2022 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION INDIA

# Harnessing SV-RNM Based Modelling and Simulation Methodology for Verifying a Complex PMIC designed for SSD Applications

Vijay Kumar, Shrikant Pattar, Yaswanth Chebrolu, Vinayak Hegde
Samsung Semiconductor India Research

***Abstract*-The objective of this paper is to present a Mixed-Signal Verification (MSV) flow to verify a PMIC designed for SSD applications. The PMIC consists of analog-centric blocks such as buck converters, on-chip voltage regulator, high frequency oscillator, etc. The challenges encountered during verification were three-fold, viz., (a) Simulation performance (runtime), (b) Checking the default values loaded onto the hundreds of register banks and (c) Achieving coverage that mandates writing onto register banks to verify the buck voltage regulation values. The authors have framed a three faceted verification methodology to address these challenges. Use of Real Number Modelling (RNM) using SystemVerilog (SV) "real" built-in nettype leveraging SV IEEE 1800-2009 features, automation of the default values check across registers using Python scripts, and randomization of register write from I2C to check the buck voltage regulation values. The results obtained have been quantified to show a performance improvement of the order of 100-1000x than schematic-based simulations.**

## I. INTRODUCTION

Power Management Integrated Circuits (PMICs) are becoming more pronounced these days owing to their inherent capabilities to perform DC-DC conversion as in Buck Converters, Dynamic Voltage Scaling (DVS) and ability to transition through various modes such as active, standby and sleep modes [1][2]. Depending on their end application, PMICs could be broadly classified as System-on-Chip (SoC) PMIC, Display PMIC, Memory PMIC or Interface PMIC, though there could be few other application-specific ones. Each of these categories of PMIC comes with its own challenges, both during the design and the verification phases, owing to their complexity and constraints.

A Memory PMIC, which is typically used for Solid State Drives (SSD) applications usually consists of one or more synchronous type buck converters, current limiting circuits, housekeeping blocks, and voltage level detectors inside the analog part of the chip. In addition, they contain on-chip voltage regulators (LDO), High-Frequency Oscillators (HFO), POR and reference level detector, and ESD protection circuits. There is also the core analog block that is responsible for generating the reference and bias voltages for all the other blocks. The output voltage of each buck converter provides DVS, forced discharge mode, power good (PG) function, as well as power-off using I2C interface. Additional features include default voltage setting mode, through which PMIC can support to change default voltage of each output up to four combinations during power on. The PMIC has been designed to use very small size inductors with multilayer chip type.

Main Features of the PMIC include:

- Independent PMIC reset, viz., Making all buck converter channel outputs return to the default voltages according to the external address setting at the same time without PMIC power off/on
- Control from I2C interface for a host of operations such as
    - Pull down mode to discharge all channels forcedly
    - Adjustable discharge resistance
    - DVS and disabling of each channel output

- One Time Programmable Start up Delay Time, RSTO Delay Time and RSTO threshold
- Fully integrated MOSFET switches
- Thermal shutdown, over voltage (external/output) and over current protection capabilities

A block diagram representation of the architecture of a Memory PMIC is shown below.

