Fault Coverage of a Test Set on Structure-Preserving Siblings of a Circuit-Under-Test

Manobendra Nath Mondal[1], Animesh Basak Chowdhury[1,3], Manjari Pradhan[1],
Susmita Sur-Kolay[1], and Bhargab B. Bhattacharya[1,2]

[1]Advanced Computing & Microelectronics Unit, Indian Statistical Institute, Kolkata, India
[2]Dept. of CSE, Indian Institute of Technology Kharagpur, India
[3]Centre for Cyber-Security, New York University, USA

***Abstract*—Most of the Automatic Test Pattern Generation (*ATPG*) algorithms for digital circuits rely heavily on netlist description that comprises both network interconnect structure among logic gates and the functionality of each gate. The performance of an *ATPG* tool on a circuit-under-test (*CUT*) C is determined by the size of the test set T and its fault coverage (*FC*). Despite extensive research in the field of testing, the following question remains unanswered: Is the structure or the functionality of C dominant in determining FC of a test-set T for C? In this paper, we present empirical evidence in favour of the dominance of structure on FC by randomly selecting a logic gate from a synthesized netlist for C, and replacing it by a different type of gate. Our experiments provide an un-intuitive result that FC of a test-set T for C under the single stuck-at fault model remains nearly the same on other sibling circuits that have identical structure as of C but with different gate functionality, provided these have similar extent of fault redundancy. This observation supports the view that feeding structural information alone may suffice to train machine-learning models that are currently being used to expedite different problems of digital circuit testing and diagnosis.**

***Keywords*—ATPG, Engineering Change Order (ECO), Fault Coverage (FC), Stuck-at Fault (SAF)**

I. Introduction

With rapidly increasing complexity of ICs, testing, validation and diagnosis pose serious challenges to test engineers. In order to overcome inherent complexities, various ML-based techniques have emerged for tackling a diverse set of problems in design validation and testing. Most of the state-of-the-art *ATPG* algorithms rely heavily on structural network of a circuit as well as on the functionality of each of its logic gates. Although such algorithms deploy efficient heuristics, their scalability with respect to time and resources for large sized circuits (# gates $> 10M$) continues to be a bottleneck. Moreover, in the light of incremental changes needed during small Engineering Change Order (ECO), re-generation of *ATPG* consumes a great deal of time-bandwidth.

Extensive work was done in the past on designing fault detection test sets purely from the functional description of digital circuits, provided their structural realizations satisfy certain constraints. As an example, generation of a complete test set for a special class of circuits called unate gate networks was described in [1]. Designing a Universal Test Set (UTS) for functionally invariant CUT was reported in [2], based on the functional description of blocks where the networks are independent. Recently, researchers have been leveraging structural information of digital circuits to feed machine-learning algorithms to explore various classical problems in VLSI testing. These problems previously required manual effort of experienced engineers having sheer domain expertise. In a couple of recent works [3, 4], the authors have used Support-Vector Regression and Graphical Convolution Networks to predict X-sensitivity during defect-diagnosis and intelligent test-point insertion to improve overall circuit testability, respectively. Both the techniques have mapped the circuit structure into a graph, extracted the structural features from the graph to train the model and then used it for their prediction problem. The authors have claimed high precision and recall score from the trained model. However, there is no underlying justification of such high accuracy as most of the features are taken solely from structural attributes of circuit graph. In this work, we study the variation of fault coverage of a given test-set obtained for a CUT on similar structure-preserving circuits. Our empirical evaluation on structurally-invariant and functionally-diverse circuits has revealed that the fault coverage of a test set is dominated mostly by structural properties of the underlying graph rather than functional differences of the gates present in the net-list.

In the rest of this paper, Section II has a motivational example in support of our hypothesis. Sections III and IV introduce the problem and propose the experimental set-up, respectively. Simulation results appear in Section V. Functional diversity of test-cases is discussed in Section VI and the conclusion in Section VII.

II. Motivation

Let us consider gate-level netlists for two circuits C_1 and C_2 whose structures are identical, whereas their functionalities are different, as shown in Fig. 1.

