Advanced UVM Based Chip Verification Methodologies with Full Analog Functionality

Simul Barua
Ulkasemi Inc.
20045 Stevens Creek Blvd,
Suite 2B
Cupertino, CA 95014
simul@ulkasemi.com

FNU Farshad
Ulkasemi Inc.
20045 Stevens Creek Blvd,
Suite 2B
Cupertino, CA 95014
farshad@ulkasemi.com

Henry Chang
Designer's Guide Consulting, Inc.
3043 Meridian Ave #35
San Jose, CA 95124
henry@designers-guide.com

***Abstract-* As the demand for high-performance computing System on Chips (SoCs) increases, the importance of full-chip verification with both analog and digital subsystems cannot be overstated. However, current full-chip verification solutions focus heavily on UVM-driven digital verification testbenches, which are not feasible for analog circuits due to the high simulation run time of SPICE models. In this paper, we discuss the limitations of traditional full-chip verification flows and propose a comprehensive verification approach that includes both digital and analog functionality. We also explore two prevailing approaches to the modeling of analog blocks, the analog mixed-signal (AMS) approach and the digital mixed-signal (DMS) approach, and discuss the trade-offs when using both approaches. Lastly, we propose a behavioral model solution to integrate analog functionality into the UVM-based chip-level verification environment, which can significantly reduce the risk of failure and ensure the complete functionality of the SoC.**

I. Introduction

Due to the rise of AI, Internet of Things (IoT), and cloud computing, demand for high-performance computing System on Chips (SoC) is higher than ever. To ensure the correct functionality of these complex SoCs, there is a plethora of digital verification solutions offered by various vendors and in-house verification teams. But in order to ensure first functional silicon, we need to perform full-chip verification with both the analog and digital subsystems. Currently, there is no standard methodology for performing full chip verification and existing functional verification solutions focus heavily on UVM-driven digital verification testbenches. The UVM-AMS Working Group at Accellera is working on standardizing the full chip verification by extending UVM's support for analog verification [1]. Due to the high simulation run time of SPICE models and analog circuits, it is not feasible to run analog circuits in schematics along with UVM testbenches. So, it is desirable to model the functional behavior of analog circuits using a higher-level language such as SystemVerilog, Verilog-AMS, etc., such that they provide full functional coverage and analog assertions to verify connectivity along with reasonable accuracy and faster simulation run times.

There are two prevailing approaches to the modeling of analog blocks. The first one is the analog mixed-signal (AMS) approach, where complex circuit behaviors are modeled accurately by defining signals in electrical and discrete domains using Verilog-AMS or VHDL-AMS. In this approach, the analog signals are modeled as a continuous domain electrical signal with voltage and current disciplines. The second one is the digital mixed-signal (DMS) approach using Verilog-AMS WREAL or SystemVerilog, where analog electrical behavior is modeled in event-driven discrete form. DMS is gaining popularity for its faster simulation run time so that system-level issues can be found earlier in the design cycle. With AMS models, it is easier to model continuous-time behavior, and AMS models co-simulate with schematics more smoothly when there is complex loading behavior at the interfaces. Both approaches also have different ramifications at the chip level.

In this paper, we will discuss digital-driven chip-level verification and the trade-offs when using both approaches, along with examples to illustrate our approach.

II. Limitation of Traditional Full-Chip Verification Flow

In today's technological landscape, SoCs are becoming increasingly complex as analog and digital circuitry are becoming more intricate. However, the current chip-level verification flow tends to focus mainly on digital functionality, leaving another critical part of the SoC, which resides within the analog portion, unverified. In its minimal form, the analog side is responsible for powering up the chip and generating clocks for the entire SoC, making it essential for starting up and running operations. Moreover, for most SoCs, the interface with real-world signals, such as sound, light, temperature, pressure, etc., uses an analog interface to capture or transmit information. Traditional UVM-based chip-level verification environments often assume either analog design is functionally correct and analog blocks are replaced with simple digital behavioral models or analog signals are tied to either logic 1 or 0. The powerup sequence is usually assumed to be working correctly and skipped. However, these assumptions can lead to undiscovered issues with analog-digital communication, increasing the chance of chip failure.

Therefore, analog functionality should not be assumed or avoided during chip-level verification. It should be included and prioritized in the verification environment to ensure that the whole SoC operates smoothly and efficiently. A comprehensive verification approach that includes both digital and analog functionality can significantly reduce the risk of failure and ensure the complete functionality of the SoC.

However, integrating analog functionality into the chip-level-verification environment is a critical challenge the semiconductor industry faces today. This is because analog schematics are developed in SPICE, which are slower to simulate than the digital RTL to a great extent. Integrating these slower schematics directly into the UVM environment poses several challenges and requires additional effort and time. Furthermore, simulation time increases significantly, leading to a significant impact on the overall design cycle.

To address this challenge, researchers have devised a behavioral model solution where the behavior of the analog design is written using different EDA tool-supported modeling languages. These models can then be validated against the schematics to ensure they fully replicate the schematic behavior. These validated models are then easy to integrate into the UVM-based chip-level verification environment, speeding up simulation time and aiding full-chip-level verification. In Fig. 1, we demonstrate the contrast between a digital-only verification method where the analog blocks are opaque and our approach, which integrates the verified analog models into the chip-level verification environment, thereby illuminating the analog section. This integration enables the creation of combined digital and analog test cases that ensure seamless full-chip functionality.

