Testing Digital Circuits: Studying the Increment of the Number of States and Estimating the Fault Coverage

Evgenii Vinarskii[1], Andrey Laputenko[2], Jorge López[3], Natalia Kushik[3]
[1]Lomonosov Moscow State University, Moscow, Russia
[2]Tomsk State University, Tomsk, Russia
[3]SAMOVAR, Télécom SudParis, CNRS, Université Paris-Saclay, Évry, France

***Abstract* – Testing of digital circuits is very important, especially for guaranteeing the correct and reliable functioning of electronic devices. One of the possibilities for deriving high quality test suites is using test generation methods for a corresponding Finite State Machine simulating the circuit behavior. In this paper, we estimate the number of implementation states whenever a circuit mutant is introduced. Experimental evaluation is performed for three types of mutants, namely Single Stuck-At Fault Mutants, Single Bridge Fault Mutants, and Hardly Detectable Fault Mutants. Experiments with the ITC'99 benchmarks (second release) show that in most cases the injection of a fault does not increase the number of states. Moreover, whenever the number of states is increased, the increment is on average 20%. Given this increment, we perform the experiments to showcase that for testing circuits with guaranteed fault coverage with respect to the listed faults, one can apply the W-method with the upper bound $m = 1.2n$ states, for n states in the specification (circuit) FSM.**

***Index Terms* – Testing, Digital/Logic Circuits, Finite State Machines.**

I. INTRODUCTION

ELECTRONIC DEVICES and their development are rapidly progressing and thus, thorough testing and verification of their components, such as for example digital circuits, is crucial. Moreover, testing with guaranteed fault coverage (100% fault detection) is extremely important for digital circuits that are used in critical systems. Such testing and formal verification may be performed through addressing formal models, such as Finite State Machines (FSM)[1].

In this paper, we study logic circuits with memory, i.e., sequential circuits that contain simple gates implementing Boolean functions and latches represented by D-triggers. Following [2], we consider three types of faults that can occur in the circuit implementation, namely Single Stuck-At Faults (SSFs), Single Bridge Faults (SBFs), and Hardly Detectable Faults (HDFs). A circuit that contains a fault is referred to as a mutant [3], as usual.

We focus on distinguishing such mutants from the original circuit specification. Despite the fact that randomly generated test sequences are widely used, our experiments clearly show they are not sufficient for getting high fault coverage w.r.t. the listed faults. The reader can refer to Fig. 1 where the experimental results for benchmark b01 of the ITC'99 package (second release) [4] are presented. It is clearly seen that there exists a limit w.r.t. the fault coverage and the length of a random sequence, i.e., the fault coverage cannot be improved after certain length of *a random test sequence* is reached [5].

Fig.1. The fault coverage for *random tests* generated for the benchmark b01

In our previous work [2], we tried to apply an FSM based test derivation strategy to test the ITC'99 circuits. In particular, we applied the transition tour over the corresponding FSM; a high fault coverage and short test sequences were obtained. However, we did not reach 100% fault coverage neither, as it is also known that the transition tour can only guarantee the detection of all output faults. For detecting transition faults, test sequences derived based on the so-called W-method can be applied [6]. Therefore, we currently study how often transition faults actually occur, when introducing a mutant of one of the three types mentioned above, and what is the fault coverage of tests returned by the W-method for different upper bounds of implementation states. We try to find out the average increase in the number of states when injecting a fault as classical FSM based test generation techniques return 'good' test suites of polynomial length whenever a mutation does not increase the number of specification

978-1-5386-5021-9/18/$31.00 ©2018 IEEE

states. On the other hand, when the number m of mutant states is greater than that of the specification, the length of a test suite with guaranteed fault coverage can become exponential [6].

Similar to random test sequence generation, we performed the experiments with the benchmarks of the ITC'99 package. The obtained results show that in most cases the injection of a fault does not increase the number of states in the mutant. Moreover, whenever the number of states gets increased the difference in the number of states is on average 20%. Experimental results confirm our hypothesis that the upper bound of $m = 1.2n$ states, where n is the number of states in the specification FSM, is sufficient for deriving high quality test suites using W-method (or any modification of it [7]).

