Methods to measure and to enhance the testability of behavioraldescriptions of digital circuits

Jean-François Santucci, Gérard Dray, Marc Boumedine, Norbert Giambiasi

Methods to Measure and to Enhance the Testability of Behavioral Descriptions of Digital Circuits.

Jean-François Santucci Gérard Dray Marc Boumédine Norbert Giambiasi

L.E.R.I.
Parc Scientifique G. Besse 30 000 Nîmes, France
E-mail : santucci@eerie.eerie.fr

Abstract

This paper presents an approach to reduce the cost of test pattern generation of behavioral descriptions. This approach utilizes a group of methods allowing the designer to assess and to enhance the testability of behavioral descriptions.

1: Introduction

Recently, new automatic test generation methods for digital circuits described according to a behavioral view have been developed. A behavioral view allows a circuit to be described independently of its internal structure by defining how its outputs react when values are applied to its inputs. In order to describe the circuit behavior, specific languages called H.D.L. (Hardware Description Languages) are used, for example DACAPO [1] and VHDL [2]. Behavioral test pattern generation is useful for three main reasons :

- the structural description of a circuit is not always available. The only knowledge about the circuit under test may come from data sheets or measurements of signals.
- the circuit complexity may not allow us to use conventional test pattern generation tools developed for structural descriptions.
- the test generation process is an integral part of the design process and one must start implementing it during the behavioral design phase.

Motivated by these facts, various conceptually different ATPG methods for behavioral descriptions have been proposed in the recent past [3,4,5, 6,7]. Judging from the literature, two different approaches have been used : test pattern generation using symbolic simulation [7] and test pattern generation using path sensitization [3,4,5,6]. The first approach is limited because for behavioral descriptions involving complex data and control structures, the analysis of symbolic expressions and the resolution of the equations involved in this approach can be difficult and even impossible. The second, concerning test generation, is based on a fault detection technique and a path sensitization principle. This implies the definition of a behavioral fault modelling scheme [8] and the resolution of several well-known problems : manifestation of a fault, sensitization, propagation and justification. Any test generation algorithm presented in the previous work builds a decision tree and applies a backtracking search procedure in order to find a solution. Because of the exhaustive nature of the search process, the number of operations performed to find a solution can, in the worst possible case, increase exponentially with the number of lines of code involved in a behavioral description. The aim of this paper is to propose various methods in order to reduce the computational cost of the behavioral test generation process. We propose a new approach in order to enhance the testability of digital circuits[1] whose behavior is described by an H.D.L. The approach consists in developing a set of methods which can be used in the following two situations : early in the design phase (manufacturing phase) and after the design of digital circuits (acceptance phase). This set is divided into the following three methods :

- the first method allows the user to modify the initial behavioral description without adding any functionalities to the circuit behavior;
- the second defines cost functions allowing an estimate of the testability of elements involved in a behavioral description which can be used both in guiding the behavioral test generation process and in the design process;
- the third consists in the modification of the behavioral description by adding functionalities. Since some functionalities are added to the circuit behavior, this last method cannot be performed if the circuit has already been designed.

Behavioral descriptions can be partitioned into two categories, namely non-procedural descriptions and procedural descriptions. In this paper, we are only interested in procedural descriptions. They are characterized by a control flow and a data flow. The control flow corresponds to the sequence of operations according to control structures involved in the descriptions. The data flow corresponds to the objects handled in the description and the operations which manipulate these objects. In order to facilitate the definition of the three kinds of tools previously introduced, it is essential to explain in detail the basic concepts linked to behavioral descriptions. We

[1]*This work is supported by the European ESPRIT-EVEREST project.*

have therefore defined an internal model which highlights, on the one hand, the sequential and concurrent aspects and, on the other hand, the separation and interaction between the control flow and the data flow [9]. The remainder of the paper is organized as follows. Section 2 gives an overview of the approach. The internal model derived from behavioral descriptions is presented in Section 3. In Section 4 we present a brief overview of the behavioral test pattern generator which has been implemented. Section 5 deals with the first method aimed at simplifying behavioral descriptions without adding functionalities. In Section 6 we describe the cost functions which have been defined to evaluate the testability of behavioral descriptions. The last method involving modifications of behavioral descriptions by adding functionalities is described in Section 7. Experiment results are presented in Section 8.

