

TEMPORAL VERIFICATION of BEHAVIORAL DESCRIPTIONS in VHDL

D. BOUSSEBHA, N. GIAMBLASI, J. MAGNIER
LERI (Laboratoire d'Etude et Recherche en Informatique)
Parc Scientifique Georges Besse 30000 Nimes, France

Abstract.

This paper presents an approach for verifying the temporal sequencing of VHDL behavioral models. The goal is to verify that the control flow of a behavioral description satisfies the behavioral specifications described in a formalism based on reified temporal logics and on a notion of activity.

1. Introduction.

The rapid progress of VLSI technology in recent years demands effective verification for the complex hardware design. To ensure a successful hardware production, the hardware design should be verified at the first stage of the design, usually in a hardware description language (HDL) such as VHDL (VHSIC Hardware Description Language). With the increasing complexity and the sophistication of HDL, however, it is becoming increasingly difficult to ensure the correctness of a hardware description. This problem has prompted some researchers to study and develop a firm theoretical basis to verify a description of a hardware system. Mathematical methods have been proposed [1] to verify by formal proof, that a description meets its rigorous specifications. The formal verification requires : (1) the definition of a formal model, including the expression of the specifications and of the primitives of the HDL in terms of the formal model; (2) Extraction of the functional description from the hardware description, and the comparison of this description with the specifications.

In this paper, we propose a formal verification technique that consists in verifying the temporal behavior of a VHDL description [IEEE Std 1076.1989]. We are especially interested in the behavioral subset of VHDL defined in [2].

A VHDL behavioral description can be written in a declarative way by a set of concurrent processes. Each process is described in a procedural way : it is characterized by a control and data flow. We are interested here in the control flow which corresponds to the sequence of actions according to the control structures (sequential, parallel, iterative and selective).

The aim of this work is thus to verify that the temporal sequencing of these actions respects the sequencing defined by the behavioral specifications.

In our approach, we have used temporal reified logic [3] as a formalism for expressing the behavioral specifications. This formalism expresses a notion of activity that has been formally defined for the VHDL actions which represent a real circuit phenomenon . It also allows us to represent the temporal constraints between the actions explicitly defined by a procedural description. A preprocessing is developed to make the work of the demonstration process easier.

To facilitate the behavioral extraction mechanism, we have defined an internal model which highlights the separation and interaction between the control and data flows. The control flow in which we are interested is modelled into a interpreted and timed Petri Net. The extraction operation comes down to validating the execution paths on the Petri Net. We have then developed a demonstrator which allows us to show

whether the temporal constraints defined by the behavioral specifications are satisfied on the Petri Net.

This paper is structured as follows : section 2 introduces the temporal model that we have chosen to describe the behavioral specifications. The definition of the behavioral specifications and the specification formalism which we have defined are described in section 3. Section 4 presents the methodology of the behavioral extraction, including the demonstration technique developed. Finally, conclusions, experimental results and future directions are addressed in section 5.

2. THE TEMPORAL MODEL FOR BEHAVIORAL SPECIFICATIONS.

Shoham's logic [3] which we use to define the temporal entities of the specification formalism and the notion of activity is a variety of reified temporal logic. In this section, we first give a brief introduction to these logics. We then present a very brief overview of Shoham's logic. The rest of this section will provide the definition of the concepts of temporal entities and activity. Examples will be presented to illustrate these concepts.

2.1. Reified temporal logics.

Many approaches have been proposed to better understand and represent time. We have chosen a symbolic representation of time in the form of reified logics. These logics manipulate pairs : $\langle \text{logic assertion} : q, \text{temporal qualification of } q : t \rangle$.

Such a representation allows us to clearly separate the non-temporal logic component from the temporal component. Thus, the reasoning can be separated into two axes : one non-temporal (use of classic theorem demonstrators) and the other temporal (use of a time map manager).

The important contributions in this area are to be found in the work of Allen [4,5], McDermott [6], Shoham [3] and Kowalski and Sergot [7].

