Virtual Prototyping in SystemC AMS for Validation of Tight Sensor/Firmware Interaction in Smart Sensors

1st Alexandra Küster, 2nd Rainer Dorsch
Bosch Sensortec GmbH
Reutlingen, Germany
alexandra.kuester@bosch-sensortec.com

3rd Christian Haubelt
University of Rostock
Rostock, Germany

4th Karsten Einwich
COSEDA Technologies GmbH
Dresden, Germany

***Abstract*—The growing number of ultra-low power sensor applications drives the processing requirements "on-the-edge" and increases the demand for smart sensors, which implement signal processing and algorithmic features in firmware. Virtual prototyping in SystemC has become a major field of interest to validate the firmware and allow a seamless integration. However, the capability of SystemC is limited to discrete-time applications and cannot handle the full sensor system including its analog front end and mechanical part. Consequently, the firmware validation is often limited to oversimplified scenarios. In this paper, we present a virtual system prototype (VSP) using SystemC and its analog/mixed-signal (AMS) extensions which permits the validation of complex firmware features with tight interaction to the sensor element. The key benefit of this approach is an improved controllability and observability during the sensor firmware development even in early design phases. An industrial case study of a MEMS accelerometer and gyroscope is used throughout the paper to illustrate the proposed approach. A performance analysis proves the pracitcal relevance of our full-stack VSP as the simulation time is increased by a factor less than five compared to a pure SystemC approach without any functionality in the analog or physical domain.**

***Index Terms*—SystemC, SystemC AMS, Virtual Prototype, MEMS**

I. Introduction

In recent years, the amount of sensors in the consumer electronics market has increased rapidly. Smart sensing has pushed forward new product areas as wearables and hearables integrating smart algorithms as gesture recognition or advanced built-in self-test methods. The emerging complex functionalities together with the demand for ultra-low power devices lead to an increasing amount of integrated firmware. However, the consumer market is changing fast and thus the pressure for low time-to-market is high. Thereby, the hardware/software integration becomes a major bottleneck. Hardware/software mismatches are often detected late due to loose cross-checking during the design flow. In order to discover mismatches earlier, system-level simulation is used to validate the firmware and its interaction with the hardware before the latter is available. Thus, virtual system prototypes (VSPs) have become a major field of interest within the last years.

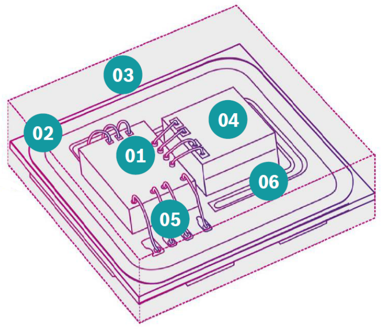


Fig. 1: Schematic of a MEMS sensor. (1) Application Specific Integrated Circuit (ASIC), (2) Printed Circuit Board (PCB), (3) Housing, (4) MEMS element, (5) Bonding wires, (6) Decoupling unit

VSPs allow system-level simulation beyond the scope of functional modeling. Instead, they can implement timing or performance related simulations [1]. Moreover, they offer high reuse and are available in very early stages. SystemC has developed as a standard for system-level simulations. As it is implemented as a C++ library, it inherently offers possibilities to simulate hardware models and software together. Its event-driven nature makes simulations fast. Nevertheless, it also restricts it to the discrete-time domain missing out not only the analog but also the multi-physical domain needed to simulate a heterogeneous smart sensor system. In order to overcome the domain limitations of SystemC, efforts have been undertaken to develop an analog/mixed-signal (AMS) extension for SystemC, called SystemC AMS. The current standard has been released as IEEE Std 1666.1-2016 [2].

Traditional firmware use cases like step counters implement post-processing procedures on the sensor data which allows easy validation on recorded data. However, in modern sensors, the firmware functions running on the device are tightly coupled with the sensor functions and bidirectionally interact with the sensor element. This generates a feedback loop between sensing unit and firmware, i.e. between the multi-physical hardware and the software. If the sensor element

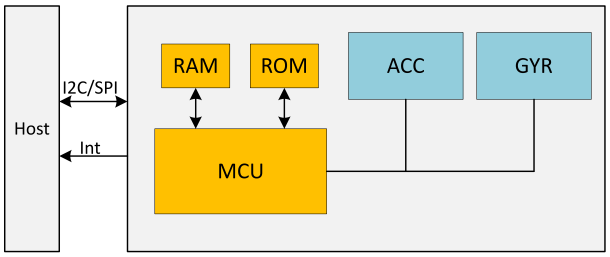


Fig. 2: Schematic view of a smart sensor. It includes a microcontroller unit (MCU) to process the raw sensor data on the device.

is not represented in the VSP, the feedback loop is broken. Excitation schemes normally controlled by the firmware are not supported and must be realized using complex and error-prone test scripts.

