Equivalence Validation of Analog Behavioral Models

Hardik Parekh[*], Manish Kumar Karna[*], Mohit Jain*, Atul Pandey[+], Sandeep Mittal[++]
[*]ST MICROELECTRONICS PVT. LTD., GREATER NOIDA,INDIA
{ HARDIK.PAREKH, MANISH.KARNA,MOHIT-JAIN.CRD}@ST.COM
[+]MENTOR GRAPHICS,MUNICH,GERMANY,[++]MENTOR GRAPHICS,NOIDA,INDIA
{ATUL_PANDEY,SANDEEP_MITTAL}@MENTOR.COM

***Abstract*—Mixed signal system-on-chips (MSSoC) integrate digital and analog functions on the same chip. The increased analog content in today's SoCs is tightly integrated to the digital portion of the design. Market segments such as power management, automotive, communication, and security applications are driving ever more integration of analog and digital content on MSSoCs. SoC verification requires a lot of effort to achieve good functional coverage. Additional complexity in MSSoCs arises from the interconnection of signals flowing between digital and analog domains. To achieve good verification coverage on mixed signal SoCs, abstract models of analog components (henceforth called analog IP) are used. These abstract models, commonly called behavioral models, capture functional features of analog behavior in digital HDL languages and are orders of magnitude faster than simulating SPICE views of analog IP. To effectively use the behavioral models in SoC level verification, it is important to establish the equivalence between the model and the implementation (SPICE). This paper will present essential components of an equivalence validation environment and commonly used methods.**

Keywords— MSSoC; behavioral models; IP; eqVC; HDL; verification plan; test plan; event-driven-simulation; assertion based verification

I. INTRODUCTION

Mixed signal system-on-chips (MSSoC) integrate analog and digital functions on a single chip. As the complexity of MSSoCs increase, the task of functional verification becomes harder and harder. Verification of MSSoCs requires verifying the analog as well as the digital functional specification of the complete design in a single verification environment. Traditionally, this was done using mixed signal simulation. Mixed signal simulation integrates a SPICE (analog) solver with an event-based (digital) solver. The main benefit of mixed signal simulation is that the digital and analog IP in the design can be verified together, including their interconnections. Mixed signal simulation provides SPICE-level accuracy for the analog IP and is typically faster than pure SPICE simulation due to the use of a digital solver for the digital IP in the design. However, simulation performance of mixed signal simulators is still not as fast as a pure event driven (digital) simulator. Another limitation of Mixed Signal simulators is capacity. The current trend of integrating more functions on SoCs means more transistors in the design. The analog solvers in mixed signal simulators cannot handle large SPICE (analog) content.

Due to the above mentioned limitations, behavioral models of analog IP are used in the verification of SoCs. The analog behavioral models are event driven and analog behavior is captured using either real-number (RN) or logic data types in HDLs. Analog behavioral models are compatible with digital simulators and are orders of magnitude faster than a mixed-signal simulation [1]. Functional verification of SoCs use behavioral models for analog design units to increase verification coverage on digital-analog interactions in the design. The use of behavioral models with digital simulation alleviates the capacity limitation of traditional mixed signal simulators. This approach enables the application of digital verification methodologies, such as verification planning and coverage tracking, to the analog IP in the design.

The behavioral model of analog IP is an abstract model of analog behavior. The more accurately the behavioral model represents analog behavior the slower its speed. Thus, to make it run faster, only relevant features for functional verification are implemented in the model; hence proving equivalence between the behavioral model and the actual implementation is a critical task.

This paper will describe our experience in setting up an equivalence validation environment for analog behavioral models with respect to the actual implementation (SPICE view). Section II presents a process oriented implementation of this environment. The need for verification planning, creating a test plan, monitoring progress, and coordinating team activity is discussed, as people with different skill sets are involved in this activity. The emphasis is on developing a methodology that can be easily extended and reused across different projects. Section III describes commonly used methods for equivalence validation. We refer to these methods as "equivalence validation components" (eqVC). The method chosen is based on the skill set of the verification team. The set up effort and various pros and cons of deploying each eqVC are also discussed. Section IV summarizes the results obtained and offers further recommendations.

