DMS Verification environment for Gyroscope System

Extended DMS Environment: from Design to System Verification

Thierry Nouguier, NXP, Toulouse, France (*thierry.nouguier@nxp.com*)

Ajay G Nayak, NXP, Bangalore, India (*ajayg.nayak@nxp.com*)

Fabrice Boissieres, NXP, Toulouse, France (*fabrice.boissieres@nxp.com*)

***Abstract*— Gyroscope system is inherently complex, consisting of both mechanical and electronic parts, often with feedback loops that interact through state machine and digital controls. While tools like MATLAB are widely used for verifying system architecture, Analog Mixed-Signal (AMS) environments are more commonly employed for verifying the actual implementation and ensuring functionality. However, Digital Mixed-Signal (DMS) environment has not been extensively explored for verifying gyroscope systems, despite their potential for improving verification efficiency. This paper aims to explore the challenges of modeling a gyroscope system (ASIC and Micro-Electronical-Mechanical Systems (MEMS)) in a DMS environment, focusing on the integration of mechanical and electronic subsystems. Furthermore, we demonstrate how DMS enhances verification efficiency by offering an integrated platform that enable better coverage.**

Keywords—gyroscope; drive; systemverilog; real number modeling; user-defined nettype; UVM-MS; Python DPI

I. Introduction

Digital Mixed-Signal (DMS) environment, leveraging efficient modeling languages like wreal and SystemVerilog, are widely used to accelerate simulation and improve coverage. While effective for standard applications, DMS is often complemented by Analog Mixed-Signal (AMS) simulations conducted in parallel to ensure higher accuracy, especially in complex system such as MEMS including gyroscopes, where precise analog behavior must be validated. Traditionally, DMS has focused on verifying digital control logic, with AMS limited by slower performance. To bridge this gap, SystemVerilog UDN (User Defined Nettype) [1] introduces enhanced modeling capabilities that improve DMS accuracy. Building on this modeling approach, we propose a DMS simulation environment that shifts much of the AMS workload into the digital domain. This enables faster, scalable, and sufficiently accurate system-level verification, reducing dependence on full AMS simulations while preserving confidence in both functional and parametric correctness.

But, during the development of our verification environment, we encountered several challenges: keep the accurate behavioral representation of the MEMS and the analog blocks, efficient simulation performance and robust measurement capabilities. So, rather than relying on the conventional EEnet provided by Cadence, we developed a custom UDN structure with a dedicated resolution function. This allowed us to maintain a balance between simulation speed and modeling accuracy, ensuring that the models remained representative of real-world behavior. In addition, we introduced a time-step tuning strategy across different models (MEMS and Capacitance-to-Voltage converter (C2V)). By adjusting the simulation step size selectively, we were able to preserve accuracy in critical regions without incurring excessive simulation overhead.

To support the gyroscope study, which requires detailed signal analysis, we enhanced our testbench to handle complex measurements (like FFT, frequency, amplitude etc...) more effectively. Traditional SystemVerilog-based methods posed challenges in computing such metrics, prompting us to integrate a Python-based measurement strategy via the Direct Programming Interface (DPI). This integration allows signal data to be extracted and analyzed directly in Python, leveraging its powerful numerical and signal processing libraries. As a result, we achieved more accurate, flexible and efficient computation of key signal characteristics. These monitors are integrated in a UVM testbench based on UVM-MS [2], forming the foundation of a unified AMS/DMS verification environment.

II. Description of the Work

A MEMS gyroscope system is a compact and integrated sensor used to measure angular velocity—how fast an object is rotating. It consists of two tightly connected components: a micro-mechanical structure built using MEMS (Micro-Electro-Mechanical Systems) technology, and an electronic processing unit known as an ASIC (Application-Specific Integrated Circuit).

