the performance of the proposed system and keep fair comparisons

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

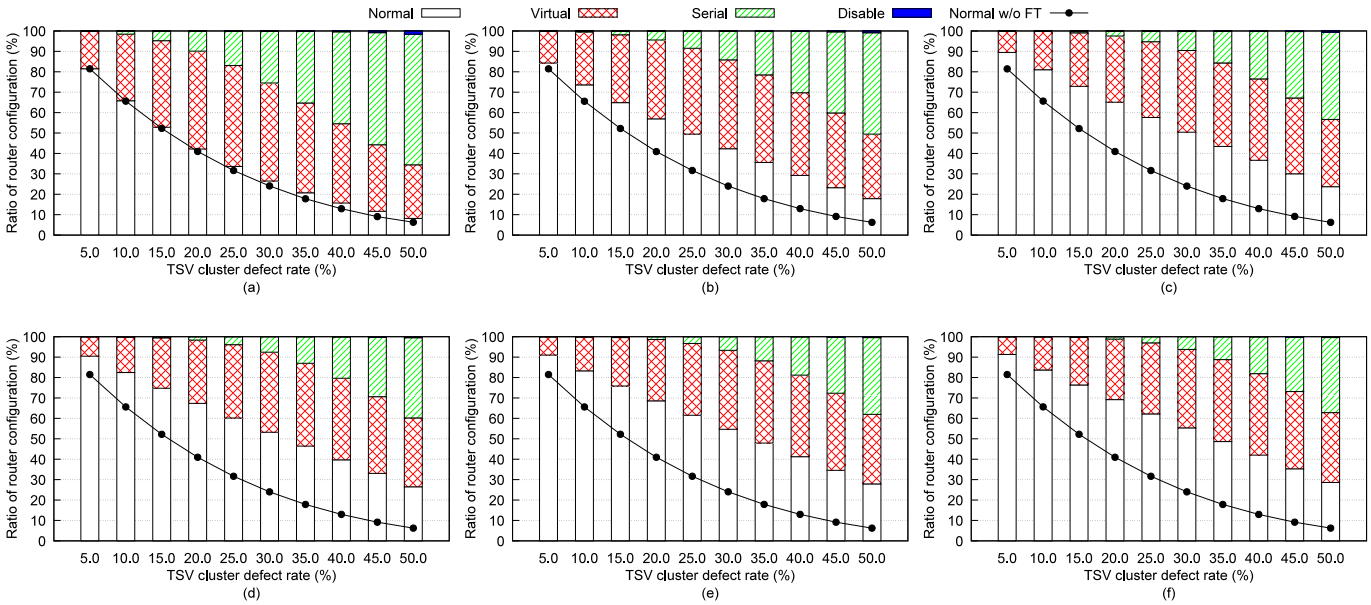

Fig. 9: Defect-rate evaluation: (a) Layer size: 2×2 (4 routers, 16 TSV clusters); (b) Layer size: 4×4 (16 routers, 64 TSV clusters); (c) Layer size: 8×8 (64 routers, 256 TSV clusters); (d) Layer size: 16×16 (256 routers, 1024 TSV clusters); (e) Layer size: 32×32 (1024 routers, 4096 TSV clusters); (f) Layer size: 64×64 (4096 routers, 16384 TSV clusters).

