Real Number Modeling

How to verify mixed-signal behavior using event-based simulation

Tom Cole
Verification Manager
IBM

Wes Queen
Verification Manager
IBM

Mark Kautzman
Verification Engineer
IBM

Dan Romaine
Solutions Engineer
Cadence Design Systems

Abstract

This paper focuses on how IBM created and deployed real number models to verify a recent mixed-signal project. The intent is to show how switching from SPICE-level and conservative electrical model co-simulation, to using real number models in Verilog-AMS, and running both analog and digital portions of the design in an event-based simulation, was used effectively to verify the functionality of a mixed-signal IC. In this paper, we will also review the motivation behind Real Number Modeling (RNM) and show a brief comparison to other modeling and simulation approaches—also taking some time to discuss the language choices available for this approach.

This paper also explores the engineer ramp-up process, and the resource allocation used to effectively collaborate across the analog and digital engineering disciplines. The design itself will not be discussed in detail. Instead, we will focus on the model creation process, the languages and approaches taken, the productivity and quality impact, and the engineering disciplines deployed as we proceeded through the project.

Introduction

Most system-level verification of analog, digital, and mixed-signal designs is accomplished through simulation. To meet the verification goals, simulation data and data accuracy are required, which means that a detailed analysis of a component such as an RF low noise amplifier, requires very high simulation accuracy, but a single RF sinusoid period might be sufficient to achieve the goals. Conversely, a pin connectivity check for a large digital block has an extremely low sensitivity towards accuracy, but may require a long transient simulation time to cover an array of events and states. As a result, a full-chip simulation using high simulation accuracy would be desirable. The limiting factor in this context is mixed-signal simulation performance. A practical way around this problem is a hierarchical verification approach that uses different levels of design abstractions to meet different verification goals.

Real Number Modeling is an innovative addition to classical mixed-signal verification approaches. It uses a hybrid model, with sufficient analog content to apply digital simulation techniques (and performance levels) while still maintaining a level of accuracy suitable for true mixed-signal verification.

Running at Digital Speeds with Event-Based Simulation

Traditional mixed-signal simulation, sometimes referred to as analog mixed-signal simulation (AMS), involves executing SPICE/analog simulation engines in tandem with digital, event-based simulation engines. When simulating using SPICE or transistor-level schematics, the analog engine used to solve the design equations produces extremely accurate results, but requires orders of magnitude more simulation time than a digital event-driven simulation would require.

Digital-centric mixed-signal verification (DMS) refers to—but is not limited to—mixed-signal verification using only event-based simulators. By executing only on a digital simulator, the DMS approach provides a reasonable trade-off between accuracy and performance. Additionally, this approach enables digital-

centric verification methodologies (coverage tracking, assertions, and so on.) to be applied to analog portions of a mixed-signal simulation.

What is Real Number Modeling?

The simulation approaches in analog and digital designs are fundamentally different due to the structure of the underlying equation systems being solved. While the digital solver is solving logical expressions in a sequential manner based on triggering events, the analog simulator must solve the entire analog system matrix at every simulation step. Each element in the analog design can have an instantaneous influence on any other element in the matrix and vice versa. Thus, there is not an obvious signal flow in one direction or the other——time and values are continuous. In digital simulation, time and values are discrete. The solver can apply a well-defined scheme of signal flow and events to solve the system.

Real Number Modeling (RNM) is a mixed approach, borrowing concepts from both the analog and digital domains. In RNM, the values are continuous—floating-point (real) numbers—as they are in the analog world. However, time is discrete, meaning the real signals change values based on discrete events. In this approach, we apply the signal flow concept, so that the digital engine is able to solve the RNM system without support of the analog solver. This delivers high simulation performance, in the range of a normal digital simulation.