To deal with problems of complexity, monotonic and complete [8, 9, 10], we have chosen Shoham's logic [3] as the basis of the time representation. This formalism uses intervals as well as time points. The manipulation of such a representation scheme is based on the notion of time map managers, whose role is to keep the temporal relations base consistent and accessible and to update by adding or retrieving relations to the base. Ghallab's approach [11,12] has been chosen for the management of our temporal base. This choice is justified by the efficiency and good performance of the approach in managing the temporal graph including both the symbolic and numeric temporal relations.

We now present a survey of the characteristics of Shoham's logic. Ghallab's work will be described in the next section.

2.2. Shoham's logic

We will take Shoham's work as a basis. Shoham's logic defines in quite a general way a temporal proposition by the formula: True (t,p) which means that proposition p is true at 't' (point or interval defined by a pair of points : $I = (\text{start}(I), \text{end}(I))$).

Syntax and semantics have been precisely defined in [3].

We need to express intervals and time points, together with their relationships.

- qualitative relations between two intervals : before (<), meet (m), overlap (o), start (s), during (d), finish (f), equal (=), and their inverses (after (>), is-met-by (mi)... see [4]) :

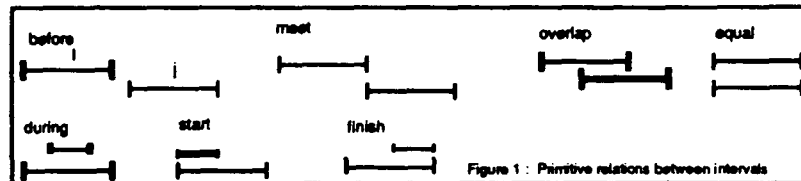


Figure 1 : Primitive relations between intervals

- qualitative relations between two time points or extremities of an interval : before (<), after (>) and equal (=).
- qualitative relations between an interval and a time point are : before, after, during, start-at, end-at (see figure 2). Other relations can be expressed by combining those above, eg : (start-before a T) is equivalent to (or (during a T) (end-at a T) (before a T)).



Figure 2 : Relations between an interval and a time point

From of this formal model, we have defined two concepts on which our specification formalism is based : temporal entities and the notion of activity [13].

2.3. Basic concepts.

Temporal entities express the notion event (pulse signal) which determines a change of state, corresponding to the transition of the proposition p : from $\neg p$ to p . If this transition is instantaneous, then this event (note $e(I, p)$) will be temporally localized by a time point. Otherwise (it is achieved with a duration), an event (called "fact" and note $f(I, p)$) will be localized by a time interval (see [13]).

In the timing model VHDL, the behavioral actions which model a physical reality in the circuit are : signal and variable assignments and the wait statement. Each action express a notion of activity which reflects the physical phenomenon that takes place in a circuit. For example, the inertial delay of a signal assignment.

As an example of activity, we will only define the transport model and we will illustrate how we have formalised this model. We refer the reader who is interested in the definition and the formalization of the other activities to [13].

• Example

In VHDL, there are two types of delay in signal assignment statements, inertial and transport delay. The example below shows the form of a transport delay :

$S1 \leftarrow \text{transport } Y \text{ after } 4 \text{ ns};$ ----- transport delay

The assignment statement expresses that all changes on Y will propagate to $S1$ regardless of how long the changes stay at the new level.

Consequently, the activity of a signal assignment transport-type is defined as (see example below) :

- the time necessary for the value of a signal to change, eg : the specified delay.

To formalize this activity, we have defined a temporal representation based on the concepts that have been defined previously (section 2.2).

Thus, the activity of a transport model will be temporally qualified by a time interval (I) over which it holds. This interval (I) represents the time span during which the change of a signal occurs. Consequently, the duration of activity of a signal of the transport type is equal to a given delay. Such an activity is defined by a triple :

$\langle \text{name (parameter), value-of-signal, duration} \rangle$.

