Aspect enhanced functional coverage driven verification in the SystemC HDVL

Christoph Kuznik, Wolfgang Müller

Faculty of Electrical Engineering, Computer Science and Mathematics University of Paderborn/C-LAB D-33102 Paderborn, Germany

christoph.kuznik@c-lab.de
wolfgang@acm.org

***Abstract*—As embedded systems incorporate more and more amounts of IP and embedded software the functional and non-functional verification task is one of the key bottlenecks in the design process. Despite proprietary design and verification languages such as IEEE-1800 SystemVerilog and IEEE-1647 *e* offer CDV functionalities neither SystemC or the SCV add-on library contain these features. Moreover, as programming languages and verification paradigms of the hardware and software domain continue to converge the verification techniques and methodologies need to take account of that, e.g. by adaption of the aspect-oriented programming scheme. In this paper we describe an approach for enhancing the functional coverage collection in the SystemC ecosystem by means of aspects, allowing cross-cutting the concern of CDV verification in stand-alone aspects, increasing the overall verification productivity.**

I. Motivation

Embedded systems incorporate an ever increasing amount of IPs and embedded software, so the verification of the functional and non-functional properties is essential for dependability and robustness. Due to the complexity, the verification process itself is one of the main bottlenecks of every SoC design flow, preventing the industry from better numbers in first silicon success. Constrained-random verification (CDV) in conjunction with functional coverage analysis is a useful tool for verification closure and can help to cope with complexity. However, as systems-on-chip and hardware design in general become more and more software dependent, the programming languages and verification paradigms of both domains converge. The verification methodologies and techniques for embedded systems design need to take account of that, increasing the productivity and reducing time-to-market. While specialized hardware design and verification languages (HDLV) such as IEEE-1800 SystemVerilog [1] and IEEE-1647 *e* [2] incorporate functional coverage language features, these functionalities are neither available in the IEEE-1666 SystemC standard [3], the SCV add-on library [4] nor complete compared to the aforementioned in any publicly available SystemC library. Moreover, to our knowledge there is no particular activity of the SystemC working groups to add these functionalities to the next versions of SystemC or SCV. In previous conducted work [5], [6] we presented a functional coverage library for SystemC that implements parts of the SystemVerilog `covergroup` metric to enable coverage-driven verification of SystemC designs on multiple levels of abstraction. To improve the usability of the approach and to extend the feature set of our library we propose the utilization of the aspect-oriented programing scheme (AOP) for enhanced coverage-driven verification with SystemC. Although the programming technique originates from the software domain, it also gained interest in the embedded systems domain, as can be seen in the related work chapter.

In this article we present an aspect-enhanced functional coverage library to enable non-intrusive coverage-driven verification of SystemC designs on multiple levels of abstraction, implementing parts of the SystemVerilog `covergroup` metric. We believe that enabling the coverage-driven verification paradigm for verification closure in SystemC design flows using a functional coverage library will boost SystemC's role as HLDV language, in doing so, still being a fully free of charge and open source ecosystem. The remainder of the paper will be structured as follows. In section II we will briefly revise the history of AOP in embedded system verification and discuss related work. In section III we will highlight our proposed aspect enhanced library architecture as well as the aspect-enhanced verification process for more productive CDV and functional coverage analysis in SystemC before we conclude in section IV.

II. AOP in Embedded Systems Verification for functional coverage and related work

Various works have been conducted on the topic of aspects in embedded systems and combined hardware/software verification. In contrast to pure software languages which offer extensive run-time support, the possibilities of aspects in conjunction with C++ based hardware design and verification languages are limited, especially when it comes to dynamic weaving. An overview about AOP in C++ can be found in [7], a summary about the use of aspects in hardware in [8]. Despite AOP for C++ based languages may also be achieved by means of templates, doing so manually is like "*doing OOP with standard C*" and cumbersome. Therefore, the AspectC++ implementation [7] has become a widespread solution for the AOP pattern and is widely used by academic researchers.

A. Industry

The IEEE-1647 *e* and the IEEE-1800 SystemVerilog verification languages offer assertion based verification and functional coverage collection. Moreover, *e* was the first industry-used verification language to make use (in parts) of aspect-oriented programming. Besides, the recently released verification methodologies Open Verification Methodology (OVM) and its successor the Universal Verification Methodology (UVM) [9] take account of the AOP pattern too. For example, a test may re-configure and change the testbench and its elements. Anyhow, the coverage functionality of both SystemVerilog and *e* mainly target RTL designs and are not applied for abstract models and communication scheme as common for virtual prototype verification on higher abstraction levels.

