Parameterized Virtual Testing and Simulation of Verilog-AMS Behavioral Models

Thorben Schey*, Khaled Karoonlatifi†, Michael Weyrich* and Andrey Morozov*

*Institute of Industrial Automation and Software Engineering,
University of Stuttgart
Stuttgart, Germany
Email: {first.last}@ias.uni-stuttgart.de
†Advantest Europe GmbH
Böblingen, Germany
Email: {first.last}@advantest.com

***Abstract*—Traditional approaches to developing test programs for Analog Mixed-Signal (AMS) circuits are characterized by time constraints and high costs and often depend heavily on physical chip validation. These tests require the use of specific test inputs, a process that relies significantly on expert knowledge due to the complexity of the AMS circuit under test (CUT). Therefore, there is a growing demand for a virtual test framework that facilitates the execution of test programs prior to chip fabrication and provides early insights into their validity. This paper presents a simulation framework that uses architecture independent behavioral models of AMS circuits with customizable error characteristics in Verilog-AMS. The proposed framework also incorporates models of test equipment and signal transmission paths. Exemplary simulation results using the Siemens Questa simulator that demonstrate the feasibility of this approach are provided.**

***Index Terms*—virtual test, framework, behavioral model, Verilog-AMS, simulation, analog mixed-signal, error model**

I. Introduction

In semiconductor manufacturing it is essential to ensure the correct functionality of the Integrated Circuits (ICs). Therefore, the chips are extensively tested after the production phase using specialized testing equipment. This process involves the execution of a test program on the testers, consisting of test sequences with defined inputs and expected outputs, tailored to evaluate specific functions of the chip. When testing the Analog Mixed-Signal (AMS) circuits of the Chip Under Test (CUT), the design process and the validation of the test program is especially complex and challenging.

This is why creating effective test programs requires expertise and time from test engineers. In addition, verification of these programs depends on the availability of physical chips, further complicating the process and delaying validation until the first chips finished production. Prior to chip manufacturing, an urgent need exists in the industry for more efficient methods of validating test programs in order to achieve time and cost savings. To address this limitation and enable pre-tapeout validation of the test programs, a simulative approach is pursued [1]. Although simulations are commonly used during chip design, they are mainly performed as SPICE simulations

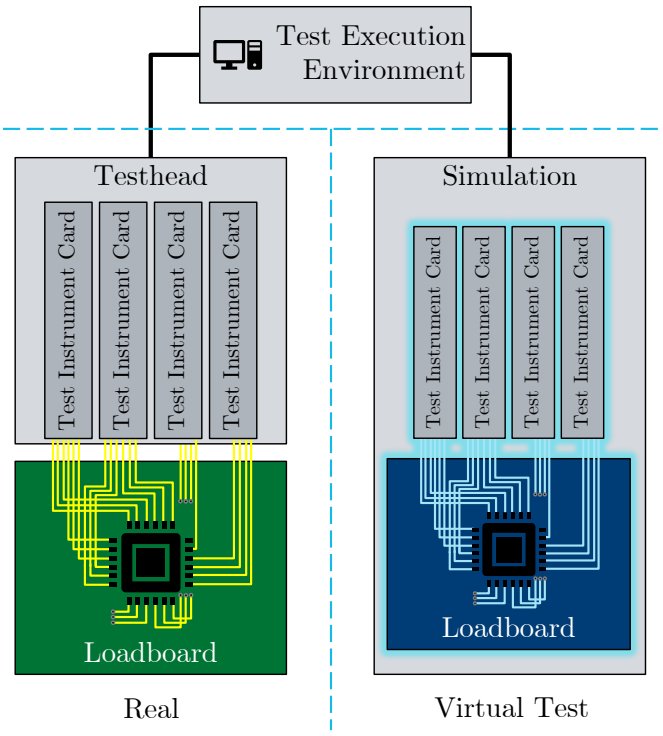

Fig. 1. Schematic comparison between real and virtual test.

when it comes to AMS circuits. As they are modeling the circuits at transistor level, they provide highly precise results, but also require extensive computational resources and time. Consequently, their use for test program development and validation in a virtual testing approach is not desired. Even though obtaining realistic simulation results is ideal, there is no immediate requirement to model the chip based on its individual circuit elements and architecture. Hence an alternative approach based on behavioral models can be used. For that, the CUTs behavior is modeled in a Hardware Description Language (HDL) such as Verilog-AMS with additional support for analog signals [2].