II. EQUIVALENCE VALIDATION ENVIRONMENT

Behavioral modeling and verification of analog design units involves collaboration between analog, digital, and verification teams. Analog behavioral models range from simple models, that are connected together to maintain hierarchical equivalence with an analog design, to complex models capturing behavior across design units into a single model. The people responsible for model equivalence validation can be verification experts, or, in the case of simple models, analog designers may perform the task. The models and circuits keep changing over the project cycle. A lot of analog IP and their behavioral models are reused in multiple projects. Data and information exchange between the model

creator, circuit designer, and verification team must be efficient. While behavioral models of analog IP are used for functional verification signoff, it is the circuit created in the schematic and layout that goes in silicon. Hence, it is extremely important to prove equivalence between the behavioral model and the implemented circuit to keep state of model and circuit in-sync. If the implementation of features in a behavioral model is different or incomplete compared to the actual analog circuit, their use in SoC full-chip verification would either capture false mistakes or no mistakes at all. Many design and interface errors due to these differences are observed only after post silicon verification, resulting in costly design re-spins. It is prudent to set up a behavioral model equivalence framework consisting of verification planning, automation, and standardized information exchange. The framework should be such that it can be reused and extended across multiple projects.

A. *Framework Overview*

As mentioned earlier, analog behavioral models implement a selected set of functions of the actual circuit; it is therefore very important to have a specification document that details the functions implemented in the model. The involvement of the analog designer, model creator, and verification engineer are crucial for the success of this activity. Figure 1 shows the analog behavioral model validation framework used in this paper. The equivalence validation environment consists of a verification plan, a test plan, simulation, equivalence validation components (eqVC), results reports and review, and a waiver mechanism (for handling exceptions). The format test plan and report vary among the different eqVCs used. These components are chosen to establish a re-usable equivalence validation methodology that covers many aspects of modern verification techniques.

B. *Verification Planning*

At the beginning of the equivalence validation activity, a kickoff meeting with the stakeholders in the modeling, circuit implementation, and verification teams is set up. The aim of this meeting is to enable understanding of the objectives of equivalence validation and establish a communication channel between stakeholders. The communication between stakeholders is a very important aspect as the specifications might change during a project, which leads to changes in the circuit implementation and behavioral model, and, hence, equivalence validation has to be run again to ensure the behavioral model is compliant with the changes and the circuit and model remain equivalent. The outcome of this kick-off meeting is a verification plan document and testplan. Verification plans act as a reference for all the activities during the validation process and contains the following information:

1) *A list of the features of the analog behavioral model to be validated.*
2) *A list of tests to verify those features.*
3) *To test each feature, a list of checks (coverage) are specified in the test plan.*
4) *The equivaelnce validation method used and data organization.*
5) *Criterion for validation closure.*
6) *Method of review, reporting, and exception handling.*

