Virtual Test for mixed-signal Circuits: Digital Twin based Development of Post-Silicon Tests

Milestone Report – GS-IMTR

Thorben Schey

August 30, 2024

Advisor / main examiner: Jun.-Prof. Dr.-Ing. Andrey Morozov

Co-examiner: Prof. Dr.-Ing. Bin Yang

Mentor: Khaled Karoonlatifi

1 Introduction

This technical report outlines the progress made, current research activities, and future directions for Project 11 (P11) within the Graduate School - Intelligent Methods for Test and Reliability (GS-IMTR). P11 started in April 2023 and is scheduled for completion in March 2026. The project is titled "Virtual Test for Mixed-Signal Circuits: Digital Twin-based Development of Post-Silicon Tests". The goal of this project is to develop and enhance virtual testing methods using digital twin technologies, with a focus on improving the development and validation of post-silicon tests for Analog Mixed-Signal (AMS) circuits.

1.1 Brief Review

The project began in April 2023 with a comprehensive literature review focusing on the latest advances in virtual testing and modeling for AMS circuits. After a comprehensive analysis, it was decided in September 2023 that the project would implement a virtual test system based on behavioral models. This approach was chosen because it provides a balance between simulation accuracy and computational efficiency, making it ideal for the project's objective.

In October 2023, Siemens Questa was acquired as the simulator for Verilog-AMS due to its robust support for mixed-signal simulations and compatibility with the project's requirements. Afterwards, the simulation infrastructure was set up in November 2023. That same month, work began on creating the first behavioral models, focusing on Analog-to-Digital Converters (ADCs), which serve as an example for an AMS Chip-Under-Test (CUT) for the project's virtual test framework.

The development of the virtual test framework itself began in January 2024 with the aim of simulating behavioral models of the CUT, transmission lines and test instruments. In March 2024, an algorithm was designed and investigated to maintain the Integral Non-Linearity (INL) during the ADC model instance creation process for error injection. This was followed in April 2024 by an idea for integrating our clocked noise approach into the behavioral models in order to significantly improve simulation times.

In July 2024, a cooperation with P18 was launched in which a behavioral model of a time-to-digital (TDC) converter is being built and simulated in the existing framework. The focus lies on the generation of synthetic test data with consideration of process variations. These are then used as part of an artificial intelligence based tuning process.

Most recently, in August 2024, the development of test instrument models was started, which are expected to model the behavior of the signal sources and measuring instruments of the Advantest testers.

During this time, we successfully submitted an extended abstract to ETS24 and received a poster presentation slot at ITC24. Other paper submissions were less successful and are currently being revised and extended.

1.2 Report Outline

This report describes the state of the art in Section 2. Section 3 discusses the framework that has been conceptualized. Section 4 is dedicated to the additional research findings: The INL algorithm and the clocked noise approach for noise simulation. Finally, the plan for the second half of P11 is considered in Section 5.

