A New Monitor Insertion Algorithm for Intermittent Fault Detection

Hassan Ebrahimi and Hans G. Kerkhoff
Testable Design and Test of Integrated Systems (TDT) Group,
University of Twente, Enschede, the Netherlands
{h.ebrahimi, h.g.kerkhoff}@utwente.nl

***Abstract*—The dependability of highly dependable systems relies on the reliability of its components and interconnections. One of the most challenging faults that threatens the reliability of interconnections in a system are intermittent resistive faults (IRFs). They may occur randomly in time, duration and amplitude in every interconnection. The occurrence rate can vary from a few nanoseconds to months. As a result, evoking and detecting such faults is a major challenge. In this paper, IRF detection at the chip level has been tackled by utilising a fully digital in-situ IRF monitor. This paper introduces a new algorithm for inserting IRF monitors in a design. The goal of this algorithm is to minimise the number of IRF monitors while providing a high fault coverage for IRFs. The algorithm has been validated using software-based fault injection. The simulation results show that the proposed algorithm improves the IRF coverage at the chip level at the cost of a small area and power-consumption overhead.**

***Index Terms*—Reliability, No Faults Found, Intermittent Resistive Faults, Intermittent Fault Detection, Chip-level and board-level fault detection**

I. Introduction

The continuous shrinking of the minimum feature size together with the ever increasing growth in complexity makes the reliability of electronic integrated systems a major design challenge. The complexity of electronic systems grows rapidly with having more logic elements in a smaller area. This leads to more layers of interconnections and thinner interconnects in a system. As interconnections are yet extremely dominant, their reliability is becoming increasingly important. One of the key interconnect reliability challenges that threaten highly dependable systems are intermittent resistive faults (IRFs).

The most common causes of IRFs are marginal or unstable interconnections. The interconnections at both board and chip levels such as tracks, vias, and solder joints are susceptible to IRFs. In addition, temperature and mechanical stress, electromigration, and corrosion cause increased instability in interconnections. IRFs manifest themselves as a sequence of low-level resistance changes in an interconnection. This can lead to a timing error or performance degradation in a system.

During the operational mode of a system, IRFs might occur intermittently in any interconnect at any time. Besides, their occurrence rate at a location may gradually increase and become increasingly severe during the lifetime of the system. Ultimately, they may evolve into a permanent fault [1]. Therefore, it is vital to detect IRFs before they become permanent and result in a system failure especially in safety-critical systems.

Intermittent-fault detection is difficult. Due to the nondeterministic behaviour of intermittent faults, the probability that an IRF is activated while in test mode is very low. Therefore, conventional test methods are highly unlikely to detect these faults. Two alternative methods for intermittent fault detection are periodic testing [2] and in-situ on-line monitoring [3]. The periodic testing method is basically retesting the system periodically which increases the probability of detecting intermittent faults. The on-line monitoring method is to monitor the health of a system using embedded instruments during the operational mode. Since IRFs may remain inactive during test phases, the in-situ on-line monitoring technique is a better option to detect IRFs in the case they become active while a system is in operational mode.

A selection algorithm is required to find the locations in a system to be monitored. Using the IRF monitor for all interconnects in a system is not practical because of the extra cost of the area and power-consumption overhead. Therefore, in order to have the maximum coverage of IRFs using the on-line monitors, an efficient location selection algorithm is required. In this paper, we propose an efficient selection algorithm for IRF monitoring which provides the maximum IRF coverage for a given area overhead constraint. The efficiency of the algorithm is verified using software-based fault injections.

The rest of the paper is organised as follows. Section II reviews related work and the background of the on-line monitor selection and insertion algorithm. The IRF monitor which is used by the proposed algorithm is introduced in section III. In section IV, the proposed algorithm is described. The simulation setup and simulation results are presented in sections V and VI, respectively. Finally, conclusions are drawn in section VII.

II. Related work

In-situ on-line monitoring is widely used to detect timing-faults [4]–[6] at the chip level. Timing violation along a data path can occur due to process, voltage and temperature variations, aging and IRF [7]. In-situ on-line monitoring techniques can be used for timing-error detection and correction in critical paths [4], timing slack measurements [6], adaptive voltage scaling [8], aging detection [9] and IRF detection as well [3].