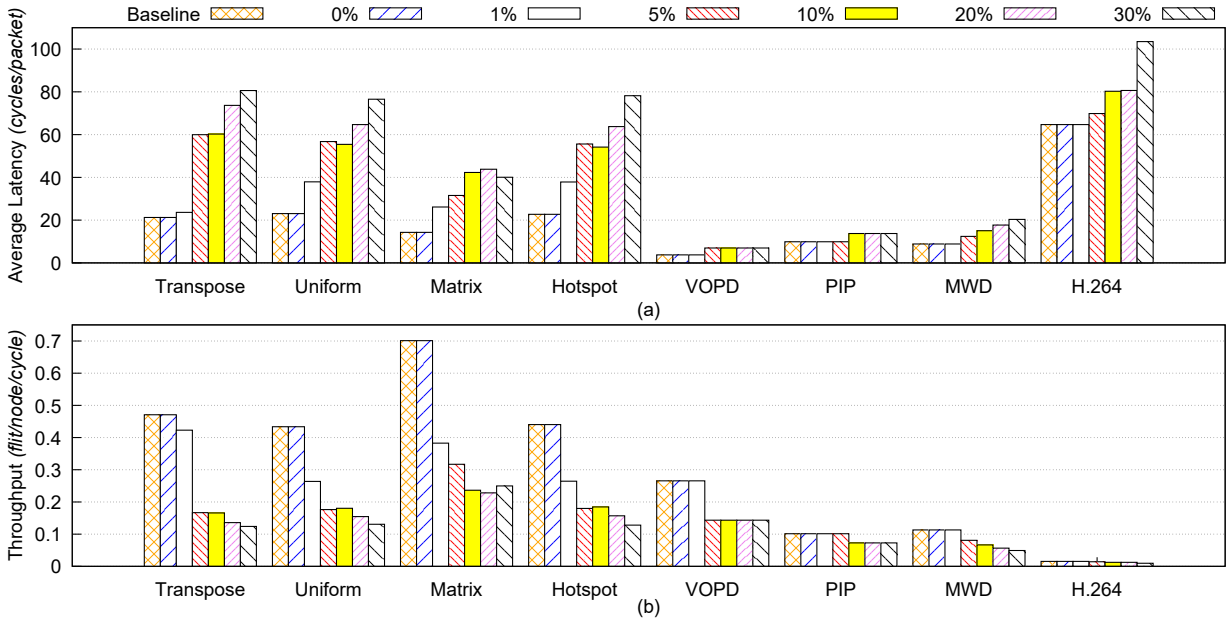

Fig. 10: Evaluation result: (a) Average Packet Latency; (b) Throughput.

TABLE 2: Simulation configurations.

Benchmark	Matrix	Transpose	Uniform	Hotspot
Network size (x,y,x)	(6, 6, 3)	(4, 4, 4)	(4, 4, 4)	(4, 4, 4)
#Packets	1,080	640	8,192	8,192
Packet's Size	10	10	10[1]	10
Benchmark	**H.264**	**VOPD**	**MWD**	**PIP**
Network size (x,y,x)	(3, 3, 3)	(3, 2, 2)	(2, 2, 3)	(2, 2, 2)
#Packets	8,400	3,494	1,120	512
Packet's Size	10	10	10	10

[1] For the hot spot nodes, there are additional 10% of flits.

to the baseline, we adopted both synthetic and realistic traffic patterns as benchmarks. We selected Transpose, Uniform, Matrix-multiplication, and Hotspot 10% [35] as the synthetic benchmarks. For realistic benchmarks, we chose H.264 video encoding system, Video Object Plane Decoder (VOPD), Picture In Picture (PIP) and Multiple Window Display (MWD) [35]. The configurations of these benchmarks are shown in Table 2. The packets are injected until the saturation point of the network is reached. In order to keep a fair comparison, only TSV defects are injected. This means that the other fault-tolerant mechanisms [35] are disabled to not affect the performance.

5.3.1 Latency Evaluation

In this experiment, we evaluate the performance of the proposed architecture in terms of Average packet Latency (APL) over various benchmark programs and defect-rates. The simulation results

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

are shown in Fig. 10 (a). From this graph, we notice that with a 0% of defect-rate, the fault-tolerant system has similar performance in comparison to the baseline system.

When we increase the defect-rates in the proposed system, it has demonstrated additional impacts on APL. At a 1% fault-rate using Matrix, Uniform, Transpose, and Hotspot 10% benchmarks, the system increases the APL by 83.24%, 64.46%, 11.30% and 66.55%, respectively. These high impacts are due to the occurrences of bottleneck inside the network. Because all vertical connections are utilized, Virtual TSV has caused congestion by sharing the TSV between two routers. The serialization is already a bottleneck technique. These bottlenecks effects are even higher at a 30% of defect-rate where the APL can be over three times that of the 0% case in the synthetic benchmarks.

With H.264, PIP, MWD and VOPD benchmarks, the APL incrementations is significantly reduced due to the low utilization rates of TSVs. We can observe the same performances of VOPD benchmark from a 1% to a 30% defect-rates. With the PIP benchmark, the system under 1% defect-rate has similar performance to 0% thanks to the optimization process which disables the unused clusters. With the MWD and H.264 benchmarks, the impact on APL is gradually increased when increasing the defect-rate. Even at a 30% of defect-rate, the APL values of MWD and H.264 are increased by 129.91% and 60.04%, respectively. Because there is no optimized routing technique for these benchmarks, the bottleneck effect is expected to happen.