A simple temporal specification of this activity is as follows :

$\text{Sig-Transp}(I, S) \leftrightarrow \text{activity}(\langle S \rangle, \langle \text{val-}S \rangle, \langle d \rangle) \wedge e(I, \langle S \rangle) \wedge \text{duration}(I, \text{delay})$

The "duration" predicate gives the correspondence between the interval " I " and its duration "delay". For example, the specification of a signal assignment to $S1$ is represented by :

$\text{Sig-Transp}(I, S1) \leftrightarrow \text{activity}(\langle S1 \rangle, \langle \text{value-of-}Y \rangle, \langle 4 \rangle)$

and it is illustrated by the following diagram :

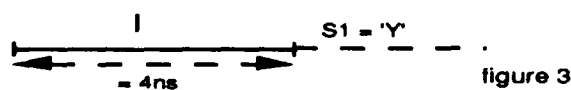


figure 3

3. BEHAVIORAL SPECIFICATIONS.

This section introduces a knowledge representation formalism and temporal structure that will enable us to express the behavioral specifications. They will be argued through simple example.

Specifications define the temporal behavior expected by a given VHDL description.

A specification program is a set of behavioral rules. Each rule defines one or several temporal behaviors. Each of these behaviors describes a list of activities to be undertaken if certain events or facts take place. The definition of a specification has two parts : one defines the conditions needed to trigger the activities and the constraints on these conditions, the other defines the implied activities and the temporal constraints between them. We have a description which is based essentially on the concepts defined in section 2.3 : the events and the facts. Such specifications will be describe as in the following example which defines a behavioral rule describing the intended behavior of a D-Latch :

```

whenever (CKL = 1) at I0, IN = 0 at I1
  if CKL at I2
    fact I2 end-at I0
  do affect OUT1 at I3 [2], affect OUT2 at I4 [2.85]
    action I3 starts I4.
  
```

Figure 4 : behavioral specification of a D-Latch.

The **<whenever>** and **<If>** fields define the conditions necessary (represented respectively in the form of events or facts) to activate the part **<Do>**, describing the set of activities : of the signal assignments (**affect**), of the wait statement (**wait**) or of the processes (**process**). The word **<at>** specifies the association of a condition or activity with the temporal objects "I" (a time point or an interval).

The **<Fact>** and **<action>** fields define, on the one hand, the temporal constraints between facts, and temporal relations between activities on the other.

The temporal constraints can be of two types : symbolic constraints (for example $I_1 > I_2$); and numeric constraints (for example $-10 < \text{start}(I) - \text{end}(J) < 5$).

The symbol [time] allows the duration of a condition or of an activity to be given.

To simplify a maximum the work of the demonstrator mechanism and to verify that the behavioral specification descriptions given are correct, we have transformed these specifications into structures much simpler to manipulate.

3.1. Compilation of the behavioral specifications.

The compilation is based on an original approach [11, 12], which proposes an very efficient time map manager (called IxTeT : Indexed Time Table) in the form of a time lattice using time points as tokens. The complexity of the access and updating operations is linear according to the size of the lattice. If the addition of a relation causes an inconsistency, the system can easily find the cause.

Consequently, thanks to this time map manager, the temporal constraints described in the part **<action>** and **<fact>** of the specification description are translated into a time lattice (figure 5) that is easy to manipulate.

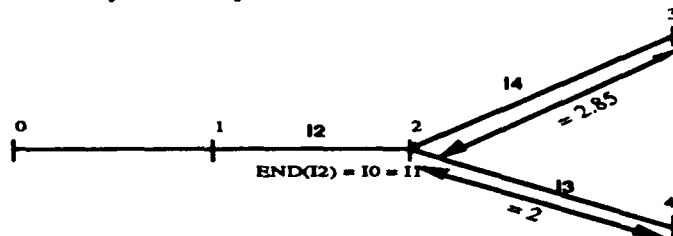


Figure 5 : time lattice associated with temporal constraints of specifications

Two tables (figure 6) which integrate the set of conditions and activities used, the time points in which they intervene (beginning and end) and a link indicating the cause of their insertion correspond to this time lattice.

condition table	0	1	2	5	4
CKL		<p	>		
CKL= 1			∅		
IN = 0			∅		
activity table	0	1	2	5	4
affect OUT1			<	>	
affect OUT2			<		>

Figure 6 : tables associated with time lattice.