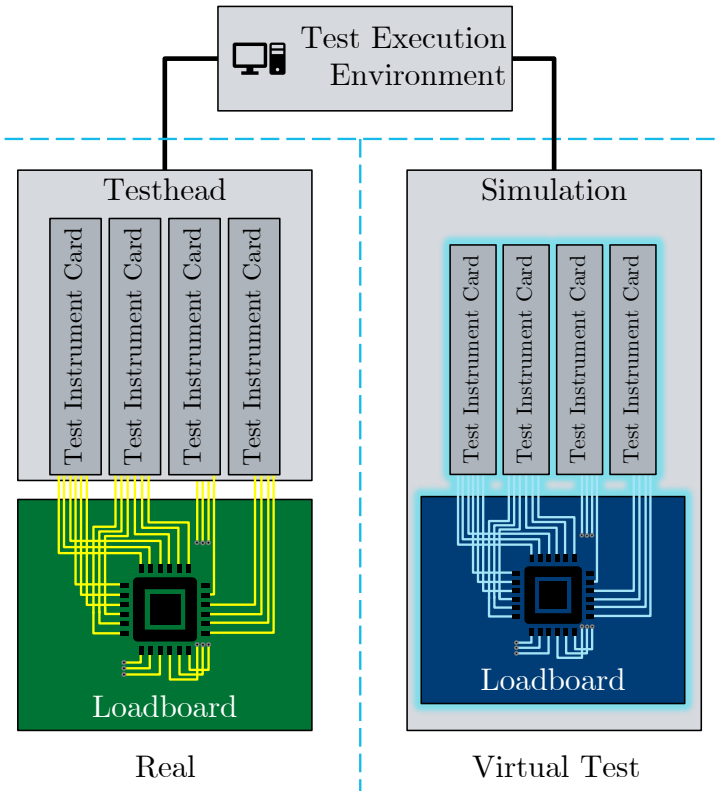

Figure 1: Schematic comparison between real and virtual test.

2 Background

In semiconductor manufacturing, ensuring the correct functionality of Integrated Circuits (ICs) is essential. After production, chips undergo extensive testing using specialized equipment to validate their functionality. This process involves executing a test program on Automated Test Equipment (ATE), which consists of specific test sequences with defined inputs and expected outputs tailored to evaluate the functions of the Chip Under Test (CUT) [1].

For Analog Mixed-Signal (AMS) circuits, the design and validation of these test programs present additional challenges. Due to the complexity of AMS circuits, creating effective test programs requires significant expertise and time from test engineers. Furthermore, the verification of these programs depends on the availability of physical chips, delaying the validation process until the chips have completed production. This traditional approach not only consumes time but also leads to high costs during both the development and execution of the test programs [2].

Prior to chip manufacturing, the industry faces an urgent need for more efficient methods of validating test programs. To address this limitation and enable pre-tapeout validation, a simulative approach is pursued [2]. Although simulations are commonly used during chip design, they are typically performed as SPICE simulations for AMS circuits. These simulations model the circuits at the transistor level, providing highly precise results but requiring extensive computational resources and time [2]. Consequently, their use for test program development and validation in a virtual testing approach is not desired.

While obtaining realistic simulation results is ideal, it is not necessary to model the chip based on its individual circuit elements and architecture. Instead, a more efficient alternative involves using behavioral models, which simulate the CUT's behavior at a higher level of abstraction. These models can be implemented using Hardware Description Languages (HDLs) like Verilog-AMS, which supports both analog and digital signals [3]. Comparative studies have shown that using HDLs for behavioral modeling significantly reduces computational requirements compared to SPICE simulations [4, 5].

Behavioral models for AMS components have been around since the development of AMS extensions for HDLs [3]. These models are often tailored to specific architectures, and in cases where they are architecture-independent, they usually focus on idealized behavior, neglecting the inclusion of errors [6].

Recent work has introduced generic error behavior models, such as those targeting MOSFETs, which include various error modes. However, these models are lower-level and do not describe errors in system components like ADCs or Digital-to-Analog Converters (DACs) [7].

Behavioral models, unlike SPICE models that can be automatically generated from chip design software, must be created manually. However, their potential to accelerate the validation of test programs by enabling simulation-based verification before tapeout represents a promising way to increase efficiency in the semiconductor industry [4]. As such, a standardized approach to the creation and use of behavioral models is needed to make them more accessible and faster to implement. However, not only models of the CUT are needed, but also models of the test infrastructure.

This need for surrounding testbenches that provide input stimuli in a simulative approach has been recognized. Methods for transferring circuit diagrams into testbenches for behavioral models have been proposed, though they are not automated and rely on reusing standard behavioral models [7, 8, 9]. Concepts for virtual test instruments, which include specifications and behavior are also being explored [10, 11].

