The era of “Internet everywhere” is creating a spectrum of applications targeted toward low-power and mixed-signal design, in segments ranging from health care to automotive to communications. Meanwhile, design challenges such as intellectual-property selection and integration as well as SOC- and system-level verification are spawning a whole new class of problems for EDA tools.

Mixed-signal design engineers face increasing difficulties in design and verification of complex mixed-signal SOCs. In a survey of mixed-signal design engineers during the 2011 Mixed-Signal Tech on Tour, a worldwide series presented by Cadence Design Systems Inc, the 561 respondents identified mixed-signal verification as a top customer challenge.

The performance of Spice simulation is prominent in the difficulties being reported (Figure 1). Analog Spice and Fast-Spice simulators are orders of magnitude slower than digital simulators and are slower still when compared with emulators and hardware accelerators. A June 2011 Design Automation Conference panel discussed the need for analog design and verification to become more like digital—that is, to become more structured and more top-down (Reference 1). Verification planning tools are required, and debug methodologies such as ABV (assertion-based verification), MDV (metric-driven verification), and UVM (universal verification methodology)-like self-checking test benches must be created for analog/mixed-signal.
To tackle simulation-throughput issues, designers are turning to behavioral-modeling techniques, which can increase simulation speed. Such techniques include event-driven simulation based on Verilog-A, Verilog-AMS, and RNM (real-number modeling).

Analog behavioral models are typically written in Verilog-AMS, Verilog-A, VHDL-AMS, or SystemVerilog.

Verilog-A is a pure-analog subset of Verilog-AMS and is mainly used for detailed analog models for performance verification. The language is quite simple, but it is challenging to write a good behavioral model with Verilog-A that provides significant performance gains while retaining the right level of accuracy. The advantage of Verilog-A is the ability to use models in pure-analog simulations as well as in the mixed-signal environment. The models are too low-level, however, to enable efficient SOC-level verification of mixed-signal designs.

The RNM technique models electrical signals by representing them as real values. Provided that the modules are at a sufficiently high level of abstraction, the interfaces can be described by passing real numbers between blocks to represent the voltage, or current, signal being transferred. This is a powerful way to simulate complex systems rapidly. Traditionally, iterating to a solution involving feedback would require an analog solver (see sidebar, “Analog versus digital solvers”).

RNM is available in the Verilog-AMS, SystemVerilog, and VHDL-AMS languages. A commonly used RNM approach is the wreal data type in Verilog-AMS. RNM uses a discrete event solver—without an analog solver—and can be used to simulate mixed-signal systems at incredible speeds. It is primarily limited to modeling at a high enough level of abstraction that bidirectional analog interactions between blocks are not significant. In other words, typical RNM defines blocks in terms of input/output transfer characteristics, with no strong direct feedback present among the blocks. Logic can be modeled naturally in these languages, so RNM is also a good choice for systems with only a small amount of analog content.

**Top-down or bottom-up**

Designers use two principal methodologies based on the creation of behavioral models for mixed-signal design. In a top-down methodology, models are developed before the circuits are designed.
The behavioral models can be simpler ones that are sufficient for functional verification at the system level. In a bottom-up methodology, the models are written to match an already implemented block for performance verification and usually result in a more accurate but slower-running model.

The creation of analog behavioral models can be challenging. Analog designers are in the best position to create such models because they are familiar with their own circuits. Many analog designers lack the programming skills and knowledge required to construct behavioral models, however, and few are familiar with Verilog or VHDL. Digital designers, conversely, have expertise with behavioral models but know less about analog circuits.

That dichotomy creates an opportunity for tool vendors to offer an automated or semi-automated solution for generating analog behavioral models. For example, automated, netlist-driven model-generation technologies can create a fairly accurate parametric behavioral model that considers PVT (process, voltage, and temperature) and loading variations for functional verification. Such an approach has shown some limited success on a subset of analog-circuit architectures under specific contexts, but there is still much work to be done to develop a general model-generation methodology with high accuracy that can be applied to any analog or mixed-signal design.

Creating behavioral models is only one part of the process of using those models in a mixed-signal verification flow. If the model and implementation do not match, the effort is worthless; worse, it can damage the entire design process. As a result, there is a need for a methodology to validate the accuracy of a behavioral model automatically against the corresponding design. The model also must be updated to keep it in sync with any changes made in the implementation.

**Assertions’ role**

The model-validation flow shown in Figure 2 simulates the implementation and behavioral models using the same test bench, with the relevant tools and flows creating the required measured-results behavior and waveform for each model. The designer can use the flow to validate that:

- The implementation- and behavioral-model measured results are within user-provided tolerances;
- The implementation- and behavioral-model waveforms are within user-provided tolerances; and
- Selected elements of the interfaces and pins of the behavioral and implementation models match.
This approach provides a self-contained, automated analog-behavioral-model verification and debugging environment. The method lets users provide verification requirements through interactive configuration, and it uses the existing mixed-signal simulation setup to validate both the waveform signals and the measured results. Interactive verification-failure debug lets designers quickly identify problems in a model and rerun validation after fixing them.

