Journal of Electronic Testing (2024) 40:315–328
https://doi.org/10.1007/s10836-024-06124-8

Design and Verification of a SAR ADC SystemVerilog Real Number Model

Nikolaos Georgoulopoulos[1] · **Theodora Mamali**[1] · **Alkis Hatzopoulos**[1]

Received: 29 November 2023 / Accepted: 6 June 2024 / Published online: 1 July 2024
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract
Mixed-signal applications have emerged as a significant trend in the semiconductor industry, with considerable efforts directed towards developing fast and accurate designs that integrate both analog and digital components. However, mixed-signal verification presents a major challenge due to the slow verification time and limited robustness of traditional verification techniques. In this study, a verification architecture for a successive-approximation register (SAR) analog-to-digital converter (ADC) real number model (Real Number Modeling – RNM) using SystemVerilog is presented, which utilizes an efficient UVM-based methodology. The proposed approach combines the UVM capabilities with the RNM model of the SAR ADC to generate a reusable, fast, and robust verification environment with a reduced time-to-market. The testbench creation and simulation were carried out using Cadence Xcelium. The proposed verification architecture employs constrained-random stimulus generation, analog assertions, and coverage metrics to enhance verification effectiveness. Additionally, aim of this work is to emphasize on the RNM efficiency with SystemVerilog, and apply its modeling capabilities for a SAR ADC. The presented real number model was compared to a Verilog-AMS model. The conducted experiments provided evidence that the proposed RNM model exhibits a significant improvement in simulation efficiency compared to previous works documented in the literature (simulation time was at 0.5 s, compared to a Verilog-AMS reference model's simulation at 20 s). This improvement in efficiency is achieved without compromising on the accuracy of the simulation, ensuring that the model maintains a satisfactory level of precision.

Keywords Analog-to-digital converter · Functional verification · Real number modeling · Successive-approximation register · SystemVerilog

1 Introduction

Most modern System-on-Chip (SoC) designs require a combination of analog and digital elements. Developing these mixed-signal SoC designs can pose several challenges, such as integrating both analog and digital elements into one system with high performance and accuracy. Additionally, evaluating mixed-signal verification is another crucial factor that needs to be considered.

To ensure the accurate and swift development of such systems, the simulation of designs and appropriate modeling are crucial. For most analog, digital, and mixed-signal designs, verification is dependent on simulation runs. Achieving the desired level of verification involves the incorporation of simulation data and data accuracy, which can become complicated depending on the design. However, verifying a mixed-signal design can be a time-consuming process, which can result in delays in the manufacturing process. One solution to this problem is to create a mixed-signal design using Real Number Modeling (RNM) and verify it using Universal Verification Methodology (UVM) while utilizing SystemVerilog [1–4].

RNM is a concept that combines elements from both analog and digital scopes [2–9]. It represents the operations of analog blocks using a signal flow model, where

Responsible Editor: S. Sindia

✉ Nikolaos Georgoulopoulos
ngeorgou@ece.auth.gr

Theodora Mamali
tsmamali@ece.auth.gr

Alkis Hatzopoulos
alkis@ece.auth.gr

[1] Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki, Thessaloniki, Greece

real-number values are employed to simulate the voltage behavior of the analog components. Furthermore, it relies solely on digital solvers to achieve high-speed simulations. RNM offers a practical and effective approach for rapidly and accurately modeling mixed-signal designs.

UVM is recognized as the most prevalent verification standard in modern digital circuits [10–20]. Although it is primarily used in digital circuits, it is also applied in mixed-signal applications. In digital circuits, UVM is utilized to leverage many capabilities such as constrained-random stimulus generation, verification planning, assertions, and coverage metrics creation. Conversely, the verification of analog elements in mixed-signal components has traditionally been accomplished using directed testing, Monte Carlo simulations, and corner analysis. However, modern analog solvers, processes, and methods often lack coverage metrics or testbench automation. As a result, the combination of Real Number Modeling (RNM) and UVM constitute a major key for establishing a fast and reusable verification environment for mixed-signal designs.

In this work, a four-bit successive-approximation analog-to-digital converter (SAR ADC) model is described with RNM using SystemVerilog. A SAR ADC is an analog-to-digital converter that converts a real value (voltage) into a digital representation of n bits, while using an efficient algorithm for finding the correct bit and an internal digital-to-analog converter (DAC) for reference, before finally converging upon a digital output for each conversion [21, 22]. In general, the SAR ADC consists of a SAR register, an internal Digital-to-Analog Converter (DAC), a comparator, and a sample-and-hold circuit. It converts an analog input voltage into a digital signal with n-bit resolution. Initially, the SAR register sets the most significant bit (MSB) to 1 and the other bits to 0. This digital output is sent to the internal DAC, which converts it back to an analog signal using a reference voltage (Vref). The proposed SAR ADC consists of a successive-approximation register (SAR register), a sample-and-hold (S/H) circuit, a comparator, and an internal DAC for reference. Except that, the RNM model takes into consideration two non-idealities, clock jitter and settling noise of S/H and provides measurement of the error rate of the four-bit SAR ADC in various frequency modes. Moreover, a UVM-based verification architecture for the SAR ADC RNM model is presented. The suggested testbench fully utilizes the benefits of UVM to provide resilience and fast execution speed for verifying the Design Under Test (DUT). It is worth noting that this architecture can be applied universally to any type of block-level verification for mixed-signal designs.

It is critical to emphasize the technical innovations of this work at this stage. The simulation performance of the presented SAR ADC is compared to a commercial Verilog-AMS setup with the analog circuit represented at transistor-level [23]. As part of the suggested concept, the SystemVerilog testbench was created and simulated in Cadence Xcelium and SimVision. With the given design, substantial gains in simulation time were achieved in all scenarios as compared to the reference testbench. The study then goes on to present various measures for assessing verification quality, which can be used to compare this verification architecture more effectively to future approaches.

The organization of this paper is as follows: Section 2 provides an overview of previous AMS modeling techniques. Section 3 illustrates the fundamental operation of the proposed SAR ADC RNM model. The basic UVM testbench architecture is evaluated in Section 4. The verification architecture for the proposed SAR ADC is then presented in Section 5. In Section 6, simulation results are analyzed, including simulation performance and verification efficiency, as well as a validation of the proposed model against a reference model in terms of simulation speed. Finally, Section 7 offers a discussion of the proposed model and its verification architecture, concluding the paper.

2 Analog and Mixed-signal Modeling Methods

This section covers various approaches for modeling analog and mixed-signal circuits, including Verilog-A from the analog modeling field, as well as Verilog-AMS (wreal), VHDL-AMS, and SystemVerilog from the real number modeling domain. Previous work, such as [24], prove the efficiency of using a SystemVerilog-based RNM model instead of the mainstream mixed-signal methods, but a brief reference is required.