In order to solve these issues, we propose a VSP implementing the full stack from sensor element to firmware for a MEMS (Micro-Electronic Mechanical System) inertial measurement unit (IMU) including three-axis accelerometer and three-axis gyroscope. Fig. 1 shows the schematic structure of such a MEMS sensor. For smart sensors, the ASIC (Application Specific Integrated Circuit) includes a microcontroller unit (MCU) and memory in order to directly process the raw sensor data on the device using integrated firmware. A schematic view is given in Fig. 2.

In the following, we refer to the sensor element as the MEMS element (shown in blue in Fig. 2), or simply MEMS, and its analog front end which is part of the ASIC. The sensor element is modeled in SystemC AMS. The rest of the ASIC, i.e. the digital part of the prototype, is written in SystemC. As system-level modeling in SystemC is well-known [1], we focus on the implementation of the AMS part. Afterwards, the VSP is applied to one test case per sensor to prove its ability for firmware validation. Thereby, the benefits and costs are discussed including a performance analysis that shows the suitability of our approach for industrial use.

The rest of the paper is structured as follows. Section II gives an overview of related work. Section III introduces the modeling language and the fundamentals of MEMS-based accelerometers and gyroscopes in preparation of section IV, where the implemented model is described. The model is then applied to one firmware feature per sensor, and benefits and costs of the new approach are discussed as a case study in section V. The paper is concluded in section VI.

II. Related Work

As the complexity of heterogeneous systems in general and smart sensors in particular is rising fast, virtual system prototypes have gained further interest within the last two decades. System-level models for sensor elements have been proposed manifold with a wide range of different abstraction levels. In [3], the authors present a generic mathematical model built in Matlab Simulink that can be applied to all sensor types as it only describes the input-output behavior using a scale factor and some non-ideal terms as nonlinearities and noise. It is not suitable for our application as it misses any dynamics and the implementation of test modes. Andryakov et al. [4] propose the usage of Verilog-AMS for the MEMS modeling in order to get a common simulation framework for MEMS and ASIC to ease their co-design. The approach works likely using VHDL-AMS as it is used in [5]. There, a combined top-down and bottom-up design flow is proposed and illustrated using a microaccelerometer model. They extend the VHDL-AMS modeling on system-level by finite element methodical simulations for the lower abstraction level and feed back the gained parameterization to the system-level model. However, both hardware description languages reach their limits if the simulation scope is extended to embedded software. Moreover, the detailed analysis and comparison of those modeling languages in [6] concludes in the statement that both might not be suitable for full system simulation of complex heterogeneous systems and the authors describe the need for an AMS extension to SystemC.

After SystemC AMS has been released, several studies have been conducted to show the ability of SystemC AMS for different use-cases. In [7], the authors propose a top-down design flow using SystemC AMS and apply it to a temperature sensor. Thereby, models of different abstraction levels are created that can be used as executable specification, for architecture exploration and integration validation. Nevertheless, they do not take into consideration complex heterogeneous systems with hardware/software interaction. In [8], this scope is extended to the joint simulation of SystemC and its AMS extension implementing a VSP for a CMOS video sensor. Still, no embedded software integration is discussed.

[9] presents a virtual prototype for a MEMS sensor hub including the firmware, although their contribution is more focused on an automatic firmware generation flow. Thereby, the sensor element only feeds through prerecorded data from a CSV file at a specified data rate. Thus, the signal flow is somehow unidirectional, and their virtual system prototype could not handle tight sensor element-firmware interaction.

Our work can be best seen as a continuation of [10]. There, a high-level model for a sensor system including accelerometer, magnetometer and gyroscope has been developed using SystemC AMS. They describe the usability limits of widely used high-level modeling languages as Matlab Simulink and the AMS extensions for VHDL and Verilog when the validation scope is extended towards embedded software. To overcome those limits, they propose a hardware/software virtual prototype using SystemC, its AMS extension and embedded software in C/C++. Their work focuses on the implementation details of this virtual prototype and the comparison of different modeling styles using different models of computation (MoCs). In our work, we investigate the applicability of such VSPs to firmware validation in an industrial environment. A case study on a real-world example of a MEMS inertial measurement unit is given for that purpose, and benefits and costs are discussed.