In comparison to SPICE models which can be exported directly from the chip design software, the behavioral models must be created in an additional step that is not automated. However, their potential to accelerate the validation of test programs by enabling simulation-based verification prior to tapeout represents a promising way to increase efficiency in the semiconductor industry. Therefore, a standardized approach is needed to make the use of behavioral models more accessible and faster.

Contributions: This paper presents a framework for virtual testing based on behavioral models. The architecture-independent models have an ideal behavior that can be adapted to typically occurring errors. We demonstrate the seamless integration of the behavioral models into our framework by using standardized wrappers. Furthermore, we introduce standardized input sources that support automatic testbench creation. Finally, we discuss the framework's interface to commercial simulators and the essential data post-processing procedures to ensure applicability in real-world scenarios.

The organization of this paper is as follows. In Section II we give an overview of the state of the art. An example for a behavioral model of an Analog-to-Digital Converter (ADC) with parametrizable errors is presented in Section III. In Section IV the architecture of the framework is addressed, including the standardized model wrapper, input source models, output measurement models, and the simulation interface. Simulation results of the models generated with the framework are presented in Section V.

II. State of the Art

In this section, we first provide an overview of the testing process of ICs and AMS circuits in particular. Second, we explain AMS circuit testing using an ADC as an example. Third, we give background information about existing work on simulative approaches and behavioral modeling of AMS circuits.

A. The Chip Testing Process

The used automated chip test equipment includes a test head, mainframe, and a computer [3]. While the main power supply and temperature regulators are in the mainframe, the measurement equipment is located in the test head in the form of exchangeable instrument cards (see Figure 1). Since not only one type of chip is tested on the tester, a loadboard with the corresponding chip socket is developed for each CUT, which establishes the connections between the test instruments and the chip. The connected computer runs a test program specifically designed for the CUT. During testing, the computer sends the test patterns to the instruments memory and receives the measurement results afterwards for further analysis [1], [3].

The test program consists of program blocks tailored to specific parts of the circuits. When working with AMS circuits, the program code must be fine-tuned by the test engineers. In contrast to digital logic tests, where the results are binary (pass or fail), more nuanced results can be expected when testing AMS circuits as parametric shifts caused by process variations occur. Since parametric shifts can lie within the tolerance range, they do not necessarily result in a fail binning, but can still be considered to be functioning properly [3].

B. Testing of an ADC

An Analog-to-Digital Converter (ADC) is a fundamental AMS component that acts as an interface element between analog and digital signals. As such, the ADC converts an analog input voltage into a digital representation at its output. In this process, the input voltage is mapped to specific digital codes, with the lowest and highest digital codes representing the negative and positive reference voltages applied to the ADC respectively.

The basic idea of ADC testing is to check the correct assignment of output codes to input voltage intervals. Consequently, a voltage ramp starting from the negative reference voltage to the positive reference voltage is a common choice for the input voltage stimuli (see Figure 4). This approach ensures that all possible input voltages are applied sequentially, allowing for the verification of each digital code. Based on the combinations of applied input voltage and corresponding measured output codes, the prominent error characteristics of an ADC can be derived:

- Offset error,
- Gain error,
- Differential Non-Linearity (DNL),
- Integral Non-Linearity (INL),
- Missing codes.

To save time when calculating these error characteristics, the histogram method is used, in which the number of sample hits per code are counted. The individual errors can be calculated using irregularities in the histogram. In the case of a ramp input voltage, subsequent weighting as with sine functions is avoided. Nevertheless, the slope of the ramp needs still to be adjusted so that there are sufficient hits per code for the histogram results to be reliable.