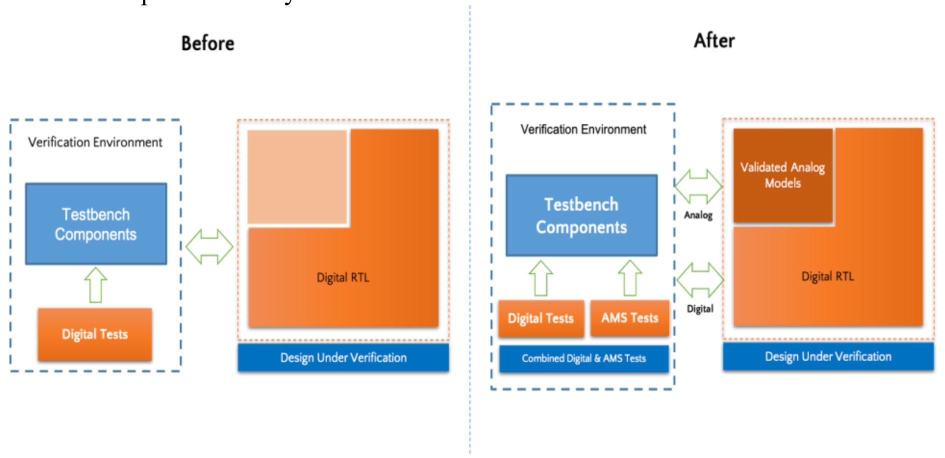


Figure 1. Chip-level verification environment with no analog models where there are only digital tests (“Before”); integration of validated analog models allowing for both analog and digital tests to verify the full chip (“After”)

III. Different Analog Modeling Approaches

Numerous industry-standard approaches are available for modeling analog behavior, each with advantages and limitations. Verilog-AMS and the DMS approach are two popular options for mixed-signal verification.

The AMS approach using Verilog-AMS is a flexible option for mixed-signal systems, allowing continuous and discrete signals to be modeled. It is widely used in the industry and has become a standard for mixed-signal simulation, ensuring compatibility across different tools and platforms. However, using Verilog-AMS in a chip-level verification environment can be challenging, as it requires both the continuous and discrete solver, while the UVM testbench uses a discrete solution. This approach is more suited for finding more subtle electrical bugs, such as those that can only be seen in a continuous time simulation or ones involving detailed electrical interfaces.

On the other hand, the DMS approach with real-number modeling (SV-RNM) or user-defined nettype (SV-UDN) in SystemVerilog provides a high level of precision for analog behavior by modeling continuous signals as real-valued numbers. In addition, the SV-UDN, also known as the discrete electrical approach in SystemVerilog, allows the creation of custom data types for electrical modeling involving voltage and current. This approach, with the help of the resolution function, enables the use of Ohm's law to solve for voltage and current for Thevenin and Norton equivalents. However, both the DMS approaches have drawbacks and limitations, especially when modeling complex analog behavior. They are more typically used when the emphasis is more on chip-level system functionality, where the details of the analog have been verified at a lower level. As such, DMS approaches focus more on finding bugs in the RTL and the communication between the RTL and the analog.

All the AMS and DMS approaches find analog supply and bias connection bugs, interface bugs between the digital and the analog, basic functional bugs such as in the startup sequence, bugs in the programming of the analog, bugs in analog signal flow, bugs in digital/analog dataflow as well as all the bugs in a digital-only verification approach.

IV. Case Study: Full-chip verification of a SoC with Analog Functionality

We will demonstrate how integrating analog functionality can enhance the verification of a simple System-on-chip (SoC) design. We will create behavioral models of the analog subsystem using the three previously discussed modeling approaches, i.e., Verilog-AMS, SV-RNM, and SV-UDN, and then integrate these models into the chip-level UVM verification environment. We will then perform various chip-level simulations to assess the performance and challenges associated with each modeling approach.

A. Description of the SoC Design

Due to proprietary concerns and to provide a more straightforward explanation of the different modeling approaches, we use an example System-on-chip (SoC) design, as shown in Fig. 2.

The analog subsystem of the SoC contains a power management unit (PMU) and a clock management unit (CMU). The PMU comprises sub-blocks such as low-dropout (LDO) regulators, power-on-reset (POR), voltage reference blocks, and bias generators, providing power and reset for the whole SoC. On the other hand, the CMU contains oscillators and a programmable phase-locked loop (PLL).

The simple digital subsystem designed using Verilog RTL mainly consists of different status and control registers to manage and monitor the PMU and CMU. For communication between the input/output (I/O) interface and the registers, we use a simple Advanced Microcontroller Bus Architecture (AMBA) Advanced High-performance Bus (AHB) protocol.