2: Basic strategies : a brief overview

As already mentioned, our approach for reducing the computational cost of behavioral test pattern generation is based upon three main methods :

- a method aimed at simplifying behavioral descriptions without adding functionalities.
- a method designed to evaluate the testability of behavioral descriptions.
- a method to improve the testability of behavioral descriptions with the adding of functionalities.

One must keep in mind the constraints within which we have to work : circuits are described according to only a behavioral point of view and we have to keep the cost of computing the various methods significantly lower than the Test Generation cost they are meant to reduce.

Let us now briefly outline the most important strategies applied in our approach :

- Before starting to reduce the computational cost of behavioral test generation, we transform the textual behavioral description into an internal model allowing the main features of behavioral descriptions to be explicitly represented : sequential and concurrent aspects and separation between the control flow and the data flow. In order to facilitate the definition of the previously mentioned methods, the model support of the application of the algorithm has to meet two requirements: it should be easy to handle and explain in detail the basic concepts linked to behavioral descriptions. In order to satisfy the first requirement, the definition of the internal model is based on hierarchical multi-view modelling itself based on graph concepts. To address the second requirement, there are two levels of modelling : the internal model associated with a behavioral description has two views. These views include an external view which represents the inputs/outputs of the descriptions and an internal view made up of two parts the separation and interaction of which are explicit : a control model representing the sequencing of the operations involved in the description and a data model representing the objects(variables,signals, constants) and the handled operations in the behavioral description.
- The goal of the first method is to define rules that modify behavioral descriptions in order to increase the performance of the test generation process. These modifications are performed without adding any functionalities to the descriptions : they do not change the functionalities of the circuit, but reorganize a given description in such a way that the computational complexity of the test generation process applied on the new description is reduced. To achieve such a goal, a first problem must be solved : we have to measure the performance of the test generation process on a given description. Such measurements require the following :
  - Their computational cost must be lower than the cost of the test generation process.
  - They must be independent of the H.D.L. used for describing the behavior of a given circuit.
  - They should not express the complexity of the algorithm involved in the circuit behavior, but they have to express how the manner in which this behavior is described can increase or reduce the test generation cost.

  To respond to this problem, we have defined complexity measures of a behavioral description based upon complexity measures developed in software engineering [10,11]. We have chosen such a solution for the following reasons :
  - A behavioral description of a digital circuit is a program so that software complexity metrics can be applied on it.
  - The properties of software engineering complexity measures accord with the requirements of the complexity measures of behavioral descriptions. Software complexity measures are easy to compute, not related to a particular programming language, and different from the complexity notion associated with the algorithm performed by the software.

  Once we are able to measure the complexity of a given behavioral description, we can define modifications involved in a given behavioral description in order to produce a less complex description according to the previous measures. The modifications are carried out by means of rules allowing the user to change the way in which the behavior of the circuit is described.
- The second way to reduce the computational cost is a testability analysis method. The objective of this method is twofold. The first is to provide information in order to speed up the search process of behavioral test generation methods. The second is to point out parts of descriptions which will produce potential testing problems. As for most testability analysis approaches for structural descriptions [12], we use the concepts of controllability and observability to define the testability of an element of the internal model of a behavioral description [13]. This definition involves two steps :
  - the definition of controllability and observability measures for the basic elements of the internal model.

• the definition of different ways of computing the testability measures according to the authors' suggestions [14] in order to compensate for the fact that testability measures are not totally accurate. They suggest switching among the testability measures during the test generation process.

We therefore propose several ways of computing behavioral testability measures based upon the computations of testability measures defined for structural descriptions.

• The third way to reduce the test generation cost is to define design for testability rules for behavioral descriptions as well. These rules propose modifications of behavioral descriptions which increase testability by adding some functionalities to the circuit behavior. One has to note that this process of designing for testability is dedicated to behavioral descriptions of digital circuits ; thus this process does not imply anything about the internal structure of devices. The only thing this process must do is seek ways for ensuring a good testability for a behavioral description while perturbing it as little as possible. These transformations of a behavioral description can be performed only during the design phase of an integrated circuit. If a circuit corresponding to the behavioral description we are dealing with has already been designed, the proposed testability enhancement cannot be applied.

