Verification challenges of complex system-on-chip devices

M. Horauer OVE, ASME, IEEE, D. Widhalm, S. Tauner IEEE, S. Mirtl

In recent years rising complexity, shrinking silicon feature sizes, and reduced design cycles have posed new challenges on the verification of modern system-on-chip solutions. To tackle the issues caused by the rising complexity various design and verification languages, as well as methodologies and tools have been introduced. Likewise, new and better physical process models allow for improved simulation of both analog and digital designs. Finally, strict management plans are used to cope with the shrinking design and verification cycles. Despite all these efforts, however, many problems exist in industrial state-of-the-art processes and tools.

This article gives some insights and presents some lessons learned from the design and verification of a recent automotive microcontroller, a complex system-on-chip solution. Based on these findings, a new verification flow is proposed that closes an identified gap between pre-silicon and post-silicon verification.

Keywords: system-on-chip; pre- and post-silicon verification; user-mode testing

Herausforderungen an die Verifikation komplexer System-on-Chips.

Mit der steigenden Komplexität und immer kürzer werdenden Entwicklungszyklen ergeben sich neue Herausforderungen für die Verifikation von modernen, komplexen System-on-Chip-Lösungen. Neue Sprachen, Werkzeuge und Methoden sowie Design-Ansätze auf höherem Abstraktionsniveau, gepaart mit der Verwendung von IP-Cores, ermöglichen es, mehr Funktionalität rascher zu entwickeln. Kleinere Prozessgeometrien und bessere physikalische Simulationsmodelle erlauben es zudem, dies auch auf stets kleinerer Siliziumsfläche mit geringerem Energiebedarf zu realisieren. Der gesamte Entwurfs- und Verifikationsprozess wird durch stringente industrielle Prozessabläufe gelenkt und geleitet. Insbesondere durch das Wechselspiel digitaler und analoger Elemente können einzelne Probleme dennoch erst spät entdeckt werden.

Dieser Beitrag gibt Einsicht in die Verifikation moderner Mikrokontroller und zeigt, welche Lehren aus diesem Prozess gezogen werden konnten. Basierend auf diesen Erkenntnissen wird ein neuer Verifikations-Prozess vorgestellt, der eine festgestellte Lücke zwischen Pre- und Post-Silizium-Verifikation zu schließen versucht.

Schlüsselwörter: System-on-Chip; Verifikation; User-Mode-Test

Received May 5, 2015, accepted June 2, 2015, published online August 15, 2015

1. Introduction

Features like increased functionality, environmental efficiency, safety, security, or comfort drive the development of new electronic devices and solutions. As an obvious implication, modern microcontrollers are turning into complex System on Chip (SoC) solutions. This rising complexity is often tackled by (re-)using a variety of IP cores, crafting and using simulation environments, and the integration of various design and verification languages that enable development at higher levels of abstraction. These approaches often facilitate parallelism of both development and verification efforts and thus pave a way to cope with shorter development cycles.

An example in this regard is the development of recent 32-bit microcontroller derivatives for advanced automotive electronic control units (ECUs). These complex SoC solutions comprise multiple asymmetric super-scalar CPU cores, a multitude of digital and analog peripherals (for example, interfaces to various high-speed buses, ADCs with nearly 100 channels) and many other features. The domain of ECUs is very competitive resulting in a high pace of innovation. New, sophisticated use cases often push the newest technologies to their limits – in fact there is a growing demand for ever more powerful computing systems coupled with very sensitive analog parts that require only a minimum of power consumption in harsh environments. As a result device suppliers and automotive tier-1 and tier-2 suppliers have to work hand in hand in order to meet the stringent requirements imposed by the short development cycles and targeted feature set. In fact new challenges and side effects need to be resolved and dealt with, e.g., ADC readout values get degraded in certain cases when some digital peripherals are in a certain state at the same time. Conveying back the necessary conditions for these effects from the automotive supplier to the verification and the development group of the semiconductor manufacturer, respectively, often proves rather difficult. First of all a user-mode test setup for post-silicon testing has to be crafted in order to reproduce the effects mentioned above within the verification group. Next, there are legal issues (e.g., regarding confidentiality) that need to be resolved. Finally, back annotation from the post-silicon verification to the pre-silicon verification proves difficult due to lack of tool support.