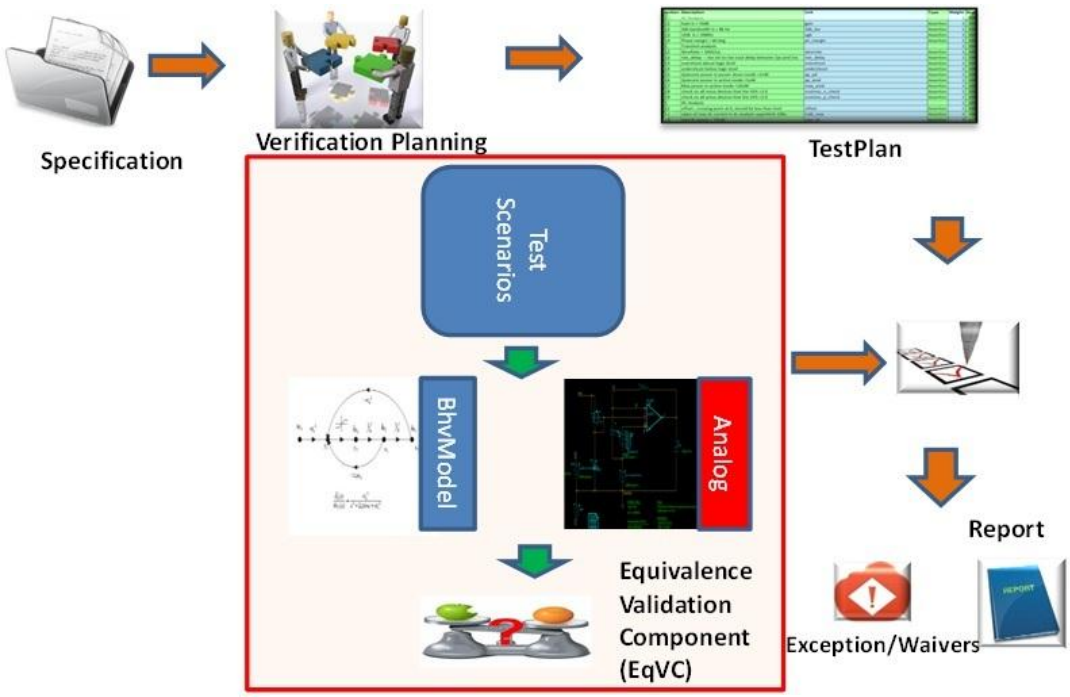

Fig. 1: Analog behavioral model validation environment.

Some feature specifications may be based on a worst case/best case value and, hence, require that the variability of analog circuits and parameters be considered. The verification plan must mention the technology corners and other parameters that need to be considered for such features.

C. Simulation Environment and eqVC

To validate equivalence between the implementation and the behavioral model of analog IP, every test has to be simulated twice. Initially, the device under verification (DUV) is a SPICE view of the analog IP, then it is changed to a behavioral model. Although the SPICE view and the behavioral model can be used in the same simulation, it is advisable to run them separately. Test scenarios are derived from the verification plan, and the eqVC (details about the types of eqVCs are in section III) is placed in the testbench environment to capture information that is used to compare if simulation of both types of DUV produce similar results. The important thing in this scheme is that exactly the same test(s) and eqVC are used for both types of DUV.

There are two common types of setups used for the testbench:

1) Analog-on-top: Schematic driven environment

2) Digital-on-top: HDL based top level.

The choice of an analog or digital on top verification environment corresponds to the verification team's proficiency in implementing stimulus. In this paper, a digital-on-top testbench environment is used as it enables the use of all types of eqVC methods. A mixed-signal simulator is used for equivalence validation that can handle both behavioral models and the SPICE view of analog IP without requiring a change in testbench. It is important to note that while using the SPICE view for simulation, the simulation time may be long; hence it is a good practice to optimize and simplify tests and test only the features that are realistic targets for functional verification. Another best practice to reduce simulation time is to not implement all analog functions of the design in a single behavioral model, but rather break it up into multiple, simpler models. This aligns with how analog circuits are designed in a modular way consisting of subcircuits. With such model implementation, basic functions of simpler models are validated separately for equivalence. These are quicker to simulate and, at this higher level (when these models are connected together), the validation focus is on interconnections and system functions. In some cases, depending on test objective, behavioral models of some blocks might be used to speed up simulations.

D. Verification Closure, Report, and Review

The results generated from simulations for all tests and captured in eqVCs are compared to the test plan to a generate PASS/FAIL report. The report generates a list of the checks that produce the same result for both behavioral model and SPICE simulation (PASS) and checks with different results (FAIL) for all tests. The closure criteria are established in the verification plan. Depending on the eqVC type used, the comparison technique varies. As the behavioral model is an abstract representation of analog implementation, differences between the simulation result for the behavioral model versus SPICE is quite commonplace. These differences arise from built in electrical effects in SPICE views but abstracted in behavioral model. A review meeting, involving the same stakeholders as were involved in the kick-off meeting, is held when all the tests have been performed for each DUV type and the validation report generated. The attendees of this meeting cross-check the report against closure criteria and sign-off the results if all of checks PASS. Some failed checks may be classified as PASS, with a waiver, after review. For waivers, it is important to add comments justifying the waiver and identifying the person who granted the waiver in the final report for traceability.

III. Equivalence Verification Component Types

Figure 2 lists various types of commonly used equivalence validation methods. The methods are classified based on the skill set of the verification team and the effort required for set up. In this section, the use model of these equivalence validation methods and the implementation of various components mentioned in Section II are described. QuestaADMS [2] has been used as a simulator for both analog behavioral and SPICE views of the DUV. QuestaADMS supports RealNumber data types in VHDL, Verilog-AMS, and SystemVerilog (SV) languages (WREAL). It is compatible with and extends advanced verification methodologies—such as assertion based verification (ABV), universal verification methodology (UVM), and low power—to mixed-signal simulation.

