An on-communication multiple-TSV defects detection and localization for real-time 3D-ICs

Khanh N. Dang*‡, Akram Ben Ahmed†, and Xuan-Tu Tran*
*SISLAB, University of Engineering and Technology, Vietnam National University Hanoi, Hanoi, 123106, Vietnam
†Department of Information and Computer Science, Keio University, Yokohama, 223-8522, Japan
‡Adaptive Systems Laboratory, The University of Aizu, Aizu-Wakamatsu, Fukushima 965-8580, Japan.
Email: khanh.n.dang@vnu.edu.vn; khanh@u-aizu.ac.jp

***Abstract*—This paper presents "On Communication Through-Silicon-Via Test" (*OCTT*), an ECC-based method to localize faults without halting the operation of TSV-based 3D-IC systems. OCTT consists of two major parts named *Statistical Detector* and *Isolation and Check*. While *Statistical Detector* could detect open and short defects in TSVs that work without interrupting data transactions, the *Isolation and Check* algorithm enhances the ability to localize fault position. The Monte-Carlo simulations of *Statistical Detector* show $\times 2$ increment in the number of detected faults when compared to conventional ECC-based techniques. While *Isolation and Check* helps localize the number of defects up to $\times 4$ and $\times 5$ higher. In addition, the worst case execution time is below 65,000 cycles with no performance degradation for testing which could be easily integrated into real-time applications.**

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

I. Introduction

Thanks to the extremely short lengths and low latencies of Through-Silicon-Vias (TSVs), TSV-based 3D-ICs could offer high speeds of communication [1]. However, due to their low yield rates [2], vulnerability to thermal [3], [4] and stress, and the cross-talk issues [5], [6], reliability is one of major concern of TSVs. Also, TSVs are susceptible to Electron-Migration (EM) [7] and the crosstalk challenge [8], [9]. Moreover, different materials usually have different thermal expansion coefficients while the layers' temperatures also vary causing stress issues that may crack the TSV during operation. 3D-ICs is also more challenging in terms of fault rate acceleration than traditional 2D-ICs because of their difficulty in thermal dissipation [3], [4].

Among existing method for detecting and correcting faulty TSVs, there are three major phases: detection, localization, and recovery. Using a Built-in-self-test (BIST) [10], [11] or an external testing [12], [13] are common for testing and localizing defects. Error Correction Codes (ECCs) [14] or dedicated circuits [15]–[17] also support detecting and correcting faults. For recovery, there are several approaches such as *hardware fault-tolerance* (i.e., correction circuits [16], redundancies [18], reliability mapping [6]), *information redundancy* (i.e., coding techniques [8], [14], [19] or re-transmission request [20]) and *algorithm-based fault-tolerance* (i.e., fault-tolerant routing [21], [22], run-time repair [7] or remapping [18], [23]).

Although numerous methods have been proposed to solve the reliability issues of TSVs, we can observe that on-line detection is still one of the major challenges for safety-critical applications which require short response times to newly occurred faults. The system could perform a testing process periodically using *BIST* [11], [24] (*Periodic BIST*: P-BIST) to reuse the existing test infrastructure, or external testing [12], [13]. Using ECC can provide a short response time while not blocking the communications. However, *P-BIST* usually has long response time that is not suitable for real-time applications and *ECCs* usually has low detection/localization rate.

To solve those problems, this paper proposes a short response time fault detection and localization method working in-parallel with the system operation. In order to have a better response time to new faults while not interrupting the system operation, we use *OCT* (On-communication Test) [25]. To solve the low coverage of *OCT*, we propose the "On Communication Through-Silicon-Via Test" (*OCTT*) methodology as follows:

- A comprehensive *OCT* set of algorithms and architectures based on two phases to improve the coverage: *Statistical Detection* and *Isolation-and-Check*.
- *Statistical Detection* mark the potential fault positions based on the outputs of *ECC* decoder within a certain number of transactions. Here, we use Parity Product Code (PPC) [26] which is one-bit correction Orthogonal Latin Square Code with an extra bit as the baseline. The Monte-Carlo simulation shows that *Statistical Detection* helps to localize 100% of two faults despite the limitation of one fault localization of PPC.
- The *Isolation-and-Check* enhances the localization ability of based on the suspicious position from *Statistical Detection*. Our evaluations show that *Statistical Detection* can detect 100% of 5 defects cases.