C. Simulation of AMS circuits

To be able to make initial predictions before the chips are physically present and thus before the first real measurements are taken, simulations are used [1]. Already during the design phase of chips, simulations are executed in order to be able to test their functionality. These SPICE simulations (Simulation Program with Integrated Circuit Emphasis) are carried out at transistor level. By considering individual transistor characteristics they offer precise and detailed results with significant computational costs. Another method for simulating AMS circuits are behavioral simulations using Hardware Description Languages (HDLs) such as Verilog-AMS. In contrast to SPICE simulations, behavioral simulations focus on the response of circuits at a high-level and are therefore suitable for system-level analysis. The level of detail of the behavioral model determines the accuracy of the results and the computing time required. Verilog-AMS is particularly characterized by its ability to describe both analog and digital behavior within

the same HDL. As an extension of the regular Verilog, which can only represent digital aspects, it also makes it possible to model and simulate AMS components by integrating analog signals in the form of the electrical discipline [2]. A comparison between a behavioral model approach based on different HDLs with a regular SPICE simulation shows that the computational requirements are significantly less when using HDLs [4], [5].

Models for AMS components have existed for some time, starting with the development of AMS extensions for HDLs [2]. Mostly, behavioral models are tailored to specific architectures [6] or in the case of architecture-independent models focus primarily on idealized behavior [2], [7] and neglect the incorporation of errors.

More recently, a proposal in [8] introduces a generic error behavior model targeting the behavior of MOSFETs. This model includes four modes: (i) free (no error), (ii) open drain, and shorts ((iii) gate-drain, (iv) drain-source). As such, the model refers to a lower level and does not describe errors of system components such as ADCs or Digital-to-Analog Converters (DACs). Regardless of the abstraction level, [9] recommends an efficient Monte Carlo approach for determining error parameters based on SPICE simulations. Once determined, these parameters can be integrated into behavioral models.

However, not only models are required, but also the surrounding testbench, which provides the input stimuli in a simulative approach. In [10], a method for transferring circuit diagrams into testbenches for behavioral models is described. Although the testbench generation is not automated, the method suggests the reuse of standard behavioral models. This approach has also been pursued by [8], [11].

A concept for virtual test instruments is described in [12], which contains specifications for test instruments and approaches for analyzing test duration and utilization. This idea is similarly extended in [13], where instrument models are integrated into a test framework. Furthermore, [14] presents a pre-silicon verification environment that enables virtual test development, demonstrating how behavioral models can be effectively integrated into early validation stages.

Building on these foundations, our work extends the current state of research by proposing a framework for virtual testing that integrates behavioral models with an interface to test execution environments. Unlike previous approaches, our framework is designed to be architecture-independent and incorporates parameterizable error behavior to support test program validation.

III. ADC Behavioral Model

In this section, we discuss the behavioral approach to modeling a generic ADC. This model is an example of how ideal behavioral models can be extended with parameterizable error effects to represent system components.

A. Ports and Ideal Behavior

The behavioral model of the ADC comprises the inputs ground (GND) as a general reference for the other voltages, the positive and negative reference voltage, the to be converted analog voltage input and an input for the digital clock. When a rising edge of the clock signal is detected, the ADC performs a conversion of the analog input voltage. The resulting digital output code is determined by the position of the input voltage relative to the interval defined by the reference voltages.

B. Parameterizable Errors

To increase the realism of the model, we introduce parameterizable errors that are commonly observed in ADCs. These errors already mentioned in Section II-B (offset and gain errors, DNL, INL, and missing codes), are adjustable by parameters that define maximum error limits. Within these limits, errors are integrated randomly during the simulation of the model. By including these errors in the model, we can simulate the effects of real errors on ADC performance and then utilize them in the validation of the test program.

C. Additional useful Parameters

In addition to the error parameters, the model considers the propagation time for input and output signals as well as the bit width of the ADC. While the propagation time parameters enable a more precise representation of the time behavior, the bit width parameter can be used to adjust the resolution of the ADC.

IV. The Framework

Throughout this section, we present our framework for virtual testing using parametrizable behavioral models. In Figure 2, the schematic structure of the framework is shown. In the following, the individual components are described in further detail.

A. Standardized Model Wrapper

In order to integrate the generic behavioral models written in Verilog-AMS into the framework, a wrapper approach was chosen. Even if it does not achieve the full flexibility of a parser for Verilog-AMS files, the standardized structure of the wrapper provides an optimal structure for template-based integration.

Each wrapper holds a list of the parameters that exist in the associated behavioral model. In addition, there is another list with all ports including their properties, such as signal type, bus width and also the information on whether it is an input, output or inout. Both the parameters and the port information can be accessed via an interface to the automated testbench generation, which also allows the creation of the model instance code.