2.1 Verilog-A

Verilog-A is a language standard used for modeling analog circuits, which forms a part of Verilog-AMS [22]. Verilog-A can be utilized by users of SPICE class simulators to create complex or simple analog models for their simulations. In SPICE simulators [25, 26] a system of nonlinear differential equations is generated and solved to describe the analog circuit in question. Key equations include Kirchhoff's laws: Kirchhoff's Current Law (the sum of currents entering a node equals zero) and Kirchhoff's Voltage Law (the sum of voltages around a loop equals zero). Additionally, constitutive equations are used, such as the current through a capacitor. Unlike in Verilog-A, where a constant-time subset is employed, these equations must be solved all at once. In contrast to SPICE, Verilog-A does not require the measurement of the entire device matrix at each solution point. Due to this difference, SPICE simulations tend to be slower than Verilog-A models [22, 27–33]. Figure 1 illustrates the description of a simple resistor in Verilog-A.

```
module resistor(p,n) ;

parameter real resistance = 1 ;
electrical p, n ;

analog

I(p,n) <+ V(p,n)/resistance ;

endmodule
```

Fig. 1 Example of a simple resistor in Verilog-A

2.2 Verilog-AMS

Verilog-AMS is an extension of the Verilog language that permits the definition of analog and mixed-signal circuit behaviors by incorporating analog and mixed-signal (AMS) modeling capabilities. This extension includes concepts for event-driven and continuous-time modeling, making it appropriate for modeling digital, analog, and mixed-signal systems. Procedures and statements in the digital domain operate in the same way as in Verilog, while all analog domain components function similarly to Verilog-A [3, 34–39]. Essentially, Verilog-AMS integrates analog and digital modeling by using event-driven semantics for digital parts and continuous-time semantics for analog parts. Digital events, such as changes in signal state, trigger updates in the analog simulation. This approach ensures synchronized operations between digital and analog domains. For instance, a digital clock edge can prompt an evaluation in an analog block, maintaining coherent system behavior.

In traditional Verilog, real numbers were represented by 64-bit vectors. However, this approach posed a challenge due to the unclear mapping of a real value (VHDL or Verilog) to 64-bit vectors. To overcome this difficulty, Verilog-AMS has introduced a new solution, wreal. Wreal is a net or wire that is inherently real-valued and connects structural elements with real-number physical connections. Recent advancements in wreal nets have introduced several advantages. Notably, wreal can now be associated with a SystemVerilog real value or a VHDL real signal, which is a significant improvement. Figure 2 depicts a sample-and-hold circuit utilizing wreal.

2.3 VHDL-AMS Approach

VHDL-AMS serves as a modeling language that caters to the needs of digital, analog, and mixed-signal systems. Although

```
module sample_and_hold(out, in, rst, clock);

input in,rst,clock;
electrical in;

output out;
wreal out;

real sampled;

always@(posedge clock or negedge rst)
out=V(in);
assign out = sampled;

endmodule
```

Fig. 2 Example of a sample-and-hold in Verilog-AMS with wreal

it derives its foundation from the IEEE standard 1076–2008 (VHDL), VHDL-AMS introduces additional features and modifications to enable the simulation and implementation of analog and mixed-signal models [6, 40–43]. This language bears some resemblance in concept to Verilog-AMS, and it incorporates a user-defined real subtype. An example of a VHDL-AMS-based 4-bit digital-to-analog converter (DAC) is presented in Fig. 3.

2.4 SystemVerilog Approach

In the past, nets were not able to be assigned real values, but SystemVerilog addressed this limitation by introducing the notion of real-number variables that could be manipulated like nets. This was particularly helpful in facilitating the process of real number modeling in mixed-signal circuits.

```
module dac (out, in);

parameter fullscale = 1;
input [3:0] in;
electrical out;

analog
V(out) <+ in * (fullscale/16);

endmodule
```

Fig. 3 VHDL-AMS model of a four-bit digital-to-analog converter

```
nettype real netreal;
netreal in, out;
assign out=in;
```

Fig. 4 SystemVerilog nettype with real data type

Real variables were able to be connected to input and output ports in SystemVerilog, but they could only be driven by a single driver. However, the nettype declaration included in the IEEE 1800–2012 SystemVerilog standard allowed for the resolution of this obstacle.

SystemVerilog IEEE 1800–2012 standard offers innovative frameworks that enable effective AMS modeling. Specifically, the integration of a new collection of nettypes with a real data form is key to achieving this. This real data form is essentially the same as the wreal nets found in Verilog-AMS [3, 4, 6, 8, 24, 44–47]. One significant advantage of the nettype approach is that it allows for the use of multiple drivers, making it a typeless net that is exclusively used for ports or interconnect building. Figure 4 offers a description of the real type nettype, where any changes made to the value of the real nettype in will result in changes to the real nettype out. Thanks to the additional utilization of UVM [13, 48–52], the SystemVerilog method for real number modeling is optimal for mixed-signal verification and offers exceptional verification performance.

2.5 Simulation Efficiency and Accuracy

The various levels of abstraction commonly used in analog segments of mixed-signal designs include the transistor netlist, transistor-level SPICE simulation, fast SPICE simulation, traditional analog modeling, real number modeling, and pure digital modeling. Figure 5 illustrates the relationship between simulation efficiency and accuracy for these levels of abstraction.

The transistor netlist offers the highest level of accuracy, at the penalty of high simulation time. SPICE and fast SPICE simulations are almost equally precise, but fast SPICE outperforms SPICE in terms of simulation speed. Traditional analog modeling techniques such as Verilog-A, Verilog-AMS, and VHDL-AMS [40] reduce simulation time compared to SPICE [25, 26] simulations. Real number modeling techniques, such as wreal and SystemVerilog RNM [5, 7–9], significantly enhance simulation efficiency to a level comparable to pure digital models. Analog models, such as Verilog-AMS analog and Verilog-A, employ an analog solver with greater computational complexity, whereas real number models (wreal, SystemVerilog RNM) and pure digital models use a digital solver. This distinction may account for the notable difference in simulation run times.

3 Proposed SAR ADC Model

This section introduces the SystemVerilog real number model for a SAR ADC, aiming to improve simulation efficiency while ensuring accurate modeling to meet satisfying levels.

3.1 Function of the SAR ADC

The SAR ADC, depicted in Fig. 6, is an analog-to-digital converter comprising a SAR register, an internal DAC, a comparator, and a sample-and-hold circuit. It is utilized to convert an analog input voltage into a digital representation with n-bits. The SAR register is configured with the most significant bit (MSB) set to 1 and the remaining bits set to 0. The resultant output is then directed to the internal DAC, which converts the digital representation back to an analog signal, utilizing a reference voltage (Vref).

The analog output is fed into the comparator circuit to be compared with the sampled input voltage (Vin). If the value from the DAC is higher than Vin, the corresponding

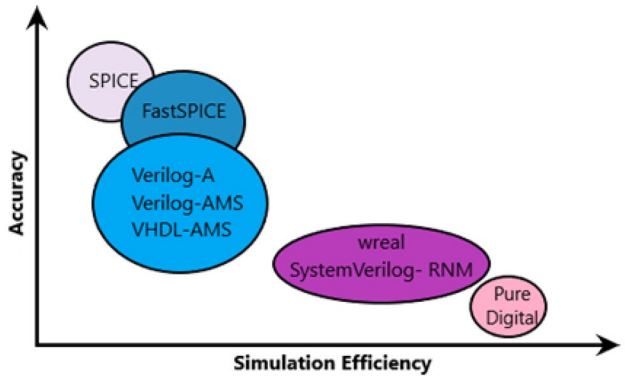

Fig. 5 Simulation Efficiency – Accuracy relation

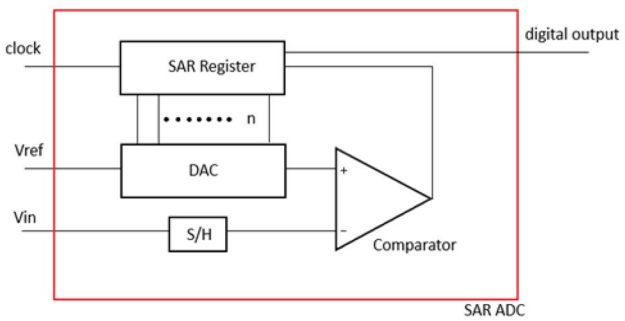

Fig. 6 The components of the SAR ADC

```
#(parameter n=4, RESET_TYPE="ASYNC")
 input logic  clock,
 input logic  reset,
 input logic  start,
 input real   vin,
 input real   vref,
 output logic [n-1:0] sar_result

assign mod_connect.clk=clock;
assign mod_connect.rst=reset;
...

sar_register#(n,RESET_TYPE) thesarregister(...);
dac#(n)                     thedac(...);
sample_and_hold#(RESET_TYPE)thesampleand_hold(...);
comparator                  thecomparator(...);
```

Fig. 7 Part of the proposed SAR-ADC top-level module

bit in the SAR register is set to 0, otherwise, the bit remains as 1. This process repeats for each bit in the SAR register, sequentially changing them to 1 in order to perform the same comparison. The resulting binary code represents the digital approximation of the input voltage.