There are three different hardware description language (HDL) standards that support RNM. They are:

- Verilog-AMS
- VHDL
- SystemVerilog

Traditional SPICE + RTL simulation at the full-chip level is extremely costly (both in terms of time and license cost) because a significant amount of the design is simulated inside the analog engine. Real number modeling is one way to reduce the time and expense required to verify an SoC, while trading off some accuracy that is not needed at this high level of integration. By replacing the analog portions of the SoC with functionally equivalent digital models, which do not require the analog engine, we achieve a very attractive speed-up in simulation performance and a reduction in the license cost. Meanwhile, typical analog simulation problems such as convergence issues are totally eliminated.

Hardware Description Languages Used for Modeling

In order for you to better understand the approach we took in using RNM, we will review the background of related modeling and hardware description languages.

The languages most widely used to model analog behavior are:

- Verilog
- Verilog-A
- Verilog-AMS
- VHDL
- SystemVerilog

The primary languages for creating behavioral models of analog circuits using real number representation have been Verilog, VHDL, and Verilog-AMS. SystemVerilog is a newer option, as real number capability is being added to the language in the upcoming language reference manual (LRM). In the following sections we provide an overview of each language and describe some of the features that apply to analog behavioral modeling.

Verilog (IEEE 1364)

Verilog is a hardware description language that was originally standardized in 1995. The real number capabilities of Verilog are limited to variables inside a module, which restricts model connectivity (no real ports). Because of this limitation, some simulators provide conversion utilities to turn real numbers into 64-bit wide buses. While this workaround is helpful, it is quite restrictive and produces models that are not pin-for-pin compatible with the analog circuit.

Verilog-A

Verilog-A is an analog-only modeling language that was designed to represent Spice-level functionality in a behavioral modeling language. The Open Verilog International body (now called Accellera) agreed to support the language standard so long as it was part of a mixed-signal language effort (Verilog-AMS). Today, the analog portion of the Verilog-AMS standard comes from the original Verilog-A language specification. Verilog-A is an analog-only representation of circuits and is not used to create real number models.

Verilog-AMS

Verilog-AMS consists of the complete Verilog specification (IEEE std 1364-2005), an analog modeling language (Verilog-A), and extensions to both for specifying mixed-signal descriptions. Verilog-AMS includes *wreal* net types, useful for structurally connecting design elements together using real numbers.

Verilog-AMS includes a unique mixed signal feature for specifying net and register behavior called disciplines. A discipline defines the domain (analog or digital) and the nature (voltage and/or current) of a net and includes additional attributes for continuous nets used in analog modeling.

Wreal nets

Verilog-AMS includes real number nets called *wreals* in addition to the Verilog net types (*wire*, *wor*, *wand*, etc). *Wreal* nets behave like wires and have a real value type.

The language standard does not provide any further details on the application of disciplines and the ability to support multiple drivers and signal resolution for *wreal* nets. Some simulators have additional capabilities for *wreals* which allow X and Z states on the nets, multiple outputs to be combined structurally using wires resolution, as well as *wreal* connectivity to logic and electrical nets.

VHDL (IEEE 1076)

VHDL is a strongly typed hardware description language which includes an AMS subset (VHDL-AMS 1076.1). The real number modeling capabilities of VHDL are robust, including real number variables and nets, compound nets (multiple values), custom resolution functions, and arrays. While these features are strong within the VHDL language, the challenging part of real number modeling in VHDL revolves around inter-language connectivity.

SystemVerilog (IEEE P1800)

SystemVerilog is a unified hardware design, specification, and verification language based on extensions to Verilog (1364-2005). The language is composed of digital design constructs, object-oriented verification constructs, and assertions. The real number modeling capability consists of real number variables inside modules and real number *input* and *output* ports. Because real numbers are variables, not signals, their ability to model analog connectivity is limited. Real value variables cannot have multiple contributors (drivers) and cannot be used for bi-directional port connections.

The IEEE P1800 committee has developed a new version of the LRM, which includes additional modeling capability. This capability will remove the current restrictions for real number ports by allowing user-defined types (UDTs) and user-defined resolution functions (UDRs). For more information, please refer to the IEEE P1800 committee.