Automated test equipment used in chip testing includes a test head, mainframe, and a computer [1]. The mainframe contains the primary power supply and temperature regulators, while the measurement equipment is located in the test head in the form of exchangeable instrument cards. A loadboard with a corresponding chip socket is developed for each CUT to connect the CUT to the test instruments (see Figure 2). The connected computer runs a test program specifically designed for the CUT, sending test patterns to the instrument's memory and receiving measurement results for further analysis [1, 2].

The test program consists of blocks tailored to specific parts of the circuits. When working with AMS circuits, test engineers must fine-tune the program code. Unlike digital logic tests, where results are binary (pass or fail), testing AMS circuits involves more nuanced outcomes due to parametric shifts caused by process variations. These shifts can fall within acceptable tolerance ranges and may not necessarily result in a failed test, even if some parameters deviate from their ideal values [1].

One example of AMS circuit testing is the Analog-to-Digital Converter (ADC), which converts an analog input voltage into a digital output code. The basic idea of ADC testing is to verify the correct assignment of output codes to input voltage intervals. A common approach is to apply a voltage ramp from the negative to the positive reference voltage, ensuring that all possible input voltages are tested sequentially. This method allows verification of each digital code, and the resulting data can be used to calculate error characteristics such as offset error, gain error, Differential Non-Linearity (DNL), Integral Non-Linearity (INL), and missing codes [6].

The corresponding test program would then perform a histogram analysis, where the number of sample hits per code are counted. The individual errors are then calculated based on irregularities in the histogram. Although the ramp's slope must be adjusted to ensure sufficient hits per code, this method avoids the need for weighting functions as used in sine wave testing [6].

In this context, our work proposes a framework for virtual testing based on behavioral models with an interface to test execution environments. This framework integrates architecture-independent models with parameterizable error behavior for test program validation, extending current research on virtual testing which is only focusing on the optimization of ICs.