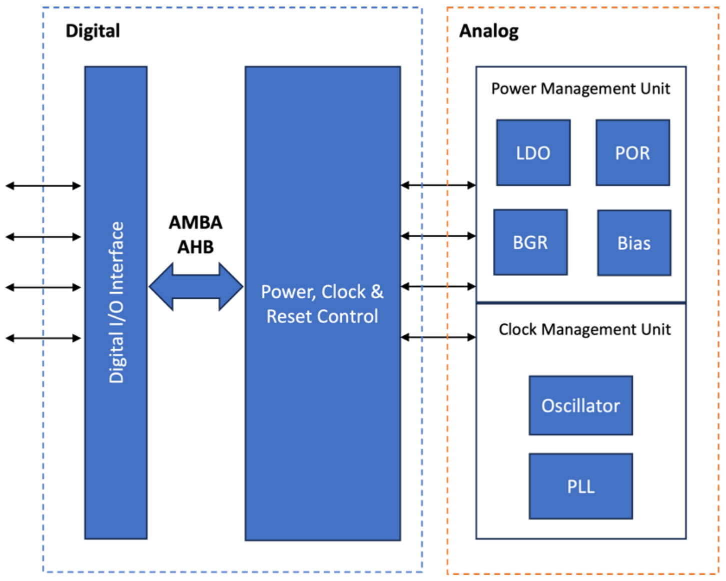
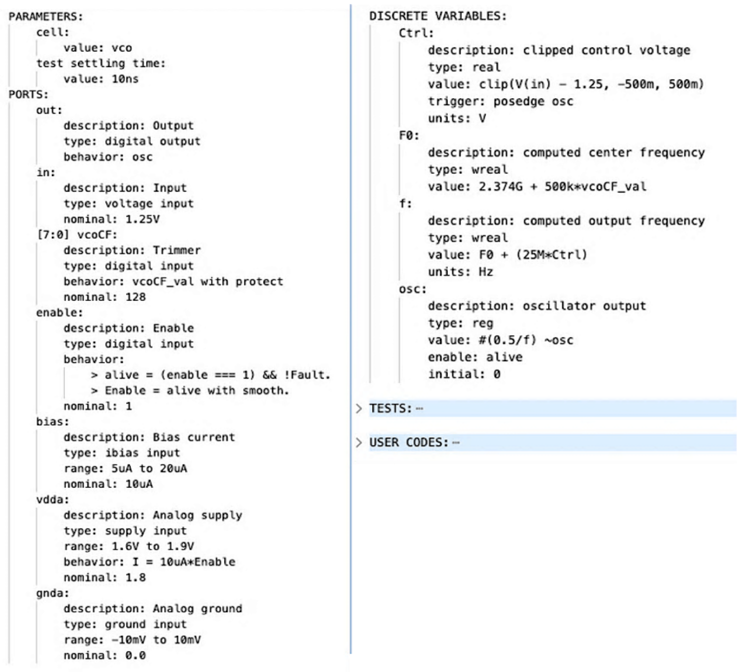


Figure 2. An example SoC design to demonstrate different modeling approaches for full-chip verification.

B. Modeling process of the analog subsystem

We have created fully functional models for each analog subsystem block using AMS and DMS approaches. We aimed to achieve full functionality, so we considered the modeling of the supply current of each analog block. To speed up our work, we used the Model in Minutes (MiM) tool [2] to generate the models. The tool can generate various AMS and DMS models based on a text-based specification input. It also generates a Verilog-AMS testbench and the necessary scripts to validate the individual models against their schematics [3]. Fig. 3 shows the simple input specification of a voltage-controlled oscillator (VCO) used for the MiM tool to generate the models for the three different approaches.

