A Mixed-Signal Simulator for VHDL–AMS

Xiao Lyi  Li Bin  Ye Yizheng  
Microelectronic Center  
Harbin Institute of Technology  
Harbin, No.92 West Dazhi St. 150001  
Tel: 86-0451-6413405  
Fax: 86-0451-6415830  
e-mail: yeyizhen@public.hr.hl.cn

Huang Guoyong  Guo Jinjun  Zhang Peng  
China Huada Integrated Circuit Design Center  
Beijing, No.1 Gao Jia Yuan Dongzheximenwai, 100015  
Tel: 86-010-64365577-2008  
Fax: 86-010-64364487  
e-mail: huanggy@cide.com.cn

Abstract—Capable and efficiency simulators are in demand for designing complex analog and mixed-signal circuits and systems. With the standardization of VHDL–AMS, the demand is being realized. VHDL–AMS is an Analog and Mixed-Signal Extensions to VHDL. This paper introduces a mixed-signal simulator for it. The simulator was developed on the original VHDL digital simulation environment. An analog kernel has been integrated into the environment for the simulation of the continuous behavior of a model. The paper presents the algorithms adopted in the analog kernel and the synchronization of the digital and analog executions. The performance of the simulator is examined by mixed-signal examples.

1. INTRODUCTION

With the ever-increasing complexity of analog and mixed-signal systems, high level modeling and simulation have become more and more necessary. High level behavioral modeling and simulation provide many advantages to the designers. It enables a precise description of a system to be developed before considering implementation details. Where behavioral simulation is available, this early verification avoids the wasting of valuable design time and engineering resources and reduces design cycles. Several proprietary languages were developed in the mid.1980's and early 1990's to address the high level analog modeling. There are MAST[1], HDL–A[2], SpectreHDL[3] and so on. As these languages and simulation environment are completely independent, models developed for one system are not readily usable in other environment. So the application is highly restricted. Efforts to overcome these limitations have resulted in the development and standardization of VHDL–AMS[4]. It is a mixed-signal language and will play an important role in the specification and verification of mixed-signal systems.

The outline of the paper is as follows. The main concepts related to the analog and mixed-signal extensions to VHDL are introduced first. We focus on the mixed-signal interface mechanism. Next a mixed-signal simulator is presented, including its main structure, the analog kernel, and the synchronization method. The performance of the simulator is examined by mixed-signal examples. The paper ends with some conclusions.

2. THE MAIN CONCEPTS IN VHDL–AMS

VHDL–AMS is an Analog and Mixed-Signal Extensions to VHDL’93. It is the new IEEE standard 1076.1-1999. It supports the description and simulation of both non-conservative and conservative continuous and mixed discrete/continuous systems. It obeys the syntax and semantics of VHDL. In order to allow analog and mixed-signal descriptions, new constructs have been added in VHDL. Assuming the reader is familiar with VHDL, we give a brief description of the analog and mixed-signal extensions in VHDL–AMS.

Continuous variable in analog domain is defined using reserved word quantity. Quantity belongs to a new class of objects. It has piecewise continuous waveforms function of time and may only take floating point values. Some implicit quantities, such as Q'Dot and Q'Integ(Q is a scalar quantity) represent the time differentials and integrals of quantities respectively.

The other reserved word terminal is the second new object introduced in VHDL–AMS. It represents an interface port or an internal node in a continuous system. Terminal does not bear any value by itself. It is defined to belong to a specific nature.

The continuous systems are modeled using Differential Algebraic Equations(DAEs). Quantities are the unknowns in DAEs. DAEs are expressed in VHDL–AMS using simultaneous statements. The simultaneous statements may appear at the same level as VHDL concurrent statement. There are five forms of simultaneous statements. The basic form is the simple simultaneous statement, which has the following syntax:

simple_expression == simple_expression

The expression may include constants, literal, signals, quantities and user defined calls. In elaboration each simultaneous statement is rewritten to a form called characteristic expression which is the different between the value of the right-hand simple expression and the value of the left-hand simple expression.