Horauer, Martin, Department of Embedded Systems, University of Applied Sciences Technikum Wien, Höchstädtplatz 6, 1200 Wien, Austria (E-mail: horauer@technikum-wien.at); **Widhalm, Dominik,** Department of Embedded Systems, University of Applied Sciences Technikum Wien, Höchstädtplatz 6, 1200 Wien, Austria; **Tauner, Stefan,** Department of Embedded Systems, University of Applied Sciences Technikum Wien, Höchstädtplatz 6, 1200 Wien, Austria; **Mirtl, Stefan,** Infineon Technologies Austria AG, Siemensstraße 2, 9500 Villach, Austria

The remainder of this article presents the development flow for complex microcontrollers at Infineon in more detail, proposes a concept and describes an implementation that allows closing the gap between pre- and post-silicon verification, and briefly shows an evaluation of this approach.

2. Microcontroller verification flow

While the proposed approach is generic in nature, it was developed primarily for the verification of some of Infineon's current 32-bit microcontrollers.

In order to efficiently achieve high verification coverage the verification activities have to be appropriately planned in advance. The planning process includes the determination of the verification objectives and also outlines how to achieve them. Up to now this planning is done separately for pre- and post-silicon verification which may hamper the exploitation of possible synergies.

In pre-silicon verification a variety of methods (e.g., simulation and formal verification) is used to verify the design and to check it against its specification. At Infineon primary simulation is used in pre-silicon verification which is applied on different levels of abstraction. On module level the basic functionality of each module is checked, mainly by the use of white-box tests. A subsystem is a group of modules whose functionality is tightly coupled (e.g., the ADC and generic timer module (GTM)). On subsystem level the modules are treated as black boxes and the focus is on the interfaces and interactions of these modules, assuming that the functionality of the modules have already been properly verified on module level. Chip- and system-level tests consider all modules of the design. Their main intent is to test realistic use cases (which are sometimes even provided by the customers).

Simulation requires the presence of a suitable testbench. Traditionally, testbenches are mainly written by hand resulting in a relatively high creation time. To lower this effort and to support the simulation process a powerful SoC-level verification environment was developed at Infineon termed *AGENtiX*. It provides front-end tools, C header files, a basic library for functions and other testbench elements for the mixed-language simulation of Infineon's automotive microcontrollers. The testbenches as well as the test cases are written in SystemC allowing to use abstract models of modules and bus functional models (BFMs) for the purpose of hardware/software co-verification. *AGENtiX* is further able to simulate multiple CPUs working in parallel. Additionally, it offers a virtual communication network to control the single modules of the testbench. In *AGENtiX*, analog and mixed-signal modules are included as abstract BFMs also connected to the communication network.

Post-silicon verification additionally focuses on issues not covered by pre-silicon tests (e.g., issues on the real silicon introduced by process technology variations). The tests are executed on real-silicon prototypes, which are most often assembled on an engineering board. In contrast to pre-silicon simulation where all necessary parts are included in the testbench, in post-silicon verification additional lab equipment is needed. Stimuli applied to the design under verification (DUV) are either generated by power supplies (for simple signals) or by pattern generators.

To drive these control and measurement instruments and to automate parts of the post-silicon verification process mainly VBA scripts embedded in Excel sheets, or MATLAB scripts are used. These augmented sheets/scripts gather the measurement results and prepare pass/fail matrices as well as various diagrams. One of the main drawbacks of this approach is that for every product and test setup the Excel sheets or MATLAB scripts have to be adapted manually. Furthermore, only functionality available as VBA or MATLAB scripts can be supported.

3. Related work

Although the gap between pre- and post-silicon verification is a well-known issue, only a few approaches for bridging have been published in the literature so far.

In [1] and [2], Adir et al. present a method to create a common platform for pre- and post-silicon verification. The common platform bases on a so-called *Exerciser*. The name *Exerciser* origins in the function of the program—it exercises the DUV by testing predefined scenarios. It runs on the DUV and provides a simulator-like environment for post-silicon verification. As stated in [1], modern *Exercisers* are self-contained solutions since they are able to generate individual test cases for predefined scenarios, run them on the target and do the checking. For this purpose they are equipped with OS-like services which are required by the test cases. Another advantage of *Exercisers* is that they can run in an infinite loop. Therefore the firmware only has to be loaded once on the DUV and is then capable of performing several tests successively. According to [2] there is a need "to consider pre- and post-silicon efforts as a single effort that is executed on different platforms".