3.2 Proposed SAR ADC SystemVerilog Real Number Model

In this work, a four-bit SAR ADC real number model is introduced using SystemVerilog. In Fig. 7, part of the top-level module of the SAR ADC real number model is presented. To be more precise, the proposed top consists of a SAR register, a DAC, a S/H and an analog comparator. Nettype with real type is used for the input ports, which consist of the analog input and the reference voltage levels. The design can operate with two different modes, asynchronous and synchronous reset. In asynchronous reset mode, the circuit will reset whenever reset signal is active 'irrespective' of the clock. In the opposite direction, a synchronous reset signal will only reset on the positive edge of the clock.

In order to fully transform the circuit from pure digital into mixed-signal, the insertion of two non-idealities was decided, clock jittering and settling noise in sample-and-hold circuit. Regarding clock jitter, it is another phenomenon that has drawn interest in the design of ADCs. Sampling of

```
integer jitter = 0.02*clk_period*1ps;
integer seed=10*1ps;

//Generate Clock and Clock jitter
always begin
  #((clk_period/2*1ps)+$dist_uniform(seed,-jitter,jitter));
  clk = ~clk;
 end
```

Fig. 8 Generation of clock jitter for model accuracy improvement

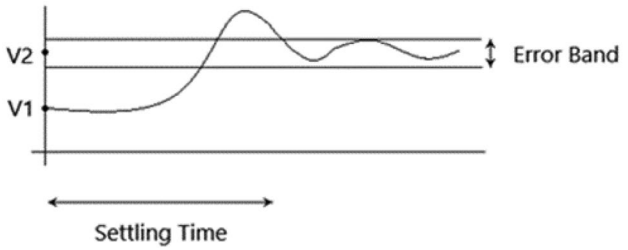

Fig. 9 Example of settling time with error band

the analog input does not always happen at the exact moment that is needed. This non-ideal element causes a conversion error proportional to the slope of the signal, which is generally referred to as jitter at the sampling moment, as mentioned in [21]. The design embeds a ± 2% clock jitter, out of the period T clock jitter, as shown in Fig. 8. Clock jitter is embedded in the model by adding/subtracting a delay to/from the clock period with the usage of a uniform distribution function.

The sample-and-hold circuit is a basic element of the proposed SAR ADC circuit. It includes an "always @ (posedge clk or negedge rst)" block, which claims that at every positive edge of the clock or at every negative edge of the reset, the modulation function of the S/H is executed. In simple words, the value of the input voltage (nettype real) is kept until the starting flag is set. Settling time is the most significant non-ideal factor in today's S/H circuits (SI memory cells). It describes the time it takes for the output to achieve its final value within a given settling error. In Fig. 9, the behavior of a standard S/H settling behavior according to [21] is illustrated. Figures 10 and 11 displays part of the proposed S/H circuit, where a settling delay equal to 10% of the clock period [53] and a ± 2% error band [54] are inserted. The impact of these two non-idealities in terms of SNR on ADC's performance is significant. More specifically, clock

```
module sample_and_hold
...
generate
if(RESET_TYPE=="ASYNC")
begin : gen_async
 always @(posedge clk or negedge rst_)
   if (!rst_)
   begin
    #(0.1*clk_period*1ps)
    vout<=(vin *(0.0001*($urandom_range(9800,10200))));
   end
   else if(start && finito)
   begin
    #(0.1*clk_period*1ps)
    vout<=(vin *(0.0001*($urandom_range(9800,10200))));
end
...
endmodule
```

Fig. 10 Part of the proposed sample-and-hold circuit

```
module dac

always_comb
 begin
  dac_out=0;
  for (int i = 0; i <n; i++)
  begin
   dac_out = dac_out + sar_register_out[i]*(vref/(2**(n-i)));
  end

end

endmodule
```

Fig. 11 Part of the proposed DAC circuit

jitter adds timing uncertainty, reducing SNR and settling noise results from incomplete DAC settling, adding noise, so the proposed non-ideal ADC model has a degradation of 6 dB, having a total SNR of 19.84 dB.

The analog comparator is the simplest module of the proposed design. It includes two nettype real inputs and a digital output. If the input voltage is higher or equal to the voltage approximation from DAC, the output holds a value of digital '1', or else '0'.

In Fig. 12, the proposed SAR register is shown. The displayed code represents the asynchronous reset modes. More specifically, for every positive edge of the clock or for every negative edge of the reset, it performs the following code. If the reset is in '0' then the variables are initialized. The most important bit is initialized as '1' and the rest as '0'. For the following cycles, it checks each individual bit by turning it into '1'. If the input voltage is higher that the digital output of the module then it keeps the bit '1', else it changes it back to '0'. After that, it finishes checking every bit, then it proceeds to output the digital approximation of the input voltage and activate the finish flag. For the synchronous reset, it follows the exact same procedure, but it only changes the ticking of the "always" block for every positive edge of the clock.

4 Basic UVM Testbench Architecture

UVM represents a methodology employed for verifying the functionality of SoC designs through the utilization of SystemVerilog. This approach involves the creation of verification elements and transaction-level modeling (TLM) to achieve efficient component interconnection [11, 14, 16, 19, 48, 49, 52, 55]. Moreover, it leverages a foundational class library to facilitate the development of Universal Verification Components (UVCs) and testbench modules, thereby augmenting the adaptability and potential for reuse within the verification environment.

Verification of a specific design, module, interface, protocol, or logic function necessitates a compilation of elements

```
logic[1:0]  flag;

generate
if(RESET_TYPE=="ASYNC")
begin : gen_async
always @(posedge clk or negedge rst_)
  if (!rst_)
  begin
   flag<=n-1;
   sar_register_out<=0;
   sar_register_out[n-1]<=1;
   finito<=1'b0;
   sar_result<=0;
  end
  else
  begin
        if(!finito)
        begin
            if(flag!=0 )
            begin
             if(comp_out==0)
             begin
             sar_register_out[flag]<=1'b0;
             end
             flag<=flag-1;
             sar_register_out[flag-1]<=1'b1
            end
            else
            begin
             if(comp_out==0)
             begin
             sar_register_out[flag]<=1'b0;
             end
            finito<=1'b1;
            end
         end
```

Fig. 12 Part of the proposed SAR register

related to stimulus, monitoring, verification, and coverage. Within this context, a UVC encompasses a crucial suite of fundamental software and associated components. UVCs are delineated through a series of classes present in the UVM library, each class signifying a significant building block. Nonetheless, the most pivotal element within this assembly is responsible for relaying the stimulus produced by the testbench to the DUT. Furthermore, the construction of a UVC can be contingent on the type of the DUT and may involve one or more agents. Agents, in turn, can be affixed to specific interfaces with the aim of evaluating the DUT's functionality.

