2019 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION UNITED STATES

SCALABLE, RE-USABLE UVM DMS AMS BASED VERIFICATION METHODOLOGY FOR MIXED-SIGNAL SOCS

Nilesh Sonara
Broadcom Limited
San Diego CA
neel.sonara@broadcom.com

Noorulla Mohammad
Broadcom Limited
San Diego CA
noorulla.mohammad@broadcom.com

Poonam Singh
Broadcom Limited
Santa Clara CA
poonam.singh@broadcom.com

David Stoops
Broadcom Limited
Irvine CA
david.stoops@broadcom.com

Joseph Fernando
Broadcom Limited
Irvine CA
joseph.fernando@broadcom.com

Kartik Sudarshana
Broadcom Limited
Santa Clara CA
kartik.sudarshana@broadcom.com

Abstract- **The traditional approach of separate analog and digital verification of current complex mixed-signal SoCs is not sufficient to achieve high quality silicon within given time and resources. In this paper full chip mixed-signal coverage driven constraint random verification environment based on Universal Verification Methodology (UVM) is introduced which supports different abstraction of analog functionality from systemverilog user defined nettype (SV UDN) to Verilog AMS (VAMS) to transistor model and any combination of these. We explain how a reusable, scalable approach helped to use the best of both analog and digital verification expertise to find issues early and improve overall productivity.**

Keywords—UVM (Universal Verification methodology) ; AMS (Analog Mixed Signal); DMS (Digital Mixed Signal); VAMS(Verilog AMS); SV (Systemverilog); UDN (User Defined Nettype); mixed-signal verification; analog assertion;

I. INTRODUCTION

There is no doubt that mixed-signal applications are among the fastest growing market segment in semiconductor industry. From mobile phones to making payments online on a phone, controlling home using voice and gestures, autonomous cars, etc. consumers expects electronics to do more and more. As process node shrinks, SoC architects are adding more analog and mixed-signals blocks, along with more and more digital and embedded software based controls to monitor, react to and regulate analog functionality dynamically in real time. Most of the mixed-signal SoCs contain multiple feedback loops and complex interactions between analog and digital blocks and embedded software. This growing complexity poses several challenges and uncertainty for SoC and system level analog mixed-signal verification.

Analog designers are good at simulating small blocks but lack the ability to create complex verification scenarios at the SoC level. At most, they use simple Verilog based verification to create few basic tests at SoC level. Digital verification engineers are very good at creating and checking complex scenarios at the SoC level but lack the knowledge of then analog domain for debug and analysis. Many times simple behavioral models are used for analog functionality and they are assumed to be pre-verified. As more complexities are added to mixed-signal SoCs, traditional separate verification by analog and digital teams is not efficient or sufficient to achieve bug free silicon at first pass within give time and resources constraints.

UVM is the de facto industry standard for digital verification [1]. UVM based digital verification techniques have been used for mixed-signal verification [2], [3], [4] using VAMS models. However, lots of digital mixed-signal (DMS) verification is done using UVM at the behavioral model level but very little is reused for VAMS and transistor model based verification at SoC level. This paper presents a solution used by our team to overcome different mixed-signal verification challenges enabling reuse and scalability. It describes UVM based constraint random verification environment and its independence from mixed-signal design for reusability. It describes SV UDN and UDR function based analog behavior modelling and its advantages in finding analog bugs early in design

cycle. Later it describes how SV UDN models are replaced with VAMS and transistor models and a mix of both using different configurations. It also explains how we achieved higher productivity and were able create more scenarios and selectively run more meaningful VAMS and mixed model simulations in less time.

II. UVM VERIFICATION ENVIRONMENT

Constraint random coverage driven UVM based verification environment was created keeping mixed-signal verification in mind. Universal Verification Component (UVC) is reusable, pre-verified, configurable, plug-and-play verification component for a particular protocol interface or logic function, and compatible with the UVM. As shown in figure 1 interface UVCs were connected to the chip pins only, making it independent of the analog model used. The main purpose of these UVCs is to program registers in the design, send some data to the host, send or receive triggers on general purpose IOs. Clock and reset UVC is used for supplying configurable clocks and resets needed by the SoC and Scoreboard, which collects transactions from UVC monitors, other monitors in the digital domain and perform needed checking. Virtual sequencer coordinates the sequences of commands and data going on different UVCs. A UVM register package is used to access all registers using interface UVCs. Analog UVC is a waveform generator, which can generate sine, triangular, trapezoidal and square with different frequency, amplitude and active time. A Board module contains external passive elements needed to emulate analog loading effects. It is configurable to select different values, supports SV behavioral and VAMS modelling which can be selected by config views.