Melani et al. propose an approach in [3] that uses a common description language for creating an environment compatible with pre- and post-silicon verification. This approach enables the reproduction of pre-silicon test cases in a lab environment and features an automatic extraction of the test results which are then compared with the expected values from simulation. To that end, the authors use a common mixed-signal hardware description language (MSHDL), namely VHDL-AMS, to build the simulation environment including not only the system model but also the input signal generation as well as the elaboration of the results. For the lab environment a script extracts all settings required for configuring the DUV, signal generation and data acquisition from the firmware. The basic idea is to augment the firmware with additional information in form of annotations, which defines how to set up and execute the test in any environment. Therefore, the firmware is used as a kind of glue which links pre- with post-silicon verification. The approach gains a reduction of the overall verification time by automating the information exchange between design and test environments.

Both the "*Exerciser*" and the "MSHDL" approach are tailored for a specific area and are not commonly usable. But they offer some valuable considerations and ideas for developing a suitable bridging approach to be used at Infineon Technologies Austria AG.

4. Proposed methodology

As described earlier, Infineon uses a number of methods to verify their products in appropriate ways. During the status quo analysis it was revealed that there are two areas offering great potential for improvements:

1. The flow of information between the individual methods.
2. The simplification of test entry in post-silicon verification.

There was a lack of communication and bridging efforts between the individual pre- and post-silicon verification steps. Furthermore, the automation in post-silicon test case creation and execution needed to be improved. Previously, a lot of effort was spent in recreating basic test cases for each DUV although these basic tests are similar for different products. This effort can be minimized by the use of already existing test cases (e.g., from simulation) and improving code reuse (e.g., by modularization).

To that end, our approach comprises a common platform for the definition of shareable test cases. This allows defining and writing test cases only once, while using them for various verification tasks in different environments. Furthermore, if an error in the test sequence is found, or if the test case is modified, only the common test

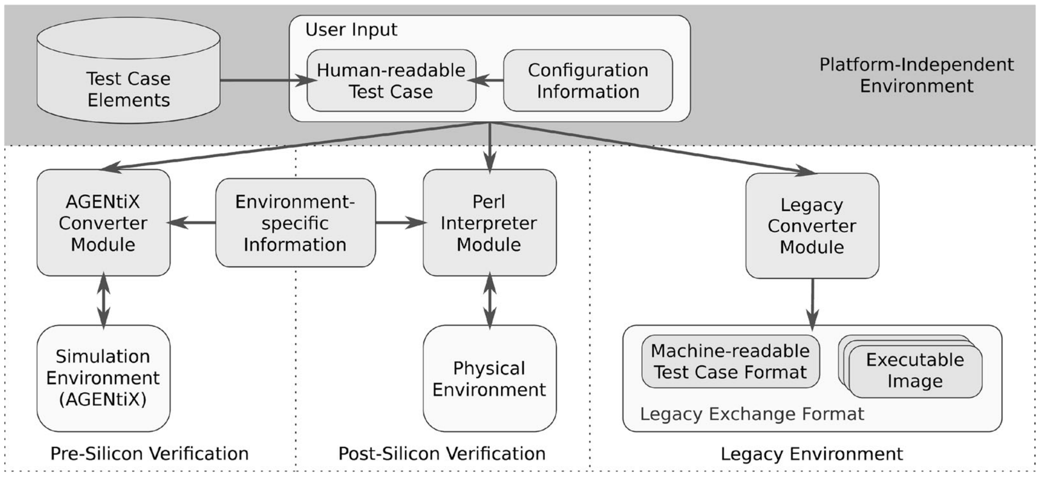

Fig. 1. Basic concept of the proposed approach

case description needs to be changed. These changes can then be distributed automatically by regenerating the environment-specific test cases.

The basic concept of the proposed approach is shown in Fig. 1. Common environments as well as common languages are used to describe the test cases and to share them between single verification activities. The most important parts of the proposed approach are the *user input files*, here especially the *human-readable test case*. This common format allows defining test cases once that are further usable in pre- and post-silicon activities. Additionally, legacy environments can be supported by exporting test cases appropriately.

