A formal description of VHDL-AMS analogue systems

Tom Kazmierski, University of Southampton, tjk@ecs.soton.ac.uk

Abstract

A formal definition of the general VHDL-AMS analogue system has been proposed to relate the way in which the language affects the specification of a non-linear discontinuous analogue system. It has been suggested to model the break set as a separate system in order to facilitate the interaction between the analogue equation set and the digital abstract machine. The significance of the proposed model is that it may be used in semantic validation of VHDL-AMS description and may also facilitate mixed-signal equation formulation for an underlying VHDL-AMS simulator.

1. Introduction

An extension of VHDL to allow descriptions of mixed-signal and analogue systems that exhibit continuous behaviour has recently been published by the IEEE 1076.1 Working Group [1]. The new language, informally called VHDL-AMS is a superset of VHDL 1076-1993. Several public domain VHDL-AMS parsers are available for public use [2,3,4] and some VHDL-AMS validation examples [5] have been published. The key enhancements, which the new standard introduced to VHDL, are:

- support for quantities and terminals as abstract objects representing analogue waveforms and network connectivity,
- support for simultaneous equations in entity architectures to model analogue behaviour,
- support for break statements to model discontinuities in analogue waveforms.

Architectures with quantity declarations and simultaneous equations lead to descriptions of continuous or discontinuous non-linear dynamic systems to allow true analogue or mixed-signal. descriptions.

One of the most urgent needs is to provide VHDL-AMS developers with tools for analogue semantic analysis and verification. A future VHDL-AMS elaborator must be able to perform semantic error checks in a similar way to that used by a general programming language compiler, which runs a semantic analysis to verify whether each part of the program makes sense in its context. The complexity of VHDL-AMS makes any semantic analysis difficult unless it is supported by formal definitions of the VHDL-AMS behaviour.

A rigorous description of an abstract digital simulator using EA automatons has been proposed to cover the full behaviour of the VHDL 1079-1993 [6] and an attempt has been made to extend this result using the concept of Hybrid Abstract State Machines [7] to cover more general mixed-signal VHDL-AMS systems [8]. The latter work has demonstrated that a formalisation of VHDL-AMS models may not only provide suitable methodologies for the semantic analysis phase of VHDL-AMS elaboration but may also help to identify deficiencies in the draft standard of the language while it is still being validated.

This contribution proposes a formalisation of the analogue part of a VHDL-AMS model. A formal definition is given to relate the way in which the language effects the specification of a non-linear dynamic system and the interaction between the analogue and digital parts of the model. An example of how a simulation algorithm for an analogue system with hysteresis can be derived from the proposed model has also been presented.

It has been pointed out [8] that VHDL-AMS break statements are cumbersome and make an extension of the VHDL'93 abstract models based on abstract EA automatons to cover the full VHDL-AMS difficult. The model presented here separates break sets from the digital part of the hybrid automaton. Break sets are effectively a part of the analogue and not the digital part of the system because their existence may modify certain characteristic equations in the analogue part.

