The BlackParrot Processor

An Open-Source Industrial-Strength RV64G Multicore Processor

Zahra Azad, Leila Delshadtehrani, Boyou Zhou, Ajay Joshi
Department of Electrical & Computer Engineering
Boston University
Boston, MA 02215
{zazad, delshad, bobzhou, joshi}@bu.edu

Farzam Gilani[†], Katie Lim[‡], Daniel Petrisko[‡], Tommy Jung[†], Mark Wyse[‡]*, Tavio Guarino[†], Bandhav Veluri[†], Yongqin Wang, Mark Oskin[†‡], Michael Taylor[†‡]
[‡]Paul G. Allen School of Computer Science & Engineering
[†]Deptartment of Electrical & Comptuer Engineering
University of Washington
Seattle, WA
{katielim, petrisko, wysem, oskin, profmbt}@cs.washington.edu,
[†]{farzamgl, tavio, dcjung, bandhav, yongqw2}@uw.edu
*presenter

***Abstract*—In this paper we introduce the BlackParrot multicore processor, a mainstream industrial-strength open-source implementation of the RISC-V RV64G architecture. BlackParrot is a clean-slate processor design with a lean, energy-efficient, and highly performant implementation. The key differentiator between BlackParrot and prior RISC-V processor efforts is that our goal is to distribute stewardship across industry and government stakeholders instead of adopting a "freemium" model where the source is controlled by a private startup that does not release the actual code it tapes out. Our approach enables a pathway for creating the equivalent of Linux for RISC-V: a truly open RISC-V processor to power the open-source hardware revolution and the Age of Bespoke Silicon.**

Keywords—open-source hardware; architecture; RISC-V

I. Introduction

Over the past few decades architects have reaped the benefits of traditional silicon transistor technology scaling to enable energy-efficiency and performance gains with each new technology process node. However, the past decade has witnessed the breakdown of traditional scaling referred to as Moore's Law, the failure of Dennard scaling, and an explosive growth in new application areas and demands. Together, these trends have sparked a revolution in hardware acceleration, specialized processors, and custom system design; an Age of Bespoke Silicon [1].

Inspired by the success of open-source software, many believe that an open-source hardware revolution will accompany the rise in custom hardware designs as architects explore the benefits of specialization and acceleration. However, one key challenge for building custom systems is choosing a base architecture for the system. Existing architectures are too costly or simply unusable for custom and heterogeneous systems, causing architects to search for new alternatives. Fortunately, the recent introduction of the RISC-V architecture [2], a fully open instruction set architecture (ISA), promises to solve this problem. Since its introduction, several processors have been designed and released for the RISC-V architecture. While we commend these efforts, each one has shortcomings that make it unsuitable as the processor of choice for the open-source hardware ecosystem.

In response, we have designed the open-source BlackParrot multicore processor. BlackParrot implements the RISC-V RV64GC architecture and is designed from the ground up to be a mainstream, industrial-strength processor capable of powering the open-source hardware revolution. A key research question for BlackParrot is how to architect the processor microarchitecture with near-zero overhead interfaces to enable a decoupling of the hardware design and testing, enabling the project to scale to a large number of open-source contributors and maintainers. In the rest of this paper we briefly describe the RISC-V architecture and then describe the design of the BlackParrot processor, its cache coherence system, and the metrics of success we will use to evaluate our design.

II. The RISC-V Instruction Set Architecture

The RISC-V instruction set architecture (ISA) [2] is an open-source instruction set designed for the future of computing. Since its introduction, it has emerged as the best-of-class architecture for the open-source hardware ecosystem and has gathered the support of many throughout academia and industry. The RISC-V architecture was designed from the ground up to be a real, usable architecture, integrating lessons learned from decades of architecture development, deployment, and use.

