Software-Based Testing of Sequential VHDL Descriptions

Mathieu Scholivé, Vincent Beroulle, Chantal Robach, Marie-Lise Flottes, Bruno Rouzeyre

Software-based Testing of Sequential VHDL Descriptions

M. Scholivé[1], V. Beroulle[1], C. Robach[1], M.L. Flottes[2], B. Rouzeyre[2]
[1] *LCIS-ESISAR, 50 rue B. de Laffemas, BP 54, 26902 Valence, France*
[2] *LIRMM, Université de Montpellier, 34090 Montpellier, France*
<Mathieu.scholive, Vincent.beroulle, Chantal.robach>@esisar.inpg.fr
<flottes, rouzeyre>@lirmm.fr

Abstract

In this paper, we propose a new high-level test pattern generation technique for sequential circuits. The main motivation is two-fold: on one hand, we elaborate test data for design validation; on the other hand, we deal with the problem of structural test development at functional level. The proposed test method, i.e. mutation testing, allows us to work with a fault model at software level on VHDL descriptions; this approach has already shown its efficiency on combinational descriptions. In order to tackle the specific problem of sequential circuits, the description is modified so that the state variables are made observable and controllable.

Keywords : validation, test, sequential circuit, VHDL description, software-based testing, mutation.

1. Introduction

Increasing complexity of integrated circuits drives test costs up. Synthesis tools adopt new solutions relying on high-level circuit descriptions to deal with "Time-to-Market" constraints. Concurrently classical Automatic Test Pattern Generators (ATPG) become very memory - and time -consuming due to the high number of integrated gates. In turn, *front-end* (first stages of the design flow) tends to be restricted to behavioural VHDL descriptions whereas, at this level, no classical ATPG performs it task. These remarks drive the development of high-level test generation technique.

Our high-level test approach reuses a well known software test techniques: the mutation testing [1]. Firstly, this technique is used to generate test data for design flow validation from design specifications to gate level descriptions. Then, in a second step, the same test data are used for structural test purpose. If the proposed test sequences do not achieve sufficient fault coverage, gate-level ATPG efforts will only focus on few remaining untested faults. Recycling test data for validation into test data for structural testing decreases low level test efforts and related costs.

First encouraging results were achieved using this design validation technique on behavioural VHDL descriptions of combinational circuits [2]. So, to go further, this article presents a study on high level sequential VHDL descriptions. In particular, we choose to deal with a first subset of sequential circuits corresponding to finite state machines (FSMs) where all possible states are defined (no invalid states).

The sequel of this paper presents an overview of our high-level test pattern generation technique and first experimental data.

2. Mutation testing improvement for sequential descriptions

In the process of mutation testing, we select test vectors that can distinguish a program from a set of faulty versions of this program, called "mutants". One mutant is generated by injecting one single fault in the original program; a fault is a "small" syntactically correct modification of one code line. Figure 1 illustrates the procedure composed of three main tasks: Mutant generation, Test data generation, Test evaluation.

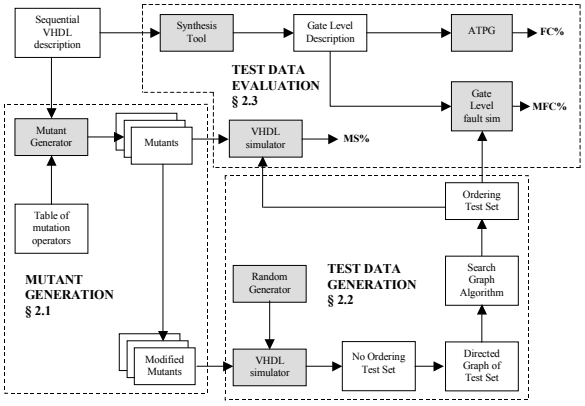

figure 1. Synoptique

2.1. Mutant generation

Mutant generation for sequential circuits can be broken down into two steps. The first consists of generating mutants from the sequential description. The

second step consists of transforming the mutant sequential VHDL descriptions into combinational descriptions for test generation purpose.

