TSV-OCT: A Scalable Online Multiple-TSV Defects Localization for Real-Time 3-D-IC Systems

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

Abstract—In order to detect and localize through-silicon-via (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 online testing. This article presents an error correction code (ECC)-based method named “TSV on-communication test” (TSV-OCT) to detect and localize faults without halting the operation of TSV-based 3-D-IC systems. We first propose a 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 $x2$ 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 $x4$ and $x5$ 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 3-D Network-on-Chip (NoC) router shows no performance degradation for testing while having a reasonable area overhead.

Index Terms—Error correction code (ECC), fault localization, fault tolerance, through-silicon via (TSV), product code.

I. INTRODUCTION

THROUGH-SILICON VIAS (TSVs) serve as vertical wires between two adjacent layers in 3-D ICs. Thanks to their extremely short lengths, their latencies are low, which could offer high speeds of communication [1]. Moreover, as a 3-D-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, Networks-on-Chip (NoC) have been widely considered as one of the most promising replacements for traditional communication paradigms (e.g., bus and point-to-point) with better scalability and parallelism [4]. By combining these two advanced technologies, 3-D 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 crosstalk 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 have 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 3-D ICs is more challenging than traditional 2-D 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 research studies have been focusing on recovery where there are several approaches such as hardware fault tolerance (i.e., correction circuits [20], redundancies [11], and reliability mapping [10]), information redundancy (i.e., coding techniques [13], [18], [23] or retransmission request [24]), and algorithm-based fault tolerance (i.e., fault-tolerant routing [4], [25], runtime 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 nonblocking solution helps preventing expensive consequences of operating systems under faults.

To maintain highly reliable real-time systems, fault detection and recovery are important tasks. Therefore, 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. As 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 test 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 online methods to detect and localize faults.

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 thousand or even millions of cycles. Meanwhile, P-BIST needs 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 a 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], upscaling 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 ECC’s efficiency.

5) Intermittent defect of TSV could happen. Since TSV is sensitive to thermal and stress issues, and 3-D 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. Fig. 1 depicts our motivation of using “OCT.” In normal P-BISTs, depicted as the periodic test, the response time is $\Delta_{\text{Task}} + \Delta_{\text{BIST}}$ ($\Delta_{\text{Task}} \leq \text{Period}$) where $\Delta_{\text{Task}}$ and $\Delta_{\text{BIST}}$ are the execution time of regular tasks and BIST, respectively. Because the testing time $\Delta_{\text{BIST}}$ requires thousands or even millions of cycles [29], [30], [36], the period of the test should be a larger value to reduce the performance degradation. Assuming the performance degradation is 10%, the period should be $10 \times \Delta_{\text{BIST}}$. 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.

Fig. 1(b) shows the case of using OCT, which is usually nonblocking (or with graceful degradation), the system can respond to the new fault after the test time $\Delta_{\text{OCT}}$. According to Fig. 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 a 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, and exceptions). ECCs/error detection codes (EDCs) could be classified as an OCT method.

Although OCT has a shorter response time, it has lower coverage than BIST. For instance, using single error correction-double error detection (SECDED) [23] 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 shown in Fig. 1(c). Because the first fault is missed by OCT, the system keeps running until another fault or a different behavior of the previous fault happens. Fig. 1(c) shows the case with the response time $\Delta_{\text{Task}}$, which is larger than P-BIST’s ($\Delta_{\text{Task}} + \Delta_{\text{BIST}}$).

In order to have a better response time to new faults, we use OCT to obtain an efficient tradeoff between the response time
and the system’s availability. To solve the low coverage of OCT, we propose the “TSV-OCT” methodology that includes a set of algorithms and architectures as follows.

1) A comprehensive OCT set of algorithms and architectures based on two phases to improve the coverage: statistical detection and isolation-and-check.

2) Statistical detection: the system localizes faults based on the fault behaviors from the ECC decoder of multiple transactions. In this article, we opt for the parity product code (PPC) adopted in our previous work [37] as the baseline. The Monte Carlo simulation shows that the statistical detection helps to localize 100% of two faults despite the limitation of one fault localization of PPC.

3) 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 reattaches these TSVs back to the encoding/decoding to ensure the faulty status. In our evaluation, we show that TSV-OCT can detect 100% of five defects cases.

The organization of this article 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 this article.

II. BACKGROUND

This section first introduces PPC—the baseline ECC for our TSV-OCT. Later, we present the existing works in testing and localizing TSV defects.

