UVM Based Approach To Model Validation For SV-RNM Behavioral Models

Donald Lewis and Courtney Fricano
Analog Devices, Inc.
3 Technology Way
Norwood, MA 02062

***Abstract*- The prevalence and complexity of mixed-signal designs are rapidly increasing in the industry. With increased complexity, the need for verification of these designs becomes more critical. Often, top level co-simulations of the full chip are not feasible, resulting in the use of behavioral models for analog sub-blocks to achieve simulation speed-up. Model validation is the process of confirming that these digital models correctly simulate the analog functionality of the block. It is generally easier to debug and verify behavioral models by running block level simulations. This paper discusses a UVM based approach to model validation at the block level.**

I. Introduction

Across the industry, analog and mixed-signal (AMS) designs are becoming more common. The AMS portions of chips are also increasing in scope, complexity, and integration with the chip. Additionally, there are more instances of the signal path and feedback crossing between the analog and digital domains of the chip. These factors make verification of the AMS designs critical to the success of chip tapeouts.

For the digital domain, metric driven verification (MDV) is used to ensure high verification quality. Figure 1 below shows the basic MDV closed loop flow. The MDV framework is enabled by verification planning, testbench development using the Universal Verification Methodology (UVM), and running regressions. A typical UVM testbench makes use of SystemVerilog's [1] constrained randomization capabilities and regular regressions to collect coverage over many runs. Simulations that are exclusively digital are able to be simulated very quickly, allowing the MDV loop to be iterated many times over the course of the verification process. The MDV closed loop flow, when followed, has been proven in the industry to result in successful verification of designs.

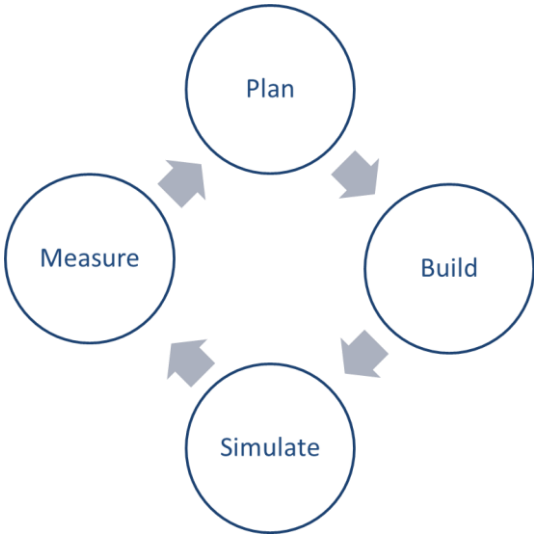

Figure 1. MDV Closed Loop Flow

AMS designs typically simulate at much slower speeds than digital designs due to their reliance on numerical solvers. Top-level simulations with multiple AMS blocks can take days, if not weeks, depending upon the design. The MDV flow requires multiple iterations and typically relies on the ability to complete many simulations a night, which is not possible with very slow AMS simulations. The solution is to use behavioral models of the analog blocks to reduce the simulation speed at the top level.

With the enhancement to SystemVerilog's handling of real-numbered ports and nets in SystemVerilog-2012, writing real-number models (RNM) in SystemVerilog (SV) has become more straightforward and easier to do. However, a key issue whenever behavioral models are used is the validation of the models with respect to the actual analog design. This paper describes a UVM based approach to model validation. This approach provides a high-level of confidence in the analog model which allows for the use of the iterative MDV flow at the top level. The approach also enables the re-use of testbench components from the block level testbench at the top level to provide detailed functional checking in top level simulations.

II. Approach

A SV-RNM behavioral model allows for fast simulation at the top level as well as easy integration into the digital testbench environment. For system level UVM based testbenches, being able to run the whole chip with valid models is critical for identifying potential issues. Typically, at the system level, the testbench is validating some functional aspects as well as the connectivity of the AMS blocks through their models. In order to validate and verify the model, the approach that we used was to run both the model and the AMS block through the same block level UVM testbench.