The symbol "<" indicates the beginning of an activity, whereas the symbol ">" indicates the end of an activity. The letter "p" indicates the presence of the event as the activation condition for a process. The symbol "<>" indicates the instantaneous events.

4. DEMONSTRATION.

The first part of this section describes the strategy used for the extraction of the sub-behaviors from the VHDL procedural descriptions. The second part shows how the temporal demonstration is made. Each phase in the demonstration process will be illustrated with an example.

4.1. Principles behind the demonstration .

The model we wish to validate in a temporal way is written in the VHDL behavioral subset defined in [2]. A step in the demonstration process will consist in the extraction of the temporal behavior in order to verify its conformity with specifications. We use an internal model which has been developed in order to constitute the general model. Many applications are linked to it [14] : behavioral test, testability measure, symbolic simulation, etc... This internal model explains the data and control flows of a behavioral description. The control flow is modelled by a timed and interpreted Petri Net and the data flow by a graph structure based on the concepts of a hierarchical, multi-view model [15]. In our demonstration problem, we are only interested in the control model.

A data model action is associated with each Petri Net transition. Actions with explicit delay (signal assignment, wait statement) are associated with timed transitions; their transition life is considered to be equal to the delay value.

It should be pointed out that places indicate only the network state at the present time. We give, as an example, the translation of a behavioral description [16] modelling the behavior of the latch (figure 7).

```

architecture X of LATCH is
begin
  P1 : process
    constant T01 : TIME := 3 ns;
    constant T10 : TIME := 2 ns;
    constant T3  : TIME := 850 ps;
    variable DELAI : TIME;
    variable OO   : BIT;
  begin
    wait on IN, CKL;
    if CKL = '1' then
      OO := IN;
      case OO is
        when '0' => DELAI := T10;
        when '1' => DELAI := T01;
      end case;
      OUT1 <= OO after DELAI;
      OUT2 <= not OO after DELAI + T3;
    end if;
  end process P1;
end X;

```

Figure 7 : behavioral description in VHDL of D-latch.

This description is translated by the following interpreted and timed Petri Net, in

which O_i corresponds to the statements of the VHDL description :

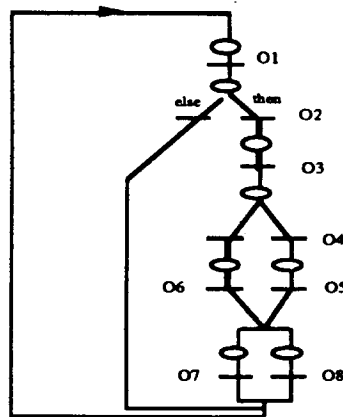


Figure 8 : petri net associated with process P1.

This translation is composed of a simple concatenation of the conditional subnets and of a parallel subnet [14]. The first subnet represents two conditional structures 'IF' and "case" of the VHDL description, and the second the parallelism of the signals O7 and O8.

4.2. Extraction of the temporal behavior from the Petri Net.

On the control model previously defined, the extraction of the behavior comes down to the classic problem involving the search for paths in a graph.

Indeed, the specifications express the conditions in which certain temporal relations should exist. These conditions define a path or a sub-graph of symbolic execution on the control model.

The extraction problem involves two phases : (1) To search for the valid path(s) in the graph (paths in which the decision-type nodes are specified in the conditions table) and calculate their durations; (2) To stress the temporal relations which can link the various actions of a valid path. For example, on the Petri graph modelling the process P1 (figure 7), the given specification expresses the conditions (CKL = 1 and IN = 0), which validate the path : $c_{v1} = (O1 O2 .. O5 O7 O8)$. The duration of a path is calculated according to the action-types which compose the path [13]. The previous valid path c_{v1} contains two action-type signal; their duration is defined by : $\text{duration}_{1j}(cv1) =$

$$\text{Max}_{i \in \{1..j\}} d(x_i); \text{ eg : } \text{duration}(cv1) = \text{Max}_{i \in \{1..2\}} [d(O7) d(O8)] = 2.85\text{ns.}$$

If we consider each node of a valid path as a time interval (or a time point) during which the action it holds, using the basic concepts defined in section 2, we can easily represent the temporal relations between the actions.

