Work-in-Progress: Fine-Grained Acceleration using Runtime Integrated Custom Execution (RICE)

Leela Pakanati*
pakanalk@rose-hulman.edu
Rose-Hulman Institute of Technology
Terre Haute, Indiana

John T. McMichen*
mcmichjt@rose-hulman.edu
Rose-Hulman Institute of Technology
Terre Haute, Indiana

Zachary Estrada
estrada@rose-hulman.edu
Rose-Hulman Institute of Technology
Terre Haute, Indiana

ABSTRACT

Runtime Integrated Custom Execution (RICE) relocates traditional peripheral reconfigurable acceleration devices into the pipeline of the processor. This relocation unlocks fine-grained acceleration previously impeded by communication overhead to a peripheral accelerator. Preliminary simulation results on a subset of the PARSEC benchmark suite shows promise for RICE in HPC applications.

CCS CONCEPTS

• **Computer systems organization → Reconfigurable computing**; • **Hardware → Hardware accelerators**.

KEYWORDS

RISC-V, Hardware Acceleration, FPGA, Hardware-Software Co-Design, Gem5, PARSEC

ACM Reference Format:
Leela Pakanati, John T. McMichen, and Zachary Estrada. 2019. Work-in-Progress: Fine-Grained Acceleration using Runtime Integrated Custom Execution (RICE). In *2019 International Conference on Compliers, Architectures and Synthesis for Embedded Systems Companion (CASES'19 Companion), October 13–18, 2019, New York, NY, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3349569.3351536

1 INTRODUCTION

Runtime Integrated Custom Execution (RICE) reimagines the traditional location and role of a reconfigurable accelerator, moving from a peripheral I/O device to being integrated within the datapath of the processor itself. This approach aims to open up the possibilities for small scale, fine-grained accelerations by avoiding the pitfall of communication overhead that prevents traditional reconfigurable accelerators from accomplishing this style of acceleration. This relocation allows the accelerator to be treated as just another execution unit. The processor is then able to utilize the accelerator through the regular instruction stream, issuing the associated custom instructions, and committing their output to the register file. Since the accelerator is implemented in a reconfigurable fabric, the accelerator can be reconfigured during processor runtime to match the accelerator to the running application, switching between programs.

Traditional peripheral accelerators must communicate to an off-chip device to pass along data and instructions, typically through a memory-based interface with latency timings to match [6]. When integrated into the datapath, the accelerator receives its data as an execution issue just like any other execution unit. With the peripheral communication overhead removed, many previously inefficient or unfeasible finer-grained acceleration techniques start to become possible avenues for developers. Those possibilities allow one to accelerate not only applications with large, commonly used functions, but smaller traces that are repeatedly executed.

In traditional peripheral accelerators, much of the infrastructure handling the accelerator's execution is very similar to the existing infrastructure in the processor, resulting in repeated logic. By incorporating the accelerator into the datapath as its own instruction, the accelerator is able to take advantage of the existing front-end infrastructure of a modern out-of-order processor. This reduces resource utilization on the reconfigurable fabric while performing the same acceleration. The need to handle the movement of these custom instructions through the pipeline is reduced by the use of an out-of-order design which allows the custom functional unit to be treated like a black box. When issuing the custom instructions, the front-end need only provide the relevant operands. The custom instruction can then be resolved using a simple handshake protocol similarly to any other instruction upon retirement.

2 RELATED WORK

ReMAP is a reconfigurable architecture that promotes acceleration of heterogeneous multiprocessors [6]. Threads share a common reconfigurable fabric that can be used by an individual thread for computation, or to provide communication between threads with built in computation. The communication to and from the reconfigurable fabric is handled through the L1 cache, limiting the opportunities for fine-grained accelerations due to the long latency of memory accesses.

The Just-in-Time Customizable (JiTC) processor considers the design space between ASICs and reconfigurable peripherals [3]. JiTC implements a functional unit that is tightly integrated to the processor pipeline and is responsible for handling an instruction set extension. This functional unit uses operation chaining to produce complex operations by combining multiple simple operations to form the accelerated instructions. This approach reduces the inefficiencies of a pure reconfigurable fabric, but the operations that are available for chaining limits the freedom in design granted to custom instruction developers.

*Both authors contributed equally to this research.

CASES'19 Companion, October 13–18, 2019, New York, NY, USA

ACM ISBN 978-1-4503-6925-1/19/10…$15.00
https://doi.org/10.1145/3349569.3351536

3 PROPOSED APPROACH

RICE implements accelerators as their own custom instruction within the RISC-V ISA. To create a new instruction, software infrastructure must be in place to support the new encoding. Instruction stubs must be added to the compiler toolchain to allow for custom instructions to access any legal combination of operands, as specified by the RISC-V ISA. For register accesses, a destination register, and up to two or three source registers may be used, following traditional RV64-G stereotypes [5]. It is imperative that the stubs available in the software be supported by the existing processor architecture as doing otherwise would require major changes to the infrastructure.