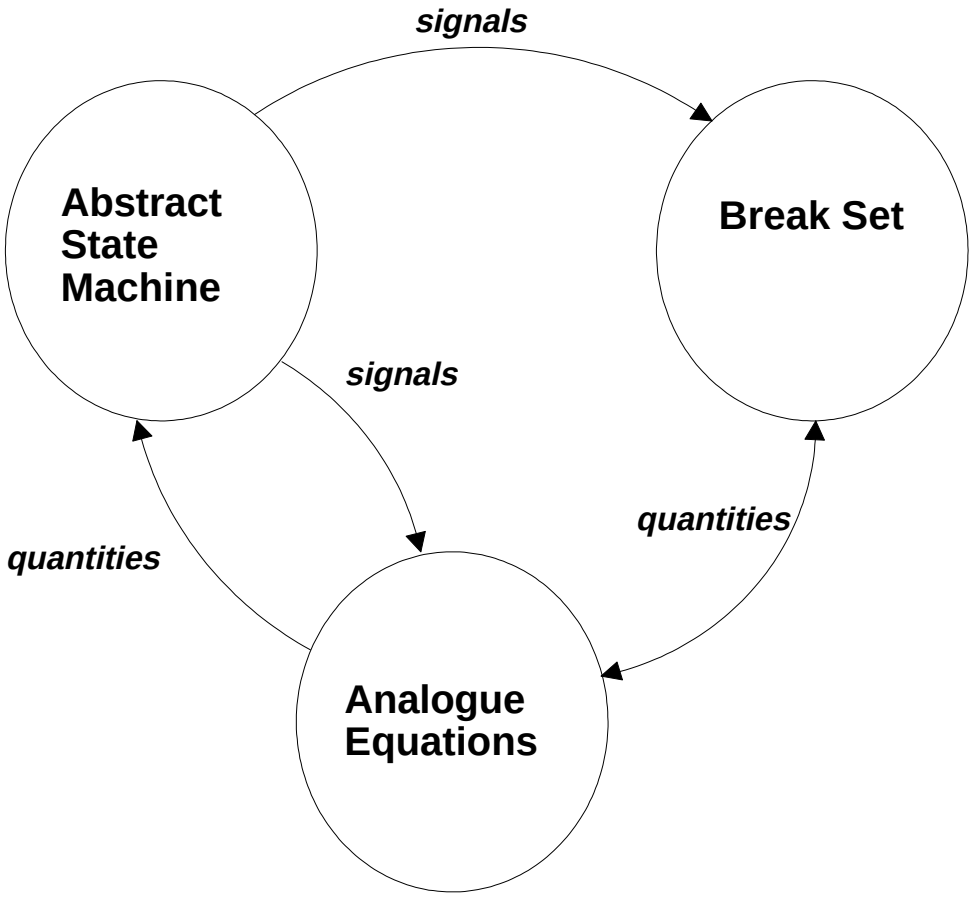

Figure 1. A VHDL-AMS Model.

2. A formal VHDL-AMS model

The extension of VHDL 1076-1993 to the analogue and mixed-signal domains has been implemented by enhancing the language with two new fundamental kinds of objects: quantities and terminals. Terminals specify the interconnections of network systems which are usually described using modular design blocks and quantities are value-bearing objects that are evaluated by solving the underlying dynamic equation set defined by the simultaneous statements of an elaborated design.

The dynamic behaviour of conserving systems can be modelled by using the following predefined attributes q'DOT, q'INTEG and q'DELAYED(T) of a quantity q. These attributes represent correspondingly: the time derivative of q at the time the attribute is evaluated, the time integral of q from time 0 to the time the attribute is evaluated and a quantity equal to q delayed by T with regard to the time point at which the attribute is evaluated. The differential, integral and algebraic equations associated with the analogue model are specified by means of simultaneous statements, which are allowed in architecture bodies, just like concurrent statements. VHDL-AMS processes may affect quantity values by executing break statements. This is a new feature of major significance since the execution of break statements intervenes with the analogue solver and may force discontinuities in the quantity waveforms. Thus, break statements provide a means of transmitting information from the digital to the analogue part of the model. T he predefined attribute q'ABOVE(E) of a quantity q, where E is an expression of the same type as q, denotes an implicit signal of the Boolean type. The value of each implicit q'ABOVE(E) signal is TRUE when q is sufficiently greater than E, FALSE when q is sufficiently less than E and undefined otherwise. The analogue solver assigns events to the drivers of all contradictory q'ABOVE(E) signals while the kernel process updates the signal values. A q'ABOVE(E) signal is set to be contradictory when its value is TRUE and q is sufficiently less than E or when its value is FALSE and q is sufficiently greater than E. The evaluation of this attribute by the analogue solver may give rise to new events and trigger processes

thus providing a means of information transfer from the analogue to the digital part of the model.

Let a VHDL-AMS system contain N signals and variables, including the implicit signals defined by the predefined attributes, and M quantities, including the implicit quantities defined by the predefined attributes such as 'DOT and 'INTEG.