The DFT methodology we propose is based on the following idea:

• When the behavioral description is declared difficult to test according to the previously introduced behavioral testability measures, we can use the controllability and observability measures associated with each basic element of the internal model (nodes involved in the control and data model) in order to select areas of the internal models which have to be modified in order to increase the weakest testability measures.

3: The internal modeling.

This section briefly presents the internal model derived from H.D.L. behavioral descriptions [9] . There are two levels of internal modeling : an external view and an internal view. The external view represents the inputs/outputs of the description. The internal view is made up of two parts, the separation and the interaction of which are explicit :

• A control model represented by a bipartite graph involving two kinds of nodes : transitions and places. This graph has the following two characteristics :

- the graph is represented in a hierarchical way insofar as a sub-graph can be associated with a transition. A transition with which a sub-graph is associated is called "decomposable". The upper hierarchical level transition is called the initial transition.

- the graph is structured. Four types of sub-graphs allow control structures to be modeled : sequential, selective, repetitive and parallel.

• A data model represented by a graph which has two types of nodes : operation and data.

- The data nodes which represent variables, signals and constants.

- The operation nodes which are of two types : assignment and decision. A sub-graph is associated with each operation node. This sub-graph represents the expression involved in the modeling operation. It is composed of nodes representing the basic operators involved in the language.

• The interaction between these models takes place thanks to the association of an operation node with a transition of the control model. Such a transition is called a non-decomposable transition and can therefore represent the assignment of a variable or a decision point for the selection of a branch. A transition with which an assignment node is associated is called operative. A transition with which a decision node is associated is called a control transition.

4: Brief overview of the test pattern generator.

In this section, we outline the main features of a deterministic behavioral test pattern generator [6] that we have implemented in CommonLISP in an object-oriented environment.

The deterministic test pattern generation algorithm for behavioral descriptions that we have defined comes from the framework of studies carried out by [3,4,5]. Once again, one has to note that this process of Test Pattern Generation is dedicated to behavioral descriptions of digital circuits ; thus the testing process is independant from the structure of the CUT (Circuit Under Test). Since it is a fault-oriented automatic test pattern generator, an exhaustive fault model has been defined on the basic elements of the internal model. A sub-set of conventional faults [8] has been selected for test generation. These fault hypotheses are classified as follows :

• on the elements of the data model : stuck-at fault hypotheses of an element of the data model,

• on the elements of the control model : hypotheses of a bad path selection in a selective control structure,

• on the elements of interaction between control/data : hypotheses of operation skipping.

Searching for a test pattern consists in solving three kinds of problems : problems concerning local fault effect manifestation (the first to be solved), constraint justification and fault effect propagation. The search for a solution involves a decision process. Whenever there are several alternatives to solve a justification problem or to propagate a fault effect, we choose one of them. But, by doing so, we may select a decision that leads to an inconsistency (conflict) and thereby involving a failure in the test pattern generation process. Therefore, in our

search for a test, we use a backtracking algorithm that allows a systematic exploration of the complete space of possible solutions. In order to minimize the number of incorrect decisions (and, as a matter of fact, the number of backtracks), methods have been defined in order to reduce the cost of the test pattern generation process.

5: Transformation of descriptions.

Because VHDL allows the designer to express the same functionalities in many different abstract ways, it is advisable to provide the test generation process with an optimal description according to complexity metrics. This section presents the method aimed at simplifying behavioral descriptions without adding functionalities. The goal of this method is to apply rules that modify behavioral descriptions in order to increase the performance of the test generation process. These modifications are performed depending on complexity metrics associated with the behavioral descriptions. To reach this objective, two techniques have been developed: a technique to compute the complexity metrics of behavioral descriptions and a technique to modify behavioral descriptions according to their computed complexity.

5.1: Metrics for behavioral descriptions.