978-1-7281-4312-5/20/$31.00 ©2020 IEEE

There are several works that have investigated the path selection approach for online aging and process variation detection. In [9] a critical-path selection algorithm has been proposed for dynamic frequency scaling under BTI-aging and process variations effects. A path-selection flow for online aging monitoring in a reconfigurable architecture has been presented in [10]. Their algorithm first selects the aging-prone path based on path delay, temperature, duty cycle, and switching activity. Then, the selected paths get pruned based on fan-out, and physical location of the path endpoints.

Few works have investigated the aging-monitor insertion at internal nodes of the circuit [5], [11]. The motivation behind is to capture timing violations as early as possible before they become masked. The proposed method in [5] inserts timing-slack monitors as probes at the endpoints and internal nodes of critical paths.

In [11], the authors have proposed a procedure for selecting the monitoring nodes only among the internal nodes of the circuit. This increases the accuracy of age-monitoring mechanisms and also decreases the number of nodes which results in a low hardware overhead.

The techniques mentioned above are based on the detection of delay violations which can be caused by aging or process variation based on monitoring the critical path(s) and near-critical path(s). The measurement of paths delays, temperature and switching activity play main roles in the aging estimation. Since the critical path and near critical paths are more prone to aging effects, they are good candidates for aging monitoring. Despite the aging effects, IRFs can happen randomly in any path of a circuit and can result in timing violation at the path endpoint. Therefore, the critical path and near-critical paths are not necessarily the best candidates for IRF detection.

In this paper, a new node-selection algorithm is proposed. The objective of this algorithm is to maximise the IRF coverage of a circuit. The algorithm selects the best locations for IRF monitoring in a circuit. It maximises the fault coverage for the most vulnerable parts of the circuit to IRFs such as wire interconnections, vias and input pads. Two different methods based on end nodes selection as well as internal nodes selection have been investigated. The experimental results show the efficiency of the proposed algorithm.

III. The IRF Monitor

The IRF monitor used in this paper is based on the monitor which has been introduced in [12] with minor changes. Fig. 1 shows how the IRF monitor can be used at the end of a data path. The output of the monitor is a *Warning* signal which indicates whether an IRF is detected or not.

To make sure that the correct data at an endpoint will be captured, the data signal at the end of a data path should stay stable during a timing window; otherwise the flip-flop at the endpoint may capture wrong data. The IRF monitor checks whether or not a late transition in the data path occurs. The timing window can be generated using a guard-band signal or employ delay elements. In this monitor, the timing window is created by a delay chain.

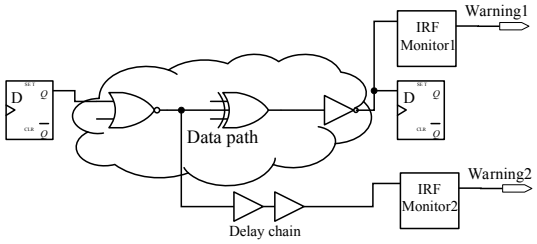

Fig. 1: An example of the insertion of IRF monitors at the endpoint and an internal node of a data path.

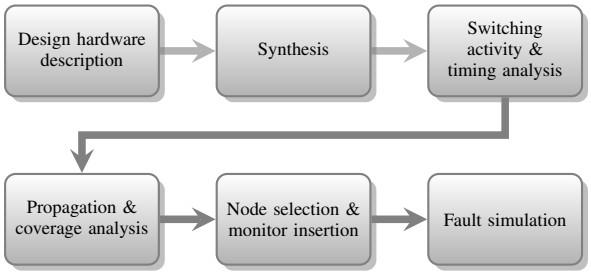

Fig. 2: The flow of the proposed node selection and the IRF monitor insertion.

IV. The Endpoint Selection Flow and the Proposed Algorithm

In this section, the nodes selection flow and the proposed algorithm are introduced and explained in detail.

A. The selection flow

The flow of the proposed node selection and the IRF monitor insertion is depicted in Fig. 2. After synthesising a given design, the switching activity of each net is extracted. The static timing analysis reveals the slack of paths, and consequently the critical paths and near critical paths of the fresh design. In order to avoid performance degradation, the proposed algorithm tries to not insert any monitor in the critical paths and near critical paths.

In the next phase, the probability that an IRF is being propagated and being captured by an monitor is calculated for each net. The result of these calculations will be a set of nets for every (internal and end) node. Each set contains all nets that can be covered by inserting a monitor at the corresponding node. The nodes that cover more nets with higher fault probability are good candidates for monitor insertion.