The idea behind this approach is that the block level testbench will be able to verify functionality and pin behavior. Using the MDV flow with a UVM testbench allows us to use constrained random test cases and regressions to fully verify the block. Then, once the testbench is in place, it can be used to check both the RTL model as well as the analog netlist. Figure 2 provides an overview of how this approach works. By running the same test cases, stimulus, and checks against both the analog netlist and the RTL model, a 1 to 1 comparison can be made to validate the model.

This UVM model-validation approach requires the ability to run Verilog co-simulations with the analog solver, as well as a standalone Verilog simulator. Additionally, a way to manage and compare the regressions of the behavioral model simulations to the co-simulations is required. This will work well for analog blocks that are small enough such that the simulation time is fast enough to allow for regular regressions. The ability to run regressions regularly (nightly in our case) is important as the regression delta is key to monitoring the health and accuracy of the model. Creating a UVM testbench for each analog block does result in some additional overhead. However, provided the system level testbench is also UVM based, there often will be an opportunity for agent re-use.

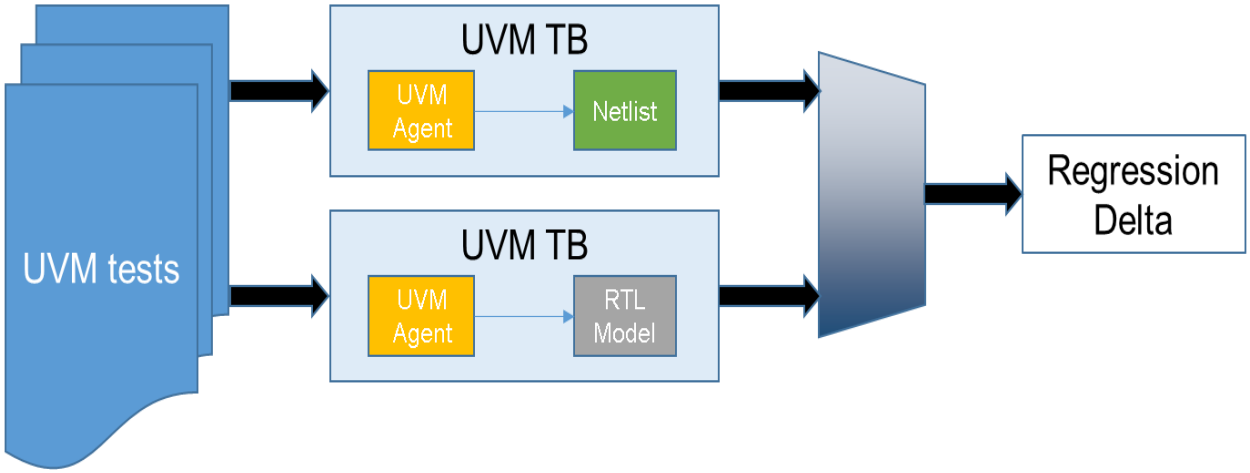

Figure 2. UVM based Model Validation Approach

Figure 3 depicts how the block level UVM agents could be re-used vertically at the system level. The re-use of the UVM agents provides additional confidence in the model and can provide insight into the veracity of the signals provided to the block at the system level. Re-using the monitors for the block in a passive manner at the system level ensures that any issues with signals going to the block will be found. Additionally, the re-use of TB agents helps to mitigate the impact of the upfront overhead related to creating a UVM testbench for each analog block.

