Universal Verification Methodology (UVM)-based Random Verification through VCS and CustomSim in Analog Mixed-signal Designs for Faster Coverage Closure

Ravi Ram
Warren Anderson
Shyam Sivakumar
AMD, Inc.
Sunnyvale, Calif., Boxborough, Mass., USA
www.amd.com

Vijay Akkaraju
Synopsys
Mountain View, Calif., USA
www.synopsys.com

ABSTRACT

This paper describes application of RTL verification methodology in analog mixed-signal (AMS) designs. Given the increasing complexity of mixed-signal circuits, it is important to increase the volume and variance in stimulus to ensure functional correctness. Lengthy run times of SPICE-based mixed-signal simulations are a significant productivity bottleneck in coverage closure and testplan completion. Coupled with limitations of Verilog only analog models, alternative flows are needed. In this paper we consider one such approach: granular representation of low-level analog blocks in Verilog-AMS, using them to build complete analog circuits that can be instanced in a System Verilog top-level testbench for mixed-signal simulations run through VCS and CustomSim. We achieved a decrease of more than an order of magnitude in simulation time and an increase in the number of tests run, enabling faster identification of coverage holes. We expect this flow to form a framework for additional System Verilog concepts in mixed-signal contexts and to increase silicon quality significantly.
Table of Contents

1 Introduction .......................................................................................................................... 3
2 Target Design and Testbench .............................................................................................. 3
   2.1 DESIGN .......................................................................................................................... 3
3 Verification Methodology and Randomization .................................................................... 5
   3.1 DIGITAL VERIFICATION CONCEPTS ......................................................................... 5
      3.1.1 Universal Verification Methodology (UVM) .............................................................. 5
      3.1.2 Coverage .................................................................................................................. 5
      3.1.3 Joint Test Action Group (JTAG) ............................................................................... 5
3.2 VERIFICATION ENVIRONMENT AND TOOLS .............................................................. 5
3.3 RANDOMIZATION............................................................................................................ 6
4 Verilog-AMS Implementation and Simulation .................................................................... 7
   4.1 CODING ARCHITECTURE ............................................................................................ 7
   4.2 EXAMPLE VERILOG-AMS MODEL: OUTPUT DRIVER ................................................ 8
   4.3 EXAMPLE VERILOG-AMS MODEL: INPUT RECEIVER ................................................ 11
   4.4 VERILOG-AMS MODULE SIMULATION FOR MODEL CALIBRATION ..................... 13
   4.5 HIERARCHY AND MODEL RE-USE ........................................................................... 14
5 Results .................................................................................................................................. 14
   5.1 SIMULATION OUTPUT COMPARISON ......................................................................... 14
   5.2 SIMULATION TIME COMPARISONS FOR SOME TESTS ........................................... 15
   5.3 MIXED-SIGNAL VERIFICATION: COST VERSUS BENEFIT .................................... 15
   5.4 ISSUES AND VERILOG-AMS LIMITATIONS ............................................................. 16
6 Conclusions ........................................................................................................................ 16
7 Acknowledgements ........................................................................................................... 17
8 References ......................................................................................................................... 17

Table of Tables

Table 1: Simulation time comparisons ................................................................................. 15

Table of Figures

Figure 1: Block diagram of the DDR physical layer interface. .............................................. 4
Figure 2: Phy System Verilog test harness. .......................................................................... 6
Figure 3: Example code of analog variables that were randomized. .................................... 7
Figure 4: Output driver schematic. ...................................................................................... 9
Figure 5: Functional portion of the driver Verilog-AMS model. ......................................... 10
Figure 6: Output driver pull-down characteristics after calibration, representing one 240-ohm
   segment of the driver illustrated in Figure 5..................................................................... 11
Figure 7: Receiver block diagram. ....................................................................................... 12
Figure 8: Receiver comparator Verilog-AMS model. ......................................................... 13

SNUG 2012
UVM-based Random Verification in AMS Designs for Faster Coverage Closure
1 Introduction

Verification of analog and mixed-signal circuits in AMD system-on-a-chip designs (SoCs) presents unique challenges that do not easily lend themselves to adoption of conventional RTL verification methodologies. A significant bottleneck is the 12+-hour run times for typical simulations when running designs in mixed mode, using SPICE models for analog blocks, run in a fast SPICE simulator, coupled to Verilog models for digital blocks. Often, that limitation leads to only a small sub-set of the complete regression tests being run in the pre-silicon stage of the design.