Based on the extracted information in the previous phases, the proposed algorithm selects a set of nodes which provide a maximum IRF fault coverage within a given area-overhead constraint. Finally, the IRF monitor will be inserted in the selected nodes.

Synopsys tools are used during the synthesis phase and performing timing analysis. The fault injection and simulation have been executed using the QuestaSim tool. The switching activity is extracted for every benchmark using the QuestaSim

tool. The other phases of the flow have been implemented using Python scripts.

The fault-propagation probability calculation and the proposed selection algorithm are explained in detail below.

B. Fault propagation probability

The fault propagation strongly depends on the circuit structure and input stimuli. IRFs can occur in every net (wire interconnects and vias) of the design. IRFs may cause timing delays in the affected net. The induced delay fault may be either masked by a logic gate in the path or may be propagated via the path and being captured by a flip-flop at the end of the path and cause a logical error. Therefore, to know which locations are suitable for monitoring IRFs, the fault propagation probability for each net of the circuit should be calculated. This information will be used by the proposed selection algorithm to select nodes which can cover the nets that have a high probability of fault propagation. It should be noted that the probability that an IRF occurs in a net could vary based on the layout of the design after the place and route phase. The vulnerability of each net to IRFs depends on the number of wire interconnects and vias it consists of. The information on the layout can be used for extra refinement later. Since, this information is not available at the synthesis phase, we have estimated the number of wire interconnects and vias based on the fan-out number for each net. Our experimental results show the accuracy of this estimation.

An example of the calculation of the propagation probability for a simple circuit is shown in Fig. 3. This Fig. shows the calculation of the propagation probability for the net x. As can be seen, the signal x can be propagated via two paths. One is shown by the blue line and another with the red line. The signal probability of each nets are written on top of the net. The propagation probability of each gate is written underneath it. In this example, the signal probability for the input nets are assumed to be 0.5. In the red path, the probability that the signal x propagates via the NAND gate is 0.5. After the NAND gate, the signal x can be propagated via the NOT and XOR gate without being masked. In the case of blue path, the propagation probability of NOR gate is 0.25 (1 - 0.75). Therefore, the probability that a fault on x is propagated to a flip-flop (either or both FF1 and FF2) is 0.5 + 0.25 - (0.5 * 0.25) = 0.625. Since, the propagation probability of the red path is higher than the blue one, the flip-flop FF1 is a better choice for IRF monitoring.

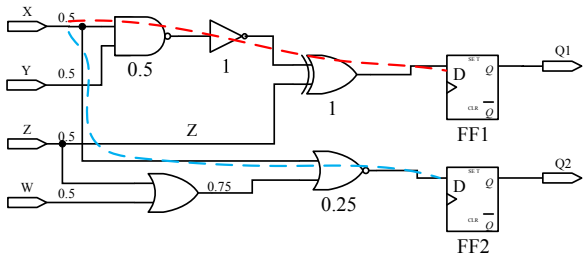

Fig. 3: An example of the calculation of the propagation probability

C. The proposed selection algorithm

The proposed node selection algorithm is shown in the Algorithm 1. The inputs of this algorithm are a graph of netlist G, an area constraints A and the timing data T. The netlist is a directed graph. In this graph, edges are nets and vertices are either logic gates or nets. The timing data T is the output of the synthesis tool which contains the information on the switching activity of each net. The output of the algorithm is a set of selected nodes S which will be used for the IRF monitor insertion. The proposed algorithm consists of three main functions and a while loop.

