Initialization of Mixed-Signal Systems in VHDL-AMS

Ken G. Ruan
Avant! Corp. Gemini Drive, Beaverton, OR 97008
TEL: (503) 626-9700, FAX: (503) 643-3361

Abstract

Complex systems, such as telecommunication systems, control systems, etc., most often consist of both analog circuitry and digital circuitry. It is more efficient in simulation to represent such complex systems as mixed-signal systems that both analog description and digital description are adopted.

VHDL-AMS is an IEEE standard which is designed to describe multi-discipline, mixed-signal systems. Three port types are supported in VHDL-AMS including terminal, quantity and signal. Properties of a port vary significantly with port type. It is crucial to have good understanding to these properties which have significant impact on the behavior in simulation.

DC analysis is the most essential analysis for mixed-signal systems that leads the system to an appropriate initial point for transient analysis.

This paper first discussed the port properties of all three port types. Then, the important language features in VHDL-AMS for mixed-signal system description are discussed. A generic model for mixed-signal systems and the criteria that a DC solution of a mixed-signal system is reached is presented. Then, analysis on the interaction between different signal types in DC analysis is presented to explore the common issues in mixed-signal system simulation. Finally, with an interface model as an example, DC simulation results of a mixed-signal system are presented and analyzed. The technique for modeling mixed-signal system in VHDL-AMS are summarized.

1. Introduction

VHDL-AMS is truly a superset of VHDL [1]. It supports description of mixed-signal systems and is capable of describing multi-discipline systems, including electrical, mechanical, hydraulic systems, etc., at various levels of complexity.

VHDL-AMS supports three port types including TERMINAL, QUANTITY and SIGNAL. TERMINAL is a type of port that energy conservation is maintained. Two types of quantity, including through quantity and across quantity, are associated with a TERMINAL. Energy conservation is maintained by the constrain that sum of all through quantities associated with one terminal is equal to zero. TERMINAL may have different NATURE that is defined for a particular discipline. There are five packages proposed by IEEE design automation standards committee working group that a number of popular natures are defined [3]. TERMINAL is more adequate in representing a connection of a physical system. However, due to the energy conservation constrain, it is also the most expensive type from computational efficiency point of view.

A port in QUANTITY type is also for analog connection. Its waveform is continuous. But, it does not comply with energy conservation law. It is adequate to consider a QUANTITY port a connection of an ideal driver that is capable of providing unlimited current. It is useful for a port that is “unidirectional”.

The third port type is SIGNAL. SIGNAL may be in different type, such as real, integer, logic, etc. In VHDL-AMS, the predefined logic type is std_logic. The waveform of a signal consists of a sequence of transactions. A transaction is represented by a (value, time) pair. The type of value is the same as the signal, time is the instance when the transaction is active that the value of the signal is updated to value. A change from one transaction to another in a signal is said an event occurs on that signal.

Change of a logic signal is discrete in both value and time. Change of real signal is continuous in value in the sense that any number in REAL can be taken, however, discrete in time that a value can only be taken at discrete time instances. The waveform of a signal is a high level abstraction of an actual waveform. No analog solver is involved in resolving the
A variable signal assignment statement is used only to continues until simulation ends. Since this is in a loop statement, the process is then generated again. Since this is in a loop statement, the process continues until simulation ends.

In this section the most important issues of mixed-signal system modeling in VHDL-AMS are discussed. Although our discussion is focused on using VHDL-AMS, the fundamental concept should be useful in dealing with other description languages.

A. Coupling of logic signal and quantity

Since properties of logic signal and real signal differ significantly, the two signals are discussed separately.

