Verilog-AMS Eases Mixed Mode Signal Simulation

Ira Miller
Thierry Cassagnes

Motorola
1300 North Alma School Road, Chandler, Arizona, 85224

ABSTRACT

The ability to design and verify mixed mode (digital, analog, electrical, and non-electrical) systems is key to the development of new products for the ever expanding electromechanical market. Although there are several individual point tools that can address specific phases of the development flow, tools to make the total process more seamless and less disparate should start to appear at the start of this new millennium, fueled by the introduction of Verilog-AMS Hardware Description Languages(HDL) and other analog HDL languages.

The Language Reference Manual (LRM) for Verilog-AMS, developed by and available from the Open Verilog International (OVI) group, forms the basis for the new Verilog-AMS language. The document describes the extensions to the IEEE standard digital simulation language Verilog that enable the description of analog and non-electrical behavior. The document soon to be made available to an IEEE standards organization has been going through the OVI standardization process since about 1995. Verilog-A simulators based on the OVI LRM 1.0 are currently available.

Keywords: Spice, OVI, Verilog, HDL, Behavioral

1 Seamless Tools

Mixed mode simulation includes the simulation and verification of digital and analog functions containing electrical, electromagnetic, thermal, control, hydraulic, and mechanical behavior.

Currently there are several separate simulation and verification tools to automate the product development process. The tools are slowly merging and migrating to a seamless design and verification flow, with capability to move from high level behavioral abstraction, down to silicon implementation, including some optimization capability. Disciplines such as test, applications support, and failure analysis are being added to the flows.

The demand for mixed mode development tools is increasing and is being driven by increasing systems integration requirements, reduced product development time, and shrinking product profit margins requiring optimized designs. An example mixed mode integrated circuit is illustrated in Figure 1.

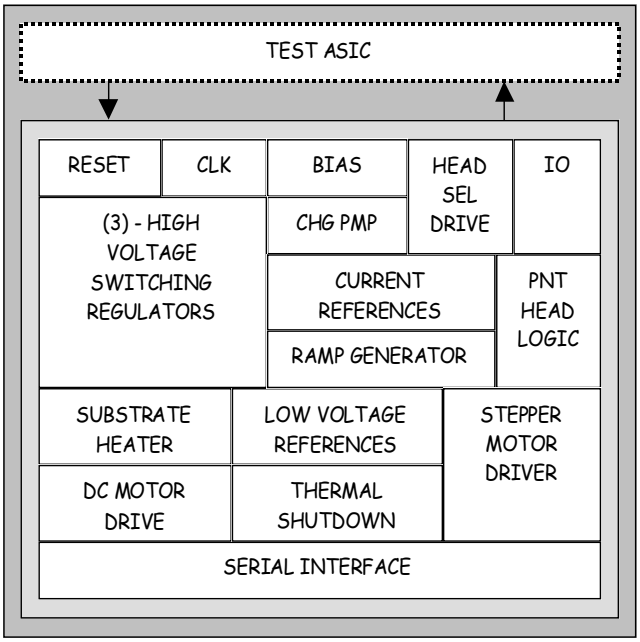

Figure 1. Mixed Signal I.C.

2 Simulators

Simulators for analog and digital signal types consisting of conservative and non-conservative system descriptions coupled with structural, behavioral and continuum modeling techniques are currently simulated at various levels of abstraction, requiring careful analysis and translation between simulators and abstractions.

2.1 Verilog

Verilog Hardware Descriptions Language (HDL) is extensively used in the design and verification of *digital* systems due to ease of use and ability to structurally verify abstraction at the circuit, gate, and register level. Behavioral performance of data flow and timing, as well as algorithm, truth and state table, and equation verification is included in the simulation capability. A rich offering of support tools such as synthesis, delay calculators, extraction, design rule checking, and testability optimization exist. Several texts and training materials are also available. [1]

2.2 SPICE

A Computer Program to Simulate Semiconductor Circuits" Spice2, appeared about 1975.[2] Spice2 which is an evolution of a simulation program called CANCER, determines the quiescent operating point, calculates the time domain behavior and the small-signal frequency-domain response of circuits. It contains built-in models for common passive and active electrical components and defines a circuit on an element by element basis. Early SPICE simulators contain a limited number of fixed linear and nonlinear elements as shown in Table 1 and 2.