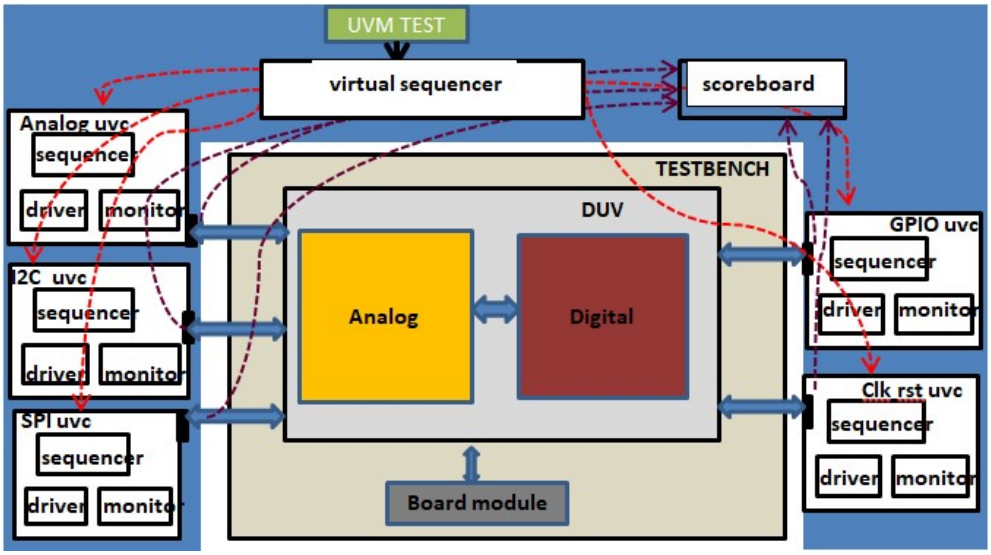


Figure 1: UVM AMS verification environment

III. TESTBENCH CONFIGURATIONS

Different testbench configurations were used to serve different purposes keeping reuse of complete UVM environment in mind. SV UDN models are completely run on event driven simulator and, while less accurate but still provide realistic behavior. VAMS models are run with SPICE like simulator and event driven simulators in parallel but only interact at analog digital signal boundary. Transistor level analog models, like SPICE netlist are very accurate but simulation effort increases greatly with every increase in circuit nodes, as it is evaluated for current and voltage at every time step in the simulation process. They are very slow when compared to event driven digital simulations.

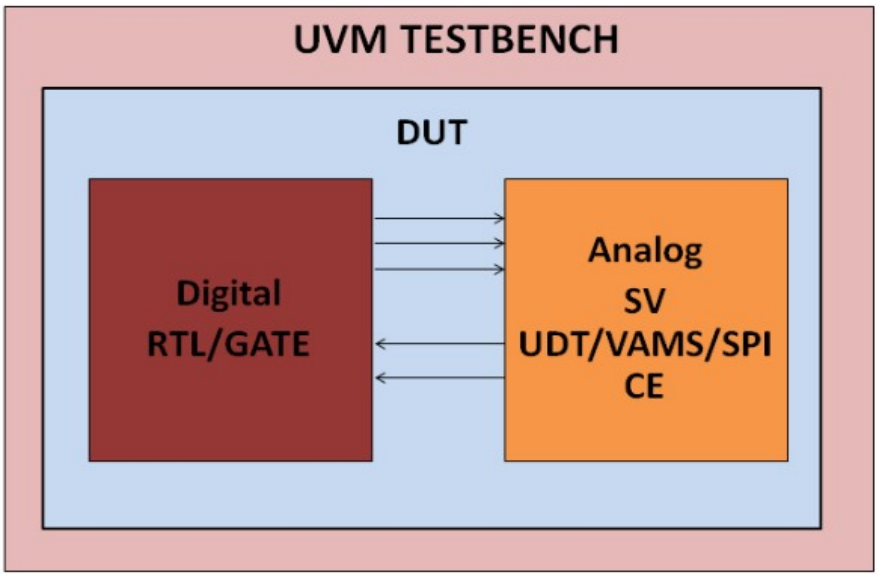


Figure 2: Scalable testbench configurations in same UVM verification environment

Figure 2 shows how the same verification environment and testbench are reused for different analog abstractions. Engineers can select SV UDN, VAMS, SPICE or mix of any of those for analog behavior and run selected test. An interface element file is created having information for different voltage domain and discipline. The simulator automatically inserts virtual elements for logic-to-real (L2R) and real-to-logic (R2L) conversion for SV UDN models.