Algorithm 1 Selection algorithm

Input: G; A; T
Output: S ={Selected nodes}

$P \leftarrow Propagation\_and\_Coverage\_Analysis(G)$
while Area_overhead $<$ A **do**
  $e \leftarrow Find\_best\_node(T, P)$
  Add e to S set
  Area_overhead $\leftarrow Estimate\_area\_overhead(S)$
function FIND_BEST_NODE(T, P)
  E={all nodes or endpoints in the netlist G}
  for all $e \in E$ **do**
    $varFaultCoverage \leftarrow Fault\_coverage(P, e)$
    $varMaximum \leftarrow 0$
    $objective \leftarrow \alpha\ (varFaultCoverage)$
    if $objective > varMaximum$ **then**
      $varMaximum \leftarrow objective$
      $SelectedNode \leftarrow e$
    Remove_nets_from_search_space(SelectedNode)
  return $SelectedNode$
function PROPAGATION_AND_COVERAGE_ANALYSIS(G)
  N ={all nets in the netlist G} ▷ Set of all nets
  V ={} ▷ Set of visited nets
  for all $n \in N$ **do**
    if $n \notin V$ **then**
      $L \leftarrow DFS(G, V, n)$
  $P \leftarrow Create\_list\_of\_nets\_for\_each\_node(L)$
  return P
function DFS(G, V, n)
  F ={Flip-flops and Outputs}
  $S \leftarrow Extract\_direct\_successors(n)$
  for all $s \in S$ **do**
    if $s \notin V$ & $s \notin F$ **then**
      Calculate the propagation probability for s
      $DFS(G, V, s)$
    else
      Update the list of nodes for net n
  return $L = \{the\ list\ of\ nodes\ for\ net\ n\}$

The function *Propagation_and_Coverage_Analysis* is used to calculate the fault-propagation probability for all nets of the

netlist (lines 17-24). This function receives the graph of the netlist G as an input. The output of this function is a list for each endpoint/internal node of the circuit. The lists contain a set of nets that can be covered by the corresponding endpoint/internal node. For every net, the propagation probability is calculated using the equations that have been presented in the previous subsection. This function uses a depth-first search (DFS) to traverse the netlist. The DFS function will be called for every net of the netlist ($n \in N$) only if the net is not processed before ($n \notin V$). The output of the DFS function is a list of nodes (L) for net n. From these lists, a list of nets (P) will be extracted for each node. List P allows to know which nets are connected to a certain node and how much is the value of the propagation probability for each net. It should be noted that the correlation between paths due to reconvergent fanout [13] are considered in this function. This provides a more accurate and less pessimistic fault-propagation probability calculation.

A pseudocode of the function *DFS* is presented in the lines 25-34. This function is a recursive function and calls itself until it reaches either an endpoint or till all nets of the netlist have been visited. The function calculates the propagation probability for each net while traversing it (line 30). This information will be returned to e the function *Propagation_and_Coverage_Analysis* as a list L.

The function *Find_Best_Node* receives the information of timing (T) and the coverage and propagation probability of nets (P) as inputs. The value of IRF coverage (*varFaultCoverage*) will be calculated for each node using the following equation:

$$IRF\_Coverage = \sum_{n=1}^{N} P_{F_n} \tag{1}$$

where N is the number of nets which is covered by the given node. The variable P_{F_n} is the fault-propagation probability for each net n which can be calculated from equation 2.

$$P_{F_i} = \prod_{g=1}^{n} P_{T_g} \tag{2}$$

where P_{T_g} is the propagation probability for a gate g and n is the number of gates in the path. This equation indicates that an IRF in a net i can be propagated via a path and being captured by an endpoint only in the case that it is able to pass via all the gates in the path.

The objective of the function *Find_Best_Node* is to find a node which has the maximum value for the IRF fault coverage. At the end of the function calculation, the covered net by the selected node will be removed from the search space.

Using the above mentioned functions, the algorithm in a while loop selects a set of nodes (endpoints or internal nodes) with maximum fault coverage. The algorithm estimates the amount of area that is required for the IRF monitoring and continues its selection procedure till the area overhead reaches the given limit A. If the area overhead is not a concern but instead the amount of total IRF fault coverage is important, the condition of the while loop can be easily substituted to satisfy this new objective.

TABLE I: Range of used uniform distributed parameters in the IRF generator during fault emulation

Parameter	Minimum	Maximum
Start time	1 ns	20 ns
Delay	10 ps	500 ps
Active time	0.1 ns	12 ns
Inactive time	0.1 ns	12 ns
Burst length	1	10
Safe time	1 ns	1 ms

V. Simulation Setup

A. The Intermittent Resistive Fault Model

The IRF model being used for the fault injection is based on our and others' experiences in practice. In this model, a burst of IRF pulses is generated based on six parameters. These parameters consist of the number of pulses in a burst, start and stop times of the burst, active and inactive times, and the amount of induced delay for each pulse.

