2020 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Boosting mixed-signal design productivity with FPGA-based methods throughout the chip design process

Gabriel Rutsch, Infineon Technologies AG, Neubiberg, Germany (Gabriel.Rutsch@infineon.com)

Simone Fontanesi, Infineon Technologies Austria AG, Villach, Austria (Simone.Fontanesi@infineon.com)

Steven G. Herbst, Stanford University, Stanford, USA (sherbst@stanford.edu)

Steven Tan Hee Yeng, Infineon Technologies Austria AG, Villach, Austria (Hee-Yeng.Tan@infineon.com)

Andrea Possemato, Infineon Technologies Austria AG, Villach, Austria (Andrea.Possemato@infineon.com)

Gaetano Formato, Infineon Technologies Austria AG, Villach, Austria (Gaetano.Formato@infineon.com)

Mark Horowitz, Stanford University, Stanford, USA (horowitz@ee.stanford.edu)

Wolfgang Ecker, Infineon Technologies AG, Neubiberg, Germany (Wolfgang.Ecker@infineon.com)

***Abstract*—In this paper, we show the impact on overall design productivity when employing FPGA-based prototyping methods throughout the mixed-signal chip design process. We made use of an open source modeling framework for generating synthesizable functional models of analog behavior that can be mapped on Xilinx FPGAs from an abstract model specification. In order to increase automation, ease-of-use and configurability when creating and working with FPGA-based prototypes, we used an open source FPGA automation framework. Two FPGA-based prototypes for a smart magnetic sensor used in automotive applications were developed and applied during product definition, pre-silicon verification, post-silicon verification and design-in activities. One of the prototypes is intended to be part of a Hardware-in-the-Loop setup; it is real-time capable and small enough to be integrated into a customer's system. The other prototype serves as a computing platform and enables conducting parameter sweeps, regressions and interactive simulations. Compared to state-of-the-art CPU-based simulation, we were able to reduce regression runtime from roughly a month to hours. Additionally, activities related to product definition, customer design-in and post-silicon verification were conducted earlier and with more confidence in the application.**

***Keywords—FPGA-based simulation, Hardware-in-the-Loop prototyping, mixed-signal emulation, chip design process***

## I. INTRODUCTION

Current automotive trends such as autonomous driving, electrification, and the idea of shared, connected and Over-The-Air updated cars are disrupting the automotive world [1]. Even for standard components like engine, transmission and wheel speed sensors [2, 3], new use cases arise and new requirements need to be satisfied. OEMs and First Tier suppliers want to try new electronic control unit (ECU) algorithms, system architectures and sensing solutions directly in the field. Therefore semiconductor suppliers are requested to share virtual and hardware prototypes that can be connected to and communicate with the complete system. In order to be able to provide a prototype to customers that can reliably emulate the product even before first silicon is available, it is necessary to gain a high level of confidence in the product's definition and implementation early on.

This paper is about a new methodology to meet these customer requirements. FPGA-based techniques are utilized to improve efficiency and effectiveness of the chip design process, which are the two key features to measure design productivity. Efficiency is the ratio between benefits, including technical value of the design and how much development time it took, and cost (capital, labor and overheads). How well a desired effect or outcome is achieved, is considered the effectiveness of a method. Benefits shown in this paper through making use of FPGA-based techniques, such as improved confidence in the application and design, faster alignment with customers and an earlier start or shortened duration of an activity, either have a positive effect on efficiency or effectiveness. [4]

For purely digital products, making use of FPGA-based Hardware-in-the-Loop (HIL) prototypes has been an established and successful solution in the past, as it provides more system simulation performance and confidence in the product compared to simulation of a purely virtual prototype (VP). Utilizing FPGA-based prototypes similarly for mixed-signal chip development, however, imposes new challenges having to do with configurability, scalability, resource utilization and simulation performance [5-7]. Our work addresses these issues by providing automated methods to easily set up, debug, and control an FPGA-based HIL prototype.

In addition, we developed an FPGA-based simulation prototype (FSP) that served as a computing platform without external hardware connections. This computing platform is comparable to a system simulation environment such as MATLAB/Simulink with respect to usability and configurability, and was used to conduct parameter sweeps, regressions and interactive simulations.

