Analysis of Production Tests Coverage

Niveditha Manjunath*§, Dieter Haerle†, Christian Manthey‡, Mikko Väänänen‡, Stephen Sabanal‡,
Herbert Eichinger‡, Hermann Tauber‡, Andreas Machne‡, Radu Grosu§and Dejan Ničković*
*AIT Austrian Institute of Technology, Vienna, Austria
†KAI, Villach, Austria
‡Infineon Technologies AG, Villach, Austria
§Vienna University of Technology, Vienna, Austria

***Abstract*—In the semiconductor industry, field returns have a negative impact with large costs and potential loss of reputation. As a consequence, a good coverage of the production tests with respect to the common manufacturing defects is essential to ensure the quality of the product to be delivered. Defect simulation is imperative to obtain coverage, however long simulation duration of the production tests can be a huge obstacle. Hence, there is an emergent need for novel methodologies to obtain coverage analysis of AMS chip production tests. In this paper, we address several aspects that are necessary to develop such a methodology. We first propose a method to identify a fault model that mimics the common manufacturing defects and extract all such faults from the DUT layout, we then develop a test ordering procedure that for a given fault selects the test from an existing test suite that is the most likely to detect the fault. The test ordering technique allows to avoid the execution of many tests during the coverage analysis and thus save considerable amounts of simulation time.**

We demonstrate the applicability and efficiency of the resulting techniques on an AMS design from Infineon Technologies AG.

***Index Terms*—coverage analysis, analog and mixed-signal validation, production testing**

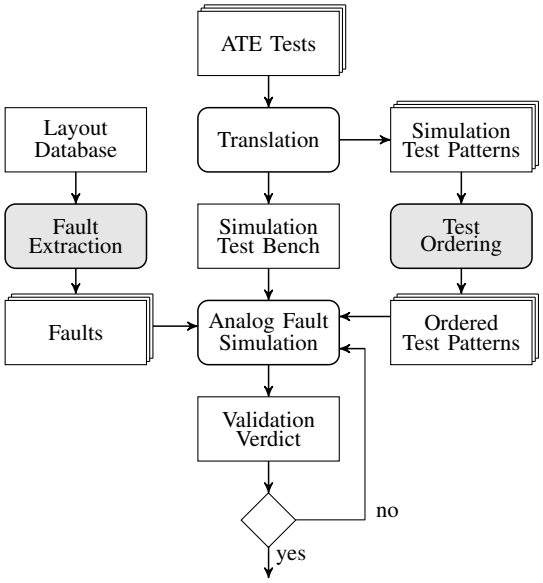


Fig. 1. Flow diagram of the fault coverage analysis.

I. Introduction

Chip fabrication is a complex process involving many processing steps that can introduce manufacturing defects. The resulting faults can affect the chip's functionality before its delivery to the customer. The *field returns* of faulty chips have a negative impact on the semiconductor industry – they can result in huge costs and potential loss of customers. In order to address this issue and to minimize the possibility of delivering faulty chips, the industry uses *production testing* for detecting the defective circuits. These production tests are executed with the *automatic test equipment* (ATE), which enables to physically connect the device-under-test (DUT) to an environment, allowing efficient automatic measurements and evaluating test results.

Production testing of purely digital circuits is a mature field and the current state-of-the-practice already achieves design of robust test suites that efficiently detect manufacturing defects, thus minimizing to a great extent the costly field returns. In contrast, production testing *analog and mixed-signal* (AMS) chips is a much more volatile process. Due to the inherent complexity of the AMS domain, there is a lack of agreed-upon analog fault models and the coverage of the existing production test suites is not sufficiently studied and understood.

The work in this paper is based on the idea of injecting a fault into a model of the DUT and using simulation to determine whether an existing ATE test pattern is able to detect that fault, as illustrated in Figure 1. This approach aids the coverage analysis of production test suites for AMS circuits. More precisely, an effective implementation of our approach consists of the following steps:

1) Translate the ATE test environment, including the ATE test patterns, to a simulation test bench;
2) Identify an appropriate fault model, extract all faults from the DUT layout and inject faults (one fault at a time) to the DUT model;
3) For each fault, order the test patterns according to their likelihood to detect that fault;
4) Simulate all the faulty DUT models with the test patterns in the computed order; and
5) Evaluate the simulation outcomes and check whether the test pattern is able to detect the fault.