5.3.2 Throughput Evaluation

Figure 10 (b) depicts the throughput evaluation with different benchmarks. At 0% defect-rate, the proposed system's throughput is similar to that of the baseline. When defects are injected into the system, we can observe some degradation in throughput caused by the bottleneck effects on the system. Similar to APL, the throughput degradation on realistic traffic benchmarks (VOPD, H.264, MWD, and PIP) is significantly better than the synthetic ones. The system at a 20% defect-rate provides a decreased throughput by 71.17%, 64.36%, 67.44% and 64.37% for Transpose, Uniform, Matrix, and Hotspot 10%, respectively. At the same defect-rate, VOPD, MWD, PIP and H.264 have 46.03%, 50.04% 28.17%, and 19.79% of throughput degradation. This lower impact is caused by the low utilization of vertical connection rate and the optimization process. The throughput values of realistic benchmarks are naturally smaller than the synthetic ones because of the specific tasks order of execution that was observed in the task graphs [36], [37].

Although there are considerable degradations in the throughput evaluation, the system still maintains over 0.1 *flit/node/cycle* in the highly stressed benchmarks, even at extremely high defect-rates.

5.3.3 Performance comparison

Table 3 shows the comparison results of our work with two other inter-layer fault-tolerant communication methods. The selected two works were presented in [38] and [37] which target fault-tolerant customized 3D-NoCs and hybrid-3D-NoC, respectively. Both two works support a maximum of one faulty vertical link. Since the benchmarks' configurations are not provided in [38], we used normalized values representing the performance ratio of the fault-tolerant works over the baseline ones.

As shown in Table 3, our work provides better performance when compared to the customized 3D-NoC [38]. At the absence of defected links, the hybrid-3D-NoC [37] shows improvements in terms of APL thanks to the efficiency of their routing algorithms.

TABLE 3: Normalized Average Packet Latency (APL) and Throughput (TP) comparison.

Benchmark	#Defect Link	[38]		[37]		This work	
		APL	TP	APL[1]	APL[2]	APL	TP
H.264	0	N/A		0.92	0.83	1	1
	1[4]	N/A		1.030	0.89	1.008	0.992
PIP	0	1.351	1.012	N/A		1	1
MWD	0	1.988	0.998	N/A		1	1
VOPD	0	2.630	0.900	N/A		1	1
Average[3]	1	2.536	0.338	1.030	0.89	1	1

[1] Routing algorithm: AdaptiveZ.
[2] Routing algorithm: AdaptiveXYZ.
[3] For [38], we used their value for seven benchmarks, with three layers configuration.
[4] In order to compare with [37], this work is inserted defect TSV clusters to create a defected all layers link.

However, it shows some degradation when a single defect is detected. On the other hand, our work maintains the similar APL and throughput values in the absence and presence of a single faulty TSV where the degradation is less than 1%. Moreover, the proposed technique even provides high reliability which allows the system works with multiple defected TSV clusters, as shown in Figure 10. The customized 3D-NoC suffers from a significant performance degradation due to the lack of routing paths and frequent occurrence of bottlenecks which increase the APL by nearly 2.5 times and reduce the overall throughput by nearly three times. In summary, our proposed technique provides the similar performance as the baseline one while providing high resiliency against TSV defects.

5.4 Router Hardware Complexity

Table 4 illustrates the hardware complexity breakdown of the proposed router in terms of area, power (static, dynamic, and total), and speed. In comparison to the router in which we implement the proposed techniques, the area and power consumption have increased by 30.42% and 18.66%, respectively. The maximum speed has also slightly decreased by 12.37%. In comparison to the baseline model, the proposed system almost doubles the area cost and power consumption while decreasing the maximum frequency by about 50%. However, the TSV sharing and Serialization modules incur reasonable area and power consumption overheads which are 47.99% and 38.89% in comparison to the baseline router, respectively. Here, the TSV Sharing module handles the sharing algorithm and the *Virtual TSV* process While the Serialization module helps the router communicating in *Serialization* mode. Notably, the Serialization module, which occupies 8.54% and 8.68% of the total area cost and power consumption, respectively, can be removed from the architecture. As we previously discussed, the need of serialization is usually necessary for high defect rates.