The structure of the paper is as follows. Section II contains preliminaries. Section III presents the experimental set up while Section IV contains the experimental results. Section V concludes the paper.

II. PRELIMINARIES

In this section, we briefly sketch the notions and definitions that are used in the paper. As the current work is a continuation of [2], most of the preliminaries are taken from the previous work.

A *combinational circuit* is composed of logic gates; each logic gate implements a Boolean function. *Sequential circuits* include on one hand, a combinational logic, i.e., a combinational circuit, and on the other, memory elements, namely *latches*. Such memory elements can differ, and in this paper, we consider simple D-triggers that 'store' an internal value for one tick during the circuit functioning.

An *initialized Finite State Machine* (*FSM*)**,** is a 5-tuple $\mathbf{S} = <S, I, O, h_S, s_0>$, where S is a finite nonempty set of states; I and O are finite input and output alphabets; and $h_S \subseteq S \times I \times O \times S$ is a behavior (transition) relation; $s_0 \in S$ is the initial state [1].

An FSM modeling the behavior of a given logic circuit can be obtained via direct simulation of the circuit behavior at each (latch) state under each possible input. In this case, the set S of FSM states consists of Boolean vectors (probably hashed as integers) representing all possible combinations of the states of the latches. Correspondingly, the set I of FSM inputs has Boolean vectors that can be applied to the sequential circuit.

As an example, consider a sequential circuit shown in Fig. 2. An FSM describing the behavior of this digital circuit is presented in Fig. 3.

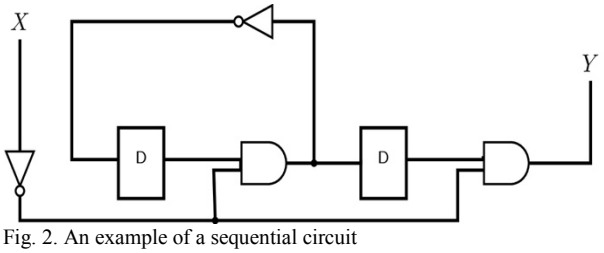

Fig. 2. An example of a sequential circuit

This circuit has one input, two latches and one output. The FSM simulating its behavior has two inputs, two outputs and three states. Each initial latch value is considered to be zero, therefore the initial FSM state is (00). Note, that the state (11) is not reachable from the initial one and thus, it is not included into the set $S = \{(00), (01), (10)\}$ of FSM states.

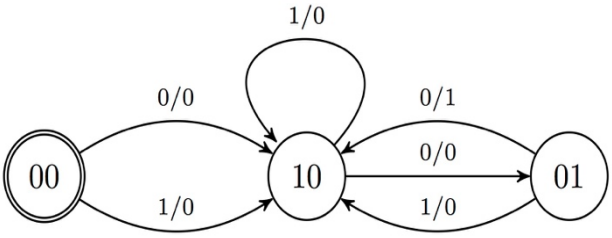

Fig. 3. An FSM describing the behavior of the circuit in Fig. 2

Following [2, 8], in this paper, we consider three types of circuit mutants.

- *Single Stuck-At Fault (SSF) Mutants* when one circuit gate gets stuck at a given logical value ("1" or "0");
- *Single Bridge Fault (SBF) Mutants* when a given input of a given logical gate is wrongly wired (bridged);
- *Hardly Detectable Fault (HDF) Mutants* when a single gate changes its output for a single input.

The goal of this work is to study the correlation between the mutants of a sequential circuit and those of the corresponding FSM. In particular, we furthermore present the experimental set up and the corresponding results on how the mutants listed above affect the number of FSM states and thus, the length of a test suite derived using FSM based strategies.

III. EXPERIMENTAL SET UP

In this section, we briefly discuss the tools used in the experimental evaluation as well as the set of benchmarks that were utilized as experimental samples.

Given the specification circuit **C**, in our experimental evaluation of the number of states in mutant (implementation) FSM, we sequentially execute the following steps.