For example, the temporal relation which linked actions O7 and O8 is a relation-type 'start', because the two actions begin at same moment and are executed in parallel.

In summary, we obtain the following sub-behavior :

$$\text{Sub-Beh} = \{IO1 \text{ end-at } IO2, IO7 \text{ starts } IO8, IO2 < IO3, IO3 < IO4, \text{duration}(IO7) = 2, \text{duration}(IO8) = 2.85\}$$

This set of constraints will constitute the temporal sub-behaviors which will be compared with the behavioral specifications.

4.3. Demonstrator.

The principles of the demonstrator consist in :

1-verifying if the activities expressed by a valid path belong respectively to the table of activities. For example, activities O7 and O8 of the signal-type, in the valid path c_{v1} correspond well to those defined in the activities table (see figure 6).

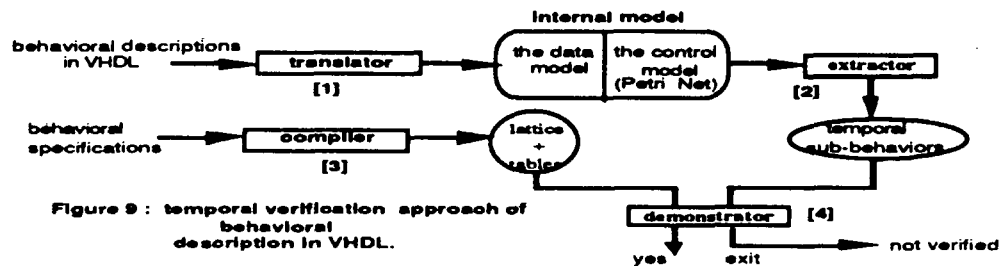
2- verifying that the temporal structure of this path is compatible with the one defined in the time lattice. This means verifying the compatibility of the numeric and symbolic constraints. These two properties describe the existence of a temporal path in the graph which is identical to the behavioral specification.

For example , the duration-type numeric constraint made explicit by action O7 in the path c_{v_1} is correct : $\text{duration}(I_{O7}) = \text{duration}(OUT1) = 2\text{ns}$.

The symbolic constraints expressed in the sub-behaviors "sub-beh" are satisfied by the time lattice. For example, in the "sub-beh" by taking two temporal propositions : (IO7, OUT1) and (IO8, OUT2) such that (IO7 starts IO8), we find the same propositions (OUT1 and OUT2) linked by the same temporal relation start in the activities table. In summary, we can conclude that the temporal sequencing of actions within the process P1 involved with the behavioral description of figure 7 meets the behavior specifications defined in Section 3.

5. Implementation and results.

Figure 9 below schematizes an overview of the architecture of the system :



The modules : translator, extractor, compiler and demonstrator, have been implemented in Common Lisp, using the Object Representation Language ORL [17], to support object-oriented programming. The system runs under Common Lisp V4.0 on a Sun 4 work station 360 running Unix. The system is in beta test and shows good performances. The following table shows the results in terms of running time for a benchmark of procedural behavioral descriptions in VHDL. It gives an idea of the lattice size and memory space for a number of behavioral rules (column (A)). The experimental results demonstrates that the algorithms used are linear (low level complexity) according to the number of actions (column (B)) in a VHDL description. The CPU times in the last column, indicate that the system is fast enough to be of practical use.

behavioral rules	lattice / memory (size) (A)	compiler (millisec)	VHDL description name	number of actions (B)	translator	extractor	demonstrator	global time
2	8 / 0.002 Kbytes	10	traffic light controller	12	22.8	23.2	4.97	41.0
12	60 / 0.08 Kbytes	30	ALU64	120	40.0	32.3	12.67	85.0
8	38 / 0.008 Kbytes	18	16-Bit CPU	35	37.3	15.8	6.93	60.5
5	21 / 0.005 Kbytes	15	74hc7537	21	26.78	16.3	6.0	49.0

Table : Run times (in seconds) for four conventional procedural behavioral descriptions.

