HotCluster: A thermal-aware defect recovery method for Through-Silicon-Vias Towards Reliable 3-D ICs systems

Khanh N. Dang, Akram Ben Ahmed, Abderazek Ben Abdallah, and Xuan-Tu Tran

***Abstract*—Through Silicon Via (TSV) is considered as the near-future solution to realize low-power and high-performance 3D-Integrated Circuits (3D-ICs) and 3D-Network-on-Chips (3D-NoCs). However, the lifetime reliability issue of TSV due to its fault sensitivity and the high operating temperature of 3D-ICs, which also accelerates the fault-rate, is one of the most critical challenges. Meanwhile, most current works focus on detecting and correcting TSV defects after manufacture without considering high-temperature nodes' impact on lifetime reliability. Besides, the recovery for defective clusters is also challenging because of costly redundancies. In this work, we present *HotCluster*: a hotspot-aware self-correction platform for clustering defects in 3D-NoCs to help understand and tackle this problem. We first give a method to predict normalized fault rates and place redundant TSV groups according to each region's fault rate. In our particular medium fault-rate (normalized to the coolest area), *HotCluster* reduces about 60% of the redundancies in comparison to the uniformly distributed redundancies while having a higher ratio of router working in a normal state. Furthermore, *HotCluster* integrates both online (weight-based) and offline (max-flow min-cut offline method) mapping algorithms to help the system correct the faulty TSV clusters. The experimental results show that both the max-flow min-cut offline method and weight-based online mode with a redundancy of 0.25 exhibits less than 1% of routers disabled under 50% defect-rates.**

***Index Terms*—3D-NoCs, Fault-tolerance, Reliability, Architecture and Design, Through Silicon Vias, Maximum flow minimal cut.**

I. Introduction

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

Khanh N. Dang and Xuan-Tu Tran are with the VNU Key Laboratory for Smart Integrated Systems (SISLAB), VNU University of Engineering and Technology, Vietnam National University, Hanoi, Hanoi 123106, Vietnam, (Corresponding authors' e-mail: {khanh.n.dang;tutx}@vnu.edu.vn). Xuan-Tu Tran is also with VNU Information Technology Institute (VNU-ITI).

Akram Ben Ahmed is with National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba 305-8568, Japan.

Abderazek Ben Abdallah is with Adaptive Systems Laboratory, Graduate School of Computer Science and Engineering, the University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan.

Despite having many advantages, 3D-ICs exhibit two major drawbacks: (1) *reliability* and (2) *difficulty in thermal dissipation*. In terms of reliability, due to the imperfection of the manufacturing process, the yield rates of TSV-based 3D-ICs have been considered as a critical factor [3], [4]. Moreover, by stacking multiple layers of wafers, 3D-ICs suffer from the stress issue due to the difference between thermal expansion coefficients of the implementation materials [5] that could create defects in TSVs. The temperature variation between two layers has been reported to reach up to 10°C [6] which negatively affects the *Time Dependent Dielectric Breakdown*, *Thermal Cycling* and *Electromigration* [7]. Here, Cooper TSV lifetime expectancy can be predicted using the Black's model [8] where the fault rate is accelerated exponentially by the operating temperature. As a result, TSV is one of the most thermally critical parts in terms of reliability in 3D-ICs.

Even within a single layer, there are also temperature differences since there are hotter and cooler areas [9]. Because of these differences in temperature, TSVs on different positions and the layers have different lifetime expectations. As we later investigate in Section IV-B, the fault-rate accelerates exponentially with the operating temperature in most academic and industry models [8], [10]. Apparently, hotspot areas must be focused on when it comes to fault-tolerance since: *(1)* the probability of having defects is higher and *(2)* this area plays an important role in the operation because of the high computation/communication utilization. Although using advanced cooling methods such as liquid or cooling TSVs could help reducing the operating temperature, these methods are still immature for near-future applications.

As explained in details in Section II, existing works presented so far have dealt with the high fault-rate of TSVs in different approaches: *(1)* improving the manufacturing process to enhance the reliability of TSVs [11]; *(2)* accounting the potential defects in the design stage [5] [12]; correcting the defected TSVs by using supporting circuits [13]–[15], redundancy [16]–[19], or *Error Correction Codes* [20]; and (3) using an alternative channel to avoid the defected TSV channel (e.g., using fault-tolerant routing [2], [21] in NoCs). To tackle the thermal issue, hotspot/thermal aware routing [22] is proposed; in addition, advanced cooling [9] can be another solution.

Although these works have significantly enhanced the reliability of TSV-based systems, there are two major issues that need to be addressed:

1) *Clustering distribution of TSV defects is a challenging*

problem to be addressed: While adding redundancy for corrections works well with random defects, it is not feasible for clustering defects since the number of defects in a group is too high. Adding a high volume of redundancies for each group is also impractical due to the high area overhead. In [21], we propose a method for correcting clustering TSV defects without redundancy in 3D-NoCs; however, it leads to serialized or disabled vertical connections, which degrade the system performance.

2) *Thermal-impact on TSV defect:* where most TSV fault-tolerant techniques focus on correcting the defects after manufacturing, they also need to investigate the impact of thermal distribution on lifetime reliability. Hotspots usually have higher chances of faults and could accelerate the crack or misalignment in TSVs. This high-temperature issue also correlates strongly with the reliability, which leads to clustering TSV defects.

Starting from the facts mentioned above, in this paper, we propose *HotCluster*: a thermal-aware correction framework for clustering defects in 3D-NoCs. Based on our previous design [21], each router has four clusters around it and could use its neighbors' clusters to establish communication. Here, we first add redundant clusters within the layer to allow a better recovery rate. Moreover, we design an algorithm that considers the operating temperature to support certain specifications and layouts. Experimental results show that the proposed solution can help 3D-NoCs to work around TSV-cluster defects without degradation at a certain fault-rate. Therefore, the reliability at a reasonable overhead is guaranteed and improved thanks to the thermal-aware design. We also solve the problem of reducing the amount of redundancy while still ensuring the same reliability. Compared to our previous work [21], the followings are the new main contributions:

- Four options are presented to address the placement of redundant TSV clusters. They are *int. red.*: each router has one redundant TSV cluster, *ext. red.*: adding redundant clusters at the border of the layer, *hyb. red.*: combination of *int. red.* and *ext. red.*, and *irr. red.*: use our *HotCluster* framework to place redundant TSV clusters.
- When addressing the solution for *irr. red.*, a thermal aware spare TSV cluster placement is introduced to reduce the number of needed redundancies while still maintaining the same reliability with uniform placement (*int. red.*, *ext. red.*, and *hyb. red.*).
- Online and offline algorithms that help the recovery of defective TSV clusters. We first present SAWI (Spare Availability Weight Initialization) to generate the first weight for online recovery. We offer a method to build, adapt, and solve the issue using the Ford-Fulkerson method for offline recovery. The Ford-Fulkerson method is considered as the optimal method for offline mapping without considering the priority of the router.

In summary, the *HotCluster* platform combines the proposed techniques, architectures, and algorithms for tackling the thermal-induced defects in TSVs. This paper provides a comprehensive approach for protecting TSV clustering defects in 3D-ICs because we offer both online and offline mapping and placement consideration, in addition to the recovery from the thermally accelerated faults. The paper is organized as follows. Section II presents the prior works. In Section III, we describe the proposed TSV fault-tolerant architecture. Section IV discusses about the thermal issues and introduces the appropriate solutions. Section V shows our evaluation and comparison results. Finally, Section VI concludes the paper.

II. Related Works

In this section, we cover the reliability issues of TSVs in 3D-ICs. Then, the fault correction methods are also summarized. We also discuss the thermal/hotspot awareness in designs and run-time techniques.

A. Reliability Issues of TSV-based 3D-ICs

The defect-rate of TSVs is considerably high which negatively affects the final yield. In [4], 0.63% of the TSVs are reportedly defected and the final yield without spare is only 15%. Besides the high defect-rate during the manufacturing stage, TSVs under operation also face several challenges with stress and thermal issues, as reported in [5]. As a result, TSVs are one of the most vulnerable components in 3D-ICs.

One of the matters which is still under investigation is the TSV failure distribution. In general, there are two main assumptions for the failure distribution: *Random* [23] and *Clustering* distributions [3], [16], [17], [19], [24]. *Random* TSV defects are efficiently dealt with by adding redundancy and recovery methods; but, *Clustering* defects still remain as an important challenge. Because of the stress and thermal issues, TSVs may also be defected after manufacturing. In [7], the authors presented several *Mean Time To Failure* equations of 3D-ICs when affected by *Time Dependent Dielectric Breakdown*, *Thermal Cycling* and *Electro-migration* where the temperature values play an important role. Because of the clustering thermal behavior in 3D-ICs [16], the obvious result was found to be the TSV-cluster defect.

