Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A non-blocking non-degrading multiple defects link testing method for 3D-Networks-on-Chip

KHANH N. DANG[1], MICHAEL CONRAD MEYER[2], AKRAM BEN AHMED[3], (MEMBER, IEEE), ABDERAZEK BEN ABDALLAH[4] AND XUAN-TU TRAN[1], (SENIOR MEMBER, IEEE)

[1]VNU Key Laboratory for Smart Integrated Systems (SISLAB), VNU University of Engineering and Technology (VNU-UET), Vietnam National University, Hanoi (VNU), Hanoi 123106, Vietnam
[2]G.S. of Information, Production and Systems, Waseda University, Kitakyushu 808-0135, Japan
[3]National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, 305-8568, Japan
[4]Adaptive Systems Laboratory, The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan.

Corresponding authors: Khanh N. Dang (e-mail: khanh.n.dang@vnu.edu.vn) and Xuan-Tu Tran (e-mail: tutx@vnu.edu.vn).

This research is partly funded by Vietnam National University, Hanoi (VNU) under grant number QG.18.38.

ABSTRACT As one of the most promising technologies to realize 3D Integrated Circuits (3D-ICs), Through-Silicon-Via (TSV) acts as the inter-layer link inside 3D Networks-on-Chip. However, the reliability issues due to the low yield rates and the sensitivity to thermal hotspots and stress issues are preventing TSV-based 3D-ICs from being widely and efficiently used. To ensure the correctness of TSV connections at run-time, detecting multiple (clustering) defects is an important feature. While Error Correction Codes are limited by a certain number of detectable faults, using Built-In-Self-Test (BIST) prevents the system from operating normally during the test time. This paper first presents a Parity Product Code (PPC) with the ability to correct one fault and detect, at least, two faults. Second, we present extended PPC (EPPC) to detect multiple defects within the links of Networks-on-Chip by using two or more additional matrices. Furthermore, we present the distance-aware version of EPPC to detect multiple defects by using only one extra matrix. The results show that the distance-aware EPPC can detect 100% of clustering defects and multiple random defects within two and three cycles, respectively. The performance evaluation for Network-on-Chip testing also shows no degradation while providing an extremely short response time (2-3 cycles).

INDEX TERMS 3D-ICs, Fault-Tolerance, Error Correction Code, Through-Silicon-Via, Product Code, Parity Check, Networks-on-Chip, Fault-Detection

I. INTRODUCTION

As a result of the fusion of 3D-Integrated Circuits (3D-ICs) and the mesh-based Network-on-Chips (NoCs), the 3D-Network-on-Chip (3D-NoC) paradigm [1] is considered to be one of the most promising architectures for IC design by providing a highly scalable multi/many-core infrastructure. Through-Silicon-Vias (TSVs) serve as vertical wires between two adjacent layers in 3D-ICs and 3D-NoCs. Because TSVs have extremely short lengths, their latencies are low which could offer extremely high communication speeds [2], [3]. Moreover, as a 3D-IC technology, TSV-based ICs can have smaller footprints despite the TSV's overheads [4], and lower power consumption thanks to the shorter wires [5].

Despite having multiple advantages, one of the major concerns of TSVs is reliability due to their low yield rates [6], [7]. In [7], the results show that the defect rate of TSVs is nearly 0.63%. Moreover, TSVs are also vulnerable to thermal and stress [8], [9]. Because of the natural parallel structure, TSVs also face the crosstalk challenge [10], [11]. Furthermore, the difference in thermal expansion coefficients of materials and temperature variations between two layers, which has been reported to reach up to 10°C [12], could lead to stress issues. In summary, 3D-ICs must adequately consider the reliability of TSVs.

To recover from defective TSVs, there are three main approaches: *(1)* hardware fault-tolerance such as correction circuits [13], redundancies [14], reliability mapping [9]; *(2)* information redundancy such as coding techniques [10], [15], [16] or re-transmission request [17]; or *(3)* algorithm-based

fault-tolerance [1], [18], [19]. Most of the recent works have been focusing on how to recover from defective TSVs in both random and clustered distributions. However, prior to recovery, the system must know the position of defective group of TSVs.

In order to detect defects in TSVs, there are two major approaches: (1) *online* and (2) *offline testing*. While *online testing* allows the system to operate during the test, *offline testing* might partly or totally interrupt the system operation. There are several methods for offline testing such as Built-in-self-test (BIST) [20], [21] and external testing [22], [23] techniques. For online testing, simple coding techniques such as Parity, Hamming [15] or SECDED [16] (Single Error Correction, Double Error Detection) or other coding methods such as Reed-Solomon or Bose–Chaudhuri–Hocquenghem can be used to detect defects during transmission. To correct multiple defects, Orthogonal Latin Square Code (OLSC) [24] which provides low latency and modular design can be used. However, OLSC does not provide extra detectability. Despite having multiple methods for detection and correct faults in wires/links, there are two main problems as follows:

- Built-in-Self-Test is costly in terms of power, area and execution time. Most works require partial or total preemption in order to test. Works for online NoC-testing in [25]–[28] perform with lower-bound of test periods around 16,000 to 50,000 while still degrading the system performance.
- Online testing such as Error Correction Code can provide immediate results; however, it is limited by a certain number of detectable faults. To ensure the reliability of the system, we must know when multiple defects occur.

Therefore, in this paper, we propose a new coding method named Extended Parity Product Code (EPPC) which is specially designed for correcting one defect and detecting multiple defects in TSV links. This work is based on our previous work in [29][1] where we used different Orthogonal Latin Square Matrices as alternative matrices of computation. However, the work in [29] has not taken into account the clustering fault model [14]. The new contributions of this paper are as follows:

- The complexity analyses for area and delay function of the encoding and decoding of Parity Product Code processes are presented. Here, the delay complexity is only $O(log_2(\sqrt{n}))$ while it is $O(log_2(n))$ for Hamming and SECDED (n is the input's bit-width).
- An Extended PPC (EPPC) to break the limitation of two-fault detection by using the *matrix-switch* method. The *matrix-switch* is based on a shifting matrix to break the undetectable pattern of PPC instead of multiple Orthogonal Latin Square matrices. The proposal can detect multiple defects within 2-3 cycles.
- Based on the clustering fault model, we present the distance-aware EPPC to reduce the needed additional matrices. The results show that we need only two cycles to tackle this model of fault.
- An analysis to point out the optimal value of s (shifting value) to obtain the best extension in distance-aware EPPC.
- Network-on-Chip integration and evaluation shows that the PPC can provide low latency multiple-defect testing without blocking the communication. When a defective link is detected, a fault-tolerant routing algorithm [18] is adopted for recovery.

The organization of this paper is as follows: Section II reviews the existing literature on detecting TSV defects. Section III shows the Network-on-Chip infrastructure to support testing and Section IV presents the baseline PPC. Section V shows the extended PPC to provide the ability to detect multiple defects. Section VI provides the evaluation environment and results. Finally, Section VII concludes the paper.

II. RELATED WORKS

As previously mentioned, we can classify the TSV fault detection into two main approaches: online and offline testing. For offline testing methods, we can further distinguish them as pre-bond and post-bond testing. For online testing methods, there are two major approaches: blocking and non-blocking tests. This section aims to briefly discuss the approaches for TSV testing.

In order to realize *offline testing*, one of the prerequisites is having additional circuits that perform the test. For *pre-bond testing*, since the tested TSV is not fully connected to two layers, most works focus on exploiting the characteristic of the open wire to assess its quality. In [30], the authors performed short-to-circuit test by classifying the voltage of a tested TSV after supplying a Vdd voltage using a voltage comparator. This method can help detect short-to circuit defects and provide a certain correction using a voltage comparator. However, the cost of additional circuits for each TSV is considerable. *Noia et al.* [22] use a probe needle to measure the output value of a group of TSVs to indicate the failure status of them. Although this method is extremely efficient, the method of measurement requires extremely accurate devices and calibration. In order to solve this problem, work in [31] proposed a contact-less method using ring oscillators. The results show that they can indicate both open and short-to-substrate defects.

After bonding two or more layers together, the connected TSVs could be tested as a normal wires in *post-bond testing*. In [20], [21], the authors presented methods of TSV BIST for pin-hole and void defects. In [30], besides pre-bond testing, the authors also presented a method for post-bond testing by reusing the existing circuit of pre-bond testing. As a conventional testing method, *Huang et al.* [32] presented a scan chain for testing TSVs after bonding. TSVs are organized by a two dimensional shape and are accessed by column and

[1] [29]: Khanh N. Dang, Michael Meyer, Akram Ben Ahmed, Abderazek Ben Abdallah, and Xuan-Tu Tran,"2D-PPC: A single-correction multiple-detection method for Through-Silicon-Via Fault", 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), Nov. 11-14, 2019

row selections. In [33], the authors presented a method to group TSVs in a two dimensional shape in order to reduce the test time. They also provided the ability to test not only open and short defects but also perform bridge TSV-to-TSV defects which could be critical.

One of the most basic methods for *online testing* is to use Error Correction/Detection Code. In other words, to be able to detect and correct faults, a code-word with redundant bits is used instead of the original data in the channels. Hamming code [15], which can detect and correct one faulty bit, is one of the most important coding techniques. SECDED by *Hisao* [16] is also extremely useful with the help of the HARQ (Hybrid-Automatic Retransmission Request) mechanism. SECDED can detect two faults in a flit which could be retransmitted for further correction as HARQ. In [34] and [35], the authors presented several variations of Hamming code using specified matrices which can correct two or even three adjacent faulty bits. Thanks to their simple XOR functions, these codes are definitely simple and suitable for high-speed circuits; however, they have a limited number of detectable faults. On the other hand, to tackle the crosstalk effect, Crosstalk Avoidance Code could be used [10]. Since using a dedicated coding technique seems inflexible, using adaptive coding could be a suitable solution. In [17], packets are structured in 2D arrays and a Hamming code is used to correct a flit (column). When the decoder fails to correct the flit because of extra faulty bits, extra hamming codes for each index (row) are used; therefore, the system can further correct faulty bits. Also, there are several powerful block coding methods such as Reed-Solomon [36] or Bose-Chaudhuri-Hocquenghem code [37] to help handle more faults. However, their calculations are too complex which could lead to a significant amount of area and power consumption.

Since TSVs are used in certain applications such as stacked memory, wires in block design or on-chip communication, they can also be tested using the existing methods for *online testing*. Here, we focus on on-chip communication using TSVs. *Li et al.* [38] test TSVs "for free" by analyzing the memory test which requires inter-layer connections. In [39], an on-line TSV testing and recovery method was presented using a random number generator and a NAND gate with a threshold voltage input to capture the open/short defect. Apparently, the method requires interrupting the connection in order to start the test. Some other methods to test TSVs in a Network-on-Chip use existing techniques of wire testing [25]–[28]. With smart scheduling of these methods, we can even obtain uninterrupted testing that allows the system to maintain its operation during test. However, reducing the test period (time between two consecutive tests) can significantly reduce the system performance. In [40], [41], the authors proposed a method to test TSVs without any interruption or degradation of the system performance. The major drawback of these non-degradation tests is the long execution time. Also, the existing channel NoC testing methods are not perfect since their coverage may not be 100%. The notable works on channel NoCs testing [42]–[47] failed to have the highest coverage. The work in [48] obtains a 100% coverage; however, their testing time might not scale well.

In summary, we can observe that offline testings using circuit in pre-bond and post-bond methods are efficient to detect the defects of TSVs. However, having an offline time to test is not always feasible in real-time and safety-critical applications. On the other hand, online testing can help test the TSV during operation; however the existing methods either suffer from low coverage rate, performance degradation or long testing time. In this work, we aim to propose a low cost, fast execution testing method for TSVs while maintaining a high coverage in order to solve the aforementioned problems.

III. NETWORK-ON-CHIP ARCHITECTURE

We integrate the proposed method into our previously designed 3D-NoC router [19], as shown in Figure 1. Please note that the proposed approach is totally independent from our adopted router architecture and could be implemented into any TSV-based architecture or normal link-based system. The illustrated system uses 3D-Mesh topology where each router can connect at most six nearby routers and a local processing element as in Figure 1(a). TSVs are the interconnects between layers. The data (flit) arriving at an input port is buffered then routed and goes through one of the output ports.

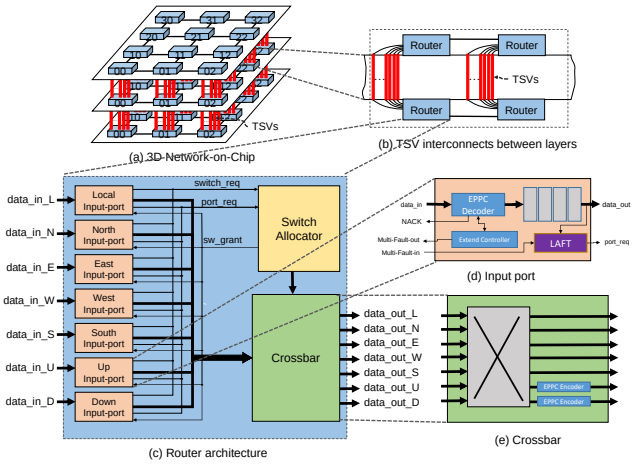

Figure 1: 3-D Mesh Network-on-Chip router architecture.