Conclusion.

The work that has been presented in this paper has lead us to develop a technique for formal verification that allows us to verifying the control flow of a VHDL procedural description. We first defined a behavioral specification formalism based on Shoham's logic, which allows us to modelled the temporal concepts expressed by the VHDL language (delays, event, time wait, duration ..) and also to describe the behavioral

specifications as set of facts or events temporally linked.

From this formalism, we then established a verification technique which starts by extracting the temporal sub-behaviors from of given VHDL descriptions and then gives them to the temporal demonstrator to prove whether they respect the behavioral specifications.

In its current state, the system does not allow the temporal verification of the data flow. This will be the next task to undertake in order to complete this study. Other extensions are under consideration : new strategies for extraction and the algorithmic optimization of the system.

REFERENCES.

- [1] P.Camurati , P. Prienetto : Formal Verification of hardware correctness : an introduction; Proc. 8th Int. Conf. CHDL. North Holland. Amsterdam. April 1987.
- [2] G.N. NURIE, P.J. MENCHINI : VHDL Model Portability, High Performance, Systems July 1989, pp 76-85.
- [3] Y.SHOHAM : Temporal Logics in AI : Semantical and Ontological Considerations; Artificial Intelligence 33, pp. 89 à 104, 1987.
- [4] J.F. ALLEN : An Interval-Based Representation of Temporal Knowledge; Proc. 7th IJCAI, pp. 221-226, 1981.
- [5] J.F. ALLEN : Maintaining Knowledge About Temporal Intervals; Comm, ACM 2(11) pp. 832-843, Novembre 1983.
- [6] D. MC DERMOTT : A Temporal Logic for Reasoning About Processes and Plans; Cognitive Science, n°6, 1982, pp 101-105.
- [7] R.KOWALSKI, M.J. SERGOT : A Logic-Based Calculus of e vents, New Generation Computing; vol 4, pp 67-95, 1986.
- [8] J.F. RIT : Modélisation et Propagation des Contraintes Temporelles pour la Planifications; These de l'INP Grenoble, Mars 1988.
- [9] M.Vilain and Kautz H. Constraint Propagation Algorithms for temporal reasoning in Proc. AAI, pp 337 à 382, August 1986.
- [10] T. GRANIER : Contribution à l'Etude du Temps Objectif dans le Raisonnement; Rapport LIFIA, RR 716-I-73, Grenoble, Février 1988.
- [11] M. GHALLAB, A. MOUNIR-ALAOUI : The Indexed Time Table Approach for planning and Actions, NASA Conference on Space Telerobotics, 31 Jan-02 Feb 1989, Pasadena - USA.
- [12] M. GHALLAB, A. MOUNIR-ALAOUI : Managing Efficiently Temporal Relations Through Indexed Spanningtrees; Proc. 11th IJCAI, pp. 1297-1303, 1989.
- [13] D.BOUSSEBHA, N.GIAMBIASI, J. MAGNIER : première etude formelle du démonstrateur temporel; rapport de recherche n°2, LERI decembre 90.
- [14] N. GIAMBIASI, J.F. SANTUCCI, G.Dray : A Methodololy to Reduce the Computational Cost of behavioral Test Pattern Generation, to appear in Design Automation Conference june 1992 California USA.
- [15] M.OUSSALAH, N.GIAMBIASI, J.F.SANTUCCI : Expert System Based on Multi-view Multi-level Models Approach for Test Pattern Generation; Tzafestas,S.G,"Knowledge-based System Diagnosis, Supervision and Control", Plenum Publish, N.Y. 1989.
- [16] R. AIRIAU, J-M BERGE, V. OLIVE : VHDL du Langage à la Modelisation. Presses Polytechnique et Universitaires Romandes et CNET-ENST , 1991.
- [17] N. GIAMBIASI, M.OUSSALAH, L.TORRES : ORL, Clean Semantics of Multiple Inheritance, EUUG Spring 90, Munich.