# TSV-OCT: A scalable online multiple-TSV defects localization for real-time 3D-IC systems

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

***Abstract*—In order to detect and localize TSV failures in both manufacturing and operating phases, most of the existing methods use a dedicated testing mechanism with long response time and prerequisite interruptions for on-line testing. This paper presents an ECC-based method named “Through-Silicon-Via On Communication Test” (*TSV-OCT*) to detect and localize faults without halting the operation of TSV-based 3D-IC systems. We first propose *Statistical Detector*, a method to detect open and short defects in TSVs that work in parallel with data transactions. Second, we propose an *Isolation and Check* algorithm to enhance the localization ability of the method. Moreover, the Monte-Carlo simulations show that the proposed *Statistical Detector* increases $\times 2$ the number of detected faults when compared to conventional ECC-based techniques. With the help of *Isolation and Check*, TSV-OCT localizes the number of defects up to $\times 4$ and $\times 5$ higher. In addition, the response time is kept below 65,000 cycles which could be easily integrated into real-time applications. On the other hand, an implementation of *TSV-OCT* on a 3D Network-on-Chip router shows no performance degradation for testing while having a reasonable area overhead.**

***Index Terms*—Fault-Tolerance, Error Correction Code, Through Silicon Via, Product Code, Fault Localization**

## I. Introduction

Through-Silicon-Vias (TSVs) serve as vertical wires between two adjacent layers in 3D-ICs. Thanks to their extremely short lengths, their latencies are low which could offer high speeds of communication [1].Moreover, as a 3D-IC technology, TSV-based ICs can have smaller footprints despite the TSV’s overheads [2], and lower power consumption thanks to the shorter wires [3]. On the other hand, Network-on-Chips (NoCs) have been widely considered as one of the most promising replacements for traditional communication paradigms (e.g., bus, point-to-point) with better scalability and parallelism [4]. By combining these two advanced technologies, 3D-Network-on-Chips (3D-NoCs) [4], [5] could open a new horizon for high performance and low power designs which have become crucial to satisfy the strict requirements of future complex applications.

Despite the aforementioned advantages, reliability has been a major concern of TSVs due to their low yield rates [6], vulnerability to thermal [7], [8] and stress, and the cross-talk issues [9], [10]. Defects on TSVs can occur in both random and cluster distributions [11] which create concerns about their fault-tolerance efficiency. TSVs are also shown to be susceptible to Electron-Migration (EM) [12] which could be a critical factor to lifetime reliability. Because of the natural parallel structure, TSVs also encounter the crosstalk challenge [13], [14] which might cause timing violations. Furthermore, the differences in thermal expansion coefficients of materials and temperature variations between two layers, which has been reported to reach up to 10°C [7], could lead to stress issues that may crack the TSV during operation. Also, since thermal dissipation in 3D-ICs is more challenging than traditional 2D-ICs, the fault rate could be exponentially accelerated [8].

To enhance the reliability of TSVs, we classify the fault-tolerance process into three major phases: detection, localization (or diagnosis), and recovery. For detection and localization, Built-in-self-test (BIST) [15], [16] and external testing [17] techniques are two common methods to determine whether a TSV has a defect. Error Correction Codes (ECCs) [18] or dedicated circuits [19]–[22] also support detecting and correcting fault. On the other hand, most recent researches have been focusing on recovery where there are several approaches such as *hardware fault-tolerance* (i.e., correction circuits [20], redundancies [11], reliability mapping [10]), *information redundancy* (i.e., coding techniques [13], [18], [23] or re-transmission request [24]) and *algorithm-based fault-tolerance* (i.e., fault-tolerant routing [4], [25], run-time repair [12] or remapping [11], [26]). Although commercial CAD tools and existing solution have become mature for defect localization and detection, having an online and non-blocking solution helps preventing expensive consequences of operating systems under faults.

To maintain highly reliable real-time systems, fault detection and recovery is an important task. So, it must be deadline-driven and still maintain other tasks’ operations [27]. However, most of the existing methods on solving the reliability issues of TSVs focus on manufacturing test and recovery while the online lifetime reliability is not properly addressed. Because of the consequences of silent defects could be expensive, the defect detection tasks require short response time and less performance degradation. To reuse the existing test infrastructure, the system could perform a testing process periodically using *BIST* [12], [15], [16] or external testing [17]. Here, we use the term *Periodic BIST* (P-BIST) [28] for this kind of tests because they are active periodically. ECC can also act as a near-instantaneous fault detection and localization method. Although *P-BIST* and ECC could handle fault detection and recovery, there are five major issues in conventional on-line methods to detect and localize faults:

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).

Akram Ben Ahmed is with Department of Information and Computer Science, Keio University, Yokohama, 223-8522, Japan.

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

1) *Reusing BIST as P-BIST usually takes a considerable amount of time while blocking the system operation.* Previous works [29]–[32] on conventional *BIST* already show that completing testing a system could cost hundred thousands or even millions of cycles. Meanwhile, *P-BIST* need to completely or partly preempt the system operation for execution which raises concerns about real-time requirements. Online testing with data traffic priority [12], [33] could lower the testing time while maintaining the system's operation.
2) *The testing period, which is the interval between two consecutive testings, is also a critical parameter.* By having a small *testing period*, the system has a lower risk of operating under faults; however, the impact on performance could be substantial. To reduce the performance impact, having longer *period* may help, but it may leave the system under faults for a longer time [12], [33].
3) *Scalability is also a major challenge for multi/manycore SoC testing as we increase the number of cores and interconnects.* Since testing the whole system requires an enormous amount of time [29]–[32], up-scaling to thousands of cores could lead to unattainable testing time.
4) *ECCs are usually limited by their ability to detect and correct faults.* As we mentioned, clustering defects [11] could appear in a TSV group, which leads to multiple faults within a group of TSVs; thus, challenging *ECCs* efficiency.
5) *Intermittent defect of TSV could happen*. Since TSV is sensitive to thermal and stress issues, and 3D-ICs usually encounter different scenarios in thermal distribution [7], TSVs under *P-BIST* may not expose their faulty behavior. For instance, high frequency TSVs [34] could have timing violations due to small crack and higher resistance under higher operating temperature; however, cooled down TSVs could not be detected by normal testing. Because the defects could be intermittent, it is difficult to detect and correct them.

To solve those problems, a short response time fault detection and localization method working in-parallel with the system operation in order to detect and correct faults [35] is needed. Here, we use "On-Communication/Computation Test" (OCT) as the terminology for this kind of methods in order to make the distinction from *P-BIST*. Figure 1 depicts our motivation of using "On-Communication Test". In normal P-BISTs, depicted as the periodic test, the response time is $\Delta_{Task}+\Delta_{BIST}$ ($\Delta_{Task} \leq Period$) where $\Delta_{Task}$ and $\Delta_{BIST}$ are the execution time of regular tasks and BIST, respectively. Because the testing time $\Delta_{BIST}$ requires thousands or even millions of cycles [29], [30], [36] the period of test should be a larger value to reduce the performance degradation. Assuming the performance degradation is 10%, *Period* should be $10 \times \Delta_{BIST}$ . Therefore, in order to respond to the new fault in a short interval, using *P-BIST* is not ideal because the system might respond to new faults after an enormous number of cycles.

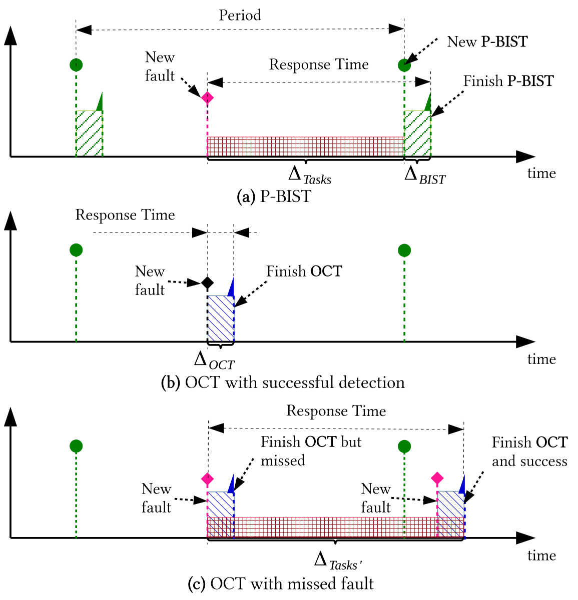

Figure 1: Motivation for On-Communication Test (OCT).

Figure 1(b) shows the case of using *OCT* which is usually non-blocking (or with graceful degradation), the system can respond to the new fault after the test time $\Delta_{OCT}$. According to Figure 1(a), the response time is considerably reduced. Also, if the system still doubts the quality of its *OCT*, it can yield a *BIST* later. As discussed in [28], *OCT* could be performed at lower-level using *dynamic verification* or *redundancies* (double or triple execution for comparison) or higher level using *anomaly detection* (i.e., Operating System, page failures, traps, exceptions). ECCs/EDCs could be classified as an *OCT* method.

Although *OCT* has shorter response time, it has lower coverage than *BIST*. For instance, using *SECDED* [23] (Single Error Correction Double Error Detection) code as *OCT* can guarantee to detect two faults; therefore, it could miss if there are three faults. If a fault is missed by *OCT*, the system could have very long response time as Figure 1(c) represents. Because the first fault is missed by *OCT*, the system keeps running until another fault or a different behavior of the previous fault happens. Figure 1(c) shows the case with the response time $\Delta_{Task'}$ which is larger than *P-BIST*'s ($\Delta_{Task} + \Delta_{BIST}$).

In order to have a better response time to new faults, we use *OCT* to obtain an efficient trade-off between response time and the system's availability. To solve the low coverage of *OCT*, we propose the "Through-Silicon-Via On Communication Testing" (*TSV-OCT*) methodology that includes a set of algorithms and architectures as follows:

- A comprehensive *OCT* set of algorithms and architectures based on two phases to improve the coverage: *Statistical Detection* and *Isolation-and-Check*.
- *Statistical Detection*: the system localizes faults based on the fault behaviors from the *ECC* decoder of multiple transactions. In this paper, we opt for the Parity Product Code (PPC) adopted in our previous work [37] as the baseline. The Monte-Carlo simulation shows that *Statisti-*

*cal Detection* helps to localize 100% of two faults despite the limitation of one fault localization of PPC.

- The *Isolation-and-Check* algorithm further enhances the localization ability of the method. It first virtually isolates the suspicious TSVs by disconnecting them from the encoding/decoding process to find more hidden defects. After no more defects are found, it re-attaches these TSVs back to the encoding/decoding to ensure the faulty status. In our evaluation, we show that *TSV-OCT* can detect 100% of 5 defects cases.

The organization of this paper is as follows: Section II reviews the background and existing literature on coding techniques and TSV fault-tolerances. Section III presents the proposed algorithms and architectures. Section IV provides the evaluation environment and results. Finally, Section V concludes the paper.