The values and distributions applied for the fault injection in this paper are shown in Table I. The fault injection starts when a random *Start Time* is passed from the simulation, then the fault injection starts with injecting a burst of pulses. Each pulse in the burst has a random delay value Δt. Each pulse remains active during a random activation time (*Active Time*) and after that stays inactive for a random time (*Inactive Time*). The delay pulse generation continues until the same number of pulses as the burst length are produced. At the end, a fault-free situation will occur which is called *Safe Time*.

The IRF model has been implemented in a Python script. The script randomly generates IRFs based on the parameter of Table I. The output of the script is a TCL file that will be the input for the fault-injection procedure.

B. Simulation-based fault injection

The well-known QuestaSim tool was used for the fault-simulation experiments. The tool receives the procedure of random IRF generation via a TCl file. The technique of saboteur [14] is used for the fault injection. It means a HDL component named as saboteur is defined for IRF injection. This component was inserted in every net of the benchmark which were selected for fault injection. A saboteur is inactive during fault-free operation, but when it becomes active it injects a random IRF in the selected net.

VI. Simulation Results

All experiments were performed on a Linux system with 8 Intel 2 GHz cores and with 16GB RAM. The execution time of the proposed selection algorithm is shown in Table II. In this table, the execution for two different methods, internal nodes and endpoints selection, are shown. Five of the largest benchmarks of ISCAS'89 and an AES-128 encoder [15] were used for our experiments. The benchmarks have been synthesized using the Synopsys Design Compiler tool

TABLE II: The measured execution time [Sec] of the proposed algorithm.

Benchmarks	Internal nodes	Endpoints
s13207	1,79	1,60
s15850	2,44	2,15
s35932	2,82	2,39
s38417	7,42	6,22
s38584	10,64	9,10
AES-128	20,23	16,77

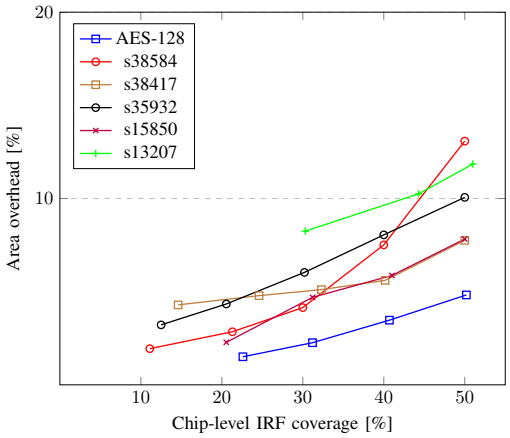

Fig. 4: Area overhead vs. fault coverage

with TSMC 40nm technology. After a performance-driven synthesis the large benchmarks s38584, s38417 and AES-128 have been optimised for a clock of 1.25 GHz, and the smaller benchmarks have been optimised for a clock of 1.66 GHz.

Table III gives an insight into the structure of the benchmarks such as the number of inputs, outputs, sequential and combinational cells. In addition, the areas for combinational and sequential parts are presented at a scale of square micrometers.

TABLE III: Synthesis reports for the benchmarks (Area $[\mu m^2]$)

Benchmarks	AES	s38584	s38417	s35932	s15850	s13207
Inputs	36	38	28	35	77	62
Outputs	38	317	106	320	150	152
Comb. cells	4619	5772	4163	2445	1885	1379
Seq. cells	910	1275	1564	1728	513	625
Comb. area	24458	23633	17131	12045	6732	4973
Seq. area	12556	16782	20214	920	6805	7760

The results of the proposed algorithm for the benchmarks are shown in Fig. 4. For each benchmark, the result of the area overhead against fault coverage of IRFs is depicted. For example, the result of benchmark s35932 is drawn with a black line. It shows the algorithm can provide 20%, 40% fault coverage at the cost of 4.3% and 8% area overheads, respectively. In the case of s38584, the same fault coverage can be obtained with 2.8% and 7.5% area overheads, respectively.

In Fig. 4, the fault coverage for each benchmark starts with a different number because the algorithm first starts with providing full board-level coverage. The reason is that input ports are more susceptible to IRFs than internal vias and interconnections. For example, covering all inputs of S13207 in the first stage of the algorithm results in 30 % fault coverage. This fault coverage includes full input coverage and partial internal connections coverage. Another example is the fault coverage for the AES-128 benchmark. As shown in Fig. 5, a full input coverage can be reached by only 9 monitors with 22% total IRF coverage.