B. TSV Fault Tolerance

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

In *Physical layer*, an improvement of TSV manufacturing can help reduce the defect-rate [11]. Designers can optimize the physical layout, use thermal-aware routing and placement methods to improve the reliability of 3D-ICs [5]. 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 to add redundant TSVs to correct the defected ones [16], [18]. The

major concern in this method is to route from a defected TSV to a spare one efficiently. There are several solutions that have been proposed to place and route the spare TSVs such as signal switching [4], single shifting [23], crossbar, router [16], ring [25], honeycomb [17] and cobweb [19]. Because of the cluster defect, adding redundancies becomes a costly technique with a high number of needed spare TSVs (up to 50% in [16], [24]). In [3], the authors propose a mapping method to reduce the impact of cluster defects. TSVs in the same group are mapped into random positions with the help of an optimization process. On the other hand, *Zhao et al.* [24] analyzed the grouping method to achieve the best recovery. The router method in [16] introduces a new approach for TSV mapping by creating a network and implementing an algorithm for re-routing the defected TSVs. Since the router or the ring methods have limitations on the maximum flow from the faulty TSVs to the spare ones, the works in [17] and [19] present alternative shapes such as honeycomb or cobweb to tackle the bottleneck on the flow, which allow more efficient recovery. On the other hand, *Ni et al.* [26] proposed a Time Division Multiplexing Access for TSVs, which can help to correct defects with low area overheads. As defects induced EM of TSVs is an important issue, *Chen et al.* [12] analyzed and proposed a framework to balance the current flow and provide recovery methods to enhance the system reliability. Because TSVs manage the vertical connections in a 3D-NoC, *Error Correction Coding* [20] is also a prominent method for detecting and correcting the defected TSVs. However, this type of solution requires extra bits, significantly increasing the area cost and power consumption.

In the *Network layer*, where we mainly focus on 3D-NoCs, using a fault-tolerant routing algorithm [2] is one of the most suitable 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, most of these works proposed off-line testing and recovery schemes that are not suitable for post-manufacturing. The system's operation has to be a halt in order to be tested and recovered. In [18], the authors presented an online testing function. 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 occur frequently. The most efficient solution for correcting random defects is the grouping and adding redundancy. Nonetheless, they are still inefficient for the cluster defect and require an expensive extra area for redundancy. Therefore, fault-tolerance for cluster defect is the main interest of this paper.

C. Thermal/Hotspot Awareness

As we previously discussed, the high temperature drives higher fault-rates in semiconductor devices, in general. Apparently, 3D-ICs also encounters heat dissipation issues that increase their steady temperatures [28].

To solve the high temperature/hotspot issues, there are two approaches: (1) *advanced cooling* such as thermal TSVs or liquid cooling [9], [29], [30], and (2) *design/algorithm aware method* [22], [31]. For *advanced cooling*, works in [29], [30] explored the ability to use TSV as cooling devices. *Cuesta et al.* [9] showed that liquid micro-channels could further reduce the steady temperature. While these works show prominent results, they are difficult to be manufactured and maintained due to their complex structures and high area overheads. Therefore, another method is to prevent runtime defects by smartly lowering down the operating temperature using design/algorithm methods.

To solve the thermal issue from a design/algorithm aspect, most works try to limit the highest temperature by cooling down the hotspots. *Cong et al.* [32] took into account the temperature in the cell and the TSV placement while [5] considering the position of the "thermal TSV" to cool down the chip. On the other hand, designing hotspot-/thermal-aware design/algorithms for 3D-NoCs is another critical challenge. The work in [27] introduced the design for 3D-NoCs with both thermal management and routing algorithm. *Chao et al.* [22] used traffic as a measurable value for allowing lower temperatures. In order to recover efficiently, *Wang et al.* [33] showed the thermal awareness placement for spare and routing. On the other hand, the work in [31] explored the trade-off between lowering the power consumption by allocating heavily communicating tasks within the same vertical stack and the resulting hotspots due to stacking these tasks. Although this type of method could reduce the operating temperature, the final temperature is still significantly high in their experiment, which leads to the fact that the TSVs in hotspots have a high defect-rate. Since TSVs in the same area have similar temperature, it could lead to defective regions instead of random ones. Consequently, clustering defects could happen making adding spares inefficiently.

III. 3D Network-on-Chip Architecture

In this section, we first show the system architecture of our preliminary work [21] named *TSV-Sharing* which is the baseline of this work. Then, we analyze the problems of this preliminary work and present the architecture which consists of additional redundancies for recovery. This architecture will be the backbone for the *HotCluster* platform in Section IV.

A. TSV defect pattern

As being presented in [4], [18], [23], TSV defect can be illustrated as a random distribution and corrected by simply adding redundancies. In [16], the authors suggest about the potential clustering defects. In our preliminary work [21], we also adopt the clustering distribution into account as our major target.

Figure 2 illustrates both random and clustering distributions. A typical TSV group is organized with functional and spare TSVs as in Figure 2 (a). Once random defects, the group can recover by using spare TSVs (Figure 2 (b)). However, a clustering defect leads to a high number of defects per group, making the TSV group unrecoverable, as in Figure 2 (c). Finally, Figure 2(d) illustrates our approach by adding redundant groups for recovery. As adding redundant clusters has high area overhead, we will optimize by using our method in Section IV.

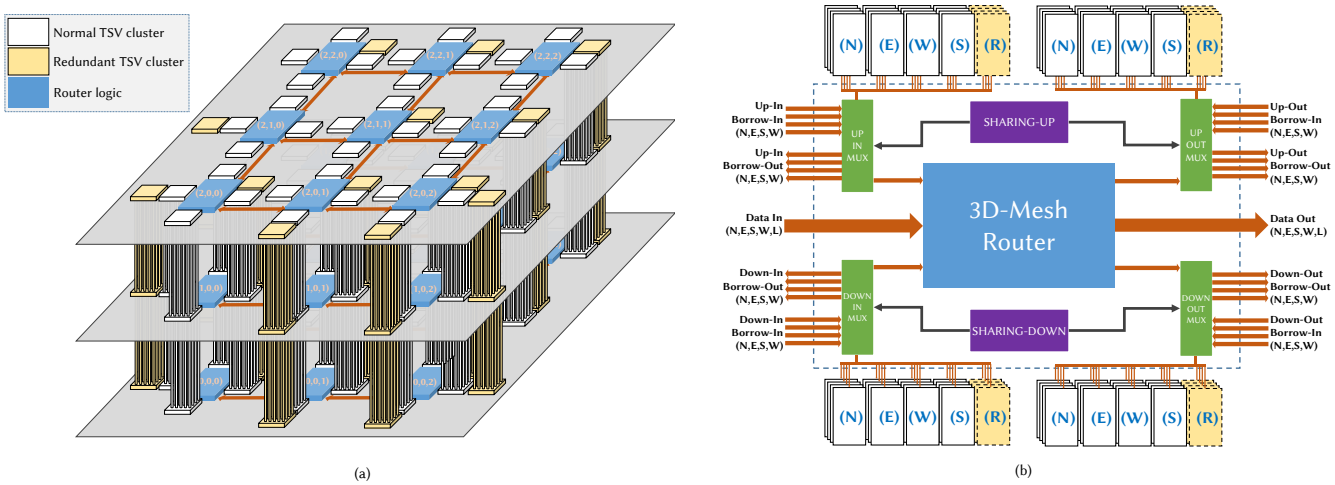

Fig. 1: The proposal TSV-based 3D-NoC: (a) Example of four layers with *irr.* redundant TSV cluster; (b) Router connection.

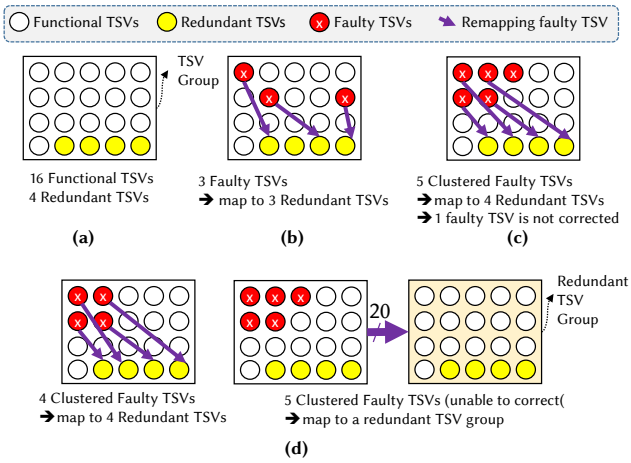

Fig. 2: TSV defect distribution and recovery approach: (a) TSV group with redundancy (16:4); (b) recovery from random defects with redundancy; (c) failing to recover clustering defects due to the lack of spares; (d) adding a redundant group to recover the failing one.

B. Preliminary work

In [21], our solution is to share TSVs between neighboring routers. The TSVs of a router are divided into four clusters. In [21], we use tri-state gates to redirect the signal from its original TSV to the spare one; therefore, we can support both uni- and bi-directional TSVs and we can share the spare TSVs between two directions (up and down). 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 designs optimization methods to improve the reliability of the system. In this previous work, we did not add any redundancy to maintain the low area overhead.

Once there are defective TSV clusters, the higher-weighted router borrows a cluster from a lower-weighted nearby one. Consequently, the system creates a chain of borrowing from the highest-weighted router to the layer's lowest-weighted ones. The lower-weighted routers usually have less than four clusters to perform their communications. To maintain their connectivity, work in [21] proposed two solutions: (1) *Virtual-TSV*: temporally borrow one or more TSV clusters from a higher weighted router to perform communication; (2) *Serialization*: use one or two TSV clusters left to perform the communication in a serial mode (4:1 or 2:1). The router that cannot perform either *Virtual-TSV* or *Serialization* will use a *fault-tolerant routing* algorithm to avoid the disconnected link.

Despite its efficiency in recovering defective TSVs and maintaining communication, *TSV-Sharing* [21] has two significant drawbacks. The first drawback is that it does not support redundant clusters, leading to performance degradation due to the three fault tolerating techniques. In fact, *virtual-TSV*, *serialization* and *fault-tolerant routing* can create bottlenecks within the network. The second one is it does not consider the impact of thermal on the reliability and assumes the center of the layer routers as the highest weighted ones. Therefore, in this work, we redesign the architecture and algorithm to overcome this preliminary work's problems.