3.1 Fine-Grained Acceleration

To select points of interest for acceleration, the GNU Profiler (gprof) is utilized to breakdown the program runtime by function. Compressing instruction traces that display the longest cumulative runtime into a single instruction frees up space in the execution window of the processor. Programs can experience a speedup in design spaces that would otherwise be inaccessible by performing hardware acceleration on these fine-grained acceleration targets. The reuse and design around existing processor architecture allows one to have more control over how these operations are performed and speedups can be attained by removing dependencies from the pipeline. Although these execution units will generally have longer latencies than regular instructions—due to the increased complexity in the custom instructions and the slower clock rate of the reconfigurable fabric—that latency is recoverable by the increased instruction density.

3.2 Instruction Set Extension

By adding custom instructions onto a general-purpose ISA, the processor can better handle domain-specific tasks. This approach allows for more flexibility in design and enables the platform to be easily applied to a variety of implementations. Implementing the custom instructions on a reconfigurable fabric also allows a general-purpose design to dynamically adapt to the needs of the currently running program.

3.3 Future Work

The primary focus of future work is understanding the trade-offs of RICE over traditional peripheral accelerators. This space can be explored by running the benchmarks without acceleration, accelerated using a peripheral accelerator, and accelerated using RICE. In addition to testing HPC applications, performing fine-grained acceleration of ALPBench—a benchmark suite targeting common multimedia applications such as MPEG-2 encoding/decoding, recognition, and ray tracing [4]—will give insights into other domains where RICE could be applied.

Table 1: Runtime comparison with RICE accelerator, runtimes presented in seconds of execution time.

Benchmark	blackscholes		freqmine		fluidanimate	
Working Set	**Small**	**Medium**	**Small**	**Medium**	**Small**	**Medium**
Baseline	0.1426	0.7033	1.0772	4.2090	0.3953	0.8449
Accelerated	0.0805	0.4169	1.0651	4.1856	0.3767	0.7955
Speedup	1.771	1.687	1.011	1.006	1.049	1.062

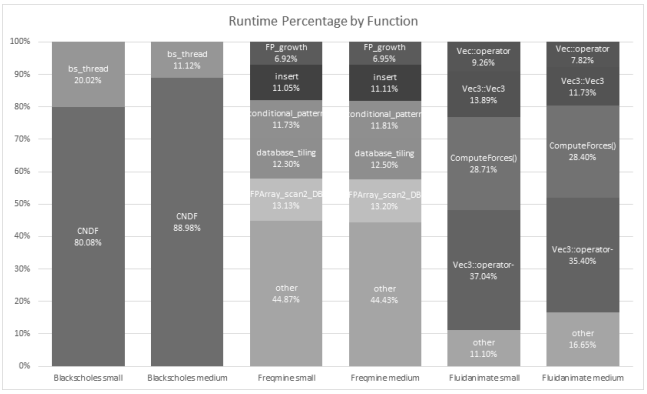

Figure 1: Benchmark profiling of runtime used by each function

4 EXPERIMENTS

Preliminary testing of RICE was performed through simulations. Gem5 [2] was used to simulate a subset of the PARSEC [1] benchmark suite to provide insights into how RICE applies to HPC. The O3CPU model, a general-purpose out-of-order processor, was used, as RICE depends on an out-of-order infrastructure. Figure 1 shows the runtime profile from gprof of the benchmarks that were selected for their variety of runtime profiles. For each benchmark run, the associated accelerator was added as an execution unit to run the custom instruction accelerators as determined by the function runtimes. The selected benchmarks were run on small and medium data sets, the results of these simulations can be found in Table 1.

The preliminary experiments show that `blackscholes`, which involved a more centralized function runtime with a denser instruction profile was very successfully accelerated. The other two benchmarks, where the runtime was more evenly distributed across functions, were still able to realize noticeable accelerations with small resource utilization (Table 2) by combining only a few instructions. The experimental results indicate that the application of RICE has varying effectiveness and will depend on how widespread the acceleration can be applied to the program.

Table 2: FPGA resource utilization of each accelerator.

Benchmark	LUTs	Slice Registers	DSPs	Num. Instr.
`blackscholes`	2275	851	6	1
`freqmine`	68	32	0	2
`fluidanimate`	578	493	4	1

REFERENCES

[1] Christian Bienia. 2011. Benchmarking Modern Multiprocessors. Ph.D. Dissertation. Princeton University.

[2] Nathan et al. Binkert. 1984. The gem5 simulator. ACM SIGARCH Computer Architecture News 39, 2 (1984), 1–7.

[3] Liang Chen, Joseph Tarango, Tulika Mitra, and Philip Brisk. 2013. A Just-in-Time Customizable processor. In Proceedings of the IEEE/ACM International Conference on Computer-Aided Design.

[4] Man-Lap Li, Ruchira Sasanka, Sarita V. Adve, Yen-Kuang Chen, and Eric Debes. 2005. The ALPBench Benchmark Suite for Complex Multimedia Applications. IEEE International Symposium on Workload Characterization (2005).

[5] Andrew Waterman and Krste AsanoviÂć. 2017. The RISC-V Instruction Set Manual: User-Level ISA. (2017).

[6] Matthew A. Watkins and David H. Albonesi. 2010. ReMAP: A Reconfigurable Heterogeneous Multicore Architecture. In Proceedings of the IEEE/ACM International Symposium on Microarchitecture.