Development and Verification of RISC-V Based DSP Subsystem IP: Case Study

Pascal Gouedo, Damien Le Bars, Olivier Montfort, Lee Moore, Aimee Sutton, Larry Lapides
Dolphin Design: pascal.gouedo@dolphin.fr, damien.lebars@dolphin.fr,
olivier.montfort@dolphin.fr
Imperas Software: moore@imperas.com, aimees@imperas.com, larryl@imperas.com

Abstract
The RISC-V ISA has been gaining momentum in the semiconductor community because of the design freedoms of the open specification. To date, the focus in the community has been on the development and verification of individual domain-specific processors and the software that runs on them. One of the use cases for those domain-specific processors is the instancing of multiple processors to create processing subsystems for AI/ML, audio processing and general-purpose digital signal processing. This subsystem level of processor integration poses new challenges for both hardware and software, not only with the individual RISC-V processors but also at the subsystem level.

New challenges faced include the verification of custom features in the processors, development of the communication fabric for the multiple processors and software development on the processor subsystem. There are also some existing challenges, specifically the verification of the base RISC-V processor.

To accelerate the development of software, a virtual platform (software simulation) flow is used. This starts with just a single processor model, the same OVP model used for DV, instantiated in a SystemC environment for basic bare metal software bring up.

This paper will present the single processor DV and virtual platform methodologies and results, and discuss the extensions to these methodologies for DV and software development for the processing subsystem.

Introduction

The open standard RISC-V Instruction Set Architecture (ISA) [1] provides users with the ability to use an ISA with accompanying ecosystem to develop domain-specific processors, thus replacing proprietary processors and accelerators. The RISC-V ISA can also be used to replace processors using traditional commercial ISAs, such as Arm, especially in use cases where legacy software is not a significant factor in choice of processor. In this situation, not having to support legacy software can allow the elimination of processor features and capabilities, thus enabling potential improvements in performance, power and area (PPA) for the processor implementation.

The development of a domain-specific processor or processing subsystem involves multiple tasks. Some of these are similar to the tasks if commercial processor IP were being used, such as pre-silicon software development. However, there are several tasks that are unique to this situation, including design verification (DV) of the individual processors. Also, while there is a processing subsystem DV task when commercial processor IP is used, this task is more complex with a new processor due to the novelty of the processor and the absence of a proven track record of past silicon success.

This need for comprehensive RISC-V processor DV is new to the semiconductor community. Previously with the standard single-sourced proprietary processor IP cores, users would license the IP confident that the processor IP vendor would deliver a pre-verified standard product. Users just needed to worry about integrating the processor IP into the rest of the SoC. This is the same for the RISC-V processor cores from the processor IP vendors: users only need to worry about the integration testing with the rest of the SoC. However, for open-source cores or new designs developed from the open standard specification, users have to do the complete RISC-V processor functional design verification work themselves.

The focus of this paper is the OpenHW Group Core-V CV32E40Pv2 processor [2]. This is a 32-bit RISC-V core, originally based on the RI5CY core developed by ETH Zurich for the PULP project [3]. The RI5CY was then productized by OpenHW as the CV32E40P, with some features of the RI5CY core not implemented. The CV32E40Pv2 now restores those remaining RI5CY features. To this base core are added some custom features to improve data processing and inter-processor communication in the processor subsystem fabric.

In this paper the new task of single processor DV is one focus, along with pre-silicon software development. In addition, the extension of the DV methodology to processor subsystem DV is briefly discussed.

RISC-V Processor DV

The OpenHW core-v-verif [4] processor verification methodology is used for the verification of the base processor plus custom features. This methodology is an asynchronous step-and-compare flow, with co-simulation of the RTL processor implementation and the instruction-accurate reference model. The asynchronous step-and-compare flow enables verification of asynchronous events such as interrupts, as well as features such as the RISC-V Debug mode. An instruction stream generator is used for constrained random test generation, with functional coverage in the DV environment as the quality metric. A diagram of this flow is shown in Figure 1.

There are 6 keys pieces to this DV flow:

1. RTL tracer module
2. RISC-V reference model
3. Verification IP (ImperasDV)

4. RISC-V Verification Interface (RVVI) between DUT and reference model subsystem
5. Test stimuli
6. Functional coverage

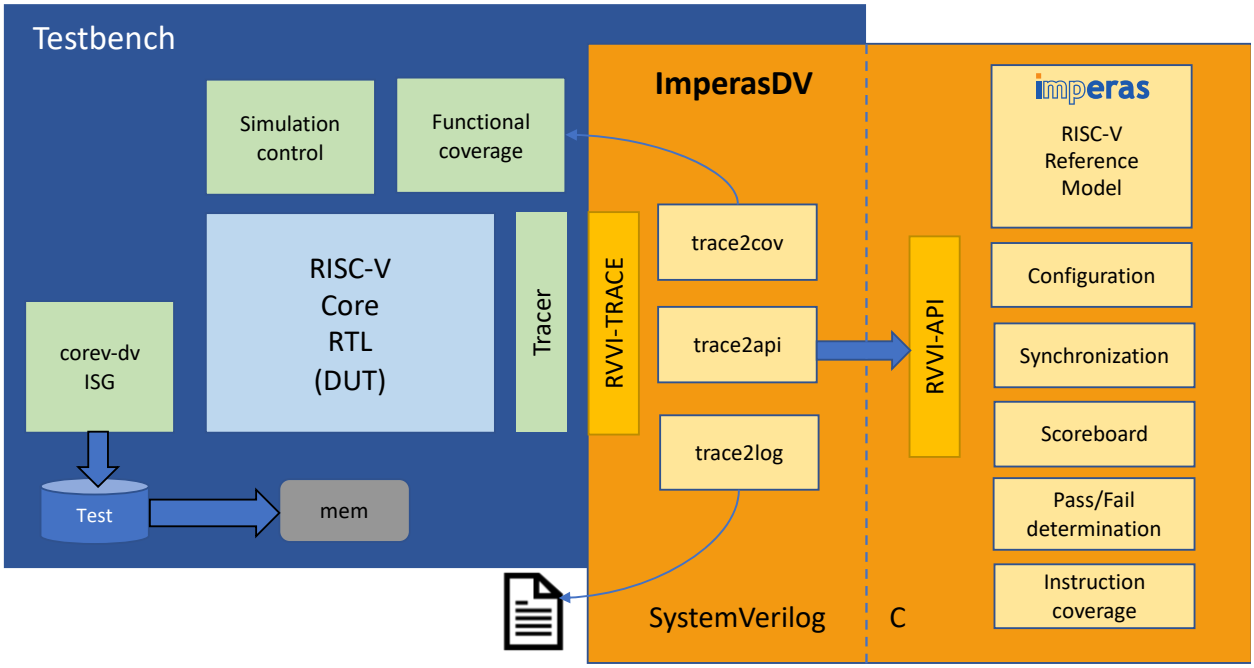

Fig. 1. DV environment for the CV32E40Pv2

RTL Tracer Module

This is the module, written in SystemVerilog, which provides instruction tracing, memory information and processor state data to the test environment. This is the limiting factor in processor DV, as any information not communicated by the tracer module cannot be verified. For ease of integration with the test environment, the tracer module should use the RVVI-TRACE standard.

RISC-V Reference Model

Open Virtual Platforms (OVP) processor modeling APIs (C language) [5] were used for building the reference model for the design verification (DV) flow. This model, with a SystemC wrapper, is also used for software development in a virtual platform (software simulation) environment. An Imperas simulator, which supports the OVP APIs, is used together with the OVP processor model for both the DV and software development flows. The OVP APIs are publicly available from the OVP website, and the OVP models are open source. The OVP processor model supports the full processor functionality, however, it does not model the pipeline and other micro-architectural details. Custom features, including custom instructions and registers, are added to the model using an external library. While the base OVP model is open source, the external extension library is under the control of the developers since the complete source tree is available under the Apache 2.0 open-source license. All OVP RISC-V models used the same base RISC-V reference model, with appropriate configuration parameters. The advantage of this

model architecture is that the base RISC-V OVP model is used by 10s of users every day, and so is rigorously tested both by Imperas and the users. As the model is upgraded to support the custom features in the processor, this upgraded model is used in both the design verification (DV) and software development flows.

Verification IP

The original OpenHW core-v-verif flow for the CV32E40P was developed on an ad hoc basis, with pieces added as they were needed. This was successful: the CV32E40P was verified and has been implemented in silicon [6]. However, this flow was not easily extendable to the next OpenHW cores. To add flexibility and extendibility, many of the functions in the test environment were combined into modular, parameterized verification IP.

This verification IP, commercialized as the ImperasDV product, includes

- Encapsulation of the reference model in the test environment
- Selection of the model variant, and any configuration needed of the model
- DUT reference state storage
- Synchronization technology: can run in synchronous or asynchronous modes
- Comparison technology
- Instruction coverage: simple C-based coverage metrics for just the instructions; this is not functional coverage

Being able to run in both synchronous or asynchronous modes enables DV of not only the RISC-V instructions by themselves, but also DV of features triggered by asynchronous events such as interrupts, Debug mode and multi-hart features.

The comparison technology enables comparisons at DUT/Reference Model processor events. While this usually means that comparisons are done on instruction retirement, this also enables comparisons for processors that have Out of Order (OoO) and/or multi-issue pipelines.