In this paper, we propose methods for *fault extraction*, and *ATE test ordering* – the activities highlighted in Figure 1.

It is observed that the most common manufacturing defects involve *shorts* and *opens* in transistors and metal connection lines from [2]. This observation gives us an effective notion

of a *fault model* that is appropriate for the quality analysis of production tests. It is proposed to translate the production test setting that includes the ATE, the DUT and the test patterns to a *simulation environment*, which gives sufficient freedom to inject faults and assess the test suite quality [1]. While the simulation environment provides the necessary flexibility to do extensive coverage analysis, it suffers from the scalability issues – a single simulation of an industrial AMS circuit can take days or even weeks of computation time. In order to tackle this challenge, we develop a *test-pattern ordering method*, which given an AMS circuit and a specific fault (*open* or *short*) computes an ordering of the existing test cases according to their likelihood of detecting that fault. These test cases are then executed in the computed order. As soon as one of the executed test cases detects the fault, the other test cases in the suite can be skipped. It follows that our test case ordering technique allows to reduce significantly simulation time by smart ordering and execution of the test cases. While we have come across various works on ordering faults for reducing test durations, we are not aware of any work that orders tests. For a given fault and a set of tests, the test ordering algorithm selects the test that is most likely to detect the injected fault.

We discuss the related work in Section II, fault extraction in Section III and test ordering in Section IV. The approach has been evaluated and discussed in Section V and conclusion and future work has been discussed in the last section.

II. Related Work

Defect-oriented testing (DOT) has been used in AMS in various contexts. It was proposed in [3] and further developed in [4] and [5]. An industrial experience in using DOT was presented in [6], where it has been shown that this testing approach results in higher test coverage. These works concentrated on generating appropriate tests that are able to detect specific faults. In contrast, we consider the coverage analysis of existing tests with respect to specific faults in this paper.

Fault analysis and coverage is matter of high significance. [7] suggests an approach to extract layout faults and obtain coverage, however the selection of injected faults was based on availability of computational resources. Usual practice to obtain coverage is by sampling faults [8], [9] and ordering them based on their probability of occurrence has been introduced in [10] to obtain fault coverage swiftly. An analysis of various methods has been explored in [11]. The methodology presented in [12] is practical, alternatively, we propose ordering tests to obtain faster coverage as opposed to ordering faults.

Test case prioritization is the topic that has been mostly studied in the context of regression testing of software [13]. In the context of analog hardware design [14], the authors develop methods to select tests to detect structural faults in analog integrated circuits in the presence of process variations. In contrast to our work, their focus is on the construction of optimal tests, rather than the ordering of the existing ones.

The minimization of production testing time is addressed in [15] by designing fault-oriented tests and in [16] by proposing a semiautomated practical simulation flow that addresses circuits with long simulation duration. [17] proposes a method to generate an analog parametric fault model that speeds up Monte-Carlo simulations. The approach in [18] uses neural networks to learn the behaviour of circuits, thus when the neural network model replaces parts of actual circuit during simulation, there is a dramatic reduction of transient simulation time.

III. Fault Model and Fault Database

The most common chip manufacturing defects are *shorts* and *opens* in metal layers. In this section, we define a model for such faults. In addition, we propose a method for extracting all shorts and opens from the simulation model of a circuit.