C. System design

The proposed system architecture is shown in Figure 1 where each router connects at most four neighboring intra-layer (north, east, south, and west) and two neighboring inter-layer (up and down) routers. Note that the number of neighbors might be reduced at the borders/corners of the NoC topology. Figure 1(b) shows the router architecture which consists of two parts: *(1) normal routing logic* (blue box) and *(2) sharing/borrowing logic* (around blue box).

1) Routing logic: The routing logic handles the communications between routers and PEs inside the network. It receives packets and routes them in proper directions. Since our method could work with any 3D-Mesh NoC or even other NoC that utilize TSV-based connections, we do not limit routing logic selection. Here, we adapt our previously developed 3D-Mesh NoC [2] that has Look-Ahead-Fault-Tolerant (LAFT) routing algorithm to re-route the packets if needed. We want to note that other NoC designs could be easily integrated.

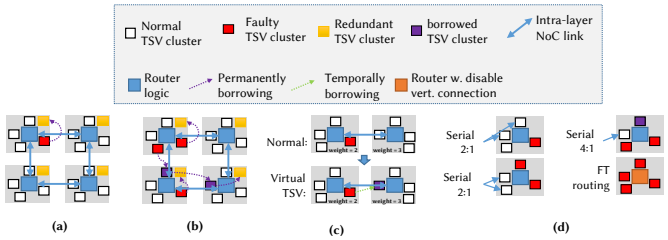

Fig. 3: TSV Borrowing Mechanism and Fault-tolerance cases: (a) Repair internally using redundant cluster. (b) Repair internally and externally using redundant and nearby clusters. (c) Virtual TSV with lower weighted (2) router temporally borrows a cluster from a higher weighted (3) one. (d) Serial 2:1 with 2-3 normal clusters, 4:1 with 1 normal cluster and fault tolerant routing with zero normal cluster cases.

2) Sharing/borrowing logic: The sharing/borrowing method is inherited from our previous work [21]. Each TSV-based connection is divided into four clusters and placed in four borders of the router (see white TSV cluster Figure 1). Compared to [21], we offer an extra internal redundant cluster within the router (yellow TSV cluster). This allows the cluster to be separated, which avoids clustering defects and fewer borrowing wires between routers. Based on the clustering design, each router naturally has four original TSV clusters, optionally one redundant cluster, and at most four nearby clusters.

Figure 3 (a) shows the case of self-repair with an internal TSV cluster. Figure 3 (b) represents the borrowing mechanism. When there is a defective cluster, the router tries to correct using a redundant cluster within the router. However, if there are two or more defective clusters, it must borrow from its neighbors to correct them. Figure 3 (c) shows the case of *Virtual TSV* where the lower-weighted (2) router requests to borrow a cluster from a higher weighted one temporally. After it completes its packet transmission, it returns the cluster to the higher-weighted router. Figure 3 (d) represents the *serialization* 2:1 and 4:1 and the *fault-tolerating routing* case. If there are two or three available clusters, the router can perform 2:1 *serialization*. If there is one available cluster, a 4:1 *serialization* can be utilized. In the case of a non available cluster, a *fault-tolerant* routing is used to find an alternative path of transmission.

D. Redundancies

One of the significant problems of the work presented in [21] is it runs into *serial* (use 25% or 50% of TSV for transmission) or *virtual* (sharing clusters between two routers) mode when a defective cluster occurs due to lack of redundancy. Adding clusters is needed to help correct faulty clusters that improve the overall performance since the system does not need to perform virtualization or fault-tolerant routing. Here, we offer two options for adding redundant clusters:

1) *Adding redundant clusters at the border of a layer*: as the principle of borrowing is to find a cluster nearby, adding redundancy at the border of the layer can inherit the algorithm in [21], where the border can be considered as a virtual router with the lowest weight. Here, the system can borrow the TSV cluster outside of its layer.
2) *Adding redundant clusters within the router*: We notice that adding redundant clusters within the router is more efficient; especially, borrowing chains can block each other. Because hotspot areas can be more faulty, we can add more redundant clusters to ensure system reliability.

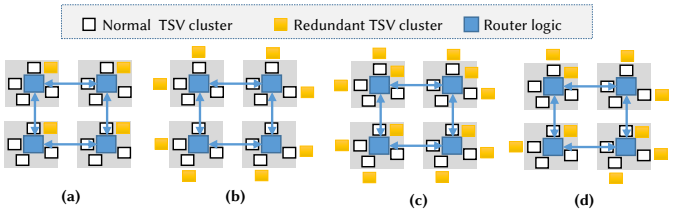

Fig. 4: The redundant clusters: (a) *int.red.*: Within each router; (b) *ext.red.* Outside of the layer; (c) *hyb.red.*: Outside of the layer and within each router; (d) *irr.red.*: Irregular assignment from the *HotCluster* framework.

Figure 1 shows the proposed TSV NoC architecture. The topology is 3D-mesh, where redundant (spare) TSV clusters could be added to the router and outside of the layer. The borrowing mechanism is presented in Figure 3. Figure 4 shows four examples of a 2×2 layer which consists of: *(a)* redundancies within each router (*int.red.*); *(b)* redundancies outside of the layer (*ext.red.*); *(c)* combination of (a) and (b) (*hyb.red.*); and *(d)* an irregular assignment by the *HotCluster* framework (*irr.red.*).

Besides the external, internal, and hybrid options, we also consider the irregular cases (*irr.*) as it will be later designed in Section IV. Here, naively placing the redundant cluster can cost a large area. In [34], the authors analyzed the cost of TSV where a $5\mu m$ TSV is around $150\times$ of a 28nm NAND gate, which creates a large overhead for adding redundancy. Therefore, in this work, we consider the placement of *irr.* by taking into account the thermal map of 3D-ICs. Another important issue is a potential timing violation due to the increase of delay brought by high temperature regions. Here, we can treat the violated timing TSVs are defective and temporarily provide a recovery method until it cools down, or permanently if there are spare TSVs. Another approach is to reduce the frequency of the connection to satisfy the timing requirements. Nevertheless, this issue should be addressed with a low-level (for detecting timing issue) system management mechanism (for frequency adaptation). Therefore, in this paper, we only consider the permanent defective TSVs as the target.

IV. Proposed HotCluster Platform

A. Platform overview

Figure 5 illustrates our proposed *HotCluster* platform. It takes the system model (e.g., Verilog HDL), benchmarks or applications, and the configurations (3D-IC structure, heatsink, TSV technology, library) as the inputs. The first phase is to assign redundant TSV clusters. Here, the platform performs a fault rate prediction, which is later shown in Section IV-B.

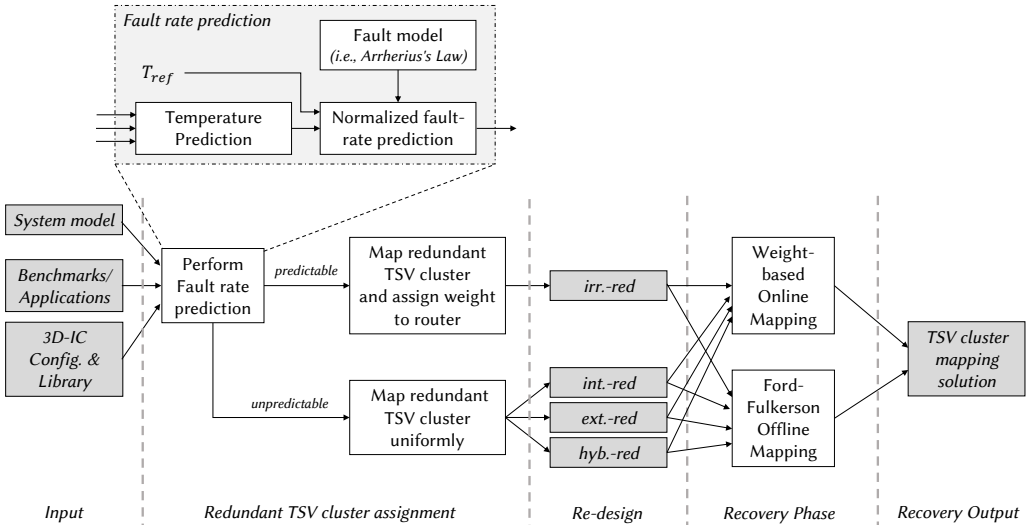

Fig. 5: The proposed *HotCluster* platform.

By predicting the critical regions (usually the hotspot area), it inserts redundant clusters for recovery (see Section IV-C). Obviously, there is a case where the fault rates are undetermined (i.e., unknown applications) which leaves the system to insert the TSV clusters uniformly (i.e., *int.-red.*, *ext.-red.* or *hyb.-red.* in Figure 4). The redundant patterns are used in the redesign phase. Once the system is manufactured, the detected faulty clusters are recovered with an online mapping method (weight-based) or an offline mapping method, which are both presented in Section IV-D.