Fig.1 shows a missile ballistic trajectory model in VHDL–AMS and the simulation results. We hope this model can help to understand what we have stated.
entity missile is
end entity missile;
architecture ballistic of missile is
constant v'0 real := 1000.0;
constant phi0 : real := 40.0*math.pi/180.0;
constant g real := 9.81;
constant air_res real := 0.001;
--quantity declaration
quantity x, z, t, v0 real;
quantity vx, vz real;
signal stopped boolean := false;
begin
--specify initial conditions
  break x := 0.0, z := 0.0, vx := v0*cos(phi0), vz := v0*sn(phi0);
--simple simultaneous statement
  --x coordinate
  x'dot = = vx;
  vx'dot = = air_res vx vx;
  z'dot = = vz;
  break on stopped;
--simultaneous if statement
  if stopped
    v'z dot = = 0.0;
  elsif vz > 0.0
    v'z dot = = -g - air_res vz vz;
  else
    v'z dot = = -g + air_res vz vz;
  end use;
--process sensitive to a Q'above(E) signal
  ctrl: process
    begin
      wait until not z above(0.0);
      stopped := true;
      --announce a discontinuity & reset
      --velocity values
      break vx := 0.0, vz := 0.0;
    end process;
  end architecture ballistic;

3. THE MIXED-SIGNAL INTERFACE MECHANISM IN VHDL-AMS

As a mixed-signal modeling and simulation language, VHDL-AMS allows the description of digital and analog behavior in a same entity. The communication of the two parts is important to mixed-signal simulation. We will description the interface mechanism more details in this section.

3.1. D to A Conversion

A new kind of statement, break statement, in VHDL-AMS can be used to D to A conversion. It has two forms, sequential break statement in process and concurrent break statement. When simulation, the concurrent break statement is equivalent to a process statement that has no sensitivity list, an empty declaration part, and a statement part that consists of a break statement followed by a wait statement. The execution of a break statement set a break flag and notifies the analog kernel a discontinuity has been occurred. It breaks the continuity of the analog kernel. It can also specify new initial values for quantities(see Fig.1).

3.2. A to D Conversion

In mixed-signal simulation, the information in analog parts must be able to be delivered to digital parts. The VHDL-AMS language uses a predefined attribute Q'above(E) for A to D conversion. The prefix Q represents a scalar quantity, E is an expression of the same type as Q. The kind of Q'above(E) is an implicit boolean signal. When the value of quantity Q is below the threshold E, the value of Q'above(E) is FALSE, and the value is TRUE when the value of quantity Q is above the threshold E. An event on this signal occurs when the sign of the expression Q-E changes. The event can trigger a process when the signal Q'above(E) is in the sensitive list.

The name of quantities can be used in any expression where a value of the type is allowed. A process can access the value of a quantity by simply using its name in an expression.

4. TIME DOMAIN MIXED-SIGNAL SIMULATION

Our VHDL-AMS simulation environment was constructed as shown in Fig.2. The environment was built for VHDL previously. We have extended the analyzer to be able to check the syntax and static semantic added in VHDL-AMS. An analog kernel has been integrated into the simulator. So it is suitable to simulate VHDL-AMS models. Each item in the environment is important. But we will focus on the elaboration and simulation phases. Assuming the VHDL simulation is familiar to us, we will emphasis on the elaboration and simulation of the continuous behavior of a model.

4.1. Elaboration

The elaboration of a design hierarchy creates a collection of process interconnected by nets and certain characteristic expressions. These characteristic expressions are grouped into four sets:
- **Explicit set**: formed by simultaneous statements
- **Structural set**: formed by declaration and association of quantities and terminals
- **Augmentation set**: each source quantity declared in the model and an implicit quantity of the form Q'Dot, Q'Integ and Q'Delayed(T) form three different augmentation sets. They are:
  - Quiescent state augmentation set
  - Time domain augmentation set
  - Discontinuity augmentation set

One of the augmentation sets combined with the structural set and explicit set play specific role in the simulation. At that time, it is the current augmentation set.