Let $t_i : i = 0,\ldots,k; t_i \geq t_{i-1} : i = 1,\ldots,k; t_0 = 0$ represent the sequence of VHDL-AMS simulation time points at the simulation cycles $0,\ldots,k$, let $s_i \in \mathbb{R}^N : i = 0,\ldots,k$ represent the vectors of signal and variable values at these time points and let $q_k \in \mathbb{R}^M$ represent the vector of quantity values at the time point t_k. A formal mathematical model describing a full, mixed-signal VHDL-AMS system at the time point t_k can therefore be proposed as follows:

$$T_k(s_0,\ldots,s_{k-1},q_k,t_0,\ldots t_k) \rightarrow s_k \qquad (1)$$

$$B_k(s_0,\ldots,s_{k-1},q_k,t_0,\ldots t_k) \rightarrow q_k \qquad (2)$$

$$F_k(s_k,q_k,t_0,\ldots t_k) = 0 \qquad (3)$$

where: T_k represents the algorithm executed by an abstract state machine which models the concurrently running processes and set of transition rules applied by the VHDL-AMS kernel to update the values of s_k at the time point t_k, B_k is the break set at the simulation cycle k that forces new values of q_k and F_k is the set of characteristic expressions which constitute the algebraic, differential and integral equations defined by the simultaneous statements for the time point t_k. Note that F_k may be a function of the past time points $t_0,\ldots t_{k-1}$ if the predefined attribute 'DELAY(T) is used in the simultaneous statements of the model.

The application of this model consists of the steps presented in the following pseudo-code:

INITIALISATION
REPEAT -- simulation cycles k = 0,1,...
Execute the analogue solver to:
apply the break set B_k and solve eqn (3) to find q_k
assign events after 0.0 sec to the drivers of contradictory 'ABOVE(E) signals

Set current time Tc_k to next time Tn_k

IF Tn_k = TIME'HIGH and there are no active drivers THEN
Render the simulation complete
IF the simulation is not complete THEN
Execute the kernel process to update the implicit and active explicit signals

Resume active processes

Execute concurrently the non-postponed processes that have resumed

Calculate the next time point Tn_{k+1}

IF $Tn_{k+1} > Tc_k$ THEN
execute the postponed processes
re-calculate the next time point Tn_{k+1}

Increment the simulation cycle index k

UNTIL simulation complete.

The initialisation phase has been specified in sufficient detail in [1]. At each simulation cycle *k*, the analogue solver solves the analogue equation set F_k = 0 (3) and applies the break set B_k to determine the quantity values q_k. The application of B_k may override certain characteristic expressions in F_k. The analog solver suspends either when it reaches the next simulation time point or when a changing quantity q generates an event at its associated implicit signal q'ABOVE(E). When the analogue solver suspends, the active concurrent user processes execute until they suspend and then the kernel process resumes and updates the explicit and implicit signal values. The processes may execute break statements which form a break set to be applied by the analogue solver in the next simulation cycle.

3. Example

Consider the following VHDL-AMS description of an analogue Schmitt trigger behaviour [5] with the hysteresis loop shown in figure 2:

```
entity AnalogueSchmitt is
   generic( -- low and high threshold:
         Vtl: voltage := 1.2; -- low
         Vth: voltage := 2.4  -- high
          );
   port (
      -- input interface quantity
      quantity Vin: in voltage;
      -- terminal driven by a voltage
```

```
      -- source
      terminal OutTerminal: electrical
     );

end entity AnalogueSchmitt;
```