We run $ATPG$ for single stuck-at faults (SAF) on both C_1 and C_1 to obtain the respective test sets $TS(1)$ and $TS(2)$ with nine and eight vectors respectively. By fault simulation on C_1, we observe that $FC(1, 1) = 100\%$ and $FC(2, 1) = 86.36\%$. Similarly, for C_2, $FC(2, 2) = 91\%$ (due to redundant faults) and $FC(1, 2) = 86.36\%$ (vide Table I for the notation).

The above example motivates us to explore whether for a set of structure-preserving circuits, a test set obtained for any one of these provides satisfactory fault coverage for all the

978-1-7281-2695-1/19/$31.00©2019 IEEE

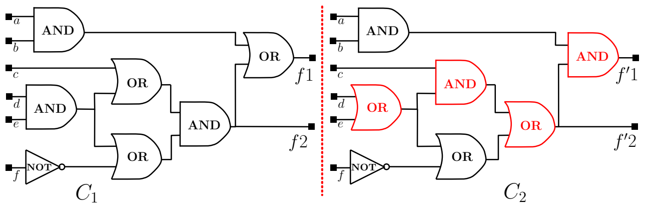

Fig. 1: C_2 is obtained from C_1 by replacing 2 of the AND gates by OR, and two of the OR gates by AND.

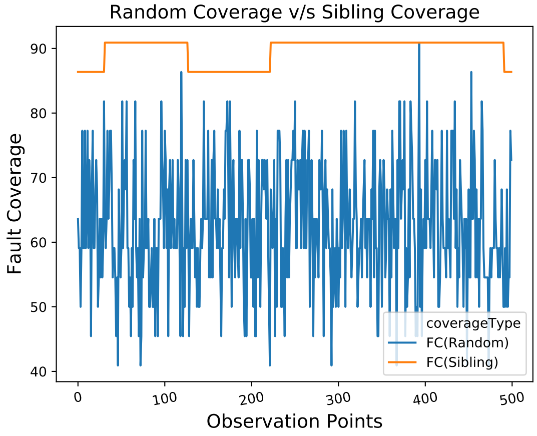

Fig. 2: Fault coverage by sibling test-set *vs*. random test-set for the motivating example in Fig. 1

other ones in the set. Further, we compare such fault coverage with that for a randomly generated test set of same size, as shown in Fig. 2 for the motivating example in Fig. 1. For each trial, we plot fault coverage on C_2 using a random set of test-vectors (*FC(random)*) versus that by a test-set generated for C_1 (*FC(sibling)*) by ATPG. The plot indicates that the fault coverage on C_2 by an ATPG generated test-set for C_1 (which may be differ with trials), dominates the fault coverage on C_2 by a randomly generated test-set.

Our hypothesis is that the gate-level structure of a logic circuit plays a major role in determining fault-coverage of a test set rather than the functionality of logic nodes and hence, similar coverage is expected on structure-preserving circuits when the same test-set is reused.

In this work, we address the following salient questions:

- Does the structural information of a circuit dominate the generation of a single SAF test-set by ATPG?
- To what extent the fault coverage of a test set for single SAFs is impacted when randomly selected gates are replaced by other gates, keeping the interconnecting structure unchanged?

III. Preliminaries

We first introduce a few terminologies used henceforth.

Dual Gate $D(g_i)$ of a boolean logic gate g_i is defined as:

- $D(AND) = OR$, and vice-versa.
- $D(NAND) = NOR$, and vice-versa.
- $D(1) = 0$, and vice-versa.

Sibling Circuit $S_i(C)$ of a given circuit C in terms of its gate-level netlist of a circuit C, is defined as the netlist obtained by replacing a fraction of its logic gates at randomly chosen locations, with their corresponding dual gates. $S_i(C)$ denotes the i^{th} sibling circuit of C.

TABLE I: Glossary of Notation

Description	Notation
Circuit-Under-Test (CUT)	C
Logic gate identifier in CUT	$g_i(C)$
i^{th} Sibling Circuit of CUT	$S_i(C)$
Set of Sibling Circuits of CUT	$\Pi(C)$
ATPG test-set for Sibling Circuit $S_i(C)$	$TS(i)$
Fault Coverage of $TS(i)$ on $S_j(C)$	$FC(i,j)$
Number of redundant, testable and overall faults in $S_i(C)$	ν_i^R, ν_i^T, ν_i^O
Redundancy Bin, a subset of Sibling circuits with ν_i^R greater than m and at most n	$R_{m,n}$
Normalized boolean difference between Sibling circuits $S_i(C)$ and $S_j(C)$	$\eta(i,j)$

Alpha (α) denotes the fraction of logic gates replaced by dual gates to obtain $S_i(C)$ from C.