As designs increase in complexity at 22-nm and below geometries, this model becomes unsustainable, both for run time and for coverage loss. Specifically, an increasing number of interactions between the analog and digital portions of the design expose many regions of functional coverage that can be hit only in mixed mode. Verilog models are not enough. We need a middle ground in which the scope and accuracy of analog modeling falls between simple behavioral code and SPICE. The former, for example by representing analog behavior with digital RTL, is a highly inaccurate and totally unsuitable for complex analog functions. The latter, such as raw SPICE or FastSPICE, is too slow and can be difficult to integrate into a full test environment. Any approach that looks to resolve the issue can also help in leveraging the typically large number of tests that are available from pure RTL (digital) verifications.

This paper describes one such approach. Specifically, we discuss building representations of complex analog circuits through granular Verilog-AMS models. Although these models are at a higher analog abstraction level than the equivalent SPICE netlist, they provide sufficient accuracy to do mixed-signal simulations. More importantly, these models enable increasing the number of tests that can be run in such a mode.

We describe the target design in Section 2 and verification environment in Section 3, followed by examples of Verilog-AMS code and its equivalent circuit in Section 4. Section 5 reviews the measurements and impact of this within the overall verification flow and quantifies the benefits of using this approach.

2 Target Design and Testbench

2.1 Design

The top level of the design is the physical layer (phy) of a DDR sub-system. This is integrated as IP in several SoCs. The focus of this paper will primarily be on this DDR phy, which transfers across a bi-directional bus between the processor and the DDR3 DRAM memory. As shown in Figure 1, the phy consists of two 64-bit channels driving a DDR3 memory interface. The channels consist of a central control section, a command interface, and a data interface. Data is transferred across the 64 data lanes through eight instances of a data byte block, shown in the inset.
Each data byte contains eight lanes of bi-directional data, with associated output drivers and input receivers, and two differential strobe pairs, along with three DLLs to control interface timing. The central control and calibration section is responsible for training clock delays, controlled through the DLL, and output and termination impedances.

![Block diagram of the DDR physical layer interface.](image)

The phy consists of numerous analog circuits, including the output driver; input receiver; input line impedance termination; calibration; reference voltage, current generation, and distribution; and, DLL. Each of these must function in the overall context of the phy, which is mostly digital. As such, the natural architecture is to control the analog functions through a digital interface into each analog block, leading to a large number of mixed-signal interactions that must be verified.

Details of the construction of the analog blocks and examples of verifying the mixed-signal interactions will be described in Section 4.
3 Verification Methodology and Randomization

3.1 Digital Verification Concepts:
This section briefly introduces some widely used concepts in digital verification for analog specialists in the targeted audience of the paper. For additional information on these topics, see the resources identified in the references section.

3.1.1 Universal Verification Methodology (UVM)
UVM is an industry-standard verification methodology developed to increase design verification efficiency and optimize testbench and testcase development time.

3.1.2 Coverage
Coverage is a metric used to measure the completeness of verification of a design. It can be broadly divided in two categories:
   a) Code coverage is the portion of RTL code (Verilog) exercised in a regression run of all test cases.
   b) Functional coverage is the portion of code exercised under tests specifically written towards device functionality using the System Verilog covergroup construct.

3.1.3 Joint Test Action Group (JTAG)
JTAG is the common name for the IEEE 1149.1 Standard Test Access Port and Boundary Scan Architecture. It is an industry standard originally developed for testing interconnects in PCBs, but is now widely used as a standard access mechanism to a variety of debug hardware on an integrated circuit. Implementation of digital logic (RTL) for this functional is essential to improve testability of the silicon in the bring-up and production stages.