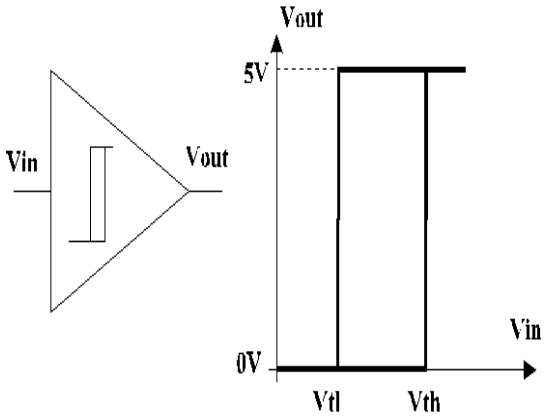

Figure 2. Analog Schmitt trigger.

```
architecture Hysteresis of
AnalogueSchmitt is
-- declare a signal to memorise the
-- hysteresis state:
  signal State: voltage := 0.0;
  -- initial state is low

-- declare a through branch for the
-- output voltage source
  quantity Vout across Iout through
         OutTerminal to ground;

begin
-- the architecture consists of two
-- architecture statements:
-- a conditional concurrent signal
--assignment to implement the
-- hysteresis
-- a simultaneous statement to
-- implement the output source equation

-- hysteresis:
  State <=     0.0 when Vin'ABOVE(Vth)
-- trigger event when Vin > Vth
   else 5.0 when not Vin'ABOVE(Vtl);
-- trigger event when Vin <= Vtl

-- output voltage source equation:
   Vout == State'RAMP;
-- the use of 'RAMP assures that
-- when a discontinuity in State
-- arises, it is announced to the
-- simulator
end architecture Hysteresis;
```

The analogue memory associated with the hysteresis behaviour is modelled by a signal of a floating type and a concurrent selected signal assignment statement. The process associated with the signal assignment statement drives the internal analogue state of the Schmitt trigger and is sensitive to the Vin'ABOVE(E) signal. Whenever Vin exceeds the high threshold or drops below the low threshold of the hysteresis loop, an event is scheduled on this implicit signal. The execution of the formal model defined by (1), (2) and (3) can be illustrated by the following pseudo-code:

INITIALISATION
REPEAT -- simulation cycles k = 0,1,...

Execute the analogue solver:
Evaluate the input excitation Vin
Evaluate Vout from the characteristic expression for
the voltage source equation Vout = State'RAMP
IF Vin'ABOVE(Vth) becomes contradictory
THEN
Assign an event to Vin'ABOVE(Vth)
after 0.0 sec
IF Vin'ABOVE(Vtl) becomes contradictory
THEN
Assign an event to Vin'ABOVE(Vtl)
after 0.0 sec
Set current time Tc_k to next time Tn_k

IF Tn_k = TIME'HIGH and the driver of State is inactive THEN
Render the simulation complete

IF the simulation is not complete THEN
Execute the kernel process to update signal State

IF an event has occurred
on Vin'ABOVE(Vth) or Vin'ABOVE(Vtl)
THEN
Resume the concurrent signal assignment

IF the concurrent signal assignment has resumed THEN
Execute it,

Calculate the next time point Tn_{k+1}

Increment the simulation cycle index k

UNTIL simulation complete.

