Final submission

ClockHands: Rename-free Instruction Set Architecture for Luis D. Pena

Out-of-order Processors EECE.5821

# **Summary**

## **What is the problem the paper tries to solve?**

The paper addresses the problem of speeding up irregular programs while improving power efficiency. In irregular programs the control flow and memory access patterns are data-dependent and statically unpredictable. These characteristics of irregular programs require to read and write data at random based in the input data is continuously generated, therefore suitable data structures are required to address this unpredictable data access such as pointer-or linked list-based data structures. Irregular programs are key in the execution, optimization, and compilation of interpreted and high-level programming languages, gaming, and social network analysis, to name a few applications. This contrast with regular programs in which memory accessed is more linear in nature and less computing time is spent navigating the data structure. According to the paper, the only solution that the industry offers to speed up irregular programs is to integrate more massive out of order cores, which improve processor’s single thread performance, which is key to reduce latency. However, although this unique solution seems to solve the problem, it does so with poor power efficiency. To address this issue, instead of increasing out of order cores the paper propose a novel instruction set architecture called ClockHands and compare its power performance with existing instruction ISA’s such as RISC-V and STRAIGHT. The benchmark programs used for their evaluation were bzip2, mcf\_s, lbm\_s, and xz\_s included in SPEC2006/2017 and CoreMark. The authors use these benchmarks because they are written entirely in C, which was important since authors were only capable of developing a C compiler for their novel ClockHands ISA due to the complexity that other programming languages such as C++/Fortran compilers entitle in terms of development time and effort.

## **What are the main ideas or insights of the paper?**

The paper tries to solve the problematic with a different approach, abandoning the idea of increasing out of order cores and focusing its attention on the ISA, how it manages register operands, and how this affects power efficiency. The paper reviews two existing instruction set architectures: conventional RISC-V, and STRAIGHT. The paper points out that in RISC architecture, a register number (name) needs to be specified and they are fix, so to access data we need to read from memory to these registers execute the instructions and then write back to the memory and repeat the process, this process is referred as renaming the registers. Renaming occurs more frequently in irregular programs because of the need to read and write memory dynamically based on the incoming data. Due to register renaming poor power efficiency emerges, preventing an increase in the front-end width. In this context, the “front-end width” relates to the number of instructions fetched from memory during each clock cycle. It’s relevant in out-of-order execution processors.

In contrast, the second architecture is called “STRAIGHT” because it executes each instruction directly without renaming its operands, instead it has a unique instruction format in which the source operand is given based on the distance from the producer instruction, which eliminates register renaming since register renaming is completely removed from the pipeline. However, due to its strong constraint in instruction placement, a large increase in number of instructions is generated when the C code is compiled for this architecture, in comparison to RISC, which does not help with power efficiency, since more instructions implies consuming more power. After this comparison, the paper then propose a novel solution called ClockHands and compare its power efficiency with RISCV and STRAIGHT. ClockHands try to take the good of both worlds, since as opposed to RISCV it does not require register renaming and in contrast to STRAIGHT it has looser constraints on instruction placement which reduce number of instructions generated at levels like RISCV when compiling the same C code.

# **Strengths**

## **Does the paper solve the suggested problem well?**

Methodology wise, the paper did a good job measuring irregular program execution for each ISA against a common ruler or benchmark. The programs used for their evaluation were bzip2, mcf\_s, lbm\_s, and xz\_s included in SPEC2006/2017 and CoreMark benchmarks. For this purpose, implementing cycle-accurate simulator, FPGA implementation, and first step compiler for ClockHands was necessary. This effort also allowed instruction count comparison of STRAIGHT and ClockHands which provided an empiric evidence of the cause of increased instruction count in STRAIGHT, and a method of verification that this issue does not occur in ClockHands, but in the contrary that ClockHands is similar RISCV in terms of instruction counts, which is good since fewer instructions count improve power consumption.

The solution presented ClockHands, simulation was fundamental to estimate power consumption. Overall, ClockHands did what it promised, reduce power consumption. Results presented show the following remarks:

1. On a machine with an eight-fetch width:
   1. results showed that Clock-hands consume 7.4% less energy than RISC.
   2. And its performance was comparable to RISC.
2. simulating a futuristic up-scale processor with a 16-fetch width.
   1. results showed that Clock-hands energy reduction increases up to 24.4%.
   2. which shows that Clock-hands enables a more power efficient and wider front end?

# **Weakness**

Although they offer cycle accurate simulation, I think would’ve been beneficial to include different simulation tools, for example in the paper they used We used a cycle-accurate simulator, Onikiri2, for the performance evaluation and McPAT for the energy consumption simulation. However, there are other popular simulation tools like SESC simulator and Qemu DBI tool. Simulation will always be a simulation, and although it can predict the trends it can never mimic reality, that is why having multiple methods to validate a proposed architecture is always welcome and simply good practice. Also, taking in consideration that each ISA is unique and perform differently it would’ve been good to compare ClockHands with others ISA besides RISCV and STRAIGHT, since ClockHands took their strength and build upon them implementing a solution which didn’t includes their weaknesses. I think there is room for improvement there.

## **What have you learned and what were your likes and dislikes in the paper?**

Although we covered the topic of how the different variables affect performance, CPI, ISA, etc. in class, it was refreshing to see in practice how the ISA can affect performance, specifically power consumption. Also, I liked also that I could relate with the terminology, and that the terms and concepts that we covered in class were relevant to understand the paper. In addition, I’m taking an advance digital design class in which we are introduce FPGA design, which I’m very excited about, and it was good to see that this paper used FPGA in the problematic of solving Computer architecture problematic, which is a plus.

Overall, I didn’t have major dislikes, the paper was well written, however it was a little be confusing at first, since in the abstract section the paper first states the following “a recently proposed architecture called STRAIGHT specifies operands by inter-instruction distance, thereby eliminating register renaming”. Then a paragraph after they state “ClockHands does not require register renaming as in STRAIGHT” which is a contradiction, or maybe a bad choice of words, I think they meant to say ClockHands does not require register renaming like RISCV does, this idea was later clarified as we read the paper.