Resistor (R)	Capacitor (C)
Inductor (L)	Mutual Inductor (K)
Independent V Sources (V)	Independent I Sources (I)
Linear V-C I Sources (G)	

Table 1 Linear Elements

Nonlinear VC I Sources (N)	Diode (D)
Bipolar Transistors (Q)	Field Effect Transistors (J)
MOS Transistors (M)	

Table 2 Nonlinear Elements

SPICE, in its many released versions, has been the main simulation tool used for analog design and verification for over 25 years. Several commercial products based upon the initial release of Spice have been introduced with enhanced features and capability to address the continually increasing circuit complexities.

Modular functional blocks, created with extensions to the definitions of controlled sources, has provided higher levels of abstraction. The technique known as "macro-modeling", which first appeared in mid 1974, was the basis of the added capability.[3] Although this added capability extended the life of SPICE simulation, circuit complexity has continued to increase and require even higher levels of abstraction.

2.3 Math Calculation Software

Programs such as Mathcad and Mathlab were developed to provide versatile and powerful calculation software packages for applied math in technical fields, providing solvers for vectors, matrices, linear algebra, Laplace, Z-transforms, Fourier, and wavelets.[4] They have been successfully used for higher levels of modeling behavior, but lack concise language for use in environments commonly found in integrated circuit development, and are not easily linked to tools such Spice and Verilog.

3 Open Verilog International (OVI)

OVI was founded in 1990 to promote and further the evolution and usage of the Verilog HDL, which is an IEEE standard language, whose future is guided and managed by OVI and driven by the needs of Verilog HDL users. OVI has broad industry support among systems companies, EDA companies and ASIC vendors. Verilog HDL is the dominant HDL in use today by 35,000 active designers and is used in more than 200 universities worldwide to teach top-down design methods. The number of EDA vendors offering Verilog HDL compliant tools is growing rapidly. OVI encourages EDA vendors to offer tools to support interoperability of HDL models written in any of the popular HDLs.

The Verilog-AMS Technical Subcommittee was created under the auspices of OVI with the charter to develop, update and promote analog and mixed signal extensions to the Verilog (IEEE-1364) language.

The subcommittee's activity has already resulted in the OVI approval of the Verilog-A LRM in June of 1996 and a prerelease of the Verilog-AMS LRM in August of 1998 that now supersedes the Verilog-A LRM.

4 Verilog-A

Verilog-A combines' structural modeling with language based behavioral modeling. Verilog-A is a mixed conservative and non-conservative simulator that applies the conservative laws whenever the signal disciplines are conservative and passes the signals consistently between conservative and non-conservative nodes. It is a language based behavioral code that can be used to directly resemble the intended equations for the system behavior, making debug much easier. Important features of the language include coupled algebraic-integro-differential equations, and rich set of analog operators, if-then and case statements, and implicate and explicit equations.

The format of a Verilog-A module is very similar to the Verilog-D module shown in Figure 2. The "**always**" statement in the Verilog-D module is replaced with a continuous time statement "**analog**" in an analog module.

```
module comp(<signal>);
  <sig_declarations>

  <param_definitions>

  <local_structure>

  always begin
      <behav_definitions>
  end
endmodule;
```

Figure 2 Verilog Module Format

4. 1 Resistor Model

Analog behavioral descriptions are encapsulated within analog statements (blocks) in a module definition. Behavioral descriptions are mathematical mappings which relate the input signals of the module to output signals, in terms of a large-signal or time-domain behavior description. The mappings use the Verilog-A language contribution operator "**<+**" to assign an expression to a signal. The assigned expression can be linear, non-linear, algebraic, and/or differential functions of the input signals. The large-signal behavior descriptions that define the constitutive relationship of the module take the form:

output_signal <+ f(input_signal);