A logic signal can be generated from a quantity Q using Q’above(E) attribute where E is an expression. Q’above(E) returns ‘TRUE’ if Q-E > 0.0. ‘FALSE’ if Q-E < 0.0. Otherwise, returns Q. Following is an example to demonstrate how the coupling can be done in VHDL-AMS. It is a process.

```vhdl
a2d: process
  begin
    region := zero;
    if vin >= vxh then
      region := one;
    elsif vin > vxl then
      region := unknown;
    end if;
    LOOP
      CASE region is
        when one =
          d <= logic_high;
        when unknown =>
          d <= logic_unknown after td;
        when zero =
          d <= logic_low;
      END CASE;
      WAIT on vin’above(vxh), vin’above(vxl);
      above_vxh := vin’above(vxh);
      above_vxl := vin’above(vxl);
      region := check_level(above_vxh, above_vxl);
      END LOOP;
  end process;
```

The input voltage vin is a quantity. There are two threshold levels vxh and vxl that partition a real into three regions named zero, one and unknown. At the beginning, region is determined according to vin. Then, the model enters a loop. Based on region, a logic signal d is generated using signal assignment statement. The model then waits until vin crosses either vxh or vxl. Two logic variables above_vxh and above_vxvl are assigned return values from the ‘above attributes and are used by check_level function to determine next region. A new event on the digital signal d is then generated again. Since this is in a loop statement, the process continues until simulation ends.

Note that, except vin, only d is a signal, all others are variable. signal assignment statement is used only to d. The advantage to use variable assignment is that the new value of a variable is available immediately after the assignment and therefore can be used by the following statements. For a signal assignment, its value will be available after the event is processed. Even the delay time is zero, there is still a delta cycle delay in simulation. Refer to section 12.6.4 The simulation cycle delay in simulation. Refer to section 12.6.4 The simulation cycle delay in simulation. Refer to section 12.6.4 The simulation cycle delay in simulation. Refer to section 12.6.4 The simulation cycle delay in simulation.
3. A generic model of mixed signal system

A generic model of a mixed-signal system can be derived from the generic model of an A/D converter [5] and is presented in Figure 2.

![Block diagram of a hypermodel mos_d2an coupling a digital output to an analog input](image)

**Figure 2. A generic model of mixed-signal system**

Block A is the subsystem that all blocks in this subsystem are analog, i.e., using only quantities. The input port and output port of this subsystem are either Quantity or terminal. Block D is the logic subsystem using only logic signal in representation. Block R is the subsystem that uses non-logic signal in representation. Both R and D blocks are represented in Canonical form for asynchronous system [2]. The coupling among these subsystems are also illustrated. A2D is a logic signal converted from a quantity, R2D stands for a logic signal converted from a real signal. D2R is real signal. R2A is a quantity converted from a real signal. An appropriate conversion process as described in previous section is involved in each coupling path.

Note that there are no “A2R” and “D2A” paths in this model. D2A is accomplished by two steps, D2R and R2A. A2R does not have practical meaning as discussed earlier.

Initial state is regard as the state where a time domain simulation starts. In analog system, initial state is a DC solution of the system. Historically, there is no DC solution concept in digital system. Time domain simulation may start with/without initialization process. Actually, initialization process itself is a time domain analysis that the input vectors are predefined [2].

It is natural to extend the DC solution concept in analog system to cover the entire mixed-signal system. The initial state of a mixed-signal system is a DC solution of the system. From now on, these two terms “Initial state” and “DC solution” are equivalent in a mixed-signal system.

Based on the generic model in Figure 2, the state of a mixed-signal system can be described by equations (1),

\[
\begin{align*}
(A_o' A2D) &= A(A_i', R2A) \\
(D_o' D1' D2R) &= D(D_i', D1', A2D, R2D) \\
(R_o' R1' R2A, R2D) &= R(R_i', R1', D2R) \\
R_{i1} &= \Delta(R_{o1}) \\
D_{i1} &= \Delta(D_{o1})
\end{align*}
\]

(1)

where \(\Delta\) is a delay operator. The initial state of a mixed-signal system can be defined as a solution of equation (1) at time \(t = 0.0\) while all inputs \(A_i, D_i, R_i\) are known.

The execution of a model consists of a initialization phase followed by the repetitive execution of process statements. Each such a repetition is called a simulation cycle [1]. See Figure 3.

When analog solver is about to start, all signals are initialized. All quantities are set to an initial value which is normally a value determined by a simulator, for example, 0.0 [6]. When the solver reaches a solution, value of quantities \(A_o\) may change. This change may cause changes in \(A2D\) that is implemented using \(Q'\) above attribute in VHDL-AMS. By definition, \(Q'\) above is one of implicit signals. Therefore, both explicit signals and implicit signals are updated, i.e. signals on the left hand side of the second through fourth equations in (1), and any activated process are executed until it suspends, for example, reaches a wait statement.