TABLE 4: Hardware complexity breakdown of a single router.

Model		Area (μm^2)	Power (mW)			Speed (Mhz)
			Static	Dynamic	Total	
Baseline router [34]		18,873	5.1229	0.9429	6.0658	925.28
Proposal	Router	29,780	10.017	2.2574	12.3144	613.50
	Serialization	3,318	0.9877	0.2807	1.2684	-
	TSV Sharing	5,740	0.7863	0.2892	1.0300	-
	Total	38,838	11.7910	2.8273	14.6128	537.63

The layout of a layer is shown in Fig. 11 where the sharing TSV areas are depicted by the red boxes. As shown in Section 3.2, a TSV sharing area consists of eight clusters. For each port, *R(1,1,1)* can access *T(E)* of *R(1,1,0)* and *R(1,1,0)* can access

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

TABLE 5: Comparison results between the proposed approach and the existing works.

Model	TSV Network [15]					TSV Grouping [19]			This work	
Technology	65 *nm*					N/A			45 *nm*	
#TSV	1000					6000			8448	
Configuration	4:2	8:2	$4\times4:8$	$8\times8:16$	$16\times16:32$	4:4	8:4	20:5	$11\times4\times4$:0	
#Spare TSV	512	256	512	256	128	6000	3000	1500	0	
45*nm* Arbiter Area (μm^2)	372 [2]	744 [2]	1,116 [2]	1,116 [2]	1,116 [2]	11,160 [1]	11,160[1]	12,555[1]	434,784[3]	
Average Area/TSV (μm^2)	151.572	126.244	152.316	126.716	128.03	113.916	151.86	127.09	151.47	
Reliability	100%	99%	100%	100%	100%	100%			100%	98.11%
Fault Assumption	(δ_{TSV} = 0.01%, α = 2)[4]					(δ_{TSV} = 1%,α = 2)[4]			(δ_c = 1%)[4]	(δ_c = 50%)[4]

[1] The authors use 2:1 multiplexers [19]. For comparison, we use the area cost of multiplexer from Nangate 45*nm* [32] (MUX2_X1: 0.186μm^2) .
[2] The authors use 1-to-3 multiplexers [15] which consists of two MUX2_X1 multiplexers ($2\times0.186\mu m^2$ [32]).
[3] For fair comparisons, our arbiter only consists of the TSV sharing and serialization modules, as shown in Table 4.
[4] δ_{TSV}: TSV defect-rate. α: parameter of Poisson distribution [15], [19]. δ_c: TSV cluster defect rate.

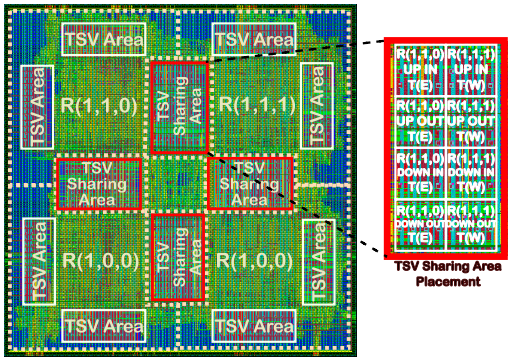

Fig. 11: Single layer layout illustrating the TSV sharing areas (red boxes). The layout size is $865\mu m \times 865\mu m$.

T(W) of *R(1,1,1)*. By placing the shared cluster areas between two routers, we can ensure a small extra wire delay for rerouting.

5.5 Comparison

In order to understand the efficiency of the proposed approach, we compare it with existing solutions for cluster TSV defect, as shown in Table 5. Here, we analyze our proposal with a network size of $4\times4\times4$. Because the router and its TSV clusters structure are identical, similar results can be obtained with the others network sizes. *TSV Grouping* [19] optimized the configuration of redundancy to deal with TSV-cluster defects. *TSV Network* [15] established TSVs into a network which allows routing from defected TSVs to redundant ones. We select the best results on these two works [15], [19] for the comparison. From this table, we can see that the average area of our proposal is 151.47μm^2 per TSV and, for a TSV size of $10\mu m\times10\mu m$, the area overhead is about 51.47%. The *TSV Network* [15] has a similar value for 4:2 configuration (4 original TSVs and 2 redundant TSVs). With 8:4 configuration, *TSV Grouping* also obtained an average area of 151.86μm^2. Because both *TSV Grouping* and *TSV Network* use redundant TSVs for recovery, the proposed method helps to reduce the total number of TSVs by eliminating the need for redundancy. In other words, the proposed approach relies on the existing number of TSVs and does not require any additional ones to maintain correct functionality.