B. Academic

Despite aspect-enabled verification techniques have spread into industry design and verification flows they are based on proprietary languages, tools and ecosystems. So while these specialized HDVL languages, e.g. *e*, incorporate functional coverage and assertion based verification, these features are not available in the IEEE-1666 SystemC [3] standard or the SCV add-on library [4]. Moreover, to our knowledge there is no particular activity of the SystemC working groups to add these functionalities to the next versions of SystemC or SCV.

Anyhow, there are various research activities being undertaken to enhance the SystemC ecosystem in terms of verification capabilities by means of aspect oriented programming. In [10] the authors estimate the amounts of data transmitted via user-defined SystemC communication channels using aspects. In [11] the authors use UML2 to model the dynamic behavior view of SystemC models and to generate aspects for expected state transitions. In contrast to functional coverage collection in terms of SystemVerilog `covergroup`, they focus on state transition coverage. In [12] the authors validate temporal LTL properties, in fact sequences of function calls, with help of aspects on TLM 2.0 complaint models. The authors improve their approach to a runtime verification framework in [13], again focusing on assertion functionality. In [14] the authors model system aspects such as metrics measure and cache policy. In [15] an aspect parsing architecture is proposed, called ASystemC, to enable static aspect-oriented features for synthesizable RTL SystemC code. Moreover, the authors focus on LTL verification of RTL models as well as estimation of circuit sizes. In [16] AOP is used as atoms for LTL property checking in ASystemC as well as ASpecC.

As conclusion we can state that there are various approaches to mimic LTL checker functionality as can be found in PSL or SVA within SystemC by means of aspects. In contrast to ABV functionality or specialized metrics we focus on the implementation of a general and common coverage metric, which is in fact the SystemVerilog `covergroup` metric, enabling verification engineers to use the same familiar feature set from SystemVerilog in SystemC designs, but besides RTL also on TLM level and also by means of non-intrusive AOP.

III. Enhancing functional coverage collection in SystemC with aspects

While extending the SystemC ecosystem with enhanced functional coverage we restrict the core library functionality to coverage collection and evaluation, so other parts of verification capabilities of SystemVerilog such as assertions, randomization and constraint-solving are not considered for implementation. It is assumed that the used SystemC testbench environment provides these features, e.g. with help of the SCV library or another library, to allow true coverage driven verification closure in SystemC. To enable great flexibility to the verification engineer and to capture the cross-cutting concern of verification in stand-alone aspects we propose a multi-paradigm functional coverage library for SystemC. In detail, the coverage metric maybe defined and sampled by

- adding coverage metric code *into* the test and testbench. Actually, this corresponds to the SystemVerilog `coverpoint` coding style.
- using non-intrusive cross-cutting concerns, modeled by means of coverage aspects. This allows a clean separation of multiple (independent) verification concerns and versatile metrics.

The library itself was designed as a singleton factory class, which is the main facility of the library, providing all necessary setup and management API calls for the creation and administration of every element of the implemented SystemVerilog coverage metric. The architecture of the library is depicted in figure 1. It may be used upon the standard OSCI SystemC kernel on any C++ framework as it was designed as an add-on library for C++/SystemC environments. This also eases the

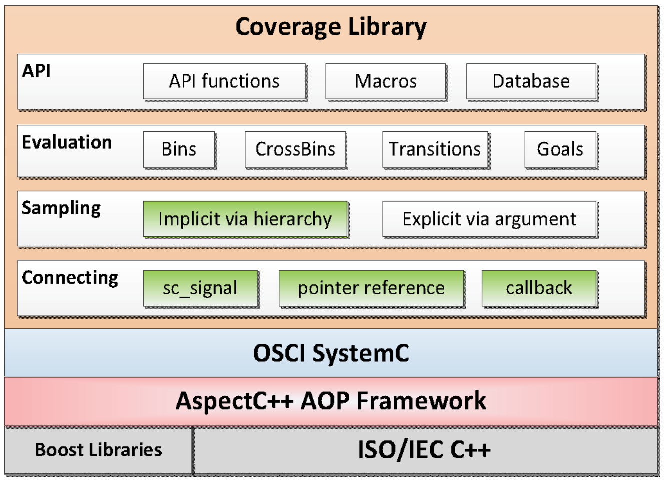

Fig. 1. Multi-paradigm functional coverage library for SystemC.

addition of functional coverage collection and evaluation to existing testbenches. Moreover, we make use of the Boost header-only libraries, e.g. function pointers. To allow multiple samplings of the coverage metric within one SystemC simulator delta cycle we do not rely on `SC_MODULE`s and clock sensitive `SC_METHOD`s within the implementation. The sampling process can be triggered in a method call fashion.