The common test description has to be written in a general, platform-independent form to be universally usable in all environments. However, since it is necessary to define a mapping between the signals described in the test cases and their manifestation in the respective execution environment, there is also *configuration information*, which might be partially environment-specific. Before and while test execution this *configuration information* is used together with *environment-specific information* (e.g., DUV-specific register addresses, instrument definitions) to describe the test environment.

To actually design a test case the engineer needs to create instances of the respective DUV and (abstract) laboratory instruments which are then connected appropriately. In physical environments this needs to be done manually by the laboratory operators according to a connection scheme stored in additional mapping files that link the physical device addresses with their virtual representation in the test cases. In simulations these pin mappings are stored together with the test cases and the information is automatically extracted.

To spare test engineers from rewriting often used test sequences manually, such sequences are stored in *test case element* pools. This library also contains standard functions (e.g., calculations) and universally applicable information (e.g., descriptions of complex waveforms).

Since test cases often require complex sequences, the use of a language that supports constructs like loops or branches is required. Several options were examined and it was determined to use Perl to describe test cases because it is already in heavy use at Infineon and therefore well supported in the target working environment. The test engineer can reuse the provided test bench elements and helper classes, but nevertheless some knowledge of Perl is required.

The actual converter modules are specific to the environment. For this reason they are described separately in the following.

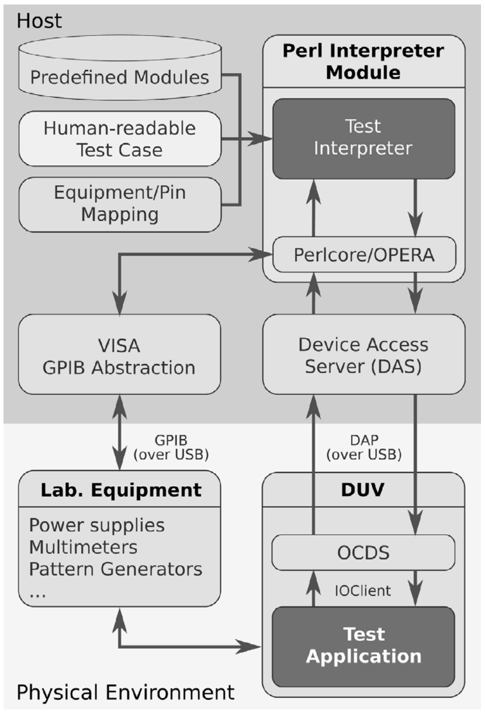

Fig. 2. Information and control flow in post-silicon verification

Figure 2 depicts the information flow in the post-silicon environment, showing the already explained input files (top left). In the post-silicon environment the successful execution of test cases requires communication with the DUV and laboratory instruments. Our test case interpreter manages the execution of the tests by sending commands to laboratory equipment via the common general purpose interface bus (GPIB) (lower left) while the microcontrollers are accessed via a proprietary two-pin debugging interface (Device Access Port (DAP), lower right). To that end, our approach reuses various efforts at Infineon to provide abstract interfaces to various classes of hardware (e.g., Device Access Server (DAS), Perlcore).

Interaction with the test firmware uses a feature of the microcontrollers On-Chip Debug Support (OCDS) named *IOClient* that allows communicating asynchronously with the firmware running on the microcontroller without disturbing its progress by a simple mailbox-like mechanism. In one of *IOClient*'s modes (i.e., Communication mode) any read or write from the host results merely in a bit flip in a flag register that can be polled by the firmware or, if requested by the application, an interrupt is triggered. The firmware is then free to decide when to fulfill the request. Additionally, the *IOClient* mode of the OCDS provides a deterministic way to reset the communication between the host and the firmware. This is used for the initial synchronization of the DUV with the host at the beginning of a test run.

The current firmware is tailored for each test case and synchronizes with the host software by using *IOClient* transfers. Any measurements or other results are sent back to the host via *IOClient* as well.

For simulations the test cases need to be converted to SystemC files to be fed into the *AGENtiX* environment. To that end, a global configuration variable is set at startup depending on the target environment to influence the runtime behavior of the Perl scripts. If the variable indicates a simulation environment then the low-level functions no longer control laboratory instruments or the DUV but instead emit SystemC code that interacts with the various BFMs representing the DUV and test bench elements. The resulting code can then be simply compiled and simulated by *AGENtiX*.