The schematic representation of a general UVM testbench architecture is presented in Fig. 13. More specifically, the Top is where all verification components, interfaces, and the DUT are instantiated. The Test is responsible for defining the test scenario within the testbench, enabling the examination and validation of specific features and functionalities. Within the Test class, the environment, environment configuration properties, and various other attributes can be found. In order to execute the test scenario, various sequences within the Test are created and initiated. When dealing with highly intricate

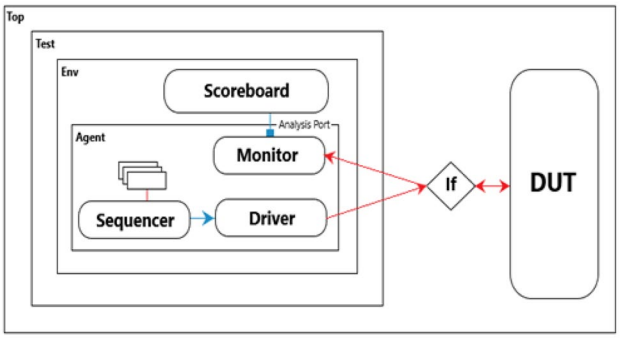

Fig. 13 Example of a basic UVM testbench architecture

designs, it becomes necessary to run multiple tests. Now, regarding the Environment, it serves as the primary container class. It encompasses one or more agents, each dedicated to different interfaces, and a shared scoreboard. Additionally, it may include other components such as a top-level monitor, checker, predictor, or even smaller sub-environments.

Concerning the Agent, it can be employed in either an active or passive way. When operating actively, it takes on the responsibility of generating the necessary stimuli and transmitting them to the DUT via an interface. An active agent typically encompasses a sequencer, a driver, a monitor, and a configuration object. Conversely, when in passive mode, the agent's role is limited to merely sampling the signals emanating from the DUT, and it consists of just a solitary monitor. In a different context, the Sequencer serves as the stimulant creator, orchestrating the flow of transactions (or sequence items) as class objects between the sequences and the driver. In addition, the Driver is an active component that continuously receives sequence items from the sequencer and propels them towards the DUT through the interface.

The Monitor functions as a passive module responsible for gathering signals from the DUT interface and translating them into transaction-level activities. Furthermore, it conveys these transactions to the evaluation components within the environment, such as the scoreboard, using its analysis ports. The Scoreboard plays a pivotal role in validating the DUT's functionality by comparing its responses to the expected values, typically sourced from a reference model, also known as predictor. The scoreboard receives transactions through its analysis outputs from the monitors. A Sequence Item represents the smallest structural unit of data in the verification process, comprising various fields of essential information required for stimulus generation. The Sequence, in turn, serves as the component responsible for generating a sequence of these items in an organized manner.

Apart from the basic architecture, there are three major UVM constructs that can play a vital role in the effective verification of a design. Sequences are fundamental in UVM for generating stimuli. They provide a flexible mechanism to create a variety of test scenarios, allowing for randomization and reuse across different testbenches. This modularity helps in achieving comprehensive test coverage by simulating numerous operational conditions. In addition, coverage metrics are vital in UVM for guiding the verification process. They help identify parts of the design that have not been tested, ensuring thorough validation. Functional coverage, code coverage, and assertion coverage are types of metrics that provide insights into test completeness and effectiveness. Finally, assertions in UVM are used for functional checking, verifying that the design behaves correctly under specified conditions. They monitor the design's response to stimuli and can catch errors that might be missed during standard simulation. Assertions are embedded in the testbench to continuously check for compliance with the design specifications.

5 Proposed SAR ADC Verification Architecture

In this section, the proposed UVM-based verification architecture for the SAR ADC RNM model is described, along with functional coverage.

5.1 UVM-based Verification Architecture

Figure 14 shows the proposed UVM-based testbench architecture for the four-bit SAR-ADC RNM model. The top module declares the SAR ADC instance, the interface, and runs various test cases. Each test instantiates the verification environment, the most important part of the testbench.

A UVC is used in the verification environment. In general, the choice between using a single agent or multiple agents in the UVC involves several trade-offs. A single

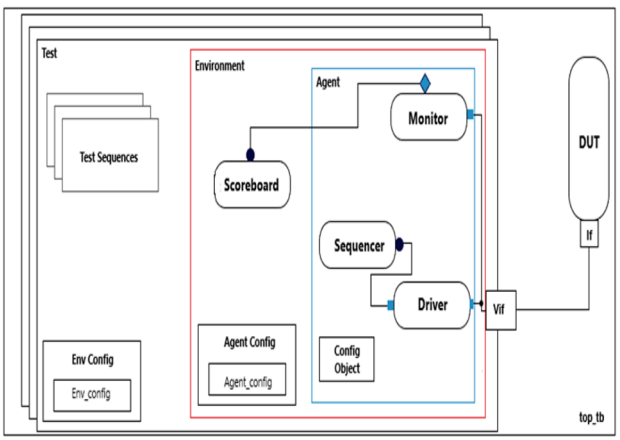

Fig. 14 Proposed UVM-based mixed-signal verification architecture

agent simplifies the testbench architecture, making it easier to manage and maintain. However, it may limit the ability to test complex interactions within the design. In multiple agents case, it allows for more detailed and thorough testing of complex interactions and scenarios, providing a higher level of coverage and validation. The disadvantage is increased complexity in managing multiple agents, which can make the testbench more challenging to develop and maintain. In this case, for the 4-bit SAR ADC, the focus is on simplicity and basic functionality, so a single agent will suffice. Thus, the UVC, which is a single agent, is included in the verification environment, along with a scoreboard and various agent configuration properties. A driver, a sequencer, and a monitor are included in the proposed agent, as well as a configuration object. The sequencer acts as a controller between the test sequences and the driver. Each sequence produces a collection of sequence items (see Fig. 15) that the driver receives. A sequence item consists of the logic-type four-bit variable of sar_result, and both the input and reference voltage, which are declared as real variables. Moreover, the different variables are limited to follow the testbench architecture's requirements.

The sequence item packets are sent to the DUT by the driver via its interface handle when it receives them, as seen in Fig. 16. The monitor transforms the signals from the DUT into sequence objects by sampling them via its device handle. The sequence items are then sent to the scoreboard via the monitor's analysis port, where they are compared to the data traffic coming from the internal predictor. The predictor is a subscriber, which receives the same stimuli as the DUT through the monitor and produces transactions with known good responses. The core of the predictor is a simplified SAR ADC model. As shown in Fig. 17, the predictor receives the input and computes the expected result of the SAR ADC model. Finally, the scoreboard's functionality is implemented with immediate assertions.

The interface serves as the bridge connecting two distinct realms, namely, the dynamic and static domains, as defined by the UVM-based classes in the mixed-signal verification testbench and the SAR ADC—DUT module, respectively. Virtual interface handles are employed to establish connection between different elements of the testbench. Within the interface, both real-type and logic-type inputs and outputs are declared, enabling communication between the DUT and the testbench. Moreover, analog concurrent assertions are integrated into the interface to monitor the actions of the SAR ADC throughout the simulation run. Figure 18 provides a visual representation of a segment of the proposed interface.