B. Standardized Input Source Models

Based on the considerations on test instruments from [13], we propose parameterizable standard input sources. For digital input sources we suggest a clock module, a heaviside step function and module for freely adjustable sequences of values. Similar functionalities are also required for analog voltage sources. In addition to an ordinary DC voltage source, it is also practical to include a ramp and sine function.

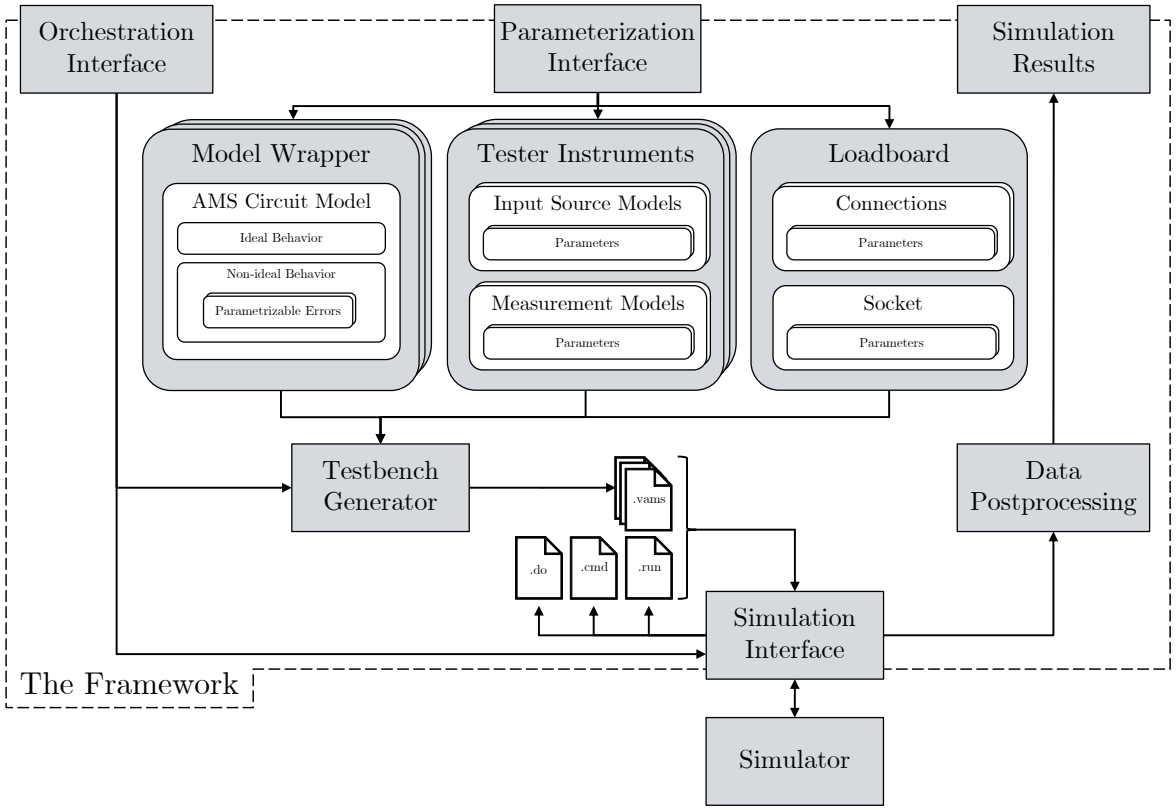

Fig. 2. The schematic of our virtual testing framework with parametrizable behavioral models of AMS circuits, tester instruments and loadboard.

Furthermore, we recommend a programmable source module that follows a sequence of voltage values based on timestamp-voltage pairs. Three different variations of a programmable source module were implemented (see Figure 3): In hard step mode, the voltages to be applied are set immediately. In the case of linear transitions, the voltage source interprets the timestamp-voltage pairs as points along a continuous linear function. This results in straight-line segments connecting adjacent points, forming linear voltage curves. The third mode considers the last voltage value in time as the target value. Taking into account a parameterizable maximum slope value, the voltage source strives for this target value. As the adjustable timestamp-voltage pairs can not be set via parameters due to their varying number, a slightly modified wrapper approach was used here, in which the Verilog-AMS code is generated at runtime allowing all points to be included.