## II. Related Work

Frank Nothaft et al. introduced a pragma-based method to convert floating to fixed-point datatypes, enabling the use of a proprietary emulation platform for the pre-silicon firmware development of a large cellular modem [5]. In this paper we consider how mixed-signal emulation can be applied to several stages of the chip development process, in addition to firmware development, including customer-facing prototypes. Consequently, we did not target a proprietary emulation platform, but rather small, low-cost FPGA boards that can easily be sent to customers and attached to their reference designs.

In that vein, Simone Fontanesi et al. showed a methodology that allows a smooth transition from an idea to a real hardware prototype [8]. Starting from a SystemC-AMS [9] model of a magnetic sensor application verified with simulations in MATLAB [10], the SystemC model of the digital core was automatically translated to synthesizable RTL code using High Level Synthesis (HLS) [11] and mapped onto a low-cost FPGA board [12].

Our work builds on Fontanesi's contributions by applying a method for automatic translation of analog functional models to synthesizable formats, which improves usability and scalability. Further, it is inspired by the idea to represent analog behavior in state-space and use Zero-order-Hold as a discretization method [6, 7]. Through an additional layer that serves as abstract model specification, reuse and reconfiguration of the model is made easier.

## III. Application

This paper focuses on FPGA emulation in the context of an application from the automotive world: magnetic sensors [13]. These sensors provide a contactless, robust and low-cost solution to measure the position, the rotational speed and the rotational angle of moving parts in wheels, steering systems, transmissions and engines [3]. Typically, a magnetic encoder is applied to the rotating element, while the sensing element scans the magnetic field variation generated by the rotation. Alternatively, a back-bias magnet can be used to generate a constant field that is modulated by the rotation of a steel wheel. These magnetic fluctuations are sensed using various techniques such as the Hall principle [14], Giant Magneto-Resistance [15], Tunnel Magneto-Resistance [16] and Anisotropic Magneto-Resistance [17]. Figure 1(a) shows the many places where magnetic sensors are used in a car, with integration details for one such application, the Anti-lock Braking System (ABS), shown in Figure 1(b).

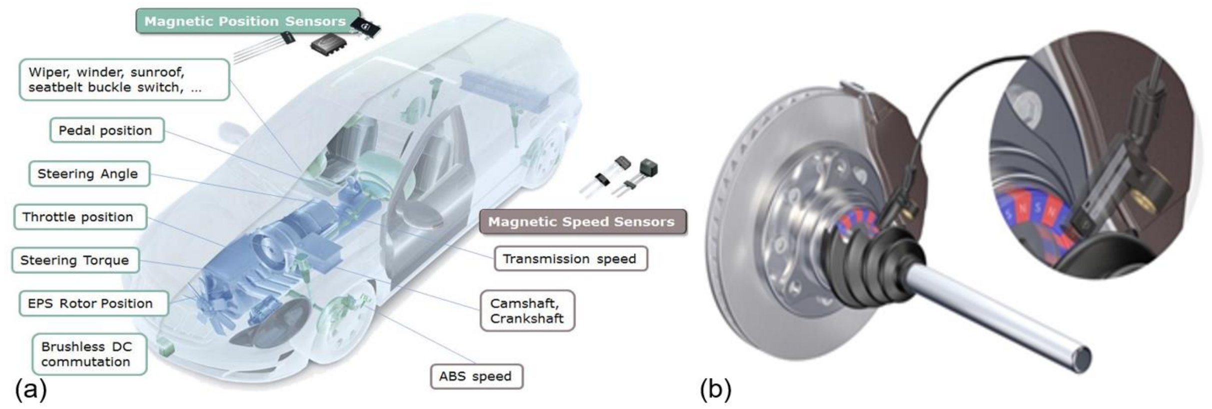

Figure 1: (a) Magnetic sensor Applications in a car; (b) Magnetic speed sensor as part of an Anti-lock Braking System application [17]

The sensors are normally connected to a local application-specific ECU via cable. They communicate with the local ECU via voltage or current modulation and transmit information such as speed, direction of rotation and safety status of the sensor. Figure 2 shows a simplified block diagram of such an application.