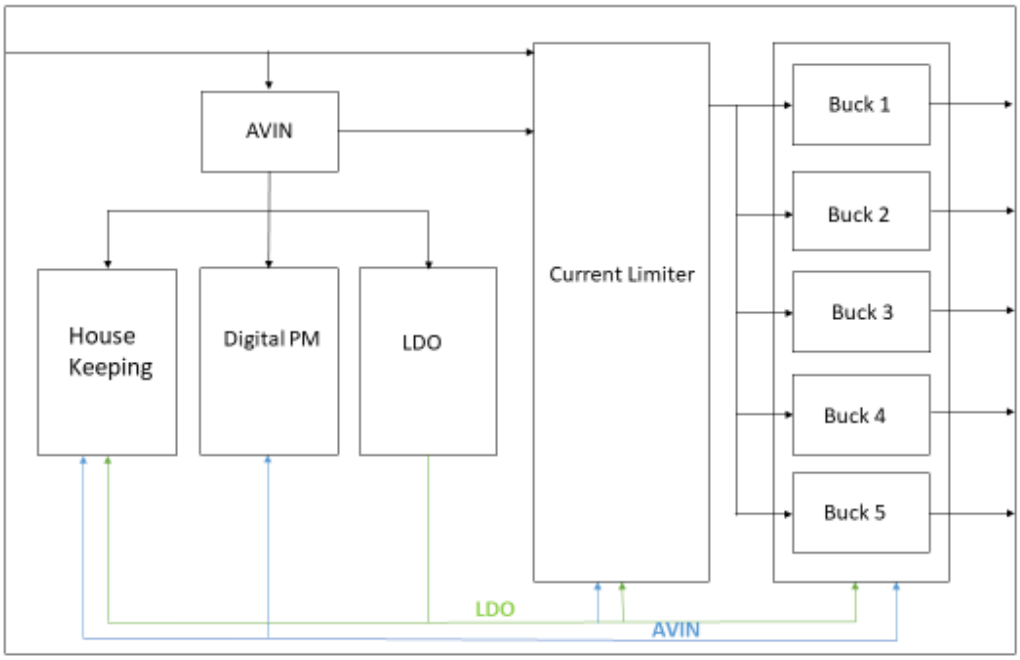


Figure 1. Block diagram of a Memory PMIC architecture

## II. SEQUENCES AND MODES OF OPERATION

When the external input voltage $V_{CL,Vin}$ supplied to the input of the current limiter exceeds its Under Voltage Lock Out (UVLO) level, the output of the current limiter $V_{CL,Vout}$ drives the 5 buck converter channels. Power sequence starts after $V_{CL,Vout}$ exceeds the UVLO threshold, with a delay time of $T_{sys}$. During this $T_{sys}$ delay time, PMIC can forcedly work in discharge mode for all buck converter channels and keeps each channel at its default voltage value, which is set up by the mode balls (M0, M1). After $V_{CL,Vout}$ reaches UVLO level, each channel enters soft-start mode after a start-up delay time, which is programmable by a One Time Programmable (OTP) unit. When $V_{CL,Vout}$ reaches the target threshold voltage, RSTO (Power On Reset) should be high after $T_{RSTO\_POR\_DELAY}$ time without $V_{CL,Vin}$ falling below the UVLO level. The $T_{RSTO\_POR\_DELAY}$ time is also programmable by OTP. If PMIC receives a reset signal, RSTO should be high after $T_{RSTO\_PMRST\_DELAY}$ time, which is again programmable by OTP.

If Reset signal (PMRST or OTP_R) is received, PMIC makes all channels return to the default voltage which is set up by mode balls. Each channel has fixed (Typ. ±5%) Enable Delay Time and Reset Delay Time when turn on by Reset signal. Reset delay time can be programmable by OTP with a tolerance of ±10%. If $V_{CL,Vout}$ goes down under UVLO level, all channels turn off after specified time (Off Delay Time) by OTP and the discharge function of channels that were enabled by OTP turns on until the possible level. The Off delay time can be programmable by OTP with a tolerance of ±10%.

Following are the scenarios under which the buck converter channels are disabled as each of these scenarios corresponds to a fault condition.

- Over Load Protection (OLP): If the output voltage remains below about 80% of the target output voltage for more than specified delay
- Short Circuit Protection (SCP): If the output voltage remains below about 33% of the target output voltage for more than specified delay
- Over Voltage Protection (OVP): If the output voltage exceeds the over voltage protection (OVP) threshold

To reactivate the channels, $V_{CL,Vin}$ should be recycled after the fault condition is removed.

