Comet: a RISC-V Core Synthesized from C++ Specifications

Simon Rokicki, Joseph Paturel, Olivier Sentieys

Comet: a RISC-V Core Synthesized from C++ Specifications

Simon Rokicki, Joseph Paturel, Olivier Sentieys

Univ Rennes, Inria, IRISA

***Abstract*—Designing the hardware of a processor core as well as its verification flow from a single high-level specification provides great advantages in terms of productivity and maintainability. In this work, we highlight the Comet RISC-V core specified from a unique C++ model. The same code is used to generate both the hardware target design through High-Level Synthesis as well as a fast and cycle-accurate bit-accurate simulator of the latter through software compilation. The object oriented nature of C++ greatly improves the readability and flexibility of the design description compared to classical HDL-based implementations. Therefore, the processor model can easily be modified, expanded and verified using standard software development methodologies. Previous work demonstrated that the core frequency and area of the generated hardware are comparable to existing RTL implementations [1]. In this poster we present the status and roadmap around Comet.**

I. Introduction

Innovations in hardware architecture are leading to better processors which can support the complex software applications being developed today. Similarly, software and programming languages have made it easier to prototype, simulate and even synthesize hardware, for example the creation of High-Level Synthesis (HLS) tools. Standard development flows for processor architecture are based around the development and maintenance of a hardware model for synthesis (usually at Register Transfer Level), and a software model (i.e. an instruction-set simulator) to validate the applications that will run on the design. Those two models have to be verified independently. To cut down the complexity of the hardware development in a processor design methodology, Comet introduces the use of a single behavioral model of the simulator, which is compiled into a cycle-accurate bit-accurate simulator by standard compilers and also synthesized into a hardware component through HLS. It is also possible to debug the behavioral model at the C/C++ level and to simply validate the behavior of the generated hardware through co-simulation and standard software development tools. The design methodology based on Comet is depicted in Figure 1.

Although high-level synthesis is making huge progress in dealing with complex structures, how far can these tools go? Can they be used to design something as complex as a microprocessor? In particular how to specify such parallel computing pipelines (e.g., core pipeline stages, cache hierarchy, communications with uncore components) with the HLS technology and to demonstrate that there is a potential high gain in design time, without jeopardizing performance and cost.

II. Contributions

In this work, we present the design of Comet, a five-stage pipelined processor implementing the RISC-V Instruction Set Architecture (ISA). Comet is fully synthesized from a unique C/C++ model using state-of-the-art HLS tools. Our results show that, if the C/C++ model is written using the methodology we describe in this paper, the generated hardware is comparable or even better than other implementations when targeting an ASIC 28 nm technology node. Moreover, a fast, cycle-accurate and bit-accurate simulator of the processor can be compiled from the same C/C++ model. The resulting maximum performance of the simulator reaches 24 million cycles per second (Mcps) on integer workloads and around 15 Mcps on average, when executing floating-point and integer workloads.

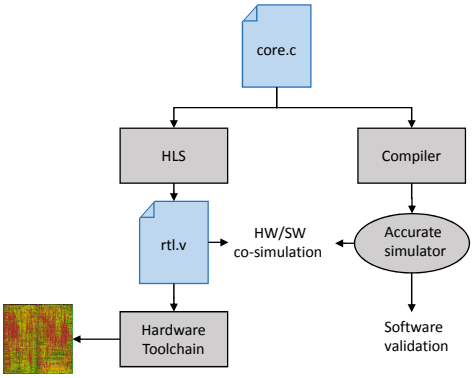

Fig. 1. Comet processor design flow.

The object oriented nature of C++ also greatly improves the readability and flexibility of the design description compared to classical HDL-based implementations. Therefore, the processor model can easily be modified, expanded and verified using standard software development methodologies.

Recent developments include compatibility with modern embedded operating systems such as Zephyr or RIOT. A shared memory multicore version supporting Linux is in its final step and an Out-of-Order version of Comet is on our roadmap. Comet is used in several projects around fault tolerance, intermittently-powered systems and hardware security.

References

[1] S. Rokicki, D. Pala, J. Paturel, and O. Sentieys, "What You Simulate Is What You Synthesize: Designing a Processor Core from C++ Specifications," in *ICCAD 2019 - 38th IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–8, Nov. 2019.