We assume that in the simulation environment we have the circuit schematic and a layout. We can extract resistance (R) and capacitance (C) parasitics from the layout. The opens and shorts in terms of R and C parasitics are modelled as follows. A resistor in R-only extraction represents the resistance of a metal wire in the circuit. In order to model an open in a metal wire, its corresponding parasitic resistor is assigned a very high resistance, typically in the range of MΩ. This represents an *open* in the cicuit. For example, an *open* between the transistors is modelled by the resistor between them in Figure 2 (Modelled Open). Similarly, a capacitor in C-only extraction represents the capacitance between two metal wires. We recall that this capacitance corresponds to Equation 1, where C is the capacitor's capacitance, ϵ is the permittivity, A is the overlapping area of the two wires and d is the distance between them.

$$C = \frac{\epsilon A}{d} \tag{1}$$

Theoretically, a short between two metal wires can be modelled by setting the distance d between the wires to 0, i.e. by setting the correspondent parasitic capacitance C to an infinite (or very high) value. But within the scope of this paper, the capacitor is replaced by a resistor and the value of its resistance is reduced severely to emulate a *short* between the 2 wires. An example of this is illustrated in Figure 2 (Modelled Short).

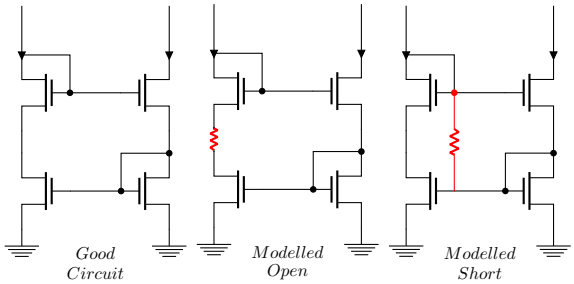


Fig. 2. Flow diagram of the fault coverage analysis.

We can see that the shorts and the opens in the circuits can be extracted from R-only and C-only parasitic extractions.

These R and C components of the layout are obtained in the form of a SPICE netlist. Each row in the extracted netlist corresponds to a parasitic resistor or capacitor, with the following information: (1) The identifier of the component from which the parasitic is extracted, (2) The value of the parasitic, (3) The net names from which it was extracted and (4) Its position in the layout. We note that with this method of extraction, only the net names from the schematic, and not the ones from the layout, are available. However, we require the extracted fault database to contain for each fault a pointer to the affected net names both in the schematic and the layout. In order to obtain layout net names associated to a given fault, we use a cross reference file available in major electronic design automation (EDA) tools and map a schematic net name to its layout equivalent. An example of such a database of fault models is presented in Table I.

IV. Production Test Selection

In order to do coverage analysis of production tests, we need to do it in a simulation environment so that we can inject faults to the DUT. There is a huge challenge to the effectiveness of this approach – simulation of AMS circuits is slow and the exhaustive coverage analysis requires in the worst case simulating all the available tests with all the injected faults. In this section, we address one angle of this problem by developing a test ordering procedure. The proposed method orders the existing test according to their likelihood to detect a given fault. We then execute the tests in the corresponding order and stop with the simulation as soon as one of the tests from the suite detects the fault. As a result, we can potentially skip the execution of many tests and thus considerably save simulation time.

We assume that we have the production test environment ported to the simulation setting, following the procedure from [1]. This environment consists in particular of the DUT M, the test suite $\mathcal{T}$ and the fault database $\mathcal{F}$ extracted from M.

Given a set of real-valued variables X, we denote by $s : X \times [0, d) \to \mathbb{R}$ a *signal* over X, a function that assigns real values to variables in x over the time domain $[0, d)$. We use $S(X)$ to denote all signals definable over X. The DUT M is the tuple $M = (I, O, f)$, where I is a finite set of *input* real-valued variables, O is a finite set of *output* real-valued variables and $f : S(I) \to S(O)$ is the function that maps its input signals to output signals.

We denote by $T \in \mathcal{T}$ an individual *test* in the test suite, where test T is simply a set of input signals in $S(I)$. We also denote by $F \in \mathcal{F}$ an individual *fault* in the fault database.