All buck converters will have a typical inductor ranging between 0.1 to 1 µH. The selected inductor has to be rated for its DC resistance and saturation current. An inductor with lowest DC resistance should be preferred for highest efficiency [1][5]. Following formula can be used to calculate the maximum inductor current under static load condition. The saturation current of the inductor should be rated higher than the maximum inductor current because during heavy load transient the inductor current will rise above the calculated value.

$$\Delta IL = VOUT \cdot (1 - VOUT/VIN) / L \cdot fSW$$

$$ILmax = ILOAD + (\Delta IL/2)$$

Where:
fSW = switching frequency
L = inductor value
ΔIL = Peak to peak inductor ripple current
ILmax = Maximum inductor current

## III. VERIFICATION CHALLENGES

Some of the challenges in verifying a Memory PMIC such as the one shown above are:

- Simulation performance (runtime) is a predominant factor when we realise these blocks at their transistor level (TL) abstractions. One buck converter typically takes 1 to 1.5 days to simulate with commercial simulators available in the industry, and with the most optimum SPICE settings.
- Verifying the default voltage regulation values of the buck converter channels by writing data to the huge set of register banks (typically hundreds)
- Addition of appropriate assertions and checkers to check for various functionalities such as power-on and power-off behavior, single-byte and multi-byte write and read from I2C interface, enabling & disabling of various power modes, and testing the protection logic
- Pin connectivity checks involving passing values between Analog and Digital (RTL), wherein the value supplied on one side is checked for propagation on the other side to ensure there are no pin mismatches (induced either due to improper spec or human-induced error during design)

Though all the above challenges are of concern, the problem of simulation runtime has always been predominant in analog-centric ICs, leading to longer verification times. One approach to handle this problem is to come up with Verilog-A behavioral models [3] for all the analog blocks and use them in lieu of their TL abstractions. However, Verilog-A requires SPICE simulator and so the simulation is still time-step driven unlike logic simulators that are purely event-driven. Also, the accuracy of the Verilog-A model to match it closely with the schematic depends on the person who is modelling. Modelling with Verilog-A is also prone to issues such as discontinuities in the signal values, etc., which can lead to convergence issues during simulation [3][4]. Finally, depending on the modelling style, it might lead to performance deterioration instead of performance improvement if proper techniques such as transition filters, bound step, etc., are not implemented in the model.

An alternative and better approach is to use Real Number Modelling (RNM), wherein the analog blocks are modelled using SystemVerilog "real" built-in nettype, that makes use of powerful SystemVerilog IEEE 1800-2009 features such as real number variables, input/output real ports, assign statements to real variables, and SystemVerilog Assertions (SVA) including real variables [6][7][8].

## IV. PROPOSED FLOW IMPLEMENTATION

Considering the above aspects, the authors of this paper worked on creating a Mixed-Signal Verification (MSV) framework for the Memory PMIC as follows. We started with the schematic of our PMIC Chip-top level, from which we created a mixed-signal configuration. In this configuration, we bound the blocks of interest to "symbol" [12] view, so that we can use behavioral models for the time-consuming analog blocks. We then extracted a Verilog-AMS netlist [4][6] of this configuration. This Verilog-AMS netlist was post-processed to convert it to SV netlist so that we can avoid electrical disciplines as our intention is to keep the simulation purely event-driven and perform a Digital Mixed-Signal (DMS) simulation. This SV netlist along with the RTL codes for the digital blocks and with the developed behavioral RNM models for analog blocks, were compiled and simulated using logic simulator that is purely event-driven [12][13]. As we were primarily interested in top-level integration checks, we were not in need of SPICE level accuracy. Hence, we integrated the developed RNM models together and took care of the Real to Logic and Logic to Real conversions using appropriate connect rules [12][13].

Creating analog behavioral models was by itself a multi-step task, which consisted of model creation, testbench creation, simulation and validation of models [7][8][12] against the corresponding schematic. Models were developed as SystemVerilog RNM models with nettype "real" and with *CDS_res_wrealsum* as the resolution function [8][10][12][13]. The developed models were simulated and the results were compared with their respective schematic simulation results using common testbenches for both, thereby validating those models [13].

A diagrammatic representation of the adopted flow is shown below.