1. Derive an FSM **S** simulating **C**'s behavior. By default, all latches are assigned to 0, namely (00…0) is the initial state of **S**.
2. Derive a mutant **C'** of the circuit **C** by injecting a fault of interest into the circuit **C**.
3. Derive an FSM **M** simulating the behavior of the mutant.
4. Compare the number of states in FSMs **S** and **M**. Report the difference, whenever the number of implementation states is greater than that of the specification.

Steps 2-4 are repeated iteratively for different types of mutants. Later on, the average value of the difference between the number of states in the specification and mutant is calculated. Note, that in this case, we are only

interested in the cases when the number of mutant states is greater than that of the specification. The reason is that in this case, the length of the test suite derived using W-based methods, can become exponential [6].

After the average increase m of the number of states for mutant FSMs is established, a test suite is derived using the W-method with the corresponding upper bound m and the fault coverage of the obtained test suite is evaluated w.r.t. the faults listed above.

In order to perform such experimental evaluation a number of software tools were utilized; those are either available online or developed in-house. The first step was performed with the use of the tool developed in the group of Prof. N. Yevtushenko (Tomsk State University) [9]. The corresponding simulator allows to model the behavior of a logic circuit starting from the initial state (00 … 0) and produce the minimal FSM in the KISS format. The same tool was utilized when executing Step 3.

For the mutant derivation, we used another tool that we previously developed [8], namely a mutant generator for the circuits written in the BLIF format. All the translations between different formats representing the logic circuits were made with the use of the ABC tool, developed at UC Berkeley [10]. Test generation using W-method was performed using the software solution FSMTest-1.0 [11]. All the steps were combined together via a script provided by the authors.

As mentioned before, for the experimental evaluation we used a subset of benchmarks of the package ICT'99. A short description of these circuits is provided in Table I [2].

TABLE I.
A SHORT DESCRIPTION OF CIRCUITS USED IN THE EXPERIMENTS

Name	Number of inputs	Number of gates	Number of outputs	Number of latches	Short description
b01	2	42	5	2	FSM that compares serial flows
b02	1	23	1	4	FSM that recognizes BCD numbers
b06	2	45	6	9	Interrupt handler

IV. EXPERIMENTAL RESULTS

In this section, we present the experimental results on the average increase of the number of states when an SSF, SBF or HDF mutant is derived. Experimental results for circuits b01, b02, and b06 are shown in Table II.

Given the results presented in Table II, one can conclude that on average the number of states is not largely increased when any of the three types of mutants are considered. In particular, on average the number of states is increased not more than twenty percent. The latter means that for testing such circuits with guaranteed fault coverage one can apply the W-method (or any modification of it [7]) with the upper bound of $m = 1.2n$ states, where n is the number of the specification FSM states.

TABLE II
EXPERIMENTAL RESULTS FOR CIRCUITS B01, B02, B06

Circuits	b01	b02	b06
Percentage of SSF mutants increasing the number of states	1%	4%	22%
Average increase in the number of states for the SSF mutants	6%	16%	6%
Percentage of SBF mutants increasing the number of states	14%	4%	56%
Average increase in the number of states for the SBF mutants	19%	13%	19%
Percentage of HDF mutants increasing the number of states	2%	9%	33%
Average increase in the number of states for the HDF mutants	19%	20%	18%

We performed these types of experiments as well, to verify our hypothesis about the lowest number m of implementation states delivering a high fault coverage for the W-method. In fact, we estimated the fault coverage of the test suites for different m values, namely, $m = n$, $m = (n + 1)$, …, $m = 1.2n$. The experimental results for SSF, SIF, and HDF mutants are presented in Figs. 4, 5, and 6, correspondingly. In Fig. 7, the fault coverage is estimated with respect to the union of mutants of all three categories.

