Changing the RISC-V Verification Paradigm with Vyoma's Verification-as-a-Service Framework

Lavanya Jagadeeswaran[1,2*]

[1]Vyoma Systems Private Limited
[2] SHAKTI Group, RISE Lab, Department of Computer Science and Engineering, Indian Institute of Technology, Madras

Abstract

Design Verification has traditionally been a closed development, high-cost resource (in terms of time, manpower) usage endeavor. In order to keep up with the growing trend of the increasing complexity of designs and its open standard methodologies, Vyoma's Verification-as-a-Service technology targets improved verification productivity leveraging state-of-the-art verification frameworks (Python-based) and compute infrastructure (Cloud-based). This provides a practical shift-left methodology for next-generation verification needs without compromising on the design verification quality.

Introduction

In recent times, the RISC-V open standard ISA has been adopted in the industry and academia alike to build highly efficient commercial processors. The ISA, being modular and extensible, has seen customization based on target applications. To leverage the benefits of the open standard RISC-V ISA, SHAKTI processor program started in 2014 at RISE Group, IIT Madras developed different class of processors targeting embedded, controller and high performance applications. The Shakti design offerings also include interconnect fabrics, peripheral IPs and customizable System-on-Chips (SoC).

Verification of these varied design components can be categorized at three different levels, namely, **block**, **core** and **system** level verification. Block level verification targets different units of the processor or peripheral whereas core level verifies the processor implementation based on the RISC-V ISA. The system level verification targets the interconnections of various core and IPs along with re-using the block and core level verification components in order to make it a comprehensive top level verification methodology. The verification levels for an example Shakti C-Class [1] RISC-V processor based SoC is depicted in Figure 1 and they are described in the subsequent sections.

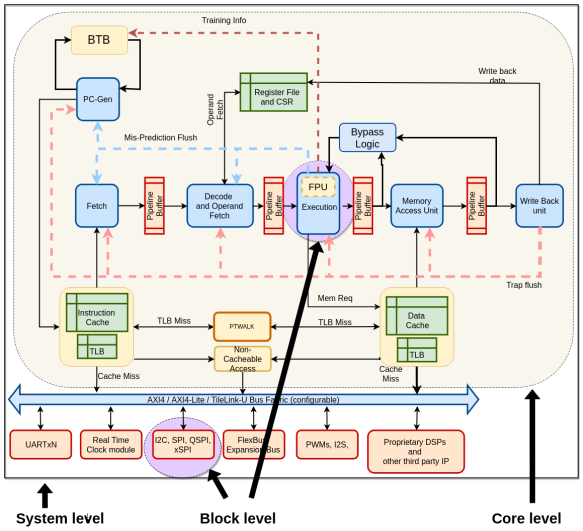

Figure 1: *Verification Levels for a Shakti C-Class based SoC*

Block Level Verification

Block level verification follows UVM methodology as shown in Figure 2. The test transactions are provided to the *driver* component which translates it to signals to be driven to the *design-under-test*. The *input monitor* observes the signal changes happening at the DUT input which is then provided to the *reference model* to obtain the expected output. Similarly, the *output monitor* observes output signal changes. Finally, both the expected and observed output are compared in the *scoreboard*. The test fails when the comparison fails during simulation. This UVM based verification framework is developed in Python. The interaction of the Python framework with the simulated DUT is provided through the CoCoTB libraries [2]. The simulation is done using Verilator.