C. Output Measurement Models

Similar to the input sources, it likewise seems practical to design the measurement instruments as models. The different properties of the measurement instruments are described with parameters within a standard model. In this case, it is not yet absolutely clear whether the measurement instruments will be completely implemented as behavioral models in Verilog-AMS and added to the testbench. Alternatively, the desired sampling behavior can be imitated afterwards with the help of data post-processing algorithms. A consideration of the different approaches is discussed later in Figure 8.

D. Automatic Testbench Generation

For the automatic testbench generation, the instances of the model wrappers including the individual parameters are required. After the wrapper instances have been referenced in lists using commands, they can be connected to each other via their ports. Initially, these are stored as one-to-one connections, but they are merged to nets when the automatic testbench generation is performed. During this merging process, all connected ports are checked for signal type and bus width compatibility. As part of the testbench code generation, signals corresponding to the identified nets are automatically generated and named.

Moreover, the instances of the Verilog-AMS modules are inserted with the previously specified parameters and connected to the generated signals. With the signal patterns embedded inside the standardized input source models, any signal pattern descriptions in the testbench can be avoided.

In addition to the integration of the CUT behavioral models and the input sources, models of loadboard signal transmission lines and sockets can also be incorporated into the testbench generation. When configuring the signal transmission lines and sockets, it is possible to either parameterize them individually

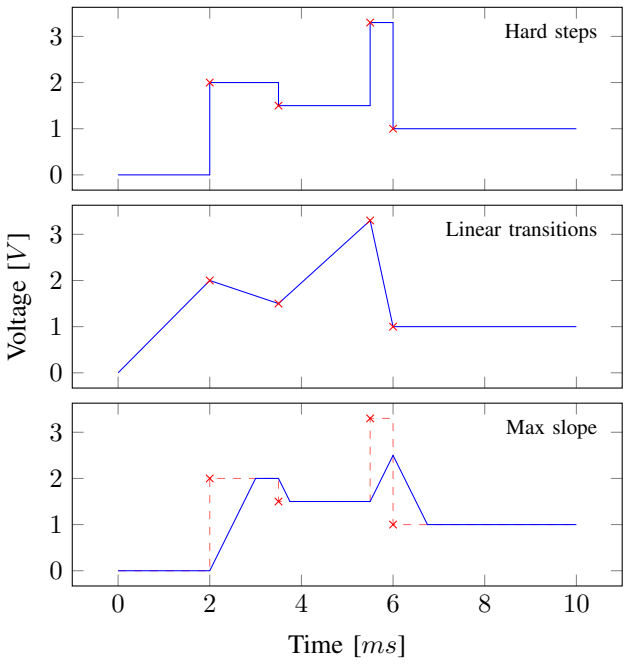

Fig. 3. Three modes of the user programmable voltage source module. The red indicators are the given timestamp-voltage pairs.

for each connection, or to use a universal parameterization for automatic placement for every connection.

E. Simulation Interface

The simulation interface serves as a link between the framework components described previously and external Verilog-AMS simulators. As the available simulators differ in usage, this component is to be tailored to the chosen simulator and provide the required files in addition to the according CLI commands. The Siemens Questa ADMS simulator was used in the work presented here.

In the required files, information such as simulator accuracy, simulated time, and the signals to be recorded must be defined. Since signals are generated as part of the automatic testbench generation and are therefore not known by name, the ports to be observed are passed to the simulation interface. The necessary signal associated with the port is then identified automatically.

Once the simulation has been carried out, the interface is also used to read in the resulting data and convert it into table form for handover to the test program execution environment. During this process, the previously performed mapping of ports to signals is taken into account so that referencing via the ports is possible.

F. Data Post-Processing

After the simulation results have been received and parsed by the simulation interface, further post-processing is necessary. As the simulator only outputs values based on event triggers or on value changes, the data points are unevenly distributed over time. In comparison with real measurement instruments, which sample the signals at a fixed clock frequency, this uneven distribution leads to incompatibility issues in the test execution environment. For this reason, artificial sampling is implemented in the framework, simulating the sampling process of real measurement instruments. Based on a parameterizable sampling frequency, the simulation data is filled in for missing time points: In the case of digital signals, the last known value is repeated. For analog signals, interpolation is performed between the known values. All simulation data points lying between two artificial sampling points are subsequently removed.