Concerning the first step, an automatic mutant generator has been integrated in "ALIEN" (a tool for mutation testing developed at LCIS-ESISAR[1]). This generator needs two inputs: the original circuit description and a table of mutation operators. It generates the list of mutants and a table for mutant descriptions including the number of generated mutants as well as for each mutant the type of mutation, the mutation performed, and where this mutation occurs.

Concerning the second step, we adapt a topological-analysis-based approach, the iterative-array model [3], where a combinational model for a sequential circuit is constructed by regenerating feedback signals from previous time copies of the circuit. This is performs by replacing all occurrences of state variables with pseudo input or output signals. In case of affectation on a state variable, we replace the state variable with an output NSn, n∈[1:m]. In the case of a condition, we replace the state variable with an input PSn, n∈[1:m]. Once this operation is achieved, we remove the clock's declaration from the primary inputs, as well as all code lines, concerning the clock.

2.2. Test data generation

The second stage of our method consists of generating test data from the new combinational descriptions (original and mutants) . Currently, test data are generated randomly and only random vectors that distinguish the original program behavior from a mutant one are kept. The behavioral analysis is performed with the help of a VHDL simulator. When the behavior of a given mutant submitted to a test pattern differs from the original program submitted to the same pattern, we say that the mutant is killed. However, for the moment, the test vectors list is not ordered and thus not adapted for fault detection on the original sequential description. The following paragraph explains how to generate this ordered test sequence.

The fundamental idea is to order the vectors so that the outputs NSn, n ∈ [1 : m] of the original combinational description submitted to vector i correspond to the required inputs PSn, n ∈ [1 : m] for vector i+1. To carry out this ordering, we propose to use a graph representation where each node represents a circuit state (PSn, NSn). There is a directed edge between two nodes when outputs NSn of the first considered node are equal to the inputs PSn of the second one. Note that, according to our limitation on the type of chosen VHDL descriptions, (FSM with only valid states) PS (and NS) are to each possible state at least once. Therefore, whatever the node, there is a path to all other nodes, the resulted directed graph is strongly connected. From this representation, it's possible, using a BFS (Breadth-First-Search) algorithm to find a minimal sequence passing through all nodes of the graph from a chosen initial state.

2.3. Test data evaluation

Firstly, test data evaluation consists of applying the previous test sequence on the mutants in order to compute the mutation score (MS% : ratio between the number of killed mutants and the total number of generated mutants).

Then, a synthesis tool is used to generate the gate level description of the circuit. Mutation fault coverage (MFC%) and Fault Coverage (FC%) are respectively computed for the stuck-at fault model using our test sequence (ALIEN) and a sequence issued from a classical gate-level ATPG (FLEXTEST).

Table 1 presents experimental data for two circuits: b01 (FSM that compares serial flows) and b02 (FSM that recognizes BCD numbers), collected on ITC'99.

	ALIEN			FLEXTEST	
Circuit	length	MS%	MFC%	Length	FC%
b01	42	98.90	98.06	75	97.42
b02	17	96.20	93.97	35	97.67

table 1. Results for two ITC'99 benchmarks

3. Conclusion

In this experiment, we have succeeded in generating structural test data from VHDL functional descriptions of sequential circuits. The proposed technique is based on a software testing technique: the mutation testing. High low-level fault coverage can be achieved with short high-level test sequences but further investigations on mutant generation and deterministic high-level test generation should improve the current results. Even if our study was initially limited to a given type of circuits, the first results obtained, whatever the length of sequences or the mutation fault coverage, encourage us to apply our approach to other types of circuits.

4. References

[1] R. De Millo, R.J. Lipton, and F.G. Sayward, "*Hints on Test Data Selection : Help for the Practicing Programmer*", *IEEE Computer*, vol. 11, No. 4, pp. 34-41, 1978.

[2] G. Al-Hayek and C. Robach, "*From Design Validation to Hardware Testing : a Unified Approach*", *Journal of Electronic Testing : theory and application 14*, pp 133-140, 1999.