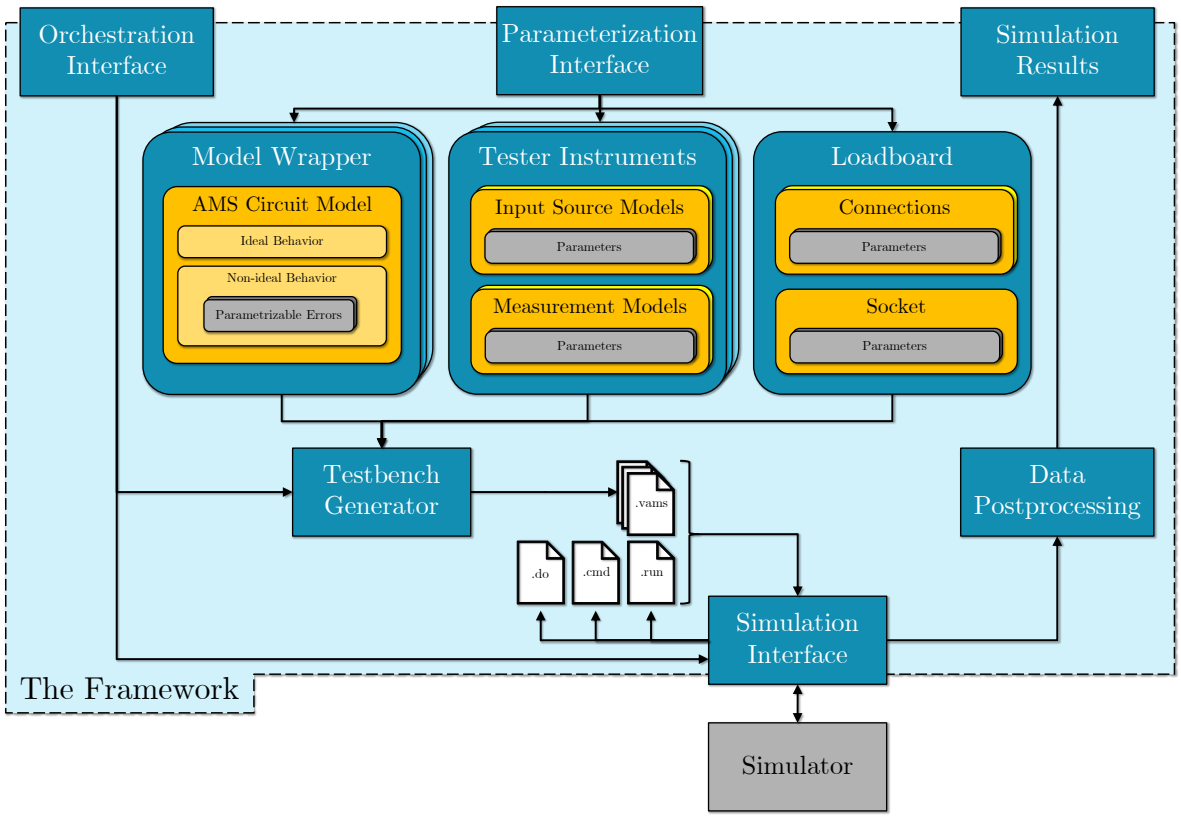

Figure 2: The schematic of our virtual testing framework with parametrizable behavioral models of AMS circuits, tester instruments and loadboard.

3 The Virtual Testing Framework

The framework for virtual testing with parametrizable behavioral models offers a comprehensive solution for validating test programs for AMS circuits. It is built around a modular and standardized approach, ensuring flexibility and scalability for a variety of testing scenarios. At its core, the framework uses a wrapper approach to integrate behavioral models written in Verilog-AMS. This method provides a robust structure that facilitates the template-based integration of different models. Each wrapper is designed to hold a list of parameters and port information, such as signal type, bus width, and whether the port is an input, output, or bidirectional. This information is accessible through an interface, allowing for seamless automated testbench generation and the creation of model instance code (see Figure 2).

The framework also includes standardized input source models, inspired by existing models and adapted for enhanced functionality. These input sources are also parameterizable, accommodating various testing scenarios. For digital input sources, the framework offers modules for clocks, heaviside step functions, and customizable value sequences. Similarly, for analog voltage sources, it provides DC voltage sources, ramps, and sine functions. A particularly noteworthy feature is the programmable source module, which can generate voltage sequences based on timestamp-voltage pairs. This module supports three operational modes: immediate voltage setting, linear transitions between points, and a mode that targets a final voltage value with a controllable slope. The flexibility of these input sources is implemented by the ability to generate Verilog-AMS code at runtime, allowing for dynamic adjustments.

Measurement instruments within the framework are also modeled to simulate the behavior of real-world instruments accurately. These models are parameterized to reflect different instrument properties, offering a realistic testing environment. While the final implementation of these instruments may vary, they ensure that the desired sampling behavior is replicated. This could be done as a complete behavioral model in Verilog-AMS or through post-processing algorithms. This adaptability is crucial for achieving realistic and reliable simulation results.

The framework includes an automatic testbench generation process that begins by parsing a netlist file, which contains the components and their connections within the circuit. Following this, the model wrappers and their parameters are instantiated, and the components are connected through their ports. These initial one-to-one connections are then merged into nets during the testbench generation process. This merging involves a compatibility check for signal types and bus widths, ensuring that the generated nets and signals match. The framework then automatically inserts instances of Verilog-AMS modules, connects them to the generated signals, and embeds the appropriate signal patterns, minimizing the need for manual input.

The simulation interface acts as a critical link between the framework components and external Verilog-AMS simulators. It is tailored to the specific simulator being used—in this case, the Siemens Questa ADMS simulator. The interface manages all aspects of the simulation process, from setting simulator accuracy and defining the simulation timeframe to specifying which signals should be recorded. Since the signals are generated dynamically during the automatic testbench generation, the interface automatically maps ports to their corresponding signals, ensuring that the correct data is captured during the simulation. After the simulation is complete, the interface reads the output data, converts it into a tabular format and reverts the signal to port mapping.

