Discerning Performance, Power, Energy and Area Efficacies of Democratized ISA Effort

ABSTRACT

Instruction Set Architectures (ISA) are fundamental to how a wide variety of modern days computer systems – ranging from simple hand held mobile devices to large scale data centers and server farms – are conceived, designed and implemented. Often, ISA designer goal is to capture most basic functions and tasks that can be then used to compose complex applications and softwares. Expectation is that a computing system should perform functions captured by ISA set in most possible efficient way in terms of performance (single-cycle) and power/energy. While an ISA is central to computer design, there have been only a handful of successful ISAs forcing designers to choose them from a small subset even though it might not be the most efficient in capturing higher level applications. Unlike compilers, OSs, drivers, and other software components, ISAs have been a proprietary component by-and-large.

Democratization of ISA was the main theme behind the advent of RISC-V. It was touted to relieve the designer community and small to mid-scale OEMs from the clutches of proprietary ISA suppliers. While this is a novel thought in spirit, much depends upon the efficacies of democratized ISA itself. In this work, we set out to discern and quantify the viability of an open source ISA such as RISC-V.

To best of our knowledge, this is first work to compare RISC-V with its popular proprietary counterparts (ARM and MIPS). We used state-of-art simulation and emulation frameworks: *qemu* for program analysis and *gem5* for microarchitectural simulations. We also present concrete cases where RISC-V clearly falls behind compared to ARM, MIPS ISAs and what addendum could possibly make it competitive. To our surprise, we also stumbled upon cases where RISC-V is better than proprietary ISAs. Overarching goal of our exploration is to enable RISC-V designers so that they can augment their designs and be able to close the gap with other state-of-art ISAs.

1. INTRODUCTION

Numerous isas have been designed but couldn't survive because they either didn't offer anything unique or they were poorly designed.

RISCV is an emerging Instruction Set Architecture (ISA) and has become an important option for both academia and industry when considering new microprocessor designs. Features like modularity, extensibility, simplicity, and being open and free to use, make RISCV an attractive option for next generation of processors especially in embedded systems domain where new, customized, low-power, and efficient cores are needed. In this paper we present a comparative study on impacts of three well-known Instruction Set Architectures (ISAs) (MIPS, ARM, and RISCV) on Performance, Power, and Area (PPA) of state-of-the-art embedded processors through a systematic measurement campaign using several different toolchains and frameworks and several standard benchmark suites. We particularly study the impact of these ISAs on important metrics such as static and dynamic Instruction Count (*icount*), Cycle Count, Microarchitectural Statistics (e.g. MPKI, Branch Prediction Accuracy, etc.), Dynamic Power, and Core's Area and report our key findings on impacts of using different ISAs on each of these metrics. We find that some of these metrics are ISA-dependent and others are dependent on other factors such as compiler, runtime libraries, and specific microarchitectural features. Our main conclusion is that while comparing to MIPS and ARM, RISCV has some shortcomings and design/toolchain issues that should be addressed and fixed, due to its intrinsic features such as modularity it provides a great opportunity for designing customized PPA-efficient cores.

Instruction Set Architectures (ISAs) has a key role in designing cores for different domains, where x86 ISA has become dominant in desktop and server domains, and ARM has become the dominant ISA in mobile, tablet, and embedded system domain. The question of impact of ISA design on different Performance, Power, Area (PPA) metrics has traditionally been an important concern for designers and semiconductor industry especially in the 1980s and 1990s when chip area and processor design complexity were the primary constraints [24, 12, 17, 7]. In the past decade, we radical changes in computing landscape and rise of mobiles and tables and increasing popularity of ARM ISA this question again becomes an important issue.

Today, with proliferation of embedded and cyber-physical systems (e.g. IoTs) and increasing popularity of domain-specific languages and emerging applications like machine-learning and more importantly, introduc-

tion of a new, open-source, modular ISA (RISCV), this question once again becomes an interesting topic for research.

To answer this question, in this paper we present a comparative study on impacts of using three different ISAs (MIPS, ARM, and RISCV) on important metrics such as static and dynamic instruction count (icount), total execution time (cycle count), dynamic power, and area. We show which of these metrics are ISA-dependent and what are the other important factors on PPA. Using these experiments we pinpoint the shortcomings, issues, and advantages of using RISCV ISA over ARM and MIPS ISAs.

2. BACKGROUND

RISC-V is an emerging open-source software and hardware ecosystem that has gained in popularity in both industry and academia [2, 11]. At the heart of the ecosystem, the RISC-V ISA is designed to be open, simple, extensible, and free to use. The RISC-V software tool chain includes open-source compilers (e.g., GNU/GCC and LLVM), a full Linux port, a GNU/GDB debugger, verification tools, and simulators. On the hardware side, several RISC-V prototypes (e.g., Celerity [4]) have been published. The rapid growth of the RISC-V ecosystem enables computer architects to quickly leverage RISC-V in their research.

3. METHODOLOGY

To study the effects of ISAs on Power, Performance, and Area, we used several different metrics using 4 different tools and more than 10 standard benchmark applications. Followings describe the frameworks, metrics, and benchmarks used in this paper. The reader can skip this section if he/she is uninterested in these details.

3.1 Framework

To find the dynamic and static instruction count (ICOUNT), we use a well-known open-source emulator called Quick-EMUlator (QEMU). QEMU is a hosted virtual machine monitor: it emulates the machine's processor through dynamic binary translation and provides a set of different hardware and device models for the machine, enabling it to run a variety of guest operating systems. We chose QEMU primarily cause it can emulates MIPS, ARM, and RISCV ISAs in user-mode.

Static icount: Static instruction count is an important metric

4. EXPERIMENTAL RESULTS AND ANALYSIS

5. RELATED WORK

6. CONCLUSIONS

7. REFERENCES