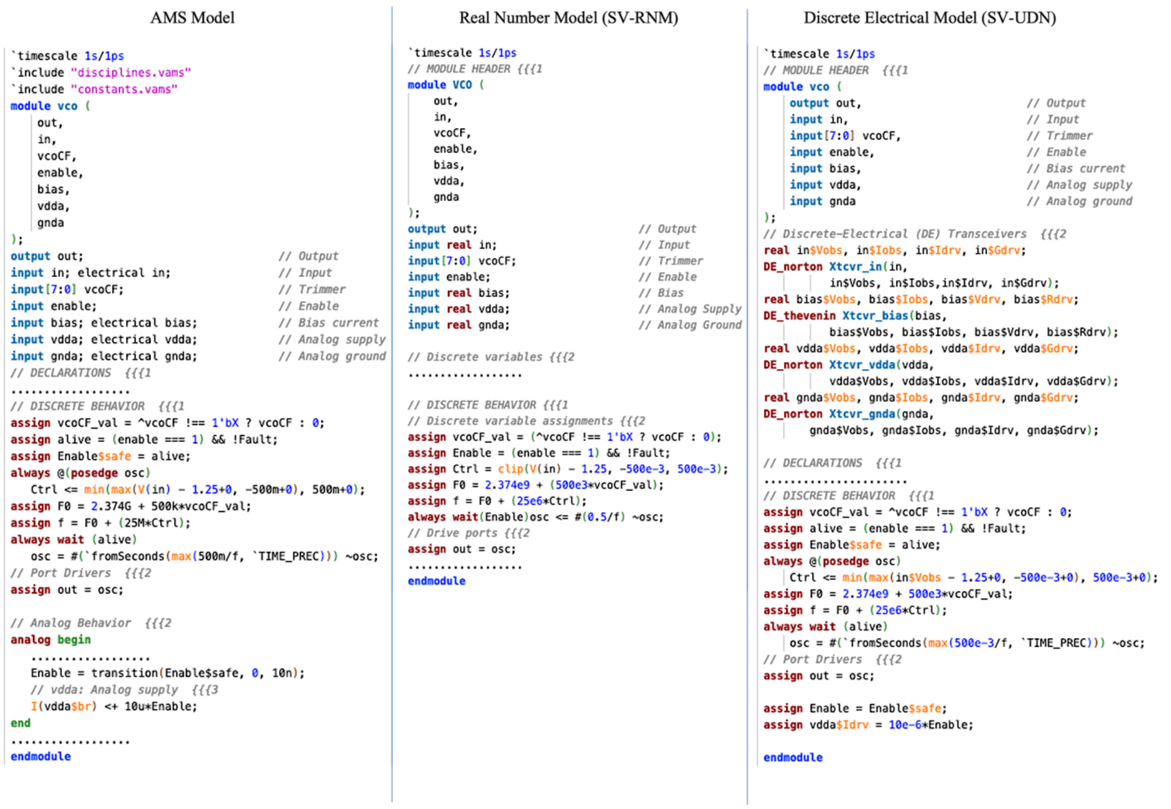


```
PARAMETERS:
    cell:
        value: vco
    test settling time:
        value: 10ns
PORTS:
    out:
        description: Output
        type: digital output
        behavior: osc
    in:
        description: Input
        type: voltage input
        nominal: 1.25V
    [7:0] vcoCF:
        description: Trimmer
        type: digital input
        behavior: vcoCF_val with protect
        nominal: 128
    enable:
        description: Enable
        type: digital input
        behavior:
            > alive = (enable === 1) && !Fault.
            > Enable = alive with smooth.
        nominal: 1
    bias:
        description: Bias current
        type: ibias input
        range: 5uA to 20uA
        nominal: 10uA
    vdda:
        description: Analog supply
        type: supply input
        range: 1.6V to 1.9V
        behavior: I = 10uA*Enable
        nominal: 1.8
    gnda:
        description: Analog ground
        type: ground input
        range: -10mV to 10mV
        nominal: 0.0
```

```
DISCRETE VARIABLES:
    Ctrl:
        description: clipped control voltage
        type: real
        value: clip(V(in) - 1.25, -500m, 500m)
        trigger: posedge osc
        units: V
    F0:
        description: computed center frequency
        type: wreal
        value: 2.374G + 500k*vcoCF_val
    f:
        description: computed output frequency
        type: wreal
        value: F0 + (25M*Ctrl)
        units: Hz
    osc:
        description: oscillator output
        type: reg
        value: #(0.5/f) ~osc
        enable: alive
        initial: 0

> TESTS: ⋯

> USER CODES: ⋯
```

Figure 3. Specification of the PLL's VCO used as the input to MiM

Fig. 4 shows the three analog behavioral models generated using the specification shown in Fig. 3. The MiM-generated SV-UDN or the discrete electrical model can either be used with its own UDN library or can utilize a wrapper to be used with the Cadence EEnet package [4]. To ensure the proper functionality and correctness of the models, we validated all the developed models against their respective schematics using the MiM-generated Verilog-AMS testbench, which runs through a set of tests to verify that the schematics and all three of the models are functionally equivalent.



AMS Model

```
`timescale 1s/1ps
`include "disciplines.vams"
`include "constants.vams"
module vco (
    out,
    in,
    vcoCF,
    enable,
    bias,
    vdda,
    gnda
);
output out;                                 // Output
input in; electrical in;                    // Input
input[7:0] vcoCF;                           // Trimmer
input enable;                               // Enable
input bias; electrical bias;                // Bias current
input vdda; electrical vdda;                // Analog supply
input gnda; electrical gnda;                // Analog ground
// DECLARATIONS  {{{1
..................
// DISCRETE BEHAVIOR  {{{1
assign vcoCF_val = ^vcoCF !== 1'bX ? vcoCF : 0;
assign alive = (enable === 1) && !Fault;
assign Enable$safe = alive;
always @(posedge osc)
   Ctrl <= min(max(V(in) - 1.25+0, -500m+0), 500m+0);
assign F0 = 2.374G + 500k*vcoCF_val;
assign f = F0 + (25M*Ctrl);
always wait (alive)
   osc = #(`fromSeconds(max(500m/f, `TIME_PREC)) ~osc;
// Port Drivers  {{{2
assign out = osc;

// Analog Behavior  {{{2
analog begin
   ..................
   Enable = transition(Enable$safe, 0, 10n);
   // vdda: Analog supply  {{{3
   I(vdda$br) <+ 10u*Enable;
end
..................
endmodule
```