The coverage analysis is done as follows. For every fault $F \in \mathcal{F}$, we inject F in M to obtain the mutated design M_F. Given a fault $F \in \mathcal{F}$, a DUT M, its faulty version M_F and a test $T \in \mathcal{T}$, we denote by $d(f(T), f_F(T))$ the distance of M from M_F under test T, where d can be any distance metric. This distance measures the difference in the observable behavior that the fault F induces under the test T. Given a constant k that defines the acceptable distance between a model M and its faulty mutant M_F, we say that the test T *kills* the mutant M_F, denoted by $\mathsf{kills}(M, F, T)$, if and only if $d(f(T), f_F(T)) > k$.

In practice, we have seen that the straightforward application of the above procedure is too expensive. A realistic design of a medium-size AMS can have an associated test suite containing at least several dozens of tests, and can have tens of thousands of injected faults. It is clear that the exhaustive coverage analysis is far from being practical. We propose to minimize the testing effort by executing the test that catches a given fault first. We formalize the notion of test ordering in Definition 1.

We now propose a definition that captures the test ordering with respect to its ability to detect a given fault.

Definition 1 (Test Ordering): Given a DUT M, a test suite $\mathcal{T}$, a fault $F \in \mathcal{F}$ and two arbitrary tests $T_1, T_2 \in \mathcal{T}$, we say that T_1 *precedes* T_2, denoted by $T_1 \preceq T_2$ if and only if $\mathsf{kills}(M, F, T_1)$ implies $\mathsf{kills}(M, F, T_2)$.

Computing the precedence order between tests for a given fault is as expensive as executing all the tests on the faulty model of the DUT, hence it is impractical. Instead, we propose to use a heuristic that approximates this order with a *score* function that is fast to compute. This score function, which we denote with $\sigma(M, F)$, assigns to every test-value (T, F) pair a value in $[0, 1]$. This value gives the likelihood of T detecting F. We propose to compute such a score function based on two main ingredients – *distance* and *activity* measures. The distance is a measure of how far a fault is from an observable output. We consider that a fault that is close to an observable output is more likely to be detected by a test that affects that output. The activity is a measure of how much activity a test creates in an observable output[1]. We consider that a test that generates a lot of activity in an output during the non-faulty simulation is more likely to affect that same output when the fault is injected. We finally note that we also take the simulation time into account for our test ordering – in case multiple tests have a similar distance and activity value, we will prefer the test with the smallest simulation time.

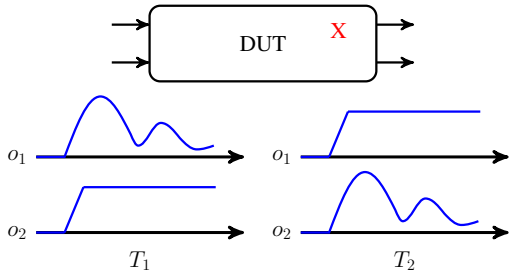


Fig. 3. Combining distance and activity measures.

We illustrate this idea in Figure 3. We can see a DUT with two outputs o_1 and o_2, a fault X that is close to o_1, and two tests T_1 and T_2. The test T_1 creates some activity in o_1, while the output o_2 is only powered up. The situation is reversed for the test T_2. By combining the distance and activity information, we can deduce that the test T_1 is more

[1]It is measured in the output signal obtained during the simulation of the DUT without any injected fault.

Fault Model	Parasitic Element	Parasitance Value	Schematic Netname1	Schematic Netname2	Layout Netname1	Layout Netname2	X-coordinate	Y-coordinate
Short	C21	3.5487E-12	out	gnd	out	gnd	2.56	-40.214
Short	C22	4.5147E-15	net014	I7/I3/net9	152	2942	588.02	412.526
Short	C23	2.7142E-16	I17/close	I17/I15/gate	421	287	212.754	415.847
Open	Rw_m1_124	3.041E-03	I3/net04#254	I3/net04#255	587#254	587#255	2209.0542	-751.36
Open	Rw_m1_129	1.0151	osc#32	osc#33	614#32	614#33	-186.235	58.785
Open	Rw_m1_132	7.064E-04	I2/net34#123	I2/net34#124	1298#123	1298#124	153.354	1354.443
Open	Rw_m1_135	0.4862	net021#74	net021#75	374#74	374#75	-243.153	156.65