Redundancy Bin $(R_{m,n})$ denotes a set of *sibling circuits*, where ν_i^R the number of redundant faults in these, are greater than m and at most n.

Normalized Boolean Difference $\eta(i,j)$ between a pair of *sibling circuits* $S_i(C)$ and $S_j(C)$ is defined as the ratio of the number of mismatched outputs between these two sibling circutis to the total number of outputs. For example, $\eta(i,j) = 1$ implies that all the output functions of $S_i(C)$ and $S_j(C)$ are different, whereas $\eta(i,j) = 0$ if both the siblings are functionally equivalent circuits.

A. Objectives

Given a gate-level netlist of a CUT C, a population of sibling circuits $\Pi(C)$ is generated for a finite set of values of (α). An ATPG tool produces a test-set $TS(i)$, $\forall S_i(C)$ $\in \Pi(C)$. *Our main objectives are to compare*

(i) $FC(i,i)$ with $FC(i,j)$ for all $S_i(C)$, $S_j(C) \in \Pi(C)$ where $i \neq j$.

(ii) functional diversity among the sibling circuits in $\Pi(C)$.

IV. Proposed Method

Our main focus is to analyze the exclusive contribution of network structure and gate functionality while generating an ATPG test-set for single SAFs. In order to evaluate fault coverage using a sibling's test-set, we first require a set (population) of sibling circuits $\Pi(C)$ for a given CUT. After $\Pi(C)$ is created, we generate a test-set for each sibling using an ATPG tool and evaluate its fault coverage on all members of the entire population. In subsequent sections, we discuss the techniques for generating sibling circuit population and our approach to compute the fault coverage.

A. Sibling Circuit Generation

For creating a population for sibling circuits, we need a CUT C, a pre-defined α and the number of siblings $\sigma(\alpha)$ to be generated. In order to have siblings with same structural representation and diverse functionality, we need to ensure that the range of values of α is large.

It is known that unlike other basic gates, the controllability at a side input of an XOR or XNOR gate does not impact the observability of an error arriving at its other input. Hence, we do not change any XOR or XNOR gate. In fact, there is no XOR or XNOR gate present in any of the benchmark circuits (ISCAS85, ISCAS89, ITC99, EPFL) used for our experimental validation. Further, NOT-gates, if any, cannot be replaced by other gates. Thus, while generating Si(C) from C, we have considered replacing only the AND, OR, NAND, NOR gates by their corresponding dual at random locations.

In Algorithm 1, we present our method for producing of $\Pi(C)$, with the objective of creating uniformly varied functionality in sibling circuits. Once $\Pi(C)$ is generated, we obtain a test-set for each of the sibling circuits using an ATPG tool and perform fault simulation on all sibling circuits using the same. Note that following the generation of $\Pi(C)$, a thorough evaluation is required regarding the extent of redundancy caused by gate replacement while obtaining the sibling circuits. The presence of a relatively large number of redundant faults in a sibling circuit compared to those in the original C, can effectively reduce the number of detectable faults in the former and hence, the size of test-set generated by ATPG tools would be smaller. Consequently, such a test-set may fail to ensure high coverage on C or its other sibling circuits that have fewer redundant faults. Thus, the underlying irredundant sub-structures of a network might be drastically affected when large redundancy is introduced due to gate replacement. Although a test-set generated for a sibling circuit with low redundancy may provide substantial fault coverage on another sibling circuit with high redundancy, the converse may not be true. If a SAF is redundant, during test generation an ATPG will either announce it as redundant or abort execution because of time-out. However, as sibling circuits are structurally invariant, the redundant node in $S_i(C)$ may not be redundant in another sibling $S_j(C)$. Hence, a test-set generated for sibling circuits having high ν_i^R would fail to provide good coverage on sibling circuits having low ν_i^R.

