HAL
open science

Behavioral Fault Simulation for VHDL Description using DEVS Formalism

Laurent Capocchi, Fabrice Bernardi, Paul-Antoine Bisgambiglia, Dominique Federici

To cite this version:

Laurent Capocchi, Fabrice Bernardi, Paul-Antoine Bisgambiglia, Dominique Federici. Behavioral Fault Simulation for VHDL Description using DEVS Formalism. IEEE Pacific Rim Dependable Computing International Conference (PRDC), Aug 2004, Papeete, France. hal-00165465

HAL Id: hal-00165465

https://hal.science/hal-00165465

Submitted on 26 Jul 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Behavioral Fault Simulation for VHDL Description using DEVS Formalism

Laurent Capocchi, Dominique Federici, Fabrice Bernardi and Paul Bisgambiglia
University of Corsica, SPE Laboratory, UMR CNRS 6134
20250 Corte, France
{capocchi, federici, bernardi, bisgambi}@univ-corse.fr

1. INTRODUCTION

Due to the ever-increasing complexity of VLSI circuits, the use of VHDL behavioral descriptions (see [1] for more explanations) in the fields of test generation and fault simulation becomes advised. Test generation tools research has been oriented for the last fifteen years towards generation tools for circuits modeled at a high abstraction level according to a behavioral view as shown by [2, 3, 4, 5, 6, 7, 8].

The aim of the test pattern generation process is to define patterns to apply on primary inputs. This is used to test eventual physical defects, eg. the defects can be detected only if they induce an irregular behavior called a fault. The fault effect or error is measured by a difference between the state of the fault-free model (reference model) and the state of the faulty model (model in which a fault hypothesis is injected). The fault simulation process consists of simulating a circuit in the presence of faults, and comparing the results of fault simulation with the fault free simulation of the same circuit with the same input test pattern.

J. Lee and al. defined in [9] a fault simulator for behavioral descriptions; the inconvenient is that the circuit description is not purely behavioral. In their approach described in [10], P.C. Ward and J.R. Armstrong proposed a Behavioral Fault Simulation (BFS) method based on simulation of behavioral VHDL descriptions. However this fault simulation process involves as many VHDL simulations as number of fault hypothesis.

Thus, one of the main problems is that today the tools for test generation are unable to quickly and easily create and simulate behavioral fault models directly from the VHDL descriptions. A way to solve this problem is to encapsulate these descriptions in easily simulable and evolutive models using DEVS formalism [11] and to define a Behavioral Fault Simulator based on fast fault list propagation technique allowing the reduction of the number of VHDL simulation.

2. TRANSFORMATION OF VHDL DESCRIPTIONS INTO DEVS MODELS

Introduced by B.P. Zeigler in the early 70's, DEVS is a set-theoretic formalism that provides a mean of modeling discrete event systems in a hierarchical and modular way [11]. Using this formalism, we can perform more easy modeling in decomposing a large system into smaller component models with coupling specifications between them. DEVS defines two kinds of models: atomic models and coupled models. An atomic model is a basic model with specifications for the dynamics of the model. It describes the behavior of a component, which is indivisible, in a timed state transition level. Coupled models tell how to couple several component models together to form a new model. This kind of model can be employed as a component in a larger coupled model, thus giving rise to the construction of complex models in a hierarchical fashion.

Our choice for DEVS has been motivated by three main points. First, digital systems can be represented by discrete event systems, and DEVS is the leading formalism in discrete event modeling and simulation. Second, a great advantage of DEVS is that it separates the modeling and simulation parts in a very efficient way, since the simulator is built directly from the model. Last, An important feature of DEVS is the possibility to define, in addition to the model scheme, the way the various components interact in the time evolution. We can easily add a desired faulty behavior to our models by changing the transition and output functions and by adding a new fault transition function.

Our basic approach for the transformation is to associate each VHDL sequential statements with a DEVS atomic model, and each process with a DEVS coupled model. We defined a library composed by four basic atomic models. The first one is the "allocation model" kind that represents the VHDL allocation instructions. We can note that this model is able to handle time delayed allocations. The second is the "junction model" kind that represents how two atomic models are linked together or classical control instructions (end if, end case). The third is the "conditional model" kind that represents the classical control instructions (if, case, for,...). Finally, the last one is the "parallel model" kind that manages the parallel execution of processes. All these basic models can be easily personalized and combined in order to create complex behavioral models.

