Equivalence Validation of Analog Behavioral Models

Manish Kumar Karna*, Hardik Parekh*, Mohit Jain*, Atul Pandey+, Sandeep Mittal++
*ST Microelectronics Pvt. Ltd., Greater Noida,India {manish.karna,hardik.parekh,mohit-jain.crd}@st.com
+Mentor Graphics,Munich,Germany,++Mentor Graphics,Noida,India {atul_pandey,sandeep_mittal}@mentor.com

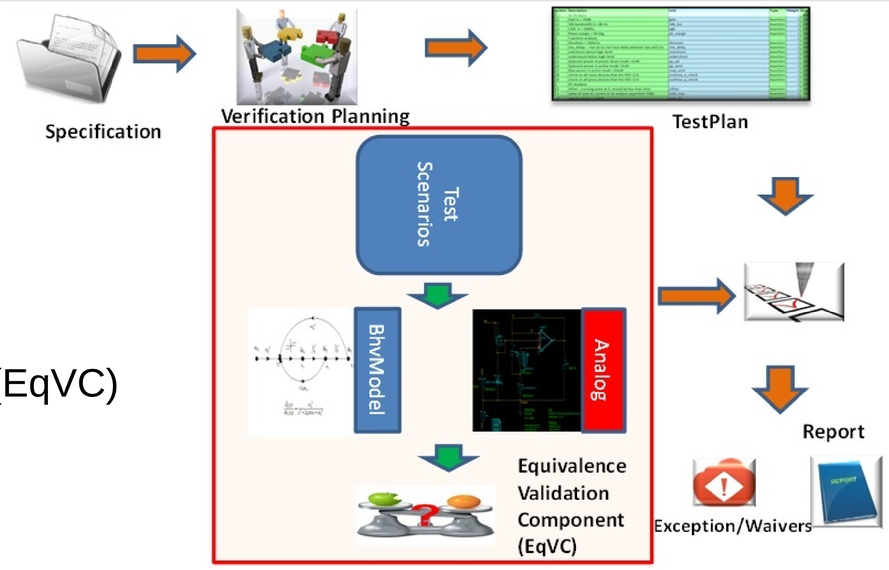
Overview

- Behavioral Models are used in verification to increase coverage on mixed signal SoCs.
- Behavioral abstract models capture functional features of analog behavior in event-driven HDL languages.
- Behavioral Models are orders of magnitude faster than simulating SPICE view of analog IP
- Critical to ensure equivalence between behavioral models and implementation (SPICE)

Framework components

- Specification document for models
- Verification Plan
- TestPlan
- Verification environment
- Equivalence validation component(EqVC)
- Equivalence check Report
- Exception Waivers

Analog behavioral model validation Environment



Verification Planning and TestPlan

Verification Plan document contains:

- List of features of the analog behavioral model to be validated.
- A list of test to verify those features.
- To test each feature, a list of checks (coverage) are specified in **TestPlan**.
- Equivalence Validation method used and data organization.
- Criterion for validation closure.
- Method of Review, Reporting and Exception handling.

Equivalence Validation Components

- Simulation environment consist of Testbench,DUV and equivalence components (EqVC)
- Same test is applied to Model and netlist DUV
- EqVC chosen according to skillset and effort

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

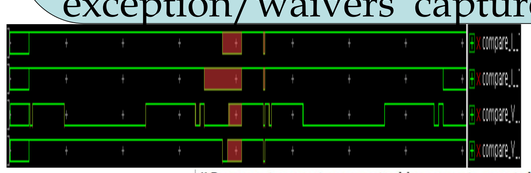
Figure 2: Equivalence validation component (EqVC) types

Verification closure, Report and Review

- Results generated from simulations for all tests and captured in eqVCs are compared to TestPlan to generate PASS/FAIL report
- The difference arise from electrical effects inbuilt in SPICE views and abstracted in behavioral model
- Exception/Mismatches are reviewed and “Waiver” issued and recorded in closure report

Waveform comparison method

- ❖Continuous waveform comparison
- ❖Quick setup time
- ❖Works for both AoT and DoT verification
- ❖TestPlan specifies signals to compare and tolerances
- ❖ Report generated and exception/waivers captured



TestPlan

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						

Report

#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

Assertion based verification method

- ❖A Library of SystemVerilog Assertions
- ❖Assertions work on both Model and netlist
- ❖Expertise in assertions language and ABV required

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

```
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))
```

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%	

Advanced Verification

- ❖ UVM constrained random stimulus for analog properties.
- ❖ Increased verification coverage
- ❖ Compatible with ABV

```
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
```

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
```