The Verilog-A listing for the resistor symbol in Figure 3 is given in Figure 4. [5]

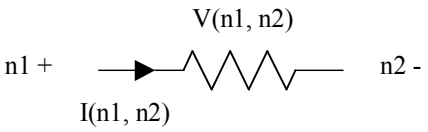

Figure 3 Resistor Model

```
module resistor(ne, n2);
  inout n1, n2;

  electrical n1, n2;

  parameter real R = 1.0;

  analog begin
      I(n1, n2) <+ V(n1, n2)/R;
  end
endmodule;
```

Figure 4 Verilog-A Module Listing

4.2 Mixed Level Description

The listing as shown in Figure 5, "op_amp.va" is a model that defines the behavior of an operational amplifier and contains a test routine to measure the step response settling time. (Test routines are often separate modules and are referred to as "Test Benches").

A circuit file will call the model file and included it as part of the structure defined.

```
`include "std.va
`include "const.va"
// - opamp.va
module opamp(inm, inp, out, gnd);
inout inm, inp, out, gnd;
  electrical inm, inp, out, gnd;
  parameter real gain = 250k;
  parameter real rp = 2.3k;
  parameter real cp = 30p;
  parameter real rin = 2Meg;
  electrical n1, n2;
    analog begin
        I(inp, inm) <+ V(inp, inm)/rin;
        V(n1) <+ laplace_nd(gain*V(inp, inm),
        {1, 0}, {1, 5e-7 });
        I(n1, n2) <+ V(n1, n2)/rp;
        I(n2, gnd) <+ cp*gain*ddt(V(n2, gnd));
        V(out, gnd) <+ V(n2, gnd);
     end
endmodule
// determine the settling time of an opamp
module settling_test(stim, meas);
inout stim, meas;
electrical stim, meas;
parameter real v0 = 0.0;
parameter real v1 = 5.0;
parameter real tstart = 1.0u;
parameter real interval = 10.0u;
real vstim;
real last;
  initial begin
    vstim = v0;
  end
  analog begin
// generate stimulus
  @(timer(tstart)) begin
    vstim = v1;
    last = tstart;
  end
    V(stim) <+ transition(vstim, 0.0, 1.0n, 1.0n);
    $strobe("%g: stimulus = %g", $realtime(), V(stim));
// measure results - opamp is in inverting/unity-gain config.
  @(cross(V(meas) - 1.1*v0, -1.0)) begin
    last = $realtime();
  end
    @(cross(V(meas) - 0.9*v0, +1.0)) begin
    last = $realtime();
  end
// report at end of measurement interval.
    @(timer(interval)) begin
    $strobe("settling time = %g s.", last - tstart);
  end
endmodule
```

Figure 5 Verilog-A Model File

The circuit file for the operational amplifier model "opamp.va" is shown in Figure 6. During simulation of the model a settling time = 2.84641 uSec was reported for the waveform shown in Figure 7.

```
*- OpAmp
 .verilog "opamp.va"
xamp1 inm 0 out 0 opamp
rfb inm out 100k
rload out 0 100k
cload out 0 20p
xtest vin out settling_test v0=500m v1=-500m interval=10u
rin vin inm 100k
.tran 10p 10u 0.0 0.01u
.end
```