Algorithm 1: Sibling Circuit Population Generation

```
Data: Synthesized CUT $C$; Set $A$ of values of $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_t\}$
;
$\sigma(\alpha)$, the no. of siblings to be generated for each $\alpha$
Result: $\Pi(C)$ - Sibling Circuit Population $\{\bigcup_{i=1}^{p} S_i(C)\}$
$\Pi(C) \leftarrow \emptyset$;
$K \leftarrow$ No. of logic gates in $C$;
for $\alpha \in A$ do
    $numGatesToBeReplaced = \lfloor \alpha \times K \rfloor$
    for $l \leftarrow 1$ to $\sigma(\alpha)$ do
        $S_l(C) \leftarrow C$;
        $G_K \leftarrow \{g_i(C) \mid i \in [1, K]\}$
        for $i \leftarrow 1$ to $numGatesToBeReplaced$ do
            Step 1 : Choose gate $g_r(C)$ randomly from $G_K$.
            if $g_r(C) \notin \{"XOR","XNOR","NOT","BUFF"\}$ then
                Replace $g_r(C)$ with $D(g_r(C))$;
            else
                Go to Step 1;
            $G_K \leftarrow G_K - g_r(C)$ ;
        $\Pi(C) \leftarrow \Pi(C) \cup S_l(C)$ ;
Report Sibling Circuit Population $\Pi(C)$ of CUT $C$;
```

B. Binning using Redundant Faults

In order to have fair comparison of fault coverage among structurally-preserving sibling circuits, it is important that the amount of redundancy among them are within a comparable range. Therefore, we impose a constraint while evaluating fault coverage among the sibling, namely redundancy-binning, and group them into separate bins based on the percentage of fault-redundancy. This step ensures that fault coverage is compared only among those siblings, which are grouped in the same bin, as given in Algorithm 2.

Algorithm 2: Redundancy-based Population binning and Test-set Evaluation

```
Data: Sibling circuit population $\Pi(C)$ of size $p$;
$B$, no. of Redundancy bins.
Result: Set of Redundancy Bins $R_{bins} = \{R_{m_1,n_1}, \ldots, R_{m_B,n_B}\}$;
$\forall R_{m_k,n_k} \in R_{bins}, k \in \{1, 2, \ldots, B\}, \forall S_i(C) \in R_{m_k,n_k}$,
$FC(i,i)$ using test-set for $S_i(C)$ and
$FC(i,j)_{i \neq j}$ using test-set of its Sibling $S_j(C) \in R_{m_k,n_k}$.
for each sibling $S_i(C) \in \Pi(C)$ do
    Run structural ATPG on $S_i(C)$ to generate Test-set $TS(i)$;
    Compute $\nu_i^R$, $\nu_i^T$ & $\nu_i^O$ of $S_i(C)$.
Compute $\nu_{min}^R$ and $\nu_{max}^R$, the min. & max. no. of redundant faults
  among all Siblings $\in \Pi(C)$ respectively. ;
$\Delta \leftarrow \lceil (\nu_{max}^R - \nu_{min}^R)/B \rceil$;
Create $B$ bins $R_{m_k,n_k}$, each with redundancy fault range of $\Delta$ and
initially empty;
for $i \leftarrow 1$ to $p$ do
    $k \leftarrow (\nu_i^R - \nu_{min}^R)/\Delta$;
    $< m_k, n_k > \leftarrow < k, k + \Delta >$;
    $R_{m_k,n_k} \leftarrow R_{m_k,n_k} \cup S_i(C)$;
for each $R_{m_k,n_k} \in R_{bins}$ do
    for each sibling $S_i(C) \in R_{m_k,n_k}$ do
        $\forall S_j(C) \in R_{m_k,n_k}, i \neq j$ Perform fault simulation using
          $TS(i)$, ;
        Compute $FC(i,i)$ and $FC(i,j)_{i \neq j}$.
$\forall R_{m_k,n_k} \in R_{bins}, \forall S_i(C) \in R_{m_k,n_k}$,
Report $FC(i,j)_{i=j}$, and $FC(i,j)_{i \neq j}, \forall S_j(C) \in R_{m_k,n_k}$.
```

V. Experimental Evaluation

Simulation experiments on several benchmark circuits demonstrate that structural information predominates fault coverage of a test-set and possibly testability measures of a circuit. Our empirical studies show that SAF-based ATPG generated test-sets provide considerably good coverage on structurally invariant and functionally diverse circuit structures provided they have similar extent of fault redundancy.