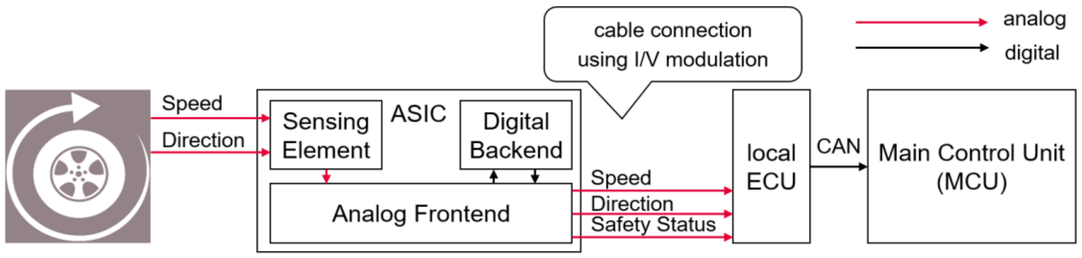

Figure 2: Block diagram of a typical magnetic sensor application

Automated driving and electrification are bringing new requirements to these applications. For instance, higher resolution is required for new ABS sensor use cases such as automated parking and driving, and increased magnetic stray field robustness is needed for transmission sensors. It is important to assess these kinds of new requirements with prototypes, so as to align properly with customers before and after the development of the new ASIC.

## IV. FPGA Prototypes

We developed two FPGA prototypes for the magnetic sensor application; one for interaction with real hardware and one for running fast simulations of the entire application. For both prototypes, the open-source Python-based modeling framework MSDSL [19] was applied to generate synthesizable models for the analog blocks. MSDSL supports both fixed-point and floating-point arithmetic and consequently makes it easy to develop synthesizable functional models of analog behavior. While Xilinx FPGA boards are the default target, other FPGAs and even commercial emulators can be targeted with minimal effort, because MSDSL generally uses vendor-neutral HDL as opposed to vendor-specific IPs or features.

In addition to MSDSL, the Python-based open-source automation framework for FPGA emulation ANASYMOD [20] was used. ANASYMOD offers automation of Xilinx FPGA project creation, functional model generation via MSDSL, bitstream generation and the possibility to run simulations either via CPU-based simulator or on the FPGA. This greatly eases version control of an FPGA prototype and consequently working together in a larger team. To increase applicability throughout chip development, multiple operating systems are supported, since architecture exploration and concept studies are often conducted on Windows, while ASIC development typically takes place on a Linux system. Different RTL simulators are supported to offer convenient choices for different activities. During model development, fast turnaround cycles between individual simulation runs are needed, making a lightweight RTL simulator a good fit. When running comprehensive regressions however, an RTL simulator with more simulation performance and better UVM support is a better choice. Besides hosting RTL simulations, running regressions and interactive simulations on FPGA is possible. For this purpose, configurable signal tracing and control instrumentation is added to each FPGA project and a Python API allows communication from host PC to FPGA.

### *A. FPGA-based Hardware-in-the-Loop Prototype*

The FPGA-based HIL prototype is depicted in Figure 3. A stimulus is generated by the magnetic encoder wheel and can continuously vary in speed and direction, to emulate the wheel behavior of a car. The laboratory test bench can correctly emulate multiple application scenarios, such as different mechanical air-gaps (distance between sensor and magnetic encoder wheel), displacements, tilts and twists around the nominal mounting position. Depending on the mounting position, the magnetic field seen by the sensor changes, but as long as the sensing element operates within the specified operating range, the ASIC (and the prototype) keeps behaving correctly, providing speed, direction and safety status information to the ECU. Closely attached to the magnetic encoder wheel, the sensing element can either be directly plugged into the power board of the analog front-end (AFE), shown in Figure 3(e), or used as satellite and connected to the AFE via cable. We split the AFE of the sensor into two parts; one part was represented with discrete components on the power board, while the other part was modeled in the FPGA. Besides the AFE's PWM-controlled current interface to the ECU, some additional functionality, which is not part of the ASIC design, was implemented on the power board. For instance, it includes an additional amplifier stage to compensate for effects that occur due to the prototype's structure. The remainder of the AFE, including amplifiers, comparators, offset compensation and a range decoder, was modeled in the FPGA as shown in Figure 3(d). Those blocks compensate changes of input stimuli from the sensing element and are custom tailored to a specific use case, which means that they need to be highly configurable and interchangeable. Together with the magnetic sensor's digital core, those functional models were mapped onto the Zynq-7000 FPGA of the Snickerdoodle board. The models had to run in real time in order to properly interact with physical components external to the FPGA; we met this requirement by discretizing the dynamics of emulated analog blocks to a fixed timestep equal to the clock period of the digital core (~100 ns). We could do this with reasonable accuracy because the time constants of emulated analog blocks were much longer than that period. Communication from sensing element to the AFE's functional models was established utilizing the FPGA's built-in XADC blocks, and a digital signal drove a current interface on the power board, which generated the analog output PWM signal. This split of the magnetic sensor's design allowed us to rapidly implement changes, ranging from a wide architectural modification to a bit-width variation in a register or a new capacity value. We implemented any non-configurable part of the AFE in discrete hardware on the power board, to keep modeling efforts to a minimum. Finally, the PWM signal was connected to a host PC, in order to process and display results.