3.2 Verification Environment and Tools
The verification environment is implemented using a System Verilog-based UVM flow. We have separate agents for each 32-bit channel, JTAG, global, and passive agents with only active monitors on the memory side. Each active agent has a sequence, driver, and monitor. Transactions are sent through hierarchical sequences and a virtual sequencer. Randomization is done based on a shared global config object passed around all these agents. As part of the AMS flow, we swap the existing Verilog model with a Verilog-AMS model and run the same sequences. For configuring the vendor memory model, we have a handle to a System Verilog object inside the memory model through which we configure the memory as part of System Verilog test through global object randomization.

Mixed-signal simulations were performed with Synopsys’ integrated VCS and CustomSim environment. The directly integrated kernel in this combination can co-simulate RTL and analog functions, where each can come from a variety of representations. In our case of Verilog plus Verilog-AMS, the VCS digital kernel performs event-based simulation of the Verilog RTL and the digital sections of the Verilog-AMS modules, while the conservative matrix solver in CustomSim evaluates the analog blocks within the Verilog-AMS modules.

We compare the results using the methodology described here with simulations in an earlier but similar project where we used the Nanosim and CustomSim FastSPICE simulators running on a
transistor netlist. These tools are now integrated into CustomSim, but we will continue to refer to this earlier method as FastSPICE to distinguish the transistor-level from the Verilog-AMS representations.

3.3 Randomization

In RTL verification, the stimulus in the design is randomized (within certain constraints) to ensure modeling as many scenarios possible during regular function of the silicon. The goal is to complement directed test cases and expose the RTL to corner cases not predicted in the testplan.
This randomization is achieved in the DDR PHY testbench using the constraint construct in System Verilog. A constraint block specifies the range in which variables in the testbench will be randomized by the digital simulator (VCS). This randomization itself is a simulation-stage behavior set after the test case identifies the top-level functionality it is looking to verify. To perform the randomization, we use the VCS constraint solver, which is a randomization engine in the simulator that takes as input the variables specified in the System Verilog constraint construct and then randomizes them within the specified range during simulation of the design+testbench.

Including randomization in mixed-signal test cases adds significant coverage to digital/analog interactions. Examples of parameters that can be randomized include comparator and receiver offsets, process and temperature conditions, which affect analog drive strengths, and delays. When verification runs span the allowed range for these parameters, it enables verification of a wide set of possible interactions that could occur on silicon. Section 4 shows examples of Verilog-AMS representations that contain such parameters. Using the model from Figure 5, Figure 3 shows analog values that can be randomized using the UVM constraint randomization flow. Even though this example shows static constants, there is a flow developed to randomize them dynamically.

```verbatim
parameter real PFactorResProc = 1.0 from [0.9:1.12]; // Process factor for resistance process variation // Minimum value is max R, maximum value is min R parameter real PFactorResTemp = 1.0 from [0.97:1.0]; // Process factor for resistance temperature variation // T=0 is 1.0, T=100 is min value parameter real PFactorXtorProc = 1.0 from [0.74:1.29]; // Process factor for transistor process variation // Min value is slow process, max value is fast process parameter real PFactorXtorTemp = 1.0 from [0.85:1.0]; // Process factor for transistor temperature variation // T=0 is 1.0, T=100 is min value
```

Figure 3: Example code of analog variables that were randomized.

4 Verilog-AMS Implementation and Simulation

4.1 Coding Architecture

We have represented the behavior of mixed-signal blocks through Verilog-AMS models. Verilog-AMS was chosen based on the construction of these blocks, which generally consists of digital inputs, such as enable signals and DAC buses controlling the analog function. Verilog-AMS lends itself nicely to retaining the digital nature of the control signals, allowing easy conversion of enable signals into on-off switches and DAC buses into numerical values that can be arithmetically applied in analog calculations. Although the same functions could be coded in Verilog-A, Verilog-A forces all nets in the design to be analog, resulting in more converters at the block boundary and more internal analog nets that must run through the analog conservative matrix solver. The calculation-based approach afforded through Verilog-AMS minimizes the number of analog nets, thereby decreasing simulation time.
We took great care in structuring the design of the mixed-signal blocks to enable Verilog-AMS representation. Many mixed-signal blocks contain both digital and analog functions. Digital logic was coded at the top level of the mixed-signal block module, but the Verilog-AMS function was placed into a sub-module. The circuit’s schematic was constructed in a similar fashion, matching the pins of the analog sub-module. Although schematic hierarchy was allowed underneath each of the digital and analog sections, matching the pins and function of the Verilog-AMS sub-module allowed the Verilog-AMS model to be checked for equivalence against a matching schematic at any level of hierarchy, which created significant flexibility when constructing equivalence checks.