Complexity metrics for behavioral descriptions are issued from software complexity metrics [10,11]. They are based on the characteristics of both the control model and the data model. The control model metrics reflect the difficulty to access each transition of the graph from the initial transition of the model. This difficulty is defined in terms of the number of control transitions to be crossed in order to reach each transition of the graph, i.e. the nesting level of each transition of the graph. This number is called the *accessibility index(AI)*. It allows the user to obtain two measures: the absolute measure (AAI) and the relative measure(RAI). The absolute measure defines the complexity independently of the number of transitions of the model. The relative measure is the average nesting level of transitions in the control model.
The data model complexity metrics estimate the difficulty in reaching each data node in the model. This difficulty is defined as the number of operation and data nodes contained in the cone of influence of each data node. The cone of influence of a node N is a sub-graph of the data model that includes the edges and nodes of all the paths that start at a primary input data node and terminate at N. This difficulty is estimated by defining two measures: the absolute complexity (ADI) and the relative complexity (RDI). The absolute measure represents the definition depth of data nodes in the data model. The relative measure is the average depth of data nodes in the data model.

5.2: Transformation rules.

In order to reduce the complexity of behavioral descriptions we have defined two types of transformation rules : optimizing rules and precedence rules.
Behavioral descriptions are represented by means of complex algorithmic expressions, data structures, procedures, processes, variables and signals and they may be described in several ways. In order to provide the ATPG with the least complex behavioral description possible, a rule-based technique is used to improve descriptions whose complexity is too great. This technique allows the user to modify the initial description by applying a set of transformation rules that allow those configurations having a less complex alternative to be identified and modified. These transformations are similar to those used in optimizing compiler techniques [15]. They attempt to limit the number of language artefacts of a textual procedural language and propose a more flexible representation with a more efficient control and/or data structure for testing purposes.
Another way to modify the behavioral description in order to obtain one less complex is to define a new sequencing between operations involved in the data model for which the execution results are the same as for the initial sequencing. Let us call the initial sequencing defined by the original description the reference chronology. It goes without saying that the reference chronology is an acceptable chronology. Depending on the chosen chronology, the test pattern generation methodology will be more or less performant. Thus, it may be relevant to make a choice among several chronologies before applying test pattern generation. The acceptable chronologies are defined from the dependency study carried out between the operations described in the reference chronology, taking into consideration the relation between their input and output data. The data dependency between operations or, more generally, between tasks [16] can be represented by a preorder graph which implicitly includes all the possible acceptable chronologies.
According to the complexity metrics presented in 5.1, before performing the actual test pattern generation, we select the acceptable chronology which has the lowest complexity.

6: Behavioral testability measures.

In this section, we give a brief overview of the different ways to compute the controllability and observability measures. More details about these computations have been presented in [13].

6.1: Controllability and observability of basic elements of the internal model.

Behavioral testability measures are dedicated to measuring the "difficulty" in solving a given justification or fault-effect propagation problem (see Section 4). Since elements of the control and data model are involved in the solving of such problems, we define testability measures

in terms of controllability and observability for each basic element of the internal model.
For the basic elements of the data model involved in the generation process :

- the controllability measures associated with an operation node indicate the difficulty in setting the output of the operation node at a given value and the observability measures indicate the difficulty in propagating a fault effect from the output of the operation node to a primary output.
- the controllability measures associated with a data node indicate the difficulty in setting the data node to a given value and the observability measures indicate the difficulty in propagating a fault effect from the data node to a primary output.

For the basic elements of the control model involved in the generation process :

- the controllability measures for a transition T indicate the difficulty in reaching the transition T from the initial transition and the observability measures indicate the difficulty in reaching an observable transition from the transition T.

6.2: Principles of controllability and observability computations.

Controllability and observability measures associated with elements of the data model (resp. control model) are computed by taking into account only information given by the data model (resp. control model). We have defined three different ways to compute the measures associated with the basic elements of the data model (distance-based measures, recursive formulae measures and fanout-based measures) while only two different ways are defined on the control model (distance-based and recursive formulae measures).