A. Set-up

All our experiments are performed on a 4-core 3GHz Intel Xeon processor with 32GB RAM. The platform operating system used was CentOS v7.4. We have taken diverse set of benchmarks from ISCAS85 [5], ISCAS89 [6], ITC99[7] and EPFL arithmetic and random control benchmarks [8]. We have used ABC [9] as a synthesis tool along with $180nm$ technology library from Cadence. Note that for all sequential circuits in our benchmark suite, we have considered their full-scan versions of the same. We have used Mentor's Tessent 2019.1 ATPG Tool for the purpose of test-pattern generation and fault simulation. We have implemented Algorithm 1 & 2 using Python3.6. All our experimental artefacts are available at https://gitlab.com/nanontechResearchTriangle/atpg_structural.

B. Observations and Discussion

In our experiments, we have chosen five distinct values of α viz. $\{0.1,\ 0.3,\ 0.5,\ 0.7,\ 0.9\}$ in order to get functional diversity among the circuits with the same structure. For scalable and effective evaluation, we have taken $\sigma(\alpha)$, the number of siblings generated for each α (via Algorithm 1) to be 40. So, for each benchmark circuit we have created 200 siblings. In order to strike a balance between the same amount of structural redundancy among all sibling circuits of a given C, we have binned $\Pi(C)$ into five disjoint bins. We evaluate the fault efficiency as follows :

(i) For a sibling circuit, measure testable fault-coverage using ATPG generated test-set.

(ii) Evaluate testable-fault coverage using test-set of the other siblings of the same redundancy bin.

In Fig. 4, we show statistical plots for overall fault coverage, fault efficiency and the variation of testable faults with α for two benchmark circuits. The statistical data for other benchmarks have been presented in TABLE II. Note that, we have reported only the median value for overall fault coverage, fault efficiency and variation of testable faults with α.

1) Effects of alpha: In our experimental findings, for most of the benchmarks, ν_i^R is minimum for $\alpha \in \{0.1, 0.9\}$, and maximum for $\alpha = 0.5$. This indicates that for a near-optimal synthesized CUT, fault coverage/redundancy is affected most when half of the logic gates of netlist are changed randomly.

2) Effects of degree of redundancy: With increasing amount of redundancy, the overall fault coverage drops down. However, fault efficiency remains high and small perturbations are observed across various redundancy bins. Thus, fault coverage using ATPG generated test-sets and those for other siblings are comparable. This empirical evaluation consolidates the hypothesis that the overall fault coverage for a test set remains minimally affected by random functional changes at gate-level, and is mostly dominated by the structure of the CUT.

VI. Threats to Validity

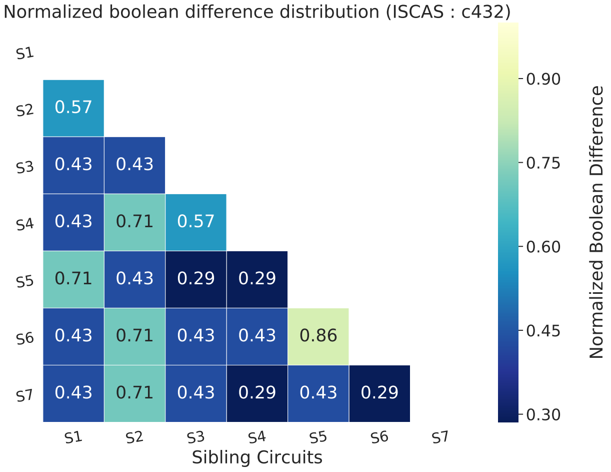

Fig. 3: Distribution of $\eta(i, j)$ among randomly chosen seven Sibling Circuits of $c432$, within bin range = $(3.67 - 68.8]$

The number of testable faults decreases as α reaches 0.5.

One may raise a counter-argument that the sibling circuits in the same redundancy bin could possibly become functionally near-equivalent, thereby resulting in similar fault coverage/fault efficiency by a test-set for CUT for its siblings in our experiment. In the following experiment, we show that the observed match in FC is neither coincidental nor due to random functional equivalence. In order to establish functional diversity, we measure the variation of normalized Boolean difference $\eta(i, j)$ among the siblings in the same bin. A diverse value of $\eta(i, j)$ indicates higher functional heterogeneity, and thus favors our hypothesis. In our experimental setup, we have randomly picked up seven siblings of $c432$ from a redundancy bin and computed $\eta(i, j)$ for all pairs, as shown in Fig. 3. We used *cec* module of ABC [9] for measuring $\eta(i, j)$. On an average, we notice that all pairs of siblings are almost 50% functionally different. Similar results have been obtained with other benchmark circuits.