## II. Background

This section firstly introduces Parity Product Code - the baseline ECC for our *TSV-OCT*. Later, we present the existing works in testing and localizing TSV defects.

### A. Parity Product Code

This part presents the baseline ECC: Parity Product Code (PPC) which is based on the Product-Code [38]. The TSV grouping method and column/row check can also be found in [39].

*1) Encoding:* For each transmission, a coded flit $F$ is represented as follows:

$$F_k = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & \cdots & b_{0,N-1} & r_0 \\ b_{1,0} & b_{1,1} & b_{1,2} & \cdots & b_{1,N-1} & r_1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ b_{M-1,0} & b_{M-1,1} & b_{M-1,2} & \cdots & b_{M-1,N-1} & r_{M-1} \\ c_0 & c_1 & c_2 & \cdots & c_{N-1} & u \end{bmatrix}$$

where

$$\begin{aligned} r_i &= b_{i,0} \oplus b_{i,1} \oplus \cdots \oplus b_{i,N-1} \\ c_j &= b_{0,j} \oplus b_{1,j} \oplus \cdots \oplus b_{M-1,j} \\ u &= \oplus_{i=0}^{N-1} \oplus_{j=0}^{M-1} (b_{i,j}) \end{aligned} \tag{1}$$

Note that the symbol $\oplus$ stands for XOR function. One TSV handles the transmission of a bit in flit $F$.

*2) Decoding:* By using parity checking, the decoder can find the column and row indexes of the flipped bit. The parity equations are as follows:

$$\begin{aligned} sr_i &= b_{i,0} \oplus b_{i,1} \oplus \cdots \oplus b_{i,N-1} \oplus r_i \\ sc_j &= b_{0,j} \oplus b_{1,j} \oplus \cdots \oplus b_{M-1,j} \oplus c_j \\ sc_N &= r_0 \oplus r_1 \oplus \dots r_{M-1} \oplus u \\ sr_M &= c_0 \oplus c_1 \oplus \dots c_{N-1} \oplus u \end{aligned} \tag{2}$$

where the bits $b_{i,j}$, $c_j$, $r_i$ and $u$ is taken from the corresponding TSVs. The outputs of Eq. 2 are two arrays of column check ($sc$) and row check ($sr$). If there is one or no flipped bit, the decoder can correct it using a $(N+1) \times (M+1)$ mask matrix $m$ where

$$m_{i,j} = \begin{cases} 1 \text{ if } sr_i == 1 \text{ and } sc_j == 1 \\ 0 \end{cases}$$

For each received flit $\hat{F}_k$, the corrected flit $F_k$ is obtained by:

$$F_k = \hat{F}_k \oplus \mathsf{m}$$

The decoder fails to correct when there are two or more faults. To support fault detection, the decoder uses the following equation:

$$\begin{aligned} fr = \sum_{i=0}^{N+1} sr_i; \; fc = \sum_{i=0}^{M+1} sc_i; \\ \mathsf{Fault\_Detected} = (fr \geq 2) \vee (fc \geq 2) \end{aligned} \tag{3}$$

*3) Correctability and Detectability:* In general, PPC can ensure the ability to correct one and detect two flipped bits. Moreover, if there are more than two flipped bits not sharing row or column indexes, PPC also has chances to detect them using Eq. 3. However, there is a weak point in its detection approach that always prevents it from detecting three faults. For instance, if bits with indexes $(i, j)$, $(i, k)$ and $(l, j)$[1] are flipped, both $cr_i$ and $sc_j$ are '0' which make the decoder fails to detect while both $cc_k$ and $sr_l$ could be '1'. This syndrome makes the decoder understand that there is one fault and correct the bit $b_{l,k}$.

In summary, PPC could detect multiple faults; however, it is still limited by the undetected patterns. In this work, we will further discuss the behaviors with TSV and the method to overcome the limitation of PPC.

### B. Related Works

This sub-section reviews the fault detection and localization for TSVs or on-chip wires. We classify the existing work into two categories: *TSV testing circuit* and *test scheduling*

*1) TSV testing circuit:* At first, Error Detection/Correction Code [18], [23] (EDC/ECC) could help detect and locate faults in TSVs as normal wires. While EDCs/ECCs usually provide immediate response time, they are limited by a certain number of detectable/correctable defects.

The other approaches is to use testing circuits or BISTs. In [19], [20], [22], the authors present a more fine grain method which could detect open defect using a simple circuit. In [39], the authors presented a grouping method with column and row check for testing both open and bridge defects and reduce the testing time. *Zhao et al.* [19] also injected test pattern to the TSV and capture the output and detect open defects using a NAND gate with logic threshold voltage. For lifetime monitoring, *Serafy et al.* [40] presented a resistance tracking method and *BIST* to overcome the aging in TSVs. The work presented in [9] also proposed a test pattern generator to test open TSV defects while *Loi et al.* [41] used a test access point for injecting and collecting test vectors. In [11], [36], [42], testing is pre-scheduled in order to ensure the

[1]We use the index $(a, b)$ to represent the $a^{th}$ row and $b^{th}$ column. Indexes start from zero.

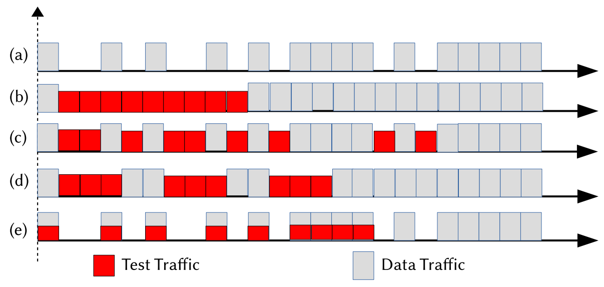

Figure 2: The sequence of data and test traffic under different strategies: (a) application traffic; (b) block test; (c) free time test traffic injection; (d) split free time test [12]; (e) on-communication test.

correctness. In [15], [16], the authors also presented other methods of TSV's *BIST* for pin-hole and void defects. [29] reuses memory BIST for TSV to reduce the test time. Probing before bonding with external testers [17] is also helpful to improve the overall yield rate. While this type of methods (BISTs/dedicated circuits) could provide good faults coverage, their main problem is the need of detaching tested devices/-modules from the system which is not affordable in critical applications.

*2) Test scheduling:* For allocating on-line testing, we adopt the classification in [28]: *anomaly detection*, *Periodic BIST*, *dynamic verification*, or *redundant execution*.

*Periodic BIST* (P-BIST) is the method that activates *BIST* periodically. Here, we focus mainly on NoC testing. In [12] and [33], the tester activates periodically, but only being executed during the free time slots to avoid deactivating the router of NoC under test. They also provide accessibility to the core during the test time. *Huang et al.* [43] also presented a non-blocking testing for NoCs which is a similar idea. In [30], testing for NoC fabrics, which can be used for 3D-NoCs, is presented using dedicated test data and structure. The common goal between these methods is to provide a smart schedule to avoid creating congestion/degradation on the system. Because their experiments are limited in terms of size, the execution time could escalate by complicating the system.

*Redundant execution* could be found in split-link transmission [44] and channel coding [45]. *Dynamic verification* are presented by *Prodromou et al.* [35] with several invariants for on-line testing NoC, [46] with another set for checking transient faults in NoC. *Shamshiri et al.* [47] with end-to-end monitoring. *Anomaly detection* (i.e *Serafy et al.* [40]) uses low-cost hardware or software to indicate anomalous behaviors of TSVs. Although these methods are efficient with deep integration into the system, the vulnerability of TSVs should be delicately addressed (i.e. defect location, real-time detection).

Figure 2 illustrates the different testing strategies. While the blocking test (i.e. P-BIST), depicted in Strategy (b) needs to block the data traffic in order to send the test traffic, Strategies (c) and (d) schedule the test traffic to have less congestion. Strategy (e) represents our OCT methods, where the test is performed together with the data transaction causing no congestion nor performance degradation.

## III. Proposed Defect Detection and Localization

In this section, we present a defect detection and localization method that offers the ability to localize additional faults. First, the localization accuracy is presented. Then, we introduce the *Statistical Detector* and *Isolation-and-Check* methods. Later, applying the *TSV-OCT* to 3D-NoCs is discussed.

### A. Localization accuracy

In terms of detection and localization accuracy, there are four types of results that can be given, as shown in Table I.

Since the data could still use suspicious TSVs, we consider the *false positive* case as acceptable. In *TSV-OCT*, *false negative* is the most critical issue because it makes the system works under unknown defects without any awareness (missed fault case).

Table I: Detection and localization cases.

| | | **TSV status** | |
|---|---|---|---|
| | | Faulty | Healthy |
| **Detection result** | Faulty | *True negative* | *False positive* |
| | Healthy | *False negative* | *True positive* |

### B. Statistical Detector

*1) Hidden error effect:* One of the natural behaviors of an open and short defect is its inconsistency on flipping bit. If a TSV has a short-to-substrate defect and transmits a '0' value, there is no error on the receiver. On the other hand, transmitting a value '1' via short-to-substrate TSV causes flipped bit. If a timing violation occurs due to an open defect, sending the same value as the last transmitted value causes no errors while sending a different value may cause a flipped bit. Due to this characteristic, a TSV region with $N$ defects is likely to have less than or equal to $N$ faults at the same time.

*2) Statistical Detector algorithm:* As we presented in Section II-A, PPC can localize one fault and detect two faults. Here, we exploit the chance that the hidden fault can reduce the number of affected TSVs.

Once the data is received, the decoder tries to detect and localize the faulty positions. Naturally, a detector can correct up to $J$ and detect up to $K$ faults ($J \leq K$). In $T$ transmissions, the detector accumulates faults which are under the localization limitation (less than $J$). After $T$ transmissions, it compares the accumulated number of faults to a threshold (*Thres_Loc*) to find out the possible corruptions. To reduce the cost, we simply set the threshold to 1; however, for removing soft-errors which could be causing flipped bits, we can set *Thres_Loc* to higher values. The details of this method are shown in Algorithm 1. Here, we use two different options for *Statistical Detector* as follows:

- *Cautious localization* (**Opt.=1**): Only indicates the fault position when one fault is left. For instance, only the cases in Figs. 3 (c), (d), and (e) give faulty position.
- *Greedy localization* (**Opt.=2**): As long as the row and column check fails, it determines the position with the corresponding indexes as faulty. For instance, with the syndrome depicted in Figure 3(b), four positions are considered as faulty: (2,0), (2,4), (3,0) and (3,4). Although

this result consists of *false positive* cases, the impact on reliability is not critical.