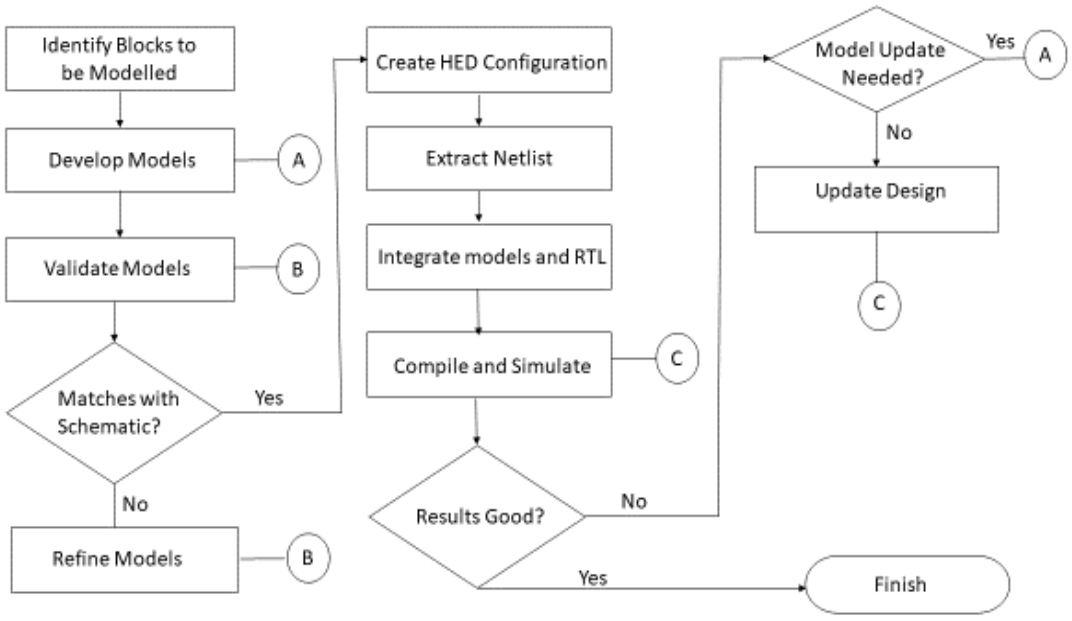


Figure 2. Adopted flow for Mixed-Signal Verification for the Memory PMIC

The above flow was implemented on an industrial design of a Memory PMIC that was used for SSD application. The results thus obtained have been quantified to show a performance improvement in the order of 100-1000x compared to schematic-based simulations.

## V. AUTOMATION AND COVERAGE

Next task was to check the default values loaded onto the hundreds of register banks. For this purpose, we developed a Python script that takes as its input the register map information provided by the designer. All the key data from the register map such as register name, range and the default (reset) value of the register are passed as inputs to the script. The script would then create a SystemVerilog file that defines a module and declares local logical variables for each register name, and maps these variables to the actual register names in the design using $xm_mirror() [11] utility available in the logic simulator. This is followed by SystemVerilog Assertions (SVA) in the module, wherein we compare the values of each field in the simulation against the default values coming from the register map. If there are mismatches, assertion violation messages are printed.

TABLE I
REGISTER MAP INFORMATION

| Address | Register Name | Range | Reset Value |
|---|---|---|---|
| 0x00 | CH1_OUT | [6:0] | 0x01 |
| 0x02 | CH2_OUT | [6:0] | 0x02 |
| 0x03 | CH3_OUT | [6:0] | 0x03 |
| 0x04 | CH4_OUT | [6:0] | 0x04 |
| 0x05 | CH5_OUT | [6:0] | 0x05 |

A pseudo code of the auto generated SV module from the script is shown below.

```
//Declarations
      logic  CH1_OUT;
      logic  CH2_OUT;

//Mirroring
       If (reg_value_check=1)
           mirror(`MAIN_CH1_OUT ->   CH1_OUT);
           mirror(`MAIN_CH2_OUT ->   CH2_OUT);
       end

//Assertion
       if  (CH1_OUT == 0x01)       true => Do nothing
                                   False => Display there is a mismatch
       if  (CH2_OUT == 0x02)       true => Do nothing
                                   False => Display there is a mismatch
```

Final task was to get a good coverage by checking the voltage regulation values of all the buck converter channels for different data byte written from I2C. This was achieved by using the SystemVerilog *randomize()* method to write random values to the buck registers and checked the corresponding output voltages automatically using the formula based checker task. We also covered the minimum, default and maximum output values using directed stimulus approach.