The EPPC is integrated as an ECC module in the input port and the *matrix-switch* feature is only integrated into two vertical ports (UP and DOWN) to monitor and detect faults of TSVs. We could also provide checking for intra-layer links by adding the *matrix-switch*. The data from the TSV is brought to the *EPPC decoder* to check. The output (detection) signal is collectively received and analyzed by the *Extend Controller*. The output multi-faults detection is updated to the nearby routers to use a fault-tolerant routing algorithm (LAFT) to avoid faulty connections [18]. When sending out the data, we also use a matrix-switch with 2-bit counter and a normal decoder. Note that the counter must be

synchronized between two layers. By using adaptive routing, we can maintain the reliability of the system even under a high number of defects. We would like to note that there are some vulnerable links inside the network such as the link between Network Interface and router where a failure on one of these link immediately corrupt the system reliability [49]. However, since we mostly consider the inter-layer links (using TSVs) in this work, LAFT routing algorithm can efficiently work without considering the vulnerability of those links.

In this implementation, we assume that the synchronization between routers could be done by a reliable channel. The fault information should be synchronized throughout the network in a broadcast manner.

IV. PARITY PRODUCT CODE

This section presents the proposed Parity Product Code (PPC). It is based on the Product-Code [50]–[52] approach and exploits the natural 2D array placement of TSVs. We first present the considered fault types then TSV organization is discussed. The following parts demonstrate the encoding and decoding processes with equivalent circuits. The analytical analyses for encoding and decoding processes are also presented.

A. FAULT CONSIDERATION

In this work, we mainly consider permanent defects. There are two types of permanent defects: manufacturing defects and operating defects. Due to imperfections during the manufacturing process, permanent TSV defects are more frequent than other types of faults. TSV defects are usually leakage (short), open (void), or bridge type [20], [53]. A TSV could be shortened to ground or Vdd which causes 0/1 stuck-at faults. A bridge defect between two or more TSVs prevents them from transmitting different values at the same time. An open defect on a TSV increases its resistance which electrically disconnects its terminals or causes a transition delay. Aging, process variation or even temperature variation which cause stress issues could further increase the fault probabilities. Besides manufacturing defects, operating defects are also a considerable issue of TSV-based 3D-ICs. Due to the high temperature of 3D-ICs, other fault factors such as Electro-Migration, Time-Dependent-Dielectric-Breakdown, etc. are accelerated. Thermal Cycling is also another fault source due to the high difference in temperature between layers.

There are two basic models of fault distribution: random and clustered. While random defects are uniformly distributed across the 2D layer, clustered defects are focused within a certain area (a cluster). Previously, either Poisson [54] or center-satellite model [55] were used for this type of distribution. Here, to model them, we adopt the model from [14] as follow:

$$p_i = F_b \sum_{j=1}^{N_c} (\frac{1}{d_{ic}})^{\alpha} \quad (1)$$

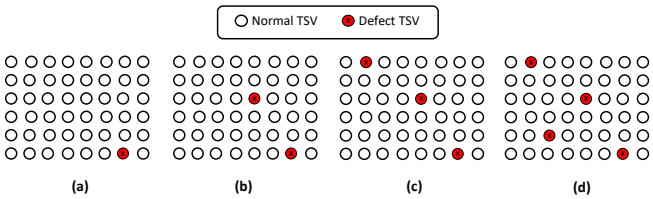

Figure 2: An illustration of random defective TSV: (a) one defective TSV; (b) two defective TSVs; (c) three defective TSVs; (d) four defective TSVs and (e) five defective TSVs.

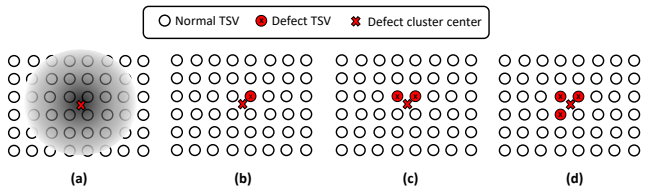

Figure 3: An illustration of clustering defect following Equation 1: (a) fault model in Equation 1 - darker area has higher defect rate; (b) one defective TSV; (c) two defective TSVs; (d) three defective TSVs.

where p_i is the defect probability of a TSV i, d_{ic} is the Euclidean distance between a TSV i and an existing cluster defect center, F_b is the fault rate of the defected cluster's center, N_c is the number of the defected cluster's centers and α is the clustering effect. To perform our evaluation, we set the $F_b = 1$ to guarantee faults at the defected cluster's center. For the nearby TSV i, we select according to the probability p_i.

For the ease of understanding, we illustrated in Figures 2 and 3 the examples of random and clustering defects, respectively. With random defects, the position of defective TSVs are totally unconstrained. Depending on different scenarios, the fault patterns are different. In our evaluation section (see Section VI), we use Monte-Carlo simulation with 10,000 cases per configuration to investigate different defect patterns. For clustering defects, by following Equation 1, the TSVs near the center of the defected cluster have a higher chance of being defective, as well. Therefore, in our evaluation, these TSVs are usually injected as defective ones. The position of the center is randomized in the group of TSVs.

Regarding behavior, we could model the possible faults as stuck-at faults. For instance, the output logic value of a TSV is stuck to '0' or '1'. These behaviors are generally applied to soft errors as single event upset. Therefore, if soft errors occur for several cycles or an intermittent fault occurs, EPPC could detect them with its coding method. The permanent defects could be physically modeled as RC models where the open and short resistances play important roles in their operations [39]. Delays caused by crosstalk and permanent faults could violate the timing constraints leading to sample the old values or metastability phenomenon could occur. This behavior is extremely hazardous for digital circuits and needs to be addressed appropriately using dedicated circuits [13],

[56]. A bridged TSV-to-TSV defect can also be modeled as a considerably small resistance between two or more TSVs shorted together [33]. Once these defective TSVs are shorted, their output voltage could be either at '0', '1' or metastability. Here, we assume a metastability immune circuit or voltage comparators [13], [56] are used to push the output to either '0' or '1'. Once the output of TSVs are binary, our coding method could help detect faults.

In this work, we use the "inverse" fault (flip-bit): where the value of a TSV is reversed during operation. The "inverse" model can be used to model open or short-to-substrate defects after the metastability. With bridge TSV-to-TSV, depending on which value we get at the output of TSVs, their values could be flipped or not. By adding the distribution in Equation 1, we can model the bridge defects. For example, Figures 3(c-d) show the fault patterns of clustering defects. Because the TSVs near the center are injected as faulty, we can easily observe a high chance of two or more nearby TSVs being faulty at the same time. Consequently, this could be modeled as a bridge defect of these TSVs. However, we would like to note that this fault pattern is a square-like shape and might not be able to model the bridge defect of TSV in different shapes (i.e. row, column, or diagonal line).

B. TSV ORGANIZATION

Assuming that a group of TSVs is organized in a 2D array of $M \times N$ (M rows and N columns). Originally, a set of TSVs is organized as follows:

$$\text{TSVs} = \begin{bmatrix} T_{0,0} & T_{0,1} & \dots & T_{0,N-1} \\ T_{1,0} & T_{1,1} & \dots & T_{1,N-1} \\ \dots & \dots & \dots & \dots \\ T_{M-1,0} & T_{M-1,1} & \dots & T_{M-1,N-1} \end{bmatrix} \quad (2)$$

where $T_{i,j}$ represents the TSV in the i^{th} row and the j^{th} column. As a product-code, for each row i and column j, we add an array of *row* parity-bits TSVs (CR_i) and an array of *column* parity-bits TSVs (CC_j). Then, there is an extra TSV CU for the *ultimate* check bits. The coded TSVs are as follows:

$$\text{Coded\_TSVs} = \begin{bmatrix} T_{0,0} & \dots & T_{0,N-1} & CR_0 \\ T_{1,0} & \dots & T_{1,N-1} & CR_1 \\ \dots & \dots & \dots & \dots \\ T_{M-1,0} & \dots & T_{M-1,N-1} & CR_{M-1} \\ CC_0 & \dots & CC_{N-1} & CU \end{bmatrix} \quad (3)$$

Even when a group of TSVs is not organized as a two dimensional array, we still can manage its data in a 2D array to apply the proposed technique. For instance, a group of 15 TSVs can be considered as a 4×4 group with one dummy value.

C. ENCODING

For each transmission, a TSV $T_{i,j}$ sends a bit $b_{i,j}$, CR_i sends a row-parity bit r_i, CC_j sends a column-parity bit c_j and CU sends an ultimate-parity bit u which is a member of a coded flit F:

$$F_k = \begin{bmatrix} b_{0,0} & b_{0,1} & \dots & b_{0,N-1} & r_0 \\ b_{1,0} & b_{1,1} & \dots & b_{1,N-1} & r_1 \\ \dots & \dots & \dots & \dots & \dots \\ b_{M-1,0} & b_{M-1,1} & \dots & b_{M,N-1} & r_{M-1} \\ c_0 & c_1 & \dots & c_{N-1} & u \end{bmatrix} \quad (4)$$

where

$$\begin{aligned} r_i &= b_{i,0} \oplus b_{i,1} \oplus \dots \oplus b_{i,N-1} \\ c_j &= b_{0,j} \oplus b_{1,j} \oplus \dots \oplus b_{M-1,j} \\ ur &= r_0 \oplus r_1 \oplus \dots \oplus r_{M-1} \\ uc &= c_0 \oplus c_1 \oplus \dots \oplus c_{N-1} \\ u &= ur = uc = \oplus_{i=0}^{N-1} \oplus_{j=0}^{M-1} (b_{i,j}) \end{aligned} \quad (5)$$

Note that the symbol $\oplus$ stands for the XOR function.

The architecture of PPC encoders is shown in Figure 4 where the *Row Encoder*, *Col. Encoder*, and *Ulti. Encoder* are for encoding the rows, columns and ultimate bits, respectively. These encoders share the same parity encoder architecture (known as XOR-tree), as shown in Figure 5. If designers do not desire to detect faults on the encoder, this signal can be simply removed to reduce the area cost (one XOR-tree and one XOR gate).

The expected number of gates ($\mathbf{G}$) and expected delay (τ) of the encoding process is shown in Equation 8 and Equation 10, respectively. The proofs of these equations are presented as Lemma IV.2 and Lemma IV.3.

From Equation 8 and Equation 10, with a given n bit ($M = N = \sqrt{n}$), the area cost and delay complexity are $O(n)$ and $O(log_2(\sqrt{n}))$, respectively. In comparison, Hamming's and SECDED's area cost and delay complexities are $O(n)$ and $O(log_2(n))$, respectively. This means that PPC provides better scalability in terms of delay.

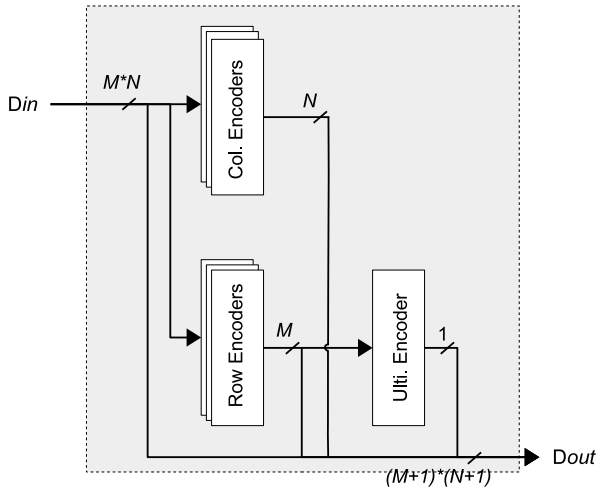

Figure 4: PPC Encoder Architecture.

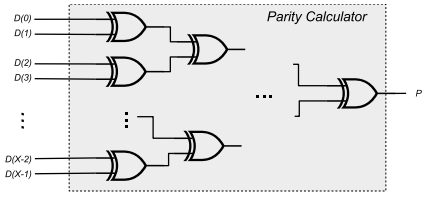

Figure 5: Parity architecture using XOR tree.

Lemma IV.1. *Given an* O_2X1*-tree (*O_AX1 *is a gate with A inputs) of Z inputs, the number of gates (*$\mathbf{G}$*) and delay (*τ*) are:*

$$\begin{aligned} \mathbf{G}_{\mathsf{O\_2X1}} &= Z - 1 \\ \tau_{\mathsf{output}} &= ceil(log_2(Z)) \times (\tau_{\mathsf{O\_2X1}}) \end{aligned} \tag{6}$$

Proof. In order to build an O_2X1-tree, the inputs of the tree are fed to the first layer of O_2X1. The outputs of a layer are fed to the inputs of the next layer until reaching a single output. The delay of each layer will be the delay of an O_2X1 gate ($\tau_{\mathsf{O\_2X1}}$) regardless of the wire delay. Because at each layer, the number of values is divided by two and rounded up (i.e., 4 inputs get 2 outputs, 3 inputs get 2 outputs), the number of layers is: ceil($\log_2(Z)$). Here, we assume the O_4X1 and O_3X1 (gates with four and three inputs) are constructed from two layers of the O_2X1 gate.