Language Comparison

The following figure shows a table of these various modeling languages and their related features for analog modeling and connectivity.

		Feature	Verilog	VHDL	Verilog-AMS (wreal)	SV 2009	SV 2013
Modeling	1	Real value variables	✓	✓	✓	✓	✓
	2	Real valued Signals/nets declarations	✗	✓	✓	✗	†
	3	User Defined Types with reals (multi-field records)	✗	✓	✗	✗	†
	4	Resolution functions for multiple real signal drivers	✗	User-Defined	6 Built-in functions	✗	†
Interaction	5	Ability to convert an interconnect to real/logic based on the hierarchical connectivity	✗	✓ Not as flexible as Wreals	Wreal coercion (incl. electrical)	C	†
	6	Ability to connect Real-to-Logic and automatic conversion of values	✗	✗	C	C	C
	7	Interact with electrical signals (Automatic Connect Module)	✗	✗	C	C	C
	8	Testbenches with mixed signal content	✗	✗	✗	✗	†

✓ : Standardized & Implemented
C : Vendor-specific
† : On the standardization roadmap; partial implementation
✗ : Not in the standards/roadmap; not implemented

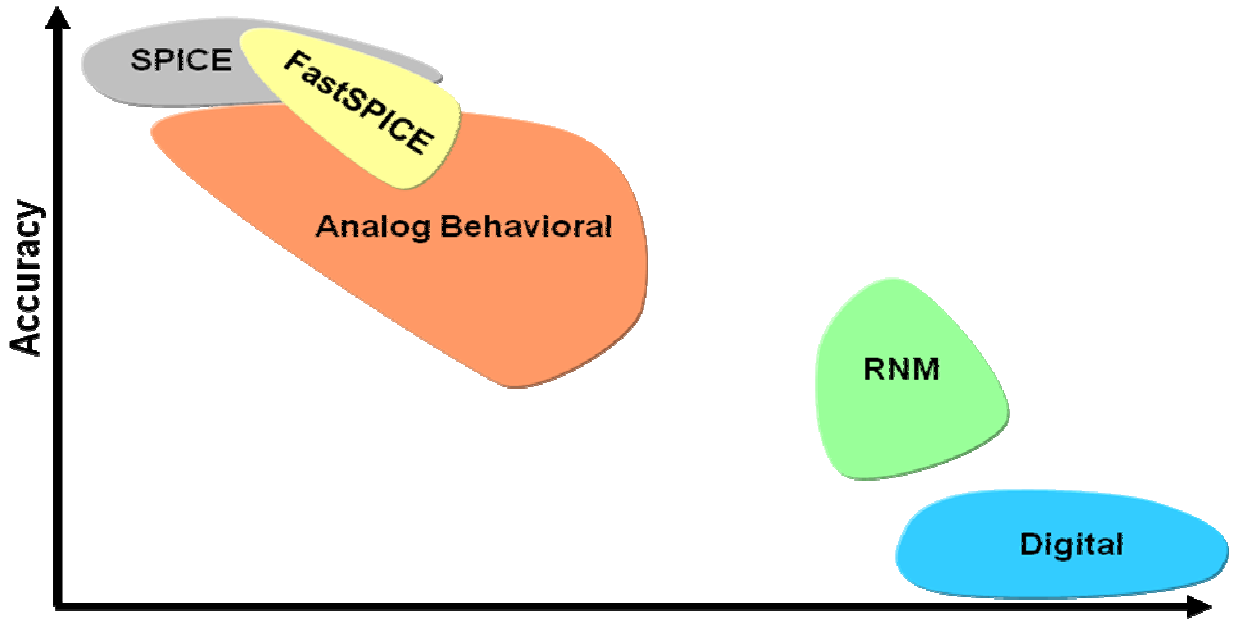
Simulation Performance

For analog sub-systems in a mixed-signal SoC, common levels of abstraction include:

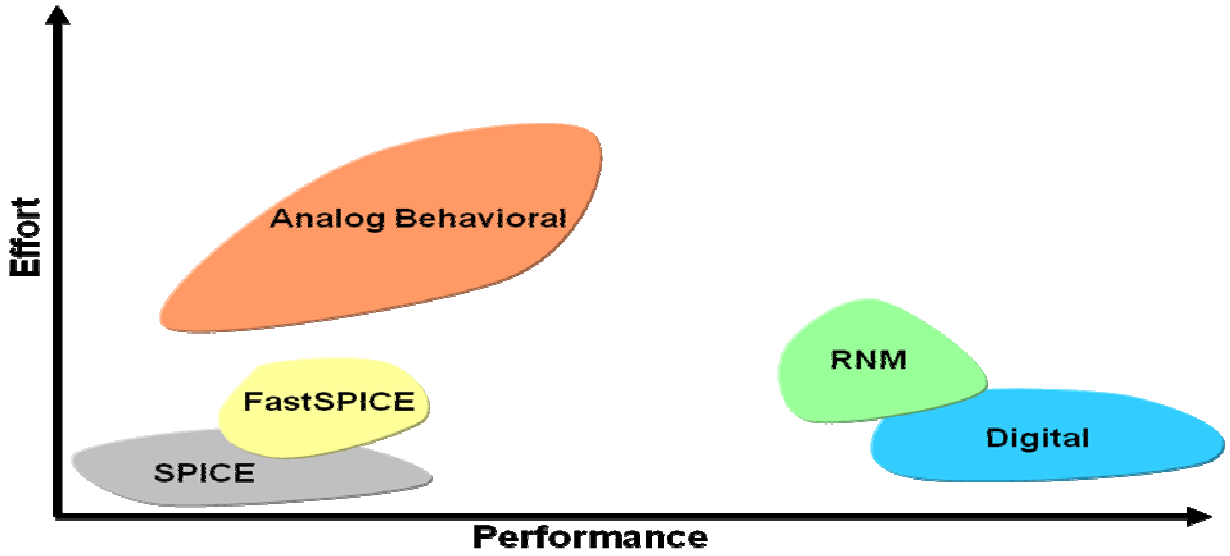
- Transistor-level
- Analog behavioral (Verilog-A)
- Real number representation
- Digital representation

There is a tradeoff between simulation performance and analog accuracy for each level of abstraction. Depending on your simulation goals, each level of abstraction is useful for a specific kind of simulation. There is no general recommendation on what level of abstraction might be useful or not. All have particular advantages and disadvantages, and thus a useful application area.

The following picture shows a general trend in the accuracy/performance tradeoff. The numbers are generic and can vary significantly for different applications.



Additionally, each model of an analog circuit requires a different amount of creation effort and simulation setup. Of course, there can be variations within a given language, but the following picture shows a relative scale for the amount of man hours needed to create specific analog abstractions.



Verification at the Digital/Analog Boundary

The design quality at the intersection of a core's digital and analog functionality has proven to be critical to the success of the core, yet extremely challenging to verify. Finding the right level of AMS modeling detail, which is enough detail to assure quality simulation and accurate schematic behavioral modeling, yet not so much detail as to drive long simulation times, proved elusive, with multi-week long analog mixed-signal model simulations limiting the scope of achievable verification.

As might be expected, week-long simulation times prove very problematic in terms of debug and turnaround times, compute farm resource limitations (availability and mean-time-to-failure), and severe limits to simulation throughput. These challenges drove our verification work to an over dependence on high-level abstracted, performance driven, analog behavioral models in order to gain visibility into the many *corner cases* we wished to observe before core release, which impacted core quality.

The development of AMS[1] models is a non-trivial task, and the time it takes to develop and verify the models is proportional to the complexity (as measured by the number of schematic blocks in a design that need to be modeled) and the available resources, both computationally and professionally. Usually, the task of developing AMS models proceeds as follows:

1. Documentation of the intended schematic design, to the level that it can be modeled.

2. First pass development of analog behavioral models, to the degree they are understood from the documentation.

3. Development of the simulation environment based on a first pass simulation of the AMS models. Usually the full verification suite is eventually run against these AMS models. This phase requires an all-hands-on-deck approach to flush the models and work through any tool issues that may arise. Performance of the models is evaluated during this phase and fed back to the AMS developers.