Figure 3 illustrates the operation of the *Statistical Detector* for TSV regions with 16-data bit, PPC(4×4) and three short (stuck-at-0) defects (positions: (0,3), (2,0), and (3,4)). Because of the hidden effect, there are four possible cases could happen:

**Algorithm 1:** Statistical detector using PPC.

```
// Option for cautious/greedy localization
Input: Opt.
// Column Check (CC) and Row Check (RC)
Input: CC[1:N], RC[1:M]
// Threshold for Localization
Input: Thres_Loc
// Fault indexes
Output: Fault[1:N][1:M]
1  Fault[1:N][1:M] = 0;
2  for (i = 1; i <= N; i++) do
3      for (j = 1; i <= M; i++) do
           // Cautious localization
4          if Opt. == 1 and $\sum CC$ == 1 and $\sum RC$ == 1 then
5              if CC[i] == 1 and RC[j] == 1 then
6                  Fault[1:N][1:M]++;
           // Greedy localization
7          if Opt. == 2 and CC[i] == 1 and RC[j] == 1 then
8              Fault[1:N][1:M]++;
9  for (i = 1; i <= N; i++) do
10     for (j = 1; i <= M; i++) do
11         if Fault[i][j] >= Thres_Loc then
12             Fault[i][j] = 1;
13         else
14             Fault[i][j] = 0;
```

1) **0 hidden defect** (Figure 3(a)): because all three defects cause flipped bits, the detector fails to correct.
2) **1 hidden defect** (Figure 3(b)): because two defects cause flipped bits, the detector fails to correct but it could alert the system.
3) **2 hidden defects** (Figs. 3 (c), (d), and (e)): the detector succeeds to localize one fault position.
4) **3 hidden defects** (Figure 3(f)): the system cannot be alerted because of hidden errors.

The *Statistical Detector* with $Opt. = 1$ (*cautious localization*) only catches the "2 hidden errors" cases (Figs. 3 (d), (e), and (f)) for localization. As shown in Figure 3(g), the system uses $T = 32$ transmissions for *Statistical Detector* with 32 data values (D0, D1, ..., D31). The cases (c), (e), and (f) are hit with D7, D20 and D25; therefore, the *Statistical Detector* can indicate those defects. If one of them is missed, the *Statistical Detector* fails to localize the position. Because the missing case could happen, we observe that using *Greedy localization* could find more faults.

In this work, we opted to use the greedy version (*Opt.=2*) because the *false positive* case is not a critical issue. As shown in Figure 3(g), the *Greedy localization* option tries to cover as much as possible the faulty positions. When one hidden defect (Figure 3 (b)) is hit, it indicates four positions (2,0), (2,4), (3,0) and (3,4) as faulty. Those *false positive* could be remove by using the Isolation and Check algorithm in Sec. III-C.

*3) False negative and false positive cases:* Apparently, the *Statistical Detector* works correctly with two defects and may work with specific sets of 3+ defects. However, with 3+ defects, there are chances to have missed faults (*false negative*) or incorrectly localized position (*false positive*).

Figure 4 shows the pattern when it may miss and incorrectly indicate position. Due to the transmitted data values, only case (a) and (c) indicate two positions where a *false positive* case occurs. Hidden defects could still exist inside the system without being noticed. Therefore, after the *Statistical Detector* completed, there are one *false negative* (TSV at (0,3)) and one *false positive* (TSV at (2,3)).

Here, we observe that the system could further enhance the result of *Statistical Detector*. By improving the detection or localization rate, the system can eliminate the need of dedicated testing while ensuring the reliability of the system.

## C. Isolation and Check

*Isolation and Check*, illustrated in Algorithm 2, is used to solve both *false positive* and *false negative* cases. Because dedicated tester may not be approachable, the *Isolation and Check* method targets to solve this issue based on re-using PPC. The proposed algorithm follows these steps:

- **Step 1:** Using *Statistical Detector* to detect the fault position. These locations are considered as *suspicious TSVs*. We use *Greedy localization* to catch as much as suspicious TSVs as possible. The *false positive* TSVs will be re-checked and corrected later.
- **Step 2:** The system virtually isolates the *suspicious TSVs* from encoding/decoding process; but, they are still being used for data transaction. In other words, *suspicious TSVs* are removed from the parity bit functions in Eq. 1 and 2. It is worth noting that column, row and ultimate parity bits could not be removed but the system can switch the parity bit to different position if needed.
- **Step 3:** Re-run the **Step 1** to **Step 3** until no fault is detected or out of time (until deadline).
- **Step 4:** Reassign each isolated TSV. The TSV could be re-attached to the encoding and decoding process. If a dedicated test is available, using it could reduce the testing time.
- **Step 5:** After Step 4, if the TSV region with isolated TSVs is still detected as faulty, there are unrecognizable faults by *Isolation and Check*. Here, we consider that the whole TSV region as faulty. The system can also consider repeating the *Isolation and Check* to have higher coverage.

By disabling all *suspicious TSVs* and re-running the *Statistical Detector*, the system can localize more faults. Let us consider the case in Figure 4, after using *Statistical Detector* once, two TSVs (0,1) and (2,3) are removed from the decoding and encoding as shown in Figure 5 (a). After isolating the suspicious TSVs, the system keeps running until the checking time is finished ($T = 32$ transactions). If the one hidden defect case (Figure 4 (b)) is hit again, as shown in Figure 5 (f) with $D7$, the system can detect the position (2,1). After concluding that position (2,1) is suspicious, the system isolates it for the next run. Once (0,1), (2,1) and (2,3) are isolated, a hit of 0 hidden defects (Figure 4 (a)) can indicate the last defect at

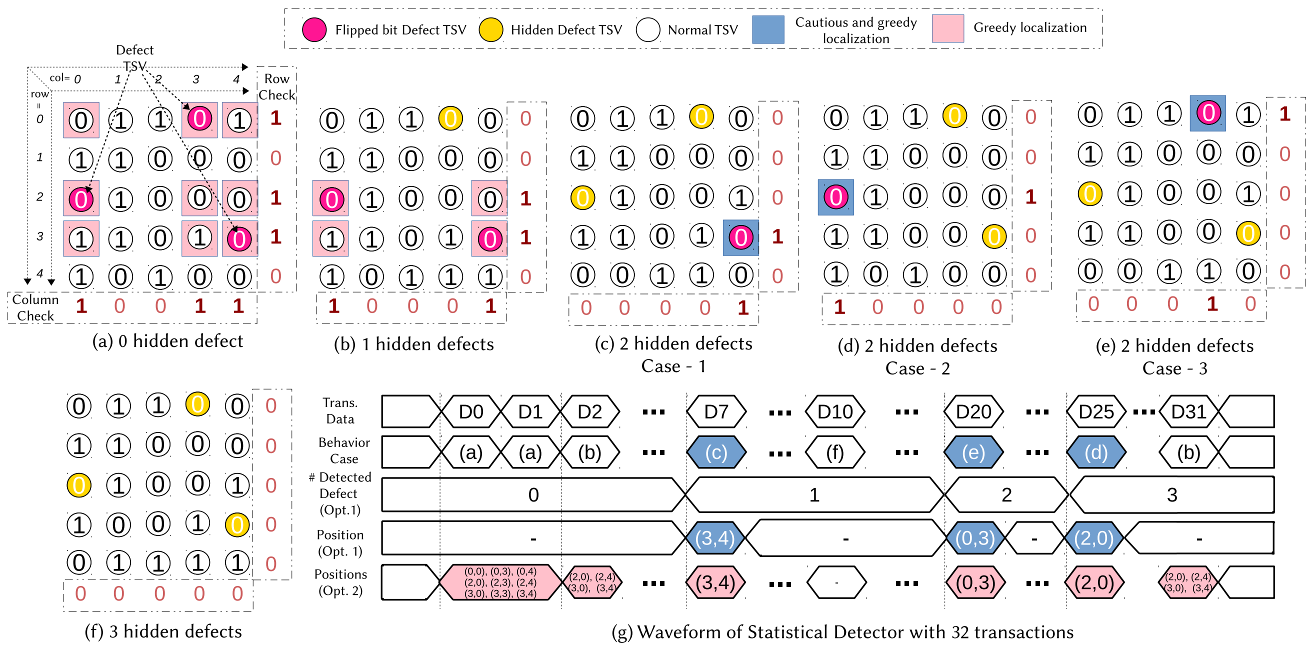

Figure 3: Illustration of Statistical Detector of a TSV region with 16-data bit, PPC (4×4) and three short (stuck-at-0) defects. Flipped bit Defect TSV: input '1', output '0'; Hidden Defect TSV: : input '0', output '0'.

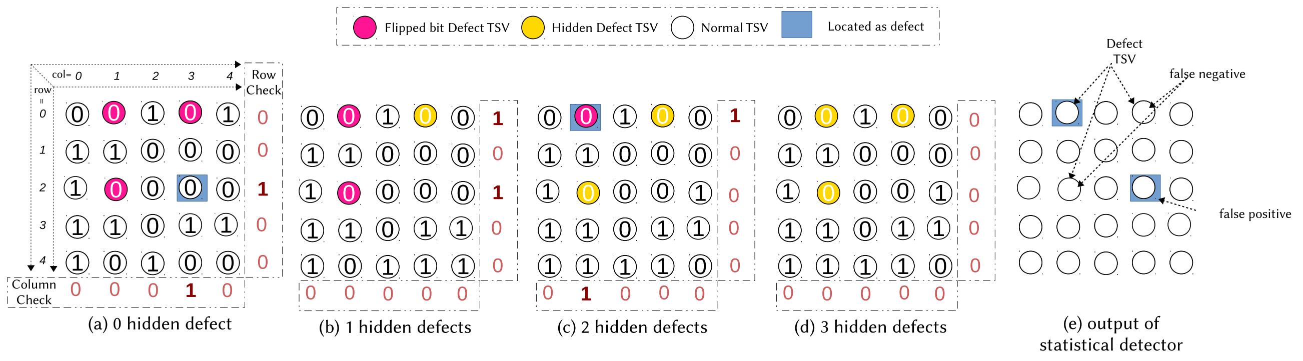

Figure 4: Illustration of *false negative* and *false positive* cases with Statistical Detector for a TSV region with 16-data bit, PPC (4×4) and three short (stuck-at-0) defects. Due to the transmitted data values, only cases (a) and (c) indicate two positions as faulty where a false positive case occurs in (a).

(0,3) (see Figure 5 (d, f) with $D55$). At the end of *Step 3* with no more position detected, the *Isolation and Check* can cover all faulty positions. However, there is a *false positive* case remaining.

*Steps 4 and 5* of the *Isolation and Check* algorithm are illustrated in Figure 6. At the end of *Step 3* (see Figure 5), four positions are indicated as suspicion. In *Step 4-5*, the algorithm re-enable each suspicious TSV to confirm it correctness. The algorithm first enables the TSV (0,1) and performs data transactions. Because this TSV is faulty and $D11$ in Figure 5(f) causes a faulty output, the system can easily conclude after $T$ transmissions that it is defected. If the *false positive* case (TSV (2,3)) is re-enabled, no faulty output is found. The system can conclude it as non-faulty and remove it from the list. After testing each suspicious TSV, the system can finally conclude the faulty positions.