The delay to the final output τ_{output} will be the number of layers (ceil($\log_2(Z)$)) multiplied by the delay of each layer ($\tau_{\mathsf{O\_2X1}}$). Therefore:

$$\tau_{\mathsf{output}} = \text{ceil}(\log_2(Z)) \times (\tau_{\mathsf{O\_2X1}}) \tag{7}$$

For each O_2X1-gate, two inputs will always give one output. Regardless of the O_2X1-tree structure, in order to obtain a single output from Z inputs, the number of inputs needs to be omitted is $Z - 1$. In other words, the number of gates is $Z - 1$.

□

Lemma IV.2. *The expected number of gates (*$\mathbf{G}$*) of the encoder is:*

$$\mathbf{G}^{encoder}_{\mathsf{XOR\_2X1}} = 2MN - 2 \tag{8}$$

Proof. For the row and column encoding, there are two branches:

1) Row encoder: the inputs are M rows of N bits. Therefore, the number of XOR gates and the delay are $M \times (N-1)$ and ceil($\log_2(N)$) $\times$ ($\tau_{\mathsf{XOR\_2X1}}$), respectively.
2) Column encoder: the inputs are N columns of M bits. Therefore, the number of XOR gates and the delay are $N \times (M-1)$ and ceil($\log_2(M)$) $\times$ ($\tau_{\mathsf{XOR\_2X1}}$), respectively.

For the ultimate bit encoding, there are also two branches:

1) Encode from row-parity-bit: the inputs are M bits. Therefore, the number of XOR gate and the delay are $M-1$ and ceil($\log_2(M)$) $\times$ ($\tau_{\mathsf{XOR\_2X1}}$), respectively.
2) Encode from column-parity-bit: the inputs are N bits. Therefore, the number of XOR gate and the delay are $N-1$ and ceil($\log_2(N)$) $\times$ ($\tau_{\mathsf{XOR\_2X1}}$), respectively.

$$\begin{aligned} \mathbf{G}^{\text{encoder}}_{\mathsf{XOR\_2X1}} &= M \times (N-1) + N \times (M-1) + (M-1) + \\ &\quad (N-1) \\ \mathbf{G}^{\text{encoder}}_{\mathsf{XOR\_2X1}} &= 2MN - 2 \end{aligned} \tag{9}$$

□

Lemma IV.3. *The expected delay (*τ*) of the encoding process is:*

$$\tau^{encoder}_{\mathsf{output}} = \begin{cases} \tau_{\mathsf{XOR\_2X1}} \times (ceil(log_2(max(M,N))) \\ \qquad +ceil(log_2(M)))) \; if \; u = ur \\ \tau_{\mathsf{XOR\_2X1}} \times (ceil(log_2(max(M,N))) \\ \qquad +ceil(log_2(N)))) \; if \; u = uc \end{cases} \tag{10}$$

Proof. The number of layers of the column and row parity-bit encoding decide the output delay. If $M > N$, then the delay of the row encoding could be larger than the column encoding. Therefore, the delay will be $\tau_{\mathsf{XOR\_2X1}} \times$ (ceil($\log_2$(max(M,N))). Also, the ultimate bit depends on which value is used (ur or uc).

Therefore, the delay of output is as Equation 10 shows.

□

D. DECODING

By using parity checking, the decoder can find the column and row indices of the flipped bit. The parity equations are as follows:

$$\begin{aligned} sr_i &= b_{i,0} \oplus b_{i,1} \oplus \cdots \oplus b_{i,N-1} \oplus r_i \\ sc_j &= b_{0,j} \oplus b_{1,j} \oplus \cdots \oplus b_{N-1,j} \oplus c_j \\ sr_N &= r_0 \oplus r_1 \oplus \ldots r_{M-1} \oplus u \\ sc_M &= c_0 \oplus c_1 \oplus \ldots c_{N-1} \oplus u \end{aligned} \tag{11}$$

The outputs of Equation 11 are two arrays: parity column (sc) and parity row (sr). If there is one or no flipped bit, the decoder can correct it using a mask:

$$\mathsf{Mask} = \begin{bmatrix} m_{0,0} & \ldots & m_{0,N-1} & m_{0,N} \\ m_{1,0} & \ldots & m_{1,N-1} & m_{1,N} \\ \ldots & \ldots & \ldots & \ldots \\ m_{M-1,0} & \ldots & m_{M-1,N-1} & m_{M-1,N} \\ m_{M,0} & \ldots & m_{M,N-1} & m_{M,N} \end{bmatrix} \tag{12}$$

where

$$m_{i,j} = \begin{cases} 1 \text{ if } sr_i == 1 \text{ and } sc_j == 1 \\ 0 \text{ otherwise.} \end{cases} \tag{13}$$

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

$$F_k = \hat{F}_k \oplus \mathsf{Mask} \tag{14}$$

The decoder fails to correct when there are two or more faults. In this fashion, the decoder sends a NACK signal and

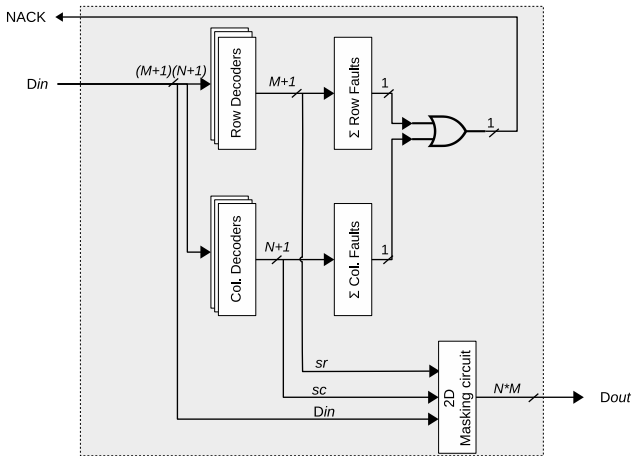

Figure 6: PPC Decoder Architecture.

a hybrid automatic retransmission request (HARQ) is used to perform correction. To support HARQ, the decoder has to detect the occurrence of faults by summarizing the number of flipped bits in the row and column as follows:

$$\begin{aligned} fr &= \sum_{i=0}^{N+1} sr_i \\ fc &= \sum_{i=0}^{M+1} sc_i \\ \mathsf{NACK} &= (fr \geq 2)\ \mathsf{OR}\ (fc \geq 2) \end{aligned} \tag{15}$$

Note that the above equations require adders and comparators which are probably over-complicated for high-speed coding techniques. To simplify the calculation of NACK, decoders can simply check either sc or sr if they are not all-zeros or one-hot values. For instance, with $M = 4$ and $N = 3$, fr, fc, and NACK can be expressed as:

$$\begin{aligned} fr = \neg\ (&\overline{sr_0}sr_1\overline{sr_2}\overline{sr_3} + sr_0\overline{sr_1}sr_2\overline{sr_3} \\ &+ \overline{sr_0}sr_1\overline{sr_2}sr_3 + \overline{sr_0}sr_1sr_2\overline{sr_3} \\ &+ \overline{sr_0}sr_1\overline{sr_2}sr_3) \\ fc = \neg\ (&\overline{sc_0}sc_1\overline{sc_2} + sc_0\overline{sc_1}\overline{sc_2} + \overline{sc_0}sc_1\overline{sc_2} \\ &+ \overline{sc_0}\overline{sc_1}sc_2) \\ \mathsf{NACK} = fr + fc \end{aligned} \tag{16}$$

Figure 6 shows the architecture of the decoder. Similarly to the encoder, there are modules using XOR-trees (*Col. Decoder* and *Row Decoder*). Then, two arrays sr and sc are used for masking the faults. By taking the sum the number of faults in rows and columns ($\sum$ Row Faults and $\sum$ Col. Faults), the decoder can determine whether multiple faults are occurring. The NACK signal is used for retransmission using the HARQ protocol.

The expected number of gates ($\mathbf{G}$) and delay (τ) of the decoding process are shown in Equation 17 and Equation 18, respectively. The proofs of these equations are also presented in Lemma IV.4 and Lemma IV.5. Note that the synthesizer could pick different gates with multiple inputs to optimize the area and timing.

From Equation 17 and Equation 18, with a given n data-bit ($M \times N = n$), the area cost and delay complexity are $O(n)$ and $O(log_2(\sqrt{n}))$, respectively. In comparison, both of Hamming's and SECDED's area cost and delay complexities are $O(n)$ and $O(log_2(n))$.

Lemma IV.4. *The expected number of gates ($\mathbf{G}$) of the decoder is :*

$$\begin{aligned} \mathbf{G}^{decoder}_{\mathsf{XOR\_2X1}} &= M(N+1) + N(M+1) + MN \\ \mathbf{G}^{decoder}_{\mathsf{INV}} &= N + M + 2 \\ \mathbf{G}^{decoder}_{\mathsf{AND\_2X1}} &= (N+2)N + (M+2)M \\ \mathbf{G}^{decoder}_{\mathsf{OR\_2X1}} &= M + N \end{aligned} \tag{17}$$

Lemma IV.5. *The expected delay (τ) of the decoding process is:*

$$\begin{aligned} \tau^{decoder}_{\mathsf{Mask}} &= \tau_{\mathsf{XOR\_2X1}} \times ceil(log_2(max(M+1, N+1))) \\ \tau^{decoder}_{\mathsf{Dout}} &= \tau^{decoder}_{\mathsf{M}} + \tau_{\mathsf{XOR\_2X1}} \\ \tau^{decoder}_{\mathsf{Sum\_Faults}} &= \tau_{\mathsf{INV}} + ceil(log_2(max(M+1, N+1))) \\ &\quad \times (\tau_{\mathsf{AND\_2X1}} + \tau_{\mathsf{OR\_2X1}}) \\ \tau^{decoder}_{\mathsf{NACK}} &= \tau^{decoder}_{\mathsf{M}} + \tau^{decoder}_{\mathsf{Sum\_Faults}} + \tau_{\mathsf{OR\_2X1}} \end{aligned} \tag{18}$$

Proof. The number of XORs of the row and column decoders can be calculated as XOR-trees. Note that the inputs are (N+1) columns and (M+1) rows. The number of XORs is $(M(N+1) + N(M+1))$ and the delay is $\tau_{\mathsf{XOR\_2X1}} \times \text{ceil}(\log_2(\max(M+1, N+1)))$.

The sums of row and column faults in this design are based on the one-hot and all zeros comparators (Equation 16). For each sum of Z bits, there are Z NOT gates and $Z+1$ AND-trees of Z inputs. To take the sum, an OR array of Z inputs is used.

$$\begin{aligned} \mathbf{G}_{\mathsf{INV}} &= Z \\ \mathbf{G}_{\mathsf{AND\_2X1}} &= (Z-1)(Z+1) \\ \mathbf{G}_{\mathsf{OR\_2X1}} &= Z - 1 \end{aligned} \tag{19}$$

The delay of the Sum_Faults signal is:

$$\begin{aligned} \tau_{\mathsf{first\ INVs}} &= \tau_{\mathsf{INV}} \\ \tau_{\mathsf{AND-trees}} &= \text{ceil}(\log_2(Z)) \times \tau_{\mathsf{AND\_2X1}} \\ \tau_{\mathsf{OR-tree}} &= \text{ceil}(\log_2(Z)) \times \tau_{\mathsf{OR\_2X1}} \\ \tau_{\mathsf{Sum\_Faults}} &= \tau_{\mathsf{INV}} + \text{ceil}(\log_2(Z)) \times (\tau_{\mathsf{AND\_2X1}} + \tau_{\mathsf{OR\_2X1}}) \end{aligned} \tag{20}$$

It worth mentioning that this circuit could be further optimized using a multiple input circuit with inversed values. For instance, in our experiment using Synopsys Design Compiler with $M = N = 3$, the CAD tool picks one AOI22_X1 gate, two OAI21_X1 gates, and one NAND3_X1 gates to obtain the value. The mask circuit is basically a 2D $M \times N$ XOR array. The delay is $\tau_{\mathsf{XOR\_2X1}}$ and the number of XOR is $M \times N$. By using $Z = N + 1$ and $Z = M + 1$, we can obtain Lemma IV.4 and Lemma IV.5. □

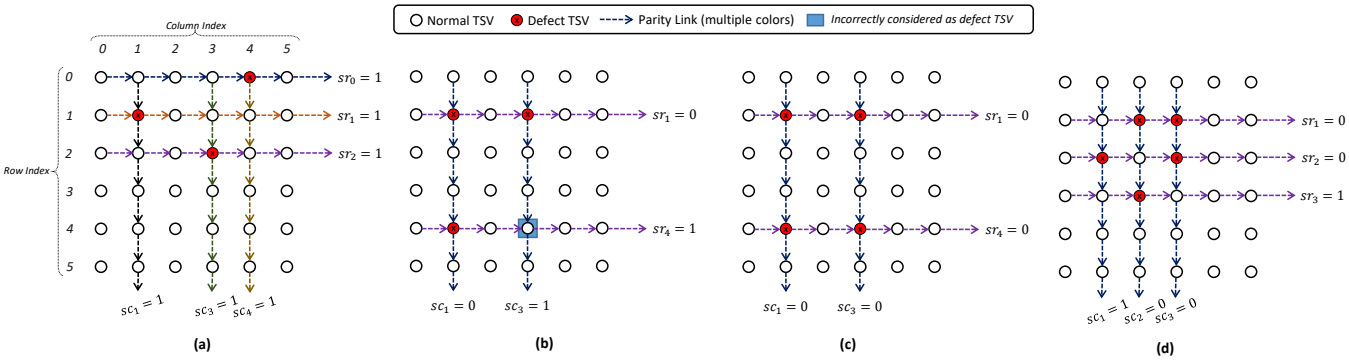

Figure 7: Multiple defects patterns with PPC: (a) Detectable case; (b) Undetectable pattern which incorrectly indicate defective TSVs; (c)Undetectable pattern which PPC recognizes the TSV group as healthy ; (d) Undetectable pattern which PPC corrects one failed TSV and considers the rest as healthy.