Quick Start	Assertion based Verification	Advanced Verification	Efficient
➢ Waveform compare ➢ Low setup effort ➢ Sufficient for simple models	➢ Assertion based Property Checkers and Monitors ➢ Medium setup effort ➢ Library of checkers/monitors	➢ UVM-AMS ➢ Advanced verification expertise ➢ Reusable verification environment	Wave Compare + Assertions + UVM-AMS

Figure 2: Equivalence validation component (EqVC) types

A. Waveform Compare

The wave compare method of equivalence validation requires the use of a waveform tool that has the capability to compare continuous waveforms and support automation through scripting. The EZWave[3] waveform tool is used in this paper as it offers the features mentioned above.

The advantage of this method is quick set up time, and it works for both analog-on-top and digital-on-top testbench verification environments. In many practical cases, especially for simple behavioral models, analog designers may be

required to verify model equivalence to the implementation. Analog designers prefer to use a schematic-driven verification environment and reuse a subset of design verification testbenches for equivalence validation. So this method provides a quick-start to the behavioral model validation effort. Digital-on-top testbench environments would use it in the absence of an ABV methodology. However, it is difficult to automate cross-referencing of simulation results to the test plan. Hence, some manual effort is required to ascertain quality of results. In this paper, automation is achieved through scripts that post-process the results and generate the PASS/FAIL report.

The kick-off meeting delivers a verification plan that defines the tests to be run. The test plan has a list of waveforms to be compared for each test (in some cases, only a single waveform list is generated, which is used for every test). Digital (behavioral models) and analog signals have differences arising from the electrical characteristics of analog circuits and the absence of these in the behavioral model. The following features are implemented in the waveform compare method to compensate for this effect:

1) *Tolerances are specified on waveforms for waveform comparison. The tolerances can be specified globally or on a per waveform basis. The tolerances are, typically, for the value of signals and delay in time.*
2) *The comparison between waveforms can be specified only in certain window of time.*
3) *Certain waveforms may need to be checked only at certain time points (clocked comparison). This clock can be a signal in the design or a virtual clock.*
4) *Many analog behavioral models have a current input or output port. Every net in a SPICE view has a flow (current) and branch (voltage) value; unlike behavioral models (except when using VHDL record type) where each signal carries one value. Comparison to the voltage or current value of the analog signal can be specified.*

The waveform compare method is used for BIAS and PLL models of HDMI IP. Figure 3 shows an example of a test plan for one of the tests for a behavioral model of a BIAS block of HDMI IP. The test plan contains the design to validate, the test name, the waveforms to compare, and the tolerances. Tolerances can be on the time or signal value. Clocked comparison is supported, and voltage or current comparisons can be specified locally or globally.

test_unit	BAIS_UNIT								
test_name	bmode								
analog_simulation_wdb	BIAS_UNIT_Top_Simulation_Analog.wdb								
Digital_Simulation_wdb	BIAS_UNIT_CALIB_Top_Simulation_Digital.wdb								
Waveforms to compare	Analog	Digital	voltage/cu	refDelay	fixedthres	start	end	xTol	yTol
	I_BIAS_UNIT:compout1v0		voltage	0n	0.5	15u	250u		
	I_BIAS_UNIT:vbgout		voltage	0n		10u	250u		
	I_BIAS_UNIT:xicalib:ipoly50ua1		current	0n	5.10E-05	20u	75u		
	:I_BIAS_UNIT:xicalib:irext45ua1		current	0n	4.50E-05	5u	50u		
compare info	start	5.00000u	end	250.00000u					
compare options	tollead	0	toltrail	0	0.5	lower	0.2		thre
compare run	5.0000u	250.0000u							
compare start	refDelay	0	testDelay	0					

Fig. 3. Example test plan for wavecompare eqVC.

In order to automate data generation and subsequent post-processing, a convention for directory data organization is established. Repositories are created for test plans, digital and mixed signal simulation results, comparisons, and closure and waiver reports.

Generation of waveform compare scripts using the syntax of the waveform compare tool, EZWave, is automated. The test plan provides the inputs with the tolerances, etc. This approach ensures that the correct waveform comparison criteria is used as specified in the test plan and reduces the effort in creating commands for the waveform tool. The script also generates a set of signals to be probed during simulations, which is added to the corresponding simulation testbench. The behavioral model and analog view is simulated using the same testbench or testbenches. After simulation, the waveform compare script is run for each test and a report (PASS/FAIL) is generated. All the above steps are automated. As soon as any change to the model or circuit is communicated to the verification engineer, the changed model or circuit is replaced and the same steps are re-done and the reports are generated quickly by invoking these script. If the specification is changed, the tests need to be reworked. Figure 4 presents a snapshot of the waveform tool showing differences between the results of analog and behavioral models (marked in red). This helps with visual inspections to fine tune tolerances and during waiver creation. Figure 5 shows an automatically generated report file. Notice the waiver column, which is used to record exceptions and their justifications. As the final step, a review meeting is held and the results across all tests are reviewed and waivers recorded.

