RISC-V Barrel Processor for Accelerator Control

MohammadHossein AskariHemmat*, Olexa Bilaniuk†, Sean Wagner‡, Yvon Savaria*, Jean-Pierre David*
*Electrical Engineering Department, Ecole Polytechnique Montreal, Quebec, Canada,
{mohammadhossein.askari-hemmat, jean-pierre.david, yvon.savaria }polymtl.ca
†Mila, University of Montreal, Montreal, Quebec, Canada, olexa.bilaniuk@mila.quebec
‡IBM Canada, Markham, Ontario, Canada, wagnerse@ca.ibm.com

Hardware accelerators are important in the post-Moore's law era of computing. To maximize performance of such accelerators, most of the logic resources should be allocated to their execution circuits, while control mechanisms should be kept small yet flexible. In this paper, we propose a barrel processor design based on the RISC-V instruction set architecture (ISA) [1]. To the best of our knowledge, this is the first implementation of a barrel RISC-V processor made public. The purpose of this processor is to concurrently control and coordinate a set of accelerator processing elements.

A barrel processor exploits thread-level parallelism by switching between different threads on each clock cycle. However, unlike simultaneous multi-threading (SMT) that is used in modern super-scalar processors, barrel processors do not issue more than one instruction per clock cycle. Instead, a single execution pipeline is shared by all threads. Barrel processors have the advantages of 1) a simplified multi-stage execution pipeline since data hazard detection and handling logic are not needed, and 2) a small size in terms of circuit resources since the main execution pipeline is shared between threads. Each thread only needs separate circuitry for storage elements that contains its state, i.e. data/address registers, the program counter, and status registers. Such a small processor allows more circuit resources to be allocated to high-performance accelerators.

The key advantage of a barrel processor is to provide real-time threading guarantees. Indeed, it avoids stalls for the processor and the accelerator. To keep up with the accelerator's demand for data, we either have to implement a dedicated state machine (which without saying would not be programmable and would offer less flexibility), or to use a dedicated controller implemented in the form of a simple processor for each individual processing element. We believe a barrel processor is a nice solution that not only guarantees deterministic thread execution on every clock cycle, but yet it is small and programmable.

Introduced in 2011, RISC-V [2] is a free and open ISA. Compared to other industry-grade ISAs, RISC-V is simple. The entire base ISA (RV32I) is composed of only 47 instructions. On the other hand, RISC-V is extensible. Even for the base ISA, there is a generous amount of opcode space which gives room for customizing the core to one's need. Apart from the base ISA, RISC-V is available for different data path widths and computation precision (e.g. long integer, single-precision floating-point, etc). The RISC-V specification

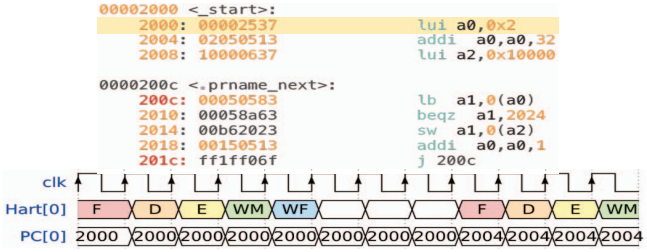

Fig. 1. This figure shows how instructions are executed in a barrel processor. Every 8 clock cycles, the program counter of the associated hart increments, which allows the pipeline to be implemented without any data or control hazard circuitry.

supports multiple hardware threads which are referred to as "harts". Each hart has its own register file and program counter, and it executes an independent sequential instruction stream [1].

In ordinary pipelined CPUs, the controller requires logic to resolve data and control hazards. Consider Figure 1: A 5-stage in-order CPU must resolve the read-after-write hazard of `0x2000` and `0x2004` on register `a0`, either by forwarding networks or by stalling `0x2004` until writeback 3 clock cycles later. It must also resolve the control hazard at `0x2010` by stalling until that branch is decided, or predict the result and potentially roll back if the prediction was incorrect. In our proposed RISC-V barrel processor, none of these cases need to be handled, since under a round-robin scheduler, each hardware thread returns to execution long enough after register/memory writeback that no forwarding paths, prediction units or stalls are required. The corresponding control logic can therefore be trimmed away.

Using an embedded barrel processor based on the RISC-V ISA attached to a multitude of accelerators is a simple way to implement a flexible software-programmable control mechanism that is compact. In a follow-up publication, we will present details on the implementation of such a processor.

The authors acknowledge support for this project from the IBM AI Horizons Network, and from the NSERC COHESA Strategic Research Network.

References

[1] *The RISC-V Instruction Set Manual*, December 2019. [Online]. Available: https://riscv.org/specifications/
[2] K. Asanović and D. A. Patterson, "Instruction sets should be free: The case for RISC-V," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146, Aug 2014.

2576-2621/20/$31.00 ©2020 IEEE
DOI 10.1109/FCCM48280.2020.00063