Additionally to the simulation and post-silicon environments our approach allows to support legacy test flows as well by converting the test cases to previously used formats. This works similarly to the code generation for AGENtiX as explained above.

As revealed by an initial prototype, an important role in our concept plays the synchronization of the test software and the DUV. If an accurate synchronization is not minded in the test case creation, it can significantly slow down the test execution or even end in wrong results. Therefore, convenience functions are provided that create barriers that synchronize the execution of the test sequence.

Another issue related to synchronization is introduced by the parallel execution on multiple targets. In some environments there are no dedicated communication channels between the controlling host and each DUV, or they are too slow for real-time interactions. If the test flow depends on intermediate results received from the DUVs the test software must somehow decide when and how to proceed at such arbitration points. The preliminary idea to solve this is to distinguish between two basic types of platforms, referred to as primary and secondary platforms. Primary platforms allow executing the test case and communicating with the host in real-time, while secondary platforms do not and therefore require another solution. To that end, the test cases need to be written with these circumstances in mind. They must not contain any communication instructions and use sufficient wait times to accommodate for variations in execution times. Test results are stored using on-board resources and are transfered to the test software after the tests have finished.

5. Evaluation

For the evaluation of our approach we focused on verifying the feasibility of the concept itself. We have chosen some existing test cases to be re-implemented with our approach. Instead of separately handcrafting respective programs for the simulation and post-silicon environment the code is only written once.

One of these test cases characterizes some aspects of the ADC unit found in microcontrollers as explained below. It is based on the well-known and widespread ADC test concept known as "*histogram method*" (refer to [4] and [5]) and will be further referred to as *histogram test*. The flow diagram of the histogram test is depicted in Fig. 3.

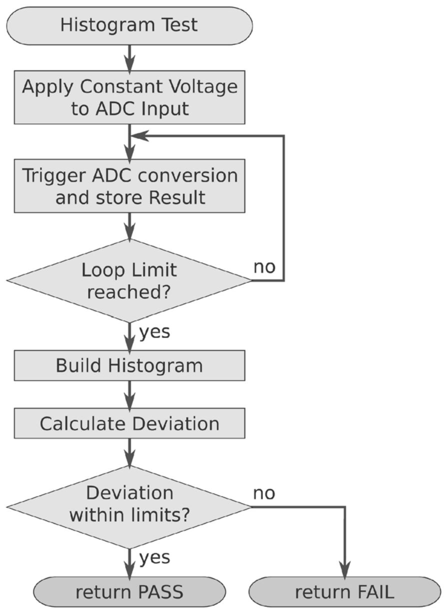

Fig. 3. Flow diagram of the histogram test

By the use of the histogram test the quality of ADC conversion results can be evaluated by measuring of the standard deviation of the conversion results. The test proceeds as follows:

1. Apply a constant voltage to desired input channel.
2. Trigger a few thousand conversions.
3. Build a histogram from conversion results.
4. Derive the standard deviation from the histogram.

To build the histogram the occurrence of every possible code word is counted. In an ideal ADC only the one code word corresponding to the applied voltage level would occur. Because of noise also adjacent code words appear in real ADCs. The counts of the code words are normally Gaussian-distributed where the variance, and respectively the standard deviation, depends on the quality of the actual ADC.

This represents a typical test case that is done in the laboratory after manufacturing to evaluate a mixed-signal design. The circumstances leading to unexpected results (e.g., a conspicuously high variance of measurements) can then (hopefully) be reproduced and investigated in simulation to find design flaws. Our approach tremendously simplifies the step needed to generate the code for these simulations.

6. Conclusion

Due to the rising complexity and shrinking product cycles, the development and proper verification of modern SoC solutions are very challenging. Although, standard flows are well-established for the design and verification processes, there is a lack of suitable concepts to link them. Additionally, verification becomes ever more challenging due to the number and complexity of features implemented, advancing process technologies, and lower supply voltages as well as effects caused by the interplay of digital and analog peripherals.