VII. Conclusion

In this work, we have studied how fault coverage (FC) of a test set is impacted when structure-invariant, functional perturbations are induced in the gate-level net-list of a CUT. Our experiments establish a crucial insight that FC is not significantly degraded under such gate-replacements whenever fault-redundancy remains similar between a given CUT and structure-preserving sibling-circuits. Thus, a test-suite for a CUT is likely to provide high fault coverage when it is exposed to structure-invariant ECO-changes. The study provides an empirical evidence that structural information of a CUT plays a major role in determining FC of a test-set rather than the functional details of each logic node. Therefore, in various machine-learning experiments related to VLSI test and diagnosis, where circuit descriptions are needed as inputs, it may be sufficient to feed only the structural information of the CUT to achieve accurate predictions. Future work would involve providing a formal proof that justifies the preservation of fault-coverage of a test-set amongst structurally-invariant siblings of a CUT.

References

[1] S. M. Reddy, "Complete test sets for logic functions," *IEEE Transactions on Computers*, vol. C-22, no. 11, pp. 1016–1020, 1973.

[2] H. Kim and J. P. Hayes, "Realization-independent ATPG for designs with unimplemented blocks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 2, pp. 290–306, 2001.

[3] M. Pradhan, B. B. Bhattacharya, K. Chakrabarty, and B. B. Bhattacharya, "Predicting X-sensitivity of circuit-inputs on test-coverage: A machine-learning approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

[4] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu, "High performance graph convolutional networks with applications in testability analysis," in *Proceedings of DAC*, 2019, pp. 18:1–18:6.

[5] D. Bryan, "The ISCAS'85 benchmark circuits and netlist format," *North Carolina State University*, vol. 25, p. 39, 1985.

[6] F. Brglez, D. Bryan, and K. Kozminski, "Notes on the ISCAS'89 benchmark circuits," *MCNC*, 1989.

[7] F. Corno, M. S. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," *IEEE Design & Test of computers*, vol. 17, no. 3, pp. 44–53, 2000.

[8] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "The EPFL combinational benchmark suite," in *Proceedings of the International Workshop on Logic & Synthesis*, 2015.

[9] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *International Conference on Computer Aided Verification*, 2010, pp. 24–40.

Fig. 4: Overall fault coverage, fault coverage efficiency and testable fault distribution versus α plots for $c7552$ & sin benchmark