Real Number Model (SV-RNM)

```
`timescale 1s/1ps
// MODULE HEADER {{{1
module VCO (
    out,
    in,
    vcoCF,
    enable,
    bias,
    vdda,
    gnda
);
output out;                           // Output
input real in;                        // Input
input[7:0] vcoCF;                     // Trimmer
input enable;                         // Enable
input real bias;                      // Bias
input real vdda;                      // Analog Supply
input real gnda;                      // Analog Ground

// Discrete variables {{{2
..................

// DISCRETE BEHAVIOR {{{1
// Discrete variable assignments {{{2
assign vcoCF_val = (^vcoCF !== 1'bX ? vcoCF : 0);
assign Enable = (enable === 1) && !Fault;
assign Ctrl = clip(V(in) - 1.25, -500e-3, 500e-3);
assign F0 = 2.374e9 + (500e3*vcoCF_val);
assign f = F0 + (25e6*Ctrl);
always wait(Enable)osc <= #(0.5/f) ~osc;
// Drive ports {{{2
assign out = osc;
..................
endmodule
```

Discrete Electrical Model (SV-UDN)

```
`timescale 1s/1ps
// MODULE HEADER  {{{1
module vco (
    output out,                         // Output
    input in,                           // Input
    input[7:0] vcoCF,                   // Trimmer
    input enable,                       // Enable
    input bias,                         // Bias current
    input vdda,                         // Analog supply
    input gnda                          // Analog ground
);
// Discrete-Electrical (DE) Transceivers  {{{2
real in$Vobs, in$Iobs, in$Idrv, in$Gdrv;
DE_norton Xtcvr_in(in,
            in$Vobs, in$Iobs,in$Idrv, in$Gdrv);
real bias$Vobs, bias$Iobs, bias$Vdrv, bias$Rdrv;
DE_thevenin Xtcvr_bias(bias,
            bias$Vobs, bias$Iobs, bias$Vdrv, bias$Rdrv);
real vdda$Vobs, vdda$Iobs, vdda$Idrv, vdda$Gdrv;
DE_norton Xtcvr_vdda(vdda,
            vdda$Vobs, vdda$Iobs, vdda$Idrv, vdda$Gdrv);
real gnda$Vobs, gnda$Iobs, gnda$Idrv, gnda$Gdrv;
DE_norton Xtcvr_gnda(gnda,
        gnda$Vobs, gnda$Iobs, gnda$Idrv, gnda$Gdrv);

// DECLARATIONS  {{{1
......................
// DISCRETE BEHAVIOR  {{{1
assign vcoCF_val = ^vcoCF !== 1'bX ? vcoCF : 0;
assign alive = (enable === 1) && !Fault;
assign Enable$safe = alive;
always @(posedge osc)
   Ctrl <= min(max(in$Vobs - 1.25+0, -500e-3+0), 500e-3+0);
assign F0 = 2.374e9 + 500e3*vcoCF_val;
assign f = F0 + (25e6*Ctrl);
always wait (alive)
   osc = #(`fromSeconds(max(500e-3/f, `TIME_PREC))) ~osc;
// Port Drivers  {{{2
assign out = osc;

assign Enable = Enable$safe;
assign vdda$Idrv = 10e-6*Enable;

endmodule
```

Figure 4. Analog behavioral model for a VCO in the three different modeling approaches

C. *Chip-level UVM Verification Environment with Analog Models*

We created a UVM testbench illustrated in Fig. 5 to test the SoC. The UVM testbench comprises two separate environments, one for handling the analog and the other for the digital side of the SoC. The analog environment has two agents: PMU_Agent and CMU_Agent. The PMU_Agent drives signals to the PMU to power up the SoC and monitors the generated subsystem supplies and resets. The CMU_Agent drives the PLL and monitors the clock frequency and locking. The Digital Environment has one AHB agent that reads and writes internal registers.

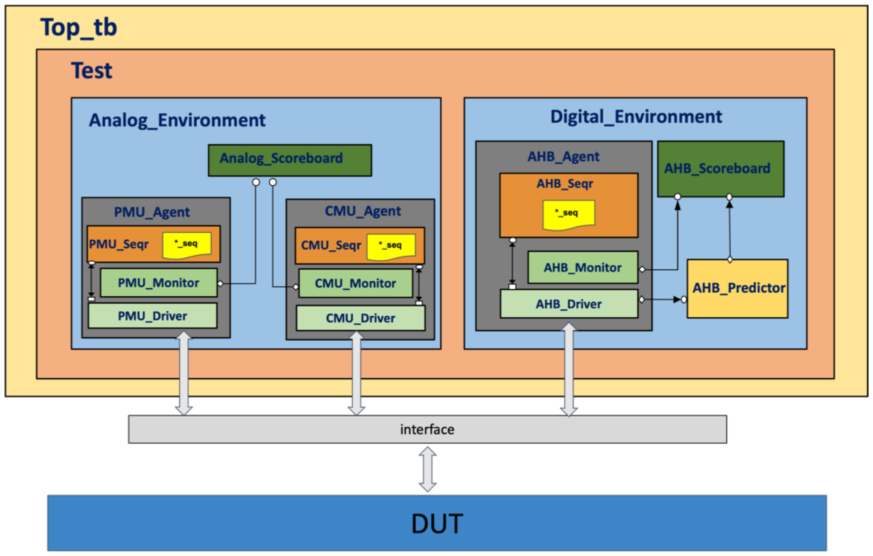


Figure 5. UVM testbench environment used to verify the chip-level functionality of the example SoC

The chip-top-level UVM test cases are designed to initiate the primary power supply. Once the SoC powers up, the internal control registers are written to configure the PLL. Finally, the phase lock time and the output clock frequency are measured once the PLL lock interrupt is detected.