TABLE I
A SEGMENT OF THE EXTRACTED FAULT DATABASE.

likely to detect the fault X than T_2 according to our ordering criterion.

a) Distance Score: Given the DUT M in the form of the SPICE netlist and a fault $F \in \mathcal{F}$, we first generate the *connectivity* graph $G_{M,F} = (V, V_I, V_O, E, w)$, where $V = \mathsf{Nets} \cup \mathsf{Instances}$ is a finite set of vertices containing all the nets and instances from the SPICE netlist, $V_I \subseteq V$ is the set of input vertices, corresponding to input variables I in M, $V_O \subseteq V$ is the set of output vertices corresponding to output variables O in M, $E \subseteq (\mathsf{Nets} \times \mathsf{Instances}) \cup (\mathsf{Instances} \times \mathsf{Nets})$ is the finite set of edges of $G_{M,F}$ and $w : V \to \mathbb{R}_{\geq 0}$ is a function that maps every vertex to a *weight*. In order to define the edges, we identify basic M components (NMOS and PMOS FETs, resistors, capacitors, diodes and connecting pads) use the rules [19] shown in Table II to define feasible transitions. We also remove trivial connections from the power supply sources to all the nets and instances from the analysis. A fault F is naturally mapped to a vertex $v_F \in V$. We define the weight function inductively as follows: (1) $w(v) = 0$ if $v = v_F$ and (2) $w(v) = i + 1$ if $i = \min\{w(v') \mid (v', v) \in E\}$. In the next step, we normalize the weights of all nodes to be included in the interval $[0, 1]$. Let $w_{max} = \max\{w(v) \mid v \in V\}$. The normalized weight $\hat{w}(v)$ of a vertex v equals to $\hat{w}(v) = w(v)/w_{max}$. The connectivity score $\gamma_M(F, o)$ in the DUT M of an observable output o with respect to the fault F equals to $\hat{w}(o)$. We note that this measure depends on the structure of the DUT and the selected fault, but not on any particular test.

b) Activity Score: We now compute another measure, the *activity* $\alpha(T, o)$ of an observable output $o \in O$ under the test $T \in \mathcal{T}$. Intuitively, in order to compute $\alpha(T, o)$, we first need to simulate the DUT model M with the test T and without any fault injection. We then store all the output signals, and assign the score to an output depending on the relative activity of its associated signal. Let the activity a of an observable output o under the test T sum up the rate of change of the signal $f(T, o)$. Formally, we have that $a(T, o) = \int |\frac{f(T,o)(t)}{dt}|$. We now normalize the activity values to be within the interval $[0, 1]$. Let $a_{max}(T) = \max\{a(T, o) \mid o \in O\}$. The normalized activity score is given by $\alpha(T, o) = a(T, o)/a_{max}$.

c) Overall Score: The overall score assigned to an output combines its connectivity and activity weights and is given by $\sigma(F, T, o) = \gamma(F, o) \cdot \alpha(T, o)$. The score assigned to a test for a given fault is $\sigma(F, T) = \min\{\sigma(F, T, o) \mid o \in O\}$.

V. EVALUATION

In this section, we evaluate the coverage approach proposed in this paper. We use a proprietary analog and mixed-signal design from Infineon Technologies AG as the evaluation DUT. The ATE used in the production testing of this circuit is Teradyne's Flex Tester. There are 60 tests associated with this design. We applied the methods from [1] to translate this production Flex Tester setting to the simulation environment, by modeling the ATE equipment around the Spice model of the DUT.

In the second step, we extracted all the possible opens and shorts from the DUT, resulting in $53,582$ distinct faults. We selected 5 faults for this evaluation and injected them one by one into the DUT.