Figure 6 Op Amp Circuit File

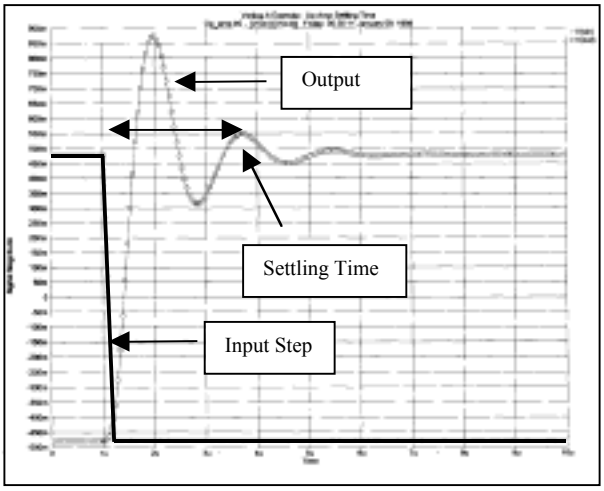

Figure 7 Op Amp Settling Time

5 Verilog-AMS

Verilog-AMS (Figure 8) benefits users by allowing them to describe and simulate analog and mixed signal designs using a top-level design methodology as well as the traditional bottom up approach.

OVI's Verilog-AMS standard supports analog and mixed signal designs at three levels: transistor/gate, transistor/gate-rtl/behavioral, and mixed transistor/gate-rtl/behavioral circuit levels. Moreover, Verilog-AMS provides powerful structural and behavioral modeling capabilities for systems in which the effects of, and interactions among, different disciplines like electrical, mechanical and thermal are important.[1] Verilog-AMS utilizes nodes, branches, and ports as terminology for these descriptions. The solutions of analog behaviors that obey the laws of conservation fall within the generalized form of Kirchhoff's Potential and Flow Laws (KPL and KFL). Both of these are defined in terms of the quantities (e.g., voltage and current) associated with the analog behaviors. Verilog-AMS HDL can also be used to describe discrete (digital) systems (per IEEE 1364-1995 Verilog HDL) and mixed-signal systems using both discrete and continuous descriptions as defined in the LRM.

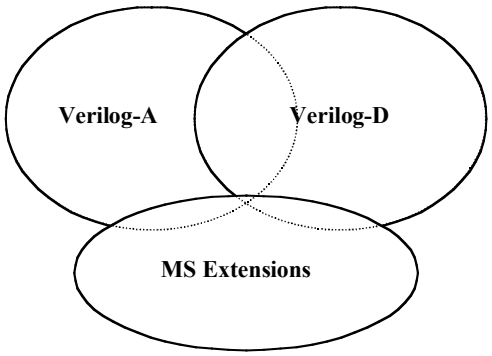

Figure 8 Verilog-AMS

The mixed-signal language extends the features of the digital modeling language (IEEE 1364-1995 Verilog HDL) to provide a single unified language with both analog and digital semantics with backward compatibility.

Signals of analog and digital types can be declared in the same module. Initial, always, and analog procedural blocks can also appear in the same module. Both analog and digital signal values can be accessed (read operations) from any context (analog or digital) in the same module. Digital signal values can be set (write operations) from any context outside of an analog procedural block, and analog potentials and flows can only receive contributions (write operations) from inside an analog procedural block. The semantics of the initial and always blocks remain the same as in IEEE 1364-1995 Verilog HDL. The discipline declaration is extended to digital signals and a new construct and connect statement is added to facilitate auto-insertion of user-defined connection modules between the analog and digital domains. When hierarchical connections are of mixed type (i.e., analog signal connected to digital port or digital signal connected to analog port), user-defined connection modules are automatically inserted to perform signal value conversion. [6]

REFERENCES

[1] OVI WEB site. http://www.ovi.com

[2] SPICE2: "A Computer Program To Simulate Semiconductor Circuits", Memorandum No. ERL-M520, Electronics Research Laboratory, College of Engineering, Berkeley, May 1975.

[3] J.Alvin Connelly and Pyung Choi, "Macromodeling with Spice," Prentice Hall, 1992.

[4] Mathcad 8 User's Guide, MathSoft, Inc, Cambridge Massachusetts 02142.

[5] Dan FitzPatrick and Ira Miller, "Analog Behavioral Modeling with the Verilog-A Language," Kluwer Academic Publishers, 1998.

[6] OVI Language Reference Manual, Version 1.9