Virtual Prototyping for Space Applications

How to get custom SoCs done fast and correct

Dr. Pascal Pieper
Pascal.Pieper@dlr.de
German Aerospace Center (DLR)
Institut für Raumfahrtsysteme

March 28, 2025

Modern satellite systems can rely on an increasing computational power with a continuously decreasing cost for that hardware. The RISC-V instruction set architecture adds to the number of possibilities with a flexible combination of standardized and custom instructions. However, with this increase in possibilities and the following shift into in-orbit processing comes the rise of software and hardware complexity. To reduce product development time and cost, satellite manufacturers usually employ single-chip combinations of an existing hard-IP processor with FPGA fabric. This enables focusing on the specific task of the system by using an existing toolchain and infrastructure, and only adding task-specific IP-cores separately.

While this approach is favorable, many companies still employ the hardware-then-software design process, which misses out on quality and development speed in comparison with more modern approaches. Opposed to this, employing *Virtual Prototypes* creates the possibility to design, evaluate, and verify an executable prototype of the system in an early design stage. This is done by modeling the future hardware on a behavioral or structural level, with a level of detail that grows together with the project.This enables both the iterative design evaluation and the *parallel development* of the software *and* (actual) hardware early in the product conception phase, where design-changing decisions still are possible. When comparing to VHDL simulators, Virtual Prototypes have the key advantages of 1. being available before every IP is finished, and 2. being faster in execution time (with a flexible trade-off between timing precision and speed [1]). Additionally, after development of the lower level hardware stages (e.g. on register transfer level, gate level, or physical hardware), Virtual Prototypes can be used as golden reference models together with test and verification methods for comparison between the system level behavior and the actual hardware (see fig. 1).

The main goal of this abstract is to encourage companies and individuals alike to use Virtual Prototypes (VPs) and invest in the comparatively low effort to create a correctly configured VP to benefit from new possibilities. Most notably, improvements in software quality [3–5], hardware quality [6–8], development speedup / design space exploration [1, 4, 7, 9–12], and accessibility [13–15] are featured in this work.

The traditional development process, especially for highly critical space applications, is comparatively rigid and document-driven. While it is important to have a clear architecture and well-defined work packages, achieving the best design decisions can be hard for complex systems. Unlike simple or complicated systems, where the general idea is known and tested, complex systems are hard to predict and thus pose a danger to the success of a product. This is the area where VPs fill the gap: Design space exploration during the project planning phase enables an early estimate on cost, speed, power consumption and overall architecture [2]. Such architecture (c.f. fig. 2) decisions can be simple questions on whether to implement a certain functionality in software or hardware [12], or more substantially, whether an idea is even possible to implement given the requirements.

Besides this, the VP that was used for architecture evaluation can be used simultaneously to already start development of low-level software to gain feasibility knowledge and, of course, improve development speed.

Once a VP is used in a project, it can also be leveraged to improve software and

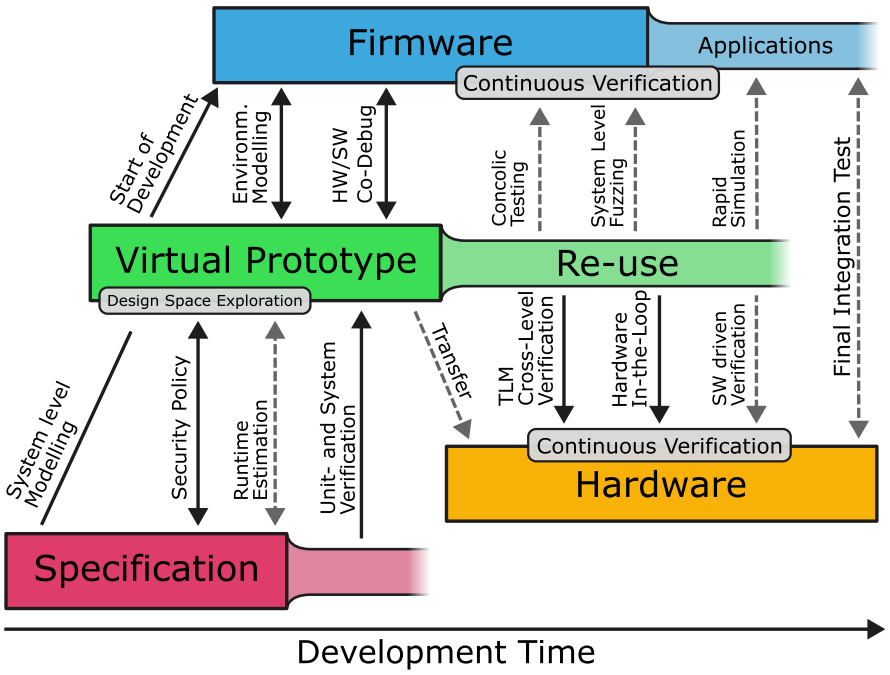

Figure 1: Virtual prototype design process after [2]

hardware quality which is especially important for space applications. It is usually not easy to test and explore software on embedded systems because of the memory and computation constraints, as well as interactions with on- or off-chip peripherals. Recent research on that topic however shows a multitude of possibilities with a VP. This includes symbolic execution [3] or coverage-guided fuzzing [16] of software with practical results, such as finding bugs in network drivers in the RIOT OS [5]. Also, possibilities of adding dynamic information flow tracking for verifying highly secure systems [4], or graphical introspection tools for modeled peripherals that help the design understanding and finding errors [9] arise.