On the other hand, the other configurations obtained lower area overheads. Nevertheless, we have to note that our arbiter not only consists of the rerouting circuit (similar to the multiplexers in *TSV Network* and *TSV Grouping*); but, also includes an online adaptive algorithm designed in hardware, in addition to the *Virtual TSV* and *Serialization* techniques. Both *TSV Grouping* and *TSV Network* have to require additional dedicated circuitries to recover from the cluster defects.

In terms of reliability, the proposed approach has proven its high resiliency, as previously shown in Section 5.2. *TSV Grouping* demonstrated a 100% of yield rate under a defect-rate of 1% and *TSV Network* obtained nearly 100% in the most cases. However, their approaches are different than our scheme, where they add redundancy to correct the defect TSVs. As a result, if the number of defected TSVs is larger than the number of redundant ones, they are unable to recover from the defected clusters. On the other hand, our technique can significantly improve the reliability by providing 98.11% of workable routers even at 50% of defected TSV-clusters. Moreover, at the low defect rates (e.g., under 5%), which is similar to [15], [19], our proposal also ensures 100% of working connections and demonstrates small performance degradation in the realistic traffic pattern benchmarks. Even with disabled vertical connections, the reliability of our system can also be improved by using a lightweight fault-tolerant routing.

6 Conclusion and Future Work

This paper presented an adaptive and scalable sharing methodology for TSVs in 3D-NoC systems to deal with the TSV-cluster defects. The results have proven the system's ability to provide high reliability that can reach up to 346.74% increase in functional routers. Moreover, the proposed approach can correctly work with a reasonable degradation, even under a 30% of defect-rate. The hardware complexity has shown a small overhead in terms of area cost (30.42%) and power consumption (18,66%) of router's logic. Since no TSV redundancy is not required in the proposed architecture and algorithms, we show that it is possible to provide a highly reliable system while maintaining the overhead reasonable.

As future work, the random TSV-defect is also an additional challenge for our 3D-NoC system. Furthermore, degradation factors on 3D-NoCs such as thermal dissipation, stress, operating voltages should be investigated.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. This work is partially supported by the University of Aizu Competitive Research funding (CRF), Ref. P-11-2016 and P-2-2017. This work is also supported by VLSI Design and Education Center (VDEC), the University of Tokyo, Japan, in Collaboration with Synopsys, Inc. and Cadence Design Systems, Inc. The first and the last authors in the author list are the main contributors of this work.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

References

[1] A. Ben Ahmed and A. Ben Abdallah, "Architecture and design of high-throughput, low-latency, and fault-tolerant routing algorithm for 3D-network-on-chip (3D-NoC)," *The Journal of Supercomputing*, vol. 66, no. 3, pp. 1507–1532, 2013.

[2] K. Banerjee *et al.*, "3-D ICs: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration," *Proc. IEEE*, vol. 89, no. 5, pp. 602–633, 2001.

[3] A. B. Abdallah and M. Sowa, "Basic Network-on-Chip Interconnection for Future Gigascale MCSoCs Applications: Communication and Computation Orthogonalization," in *Proc. of the Symp. on Science, Society, and Technology*, 2006, pp. 1–7.

[4] G. Van der Plas *et al.*, "Design issues and considerations for low-cost 3-D TSV IC technology," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 293–307, 2011.

[5] F. Ye and K. Chakrabarty, "TSV open defects in 3D integrated circuits: Characterization, test, and optimal spare allocation," in *Proc. 49th Annual Design Automation Conf.* ACM, 2012, pp. 1024–1030.

[6] U. Kang *et al.*, "8Gb 3D DDR3 DRAM using through-silicon-via technology," in *IEEE Int. Solid-State Circuits Conf.-Dig. of Tech. Papers*. IEEE, 2009, pp. 130–131.

[7] T. Zhang *et al.*, "Temperature-aware routing in 3D ICs," in *Asia and South Pacific Conf. on Design Automation*, Jan 2006, pp. 309–314.

[8] Y. J. Park *et al.*, "Thermal Analysis for 3D Multi-core Processors with Dynamic Frequency Scaling," in *IEEE/ACIS 9th Int. Conf. on Computer and Information Science*, Aug 2010, pp. 69–74.