G. Application

In order to connect the presented framework to a test execution environment, we plan to use parsers for file transfers. In form of a standardized data exchange format, all parameterizations of the test instruments and the CUT as well as the netlist describing the connections should be included. Such a data file allows all subsequent steps to be carried out fully automated using the previously described components.

V. Experimental Results

The ADC model described in Section III is used in this section as a representative example of a behavioral CUT model and thus serves as the basis for the provided simulation results. Since the ideal behaviour provides the foundation for the applied parameterizable errors, the simulation results of an ideal, error-free 8-bit ADC with a voltage ramp as input are shown in Figure 4. Its ideal behavior with steps of equal width are visible. Due to the high number of steps in an 8bit ADC, the resolution is subsequently reduced in order to better visualize the error effects.

Compared to the ideal behavior, the simulation results show the desired erroneous behavior (see Figure 5) when the gain and offset errors are introduced via the wrapper of the ADC model.

The important error effects DNL and thus INL, as well as missing codes can also be set and observed (see Figure 6). Not only can the individual shifted code edges be seen, which lead to different step widths, but also a missing code in the case of code number seven.

Our transmission line models are based on the behavior of passive electrical components, providing a more accurate representation of signal propagation. In the simulation results (see Figure 7), these effects can be observed.

Finally, the two approaches for replicating the behavior of the tester measurement instruments were evaluated. In particular, the focus lies on obtaining a uniformly sampled output signal. Figure 8 shows both the simulation results of the integrated model approach in Verilog-AMS and pure data post-processing using artificial sampling. It is evident that the data post-processing works as desired, while the integrated behavior model approach, which was implemented using a sample and hold element, provides additional undesirable samples. Although all points where the internal clock of the sample and hold element caused the simulator to record a measurement are

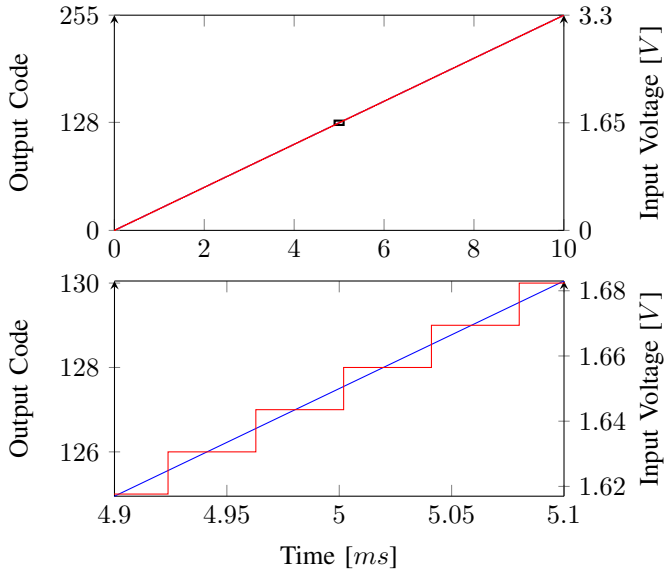

Fig. 4. Simulation results of a 8-bit ADC showing ideal behavior. The input (blue) represents a voltage ramp that ranges from the negative ($0V$) to the positive ($3.3V$) ADC reference voltage. The resulting output codes are shown in red. The clock frequency was set to $256kHz$ with 10 conversion per step.

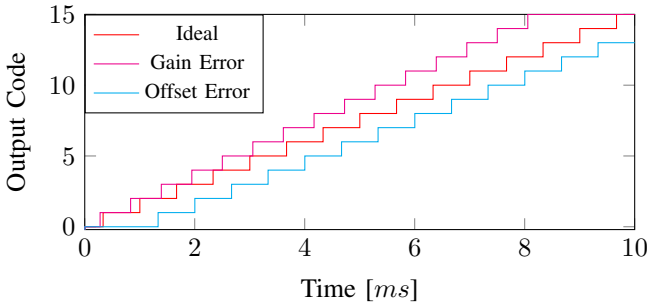

Fig. 5. Simulation results of a 4-bit ADC showing offset (cyan) and gain (magenta) errors. The ideal code edges are shown in red for comparison. The clock frequency was set to $16kHz$ with 10 conversion per step.