The organization of this paper is as follows: Section II presents the proposed detection and localization method. Section III evaluates the proposed algorithms and architectures. Then, Section IV concludes the paper.

II. Proposed Defect Detection and Localization

This section presents the proposed defect detection and localization method. First, the general testing accuracy is pre-

sented. Then, we introduce the two parts of *OCTT*: *Statistical Detector* and *Isolation-and-Check*.

A. Testing accuracy

Table I show four basic terminologies for detection and localization accuracy. Even suspicious TSVs could still be used for data transmission in *OCT*, the *false positive* cases are not critical in terms of reliability. Meanwhile, *false negative* is the most problematic issue because the system works under unknown defects without any awareness.

Table I: Detection and localization cases.

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

B. Parity Product Code

Parity Product Code (PPC) is based on the Product-Code [27] with the column and row parity check bits.

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

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

where

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

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

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

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

The outputs of Eq. 2 are two arrays of column check (sc) and row check (sr). If there is one or no flipped bit, the decoder can correct it using a $(N+1) \times (M+1)$ mask matrix m where

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

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

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

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

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

3) Correctability and Detectability: As we previously shown, PPC can correct one and detect two flipped bit. If there are more than two flipped bits, PPC also has chances to detect them using Eq. 3. However, the 2+ faults detectability is limited due to the potential hidden patterns. For instance, if bits with indexes (i,j), (i,k) and (l,j)[1] are flipped, they result both cr_i and sc_j are '0'. Therefore, the decoder fails to detect while both cc_k and sr_l could be '1'. This syndrome makes the decoder understand that there is one fault and correct the bit $b_{l,k}$.

C. Statistical Detector

1) Hidden error effect: One of the natural behaviors of an open and short defect is its inconsistency on flipping bit. If a TSV has a short-to-substrate defect and transmits a '0' value, there is no error on the receiver. On the other hand, transmitting a value '1' via short-to-substrate TSV causes flipped bit. If a timing violation occurs due to an open defect, sending the same value as the last transmitted value causes no errors while sending a different value may cause a flipped bit. Further study in open and short defect on TSVs could be founded in [15].

In summary, a TSV region with N defects is likely to have less than or equal to N faults at the same time.

2) Statistical Detector algorithm: Because of the inconsistency of open and short defects, we exploit the chance that the hidden fault can reduce the number of affected TSVs.

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

In this work, because of *Greedy localization* , the *false positive* cases could happen; however they are not critical issues for system reliability. Those *false positive* could be remove by using a dedicated testing later.

Although yielding a *BIST* could possibly solve the *false positive* cases, this may not be suitable for real-time systems

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

Algorithm 1: OCTT algorithm.

```
   // Column Check (CC) and Row Check (RC)
   Input: CC[1:N], RC[1:M]
   // Threshold for Localization
   Input: Thres_Loc
   // Fault indexes
   Output: Fault[1:N][1:M]
   // Statistical Detection function
1  Function Statistical_Detector(CC,RC,Thres_Loc) is
2      return Fault;
3      Fault[1:N][1:M] = 0;
4      for (i = 1; i <= N; i++) do
5          for (j = 1; i <= M; i++) do
6              if and CC[i] == 1 and RC[j] == 1 then
7                  Fault[1:N][1:M]++;
8      for (i = 1; i <= N; i++) do
9          for (j = 1; i <= M; i++) do
10             if Fault[i][j] >= Thres_Loc then
11                 Fault[i][j] = 1;
12             else
13                 Fault[i][j] = 0;
   // Isolation and Check
14 Statistical_Detector();
15 Isolation[1:N][1:M] = Fault[1:N][1:M]
   // Un-isolate each position and recheck the
      second time
16 for (i = 1; i <= N; i++) do
17     for (j = 1; i <= M; i++) do
18         if Fault[i][j] == 1 then
19             Isolation[i][j] = 0;
20             TempFault[1:N][1:M] = Statistical_Detector();
21             if TempFault[1:N][1:M] == 0 then
                   // not a faulty position
22                 Fault[i][j] = 0;
23             else
24                 Isolation[i][j] = 1;
```

because of critical tasks are still running. Even when the test could be executed, because *Statistical Detector* operates independently, there will be a case when multiple TSV regions having faults while parallel testing is not available. In either cases, Here, we observe that the system could further enhance the result of *Statistical Detector*. By improving the detection or localization rate, the system can eliminate the need of dedicated testing while ensuring the reliability of the system.