B. Fault rate prediction

The model of thermal accelerated fault rate has been covered in academic and military documents [8], [10]. In general, the operating temperature exponentially accelerating the fault-rates is used in most models. Here, we consider the fault-rate following Arrherius' Law [10] where π_T is the acceleration rate of fault as follows:

$$\pi_T = A \times e^{-\frac{E_a}{k_B T}} \tag{1}$$

Three constants A, E_a and k_B are the pre-exponential factor, activation energy and Boltzmann constant, respectively. Here, we assume that each cluster has a base defect-rate of λ_b. Then, the defect-rate is accelerated with a temperature factor of π_T:

$$\lambda_T = \lambda_b \pi_T = \lambda_b A e^{-\frac{E_a}{k_B T}} \tag{2}$$

By using a reference temperature ($\mathsf{T}_{\mathsf{ref}}$), we can estimate the accelerated factor regardless of the constant value. For example, as HDR4 [10] used 70°C as the threshold voltage, the acceleration factor is as follows:

$$\lambda_T / \lambda_{\mathsf{T}_{\mathsf{ref}}} = \frac{\lambda_b A e^{-\frac{E_a}{k_B T}}}{\lambda_b A e^{-\frac{E_a}{k_B \mathsf{T}_{\mathsf{ref}}}}} = e^{\frac{E_a}{k_B}(\frac{1}{\mathsf{T}_{\mathsf{ref}}} - \frac{1}{T})} \tag{3}$$

As shown in Equation 3, the normalized fault-rate only depends on the activation energy (E_a) and the operating temperature (T). Here, we would like to note that in prior works on TSV lifetime reliability, the authors used Black's model [8]; however, as we normalize it based on temperature, both models (Black's and Arrhenius') lead to the Equation 3.

In summary, by using a normalized value, we can illustrate the normalized fault-rate value of a high-temperature point. By knowing the normalized fault-rate, we can estimate the needed redundancies and design a suitable layout and algorithm for recovery. Please note that designers can adopt different models for this phase.

C. TSV cluster placement

There are two situations that can be observed in the proposed *HotCluster*: (1) *unpredictable fault rate* and (2) *predictable fault rate*. If the fault rates (or temperature map) are unpredictable, the best is to place the ration of redundancy (or repair rate) Re_h as a certain value (i.e., $Re_h = 1$). Depending on designers' analyses, the redundant clusters could be placed at the center, uniformly, or outside of the layer. In this work, we assume that there are three approaches that designers can use: *int.-red.*, *ext.-red.* or *hyb.-red.* in Figure 4. Obviously, designers can insert redundancies using different patterns (i.e., several routers have one redundant cluster or a router has several redundant clusters); however, this work focuses on the thermal-aware method where the hotspots can be predicted. Therefore, we suggest that different patterns should be later investigated. Even with varying patterns of redundancy, the recovery algorithms in the later section can also be performed.

If hotspot areas and their temperatures can be predicted, which allows us to predict the normalized fault rates in Eq. 3, we optionally place one redundant cluster within the router. It allows the system to repair itself without creating an extended borrowing chain immediately. Second, we must pre-setup a chain of borrowing with redundancy. As previously explained, the borrowing mechanism is already supported; however, placing redundancy in the chain can increase the success rate.

The TSV cluster placement and weight assignment flow for our architecture are presented in Algorithm 1. The input of

Algorithm 1: TSV cluster placement and weight assignment algorithm.

```
   Input: NF_h[rows][cols] ;                    // predicted faults
   Output: Re_h[rows][cols] ;                   // redundancies map
   Output: W[rows][cols] ;                            // Weight map
   Output: Un[rows][cols] ;                    // Uncorrected map
1  visited[rows][cols] = all false;
2  W_max = rows*cols;
3  while (visited[rows][cols] == all true) do
4      (r,c) = find maximum NF_h and unvisited router;
5      visited[r][c] = true;
6      W[r][c]=W_max;
7      W_max =- 1;
8      if (NF_h[r][c]! = 0) then
9          Re_h[r][c] = 1;
           // insert redundancy
10         NF_h[r][c] = NF_h[r][c] − Re_h[r][c];
           // correct with redundancies
11         for ([x,y] in ([r-1,c], [r+1,c], [r,c+1], [r,c-1]) ) do
12             if (NF_h[r][c] == 0 and early_break == true ) then
13                 break the for loop;
14             if (visited[x][y] == false and NF_h[x][y] < 4) then
15                 NF_h[x][y] + +;
16                 NF_h[r][c] − −;
17         if (NF_h[r][c] > 0) then
18             fail to correct;
19             Un[r][c] = true;
20         else
21             Un[r][c] = false;
22 return Re_h,W;
```

the algorithm is the number of faulty TSV groups in each router NF_h. The algorithm's outputs are the redundancy map, the weight of each router, and the uncorrected router map. The algorithm performs a heuristic loop to scan through all routers in the network. It tries to find the router with the maximum expected fault to perform the mapping. Consequently, it always finds the higher temperature routers first and the cooler ones later. Once it finds the hotspot router, which is not visited yet, it assigns the weight to this router (line 6). Note that the weight decreases in each loop, making the hotspot router a higher weight than the cool ones. Then, it starts to insert redundancy to correct the defects (line 7) if there is at least one. If there are uncorrected defects, the algorithm distributes the faults to it neighbors. Note that we only distribute the defects to the neighbors that are not visited yet and have less than four predicted defects. Due to the max-flow min-cut theorem, a router can correct at most five defect clusters. However, once it is given a faulty cluster function (line 7), it cannot correct in the reverse direction. Therefore, this router can correct at most four defective TSV clusters. Since it is given one, this router should have at most three defective clusters; otherwise, it cannot be corrected. From line 17 to 21 of Algorithm 1, we assess the remaining faults that are not mapped yet. If there are unmapped defects, we return them as the output of the algorithm.

In Algorithm 1, we also provide an option for the early break, which stops mapping the cluster when NF is 0. This helps to fit precisely the number of defects to the redundant clusters. However, since the defective position may vary among the router clusters, it may cause a smaller bottleneck path that the mapping algorithm fails to find a redundant cluster for correction. Therefore, we support a non-early break that inserts redundant TSVs to all neighbors of the current router. As a result, the number of redundancies can be increased; however, it helps to improve the reliability of the system.

Since Algorithm 1 visits each router once and each visit performs a search for a maximum NF_h, an unvisited router ($O(n)$) and a constant calculation for each router ($O(1)$), the time complexity is $O(n^2)$ where n is the number of the routers within the layer. Here, one of the optimized approaches is to perform a quick sort first and the calculation using the sorted list. By doing so, we can reduce the time complexity to $O(n\ log(n))$ (quicksort is $O(n\ log(n))$ and the placement algorithm is $O(n)$).

D. Algorithm for cluster finding

With the four redundancy options in Figure 4, we need to design a cluster finding mechanism. We first explore online mapping. Then, we explore the ability of an offline algorithm.

1) Online mapping: In [21], we use the center priority weight initialization (CPWI) that gives higher weights to the central routers and lower weights to the border ones. The weights are later optimized (adjusted) to maximize the utilization of TSV clusters. This method could be adopted for the Figure 4 (a) case.

Secondly, we realize that *CPWI* cannot work well with routers having redundant clusters. Despite of having a higher weight, a router still has spare clusters that could be borrowed from its neighbors which is prohibited in the previous work [21] (the weight-based borrowing mechanism). Therefore, we propose a new initiation of weight assignment named spare availability weight initialization (SAWI):

$$\text{Weight} = 4 - \text{unused-clusters} \tag{4}$$

Thirdly, once we have the weight from the Algorithm 1, we can use it as the input for the cluster finding algorithm. Depending on the assigned weights, the finding algorithm will work differently. In the evaluation section, we will further investigate the efficiency of these assignments.

Algorithm 2 shows the weight-based cluster finding algorithm. From the detection mechanism, this one receives the faulty information as input. Here, we can adopt an online detection method [35] or use an error detection code for determining the healthy status of the TSV group. It first tries to repair defects using the current redundancy. If it can find enough clusters, it completes the borrowing. If there are not enough redundancies, it tries to find nearby clusters. Therefore, it creates a borrowing chain. The nearby cluster could be replaced by a nearby redundancy.

However, these online weight-based methods might not be optimal for finding a cluster. Therefore, this kind of approach makes fewer routers operate in a normal state. The main reason for this approach is the locally optimized one. Unless we can predict the possible fault rates, this method is not the best option. To solve this problem, we observe that a software mapping solution could be useful. This mapping can be solved offline with a host CPU.

Algorithm 2: Online TSV Cluster Finding.

```
Input: Re ;                                   // number of redundancy
Input: NF ;                                   // number of fault
Input: Borrow-success ;                       // Ack. for borrowing
Output: Borrow;                               // Borrow cluster
if Re[r][c] >= NF[r][c] then
    repair NF[r][c];
else
    repair Re[r][c];
    find NF[r][c] − Re[r][c] candidates from routers ([r-1,c], [r+1,c],
      [r,c+1], [r,c-1]) based on weight;
    request to borrow the candidates;
    if fail to borrow then
        return borrowing clusters;
        // jump to virtual/serial mode
        if there are four accessible clusters then
            // virtual mode
            request to access everytime the router need;
        else if there is one accessible cluster then
            use 4:1 serial connection;
        else
            use fault-tolerant routing;
    else
        create borrowing chain;
```

Algorithm 3: Ford-Fulkerson TSV Cluster Finding.

```
// Build flow graph
add source s and sink t;
for router r_i in the layer do
    add node for the router r_i;
    add edge from the source s to the router r_i ;
    add capacity r_i → s = number of defective cluster in router r_i;
    add edge from the router r_i to the sink t;
    add capacity r_i → s = number of avaiable redundant cluster attached to
      router r_i;
for router r_i in the layer do
    for router r_j in the layer do
        if routers r_i could borrow a cluster from r_j then
            add edge from the router r_i to the router r_j;
            add capacity r_i → r_j = 1;
while no augmenting path do
    Breadth first search to find minimum path
      Augmenting the found minimum path with capacity Save the flow
if max-flow == # of defective TSV then
    return;
else
    Perform serial/borrowing assignment;
    return;
```

2) Offline mapping: A searching algorithm can be adopted to perform offline with a CPU inside the system [16]. It needs to collect the faulty information and find the optimal solution for cluster chains. In this offline mapping, we assume that a dedicated CPU can access this fault information and use the proposed approach to solve it. Since the system needs to reconfigure after mapping, which might need to defer its operation, we consider this approach as offline even it can be run by a dedicated CPU within the 3D-NoC, and the system can still perform as usual during the execution of the mapping.