A. PPC

This part presents the baseline ECC: PPC that 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 $F_k$ is represented as follows:

$$F_k = \begin{bmatrix} b_{0,0} & b_{0,1} & b_{0,2} & \ldots & b_{0,N-1} & r_0 \\ b_{1,0} & b_{1,1} & b_{1,2} & \ldots & b_{1,N-1} & r_1 \\ \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\ b_{M-1,0} & b_{M-1,1} & b_{M-1,2} & \ldots & b_{M-1,N-1} & r_{M-1} \end{bmatrix}$$

where

$$r_i = b_{i,0} \oplus b_{i,1} \oplus \ldots \oplus b_{i,N-1}$$

$$c_j = b_{0,j} \oplus b_{1,j} \oplus \ldots \oplus b_{M-1,j}$$

$$u = \bigoplus_{i=0}^{N-1} b_{i,j}$$

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:

$$sr_i = b_{i,0} \oplus b_{i,1} \oplus \ldots \oplus b_{i,N-1} \oplus r_i$$

$$sc_j = b_{0,j} \oplus b_{1,j} \oplus \ldots \oplus b_{M-1,j} \oplus c_j$$

$$sc_N = r_0 \oplus r_1 \oplus \ldots \oplus r_{M-1} \oplus u$$

$$st_M = c_0 \oplus c_1 \oplus \ldots \oplus c_{N-1} \oplus u$$

where the bits $b_{i,j}$, $c_j$, $r_i$, and $u$ are taken from the corresponding TSVs. The outputs of (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

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

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

$$\hat{F}_k = \hat{F}_k \oplus m.$$

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

$$fr = \sum_{i=0}^{N+1} sr_i \quad fc = \sum_{i=0}^{M+1} sc_i$$

$$\text{Fault Detected} = (fr \geq 2) \lor (fc \geq 2).$$

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 (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 $(l, j)$ are flipped, both $cr_i$ and $sc_j$ are “0,” which makes the decoder fails to detect while both $cck$ and $sr_l$ could be “1.” This syndrome makes the decoder understand that there is one fault and correct the bit $b_{i,k}$.

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

B. Related Works

This 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, EDC/ECC [18], [23] 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 correctable/correctable defects.

The other approaches are to use testing circuits or BISTs. Zhao et al. [19], Cho et al. [20], and Chen et al. [22] present a more fine-grain method that could detect open defect using a simple circuit. Lee et al. [39] 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 a test pattern to the TSV and captured the output and detect open defects using a NAND gate with a logic threshold voltage. For lifetime monitoring, Serafy and Srivastava [40] presented a resistance tracking method and BIST to overcome

1We use the index $(a, b)$ to represent the $a$th row and $b$th column. Indexes start from zero.
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], and [42], testing is prescheduled in order to ensure the correctness. Lou et al. [15] and Tsai et al. [16] also presented other methods of TSV’s BIST for pin-hole and void defects. Li et al. [29] reuse 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) that 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 online testing, we adopt the classification in [28]: anomaly detection, P-BIST, dynamic verification, or redundant execution.

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 nonblocking testing for NoCs, which is a similar idea. In [30], testing for NoC fabrics, which can be used for 3-D 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 is presented by Prodromou et al. [35] with several invariants for online testing NoC and by Yu et al. [46] with another set for checking transient faults in NoC. Shamshiri et al. [47] with end-to-end monitoring. Anomaly detection (i.e., [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 and real-time detection).

Fig. 2 illustrates the different testing strategies. While the blocking test (i.e., P-BIST) depicted in the strategy in Fig. 2(b) needs to block the data traffic in order to send the test traffic, strategies in Fig. 2(c) and (d) schedule the test traffic to have less congestion. The strategy in Fig. 2(e) represents our OCT methods, where the test is performed together with the data transaction causing neither 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 3-D NoCs is discussed.

![Fig. 2. 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) OCT.](image)
Fig. 3. Illustration of a 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” and output “0.” Hidden defect TSV: input “0” and output “0.” (a) Zero hidden defect. (b) One hidden defect. (c) Two hidden defects—Case 1. (d) Two hidden defects—Case 2. (e) Two hidden defects—Case 3. (f) Three hidden defects. (g) Waveforms of statistical detector with 32 transactions.

Algorithm 1 Statistical Detector Using PPC

```
// Option for cautious/greedy localization
Input: Opt.
// Column Check (CC) and Row Check (RC)
Input: CC[N][M], RC[N][M]
// Threshold for localization
Input: Three_Loc
// Fault indexes
Output: Fault[N][M]

Fault[N][M] = 0;
for (i = 1; i <= N; i++)
for (j = 1; j <= M; j++)
    // Cautious localization
    if Opt. == 1 and \( \sum CC[i][j] == 1 \) and \( \sum RC[j][i] == 1 \) then
        Fault[N][M]++;
    // Greedy localization
    if Opt. == 2 and \( CC[i][j] == 1 \) and \( RC[j][i] == 1 \) then
        Fault[N][M]++;
for (i = 1; i <= N; i++)
for (j = 1; j <= M; j++)
    if Fault[i][j] == Three_Loc then
        Fault[i][j] = 1;
    else
        Fault[i][j] = 0;
```

1) **Cautious localization** (Opt. = 1): Only indicates the fault position when one fault is left. For instance, only the cases in Fig. 3(c)–(e) give faulty position.