[9] A. Eghbal *et al.*, "Analytical Fault Tolerance Assessment and Metrics for TSV-based 3D Network-on-Chip," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3591–3604, Dec 2015.

[10] T. Frank *et al.*, "Resistance increase due to electromigration induced depletion under TSV," in *IEEE Int. Reliability Physics Symp.*, Apr 2011, pp. 3F.4.1–3F.4.6.

[11] Y. Zhao *et al.*, "Online Fault Tolerance Technique for TSV-Based 3-D-IC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 23, no. 8, pp. 1567–1571, 2015.

[12] J. U. Knickerbocker *et al.*, "Three-dimensional silicon integration," *IBM J. Research and Development*, vol. 52, no. 6, pp. 553–569, 2008.

[13] M. Cho *et al.*, "Design method and test structure to characterize and repair TSV defect induced signal degradation in 3D system," in *Proc. Int. Conf. on Computer-Aided Design*, 2010, pp. 694–697.

[14] M. Laisne *et al.*, "Systems and methods utilizing redundancy in semiconductor chip interconnects," 2013, US Patent 8,384,417.

[15] L. Jiang *et al.*, "On effective through-silicon via repair for 3-D-stacked ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 32, no. 4, pp. 559–571, 2013.

[16] I. Loi *et al.*, "A low-overhead fault tolerance scheme for TSV-based 3D network on chip links," in *Proc. 2008 IEEE/ACM Int. Conf. on Computer-Aided Design*, 2008, pp. 598–602.

[17] D. Bertozzi *et al.*, "Error control schemes for on-chip communication links: the energy-reliability tradeoff," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst*, vol. 24, no. 6, pp. 818–831, Jun 2005.

[18] A.-C. Hsieh and T. Hwang, "TSV redundancy: architecture and design issues in 3-D IC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst*, vol. 20, no. 4, pp. 711–722, 2012.

[19] Y. Zhao *et al.*, "Cost-effective TSV grouping for yield improvement of 3D-ICs," in *Asian Test Symp.* IEEE, 2011, pp. 201–206.

[20] A. Kologeski *et al.*, "Combining fault tolerance and serialization effort to improve yield in 3d networks-on-chip," in *2013 IEEE 20th Int. Conf. on Electronics, Circuits, and Systems*, Dec 2013, pp. 125–128.

[21] A. Topol *et al.*, "Enabling SOI-based assembly technology for three-dimensional (3D) integrated circuits (ICs)," in *IEEE Int. Electron Devices Meeting. IEDM Tech. Dig.* IEEE, 2005, pp. 352–355.

[22] B. Swinnen *et al.*, "3D integration by Cu-Cu thermo-compression bonding of extremely thinned bulk-Si die containing 10 μm pitch through-Si vias," in *Int. Electron Devices Meeting*. IEEE, 2006, pp. 1–4.

[23] N. Miyakawa *et al.*, "Multilayer stacking technology using wafer-to-wafer stacked method," *ACM J. Emerging Technologies in Computing Systems*, vol. 4, no. 4, p. 20, 2008.

[24] N. Miyakawa, "A 3D prototyping chip based on a wafer-level stacking technology," in *Asia and South Pacific Design Automation Conf., 2009*. IEEE, 2009, pp. 416–420.

[25] R. P. Reddy *et al.*, "A Cost-Effective Fault Tolerance Technique for Functional TSV in 3-D ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 7, pp. 2071–2080, July 2017.

[26] D. Xiang *et al.*, "Practical deadlock-free fault-tolerant routing in meshes based on the planar network fault model," *IEEE Trans. Comput.*, vol. 58, no. 5, pp. 620–633, 2009.

[27] K. C. J. Chen *et al.*, "Thermal-Aware 3D Network-On-Chip (3D NoC) Designs: Routing Algorithms and Thermal Managements," *IEEE Circuits Syst. Mag.*, vol. 15, no. 4, pp. 45–69, Fourthquarter 2015.

[28] D. Xiang *et al.*, "Multicast-based testing and thermal-aware test scheduling for 3d ics with a stacked network-on-chip," *IEEE Trans. Compt.*, vol. 65, no. 9, pp. 2767–2779, 2016.