Here, we convert the mapping problem to a multi-source multi-sink max flow problem and consolidate source/sink to transform it to a typical single-source single-sink max flow and solve it using Ford-Fulkerson method [36]. Specifically, we use the Edmonds-Karp algorithm with the shortest augmenting path since we want to keep the shortest borrowing chain. Also, since the edge's capacity is always '1' for non-virtual ones, it can be an optimal path as the shortest one. One more constraint is to aim for full-fill most edges instead of partly-fill, as in the Ford-Fulkerson method.

Figure 6 shows how we apply Ford-Fulkerson algorithm [36]. We first add a virtual source s and sink t into the flow graph. Once there is a defective TSV cluster, the edge between source s and the router is added with the capacity of the number of defected clusters. For each router, there is a virtual edge to the sink with capacity '1' if there is a redundancy. Between each router that nearby, there is also an edge with capacity '1'.

The complexity of Edmonds-Karp is $O(VE^2)$ where $V = 2+M\times N$ for a layer of $M\times N$ routers and $E = 4M\times N+M+N$. Therefore, the complexity of Edmonds-Karp for this application is $O(M^2N^2) = O(n^2)$ where n is the number of routers in a layer.

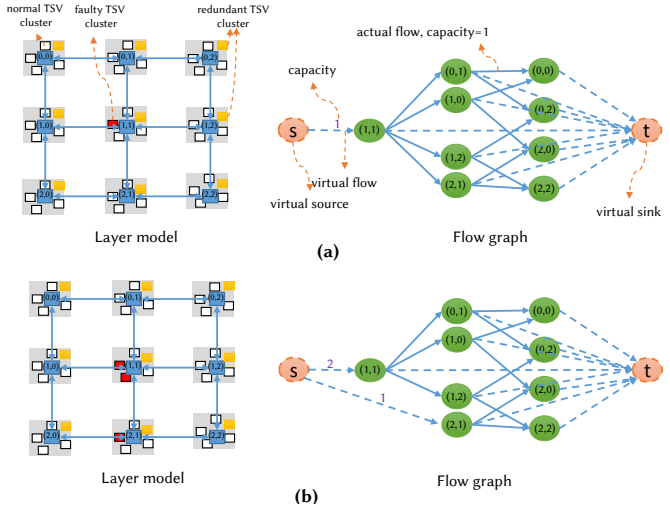

Fig. 6: Example of off-line mapping using the Max-flow Ford-Fulkerson algorithm. The layer size is 3×3 and each router (yellow circle) has one redundant TSV group (green circle). (a) Router (1,1) has one defective TSV cluster. (b) The entire layer has three defective clusters.

V. Evaluation Results

A. Evaluation Methodology

The proposed system was designed in Verilog-HDL, synthesized, and prototyped with commercial and academic CAD tools (Synopsys Design Compiler and Primetime, Cadence Innovus, HotSpot 6.0, gem5 with PARSEC benchmarks). For physical design, we use NANGATE 45nm library [37] and NCSU FreePDK3D45 TSV [38]. The TSV is designed and exported as a macro cell with the size and the pitch of $4.06\mu m\times 4.06\mu m$ and $10\mu m$, respectively. In the place and route design phase, we place the TSV macro cell as the output/input terminal of up and down directions of the 3D-NoC router. The HotCluster reliability estimation platform is built on Java in cooperation with the temperature output of HotSpot.

The proposed technique is implemented into a 3D Mesh NoC system with a 44-bit flit (2× SECDED(22,16)) and four input-buffers depths. The flow-control is Stall-Go, and the forwarding mechanism is Wormhole.

First, we evaluate the proposed architecture and algorithm

using a temperature and fault acceleration model. Here, PARSEC and synthetic benchmarks' trace values are used in our platform to estimate the power consumption. Later, we predict the temperature and reliability to be put into our HotCluster platform. Second, we evaluate the mapping algorithms by inserting faults (defects) into TSV-clusters and assessing the proposed 3D-NoC system's reliability. Here, we randomly insert defected clusters from 0% to 50% to observe the efficiency of the proposed architecture and algorithms. Although faults at high rates such as 50% are not reasonable, we still considered them to understand our method's performance in very extreme situations. Third, we use both synthetic and realistic traffic patterns as benchmarks to study the performance of the proposed system in comparison to the baseline model [39]. Last, we evaluate a single 3D router's hardware complexity and compare our system with other proposed approaches [16], [24].

B. HotCluster method evaluation

1) Simulation flow: In order to evaluate the hotspot defect tolerance of the method, we use both synthetic and realistic benchmarks. We selected Transpose, Uniform, Matrix-multiplication, and Hotspot 10% [40] 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) [40]. For each benchmark, after getting the layout, we perform power extraction using post-layout simulation switching activities. The temperature is predicted using Hotspot 6.0. We first perform the power extraction using Synopsys PrimeTime to obtain the energy per bit value for the dynamic power of the proposed NoC. The static power of NoC is considered independent of the switching activities. Then, from the benchmarks, we can obtain the switching activities (number of bit/time) and estimate the power consumption.

2) PARSEC and synthetic benchmarks: To understand the efficiency of the proposal under realistic benchmarks, we first built a 3-D Mesh topology for garnet2.0 under gem5 system simulation [41]. We then perform PARSEC benchmarks [42] and extract the switching activities of the network. We emulated the activities of the PARSEC benchmark traffics (like our realistic benchmarks) under the post-layout simulation of our router. Then, the power consumption is estimated with PrimeTime. The temperature is predicted using Hotspot 6.0 [43]. We do not consider the processor power/temperature impact with all benchmark-based evaluations since it varies between different processors (with different costs of area, frequencies, and voltages), low power, and cooling techniques. Here, we want to emphasize the impact of switching activities of the router and TSVs (inter-layer link).

We select the T_{ref} with no redundancy required in the coolest layer to normalize the reliability. Consequently, we can observe the optimization in placing the redundancies of *HotCluster*. Here, we compare with the Ford-Fulkerson method for *hyb. red.*, *int. red.*, and *ext. red.* where *irr. red.* is generated by the *HotCluster* platform with the weight-based finding algorithm. Figure 7 shows the results of our

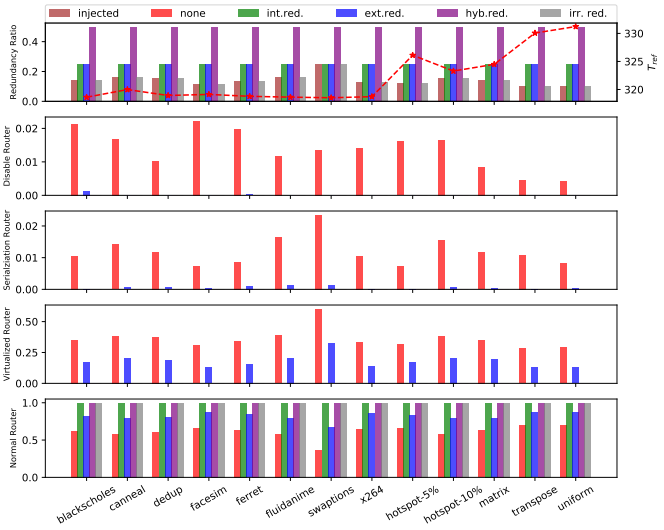

Fig. 7: Evaluation results of our platform for PARSEC and synthetic benchmarks with higher reference temperature (medium defect-rate). *int.red.*, *ext.red.*, and *hyb.red.* use Algorithm 3; *irr.red.* use *HotCluster* platform with reliability prediction and weight-based mapping (Algorithm 1 and 2).

HotCluster for PARSEC and synthetic benchmarks. We can easily observe that all regular redundancy modes (internal, external, and hybrid) have fixed ratios regarding redundancy ratio. On the other hand, *HotCluster* places the equal number of redundancy to the number of injected faults thanks to its prediction on reliability. As can be easily observed, *HotCluster* saves the number of redundancies in all cases. For instance, *hyb. red.* has a redundancy ratio of 0.5, both *int. red.* and *ext. red.* is 0.25 while *HotCluster* is below 0.17 except in *swaptions* (0.25).

Despite reducing the number of redundancies, *HotCluster* still maintain the same reliability as *hyb. red.* and dominates *ext. red.*. The *hyb. red.*, *int. red.* and *HotCluster* maintain 100% router working normally while *ext. red.* has more than 10% routers not in the normal mode.

To illustrate the efficiency of *HotCluster*, we also evaluate it with lower reference temperatures (reduced by 5 Kelvin), as shown in Figure 8. Here, the fault injection rate becomes higher and requires more optimized recovery. As depicted in Figure 8, *HotCluster* copes with the number of faults injected into the system. We can notice that the fault-rate is between 0.25 and 0.5 which makes both *int.red.* and *ext.red.* unable to correct, and only *hyb.red.* can compete with *HotCluster*. However, in terms of reliability, *HotCluster* starts to dominate the recovery efficiency. By planing beforehand the weight of hot and cool routers, it can correct more routers in the normal mode than its counterparts. In all cases, *hyb.red.* has lower rates in normal routers than *HotCluster*. This is due to the fact that the weight assignment method is only based on the number of redundancies of each router and fails to recognize the hotter routers in the layer. However, *Ford-Fulkerson* is only optimal to find replacements for faulty clusters and not optimal for having a larger ratio of normal routers.