The MEMS part includes a tiny proof mass suspended by microfabricated springs. This mass is continuously driven to vibrate along a specific axis (called the drive axis). When the device experiences rotation around an axis perpendicular to this motion, Coriolis forces act on the vibrating mass, causing it to deflect slightly along a second axis (called the sense axis). This deflection is extremely small but can be detected using capacitive sensing, where changes in capacitance between the moving mass and fixed electrodes are measured [3-5]

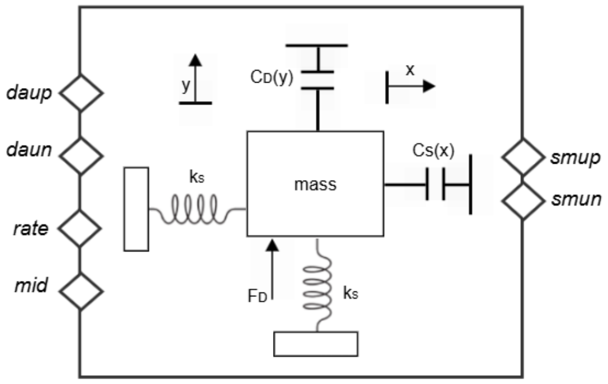


Figure 1. MEMS Architecture

The ASIC plays a critical role in making the sensor functional and accurate. It generates the drive signal to keep the mass vibrating at a stable amplitude and frequency, using feedback control such as automatic gain control (AGC). It also amplifies and filters the weak signal from the sense axis, demodulates it to isolate the Coriolis component, and converts the result into a digital signal using an analog-to-digital converter (ADC). Additionally, the ASIC handles calibration, temperature compensation, and communication with external systems through digital interfaces like SPI. The system also interfaces with other modules such as the Phase-Locked Loop (PLL) and the Power Monitor Circuit (PMC).

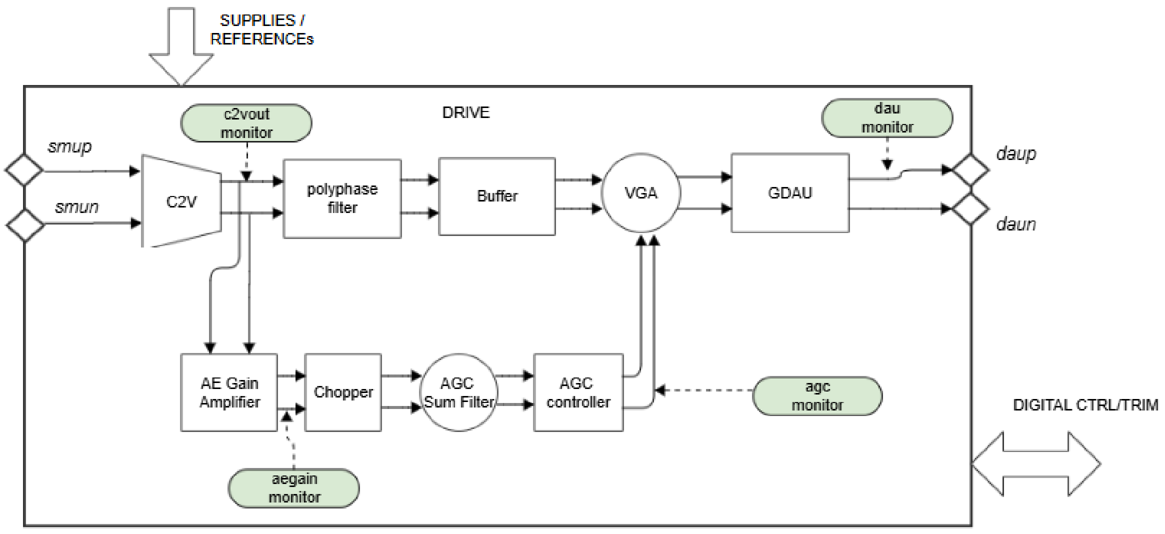


Figure 2. Drive Architecture

Figure 1 depicts the MEMS architecture which provides the capacitance to the electrical system. Figure 2 illustrates the components required to model the drive analog front-end, which includes various sub-functions such as amplifiers and filters.