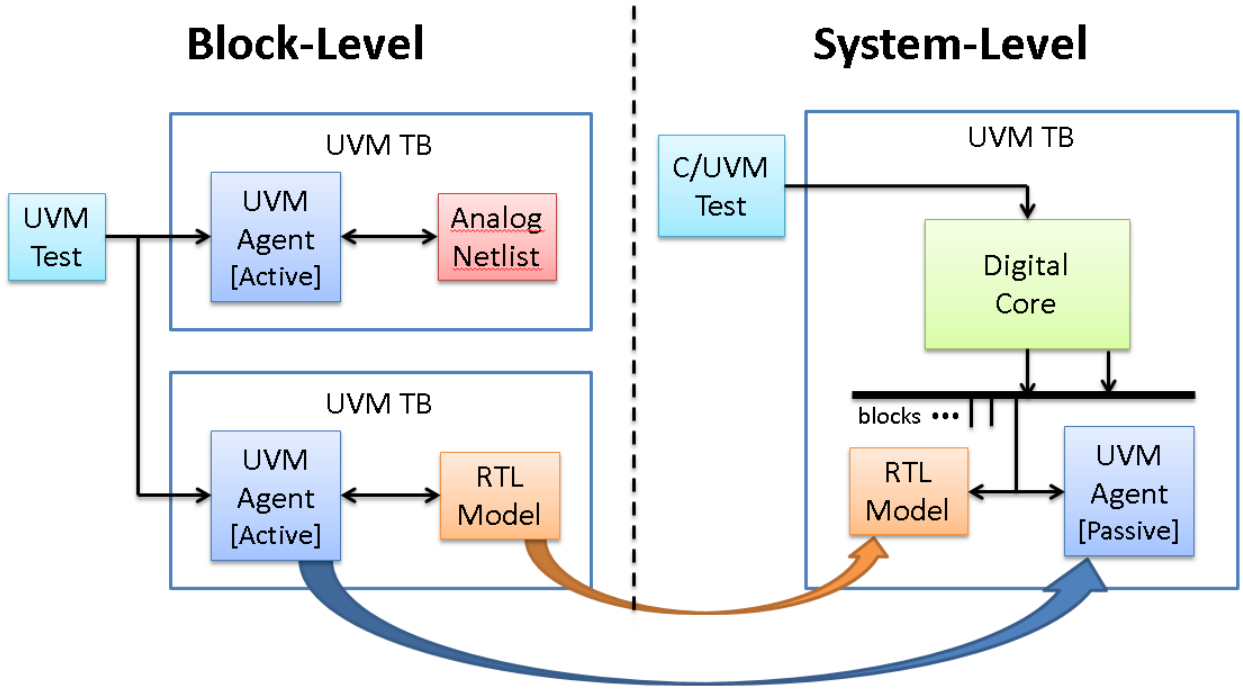

Figure 3. System Level Reuse

III. IMPLEMENTATION

To implement this strategy, the block level testbench must be architected to allow for co-simulation as well as digital only simulations. To get an accurate comparison of the behavioral model to the design, the same stimulus must be provided in both environments. As a result, the stimulus should be driven by the UVM testbench, so the testbench should always be running a digital simulator.

Generally, if the analog simulator supports co-simulations with the digital simulator as the master, a UVM testbench can be created for the behavioral model and then re-used by replacing the RTL model with the analog circuit for co-simulation. If required, the testbench can be architected to allow for either simulator to be the master and drive the simulation. Due to design requirements, some blocks require the analog simulator to be the master. The architecture that was used to allow for different simulator masters in the same testbench is shown in Figures 4 and 5.