- **Break set**: formed by the execution of sequential break statements in each simulation cycle. It is a dynamic set. After application the break set is empty.

4.2. The Simulator

Two kinds of algorithms are integrated in the simulator. One is the event-driven algorithm for execution of the digital
The execution of a break statement signals the analog solver a discontinuity has occurred. In the next simulation cycle, the analog solver must consider the discontinuity.

4.2.2. The Analog Solver

The analog solver executes at the beginning of each simulation cycle. From time $T_c$ to $T_n$, it generates a sequence of ASPs. The execution process in the analog solver can be outlined as:

At the beginning of each simulation cycle{
    Analog solver resume at $T_c$
    If break flag is set, then
        Determine the discontinuity augmentation set
        Current augmentation set $\leq$ break set
        The analog solver computes an ASP
        Break set is empty, the break flag is cleared.
    Else
        Repeat{
            Guess $T_i, T_i \in [T_c, T_n], i \geq 1$
            Determine the time domain augmentation set
            If $T_i \geq T_n$ then $T_i = T_n$
            At each time point $T_i$, the analog solver computes an ASP
            If any implicit signal $Q'$ above(E) is contradictory then
                The driver of the implicit signal $Q'$ above(E) is active.
                The analog solver suspends
            Else
                Continue
        }

The analog solver consists of a number of algorithms required to solve the set of DAEs and also advance in time. We use the direct method to solve the DAEs systems. The solution method can be explained as:

For each time point in the simulation interval{
    Discretize the set of nonlinear differential equations using trapezoid method, get a set of nonlinear algebraic equations
    Find initial guess value for the quantities
    Linearize the system of nonlinear algebraic equations using Newton-Raphson method, get a set of linear algebraic equations
    Solve the linear equations using LU decomposition, get an ASP
    Check result for accuracy
    If accuracy then advance in time, repeat above steps
    Otherwise reduce time step and repeat from the linearization step
}

We can see that the computation cost for an ASP is comparatively high. The analog simulation time is the dominant component of the total time in mixed-signal simulation. Within the accuracy scope, we should reduce the number of the ASPs computed by the analog solver. When the analog solver estimates an event occurs on any signal $Q'$ above(E), it must track back and hit the point where the
crossing threshold E occurs. We adopt the secant method used in finding the root of a nonlinear equation to determine the exact time of the Q'above(E) event. The secant method is shown in Fig.3. In the time interval [T_{i-1}, T_i], if a value of quantity Q which is a prefix of a signal Q'above(E) crosses the threshold E, the analog solver must track back to find the exact crossing time. The secant algorithm is as follows:

**Secant algorithm**

Make a secant between the two sample points \([T_{i-1}, Q_{i-1}]\) and \([T_i, Q_i]\), get a crossing point with E.

The analog solver calculates an ASP \(Q'_i\) at the time of the crossing point.

If the difference of \(|Q'_i - E|\) is within the set limit, the exact time is found.

Else

If \((Q_{i-1} - E)(Q'_i - E) < 0\), make a secant between \([T_{i-1}, Q_{i-1}]\) and \([T'_i, Q'_i]\)

Else the secant between \([T'_i, Q'_i]\) and \([T_i, Q_i]\)

Repeat above steps

To prevent the algorithm from tracking too much of a time step, the user can give a desired accuracy limit. The secant search was compared with the bisection search through examples at the same accuracy limit(1.0e-6). The results are shown in Table 1. The secant method is more effective and converges to the root rapidly than bisection method.