To enable the DMS environment for the gyroscope, we needed to develop SystemVerilog models for each of these blocks. Modeling both MEMS and electrical ASIC systems in SystemVerilog introduces a host of significant challenges that stem from the fundamental mismatch between the physical nature of MEMS and the digital abstraction of hardware description languages. MEMS devices operate in the continuous-time, analog, and mechanical domains, exhibiting complex behaviors such as resonance, damping, non-linear stiffness, and thermal drift—none of which are natively supported in SystemVerilog. Attempting to represent these dynamics using discrete-time constructs often leads to oversimplified or inaccurate models that fail to capture critical performance characteristics. Moreover, SystemVerilog lacks built-in support for multi-domain simulation, making it extremely difficult to model the tight coupling between mechanical motion and electrical signal processing. This is especially problematic in systems where phase accuracy is paramount, such as resonant sensors or closed-loop control systems, where even minor phase mismatches can lead to instability or degraded sensitivity.

Additionally, the absence of analog modeling capabilities limits the ability to simulate noise, parasitics, and signal degradation—factors that are crucial in high-precision sensing applications. The lack of interoperability with high-fidelity simulation tools further exacerbates the problem, making it hard to validate or calibrate the models against real-world behavior.

III. SOLUTIONS PROPOSED

A. *Modelling Strategy*

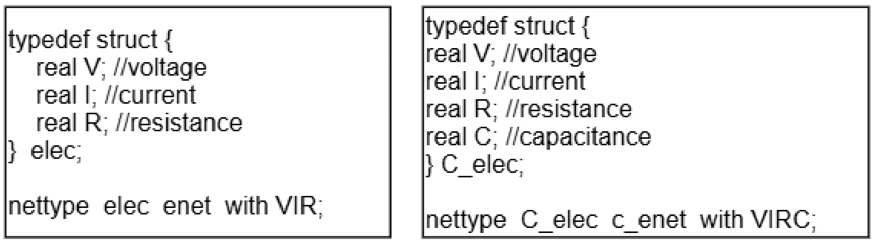
To model the accurate analog behavior and maintains the performance, Cadence provides a one-stop solution through its EEnet System Verilog UDN (User Defined Net), which allows modeling of voltage, current, and impedance on nets and nodes, but its built-in resolution function performs extensive computations that significantly slow down simulation speed. To address this, we developed a custom UDN called enet with a simplified resolution function that works on the principle of Kirchhoff's Current law whereas the Cadence EEnet works on Norton's Theorem making the resolution function complex to compute. This tailored approach retains the essential modeling capabilities while dramatically improving simulation performance, making it a practical strategy to increase the speed and maintain the accuracy. Table I highlights the performance of custom enet with Cadence EEnet for a capacitor model, showing a simulation accuracy difference of only 0.1% between enet and Spectre.

Table I. Performance comparison of enet and EEnet

	Spectre (VAMS)	EEnet	enet
Simulation time(s)	256	129	24
Speed-up	x1	x2	x10

Modeling the interaction between MEMS and ASIC systems in SystemVerilog presents deep challenges, particularly due to the fundamental differences in how mechanical and electrical domains operate and interact. In many MEMS systems, the mechanical structure produces a capacitance output—a key sensing parameter that must be accurately interpreted by the ASIC for further computation. However, standard modeling constructs in SystemVerilog, including Cadence's EEnet UDN, are limited to handling voltage, current, and impedance, and do not natively support the direct transmission of capacitance as a signal. Attempting to pass the current flowing through a capacitor as a proxy for capacitance introduces significant inaccuracies, especially in dynamic or nonlinear conditions, leading to incorrect signal interpretation and degraded system performance.

To overcome this limitation, we introduced a custom net type called c_enet, which extends the capabilities of enet by allowing a net to carry not only voltage (V), current (I), and impedance (R), but also capacitance (C) as a first-class quantity. This enhancement enables more accurate and physically meaningful modeling of the MEMS-to-ASIC interface, preserving the integrity of the mechanical signal as it enters the electrical domain. Combined with the simplified resolution function in enet—which improves simulation speed compared to the more computationally intensive EEnet - c_enet provides a powerful and efficient strategy for modeling complex electro-mechanical interactions in a unified SystemVerilog environment. Figure 3 shows the enet and c_enet structures



```
typedef struct {
    real V; //voltage
    real I; //current
    real R; //resistance
}  elec;

nettype  elec  enet  with VIR;
```

```
typedef struct {
real V; //voltage
real I; //current
real R; //resistance
real C; //capacitance
} C_elec;

nettype  C_elec  c_enet  with VIRC;
```

(a) (b)

Figure 3. (a) enet and (b) c_enet representation

B. Sampling Strategy

In MEMS-based systems, accuracy is paramount due to the analog and mechanical nature of the sensing elements. MEMS devices often operate in high-frequency domains and produce outputs such as capacitance changes. These outputs are typically phase-sensitive, meaning that even small timing errors can lead to significant deviations in the interpreted signal. For example, in resonant MEMS sensors or capacitive sensing applications, maintaining precise phase alignment is critical for correct signal demodulation, feedback control, and overall system stability.

