Denotational Semantics of a Behavioral Subset of VHDL

Félix Nicoli
LIM - ESA 6077
CMI - Université de Provence
39, rue Joliot-Curie
13453 Marseille - France
email : nicoli@gyptis.univ-mrs.fr

Abstract

This paper introduces a denotational semantics of a behavioral subset of VHDL. This subset is restricted to basic data types only and does not allow `for` *clauses in* `wait` *statement. We consider the full model of time and resolution, we give a precise definition of the simulation mechanism. Easy translation rules from VHDL to Boyer-Moore logic can be derived from that semantics.*

1 Introduction

Hardware Description Languages and more particularly VHDL [4] are more and more widely used in modern CAD tools, therefore it is a major aim to give it a formal semantics suitable for synthesis as well as for verification. Many approaches have been proposed to fulfill that need. Since VHDL is a complex language each approach focuses on a restricted subset. Here we introduce a denotational semantics of a behavioral subset of VHDL. That semantics is inspired by works done in [1, 5]. The following table summarizes the VHDL features taken into consideration by our different approaches. + means that the feature is considered and − means that it is not.

Features	[Russinoff]	[Salem, Borrione]	[Nicoli]
ϕ time	+	+	+
δ time	+	-	+
signals	+	+	+
variables	-	+	+
behavior	-	+/-	+
structure	+	+	-
data-flow	+	+	+/-
asynchronism	+	-	+
resolution	-	-	+

We follow the methodology given in [1], that is, we split the semantics into three parts : a semantics of elaboration, a semantics of simulation, and finally, the semantics of the elaboration-initialization-simulation procedure. Our data structures for time and waveforms are similar to those of [5].

2 Behavioral Subset - Preprocessing

We consider descriptions that are composed of `process` statements that include basic sequential statements, `for` clauses are forbidden in `wait` statements.

Those descriptions are preprocessed as follows. Mainly, `process` statements are transformed so as they have only one `wait` statement as last statement. That translation has been proposed in [3]. The resulting `process` is a nested if-then-else shaped process where every condition is a reactivation condition. The last else-branch only contains a `null` statement. The last statement is a `wait` statement with the general resuming condition of the process (roughly speaking, the disjunction of every `wait` resuming condition of the original process). From a semantic point of view, that `wait` statement is useless because every relevant information stands in the if-then-else statement.

3 Denotational semantics

Roughly speaking, our semantics expresses how objects of a description (signals and variables) are modified by VHDL statements during a simulation interval. We have chosen to define that semantics in a denotational way, so that it can easily be encoded in the Boyer-Moore prover [2].

Our models of time and signal values correspond to structures informally described in [5]. A simulation time is a pair (a, b) of natural numbers where a corresponds to the physical simulation time and b stands for the delta simulation time. Signal values are denoted by waveforms. A wavefom is the ordered (by increasing simulation time) list of the events associated with a signal. An event is a pair (v, t) that means that a signal receives the value v at simulation time t. Given a

signal, we associate one waveform with each process in which it is assigned and another one called "*resolution waveform*" which stores the successive resolution values of the signal. In fact we extend resolution to non resolved signals in order to have an unified processing for every signal of the description. For a non-resolved signal, we consider that its resolution function is the identity.

We define two other semantic objects : the "*sensitivity list*" of an architecture and the list of resolved signals called "*resolution list*". As we perform resolution for every signal in a description, the *resolution list* is simply the list of every assigned signal (resolved or not). The *sensitivity list* is the list of the signal used in the computation of the next simulation time. It contains the signals that appear in the `wait` statement of every process statement.

The overall semantics is composed of four parts.

Since we assume that the descriptions only include processes, the semantics of elaboration mostly consists in denoting objects, giving them initial values and in building the sensitivity and resolution lists.

The simulation cycle semantics expresses modifications on variable values and signal waveforms at the end of a simulation cycle. It is the composition of the simulation semantic functions of the process statements included in an architecture. That means that we give a sequential meaning to concurrent instructions. This is possible because two different process statements are always modifying different semantic objects even when they denote the same syntactic object (in case of a resolved signal). Indeed two processes do not share the same variables and resolved signals are associated with one waveform per process in which they are assigned. So the process statements modify different waveforms. Because of the nested if-then-else shape of processes, the wait statement is no more useful in the semantics and is simply omitted. The simulation cycle semantic function of a process is the composition of the simulation semantic functions of its constitutive sequential statements.

The simulation loop semantics formalizes how variables and signals are modified after the iteration of the simulation interval. The next simulation time and resolution are computed in that semantic function. It expresses the iteration of resolution and simulation cycles as the least fixed-point of a functional. That fixed-point can be the undefined function.

The global semantics expresses the full simulation of a description. i.e. the initialization and simulation loop phases by the composition of the previous semantic functions.

4 Conclusion

We have proposed a denotational semantics of a behavioral subset of VHDL. That semantics takes into account fundamental aspects such as the model of time, the resolution mechanism and `wait` statements. It is currently limited to simple data types. That subset will be extended so as to take into account `for` clause of wait statements and shared variables defined in VHDL'93.

Because of the functional nature of that semantics, it is easy to implement in the Boyer-Moore logic. For a given design, we obtain a set of Nqthm functions on which both simulation and partial proofs are possible. We get functions related to the simulation cycle and one "simulation" function which expresses the simulation loop. Such a split of the simulation semantics allows to perform proof on a simulation cycle as well as on the full simulation loop.

References

[1] Dominique Borrione and Ashraf Salem. Denotational Semantics of a Synchronous VHDL Subset. In Robert K. Brayton, Edmund M. Clarke, and P.A. Subrahmanyam, editors, *Formal Methods in system design*, volume 7, chapter 3, pages 53–71. Kluwer Academic Publishers, August 1995.

[2] Robert S. Boyer and J S. Moore. *A Computational Logic Handbook*. Academic Press Inc (London), 1988.

[3] David Déharbe and Dominique Borrione. Semantics of a Verification Oriented Subset of VHDL. In Paolo E. Camurati and Hans Eveking, editors, *Correct Hardware Design and Verification Methods*, number 987 in LNCS, pages 293–310. Springer, October 1995.

[4] IEEE Press. *1076-1993 IEEE Standard VHDL Language Reference Manual*, 1993.

[5] David M. Russinoff. A formalization of a Subset of VHDL in the Boyer-Moore Logic. In Robert K. Brayton, Edmund M. Clarke, and P.A. Subrahmanyam, editors, *Formal Methods in system design*, volume 7, chapter 1, pages 7–25. Kluwer Academic Publishers, August 1995.