We now moved to the DUT simulation, which was done in Cadence Spectre. We first simulated all the tests from the suite on the DUT without injected faults and stored the output signals for the test selection analysis. For each faulty circuit and test, we computed the test selection score using the procedure described in Section IV. We note that we truncated the scores to two decimal places. We ordered the tests according to their scores. In the case of equal scores, we preferred test with smaller simulation times on the circuit without fault injections. Table III highlights 3 tests with highest scores with respect to the 5 faulty circuits. In order to assess the quality of the computed scores, we executed each faulty circuit with the tests in the descending order of their computed scores in the simulation environment. Whenever a test was able to kill the mutant, we would stop the further test execution. In our experimental setting, we were able to kill the 5 faulty circuits with only 3 test, thus avoiding the execution of $5 \cdot 59 = 295$ out of 300 tests. Simulation time for these 5 tests is approximately 1 hour whereas it takes about 129 days and 18 hours to simulate 300 tests sequentially.

The experimental results demonstrate the effectiveness of the proposed approach. In particular, we can see that the test ordering procedure can result in significant saving of simulation time. In the conducted experiments, we were able to decrease the simulation times by a factor of $\tilde{3}000$. We note nevertheless that the test ordering procedure is not sufficient on its own to fully tackle the problem of production tests coverage analysis in the AMS domain. First, in the case that none of the existing tests is able to detect a fault, we would need to simulate all of them before realizing that fact, despite the test

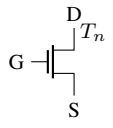
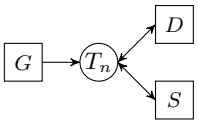
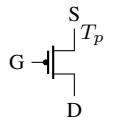
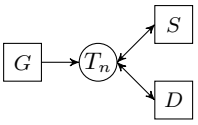
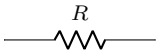
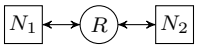
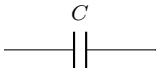
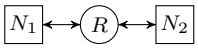
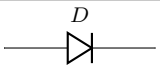
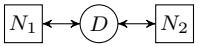
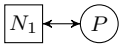
Component	Symbol	Translation	Component	Symbol	Translation
NMOS FET			PMOS FET		
Resistor			Capacitor		
Diode			Connecting Pad		

TABLE II
TRANSLATION OF CIRCUIT ELEMENTS TO NODES AND EDGES.

Test	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Time(s)
56	0.24	0.24	0.14	0.67	0.73	17m 41.9s
57	0.24	0.74	0.14	0.24	0.30	9m 22.7s
59	0.67	0.24	0.14	0.24	0.30	7m 55.8s

TABLE III
TEST SELECTION EVALUATION RESULTS.

ordering procedure. Second, simulating tens of thousands of faulty circuits remains a very time consuming activity, even with the test ordering algorithms.

VI. CONCLUSION

In this paper, we developed parts of a methodology for coverage analysis of AMS circuit production tests in the simulation environment. It includes developing a fault model that mimics common manufacturing defects. In order to decrease excessive simulation times that are needed to execute all tests on all faulty models, we developed a test selection procedure that allows the avoidance of many test executions. We have demonstrated experimentally the effectiveness of the proposed approach.

In future work, we plan to study whether our test selection procedure can be improved to provide some hard guarantees. Despite significant improvements to the overall simulation time needed to perform the full coverage analysis, test ordering does not solve the problem on its own. Also, the proposed test selection procedure is suitable for analog signals only. We plan to combine the techniques developed in this paper with simulation acceleration methods that are being developed in the same project.

REFERENCES

[1] M. Heindl, "Systematic testing of analog-mixed signal systems," in *Master Thesis, TU Wien*, 2017.

[2] K. Arabi and B. Kaminska, "Testing analog and mixed-signal integrated circuits using oscillation-test method," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 7, pp. 745–753, Jul 1997.

[3] M. B. Santos, F. M. Goncalves, M. Ohletz, and J. P. Teixeira, "Defect-oriented testing of analogue and mixed signal ics," in *1998 IEEE International Conference on Electronics, Circuits and Systems. Surfing the Waves of Science and Technology (Cat. No.98EX196)*, vol. 2, 1998, pp. 419–424 vol.2.