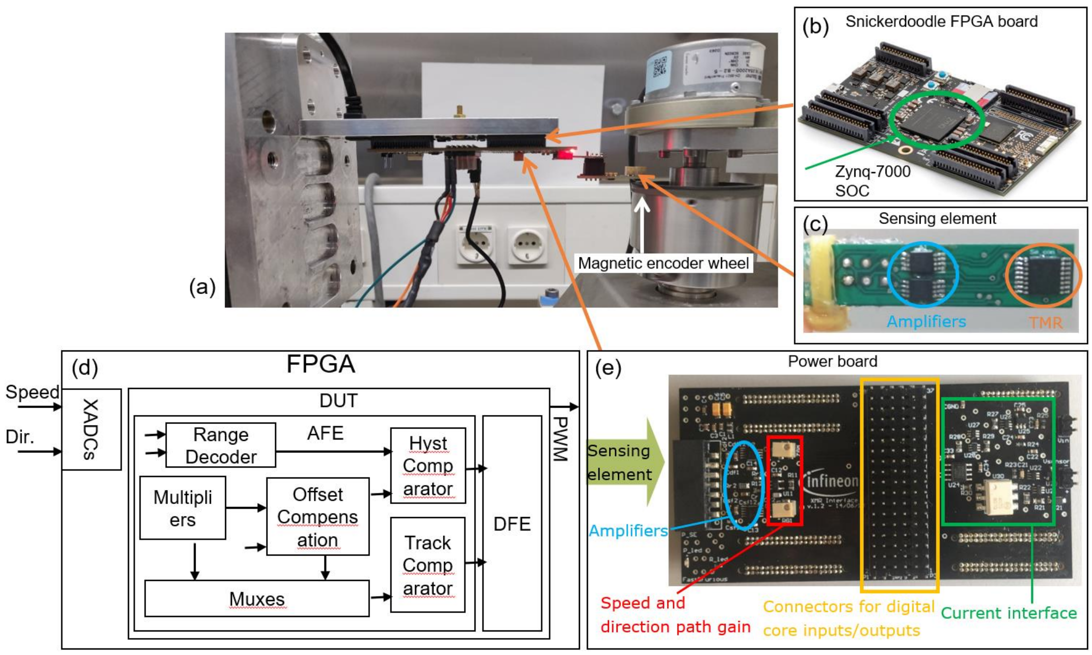

Figure 3: (a) Overview of FPGA-based HIL prototype, (b) FPGA board, (c) sensing element, (d) diagram of FPGA design, (e) power board

## B. *FPGA-based Simulation Prototype*

To perform comprehensive regressions and concept studies, a highly performant simulation environment is needed. The FPGA-based HIL prototype was not suitable for this task, as the regression required rapidly changing stimuli from the sensing element and therefore changing the lab setup dynamically. Therefore, we created an FSP of the magnetic sensor application. Figure 4 visualizes the FSP. Included is the same functional model of the AFE and digital core we presented in the previous subsection, a sinewave generator and a test bench, instrumentation for signal control and tracing, and simulation control infrastructure. Every included part of the application and instrumentation is mapped onto FPGA and there is no physical connection to the FPGA other than to the host PC for data transfer and power supply. Hence, the FPGA serves purely as a computing platform.