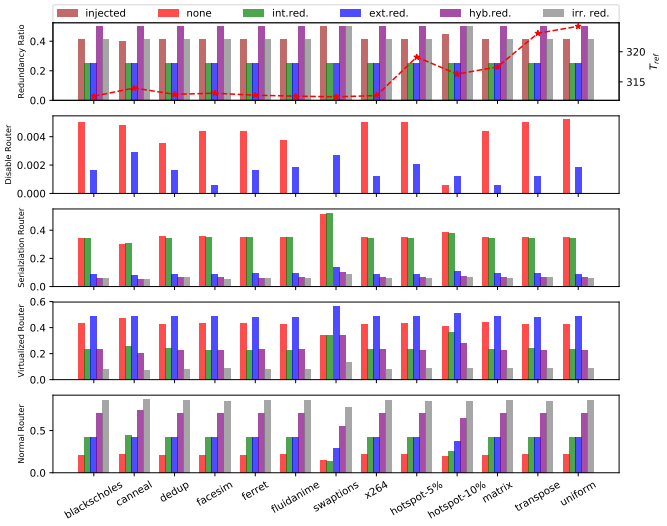

Fig. 8: Evaluation results of our tool flow for PARSEC and synthetic benchmarks with lower reference temperature (high defect-rate). *int.red.*, *ext.red.*, and *hyb.red.* use Algorithm 3; *irr.red.* use *HotCluster* platform with reliability prediction and weight-based mapping (Algorithm 1 and 2).

HotCluster brings more flexibility on mapping TSV cluster redundancies and it has the lowest number of redundancies while still maintaining a comparable reliability. However, *HotCluster* is based on 3D-NoC and relies on regular TSV placements which allow easy access to neighbor TSV clusters. For irregular TSV placements, we can still split TSVs into clusters and allow the repairing and borrowing mechanisms as in 3D-NoCs. In this case, our *HotCluster* can still be applied and provide a reliable solution for TSV placement and recovery. However, it has three significant limitations in the irregular case: (1) if some TSV regions are distant from others, we cannot allow neighboring borrowing due to the induced timing violations; (2) the borrowing chain in 3D-NoC might not be established due to the lack of diversity in the borrowing options; and (3) it is not efficient if we cannot predict the temperature and reliability.

C. Comparison of cluster finding algorithms

As we discussed earlier, the system can adopt either online or offline mapping. If the predicted temperature defines the weights as the previous section, the *HotCluster* platform can easily dominate the online mapping. However, if the temperatures and the fault rates are unpredictable, designers should consider one of the three mappings (int. red., ext. red. and hyb. red.). Here, we evaluate these three mapping methods under our proposed recovery algorithms.

Instead of using the predicted hotspot, we randomized the high fault rate areas. By randomizing the faults (or hotspots) within the layers, we can cover the case of unpredictable fault patterns. Moreover, it also can be implied as the case other module have thermal impact on TSV regions. For instance, a TSV group placed nearby a high-temperature CPU can have a high defect rate without even being frequently used as it is attached to a highly utilized router, as explained in Section V-B.

To prove our proposal's scalability, we evaluated several layer sizes: 2×2, 4×4, 8×8. TSVs are grouped into clusters, as explained in Section III-B, and the defect-rates vary from 5% to 50%. We perform the Monte-Carlo method for the proposed algorithms with 10,000 different samples and calculate the average results. We measure the ratio of four types of routers in the layer: *Normal* (healthy or corrected), *Virtual* (router with virtual TSV), *Serial* (router using serialization) and *Disabled* (routers with disabled vertical connections). Here, we evaluate three different algorithms: (1) CPWI from [21], and (2) SAWI, (3) Ford-Fulkerson implemented in HotCluster.

As depicted in Figure 9, the CPWI algorithm without redundancy system mostly operates without disabling any vertical connections with fault-rates under or equal to 50%. Thanks to the *Virtual TSV* and *Serialization* techniques, the routers having less than four clusters are still able to work. We could notice that under 50% defect-rate, there is a small percentage of routers having disabled connections. In this case, a fault-tolerant routing algorithm [2] could easily fix the communication availability. With redundancies, CPWI can reduce the number of routers operating in bottleneck modes (virtual/serial). This could be easily explained by the fact the redundancies can correct defective TSV clusters. For small size layers (2×2 or 4×4) and low defect-rates ($< 10\%$), CPWI with redundancies is very efficient with nearly 100% routers operating in full mode. However, we can easily notice that with large layer sizes and higher defect-rates, CPWI with redundancies become inefficient. Here, the SAWI and FF (Ford-Fulkerson) with different redundancy strategies can be helpful.

In Figure 9, we can see that SAWI significantly improves the availability. With both internal and external redundancy, SAWI outperforms CPWI. With less than 20% defect-rates, it maintains most of the router in normal mode. However, as previously discussed, SAWI is a local optimization method; therefore, with higher defect-rates, the FF method starts to outperform it because of its global optimizing flow. With a 45% defect rate in a 8×8 layer and *int. red.*, SAWI has 0.3% of the router having disabled TSV connections while the FF method can reduce it to 0.11%. Compared to SAWI, the Ford-Fulkerson (FF) method easily dominates at low error rate where very low percentages of virtual or serial are needed to maintain operation. With the same *int.-red.* at a 20% defect rate, the FF method has in average 1.5%, 0.7%, and 0% in *virtual*, *serial*, and *disable* mode, respectively while the results for SAWI are in average 5.39%, 1.11%, and 0.01% in *virtual*, *serial*, and *disable* mode, respectively. As the results show, we can observe that more routers operate in normal mode with FF than SAWI as in Figure 9. However, at higher defect-rates, FF method becomes less efficient than SAWI for having a smaller number of normal routers. For example, at a 50% defect rate, 4×4, and *hyb.-red.*, the FF method and SAWI maintain 18.87% and 28.40% of the routers in the normal mode, respectively. Despite being less efficient for mapping router in normal mode, we can observe that FF has fewer routers being disabled since it claims 0.31% while SAWI claims 0.73% of the routers having disabled connections. The main reason is that FF method only tries to make the

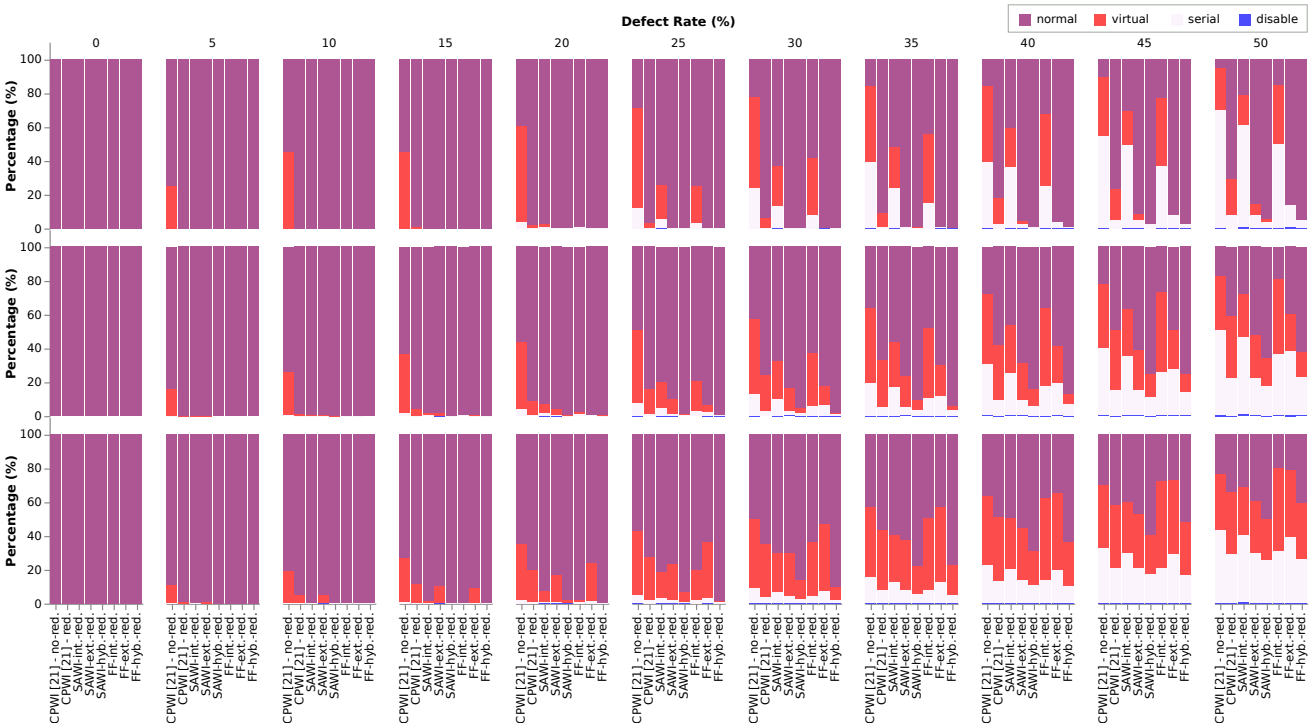

Fig. 9: Defect-rate evaluation with random defect distributions (injected to redundant TSVs): (a) Layer size: 2×2 (4 routers, 16 TSV clusters); (b) Layer size: 4×4 (16 routers, 64 TSV clusters); (c) Layer size: 8×8 (64 routers, 256 TSV clusters) .

maximum flow, which corrects as many clusters as possible. This makes FF method have fewer disabled routers; however, it has more routers in virtual or serial mode. Here, we can observe that both weight assignment adaption (SAWI) and the offline mapping (FF) easily dominate the center priority approach in [21]. Depending on the applications' needs and specifications, designers can choose a proper method for remapping TSV clusters.