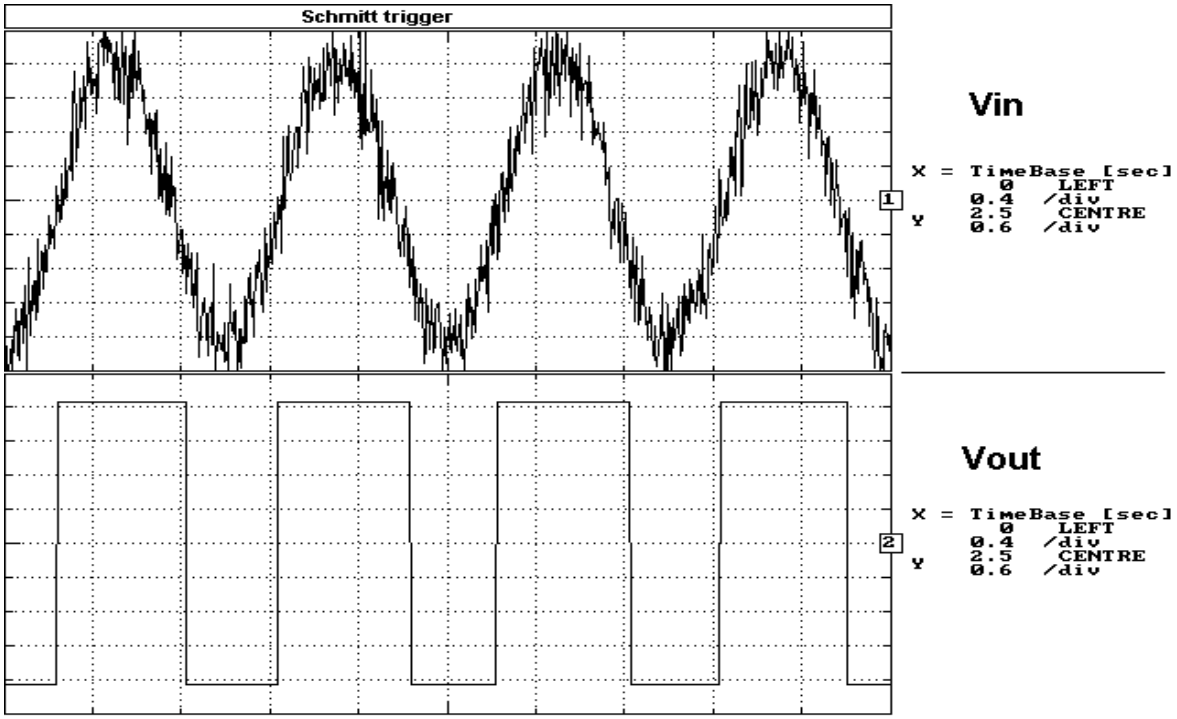

Figure 3. Analog Schmitt trigger simulation.

A test execution of this algorithm has been carried out with *TIME'HIGH = 4 secs* and *Vin = 2.5*(1+cos(6.28*(time+0.5 sec))) + 0.6*gauss_rnd*, where *gauss_rnd* is a random value of the normal Gaussian distribution The simulation results are presented in figure 3.

4. Conclusion.

A formal model has been proposed to specify the behaviour of arbitrary analogue VHDL-AMS systems. It has been demonstrated that the model is able to interact with formal VHDL 1076-1993 digital specifications based on abstract state machines. The model is independent on a particular analogue solver implementation and may provide a good basis for a semantic formalisation of VHDL-AMS. The proposed model separates break sets from the digital automaton that models the event-driven part of a VHDL-AMS description. The model may also facilitate mixed-signal equation formulation in an underlying VHDL-AMS simulator. However, in a development of a formal semantic model for VHDL-AMS, further attention needs to be given to break statements. They have been provided in the language in order to allow discontinuities in analogue waveforms by direct interaction between user processes and the analogue solver. There is as yet no proof that a rigid semantic validation of such a system is feasible.

5. References

[1] "IEEE Standard VHDL, Language Reference Manual (Integrated with VHDL-AMS Changes)", IEEE Std 1076.1, IEEE Inc., 1997.

[2] VHDL-AMS Analyzer Ver 0.4, Electronic Design Automation Research Center, Distributed Processing Laboratory, University of Cincinnati, 1997, http://www.ece.uc.edu/~vasu/index.html.

[3] VHDL-AMS compiler, LEDA S.A. 1997, http://worldserver.oleane.com/leda.

[4] VHDL-AMS Frontend, Java-Version, 1997, http://www.ti.informatik.uni-frankfurt.de/grimm/hybrid.html#VHDL-AMS.

[5] Southampton University VHDL-AMS Validation Suite, 1997, http://www.syssim.ecs.soton.ac.uk/validsui.html.

[6] Boerger E., U. Glaesser and W. Mueller, "Formal definition of an abstract VHDL'93 simulator by EA-machines", in C. Delgado Kloos and P.T. Breuer, eds., Formal Semantics for VHDL, Kluwer, 1995, pp. 1-39.

[7] Alur R., e.a. "The Algorithmic Analysis of Hybrid Systems" in Theoretical Computer Science 138:3-34, 1995.

[8] Sasaki H., K. Mizushima and T. Sasaki, "Semantic validation of VHDL-AMS by an abstract state machine", Report by Analog Task Group, EIAJ, June 1997