Novel Variability Aware Path Selection for Self-Referencing Based Hardware Trojan Detection

Vaikuntapu Ramakrishna, Lava Bhargava, Vineet Sahula
Department of Electronics and Communication Engineering
Malaviya National Institute of Technology, Jaipur-302017, India
vramakrishna409@gmail.com, lavab@mnit.ac.in, sahula@ieee.org

***Abstract*—Hardware Trojans can be inserted by an adversary at untrusted third-party fabrication houses. Many Side-channel analysis based detection techniques have been proposed in past to detect such Trojans. However, their efficiency is highly affected by process variations. Hardware Trojan (HT) inserted in an IC affects the path delays within the IC. In this work, we exploit the fact that the path delays of topologically symmetric paths in an IC will be affected similarly by process variations. We tend to choose paths that are minimally affected by process variations. In this paper, we propose a path selection technique for delay based HT detection technique. We further use the concept of self-referencing to improve detection accuracy as well as to eliminate the requirement of golden ICs. Simulations performed using ISCAS-85 benchmarks establish that the proposed method is achieving a true positive rate of 100% with a false positive rate less than 5%. We have considered maximum of 10% intra-die and 15% inter-die variations in threshold voltage (V_{th}).**

***Index Terms*—Hardware Trojan detection, Self-referencing, Process variation, Path delay.**

I. Introduction

Hardware Trojan (HT) can be introduced by an adversary at an untrusted design or fabrication house. Depending on the interests of the adversary the HT can cause change in functionality, denial-of-service, information leakage or reliability reduction. The detailed taxonomy of hardware Trojans at various abstraction levels was presented in [1]. Different methods have been presented for detecting Trojans inserted at different abstraction levels [2]. Post-silicon detection methods aim to detect HTs in fabricated ICs inserted at untrusted fabrication foundries. Even though logic testing [3] based approaches for HT detection have been proposed their efficiency is limited by the stealthy nature of HT, test generation and execution time. Majority of existing post-silicon methods are devised based on side channel analysis (SCA) i.e analyzing the side channel parameters like power, path delay and electromagnetic measurements etc. Many existing SCA based HT detection techniques depend on golden ICs (i.e ICs without any HTs) used as a reference to compare the side-channel (SC) signatures of suspect IC to decide whether HTs exist in the IC under purview. Obtaining such golden ICs is expensive as it involves invasive techniques such as reverse engineering. Another challenge for SCA methods is the process variation (PV) which impacts the efficiency of detection even if golden ICs are available. In presence of PV direct comparison of the SC signatures of suspect chip with the golden chips may not accurately detect HTs. To mitigate the effects of PVs on SC parameters researchers have proposed a technique known as Self-referencing [4]. In self-referencing based methods the SC parameter of an IC is compared with the SC parameter of the IC itself instead of comparing with golden signature. The major advantages of self-referencing methods are, requirement of golden IC is eliminated and the effect of inter-die process variation are mitigated as it affects the entire chip in a similar fashion. In this work, we chose path delay for Trojan detection because of its specific advantages such as, HT effect is local to the path and HT affects path delay irrespective of its activation status which eliminates the need to activate Trojans. Following are the contributions of this paper:

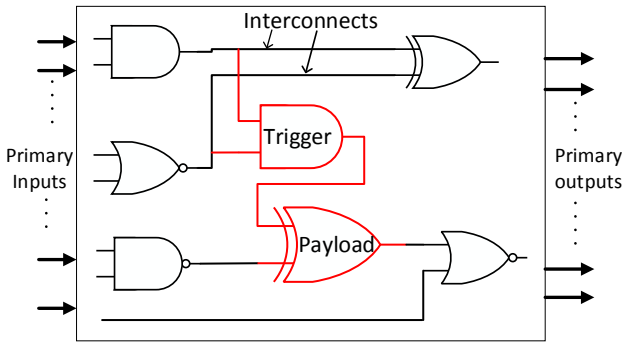

Fig. 1. Hardware Trojan structure

- It presents a methodology to detect hardware Trojans (HTs) inserted during fabrication. This method uses the concept of self-referencing and the delays of symmetric paths are analysed to detect Trojans.
- It proposes a path selection algorithm to select symmetric path pairs in such a way to mitigate the effect of inter-die and intra-die process variations on detection accuracy. It also presents a procedure to create a symmetric path pair if it does not exist in the design.