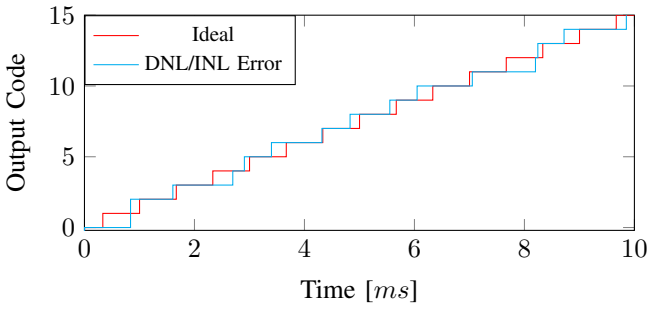

Fig. 6. Simulation results of a 4-bit ADC with DNL, INL and missing code errors in the cyan colored plot. The ideal code edges are shown in red for comparison. The clock frequency was set to $16kHz$ with 10 conversion per step.

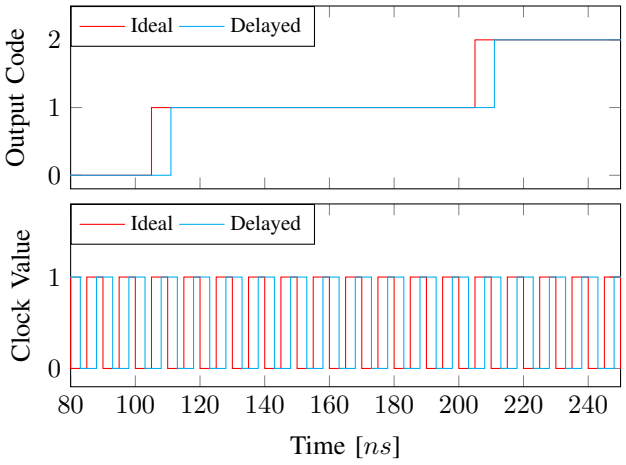

Fig. 7. Simulation results of an ideal 4-bit ADC with signal transmission line delays (cyan). The ideal code edges are shown in red for comparison. The clock frequency was set to $100MHz$ with 10 conversion per step.

correct, the unwanted additional points are caused by events triggered by other components in the simulation. We therefore recommend using data post-processing to emulate the sampling and employing a behavioral model in the simulation to address other circuit-related effects of the measurement equipment.

All results were obtained with simulations carried out entirely using the presented framework. For this purpose, only the parameter values and connections between the standard components were specified. The testbench generation, simulator control, and data transfer as well as simulation result retrieval were executed fully automated.

VI. Conclusion

In this paper, we have demonstrated a behavioral model-based approach to the simulation of analog mixed-signal circuits with parameterizable errors for validating of test programs. For this purpose, a virtual test framework was presented that allows generic behavioral models to be integrated and typical tester instruments to be simulated automatically. The key components of the framework include standardized model wrappers for behavioral models, standardized input sources, automatic testbench generation, simulation interfaces and data post-processing. Combined, these components enable a seamless integration of parameterizable generic behavioral models into the virtual test environment and allow a simulative validation process of test programs. The functionality of this framework in simulating the behavior of AMS circuits was demonstrated by the experimental results presented. In this context, the ideal and the erroneous behavior of an ADC simulated with the framework was shown as an example. Additionally, the incorporation of signal transmission line properties based on an behavioral model was demonstrated as part of the automatic testbench generation.