5.2 Functional Coverage

Coverage-driven testing of a mixed-signal architecture relies heavily on functional coverage. It is the metric that determines how well a testbench covers a design's functionality. The attainment of functional coverage is accomplished by specifying dedicated coverage models tailored to the intended DUT. A covergroup, which is a set of coverpoints, forms the coverage model in the proposed mixed-signal

```
class sar_adc_4bit_seq_item extends uvm_sequence_item;
  rand real vref;
  rand real vin;
  logic [3:0] sar_result;

  constraint vref_c{vref inside {[0.0:5.0]};}
  constraint vin_c{vin inside {[0.0:5.0]};}
  constraint vin_diff{vin <= vref;}

  `uvm_object_utils_begin(sar_adc_4bit_seq_item)
    `uvm_field_real (vref, UVM_ALL_ON)
    `uvm_field_real (vin, UVM_ALL_ON)
    `uvm_field_int  (sar_result, UVM_ALL_ON)
  `uvm_object_utils_end

  function new (string name = "sar_adc_4bit_seq_item");
    super.new(name);
  endfunction : new
endclass : sar_adc_4bit_seq_item
```

Fig. 15 Sequence item class of the proposed verification architecture

```
task run_phase(uvm_phase phase);
  @(posedge sar_adc_if.rst_);
  @(posedge sar_adc_if.clk);

  forever begin
  req = sar_adc_4bit_seq_item::type_id::create("req");

  seq_item_port.get_next_item(req);
  `uvm_info(get_type_name(), "Driver got item.", UVM_MEDIUM)

  $cast(cloned_req, req.clone());

  @(posedge sar_adc_if.clk);
  sar_adc_if.vref <= 1;
  sar_adc_if.vin <= 1;

  sar_adc_if.vin    <= req.vin;
  sar_adc_if.vref   <= req.vref;
  @(posedge sar_adc_if.clk);

  `uvm_info(get_type_name(), "Driver finished item.", UVM_LOW)
   seq_item_port.item_done();
  end
  endtask
  ...
```

Fig. 16 Part of the run_phase task from the proposed driver class

```
// Predictor functionality - Golden Reference Model
virtual function void write_sar_adc_4bit_master(sar_adc_4bit_seq_item packet);

 logic [3:0] sar_result_temp=1000;
 logic flag=0;
 real dac_temp=0;

 for (int i = 3; i <=0; i--)
 begin
   sar_result_temp[i]=1'b1;
   for (int j = 0; j <4; i++)
   begin
   dac_temp = dac_temp +sar_result_temp[j]*(packet.vref/(2**(3-j)));
   end
   if (packet.vin>=dac_temp)flag=1;
   else flag=0;
   if (flag==0)sar_result_temp[i]=1'b0;
   end

 packet.sar_result <= sar_result_temp;
 data_queue.push_back(packet);
  endfunction : write_sar_adc_4bit_master
```

Fig. 17 Part of the proposed scoreboard class: predictor—golden reference model

verification architecture. Every coverpoint is associated with a particular aspect of the SAR ADC's functionality that needs to be encompassed.

Figure 19 displays part of the intended covergroup for the real number model of the SAR ADC. Every rising edge of the reference clock signal is encompassed by every coverpoint, as are all real-valued and logic-type signals from the interface. Besides, it comprises diverse coverage bins, each linked to a range of values, and it delineates the range of real-valued signals. However, for a more practical approach, it is advantageous to subdivide this range into smaller value sets. This division allows for the evaluation of whether the input values,

```
//Actual signals
input logic clk;
input logic rst_;
input logic start;

//Nettype real variable
output real  vin;
output real  vref;

output logic [3:0] sar_result;

//Concurent assertios
assert property(@(posedge clk)  rst_==0  |-> sar_result==0);
assert property(@(posedge clk)  rst_==0  |-> vin==0 & vref==0);
assert property(@(posedge clk)  start==0 ##5 1 |-> start==1);
assert property(@(posedge clk)  start==1 ##5 1 |-> start==0);
assert property(@(posedge clk)  rst_==0  |-> start==0);
assert property(@(posedge clk)  vin<=vref);
assert property(@(posedge clk) (rst_==0) ##3 (rst_==1) ##5 1 |->sar_result!=0);
```

Fig. 18 Part of the interface, where analog concurrent assertions are added to check that analog signals behave as expected during simulation

```
covergroup cg @(posedge clk);

    c1: coverpoint vref {
          type_option.real_interval = 0.1;
           bins b_c1[50] = {[0.0 : 5.0]};
         }
   c2: coverpoint vin  {
          type_option.real_interval = 0.1;
          bins b_c2[50] = {[0.0 : 5.0]};
       }

    c1c2: cross c1,c2;


endgroup: cg

cg cov_cg = new();
```

Fig. 19 Covergroup for the analog signals vref and vin. It is utilized to measure how well specific aspects of the analog signal behavior are being tested

during a simulation run, fall within these specific sets. This involves the creation of a vector of bins for each coverpoint, with the "type_option.real_interval" field being set to 0.1 to specify the desired range of values assigned to each bin.

As a consequence, for each coverpoint, different bin vector sizes are declared. Furthermore, cross coverage is used for all coverpoints in pairs to see if a specific combination of voltage input values is used. “c1c2: cross c1, c2,” tests the degree at which the bins of c1 coverpoint cross with the bins of c2 coverpoint.

6 Experimental Results

In this section, initially a simulation performance analysis takes place, where the proposed SAR ADC SystemVerilog RNM model is compared against a reference model in Verilog-AMS. Additionally, the presented UVM-based verification architecture is evaluated for its verification efficiency.

Table 1 Simulation Performance of the Proposed SAR ADC RNM Model

	Proposed SAR ADC SystemVerilog RNM Model	Verilog-AMS Model [23]
Simulation actual execution time	0.5 s	20 s
Percentage (%) of time compared to Verilog-AMS	2.5%	100%

```
// Clock and reset
initial begin
  //randomize clock period in range 50MHz - 500MHz
  clk_period = $urandom_range(2000, 20000);
  clk_period = 2000;

  //indicate there is an activity on rst_n
  rst_n=1;
  #1ns;
  rst_n=0;

  //put rst_n=1
  #(3*((5*clk_period* 1ps)/5)*1ns);
  rst_n = 1;
end

integer jitter = 0.02*clk_period*1ps;
integer seed=10*1ps;

//Generate Clock and Clock jitter
always begin
  #((clk_period/2*1ps)+$dist_uniform(seed,-jitter,jitter));
  clk = ~clk;
 end
```

Fig. 20 Randomization of clock frequency in the top-level module

6.1 Simulation Performance of the SAR ADC RNM Model

To demonstrate the advantages of the suggested design simulation, Table 1 presents a comparison of the simulation durations needed for a single conversion at 6-bit using both the proposed SAR ADC SystemVerilog RNM model and a Verilog-AMS model [23].

The simulation time for the SystemVerilog RNM model is significantly lower (0.5 s) compared to the Verilog-AMS model (20 s) [23]. The proposed SAR ADC SystemVerilog RNM model demonstrates a notable improvement in simulation efficiency compared to its Verilog-AMS counterpart, as evidenced by the significantly reduced simulation time.

6.2 Verification Efficiency of the Proposed UVM-based Verification Architecture

Cadence Xcelium was employed to integrate and simulate the suggested UVM-based verification structure for the real-number model of the SAR ADC, as outlined in Section 3. The verification architecture was formulated using SystemVerilog with RNM. Numerous test scenarios were created to comprehensively assess the functionalities of the SAR ADC.

The testbench was crafted using SystemVerilog in adherence to the UVM methodology, featuring target output system clock frequencies spanning from 50 to 500 MHz. The reference clock frequency was randomized in each simulation run, as depicted in Fig. 20.