III. Fundamentals

A. AMS extension to SystemC

SystemC AMS supports analog/mixed-signal (AMS) modeling as a natural extension of SystemC, which is limited to discrete-time simulations by its event-driven nature. It allows to describe AMS blocks at high abstraction levels and prevents overheads caused by co-simulation frameworks or the integration of low-level AMS models. SystemC together with its extension thus allows fast system-level simulation of heterogeneous systems.

Three models of computation (MoCs) are standardized for that purpose, targeting different descriptions and mathematical formalisms. The timed data flow (TDF) MoC is based on data-flow semantics and assumes signals to be directed and uniformly sampled. Each module has a constant sampling time. The underlying synchronous data flow model supports the generation of a static schedule during elaboration and a fast execution during simulation time. Additionally, it offers an efficient incorporation of linear transfer functions and state-space equations. The linear signal flow (LSF) MoC and the electrical linear network (ELN) MoC implement continuous-time modeling styles using a differential algebraic equation solver. As non-linear differential solvers are generally slow, they are restricted to linear models. The LSF MoC offers linear signal-flow primitives to cover the non-conservative domain. Each model consists of the hierarchical instantiation of those basic blocks. The modeling procedure is similar for the ELN MoC but it uses linear electrical primitives following the conservation laws.

Moreover, SystemC AMS offers the possibility for dynamic time step assignment in TDF modules and frequency-domain simulation. Thereby LSF and ELN models inherently offer a frequency-domain implementation. For TDF modules, the time-domain and frequency-domain behavior must be implemented independently.

B. Inertial Measurement Units

An inertial measurement unit classically consists of an accelerometer for acceleration measurements and a gyroscope for angular rate measurements. In the following, the basic working principles of those devices are explained shortly.

An accelerometer measures accelerations in terms of capacitive changes caused by a deflection of the moving electrode in the presence of an acceleration force. Physically, it can be modeled as a damped spring-mass system. The suspended mass works as movable electrode. Its movement depending on the acceleration force $F = ma$ with mass m and acceleration a is described by a second-order differential equation as follows

$$m\ddot{x} + b\dot{x} + kx = F \tag{1}$$

where b denotes the damping coefficient and k the spring constant. This corresponds to a classical second-order proportional delay (PT2) element in control technology. The transfer function (TF) in the frequency domain then calculates to

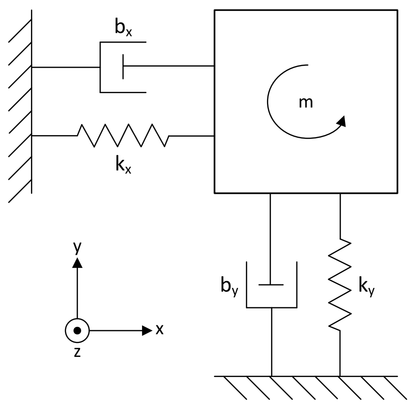


Fig. 3: Physical model of a general single-axis gyroscope. It consists of a damped spring-mass system.

$$H(s) = \frac{\omega_0^2}{s^2 + s\omega_0/Q + \omega_0^2} = k\frac{X(s)}{F(s)} \tag{2}$$

with ω_0 being the resonance frequency and $Q = \frac{\sqrt{mk}}{b}$ the quality factor. As transfer functions can be easily implemented in SystemC AMS, (2) becomes the core of our model.

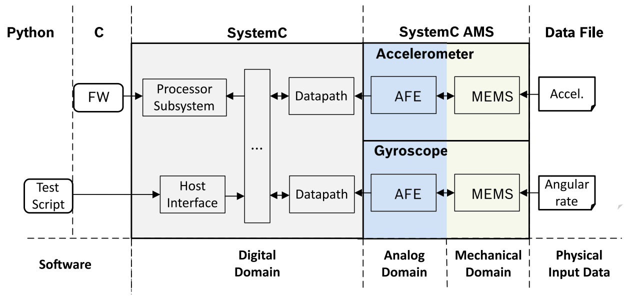
A MEMS vibratory gyroscope is based on the same mechanical fundamentals. Additionally, it relies on the Coriolis effect that occurs when an oscillating suspended mass is fixed to a rotating body. We extend our physical model with a second spring and damper system as illustrated in Fig. 3. The differential equation system calculates to

$$\begin{aligned} m\ddot{x} + b_x\dot{x} + k_x x &= F_x \\ m\ddot{y} + b_y\dot{y} + k_y y &= F_y \end{aligned} \tag{3}$$