V. EXTENDED PPC

In the previous section, we discussed the design of PPC on correcting and detecting faults. In this section, we present the Extended PPC (EPPC) which provides the ability to detect more than two faults during transmission.

A. CORRECTABILITY AND DETECTABILITY

In general, PPC can ensure the ability to correct one and detect two flipped bits. However, if there are more than two flipped bits, PPC also has chances to detect them. For instance, Figure 7(a) illustrates a case of three flipped bits with indices (1,1), (2,3), and (0,4) of a PPC (5×5) (there are column and row parity checks to have a 6×6 TSVs group). The PPC decoder outputs a row check $sr = 000111$ and a column check $sc = 011010$. By determining that the sr and sc values have multiple bits '1' (Equation 15 or 16), the decoder can detect more than two faults.

Although PPC can detect more than two faults, there is a weak point in its detection approach that always prevents it from detecting three faults. For instance, if bits with indices (i, j), (i, k) and (l, j) are flipped, both cr_i and sc_j are '0' which makes the decoder fail to detect while both sc_k and sr_l could be '1'. This syndrome makes the decoder understand that there is one fault and corrects the bit $b_{l,k}$. Figure 7(b) shows a simple illustration of such a case. In this case, a PPC (5×5) having three flipped bits with indices (1,1), (1,3) and (4,1) is decoded to have $sc = 001000$ and $sr = 010000$. Because the flipped bits belong to the same row and column, the parity check bit (sc and sr) is calculated as correct. However, because row 3 and column 3 are determined as having flipped bit, the decoder determines that bit (3,3) is flipped. Another example is shown in Figure 7(c) where bit (3,3) is also flipped. Here, PPC determines no flipped bit exists and recognizes the system as healthy. Figure 7(d) also shows another example of five defects causing misinterpretation during decoding using PPC. Here, since the parity checks lead to $sr_2 = sc_1 = 1$, the PPC decoder concludes that the TSV at the position (2,1) is defective.

Because of this behavior, we can easily observe that the detection of PPC is still limited to two faults and it cannot guarantee the ability to detect more than two defects. If we can reform the patterns into different ones, we could solve this problem and provide more multi-fault detection.

B. PRELIMINARY EXTENSION

In order to correct and detect more faults, we could extend the PPC with orthogonal Latin square (OLS) matrices, as in our previous work in [29]. Note that this will limit the shape of the PPC to a square ($M = N$). The work on low power OLSC using the OLS matrices for encoding and decoding can be found in [57]. Obviously, squared PPC ($M = N$) without a u bit is a case of OLSC. However, since we focus on permanent defects, it is more efficient to use spare TSV as correction. To break the undetectable patterns, there are two extended matrices using OLSC, as shown in Figure 8. *Matrix-0* and *Matrix-1* are used in the baseline version of PPC where the parity bits are the result of calculating the parity of columns and rows. To have a different coding scheme, *Matrix-2* and *Matrix-3* are used. We also could observe that the design for *Matrix-2* and *Matrix-3* is identical to the original matrices of PPC but they have different rows and columns. While the original matrix could be limited by the undetectable patterns, simply switching the different matrices could break these patterns. The extra cost and latency are only the area of $M \times N$ MUX 2:1 multiplexers and the delay of a single MUX 2:1, respectively.

Using OLS matrices can help to generate different row and column checks for PPC; however, it has two major drawbacks: *(1)* it only works with square matrices ($M = N$) and *(2)* it still groups nearby TSVs in the same row/column check which is not efficient for the clustering defects. In fact, the detection rate of PPC with OLSC matrices of some configurations even drop to 95+% as in [29]. In order to solve these two problems, we present the new extensions for PPC in the next sections.

C. EXTENDING PPC

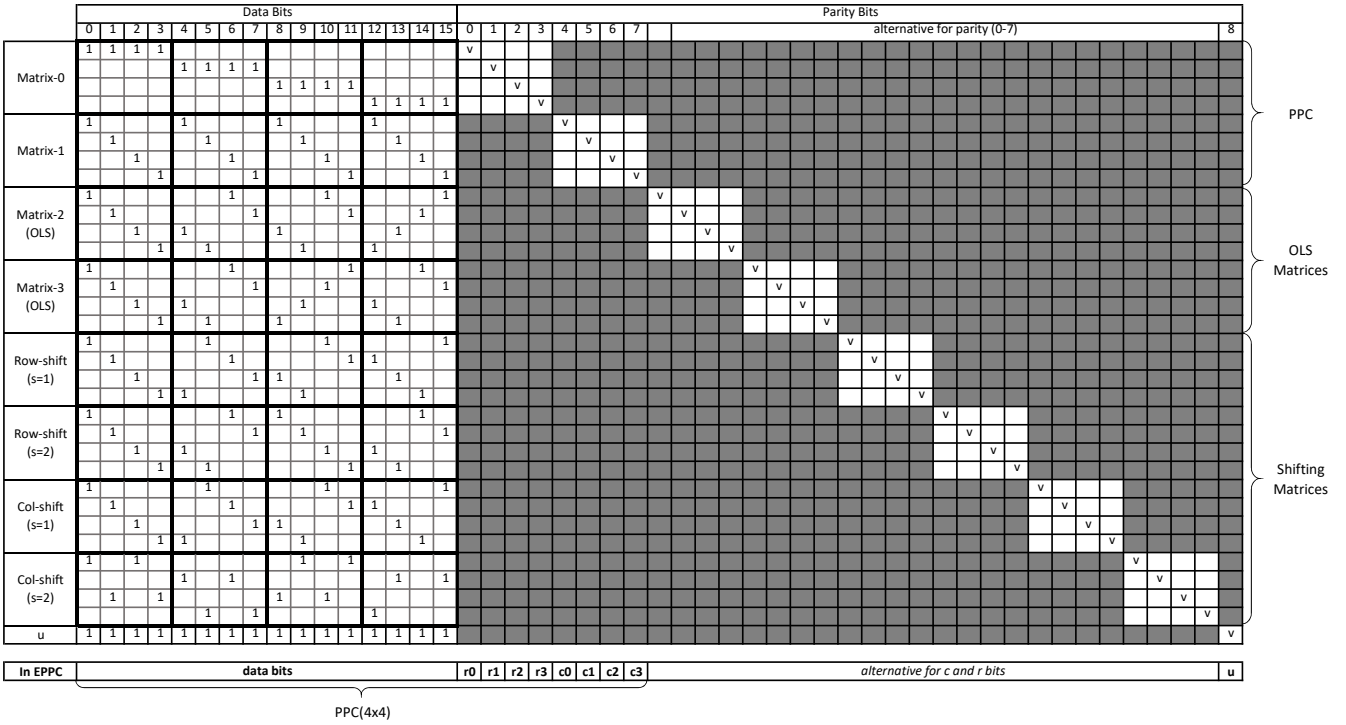

Figure 8: Extending PPC using shifting matrices.

1) Shifting matrices with $s = \pm 1$

For the data that cannot form a square matrix (i.e., 32 bits), we can use alternative rows and columns with the shifting technique shown in Figure 8. We denote s as the shift jump between rows or columns. The method is described as follows:

- *Row-shift*: In the column check, the TSV (i, j) matches with $(i + 1, (j + s \bmod (N + 1)))$ for calculating the parity.
- *Column-shift*: In the row check, the TSV (i, j) matches with $((i + s \bmod (M + 1)), j + 1)$ for calculating the parity.

We can shift to either direction (negative and positive values for s) and can combine *row-shift* with *column-shift* to have different matrices.

For the ease of understanding, Figure 9 shows examples *row-shift* and *column-shift* with $s = \pm 1$. The undetectable pattern in Figure 7(b) is used as the tackling pattern in this case. With three defects in the positions (1,1), (1,3), and (4,1) PPC fails to recognize the case and correct the healthy TSV. However, with shifting matrices, the system can break the undetectable pattern. Figure 9(a) shows the *row-shift* $s = 1$ where the row checks are kept as the normal PPC because shifting among each row does not affect the parity result. However, now the column parity is changed due to the bit indices being shifted. For instance, the 5^th column now consists of TSVs with indices (0,0), (1,1), (2,2), (3,3), (4,4) and (5,5). As a result, we can observe that the column check sc has three values of '1' which help the system detect the faults following Equation 15. On the other hand, Figure 9(b), (c) and (d) show the other shifting matrices where we could have different row and column checks depending on how we shifted them. In all cases, *row-shift* and *column-shift* can break the pattern to have a better detection rate.

2) Limitation of of $s = \pm 1$

As shown in Figure 9, we can break the pattern by using shifting matrices. The same principle can be applied for using OLS matrices in [29]. However, we also observe the limitation of this method where shifting is not efficient for breaking the undetectable patterns.

At first, it seems using only one shifting matrix is enough to break the pattern; however, there are still some patterns that cannot be broken. Figure 10 shows a case where *row-shift* or *column-shift* with $s = 1$ cannot detect the three defects. Here, there are three defective TSVs at (1,1) (1,2) and (2,2). However, once we apply *row-shift* and *column-shift* with $s = 1$, as depicted in Figures 10(b) and (c), the pattern is changed to a new pattern that also cannot be detected by the new matrix. We also observe that, by shifting with $s = 1$, there are shared bit indices between row and column checks which is one drawback of this method. For instance, sc_5 in *row-shift* with $s = 1$ is exactly the same as sr_5 in *column-shift* with $s = 1$. This repetition reduces the efficiency of the shifting method where both row and *column-shift* with $s = 1$ fails to detect the pattern.

On the other hand, using $s = -1$ can break the pattern that $s = -1$ fails to do, as represented in Figure 10(d). In

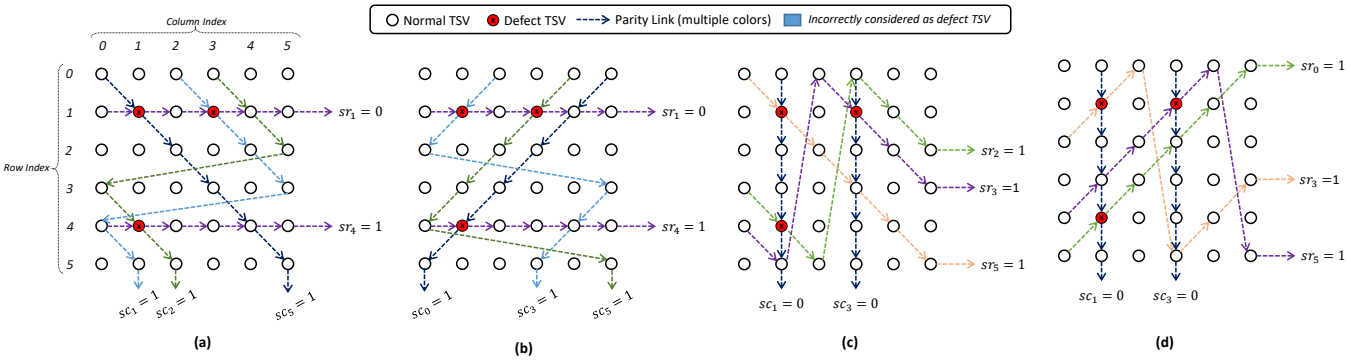

Figure 9: An illustration of shifting method: (a) *row-shift* $s = 1$; (b) *row-shift* $s = -1$; (c) *column-shift* $s = 1$; (d) *column-shift* $s = -1$.

fact, the *column-shift* with $s = -1$ can make three values of sr_i turn into '1' which helps detecting the multiple fault cases. Even though using $s = -1$ can help breaking this pattern, we observe that using this s value can also have some pitfalls. For instance, using $s = -1$ fails to break the pattern of three defective TSVs at (1,1) (1,2) and (0,2). Nevertheless, we observe using multiple shifting matrices might be needed to have a better detection rate.