This paper is organized as follows, HT's effect on path delay, potential locations of HTs and brief overview of existing HT detection methods are presented in Section II. Section III explains the proposed methodology in detail and path selection procedure is explained in Section IV. Simulation results are reported in Section V and we conclude in section VI.

II. Background and Related Work

A. HT impact on path delay

HTs can impact the delay of paths in a circuit in two ways. A gate that acts as Trojan trigger connects to existing interconnects in the circuit and activates the HT when specific activation condition is met as shown in Fig. 1. These extra connections induce some additional capacitance at the interconnects which in turn impacts delay. The gate that acts as Trojan payload is inserted by restitching a net in the original circuit (see Fig. 1) and launches its attack when the HT is activated. An additional delay equal to the delay of inserted payload gate is added to the delay of paths passing through the restitched net in the original circuit. The additional delay introduced by payload gate is typically larger than that of trigger gate.

B. Potential locations of HT

An attacker may choose to insert HT in a way that it should be stealthy i.e undetectable by both logic testing and side-channel based detection techniques. To meet such requirements, it is normally assumed that attacker attempts to insert the HT at nodes of the circuit which exhibit very low switching activity. The reason behind this assumption is Trojans inserted at such nodes are expected to show minimal impact in logic-testing and side-channel based detection techniques [5]. We have considered low switching nodes/nets as possible and vulnerable locations for HT insertion.

C. Process Variations

In general HTs are small in size in comparison with the entire design, hence their effect on SC parameters may be masked by the effects of PVs. The inter-die variation affects all devices in an IC in a similar fashion, whereas, intra-die variation affects these devices differently depending on their locations on the IC [6]. In general inter-die variations are larger than intra-die variations [7]. We have considered variations in transistor threshold voltage (V_{th}). The variation model considered in this work is shown in eq. (1) and (2).

$$v_{th}(x,y) = v_{th,nom} + \Delta v_{inter} + \Delta v_{intra}(x,y) \quad (1)$$

The intra-die component $\Delta v_{intra}(x,y)$ can further be divided as a correlated spatial component and a random component as shown in eq. (2)

$$\Delta v_{intra}(x,y) = \Delta v_{spatial}(x,y) + \Delta v_{random}(x,y) \quad (2)$$

where $v_{th,nom}$ is the nominal threshold voltage, Δv_{inter} is the inter-die variation which which is same at all locations, $\Delta v_{spatial}(x,y)$ is a spatially correlated component which can be modeled as a correlated multivariate Gaussian random variable. $\Delta v_{random}(x,y)$ is the random component which can be modeled as an independent Gaussian random variable.