By providing a standardized and automated approach for virtual testing within this framework, significant advantages are offered compared to the conventional test program development process. Test engineers can validate test programs

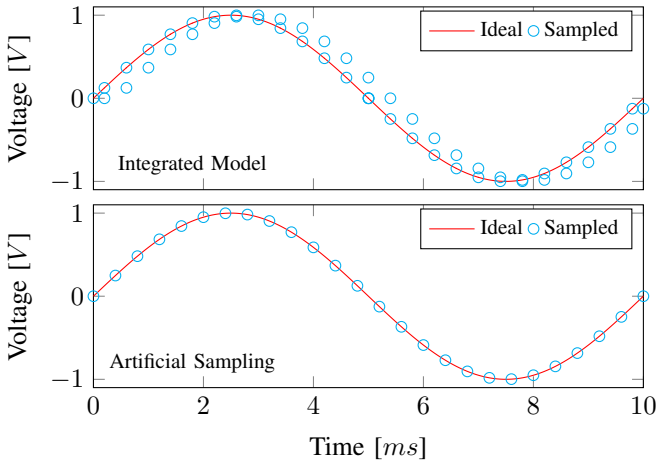

Fig. 8. Comparison of simulation results between the integrated model approach and artificial sampling in form of data post-processing. The input sine wave (red) was sampled with a frequency of $2.5kHz$.

more efficiently, reducing reliance on physical chips and accelerating the overall test development process. As semiconductor technologies continue to advance, we expect such virtual test frameworks to have a significant impact on test program validation times, thereby playing a major role in ensuring the reliability and performance of integrated circuits.

Acknowledgements

This research was supported by Advantest as part of the Graduate School "Intelligent Methods for Test and Reliability" (GS-IMTR) at the University of Stuttgart.

References

[1] L. S. Milor, "A tutorial introduction to research on analog and mixed-signal circuit testing," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 10, pp. 1389–1407, 1998.

[2] I. Miller and T. Cassagnes, "Verilog-a and verilog-ams provides a new dimension in modeling and simulation," in *Proceedings of the 2000 Third IEEE International Caracas Conference on Devices, Circuits and Systems (Cat. No. 00TH8474)*. IEEE, 2000, pp. C49–1.

[3] M. Burns, G. W. Roberts, and F. Taenzler, *An introduction to mixed-signal IC test and measurement*. Oxford university press New York, 2001, vol. 2001.

[4] R. Narayanan, N. Abbasi, M. Zaki, G. Al Sammane, and S. Tahar, "On the simulation performance of contemporary ams hardware description languages," in *2008 International Conference on Microelectronics*. IEEE, 2008, pp. 361–364.

[5] R. O. Peruzzi and J. Medero, "A systematic approach to creating behavioral models," 2015.

[6] V. Y. Ponce-Hinestroza and V. R. Gonzalez-Diaz, "System-level behavioral model of a 12-bit 1.5-bit per stage pipelined adc based on verilog®=-ams," in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2018, pp. 301–304.

[7] R. O. Peruzzi, "Verification of digitally calibrated analog systems with verilog-ams behavioral models," in *2006 IEEE International Behavioral Modeling and Simulation Workshop*. IEEE, 2006, pp. 7–16.

[8] N. Dall'Ora, S. Azam, E. Fraccaroli, R. Gillon, and F. Fummi, "Verilog-a implementation of generic defect templates for analog fault injection," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, 2023, pp. 477–481.

[9] H.-G. Stratigopoulos and S. Sunter, "Efficient monte carlo-based analog parametric fault modelling," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*. IEEE, 2014, pp. 1–6.

[10] C. Wegener, "Method of modeling analog circuits in verilog for mixed-signal design simulations," in *2013 European Conference on Circuit Theory and Design (ECCTD)*. IEEE, 2013, pp. 1–5.

[11] B. C. Lim and M. Horowitz, "An analog model template library: Simplifying chip-level, mixed-signal design verification," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 1, pp. 193–204, 2018.

[12] M. Miegler and W. Wolz, "Development of test programs in a virtual test environment," in *Proceedings of 14th VLSI Test Symposium*. IEEE, 1996, pp. 99–103.

[13] L. M. van de Logt, V. A. Zivkovic, and I. H. van Baast, "Model-driven ams test setup validation tool prepared for ieee p1687. 2," in *2019 IEEE European Test Symposium (ETS)*. IEEE, 2019, pp. 1–6.

[14] E. Aderholz, Q. Atol, B. Baptist, R. Holzner, R. Ignacio, V. Kamanuri, A. Kun, K. Ma, B. Mariacher, O. Pfabigan *et al.*, "Virtual test development using pre-silicon verification environment," in *2024 IEEE International Test Conference (ITC)*. IEEE, 2024, pp. 445–450.