Despite the ability to tackle some undetectable patterns, using multiple matrices with different values of s is still limited because there are some patterns that cannot be tackled even using four shifting matrices and the original PPC. Figure 11 (a-e) shows the case where PPC and EPPC both fail to detect the pattern. Even shifting with $s = 1$ or $s = -1$, it still cannot break the pattern and the system still behaves as there is only one defective TSV.

In summary, despite having the ability to break some undetectable patterns, there are three major drawbacks of EPPC:

1) There are still some undetectable patterns that EPPC fails to detect as shown in Figures 11 (a-e).
2) The *row-shift* and *column-shift* are repeated as we can observe the similarity between *row-shift* ($s = 1$) and *column-shift* ($s = 1$) in Figures 10 (b) and (c).
3) The fault-rate model in Equation 1 using Euclidean distance as the key parameter of clustering faults has not been considered.

These drawbacks can reduce the detection rate of EPPC. To improve the detection rate, we present the distance-aware EPPC in the next section.

D. DISTANCE-AWARE EPPC

In this section, we use a method named distance-aware EPPC by shifting by $|s| > 1$ index per row/column as in Figure 8 and Figure 9. Now, the *row-/column-shift* mechanism is as follows:

- *Row-shift:* the alternative column i consists of the TSVs with index $(j, (i + s * j) \bmod (N + 1))$ (N: number of columns in data, N+1: number of columns in TSV group) where the last index is for a parity bit. For example, with $M = N = 5$ and $s = 2$, a new column for parity check consists of bit indices (0,0) (1,2), (2,4), (3,0), (4,2) and (5,4) where (5,4) is for the parity bit.
- *Column-shift:* the alternative row j consists of the TSVs with index $((j + s * i) \bmod (M + 1), i)$ (M: number of rows in data: M+1: number of rows in TSV group) where the last index is for a parity bit. For example, with $M = N = 5$ and $s = 2$, a new row for parity check consists of bit indices (1,0) (3,1), (5,2), (1,3), (3,4) and (5,5) where (5,5) is for the parity bit.

Figure 11 (f) shows the *column-shift* with $s = -2$. Here, we observe that one row for parity check consists of (1,0), (5,1), (3,2), (1,3), (5,4) and (3,5). By distancing the bit indices within rows and column check, it can now break the undetectable pattern. Similar cases can be observed in Figure 11 (g) and (h) where we apply rows shift with $s = 2$ and $s = 3$. Obviously, using a different value of s can help us tackle the undetectable pattern of $s = \pm 1$ which is the first problem. The second problem of repetition of rows and columns for parity check can be solved by using different $|s|$ values for checking.

For the third problem, as shown in Equation 1, the fault-rate depends on the distance to the cluster center. To avoid the undetectable pattern, we need to limit the number of defective TSV per row and column of each parity check. Therefore, the optimal value s should provide the longest Euclidean distance (d_E) between indices of rows and columns in the parity check. The Euclidean distance among PPC is only '1' which makes it difficult to avoid the pattern due to clustering defect. EPPC with $s = 1$ also provides a minimum Euclidean distance of $\sqrt{2} \simeq 1.44$ which is certainly not high enough.

In order to maximize the minimum Euclidean distance between TSVs of a row and column check, the shifting value can be approximated by Equation 21. The proof of this equation is shown in Lemma V.1.

Lemma V.1. *Given an extended PPC with the size $N \times M$, the approximate s values for maximum value of minimum*

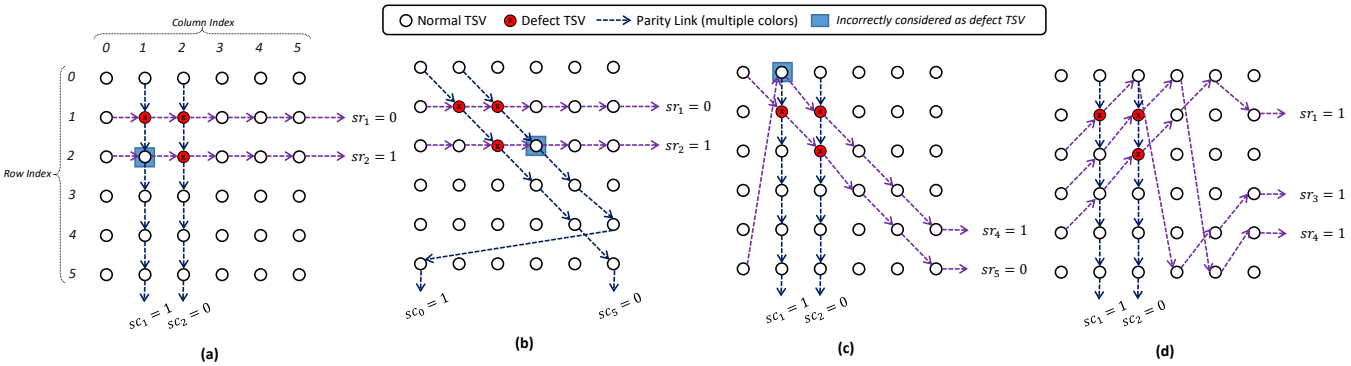

Figure 10: An illustration of undetected case with shifting method $s = 1$: (a) original PPC which fails to detect; (b) *row-shift* $s = 1$ also fails to detect; (c) *column-shift* $s = 1$ also fails to detect; (d) *column-shift* $s = -1$ can detect this case.

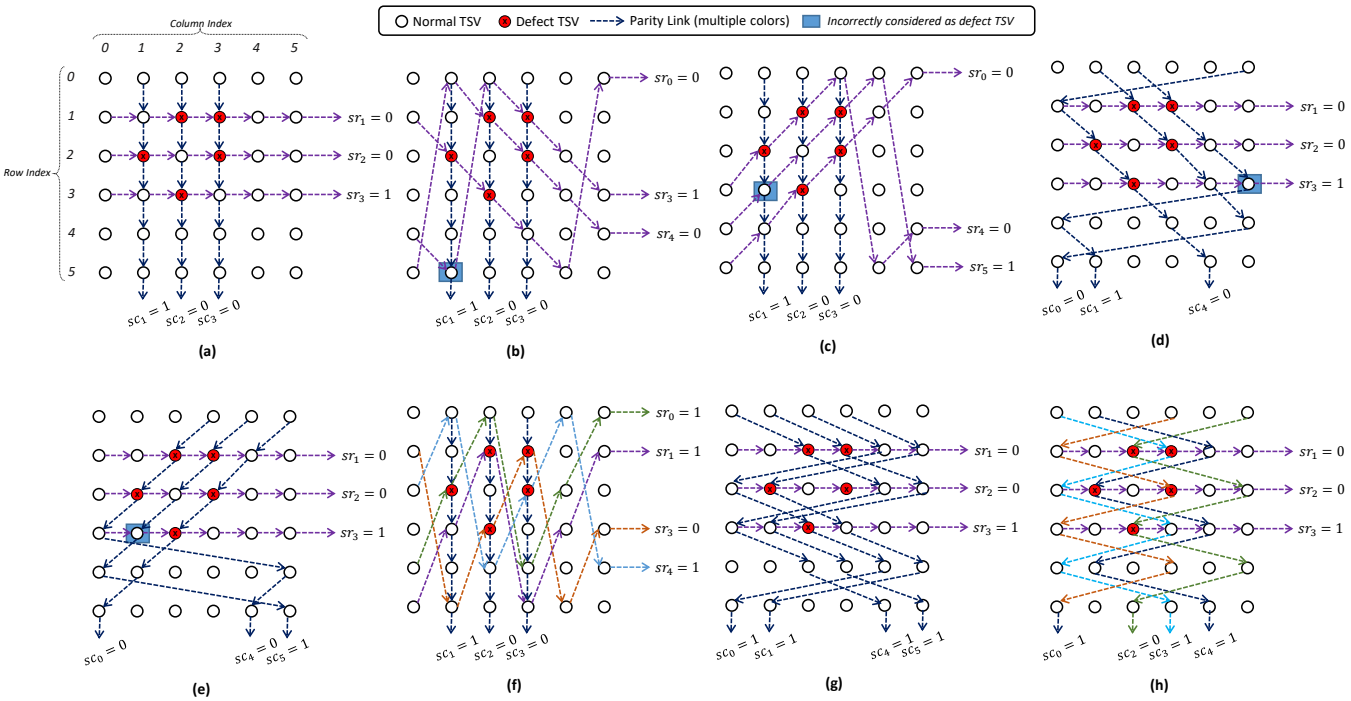

Figure 11: An illustration of undetected case with shifting method $s = \pm 1$ which only corrects one defective TSV: (a) original PPC which fails to detect; (b) *row-shift* $s = 1$ also fails to detect; (c) *row-shift* $s = -1$ also fails to detect; (d) *column-shift* $s = 1$ also fails to detect; (e) *column-shift* $s = -1$ also fails to detect this case; (f) *column-shift* $s = -2$ can detect; (g) *row-shift* $s = 2$ can detect; (h) *row-shift* $s = 3$ can also detect.

Euclidean distance are:

$$\begin{aligned} s_{row} &\simeq \pm\sqrt{N} \\ s_{col} &\simeq \pm\sqrt{M} \end{aligned} \tag{21}$$

Proof. Since both *row-shift* and *column-shift* follow the same principle of fixed distance s in each row/column check, here we consider *row-shift*.

Here, the two consecutive indices in a column check of *row-shift* are $(j, (i + s * j) \bmod (N + 1))$ and $(j + 1, (i + s * j + s) \bmod (N + 1))$. If the column index is less or equal to N, the column index is not circularly shifted using the *mod* function. Therefore, the Euclidean distance between adjacent indices is:

$$d_E^{\text{adjacent}} = \sqrt{1^2 + s^2} \tag{22}$$

Figure 12 shows the illustration for the adjacent distance. For example, we can use the calculation in Equation 22 for (0,0) to (1,s) or (0,1) to (1,s+1).

Once the column index $(i + s * j)$ is larger than N, the next index will circularly shift to $(i + s * j) \bmod (N + 1)$. For example, Figure 11(g) shifts the index from (2,4) to (3,0) instead of (3,6). The start index of column check is $(0, 0)$, the index of the TSV that is 1^{st} circularly shifted ($r_{1^{\text{st}}-\text{shift}}$,

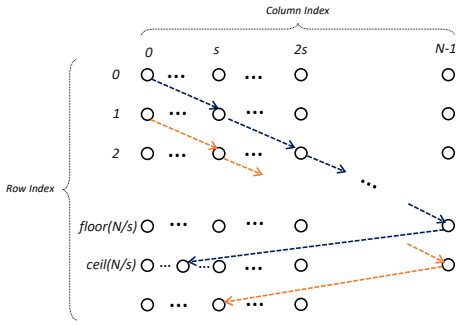

Figure 12: An illustration of the minimum Euclidean distance in shifting.

$c_{1^{st}-\text{shift}}$) is:

$$\begin{aligned} r_{1^{st}-\text{shift}} &= \text{ceil}(\frac{N}{s}) \\ c_{1^{st}-\text{shift}} &= (N \bmod s) < s \end{aligned} \tag{23}$$

For instance, with *row-shift* $s = 2$, Figure 11(g) shows the new column for parity check is (0,0), (1,2), (2,4), (3,0), (4,2) and (5,4). Here, the first shifted index is (3,0) and we realize that the shifted index is closer to (0,0) with $d_E = 3$ than the last index (2,4) with $d_E = \sqrt{17}$. From Figure 11, we could also observe that the column index is shifted by every ceil($\frac{N}{s}$) rows.