D. Isolation and Check

Isolation and Check, illustrated in Algorithm 1 lines 14-25, is used to solve both *false positive* and *false negative* cases. Because dedicated tester may not be approachable, the *Isolation and Check* method targets to solve this issue based on re-using PPC. The proposed algorithm follows these steps:

- **Step 1:** Using *Statistical Detector* to detect the fault position. Theses locations are considered as *suspicious TSVs*.
- **Step 2:** The system isolates the *suspicious TSVs* by detaching them from encoding/decoding but it keeps using the faulty TSV for data transaction.
- **Step 3:** Re-run the **Step 1** to **Step 3** until no fault is detected or out of time (until deadline).
- **Step 4:** Reassign each isolated TSV. The TSV could be re-attached to the encoding and decoding process. If a dedicated test is available, using it could reduce the testing time.
- **Step 5:** If the TSV region is detected as faulty, the faults are not localized by *Isolation and Check*. However, by repeating the *Isolation and Check* itself, higher coverage could be obtained.

By disabling all *suspicious TSVs* and re-running the *Statistical Detector*, the system can localize more faults.

III. Evaluation

A. Evaluation Methodology

OCTT was designed in Verilog-HDL, synthesized and prototyped with commercial CAD tools. The libraries are NANGATE 45nm [28] and NCSU FreePDK TSV [29]. The TSV size, pitch and Keep-out Zone are $4.06\mu m \times 4.06\mu m$, 10 μm, and 15 μm, respectively.

We first evaluate the detection and localization rate of *Statistical Detector*. The performance of *Isolate and Check* is presented later. Four different data widths: 8, 16, 32 and 64 are used together with the several numbers of transactions T= 8, 16, 32, 64 and 128. The Monte-Carlo simulation with 100,000 random samples is used for each case of these evaluations. The worst case execution time of the *OCTT* is also evaluated. Later, we compare the hardware implementation of the design with the well-known ECCs then with *BIST* and other testing methods.

B. Statistical Detector Performance

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

Even without *false positive* cases, the localization rate of the *Statistical Detect* easily outperforms the baseline ECC (Parity Product Code), as depicted in Fig. 2. *OCTT* still guarantees at least 80% of two faults in the worst case.

C. Isolation and check performance

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

As shown in Fig. 3, we can observe that *Isolation-and-Check* helps guarantee that healthy TSVs being not labeled as defected. While isolation may give mixing results between various T values, re-checking gives a more reasonable result where higher T values give better results.

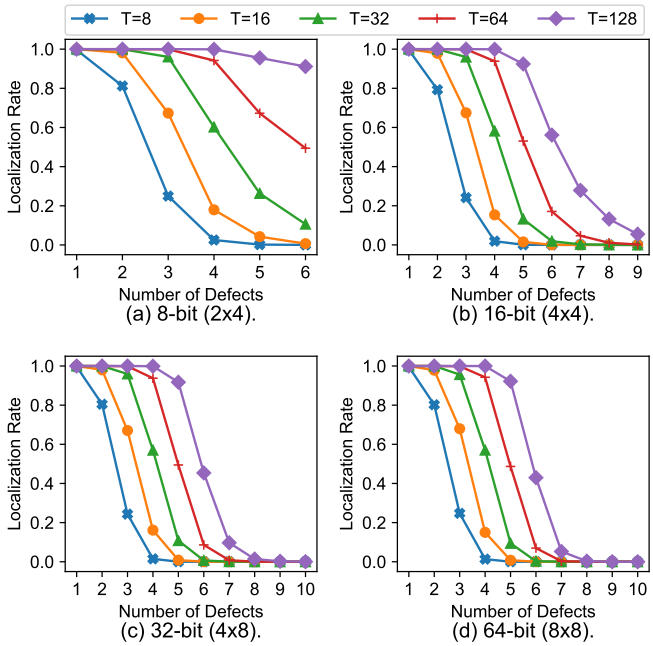

Figure 1: Result of standalone Statistical Detector (including *false positive*).