7. Behavioral design methodologies.

Another way to reduce the cost of test pattern generation is to use design methodologies for behavioral descriptions. These methodologies propose the designer a choice of descriptions modifications allowing the controllability and/or the observability of basic elements of the internal model to be increased. These modifications involve the addition of functionalities to the behavioral description; indeed they can be performed only during the design phase of an integrated circuit.
The objective is to propose a set of modification rules for the internal model allowing the designer to enhance the controllability and/or the observability of the basic elements. These enhancements are realised by creating new access paths toward basic elements of the description. The access paths can be generated by adding new test points or new statements. The addition of test points allows the designer to generate access paths to the poor controllability and observability areas. Two types of test points have been defined, referred to as control points and observation points. Control points are primary input variables used to enhance controllability; observation points are primary output variables used to enhance observability. A second way to create access paths is to add new statements to the description. These statements consist either in initializing some basics elements of the description or in creating selective access paths by defining two behavioral modes : test and normal modes. In the normal mode the behavior is the same as it was initially. The test mode allows the designer to activate the access paths to the poor controllability and observability areas.

8: Experiments.

In this section, we describe different experiments which have been carried out in order to validate the proposed approach. First, we present experiments regarding controllability and observability measures, followed by experiments done to validate complexity measures.

8.1 : Experiments concerning controllability and observability measures.

The objective of these experiments is to verify that controllability and observability measures speed up the search process of behavioral test generation and consequently reduce its cost .

		Methods						
		1	2	3	4	5	6	7
C1	Backtracks/fault	8.36	7.78	5.14	4.82	7.32	4.96	4.62
	Detected+Undetectable	45	46	48	48	46	48	48
	Dropped	5	4	2	2	4	2	2
	Rank	7	6	4	3	5	2	1
C2	Backtracks/fault	21.58	19.1	14.38	13.12	15.02	12.56	11.22
	Detected+Undetectable	27	32	38	40	34	40	42
	Dropped	23	18	12	10	16	10	8
	Rank	7	6	4	3	5	2	1
C3	Backtracks/fault	24.42	19.36	18.22	17.46	20.48	18.6	15.3
	Detected+Undetectable	14	28	31	33	28	30	35
	Dropped	36	22	19	17	22	20	15
	Rank	7	5	3	2	6	4	1
C4	Backtracks/fault	15.32	15.35	11.61	7.58	15.41	11.51	7.9
	Detected+Undetectable	22	21	23	26	21	26	30
	Dropped	9	10	8	5	10	5	1
	Rank	5	6	4	2	7	3	1
C5	Backtracks/fault	9.52	9.04	9.19	11.47	8.66	9.57	13.85
	Detected+Undetectable	21	19	19	17	19	18	16
	Dropped	0	2	2	4	2	3	5
	Rank	1	3	4	6	2	5	7
C6	Backtracks/fault	8.96	4.52	4.56	4.16	2.96	3.64	4.96
	Detected+Undetectable	20	22	22	23	24	22	21
	Dropped	5	3	3	2	1	3	4
	Rank	7	4	5	2	1	3	6

Table 1 : Results of the first experiment.

The approach used for the experiments stems from the work of Patel [14]. Seven methods have been used for the experiments. Method 1 (see Table 1) performs the test pattern generation process without any controllability and observability measures. The six others consist in performing the test pattern generation process by using the controllability and observability measures described in Section 6. The experiment has involved three behavioral descriptions of circuits (C1:ADD_BCD,C2:ALU8, C3:ALU64) and three experimental descriptions (C4, C5,

C6). The experiment results are summarized in Table 1. They allow us to highlight several important points. The first point is that among all the proposed testability measures, none is found to be consistently superior in all cases. The second point is that it will be more efficient to switch between the different controllability and observability measures than to use a sole and unique measure.

8.2: Experiments on complexity measures.

The objective of these experiments is to verify that using the complexity measures proposed in Section 5 allows the designer to reduce the cost of the behavioral test generation process. This reduction can be made by selecting the least complex description from among various descriptions of the same integrated circuit (IC).

The experiments consist in describing the same IC in different ways. The test pattern generation process is then performed for each description. These experiments have been performed on two circuits (C1 : ADD_BCD and C2 : a counter). The behavior of these circuits has been described in three different ways (d1 d2 d3). The experiments results are summarized in Table 2. They give for each behavioral description of the two circuits the complexity metrics defined in section 5 and the average number of backtracks per fault obtained after performing the test pattern generation process. They allow us to highlight that the cost of the T.G. process is reduced for the descriptions pointed out as the least complex according to the complexity metrics.