All sources are connected via pointer references or callback functions so their values may be read multiple times per delta cycle. Apart from simple value coverage boolean expression coverage can be implemented via callbacks functions.

A. Library Layers

Via the connection level the library allows connection of coverpoints to various value coverage source, such as `sc_signals`, variables or callback functions. If implicit sampling is used, each coverpoint has to be bound to a signal source. If a coverpoint is not bound to a value source upon simulation start the library will notify about that and halt the simulation. Once a coverpoint is connected to a source it may be sampled during simulation runs. This can be done via implicit sampling or explicit sampling, for example with an integer argument for an integer coverpoint. Moreover, both ways can be performed simultaneously along the entire hierarchy of covergroups, coverpoints and bins. For example, the statement `coverpoint->sample()` invokes an one-time value retrieval from the coverpoints connected source and invokes `sample(`*arg*`)` on all its `bins`. Within the evaluation level, all coverpoints and their bins are examined if the sampled value fits in one of the specified intervals. In this case, the hit counter of the specific bin will increase. Moreover, during construction of the coverage metric, the verification engineer may set coverage goals for each coverpoint such as minimal hits, targeted hit count and may associate weights. The API level provides API functions and macros to instantiate the covergroup structure, to connect sources to the coverpoints and to control and evaluate the coverage collection. Apart from that, the coverage results can be written to a simple database format. Figure 2 summarizes the implemented functions. For a detailed list of implemented features please refer to [5].

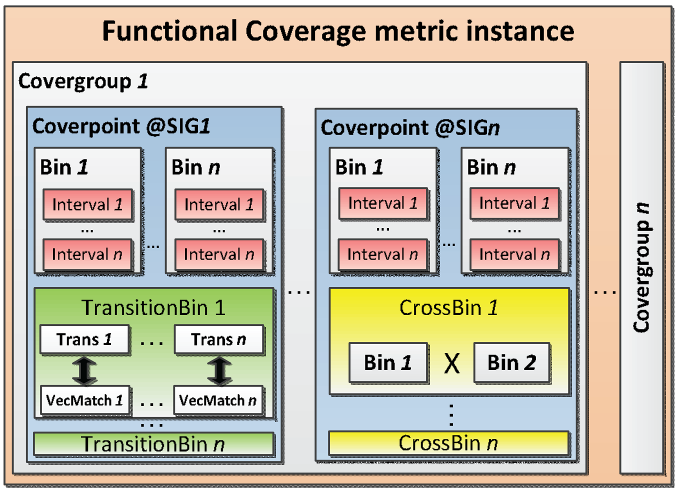

Fig. 2. Functional elements of the SystemC coverage library.

B. AOP enhanced coverage collection in SystemC

Using the AspectC++ layer we support source-to-source static weaving of aspects into the simulation for dynamic verification. The aspect itself may be simple (static), or dynamic by means of the provided `tjp` joint point API of AspectC++. Static joinpoints can be defined for functions, classes and namespaces via matching expressions. To capture certain events during the simulation the dynamic joinpoints are suitable, for example to cover events of functions calls as well as object construction and deconstruction. With help of advices the verification engineer has great flexibility to enhance his coverage metric to uncovered areas and corner case scenarios. For example, apart from SystemVerilog value and transition coverage the aspect-oriented non-intrusive mode allows to define various functional and non-functional coverage metrics. Examples are coverage collection and analysis of