The BlackParrot multicore processor leverages the RISC-V ecosystem because of its many benefits. The RISC-V architecture was designed to be an ISA that is (a) open and free for academia and industry to use, (b) suitable for both simulation and native hardware implementation, (c) build on a small required base instruction set, and (d) not over-architected to provide a clean separation between hardware implementation and software running on top of it. In addition to these design principles, the RISC-V software ecosystem is rapidly maturing such that RISC-V is Linux capable and easy to compile to. The architecture includes extensibility features that make it well-suited for integration in heterogeneous and accelerator-rich systems. These factors contribute to creating not only an open architecture, but a hardware and software ecosystem around it that can be used by the open-source hardware community.

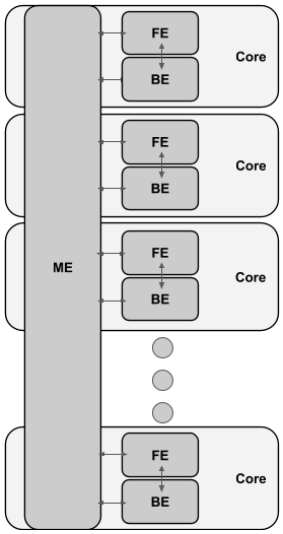

Figure 1. BlackParrot High-Level Block Diagram

III. The BlackParrot RV64G processor

BlackParrot is a clean-slate multicore processor implementation of the RISC-V 64-bit RV64GC architecture. The RV64G architecture includes the base RISC-V 64-bit integer ISA ("I"), as well as the integer multiplication and division ("M"), atomics ("A"), and single and double precision floating-point ("F" and "D") standard extensions. These extensions, with the addition of the compressed instruction ("C") extension, are sufficient to implement a fully-featured, energy-efficient, and high-performance Linux-capable RISC-V processor.

The BlackParrot multicore processor is designed to be a mainstream, industrial-strength implementation of the RISC-V architecture that will drive the future growth of the open-source hardware ecosystem. BlackParrot is built using industry standard design principles with the goal of being easy to verify and extend. We embrace the philosophy of IP reuse and modular design and will extensively leverage hardware blocks and IP from the BaseJump STL [3].

The BlackParrot multicore processor design comprises BlackParrot RV64GC processor cores and a cache coherency system. Each BlackParrot core is an in-order, 64-bit wide, pipelined processor core with virtual memory support. Each core is decomposed into two modules called the Front End and Back End. A third module called the Memory End connects to one or more processor cores and provides the L2 memory system, including cache coherence. Figure 1 shows a high-level block diagram for a BlackParrot multicore processor comprising multiple cores and a Memory End. These modules are connected via well-defined, narrow interfaces that transfer only the minimum amount of information required, such as the instruction stream and branch prediction data.

A. Front End

The Front End contains instruction fetch, branch prediction, and the L1 instruction cache. It presents an in-order, but potentially speculative, instruction stream to the Back End for processing. The primary modules contained within the Front End are the L1 instruction cache, the Instruction TLB, PC Generation, and compressed instruction decompressor. The L1 instruction cache typically has 64-byte cache lines, is 8-way associative, and has a typical capacity of 32 KB. The instruction cache interfaces with the Memory End and participates in cache coherence. The compressed instruction decoder provides support for the "C" extension of the RISC-V ISA, which is now required to support the Linux operating system. The use of compressed instructions may result in between 25%-30% code-size reduction [2]. The PC Generation module contains the branch predictor and can readily be changed to accommodate new branch prediction strategies. The baseline architecture employs a branch predictor containing a Branch History Table (BHT) and Branch Target Buffer (BTB).

B. Back End

The Back End is responsible for maintaining the correct architectural state of the processor and contains instruction execution, speculation resolution, the Data TLB, and the L1 data cache. Logically, the Back End controls the Front End and may command the Front End to flush all state or redirect instruction fetch (e.g., branch misprediction).