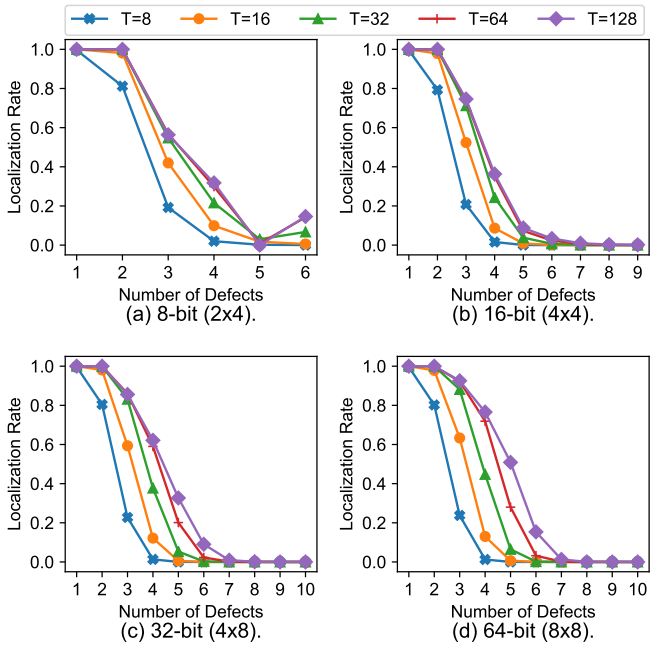

Figure 2: Result of standalone Statistical Detector (excluding *false positive*).

As shown in Fig. 3, 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 two 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,

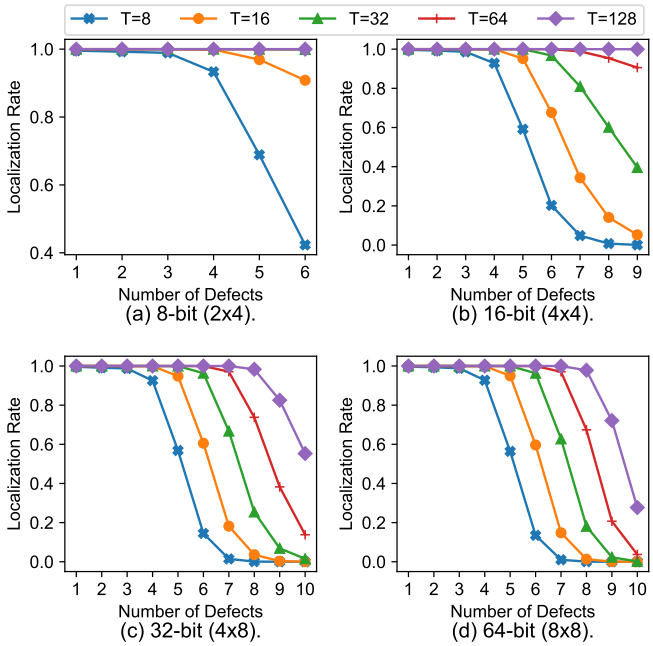

Figure 3: Result of *OCTT* (excluding *false positive*).

these results also imply a significant improvement of using *Statistical Detector*. As long as the system keeps sending flits via faulty TSVs, the *Statistical Detector* can detect them. In this Monte-Carlo simulation, we observe that the system can detect 100% of two faults up to $T = 32$ with a very short response time (it requires 32 cycles to complete 32 transactions). On the other hand, we observe that increasing the T value from 64 to 128 does not significantly improve the overall localization rate.

D. Response time

In this section, we evaluate the worst case execution time (WCET) of the proposed methods. The response time is considered as the time from the occurrence of a new fault until its detection (including *false positive* cases). Figures 4,show the WCET results. Note that each transaction is assumed to cost one clock cycle. A dummy value (for instance, [32] uses zeros and ones vector to test) could be used to test if the connection is free. For WCET, depicted in Fig. 4, we can observe the smaller T values give lower maximum response times. 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.

E. Hardware Implementation

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

Table II: Hardware implementation results.

Scheme		Tech. (nm)	k (bit)	n (bit)	Area Cost (μm^2)		Latency (ns)		Power (μW)	
					Encoder	Decoder	Encoder	Decoder	Encoder	Decoder
Hamming [14]		45	32	39	94.1640	234.8780	0.55	1.12	30.0831	96.2898
SECDED [19]		45	32	40	111.7200	253.7640	0.60	1.44	36.9622	103.1422
SEC-DAEC [30][1]		45	32	39	322	1902	0.53	1.33	-	-
TAEC [31][1]		45	32	40	264	2628	0.45	1.32	-	-
PPC(4×8)		45	32	45	76.6080	187.2640	0.30	0.68	43.0272	129.4174
OCTT	Total	45	32	45	130.3400	2161.2500	0.39	0.72	48.639	1.04×10^3
	PPC(4×8)	45	32	45	130.3400	327.4460	-	-	48.639	198.424
	Stat_det	45	32	45	-	751.1840	-	-	-	408.621
	Isol_Check	45	32	45	-	1016.3860	-	-	-	387.056

[1] We use the area optimization and lowest area cost design since our design is optimized for area cost.