Due to the similarity of the shifting index, we have the new minimum Euclidean distance between the first circular shifted indices and (0,0). This distance should also be considered. If the index is circularly shifted too fast with a large s value, the shifted index might cause a smaller minimum Euclidean distance. The Euclidean distance between the consecutive shifted indices is:

$$d_E^{\text{shift}} = \sqrt{(r_{1^{st}-\text{shift}})^2 + (c_{1^{st}-\text{shift}})^2} \tag{24}$$

Here, the worst case is N divided by s which leads to $c_{1^{st}-\text{shift}} = 0$ where the shifting method repeats the same column index (i.e Figure 11(g) has (0,0) and (3,0) in the parity column check). We now can approximate the distance as:

$$d_E^{\text{shift}} \simeq \sqrt{(\frac{N}{s})^2 + 0^2} = \frac{N}{s} \tag{25}$$

Obviously, the minimum Euclidean distance will be the minimum between these two distances:

$$d_E^{\text{minimum}} = \min(d_E^{\text{shift}}, d_E^{\text{adjacent}}) \tag{26}$$

To optimize the distance, we balance the two distances to maximize the minimum distance:

$$d_E^{\text{adjacent}} \simeq d_E^{\text{shift}} \tag{27}$$

We can use Equation 25 and Equation 22 to approximate the value of s as follows:

$$\begin{aligned} \sqrt{1^2 + s^2} &\simeq \frac{N}{s} \\ 1 + s^2 &\simeq (\frac{N}{s})^2 \\ s &\simeq \sqrt{N} \end{aligned} \tag{28}$$

For the *row-shift* we should have $s_{row} \simeq \pm\sqrt{N}$ to maximize the Euclidean between indices. For the *column-shift*, we expect $s_{col} \simeq \pm\sqrt{M}$ to have the optimal distance between TSVs in the same row/column check. □

In Figure 11, the optimal value of $s \simeq \pm\sqrt{N} = \pm\sqrt{5}$; therefore, $s = \pm 2$ is our best choice. Figure 11(b-e) show $s = 1$ with a minimum Euclidean distance of $\sqrt{2}$. Figure 11(h) shows $s = 3$ with a minimum Euclidean distance of 2. Meanwhile, Figure 11(f) and (g) show $s = \pm 2$ with a minimum Euclidean distance of $\sqrt{5}$ which is better than both $= \pm 1$ and $s = \pm 3$.

E. ARCHITECTURE

The architectures of our EPPC encoder and EPPC decoder are shown in Figures 13 and 14, respectively. In comparison to the PPC, it needs a *matrix-switch* module to help change the matrix and a counter to issue a selection signal of the *matrix-switch*.

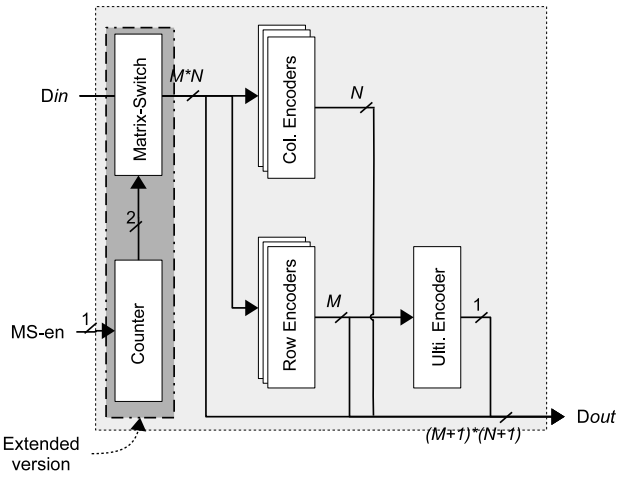

Figure 13: EPPC Encoder Architecture.

The architecture of the *matrix-switch* is shown in Figure 15. The signal D(c) can be assigned as D(a) or D(b) depending on the selection of the multiplexer. To feed to data into the row and column encoder and decoder modules, we need to switch the bit position to help with the variation of the matrix. For instance, the *row-shift*ing right method is implemented as a multiplexer between nearby-bits (i, j) and $(i, j + i)$. Note that we use two different *matrix-switches* for row and column parity if needed (in a pair of matrices, we can share).

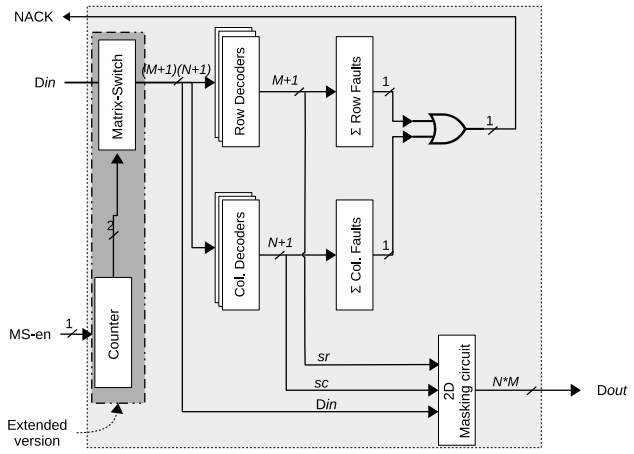

Figure 14: EPPC Decoder Architecture.

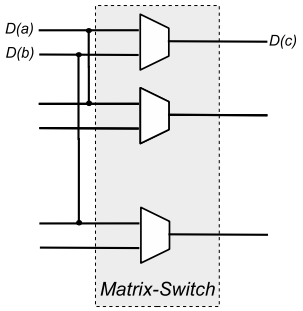

Figure 15: Matrix-Switch architecture using multiplexer.

Here, the extra area cost and delay for encoder and decoder for four matrices (two for row, two for column) are as follows:

$$\begin{aligned} \mathbf{G}^{\text{encoder}}_{\text{MUX\_2X1}} &= ZMN \\ \mathbf{G}^{\text{decoder}}_{\text{MUX\_2X1}} &= Z(M+1)(N+1) \\ \mathbf{G}^{\text{encoder}}_{\text{2-bit counter}} &= \mathbf{G}^{\text{decoder}}_{\text{2-bit counter}} = 1 \end{aligned} \tag{29}$$

$$\begin{aligned} \tau^{\text{EPPC encoder}} &= \tau^{\text{PPC encoder}} + \tau_{\text{MUX\_2X1}} \\ \tau^{\text{EPPC decoder}} &= \tau^{\text{PPC decoder}} + \tau_{\text{MUX\_2X1}} \end{aligned} \tag{30}$$

Where Z is the number of additional matrices. To support more matrices, we must increase the value of Z. In summary, the extended version still keep the complexity of area and delay as $O(n)$ and $O(log_2(\sqrt{n}))$ as PPC's, respectively.

VI. EVALUATION

The EPPC circuit is designed in Verilog-HDL with $45nm$ process technology. The design is implemented using EDA tools by Synopsys. In this evaluation, we first perform the detection rate under random distribution. Then, clustering distribution is investigated for both EPPC and the distance-aware version. Then, the real implementation results are presented and compared. Finally, we compared our work with other existing NoC testing methods.

A. DETECTION PERFORMANCE UNDER UNIFORM RANDOM DISTRIBUTION

In order to study the detection ability of EPPC, we perform a 10,000 Monte-Carlo simulation cases, represented in Figure 16. As we can observe in the results, the original PPC can detect at most two defects and fails to detect 3+ defects with smaller M and N values. This is due to the undetectable pattern easily occurring under random distribution with smaller M and N values. When we apply the shifting technique, we can break the pattern and increase the detectability with EPPC. For instance, with $M = 4$ and $N = 8$ (32-bit), PPC misses 11% of three faults cases; however, EPPC with only *row-shift* can detect 98.8% and EPPC with both row and *column-shift* can detect 100%. When we increase the size of the TSV group, we can have better coverage of PPC; however, EPPC now can guarantee 100% with only *row-shift*ing.

In summary, the proposed PPC provides a reasonable detection rate. By using extra matrices, it could help detect multiple faults without adding overwhelming extra area cost ($M \times N$ 2:1 multiplexers and a 2-bit counter).

B. DETECTION PERFORMANCE UNDER CLUSTERING DISTRIBUTION

To understand the efficiency under clustering defects (multiple defects per group), we evaluate the method using a probability model of clustered TSV faults. Figure 17 shows the evaluation results of detection rate using Monte-Carlo simulation (10,000 tests per case) with the clustering fault distribution and $s = 1$. We limited the defects among one cluster ($N_c = 1$) and constraint the number of defects. We can easily observe that the PPC detection rates drop significantly when compared to the randomly injected defects. This can be explained by the undetectable patterns occurring more frequently with clustering defects. We could notice, with even $M = 32$ and $N = 32$, that PPC can detect with random defects but fails with clustering defects. On the other hand, EPPC significantly improves the detection rate by using shifting matrices. With the *row-shift* matrix, EPPC achieves higher detection rates and with both *row-shift* and *column-shift*, EPPC obtains 100% in most cases except $M = 4, N = 4$ with 9 defects. The main reason of this exception is that, due to the high number of defects, switching matrices cannot break the undetectable patterns. In the case of nine defects, their shape is close to a 3×3. Meanwhile, the optimal distance $s = \sqrt{4} = 2$ is still smaller than the dimension (3) of clustering defect area's shape. Consequently, the distance aware EPPC is no longer efficient in this case. Nevertheless, EPPC can still detect eight defects which is still considerably high.

As we can easily notice, with only *row-shift*, EPPC can only detect less than 100% of the cases. EPPC needs at least two additional matrices (row and *column-shift*) to have a 100% detection rate. To reduce the number of matrices, we can use distance-aware EPPC. Figure 18 shows the detection rate of EPPC with $s = 2$. Compared to EPPC with only

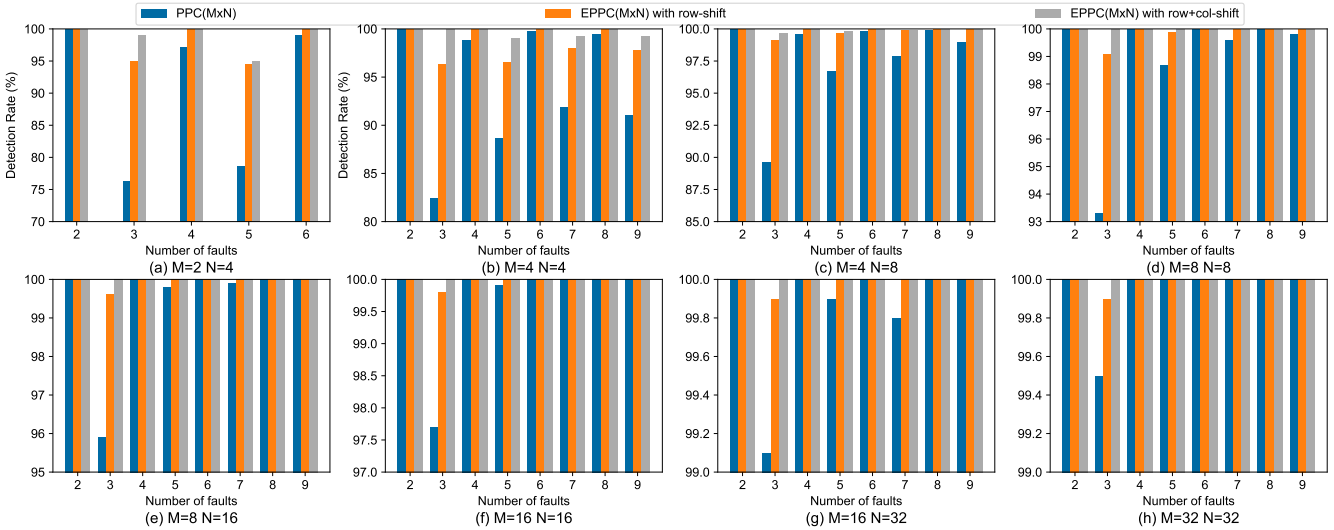

Figure 16: Detection ability evaluation of EPPC with additional matrices using shifting method with $s = \pm 1$ under random distribution.

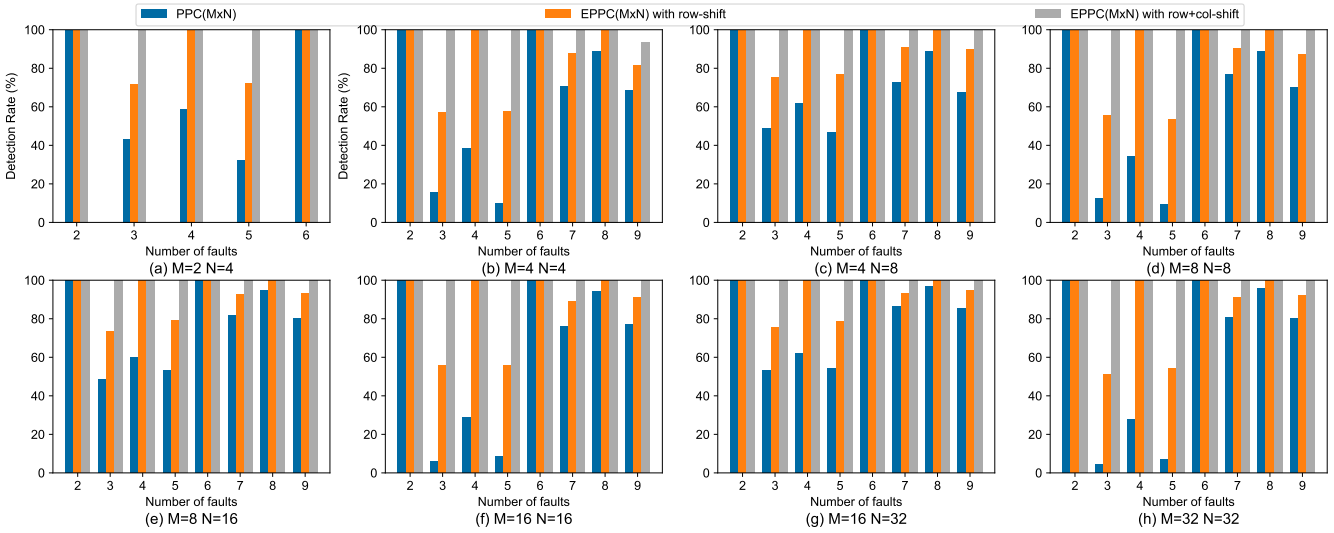

Figure 17: Detection ability evaluation of EPPC with additional matrices using shifting method with $s = \pm 1$ under clustering distribution: $F_b = 1$, $N_c = 1$, $\alpha = 3.0$.

row-shift, illustrated in Figure 17, distance-aware EPPC can obtain a 100% detection rate.