RVVI

Verification IP, reference models and the DUT are of course critical, but standards allow them to be connected efficiently. This was the central concept behind the development of the RISC-V Verification Interface, RVVI [7]. RVVI is an open standard available on GitHub that defines the key interfaces between the DUT and reference model with the necessary features to support the comparison, debug and coverage aspects for RISC-V processor verification shown in Fig 3. In addition, as this is not limited to any one core project or company, it can be seen as a universal solution that others can build on. RVVI already supports all the ratified RISC-V specifications and tracks the latest draft extensions as they stabilize.

RVVI has two primary pieces. The first is RVVI-TRACE, to interface the DUT to the test environment, as shown in Fig. 2. The second is RVVI-API, for interfacing to the reference model subsystem, as shown in Fig. 3.

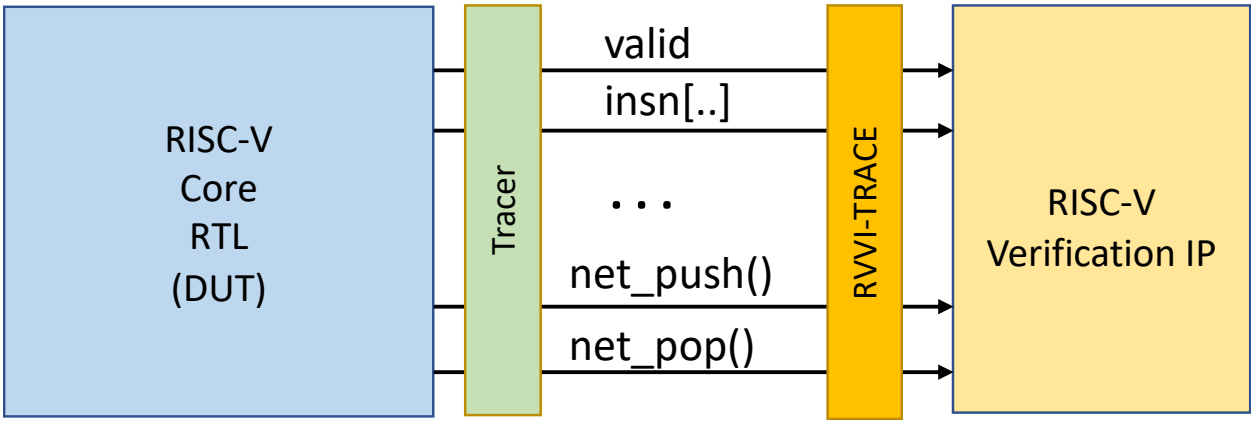

Fig. 2. RVVI-TRACE is the interface between the DUT, in particular the tracer, and the verification IP.

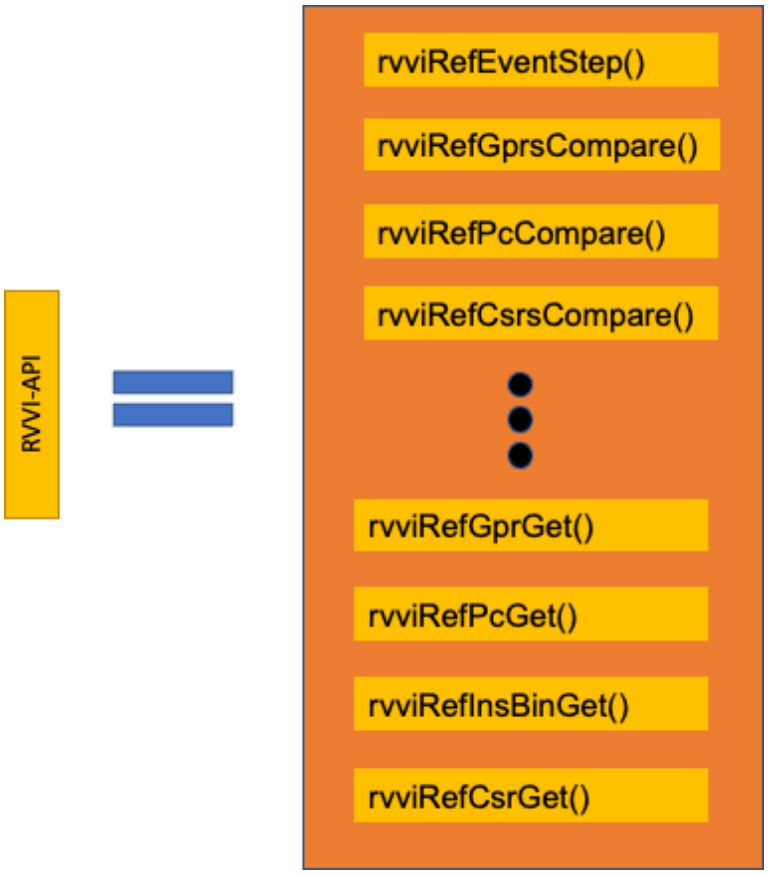

Fig. 3. RVVI-API defines standard functions that are implemented by RISC-V processor verification IP.