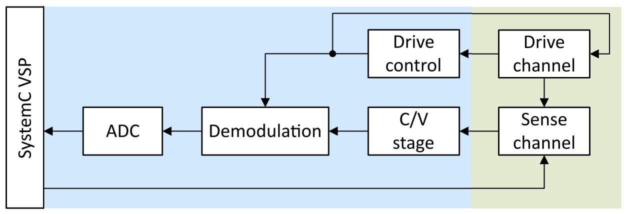
Here we assume a perfect decoupling of the two axes. Let x be the drive direction. Using a phase-locked loop and an amplitude controller, the drive mass is kept in an oscillation of constant amplitude and frequency. The frequency is chosen to be the resonance frequency to achieve highest gain. If an angular rate is applied to the system in z-direction, the Coriolis force calculates to

$$\begin{aligned} \vec{F}_{Cor} &= 2m\vec{v}_x \times \vec{\Omega}_z \\ &= -2mv_x\Omega_z\vec{e}_y \end{aligned} \tag{4}$$

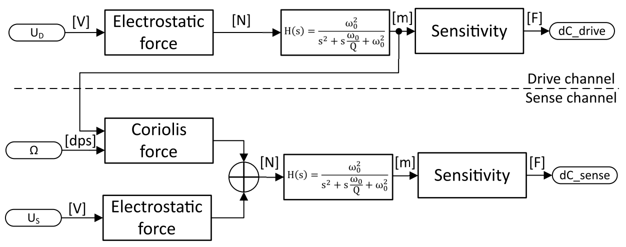
with $\vec{v}_x$ being the vectorial drive velocity and $\vec{\Omega}_z$ the angular rate. The emerging force in (negative) y-direction causes a displacement of the sense mass which changes the capacitance between the moving electrode and the fixed top and bottom electrodes. The ASIC converts the capacitive change into a voltage and demodulates it with the carrier, i.e. the drive velocity. Thus, the output is a voltage that is proportional to the angular rate in the idealized case.



(a) Top-level overview of the heterogeneous model.



(b) SystemC AMS model overview.



(c) MEMS model overview.

Fig. 4: Block-level overview of the virtual system prototype on various layers.

IV. Model Implementation

Fig. 4a shows a simplified block diagram of the overall system model highlighting the different domains. The SystemC model consists of 28 modules and more than 15000 lines of code, ranging from the digital data path up to the host interface. All of them are combined within one top-level module to allow easy instantiation. Data are transferred via TLM (transaction-level modeling). A driver connects the host interface of the model instance to the test script written in Python, i.e. the test script acts as a host that can interact with the prototype through the same channels as a real host can interact with the hardware. As SystemC is the de-facto standard for virtual digital prototypes and our approach is focused on the heterogeneous modeling with AMS extensions, we keep that rough introduction and focus on the implementation of the SystemC AMS model in the following. The model is intended for high simulation speed and comprehensive behavior. Thus, it intentionally misses some non-idealities that are not relevant for a first validation of the firmware algorithm. If the model gets too complex, it deteriorates the developer's possibilities to detect malfunctions fast and easily.

The SystemC AMS model has been developed using the commercial tool CoSIDE by COSEDA Technologies [12]. It allows the automatic code generation of all structural parts. Moreover, the code for hierarchical models is automatically generated when they are connected in a block-level scheme on the visualization dashboard. The MEMS accelerometer is modeled according to the physical basics denoted in section III-B. The transfer function of the PT2 element is modeled in the LSF MoC. It is surrounded by TDF modules for the electrostatic force generation and the calculation of the capacitive change. Please note that it is also possible to specify transfer functions in TDF modules directly, but we decided to test different MoCs for our approach. The two MoCs are connected using the standardized converter ports of SystemC AMS. The implementation of the PT2 element in CoSIDE representing the damped harmonic oscillator is given in Fig. 5. Moreover, a saturation effect is modeled to denote the physical limit of the deflection amplitude of the MEMS. The analog frontend consists of a C/V stage that converts the capacitive values via charge integration to corresponding voltage levels. Afterwards, the voltage is fed through a Sigma-Delta analog-to-digital converter (ADC) and a filtering stage. The setup is equal for all three channels.