C. HARDWARE IMPLEMENTATION

The hardware implementations of our proposed methods are presented in Figure 1. Here, we compare PPC to SECDED and Hamming with the same data bit-width. We also add the results from [34] and [35] which provide two or three adjacent fault correction coding techniques. The results from [34] and [35] are presented in both area and delay optimization while our design simply targets timing optimization.

The results demonstrate that PPC provides several benefits over SECDED and Hamming. The complexities of PPC's encoders and decoders are lower than the other two. In particular, the area cost of the encoder and decoder for 64-bit PPC are 17.01% and 15.95% less than SECDED. The latency of PPC encoders and decoders are also smaller thanks to the narrower XOR trees. For 64-bit, they are 22.67% and 49.38% lower than SECDED. In comparison to the best area optimized (AO) results in [34] and [35], and despite the fact that we use more parity bits, the proposed design (encoder and decoder) only incurs 15.96% and 14.20% extra area cost, respectively. The best delay optimized (DO) design in [34] and [35] reduces the latency by 67.14% and 64.29% when compared to PPC; however, their complexities are $8\times$ higher. Utilizing one or two additional matrices in *PPC*(8×8) maintains a latency below $1ns$ and increases the area cost by factors of 1.5x and 1.7x, respectively, but greatly improves

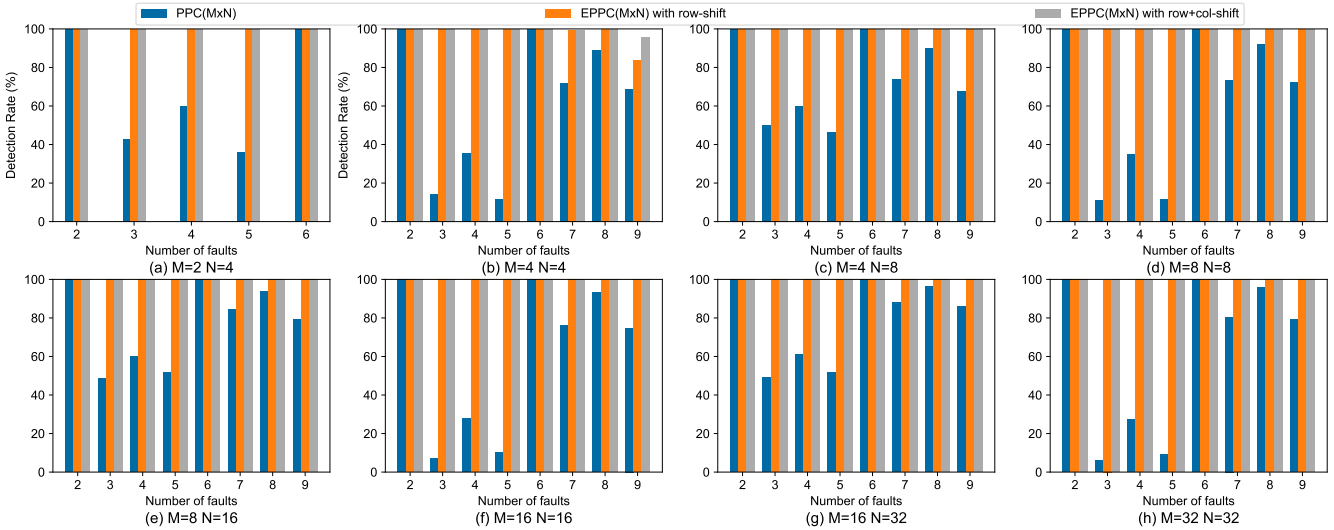

Figure 18: Detection ability evaluation of distance-aware EPPC with additional matrices using shifting method with $s_{row} \simeq \pm\sqrt{N}$ and $s_{col} \simeq \pm\sqrt{M}$ under clustering distribution: $F_b = 1$, $N_c = 1$, $\alpha = 3.0$.

Table 1: Hardware implementation results: "AO" and "DO" are Area Optimization and Delay Optimization, respectively. To scale from 65 nm to 45 nm equations presented in [58] have been used.

Scheme	**Tech.** (nm)	**k** (bit)	**n** (bit)	**Area Cost** (μm^2)				**Latency** (ns)			
				Encoder		Decoder		Encoder		Decoder	
				AO	DO	AO	DO	AO	DO	AO	DO
Hamming [15]	45	64	71	193.1200		463.1060		0.69		1.58	
SECDED [24]	45	64	72	234.6120		487.0460		0.75		1.62	
HP + HARQ-II [17]	65	64	72	9792.5				0.41		0.59	
	45 (*scaled*)	64	72	4896.25				0.29		0.42	
ARQ (CRC-5) [17]	65	64	69	3605.6				0.37		0.41	
	45 (*scaled*)	64	69	1802.8				0.26		0.29	
BCH [17]	65	64	85	77353.2				0.42		0.72	
	45 (*scaled*)	64	85	38676.6				0.30		0.51	
SEC-DAEC [34]	45	64	72	678	812	3106	4227	0.61	0.33	1.75	0.61
TAEC-64-v1 [35]	45	64	72	566	695	5279	7165	0.58	0.30	1.81	0.62
TAEC-64-v2 [35]	45	64	72	572	696	5158	6833	0.59	0.29	2.01	0.61
TAEC-64-v3 [35]	45	64	71	583	703	3672	4928	0.65	0.30	1.67	0.62
TAEC-64-v4 [35]	45	64	71	587	722	4563	5976	0.60	0.31	1.87	0.62
PPC(8×8)	45	64	81	194.7120		409.3740		0.58		0.82	
EPPC(8×8) with 1 additional matrix	45	64	81	341.2780		628.0260		0.53		0.91	
EPPC(8×8) with 2 additional matrices	45	64	81	404.5860		691.3340		0.55		0.97	

the detection rate compared to using a single matrix.

Detailed results of the 64 data bit-width implementations are shown in Table 1. Besides the works in [34] and [35], we also perform the comparison with results obtained from [17] which are implemented in 65 nm technology. Even when scaling to $45nm$, the area cost of Hamming Product Code (HP-HARQ-II) in [17] is $8.11\times$ higher than PPC. The BCH [17] code provides multiple bits correction; however, its complexity is $50\times$ greater than the proposed one. HP-HARQ-II's and BCH's latencies are 28.57% and 18.57% lower despite using older technology. However, our latency is still extremely low (0.58 ns and 0.82 ns). With the delay complexity $O(log_2(\sqrt{n}))$, the results are expected to be lower with higher data-widths. Meanwhile, the area cost is similar to Hamming and SECDED which are two simple coding techniques. It is important to mention that the area cost results have not taken into account the area of TSVs. Note that our design demands more additional TSVs than the others. With the same 64 data bit-width, PPC uses 81 code-word bit-width (or TSVs) while Hamming, SECDED, BCH use 71, 72 and 85 code-word bit-width (or TSVs), respectively.

D. COMPARISON OF NETWORK-ON-CHIP TESTING

Hereafter, we compare the proposed method with existing works targeting TSV/NoC testing in Table 2. Here, we focus on three key parameters: area overhead, performance degradation and response time.

Table 2 also shows a comparison for non-blocking online NoC testings with 32-bit flit and in 45 nm technology. It is important to mention that our NoC router has some protection mechanism for soft error in pipeline stages [59]

Table 2: Comparison of non-blocking testing circuit for Network-on-Chip router (32-bit and 45 nm technology).

Design	*Kakoee et al.* [25][a]	*Tran et al.* [26][a]	*Liu et al.* [27][a]	*Wang et al.* [28][a]	*Dang et al.* [40]	This work
Coverage	Link	Link	Router's modules	Link & Router's module	Link (TSVs)	Link (TSV)
NoC size and topology	10×8 Mesh				any NoC	any NoC[e]
Require adaptive routing	✓	✓	✓	✓	✗	✗
Performance degradation	✓	✓	✓	✓	✗	✗
Synthetic, Period = 20K cycles[b]	$> 10\times$	$> 10\times$	$1.0 - 2.5\times$	$1.5 - 2\times$	$1.0\times$	$1.0\times$
PARSEC, Period = 40K cycles[b]	$> 1.8\times$	$> 1.4\times$	$0.9 - 1.0\times$	$0.9 - 1.0\times$	$1.0\times$	$1.0\times$
Test period/Test time (*cycles*)	500K-1M[c]	200K-1M[c]	20K-	16,840-	64-16,448	2-3
Area (μm^2)	700	700	2200	2400	2291.59	1095.92[f]
Test circuit (area)	✗	✗	✗	✗	✓	✓

[a] Area and power costs are extracted by subtracting to the non-tested router design on paper [28]. Area cost and power consumption of non-tested router [28] are 0.0251 mm^2 and 391.00 μW, respectively.
[b] Performance values, which are extracted from paper [28], 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.* [28] is 16,840 cycles.
[c] The test time is extracted from [28]. Values are selected based on reasonable performance degradation ($> 5\times$ average latency and $> 1.8\times$ execution time).
[e] Our proposal could be also applied for link testing.
[f] The value represents the evaluation of a single link.

or hard errors in buffer or crossbar [60] which might degrade the performance; however, we have disabled them in this evaluation to have a fair comparison. Because our technique targets 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, this work does not require adaptive routing to perform testing because it is an on-communication test (non-blocking testing).

As can be observed in Table 2, our technique offers the smallest area cost and testing time. It is worth stating that we can localize only one defect while the other can localize more (i.e., *Dang et al.* [40] can localize 6 faults). However, beside [40], this work does not degrade the performance of the applied NoC. *Kakoee et al.* [25] and *Tran et al.* [26] 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.* [27] and *Wang et al.* [28] presented promising results under PARSEC benchmarks because of the 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, this work offers no change in the system performance under test while it guarantees a response time under 20.000 cycles. Please note that the upper bound of our technique is only 4 cycles which is significantly smaller than all compared works. For example, the lower bound of *Wang et al.* [28] and *Dang et al.* [40] is 16,840 and 64 cycles, respectively. Meanwhile, *Kakoee et al.* [25] and *Tran et al.* [26] have significantly higher lower-bounds (200.000 or 500.000 cycles) which may not be suitable for real-time applications.

For common blocking testing for NoC links, Table 3 summarizes the existing works and compare them to our proposal which is a non-blocking testing. Since our method works individually between two terminals, its testing does not scale with the network size. In comparison to all existing works, our method provides the shortest test time (TT) which is only 3 cycles and the number of gates are reasonable at nearly 1400 gates. We would like to note that the other works can determine the position of defects if needed while our method only detects the faulty link. Nevertheless, our method is the best in terms of detectablity and testing time. Moreover, we can use our method to provide a calling mechanism for BIST which can localize and extract the positions of faults by using deferred testing [40], [41].

Table 3: Comparison with NoC channel blocking testing methods.

Methods	**NoC Size**	**Coverage** (%)	**AO** (gate/%)	**TT** (cycles)	**TR**
Grecu *et al.* [61]	8 × 8	N/A	166,404	5000	9
Peterson *et al.* [42]	26 × 26	99.8	9%	5100	N/A
Cota *et al.* [62]	5 × 5	n/a	742	182	4
Herve *et al.* [43]	2 × 2	93	835	798	5
Concatto *et al.* [45]	3 × 3	66	31,8%	N/A	9
Raik *et al.* [63]	3 × 3	99.33	4%	320	2
Herve *et al.* [44]	4 × 4	99.93	742	286	12
Strano *et al.* [46]	5 × 5	99.3	11%	1104	4
Kakoee *et al.* [47]	8 × 8	85.6	58%	N/A	9
Ghofrani *et al.* [64]	10 × 10	98	9.65%	N/A	4
Kakoee *et al.* [25]	8 × 8	73.2	58%	N/A	9
Bhowmik *et al.* [48]	3 × 4	100	N/A	416	6
Aghaei *et al.* [65]	16 × 16	89.5	5.1%	70	1
Ours (non-block)	Any	100	1374	3	1

AO: area overhead, TT: testing time, TR: test round.

In summary, our technique provides the fastest execution time without the need of BIST circuits. Once the link is faulty, we adopt the look-ahead fault-tolerant routing algorithm [18] to avoid it during transmission.