IV. Results and Discussion

The models developed using each modeling approach were used individually for the analog subsystem in the UVM verification environment. Fig. 6 depicts the simulation waveform performed to test the PLL lock time and clock frequency. This waveform highlights the power-up sequence of the design, which is often avoided or assumed in traditional digital-focused UVM testbenches. Additionally, the results showcase the register write process using AHB to control the outputs of the PLL, verifying the correct interactions between analog and digital domains. Involving the analog functionality also enabled us to measure power consumption during each simulation phase, which is becoming increasingly crucial in modern systems where power efficiency is a critical design consideration.

Overall, the simulation results suggest that incorporating analog models into the verification environment offers a more comprehensive and accurate approach to chip-level verification, allowing for the detection of potential design issues that may be overlooked in a traditional verification environment.

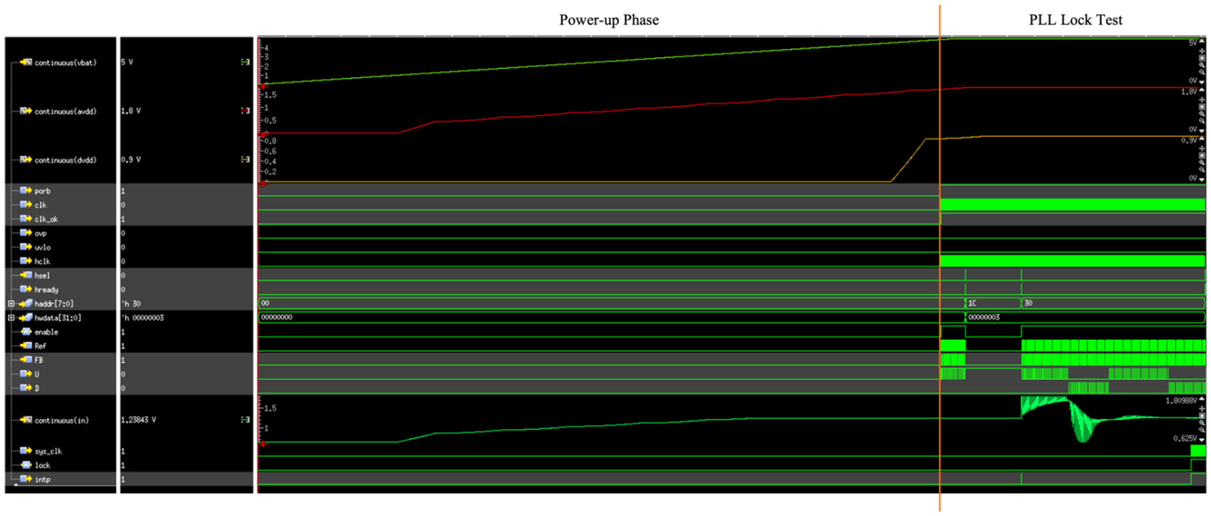


Figure 6. Simulation results of the chip-level verification testbench with analog functionality

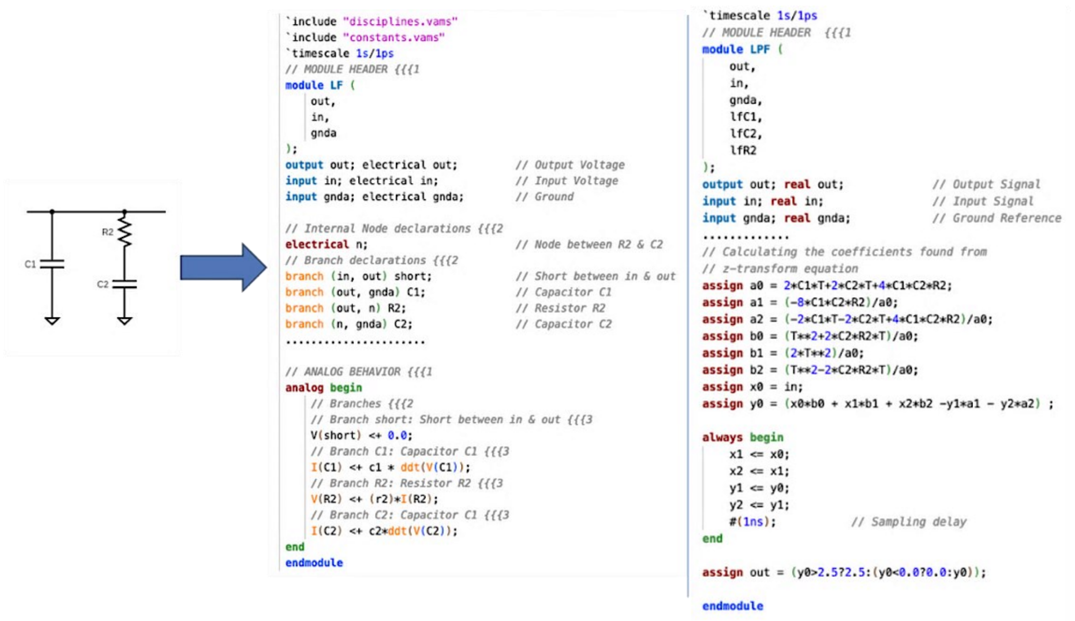
Now that we have shown the importance of analog functionality in chip-level verification, we will discuss the trade-offs between the AMS and DMS modeling approaches.