A block-level diagram of the gyroscope AMS model is given in Fig. 4b. The MEMS model divides up into the drive channel and three sense channels, one per axis. The output of those channels is a differential capacitance similar to the accelerometer case. A model subset is illustrated in Fig. 4c. Each block describes one SystemC AMS module and the signal units are highlighted. The primitive blocks can be reused from the accelerometer model. The electrostatic force blocks calculate the electrostatic force acting on the drive/sense mass depending on the applied voltages and the sensitivity. The transfer function represents the relation between applied force and displacement. Finally, the differential capacitance is calculated from the displacement and the geometry dependent sensitivity. The analog front end converts the capacitive change into a voltage using the already mentioned C/V stage. As the sense signals are modulated with the drive velocity by the Coriolis effect, a demodulation signal with similar phase and frequency is provided by the ASIC to demodulate the signals accordingly. To suppress their higher frequency components, the demodulation stage also implements a low-pass filtering. Afterwards, the signals are fed through the ADC stage. The drive control consists of a phase-locked loop and an amplitude controller to keep the drive mode in permanent oscillation. The complexity of the basic modules is low as they are kept general to keep high simulation speed and allow easy reuse. The main functionality of TDF modules is specified in their processing-function. For the electrostatic force block it is exemplarily given below.

```
void mems_e_force::processing()
{
  double diff_p = U_cm.read() - U_p.read();
  double diff_n = U_cm.read() - U_n.read();
  F_out.write(0.5 * (p.sens_p * diff_p*diff_p
      - p.sens_n * diff_n*diff_n));
}
```

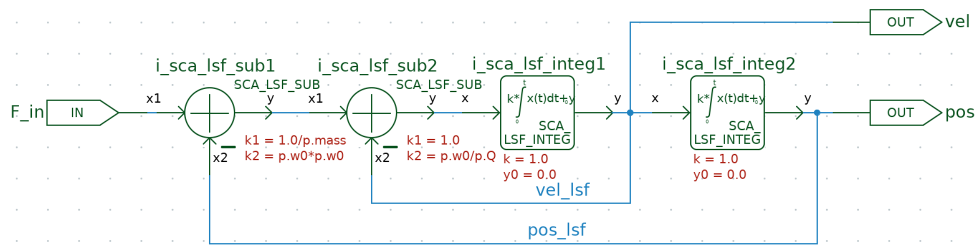


Fig. 5: Implementation of the damped harmonic oscillator in COSIDE. It is modeled in the LSF MoC.

The data transfer from the sensor frontend towards the digital data path is easily done using one of the pre-defined converter ports of SystemC AMS. The SystemC AMS model can be homogeneously integrated with pure SystemC models due to the fact that SystemC AMS is an extension of SystemC and provides interfaces which permit the binding of SystemC signals.

V. CASE STUDY: FIRMWARE VALIDATION

A. Accelerometer

Our test case for the accelerometer model focuses on the validation of a built-in self-test (BIST) procedure. In presence of an acceleration, the suspended mass is deflected. During the BIST, a deflection is generated by an electrostatic force instead. When the excitation stops, the mass returns to its rest position. The firmware enables or disables the test mode that applies a constant electrostatic force to deflect the mass. In the pure SystemC case, the offset generation is shifted from the mechanical part (which is not modeled) into the ASIC to allow its implementation in SystemC. If the test mode is enabled, a constant offset is added to the sensor data fed from the input file. During the test procedure, the offset is enabled and disabled three times resulting in the rectangular signal shown in Fig. 6. For our mixed-signal approach, we generate a voltage signal that is fed to the MEMS instead of specifying the offset directly. Within the AMS model, the voltage is used to calculate the electrostatic force that acts on the suspended mass. It is now possible to extract the timing behavior for the test procedure. Proper intervals for the force application are crucial to get the full output swing. In our example, the second and third excitation times are not long enough to fully deflect the mass to its maximum.

B. Gyroscope

1) Introduction to the Test Case: As the test case for the accelerometer is relatively simple, it cannot highlight all benefits of the full-stack prototype. We thus present another use case with higher complexity in the following. It deals with the gyroscope model. Again, electrostatic excitation is conducted, and the sensor behavior is evaluated for self-test purposes. However, in contrast to the accelerometer case, the sequence of stimuli is not fixed in advance. Instead, a firmware algorithm evaluates the response to the initial stimulus and calculates the next stimulus based on the results. This procedure is repeated

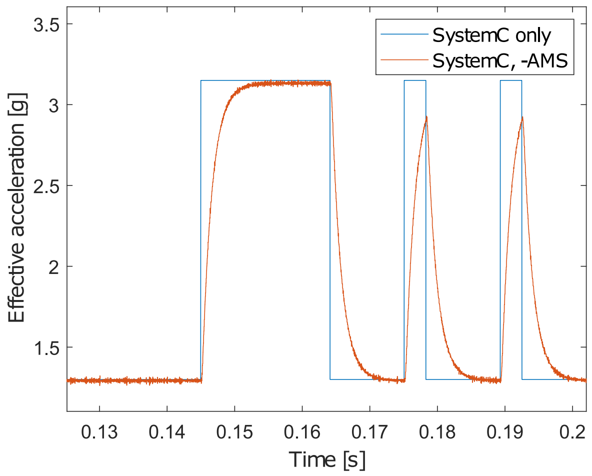
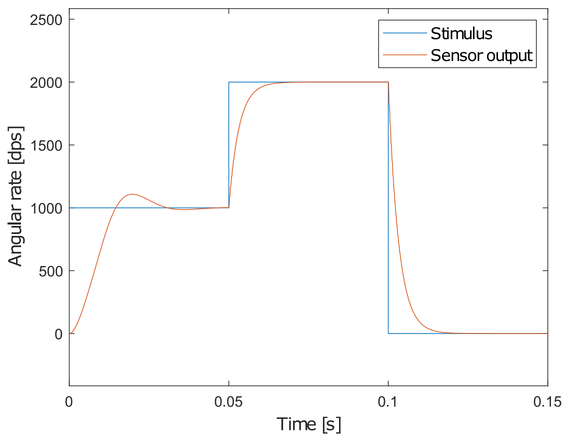