A. SV UDN DMS MODEL

A UVM based environment provides the benefit of having large number of complex tests, driving the need for systemverilog (SV) behavioral models that can accurately represent functional and transient behavior of the transistor-level analog blocks. While real number modeling improves performance with moderate accuracy [5], it does not allow multi value nets and multiple drivers. Systemverilog user defined nettypes (SV UDN) and user defined resolution function allow easier yet accurate modelling of loading effects which improves simulation performance [6]. It has enabled datatypes with multiple elements like (V, I, R) transmitting and receiving information bi-directionally with close enough resemblance to actual analog circuits. An nettype with resolution function overcomes most of the limitations of SV RNM models like having multiple drivers, support bi-directional wreal port, take average of current at a node, min/max of voltage, support resolution function etc. There are also built in Nettypes like wrealmax, wrealmin, wrealavg, from EDA vendors which helps with modeling. The full syntax, rules & advantages of nettypes are beyond the scope of this paper. Below is the sample for definition of user defined type T and its resolution function which adds field1 for each driver. It is used for adding currents coming to a net in the circuit.

```
typedef struct{
real field1; field2; field3;
} T

function automatic T Tsum(input T driver[]);
Tsum.field1 = 0.0;
foreach (driver[i])
   Tsum.field1 += driver[i].field1;
endfunction
```

nettype T wTsum with Tsum;

Coercion technique, enables us to share same net across multiple domains like wreal, electrical or logical. The EDA tool uses information from an interface element (ie) file to resolve the type of the net based on the destination. The interface element has information about Real2Logic(R2L) or Logic2Real(L2R) mapping. The simulator automatically inserts virtual elements during elaboration to connect multiple domains shared by the same net.

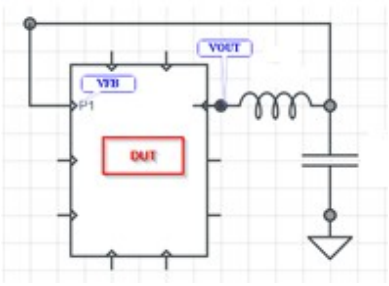
Ex: AMS Control File (ie)

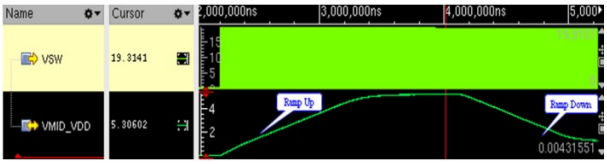
```
amsd{
  ie vsup=1.8 rhi=0.1 rlo=0.1 rout=0.1
  ie vsup=1.1 instport="`top.vdd1p1"
}
```

SV UDN behavioral models were generated bottom-up by building individual blocks while top-level connections between blocks were automatically created from schematic. This helps in finding schematic issues with digital simulation. SV models developed with this approach also have supply and ground nets, which make it suitable for power aware simulations. SV UDN model based simulations uncovered issues in complex feedback loop interactions between analog and digital circuitry. 100% code and functional coverage were achieved using this approach.

Analog – Digital Feedback loop example using Dynamic measurements using SV UDN based model

Feedback loops are the very critical and complex logic to verify in the mixed signal simulations. As it needs the continuous monitoring of the generated voltage and continuously feedback control, it becomes critical for the verification process to check for every detail of the regulatory path. Once the required target voltage has been programmed, the digital controller will start generating the voltage codes and send it to analog circuit. These codes are converted into analog by DAC and it further sent through the filters and PWM generator. The LCR filter in the bench filters and creates a clean ramp voltage. The resultant output voltage is feedback to the ADC which is compared against the reference voltage. If the generated output voltage is more than the reference voltage an over voltage indication is provided to the digital controller which is used as an interrupt to the CPU. If the resultant output voltage does not reach the reference voltage, the digital controller will further add step value to the current voltage code and send it to the analog DAC. The loop continuous till the voltage reaches to the reference voltage. The Analog UVC is the key component in the verification bench which controls the test parameters of analog circuits and also it provided required tasks to generate the over voltage and under voltage scenarios. The critical checks were performed during the ramp up and ramp down of the voltage. Below two diagrams show a typical feedback loop and the results of switching voltage ramp Up and ramp down during voltage loop operation. Ramp up is from 0V to 5V and ramp down was till 1.2 V. The filter parameters are used to control the slew rate of the output and change rate at any point of interval. System Verilog analog slew checker and voltage checker were added on the feedback node to continuously monitor the generated output voltage