To simulate different application scenarios on the FPGA, the sinewave's frequency, amplitude and phase need to be runtime-configurable. In this setup, it is possible to control the sinewave generator even during a simulation run from the hosting PC. Currently, we utilize Xilinx proprietary VIO cores, which are controllable via the Vivado TCL interpreter, to establish the communication to the hosting PC and ANASYMOD. This communication interface is

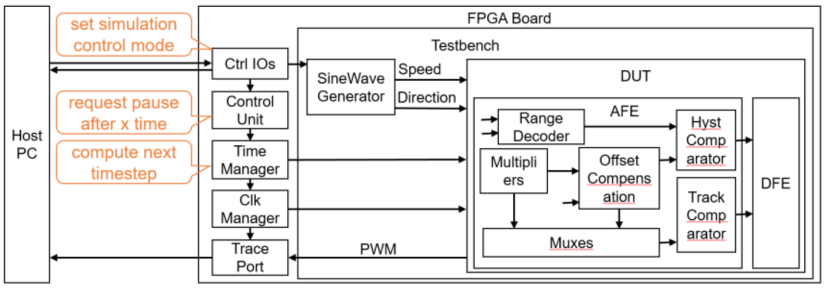

Figure 4: Structure of the FPGA-based simulation prototype

available for all Xilinx FPGAs, making it a good fallback, but we strongly recommend using a USB-UART bridge when possible, as it can reduce communication latency by up to two orders of magnitude. In addition to the VIOs, there is also a waveform capture unit to transfer data from FPGA to the hosting PC. That block is implemented using a Xilinx proprietary integrated logic analyzer (ILA) core, and is also controllable via TCL interpreter. Its bandwidth is significantly higher than that of the VIO interface, making it practical to capture waveforms of emulated analog signals. To emulate realistic application scenarios, the input stimulus needs to be changeable at any given point in time. Therefore, we added an interactive control unit that, in combination with a time manager, can pause the FPGA simulation at any given point in time, while still keeping control via the hosting PC active.

## V. RESULTS

In this section, we first compare the simulation performance of the FSP with that of a CPU-based simulator. We then discuss how we made use of both the FSP and HIL prototype at various stages of the chip design process.

In measuring simulation performance, we ran two test cases, illustrated in Figure 5, using both a CPU-based simulator and the FSP. The first test case simulated the system’s response to forward car motion, during which speed and direction signals are steady sinusoids. The second test case was more dynamic, simulating the rotational vibration that occurs when a car stops. Here, the DUT’s AFE was sent a stimulus consisting of 100 steady sinusoidal periods, followed by 10 vibration periods, and finally another 100 steady periods.

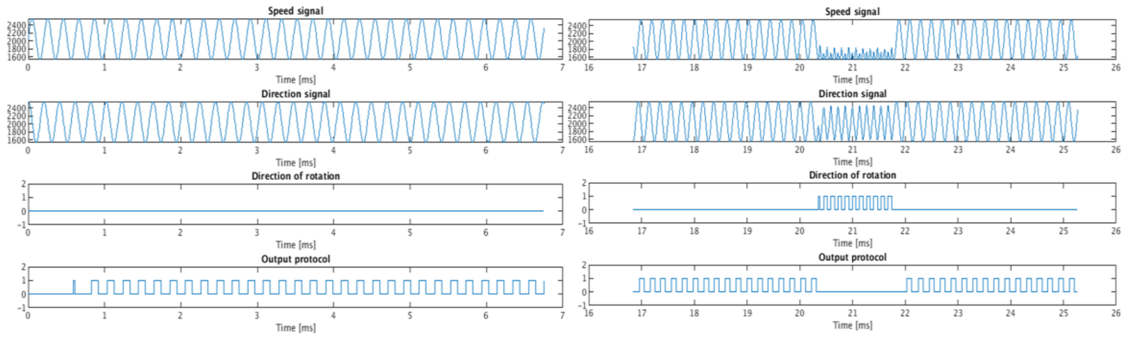

Figure 5: FPGA simulation results for normal rotation (left) and rotational vibration (right) test cases

A comparison of the simulation performance of both methods is given in Table I, with details on the simulation setups in Table II. The FSP ran at 18.35 MHz, with each emulation cycle representing an emulated time increment of 100 ns. That corresponds to 0.5s real time per 1s of emulated time, or 2x faster than real time. Compared to the CPU-based simulation, which would take 1.7 hr to simulate 1 s, this represents a speedup of ~12,500x.