Because intermittent faults occur in certain conditions and may vary among the products due to process variation, *TSV-OCT* detects TSV intermittent defects by constantly monitoring the operation of TSVs. However, it is important to mention that the condition to successfully execute this detection is that these intermittent faults must last at least 2× WCET (Worst-Case Execution Time). Such a duration ensures at least one complete run of TSV-OCT to detect the fault. Nevertheless, last changing intermittent or transient faults could be corrected by the built-in ECC in *TSV-OCT*.

## *D. Implementation on a 3D-Network-on-Chip*

In order to understand the cost of the design, we integrate the *TSV-OCT* into our previously designed 3D-NoC router [26], as shown in Figure 7. Please note that the proposed approach is totally independent from our opted router architecture and could be implemented into any TSV-based architecture. The PPC is integrated as an ECC module and the

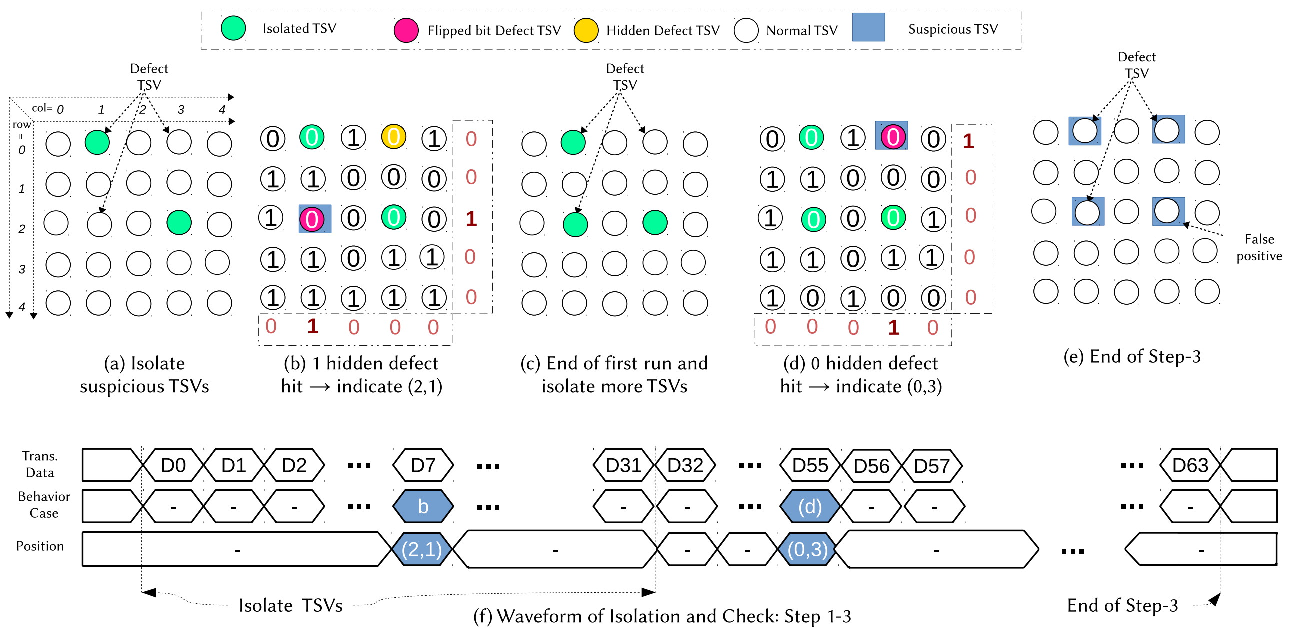
Isolated TSV
Flipped bit Defect TSV
Hidden Defect TSV
Normal TSV
Suspicious TSV
Defect TSV
(a) Isolate suspicious TSVs
(b) 1 hidden defect hit ⟶ indicate (2,1)
(c) End of first run and isolate more TSVs
(d) 0 hidden defect hit ⟶ indicate (0,3)
(e) End of Step-3
False positive
Trans. Data
Behavior Case
Position
Isolate TSVs
End of Step-3
(f) Waveform of Isolation and Check: Step 1-3

Figure 5: Isolation and Check illustration: *Step 1 to 3*.

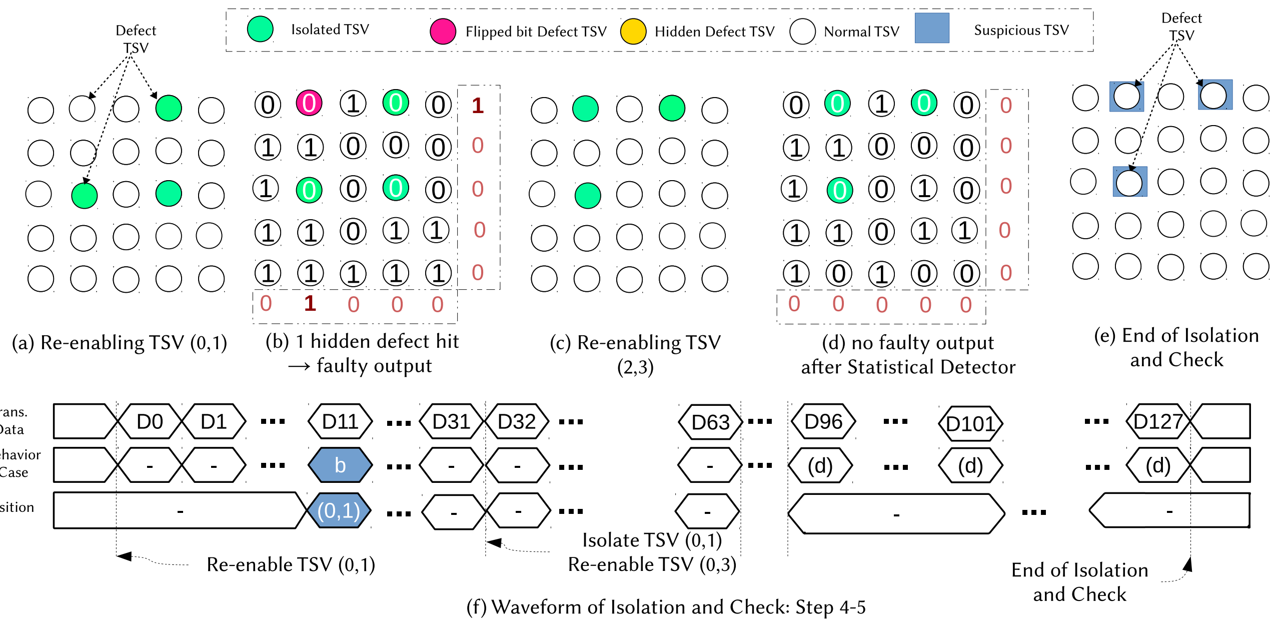
Isolated TSV
Flipped bit Defect TSV
Hidden Defect TSV
Normal TSV
Suspicious TSV
Defect TSV
(a) Re-enabling TSV (0,1)
(b) 1 hidden defect hit ⟶ faulty output
(c) Re-enabling TSV (2,3)
(d) no faulty output after Statistical Detector
(e) End of Isolation and Check
Trans. Data
Behavior Case
Position
Re-enable TSV (0,1)
Isolate TSV (0,1)
Re-enable TSV (0,3)
End of Isolation and Check
(f) Waveform of Isolation and Check: Step 4-5

Figure 6: Isolation and Check: *Step 4 to 5*.

*TSV-OCT* is only integrated into two vertical ports (UP and DOWN) to monitor and detect faults of TSVs. The data from TSV is brought to the *Statistical Decoder* then sent to the input buffer. The syndromes are collectively received and analyzed by "StatD" (*Statistical Detector*). The output is updated to the fault table ("F-table") while the controller issues a control signal for iterations and check. Previously in [26], we used two SECDED (16,22) codes to handle potential soft errors in the data. Here, we use a single PPC $(4 \times 8)$ which requires 45 codeword-bit. This leads to only one extra TSV to be added (44 + 1 in total).

In this implementation, we assume that the synchronization between two terminals (encoder and decoder) is done by a safety reliable channel. The switching between modes (isolation position, check etc...) should be synchronized with two identical timers in two layers.

In terms of scalability, by using TSV-OCT in each X-bit connection, the WCET of the system is equal to the WCET of one X-bit connection. In other words, assuming that $n$ is the total number of TSVs in the system, conventional testing methods either require $O(n)$ for testing time (serial test) or $O(n)$ for faulty information collection time (parallel test). On the other hand, the WCET of the proposed TSV-OCT system is always $O(X)$ regardless of the total number of TSVs employed in the system.

**Algorithm 2:** Isolation and Check algorithm.

```
   // Column Check (CC) and Row Check (RC)
   Input: CC[1:N], RC[1:M]
   // Threshold for Localization
   Input: Thres_Loc
   // Fault indexes
   Output: Fault[1:N][1:M]
   // Run the first time
1  Isolation[1:N][1:M] = 0;
2  Fault[1:N][1:M] = Statistical_Detector(CC, RC, Thres_Loc);
   // Isolate fault and recheck the second time
3  Isolation[1:N][1:M] = Fault[1:N][1:M] Fault[1:N][1:M] +=
     Statistical_Detector();
   // Un-isolate each position and recheck the second
      time
4  for (i = 1; i <= N; i++) do
5      for (j = 1; j <= M; j++) do
6          if Fault[i][j] == 1 then
7              Isolation[i][j] = 0;
8              TempFault[1:N][1:M] = Statistical_Detector();
9              if TempFault[1:N][1:M] == 0 then
                   // not a faulty position
10                 Fault[i][j] = 0;
11             else
12                 Isolation[i][j] = 1;
```

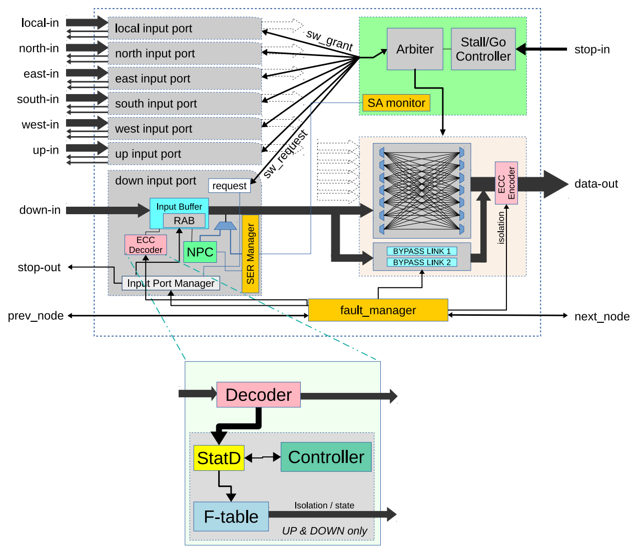

Figure 7: 3D-NoC router with the proposed on-line fault detector. Note that there are only two modules in the router for UP and DOWN connections.