The updated signal may cause new events. Some of the new events may have zero delay, i.e. the **time** component in a transaction is equal to the current global time. These zero delay events lead the simulation to a delta cycle that global time will remain unchanged, \(T_n = T_c\). A simulation cycle will not end until there is no zero delay event to process.

![Depiction of a simulation cycle](image)

**Figure 3. Depiction of a simulation cycle**

Note that the simulation cycle shown in Figure 3 is valid for both DC and Transient analysis. The only difference is \(T_n\) does not advance during DC analysis.

D. A **mos_d2an** hypermodel

A **mos_d2an** hypermodel is used as an example to explore issues in DC analysis of mixed-signal system.

A block diagram of a hypermodel **mos_d2an** coupling a digital output to an analog input is shown in Figure 4.

A logic signal is applied at the digital input port \(d\). Two d/a converters are used to convert the digital signals into analog voltage to drive the gates of two mosfets. The output is obtained from the output stage consisting of a p-type mosfet and a n-type mosfet.
and vds_snk are quantities, they all have their initial value, 0.0 and vhigh is 0.0. vth_snk is a constant, therefore, viow is vth_snk. vx will be 0.5*vth_snk. All elements in src_table and snk_table are set to these values, accordingly. The inverter output d_inv is equal to the inverse of d. Events on real signals veff_snk and veff_src are created and have the values viow and vhigh, respectively. Note that at this moment, vhigh = 0.0. Tn is set to 0.0 at the end of initialization phase.

Now a simulation cycle starts. The first thing to do is to invoke analog solver to resolve analog quantities.

Since vgs_src, vgs_snk are below threshold voltage, channel currents ids_snk and ids_src are 0.0. Due to the load resistor R_load, the output voltage at terminal a is 0.0. There is no signal to update and no more events to process, the simulation ends. The DC solution of this case is Vout = 0.0 even d = ‘H’. This is certainly not an expected solution.

The reason to reach an unexpected solution is that real signals veff_src and veff_snk are depending on the quantities vds_src and vds_snk. The solution of vds_src + vds_snk = Vcc which is 5.0 can only be available to the process p1 after the execution of analog solver. However, p1 process is active only when there is an event on the input signal d which has no change in this case. Therefore, the change in vds_src and vds_snk has no affect on veff_src and veff_snk, and vgs_src and vgs_snk stay at the same value 0.0 and vth_snk, respectively. Consequently, ids_src and ids_snk are still zero, so does the output Aout that stays at 0.0.

One may think of using vcc'above(vth) to detect the change in Vcc. The answer is NO! At the moment when a transaction on the implicit signal vcc'above(vth) is generated, the value of vcc is equal to vth. In general, vth cannot be equal to the final value of Vcc since it is not known to the model and any assumption on the value of Vcc cannot be valid in all situations.

An efficient way to resolve this issue is let veff_src and veff_snk not to depend on quantity. The block diagram of a revised mos_d2an model is shown in Figure 6.

Quantities Vgs_src and Vgs_snk are now depending on the switch state which is controlled by the input state. A switch is CLOSED when the control signal is high. It is OPEN when its control signal is low. Assume the on resistance ron and off resistance roff are identical to all switches.
sw1 through sw4. When d = ‘H’, Vgs_src, Vgs_snk can be expressed as:

\[
\begin{align*}
\text{vgs_snk} &= Vcc \times \frac{\text{ron}}{\text{ron} + \text{roff}}; \\
\text{vgs_src} &= Vcc \times \frac{\text{roff}}{\text{ron} + \text{roff}};
\end{align*}
\]

vgs_snk, vgs_src and vcc will be resolved simultaneously by the analog solver. Since roff >> ron, vgs_snk close to zero, vgs_src close to Vcc. Under this condition, the upper device is conducting and lower device is shut-off. Then, Aout close to Vcc. The results are as expected.