In analyzing the simulation performance of our FSP, it is important to consider several aspects that do not play a role during CPU-based simulation. One of those aspects is bit-stream generation, which took 25 minutes. While this is several times longer than running a single CPU-based test case (e.g., normal or rotational vibration), regression test suites typically consist of a large number of such simulations. Since bit-stream generation only needs to happen once per test suite, its amortized cost is low.  Even for a single simulation, the relative time consumption of bit-stream generation decreases rapidly as simulation length increases, since the throughput of FPGA-based simulation is orders of magnitude higher than that of CPU-based simulation.

Another FPGA-specific concern is the communication between the FPGA and PC, which takes ~130ms for a read and ~360ms for a write operation when employing VIOs. As shown in the rotational vibration testcase, effective simulation throughput is strongly I/O bound. 19.78s is spent on I/O (22x read and 47x write), 2.36s is spent on result post-processing and only 21ms is needed to run the actual emulation on FPGA. Preliminary experiments have shown, though, that the I/O overhead could be reduced by two orders of magnitude by making use of the FPGA board's USB-UART interface. Further gains in communication throughput can be achieved by using the FPGA's PCIe interface, or by employing a commercial emulation platform at the cost of simulation performance.

The last FPGA-specific consideration has to do with results post-processing. Fixed-point numbers represent analog values on the FPGA, which helps to keep resource utilization in check, but makes it difficult to directly interpret their waveforms. As a result, waveforms of analog signals captured by during emulation have to be read, converted to a floating-point format, and written to a new value change dump (VCD) file. In our test cases, this only causes an overhead of about two seconds, but this can grow quickly during debug activities, when a user observes many signals at the same time.

Table I. Result comparison for CPU- and FPGA-based simulation of the magnetic sensor application

| **Test Case** | **Normal rotation** | | **Rotational vibration** | |
|---|---|---|---|---|
| Simulation Platform | FPGA | CPU | FPGA | CPU |
| Simulated Time | 40.7 ms | | 42.13ms | |
| Bitstream Generation | 25 min | - | 25 min | - |
| Simulation Time (total) | 37.03s | 268s | 53.59s | 273s |
| Simulation Launch | 33.53s | 10s | 31.43s | 10s |
| Parameter Initialization | 1.11s | - | 1.61s | - |
| Simulation Time (effective) | 0.68s | 258s | 18.17s | 263s |
| Simulation Performance (time in s to simulate 1 s) | 0.5 | 6339 | 0.5 | 6243 |
| Result Post-Processing | 1.72s | 2.91s | 2.36s | 2.91s |

Table II. Hardware / software specification of simulation environments

| | | | | |
|---|---|---|---|---|
| HW/SW Spec | FPGA Board/Chip | PYNQ-Z1 | FPGA Chip | ZYNQ XC7Z020-1CLG |
| | CPU Name | Intel Xeon E5-2690 v4 | CPU Frequency | 2.6 GHz / 3.5 GHz boost |
| | CPU Cores | 28 | RAM | 252 GB |
| | OS Name | Red Hat Linux 6 x64 | Simulator Version | Cadence Incisive 15.20-s028 |
| | Vivado Version | Vivado v2018.2 (64-bit) | | |

Having demonstrated the simulation capabilities of our FSP, we'll discuss how we made use of the flexibility and power of the FSP, as well as the HIL prototype, in four distinct development stages of a magnet sensor product: product definition, pre-silicon verification, post-silicon verification, and customer interaction (Figure 6).

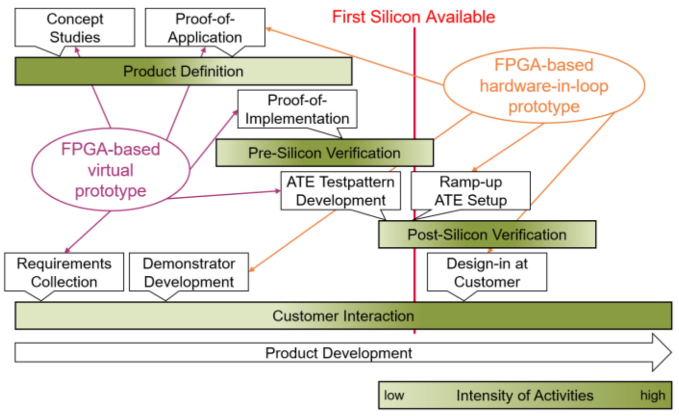

Figure 6: Development process of the magnetic sensor application; color intensity shows the relative effort of each stage over time.