[4] R. H. Beurze, Y. Xing, R. van Kleef, R. J. W. T. Tangelder, and N. Engin, "Practical implementation of defect-oriented testing for a mixed-signal class-d amplifier," in *European Test Workshop 1999 (Cat. No.PR00390)*, May 1999, pp. 28–33.

[5] B. Kruseman, B. Tasi, C. Hora, J. Dohmen, H. Hashempour, M. van Beurden, and Y. Xing, "Defect oriented testing for analog/mixed-signal devices," in *2011 IEEE International Test Conference*, Sept 2011, pp. 1–10.

[6] Y. Xing, "Defect-oriented testing of mixed-signal ics: some industrial experience," in *Proceedings International Test Conference 1998 (IEEE Cat. No.98CH36270)*, Oct 1998, pp. 678–687.

[7] J. Sequeira, S. Natarajan, P. Goteti, and N. Chaudhary, "Fault simulation for analog test coverage," in *2016 IEEE International Test Conference (ITC)*, Nov 2016, pp. 1–7.

[8] S. Sunter, K. Jurga, P. Dingenen, and R. Vanhooren, "Practical random sampling of potential defects for analog fault simulation," in *2014 International Test Conference*, Oct 2014, pp. 1–10.

[9] E. Yilmaz, G. Shofner, L. Winemberg, and S. Ozev, "Fault analysis and simulation of large scale industrial mixed-signal circuits," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2013, pp. 565–570.

[10] H. G. Stratigopoulos and S. Sunter, "Fast monte carlo-based estimation of analog parametric test metrics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1977–1990, Dec 2014.

[11] S. Sunter, "Experiences with an industrial analog fault simulator and engineering intuition," in *2015 IEEE 20th International Mixed-Signals Testing Workshop (IMSTW)*, June 2015, pp. 1–5.

[12] S. Sunter, K. Jurga, and A. Laidler, "Using mixed-signal defect simulation to close the loop between design and test," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 63, no. 12, pp. 2313–2322, Dec 2016.

[13] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: a family of empirical studies," *IEEE Transactions on Software Engineering*, vol. 28, no. 2, pp. 159–182, Feb 2002.

[14] G. Devarayanadurg, M. Soma, P. Goteti, and S. D. Huynh, "Test set selection for structural faults in analog ic's," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 1026–1039, Jul 1999.

[15] L. Milor and A. L. Sangiovanni-Vincentelli, "Minimizing production test time to detect faults in analog circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 796–813, Jun 1994.

[16] M. J. Barragan, H. G. Stratigopoulos, S. Mir, H. Le-Gall, N. Bhargava, and A. Bal, "Practical simulation flow for evaluating analog/mixed-

signal test techniques," *IEEE Design Test*, vol. 33, no. 6, pp. 46–54, Dec 2016.

[17] H. G. Stratigopoulos and S. Sunter, "Efficient monte carlo-based analog parametric fault modelling," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*, April 2014, pp. 1–6.

[18] R. M. Hasani, D. Haerle, and R. Grosu, "Efficient modeling of complex analog integrated circuits using neural networks," in *2016 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, June 2016, pp. 1–4.

[19] M. Eick, "Structure and signal path analysis for analog and digital circuits," in *Ph. D. Dissertation, TU München*, 2013.