Equivalence checking was done through two separate steps. The first step checked the digital portion of the mixed-signal function, which can be properly represented through standard Verilog RTL. We invoked a commercially available automated digital equivalence-checking tool to check the logical equivalence between RTL and schematic. During this step of the check, the Verilog-AMS sub-module was black-boxed and checked only for pin matching. The second step verified the equivalence of the analog function in the Verilog-AMS module with that of the schematic. This step must be performed manually because a continuous analog function cannot be reduced to a finite vector set as in digital equivalence checking. The stimulus fed to the analog block invokes its function, no different than a pure SPICE simulation. For rigorous comparison, an identical HSPICE stimulus file is used to drive both the schematic and Verilog-AMS models. The output waveforms are compared by hand to match critical performance parameters. Section 4.2 illustrates an example.

Accuracy of the model and its tolerance during calibration cannot be defined exactly, but must be determined through knowledge of the design, its sensitivities, and the accuracy needed to verify interactions with digital logic. Since the goal of mixed-signal simulation with Verilog-AMS is to reduce the number of analog nets in the matrix solver, the Verilog-AMS model must be sufficiently abstract to minimize these nets, but must also behave in close approximation to the transistor-level circuit. A good example is a digital-to-analog converter (DAC) that receives a digital bus and converts that to an analog voltage. In a simple representation, a Verilog-AMS model would treat the digital bus as a numerical value, divide it by its full scale, and apply that fraction to the full scale voltage to determine the DAC output. This neglects transistor mismatch, process variation, and systematic effects that occur in DACs. Therefore, a more detailed representation would add numerical parameters that generate non-linearity in a systematic or random fashion to the model. This opens a wide space in which to choose the level of detail to represent the circuit, requiring knowledge of the application to guide the model writing and calibration process. Since the main goal here is mixed-signal functional verification, the model should only be written to the level of detail needed to exhibit the behavior to be verified. We demonstrate this process through several model and simulation examples.

4.2 Example Verilog-AMS Model: output driver

Figure 4 shows the schematic diagram of the output driver. It consists of a pull-up and pull-down data path controlled through a half-rate clock (DLLclk). In each path, data is transferred to the output through a serializer mux. The final data feeds six pull-up and pull-down banks, where it passes through a slew-rate control pre-driver section and then to the final drive stage. The impedance of the pull-up and pull-down banks is respectively controlled by the 8-bit calP and calN
buses, which enable binary-weighted legs in the output stage. Banks are sized to target 120, 240, or 480 ohm and can be enabled individually to form a variety of possible impedances.

Only the final output driver section contains any analog behavior. The serialization is performed through variable-delay modeling in standard Verilog. The final pull-up and pull-down sections are modeled as a combination of parameterized transistor and resistor components. The pull-down model, shown in Figure 5, consists of a nominal resistance scaled by process variation, temperature, and bank strength. The transistor portion consists of a fourth-order polynomial fit scaled by process variation and temperature parameters.

```verilog
parameter real Rbank = 240.0;
// Resistance of bank, scalable across different drive strength banks
integer BaseWeight = 64 * 3.5 / 2.0;
// Unit of weight is the LSB of the binary legs
real Rbank_unit = 240.0;
// Unit cell used for calibration of the driver versus code curve here
real Rint = 132.0;
// Portion of the driver that is series resistance
real Rscale, Tscale, Ncal_ena;
always @* begin
   Rscale = (Rbank / Rbank_unit) / PFactorResProc / PFactorResTemp;
   Tscale = PFactorXtorProc * PFactorXtorTemp / (Rbank / Rbank_unit);
   Ncal_ena = BaseWeight + {8{ng_dat}} & ncal[7:0];
end
```