Checkers were implemented to check any overflow in the bit vectors in the digital controller.. All the internal nodes activity during shut down process is monitored by analog assertions. The scoreboard checks are implemented by creating an equivalent C model of digital controller and checks were added to compare the generated codes by design against the golden codes generated by C model

B. VERILOG AMS MODEL

One of the main goal in AMS modeling should be convergence of the simulator. If the simulator will not converge to a solution, then even the most accurate model is not helpful. VAMS models are much slower than SV UDN models and require analog simulator to run in parallel with event-based simulator. VAMS model development is started top-down from schematic with reusing some of the components from SV UDN model. Accuracy of VAMS model is very close to transistor models. To reuse the verification work done for SV UDN model, VAMS models and board components with VAMS elements are compiled replacing the SV model in simulation. Board elements add external passive elements, which are needed to avoid any convergence or stability issues. Regression script takes care of this with an extra switch added to the run the same simulation with a VAMS configuration.

C. TRANSISTOR MODEL

Transistor models based on SPICE netlist are extremely slow but model analog circuitry very accurately. Spice netlist is also compiled in this flow and config view is used to select SPICE netlists for analog blocks for transistor-level simulations. The Same board components are reused from VAMS configuration. Below is the example for config used for full spice. Verification environments stays same as it was used with SV model reusing every test.

```
// amsd portmap module
simulator lang=spectre

amsd {
//Default/Fallback discipline
ie vsup=(1.5) rhi=(0.1) rlo=(0.1) rin=(20M) rout=(0.1) vdelta=(1.5/64.0)
ie vsup=(1.8) rhi=(0.1) rlo=(0.1) rin=(20M) instport="sys.I_DUT.VDDO"

portmap subckt=ANATOP porttype=name
config cell=ANATOP use=spice

}
```

D. MIXED-MODE MODEL

SPICE netlist based simulations are very accurate but keeping all analog blocks in SPICE for many tests is not possible as simulations can run for days. Simulations using VAMS models run faster than SPICE models but are not as accurate as SPICE models. In mixed-mode configurations, any analog block instance or any module can be configured to select VAMS, SPICE or SV UDN views. SV UDN is selected as a default view. The mixed mode approach helps in improving simulation speed while keeping the accuracy of the SPICE models where it is required. The analog team, based on their expertise, decides different configurations for analog blocks for simulations. Each configuration was created with target functionality in mind. The digital team, based on their expertise, creates tests scenarios, which exercise the entire SoC, including analog blocks and their feedback to the digital design. Figure 3 shows some of the different configurations used.

In mixed-mode configurations, any analog block instance or any module can be configured to select VAMS, SPICE or SV UDN views. SV UDN is selected as a default view. Figure 3 shows some of the different configurations used.

CONFIG	TX	BIAS	RX	PMU	MISC
CFG0	SV UDT	SV UDT	SV UDT	SV UDT	SV UDT
CFG1	VAMS	VAMS	VAMS	VAMS	SV UDT
CFG2	VAMS	VAMS	VAMS	VAMS	VAMS
CFG3	SPICE	VAMS	VAMS	VAMS	SV UDT
CFG4	SPICE	VAMS	VAMS	VAMS	VAMS
CFG5	VAMS	VAMS	SPICE	VAMS	VAMS
CFG6	VAMS	SPICE	VAMS	SPICE	VAMS
CFG7	SPICE	SPICE	SPICE	SPICE	SPICE