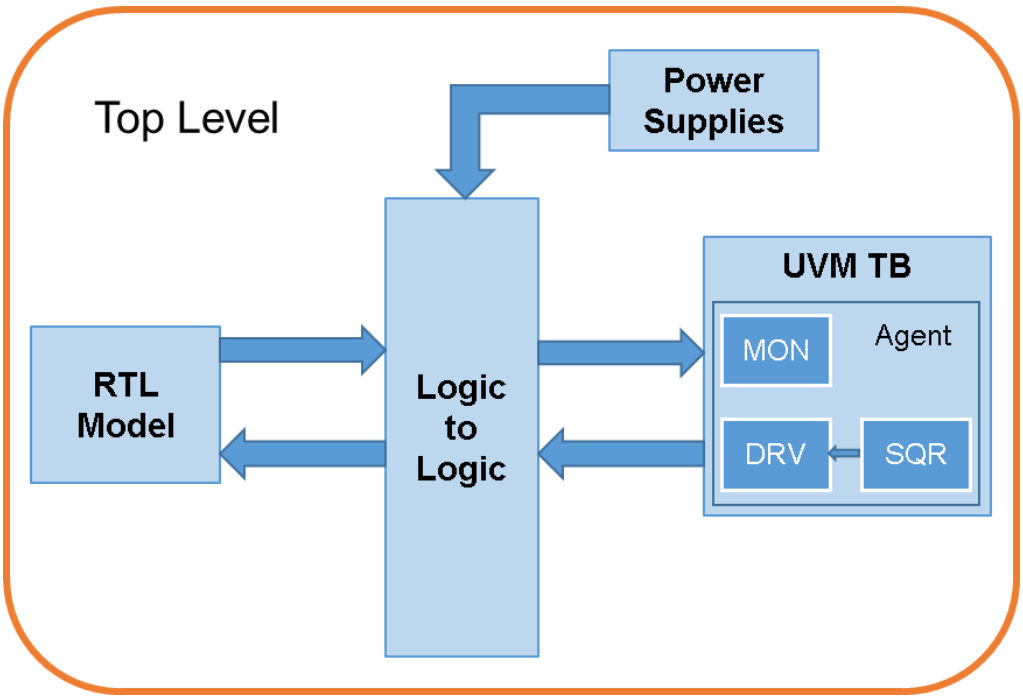

Figure 4. Block Level Digital Testbench Architecture

Figure 4 depicts the setup of the testbench for the digital only simulations involving the behavioral model, while Figure 5 shows the replacement of the model, interface, and power supplies with analog blocks.