That is just the beginning of the advantages that can be brought into the analog-verification world. The digital domain has many other tools at its disposal—such as verification planning, coverage, and assertions—that can apply to analog verification given the right language extensions.

**Assertions’ role**

Assertions capture aspects of a specification and design intent in an executable form. They act as monitors during simulation, detecting errors close to their source and reporting both errors and coverage information. Through the use of assertions, verification can start earlier; design and verification teams can detect and remove bugs faster; and designers can incorporate their intent into the design code, thereby minimizing integration issues later on. The mixed-signal design and verification communities embracing ABV methodologies can reap the benefits of assertions and tools in the following ways:

- Assertions capture design intent and can be incorporated with the design;
- Assertions detect errors closer to their source, speeding removal; and
- Assertions provide control-oriented functional coverage information.

Assertions can also capture properties about interfaces, such as the assumptions made in the design, the interactions that are expected to occur (or not occur), and errors related to behaviors such as nominal functional boundary conditions or startup. A metric-driven verification environment can use assertions as coverage points, which ensure that the environment has verified the design in every valid configuration, that it has verified all possible communications-protocol variations, that valid
combinations of inputs have been applied, and that valid combinations of outputs have been observed.

In this still-emerging field, some HDLs and HVLs include support for real-numbered functional-coverage objects; the ability to generate real-valued, constrained random numbers; and the ability to create assertions on electrical quantities, such as voltages or currents.

Two standards groups are actively working to standardize analog/mixed-signal assertions. The ASVA (Analog System Verilog Assertions) committee is targeting analog and mixed-signal extensions to the SVA subset of SystemVerilog (Reference 2); the SV-AMS (SystemVerilog-AMS) committee is defining analog/mixed-signal extensions to SystemVerilog in work that parallels the successful transformation of Verilog into Verilog-AMS (Reference 3). Participants expect the results of the ASVA committee’s work to feed into the longer-term SV-AMS effort.

Behavioral modeling continues to make inroads into mixed-signal simulation and verification as a result of behavioral-language standardization, skills development, and automation improvements. As design teams gain a better understanding of the advantages that behavioral modeling offers, more are making the initial investment necessary for the methodology shift, confident they will reap a return in the form of productivity improvements.

Behavioral modeling can be a key component of any mixed-signal verification methodology. Bringing analog and mixed-signal blocks to a higher level of abstraction enables the creation of more effective mixed-signal simulation and more complete verification environments.

References

2. Verilog Analog/Mixed-Signal Technical Subcommittee
3. SystemVerilog-AMS

Author’s biography

Qi Wang is group director for solutions marketing at Cadence Design Systems, with a focus on low-power and mixed-signal solutions. In this role, he works with product-marketing, research and development, and technical-support teams from various product groups to set the solution-wide strategy and to promote Cadence solutions for advanced low-power and mixed-signal designs.

Analog versus digital solvers

Digital simulators execute much faster than their analog counterparts. Digital simulators calculate values for outputs whenever inputs change. There is a time delay associated with those value changes. Each change is independent of other changes happening in the system.

Analog simulators are tasked with solving a set of simultaneous (nonlinear, ordinary differential) equations. During transient analysis, matrix-based numerical-analysis techniques solve the complete set of voltage and current equations at each analog time step. The process is iterative because it involves feedback; performance depends in part on how quickly the simulator can converge on a
solution.

A direct-solution method could immediately compute the voltages and currents at each time point without requiring an iterative process if the method had to define only linear systems, but real systems are never completely linear. Everything has bounds of operation, often induced by power-supply or performance limits that result from the transistors used to implement the characteristics.

The typical analog simulator breaks the time axis into sufficiently small time steps to allow it to approximate linearity over each of the steps (Figure A). It then can calculate the simultaneous-equation solution that describes what should happen over each small time step. A significant amount of theory is associated with doing this calculation accurately, predicated by the simulator’s ability to estimate how much error may occur at each step. The simulator can then decide how big a time step it can safely take, and it must iterate and converge toward a solution that sufficiently solves the Kirchhoff’s voltage and current laws at the new time point.

![Figure A](image-url) Various sampling methodologies are compared. Analog simulators break the time axis into small time steps.

With a nonlinear system, this iteration procedure is not guaranteed. Each iteration estimate of the new solution requires reevaluation of the nonlinear relationships in the system. If the solution does not converge to a sufficiently accurate value in a certain number of iterations, the simulator will typically assume that the step size was too large for the nonlinearities and give up computing that point. It will then try a smaller time step from the previous time point.

If you liked this feature, and would like to see a weekly or bi-weekly collection of related features delivered directly to your inbox, sign up for the IC Design newsletter here.