Fig. 5 shows an implementation of the proposed insertion flow for the AES-128 benchmark. In this Fig., the location and the number of IRF monitors are chosen by the proposed algorithm. In total a number of 9 monitors has been selected, to provide full IRF coverage for all input ports. The IRF monitors are coloured in red colour. The input ports, vias and interconnection wires which are covered by the monitors are coloured in green. The output, clk and reset ports are coloured in yellow.

The comparison between the proposed algorithm and a slack-based method such as [5], is shown for the benchmark s38417 in Fig. 6. In this Fig., the results of fault coverage for different area overheads are shown. For example, it depicts that the proposed algorithm can reach a fault coverage of IRFs of about 50% with an area overhead of 7.7%. Whereas, for the slack-based algorithm the area overhead is 11.6%. In total, the proposed method provides the same amount of fault coverage with about 30% less area overhead in comparison to the slack-based method.

The correlation between the number fan-outs for each net and the number of its vias after placing and routing is shown in Fig. 7. As can be seen, there is a linear relation between the amount of fan-outs and the required vias for each net. As mentioned before, the proposed algorithm uses an accurate estimation of vias to calculate the IRF probability for each net.

To evaluate the effectiveness of the proposed algorithm for IRF detection, a simulation-based fault injection was used. AES-128 benchmark was selected for fault injection. A post-synthesis simulation has been performed using the QuestaSim

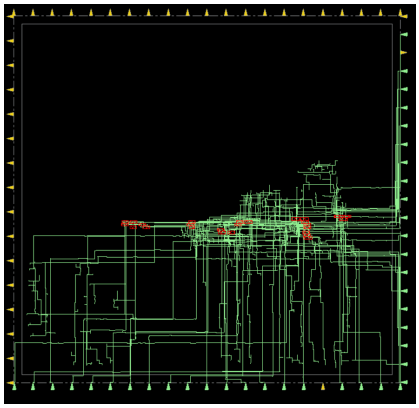

Fig. 5: An implementation of the proposed insertion flow for the AES-128 benchmark. The cells coloured in red colour are the inserted IRF monitors at selected internal and end nodes.

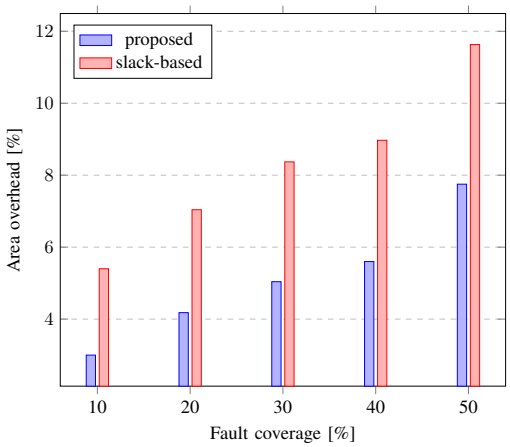

Fig. 6: Comparison of the area overhead for a slack-based approach and the proposed algorithm for benchmark s38417.

TABLE IV: Fault injection results

Fault impact	Masked	Detected	Undetected	Failed
Fault percentage	32.5%	38.2%	21.8%	7.3%

tool. The algorithm was executed on the benchmark with a 20% area constraint. A total number of 10000 IRFs have been injected on 1000 nets which were randomly selected from the benchmark. The results are shown in Table IV. As can be seen, about 32.5% of injected faults have been masked and have not propagated to an endpoint. From the rest of the faults, 38.2% have been detected by IRF monitors. Around 21.8% of faults have caused a slack reduction in the endpoint which were not selected for IRF monitoring and therefore remained undetected. Finally, 7.3% of the injected faults lead to a failure.

The maximum fault coverage for end-node and internal node methods is on average 80% and 95%, respectively. However, full IRFs coverage is not feasible. Therefore, the algorithm tries to maximise the IRF coverage for connections that are more susceptible to IRFs i.e. the input ports and the nets with a high number of vias.

VII. Conclusions

In this paper, a node selection-algorithm has been proposed to monitor IRFs in a system. The algorithm finds the best locations for IRF monitoring based on information of the propagation probability for each net of the system. It can provide a set of endpoints or internal nodes for inserting the IRF monitors. The simulation results show that the proposed algorithm can improve the fault coverage for IRFs of a system at the cost of only a small area and power-consumption overhead.