As the Front End provides instructions to the Back End, the Back End's scheduler dispatches instructions to the execution units. The execution pipeline is single-issue and non-stalling. All instructions proceed to completion unless an exceptional condition is encountered. The most common exceptional conditions are branch mispredicts and cache misses. If a branch mispredict is detected, the instructions following the branch in the pipeline, which are speculative, are poisoned and the fetch unit is redirected to the appropriate address. If a cache miss is detected, subsequent instructions are again poisoned to preserve the architectural state of the processor. After the cache miss is resolved and the line is filled into the appropriate cache, the missing load or store is replayed. The memory operation replay is handled by rolling back the instruction queue between the Front End and Back End to the memory operation and then re-executing the operation and any following instructions, which avoids refetching an instruction cache line.

Memory operations are sent to the L1 data cache, which has the same typical organization as the L1 instruction cache and is a 32 KB 8-way associative cache with 64-byte lines. The data cache also interfaces with the Memory End and participates in the cache coherence protocol.

IV. Memory End and Cache Coherence in BlackParrot

The BlackParrot processor cores, comprising a Front End and Back End, connect to the Memory End module, which provides cache coherence and the L2 memory system. The Memory End comprises a coherence message network, the Cache Coherence Engines (CCEs), Local Cache Engines (LCEs), and the L2 caches. Figure 2 depicts the high-level architecture of the Memory End, with multiple LCEs connected via the coherence network to multiple CCEs and attached L2 cache banks. Although the LCEs are tightly coupled with the instruction and data caches in the Front and Back End modules, respectively, we consider them part of the

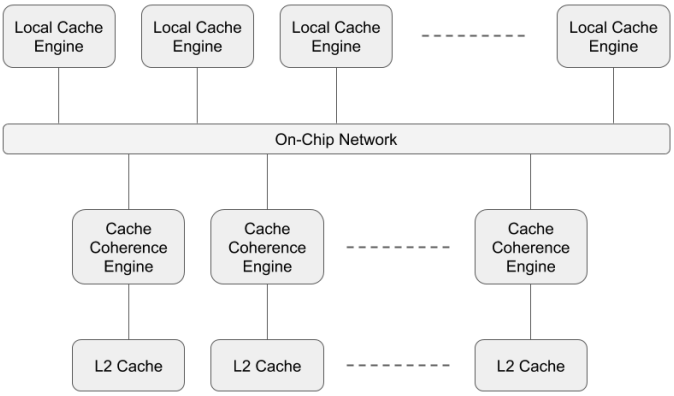

Figure 2. BlackParrot Coherence System Block Diagram

Memory End since they are the L1 coherence agents. The primary function of the Memory End is to provide coherent shared memory to one or more BlackParrot processor cores, and to support the memory consistency model.

The coherence system is designed with an emphasis on protocol simplicity and ease of verification. The coherence system comprises Local Cache Engines (LCEs) and Cache Coherence Engines (CCEs). The CCE is responsible for managing coherence for a subset of the physical address space while the LCE is responsible for initiating and responding to coherence requests for an L1 entity (data cache, instruction cache, I/O cache, etc.). There is one LCE attached to each L1 entity in the system and there may be many LCEs and CCEs in a BlackParrot processor. The LCEs connect to the CCEs via a coherence network and each CCE connects to an L2 victim cache. The L2 victim caches are non-inclusive of the L1 caches. Each CCE manages a subset of the physical address space, with addresses typically striped across the CCEs.

The CCE is responsible for managing coherence and implements a directory-based cache coherence protocol. The CCE supports EI, MSI, and MESI coherence protocols, and may be extended to support additional protocols. The coherence engine is a microprogrammed controller that executes a set of basic arithmetic, directory maintenance, and coherence message operations. The microprogrammed CCE provides considerable flexibility and adaptability when implementing coherence protocols, and additionally provides opportunities to explore unique uses of a coherence engine within the multicore processor. The coherence directory is fully inclusive of all attached LCEs and contains exactly one entry for every L1 cache block that may be cached in the system. The directory holds the true copies of all the L1 caches' tag sets as well as the coherence state for every block. The CCE performs all coherence state updates for every cache block in the system and acts as the point of serialization in the coherence protocol.