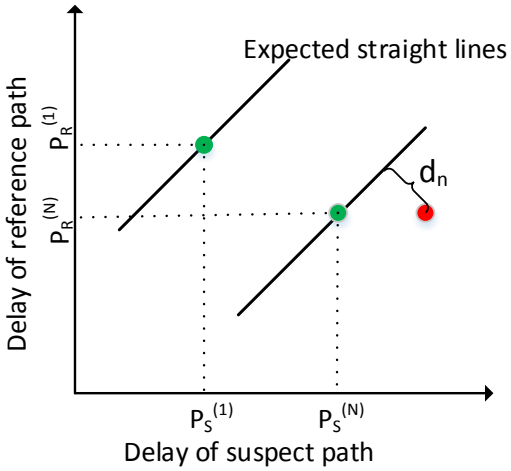

Fig. 2. Deviation from expected straight line

D. Related Work

The detailed taxonomy and classification of HTs is presented in [1]. To detect HTs, SCA based detection techniques have been considered to be more efficient than traditional functionality testing methods. For the first time, Agarwal et al. proposed to use side channel parameter (power) for Trojan detection [8] by constructing fingerprint using Golden-ICs. We present the work available in literature on detecting HT similar to the proposed work i.e. path delay analysis based HT detection. Path delays were used to generate the fingerprint of an IC family using principle component analysis in [9]. The limitations of these methods are they require a set of golden ICs and their detection accuracy is largely affected by process variations. In [10], the authors used shadow registers to measure path delays and to detect HTs and this method requires multiple clocks and requires extra registers inserted for each selected path. Efficiency of delay-based techniques to detect HTs in the presence of process variations was studied in [11]. In [12] authors proposed an embedded test structure by using existing scan structures to measure path delays so as to detect HTs. In [13], authors proposed to choose the shortest paths through each possible Trojan location to improve HT detection accuracy in the presence of process variations. A clock glitching method was proposed to measure path delays and statistical techniques are used to reduce the effects of process variations in [14]. In [15] authors proposed a clock sweeping based path delay measurement technique for detecting HTs. A self-referencing based golden-IC free HT detection method is proposed in [16]. It compares the delays of symmetric paths in a design. In [17] a pulse propagation technique is proposed to detect extra capacitance induced by HT on logical paths. A high resolution on-chip embedded test structure called a time-to-digital converter to measure path delays and a chip-averaging technique was proposed to detect delay anomalies introduced by HTs in [18]. Like any other delay based detection techniques, proposed method is also not able to detect Trojans that don't have any impact on path delays.

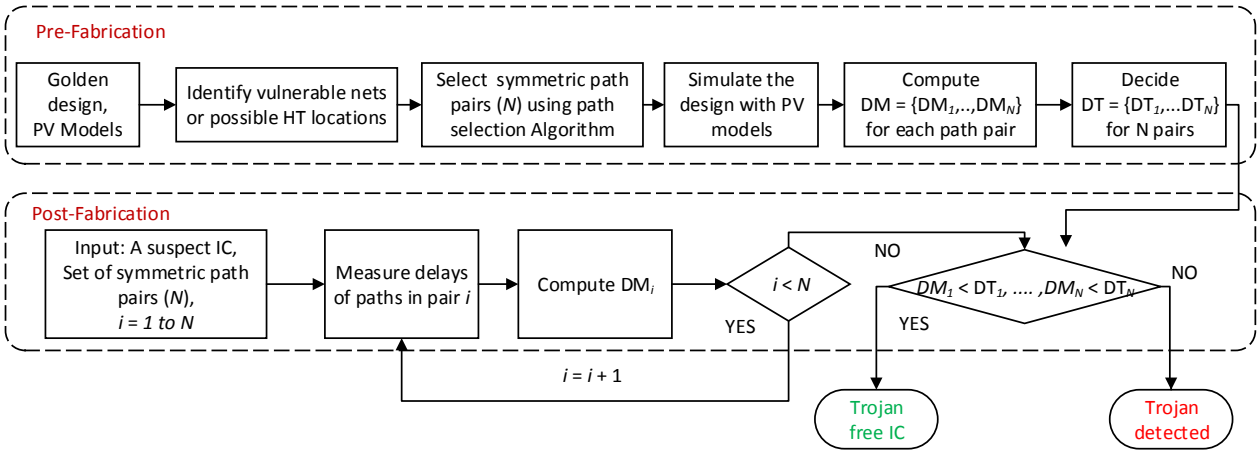

Fig. 3. Proposed Hardware Trojan Detection Methodology

III. Proposed Detection Methodology

In this section we present the proposed methodology in detail. Every IC contains many paths out of which some paths are topologically similar which we call as symmetric paths. Any two paths which have equal number of gates of same type can be defined as symmetric paths as shown in Fig.4. The design netlist has been analysed to find vulnerable nets which are considered to be potential locations of HT insertion. We chose a path through such vulnerable net which we call as suspect path and search for a symmetric path which is topologically similar which we call as reference path to form a suspect and reference path pair. The path delays of such paths may or may not be equal as it is dependent on the interconnection length and capacitive load of gates. As both the paths in a pair have similar gates they experience same inter-die process variation (global variation or die-to-die variation). We chose a symmetric path pair for each vulnerable net in the netlist. When the delays of such pairs are plotted on a two dimensional space they should follow strait lines corresponding to each pair as shown in Fig.2. If HT is inserted at a vulnerable net then the delay of suspect path will be increased by an amount of HT induced delay. Which in turn leads to some deviation from the expected straight line as shown in Fig. 2. This deviation has been used to calculate a detection metric (DM) for each selected pair in an IC under test. But, in the presence of intra-die process variations (local or within-die variations) the delays of pairs may deviate a little from the straight line. Therefore, this detection metric is compared with a pre-defined threshold (DT) to separate Trojan inserted ICs from Trojan free ICs. The proposed methodology consists two stages i.e (i) Pre-fabrication stage and (ii) Post-fabrication stage as shown in Fig. 3.