3. BEHAVIORAL FAULT SIMULATION USING FAULT LIST PROPAGATION TECHNIQUE

A fault modeling scheme allows to derive a set of fault hypothesis on the previously described model. Fault hypothesis are defined according to the elements involved in the model of the circuit under test. In order to have a behavioral fault model for which some measures of confidence are provided, we select fault hypothesis according to the fault model proposed in [12]. The selected fault hypothesis are classified into the following three groups:

- F1 : Stuck-at of a signal or variable object.
- F2 : Faults on conditional model. F2t (resp. F2f) : the true branch (resp. the false branch) of the conditional model is always selected whatever the resulting value implies on the end reporting value.
- F3 : Jump of statement instruction.

The first step for BFS is performed by the parser and consists in determining the global fault list. In the second step, we have to choose the test sequence which will be applied at the input of the circuit under test. It is obtained using the pseudo-random generator. The third step is the fault simulation. The free simulation is performed concurrently with the fault simulation thanks to DEVS transformation of the circuit under test.

The fault propagation is achieved using fault lists that are propagated through the DEVS model associated with the VHDL description under test. The propagation is performed according to the rules detailled below:

- Principle 1 - Generalities : The fault lists propagation is driven by the architecture of DEVS model.
- Principle 2 - Assignment : During the simulation with fault list propagation, when an assignment model ($D_1 \leq E(D_i)$ with E is a function with VHDL operators) is found, the fault list associated to D_1 is determined using the list associated to D_i representing the locally observable faults on D_i, the F3 type fault on D_1 and the D_1 list of faults calculated at the last cycle.
- Principle 3 - Selection: In the case of a selection model, the fault free path and the faulty path are determined. We put in no-safe ports all the faults involving a faulty path. The fault lists are propagated along the fault free path according to the principles 1, 2 and 3.

4. EXPERIMENTS AND RESULTS

The proposed BFS approach have been implemented in python and evaluated on behavioral VHDL descriptions designs of the ITC'99 benchmarks [13].

Table 1 shows the following information : the number of behavioral faults, the length of the test sequence obtained by the pseudo-random generator, the detected faults and the achieved behavioral fault coverage.

	Faults	T.S length	Detected Faults	Coverage (%)
B01	79	54	73	92.40
B02	48	56	41	85.45
B03	134	81	125	93.28
B04	105	72	95	90.47
B05	145	102	131	90.34
B06	95	72	82	86.31
B07	76	63	63	82.89
B08	64	70	53	82.81
B09	68	57	60	88.23
B10	163	110	150	92.02

Table 1: Fault Simulation Results for ITC'99

5. REFERENCES

[1] P. Ashenden, "The vhdl-cookbook," tech. rep., 1990.

[2] F. Ferrandi, F. Fummi, and D. Sciuto, "Implicit test generation for behavioral VHDL models," pp. 587–596.

[3] F. Corno, P. Prinetto, and M. S. Reorda, "Testability analysis and ATPG on behavioral RT-level VHDL," in *International Test Conference*, pp. 753–759, 1997.

[4] F. Ferrandi *et al.*, "Testing Core-Based Systems: A Symbolic Methodology," *IEEE Design & Test of Computers*, vol. 14, no. 4, pp. 69–77, 1997.

[5] C. C. M.D. Oneill, D.D. Jani and J. Armstrong, "Btg : a behavioral test generator," in *9th Computer Hardware Description Languages and their Application*, pp. 347–361, 1990.

[6] F. Norrod, "An automatic test generation algorithm for hardware description language," in *26th Design Automation Conference*, pp. 76–85, 1989.

[7] H. V. H.D. Hummer and H. Toepfer, "Functional tests for hardware derived from vhdl description," in *CHDL 91*, pp. 433–445, 1991.

[8] J. S. A.L. Courbis and N. Giambiasi, "Automatic test pattern generation for digital circuits," in *1st IEEE Asian Symposium, Hiroshima*, pp. 112–118, 1992.

[9] J. Lee and al., "Architectural level fault simulation using symbolic data," in *European Design Automation Conf. (EDAC)*, 1993.

[10] P. Ward and J. Armstrong, "Behavioral fault simulation in vhdl," in *Proc. Design Automation Conference*, pp. 586–593, 1990.

[11] B. Zeigler, H. Praehofer, and T. Kim, *Theory of the Modeling and Simulation, 2nde Edition*. Academic Press, 2000.

[12] S. Ghosh and T. Chakraborty, "On behavior fault modeling for digital designs," in *Journal of Electronic Testing : Theory and Applications*, pp. 135–151, 1991.

[13] S. Davidson, "Itc?99 benchmark circuits, preliminary results," in *IEEE ITC 99*, p. 1125, 1999.