2) **Greedy localization** (Opt. = 2): As long as the row and column checks fail, it determines the position with the corresponding indexes as faulty. For instance, with the syndrome depicted in Fig. 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.

Fig. 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 that could happen.

1) **Zero hidden defect** [Fig. 3(a)]: Because all three defects cause flipped bits, the detector fails to correct.
2) **One hidden defect** [Fig. 3(b)]: Because two defects cause flipped bits, the detector fails to correct but it could alert the system.
3) **Two hidden defects** [Fig. 3(c)–(e)]: The detector succeeds to localize one fault position.
4) **Three hidden defects** [Fig. 3(f)]: The system cannot be alerted because of hidden errors.

The statistical detector with Opt. = 1 (cautious localization) only catches the “two hidden errors” cases [Fig. 3(d)–(f)] for localization. As shown in Fig. 3(g), the system uses \( T = 32 \) transmissions for the statistical detector with 32 data values (D0, D1, …, D31). The cases in Fig. 3(c), (e), and (f) are hit with D7, D20, and D25, respectively; 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 article, we opted to use the greedy version (Opt. = 2) because the false-positive case is not a critical issue. As shown in Fig. 3(g), the Greedy localization option tries to cover as much as possible the faulty positions. When one hidden defect [Fig. 3(b)] is hit, it indicates four positions (2, 0), (2, 4), (3, 0), and (3, 4) as faulty. Those false positives could be removed by using the isolation-and-check algorithm in Section 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).
Algorithm 2 Isolation-and-Check Algorithm

// Column Check (CC) and Row Check (RC)
Input: CC(1:N), RC(1:M)
// Threshold for Localization
Input: ThreeLoc
// Fault index
Output: Fault(1:N)(1:M)

1. Isolation(1:N)(1:M) = 0;
2. Fault(1:N)(1:M) = Statistical_Detector(CC, RC, ThreeLoc);
3. Isolate fault and recheck the second time
4. Isolation(1:N)(1:M) = Fault(1:N)(1:M); // Un-isolate each position and recheck the second time

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
10. Fault[i][j] = 0;
11. else
12. Isolation[i][j] = 1;

Fig. 4 shows the pattern when it may miss and incorrectly indicate position. Due to the transmitted data values, only the cases in Fig. 4(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 the statistical detector. By improving the detection or localization rate, the system can eliminate the need for 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 reusing PPC. The proposed algorithm follows these steps:

Step 1) Using the 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 rechecked and corrected later.

Step 2) The system virtually isolates the suspicious TSVs from the encoding/decoding process; however, they are still being used for data transaction. In other words, suspicious TSVs are removed from the parity bit functions in (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 positions if needed.

Step 3) Rerun Steps 1)–3) until no fault is detected or out of time (until deadline).

Step 4) Reassign each isolated TSV. The TSV could be reattached to the encoding and decoding processes. 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 rerunning the statistical detector, the system can localize more faults. Let us consider the case in Fig. 4, after using the statistical detector once, two TSVs (0, 1) and (2, 3) are removed from the decoding and encoding, as shown in Fig. 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 [Fig. 4(b)] is hit again, as shown in Fig. 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 zero hidden defects [Fig. 4(a)] can indicate the last defect at (0, 3) [see Fig. 5(d) and (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 Fig. 6. At the end of Step 3) (see Fig. 5), four positions are indicated as suspicious. In Steps 4) and 5), the algorithm reenables each suspicious TSV to confirm its correctness. The algorithm first enables the TSV (0, 1) and
performs data transactions. Because this TSV is faulty and $D_{11}$ in Fig. 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 reenabled, no faulty output is found. The system can conclude it as nonfaulty 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 $2 \times \text{worst case execution time (WCET)}$. 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 3-D NoC

In order to understand the cost of the design, we integrate the TSV-OCT into our previously designed 3-D-NoC router [26], as shown in Fig. 7. Note that the proposed approach is totally independent of our opted router architecture and could be implemented into any TSV-based architecture. The PPC is integrated as an ECC module and the 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 × 8) that requires 45 codeword-bit. This leads to only one extra TSV to be added (44 + 1 into 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.

IV. EVALUATION

A. Evaluation Methodology

The proposed system was designed in Verilog-HDL, synthesized, and prototyped with commercial CAD tools. We use the NANGATE 45-nm 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 the 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 are 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 times 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

Fig. 8 shows the performance results when only using the statistical detector (without isolation and check), including false-positive cases. We can easily observe that the localization rate of a 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 detector localizes at least 45% of six faults, 99% of three faults, and 100% of two faults. Therefore, the statistical detector has significantly improved the localization rate of the ECC’s bound (naturally localizing one fault and detecting two faults).