4. When schematics are complete, or close to complete, it is imperative that the schematics and AMS models be correlated. Otherwise, you may end up with really nice AMS models that do not match the schematic behavior, which, in turn, will give a false sense of quality.

5. Rerun of the verification test suite against the updated AMS models. These AMS models become the primary full core verification vehicle throughout the remainder of the project. AMS models should be updated in the wake of issues, but in parallel, when changes are made to the models. Then, those changes need to be correlated back to the schematic to ensure that the behavior is still the same, or to see if a real schematic issue has presented itself.

There are two critical areas necessary to the success of utilizing AMS in a project flow: high-quality design stimulus and correlation of the model to the schematic.

The verification of a given design is only as good as the stimulus provided for it.

1. It could be that the AMS component is fully modeled and matches the schematic, but the test suite is not stimulating the core correctly. In this case, you need to close the gap using functional and code coverage

2. The regression test suite for pure digital RTL/VHDL analog behavioral verification needs to be regularly regressed against the RTL/AMS model to insure that the goals of the verification plan are being met.

Correlation of the AMS model to the schematic is imperative.

1. If this is not done throughout the life of the project, when schematics are updated or AMS models are updated, you run the risk of having very nice AMS models, in which the AMS models give you the impression that the schematics are correct, but, in fact, do not match behaviorally—and you may have a bug in the schematic.

2. There are two methods in use today to correlate a model and its corresponding schematic:

    a. Create a test bench to be run against both the schematic and the AMS

[1] For all intensive purposes AMS will refer to both AMS and DMS models as the purpose of the development and simulation are one in the same

model. This is a manual effort and is left up to the AMS designer and schematic owner to sign off on the behavior.

b. Cadence provides a tool (amsDMV) which can be used to automate the process. Essentially the test bench mentioned above is used to stimulate both an AMS model and the corresponding schematic representation. The tool provides tolerances on the I/O that will give a *pass*/*fail* result depending on whether or not the tolerances are missed, met, or exceeded.

Historically, the AMS behavioral models created by IBM were implemented using Verilog-AMS electrical constructs to model currents and voltages, and capacitance and slews, for all major circuit blocks. Over the last few generations of designs, our AMS models have shifted from a pure electrical style (with differential equations) to a combination of electrical for major circuits, and digital logic for latches and combinatorial logic at the leaf cells.

Moving to real number models enables the models to execute primarily on the digital simulator, with very little, if any of the modeling and matrix math calculations on the analog simulator. The major trade-off between accuracy and performance was to choose to model either voltage or current in between analog blocks as a *wreal*—ignoring capacitance, slews and differential current, or voltage equations, which greatly improved performance (~40x). Nominal resistances were assumed in each of the models so a corresponding voltage or current could be calculated. The accuracy tradeoffs in these cases are not as detrimental as one may anticipate as the pure analog circuit simulations should cover this accuracy tradeoff. Also, as discussed earlier, the behavior of the model is the critical factor, which should be correlated to the schematic. The following list shows examples of some of the trade-offs that were made:

- An RLC front-end network was modeled for both AC, DC, and test modes. Alternate sets of equations were needed to model voltages for different cases of interest. Capacitance and inductance was not modeled.
- Peaking amplifiers were modeled with a time domain sampled filter, which would alter the frequency response of the incoming signal. Care was taken to have only two models like this in the data path since they are mathematically expensive.
- Current sources output a current that the terminating model would convert to a voltage.
- Terminating resistors of the current sources had to be altered in a way that was different than in the actual circuit in order to produce the correct voltage output per each digital controlled step.