The suggested verification framework successfully validated the accurate functionality of the SAR ADC model under all scenarios. Furthermore, 99% was deemed to be an appropriate standard of functional coverage. The SAR ADC's functionality was covered at a rate of over 99.6%, according to the coverage report. The high coverage level leads to greater confidence in the verification process, reducing the risk of undetected bugs and ensuring robust design validation. By achieving high coverage levels across functional, code, and assertion metrics, the reliability of the verification process is significantly enhanced, ensuring that the design meets its specifications and behaves as expected under all conditions.

The simulation waveform of the presented verification testbench is illustrated in Fig. 21 with a reference clock frequency at 500 MHz. The test scenario in this case involves the creation of four thousand sequences. The number of sequences is set to be as high as possible, in order to completely test the SAR ADC's functionality. The run phase of this test case is

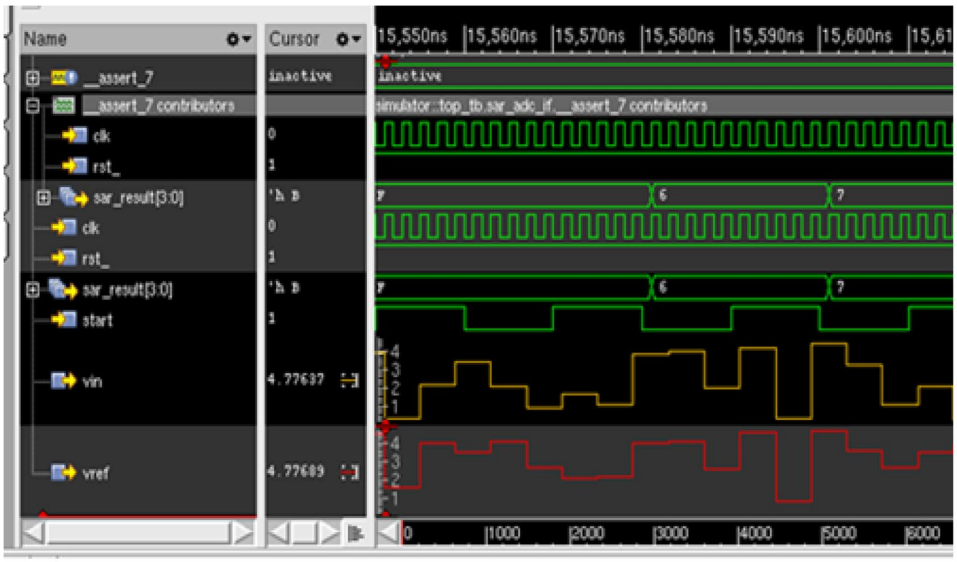

Fig. 21 Simulation results having reference clock frequency set to 500 MHz

```
task run_phase(uvm_phase phase);
  sar_adc_4bit_base_seq m_seq;

  phase.raise_objection(this, "default_sequence_test");
    for(int i=0; i<4000; i++) begin
      m_seq = sar_adc_4bit_base_seq::type_id::create("m_seq");

      if(m_seq.randomize())
        `uvm_info(get_type_name(), "m_seq just got randomized!", UVM_LOW)
      fork
        m_seq.start(sar_adc_env.sar_adc_agent.m_sequencer);

      join
    end
    #300;
  phase.drop_objection(this, "default_sequence_test");
endtask: run_phase

ndclass : default sequence test
```

Fig. 22 Run phase of the UVM-based test class

Table 2 Coverage Analysis for Different Numbers of Test Sequences

Clock Frequency set to 500MHz					
Number of test sequences	500	1000	2000	4000	8000
Coverage analysis (%)	99.63%	99.68%	99.74%	99.78%	99.79%

presented in Fig. 22, where four thousand sequences are created, randomized and then started via the sequencer.

The simulation results demonstrated that the proposed UVM-based testbench architecture can achieve high mixed-signal DUT verification performance. In addition, coverage analysis of the test scenario shown in Fig. 23 pointed out the proposed testbench efficiency. Moreover, as shown in Fig. 23, the coverage model demonstrates 99.78% coverage completeness of SAR ADC functionality.

Coverage analysis results for the proposed testbench module are shown in Table 2 for cases where the clock frequency

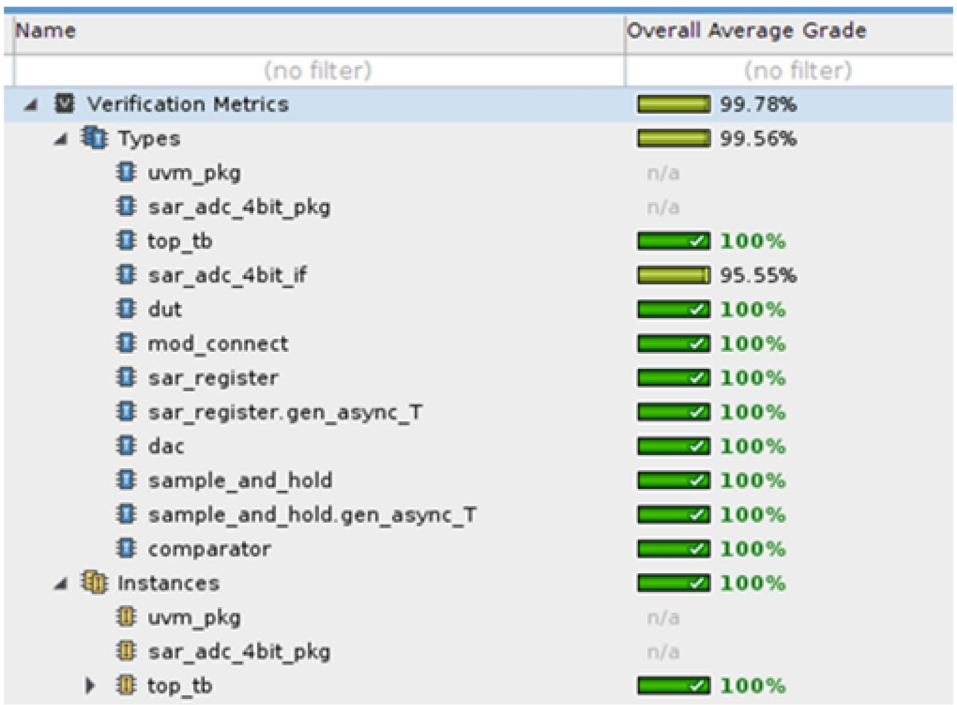

Fig. 23 Coverage analysis for a test case scenario

is set to 500 MHz and the number of sequences initiated from the test is modified. Coverage analysis metrics are derived for 500, 1000, 2000 4000, and 8000 test sequences, respectively. Each test was completed successfully in all cases, with coverage analysis levels exceeding 99% in all cases, with some increases for larger numbers of sequences. This means that the SAR ADC's normal operation was completely tested for various numbers of test sequences, without showing any errors.

7 Discussion and Conclusion

This paper presents a four-bit SAR ADC real number model implemented in SystemVerilog. It takes advantage of real number modeling, which allows for quick simulation of mixed-signal designs. In contrast to other mixed-signal or analog modeling approaches, the experiments revealed that the proposed SystemVerilog RNM SAR ADC model has a high simulation efficiency and reasonable accuracy. In addition, for the SAR ADC real number model, a UVM-based mixed-signal verification architecture was presented. It exploits the advantages of UVM's high verification performance, as well as SystemVerilog – RNM's mixed-signal design capabilities. Compared to conventional mixed-signal or analog verification approaches, the proposed testbench design demonstrated superior verification efficacy while in parallel decreasing simulation time by an exponential factor. The presented testbench architecture employs constrained-random real-valued stimulus generation, coverage metrics, multiple test scenarios, and analog assertions to accomplish substantial verification process improvements. These characteristics contribute to high testbench modularity, interoperability, and efficiency.