- sequences of function calls to implement LTL checking behavior (assertion language functionality)
- dynamic user-defined TLM payloads analysis and coverage via the joint point API during simulation
- compliance with pre-defined policies
- power estimations based on a detailed power model for both communication channels as well as value-dependent algorithm annotations

and many more. Moreover, each of these metrics may be recorded per simulation run via the database API. This allows temporal merging of multi-metric coverage results from independent simulation runs of the same coverage metric while keeping the DUT code clean and unaffected. In our work we concentrate on enhancing the verification capabilities of the SystemC ecosystems in terms of *functional* (value, transition) coverage of DUT on multiple levels of abstraction. This means, we aim to utilize the entire AOP enhanced coverage verification scheme to both RTL and TLM designs with both static coverage definitions and woven aspects. In case of aspects the metrics definition must be known at compile time as we rely on source-to-source static C++ weaving at the moment. The authors of AspectC++ framework also provide a prototype implementation of a dynamic weaver for the C++ system [17], named `dac++`, which may will be considered after an in-depth analysis ensured maturity.

C. AOP enhanced dynamic verification performance

Static woven aspects have no influence to simulation time, as there is no difference for the compiler between manually inserted or source-to-source woven code within the DUV test and testbench sources. On the other hand, the compilation time itself increases due to the parsing undertaken by AspectC++. Therefore, the increase is metric-dependent [14], [16]. Anyhow, we believe that by means of distributed development of verification aspects the overall productivity benefits from AOP usage.

D. AOP enhanced coverage-driven verification flow

An aspect-considering CDV flow, which is an adapted version of original CDV for SystemVerilog flow presented in [18], is depicted in figure 3. As test set a combination of standard metrics and non-intrusive metrics are used to perform dynamic verification. Before running the actual simulation the selected aspects are woven into the compilation. Via collecting and analysing the coverage results, the constrained-random generation can be optimized or specific directed test can be conducted. Unfortunately, so far no existing verification methodology, such as OVM/UVM, is capable of reflecting the nature of aspect enhanced testbenches directly by means of their modeling elements.

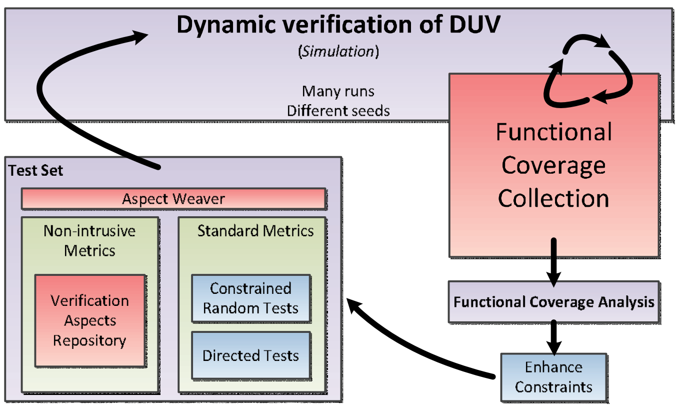

Fig. 3. Aspect-enhanced CDV flow with SystemC.

IV. Conclusion and Outlook

In this paper we have shown a novel approach for improving the verification capabilities of SystemC with functional coverage based on aspect-oriented programming. Despite this technique originates from the software domain, we believe that cross-cutting concerns have great benefits for analysis and verification of embedded systems and embedded software as methodologies and programming paradigms of both domains will continue to converge. Enabling multi-paradigm CDV on abstract models and virtual prototypes in conjunction with efficient functional coverage collection and analysis will enhance the role of SystemC as design and verification language, by doing so, still being a completely free of charge ecosystem. For further development and research activities we see a lot of interesting topics. For example, support for the upcoming Accellera Unified Coverage Interoperability Standard (UCIS) Standard [19] would enhance interoperability with other tools and proprietary flows. Moreover, the dynamic weaver version of AspectC++ could also be evaluated to allow *dynamic* weaving of verification aspects into running simulations.

In the broader scope this library will be combined with the results of other work packages of the BMBF founded SANITAS project, e.g. an enhanced OVM/UVM for SystemC, enabling early verification across the entire value-creation chain.

Acknowledgment

This work was partly funded by the DFG Collaborative Research Centre 614 and by the German Ministry of Education and Research (BMBF) through the BMBF project SANITAS (01M3088I). We greatly appreciate the cooperation with the project partners [20].