In contrast, the ASIC subsystem can process signals in discrete time steps and is generally less sensitive to fine-grained timing variations. While accuracy is still important, the ASIC can tolerate coarser time resolution without significantly affecting performance. This difference in sensitivity allows for optimization in simulation strategies.

To balance simulation accuracy and efficiency, in this paper we propose a multi-rate sampling strategy. MEMS Subsystem is simulated with a fine timestep (0.1 ns) to capture high-frequency, phase-sensitive behavior with high fidelity whereas the ASIC Subsystem is simulated with a coarser timestep (1 ns) since it processes signals at a slower rate and is less sensitive to sub-nanosecond variations. This approach ensures that the MEMS model maintains the temporal resolution needed for accurate phase tracking, while the ASIC model benefits from reduced computational complexity. To validate and observe the behavior of models under simulation, a comprehensive testbench environment was established.

C. Testbench Environment

To monitor relevant signals, we developed dedicated UVCs configured in passive mode. These UVCs share a common architecture build on the UVM-MS 1.0 standard from Accellera [6]. Each UVC is specialized for a specific signal type (e.g., sinewave, square wave, DC), extracting key characteristics from the signal.

Analog signals are discretized by a sampler within the MS Bridge. The resulting samples are encapsulated in a sample_trans transaction, passed through the proxy to the monitor, and then sent to Python scripts via the DPI interface. The Python script performs signal analysis and returns the extracted parameters to the meter via the analysis port. These results are stored in a measures_trans transaction, enabling the meter port to connect to a scoreboard form comparison against expected values.

Configuration options (core cfg) allow tuning of sampling parameters such as samples size and sampling time. The analysis configuration (analysis cfg) links each UVC to the appropriate measurements. For example, in the Figure 4, a UVM-MS monitor for sinewave signal uses a Python script called Sine Measure. This script retrieves 1000 samples (get_data), performs an FFT to extract frequency, amplitude and DC offset (meas_fft) and stores the results using (get_freq, get_ampl, get_dc) in the measures_trans transaction.