A. Comparison between the modeling approaches

The AMS approach supports analog circuitry modeling with the highest accuracy and ease. Developing models for complex analog circuitry, e.g., analog filters, can be easily achieved with the AMS approach by modeling the electrical components using the nodes and branch feature, which is unavailable in both the DMS approaches. Also, most Verilog-AMS system functions that are useful for performing complex mathematical operations on continuous signals are also unavailable in SystemVerilog DMS.

For DMS, in such cases, models are developed using their discrete-time transfer function equations to overcome these limitations, which adds extra complexity and takes more time to convert the continuous time equation to its discrete form. Although the SV-UDN models have some ability to solve this problem, both the DMS modeling approaches need a faster sampling mechanism to mimic the continuous analog system.

Fig. 7 shows the simplified circuit diagram of the PLL loop filter and its AMS and DMS models. AMS modeling is more straightforward as the electrical components can be easily modeled using nodes and branches. On the other hand, for both the DMS approaches, the low pass behavior is achieved using discrete z-transform equations and a high sampling rate.



```
`include "disciplines.vams"
`include "constants.vams"
`timescale 1s/1ps
// MODULE HEADER {{{1
module LF (
    out,
    in,
    gnda
);
output out; electrical out;          // Output Voltage
input in; electrical in;             // Input Voltage
input gnda; electrical gnda;         // Ground

// Internal Node declarations {{{2
electrical n;                        // Node between R2 & C2
// Branch declarations {{{2
branch (in, out) short;              // Short between in & out
branch (out, gnda) C1;               // Capacitor C1
branch (out, n) R2;                  // Resistor R2
branch (n, gnda) C2;                 // Capacitor C2
......................

// ANALOG BEHAVIOR {{{1
analog begin
    // Branches {{{2
    // Branch short: Short between in & out {{{3
    V(short) <+ 0.0;
    // Branch C1: Capacitor C1 {{{3
    I(C1) <+ c1 * ddt(V(C1));
    // Branch R2: Resistor R2 {{{3
    V(R2) <+ (r2)*I(R2);
    // Branch C2: Capacitor C1 {{{3
    I(C2) <+ c2*ddt(V(C2));
end
endmodule
```

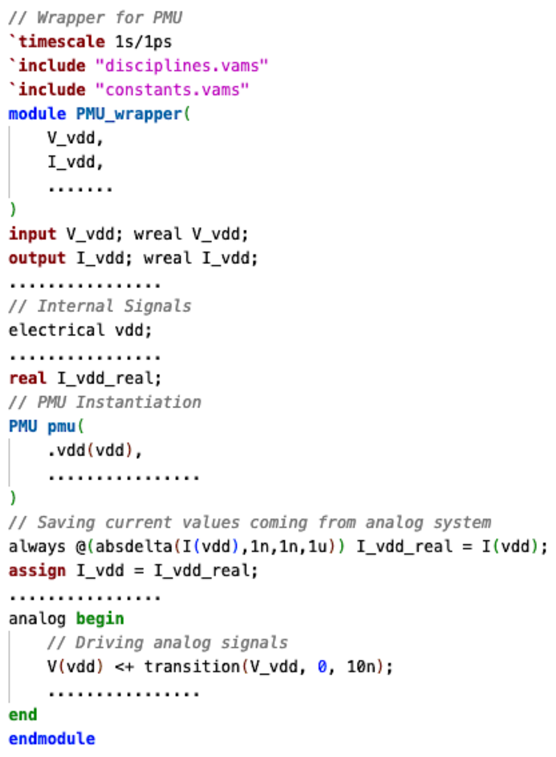
```
`timescale 1s/1ps
// MODULE HEADER  {{{1
module LPF (
    out,
    in,
    gnda,
    lfC1,
    lfC2,
    lfR2
);
output out; real out;               // Output Signal
input in; real in;                  // Input Signal
input gnda; real gnda;              // Ground Reference
.............
// Calculating the coefficients found from
// z-transform equation
assign a0 = 2*C1*T+2*C2*T+4*C1*C2*R2;
assign a1 = (-8*C1*C2*R2)/a0;
assign a2 = (-2*C1*T-2*C2*T+4*C1*C2*R2)/a0;
assign b0 = (T**2+2*C2*R2*T)/a0;
assign b1 = (2*T**2)/a0;
assign b2 = (T**2-2*C2*R2*T)/a0;
assign x0 = in;
assign y0 = (x0*b0 + x1*b1 + x2*b2 -y1*a1 - y2*a2) ;

always begin
    x1 <= x0;
    x2 <= x1;
    y1 <= y0;
    y2 <= y1;
    #(1ns);           // Sampling delay
end

assign out = (y0>2.5?2.5:(y0<0.0?0.0:y0));

endmodule
```

Figure 7. AMS and DMS model of the PLL loop filter

Also, the AMS and the SV-UDN approaches can model the voltage and current of electrical nets, which is unavailable in SV-RNM. As a result, modeling power consumption and other details are hard to achieve using the SV-RNM approach.