References

[1] IEEE, "Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language," *IEEE STD 1800-2009*, pp. C1 –1285, 2009.
[2] Open SystemC Initiative, "Standard for the Functional Verification Language e," *IEEE Std 1647-2011 (Revision of IEEE Std 1647-2008)*, pp. 1 –495, 26 2011.
[3] Open SystemC Initiative, *IEEE Standard SystemC Language Reference Manual*, Open SystemC Initiative Std., 2006.
[4] Open SystemC Initiative, *SystemC Verification Library v1.0p2*, 2006. [Online]. Available: http://www.systemc.org/downloads/standards/
[5] C. Kuznik and W. Müller, "Functional Coverage-driven Verification with SystemC on Multiple Level of Abstraction," *Proceedings of DVCON*, 2011.
[6] C. Kuznik and W. Müller, "Verification Closure of SystemC Designs with Functional Coverage," *North American SystemC User Group Meeting*, vol. 16th, 2011.
[7] R. Tartler, D. Lohmann, F. Scheler, and O. Spinczyk, "AspectC++: An integrated approach for static and dynamic adaptation of system software," *Knowledge-Based Systems*, vol. 23, no. 7, pp. 704 – 720, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705110000377
[8] M. Engel and O. Spinczyk, "Aspects in hardware: what do they look like?" in *Proceedings of the 2008 AOSD workshop on Aspects, components, and patterns for infrastructure software*, ser. ACP4IS '08. New York, NY, USA: ACM, 2008, pp. 5:1–5:6. [Online]. Available: http://doi.acm.org/10.1145/1404891.1404896
[9] Accellera Organization, "Universal Verification Methodology (UVM) standard," 2011. [Online]. Available: http://uvmworld.org/
[10] G. Jakacki, "Aspect-oriented techniques for extraction of communication models from SystemC designs," in *ASIC, 2003. Proceedings. 5th International Conference on*, vol. 1, oct. 2003, pp. 262 – 265 Vol.1.
[11] Y. Chen, W. Qiu, B. Zhou, and C. Peng, "An automatic test coverage analysis for SystemC description using aspect-oriented programming," in *Computer Supported Cooperative Work in Design, 2004. Proceedings. The 8th International Conference on*, vol. 2, may 2004, pp. 632 – 636 Vol.2.
[12] M. Kallel, Y. Lahbib, R. Tourki, and A. Baganne, "Aspect-based ABV for SystemC transaction level models," in *Microelectronics (ICM), 2009 International Conference on*, dec. 2009, pp. 304 –307.
[13] Kallel, M. and Lahbib, Y. and Tourki, R. and Baganne, A., "Verification of SystemC transaction level models using an aspect-oriented and generic approach," in *Design and Technology of Integrated Systems in Nanoscale Era (DTIS), 2010 5th International Conference on*, march 2010, pp. 1 –6.
[14] D. Déharbe and S. Medeiros, "Aspect-oriented design in SystemC: implementation and applications," in *Proceedings of the 19th annual symposium on Integrated circuits and systems design*, ser. SBCCI '06. New York, NY, USA: ACM, 2006, pp. 119–124. [Online]. Available: http://doi.acm.org/10.1145/1150343.1150378
[15] P. Borba and S. Chiba, Eds., *Companion Volume of the 10th International Conference on Aspect-Oriented Software Development, AOSD 2011, Porto de Galinhas, Brazil, March 21-25, 2011.* ACM, 2011.
[16] O. A. L. Lemos and P. C. Masiero, "A pointcut-based coverage analysis approach for aspect-oriented programs," *Inf. Sci.*, vol. 181, no. 13, pp. 2721–2746, 2011.
[17] R. Tartler, D. Lohmann, W. Schröder-Preikschat, and O. Spinczyk, "Dynamic AspectC++: Generic Advice at Any Time," in *Proceeding of the 2009 conference on New Trends in Software Methodologies, Tools and Techniques.* Amsterdam, The Netherlands: IOS Press, 2009, pp. 165–186. [Online]. Available: http://dl.acm.org/citation.cfm?id=1659308.1659322
[18] C. Spear, *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features.* Springer, 2008. [Online]. Available: http://books.google.com/books?id=tJYK-_-OV6QC
[19] Accellera Organization Inc. (2011) Unified Coverage Interoperability Standard (UCIS).
[20] Collaborative verification along the entire value-added chain; "SANITAS" research project launched under management of Infineon. [Online]. Available: http://www.infineon.com/cms/en/corporate/press/news/releases/2009/INFXX200912-018.html