**Figure 4: Output driver schematic.**

Only the final output driver section contains any analog behavior. The serialization is performed through variable-delay modeling in standard Verilog. The final pull-up and pull-down sections are modeled as a combination of parameterized transistor and resistor components. The pull-down model, shown in Figure 5, consists of a nominal resistance scaled by process variation, temperature, and bank strength. The transistor portion consists of a fourth-order polynomial fit scaled by process variation and temperature parameters.
V(PAD_VIO, pad_int) <+ I(PAD_VIO, pad_int) * Rint * Rscale;

// Transistor portion
I(pad_int, VSS) <+ Ncal_ena * Tscale *
( xivp4 * V(pad_int, VSS) +
 xivp3 * V(pad_int, VSS) * V(pad_int, VSS) +
 xivp2 * V(pad_int, VSS) * V(pad_int, VSS) * V(pad_int, VSS) +
 xivp1 * V(pad_int, VSS) * V(pad_int, VSS) * V(pad_int, VSS) * V(pad_int, VSS) );
end

Figure 5: Functional portion of the driver Verilog-AMS model.

Model parameters were calibrated through matching to the actual circuit characteristics. First, we traced the transistor portion of the driver’s base leg I-V characteristic by shorting the resistor. After some exploration through MatLab fitting functions, we found that a fourth-order polynomial reproduced the transistor curve reasonably. Note that the transistor’s gate is always tied to the supply rail, which greatly simplifies this exercise compared to fitting a full compact model. Transistor temperature and process coefficients were determined with the same configuration and entered as parameter bounds. Similarly, resistor coefficients were determined by shorting the transistor portion and measuring the resistor behavior. Note that the resistor parameters in the model are inverses of standard resistor coefficients, but were defined so minimum values of both transistor and resistor parameters always yield minimum current.

An I-V trace of the resulting model for the driver pull-down is shown in Figure 6. The model fit is quite good, particularly in the voltage range below 1.0V, which is the most important region for driver operation. The fourth-order polynomial captures the transistor non-linearity, which is important to model in order to verify the operation of the driver impedance calibration block under non-linear I-V conditions. This example was done for one particular value of the calibration code Ncal, but, as the code in Figure 5 illustrates, the behavior with other values is a matter of simple scaling.
4.3 Example Verilog-AMS Model: input receiver

The DDR receiver is intended for reading high-speed data from the memory (DRAM). It accomplishes this by slicing the incoming high-speed, low-swing data signal (on RxIn input) using an internally generated reference voltage, amplifying it, and sampling it internally to propagate CMOS levels downstream. To enable loop-unrolled distributed feedback equalization (DFE), three differential amplifier stages, a three-output voltage reference DAC, and a clocked sense amplifier are employed on the receive data path.
The combination of the amplifier stages and the clocked sense amplifier was modeled in Verilog-AMS as an “analog comparator followed by a digital flip-flop.” The comparator (code excerpt in Figure 8) acts as a single-bit A/D and captures the slicing function by accepting an analog input and a slicer voltage and transmitting a digital output to the flip-flop. Further, the DAC generating the slicer voltage is composed of a decoder in its front end and an analog voltage generator in its back end. The decoder was modeled in Verilog while the analog back-end was modeled in Verilog-AMS.

```verilog
module ddr_comparator(input p, input n, output out);

  parameter real CmprOffset = 0.0; //default
  parameter real CmprHyst = 1.0; //default
  parameter real CmprDelay = 0.0; //default

  electrical p, n, vid;
  reg out;

  real threshhi = CmprOffset + (0.5*CmprHyst);
  real threshlo = CmprOffset - (0.5*CmprHyst);

  always @(above(V(vid)-threshhi)) begin
    #(CmprDelay/1e-12) out = 1'b1;
    end

  always @(above(threshlo-V(vid))) begin
    #(CmprDelay/1e-12) out = 1'b0;
    end

endmodule
```

Figure 7: Receiver block diagram.
analog begin
    @(initial_step) begin
    end
    V(vid) <+ V(p,n);
end
endmodule

Figure 8: Receiver comparator Verilog-AMS model.

For the comparator, a user-programmable offset, hysteresis, and delay were added. For the DAC, the user could set its range and resolution. This allowed us to verify overall functionality and robustness amid analog non-idealities.