A. Pre-fabrication stage

We assume that a golden model of the design is available, as the HT is assumed to be inserted after design sign-off while fabrication. The golden design netlist is simulated using a large number of random test vectors to collect a set of vulnerable nets as $\{N_v\}$. The shortest path as reported in [13] has been chosen through each vulnerable net which we call as suspect path P_s. For each such suspect path, a topologically symmetric path is identified which we call as reference path P_r. The path selection algorithm which is used to select suspect and reference paths is explained in section IV. A set of symmetric path pairs $\{N\}$ is formed by combining a suspect path and its corresponding reference path through each net n in $\{N_v\}$. $P^i_{s,nom}$ and $P^i_{r,nom}$ represent the nominal path delays of suspect path and reference path respectively, of i^{th} symmetric path pair corresponding to i^{th} vulnerable net in $\{N_v\}$. The detection metric DM_i of i^{th} pair is calculated as the normalised distance [19] from expected straight line through the point $(P^i_{s,nom}, P^i_{r,nom})$ as shown in eq. (3).

$$DM_i = \frac{d_i}{\sqrt{(P^i_s)^2 + (P^i_r)^2}} \tag{3}$$

Here, the distance d_i is given by eq. (4).

$$d_i = \frac{1}{\sqrt{2}}(|P^i_s - P^i_r + (P^i_{s,nom} - P^i_{r,nom})|) \tag{4}$$

Monte-Carlo simulations are performed using reliable process variation models provided by the foundry to generate a set of Trojan free ICs. The $DM = \{DM_1, DM_2, ..., DM_N\}$ of all Trojan free ICs is calculated and a detection threshold is decided for each pair as a set $DT = \{DT_1, DT_2, ..., DT_N\}$.

B. Post-fabrication stage

The path delays of all $2N$ paths in N pairs are measured for each suspected IC. Techniques presented in [10], [15] can be used to measure the delays of selected paths. The DM of each pair is calculated using eq. (3). Each DM_i is compared with pre-defined detection threshold DT_i to infer whether the IC under test is Trojan free as shown below.

$$if \begin{cases} DM_1 < DT_1 \quad and \\ DM_2 < DT_2 \quad and \\ ... \\ DM_N < DT_N. \end{cases} \quad \text{then IC is Trojan free}$$

$$else \qquad \text{IC has Trojan inserted}$$

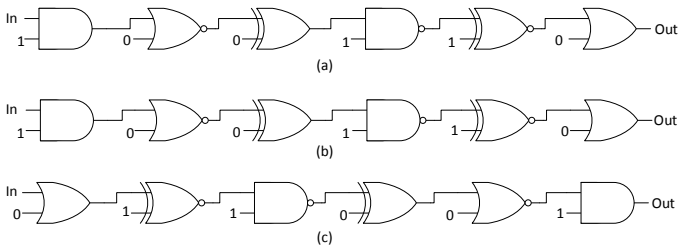

Fig. 4. Symmetric paths. (a) suspect path (b) type-1 reference path (c) type-2 reference path

IV. Path selection Procedure

A. Symmetric paths

In this work, we have considered two paths as symmetric if they pass through similar types of logic gates as shown in Fig. 4. Assume Fig. 4(a) shows a suspect path through a vulnerable net. The reference path for the suspect path in Fig. 4(a) can be anyone of the two paths shown in Fig. 4(b) and 4(c). We call path shown in Fig. 4(b) as type-1 reference path and path in Fig. 4(c) as type-2 reference path. As all the three paths in Fig. 4 have same number of logic gates of same type, they all experience same inter-die process variations [20]. The delays of the paths shown in Fig. 4 may or may not be same as they also depend on interconnection delays and capacitive load experienced by gates in the respective paths of the design post layout.