Data post-processing is an essential step within the framework, addressing the challenges introduced by the uneven distribution of simulation data points over time. Unlike real measurement instruments, which sample signals at fixed intervals, simulators typically output data based on event triggers or changes in signal values. This can lead to gaps in the retrieved data, which must be addressed to ensure compatibility with the test execution environment. The framework simulates the sampling process of real instruments by artificially filling in the missing data points. For digital signals, the last known value is repeated, while for analog signals, interpolation is performed between known values. This process ensures that the simulation data is uniformly sampled and ready for further analysis.

Finally, the framework is designed for easy integration with a test execution environment. This is achieved through the use of parsers that facilitate file transfers and a standardized data exchange format. This format includes all necessary information, such as the parameterizations of test instruments and the CUT, as well as the netlist describing the connections. By using this standardized format, the framework enables full automation of the virtual test execution process.

4 Other Research Findings

In addition to the main framework, we also explored two strategies for enhancing behavioral models of ADCs: our INL (Integral Non-Linearity) algorithm and the clocked noise model. The INL algorithm is designed to maintain a given maximum INL when distributing the shifts of individual code edges based on a random distribution. Meanwhile, the clocked noise approach injects noise selectively at critical time points, optimizing simulation speed while preserving accuracy.

4.1 INL Algorithm

To understand the objective of the developed algorithm, it is important to first consider the key metrics that define an Analog-to-Digital Converter (ADC). The ADC is a fundamental component responsible for converting analog signals into digital representations. It operates within a specified range between the upper and lower reference voltages, referred to as the Full Scale Range (FSR). Within this range, the analog input voltages are systematically mapped to corresponding digital codes. Ideally, these codes are uniformly distributed across the FSR. The Full Scale (FS) represents the total number of possible digital codes, which increases exponentially with the bit width N. The Least Significant Bit (LSB) serves as the smallest unit of measurement in this system, representing the voltage interval that each digital code encompasses.

Based on the uniform distribution of codes in the ideal case, the code edges must also be equidistant. These code edges mark the points where the digital output changes and are therefore also spaced evenly across the full scale range. However, real-world ADCs exhibit non-idealities that introduce errors such as offset error, gain error, Differential Non-Linearity (DNL), and Integral Non-Linearity (INL). Offset errors cause a uniform shift in all code edges, altering the voltage at which each code is reached, while