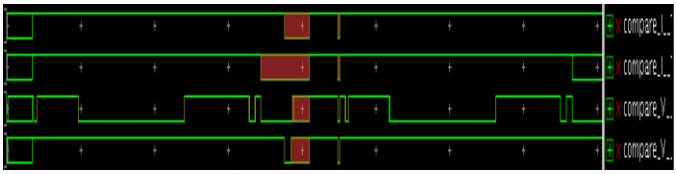

Fig. 4. Ezwave snapshot of waveform mismatch.

#Compact report generated by report_gen.tcl script			
design unit name	BIAS_UNIT		
testname name	BMODE		
user name	hparikh		
date	12/3/2013	Time	11:34:21
verification directory	eq_check_hdmi/bmode		
Analog signal name	Digital signal name	difference	Waiver
I_BIAS_UNIT:compout1v0	I_BIAS_UNIT:compout1v0	2	* time delay effect.OK
I_BIAS_UNIT:vbgout	I_BIAS_UNIT:vbgout	2	* time delay effect.OK
I_BIAS_UNIT:xicalib:ipoly50ua1	I_BIAS_UNIT:xicalib:ipoly50u	1	*acceptable
:I_BIAS_UNIT:xicalib:irext45ua1	:I_BIAS_UNIT:xicalib:irext45u	1	*acceptable

Fig. 5. Report file from wavecompare eqVC.

B. *Assertion-based Verification*

ABV is a well-established verification method in digital verification [4]. In ABV, designers use assertions to capture specific design intent and verify that the design correctly implements that intent. ABV methodologies improve design quality and verification productivity by increasing observability. SystemVerilog Assertions (SVA) are chosen as

the assertion language in this paper because of its design hierarchy agnostic nature. QuestaADMS extends the scope of SVA bind to analog objects; hence the same SVAs can be used on analog as well as digital DUVs.

Setting up ABV requires an upfront effort to write and verify assertions. This approach is applicable to digital-on-top verification and requires proficiency with the SVA language. In this approach, the test plan is an executable verification plan. The test plan is linked to functional checks (assertions) from simulation results to ascertain completeness of the verification task. This flow is compatible with a digital verification methodology and enables the use of the same tool sets as in digital verification.

To enable reuse, a library of commonly used assertions for analog characteristics is developed, including: monotonicity of signal (signal ramp up without glitch), signals crossing a threshold and staying over that threshold for a certain time, and the value of a signal at specified event or time. Some assertions are specific to a given model and have to be created based on the functional spec of that model. Figure 6 shows an example of SVAs for checking monotonicity in a specified time range.

```
property monotony_check_time(sig,starttime,endtime);
real init,slopec;
  @(sig)

 (($realtime>=starttime && !($realtime >=endtime)),init=sig) |=> sig > init;

endproperty


monotonic_reg_out: assert property (monotony_check_time(out_reg,400000,2000000))
```

Fig. 6. An example SVA.

Whereas the same assertion works when the target is signals and corresponding voltage values in analog view, some overhead is added when the target in the analog view is a current value. The current value has to be sampled before being used in assertions.

The test plan describes functions to be verified for a given model and the assertions that are used to verify that specification. Figure 7 shows a test plan for a voltage regulator. The properties are defined in a library file, and an eqVC module is instantiated to the testbench consisting of SV bind constructs for the specified checks in the test plan. The same eqVC module is used during simulation of analog or behavioral models of the DUV. The results of the simulation are saved in a universal coverage database (UCDB) /UCIS [5] format for each test. As in the wave compare method, data is organized in repositories. Each test is run twice: once using a behavioral model and another time using a SPICE view of the DUV. After tests are simulated, the UCDBs of individual tests are merged together and cross-linked to the test plan.