B. Path selection algorithm

The proposed path selection technique is discussed in this section and is presented in Algorihm 1. This algorithm returns a suspect path if any, and its corresponding reference path for each vulnerable net in the design. We call a path with minimum delay as shortest path. All paths passing through a vulnerble net n_i are collected in set $S_{suspect}$. A shortest sensitizable path p_{short} is chosen from the set $S_{suspect}$. All paths that are symmetric to p_{short} are collected in set $S_{symmetric}$. If there are no symmetric paths available for p_{short} then next shortest path in $S_{suspect}$ is selected as p_{short}. If there are any nets for which there exist no symmetric path for at least one path in $S_{suspect}$ then a symmetric path is created as explained in Section IV-C. After finding out suspect path p_{susp} for each vulnerable net and its corresponding set of symmetric paths, final layout of the netlist is prepared. This layout is partitioned into 64 grids as 8 rows and 8 columns. The purpose of partitioning the layout is to exploit spatial correlation component in intra-die variation among the adjacent grids. We rank each path in $S_{symmetric}$ according to the distance between grids that consist same gates as shown in Fig.5. A path in $S_{symmetric}$ which is near (i.e with lowest rank) to p_{susp} is selected as reference path p_{ref}. A symmetric path pair $SymmetricPpair_i$ corresponding to net n_i is formed as (p_{susp}, p_{ref}). We used Synopsys Tetramax ATPG tool [21] to decide whether a path is sensitizable.

Algorithm 1: Path selection algorithm

Input : Netlist, N_v = set of vulnerable nets
Output: Symmetric path pairs

1 **foreach** *net* n_i *in* N_v **do**
2 $S_{suspect,i}$ = all paths passing through net n_i
3 $p_{short,i}$ = a sensitizable path with minimum delay in $S_{suspect,i}$
4 $S_{symmetric,i}$ = set of paths that are symmetric(type-1 or type-2) to $P_{short,i}$ which are not passing through net n_i
5 **if** $|S_{symmetric,i}| = 0$ **then**
/* If there are no symmetric paths for $p_{short,i}$ then select next minimum delay path */
6 $S_{suspect,i}$ = $S_{suspect,i}$ - $p_{short,i}$
7 go to: step 3
8 **end**
9 **if** $\sum_{j=1}^{|S_{suspect,i}|} |S_{symmetric,j}| = 0$ **then**
/* If there are no symmetric paths for at least one path in $S_{suspect,i}$ then create a symmetric path */
10 $p_{short,i}$ = a sensitizable path with minimum delay in $S_{suspect,i}$
11 p_{sub} = a path that has a subset of total gates in the path $p_{short,i}$
12 $extragates$ = $gatesin(p_{short,i}) - gatesin(p_{sub})$
13 $p_{ref,i}$ = path p_{sub} with $extragates$ inserted at its input
14 Connect off-path inputs of $extragates$ to non-controlling values
15 **end**
16 Generate the layout of (modified) netlist and partition it into 8×8 grids
17 $p_{susp,i} = p_{short,i}$
18 $p_{ref,i}$ = a path from $S_{symmetric,i}$ which is nearest to $p_{susp,i}$ i.e the path passing through grids near to that of $p_{susp,i}$ i.e path with lowest rank
19 $SymmetricPpair_i \leftarrow (p_{susp,i}, p_{ref,i})$
20 **end**

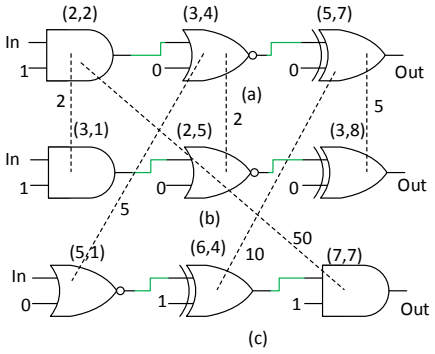

Fig. 5. Example for ranking symmetric paths and locations of gates in $8X8$ grids (a) suspect path (b) reference path with rank = 9 (2+2+5) (c) reference path with rank = 65 (50+5+10)