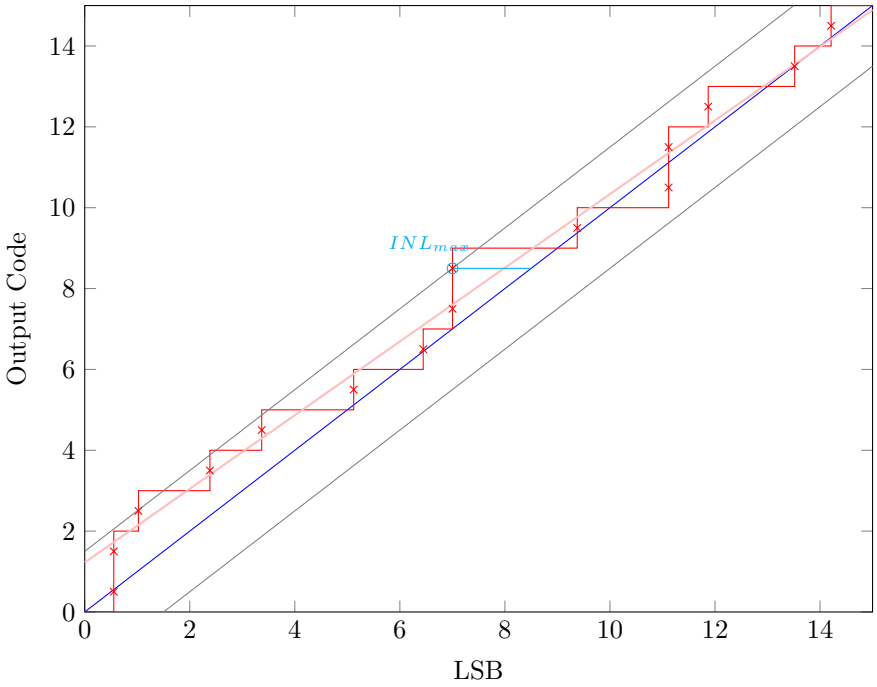

Figure 3: Example of a code edge transfer curve (red) of a 4-bit ADC considering a gain, offset and INL configuration. The code edge with the maximum INL (cyan) lies exactly on the envelope (gray) around the ideal curve (blue), which represents the specified INL value. The offset and gain of the best-fit curve through all code edges (pink) to the ideal curve are also visible.

gain errors modify the slope of the code edge transfer curve, also leading to deviations from the ideal positions of these edges.

DNL errors affect the width of individual code steps, measured as the deviation from the ideal step size in LSBs. Importantly, DNL focuses on step size variations, while INL errors consider the actual position of the code edges compared to their ideal counterparts. The INL value at any code edge reflects this deviation, thereby also including the other non-idealities like offset and gain errors. Maintaining these INL and DNL values within specified limits is crucial for ensuring the ADC's accuracy and performance, as these parameters directly influence the classification of ADCs in the binning process after testing.

To optimize test programs effectively, it is essential to evaluate their ability to accurately detect the parametric shifts in AMS circuits. In the context of virtual testing, this requires the capability to parameterize a model instance with specific errors and subsequently verify whether the test program can identify these errors. Currently, this capability is limited when it comes to ADC models.

Existing methods for influencing the behavior of architecture-representing models typically involve adjusting individual component values, such as capacitor or resistor values [8, 12, 13]. However, the exact impact of these changes on the INL cannot be precisely calculated. Consequently, simulations are run to empirically determine the INL values, rather than predicting them accurately from component changes.

What is needed is a model that can be directly parameterized with an INL error value, thereby reflecting the specific INL error behavior within the simulation. Although behavioral models exist, they fall short of this requirement. They use the INL value as an input to induce additional shifts in the code edges but do not guarantee that this INL value accurately represents the maximum INL value observed in actual tests.

Our INL algorithm addresses this challenge by adjusting the individual code edges according to a probability distribution, while simultaneously maintaining a maximum INL value, as well as specified offset and gain errors. The algorithm employs an iterative process where the code edges are linearly shifted, ensuring that all three criteria—INL, offset, and gain—are accurately met. Figure 3 shows the simulation results of an ADC behavioral model whose code edges were determined by the INL algorithm. All three error types offset, gain and INL are present. Comparative analyses indicate that our approach achieves a level of accuracy unparalleled by existing behavioral models [6, 14, 15].