Similar work has been done on the hardware-side as well. The developed peripherals in the VP can be tested thoroughly with different techniques such as constrained random verification [8], symbolic execution of the peripherals themselves [6], or by interacting with the physical world through Hardware-in-the-Loop testing [7] or co-simulation [17]. A high-quality VP then can be used as a golden reference model that can act as an *executable specification*. With that, more and advanced verification is also possible alongside the traditional testing. For example, modeling frameworks like SystemC TLM [18] offer possibilities to lift VHDL code to the VP which then can be used to analyze the devices with known tools for equivalence testing, or even synthesize high-level SystemC code down to VHDL [19]. Finally, the VPs improve the overall accessibility for small companies or even private individuals to start building their unique selling idea without having to re-invent everything around it. They can be used as a learning platform [20, 21] where the real hardware would be prohibitively expensive, or as a playground for potential customers to learn about a product before buying it.

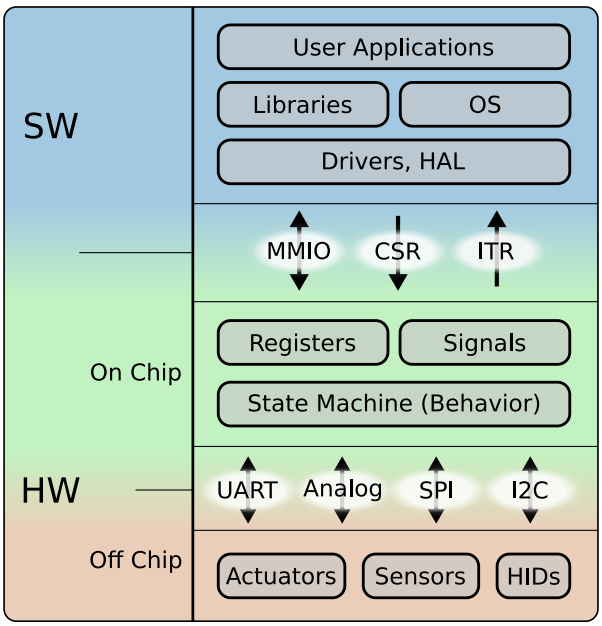

Figure 2: Embedded system as modeled with a VP [2]

In conclusion, Virtual Prototypes are very useful for designing complex systems and offer a multitude of different process improvements, resulting in a faster development speed and better quality. Especially when handling custom RISC-V instructions for highly specialized applications like SoCs with FPGA-fabric in restricted environments, the presented contributions show that the benefits of using a VP can significantly outweigh the initial work of developing the model. There are many different RISC-V centered tools and frameworks available [15, 20, 22–24], a lot published as open-source. Let's use them!

References

[1] V. Herdt, D. Große, S. Tempel, and R. Drechsler, "Adaptive simulation with virtual prototypes in an open-source risc-v evaluation platform," *Journal of Systems Architecture*, vol. 116, p. 102 135, 2021, ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2021.102135. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001016.

[2] P. Pieper and R. Drechser, *Formal and Practical Techniques for the Complex System Design Process using Virtual Prototypes, Better Early than Never*. Springer Cham, 2024, ISBN: 978-3-031-51691-7. DOI: 10.1007/978-3-031-51692-4.

[3] S. Tempel, V. Herdt, and R. Drechsler, "SymEx-VP: An open source virtual prototype for os-agnostic concolic testing of iot firmware," *Journal of Systems Architecture*, vol. 126, p. 102 456, 2022, ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2022.102456. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762122000480.

[4] P. Pieper, V. Herdt, D. Große, and R. Drechsler, "Dynamic Information Flow Tracking for Embedded Binaries using SystemC-based Virtual Prototypes," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 2020, pp. 1–6. DOI: 10.1109/DAC18072.2020.9218494.

[5] S. Tempel, V. Herdt, and R. Drechsler, "Automated detection of spatial memory safety violations for constrained devices," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2022, pp. 160–165. DOI: 10.1109/ASP-DAC52403.2022.9712570.