#	Title	Description	Link	Type
1	Analog Properties	Analog Properties		
1.1	monotonicity	The out_reg signal is monotonic at ramup	monotonic_reg_out	Assertion
1.2	slope_reg	rate of rise of out_reg	slope_reg_out	Assertion
1.3	drop_out_condition_chec	input voltage shouldn't go below dropout range (vin >	drop_out_check	Assertion
1.4	out_volt_at_trim1	value of voltage at given trim1	voltage_at_trim0011	Assertion
1.5	out_volt_at_trim2	value of voltage at given trim2	voltage_at_trim1001	Assertion
2	Logic properties	Logic Properties		
2.1	pd_check	check that enable is 0 when pd is asserted	en_in_pd	Assertion
2.2	supply_stable	supply is stable before enable	supp_stable	Assertion

Fig.7. Test plan for ABV eqVC.

The cross linking of the test plan with the simulation results is helpful in tracking the status of validation activity, as it shows a quantitative measure of the specification checks that were implemented and that passed. This process is automated via simple scripts using a command in QuestaADMS for UCDB merging and test plan linking. Figure 8a shows the status after merging assertions from behavioral model simulation, and Figure 8b shows the corresponding results for a SPICE DUV. In this case, the SPICE DUV shows 100% coverage, which means that all the checks (assertions) passed for the tests, while behavioral model simulations for the same tests and checks achieved 70% coverage, meaning there are some checks that failed in the behavioral model. The differences between behavioral and SPICE models are either due to different interpretations of the specification between the analog designer and the behavioral model creator or an error in implementation. This method gives quantitative and traceable metrics to validate equivalence. PASS/FAIL reports are generated using commands available in the simulation tool. A review meeting at the end of this activity looks at the report and any exceptions are recorded. It is a good practice to break up tests into many simpler tests, rather than verify many assertion conditions in one test. This way, many tests can be run in parallel and simulation time for SPICE DUVs is reduced. The ABV method is deployed for analog calibration module(s) of USB IP.

Verification Management Tracker

Sec#	Testplan Section / Coverage Link	Type	Coverage	Goal	% of Goal	Status
0	testplan	Testplan	70%	-	70%	
1	Analog Properties	Testplan	40%	100%	40%	
1.1	monotonicity	Testplan	0%	100%	0%	
1.2	slope_reg	Testplan	0%	100%	0%	
1.3	drop_out_condition_check	Testplan	0%	100%	0%	
1.4	out_volt_at_trim1	Testplan	100%	100%	100%	
1.5	out_volt_at_trim2	Testplan	100%	100%	100%	
2	Logic properties	Testplan	100%	-	100%	
2.1	pd_check	Testplan	100%	100%	100%	
2.2	supply_stable	Testplan	100%	100%	100%	

Fig. 8a. Merging behavioral model DUV results with test plan.

Sec#	Testplan Section / Coverage Link	Type	Coverage	Goal	% of Goal	Status
0	testplan	Testplan	100%	-	100%	
1	Analog Properties	Testplan	100%	100%	100%	
1.1	monotonicity	Testplan	100%	100%	100%	
1.2	slope_reg	Testplan	100%	100%	100%	
1.3	drop_out_condition_check	Testplan	100%	100%	100%	
1.4	out_volt_at_trim1	Testplan	100%	100%	100%	
1.5	out_volt_at_trim2	Testplan	100%	100%	100%	
2	Logic properties	Testplan	100%	-	100%	
2.1	pd_check	Testplan	100%	100%	100%	
2.2	supply_stable	Testplan	100%	100%	100%	

Fig. 8b. Merging SPICE DUV results with test plan.

C. Advanced Verification

Advanced verification method of equivalence validation is based on UVM [6], a standardized methodology used widely in digital verification. The UVM class library brings automation to the SystemVerilog language. UVM is an Accellera standard with support from multiple vendors. Major features offered by UVM are constrained random stimulus and coverage-driven verification based on reusable class libraries. UVM is used to extend the ABV methodology to include randomized stimuli and implement coverage driven verification.