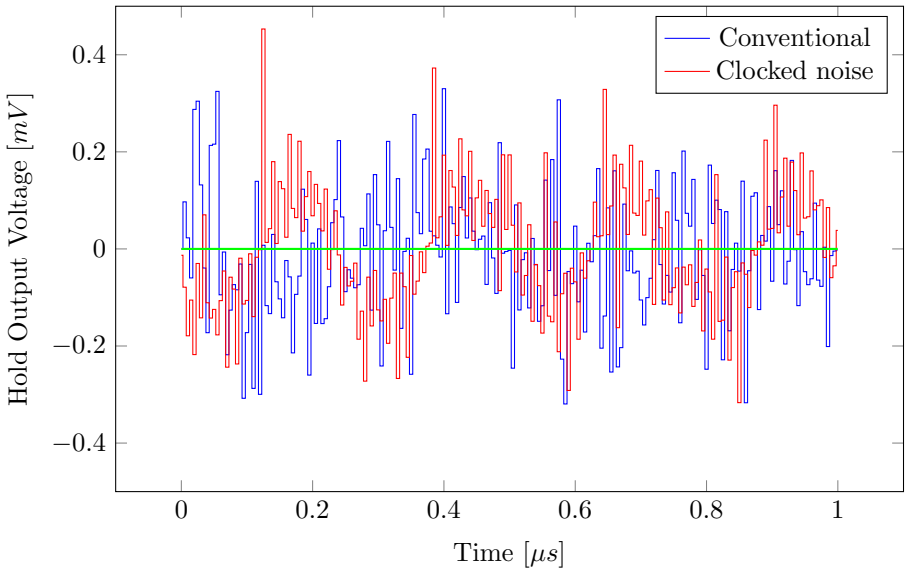

Figure 4: Output signals of a S/H circuit using the conventional transient noise analysis and our clocked noise method. The noise power was set to the level of thermal noise. The ideal analog input signal (green) was set to $0V$. The influence of the generated noise based on the traditional and the clocked noise method can be observed similarly.

4.2 Clocked Noise

In real-world applications, electrical signals are inherently subject to noise, which can significantly impact the performance of electronic circuits. Simulation of noise is therefore essential, especially in systems where precision is critical. However, traditional noise modeling methods are computationally intensive, making them impractical for many simulations, particularly when simulating behavioral models where low computational overhead is crucial. This often leads to noise being ignored or only partially considered in standard simulations.

Standard noise modeling typically involves continuous noise sources that affect the signal throughout the simulation. This method can be very slow, especially when applied to circuits with high sampling rates, as it requires the simulation to account for noise at every timestep. This is particularly cumbersome in AMS systems, where analog noise can propagate through digital parts, leading to errors that might not be accurately captured if noise is improperly modeled.

Our approach to noise injection is specifically designed for sample-and-hold (S/H) circuits, which are commonly found in ADCs. S/H circuits capture and hold the input signal momentarily to ensure a stable value for conversion. As only the noise present at the time of sampling can affect the outcome of the conversion, it is possible to model noise accurately at these specific points rather than continuously.

To improve the efficiency of noise modeling in S/H circuits, we introduce a method called clocked noise injection. Instead of applying noise continuously, we inject noise only at specific sampling points when the S/H circuit captures the input signal. This is achieved by introducing a noise generator that is triggered by the clock signal of the S/H circuit. The noise generator outputs a new noise value only when the S/H circuit samples the signal, effectively undersampling the noise in sync with the circuit's operation.

Simulation results confirm that our clocked noise injection method not only matches the accuracy of traditional noise modeling but does so with a fraction of the computational cost. This makes it a valuable tool for simulating S/H circuits in ADCs and other AMS systems, where noise modeling is necessary, but also simulation speed must be met. This is especially true for our test program validation framework, where rapid simulation speeds are crucial to the targeted test program development workflow.