Register description of the buck voltage register CH1_VOUT is taken and shown below for illustration.

TABLE II
CODE VS DATA MAPPING FOR CH1 REGISTER

| Function | Address | Code | Value |
|---|---|---|---|
| CH1_VOUT | 0x00 | 0x00 | 1.2V |
| | | 0x01 | 1.19375V |
| | | 0x02 | 1.1875V |
| | | … | … |
| | | 0x40 | 0.8V |
| | | 0x70 | 0.5V |

To check the output voltage value corresponding to each random code for a given buck converter channel as shown above, below steps were followed by using the randomization technique.

**A. Write random code to register from I2C**

```
rand bit [7:0] i2c_wdata; //declared random variable
repeat(itr) begin // more iteration (itr=10 for example) to hit different values
i2c_wdata.randomize();  //method to generate random data
`uvm_info(get_type_name(),$sformatf("rand data  i2c_wdata: 0x%x ",i2c_wdata),UVM_LOW)
`WRITE_S (`SLAVE_ADDR, `CH1_VOUT,i2c_wdata);  //I2C write to the RTL design buck1 register
```

Once the random data is written to the register, the expected buck voltage is calculated and compared against the actual buck voltage.

```
dvs_output_compare ("BUCK1",i2c_wdata[6:0] );
task dvs_output_compare (input string str, input int dvs_code);
 case (str)
 "BUCK1" : begin
          offset   = `BUCK1_OFFSET;
          max_volt = `BUCK1_MAX;
          min_volt = `BUCK1_MIN;
          dvs_step = `BUCK1_STEP;
          actual_vout = `BUCK1_ACT;
 end
 ..
 endcase
endtase
```

**B. Calculate the expected buck voltage by formula**

```
 if (offset >= max_volt) // for decreasing DVS
   expected_vout = ((offset - (dvs_step*dvs_code)) < min_volt) ? min_volt:((offset - (dvs_step*dvs_code))>
max_volt)? max_volt: (offset - (dvs_step*dvs_code));
 else  // for increasing DVS
   expected_vout = ((offset + (dvs_step*dvs_code)) < min_volt) ? min_volt:((offset +
(dvs_step*dvs_code))> max_volt)? max_volt: (offset + (dvs_step*dvs_code));
 compare_data (str, actual_vout,expected_vout);
```

**C. Compare the expected and actual buck voltage values**

```
task automatic compare_data(input string str, input real actual, input real expected);
      if($sformatf("%0.6f",actual) != $sformatf("%0.6f",expected))
         `uvm_error("COMPARE_VOLTAGE",$sformatf("%s :: Actual value = %0.6f doesnot match with
expected value = %0.6f",str,actual,expected))
      else
```

```
`uvm_info("COMPARE_VOLTAGE",$sformatf("%s :: Actual value = %0.6f match with expected value = %0.6f",str,actual,expected),UVM_LOW)
  endtask
```

With this randomization method, we could get good coverage on the buck converter channels to check if the PMIC generates proper voltages for any random code.

## VI. RESULTS AND CONCLUSION

We were able to complete the verification of the Memory PMIC with the above-mentioned approach thereby achieving the below metrics. Plot of some of the key test results are shown for illustration.