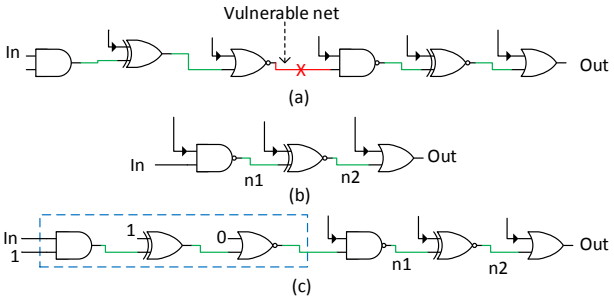

Fig. 6. Generating symmetric path (a) selected suspect path (b) a non-critical path (c) generated symmetric path to cover the vulnerable net

C. Reference path creation

There may be some vulnerable nets for which a symmetric pair does not exist, we call such nets as uncovered nets. In order cover such nets, we create a symmetric path to be used as reference path by adding few extra logic gates to the existing non-critical paths as shown in Fig. 6. We chose the shortest path through the uncovered net and search for a non-critical path which has gates that are subset of gates of selected path through the uncovered net. The remaining gates are inserted at the input of the non-critical path without changing the functionality to create a type-1 or type-2 symmetric path to be used as a reference path. The off-path inputs of extra inserted gates must be set to non-controlling values. The designer has to ensure that all paths through the input net of the created new path must be non-critical even after the addition of extra gates and functionality must remain same. The extra overhead because of adding extra gates is dependent on the number of uncovered nets and the size of selected suspect and reference paths.

V. Results

Simulations are performed on ISCAS-85 benchmark circuits [22] to evaluate proposed methodology. These benchmarks are synthesized using Synopsys generic (SAED_EDK32nm) library. To perform SPICE level simulations 32nm CMOS Predictive Technology Models (PTM) [23] have been used. The netlists are simulated over a large number of random test vectors (i.e. $10e6$) and nets with switching activity less than predefined threshold ($10e-3$) are considered as vulnerable nets as explained in section II-B. Static timing analysis (STA) is performed using Synopsys primetime [24] to generate path data and Tetramax ATPG [21] is used to decide whether a path is sensitizable. Symmetric paths are created to cover the uncovered nets as explained in section IV-C. Physical layout of design is carried out using Synopsys IC Compiler and layout is partitioned into 8×8 grids. Symmetric path pairs are selected using path selection procedure explained in section IV-B. Selected paths are extracted in SPICE netlist form. Process variation on threshold voltage (V_{th}) is modeled as explained in section II-C. 3σ values for inter-die and intra-die variation are considered as 15% and 10% respectively. As the layout has been partitioned into 8×8 grids, 64 correlated multivariate random variables are generated to model the spatial variation.

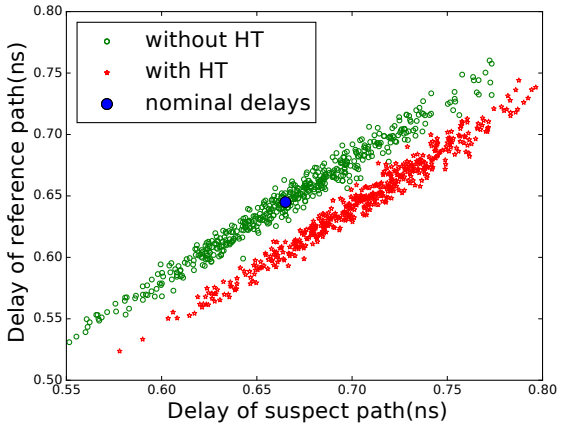

Fig. 7. Effect PV on a symmetric path pair and variation along expected straight line