Conclusively, it's worth pointing out that the presented verification architecture can be extended to any form of mixed-signal SoC design block-level verification. Exploring an extension to this research involves examining a white-box verification framework designed to assess the operational integrity of the internal components within the SAR ADC real number model. Regarding the limitations, the proposed approach may face challenges in scaling to more complex ADC architectures due to increased verification demands and complexity in stimulus generation and checking. In addition, while the approach is tailored for SAR ADCs, its application to other mixed-signal circuits may require significant modifications to handle different interaction dynamics and performance characteristics. On the other hand, future research could explore techniques to enhance scalability, such as hierarchical verification or partitioning strategies. Adapting the approach to other mixed-signal designs, such as PLLs or DACs, could also expand its utility. Finally, emerging trends like machine learning could be leveraged to automate and improve verification efficiency and coverage, potentially addressing complex verification scenarios more effectively.

Funding No funding was received for conducting this study. The authors have no relevant financial or non-financial interests to disclose.

Data Availability The data that support the findings of this study are not openly available due to reasons of sensitivity and are available from the corresponding author upon reasonable request.

References

1. Accelera (2011) Universal Verification Methodology (UVM) 1.1 User's Guide, https://accellera.org/, (Accessed 10 Aug 2021)
2. Balasubramanian S, Hardee P (2013) Solutions for Mixed Signal SoC Verification Using Real Number Models. Cadence Design Systems. https://www.cadence.com/. (Accessed 10 Aug 2021)
3. Hartong W, Cranston S (2009) Real Valued Modeling for Mixed Signal Simulation, Cadence Design Systems
4. Vogelsong R, Osman AH, Mohamed M (2015) Practical RNM with SystemVerilog. Proc. CDNLive 2015
5. Bombieri N, Ebeid E, Fummi F et al (2013) On the Reuse of Heterogeneous IPs into SysML Models for Integration Validation. J Electron Test 29:647–667
6. Chen J, Henrie M (2012) Mixed-Signal Methodology Guide, Lulu, https://www.lulu.com/, (Accessed 10 Aug 2021)
7. Cross D (2023) Real Valued Models for Verification of Silicon Photonic Systems", Proc. (2023) IEEE Photonics Conference (IPC). Orlando, FL, USA 1–2. https://doi.org/10.1109/IPC57732.2023.10360544
8. Georgoulopoulos N, Hatzopoulos A (2021) Parameterizable Real Number Models for Mixed-Signal Designs Using SystemVerilog. J Electron Test 37:685–700
9. Maurice M, Dessouky M, Salem A (2023) Increasing the Modeling Accuracy of an Analog PLL Device Executed With an Event-Driven Simulator. IEEE Access 11:79721–79738. https://doi.org/10.1109/ACCESS.2023.3299227
10. Barros JS, Schulz VH, Lettnin DV (2018) An Adaptive Closed-Loop Verification Approach in UVM-SystemC for AMS Circuits", Proc. DV (2018) 31st Symposium on Integrated Circuits and Systems Design (SBCCI). Bento Gonçalves, Brazil 1–6. https://doi.org/10.1109/SBCCI.2018.8533229
11. Biswal BP, Singh A, Singh B (2017) Cache coherency controller verification IP using SystemVerilog Assertions (SVA) and Universal Verification Methodologies (UVM),. Proc. 2017 11th International Conference on Intelligent Systems and Control (ISCO), Coimbatore, India, pp. 21–24. https://doi.org/10.1109/ISCO.2017.7855984
12. Deepak R, Parvathy SJ, Arya S, Babu V (2020) Regression Based Mixed Signal Verification of an Ambient Light Sensor Interface", (2020) IEEE International Symposium on Circuits and Systems (ISCAS). Seville, Spain 1–4. https://doi.org/10.1109/ISCAS45731.2020.9181161
13. Dharani M, Bharathi M, Padmaja N, Praveena K (2023) Design and Verification process of Combinational Adder using UVM Methodology. Proc. 2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS), Bangalore, India, pp. 359–362. https://doi.org/10.1109/ICAECIS58353.2023.10170273
14. Doshi NK, Suryawanshi S, Kumar GN (2016) Development of generic verification environment based on UVM with case study on HMC controller. Proc. 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication

