Cross-Level Verification of Hardware Peripherals

Sallar Ahmadi-Pour[1], Muhammad Hassan[1,2], Rolf Drechsler[1,2] *

[1]Institute of Computer Science, University of Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, Germany
{sallar, hassan, drechsler} @uni-bremen.de

Abstract

In this extended abstract we present a Virtual Prototype (VP) driven verification methodology for Hardware (HW) peripherals. Our verification methodology is twofold: A Coverage-Guided Fuzzing (CGF) based approach enables comprehensive verification at the unit-level, while an application-driven co-simulation approach enables verification at the system level. As a case-study, we utilize a RISC-V Platform Level Interrupt Controller (PLIC) as HW peripheral and use an abstract Transaction Level Modeling (TLM) PLIC implementation from the open source RISC-V VP as the reference model. In our experiments, we find three behavioral mismatches as well as non-functional timing behavior mismatches. As the different approaches uncover different types of mismatches, we conclude a synergy between the methods to aid in verification efforts.

Introduction

With the trend of modern computing systems leaning stronger towards innovative technologies like hardware acceleration, open instruction sets and new methodologies in the development and verification of chips, further growth in complexity of *System-on-Chip (SoC)* is natural [1]. Together with this trend, the need for early and more integrated verification methods arise, as detecting errors in later stages will become more costly to fix. To deal with the rising complexity, modern design flows for embedded systems leverage *Virtual Prototype (VP)* [2]. A VP is an abstract executable model of the entire *Hardware (HW)* platform commonly utilizing the *Transaction Level Modeling (TLM)* formalism [3], with the goal of being available as early as possible in the development process (i.e., executable specification). VPs are leveraged for early *Software (SW)* development and verification and also serve as functional reference model for the subsequent HW development stage at the *Register-Tranfer Level (RTL)*. As such VPs enable to streamline and integrate the HW and SW development and verification flows [2]. While a strong emphasis has been put on methods to verify the processor, as it is at the heart of an SoC, verification of peripherals has been comparatively neglected. However, peripherals are essential components in modern SoC by providing core functionality in the interaction with sensors, actuators, buses and other controllers. In this extended abstract, we propose a VP-driven verification methodology with focus on HW peripherals, as shown in Fig. 1. Particularly, we combine two approaches that complement each other and use the VP as readily available reference model: We use Ⓐ a fuzzing-based approach that enables comprehensive unit-testing of the HW peripheral with a TLM reference and Ⓑ a simulation-based approach that enables application-driven testing of the HW peripheral on a system-level. As a case-study, we designed a RISC-V *Platform Level Interrupt Controller (PLIC)* [4] as a HW peripheral at the RTL and use the open source RISC-V VP [5] [6], which provides a TLM PLIC, as reference model. Our experiments demonstrate the effectiveness of our verification methodology in supporting the design flow for RTL peripherals by finding mismatches with the readily available TLM reference model. While this paper provides an overview of our work, a more extensive presentation of our work and discussion of results can be found in [7].

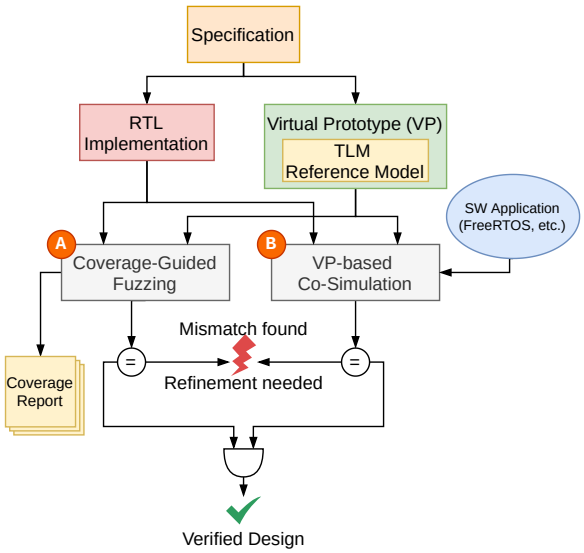

Figure 1: *Overview of the verification methodology for hardware peripherals.*

Verification Methodology

Our cross-level verification methodology utilizes two verification techniques, namely *Coverage-Guided Fuzzing (CGF)* and an application driven co-simulation. For the CGF, the readily available TLM reference and the RTL *Design Under Verification (DUV)* are simulated within a fuzzing testbench, which

*This work was supported in part by the German Federal Ministry of Education and Research (BMBF) under grant no. 16ME0127 (Scale4Edge) and grant no. 01IW22002 (ECXL).

collects coverage information in a feedback loop. As the CGF generates TLM based transactions, the DUV is provided with a TLM-RTL transactor, in order to translate between TLM transaction payloads and RTL signals over clock cycles. If a difference in the behavior is identified, the CGF stops and the input pattern is available for further debugging of the mismatch. Additionally, the coverage report can be utilized to assess the quality of the verification. For the application driven co-simulation, the RTL DUV is integrated, together with the transactor, as a drop-in replacement in a full system VP. This allows a co-simulation with the TLM reference as well as the DUV. Through the full VP simulation, embedded software applications and operating systems can be executed and the integrated interaction of the DUV can be assessed.