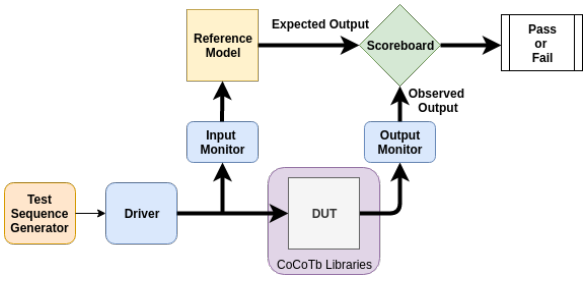

Figure 2: *UVM Block Level Verification*

*Corresponding author: lavanya@vyomasystems.com

Processor Verification

To verify these RISC-V implementations, several support tools have also been developed by the RISC-V community. These tools aid in verifying the processor in the form of providing test execution environments, configurable test benches or coverage metrics describing the verification completeness. Enumerating these open-source support tools, riscv-tests [3] provides self-checking directed assembly tests and execution environments, riscv-arch-tests [4] provide a framework to run signature based compliance tests to validate the implementation against the user and privileged specifications [5]. AAPG [6], riscv-torture [7], riscv-dv [8], microTESK [9] are used to perform random generation of assembly tests targeting various architectural scenarios. Spike [10], riscvOVPsim [11], VeeR-ISS [12] are the reference models used for simulation-based verification. In addition to these tools, there are various formal definitions of the ISA specification along with coverage capability that can greatly aid verification. At present, none of the above tools or environments can be termed as complete in itself to verify all the possible customizable RISC-V implementation. Vyoma's UpTickPro Verification framework provides a cloud-based Python infrastructure to rapidly provide verification closure and deliver high-quality RISC-V Designs.

Vyoma's UpTickPro

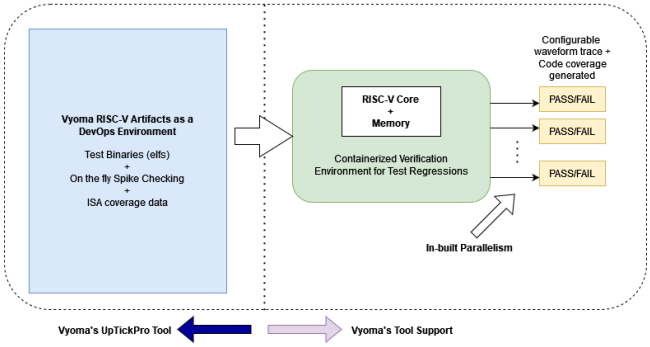

Figure 3: *Vyoma's UpTickPro*

RISC-V processor design verification will be carried out as shown above at two abstract levels, namely, block and core. The block level verification targets different design units of the processor using industry-standard UVM (Universal Verification Methodology) framework in Python using CoCoTb [2]. On the other hand, core level verification provides instruction-level comparison of the RISC-V design with Spike, the RISC-V ISS (Instruction Set Simulator). UpTickPro provides ready-to-verify RISC-V tests both manually developed and randomly generated. These tools are developed and maintained at Vyoma for rapid processor verification and their artifacts like the test binaries, processor state comparison on the fly, ISA level coverage data can be quickly provided for the verification flow. Using Vyoma's RISC-V Artifacts, verification happens in UpTickPro's environment to parallelly simulate the tests on the design and compare it with the expected ISS trace. Vyoma's UpTickPro introduces a new paradigm where we leverage the software infrastructure for hardware verification and thereby improving productivity but maintaining RISC-V design quality.

Acknowledgements

The author would like to thank the SHAKTI Group at IIT Madras for the collaboration and access to the industry-standard SHAKTI ecosystem to validate the use of UpTickPro for Verification. Vyoma Systems is a startup out of this SHAKTI Lab incubated at IITM Pravartak Technologies Foundation at IIT Madras, Chennai, India.

Author Bio

Lavanya has 12+ years of experience in the verification field at industry and academic processor teams. She has been part of IBM (IBM Z processors), ARM (ARM Cortex M processors) processor verification teams and led the RISC-V processor verification efforts at Rambus Chip Technologies and at SHAKTI IIT Madras. She has an MS at IIT Madras from the Department of Computer Science and Engineering in 2011.

References

[1] *SHAKTI C-Class.* `https://gitlab.com/shaktiproject/cores/c-class`.

[2] *CoCoTb.* `https://www.cocotb.org/`.

[3] *riscv-tests.* `https://github.com/riscv/riscv-tests`.

[4] *riscv-arch-tests.* `https://github.com/riscv-non-isa/riscv-arch-test`.

[5] *riscv-spec.* `https://riscof.readthedocs.io/en/latest/`.

[6] *AAPG.* `https://gitlab.com/shaktiproject/tools/aapg`.

[7] *riscv-torture.* `https://github.com/ucb-bar/riscv-torture`.

[8] *riscv-dv.* `https://github.com/chipsalliance/riscv-dv`.

[9] *microTESK.* `http://www.microtesk.org/`.

[10] *spike.* `https://github.com/riscv-software-src/riscv-isa-sim`.

[11] *riscvOVPsim.* `https://github.com/riscv/riscv-ovpsim`.

[12] *VeeR-ISS.* `https://github.com/chipsalliance/VeeR-ISS`.