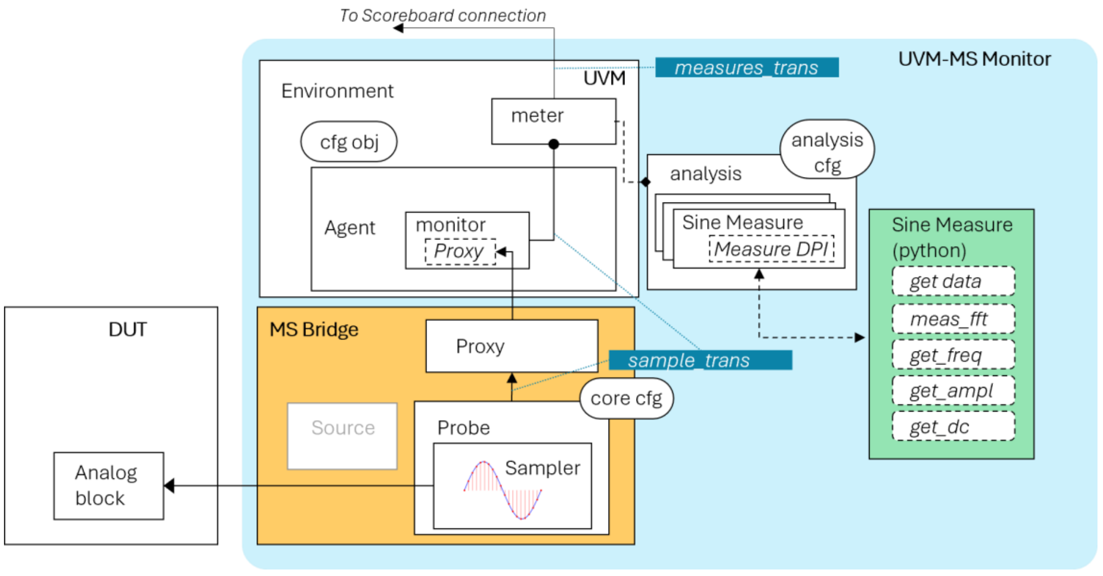
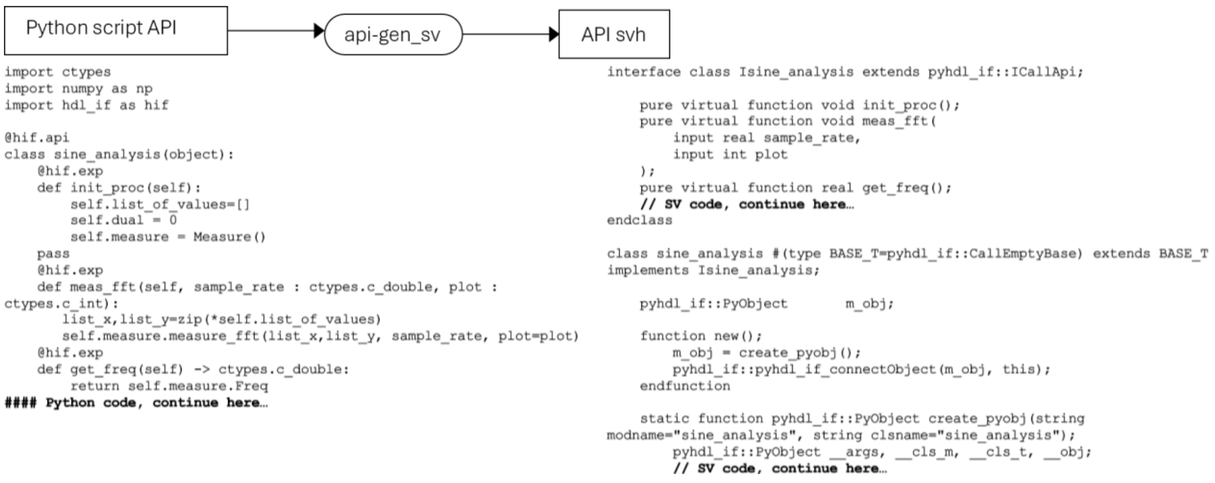


Figure 4. UVM-MS Monitor to measure sine wave

D. DPI integration

While many DPI solutions exist for integrating Python, they are often complex to implement in a UVM environment. We chose the pyhdl-if package [7], which offers a SystemVerilog-friendly API for calling Python functions without the need for C wrappers. This significantly reduces development effort and complexity.

Originally, pyhdl-if did not support real (floating-point) values, making limiting its applicability for analog signal processing. To overcome this, we extended the package to include support for both double and float types, enabling its use in mixed-signal design. Figure 5 demonstrates how easily SystemVerilog wrapper code can be generated, and the DPI configured to integrate Python functions seamlessly.



```
import ctypes
import numpy as np
import hdl_if as hif

@hif.api
class sine_analysis(object):
    @hif.exp
    def init_proc(self):
        self.list_of_values=[]
        self.dual = 0
        self.measure = Measure()
    pass
    @hif.exp
    def meas_fft(self, sample_rate : ctypes.c_double, plot :
ctypes.c_int):
       list_x,list_y=zip(*self.list_of_values)
       self.measure.measure_fft(list_x,list_y, sample_rate, plot=plot)
    @hif.exp
    def get_freq(self) -> ctypes.c_double:
        return self.measure.Freq
#### Python code, continue here…
```

```
interface class Isine_analysis extends pyhdl_if::ICallApi;

    pure virtual function void init_proc();
    pure virtual function void meas_fft(
        input real sample_rate,
        input int plot
    );
    pure virtual function real get_freq();
    // SV code, continue here…
endclass

class sine_analysis #(type BASE_T=pyhdl_if::CallEmptyBase) extends BASE_T
implements Isine_analysis;

    pyhdl_if::PyObject       m_obj;

    function new();
        m_obj = create_pyobj();
        pyhdl_if::pyhdl_if_connectObject(m_obj, this);
    endfunction

    static function pyhdl_if::PyObject create_pyobj(string
modname="sine_analysis", string clsname="sine_analysis");
        pyhdl_if::PyObject __args, __cls_m, __cls_t, __obj;
        // SV code, continue here…
```