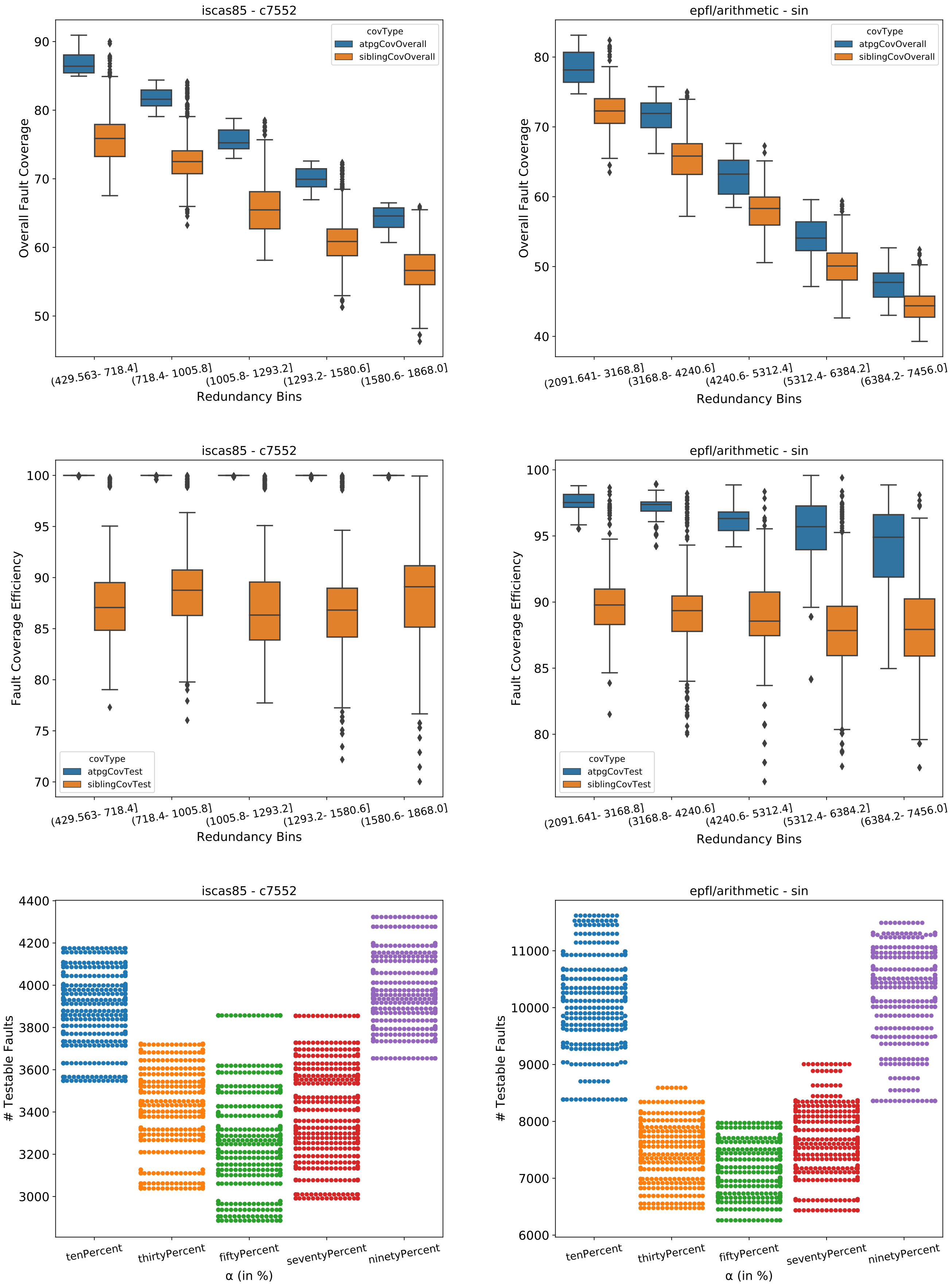

Table II: Results on Benchmark Suites – Fault coverage (overall and efficiency) w.r.t. redundancy bin; Average number of testable faults w.r.t. α

ν^D: #Faults detected by applied test-set.*NS: No sibling circuit found for that bin.