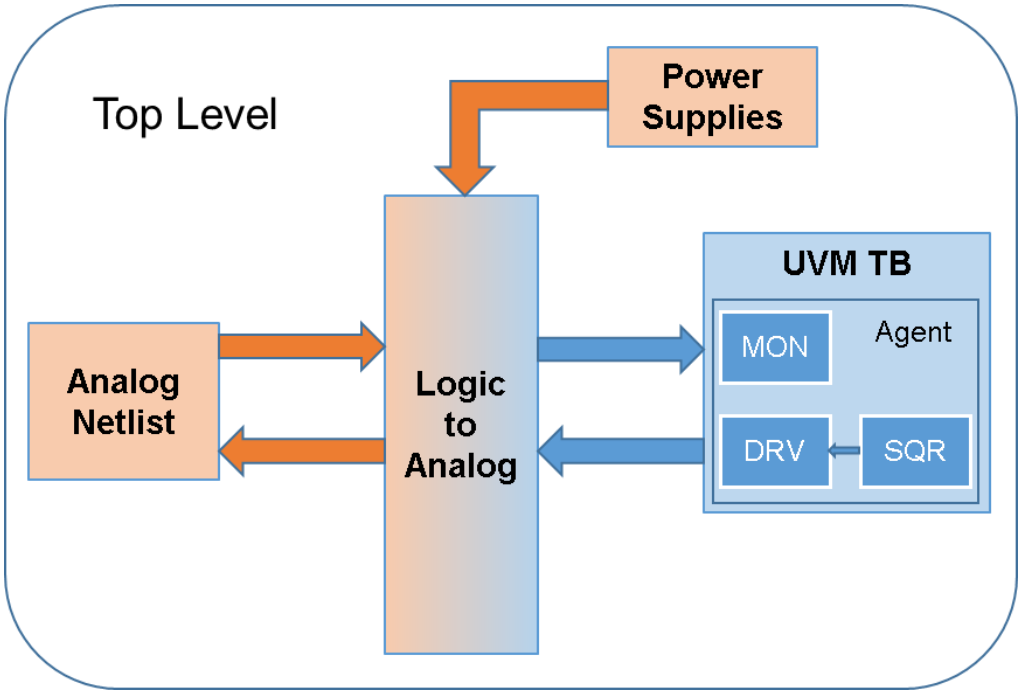

Figure 5. Block Level Co-simulation Testbench Architecture

The scope of what the behavioral models will be functionally modeling is critical. The scope will drive which aspects of the models need to be validated at the block level. In this project, the behavioral models were being used to verify functionality and connectivity of the top level digital simulation. Some deviation between the behavioral model and the analog design will be present, but the limits of what we deemed acceptable was driven by the block's specification as well as the ultimate function of the models. For example, for one of the blocks there was a specification that a signal was required to propagate within a certain time. Figure 6 shows how this was checked using a SystemVerilog assertion. This check was used in simulations for both the behavioral model as well as the analog design. If the check passed in both simulations, then the model and the design both follow the specification. There may be variation in the delay between the model and the design, but in this case this variation will not have an impact on functionality. If it does, the check should be adjusted to match the functional requirements.

```
property sva_prop_tsbh_check;
    time pdm33_3v_change;
    disable iff(!resetb_3v || fault_code === 3'h4 || !rstb)
    @(negedge tb_clk) ($rose(pdm33_3v)) |->  (((pdm33_changed - pdm33_timer) >= min_tsbh) &&
(pdm33_3v == pdm33_set_value));
endproperty : sva_prop_tsbh_check
```

Figure 6. Specification Check From Testbench

An example of a check that was adjusted to allow for variation in the signal due to the analog nature of the circuit was a check of the glitch rejection circuitry of the block. Figure 7 shows how the glitch rejection specification for one of the blocks was checked. For this specification, a window for a valid edge is defined and then verified. The behavioral model and the design passing this check shows that the model has the proper functionality.

```
//check for glitch pulses
if(this.ctl_en === 1) begin
// count increments based on voltage level
    if(evdd_in > ovlo_e_th && this.ctl_evddcompen == 1) evdd_ov = evdd_ov + inc;
        else evdd_ov = 0;
    if(evdd_in < uvlo_e_th && this.ctl_evddcompen == 1) evdd_uv = evdd_uv + inc;
        else evdd_uv = 0;
    //...
end

// check multiple thresholds to determine if an error must be flagged
if(evdd_max > 64 || evdd_ov > 100 || evdd_uv > 100 || evdd_min > 64) evdd_err_req = 1;

//...
// check multiple thresholds to determine if an error can be flagged to complete the window
evdd_err_valid = (evdd_max >= 10 || evdd_ov >= 12 || evdd_uv >= 12 || evdd_min >= 10);

//...
// when TB sees error flag
if(!evdd_err_valid ) `uvm_error(get_type_name(),"evdd_err flagged before glitch timeout")
// if no error flag is seen
if(evdd_err_req) `uvm_error(get_type_name(),"evdd_err not latched after min glitch pulse")
```

Figure 7. Glitch Rejection Check

For the blocks in this project, the goal was not to have the behavioral models exactly match the analog behavior of the analog design across process, voltage, and temperature (PVT) conditions, but rather to get a close enough matching that would be appropriate for the interface with the top level digital simulations. If the requirements of the behavioral model were to change to require tighter matching, then the model and testbench would need to be adjusted accordingly. The key for this approach is that the requirements of the behavioral model and assumptions about what is and is not modeled must match up with the assumptions made at the top level testbench.

IV. RESULTS

By using this approach we were able to successfully validate the behavioral models for the AMS blocks on our chip. Multiple issues with both the behavioral models and the design were both found and tracked. One of the issues that we found as a result of this approach was a mismatch between the way test modes were implemented between the behavioral model and the design. The regression results for the test mode tests did not match, leading us to find an issue with the way the test modes had been implemented in the design.

Another example of an issue found using this approach was that the way an oscillator trim was modeled did not match the design. This finding led to an update of the behavioral model to more accurately model the trimming implementation. In addition to finding bugs in both the behavioral model and the design, this approach allowed for quick feedback whenever there was a change in either. If the model and the design got out of sync, regression failure mismatches quickly flagged the issue. Figure 8 shows an example of our regression tracking for one of the AMS block level testbenches that used this approach. Regression failures and mismatches are indicated in red, and triggered either a model, design, or testbench update as required. The chip was successfully taped out, with no functional bugs in the AMS blocks that used this verification approach.