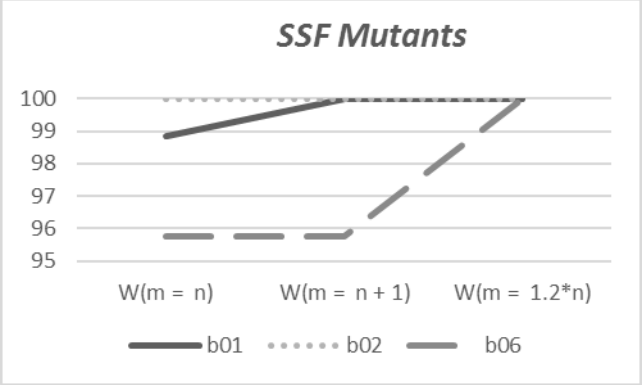

Fig 4. Fault coverage for *SSF mutants*

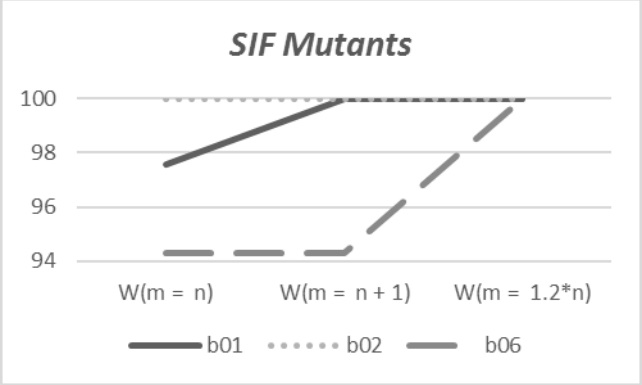

Fig 5. Fault coverage for *SIF mutants*

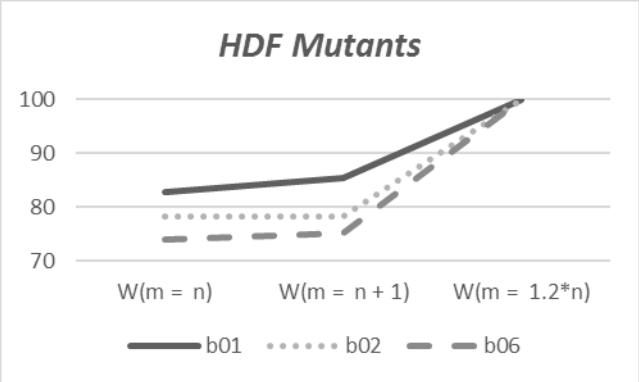

Fig 6. Fault coverage for *HDF mutants*

As the fault coverage of the derived test suites reaches 100% assuming the upper bound $m = 1.2n$ for the number of implementation states, we conclude that our hypothesis was correct and the average 20% increase of the specification states allows to derive high quality test suites using the W-method (or its modifications).

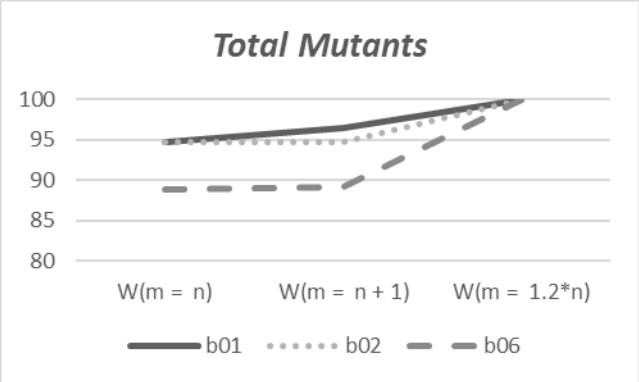

Fig 7. Fault coverage for all types of mutants

VI. CONCLUSION

In this paper, we studied the correlation between mutants of logic circuits and corresponding FSMs. In particular, we investigated how circuit mutants such as for example, single stuck-at faults affect the number of states in the corresponding mutant FSM. Experiments with a number of benchmarks from the ICT'99 package showed that on average there is a 20% difference between the number of states in the specification and its faulty implementation. Therefore, for testing such circuits with *expected* guaranteed fault coverage one can apply W-method or its modifications with the upper bound $m = 1.2n$ where n is the number of specification states. Our experimental results for different upper bounds on the number of implementation states confirmed the hypothesis of $m = 1.2n$ being the reasonable value for this purpose.