For comparison, simulation results using the two mos d2a hypermodels are presented in Figure 7. The digital input state is plotted at the top. Its initial state is ‘H’. The upper analog waveform is the output of the original mos_d2an hypermodel. The lower waveform is the output of revised mos_d2an hypermodel. It can be seen that the initial value in the upper waveform is close to zero. However, the initial value in the lower waveform is close to Vcc.

Refer to Appendix A. At the end of DC analysis, veff_src is zero. When transient analysis starts, an event occurs on the DOMAIN signal [1] and activates p1 process. The execution of the signal assignment statement in p1 updates the value of veff_src and veff_snk. Due to the simultaneous statements implemented for R2A path, vgs_src ramps up from 0 to Vcc and the upper mosfet becomes conducting. A source current flows from terminal p to terminal a and pulls up the voltage at a. That is why a transition from 0.0 to Vcc occurs in the waveform at a. Refer to the upper aout in Figure 7.

4. Summary

The DC analysis process for a mixed-signal system is described based on the generic model proposed in this paper. The simulation cycle in VHDL-AMS is analyzed to explore crucial factors for obtaining correct DC solution of a mixed-signal system in VHDL-AMS.

The technique used in mixed-signal modeling can be summarized as the following:

1. Choose port type carefully. Use Terminal for ports where analog characteristics are crucial. Use signal port for high computational efficiency. Use quantity for analog port where through quantity is not a concern.
2. Use Q'above to generate a logic signal from a quantity;
3. Use S' ramp for converting a signal to a quantity to ensure that the quantity is continuous;
4. Due to the fact that the value of a quantity can only be available after a DC solution has been reached and the value of a signal can only be updated when there is an event occurred, therefore, the direct dependency of real signal on a quantity should be avoided;
5. Use variable wherever possible. Use signal only when that signal is needed to activate a process or the timing is crucial;
6. Signal assignment with zero delay causes a delta cycle. Never create a closed signal path that total propagation delay through the path is zero. Because, this will cause unlimited number of delta cycles.

References

[7] www.VHDL-AMS.com, the web site dedicated to VHDL-AMS developers and users. Sponsored by Avant! Corp.
Appendix A. A mos_d2an hypermodel.

This is an example to show how a mixed-signal model can be written in VHDL-AMS. It couples a digital signal to an electrical terminal. This model uses two types of port, including logic signal and electrical terminal, two types of signal, including logic and real, three types of quantity, including across, through and simple. It is truly a mixed-signal system.