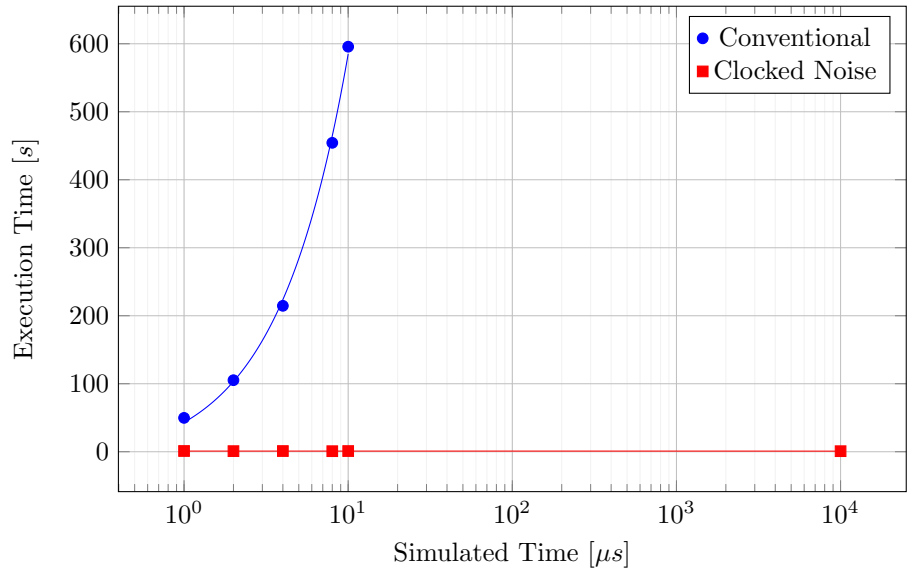

Figure 5: Required execution time of the simulator with varying simulated time. The sample rate of the S/H circuit was adjusted so that always 100 samples were executed in the simulated time.

5 Future Work

In our ongoing efforts to enhance the framework, several key areas will be addressed. First, we plan to extend the framework by integrating detailed models of tester instruments and transmission paths. This addition will allow for more accurate and comprehensive simulations that account for the influence of the testing environment.

Additionally, we aim to further optimize the framework to increase simulation speed and enable parallel execution capabilities. By improving these aspects, we expect to support more complex simulations and reduce overall computation times.

Another avenue for exploration involves reorganization and automatic generation of test programs. By optimizing the test program and also its development process, we can enhance the efficiency and effectiveness of testing workflows.

Finally, we intend to map the INL algorithm, initially developed for ADCs, to Digital-to-Analog Converters (DACs). This extension will broaden the applicability of our approach and potentially yield similar accuracy improvements in DAC performance validation.

References

[1] Mark Burns, Gordon W Roberts, and Friedrich Taenzler. *An introduction to mixed-signal IC test and measurement*, volume 2001. Oxford university press New York, 2001.

[2] Linda 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*, 45(10):1389–1407, 1998.

[3] Ira Miller and Thierry 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)*, pages C49–1. IEEE, 2000.

[4] Robert O Peruzzi and Jose Medero. A systematic approach to creating behavioral models. 2015.

[5] Rajeev Narayanan, Naeem Abbasi, Mohamed Zaki, Ghiath Al Sammane, and Sofiene Tahar. On the simulation performance of contemporary ams hardware description languages. In *2008 International Conference on Microelectronics*, pages 361–364. IEEE, 2008.

[6] Robert O Peruzzi. Verification of digitally calibrated analog systems with verilog-ams behavioral models. In *2006 IEEE International Behavioral Modeling and Simulation Workshop*, pages 7–16. IEEE, 2006.

[7] Nicola Dall'Ora, Sadia Azam, Enrico Fraccaroli, Renaud Gillon, and Franco Fummi. Verilog-a implementation of generic defect templates for analog fault injection. In *Proceedings of the Great Lakes Symposium on VLSI 2023*, pages 477–481, 2023.

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

[9] Byong Chan Lim and Mark Horowitz. An analog model template library: Simplifying chip-level, mixed-signal design verification. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 27(1):193–204, 2018.

[10] M Miegler and Werner Wolz. Development of test programs in a virtual test environment. In *Proceedings of 14th VLSI Test Symposium*, pages 99–103. IEEE, 1996.

[11] Leon MA van de Logt, Vladimir A Zivkovic, and Ingrid HA van Baast. Model-driven ams test setup validation tool prepared for ieee p1687. 2. In *2019 IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2019.

[12] Christos Sapsanis, Martin Villemur, and Andreas G Andreou. Real number modeling of a sar adc behavior using systemverilog. In *2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 1–4. IEEE, 2022.

[13] Vicente Y Ponce-Hinestroza and Victor 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)*, pages 301–304. IEEE, 2018.

[14] Modeling analog to digital converters, 2024.

[15] Ken Kundert and Olaf Zinke. *The designer's guide to Verilog-AMS*. Springer Science & Business Media, 2005.