APPENDIX

A. Complete Evaluation Results

Test	Fault 1	Fault 2	Fault 3	Fault 4	Fault 5	Time(s)
1	0,10	0,09	0,05	0,27	0,29	1d 2h 36m
2	0.04	0.14	0.03	0.04	0.06	15h 6m 28s
3	0.31	0.32	0.03	0.31	0.35	8d 7h 49m
4	0.00	0.00	0.00	0.00	0.00	3h 51m 13s
5	0.00	0.00	0.00	0.00	0.00	3h 13m 47s
6	0.42	0.44	0.05	0.42	0.48	2h 9m 48s
7	0.35	0.74	0.14	0.35	0.40	3h 35m 20s
8	0.41	0.43	0.13	0.41	0.47	1h 44m 36s
9	0.00	0.00	0.00	0.00	0.00	4h 50m 53s
10	0.07	0.07	0.02	0.08	0.09	4h 34m 27s
11	0.38	0.74	0.14	0.67	0.73	5h 50m 42s
12	0.39	0.74	0.14	0.67	0.73	23h 1m 6s
13	0.00	0.00	0.00	0.00	0.00	23h 1m 28s
14	0.00	0.00	0.00	0.00	0.00	1d 0h 19m
15	0.24	0.22	0.14	0.67	0.73	3h 18m 27s
16	0.31	0.32	0.03	0.31	0.35	3h 26m 5s
17	0.31	0.32	0.03	0.31	0.35	1h 6m 43s
18	0.42	0.44	0.05	0.42	0.48	1h 36m 51s
19	0.00	0.00	0.00	0.00	0.00	80ms
20	0.07	0.07	0.04	0.20	0.22	100ms
21	0.13	0.09	0.04	0.22	0.23	1h 55m 35s
22	0.00	0.00	0.00	0.00	0.00	3h 30m 18s
23	0.00	0.00	0.00	0.00	0.00	3h 5m 54s
24	0.24	0.22	0.14	0.67	0.73	2h 58m 18s
25	0.24	0.22	0.14	0.67	0.73	20h 29m 13s
26	0.24	0.74	0.14	0.24	0.30	4h 58m 27s
27	0.24	0.74	0.14	0.24	0.30	21h 42m 41s
28	0.24	0.22	0.14	0.24	0.30	4h 40m 9s
29	0.24	0.22	0.14	0.24	0.30	20h 25m 9s
30	0.67	0.22	0.14	0.24	0.30	12h 24m 40s
31	0.67	0.22	0.14	0.24	0.30	2h 45m 48s
32	0.24	0.22	0.14	0.67	0.73	3h 24m 40s
33	0.24	0.74	0.14	0.24	0.30	3h 40m 59s
34	0.24	0.22	0.14	0.67	0.73	34m 49.4s
35	0.07	0.06	0.04	0.18	0.20	34m 52.9s
36	0.01	0.01	0.00	0.01	0.01	2h 19m 12s
37	0.00	0.00	0.00	0.00	0.00	3h 11m 57s
38	0.07	0.07	0.04	0.20	0.21	2h 25m 45s
39	0.02	0.07	0.01	0.02	0.03	3h 4m 17s
40	0.02	0.02	0.01	0.02	0.03	2h 42m 51s
41	0.09	0.03	0.02	0.03	0.04	3h 26m 44s
42	0.10	0.09	0.06	0.27	0.30	3h 5m 53s
43	0.07	0.20	0.04	0.07	0.08	4h 30m 44s
44	0.06	0.05	0.03	0.06	0.07	3h 5m 47s
45	0.22	0.07	0.04	0.08	0.10	4h 6m 8s
46	0.00	0.00	0.00	0.00	0.00	2h 31m 56s
47	0.04	0.04	0.00	0.04	0.05	2h 30m 46s
48	0.04	0.04	0.00	0.04	0.05	4h 44m 52s
49	0.13	0.11	0.03	0.13	0.15	22h 24m 36s
50	0.62	0.64	0.14	0.67	0.73	2d 15h 39m
51	0.64	0.63	0.14	0.67	0.73	8h 21m 13s
52	0.00	0.00	0.00	0.00	0.01	11h 55m 29s
53	0.00	0.00	0.00	0.01	0.01	5h 56m 16s
54	0.24	0.22	0.14	0.67	0.73	1h 44m 17s
55	0.16	0.15	0.09	0.45	0.49	7h 51m 35s
56	0.24	0.24	0.14	0.67	0.73	17m 41.9s
57	0.24	0.74	0.14	0.24	0.30	9m 22.7s
58	0.24	0.22	0.14	0.24	0.30	15m 31.3s
59	0.67	0.24	0.14	0.24	0.30	7m 55.8s
60	0.31	0.32	0.03	0.31	0.35	70ms

TABLE IV
TEST SELECTION EVALUATION RESULTS.