The obtained results are encouraging, and they provide some future directions of our work. In particular, we plan to study other types of mutants for logic circuits as well as to try other packages of benchmarks for the experimental evaluation and other methods (or heuristics) for test derivation which return shorter tests than the W-method while preserving the fault coverage. On the other hand, we also would like to perform some theoretical investigations for estimating the upper bound on the number of implementation states when a given fault is injected into the circuit.

ACKNOWLEDGMENT

The authors acknowledge that some results of this paper were preliminary presented on the PhD Workshop of the ICTSS conference (2017). The help of anonymous reviewers of this workshop is highly appreciated.

The authors also acknowledge the help of Prof. Nina Yevtushenko for all the fruitful discussions.

Finally, the obtained results were partially supported by the Russian Science Foundation (RSF), research project No. 16-49-03012.

REFERENCES

[1] A. Gill, "State-identification experiments in finite automata", Information and Control, pp. 132-154, 1961.

[2] J. Lopez, E. Vinarsky, A. Laputenko, "On the Fault Coverage of High-level Test Derivation Methods for Digital Circuits", In Proc. of the 18th international conference on micro/nanotechnologies and electron devices EDM 2017, 2017.

[3] Ma Y., Offutt J., Kwon Y., "MuJava: a mutation system for java", Proceedings of the 28th international conference on Software engineering, Shanghai, China, pp. 827-830, 2006.

[4] F. Corno, M. Reorda, G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results", IEEE Design & Test of Computers, pp. 44-53, 2000.

[5] A. Laputenko, J. Lopez, N. Yevtushenko, "Verifying digital components of physical systems: experimental evaluation of test quality", Tomsk, Izvestija vysshih uchebnyh zavedenij vol.60, no 11, pp. 146-151, 2017

[6] M. Vasilevskii, "Failure Diagnosis of Automata", Kibernetika, pp. 98-108, 1973.

[7] R. Dorofeeva, K. El-Fakih, S. Maag, A. Cavalli, N. Yevtushenko, "FSM-based conformance testing methods: a survey annotated with experimental evaluation", Information and Software Technology, 52, pp. 1286-1297, 2010.

[8] N. Kushik, J. Lopez, N. Yevtushenko, "Investigation of Correlation of Test Sequences for Reliability Testing of Digital Physical System Components", Russian Physics Journal, pp. 1274-1280, 2016.

[9] G. Sapunkov, "The development of the information system "Finite state machine"", Master thesis. Tomsk State University, 2008.

[10] R. Brayton, A. Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool", In Proc. of the Computer Aided Verification: 22nd International Conference, pp. 24-40, 2010

[11] N. Shabaldina, M. Gromov, "FSMTest-1.0: A manual for researches," In: EWDTS. pp. 1-4, 2015.

Evgenii Vinarskii, was born in Tomsk, Russia in 1996. He graduated from the Academic Township Lyceum (Tomsk) in 2015. He is currently a third-year student of Lomonosov Moscow State University, department of computer science. His research interests include discrete mathematics, programming, algebra, and formal methods for software testing.

Andrey Laputenko was born in Yurga, Russia. He has obtained his bachelor and master degrees in radiophysics at Tomsk State University, Tomsk.Currently, he is a postgraduate student at Tomsk State University. His research interests are in the field of optimization and testing hardware systems and digital circuits.

Jorge López was born in Guatemala City, Guatemala. He obtained his PhD degree in informatics from the Paris-Saclay University in Paris, France. Currently he works as a postdoctoral researcher at Télécom SudParis/Paris-Saclay University. His research interests include optimization of code and systems, formal models and software testing.

Natalia Kushik was born in Tomsk in 1987 and has received her diploma degree in Applied Mathematics from Tomsk State University, Russia, in 2010. She has got a PhD degree in 2013. In 2015, she joined Telecom SudParis/Institut Mines Telecom and she is currently working as an assistant professor at the department of Mobile Networks and Services. Her research interests include automata theory, quality evaluation and prediction, testing and verification.