4.4 **Verilog-AMS Module Simulation for Model Calibration**

As described in Section 3.2, we used the integrated VCS + CustomSim combination to co-simulate the Verilog and Verilog-AMS modules. For simulating the functionality of a Verilog-AMS module with HSPICE stimulus, we use the SPICE-top form as shown in Figure 9 for the driver model of Figure 5. The file runsim.sh compiles and runs the simulation through VCS. It calls the driver pulldown model, drvpd.vams, a parameters file, drvpd.cfg, the default Synopsys connect modules, and sets several other switches. Details can be found in Synopsys documentation on mixed-signal simulation [5] for the CustomSim-VCS-AMS flow.

```
rundev.sh
  vcs +v2k
    drvpd.vams
    -parameters drvpd.cfg
    $AMS_HOME/snps_cm_d2a_1.vams
    $AMS_HOME/snps_cm_a2d_1.vams
    $AMS_HOME/snps_cm_bidir_1.vams
    $AMS_HOME/snps_crules_1_10.vams
    -ams
    -ad=vcsAD.init
    -ams_discipline logic
    -R

drvpd.cfg
  assign 1.00 snps_sptop.xdrv_pd_240_dut.PFactorResProc
  assign 1.00 snps_sptop.xdrv_pd_240_dut.PFactorResTemp
  assign 1.00 snps_sptop.xdrv_pd_240_dut.PFactorXtorProc
  assign 1.00 snps_sptop.xdrv_pd_240_dut.PFactorXtorTemp

vcsAD.init
  spice_top;
  choose xa -hspice top.spi -c xa.cfg -o ana;

top.spi
  *--------DUT Instantiation-----------------------------*$
  *--------Stimulus-----------------------------------*$
  VMEMIO VMEMIO VSS dc p_vmemio
  VPAD_VIO PAD_VIO VSS pwl 0 0 1.5u p_pad_vio
  Vnbase_en nbase_en VSS dc p_nbase_en
  * etc.
```

SNUG 2012
HSPICE simulation for model calibration was performed using the same top.spi file, but with the raw transistor netlist from a schematic or extracted from layout substituted for the Verilog-AMS model.

4.5 Hierarchy and Model Re-use
To minimize the amount of work to write and calibrate models, we endeavored to minimize the number of analog models through re-use of analog building block functions in the Verilog-AMS models. Such analog building block models include switches for muxing functions, DACs, and comparators. Optimal re-use kept the total number of base Verilog-AMS modules to 16 for the entire phy. These were built into models for the mixed-signal blocks described in Section 2, representing the function of the 64 data lanes, 26 address lanes, calibration circuits, and 22 unique analog blocks across the full phy.

In addition to circuit representations for the DDR phy, these models are also available for use for mixed-signal verification of other I/O interfaces, such as PCIe and HyperTransport™, on AMD products. Some modifications will be necessary, but writing the models from low-level building blocks enables easier adaptation to other I/O interfaces. Likewise, the full UVM mixed-signal methodology will soon be enabled for these other interfaces as well.

5 Results

5.1 Simulation Output Comparison
Figure 10 is a simple example of a Verilog-AMS versus a pure Verilog simulation. It shows four data bursts propagating to the off-chip driver output from Figure 4 with one-tap de-emphasis equalization enabled. The zero/one threshold is at 0.75V.

![Figure 10: Off-chip driver output in Verilog-AMS (upper) and pure Verilog (lower).](image)

The correspondence between the two simulations is clear. It is also clear that the digital simulation cannot represent the continuous levels of the output. In the pure Verilog simulation, the equalization behavior is lost and cannot be verified without other means.

SNUG 2012
We show only a simple Verilog versus Verilog-AMS example because it is not possible to show a comparison for more complex cases, such as the digitally controlled on-chip calibration loop. This circuit block adjusts the strength of driver pull-up and pull-down calibration values (calN and calP in Figure 4) to establish an effective output impedance. The digital values are adjusted until two analog signals become equal to each other (within one DAC LSB). These continuous analog signals have no corresponding equivalent in the fully digital realm. The digital equivalent simulation is not possible without inventing some kind of numerical representation.

