TCCF: Tightly-Coupled Co-simulation Framework for RISC-V Based Systems

Xavier Ruppen, Roberto Rigamonti, Alberto Dassatti
HES-SO — REDS Institute, HEIG-VD — CH-1400 Yverdon-les-Bains, Switzerland
e-mail: *name.surname*@heig-vd.ch

***Abstract*—The development of heterogeneous systems where a CPU has to interact with a custom HDL design is particularly critical; these systems must, in fact, rely on the cooperation of sub-systems developed by (usually) separate teams and tested in isolation, typically by developing a mock system to represent the counterpart and running made-up test benches for some limited scenarios. This approach often postpones the discovery of issues to the system integration stage, where locating them is more complex and fixing them more expensive.**

In this paper we present TCCF, a QEMU/ModelSim-based framework that allows the complete co-simulation of heterogeneous systems. Having full visibility on the internals during HW/SW interactions enables to quickly spot incongruences and mistakes in the interface design, as well as to perform in-depth investigations on system's failures and behavior.

We have validated TCCF on a variety of publicly-available IP cores and platforms. The framework is easily extensible to cover custom protocols and operating systems, and is available under a GPL-3.0 license.

I. Introduction

The traditional workflow in heterogeneous system design requires multiple teams, often with non-overlapping competences, to agree on a set of paper specifications, and then to separately develop the different components. Testing is typically performed by creating a mock system to represent the missing parts and running made-up test benches and scenarios. This approach not only postpones the discovery of system integration bugs late in the development life cycle, where the costs are much higher [1], but also requires each team to develop accurate simulators, with the associated financial burden (and set of bugs) [2].

In this paper we present TCCF (*Tightly-Coupled Co-simulation Framework*), a QEMU/ModelSim-based framework that allows the quick co-simulation of heterogeneous systems. TCCF handles the interactions between ModelSim[1] — via its FLI interface [3] — and a modified version of RISC-V/QEMU[2], allowing unmodified host software to run in QEMU and interact with an HDL design being simulated in ModelSim. Having full visibility on the internals during **real** interactions enables developers to quickly spot incongruences and mistakes in the interface design due to ambiguities in the specifications, enhancing at the same time the visibility on the state of the system at the time of failure [2]. A comparable visibility could only be achieved by simulating the whole system as an HDL design; for instance, Arm validates its processors by making them boot GNU/Linux [4]. However, besides requiring the HDL sources for the complete system, this approach is prohibitively time-consuming — the time scale being in the order of days. Moreover, in the case of FPGA design, being able to simulate the HDL design in a realistic setting both avoids having to regenerate a new bitstream after each bugfix and does not impose the overhead due to hardware debugging tools, which could engender timing violations and thus unpredictable behavior [2], [5].

[1]https://www.mentor.com/products/fpga/verification-simulation/modelsim
[2]https://github.com/riscv/riscv-qemu

II. Related Work

The idea of co-simulating the hardware and the software components of an heterogeneous system is not new, dating to the early 90s [6]. However, so far it has been exploited in very restricted settings, either requiring the development of complex software abstractions or limiting the scope to specific languages/systems/platforms.

The RABBITS project [7] proposes a system-level simulation based on QEMU and SystemC. The approach and the preliminary results are extremely interesting, the choice of SystemC as a working language has a major impact upon the applicability of the methodologies. Indeed, while SystemC is still a promising technology, it is not yet supported in many standard workflows.

Another related project is SimXMD [5], which focuses on using GDB to drive the simulation of a processor. Despite being similar to what we propose, this approach is limited by the use of the GDB debugger and the close ties to the chosen processor, making portability a real issue.

The technique presented in [8] is also very close to what we propose, in that a set of interfaces is used to communicate between the HW and the SW sub-systems across a custom Remote Bus. Although more general than the approaches detailed above, it requires the development of an HDL wrapper and a custom translation program using their API for each supported bus, imposing a considerable engineering effort.

Despite being investigated for almost thirty years, the co-simulation problem is far from being solved, as testified by the very recent work presented in [2]: it proposes a full-system co-simulation framework for server systems with PCIe-connected FPGAs by linking a Virtual Machine's PCIe and NIC devices to the corresponding blocks in an HDL design. While very effective in providing reduced simulation times, it is limited

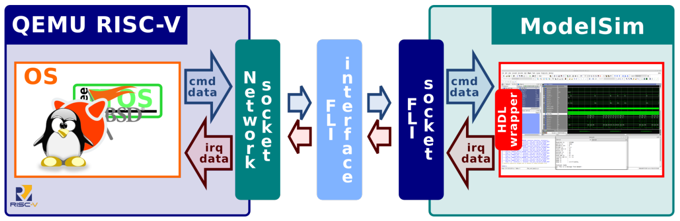

Fig. 1. Schematic representation of TCCF. Commands, data, and IRQ requests are exchanged between QEMU and the UUT via the FLI interface.

to PCIe devices and specifically aimed at server environments, while we target more generic platforms and configurations.