The implementation in VHDL-AMS is shown below.

```
library  ieee;
use  ieee.std_logic_1164.all;
use work.energy_systems.all;
use work.electrical_systems.all;
use work.ai_standard.all;
use work.semiconductor_devices.all;
use work.hypermodels.all;
use work.hypermodels.all;
use work.ai_standard.all;
use work.electrical_systems.all;
use work.energy_systems.ALL;

entity mos_d2an is
  generic (model : mos_hypermodel_model := mos_hypermodel_init);
  port ( signal d : in std_logic := 'U'; -- Digital signal input
    p,                   -- Power Supply
    m : electrical );    -- Reference Ground
end entity mos_d2an;

architecture Simple of mos_d2an is
begin
  p1 : process
  variable snk_table : real_table;
  variable src_table : real_table;
  variable d_inv : std_logic;
  variable vhigh, vlow : real;
  variable vh : real;
  variable vx, veff_snk, veff_src, reff, r_table := (vn, vn, vn, vn, vn, vn, vn, vn, vn);

  vgs_snk := veff_snk'ramp(tt);
  vgs_src := veff_src'ramp(tt);
  icout := cout * vds_snk'dot;
  vds_snk := ids_snk := ideal_channel(betasnk, vth_snk, lambda_snk, vgs_snk, vds_snk);
  vds_src := ids_src := ideal_channel(betasrc, vth_src, lambda_src, vgs_src, vds_src);
  veff_snk := vth_snk;
  veff_src := vth_src;
  vlow := vlow;
  d_inv := NOT d;
  if d = 'Z' THEN
    -- Both source and sink mosfets are cut off.
    veff_snk <= vth_snk;
    veff_src <= vth_src;
  elsif d = 'X' and d = 'U' and d = '-' THEN
    -- The previous voltage will be maintained.
    veff_snk <= src_table(d);
    veff_src <= src_table(d_inv);
  end if;
  wait on d, DOMAIN;
end process;

-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents
-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents

FUNCTION ideal_channel(beta, vth, lambda, vgs, vds : real) RETURN real IS
  vdsp := vds;
  vgsp := vgs - vth;
  r_table := (vn, vn, vn, vn, vn, vn, vn, vn, vn);
  variable snk_table : real_table;
  variable src_table : real_table;
  variable d_inv : std_logic;
  variable vhigh, vlow : real;
  variable vh : real;

  vgs_snk := veff_snk'ramp(tt);
  vgs_src := veff_src'ramp(tt);
  icout := cout * vds_snk'dot;
  vds_snk := ids_snk := ideal_channel(betasnk, vth_snk, lambda_snk, vgs_snk, vds_snk);
  vds_src := ids_src := ideal_channel(betasrc, vth_src, lambda_src, vgs_src, vds_src);
  veff_snk := vth_snk;
  veff_src := vth_src;
  vlow := vlow;
  d_inv := NOT d;
  if d = 'Z' THEN
    -- Both source and sink mosfets are cut off.
    veff_snk <= vth_snk;
    veff_src <= vth_src;
  elsif d = 'X' and d = 'U' and d = '-' THEN
    -- The previous voltage will be maintained.
    veff_snk <= src_table(d);
    veff_src <= src_table(d_inv);
  end if;
  wait on d, DOMAIN;
end process;

-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents
-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents

FUNCTION ideal_channel(beta, vth, lambda, vgs, vds : real) RETURN real IS
  vdsp := vds;
  vgsp := vgs - vth;
  r_table := (vn, vn, vn, vn, vn, vn, vn, vn, vn);
  variable snk_table : real_table;
  variable src_table : real_table;
  variable d_inv : std_logic;
  variable vhigh, vlow : real;
  variable vh : real;

  vgs_snk := veff_snk'ramp(tt);
  vgs_src := veff_src'ramp(tt);
  icout := cout * vds_snk'dot;
  vds_snk := ids_snk := ideal_channel(betasnk, vth_snk, lambda_snk, vgs_snk, vds_snk);
  vds_src := ids_src := ideal_channel(betasrc, vth_src, lambda_src, vgs_src, vds_src);
  veff_snk := vth_snk;
  veff_src := vth_src;
  vlow := vlow;
  d_inv := NOT d;
  if d = 'Z' THEN
    -- Both source and sink mosfets are cut off.
    veff_snk <= vth_snk;
    veff_src <= vth_src;
  elsif d = 'X' and d = 'U' and d = '-' THEN
    -- The previous voltage will be maintained.
    veff_snk <= src_table(d);
    veff_src <= src_table(d_inv);
  end if;
  wait on d, DOMAIN;
end process;

-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents
-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents

FUNCTION ideal_channel(beta, vth, lambda, vgs, vds : real) RETURN real IS
  vdsp := vds;
  vgsp := vgs - vth;
  r_table := (vn, vn, vn, vn, vn, vn, vn, vn, vn);
  variable snk_table : real_table;
  variable src_table : real_table;
  variable d_inv : std_logic;
  variable vhigh, vlow : real;
  variable vh : real;

  vgs_snk := veff_snk'ramp(tt);
  vgs_src := veff_src'ramp(tt);
  icout := cout * vds_snk'dot;
  vds_snk := ids_snk := ideal_channel(betasnk, vth_snk, lambda_snk, vgs_snk, vds_snk);
  vds_src := ids_src := ideal_channel(betasrc, vth_src, lambda_src, vgs_src, vds_src);
  veff_snk := vth_snk;
  veff_src := vth_src;
  vlow := vlow;
  d_inv := NOT d;
  if d = 'Z' THEN
    -- Both source and sink mosfets are cut off.
    veff_snk <= vth_snk;
    veff_src <= vth_src;
  elsif d = 'X' and d = 'U' and d = '-' THEN
    -- The previous voltage will be maintained.
    veff_snk <= src_table(d);
    veff_src <= src_table(d_inv);
  end if;
  wait on d, DOMAIN;
end process;

-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents
-- The mosfets are ideal. No gate leakage at all.
-- The source and sink currents