Figure 11 shows a simulation of the Verilog-AMS representation of the compensation loop. The loop runs through four successive approximation sequences, each of which converges with an analog value of V(pos) = V(neg) = 500mV. The loop adjusts the calN and calP values to achieve the convergence. The loop also controls 15 thermometer bits (not shown in Figure 4).

![Figure 11: Verilog-AMS simulation of the compensation loop.](image)

### 5.2 Simulation Time Comparisons for Some Tests

Table 1 shows a summary of simulation time differences for tests using fast SPICE and the same test run with Verilog-AMS model.

<table>
<thead>
<tr>
<th>Test Name</th>
<th>Verilog-AMS wall-clock time</th>
<th>FastSPICE wall clock time</th>
<th>Simulation speed-up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output impedance calibration cycle</td>
<td>3 minutes</td>
<td>28 hours</td>
<td>560x</td>
</tr>
<tr>
<td>Output driver functionality</td>
<td>6 seconds</td>
<td>30 minutes</td>
<td>300x</td>
</tr>
</tbody>
</table>

Simulation accuracy is set by the tolerance of the simulator combined with the accuracy of the Verilog-AMS models, which, as described in Section 4.1, is as much art as science. The models must be written with verification goals in mind. In the case of the compensation loop, we want to see convergence of the calibration cycle across a range of parameters representing process, voltage, and temperature variation as well as comparator offset. In the driver functionality case, we want to observe the function of the equalization logic across equalization settings to verify its operation.

### 5.3 Mixed-signal Verification: cost versus benefit

For a project of this magnitude, with requirements to ship first silicon to customers, the benefit of mixed-signal simulation far outweighed the cost. This project represented a complete re-
architecture of the DDR phy, including extensive training and control features. As such, we could not rely on prior debug and proof of mixed-signal functionality for silicon.

We found 23 unique design bugs through mixed-signal verification. Although most of these could have been found through FastSPICE simulation, the ability to write AMS models before circuit design completion enabled many of these bugs to be caught earlier in the project, when their impact was less severe. In addition, the faster simulation speed of the AMS models allowed for easier test creation through faster turnaround time.

Across this full project, we invested approximately 120 person-days to create and calibrate the Verilog-AMS models. This represents only a few percent of the total design work on the project.

From a digital verification point of view, the design in its current release had many portions that could be exercised only in a mixed-signal simulation mode. Coverage closure would require a large number of test cases, which is not practical in a reasonable time using a Verilog-plus-SPICE approach. Verilog-AMS delivered simulation times fast enough for coverage closure to be achieved in the available project design time frames.

Other immediate benefits included the ability to run a large portion of the design regression, including most random and directed test cases, in mixed-signal mode, enabling coverage of large number of corner cases.

5.4 Issues and Verilog-AMS Limitations
Among current limitations of the flow is non-availability of a tool to perform equivalency checking between schematics and Verilog-AMS code. This increases the time to ensure accuracy of the Verilog-AMS code because checking must be done by hand.

Despite the granular representation of the Verilog-AMS code described in the paper, not all of it is “plug and play” for similar designs among multiple projects. The inherent nature of design uniqueness requires modifications to the Verilog-AMS code developed from one project to other. In other words the concept of object orientation, which allows re-use of code in UVM/System Verilog, is not really practical in Verilog-AMS.

6 Conclusions
We have demonstrated the advantage of applying the UVM methodology to the verification of mixed-signal designs using Verilog-AMS representations for analog functions simulated through VCS+CustomSim tool. The Verilog-AMS models were written to include parameters that cover expected variation across process, voltage, temperature, and device mismatch. Verilog-AMS models significantly reduced simulation time compared to transistor-based FastSPICE simulations. Furthermore, we demonstrated a method for comparing the Verilog-AMS output to the transistor-based schematic output through a common HSPICE test bench.
7 Acknowledgements

The authors would like to acknowledge Mei-Cheng Huang for her contribution to integrate the Verilog-AMS methodology into AMD’s standard verification environment. We are grateful for the extensive support from Synopsys’ David Cronauer and Maureen Ladd.

8 References