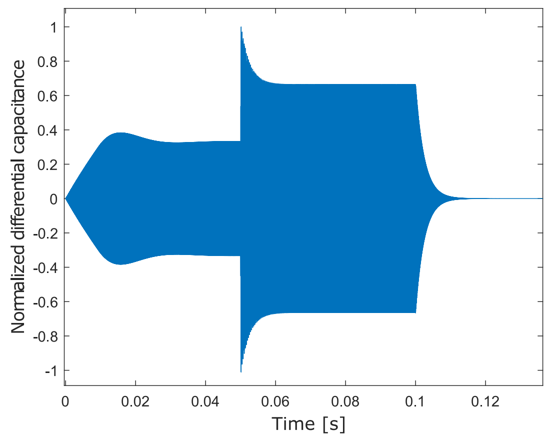
Fig. 6: Simulated sensor data during the self-test with and without sensor element modeling.

several times until the proper functionality of the device is ensured. In the following, the model of the sensor element is validated first and the firmware test case is applied to it in a second step.

2) Model Validation: In order to check that our sensor element model reacts correctly to an applied stimulus, we created a test setup with a piecewise constant angular rate input and checked whether the output voltage of the front end matches the expected target. Fig. 7a shows the applied angular rate Ω over time. The sensor output is multiplied with a conversion factor to get an angular rate value instead of the output voltage. This allows easy comparison to the stimulus here. Different effects are visible in that measurement. First, the drive oscillation needs to start-up and reach its final amplitude. Thus, the sensor does not match the target angular rate in the very beginning. Moreover, it cannot follow the instantaneous changes of the input signal. Instead, it shows a transient behavior to switch from one rate value to the other. Here, one possible error source for the upcoming firmware validation becomes visible. If the measurement samples are taken before the system has settled, it will deteriorate the accuracy of the self-test algorithm. Fig. 7b shows the corresponding output of the MEMS for reference. Due to the modulation caused by the Coriolis effect, the differential capacitance is ringing at the

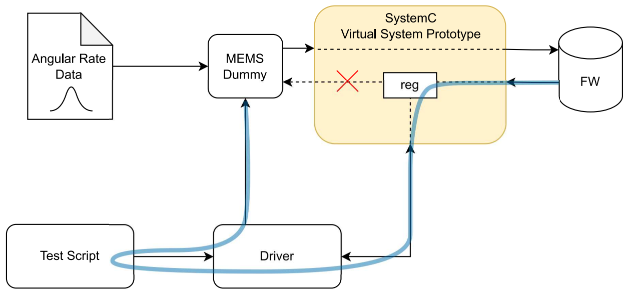


(a) Piecewise constant angular rate stimulus and the corresponding reaction of the sensor element.

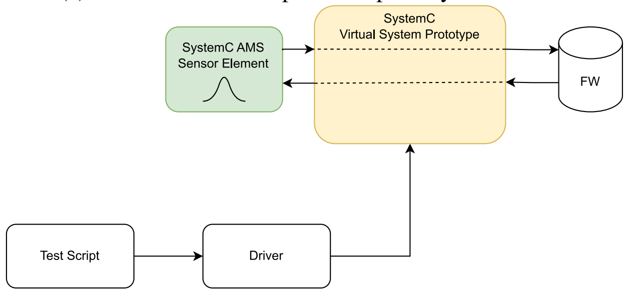


(b) Differential sense capacitance normalized to its maximum value.

Fig. 7: Simulation results for the model validation.



(a) The validation setup for the pure SystemC VSP.



(b) The validation setup for the full-stack VSP.

Fig. 8: Overview of the validation setup highlighting the data flow.

drive frequency. The amplitude is normalized to the maximum value.

