CV32E40P updates: customizing an open-source RISC-V core at industrial-grade; experiences and challenges

Pascal Gouédo[1], Yoann Pruvost[1], Xavier Aubert[1], Olivier Montfort[1]
Pasquale Davide Schiavone[2], Mike Thompson[2]

[1]Dolphin Design, [2]OpenHW Group

Abstract

This paper will focus on the design and verification strategy of an open-source RISC-V CPU for edge-computing platforms at industrial-grade.

Introduction

In 2021 Dolphin Design made a test chip using a customized RI5CY CPU from ETH Zurich in one MCU sub-system and in a Multicore platform with 16 cores.

However, the Technology-Readyness-Level (TRL) of the core was not high enough to meet standard required by production requirements.

When the core moved to the OpenHW Group organization, it achieved a TRL-5 (enough to meet industrial-grade according to the official scale from NASA and the European Commission) by applying industrial verification strategies as *step&compare* against a reference model, Universal Verification Methodology (UVM), etc., on the **RV32IMC** instructions subset supported by the core.

This paper will present the second verification project applied to the CV32E40P core to verify the missing **RV32F, Zfinx** and custom **RV32Xpulp** ISA extensions to achieve a TRL-5 on all the ISA supported by the core. We will talk about the challenges of verifying custom extensions, collaborating in an open-source ecosystem with other stakeholders, the Dolphin experience, and final outcome.

CV32E40Pv2

OpenHW Group is a not-for-profit, global organization with the objective to deliver open-source cores at industrial-grade level.

Rather than verifying RI5CY core from the ground up, Dolphin Design (France) decided to become an OpenHW Group member to drive and to execute the verification of the custom Xpulp ISA extension developed at ETH Zurich and the official F and Zfinx RISC-V extensions.

Specification and Design

Originally, the RI5CY's custom Xpulp instructions were partly disseminated in RISC-V standard extensions as well as using Custom extensions. And Pulp_Zfinx was not aligned with the ratified Zfinx extension.

Therefore, in order to have CV32E40P fully compliant with the RISC-V specifications, all Xpulp instructions have been re-encoded to fit in Custom-0 to Custom-3. Additionally, the Pulp_Zfinx was fixed to meet the official Zfinx extension specification.

Verification

Coming to the verification side, the Core-V-Verif verification environment from OpenHW group was reused. Core-V-Verif is an open-source SystemVerilog verification environment which follows the popular UVM. Core-V-Verif was originally developed for v1.0.0 release of CV32E40P, and at the time of writing, it has been extended to support four additional cores from the CORE-V family of RISC-V cores.

A key feature of Core-V-Verif is its integration of a reference model that executes the same test-program as the device-under-test (DUT). The first generation of Core-V-Verif used an instruction set simulator (ISS) from Imperas Software as the reference model. Keeping the DUT and ISS synchronized was a challenge because these models have different concepts of time: the DUT is an RTL model which by definition is cycle-timed, and the ISS is instruction-timed.

In the general case, instructions executing on an RTL model will require a variable number of clock cycles to execute, while the same instruction stream on an ISS will always take a constant number of "instruction cycles" (typically one). A technique called *step&compare* was used to keep the two models synchronized by "stepping" the ISS only after the DUT retired an instruction. At this point the state of the two models was compared. Any mismatch between the PC, CSRs or GPRs was flagged as an error.

The above strategy was successful, but inefficient because the *step&compare* logic in the testbench must compensate for the cycle-time effects of events that are asynchronous to the instruction stream such as interrupts, debug resets plus

bus errors and random delays on instruction fetch and load/store memory buses.

For verification of v2.0.0 of the CV32E40P core, the ISS was replaced by a true reference model (RM) called ImperasDV. In this context, the distinction between an ISS and an RM is two-fold:

- An RM is aware of, and explicitly compensates for, the cycle-timed behavior of an RTL DUT;
- An RM is capable of predicting multiple legal state changes and comparing them to the actual state change of the DUT.

The reference model eliminated the need for complex *step&compare* logic by running in zero time and predicting multiple legal DUT states, based on instruction execution and external asynchronous events. In addition, the Imperas reference model has been extended to support the Xpulp instructions v2 specification.

Another innovation for v2.0.0 was the adoption of a standardized tracer interface to the DUT and RM, based on the open-source RISC-V Verification Interface (RVVI). The use of well documented, standardized interfaces greatly simplifies the integration of the DUT with the RM.

Stimulus generation for a processor core comes in multiple forms:

- Automatic generation of pseudo-random test-programs was facilitated by RISCV-DV, an open-source, SV/UVM RISC-V instruction stream generator developed by Google. An OpenHW extension to RISCV-DV, called COREV-DV was used to generate random streams of any instructions, v2 Xpulp ones included.
- Supplementing the random test-programs is a library of manually generated C and Assembly programs, including an obligatory “hello-world” as well as some benchmarking tests like Coremark and Embench IOT test suite.
- Randomization of bus cycle delays on the Instruction fetch and Load/Store memory interfaces are generated by a UVM Agent for the Open Bus Interface (OBI) standard.
- UVM Agents randomize interrupts and debug requests that are completely asynchronous to program execution. These agents can be disabled/enabled at run-time so that any test-program can be used to test the effects of interrupts/debug-request.

Formal Verification

To accelerate the verification of more than 300 Xpulp instructions, Formal Verification methodology and tools have been analyzed and evaluated.

At the end of this evaluation, Siemens EDA Onespin tools have been selected for

- their processor verification capabilities,
- their RISC-V ISA app due to its ability to verify standard extension as well as easy methodology to extend it by adding custom instructions described using pseudo-code language,
- for their RTL code coverage generation capability using RTL mutation. This Formal code coverage database can be exported in standard format that can be merged with RTL simulation code coverage.

The Xpulp instructions pseudo-code description using Sail language have been added to the RISC-V ISA app to successfully formally verify all the CV32E40P instructions, including the previously verified standard IMC together with the new F, Zfinx and Xpulp extensions.

This has been applied on 5 different core configurations (controlled via SystemVerilog parameters).

In addition, to keep the already-verified RV32IMC instructions sane for v1.0.0 users, a CI flow running a logical-equivalence-checking script between v1.0.0 and v2.0.0 (parametrized like v1.0.0) has been running for each contribution to the core.

Software toolchain

One more important development is about supporting the Xpulp instructions at industrial-grade in the GNU GCC SW toolchain. This has been done by Embecosm, which started from the GCC official upstream repository based on GCC 12 on which they added the Xpulp instruction support at different levels (assembly, builtin and compiler). There was a lot of investigation on -march option management and builtin definition because no custom instruction has been pushed in GCC upstream repositories as of today. So a builtin specification for CORE-V cores using RISC-V rules has been reviewed and released in OpenHW Group. After that release, an important part of the work on the toolchain was to create and to verify more than 250 builtins. Another hot topic was the automatic generation of Hardware Loops by the compiler.

Conclusion

The successful delivery of v2.0.0 release of CV32E40P is about to be possible thanks to open-source and OpenHW Group, where close collaboration of different people coming from different companies working in areas as different as software toolchain, reference model, hardware design and verification and EDA provider was a key enabler for the project.