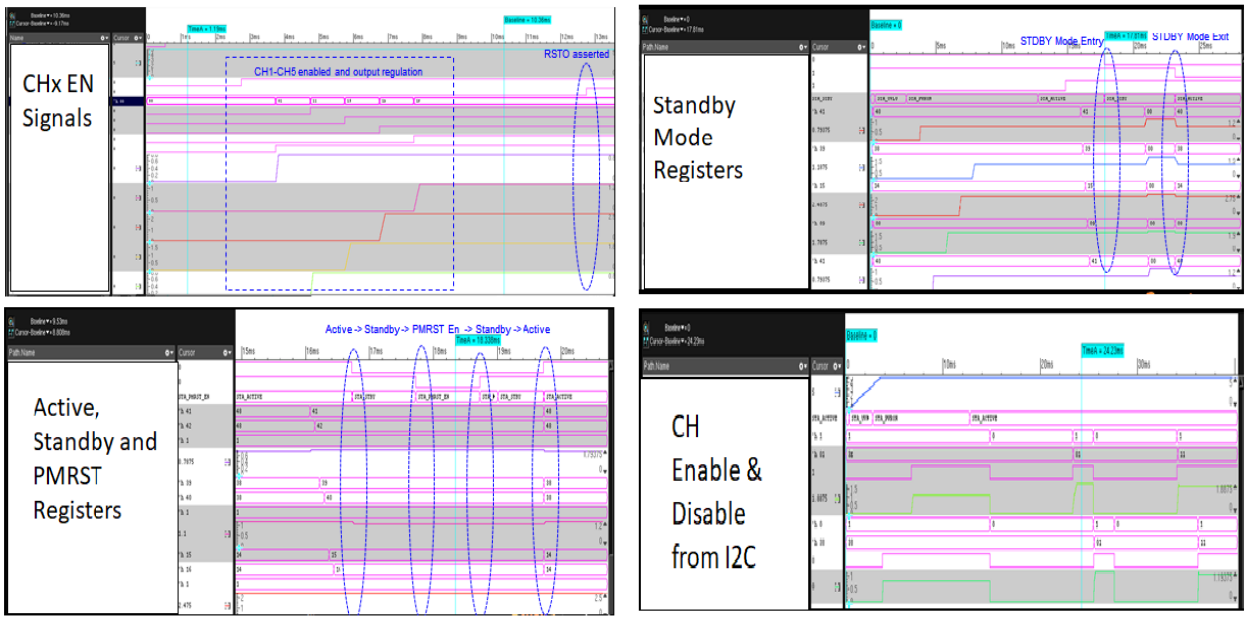


Figure 3. Key Test Results

- 68 test cases were completed on time with this approach spanning across various scenarios including below
  - Power-on and power-off
  - Write and read from I2C interface
  - Dynamic Voltage Scaling
  - Protection Logic
  - Standby mode entry & exit
  - Pin connectivity checks between analog and digital
- 25 SV-RNM models were developed and validated against the schematic
- 6 weeks TAT for the whole activity, ensuring first-pass silicon success

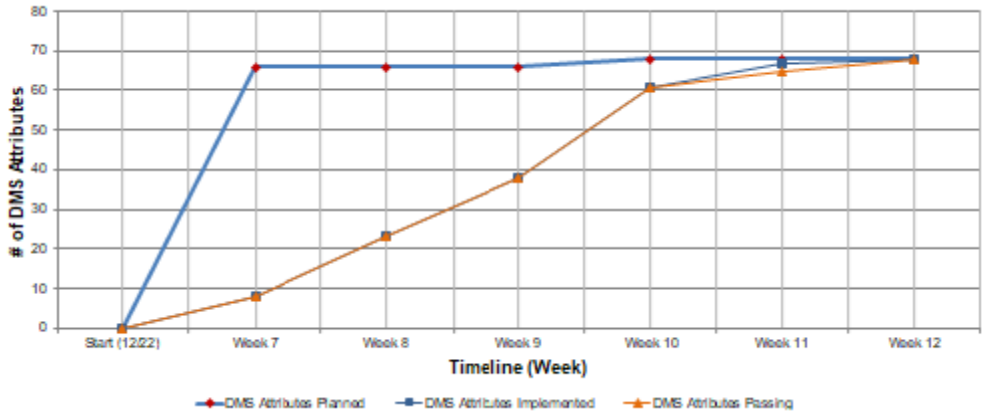


Figure 4. Verification Completion Timeline

We thus created a true mixed-signal configuration of the Memory PMIC chip that consists of RTL for the digital and SV real (behavioural) models developed and validated against the schematic for the analog parts of the design. We were able to simulate this configuration using logic simulator which is event-driven, so we got a huge performance benefit compared to SPICE simulations with TL abstraction. We also avoided potential issues such as non-convergence and finding right tolerance settings that come with a SPICE simulation that is time-step driven. We came up with register check scripts using Python for verifying the default values loaded onto the hundreds of registers. We also enabled randomization to write data onto the register banks from I2C to check the corresponding buck default regulation values, thereby achieving coverage. We could thus complete the verification activity on time with this methodology and ensure first-pass silicon success. Future scope is to automate regression of the model validation activity. Also the scope of moving from SystemVerilog real to SystemVerilog EEnet based modelling to account for voltage, current and impedance in the models are currently being explored.