Test Stimuli

The DV flow developed can use either direct or random stimuli. The RISC-V International compliance tests as well as the OVP architecture validation tests were used as directed tests. For random stimuli, the riscv-dv instruction stream generator [8], an open-source tool, was modified to support the CV32E40Pv2 and this DV environment. This was shown as "corev-dv ISG" in Fig. 1.

Functional Coverage
SystemVerilog functional coverage modules were built to support the original CV32E40P verification effort. These coverage modules, which are UVM-compliant, are being extended for the additional functionality in the CV32E40Pv2.

Pre-Silicon Software Development
To accelerate the development of software, a virtual platform (software simulation) flow is used. This starts with just a single processor model, the same OVP model used for DV, instantiated in a SystemC environment for basic bare metal software bring up. By using the same processor model as both the DV reference model and for software development, users achieve improved consistency between the hardware and software teams. Using this methodology enables easier bring up of the software on the first silicon, reducing this task to just days.

The OVP model of the processor running with the Imperas simulator has one unique feature that makes integration with the SystemC environment much easier, and the performance more efficient: the Imperas simulator can be a slave to the SystemC simulator. Most of the virtual platform will be SystemC models of the behavioral and peripheral components of the SoC or processing subsystem. The SystemC simulator, by definition of the SystemC standard including simulation kernel, needs to be the master simulator, controlling the event scheduler. The Imperas simulator can let another simulator, in this case the SystemC simulator, drive its own event scheduler.

With this slave configuration, the OVP processor model looks to the SystemC virtual platform as another native SystemC model. This means that the OVP model executes in the SystemC process, and does not require a co-simulation configuration. This also enables a low overhead (less than 1%) for integrating the OVP model with the SystemC environment. SystemC DMI and other SystemC features are also used for integration to reduce overhead. OVP processor models typically execute at 300-500 million instructions per second, and since the bulk of the simulation time is spent in the processor model, the low overhead to achieve maximum simulation performance is critical.

Further Work: Processing Subsystem

Dolphin Design's Panther processor is a multicore processing subsystem embedding several instances of an enhanced CV32E40Pv2 core optimized for digital signal and AI/ML processing. Extending the work in this paper from a single processor to the processing subsystem is the next major effort. For DV of the processing subsystem, there are different verification goals, such as verifying the interactions between two processors, verifying the interactions with any other behavioral components and verifying the processor in a many-processor subsystem with all the communications between processors. This could involve different DV environments, perhaps

even with the reference model being used instead of some number of the RTL instances in the processing subsystem DUT.

For the virtual platform flow, the mechanics of instancing multiple processor models in the virtual platform of the subsystem are straightforward. The complexities arise when modeling a complete processor subsystem which includes dedicated hardware for processor synchronization, DMA controller and shared cache, and to provide the debug and analysis capabilities of the software running on the processor subsystem. An additional complexity comes from the tradeoff between simulation accuracy and simulation performance: wanting as much timing accuracy as possible while maintaining simulation performance of near real-time.

Conclusions

For customized, domain-specific RISC-V processors to become an established, low risk choice for SoCs the use of these processors must become easier, with key methodologies and best-known practices available to the semiconductor community. One of those key methodologies is processor design verification. An asynchronous step-and-compare methodology has been presented in this paper, together with the main components of that methodology. Early, pre-silicon software development is another key methodology, and while virtual platform-based development is established for certain commercial processors, work on tools and models is needed for adoption by RISC-V users. An important piece of this work is showing that the same model can and should be used both as the DV reference model and the processor model for the virtual platform.

Acknowledgments

The authors would like to acknowledge the help provided by their colleagues at their respective companies, plus the work by other member companies on OpenHW core-v-verif, and especially Mike Thompson of OpenHW Group for his work both hands-on and coordinating core-v-verif development.

References

1. RISC-V Specifications https://riscv.org/technical/specifications/
2. OpenHW CORE-V https://www.openhwgroup.org/#core-v-family
3. PULP Platform https://pulp-platform.org
4. OpenHW CORE-V-VERIF https://github.com/openhwgroup/core-v-verif
5. OVP Documentation https://www.ovpworld.org/documentation
6. M. Thompson et al, “Jump Start Your RISC-V Project with OpenHW”, DVCon Silicon Valley, 2021 https://www.imperas.com/articles/dvcon-2021-paper-jump-start-your-riscv-project-openhw
7. RVVI – RISC-V Verification Interface https://github.com/riscv-verification/RVVI
8. Google RISC-V ISG https://github.com/google/riscv-dv