Figure 5. System Verilog wrapper code and DPI Integration

While implementing such computations directly in SystemVerilog would be complex and cumbersome, especially for tasks involving signals analysis, Python offer more flexible and efficient alternatives. By leveraging Python's rich ecosystem and third-party libraries, we can perform advanced computations with ease: phase extraction is achieved using the Hilbert transform [8], while frequency, amplitude and DC characteristics are derived from FFT analysis [9].

IV. Use Case: Safety Mechanism Verification

As an automotive safety product, the circuit incorporates several mechanisms to ensure reliable operation and effective fault detection. Among these, one of the most critical is the self-test feature, which verifies the correct functioning of the gyroscope both at startup and during normal operation. Another mechanism is monitoring DAU (Drive Actuation Unit). The DAU controls the drive loop by keeping the vibrating structure oscillating at its resonant frequency, then ensuring stable operation of the MEMS structure. So, by monitoring the DAU amplitude, we prevent overdriving the MEMS structure, which could cause mechanical damage.

The integration of a new MEMS structure has revealed significant dispersion in the drive quality factor (Qd), resulting in the DAU operating near its voltage rail limits. This reduced headroom compromises the effectiveness of monitoring and diagnostic functions, potentially masking early signs of degradation.

It is therefore essential to verify this behavior and define acceptable operating limits, enabling the MEMS designer to take corrective action if necessary. The only reliable way to perform this verification is by simulating the complete system, comprising MEMS, Drive, PLL, ADC, and digital blocks.

Initially, the MEMS model exhibited incorrect behavior as the Qd decreased, the DAU output also decreased, whereas it should have increased. This discrepancy indicated that the original MEMS model lacked sufficient accuracy to reflect the expected system dynamics, despite appearing functional under standard conditions (Typical Qd). After refining the MEMS model to improve its accuracy, the simulation results aligned with expectations: a decrease in Qd led to an increase of DAU output, confirming the corrected behavior. Then we conducted a series of simulations by varying Qd across a defined parameter space from the dispersion. For each simulation, we extracted the DAU amplitude response and performed a comparative analysis to quantify its influence on the activation threshold and reliability of the safety mechanism.

V. Results

A. *Validation criteria*

The primary validation criteria for the Drive and MEMS models include both signal-level characteristics, such as phase, amplitude across various stages, and system-level behaviors like startup time, FSM state transitions and digital control logic. Depending on the type of characteristic being verified, different validation methods are employed. Analog properties are verified using a scoreboard, while system characteristics are checked using SystemVerilog Assertions (SVA). The scoreboard includes also DPI allowing us to display results data and plot the different metrics with their standard deviation / mean / min / max.