Spatial correlation decreases with increase in distance between grids. 500 ICs without HT and 500 ICs with HT (shown in Fig. 1) are generated and SPICE level simulations are performed with the generated V_{th} profiles using HSPICE. The variation of delays of suspect and reference paths in a symmetric path pair of c880 circuit is shown in Fig. 7. The path delays vary along a straight line passing through the point of nominal delays due to inter-die variations, however, due to intra-die variations they deviate away from the straight line. It is observed that the delays in HT inserted ICs also follows a straight line but they are away from the expected straight line. Detection metric (DM) of all trojan free and with trojan ICs is calculated as per eq. (3) and shown in Fig. 8. True positive rate (TPR) is defined as number of ICs that are detected as HT inserted out of total HT inserted ICs. False positive rate (FPR) is defined as the number of ICs that are detected as HT inserted but which in fact are HT free. A TPR of 100% and a FPR of 3% is observed with a DT of 0.01. The detection threshold (DT) has been decided in a pessimistic way by allowing a false positive rate of 5% but, row 7 in table-I shows the minimum FPR that can be achieved for the TPR shown in row 6. Table-I shows the summary of simulation results. Row 2 shows total number of nets present in the circuit and row 3 presents the number of low-switching nets considered as vulnerable nets. Nets which do not have reference paths i.e uncovered nets are shown in row 4 and row 5 shows the hardware overhead (i.e number of gates) due to extra gates inserted to create a reference path for each uncovered net. Rows 6 and 7 show the TPR and FPR achieved by the proposed detection method. It is observed that maximum TPR possible is 95% even with 18% FPR for c432. It is due to the unavailability of shortest symmetric paths through the net n6 . For this net the delays of selected symmetric paths are higher than 75% of critical path delay. Even though paths are longer, the performance of the method would have been better if the symmetric paths are closer. But, in this case these paths pass through less spatially correlated grids. One way to improve the performance is to create a shorter symmetric path closer to any one of the paths through net n6.

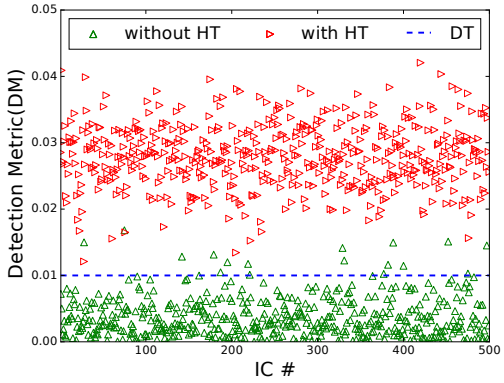

Fig. 8. Detection metric of 500 HT free and 500 HT inserted ICs for a symmetric path pair of c880

TABLE I
SIMULATION RESULTS ON ISCAS-85 BENCHMARKS

Circuit	c432	c499	c880	c1355	c1908	c2670	c5315
Total nets	120	200	238	229	398	152	734
Vulnerable nets	9	42	17	40	16	22	8
Uncovered nets	2	0	3	0	7	0	3
Overhead†	4	0	11	0	22	0	9
TPR(%)	95	99	100	100	100	98	100
FPR(%)	18	5	3	3	2	5	1

† In number of extra gates inserted.

VI. CONCLUSION

We have considered threat model of HT insertion while fabrication at untrusted foundries. In this work, we presented a self-referencing based HT detection method using path delays which eliminates the requirement of golden ICs. Further, we proposed a procedure to select paths that minimizes the effect of both inter-die and intra-die PV. We have used topologically symmetric paths to mitigate inter-die variations and selected closer paths to exploit the spatial correlation to reduce the impact of intra-die variations. This method uses simulation results to decide DT, on-chip PV monitors can be used to estimate the DT of each suspect IC which will be focused in our future work.

ACKNOWLEDGEMENT

This work is partly supported by SMDPC2SD project, sponsored by Ministry of Electronics & Information of Technology, Government of India.

REFERENCES

[1] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design and Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[2] H. Li, Q. Liu, J. Zhang, and Y. Lyu, "A Survey of Hardware Trojan Detection, Diagnosis and Prevention," in *2015 14th International Conference on Computer-Aided Design and Computer Graphics (CAD/Graphics)*, 2015, pp. 173–180.

[3] A. G. Voyiatzis, K. G. Stefanidis, and P. Kitsos, "Efficient Triggering of Trojan Hardware Logic," in *2016 IEEE 19th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, 2016, pp. 200–205.