## IV. Evaluation

### A. Evaluation Methodology

The proposed system was designed in Verilog-HDL, synthesized and prototyped with commercial CAD tools. We use NANGATE 45nm library [48] and NCSU FreePDK TSV [49]. The TSV size and pitch are $4.06\mu m \times 4.06\mu m$, and $15\mu m$, respectively.

In this section, we first evaluate the *Statistical Detector* performance in terms of detection and localization rate. Then, the performance of *Isolate and Check* is also investigated. We use two fault models: (1) stuck-at-0 for short-to-substrate defects; and (2) delay value of one clock cycle for open defects. Because the data is randomly generated, the hidden fault probability is about 0.5 for both models. Here, we use four different data widths: 8, 16, 32 and 64. We also vary the number of transactions $T$= 8, 16, 32, 64 and 128. At the end of these evaluations, we aim to provide a comprehensive coverage of the possible combinations. Note that the Monte-Carlo based method is used to perform the proposed method. We run with 100,000 random samples for each case of these evaluations. As a supplement to the detection rate, we evaluate the minimum, maximum and average response time of the *TSV-OCT*. Later, we show the hardware implementation of the design as well as some comparisons. We first compare with the well-known ECCs then with *BIST* and other testing methods.

### B. Statistical Detector Performance

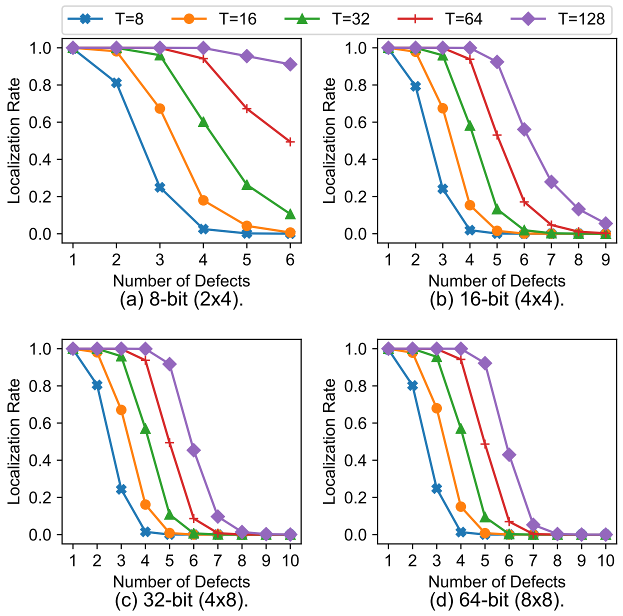

Figure 8: Result of standalone Statistical Detector (including *false positive*).

Figure 8 shows the performance results when only using *Statistical Detector* (without *Isolation and Check*) including *false positive* cases. We can easily observe that the localization rate of higher number of transactions $T$ values is better than the lower ones. This could be easily explained by the higher probability of silent faults that could be dropped. With $T = 128$, the *Statistical Detect* localizes at least 45% of 6 faults, 99% of 3 faults and 100% of 2 faults. Therefore, the *Statistical Detector* has significantly improved the localization rate of the ECC's bound (naturally localizing 1 fault and detecting 2 faults).

Even without *false positive* cases, the localization rate of the *Statistical Detect* easily outperforms the baseline ECC (Parity Product Code), as depicted in Figure 9. *TSV-OCT* still guarantees at least 80% of two faults in the worst case.

In both cases of with and without *false positive*, we can observe that the *Statistical Detector* could not give any solid performance with 3+ faults. In order to improve it, the *Isolation and Check* should be used.

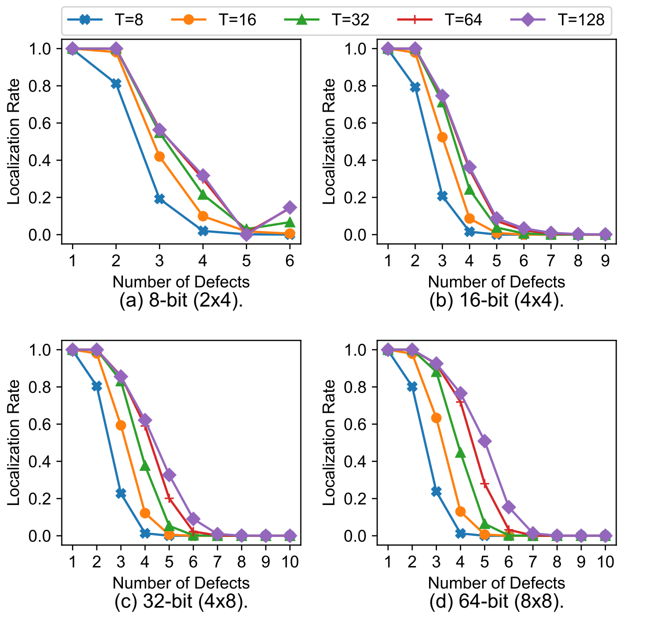

Figure 9: Result of standalone Statistical Detector (excluding *false positive*).

## C. Isolation and check performance

This section evaluates the Isolation and Check in two parts: *Step 1-3* and *Step 4-5*.

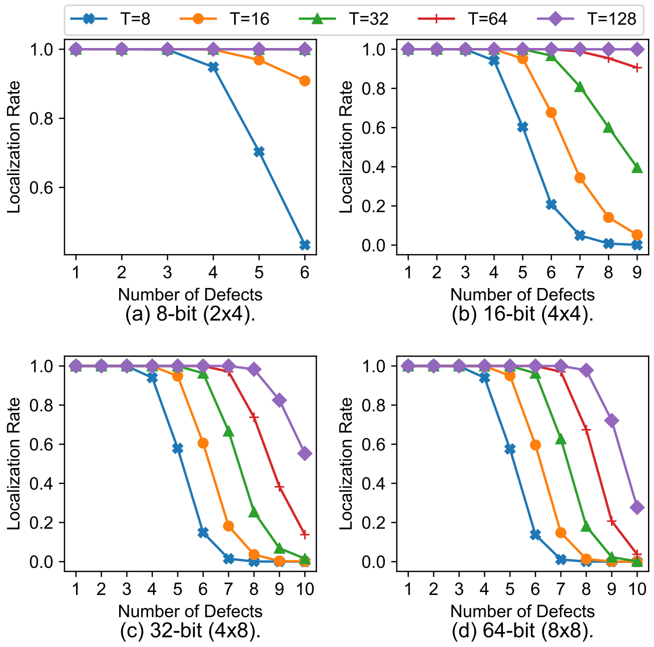

Figure 10: Result of Statistical Detector and Isolation (*Step 1-3*, including *false positive*).

*1) Step 1-3:* At first, we evaluate **Step 1-3** of the *Isolation and Check* method to see how much these three steps can improve the defect detection when compared to the *Statistical Detector*. As shown in Figure 10, we can observe that $T = 128$ can cover most of less than 6 faults (including *false positive*). The lower $T$ values give less improvement than 128; however, in comparison to *Statistical Detector*, the *Isolation and Check* method gives significant improvements ($\simeq 2\times$ localization rate).

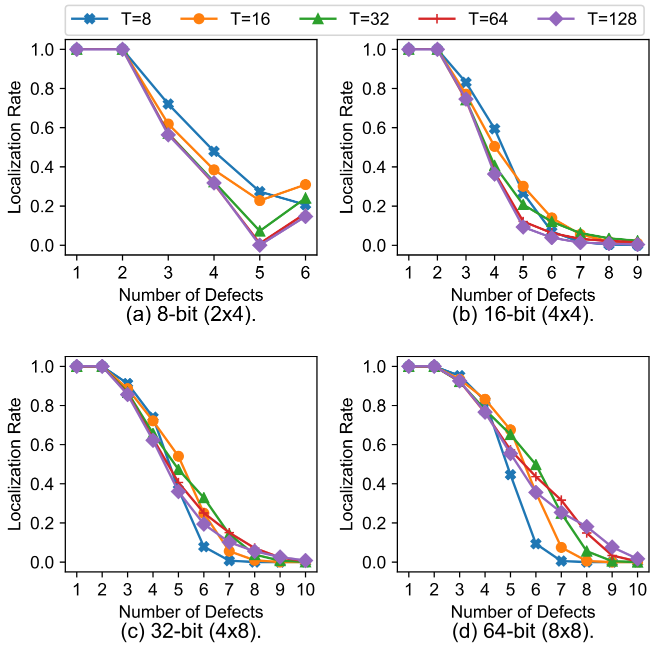

Figure 11: Result of Statistical Detector and Isolation (*Step 1-3*, excluding *false positive*).

Figure 11 shows the localization rate without *false positive*. We now can see that the system can localize two faults in all cases. For other cases, we can see the drops of accuracy because healthy TSVs are considered as faulty.

*2) Step 4-5:* As shown in Figure 12, we can observe that further extension of the algorithm to Step 4-5 helps guarantee that healthy TSVs being not labeled as defected. While isolation may give mixing results between various $T$ values, re-checking gives a more reasonable result where higher $T$ values give better results.

As shown in Figure 12, the localization rate strongly depends on the number of checked transactions $T$ by the detector. With only $T = 8$, it successfully detects less than 40% of the six defects. Once $T = 32$ is used, two defects could be 100% detected. This could be explained by the less chance of hidden errors in multiple transactions. On the other hand, these results also imply a significant improvement of using *Statistical Detector*. As long as the system keeps sending flits via faulty TSVs, the *Statistical Detector* can detect them. In this Monte-Carlo simulation, we observe that the system can detect 100% of two faults up to $T = 32$ with a very short response time (it requires 32 cycles to complete 32 transactions). On the other hand, we observe that increasing the $T$ value from 64 to 128 does not significantly improve the overall localization rate.

Considering the realistic defect rate of TSVs, the rate could vary between 0.001% to 5% [39]. Here, we assume the defect rate is 5% where 8, 16, 32, and 64 data-bit have about 1, 2, 3, and 5 defects respectively. As shown in Figure 12, our method can cover 5 faults in all cases with T=128, which is enough to

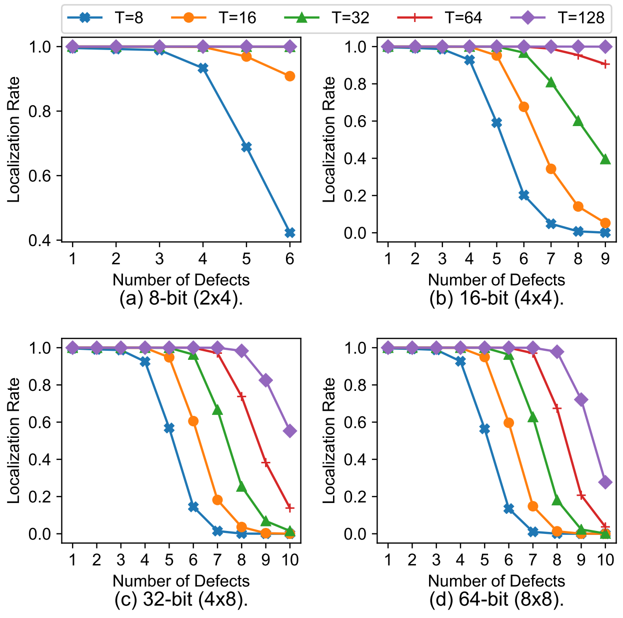
Figure 12: Result of *TSV-OCT* (excluding *false positive*).