Figure 3: configurations selecting analog blocks as VAMS or SPICE model

Tests that exercised a specific analog path used SPICE views for that particular path. For example, tests that verified the TX path would use CFG3 view. Power up, power down, brownout and connectivity tests were run as all SPICE simulations, CFG7. Below is example for mixed config where few modules in spice netlist are replaced with VAMS models.

```
simulator lang=spectre

amsd {
    //Default/Fallback discipline
    ie vsup=(1.5) rhi=(0.1) rlo=(0.1) rin=(20M) rout=(0.1) vdelta=(1.5/64.0)
    ie vsup=(1.8) rhi=(0.1) rlo=(0.1) rin=(20M) instport="sys.I_DUT.VDDO"

     portmap subckt=ANATOP porttype=name
     config cell=ANATOP use=spice

     portmap module=USB_CPD_top porttype=name
     config cell=USB_CPD_top use=hdl

     portmap module=RCO_top porttype=name
     config cell=RCO_top use=hdl

     portmap module=ADC12_top porttype=name
     config cell=ADC12_top use=hdl
    }
```

IV. ASSERTIONS AND COVERAGE

Systemverilog assertions were added to the analog to digital boundary, which was reused for all configurations. These assertions check analog to digital timing checks, protocol checks and glitch detection. Additional analog assertions using systemverilog were added at internal nodes to continuously monitor activity in the analog circuitry. There assertions include

- Limit checker: Checks that analog signal remains above or below a given threshold
- Slew checker: Checks that analog signal rises/falls with a given slew rate(+/- tolerance)
- Range checker: Checks that analog signal remains within the given high and low threshold
- Frequency checker: Checks that the analog signal frequency is within a given tolerance.

The Below sample code for range checker assertion shows that analog nodes inside analog circuitry can be probed by system tasks provided by the simulator and its value can be checked against the range. An error is flagged when the values are out of range.

```
bit node_out_of_range;
always @(posedge tb_clk) begin
 if($<sim>_analog_exists(signal_path) != 1)
    `uvm_fatal("AMS: " ,
       $sformatf("<sim>_analog_exists() failed for : %s", signal_path))
  else begin
    voltage_val = $<sim>_get_analog_value(signal_path);
    if(!((voltage_val > Vth_lo) & (voltage_val < Vth_high)))
       node_out_of_range = 1;
  end
end
property p_node_out_of_range;
   disable iff(reset)   $rose(node_out_of_range) |-> (reset_assert);
endproperty
a_node_out_of_range: assert property(p_node_out_of_range)   else
$error("When Node voltage out of range happened ,  reset is not set")
```

Digital functional coverage is discrete and mostly represents range or unique values while analog are continuous and unbounded range. One simple solution is to discretize the analog range keeping accuracy in mind. Systemverilog standard does not offer any real valued coverage but it can be solved by converting real number into bit vector or integer and multiplying it with 10's power for necessary resolution. Some simulators support real value cover points and it can be coded as below in the example.

```
covergroup ldo_sw_cov;
            option.per_instance = 1;
            option.name = "ldo_sw_cov";
            type_option.real_interval = 0.1;
  cp_ldo_sw: coverpoint I_vr_if.SW {
    option.comment = "LDO SW O/P voltage range ";
              bins vol_0[] =  {[0.0:0.9]};
              bins vol_1[] =  {[1.0:1.9]};
              bins vol_2[] =  {[2.0:2.9]};
              bins vol_3[] =  {[3.0:3.9]};
  }
endgroup
```

Below code shows how new integer variable can be declared and real value can be converted to integer. New integer variable value can be updated and its coverage can be collected to reflect real variable coverage.

```
covergroup ldo_sw_cov;
        option.per_instance = 1;
        option.name = "ldo_sw_cov";
 cp_ldo_sw: coverpoint SW_final {
   option.comment = "SW O/P voltage range ";
          bins vol_0[] = {[0:9]};
          bins vol_1[] = {[10:19]};
          bins vol_2[] = {[20:29]};
          bins vol_3[] = {[30:39]};
 }
endgroup
integer SW_final;
SW_final = I_mvr_if.SW * 10;
ldo_sw_cov.sample();
```

100% toggle coverage for the analog-digital boundary was also achieved in each configuration separately to make sure analog-digital interface is exercised in every model.

V. UPF AND GATE SIMULATIONS

As SV UDN model is created from schematic it contains supply pins, level shifters etc. and it is power aware, all outputs are gated with supply and ground voltage values. Therefore, it was possible to carry out low power simulations using UPF flow mentioned as future work in [2]. The same low power tests are reused with power- aware gate level simulations using SV UDN model. It was also possible to run power aware gate level simulation with all analog blocks as SPICE models though it is not recommended due to very long simulation time and large log file size.