In conclusion, we have demonstrated the efficiency of the three algorithms: CPWI, SAWI and FF. While SAWI and FF can dominate CPWI, they have more area cost overhead due to the additional redundant TSVs. On the other hand, we can easily observe that with less than 25% of defect-rate, the proposed system with either SAWI or FF can provide a most efficient clustering recovery mechanism.

D. 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 [44] 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 proposed system's performance and keep fair comparisons to the baseline, we adopted both synthetic and realistic traffic patterns as benchmarks. The packets are injected until the saturation point of the network is reached. To keep a fair comparison, only TSV defects are injected. This means that the other fault-tolerance mechanisms [40] are disabled not to affect the performance.

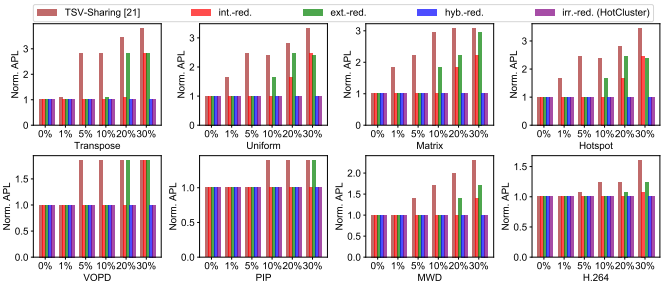

Fig. 10: Evaluation results and comparison of *HotCluster* with *hyb.red.* and TSV-Sharing [21] using CWPI in terms of Average Packet Latency. Results are normalized to the baseline 3D-NoC design [39].

In this experiment, we evaluate the proposed architecture's performance in terms of Average Packet Latency (APL) and throughput over various benchmark programs and defect-rates. We compare four different patterns where the *irr.-red.* is the output of the *HotCluster*. The simulation results are shown in Figure 10 and Figure 11. From these graphs, we notice that with a 0% of defect-rate, the system has similar performance compared to the baseline system.

As shown in Figs. 10 and 11, when we increase the defect-rates in the TSV-Sharing system [21], it has demonstrated additional impacts on APL and throughput. Meanwhile, our proposed system under four different configurations shows no degradation at less than 20% defect rate. Once we increase it to 20%, we start to obverse some degradation in performance. The external redundancy method starts to degrade at higher

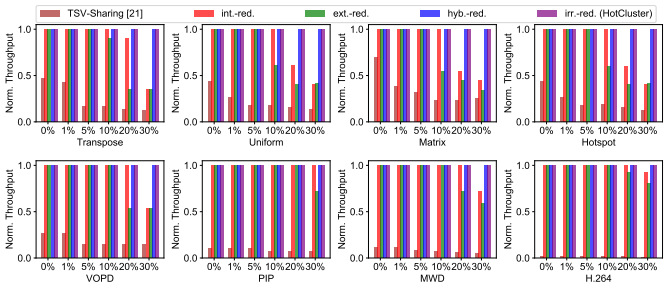

Fig. 11: Evaluation results and comparison of *HotCluster* with *hyb.red.* and TSV-Sharing [21] using CWPI in terms of throughput. Results are normalized to the baseline 3D-NoC design [39].

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

Benchmark	#Defect Link	[45]		[46]		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	1
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 [45], we used their value for seven benchmarks, with three layers configuration.
[4] In order to compare with [46], this work is inserted defect TSV clusters to create a defected all layers link.

error rates. This can be easily observed in Section V-B where the disabled connections start to appear in these benchmarks. The *int.-red.* only has some degradation in some benchmarks at very high fault rates. The main reason is that *int.-red.* inserts one redundant cluster per four TSV clusters; therefore, it can easily correct at a 25% defect rate. However, as we evaluated in Section V-C, there are some virtualized and serialized connections even with only 25% defect rates. On the other hand, both *hyb-.red.* and *HotCluster* easily dominate in this benchmark with basically no degradation. The main reason is that both methods have an abundant amount of TSV redundancies; therefore, they have no difficulties in this evaluation.

Table I shows the comparison results of our work with three other inter-layer fault-tolerant communication methods. The selected two works were presented in [45] and [46] which target fault-tolerant customized 3D-NoCs and hybrid-3D-NoC, respectively, while TSV-Sharing [21] is our previous work. As shown in Table I, *HotCluster* provides the best performance when compared to the other works [45], [46]. The customized 3D-NoCs in [45], [46] suffer from significant performance degradation due to the lack of routing paths and frequent occurrence bottlenecks, which increase the APL by nearly 2.5 times and reduce the overall throughput almost three times. In summary, our proposed technique provides similar performance as the baseline one while providing high resiliency against TSV defects.

E. Router Hardware Complexity

Table II illustrates the hardware complexity breakdown of the proposed router in terms of area, power, and speed. In comparison to TSV Sharing [21], the router of CPWI without redundancies, the area and power consumption have increased by 1.93% and 14.39%, respectively, while the maximum speed is maintained as the same. This could be well explained by the fact that the additional circuit is very low cost and has no impact on the critical path; however, the switching activities to select different TSVs for a signal cause more power consumption. 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 designed HotCluster not only supports link faults, soft error, and TSV faults, but also supports online recovery. As we showed in Section V-D, the HotCluster system totally dominates the baseline and TSV sharing systems at the presence of TSV faults.

TABLE II: Hardware complexity breakdown of a single router.

Model		**Area** (μm^2)	**Power** (mW)	**Speed** (Mhz)
Baseline router [39]		18,873	6.0658	925.28
TSV Sharing [21]	routing units	29,780	12.3144	613.50
	Serialization	3,318	-	-
	External sharing	5,740	-	-
	Router's logic	38,838	14.6128	537.63
(CPWI-no-red.)	TSV's area	2,901.1136	-	-
	Total	41,739	14.6128	537.63
HotCluster (*int.red.*)[1]	routing units	29,780	12.3144	613.50
	Serialization	3,318	-	-
	External sharing	5,740	-	-
	Internal correction	325.5840	-	-
	Router's logic	38,920	16.7161	537.63
	TSV's area [1]	3,626.3920	-	-
	Total	42,546	16.7161	537.63

TSV area without Keep-out-Zone: $4.06 \times 4.06 = 16.4836 \mu m^2$.
[1] The TSV area depends on the insertion of redundancies in *HotCluster* flow.

F. Comparison

To understand the efficiency of the proposed approach, we compare it with existing solutions for cluster TSV defects, as shown in Table III. Here, we analyze our proposal with a network size of $4 \times 4 \times 4$. Because the router and its TSV clusters structure are identical, similar results can be obtained with other network sizes.

We select the best results of these two works [16], [24] for the comparison. *TSV Grouping* [24] optimized the configuration of redundancy to deal with TSV-cluster defects. *TSV Network* [16] established TSVs into a network that allows routing from defected TSVs to redundant ones. We also compare with *TSV Sharing* [21] which is our previous work. From this table, we can see that the average area of *HotCluster* [21] is $51.93 \mu m^2$ per original TSV which is slightly larger than *TSV Sharing*. The *TSV Network* [16] 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 $51.86 \mu m^2$. Also our *HotCluster* has a higher area cost; however, it is still reasonable as the TSV with Keep-out-Zone of $10 \mu m$ cost $100 \mu m^2$ of physical area.

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

Model	TSV Network [16]				TSV Grouping [24]			TSV Sharing [21]		**HotCluster**			
Technology	65 nm				N/A			45 nm		45 nm			
#TSV	1000				6000			8448		8448			
Configuration	4:2	8:2	$64:16$	$256:32$	4:4	8:4	20:5	176:0		4:1[5]	4:2[6]	4:1[5]	4:2[6]
#Spare TSV	512	256	256	128	6000	3000	1500	0		2112	4224	2112	4224
Arbiter Area (μm^2)	372 [2]	744 [2]	1,116 [2]	1,116 [2]	11,160 [1]	11,160[1]	12,555[1]	434,784[3]		438,720[3]			
per TSV (μm^2)	51.57	26.24	26.72	28.03	13.92	51.86	27.09	51.47		51.93			
Reliability	100%	99%	100%	100%	100%			100%	98.11%	100%	100%	99.27%	99.76%
Fault Config.	$\delta_{TSV} = 0.01\%$, $\alpha = 2^4$				$\delta_{TSV} = 1\%$,$\alpha = 2^4$			$\delta_c = 1\%$[4]	$\delta_c = 50\%$[4]	$\delta_c = 1\%$[4]		$\delta_c = 50\%$[4]	

[1] The authors use 2:1 multiplexers [24]. For comparison, we use the area cost of multiplexer from Nangate $45nm$ [37] (MUX2_X1: $0.186\mu m^2$) .
[2] The authors use 1-to-3 multiplexers [16] which consists of two MUX2_X1 multiplexers ($2 \times 0.186\mu m^2$ [37]).
[3] For fair comparisons, the arbiter only consists of the TSV sharing and serialization modules.
[4] δ_{TSV}: TSV defect-rate. α: parameter of Poisson distribution [16], [24]. δ_c: TSV cluster defect-rate.
[5] : *int.red.* configuration.
[6] : *ext.red.* configuration.