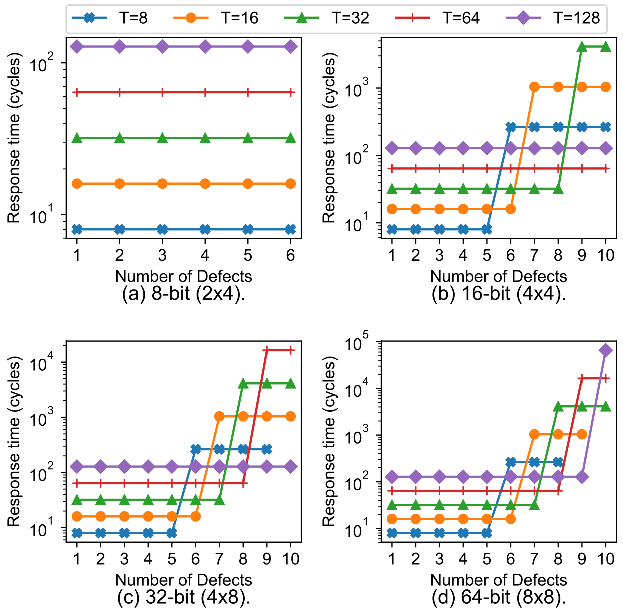
Figure 13: Minimum response time.

satisfy the assumed 5% maximum defect rate. Also, even the defects are not completely matched with lower T values, the system is still aware of the non-localized defects, as shown in Equation 3.

## D. Response time

In this section, we evaluate the response time of the proposed methods when considering three aspects: minimum, maximum, and average response time to the new fault. The response time is considered as the time from the occurrence of a new fault until its detection (including *false positive* cases). Figures 13, 14, and 15 show the three aspects' results. Note that each transaction is assumed to cost one clock cycle. A dummy value (for instance, [50] uses zeros and ones vector to test) could be used to test if the connection is free.

In terms of minimum response time, as illustrated in Figure 13, smaller $T$ values give much faster response time. Please note that this is the minimum value of the successful cases. By having more faults in the system, the minimum response time could be increased. For smaller $T$ values (8, 16), they have lower minimum response time when increasing the number of defects; however, their coverage has been dropped significantly. We can also notice the non-existing cases: [$T = 8$ 32-bit 10-fault], [$T = 8$ 64-bit 9-fault], [$T = 8$ 64-bit 10-fault] and [$T = 16$ 64-bit 10-fault] where the system fails to correctly localize any case in the Monte-Carlo simulation.

For the case of maximum response time, depicted in Figure 14, we can observe a similar behavior as minimum response time where the smaller $T$ values give lower maximum response time. This is predictable because the number of cycles in the algorithm is shorter. However, when increasing the number of faults, the number of required cycles becomes significantly high. Especially, using $T = 128$ costs more than 60.000 cycles to finish in high defect rates.

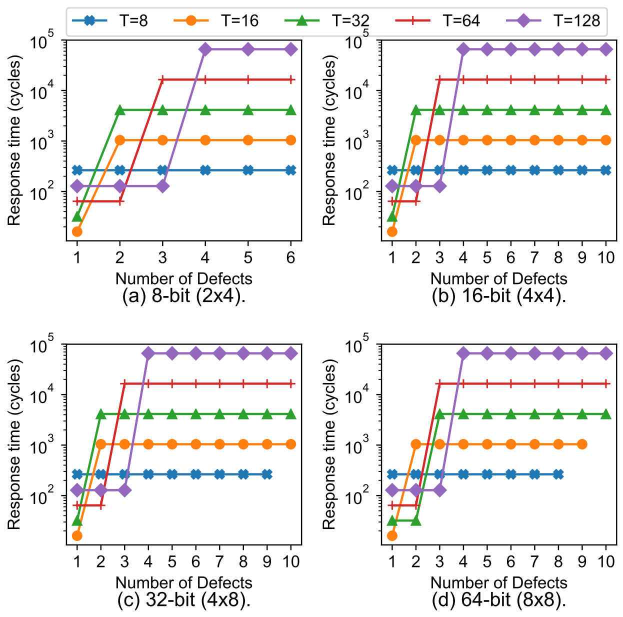
Figure 14: Maximum response time.

Figure 15 shows the average response time. As we can notice, the average response time could be significantly increased up to around 16,000 cycles when using $T = 64$. As presented in the previous evaluations (minimum and maximum response time), the expected response time of higher values of $T$ are worse than smaller ones. However, we also can observe the case where smaller $T$ values give higher average response time. This is due to the fact that having smaller T values leads to higher hidden fault probabilities which need more iterations for localization. Also, the drop on coverage of smaller $T$ values should be considered.

In summary, we can easily observe the linear relation between the response time with $T$. With smaller number of

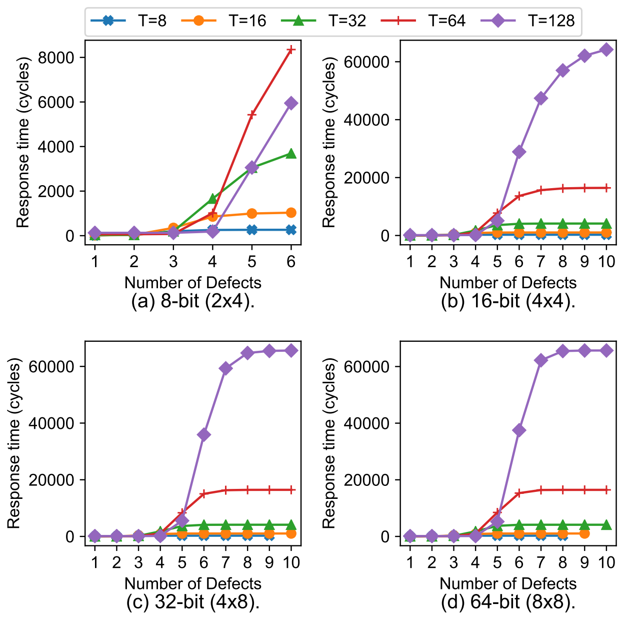

Figure 15: Average response time.

defects, this could be a critical issue. However, with a greater number of defects, the trade-off between coverage rate and response time should be considered. Also, by repeating the smaller $T$ values, the system may catch more faults during operation. For instance, running $T = 8$ twice could give similar fault coverage as $T = 16$. Please note that diving TSV into clusters does not increase the testing time. Meanwhile, *P-BIST* encounter scalability issues and blocking issues for large scale testing which may significantly impact the performance. Authors in [12] pointed out their lower bound of testing period of a $10 \times 8$ mesh NoC is 16,840 cycles.

## E. Hardware Implementation

Detailed results of 32 data bit implementations are shown in Table II. For a comprehensive comparison, we select the most common techniques in ECC such as: Hamming, SECDED and several multiple fault correction techniques (SEC-DAEC, TAEC). However, this work only focuses on improving the ability of PPC instead of providing alternative error correction coding methods.

In the case of the encoder, the complexity of our method is lower than SEC-DAEC (Single Error Correction, Double Adjacent Error Correction) and TAEC (Triple Adjacent Error Correction) [51], [52] but higher than Hamming and SECDED. The area cost of the proposed decoder is higher due to the fact the system requires registers to collectively store the output syndrome. However, for a delay optimized design, we offer smaller design than multiple errors correction. It is worth mentioning that these techniques only correct adjacent errors. Also, our latency is smaller thanks to the short critical path where PPC $(4 \times 8)$ only calculates the parity for 8-bit instead of 32-bit. Our decoder area is also higher than the Hamming and SECDED and similar to TAEC and SEC-DAEC methods with area optimization. For latency optimization, our area cost is smaller than all multiple error corrections.

In comparison to the baseline PPC $(4 \times 8)$, the proposed architecture increases $1.70\times$ and $11.54\times$ the encoder and decoder area cost, respectively. Here, both of the encoder and decoder area of our method have larger area cost due to the need of isolation. The latency is also degraded by less than 0.1 $ns$, which still offers the ability to operate at extremely high frequencies. In terms of power, the encoder demands similar values as the baseline; however, the decoder requires nearly $9\times$ power. Although the decoder requires high area and power overhead, these results are expected because of the added extra modules and computation.

Table III shows the result of implementing *TSV-OCT* into a 3D-Network-on-Chip. As previously stated, we adopt our previous work [53] for comparison. For a fair comparison, we also added the result of the baseline model [54] without protection. The area cost of the final router only increases by 9.17% when compared to the previous work [53]. Within the router, the area cost of the encoder is insignificant with less than 0.5% while the decoder occupies 7.94% of the router. For two vertical connections using TSVs (up and down), the area of the decoder takes 15.88% of the total area. Despite of having higher area costs, as represented in Table II, the overhead of the proposal inside a given 3D-NoC router is totally reasonable. Also, the proposed approach offers a much better fault detection and localization rates. In comparison to the baseline model [54], *TSV-OCT* router adds 41.5% of area overhead; however, our work provides protection on links and localization of defects, which makes it totally reasonable.

## F. Comparison

In this section, we compare *TSV-OCT* with existing works targeting TSV/NoC testing in Table IV. Here, we focus on three key parameters: area overhead, performance degradation and response time.

In summary, our result with $[4\times8$, T=64] offers the smallest testing time among all other on-chip communication testing and are not affected by any scalability issue. However, these online testing methods [30]–[32] offer better coverage than ours where we only offer testing for TSV only. However, we have to notice that our technique, as an ECC-based technique, has no degradation in the overall performance; regardless of the number of faults.

In comparison to the circuit-based testing methods [19], [20], [39], [55], [56], where each TSV needs from a half to couple of cycles to be tested, our proposed technique has longer testing time (average: 85.47 cycles per TSV). In terms of area cost, our results give a reasonable overhead for TSV. The proposed TSV-OCT area is $50.92\mu m^2$ per TSV. However, all TSV-testing designs in [19], [20], [55], [56] obtain better area overhead than ours. This could be explained by the high number of registers needed in our design (accumulating faults and isolation registers). Among the aforementioned works, the one presented by *Lee et al.* [39] offers the best test-time, area per TSV and TSV-to-TSV bridge defect detection. Nevertheless, *TSV-OCT* is the method could work online without any degradation while these circuits must interrupt the connect to test.