The extension of UVM for mixed-signal simulation and verification of analog behavioral models is reported in literature [7],[8]. Setting up UVM-based verification requires advanced knowledge of the SV language and the initial set up effort is high. However, its use enables randomization of stimuli and increases verification coverage, thus UVM agents are extended to include analog specifications to increase test scenarios and measure coverage. eqVC in this case includes UVM monitor agent and assertions. An example of generating constrained random stimulus for real signals is shown in Figure 9. Integer parameters are constrained and then divided by a resolution parameter to generate randomized real numbers. Coverpoints are defined on the integer bins, as shown in Figure 10. A list of parameter values or range and their cross values is used in covergroups to generate stimulus for equivalence validation testbenches. In this method assertions developed in ABV method are re-used.

```
class my_transaction extends uvm_sequence_item;

  `uvm_object_utils(my_transaction)

       randc int load;
       randc int vref;


      //list
       constraint constraints_ld { load inside {250, 500, 1000};}
      //range
      constraint constraints_vr { vref >= 2; vref < 8;}
      real load_real;
      real vref_real;

      /randomize real signals
      function void post_randomize();
      int2real();
      endfunction

      function void int2real();
      load_real = real'(load)/10.0;
      vref_real = real'(vref)/4.0;
      endfunction
```

Fig. 9. Generating constraint random real stimlus.

We used UVM based eqVCs in a pilot project to explore feasibility and effort. While constrained random stimulus increase verification coverage, simulation performance when using SPICE DUV needs attention. Optimizing test sequences and using behavioral models of some design blocks in the SPICE DUVs improves simulation performance. The following types of behavioral models are being considered for UVM-based eqVC method:

1) *Behavioral models that capture complex analog functionality with many interdependent signals/states would benefit from randomized stimlus.*

2) *Top-level behavioral models that are created by connecting other behavioral models.*

Name	Coverage	Goal
:my_pkg:coverage_collector		
TYPE voltage_range_load	100.0%	100
CVP voltage_range_load::reg_supply_ra...	100.0%	100
CVP voltage_range_load::load_range	100.0%	100
CROSS voltage_range_load::cros_c	100.0%	100

```
class coverage_collector extends uvm_subscriber #(my_transaction);

  `uvm_component_utils(coverage_collector)

        my_transaction txn;

        integer vref;
        integer load;

        covergroup voltage_range_load;

        reg_supply_range: coverpoint vref
         { bins supp_bin = {4,5,6}; }


      load_range: coverpoint load
     { bins load_range = {25,500,1000}; }


  cros_c : cross reg_supply_range,load_range ;
  endgroup
```

Fig. 10. Coverage for randomized parameters.

The verification environment remains identical to ABV method. The same tools/languages are used for the test plan, property checking (SVA library), coverage collection, and report generation.

IV. Results and Summary

This validation environment has been successfully used by the high-speed serial links verification group at STMicroelectronics for HDMI, USB, and DPHY IPs. Use of a verification planning based methodology has significantly reduced the equivalence validation effort of behavioral models. High level of automation and reuse have enabled the rapid use of this methodology across various IPs. During these projects, we have used this validation environment to discover and fix multiple functional, connectivity, and test mode differences between behavioral models and circuits. The behavioral models are delivered across multiple SoC teams and are in use for functional verification.

Our digital, analog, and verification teams are collaborating following the framework mentioned in Section A. A mix of eqVC types described in Section B has been used in these projects depending on specifications of model to be validated and skillset of person responsible. For simple models (usually leaf level models), the most commonly used eqVC is waveform compare. ABV eqVC is the preferred method on complex models as well as for top hierarchy, which consists of interconnected behavioral models. In cases where it is difficult to write property checkers that would work for both behavioral models and SPICE DUV, a wave compare eqVC is set up in addition to ABV.

REFERENCES

[1] Serge Garcia Sabiro, Event-Driven (RN) Modeling for AMS Circuits,*Forum on specification & Design Languages 2014.*

[2] Questa ADMS Reference Manual, Mentor Graphics

[3] EZWave Reference Manual, Mentor Graphics

[4]AssertionBasedVerification, *https://verificationacademy.com/courses/assertion-based-verification*

[5] Unified Coverage Interoperability Standard, *http://www.accellera.org/activities/committees/ucis*

[6] Standard Universal Verification Methodology, *http://www.accellera.org/activities/committees/uvm*

[7] Harry Wang, Wessam El-Naji, Kenneth M. Bakalar, Experience with OVM and Mixed-Signal Verification of an Impedance Calibration Block for a DDR Interface, *DVCon 2012*

[8] Neyaz Khan, Yaron Kashai, From Spec to Verification Closure: a case study of applying UVM-MS for first pass success to a complex Mixed-Signal SoC design, *DVCon2012*