3) Firmware Validation: After the functionality of the AMS model has been validated, it is connected to the SystemC model. Now the full stack from MEMS to firmware is reached to allow MEMS/firmware interaction patterns. We tried to match the interface between the AMS model and the digital part as close as possible to the real implementation. When applying firmware tests this pays off as it enables us to easily implement the test cases based on the knowledge about the hardware implementation. Moreover, the test case developed for the virtual prototype can be reused without much modification when the hardware becomes available.

In the following, we will compare the test procedure for a VSP without dedicated MEMS model to our approach to show the benefits of our methodology. Fig. 8 illustrates the differences between the two test setups. In the pure SystemC case, the MEMS is only given as a dummy module which shifts through data from an input file. As the stimuli are not known a priori, the input file contains a look-up table (LUT) of sensor outputs for different input settings. The test script can interact with the VSP via the driver as described in Sec. IV. It reads out the information on which stimulus should be applied and controls the dummy module of the MEMS to read the corresponding entry of the LUT. Please note that there is no effective path between firmware and MEMS. All interactions must be handled via the driver controlled by the test script as highlighted in Fig. 8. It requires a careful synchronization between test procedure and firmware function.

In contrast to that, the integration of the MEMS model allows interactions between MEMS and firmware through the virtual prototype. The MEMS model implements the translation from the voltage stimulus provided by the sensor front end to the electrostatic excitation. Thus, it provides the corresponding response without the need for an input file. Consequently, the test script does not need to interact with the virtual prototype during the execution of the firmware feature. It only sets the initial configuration and evaluates the result of the test case after the algorithm has stopped. The error sources of the test setup are thus mainly limited to the VSP. If the VSP is carefully verified, confidence that a test failure indicates a firmware bug is high. In the pure SystemC approach, the correct test script is crucial to get a firmware execution that corresponds to the hardware case. As the test script is unique for each firmware function, the question of test script validation arises for each feature separately whereas the validation of the VSP is valid for all test cases. If the test case fails, the heterogeneous VSP allows deep insights into the system behavior. If tracing is enabled, the procedure can be visualized helping the engineer to find the problem. In

TABLE I: Comparison of the testing procedures with pure SystemC or full-stack VSP

	Virtual Prototype Version	
	Pure SystemC	**Full-stack**
File input	Hardware measurements	None
Complexity of test case	High	Low
Error sources	Test case, VSP	VSP
Insights into system behavior	Limited to algorithmic, no physical insights	Algorithmic and physical insights
Timing errors covered	No	Yes
Reusability for HW	Low	High
Confidence in test result	**Low**	**High**

this aspect, firmware validation with VSPs is advantageous to hardware tests where insights into the system are very limited. Using the full-stack prototype with AMS part, timing errors can be found that have not been covered before. For the proposed firmware feature, it is crucial to wait until the system has settled onto the new stimulus before the sensor values can be read out. Otherwise, transient processes can deteriorate the accuracy of the measurement. The benefits of the mixed-signal modeling approach are summarized in Table I.

C. Performance Analysis

The last sections have exemplarily shown the advantages gained by the mixed-signal VSP. Nevertheless, they are obtained at the expense of increased simulation time as the size and complexity of the virtual prototype increases. To assess the value of our approach, we quantify the simulation overhead. We measured the wall-clock time for executing the different firmware test cases for the pure SystemC VSP and the full-stack prototype. As the accelerometer model is small compared to the gyroscope model, we thereby get results for two different stages of AMS model complexity. The execution behavior for the gyroscope self-test is not exactly the same for both prototypes as it dynamically reacts onto the previous measurement results. Not only the wall-clock time but also the simulated time is depicted in Table II for that reason. The ratio is then normalized to the simulated time to allow a fair comparison between the two approaches that is independent of the test case. The measured values are moreover split into initialization procedure and firmware procedure. During the initialization procedure, the virtual prototype is configured for the following test case (boot the processor subsystem, enable SPI, etc.). Between those configuration steps, significant waiting times occur. This explains the comparatively large overhead during this section for both AMS models as the sensor elements are running in full detail during the idle states of the test script. The firmware procedure describes the part of the test script, where the firmware feature is running. Here, the overhead becomes smaller. We observe a factor of 2.5 times slower for the accelerometer and about 4.8 times slower for the gyroscope. As expected, the gyroscope runs slower than the small accelerometer model. Nevertheless, both models are able to increase the wall-clock time only by a factor less than five. Considering the benefits for the test evaluation discussed in section V-B3, the performance overhead is reasonable to take in our opinion, especially as finding errors in late design stages is extremely time-consuming. The analysis shows that it makes sense to disable the AMS models when they are not of interest. For example, raising them at the end of the initialization procedure could save significant simulation time. Working on improvements of the model efficiency, we furthermore expect the overhead to decrease.