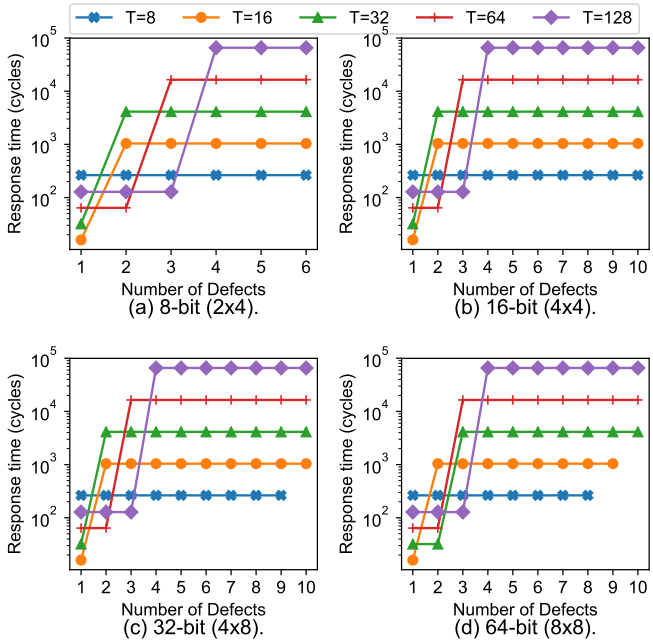

Figure 4: WCET of the OCTT.

ability of PPC instead of providing alternative error correction coding methods.

In the case of the encoder, the complexity of our method is lower than SEC-DAEC (Single Error Correction, Double Adjacent Error Correction) and TAEC (Triple Adjacent Error Correction) [30], [31] but higher than Hamming and SECDED. The area cost of the proposed decoder is higher due to the fact the system requires registers to collectively store the output syndrome. However, for a delay optimized design, we offer smaller design than multiple errors correction. It is worth mentioning that these techniques only correct adjacent errors. Also, our latency is smaller thanks to the short critical path where PPC (4×8) only calculates the parity for 8-bit instead of 32-bit. Our decoder area is also higher than the Hamming and SECDED and similar to TAEC and SEC-DAEC methods with area optimization. For latency optimization, our area cost is smaller than all multiple error corrections.

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

IV. Conclusion

This paper presents a method to improve the localization rate of Parity Product Code (PPC) to enhance the reliability of TSV-based 3D-IC designs. From the conducted experiments, and in contrast to conventional PPC that are limited to localizing one fault at most, *OCTT* has demonstrated its ability to localize more than six faults. Furthermore, *OCTT*'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 *OCTT* to a dedicated application together with soft and hard fault tolerance to obtain a comprehensive method. Leaving defected TSV run without recovery in approximate computing is also an interesting topic. Extending the technique with adaptive coding and different based coding methods is another possible direction.

Acknowledgment

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

References

[1] 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.

[2] 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.

[3] 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.

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

[5] 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.

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

[7] J. Wang *et al.*, "Efficient design-for-test approach for networks-on-chip," *IEEE Trans. Comput.*, 2018.

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

[9] A. Eghbal *et al.*, "Analytical fault tolerance assessment and metrics for TSV-based 3D network-on-chip," *IEEE Trans. Comput.*, vol. 64, no. 12, pp. 3591–3604, 2015.

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

[11] 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.

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

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

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

[15] 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.

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

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

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

[25] K. N. Dang, A. B. Ahmed, A. B. Abdallah, and X. T. Tran, "Tsv-ias: Analytic analysis and low-cost non-preemptive on-line detection and correction method for tsv defects," in *The IEEE Symposium on VLSI (ISVLSI) 2019*, 2019.

[26] K. N. Dang and X. T. Tran, "Parity-based ECC and Mechanism for Detecting and Correcting Soft Errors in On-Chip Communication," in *IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, 2018.

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

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

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

[30] 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.

[31] 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.

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