A BlackParrot system may contain many LCEs, for example one for each data and instruction cache in each core. Each LCE is responsible for initiating and responding to coherence requests for its associated L1 entity. The LCE performs a small set of simple operations that allow it to respond to coherence requests initiated by a CCE or to request a cache block's coherence state be changed in the CCE directory. Each LCE maintains a shadow copy of the cache tags for its L1 entity and all changes to the tags must be performed by the CCE. Delegating coherence state management exclusively to the directory significantly reduces the complexity of the coherence protocol and the number of required transient states.

V. BlackParrot Success Metrics

When introducing a new processor or hardware design, it is important to define the metrics that will be used to evaluate the design. The BlackParrot processor has four key metrics for success, some that are familiar to designers, and others that are unique to the environment and use cases we are targeting. These metrics are Quality, PPA (Power, Performance, and Area), Functionality, and Virality. We briefly overview these metrics in the following subsections.

A. Quality

A key metric for success of any new hardware design is quality. A design must be high quality for users to trust that it will behave as expected and advertised. There are many factors that influence quality and it is important to produce a design that has all these properties. A high-quality design should contain code that is easy to understand and well documented. A high-quality design should be well tested, verified, and validated. A high-quality design should be secure to protect the users, applications, and environment running on it. Combining these quality factors will enable the success and adoption of BlackParrot. Users must trust that the design is of high enough quality to fabricate into silicon, which can only be achieved if the above factors are present.

B. Power, Performance, and Area (PPA)

The second metric for success used by BlackParrot is PPA, or Power, Performance, and Area. PPA is a metric frequently employed by hardware designers to evaluate the physical efficiency of a design. A successful hardware design should strive to have the greatest performance while requiring the minimal amount of power and/or energy and occupying the least amount of physical area in silicon. BlackParrot strives to be Pareto optimal in PPA. Achieving a design point on the Pareto optimal frontier will encourage other designers to incorporate the BlackParrot processor into their designs.

C. Functionality

The third measure of success for the BlackParrot processor is functionality. The processor design will be considered successful if its functionality is useful, but also well justified and not excessive. The processor must provide the features that users require, such as supporting Linux, I/O devices, shared memory, extensibility, and modularity. However, equally important, the design must not include features that users do *not* require. Too often, processor designers introduce feature after feature into the design that significantly increases the complexity and testing and verification burden for those adopting the design, while providing benefits to only a very small fraction of all users. We believe that it is better to

embrace the motto of "Be Tiny," and include only the functionality that benefits most users and does not unnecessarily increase the complexity of the design or its verification.

D. Virality

The last metric we use to evaluate the success of the BlackParrot processor is virality. Perhaps a new metric for hardware designers, virality is the measure of the community energy surrounding our design. As an open-source project, we hope to replicate the success of the open-source software movement, and therefore the success of BlackParrot must be measured in comparable ways to that of the most popular open-source software projects such as Linux and LLVM. We hope that BlackParrot will be compelling enough to convince the best hardware designers to improve it, draw large numbers of users, and to convince companies to invest in the design and become stewards of the code base. A key enabler of the open-source hardware movement will be a collection of high-virality projects, and BlackParrot aims to be the processor of choice in this environment.

VI. Conclusion

In this paper we present BlackParrot, an open-source, industrial strength processor implementing the RISC-V RV64G architecture. BlackParrot is designed from the ground up with a lean, energy-efficient, and highly performant implementation. Inspired by the success of open-source software, BlackParrot will be fully open-source with the goal of transitioning stewardship of the design and code to industry and government stakeholders. BlackParrot is the processor that will enable the open-source hardware revolution and the age of bespoke silicon.

Acknowledgment

This research is funded by the DARPA POSH program.

References

[1] M. B. Taylor, "Bitcoin and the age of bespoke silicon," 2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES), Montreal, QC, 2013, pp. 1-10.

[2] A. Waterman and K. Asanovic, Editors, "The RISC-V instruction set manual: Volume 1: User-Level ISA," Version 2.2, May 7, 2017.

[3] M. B. Taylor, "BaseJump STL: SystemVerilog needs a standard template library for hardware design," Design Automation Conference (DAC), June 2018.