Table II: Hardware implementation results.

| Scheme | | Tech. ($nm$) | k (bit) | n (bit) | Area Cost ($\mu m^2$) | | Latency ($ns$) | | Power ($\mu W$) | | Response (*cycles*) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Encoder | Decoder | Encoder | Decoder | Encoder | Decoder | Min | Max |
| Hamming [18] | | 45 | 32 | 39 | 94.1640 | 234.8780 | 0.55 | 1.12 | 30.0831 | 96.2898 | 1 | 1 |
| SECDED [23] | | 45 | 32 | 40 | 111.7200 | 253.7640 | 0.60 | 1.44 | 36.9622 | 103.1422 | 1 | 1 |
| SEC-DAEC [51][a] | | 45 | 32 | 39 | 322 | 1902 | 0.53 | 1.33 | - | - | 1 | 1 |
| TAEC [52][a] | | 45 | 32 | 40 | 264 | 2628 | 0.45 | 1.32 | - | - | 1 | 1 |
| PPC($4 \times 8$) | | 45 | 32 | 45 | 76.6080 | 187.2640 | 0.30 | 0.68 | 43.0272 | 129.4174 | 1 | 1 |
| **TSV-OCT** ($4 \times 8$), T=64 | Total | 45 | 32 | 45 | 130.3400 | 2161.2500 | 0.39 | 0.72 | 48.639 | $1.04 \times 10^3$ | 64[b] | 16,448[b] |
| | PPC | 45 | 32 | 45 | 130.3400 | 327.4460 | - | - | 48.639 | 198.424 | - | - |
| | Stat_det | 45 | 32 | 45 | - | 751.1840 | - | - | - | 408.621 | - | - |
| | Isol_Check | 45 | 32 | 45 | - | 1016.3860 | - | - | - | 387.056 | - | - |

[a] We use the area optimization and lowest area cost design since our design is optimized for area cost.
[b] More details about response time and localization rate could be seen in Section IV-C and IV-D.

Table III: Hardware complexity of 3D NoC router (32-bit).

| Design | Specification | ECCs | Module | Area | |
|---|---|---|---|---|---|
| | | | | ($\mu m^2$) | (ratio to baseline) |
| Baseline router [54] | Wormhole, 4-flits buffer, 32-bit data, 3D Mesh | None | Router | 18,873 | - |
| | | | TSVs[1] | 1,054.95 | - |
| | | | Total | 19,927.95 | - |
| SECDED router [53] | | 2× SECDED (22,16) | Router | 24,519 | (129.92%) |
| | | | TSVs[1] | 1,451.56 | (137.50%) |
| | | | Total | 25,970.56 | (130.32%) |
| TSV-OCT router | | PPC($4 \times 8$) T=64 | Router | 26,843 | (142.23%) |
| | | | Encoder[2] | 130 | - |
| | | | Decoder[2] | 2161 | - |
| | | | TSVs[1] | 1483.52 | (140.63%) |
| | | | Total | 28,326.52 | (142.14%) |

[1] TSV area: $4.06\mu m \times 4.06\mu m = 16.4836\mu m^2$.
[2] The area cost details of the employed encoder and decoder are shown in Table II.

Table V also shows a comparison for on-line NoC testings with 32-bit flit and in 45 $nm$ technology. While *TSV-OCT* targets only vertical wires (TSVs), we compare with existing techniques offering a similar or higher coverage. Please note that the results of other works do not include BIST area and power consumption. Also, *TSV-OCT* does not require adaptive routing to perform testing because it is an on-communication test (non-blocking testing).

As can be observed in Table V, our technique offers a smaller area cost than *Wang et al.* [12] and a larger area cost than other designs because our technique does include testing circuit. The power consumption of our technique is higher than existing works because the register of statistical detector and isolation-and-check cost nearly 70% of the total amount. However, in contrast to all existing techniques, *TSV-OCT* does not degrade the performance of the applied NoC. *Kakoee et al.* [33] and *Tran et al.* [57] have a significant impact on the performance which are $> 10\times$ and $> 1.4\times$ the average latency of synthetic benchmarks and execution time of PARSEC, respectively. Although *Liu et al.* [58] and *Wang et al.* [12] give promising results under PARSEC benchmarks because of low utilization rates, they still increase the average latency by up to $2.5\times$ and $2\times$ under synthetic benchmarks.

On the other hand, *TSV-OCT* offers no change in the system performance under test while it guarantees the response time under 20.000 cycles (see Section IV-D for more details). Please note that the upper bound of our technique is 16,448 cycles which is under the lower bound of *Wang et al.* [12] (16,840 cycles). Meanwhile, *Kakoee et al.* [33] and *Tran et al.* [57] have significant higher lower-bounds (200.000 or 500.000 cycles) which may not be suitable for real-time applications.

### G. Discussion

In the previous evaluations, we have presented the efficiency of *TSV-OCT*. Despite the obtained advantages, there are some challenges that should be addressed in order to further enhance the detection ability of *TSV-OCT*, as discussed hereafter.

As previously mentioned, *TSV-OCT* also tackles intermittent faults. Since the WCETs of *TSV-OCT* are fixed, designers could choose a proper configuration to ensure the detection and localization of this type of faults during the Intermittent Detection Window (IDW), which is defined as $2\times$ WCET.

In our two fault models, we have not considered the metastability phenomenon. However, our design is compatible with metastability immune circuits [21], [59]. To avoid metastability, one of these methods can be easily adopted for each TSV before the detection and localization.

Although this work has been evaluated and compared in detection and localization efficiencies, the impact of real-chip fabrication and Process Voltage Temperature (PVT) variations have not been studied. The PVT or real-chip measurement could provide more realistic result on the timing behavior of the circuit; however, theses variations have small impact on the efficiency of the algorithm. Nevertheless, PVT or real-chip measurement should be studied in the future to provide better understanding of our proposal.

Testing methods for TSV-to-TSV bridge defects [39] have not been evaluated in our paper. The performance of TSV-OCT on detecting and localizing this type defect will be investigated in the future. We would like to note that among the conducted 10,000 Monte-Carlo simulation cases for each configuration, there are multiple cases having two or more adjacent defected TSVs.

Despite the above limitations, *TSV-OCT* still provides extra defects' localization while maintaining short execution time. The exhibited overhead in a 3D-NoC implementation is also reasonable which makes *TSV-OCT* totally feasible for integration into highly reliable 3D-ICs.

## V. Conclusion

This paper presents a method to improve the localization rate of Parity Product Code (PPC) to enhance the reliability of TSV-based 3D-IC designs. From the conducted experiments, and in contrast to the baseline PPC that are limited to localizing one fault at most, *TSV-OCT* has demonstrated its ability to localize more than six faults. Furthermore, *TSV-*

Table IV: Comparison table with existing works in TSV testing. S: number of signal TSVs; R: number of redundant TSVs.

| **Work** | **Brief description** | **Test type** | **Tech.** | **Config.** | **Area w.o. TSV** ($\mu m^2$) | **Test Time** (cycles) |
|---|---|---|---|---|---|---|
| *Zhang et al.* [55] | Detect by capacitive and resistive measuring<br>Recovery using redundancies for TSV array<br>Assignment and collection using scan/config chain | post-bond | $45nm$ | S: 96<br>R: 24 | self-test: $1,128$<br>control logic: $281.3$<br>per TSV:$11.7$ | short: 3 per TSV<br>open: 2 per TSV<br>total: 1.04 per TSV |
| *Zhao et al.* [19] | Detect by using NAND gate with one input is logic threshold voltage<br>Recovery using redundancies<br>Assignment and collection using scan/config chain | online | $130nm$ | S: 9112<br>R: 114 | detection: 111,403<br>recovery: 506,310<br>routing: 312,542<br>per TSV: 33.876 | test: S+R (1 per TSV)<br>repair: S+R (1 per TSV)<br>total: 2(S+R) (2 per TSV) |
| *Cho et al.* [20] | detect the signal degradation through TSVs due to resistive shorts and variations using voltage comparator.<br>Recovery using the output of voltage comparator. | pre-bond | $90nm$<br>&<br>$45nm$ | S: 1444 | area: ≃46,656 (45nm)<br>per TSV: ≃21 (45nm) | test & recovery: 1 per TSV |
| *Jani et al.* [56] | Design for Cu-Cu hybrid bonding (pitch $\leq 2\mu m$)<br>Measure the misalignment defect & RC delay<br>Assignment and collection using scan/config chain | post-bond | $28nm$ | S: 10,000 | passive: 61,370<br>active logic: 28,600<br>per TSV: 8.997 | alignment: 1 per TSV<br>RC: 1 per TSV |
| *Lee et al.* [39] | TSV-to-TSV bridge and open defect test<br>TSVs are divide in a group of $N = n \times n$ TSVs | post-bond | $45nm$ | S: 1,000 | total: 1130.5[1]<br>per TSV: 1.1305 | Total: 0.5 per TSV |
| *Li et al.* [29] | On-chip test framework for 3D-IC<br>TSVs are tested "for free" during memory BIST<br>The time testing TSV is higher due to waiting | offline & post-bond | $90nm$ | 512 data<br>48 address | Mechanism: 75,322.8<br>BIST : 4,673.2<br>Pattern gen.:111,524.4<br>Per TSV: 342.0 | system: 11,009,580<br>data TSV: 114<br>address TSV: 307<br>Per TSV: 0 (test with mem.) |
| *Grecu et al.* [30] | NoC testing<br>The link test could be used for TSV | online | $90nm$ | link: 32<br>mesh:16×16 | (gate count)<br>unicast: 524/switch<br>multicast: 1025/switch | unicast: 223,368<br>multicast: 15,233 |
| *Amory et al.* [31] | Test method for NoC that provide scalability. Testing router by comparing output with equal inputs. Test wrapper is inserted around the NoC. | online | $0.35\mu m$ | link:20<br>mesh:5×5 | NoC: 9491 gates<br>switch:379.64 gates | 11,206 |
| *Xiang et al.* [32] | Multi-cast and thermal aware testing for 3D-ICs. The method provides lower test column and temperature. | online | gate | $4 \times 4 \times 4$ | area overhead 2.4 | 221,119 |
| **TSV-OCT** | On-communication test method | online | $45nm$ | ($4 \times 8$, T=64)<br>S: 32*G<br>R: 13*G | total:2,291.59<br>per TSV: 50.92 | best: 64 (1.43 per TSV)<br>worst: 16,448 (365.51 per TSV)<br>average: 8,346 (85.47 per TSV) |

[1] The estimation is based on the results represented in the paper.