### A. *Product Definition*

The first stage of product development is Product Definition. Here, two activities are dominant: (1) concept studies are performed to identify suitable applications for the product, and (2) customer requirements need to be understood correctly and verified in their applications. As described in Section III, the area of application for magnetic sensors in a car is large and diverse. Therefore, besides the DUT, also the application needs to be adaptable, which made it impractical to employ an FPGA-based HIL prototype. Instead, the FSP fits, because it allowed DUT-related and application-specific parameters to be varied, such as the pitch of a wheel or the external communication protocol.

The FPGA-based HIL prototype was a big benefit during interactions with customers. By attaching the prototype to a customer's reference design, turnaround times when aligning on requirements were shortened and new features could be specified precisely and tested directly in the field long before engineering samples were available.

### B. *Pre-Silicon Verification*

During pre-silicon verification, the main activity is to run extensive regressions and to verify as much of the product's functionality as possible to gain confidence in the design. Therefore, the previously shown simulation performance gains from utilizing an FSP rather that a CPU-based VP were particularly impactful. For applications similar to the magnetic sensor, a common requirement verification strategy involves running roughly 10,000 simulations. Test cases differ in complexity from simple normal rotation to highly dynamic rotational vibrations. Parameters that are dynamically swept here include angular vibration amplitude, frequency and the number of vibration periods. Both FPGA and CPU simulation can be highly parallelized. Since this can be done similarly, the simulation times were measured on sequential execution of all testcases. Performing those simulations on a CPU-based environment would take approximately 30 days. Compared to that, accounting for simulation launch and bitstream generation only once, the time needed to run 10,000 simulations of simple test cases on an FPGA would be 9.75h for VIO-based and 5.25h for USB-UART-based FPGA to PC communication, which is a speedup of 73x and 137x respectively. Test cases with more dynamic stimuli demand more interaction between the FPGA and host PC, which means simulation time to run 10,000 simulations of that kind take 61.5 h (12x speedup) for VIO-based and 7.5h (96x speedup) for USB-UART-based communication. Even employing a VIO-based communication interface between the FPGA and host PC already shows a satisfactory improvement compared to CPU simulation.

### C. *Post-Silicon Verification*

In the post-silicon verification phase, silicon samples of the design are verified. One activity is setting up the Automatic Tester Equipment (ATE). As real hardware is needed to ensure correct wiring between the tester equipment and the DUT. This conventionally starts as soon as a test chip is available. Employing the FPGA-based HIL prototype for that allows an earlier start, which shortens chip development time.

Further, comprehensive test patterns and test cases are optimized towards coverage and minimal testing time, which starts before silicon is available via simulation. Here, simulation throughput is crucial, which means the FSP is the choice to take. One of the key applications of the FSP is in the development of built-in-self-test (BIST) capabilities, which reduces test time and the total number of connections and analog interfaces from DUT to ATE.

### D. *Customer Interaction and Design-in*

Once a customer opportunity arises, the performance of the product in the customer´s system is verified. This could mean testing a new encoder wheel, a sensor orientation different to the one used for product definition or specific magnetic operating conditions (e.g., presence of stray fields). Usually, these customer needs are addressed either via measurement on the customer's system or via simulation, with simulation preferred if numerous different application scenarios need to be assessed. In both cases, the effort can be very high. For instance, the evaluation of a particular wheel geometry during design-in requires several weeks of measurement in the laboratory and simulation of test cases. The FSP is capable of simulating those new use cases, as input stimuli patterns can be changed dynamically. Consequently, it is possible to scan and store the magnetic signal generated by a new wheel and use it as stimulus for the FSP. Shorter execution time compared to the standard approaches also increases the number of conducted test cases and therefore improves confidence in the product. Additionally, since the FPGA-based HIL prototype is attached to reference hardware, the customers can use it to develop and optimize their ECUs.