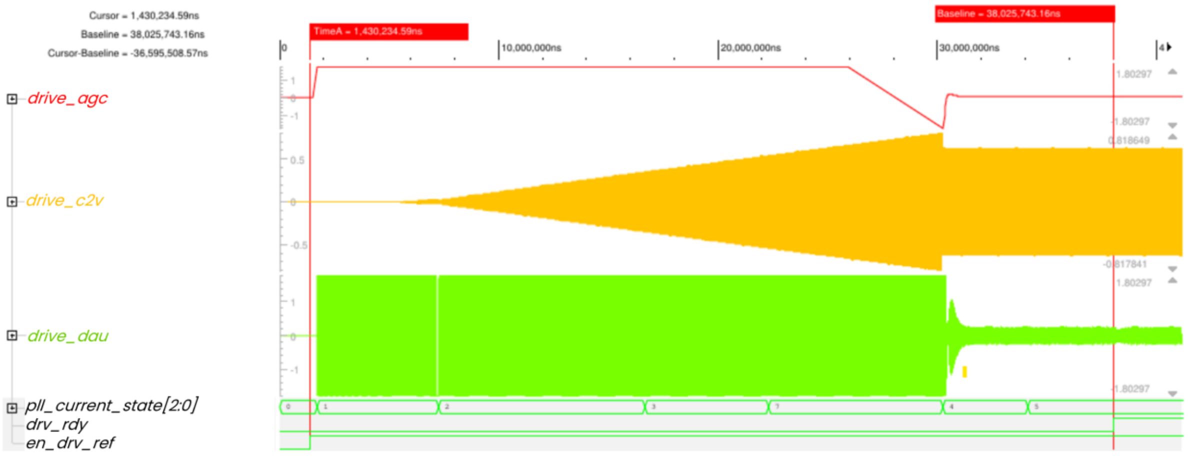
B. *Results*

The table II detailed the Drive signal-level characteristics for both DMS and AMS. While amplitude and phase measurements are largely consistent, DMS with its 0.1ns resolution, offers improved precision. However, the observed DC levels mismatches for the AGC and for the DAU phase in DMS (at 1ns resolution) expose limitations in sampling accuracy. This highlights the critical role of a precise sampling in the MEMS model for accurately capturing system behavior and enabling reliable verification. When comparing startup time, DMS at 0.1ns shows a reduction of approximately 0.66ms (1.77% faster), while DMS at 1ns demonstrates a more significant difference of 2.54ms (6.82% faster). These results highlight the influence of sampling resolution on startup performance, reinforcing the need for a high-precision sampling in the MEMS model to align closely with AMS reference behavior. Concerning CPU time, the comparison highlights significant differences in computational efficiency across the evaluated methods. The DMS configuration with 0.1 ns sampling, while offering finer temporal resolution and capturing more detailed signal precisions, nearly doubles the CPU time compared to the 1 ns case. Meanwhile, the AMS reference model, although providing the highest fidelity due to its detailed analog behavior modeling, results in the longest simulation time, thus being more appropriate for final validation stages rather than early design exploration or verification debug.

Table II. Drive Results: signal-level characteristics

Signals	**Drive signal-level characteristics**			
	Signal characteristics	***DMS [sampling:0.1ns]***	***DMS [sampling:1ns]***	***AMS [Ref]***
c2v	Amplitude [Volt]	0.6346	0.636	0.628
	Frequency [Hz]	20000	20000	20000
	Phase vs (Polyphase) [deg]	-89.21	-89.21	-90
agc	DC level [Volt]	0.04595	-0.0446	0.05
dau	Amplitude [Volt]	0.2567	0.249	0.345
	Frequency [Hz]	20000	20000	20000
	Phase vs (VGA) [deg]	-0.0062	-157.42	0.001
aegain	Amplitude [Volt]	0.67	0.67	0.664
	Frequency [Hz]	20000	20000	20000
Parameters	**Drive system characteristics / CPU**			
Startup-time	[ms]	36.59	34.71	37.25
CPU time	[h:m:s]	00:36:53	00:17:46	1:30:52

Figure 6 presents simulation waveform extracted from DMS environment, illustrating key signals from the Drive subsystem (e.g. agc, c2v, dau) alongside digital signals such as state of the PLL FSM and drive ready signal. These results exhibit a high degree of alignment with the AMS reference, thereby validating the reliability and accuracy of the DMS simulation approach. In the second graph we can observe a series of simulations across a range of Qd values linked to the Use Case section. The refined MEMS model (0.1ns sampled) accurately captures the expected DAU behavior under Qd variation. Simulation confirms that Qd dispersion can significantly affect DAU amplitude. AMS trend aligns with DMS (sampling 0.1ns), confirming model consistency. DMS (sampling 1ns), trend shows an incorrect response. The divergence between the two DMS curves highlights the importance of high-resolution sampling for accurate DAU monitoring