E. DISCUSSION

This paper has presented the method to provide one-fault correction and multi-faults detection. In this section, we discussed the advantages and drawbacks of the design.

While the distance-aware EPPC with one additional matrix can detect 100% of the clustering defects due to the fact that we know the possible distribution of defects beforehand using Equation 1. Under random defects, EPPC with two additional matrices can obtain 100% detection rate in most cases. However, the realistic distribution might vary between different parameters or mixing between random and clustering defects which can make the EPPC inefficient. In this case, a BIST approach is more suitable than a low-cost method

such as EPPC. On the other hand, with expected low defect rate, our PPC could be a better method which still provides a certain level of detection coverage while having lower area cost and shorter latency.

Soft errors might occur during the runtime that drive the circuit to an inaccurate result. Our EPPC approach has not been protected from such kind of errors. However, we would like to note that PPC itself has the ability to protect the encoder by comparing the u bits obtained from XOR-ing the row and column check bits. On the other hand, we can apply a technique that we previously proposed to provide protection from soft errors in the pipeline stages in [59]. By repeating the process twice and comparing the results, we can detect the occurrence of soft-errors and execute the stage one more time by using a majority voting as recovery.

In [40], we presented a technique named statistical detection where the decoder records the results for 32 or 64 cycles to detect and localize the fault. Here, we can apply the same principle to obtain perfect detection rate by recording it. However, as we observe in the evaluation, our detection rate reaches 100% which is enough for most safety-critical applications. Recording a high number of cycles might affect the area overhead and power consumption.

One of the common methods is 'walking-1' or 'walking-0' [66] which can indicate stuck-at and bridge models. However, this method requires several cycles to test and the connection should be preempted. Moreover, this method is not efficient for handling crosstalk because the behavior of crosstalk heavily depends on the transition of transmitting values of the aggressor and the victim TSVs.

Our work focuses on how to detect multiple defects in the link; therefore, it cannot localize/diagnose the faulty positions. This is due to the fact that multiple defects might not be correctable due to the lack of redundancies. In fact, we do not use any redundancy in this work and rely on the LAFT algorithm [18] to provide fault-tolerant routing. If readers are interested in the localization of multiple defect online, our work in [40] shows an efficient method without any degradation in system performance. Recovery for clustering defects can be also found in [14], [67].

In our fault consideration, we mentioned that our fault model does not have the ability to model a bridge defect in different shapes such as in a row, column or diagonal line. However, we also want to note that with this type of patterns, PPC can easily detect multiple defects since the output will indicate multiple flipped bits in both rows and columns.

Although our proposal still has some parts that need to be improved, the overall evaluation has shown an adequate value where EPPC can correct one and detect multiple defects. Notably, the worst case execution time of our method is considerably low (3 cycles) which easily outperforms the existing techniques.

VII. CONCLUSION

This paper presents the Extended Parity Product Code (EPPC) to enhance the reliability of TSV-based 3D-IC designs. By exploiting the inherent 2D array organization of TSVs, the proposed approach can efficiently represent the fault manifestation in TSV-based systems allowing it to correct one and detect multiple faults in a set of TSVs. From the conducted experiments, and in contrast to conventional coding schemes that are limited to detecting two faults at most, the proposed EPPC has demonstrated its ability to detect 2-8 faults with 100% detection rate. EPPC also use 2-3 cycles to perform its test instead of thousands of cycles like traditional BISTs. Our analysis also showed that the delay complexity of EPPC is $O(log_2(\sqrt{n}))$ which is significantly lower than that of Hamming/SECDED ($O(log_2(n))$).

As a future work, we plan to investigate the thermal impact on 3D-IC systems. Extending the technique with adaptive coding to support multiple bit localization is another possible direction.

ACKNOWLEDGMENTS

The authors would like to thank the editors and reviewers for their helpful comments to improve the manuscript. This research is partly funded by Vietnam National University, Hanoi (VNU) under grant number QG.18.38.

References

[1] 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)," The Journal of Supercomputing, vol. 66, no. 3, pp. 1507–1532, 2013.

[2] 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.

[3] 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.

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

[5] 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.

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

[7] U. Kang et al., "8 Gb 3-D DDR3 DRAM using through-silicon-via technology," IEEE J. Solid-State Circuits, vol. 45, no. 1, pp. 111–119, 2010.

[8] 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.

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

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

[11] 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.

[12] 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. (ICIS). IEEE, 2010, pp. 69–74.

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

[14] 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.

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

[16] 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.
[17] 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.
[18] A. B. Ahmed and A. B. Abdallah, "Adaptive fault-tolerant architecture and routing algorithm for reliable many-core 3D-NoC systems," Journal of Parallel and Distributed Computing, vol. 93, pp. 30–43, 2016.
[19] 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.
[20] Y. Lou et al., "Comparing through-silicon-via (TSV) void/pinhole defect self-test methods," J. of Electronic Testing, vol. 28, no. 1, pp. 27–38, 2012.
[21] 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.
[22] B. Noia et al., "Pre-bond probing of TSVs in 3D stacked ICs," in 2011 IEEE Int. Test Conf. (ITC). IEEE, 2011, pp. 1–10.
[23] P.-Y. Chen et al., "On-chip TSV testing for 3D IC before bonding using sense amplification," in 2009. ATS'09. Asian Test Symp. IEEE, 2009, pp. 450–455.
[24] M. Hsiao, D. Bossen, and R. Chien, "Orthogonal latin square codes," IBM Journal of Research and Development, vol. 14, no. 4, pp. 390–394, 1970.
[25] 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.
[26] 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.
[27] 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.
[28] J. Wang et al., "Efficient design-for-test approach for networks-on-chip," IEEE Trans. Comput., vol. 68, no. 2, pp. 198–213, 2019.
[29] K. N. Dang, M. Meyer, A. B. Ahmed, A. B. Abdallah, and X.-T. Tran, "2D-PPC: A single-correction multiple-detection method for Through-Silicon-Via Faults," in 2019 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS 2019), 2019.
[30] M. Cho, C. Liu, D. H. Kim, S. K. Lim, and S. Mukhopadhyay, "Pre-bond and post-bond test and signal recovery structure to characterize and repair tsv defect induced signal degradation in 3-d system," IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 1, no. 11, pp. 1718–1727, 2011.
[31] S. Deutsch and K. Chakrabarty, "Contactless pre-bond TSV test and diagnosis using ring oscillators and multiple voltage levels," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 33, no. 5, pp. 774–785, 2014.
[32] Y.-J. Huang, J.-F. Li, J.-J. Chen, D.-M. Kwai, Y.-F. Chou, and C.-W. Wu, "A built-in self-test scheme for the post-bond test of TSVs in 3D ICs," in 29th VLSI Test Symposium. IEEE, 2011, pp. 20–25.
[33] Y.-w. Lee, H. Lim, and S. Kang, "Grouping-based TSV test architecture for resistive open and bridge defects in 3-D-ICs," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 10, pp. 1759–1763, 2016.
[34] 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.
[35] 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.
[36] S. B. Wicker and V. K. Bhargava, Reed-Solomon codes and their applications. John Wiley & Sons, 1999.
[37] I. S. Reed and X. Chen, Error-control coding for data networks. Springer Science & Business Media, 2012, vol. 508.
[38] 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.
[39] 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.
[40] K. N. Dang, A. B. Ahmed, A. B. Abdallah, and X.-T. Tran, "TSV-OCT: A Scalable Online Multiple-TSV Defects Localization for Real-Time 3-D-IC systems," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019.
[41] K. N. Dang, A. B. Ahmed, B. A. Abderrazak, and X.-T. Tran, "TSV-IaS: Analytic Analysis and Low-Cost Non-Preemptive on-Line Detection and Correction Method for TSV Defects," in Proc. IEEE Computer Society Annual Symp. VLSI (ISVLSI), 2019, pp. 501–506.
[42] K. Petersen and J. Oberg, "Toward a scalable test methodology for 2D-mesh network-on-chips," in Proc. Automation Test in Europe Conf 2007 Design Exhibition, Apr. 2007, pp. 1–6.
[43] M. Herve et al., "Diagnosis of interconnect shorts in mesh NoCs," in Proc. 3rd ACM/IEEE Int. Symp. Networks-on-Chip, May 2009, pp. 256–265.
[44] ——, "Concurrent test of Network-on-Chip interconnects and routers," in Proc. 11th Latin American Test Workshop, Mar. 2010, pp. 1–6.
[45] C. Concatto et al., "Improving yield of torus nocs through fault-diagnosis-and-repair of interconnect faults," in Proc. 15th IEEE Int. On-Line Testing Symp, Jun. 2009, pp. 61–66.
[46] A. Strano et al., "Exploiting network-on-chip structural redundancy for a cooperative and scalable built-in self-test architecture," in Proc. Automation Test in Europe 2011 Design, Mar. 2011, pp. 1–6.
[47] M. R. Kakoee, V. Bertacco, and L. Benini, "A distributed and topology-agnostic approach for on-line NoC testing," in Proc. Fifth ACM/IEEE Int. Symp, May 2011, pp. 113–120.
[48] B. Bhowmik et al., "Impact of NoC interconnect shorts on performance metrics," in Proc. Twenty Second National Conf. Communication (NCC), Mar. 2016, pp. 1–6.
[49] 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.
[50] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," IEEE Trans. Commun., vol. 46, no. 8, pp. 1003–1010, 1998.
[51] F. Chiaraluce and R. Garello, "Extended Hamming product codes analytical performance evaluation for low error rate applications," IEEE Trans. Wireless Commun., vol. 3, no. 6, pp. 2353–2361, 2004.
[52] K. N. Dang et al., "2D-PPC: A single-correction multiple-detection method for through-silicon-via faults," in IEEE Asia Pacific Conference on Circuits and Systems, 2019.
[53] K. Chakrabarty et al., "TSV defects and TSV-induced circuit failures: The third dimension in test and design-for-test," in 2012 IEEE Int. Rel. Physics Symp. (IRPS). IEEE, 2012, pp. 5F–1.
[54] I. Koren and Z. Koren, "Defect tolerance in VLSI circuits: techniques and yield analysis," Proceedings of the IEEE, vol. 86, no. 9, pp. 1819–1838, 1998.
[55] F. J. Meyer and D. K. Pradhan, "Modeling defect spatial distribution," IEEE Transactions on Computers, vol. 38, no. 4, pp. 538–546, 1989.
[56] K. A. Bowman et al., "Energy-efficient and metastability-immune resilient circuits for dynamic variation tolerance," IEEE Journal of Solid-State Circuits, vol. 44, no. 1, pp. 49–63, 2009.
[57] S. E. Lee, Y. S. Yang, G. S. Choi, W. Wu, and R. Iyer, "Low-power, resilient interconnection with orthogonal latin squares," IEEE Design & Test of Computers, vol. 28, no. 2, pp. 30–39, 2011.
[58] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of cmos device performance from 180 nm to 7 nm," Integration, vol. 58, pp. 74–81, 2017.
[59] K. N. Dang, M. Meyer, Y. Okuyama, X.-T. Tran, and A. B. Abdallah, "A soft-error resilient 3D network-on-chip router," in IEEE 7th International Conference on Awareness Science and Technology, 2015.
[60] K. N. Dang, M. Meyer, Y. Okuyama, and A. B. Abdallah, "A low-overhead soft–hard fault-tolerant architecture, design and management scheme for reliable high-performance many-core 3D-NoC systems," The Journal of Supercomputing, vol. 73, no. 6, pp. 2705–2729, 2017.
[61] C. Grecu et al., "BIST for network-on-chip interconnect infrastructures," in 24th IEEE VLSI Test Symposium. IEEE, 2006, pp. 6–pp.
[62] E. Cota et al., "A high-fault-coverage approach for the test of data, control and handshake interconnects in mesh Networks-on-Chip," IEEE Transactions on Computers, vol. 57, no. 9, pp. 1202–1215, Sep. 2008.
[63] R. Ubar and J. Raik, "Testing strategies for networks on chip," in Networks on Chip. Springer, Jan. 2003.
[64] A. Ghofrani et al., "Comprehensive online defect diagnosis in on-chip networks," in Proc. IEEE 30th VLSI Test Symp. (VTS), Apr. 2012, pp. 44–49.
[65] B. Aghaei, "A high fault coverage test approach for communication channels in network on chip," Microelectronics Reliability, vol. 75, pp. 178 – 186, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026271417302950
[66] D. Das and N. A. Touba, "A low cost approach for detecting, locating, and avoiding interconnect faults in FPGA-based reconfigurable systems," in Proceedings Twelfth International Conference on VLSI Design.(Cat. No. PR00013). IEEE, 1999, pp. 266–269.

[67] T. Ni, Y. Yao, H. Chang, L. Lu, H. Liang, A. Yan, Z. Huang, and X. Wen, "LCHR-TSV: Novel Low Cost and Highly Repairable Honeycomb-Based TSV Redundancy Architecture for Clustered Faults," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019.