## VI. CONCLUSION

In this paper, we illustrated the impact on overall design productivity when employing FPGA-based prototyping methods throughout the mixed-signal chip design process. The key challenges that we addressed are representing analog behavior on a FPGA efficiently and easy debug and control of the FPGA-based prototype. In addition, overhead compared to a CPU-based simulation during setup and usage of the FPGA-based prototype was minimized. Therefore, we made use of the open source modeling framework MSDSL for generating synthesizable functional models of analog behavior from an abstract model specification. In order to increase automation, ease-of-use and configurability when creating and working with FPGA-based prototypes, we used the open source FPGA automation framework ANASYMOD. Two FPGA-based prototypes for a smart magnetic sensor for automotive applications were developed and applied during product definition, pre-silicon verification, post-silicon verification and design-in activities. One of the prototypes was intended to be part of a HIL setup, real-time capable and small enough to be integrated into a customer's system. The other prototype served as a computing platform and enabled conducting parameter sweeps, regressions and interactive simulations of the entire application. Besides quantitative benefits, such as a reduction on pre-silicon regression runtime from 30 days to 5.25h compared to CPU-based simulations, we also observed qualitative benefits. Design-in activities at the customer side can start before silicon is available, which also shortens product definition. We could also notice benefits when employing FPGA-based prototyping for post-silicon verification related activities, such as the speedup of ATE test pattern development and an early ATE bring-up. We conclude that employing FPGA-based methods significantly improves design productivity for products similar to the magnetic sensor application. Still, we see potential for further improvement with respect to broadening the area of applicability during chip development, adding more automation for prototype/functional model validation and communication speedup between hosting PC and FPGA.

**Acknowledgement**: The authors would like to thank the Hertz Foundation Fellowship, Stanford Graduate Fellowship and the Federal Ministry of Education and Research (Germany) for supporting this work.

## REFERENCES

[1] F. Kuhnert, C. Stürmer, A. Koster, "Five trends transforming the Automotive Industry", PricewaterhouseCoopers GmbH, 2018

[2] Infineon Technologies AG, "Sensing the world - Sensor solutions for automotive, industrial and consumer applications", April 2016

[3] C.P.O Treutler, "Magnetic sensors for automotive applications", in Sensors and Actuators, Volume 91, Issues 1-2, June 2001

[4] A. H. B. Duffy, "The Design Productivity Debate", Springer, January 1998

[5] F. A. Nothaft, L. Fernandez, S. Cefali, N. Shah, J. Rael, "Pragma-Based Floating-to-Fixed Point Conversion for the Emulation of Analog Behavioral Models", ICCAD, 2014

[6] S. Herbst, B.C.Lim, M. Horowitz, "Fast FPGA Emulation of Analog Dynamics in Digitally-Driven Systems", ICCAD, 2018

[7] T. Bruckner, M. Lorenz, C. Zorn, J. Becker, W.Mathis, M. Ortmanns, "Hardware-Accelerated Simulation Environment for CT Sigma-Delta Modulators Using an FPGA", IEEE Transactions on Circuits and Systems II: Express Briefs ( Volume: 59 , Issue: 8), 2012

[8] S. Fontanesi, G. Formato, T. Arndt, A. Monterastelli, "Fast and Furious: Quick Innovation from Idea to Real Prototype", DVCON 2018

[9] Accellera System Initiative, SystemC, in http://accellera.org/downloads/standards/systemc

[10] MATLAB, in https://de.mathworks.com/products/matlab.html

[11] Catapult® High-Level Synthesis, in https://www.mentor.com/hls-lp/catapult-high-level-synthesis/

[12] Snickerdoodle SoC board, in https://krtkl.com/

[13] S. Fontanesi, S. H. Kim, G. Binder, A. Monterastelli, "Mechanical mounting variation effects on magnetic speed sensor applications", DVCON Europe 2017

[14] R.S. Popovic, "Hall Effect Devices, Second Edition", CRC Press, December 2003

[15] C. Reig, S. Cardoso, S. C. Mukhopadhyay, "Giant Magnetoresistance Sensors - From Basis to State-of-the-Art Applications", 2013

[16] H. Jin, T. Miyazaki, "Tunnel Magnetoresistance Effect", Springer, August 2012

[17] M. J. Haji-Sheikh, K. Allen, "Anisotropic Magnetoresistance (AMR) Magnetometers", in High Sensitivity Magnetometers, Volume 19 of the series Smart Sensors, Measurement and Instrumentation pp 167-199, September 2016

[18] Infineon Technologies AG, "Sensing the world", Infineon Technologies AG, 2016

[19] S. Herbst, "msdsl", 2020, GitHub repository, https://git.io/msdsl

[20] G. Rutsch, "anasymod", 2020, GitHub repository, https://git.io/anasymod