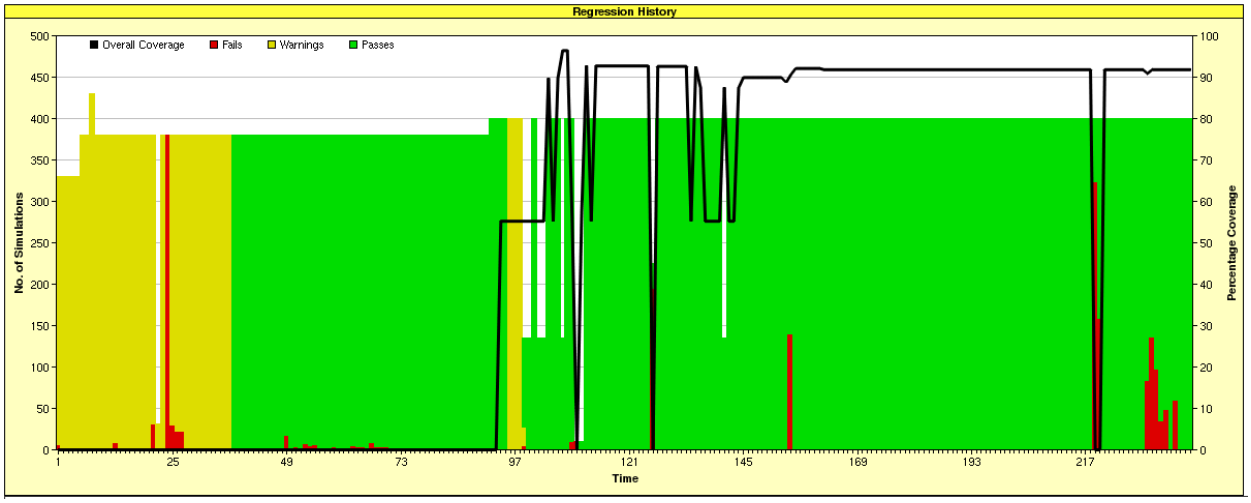

Figure 8. Regression History for AMS Block

The behavioral models were also critical in identifying issues and bugs at the system level. In one instance, the polarity of a reset pin was reversed in the system level and was identified through the block level monitors. Additionally, a signal sequencing error at the system level was found in the handshaking between an AMS block and another digital block. The resulting fix included a specification update, as the offending behavior existed in both the design and the behavioral model, which is the type of bug that would not have been caught without this model validation approach.

V. CONCLUSIONS

The approach for model validation presented in this paper had the following benefits:

- Allowed for thorough validation of an analog behavioral model from a functional perspective
- Provided quick feedback on the impact of model or design changes
- Facilitated the use of modern MDV methodologies on AMS blocks
- Enabled re-use of testbench agents at system level to reduce development time and ensure coherence between block and system level

The methodology presented in this paper successfully verified several new and modified analog blocks on our large multi-core SoC. The validated models were key in verifying top-level connectivity and functionality between blocks, and this approach found multiple bugs at both the block level and the top level.

This UVM based model validation approach is generic and simulator independent. The level of matching between the behavioral model and the AMS design can vary from purely functional to accurate across PVT variation as required by the design. The UVM testbench can be designed to validate the model at either block or top level. However, this approach is recommended for models that are focused on functional verification and connectivity, as SV-RNM models are generally not sufficient for checking chip power up and power draw analysis.

ACKNOWLEDGEMENTS

The following Analog Devices employees provided input and feedback that was critical to the writing of this paper: David Brownell, John Mackintosh.

REFERENCES

[1] IEEE Standard Verilog Hardware Description Language, IEEE Standard 1364, 2001.