VI. REGRESSIONS

All the tests were initially developed with SV model and nightly regressions were run. Separate AMS verification plan was created and different configs were defined with each having information about which modules is used as SPICE or VAMS. Config creation was done keeping functional scenarios in mind and modules it would exercise. All VAMS models simulation took around 10 times more cpu time while ALL spice simulations took 50-100 more cpu time than time taken byVAMS simulation. These numbers can vary widely depending on the configuration and how VAMS models are coded. Weekly regressions were run for VAMS models and spice simulations were run as needed.

VII. SIMULATION DATA

The above chart in figure 4 shows issues found in different parts of design, testbench and spec.

- 46% issues reported were digital design issues. More than 98% of those were found using SV UDN model based simulations. More than half of issues reported were found from scenarios having analog and digital interactions.
- 14% of issues reported were analog circuit implementation issues. More than 50% of those issues were found using SV UDN based simulation. Issues included connectivity issues, polarity issues, test mode and mux selection issues. 35% of those were reported in VAMS simulations which included the remaining issues which were found using SPICE simulations
- 23% and 2% issues were reported for SV UDN modelling and VAMS modelling issues respectively.
- 6% issues reported were spec related. Out of that 10% were found using SV UDN model.
- 9% issues reported were verification environment and testbench related.

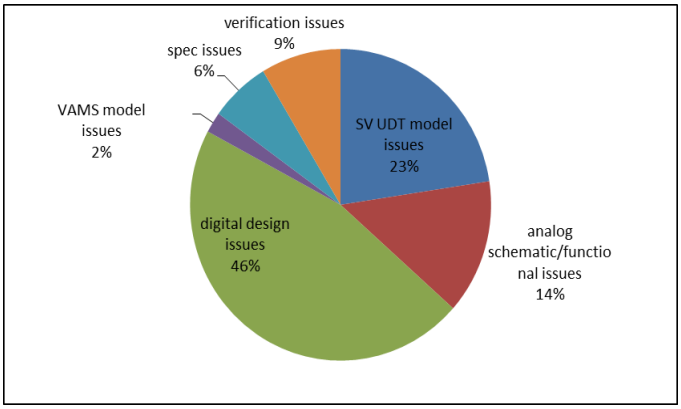


Figure 4: simulation bugs found and its distribution

VIII. CONCLUSION

A Unified reusable scalable UVM based constrained random coverage driven verification environment with configurable analog model abstractions was presented in this paper. With the help of realistic behavior of SV UDN model, many bugs were discovered in analog digital communication. It also helped in finding analog bugs in schematic. Analog designers did not have to worry about tests creation and they were able to run more simulations with VAMS and mixed mode models by reusing tests from digital team. They were also able to spend more time on analysis and debug. All scoreboards, assertions, checkers and coverage were reused for VAMS and mixed mode simulations. Coverage driven methodology helped in selecting tests to run at VAMS and mixed mode simulations.

REFERENCES

[1] Universal Verification Methodology (UVM) 1.2 User's Guide http://www.accellera.org/images//downloads/standards/uvm/uvm_users_guide_1.2.pdf

[2] C. Liang, Zhou Fang and C. Z. Chen, "Method for analog-mixed-signal design verification and model calibration," *2015 China Semiconductor Technology International Conference*, Shanghai, 2015, pp. 1-4.

[3] S. Simon, D. Bhat, A. Rath, J. Kirscher and L. Maurer, "Coverage-driven mixed-signal verification of smart power ICs in a UVM environment," *2017 22nd IEEE European Test Symposium (ETS)*, Limassol, 2017, pp. 1-6.

[4] C. Liang, G. Zhong, S. Huang and B. Xia, "UVM-AMS based sub-system verification of wireless power receiver SoC," 2014 12th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT), Guilin, 2014, pp. 1-3.

[5] D. Juneja, "On Event Driven Modeling of Continuous Time Systems," 2015 28th International Conference on VLSI Design, Bangalore, 2015, pp. 198-203.

[6] Shera and C. Wegener, "Buck converter modeling in Systemverilog for verification and virtual test applications," *2015 IEEE 20th International Mixed-Signals Testing Workshop (IMSTW)*, Paris, 2015, pp. 1-6.