Table V: Comparison of online testing circuit for Network-on-Chip router (32-bit and 45 $nm$ technology).

| **Design** | *Kakoee et al.* [33][a] | *Tran et al.* [57][a] | *Liu et al.* [58][a] | *Wang et al.* [12][a] | **TSV-OCT** |
|---|---|---|---|---|---|
| **Coverage** | Link | Link | Router's modules | Link & Router's module | Link (TSVs) |
| **NoC size and topology** | 10×8 Mesh | | | | any NoC[e] |
| **Require adaptive routing** | ✓ | ✓ | ✓ | ✓ | ✗ |
| **Performance degradation** | ✓ | ✓ | ✓ | ✓ | ✗ |
| Synthetic, Period = 20K cycles[b] | $> 10\times$ | $> 10\times$ | $1.0 - 2.5\times$ | $1.5 - 2\times$ | $1.0\times$ |
| PARSEC, Period = 40K cycles[b] | $> 1.8\times$ | $> 1.4\times$ | $0.9 - 1.0\times$ | $0.9 - 1.0\times$ | $1.0\times$ |
| **Test period/Test time** (*cycles*) | 500K-1M[c] | 200K-1M[c] | 20K- | 16,840- | 64-16,448 |
| **Area** ($\mu m^2$) | 700 | 700 | 2200 | 2400 | 2291.59 |
| **Power** ($\mu W$)[d] | 8.77 | 9.18 | 18.27 | 18.1 | 1088.639 |
| **Test circuit** (area and power) | ✗ | ✗ | ✗ | ✗ | ✓ |

[a] Area and power costs are extracted by subtracting to the non-test router design on paper [12]. Area cost and power consumption of non-test router [12] are 0.0251 $mm^2$ and 391.00 $\mu W$, respectively.
[b] Performance values, which are extracted from paper [12], are approximated values. The worst case test time (upper bound) of ours is 16,448 cycles which is under the period (20K/40K cycles). The lower bound of testing period for *Wang et al.* [12] is 16,840 cycles.
[c] Test time is extract from paper [12]. Values are selected based on reasonable performance degradation ($> 5\times$ average latency and $> 1.8\times$ execution time).
[d] Power consumption of our design is based on a 500MHz implementation. Other designs' frequency is unclear [12].
[e] Our proposal could be also applied for link testing.

*OCT*'s response time is guaranteed under a certain time which makes it suitable for real-time applications.

As a future work, we plan to apply *TSV-OCT* to a dedicated application together with soft and hard fault tolerance to obtain a comprehensive method. In depth analyses using PVT simulation and real-chip fabrications could provide different aspects on the efficiency of our method. Also, since TSV-OCT could work with different mediums, applying our proposal to normal wires or memories could also be a viable direction.

## Acknowledgment

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

The authors would like to give special thanks to the reviewers and editors for their excellent comments that help improve the paper.

## References

[1] J. Cho *et al.*, "Modeling and analysis of through-silicon via (TSV) noise coupling and suppression using a guard ring," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 1, no. 2, pp. 220–233, 2011.

[2] X. Dong and Y. Xie, "System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs)," in *Asia and South Pacific Des. Automation Conf.*, 2009, pp. 234–241.

[3] W. R. Davis *et al.*, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Des. Test. Comput.*, vol. 22, no. 6, pp. 498–510, 2005.

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

[5] K. N. Dang *et al.*, "A comprehensive reliability assessment of fault-resilient network-on-chip using analytical model," *IEEE Trans. VLSI Syst.*, vol. 25, no. 11, pp. 3099–3112, Nov 2017.

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

[7] Y. J. Park *et al.*, "Thermal analysis for 3D multi-core processors with dynamic frequency scaling," in *2010 IEEE/ACIS 9th Int. Conf. on Comput. and Inform. Sci.* IEEE, 2010, pp. 69–74.

[8] T. Frank *et al.*, "Reliability of TSV interconnects: Electromigration, thermal cycling, and impact on above metal level dielectric," *Microelectron. Reliab.*, vol. 53, no. 1, pp. 17–29, 2013.

[9] 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.

[10] F. Ye and K. Chakrabarty, "TSV open defects in 3D integrated circuits: Characterization, test, and optimal spare allocation," in *The 49th Annu. Des. Automation Conf.*, 2012, pp. 1024–1030.

[11] 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.

[12] J. Wang *et al.*, "Efficient design-for-test approach for networks-on-chip," *IEEE Trans. Comput.*, vol. 68, no. 2, pp. 198–213, 2019.

[13] R. Kumar and S. P. Khatri, "Crosstalk avoidance codes for 3D VLSI," in *Automation and Test in Europe*, 2013, pp. 1673–1678.

[14] 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, 2015.

[15] Y. Lou *et al.*, "Comparing through-silicon-via (TSV) void/pinhole defect self-test methods," *J. Electron. Test.*, vol. 28, no. 1, pp. 27–38, 2012.

[16] M. Tsai *et al.*, "Through silicon via (TSV) defect/pinhole self test circuit for 3D-IC," in *IEEE Int. Conf. on 3DIC.*, 2009, pp. 1–8.

[17] B. Noia *et al.*, "Pre-bond probing of TSVs in 3D stacked ICs," in *IEEE Int. Test Conf.*, 2011, pp. 1–10.

[18] R. W. Hamming, "Error detecting and error correcting codes," *Bell Labs Tech. J.*, vol. 29, no. 2, pp. 147–160, 1950.

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

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

[21] K. A. Bowman *et al.*, "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 49–63, 2009.

[22] P.-Y. Chen *et al.*, "On-chip TSV testing for 3D IC before bonding using sense amplification," in *Asian Test Symp.* IEEE, 2009, pp. 450–455.

[23] M.-Y. Hsiao, "A class of optimal minimum odd-weight-column SEC-DED codes," *IBM J. Res. Dev.*, vol. 14, no. 4, pp. 395–401, 1970.

[24] B. Fu and P. Ampadu, "On hamming product codes with type-ii hybrid ARQ for on-chip interconnects," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 9, pp. 2042–2054, 2009.

[25] A. B. Ahmed and A. B. Abdallah, "Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3D-NoC systems," *J. Parallel Distrib. Comput.*, vol. 93, pp. 30–43, 2016.

[26] K. N. Dang *et al.*, "Scalable design methodology and online algorithm for TSV-cluster defects recovery in highly reliable 3D-NoC systems," *IEEE Trans. Emerg. Topics Comput.*, in press.

[27] G. C. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2011, vol. 24.

[28] D. Gizopoulos *et al.*, "Architectures for online error detection and recovery in multicore processors," in *Design, Automation & Test in Europe Conference & Exhibition*. IEEE, 2011, pp. 1–6.

[29] L.-C. Li *et al.*, "An efficient 3D-IC on-chip test framework to embed TSV testing in memory BIST," in *20th Asia and South Pacific Design Automation Conference,*. IEEE, 2015, pp. 520–525.

[30] C. Grecu *et al.*, "Testing network-on-chip communication fabrics," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 12, p. 2201, 2007.

[31] A. M. Amory *et al.*, "A scalable test strategy for network-on-chip routers," in *Int. Conf. on Test*, 2005, pp. 9 pp.–599.

[32] D. Xiang *et al.*, "Multicast-Based Testing and Thermal-Aware Test Scheduling for 3D ICs with a Stacked Network-on-Chip," *IEEE Trans. Comput.*, vol. 65, no. 9, pp. 2767–2779, Sep. 2016.

[33] M. R. Kakoee, V. Bertacco, and L. Benini, "At-speed distributed functional testing to detect logic and delay faults in nocs," *IEEE Trans. Comput.*, vol. 63, no. 3, pp. 703–717, 2014.

[34] J. Kim *et al.*, "High-frequency scalable electrical model and analysis of a through silicon via (TSV)," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 1, no. 2, pp. 181–195, 2011.

[35] A. Prodromou *et al.*, "Nocalert: An on-line and real-time fault detection mechanism for network-on-chip architectures," in *IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 60–71.

[36] C. Liu *et al.*, "Reuse-based test access and integrated test scheduling for network-on-chip," in *Design, Automation and Test in Europe*, 2006, pp. 303–308.

[37] K. N. Dang and X. T. Tran, "Parity-based ECC and Mechanism for Detecting and Correcting Soft Errors in On-Chip Communication," in *IEEE Int. Symp. on Embedded Multicore/Many-core Syst. -on-Chip*, 2018.

[38] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003–1010, 1998.

[39] Y.-w. Lee, H. Lim, and S. Kang, "Grouping-based TSV test architecture for resistive open and bridge defects in 3-D-ICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1759–1763, 2017.

[40] C. Serafy and A. Srivastava, "Online TSV health monitoring and built-in self-repair to overcome aging," in *Int. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems*, 2013, pp. 224–229.

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

[42] K. Manna *et al.*, "Preemptive test scheduling for network-on-chip using particle swarm optimization," in *VLSI Design and Test*. Springer, 2013, pp. 74–82.

[43] L. Huang *et al.*, "Non-blocking testing for network-on-chip," *IEEE Trans. Comput.*, vol. 65, no. 3, pp. 679–692, 2016.

[44] T. Lehtonen *et al.*, "Online reconfigurable self-timed links for fault tolerant noc," *VLSI design*, vol. 2007, 2007.

[45] A. Ganguly *et al.*, "Crosstalk-aware channel coding schemes for energy efficient and reliable NOC interconnects," *IEEE Trans. VLSI Syst.*, vol. 17, no. 11, pp. 1626–1639, 2009.

[46] Q. Yu *et al.*, "Addressing network-on-chip router transient errors with inherent information redundancy," *ACM Trans. on Embedded Computing Syst.*, vol. 12, no. 4, p. 105, 2013.

[47] S. Shamshiri *et al.*, "End-to-end error correction and online diagnosis for on-chip networks," in *Int. Test Conf.*, 2011, pp. 1–10.

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

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

[50] A.-C. Hsieh and T. Hwang, "TSV redundancy: Architecture and design issues in 3-D IC," *IEEE Trans. VLSI Syst.*, vol. 20, no. 4, pp. 711–722, 2012.

[51] A. Dutta and N. A. Touba, "Multiple bit upset tolerant memory using a selective cycle avoidance based SEC-DED-DAEC code," in *25th IEEE VLSI Test Symp.* IEEE, 2007, pp. 349–354.

[52] L.-J. Saiz-Adalid *et al.*, "MCU tolerance in SRAMs through low-redundancy triple adjacent error correction," *IEEE Trans. VLSI Syst.*, vol. 23, no. 10, pp. 2332–2336, 2015.

[53] 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," *J. of Supercomputing*, vol. 73, no. 6, pp. 2705–2729, Jun 2017.

[54] 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.

[55] J. Zhang *et al.*, "Self-test method and recovery mechanism for high frequency TSV array," in *Int. Conf. on VLSI and Syst.-on-Chip*, 2011, pp. 260–265.

[56] I. Jani *et al.*, "BISTs for post-bond test and electrical analysis of high density 3D interconnect defects," in *European Test Symp.* IEEE, 2018, pp. 1–6.

[57] X.-T. Tran *et al.*, "Design-for-test approach of an asynchronous network-on-chip architecture and its associated test pattern generation and application," *IET comput. & digital techniques*, vol. 3, no. 5, pp. 487–500, 2009.

[58] J. Liu *et al.*, "Online traffic-aware fault detection for networks-on-chip," *J. Parallel Distrib. Comput.*, vol. 74, no. 1, pp. 1984–1993, 2014.

[59] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *IEEE/ACM Int. Symp. on Microarchitecture*, 2003, pp. 7–19.