Although the *TSV Grouping* [24] and *TSV Network* [16] obtained lower area overheads, our design not only consists of the rerouting circuit, but also includes an online adaptive algorithm designed in hardware. Both *TSV Grouping* and *TSV Network* have to require additional dedicated circuits or CPU time to recover from the cluster defects. Meanwhile, our previous work (*TSV Sharing* [21]) can compete with similar features; however, our improvements in redundancies can easily dominate the reliability and performance, as previously depicted in Figures 9, 10 and 11.

In terms of reliability, *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, when these techniques can be inefficient when a lot of TSVs become defective, our technique can still help the system works. For example, with 4:1 and 4:2 redundancies, *HotCluster* under 1% defective clusters maintains the active routers as 100%. Moreover, more than 99.27% of routers still maintain their connections even under 50% clustering defect-rate. With *hyb.red.*, there are nearly 10.8% routers working in *Serialization* mode, which means about 88% of routers can work in normal or virtual mode. Note that with 50% defect-rate, only *TSV Grouping* with 4:4 can possibly function as we assume the recovery works perfectly. It is also important to mention that less than 1% of the router were disabled in our Monte-Carlo simulation. Therefore, we use the Look-Ahead Routing Algorithm to recover the disabled routing paths.

VI. Conclusion and Future Work

This paper presented a thermal aware recovery methodology for TSVs in 3D-ICs systems to deal with the TSV-cluster defects. We first give several algorithms to help to correct the faulty TSV clusters within a 3D-NoCs. Then, thermal modeling using Arrhenius's Law is presented to help predict the critical area of the 3D-ICs in terms of reliability. Based on the thermal model, we give the cluster placement approach and its cluster finding solution based on weight to adapt with the thermal related defects. The results have proven the system's ability to provide high reliability that can reach up to 100% of working routers even under 50% defective clusters. Moreover, the proposed approach can correctly work without any degradation under a 20% of defect-rate. The hardware complexity has shown a small overhead in terms of area cost even with redundant TSVs (30.86%). Our platform even reduces the number of TSV redundancy to reduce the area cost, especially with possible hotspots.

As future work, the random TSV-defect is also an additional challenge for our 3D-NoC system. The impact of high temperature on timing of TSVs is another critical issue for our future works. On the other hand, the irregular TSV placement needs to be considered to support more flexible architectures. Furthermore, degradation factors on 3D-NoCs such as thermal dissipation, stress, operating voltages should be investigated.

Acknowledgments

This research is funded by Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.01-2018.312.

References

[1] 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.
[2] 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.
[3] 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.
[4] 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.
[5] T. Zhang *et al.*, "Temperature-aware routing in 3D ICs," in *Asia and South Pacific Conf. on Design Automation*, Jan 2006, pp. 309–314.
[6] 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.
[7] 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.
[8] J. R. Black, "Mass transport of aluminum by momentum exchange with conducting electrons," in *6th Annual Reliability Physics Symposium (IEEE)*. IEEE, 1967, pp. 148–159.
[9] D. Cuesta *et al.*, "Thermal-aware floorplanner for 3D IC, including TSVs, liquid microchannels and thermal domains optimization," *Applied Soft Computing*, vol. 34, pp. 164–177, 2015.
[10] M. White, *Microelectronics reliability: physics-of-failure based modeling and lifetime evaluation*. JPL Publ., 2008.
[11] J. U. Knickerbocker *et al.*, "Three-dimensional silicon integration," *IBM J. Research and Development*, vol. 52, no. 6, pp. 553–569, 2008.
[12] Y. Cheng *et al.*, "Alleviating through-silicon-via electromigration for 3-d integrated circuits taking advantage of self-healing effect," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 11, pp. 3310–3322, 2016.

[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] M. Yi *et al.*, "A pulse shrinking-based test solution for prebond through silicon via in 3-D ICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 4, pp. 755–766, 2018.

[16] L. Jiang, Q. Xu, and B. Eklow, "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.

[17] T. Ni *et al.*, "LCHR-TSV: novel low cost and highly repairable honeycomb-based TSV redundancy architecture for clustered faults," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2938–2951, 2020.

[18] 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.

[19] T. Ni *et al.*, "Architecture of cobweb-based redundant TSV for clustered faults," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 28, no. 7, pp. 1736–1739, 2020.

[20] 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.

[21] K. N. Dang, A. B. Ahmed, Y. Okuyama, and A. B. Abdallah, "Scalable design methodology and online algorithm for TSV-cluster defects recovery in highly reliable 3D-NoC systems," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 3, pp. 577–590, 2020.

[22] C.-H. Chao *et al.*, "Traffic-and thermal-aware run-time thermal management scheme for 3D NoC systems," in *2010 ACM/IEEE International Symposium on Networks-on-Chip*, 2010, pp. 223–230.

[23] 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.

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

[25] W. Lo, K. Chi, and T. Hwang, "Architecture of ring-based redundant tsv for clustered faults," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 848–853.

[26] T. Ni *et al.*, "A cost-effective TSV repair architecture for clustered faults in 3D IC," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–5, in-press.

[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] P. Coudrain *et al.*, "Experimental insights into thermal dissipation in TSV-based 3D integrated circuits," *IEEE Design & Test of Computers*, no. 1, pp. 1–1, 2016.

[29] P.-Y. Hsu, H.-T. Chen, and T. Hwang, "Stacking signal TSV for thermal dissipation in global routing for 3-D IC," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 7, pp. 1031–1042, 2014.

[30] J. Cong and Y. Zhang, "Thermal via planning for 3-D ICs," in *IEEE/ACM Int. Conf. on Computer-Aided Design*, 2005, pp. 745–752.

[31] Y. Cheng *et al.*, "Thermal-constrained task allocation for interconnect energy reduction in 3-D homogeneous MPSoCs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 2, pp. 239–249, 2012.

[32] J. Cong *et al.*, "Thermal-aware cell and through-silicon-via co-placement for 3D ICs," in *48th Design Automation Conference*, 2011, pp. 670–675.

[33] S. Wang, M. B. Tahoori, and K. Chakrabarty, "Thermal-aware TSV repair for electromigration in 3D ICs," in *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*. EDA Consortium, 2016, pp. 1291–1296.

[34] S. K. Samal *et al.*, "Monolithic 3D IC vs. TSV-based 3D IC in 14nm FinFET technology," in *2016 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S)*. IEEE, 2016, pp. 1–2.

[35] K. Dang *et al.*, "TSV-IaS: Analytic analysis and low-cost non-preemptive on-line detection and correction method for TSV defects," in *Proc. IEEE Computer Society Annual Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 501–506.

[36] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," in *Classic papers in combinatorics*. Springer, 2009, pp. 243–248.

[37] NanGate Inc., "Nangate Open Cell Library 45 nm," http://www.nangate.com/, (accessed June 16, 2016).

[38] NCSU Electronic Design Automation, "FreePDK3D45 3D-IC process design kit," http://www.eda.ncsu.edu/wiki/FreePDK3D45:Contents, (accessed June 16, 2016).

[39] 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.

[40] 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.

[41] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH computer architecture news*, vol. 39, no. 2, pp. 1–7, 2011.

[42] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.

[43] R. Zhang, M. R. Stan, and K. Skadron, "Hotspot 6.0: Validation, acceleration and extension," *University of Virginia, Tech. Rep*, 2015.

[44] 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.

[45] 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.

[46] 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.

Khanh N. Dang received his Ph.D. degree from The University of Aizu, Japan in 2017. Since 2017, he has been an assistant professor at VNU Key Laboratory for Smart Integrated Systems, Vietnam National University Hanoi (VNU), Hanoi Vietnam. Dr. He was visiting researcher at University of Aizu in 2019 and 2020-2021. His research interests include System-on-Chips/Network-on-Chips, 3D-ICs, neuromorphic computing, machine learning and fault-tolerant systems.

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 later joined Keio University, Japan, as a postdoctoral researcher. He is currently a Research Scientist at the National Institute of Advanced Industrial Science and Technology (AIST), Japan. His research interests include on-chip interconnection networks, reliable and fault-tolerant systems, and ultra-low-power embedded systems.

Abderazek Ben Abdallah is a full Professor of computer science and engineering and the Head of the Division of Computer Engineering, The University of Aizu, Japan. He received his Ph.D. degree in computer engineering from the University of Electro-Communications, Tokyo, in 2002. His research falls primarily in computer systems and architectures with an emphasis on adaptive systems, interconnection networks, powerand reliability-aware architectures, and neuromorphic computing systems.

Xuan-Tu Tran received a Ph.D. degree in 2008 from Grenoble INP (at the CEA-LETI), France, in Micro Nano Electronics. He is currently an associate professor at VNU-UET, Vietnam National University, Hanoi (VNU). He was an invited professor at the University Paris-Sud 11, France (2009, 2010, and 2015), at Grenoble INP, France (2011), at the University of Electro-Communications, Japan (2019), and adjunct professor at UTS, Australia (2017-2020). He is currently the director of the VNU Key Laboratory for Smart Integrated Systems (SISLAB) and director of VNU Information Technology Institute (VNU-ITI). His research interests include design and test of systems-on-chips, networks-on-chips, design-for-testability, asynchronous/synchronous VLSI design, low power techniques, and hardware architectures for multimedia applications, cryptography.