Even without the false-positive cases, the localization rate of the statistical detector easily outperforms the baseline ECC (PPC), as depicted in Fig. 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.

C. Isolation-and-Check Performance

This section evaluates the isolation and check in two parts: Steps 1–3) and Steps 4) and 5).

1) Steps 1–3): At first, we evaluate Steps 1–3) of the isolation-and-check method to see how much these three steps can improve the defect detection when compared to the
As shown in Fig. 10, we can observe that $T = 128$ can cover most of less than six faults (including false positive). The lower $T$ values give less improvement than 128; however, in comparison to the statistical detector, the isolation-and-check method gives significant improvements ($\approx 2 \times$ localization rate).

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

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

As shown in Fig. 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 in using the 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% and 5% [39]. Here, we assume the defect rate is 5%, where 8, 16, 32, and 64 data bits have about 1, 2, 3, and 5 defects, respectively. As shown in Fig. 12, our method can cover five faults in all cases with $T = 128$, which is enough to 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 nonlocalized defects, as shown in (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 times 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). Figs. 13–15 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 a minimum response time, as illustrated in Fig. 13, smaller $T$ values give much faster response time. 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 significantly dropped. We can also notice the nonexisting cases: $[T = 8 \text{ 32-bit 10-fault}]$, $[T = 8 \text{ 64-bit 9-fault}]$, $[T = 8 \text{ 64-bit 10-fault}]$, and $[T = 16 \text{ 64-bit 10-fault}]$, where the system fails to correctly localize any case in the Monte Carlo simulation.

For the case of the maximum response time, depicted in Fig. 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.

Fig. 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 times), the expected response time of higher values of $T$ is worse than smaller ones. However, we also can observe the case where smaller $T$ values give a 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 relationship between the response time with $T$. With a smaller number of defects, this could be a critical issue. However, with a greater number of defects, the tradeoff between the coverage rate and the 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 a similar fault coverage as $T = 16$. Note that diving TSV into clusters does not increase the testing time. Meanwhile, P-BIST encounters scalability issues and blocking issues for large-scale testing, which may significantly impact the performance. Wang et al. [12] pointed out that their lower bound of testing period of a 10×8 mesh NoC is 16,840 cycles.

### E. Hardware Implementation

Detailed results of 32-data-bit implementations are given 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 [single-error correction, double-adjacent error correction (SEC-DAEC) and triple-adjacent error correction (TAEC)]. However, this article 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 and TAEC [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 a smaller design than multiple-error 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 (4×8) only calculates the parity for 8 bits instead of 32 bits. 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 (4×8), the proposed architecture increases 1.70× and 11.54× the encoder and decoder area costs, respectively. Here, both of the encoder and decoder areas of our method have larger area cost due to the need for 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× power. Although the decoder requires high area and power overhead, these results are expected because of the added extra modules and computation.

Table III gives the result of implementing TSV-OCT into a 3-D NoC. 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 that in [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 having higher area costs, as represented in Table II, the overhead of the proposal inside a given 3-D-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 the 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 $[4 \times 8, T = 64]$ offers the smallest testing time among all other on-chip communication testing and is 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. However, we have to note 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], and [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 that could work online without any degradation while these circuits must interrupt the connection to test.

Table V also shows a comparison for online 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. Note that the results of other works do not include the BIST area and power consumption. Also, TSV-OCT does not require adaptive routing to perform testing because it is an OCT (nonblocking testing).

As can be observed in Table V, our technique offers a smaller area cost than that in [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, respectively.

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). Note that the upper bound of our technique is 16,448 cycles, which is under the lower bound of that in [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 article has been evaluated and compared in the detection and localization efficiencies, the impact of real-chip fabrication and process–voltage–temperature (PVT) variations has not been studied. The PVT or real-chip measurement could provide more realistic results on the efficiency of our algorithm. Nevertheless, PVT or real-chip measurement should be studied in the future to provide a better understanding of our proposal.

Testing methods for TSV-to-TSV bridge defects [39] have not been evaluated in our article. The performance of TSV-OCT on detecting and localizing this type of 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-mentioned limitations, TSV-OCT still provides extra defects’ localization while maintaining short execution time. The exhibited overhead in a 3-D-NoC implementation is also reasonable, which makes TSV-OCT totally feasible for integration into highly reliable 3-D ICs.

V. CONCLUSION

This article presented a method to improve the localization rate of PPC to enhance the reliability of TSV-based 3-D-IC designs. From the conducted experiments, and in contrast to the baseline PPC that is limited to localizing one fault at most, TSV-OCT has demonstrated its ability to localize more than six faults. Furthermore, TSV-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 tolerances 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

The authors would like to thank the reviewers and editors for their excellent comments that helped improve this article.

REFERENCES