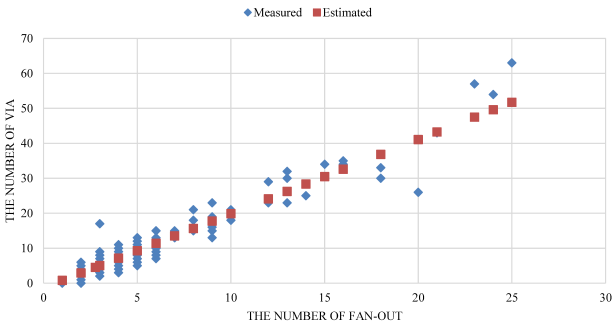

Fig. 7: Comparison of the number of vias.

Acknowledgement

This research was carried out within the EU-PENTA project "HADES", financed by the European Commission (EC) and the Netherlands Enterprise Agency (RVO).

References

[1] J. P. Hofmeister, P. Lall, D. Panchagade, N. N. Roth, T. A. Tracy, J. B. Judkins, and K. L. Harris, "Ball grid array (BGA) solder joint intermittency detection: SJ BIST," in *IEEE Aerospace Conference*, pp. 1–11, 2008.

[2] N. Kranitis, A. Merentitis, N. Laoutaris, G. Theodorou, A. Paschalis, D. Gizopoulos, and C. Halatsis, "Optimal periodic testing of intermittent faults in embedded pipelined processor applications," in *IEEE Design, Automation and Test in Europe (DATE)*, vol. 1, pp. 1–6, 2006.

[3] H. Ebrahimi, A. Rohani, and H. G. Kerkhoff, "Detecting intermittent resistive faults in digital CMOS circuits," in *IEEE Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 87–90, 2016.

[4] S. Das, C. Tokunaga, S. Pant, W. Ma, S. Kalaiselvan, K. Lai, D. M. Bull, and D. T. Blaauw, "RazorII: In situ error detection and correction for pvt and ser tolerance," *in IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, 2009.

[5] L. Lai, V. Chandra, R. C. Aitken, and P. Gupta, "SlackProbe: A flexible and efficient in-situ timing slack monitoring methodology," *in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 8, pp. 1168–1179, 2014.

[6] M. Sadi, L. Winemberg, and M. Tehranipoor, "A robust digital sensor IP and sensor insertion flow for in-situ path timing slack monitoring in SoCs," in *IEEE VLSI Test Symposium (VTS)*, pp. 1–6, 2015.

[7] H. G. Kerkhoff and H. Ebrahimi, "Investigation of intermittent resistive faults in digital CMOS circuits," *in World Scientific Journal of circuits, systems and computers*, vol. 25, no. 03, p. 1640023, 2016.

[8] W. Shan, L. Shi, and J. Yang, "In-situ timing monitor-based adaptive voltage scaling system for wide-voltage-range applications," *in IEEE Access*, vol. 5, pp. 15831–15838, 2017.

[9] A. F. Gomez and V. Champac, "Selection of critical paths for reliable frequency scaling under bti-aging considering workload uncertainty and process variations effects," *in ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 23, no. 3, p. 27, 2018.

[10] M. Ebrahimi, Z. Ghaderi, E. Bozorgzadeh, and Z. Navabi, "Path selection and sensor insertion flow for age monitoring in fpgas," in *IEEE Design, Automation and Test in Europe (DATE)*, pp. 792–797, 2016.

[11] S. Sadeghi-Kohan, A. Vafaei, and Z. Navabi, "Near-optimal node selection procedure for aging monitor placement," in *IEEE International Symposium on On-Line Testing And Robust System Design (IOLTS)*, pp. 6–11, 2018.

[12] H. Ebrahimi and H. G. Kerkhoff, "Intermittent resistance fault detection at board level," in *IEEE Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, pp. 135–140, 2018.

[13] H. Jahanirad, "CC-SPRA: Correlation coefficients approach for signal probability-based reliability analysis," *in IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 4, pp. 927–939, 2019.

[14] D. Gil-Tomás, J. Gracia-Morán, J.-C. Baraza-Calvo, L.-J. Saiz-Adalid, and P.-J. Gil-Vicente, "Injecting intermittent faults for the dependability assessment of a fault-tolerant microcomputer system," *in IEEE Transactions on Reliability*, vol. 65, no. 2, pp. 648–661, 2015.

[15] Opencores, "Advanced encryption standard AES-128, available: http://www.opencores.org/," 2019.