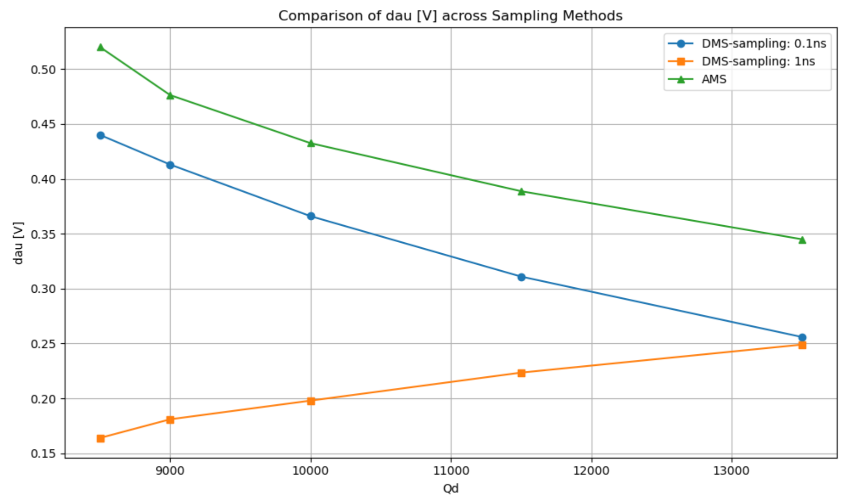


Figure 6.

VI. Conclusion

Modeling MEMS and the analog signal chain using DMS has proven both feasible and effective, maintaining the accuracy required for robust system-level verification. A key contributor to this accuracy is the precise configuration of the MEMS model's sampling time, which critically affects behavioral fidelity and ensures strong correlation with AMS reference outcomes. Our custom enet structure further enhances accuracy, particularly for c2v signals, enabling more reliable verification of use case such as DAU monitoring. The integration of Python DPI within our UVM-MS verification environment facilitates efficient extraction of signal characteristics and supports complex measurements for validating system behavior.

Looking ahead, future work includes developing a dedicated DMS environment for MEMS verification, extending the approach to accelerometers, and enhancing gyroscope modeling with features like shock detection. We also aim to integrate AI/ML techniques for parameters tuning, anomaly detection, and predictive modeling to further improve verification efficiency and insight.

References

[1] Cadence Design Systems, *SystemVerilog Real Number Modeling (SV-RNM) Based Advanced Verification*, 2024. [Online]. Available: https://www.cadence.com/en_US/home/training/all-courses/86274.html
An advanced training resource covering SV-RNM techniques, including EEnet, connect modules, and mixed-signal simulation strategies.

[2] T. Fitzpatrick, P. Lynch, X. Li, S. Chetput, J. Westendorp, and L. Balasubramanian, "*UVM-AMS: A UVM-Based Analog Verification Standard*," presented at DVCon U.S. 2021, Accellera Systems Initiative

[3] W. Wang, X. Lv and F. Sun, "Design of Micromachined Vibratory Gyroscope With Two Degree-of-Freedom Drive-Mode and Sense-Mode," in IEEE Sensors Journal, vol. 12, no. 7, pp. 2460-2464, July 2012, doi: 10.1109/JSEN.2012.2192497.

[4] J. Nazdrowicz and A. Napieralski, "Modelling, Simulations and Performance Analysis of MEMS vibrating Gyroscope in Coventor MEMS+ Environment," 2019 20th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE), Hannover, Germany, 2019, pp. 1-5, doi: 10.1109/EuroSimE.2019.8724520.

[5] F. Cenni, O. Guillaume, M. Diaz-Nava and T. Maehne, "SystemC-AMS/MDVP-based modeling for the virtual prototyping of MEMS applications," 2015 Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP), Montpellier, France, 2015, pp. 1-6, doi: 10.1109/DTIP.2015.7160972.

[6] Accellera Systems Initiative, "*UVM Mixed-Signal (UVM-MS) 1.0 Standard*," Jan. 2025

[7] PyPI, "pyhdl-if," *Python Package Index*, Mar. 13, 2025. [Online]. Available: https://pypi.org/project/pyhdl-if/

[8] I. G. Roy, "Hilbert Transform: A Brief Overview," *Resonance*, vol. 29, no. 5, pp. 671–689, Aug. 2024. [Online]. Available: https://link.springer.com/article/10.1007/s12045-024-0671-7

[9] T. Haslwanter, *Hands-on Signal Analysis with Python*. Springer, 2021. [Online]. Available: https://link.springer.com/book/10.1007/978-3-030-57903-6