## ACKNOWLEDGEMENTS

Authors would like to kindly acknowledge the following people from the industry for their support and guidance towards this work.

- Manish Parmar, Manish Goel, Balajee Sowrirajan of **Samsung Semiconductor India Research**
- DaeHoon Han of **Samsung Electronics**
- Nandeesh Pattanashetty, Sundararajan A, Arumugan A, Yaswanth Sai Darimadugu of **Cadence Design Systems**

## REFERENCES

[1] Yong Liang, L.F. Tao, Colin Xing, "A New Solution of Power Management Integrated Circuit One Time Programable Test", China Semiconductor Technology International Conference (CSTIC), 2021

[2] Thinh Tran-Dinh et all, "Power Management IC With a Three-Phase Cold Self-Start for Thermoelectric Generators", IEEE Transactions on Circuits and Systems I: Regular Papers, 2021

[3] Jie Liu, Yu Ban, "A parametric model for a high speed heterogeneous current-steering digital-to-analog converter based on compiled Verilog-A and SPICE", IEEE International Conference on Integrated Circuits, Technologies and Applications (ICTA), Nov 2021

[4] Binbin Yang., et all, "Simulation of serial RRAM cell based on a Verilog-A compact model", 2021 XXXVI Conference on Design of Circuits and Integrated Systems (DCIS), Nov 2021

[5] Chengwu Tao, Ayman Fayed, "Analysis and modelling of buck converters output spectrum in CCM with PWM control", IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS), Aug 2011

[6] Constantina Tsechelidou, Nikolaos Georgoulopoulos, Alkiviadis Hatzopoulos, "Design of a SystemVerilog-Based Sigma-Delta ADC Real Number Model", 22nd Euromicro Conference on Digital System Design (DSD), Aug 2019

[7] Mina Louis, Mohamed Dessouky, Ashraf Salem, "PLL Real Number Modelling in SystemVerilog", 16th International Conference on Synthesis, Modelling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), July 2019

[8] Nikolaos Georgoulopoulos, Athanasios Mekras, Alkiviadis Hatzopoulos, "Design of a SystemVerilog-Based VCO Real Number Model", 8th International Conference on Modern Circuits and Systems Technologies (MOCAST), May 2019

[9] Nikolaos Georgoulopoulos, Ioannis Giannou, Alkiviadis Hatzopoulos, "UVM-Based Verification of a Mixed-Signal Design Using SystemVerilog", 28th International Symposium on Power and Timing Modelling, Optimization and Simulation (PATMOS), July 2018

[10] Nikolaos Georgoulopoulos, Alkiviadis Hatzopoulos, "Real number modelling of a flash ADC using SystemVerilog", Panhellenic Conference on Electronics and Telecommunications (PACET), Nov 2017

[11] Daniel Stanley, Can Wang, Sung-Jin Kim, Steven Herbst, Jaeha Kim, Mark Horowitz, "Fast Validation of Mixed-Signal SoCs", IEEE Open Journal of the Solid-State Circuits Society, October 2021

[12] Erich Barke., et all, "Embedded tutorial: Analog-/mixed-signal verification methods for AMS coverage analysis", Design, Automation & Test in Europe Conference & Exhibition (DATE), April 2016

[13] "Mixed Signal Verification – System Verilog Real Number Modelling Overview", Cadence Online Support

[14] Modelling a Fully Differential Ring Amplifier with SystemVerilog Real Numbers (RAK), Cadence Online Support

[15] "AMS Design and Model Validation User Guide ICADVM18.1", Cadence Online Support