[6] P. Pieper, V. Herdt, and R. Drechsler, "Verifying SystemC TLM peripherals using modern C++ symbolic execution tools," in *2022 59th ACM/IEEE Design Automation Conference (DAC)*, 2022, pp. 1–6. DOI: 10.1145/3489517.3530604.

[7] P. Pieper, S. Ahmadi-Pour, and R. Drechsler, "Virtual-peripheral-in-the-loop: A hardware-in-the-loop strategy to bridge the VP/RTL design-gap," in *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '23, Hamburg, Germany: Association for Computing Machinery, 2023. DOI: 10.1145/-pending-.

[8] S. Ahmadi-Pour, V. Herdt, and R. Drechsler, "Constrained random verification for risc-v: Overview, evaluation and discussion," in *MBMV 2021; 24th Workshop*, 2021, pp. 1–8.

[9] P. Pieper, R. Wimmer, G. Angst, and R. Drechsler, "Minimally invasive HW/SW co-debug live visualization on architecture level," in *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '21, Virtual Event, USA: ACM, 2021, pp. 321–326, ISBN: 9781450383936. DOI: 10.1145/3453688.3461524.

[10] J. Zielasko, S. Tempel, V. Herdt, and R. Drechsler, "3D visualization of symbolic execution traces," 2022, pp. 1–8. DOI: 10.1109/FDL56239.2022.9925664.

[11] F. Böseler, J. Walter, and B. R. Perjikolaei, "A comparison of Virtual Platform Simulation Solutions for timing prediction of small RISC-V based SoCs," 2022, pp. 1–8. DOI: 10.1109/FDL56239.2022.9925667.

[12] F. Vahid, "What is hardware/software partitioning?" *SIGDA Newsl.*, vol. 39, no. 6, p. 1, 2009, ISSN: 0163-5743. DOI: 10.1145/1862900.1862901. [Online]. Available: https://doi.org/10.1145/1862900.1862901.

[13] P. Pieper, V. Herdt, and R. Drechsler, "Advanced environment modeling and interaction in an open source RISC-V virtual prototype," in *Proceedings of the Great Lakes Symposium on VLSI 2022*, ser. GLSVLSI '22, Irvine, CA, USA: ACM, 2022, pp. 193–197, ISBN: 9781450393225. DOI: 10.1145/3526241.3530374.

[14] P. Pieper, V. Herdt, and R. Drechsler, "Advanced embedded system modeling and simulation in an open source RISC-V virtual prototype," *JLPEA*, vol. 12, no. 4, 2022, ISSN: 2079-9268. DOI: 10.3390/jlpea12040052. [Online]. Available: https://www.mdpi.com/2079-9268/12/4/52.

[15] S. Ahmadi-Pour, V. Herdt, and R. Drechsler, "The microrv32 framework: An accessible and configurable open source risc-v cross-level platform for education and research," *Journal of Systems Architecture*, vol. 133, p. 102 757, 2022, ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2022.102757. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762122002429.

[16] V. Herdt, D. Große, J. Wloka, T. Güneysu, and R. Drechsler, "Verification of embedded binaries using coverage-guided fuzzing with systemc-based virtual prototypes," in *Proceedings of the 2020 on Great Lakes Symposium on VLSI*, ser. GLSVLSI '20, Virtual Event, China: Association for Computing Machinery, 2020, pp. 101–106, ISBN: 9781450379441. DOI: 10.1145/3386263.3406899. [Online]. Available: https://doi.org/10.1145/3386263.3406899.

[17] "Co-simulating your verilated model," Accessed: 2025-02. [Online]. Available: https://renode.readthedocs.io/en/latest/tutorials/co-simulating-custom-hdl.html.

[18] *OSCI TLM-2.0 Language Reference Manual*, OSCI, 2009.

[19] P. Coussy, A. Takach, M. McNamara, and M. Meredith, "An introduction to the systemc synthesis subset standard," 2010-10, pp. 183–184. DOI: 10.1145/1878961.1878993.

[20] P. Pieper. "Virtual breadboard GUI," Accessed: 2022-12-20. [Online]. Available: https://github.com/agra-uni-bremen/virtual-breadboard.

[21] M. Koenig and R. Rasch, "Digital teaching an embedded systems course by using simulators," in *2021 ACM/IEEE Workshop on Computer Architecture Education (WCAE)*, 2021, pp. 1–7. DOI: 10.1109/WCAE53984.2021.9707146.

[22] P. Pieper, V. Herdt, S. Tempel, K. A. Rudkowski, S. Ahmadi-Pour, and N. Bruns. "RISC-V virtual prototype," Accessed: 2022-12-20. [Online]. Available: https://github.com/agra-uni-bremen/riscv-vp.

[23] "Renode," Accessed: 2022-04. [Online]. Available: https://renode.io/.

[24] "Synopsis virtualizer," Accessed: 2022-04. [Online]. Available: https://www.synopsys.com/verification/virtual-prototyping/virtualizer.html.