To the best of our knowledge, no approach available in literature is as easily re-targetable as TCCF, and therefore none can be adapted with very small changes to the RISC-V ecosystem. In particular, TCCF demands that neither the OS code nor the HDL design have to be altered, as just a set of easily-implemented wrappers between the two domains has to be created. As a consequence, **adding the support for a new design in TCCF can take as little as 15 minutes of work**. Furthermore, when a standard bus such as AXI [9] or Wishbone [10] is used, these wrappers are common to different designs and thus no adaptation is required.

III. Proposed Approach and Results

TCCF is structured in three main parts, as depicted by Fig. 1:

- QEMU/RISC-V: this component executes the OS, with the Unit Under Test (UUT) driver if available, and interacts with the HDL simulation through a network socket. In particular, it sends both commands and data to the UUT, and receives responses and interrupt requests. From the OS stance, the UUT appears as a (simulated) memory-mapped device. Operations directed to particular memory regions are intercepted and redirected to the socket connected to the FLI interface.
- FLI interface: it is a simple bridge that interfaces the sockets used by QEMU to the FLI socket exposed by ModelSim.
- ModelSim and its FLI socket: it interacts with a wrapper to the HDL design interfacing the UUT with the framework. The wrapper is very easy to write — it simply connects signals from the FLI socket to the bus used by the design, and thus could be easily automated. Although a Bus Functional Model (BFM) is required, BFMs for the major buses can be freely obtained from the UVVM Library[3] and imported as is.

Due to limitations in FLI, the hardware simulation cannot be controlled by QEMU — meaning that the UUT has to be started and stopped from ModelSim's interface or a script [3].

We have performed tests with several systems and HDL designs to validate our proposal and explore its limits. We have started from a simple VHDL UUT composed by four registers connected using an AXI-lite slave [9] and a RISC-V64U machine emulated in QEMU/RISC-V. In particular, we have taken the latest Linux kernel available for the SiFive Freedom U500 board — compiled with the SiFive GNU Embedded Toolchain — and the BusyBear Linux[4] root filesystem image targeting the VirtIO board in QEMU/RISC-V. After booting the virtual board, using the AXI-lite UVVM's BFM and a simple wrapper connecting AXI operations with the FLI interface, we have been able to read/write from the registers via the `devmem2` command.

We have then taken a freely-available Verilog UART 16550 core[5] that uses the Wishbone bus. UVVM does not have the BFM for the Wishbone bus, but it has the BFM for the Avalon bus [11] which differs from it in just few minor points. We were thus able to write a wrapper in few minutes, taking care to instantiate twice the UART port so as to be able to use them to make two separate QEMU instances communicate. In GNU/Linux we have then mapped the UART port to a chosen memory address where QEMU is able to intercept the read/write operations[6]. As a result, we were able to gain full-observability on a bi-directional serial communication between two systems via the simulated HDL design.

IV. Conclusion

In this paper we presented a framework for the quick co-simulation of heterogeneous systems. The clear advantage with respect to competing approaches available in literature is the versatility and the reduced effort required to adapt to custom platforms. Besides its applications in testing and debug, TCCF could be used in an educational setting to introduce students to an architecture and its interactions with a set of simulated peripherals[7].

References

[1] J.M. Stecklein et al., "Error Cost Escalation Through the Project Life Cycle," in *INCOSE Symposium*, 2004.
[2] S. Cho et al., "A Full-System VM-HDL Co-Simulation Framework for Servers with PCIe-Connected FPGAs," in *FPGA Conference*, 2018.
[3] Mentor Graphics, *ModelSim Command Reference Manual*. Mentor Graphics Corp., 2015.
[4] M.S. Hrishikesh, M. Rajagopalan, S. Sriram, R. Mantri, "System Validation at ARM," Arm Holdings, Tech. Rep., 2011.
[5] R. Willenberg, P. Chow, "Simulation-Based HW/SW Co-Debugging for Field-Programmable Systems-on-Chip," in *FPL Conference*, 2013.
[6] R.K. Gupta, C.N. Coelho, G. De Micheli, "Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components," in *DAC Conference*, 1992.
[7] TIMA Lab - Grenoble Institute of Technology. (2010) RABBITS : an environment for fast and accurate MPSoC simulation. [Online]. Available: http://tima.imag.fr/sls/research-projects/rabbits/
[8] P. Crosthwaite, J. Williams, P. Sutton, "A Unified Emulation/Simulation Environment for Reconfigurable System-on-Chip Development," in *FPT Conference*, 2011.
[9] Arm, *AMBA AXI and ACE Protocol Specification*. Arm Holdings, 2011.
[10] OpenCores, *Wishbone B4: WISHBONE System-On-Chip (SoC) Interconnection Architecture for Portable IP Cores*. OpenCores, 2010.
[11] Intel Corp., *Avalon Interface Specification*. Intel Corp., 2017.

[3] https://github.com/UVVM/UVVM_All

[4] https://github.com/michaeljclark/busybear-linux

[5] https://opencores.org/project,uart16550

[6] This can be easily done in the device tree for systems supporting it.

[7] Please refer to our previous work with a custom educational board, available at https://github.com/reds-heig/FSS