### 4.2.3. The Synchronization between the Kernel Process and the Analog Solver

The VHDL-AMS simulation cycle precisely defines the synchronization between the kernel process and the analog solver. Actually, the digital event is the controller, the analog solver shake hands with the kernel process at each digital event time. It is a lock step synchronization approach[5] as shown in Fig.4. Assuming the kernel process and the analog solver are synchronized at Tc. The analog solver advances from Tc to Tn. When arriving Tn, the solver suspends, the kernel process resumes, this case is shown in Fig.4.(a). If a Q'above(E) event occurs at the time interval, the analog solver suspends, the kernel process must take it into account, then the kernel process and the solver are again synchronized, this is shown in Fig.4.(b). Fig.4.(c) shows at time Tn, a discontinuity occurs the analog solver must resume and calculate an ASP. The synchronization approach need not back tracking.

### 5. SIMULATION EXAMPLES

The simulator has been tested using a number of analog and mixed-signal examples. We choose two models to show the performance of the simulator.

Example one is an analog to digital converter. The input is a ramp voltage, the output is a 8 bit digital signals. The results are shown in Fig.5.

![Fig.3. Secant search for the crossing time](image)

**Table 1. The Compare Results of Secant and Bisection Search Method**

<table>
<thead>
<tr>
<th>Model</th>
<th>Backtracking Time(s)</th>
<th>Simulation Time(s)</th>
<th>Execution Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-state</td>
<td>2.16</td>
<td>1 min</td>
<td>6.39sec</td>
</tr>
<tr>
<td>Bouncingball</td>
<td>415</td>
<td>3.5min</td>
<td>3.2sec</td>
</tr>
<tr>
<td>AnalogSimit</td>
<td>320</td>
<td>40sec</td>
<td>9.91sec</td>
</tr>
</tbody>
</table>

SS = Secant search  
BS = Bisection search

![Fig.4.(a) No Q'Above(E) event occurs when the analog solver from Tc to Tn](image)

![Fig.4.(b) Q'Above(E) event occurs when the analog solver from Tc to Tn](image)

![Fig.4.(c) A discontinuity occurs at Tn](image)

The other is a mixed-signal phase lock loop(PLL) model. PLL often used as benchmark circuit for testing the performance of mixed-signal simulators. PLL is a feedback system that exhibits a strong nonlinear behavior especially when the phase difference of the two compared signals is large and the system is trying to null it. The architecture selected for the PLL is a charge-pump PLL. The block diagram of the PLL is shown in Fig.6.(a) The circuit is comprised of four components:

- The phase detector(PD), the PD compares the phase of the input reference frequency(fr) to the output from the VCO, and generates digital pulses Up and Down as output. The PD is described as a purely digital component at the behavioral level.
The charge-pump (CP), the CP delivers a pump current which is driven by Up and Down. The CP block is described as a mixed-signal model. The input signals are digital and the output is analog.

The loop filter (LF), a low pass filter converting the current from CP to an analog VCO control voltage. The LF is implemented as an analog structure model.

The voltage controlled oscillator (VCO), produces an output frequency based on the input voltage that is fed back to the phase detector. The VCO function is described at the behavioral level. The output is a sinusoidal waveform.

The top level of the PLL VHDL-AMS code and the simulation results are shown in Fig.6. The VCO has a central frequency at 15kHz and a slope of 2kHz/V. The simulation time is 10ms. The execution time is 1051.28sec (on a SUN ULTRA2 workstation). The PLL was locked after 6ms. Vc is the VCO control voltage.

---

**Fig.5. The input and output of ADC**

**Fig.6. (a) The PLL block diagram**

**Fig.6. (b) The digital signals in PLL model**

**Fig.6. (c) The VCO input control voltage**

**Fig.6. (d) The top level VHDL-AMS code of the PLL model**

---

6. CONCLUSIONS AND FUTURE WORK

We introduce a mixed-signal simulator based on VHDL-AMS language. It supports the simulation of digital, analog, and mixed-signal circuits and systems. The efficiency of the simulator was examined by mixed-signal examples. The simulator permits to verify the system behavior at early stage and facilitates exploration of different architecture options. Our future work will include the frequency and noise domain simulation. It will also be necessary to check the performance of the simulator with more complex examples.

---

**REFERENCE**