[29] Y.-J. Huang and J.-F. Li, "Built-in self-repair scheme for the TSVs in 3-D ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 10, pp. 1600–1613, 2012.

[30] M. Palesi *et al.*, "Application specific routing algorithms for networks on chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 3, pp. 316–330, Mar 2009.

[31] Y. Ghidini *et al.*, "Lasio 3D NoC vertical links serialization: Evaluation of latency and buffer occupancy," in *26th Symp. on Integrated Circuits and Systems Design*, Sep 2013, pp. 1–6.

[32] NanGate Inc., "Nangate Open Cell Library 45 nm," http://www.nangate.com/, (accessed 16.06.16).

[33] NCSU Electronic Design Automation, "FreePDK3D45 3D-IC process design kit," http://www.eda.ncsu.edu/wiki/FreePDK3D45:Contents, (accessed 16.06.16).

[34] A. Ben Ahmed and A. Ben Abdallah, "LA-XYZ: low latency, high throughput look-ahead routing algorithm for 3D network-on-chip (3D-NoC) architecture," in *IEEE 6th Int. Symp. on Embedded Multicore Socs*. IEEE, Sep 2012, pp. 167–174.

[35] K. N. Dang *et al.*, "A low-overhead soft–hard fault-tolerant architecture, design and management scheme for reliable high-performance many-core 3D-NoC systems," *The Journal of Supercomputing*, vol. 73, no. 6, pp. 2705–2729, 2017.

[36] D. Bertozzi *et al.*, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, 2005.

[37] A.-M. Rahmani *et al.*, "High-performance and fault-tolerant 3D noc-bus hybrid architecture using arb-net-based adaptive monitoring platform," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 734–747, 2014.

[38] K. S.-M. Li and S.-J. Wang, "Design methodology of fault-tolerant custom 3d network-on-chip," *ACM Trans. on Design Automation of Electronic Syst.*, vol. 22, no. 4, p. 63, 2017.

Khanh N. Dang received his B.Sc., M.Sc., and Ph.D. degree from VNU University of Engineering and Technology, University of Paris-Sud XI, and The University of Aizu, Japan in 2011, 2014, and 2017, respectively. His research interests include System-on-Chips/Network-on-Chips, 3D-ICs, and fault-tolerant systems. Since October 2017, Dr. Khanh N. Dang has moved to SISLAB, The University of Engineering and Technology, Vietnam National University Hanoi, Hanoi, Vietnam He is a Student Member of IEEE.

Akram Ben Ahmed received his M.S.E. and Ph.D. degrees in Computer Science and Engineering from the University of Aizu, Japan, in 2012 and 2015, respectively. He is currently a postdoctoral researcher in the Department of Information and Computer Science, Keio University, Japan. His current research interests include on-chip interconnection networks, reliable and fault-tolerant systems, and ultra-low-power embedded real-time systems. Dr. Akram Ben Ahmed is a Member of IEEE.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TETC.2017.2762407, IEEE Transactions on Emerging Topics in Computing

Yuichi Okuyama is an associate professor at the University of Aizu. He received his Masters and Ph.D. degrees in computer science and engineering from the University of Aizu in 1999 and 2002 respectively. His research interests include reconfigurable architecture design, parallel programming for pattern recognition and education of computer fundamentals.

Abderazek Ben Abdallah received the Ph.D. degree in computer engineering from the University of Electro-Communications at Tokyo, Chofu, Japan, in 2002. He was a Research Associate with the Graduate School of Information Systems, University of Electro-Communications at Tokyo, from 2002 to 2007. He is currently a Full Professor of computer science and engineering and the Head of the Division of Computer Engineering, The University of Aizu, Aizuwakamatsu, Japan. He has been a Faculty Member with The University of Aizu since 2007. He has authored three books, published over 150 journal articles and conference papers in these areas, received numerous awards, and given invited talks and courses at several universities and conferences worldwide. His current research interest is in the area of computer system and architecture, with an emphasis on adaptive/self-organizing systems, network-on-chip/system-on-chip, processor microarchitecture, power and reliability-aware architectures, neuro-inspired systems, and VLSI design for 3-D-ICs. Dr. Ben Abdallah is a Senior Member of ACM and IEEE and a member of IEICE. He has been a Principal Investigator or a Co-Principal Investigator of several projects for developing next-generation high-performance reliable computing systems for applications in general purpose and pervasive computing.