B. Integration of the models to the UVM chip-level verification environment

It is worth noting that integrating both SV-RNM and SV-UDN models into the UVM testbench is straightforward since they operate on the same discrete domain. As a result, less effort and time are required to bring them into the UVM environment. However, integrating the AMS models is more complex. Since these models rely on an analog solver to run the analog behavior, additional connect modules are necessary to convert signals between the electrical and discrete domains. Additionally, the AMS models require conversion wrappers to access voltages and currents directly from the UVM testbench. Fig. 8 illustrates a simplified version of the wrapper utilized for the AMS models of the PMU subsystem. The figure demonstrates how the analog signals are driven using signals originating from UVM and how the analog values are read and converted to the real-number values supported by the testbench.



```
// Wrapper for PMU
`timescale 1s/1ps
`include "disciplines.vams"
`include "constants.vams"
module PMU_wrapper(
    V_vdd,
    I_vdd,
    .......
)
input V_vdd; wreal V_vdd;
output I_vdd; wreal I_vdd;
................
// Internal Signals
electrical vdd;
................
real I_vdd_real;
// PMU Instantiation
PMU pmu(
    .vdd(vdd),
    ................
)
// Saving current values coming from analog system
always @(absdelta(I(vdd),1n,1n,1u)) I_vdd_real = I(vdd);
assign I_vdd = I_vdd_real;
................
analog begin
    // Driving analog signals
    V(vdd) <+ transition(V_vdd, 0, 10n);
    ................
end
endmodule
```

Figure 8. Verilog-AMS wrapper used for the AMS models of the PMU

C. Performance at the UVM chip-level verification environment

To determine the model performance of each approach, we measured the simulation time each took to run the PLL lock test. Table 1 shows the comparison between the simulation time each approach took. The performance of both the DMS approaches was quite similar, with only a slight difference in simulation time, taking a few minutes to complete. On the other hand, the AMS models took significantly longer, clocking in at around 20 minutes to complete the simulation. However, even this slowest AMS approach proved better than the SPICE level simulation, which took us several hours to run the standalone PLL design.

TABLE I
SIMULATION PERFORMANCE COMPARISON

Modeling Approach	Total Simulation Time
Real Number Models (SV-RNM)	~2 minutes
Discrete Electrical Models (SV-UDN)	~2 minutes
AMS Models	~20 minutes

IV. Conclusion

In conclusion, it is evident that full-chip verification, including both analog and digital subsystems, is crucial for ensuring the proper functioning of complex SoCs. The conventional chip-level verification flow that concentrates solely on digital functionality is insufficient since it ignores the analog portion, resulting in potential issues with analog-to-digital communication and increasing the risk of chip failure. We have explored two leading approaches to analog modeling, AMS and DMS, and discussed their advantages and disadvantages. While AMS models provide accurate analog functionality modeling, they perform poorly in the chip-level verification environment. On the other hand, SV-RNM models demonstrate high performance in the chip-level verification environment but have limitations in detailed functional modeling. Finally, SV-UDN models exhibit excellent performance at the chip level and can overcome most of the drawbacks of SV-RNM. However, they entail a steeper learning curve than traditional hardware description languages. Finally, our analysis suggests that a hybrid approach employing AMS and DMS models can overcome their limitations and achieve better chip-level verification outcomes.

Acknowledgment

We would like to express our gratitude to the Ulkasemi AMS Verification team members - Uchchhash Sarkar, Imrul Hasan Sabid, and Anowarul Azim, for their invaluable contribution and support in developing the DMS models and UVM testbench. We would also like to thank Designer's Guide Consulting Inc. for providing us access to the MiM tool for generating analog models.

References

[1] Accellera.org., "Workshop: UVM-AMS: A UVM-Based Analog Verification Standard," [Online]. Available: https://www.accellera.org/resources/videos/uvm-ams-workshop-2021. [Accessed: 15-Sep-2023].

[2] Designer's Guide Consulting, Inc., "Analog Verification Products: Models in Minutes," [Online]. Available: https://designers-guide.com/main/products/. [Accessed: 13-Nov-2023].

[3] H. Chang and K. Kundert, "Specification Driven Analog and Mixed- Signal Verification." [Online]. Available: https://dvcon-proceedings.org/wp-content/uploads/specification-driven-analog-and-mixed-signal-verification.pdf. [Accessed: 13-Nov-2023].

[4] J. Brennan, T. Ziller, K. Fotouhi, and A. Osman, "The How To's of Advanced Mixed-Signal Verification." [Online]. Available: https://dvcon-proceedings.org/wp-content/uploads/the-how-tosof-advanced-mixed-signal-verification.pdf. [Accessed: 13-Nov-2023]