VI. Conclusion

This paper presented a heterogeneous virtual prototype of an inertial measurement unit in SystemC and SystemC AMS. It enables effective firmware validation even if the firmware function is tightly coupled with the sensor behavior. This has been demonstrated in an industrial case study on the validation procedure for the BIST in MEMS accelerometers and an advanced self-test in MEMS gyroscopes. The approach offers deep insights into the system behavior and permits to find timing errors. As the interaction patterns match closely the hardware function, validation is executed without the creation of complex and error-prone test cases. The developer pays for these advantages with increased simulation time and the maintenance of the SystemC AMS model. Nevertheless the simulation time for common test cases in a state-of-the-art IMU stays in the range of several minutes and the reuse of the AMS models is high due to their abstraction level.

References

[1] J. Rudolf, D. Gis, S. Stieber, C. Haubelt and R. Dorsch, "SystemC Power Profiling for IoT Device Firmware using Runtime Configurable Models," 2019 8th Mediterranean Conference on Embedded Computing (MECO), 2019, pp. 1-6, doi: 10.1109/MECO.2019.8759994.

[2] F. Pêcheux, C. Grimm, T. Maehne, M. Barnasconi and K. Einwich, "SystemC AMS Based Frameworks for Virtual Prototyping of Heterogeneous Systems," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1-4, doi: 10.1109/ISCAS.2018.8351864.

[3] J. M. R. Velzquez, F. Mailly and P. Nouet, "A generic model for sensor simulation at system level," 2018 Symposium on Design, Test, Integration & Packaging of MEMS and MOEMS (DTIP), 2018, pp. 1-4, doi: 10.1109/DTIP.2018.8394198.

[4] Y. Andryakov, A. Anikina, Y. Belyaev, A. Belogurov, D. Kostygov and D. Puzankov, "ASIC and MEMS co-design methodology," 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW), 2016, pp. 120-123, doi: 10.1109/EIConRusNW.2016.7448136.

[5] M. Shafique, A. Menon, K. Virk and J. Madsen, "System-Level Modeling and Simulation of MEMS-based Sensors," 2005 Pakistan Section Multitopic Conference, 2005, pp. 1-6, doi: 10.1109/INMIC.2005.334503.

[6] F. Pêcheux, C. Lallement and A. Vachoux, "VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, no. 2, pp. 204-225, Feb. 2005, doi: 10.1109/TCAD.2004.841071.

[7] M. Barnasconi and S. Adhikari, "Invited: ESL design in SystemC AMS," 2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC), 2017, pp. 1-5, doi: 10.1145/3061639.3072951.

[8] F. Cenni, S. Scotti and E. Simeu, "A SystemC AMS/TLM platform for CMOS video sensors," Proceedings of the 2011 Conference on Design & Architectures for Signal & Image Processing (DASIP), 2011, pp. 1-6, doi: 10.1109/DASIP.2011.6136873.

[9] J. Rudolf, M. Strobel, J. Benz, C. Haubelt, M. Radetzki and O. Bringmann, "Automated Sensor Firmware Development - Generation, Optimization, and Analysis," MBMV 2019; 22nd Workshop - Methods and Description Languages for Modelling and Verification of Circuits and Systems, 2019, pp. 1-12.

TABLE II: Comparison of the Simulation Performance for the Pure SystemC and Full-stack VSP

		Simulation Times					
		Initialization Procedure		**Firmware Procedure**		**Test Execution**	
		Wall-clock time	***Simulated time***	***Wall-clock time***	***Simulated time***	***Wall-clock time***	***Simulated time***
Gyr.	***Pure SystemC***	1.3 s	0.136 s	113.0 s	1.923 s	114.3 s	2.059 s
	SystemC, -AMS	20.6 s	0.136 s	487.1 s	1.739 s	507.7 s	1.876 s
	Ratio normalized to sim. time	15.85	-	4.77	-	4.88	-
Accel.	***Pure SystemC***	0.173 s	0.160 s	0.303 s	0.045 s	0.476 s	0.205 s
	SystemC, -AMS	1.655 s	0.160 s	0.742 s	0.045 s	2.397 s	0.205 s
	Ratio	9.59	-	2.45	-	5.04	-

[10] 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), 2015, pp. 1-6, doi: 10.1109/DTIP.2015.7160972.

[11] K. Einwich, ”Introduction to the SystemC AMS extension standard,” 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems, 2011, pp. 6-8, doi: 10.1109/DDECS.2011.5783036.

[12] Coseda Technologies GmbH. Coside. http://www.cosedatech.com/coside-overview.