Real Number Modeling Deployment Challenges

For this project, at the beginning of the last verification cycle, an experiment was done with real number models using Verilog-AMS wired-real models (*wreal* models). This was done to compromise between the detailed electrical models and high-level accelerated-performance models.

The expectation was to achieve a 10x test case performance improvement over the previous detailed models, and reduce our test case turn-around-times to a few days while still maintaining a detailed level of results. There were three distinct challenges before the experiment could move into production:

1. Design resources
2. Education
3. Verification tracking

We will discuss each of these below.

Design Resources

Finding analog design resources to add yet another model, which was only a deliverable to the verification team was a challenge. The analog design team is always on a very tight schedule with constrained resources due to challenging schedules and the motto is always: *"do you want schematics or AMS models?"*.

The request to the analog team was viewed from that perspective and it was quickly realized that some persuasion was needed. Some of the reasons that make this persuasion necessary are:

- Who develops the *wreal* models?
  The analog team.
- Who has to learn a new modeling language that is not part of their normal skill set?
  The analog designers.
- Who maintains and adjusts those models as they proceed through schematic design?
  The analog team.
- Who has to field questions and fix problems discovered by the verification team in the models while they are in the middle of designing complex analog blocks?
  The analog team.
- Who is most concerned about the accuracy of higher order models not corresponding to actual silicon behavior, and who gets blamed when it doesn't correlate?
  The analog team.
- Whose already tight schedule gets hammered by model development?
  The analog team.
- Who asks (justifiably) *What is in this for me*, when presented with the magic of the new models?
  The analog team.

Education

Educating the analog team in creating effective and efficient wreal modeling was a challenge. For this, we relied heavily on the Cadence Application Engineering teams to provide multiple education sessions, one-on-one guidance, and also hands-on generation of several models as template/examples.

Verification Tracking

The measures of verification and ultimate product quality are key to knowing that a methodology and modeling change has achieved its objective. In our case those same measures were also key to justifying the investment in *wreal* model development. One challenge we faced in this recent project was tracking verification progress when running mixed-signal simulations.

There are a few ways to track metrics. One is to run with the high-level abstracted analog model to know that you have fully achieved coverage (functional and code) targets after *X* number of simulations. With this number, you can then, in turn, run the same number of simulations with the AMS model to achieve a very high level of confidence that all scenarios have been hit. Looking forward, with the adoption of a full DMS methodology, we will be able to take advantage of utilizing code coverage across the DMS models.

Effecting Change

In was understood that a broad management-dictated approach to this project would not be successful. In order to prove value, it was necessary for us to find a way to show that the model development was feasible, and dramatically beneficial for overall core design and development objectives. At the same time, it was also realized that it was necessary to clearly demonstrate those benefits before engaging the broader community of analog designers.

It was also necessary characterize the actual learning process that was required for an analog designer to develop and maintain a high quality *wreal* model of the design.

Lastly, there was a need to find areas where the time spent developing the model contributed to a more efficient analog design and verification process.

We decided to carefully select a pilot model development project for a few key models, and then engage with a set of actions to overcome the natural barriers to adoption. Our approach was as follows:

1. The first action, as a shared goal of the analog design and digital verification teams, was to prove the value of *wreal*

models in terms of simulation performance and design quality.

2. Analog designers were selected who were both open to new learning experiences and respected in the analog community.

3. Cadence was relied on heavily to provide education sessions, personal guidance, and templates and examples. This proved essential to getting the experiment kicked-off and pushed forward during the early phases when the risks of abandonment were highest.

4. A process was set up for rapidly incorporating the models into the AMS verification environment and providing timely feedback to the analog team, thus showing the downstream benefits. This rapid feedback was essential in identifying corrections to the model process. But more importantly, it proved to be a real winner in demonstrating, what for us, were game changing performance jumps—much greater than the 10x improvement over the detailed models that we had set as an initial expectation.

The impact of this approach was quick, definitive feedback to the entire team, detailing the actual design resource expenditures and time to design closure.