This article presents a concept to share test cases between different pre- and post-silicon verification activities. A common test description is introduced which can then be automatically converted to environment-specific test cases. This offers the advantage that test cases need only to be written once and can then be commonly used in different verification activities and environments. Due to the common basis, forward and backward annotation between verification activities is enabled.

In addition, this article presents some implementation issues in order to improve existing verification processes. The feasibility of the proposed approach was verified using predefined test cases. It shows that test cases can easily be defined using the capabilities of the proposed description language.

Acknowledgements

This work has been supported by Infineon Technologies Austria AG and by the publicly funded Josef Ressel Center for Verification of Embedded Computing Systems (VECS) managed by the Christian Doppler Gesellschaft (CDG).

References

1. Adir, A., Copty, S., Landa, S., Nahir, A., Shurek, G., Ziv, A., Meissner, C., Schumann, J. (2011): A unified methodology for pre-silicon verification and post-silicon validation. In Design, automation test in Europe exhibition, DATE (pp. 1–6). Available online at: http://dx.doi.org/10.1109/DATE.2011.5763252. doi:10.1109/DATE.2011.5763252.
2. Adir, A., Nahir, A., Shurek, G., Ziv, A., Meissner, C., Schumann, J. (2011): Leveraging pre-silicon verification resources for the post-silicon validation of the IBM POWER7 processor. In 48th ACM/EDAC/IEEE design automation conference, DAC (pp. 569–574). Available online at: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5981978.
3. Melani, M., D'Ascoli, F., Marino, C., Fanucci, L., Giambastiani, A., Rocchi, A., De Marinis, M., Monterastelli, A. (2006): An integrated flow from pre-silicon simulation to post-silicon verification. In Ph.D. research in microelectronics and electronics (pp. 205–208). Available online at: http://dx.doi.org/10.1109/RME.2006.1689932. doi:10.1109/RME.2006.1689932.
4. Azaïs, F., Bernard, S., Bertrand, Y., Renovell, M. (2001): A low-cost BIST architecture for linear histogram testing of ADCs. J. Electron. Test., 17(2), 139–147. Available online at: http://dx.doi.org/10.1023/A:1011173710479. doi:10.1023/A:1011173710479.
5. Azaïs, F., Bernard, S., Bertrand, Y., Renovell, M. (2001): Implementation of a linear histogram BIST for ADCs. In Design, automation and test in Europe proceedings, DATE (pp. 590–595). New York: IEEE Press. Available online at: http://dl.acm.org/citation.cfm?id=367072.367829.

Authors

Martin Horauer
received his master's degree in Electrical Engineering from Vienna University of Technology, Austria, in 1994 and his doctorate in 2004, respectively. He heads the bachelor program in Electronics Engineering and is Associate Head of the master program in Embedded Systems at the University of Applied Sciences Technikum Wien, Vienna, Austria. His research interests include test and verification of embedded systems.

Dominik Widhalm
received his master's degree in Embedded Systems from the University of Applied Sciences Technikum Wien, Vienna, Austria, in 2014. He is part of the Josef Ressel Centre for Verification of Embedded Computing Systems at the University of Applied Sciences Technikum Wien, where he is working on mixed-signal SoC verification in cooperation with Infineon Technologies Austria AG. His research interests include test and verification of mixed-signal SoC designs as well as embedded software design.

Stefan Tauner
was born in Vienna, Austria, in 1983. In 2003 he began to study Computer Engineering (in the bachelor program) at the Vienna University of Technology. He continued with the master program in 2009 which he finished with a thesis about unification of sensor data retrieval in mobile robots.
In fall 2013 he joined the Josef Ressel Center VECS at the University of Applied Sciences Technikum Wien working in cooperation with Infineon Technologies Austria AG on bridging the gap between post-silicon validation and pre-silicon verification. Since September 2014 he has been working additionally on automating fault injection in FPGA designs in collaboration with Siemens AG Österreich.
Besides his research he is also teaching bachelor and master students in various courses on operating systems, low-level programming and embedded systems.

Stefan Mirtl
received his bachelor degree in Information Management in 2013 from the Alpen Adria University in Klagenfurt, Austria, and is now doing his master degree in Business Economics. He has been employed at Infineon Technologies Austria AG since 2002. He started working as Maintenance Engineer, he then moved to the Infineon's Microcontroller Validation department in 2011.