Technology (RTEICT), Bangalore, India, pp. 550–553. https://doi.org/10.1109/RTEICT.2016.7807882
15. Fu Y, Huang K, Zhang L, Liu F (2020) A System Function Verification Flow For Mixed-signal SoC", Proc. F (2020) 7th International Forum on Electrical Engineering and Automation (IFEEA). Hefei, China 738–741. https://doi.org/10.1109/IFEEA51475.2020.00157
16. Gayathri M (2016) A SV-UVM framework for Verification of SGMII IP Core with reusable AXI to WB Bridge UVC, Proc. 3rd International Conference on Advanced Computing and Communication Systems, ICACCS
17. Khalifa K (2017) Extendable Generic Base Verification Architecture for Flash Memory Controllers Based on UVM, Proc. IEEE 21st International Conference on Computer Supported Cooperative Work in Design, CSCWD
18. Lim BC, Horowitz M (2019) An Analog Model Template Library: Simplifying Chip-Level, Mixed-Signal Design Verification. IEEE Trans Very Large Scale Integr (VLSI) Syst 27(1):193–204. https://doi.org/10.1109/TVLSI.2018.2873387
19. Salah K (2014) A UVM-Based Smart Functional Verification Platform: Concepts, Pros, Cons, and Opportunities, Proc. 9th International Design and Test Symposium, DTS
20. Zivkovic VA, Palazzi M, ChuenAlvan Lam M, Isager M (2022) AMS Test Vector Generation using AMS Verification and IEEE P1687.2. Proc. 2022 IEEE European Test Symposium (ETS), Barcelona, Spain, pp. 1-4. https://doi.org/10.1109/ETS54262.2022.9810471
21. Hanfoug S, Bouguechal NE, Barra S (2014) Behavioral non-ideal Model of 8-bit Current-Mode Successive Approximation Registers ADC by using Simulink, Proc. Int J u- and e-Service, Sci Tech 8(3):85–102
22. Santhanalakshmi M, Yasoda K (2015) Verilog-A implementation of energy-efficient SAR ADCs for biomedical application, Proc. 19th International Symposium on VLSI Design and Test, VDAT
23. Carsten W (2013) Method of modeling analog circuits in verilog for mixed-signal design simulations, Proc. 2013 European Conference on Circuit Theory and Design, ECCTD
24. Georgoulopoulos N, Hatzopoulos A (2018) Efficiency Evaluation of a SystemVerilog-based Real Number Model, Proc. 7th International Conference on Modern Circuits and Systems Technologies, MOCAST
25. Narayanan R, Zaki MH, Tahar S (2010) Using Stochastic Differential Equation for Verification of Noise in Analog/RF Circuits. J Electron Test 26:97–109
26. Subrahmaniyan Radhakrishnan S, Ozev S (2011) Adaptive Modeling of Analog/RF Circuits for Efficient Fault Response Evaluation. J Electron Test 27:465–476
27. Aisola R et al (1996) Verilog-A Language Reference Manual, Open Verilog International, OVI
28. Belay YA, Cabrini A, Torelli G (2016) A comprehensive Verilog-A behavioral model of Spin-Transfer Torque memory cell," Proc. 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), Lisbon, Portugal, pp. 1–4. https://doi.org/10.1109/PRIME.2016.7519522
29. Chandrasekaran S et al (2009) Verilog-AMS Language Reference Manual, Accellera, https://www.accellera.org/, (Accessed 10 Aug 2021)
30. FitzPatrick D, Miller I (2003) Analog Behavioral Modeling with the Verilog-A Language, Kluwer Academic Publishers
31. Jimenez-Dominguez E, Gonzalez-Diaz VR, Rodriguez-Dominguez AM (2016) Behavioral model of a VCO varying its Kvco with Verilog-A," Proc. 13th International Conference on Power Electronics (CIEP), Guanajuato, Mexico, pp. 70–74. https://doi.org/10.1109/CIEP.2016.7530733
32. Kuo P-Y, Sie L-F (2015) Analyze the behavior model based on Verilog-A for Sallen-Key low-pass filter. Proc. 2015 IEEE International Conference on Consumer Electronics - Taiwan, Taipei, Taiwan, pp. 460–461. https://doi.org/10.1109/ICCE-TW.2015.7216998
33. Lena D, Grosso M, Bocca A, Macii A, Rinaudo S (2016) A compact IGBT electro-thermal model in Verilog-A for fast system-level simulation. Proc. IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society, Florence, Italy, pp. 3793–3798. https://doi.org/10.1109/IECON.2016.7793376
34. Cadence Design Systems (2015) Verilog-AMS Real Valued Modeling Guide
35. Cheng K-H, Jou CF (2003) 2.4 GHz CMOS VCO design with Verilog-AMS, Proceedings of the 12th International Conference on Fuzzy Systems, FUZZ
36. Cundert KS, Zinke O (2004) The Designer's Guide to Verilog AMS, Kluwer Academic Publishers
37. Hujer M, Manasek R, O'Mahony J, Feerick P, Barry M, Walsh B (2006) Nanometer Wireless Transceiver Modeling using Verilog-AMS and SystemC. Proc. IEEE International Behavioral Modeling and Simulation Workshop, San Jose, CA, USA, pp. 150–155. https://doi.org/10.1109/BMAS.2006.283486
38. Jakobsson A, Serban A, Gong S (2015) Implementation of Quantized-State System Models for a PLL Loop Filter Using Verilog-AMS. IEEE Trans Circuits Syst I Regul Pap 62(3):680–688. https://doi.org/10.1109/TCSI.2014.2377411
39. Lian-xi L, Yin-tang Y, Zhang-ming Z, Yani L (2005) Design of PLL system based Verilog-AMS behavior models," Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, 2005., Suzhou, China, pp. 67–70. https://doi.org/10.1109/IWVDVT.2005.1504466
40. Godambe NJ, Richard Shi CJ (1998) "Behavioral Level Noise Modeling and Jitter Simulation of Phase-Locked Loops with Faults Using VHDL-AMS." J Electron Test 13:7–17
41. Hernández FAI, Canesin CA (2012) Electrical Power Distribution System modeling with VHDL-AMS for the construction of a Real-Time Digital Simulator using FPGAS devices, Proc. 10th IEEE International Conference on Industry Applications, IAS
42. Pecheux F, Lallement C, Vachoux A (2005) VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems. IEEE Trans Comput Aided Des Integr Circuits Syst 24(2):204–225. https://doi.org/10.1109/TCAD.2004.841071
43. Szermer M, Daniel M, Napieralski A (2003) Modeling and simulation sigma-delta analog to digital converters using VHDL-AMS," The Experience of Designing and Application of CAD Systems in Microelectronics, 2003. CADSM 2003. Proceedings of the 7th International Conference., Slavske, Ukraine, pp. 331–333. https://doi.org/10.1109/CADSM.2003.1255082
44. Georgoulopoulos N, Hatzopoulos A (2017) Real number modeling of a flash ADC using SystemVerilog, Proc. Panhellenic Conference on Electronics and Telecommunications, PACET
45. Lotfy A, Farooq SFS, Wang QS, Yaldiz S, Mosalikanti P, Kurd N (2015) A system-verilog behavioral model for PLLs for pre-silicon validation and top-down design methodology", Proc. (2015) IEEE Custom Integrated Circuits Conference (CICC). San Jose, CA, USA 1–4. https://doi.org/10.1109/CICC.2015.7338432
46. Shera E, Wegener C (2015) Buck converter modeling in System-Verilog for verification and virtual test applications, Proc. IEEE 20th International Mixed-Signals Testing Workshop, IMSTW
47. Yang X, Niu X, Fan J, Choi C (2013) Mixed-signal System-on-a-Chip (SoC) verification based on SystemVerilog model. Proc. 45th Southeastern Symposium on System Theory, Waco, TX, USA, pp. 17–21. https://doi.org/10.1109/SSST.2013.6524952
48. Bromley J (2013) If SystemVerilog is so good, why do we need the UVM? Sharing responsibilities between libraries and the core language, Proceedings of the 2013 Forum on specification and Design Languages, FDL

49. Georgoulopoulos N, Giannou I, Hatzopoulos A (2018) UVM-Based Verification of a Mixed-Signal Design Using SystemVerilog. Proc. 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), Platja d'Aro, Spain, pp. 97–102. https://doi.org/10.1109/PATMOS.2018.8464148
50. Simon S, Bhat D, Rath A, Kirscher J, Maurer L (2017) Coverage-driven mixed-signal verification of smart power ICs in a UVM environment," Proc. 22nd IEEE European Test Symposium (ETS), Limassol, Cyprus, pp. 1–6. https://doi.org/10.1109/ETS.2017.7968237
51. Simon S, Karaca Ö, Kirscher J, Rath A, Pelz G, Maurer L (2016) Safety-oriented mixed-signal verification of automotive power devices in a UVM environment. Proc. 13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Lisbon, Portugal, pp. 1–4. https://doi.org/10.1109/SMACD.2016.7520718
52. Yun Y-N, Kim J-B, Kim N-D, Min B (2011) Beyond UVM for practical SoC verification. Proc. B 2011 International SoC Design Conference, Jeju. Korea (South) 158–162. https://doi.org/10.1109/ISOCC.2011.6138671
53. Nicollini G, Confalonieri P, Senderowicz D (1989) A fully differential sample-and-hold circuit for high-speed applications. IEEE J Solid-State Circuits JSSC
54. Tay TT, Mareels I, Moore JB (1998) High performance control. https://www.researchgate.net/publication/235683246_High_Performance_Control, pp 93
55. Moskala M, Kloczko P, Cieplucha M, Pleskacz W (2015) UVM-based verification of bluetooth low energy controlle. Proc. IEEE 18th International Symposium on Design and Diagnostics of Electronic Circuits Systems, DDECS

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

Nikolaos Georgoulopoulos is a Postdoctoral Researcher at the Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki. His research interests include digital IC design and testing (full RTL-to-GDSII flow implementation), ASIC design verification, digital and mixed-signal IC modeling, 3D IC testing.

Theodora Mamali is a researcher at the Aristotle University of Thessaloniki's Department of Electrical and Computer Engineering. Her research interests include machine learning in verification, power management in ultra-low power processors, mixed signal verification using digital on top of SoCs, and functional verification methodologies.

Alkiviadis (Alkis) Hatzopoulos is a full professor at the Department of Electrical and Computer Engineering, Aristotle University of Thessaloniki. His current research interests include: modeling, design and testing of integrated circuits and systems (analog, mixed-signal, high-frequency), three dimensional integrated circuits (3D ICs), electronic communication circuits, instrumentation electronics, bioelectronics systems, space electronics.