Debug times were being reduced from weeks, to days and hours! Even the analog team members were seeing direct benefits. With this small success in hand, upper management and outside technical teams became interested in moving to larger and more complex models. At this point, the analog teams had bought in to the process and decided to take the leap.

An expanded list of models was proposed by the verification team and accepted by the design teams. The *experiment* had just moved closer to becoming the *plan*. As each wreal model became available, and the results reviewed, it became clear that *wreal* models could become a game changer for the AMS verification methodology.

The performance increase was proving to be such that DMS test cases could now be simulated in hours, making DMS simulation a viable candidate for overnight regressions alongside standard all digital simulations.

No longer would AMS need to be relegated to a small sampling of dedicated week long test cases or lower precision accelerated performance models developed by the digital team as poor surrogates until the analog models were extracted from the schematics. AMS could now be incorporated into the constrained random metric driven verification environment allowing extensive probing of corner cases before, rather than after, the official core release.

Managerial Perspective

One of the keys to success was that these achievements were shared significantly and directly with the analog team. The models had become the key to higher quality cores at a earlier time in the schedule. The first barrier had been crossed from both analog and verification, and some ancillary benefits were found that applied more directly to the analog design productivity.

For one thing, the success of the DMS model allowed removal of one of the high-level abstracted analog models from the analog requirements, so it became a replacement and not an addition. There was also added value in getting early feedback to the analog designer based on the model, well before the availability of the completed analog schematic design. Some observed values to the analog team included:

1. **Time Savings**—Developing a model consumes time, but it serves as a valuable tool for the analog designer to use to validate the design intent of the architect and the specification. Being able to then have that model run in the AMS verification environment provides the analog designer with further feedback on the critical interface logic being co-developed by the digital team. Even the development of a

testbench to verify the model creates a reusable base for the schematic-level verification.

2. **Earlier Bug Detection**—No one is happy to have a bug discovered in a design at any time. But having one found very near the scheduled release date is everyone's second worst nightmare. Yet, the previous AMS verification process almost guaranteed that pre-release bugs found in the analog design would be found very late in the development cycle. This is when the time-line to release is shortest, pressure to correct bugs the greatest, and the time to validate them is measured in weeks. The *wreal* model approach directly helps the analog design team by finding problems earlier in the process and by dramatically reducing the verification time for a fix, whenever the bug is discovered.

3. **Better Post-Release Quality**—Obviously the worst nightmare for the analog designer is to have a bug found post-release, in system testing, or by the end customer. The far more extensive verification that *wreal* enables does not eliminate this nightmare, but it goes a long way to making it a rare occurrence.

4. **Improved Model Accuracy**—Improved methods to insure the accuracy of the *wreal* models are underway. Cadence has automation tools which help us easily maintain a correlation between the schematic analog design and the *wreal* model. This makes it possible to keep the *wreal* model behavior in step with the analog development and detect discrepancies between the model and the SPICE results.

Conclusion

Looking back on the progress to date, the approach of piloting the project, marshalling a lot of support to cross the most difficult transition points, and positioning management involvement, it is clear this process helped proliferate best engineering successes, rather than dictating what they should be. In addition, establishing a focus on continuous engineering process optimization through tools and education has worked well.

A culture of productivity recognizes that the big leaps forward often require a view of the larger scopes of influence. In the case of *wreal* modeling, doors have been opened to include DMS in our broad push forward into metric-driven verification, which is helping move DMS and digital simulation down a much more common path than we previously thought was possible.

There is no question that the *wreal* model development will have a significant and continuing improvement on overall development productivity. Whether that new model development has positively or negatively impacted the analog team's productivity remains a question to be answered after broader adoption of our process.

The experience from previous transitions on the digital side, which was to SystemVerilog for testbenches, suggests that model development in a new language is an acquired skill, improving in both time and accuracy with practice and lessons learned. Also, model reuse from project-to-project can shorten development over time, and familiarity with the tools for development and verification can dramatically affect productivity.