Series	Name		Overall Fault Coverage= $\frac{\nu^D}{\nu^O} \times 100$					Fault Coverage Efficiency=$\frac{\nu^D}{\nu^T} \times 100$					Testable Faults, $\nu^T = \nu^O - \nu^R$				
			bin-1	bin-2	bin-3	bin-4	bin-5	bin-1	bin-2	bin-3	bin-4	bin-5	α=0.1	α=0.3	α=0.5	α=0.7	α=0.9
ISCAS89	s526	FC(i,i)	92.4	86.3	80.9	74.7	67.9	100	100	100	100	100	519	475	462	472	520
		FC(i,j)	79.9	72.9	68.9	62.5	56.2	85.9	84.9	85.3	84.2	84.4					
	s1494	FC(i,i)	80.6	73.2	59.6	52.8	41.9	100	100	100	100	100	1255	925	868	930	1246
		FC(i,j)	66.1	62.3	47.8	42.1	37.1	83.2	85.5	79.5	79.8	91.5					
	s9234	FC(i,i)	84.1	79.2	70.6	66.2	60.2	100	100	100	100	100	4242	3424	3272	3441	4203
		FC(i,j)	66.8	64.7	58.1	55.5	51.8	80.1	81.1	82.0	84.3	85.8					
	s15850	FC(i,i)	88.2	85.5	79.9	76.9	73.2	100	100	100	100	100	8434	7470	7294	7495	8367
		FC(i,j)	76.7	74.4	71.5	69.3	67.6	86.4	87.3	89.4	90.4	92.2					
	s38584	FC(i,i)	86.9	*NS*	*NS*	73.3	69.1	100	*NS*	*NS*	100	100	28641	24140	22802	24266	28747
		FC(i,j)	78.7	*NS*	*NS*	66.8	62.9	90.7	*NS*	*NS*	91.0	91.3					
ISCAS85	c432	FC(i,i)	92.3	84.2	74.6	58.5	48.3	100	100	100	100	100	540	513	493	498	536
		FC(i,j)	69.3	52.0	42.9	35.1	29.4	75.0	62.5	59.0	59.5	60.8					
	c1908	FC(i,i)	93.3	88.8	83.9	79.8	72.5	100	100	100	100	100	909	879	864	873	916
		FC(i,j)	82.9	80.8	77.5	74.0	69.9	88.3	90.9	92.6	92.9	96.6					
	c3540	FC(i,i)	82.6	76.9	62.4	53.8	44.3	100	100	100	100	100	2051	1536	1427	1500	2026
		FC(i,j)	67.5	61.2	47.5	40.7	33.9	80.9	80.5	75.8	75.4	76.1					
	c7552	FC(i,i)	86.4	81.6	75.2	69.9	64.6	100	100	100	100	100	3879	3431	3325	3419	3926
		FC(i,j)	75.9	72.5	65.5	60.9	56.6	87.1	88.8	86.3	86.8	89.1					
ITC99	b10	FC(i,i)	90.9	83.7	75.4	67.8	58.8	100	100	100	100	100	452	392	374	389	454
		FC(i,j)	80.4	70.2	61.2	53.9	47.1	87.9	84.2	81.8	79.9	81.7					
	b14	FC(i,i)	71.7	63.4	51.2	42.9	36.1	99.9	99.9	99.8	99.8	99.9	9275	6556	6194	6521	9882
		FC(i,j)	52.5	46.9	40.8	35.3	30.5	73.8	73.7	78.4	82.1	85.5					
	b22	FC(i,i)	70.1	65.6	53.9	47.1	41.4	99.9	99.9	99.8	99.8	99.7	32227	21944	19606	22423	33007
		FC(i,j)	56.1	51.3	45.2	39.4	34.9	79.6	78.9	82.9	83.9	85.2					
Arithmetic (EPFL)	adder	FC(i,i)	95.5	*NS*	90.5	89.5	87.6	100	*NS*	100	100	100	2925	2740	2685	2735	2923
		FC(i,j)	93.3	*NS*	88.0	86.0	84.0	97.6	*NS*	97.2	96.7	96.4					
	max	FC(i,i)	46.5	42.3	35.2	27.9	21.4	100	100	100	100	100	3856	2971	2932	3061	3783
		FC(i,j)	42.5	34.0	26.4	18.7	15.6	90.7	82.8	75.8	67.5	79.21					
	mult	FC(i,i)	94.4	*NS*	88.3	87.3	84.7	99.9	*NS*	99.9	99.9	99.9	56235	51978	50668	52541	56647
		FC(i,j)	93.6	*NS*	87.6	86.5	83.7	99.3	*NS*	99.2	98.8	98.7					
	sin	FC(i,i)	78.1	71.9	63.2	54.1	47.7	97.5	97.4	96.3	95.7	94.9	10100	7604	7195	7729	10282
		FC(i,j)	72.3	65.8	58.3	50.1	44.4	89.8	89.4	88.6	87.8	87.9					
random_control (EPFL)	arbiter	FC(i,i)	44.4	51.4	27.2	29.0	31.3	100	100	100	100	100	7988	8316	9581	8250	8065
		FC(i,j)	32.0	32.0	17.0	18.0	19.0	73.1	70.5	65.3	60.8	58.6					
	cavlc	FC(i,i)	72.7	65.4	50.0	38.6	28.5	100	100	100	100	100	1205	652	571	688	1223
		FC(i,j)	50.9	40.9	30.9	24.5	17.9	72.2	66.3	63.3	63.3	62.3					
	i2c	FC(i,i)	91.7	89.4	83.1	80.3	76.8	100	100	100	100	100	3270	2886	2848	2904	3254
		FC(i,j)	79.2	78.5	68.3	66.1	63.3	86.2	87.8	82.1	82.6	82.9					
	int2float	FC(i,i)	79.0	73.0	34.9	25.2	82.7	100	100	100	100	100	444	286	282	305	433
		FC(i,j)	53.7	47.4	21.9	20.6	67.4	66.6	66.8	65.8	82.2	79.1					