[4] D. Du, S. Narasimhan, R. S. Chakraborty, and S. Bhunia, "Self-referencing: A Scalable Side-channel Approach for Hardware Trojan Detection," in *Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems*, vol. CHES'10, 2010, pp. 173–187.

[5] H. Salmani, M. M. Tehranipoor, and J. Plusquellic, "New Design Strategy for Improving Hardware Trojan Detection and Reducing Trojan Activation Time," in *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, 2009, pp. 66–73.

[6] C. Hongliang and S. S. Sapatnekar, "Statistical Timing Analysis Considering Spatial Correlations Using a Single PERT-like Traversal," in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486)*, 2003, pp. 621–625.

[7] X. Chen, L. Wang, Y. Wang, Y. Liu, and H. Yang, "A General Framework for Hardware Trojan Detection in Digital Circuits by Statistical Learning Algorithms," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 0070, no. c, pp. 1–1, 2016.

[8] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection using IC Fingerprinting," in *Proceedings - IEEE Symposium on Security and Privacy*, 2007, pp. 296–310.

[9] Y. Yier, Jin and Makris, "Hardware Trojan Detection Using Path Delay Fingerprint," in *Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, 2008, pp. 51–57.

[10] J. Li and J. Lach, "At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust, HOST*, 2008, pp. 8–14.

[11] J. Rai, D. and Lach, "Performance of Delay-Based Trojan Detection Techniques under Parameter Variations," in *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, 2009, pp. 58–65.

[12] C. Lamech and J. Plusquellic, "Trojan Detection based on Delay Variations Measured using a High-Precision, Low-Overhead Embedded Test Structure," in *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*, 2012, pp. 75–82.

[13] S. K. Cha, Byeongju and Gupta, "Trojan Detection via Delay Measurements: A New Approach to Select Paths and Vectors to Maximize Effectiveness and Minimize Cost," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, 2013, pp. 1265–1270.

[14] I. Exurville, L. Zussa, J.-b. Rigaud, and B. Robisson, "Resilient Hardware Trojans Detection based on Path Delay Measurements," in *Hardware Oriented Security and Trust (HOST), 2015 IEEE International Symposium on*, 2015, pp. 151–156.

[15] K. Xiao, X. Zhang, and M. Tehranipoor, "A Clock Sweeping Technique for Detecting Hardware Trojans Impacting Circuits Delay," *IEEE Design & Test*, vol. 30, no. 2, pp. 26–34, 2013.

[16] N. Yoshimizu, "Hardware Trojan Detection By Symmetry Breaking In Path Delays," in *IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014*, 2014, pp. 107–111.

[17] S. Deyati, B. J. Muldrey, and A. Chatterjee, "Trojan Detection in Digital Systems Using Current Sensing of Pulse Propagation in Logic Gates," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*, 2016, pp. 350–355.

[18] D. Ismari, J. Plusquellic, C. Lamech, S. Bhunia, and F. Saqib, "On Ddetecting Delay Anomalies Introduced by Hardware Trojans," in *Proceedings of the 35th International Conference on Computer-Aided Design - ICCAD '16*, 2016, pp. 1–7.

[19] V. Ramakrishna, L. Bhargava, and V. Sahula, "Golden IC free Methodology for Hardware Trojan Detection using Symmetric Path Delays," in *VLSI Design and Test (VDAT), 2016 20th International Symposium on, IEEE*, 2016, pp. 1–2.

[20] I. Sutherland, B. Sproull, and D. Harris, *Logical Effort: Designing Fast CMOS Circuits*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[21] *TetraMAX ATPG User Guide, Version L-2016.03-SP2*, Synopsys, Inc.

[22] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in Fortran," in *Proceedings of IEEE Int'l Symposium Circuits and Systems (ISCAS 85)*. IEEE Press, Piscataway, N.J., 1985, pp. 677–692.

[23] (2014) Predictive technology model. [Online]. Available: http://ptm.asu.edu/

[24] *PrimeTime User Guide, Version L-2016.06, June 2016*, Synopsys, Inc.