		AAI	RAI	ADI	RDI	Backtracks /faults
C1	d1	4	0.36	35	1.45	8.36
	d2	6	0.42	56	1.86	25.62
	d3	8	0.47	80	2.35	30.74
C2	d1	28	1.77	42	2	16.5
	d2	30	2	49	2.75	25.21
	d3	32	2.53	56	3.6	56

Table 2 : Results of the second experiment.

9: Conclusions and future work.

In this paper we have presented an approach for managing the testing related aspects of behavioral descriptions of digital circuits. The goal of this approach is to keep the cost of behavioral test pattern generation process into reasonnable bound, i.e. as low as possible. This goal is addressed by combining three kinds of methods:

. a method aimed at simplifying behavioral description without adding functionnalities.

. a method designed to assess the testability of behavioral descriptions and to use the related testability measure to speed up the test pattern generation process.

. a method defined to improve the testability of behavioral description with the addition of functionalities.

The two first methods have been used in experimental studies to show that they lead to an effective test generation cost reduction. Based on these experiments we can conclude that testability measures must be used in order to accelerate the behavioral deterministic test pattern generation process.

Experiments concerning the first method (involving both complexity measures and optimising rules) are in progress. We have reported some preliminary, but encouraging experimental results.

Our future work will concentrate on a number of investigations which are briefly described below:

. an interesting question appears to be whether benefits can be gained from the application of dynamic testability measures.

. it would be interesting to develop algorithmic techniques that definitely lead to improvements of behavioral ATPG.

References :

[1] "DACAPO III", user manual Version 1.0, Dosis GmbH, September 1988.

[2] "IEEE standard VHDL Language reference manual", IEEE standard 1076, 1987.

[3] D.S. Barclay, J.R. Armstrong, "A heuristic chip-level test generation algorithm", 24th DAC, 1987, pp.257-261.

[4] U.H. Levendel, P.R. Menon, "Test generation algorithms for computer hardware description languages",IEEE Trans. on computers, Vol. C-31, N° 7, July 1982, pp. 577-588

[5] F. E. Norrod, "An automatic test generation algorithm for hardware description languages," 26th DAC, 1989, pp.429-434.

[6] N. Giambiasi, J.F. Santucci, A.L. Courbis, V. Pla, "Test pattern generation for behavioral descriptions in VHDL", Euro-VHDL 91, Stockholm, September 1991, pp.228-235.

[7] S.Y.H. Su, T. Lin, Functional testing techniques for digital LSI/VLSI systems, 21th DAC, 1984, p. 517-528.

[8] S. Ghosh, T.J. Chakraborty, "On behavior Fault modeling for digital designs", Journal of electronic testing : theory and applications, 2, pp135-151 (1991).

[9] N. Giambiasi, J.F. Santucci, A.L. Courbis, M. Boumédine, "General behavioural modeling framework for hardware systems", EDAC, March 1990.

[10] T. J. McCabe , Structured Testing, IEEE Computer Society 1983.

[11] M. Halstead,Elements of Software Science, Elsevier North-Holland, New-York, 1977.

[12] M. Abramovici, M.A. Breuer, A.D. Friedman, Digital Systems Testing And Testable Design, Chapter. 9, Computer Science Press, New York 1990.

[13]J.F. Santucci, G. Dray, N. Giambiasi, M. Boumédine, "A methodology to reduce the computational cost of behavioral test pattern generation",29th DAC, 1992, pp267-272.

[14] S. Patel and J. Patel, "Effectiveness of heuristics measures for automatic test pattern generation," 23rd DAC, 1986, pp. 547-552.

[15] A. V. Aho, R. Sethi and J. D. Ullman, "Compilers : Principles, Techniques and tools, Addison-wesley", Ch.ap. 9, 1986.

[16] R. Cosnard, "Algorithme parallèle : une étude de complexité, TSI 1987, Vol. 6, N°2.