Evaluation

In a case-study, using the RISC-V PLIC, we evaluate our cross-level methodology. The RISC-V PLIC is a suitable peripheral as it provides a combination of handling bus transactions, timing specific behavior and handling numerous I/Os and registers. We utilized the open source RISC-V VP [6], as full system configurations with a TLM based PLIC are available. An in-house developed RTL PLIC, generated from a SpinalHDL description, is compiled to SystemC RTL with Verilator and provided with a SystemC/C++ transactor, for handling TLM transaction.

In the CGF testbench we employ LLVM libFuzzer as a fuzzer. The fuzzer generated inputs are mapped to the interrupt inputs, with a configuration for the priority, as well as the configuration of the threshold of the PLIC. Our CGF approach identified three mismatches in the functionality, regarding the usage of the threshold value. Further inspection showed, that the PLIC specification allowed for ambiguity on the interpretation of this matter. Lastly, the obtained coverage for the TLM and the RTL DUV are shown in Tab. 1. The table shows the line, function and branch coverage in absolute and relative numbers, respectively.

Our application driven co-simulation uses a FreeRTOS based software application utilizing interrupts from various sources, thus integrating the use of the PLIC. Through analysis of the execution trace of FreeRTOS with the help of the Tracalyzer tool, we can identify mismatches in the behavior between the reference VP and the VP containing the RTL DUV. The comparison of the traces showed timing mismatches of the RTL DUV with the TLM PLIC, on increasing clock period in the RTL domain. This drift of the difference in internal simulation time increases non-linearly and was measurable in the order of microseconds. Lastly, we identified that the clock period applied to the RTL DUV impacts the overall simulation time of the VP, as shown in Fig. 2. With small clock periods, the simulation kernel requires more context switches, thus progressing the simulation slower.

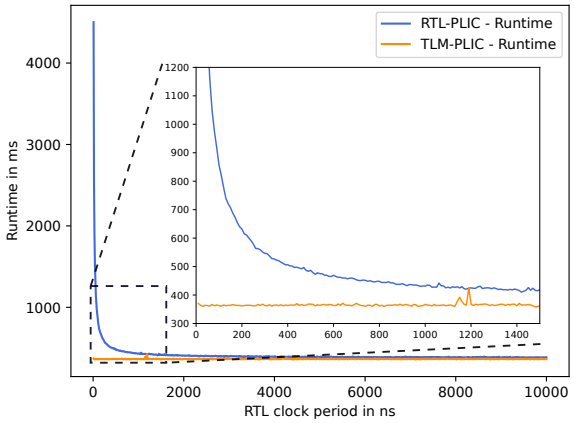

Figure 2: *Impact of clock period of the RTL component on the overall host execution time.*

Table 1: *Obtained coverage for the TLM/RTL peripheral*

Coverage Metric	TLM PLIC			RTL PLIC		
	Hit	Available	Coverage	Hit	Available	Coverage
Line coverage	119	121	98.3%	3212	3721	86.3%
Function coverage	13	13	100%	20	24	83.3%
Branch coverage	72	118	61.0%	1056	1432	73.7%

Depending on the peripheral this information can be utilized to choose between fast and accurate VP co-simulations.

Conclusion

In this extended abstract we highlighted the essential aspects of our cross-level verification methodology for hardware peripherals. We showed how VPs can be used to aid in the early verification on a unit level as well as on the system level, through our CGF and application driven co-simulation, respectively. Our results show how VPs allow synergies between unit level and system level to come into place and pave the road towards early and integrated verification methods. For future work, we plan to investigate more peripherals, different fuzzers and the inclusion of other verification techniques (e.g., symbolic execution).

References

[1] John L. Hennessy and David A. Patterson. "A new golden age for computer architecture". In: *Commun. ACM* 62.2 (Jan. 2019), pp. 48–60. ISSN: 0001-0782. DOI: 10.1145/3282307. URL: https://doi.org/10.1145/3282307.

[2] Tom De Schutter. *Better Software. Faster!: Best Practices in Virtual Prototyping*. Synopsys Press, Mar. 2014.

[3] *IEEE Standard SystemC Language Reference Manual*. IEEE Std. 1666. 2011.

[4] RISC-V International. *RISC-V Platform-Level Interrupt Controller Specification*. https://github.com/riscv/riscv-plic-spec/. 2022.

[5] *RISC-V Virtual Prototype*. https://github.com/agra-uni-bremen/riscv-vp. 2022.

[6] Vladimir Herdt et al. "Extensible and Configurable RISC-V based Virtual Prototype". In: *FDL*. 2018.

[7] Sallar Ahmadi-Pour et al. "Synergistic Verification of Hardware Peripherals through Virtual Prototype Aided Cross-Level Methodology Leveraging Coverage-Guided Fuzzing and Co-Simulation". In: *Chips* 2.3 (Sept. 2023), pp. 195–208. ISSN: 2674-0729. DOI: 10.3390/chips2030012.