Configurable Tightly-Coupled FPGA-style fabric for Fine-Grained Acceleration

David J. Schlais, Heng Zhuo, Mikko H. Lipasti
University of Wisconsin-Madison
schlais2@wisc.edu, hzhuo2@wisc.edu, mikko@engr.wisc.edu

I. Introduction

Due to the slowdown of Dennard scaling and the *dark silicon era*, computer architects introduce hardware accelerators to maintain expected performance improvements without violating power budgets. These accelerators improve energy efficiency over a processor's general-purpose pipeline and increase performance by trading off software flexibility for performance. Initial hardware accelerators often targeted large tasks, such as video encoding [1], graphics, etc. Then, loosely-coupled *sea of accelerators* [2] became a model to accelerate several different domains. On the other hand, today, we see tightly-coupled accelerator proposals for heap management, hash maps, string functions, and fine-grained tasks [3] [4]. SIMD is in the same class of fine-grained acceleration, where a small group of operations are replaced by a single instruction.

II. Problem Statements

Our first-order modeling in Figure 2 shows that if you require an accelerator to execute non-speculatively (NL, non-leading) or/and with a dispatch barrier (NT, non-trailing), fine-grained tasks benefit less from acceleration due to ROB drain/fill or CPU idle penalties. Many *prior* academic accelerator papers focused on accelerating coarse-grained tasks [5], where pipeline drains and fills created negligible impact on overall speedup (see left-hand side of Figure 2), and these performance impacts were often ignored. However, we show that at the granularity of many *current* academic paper accelerators, the four different implementations have a dramatic range of overall program speedup, where some implementations can actually cause slowdown.

Understanding the tradeoffs between each implementation will be an important step for choosing the optimal implementation based on the type of accelerator, processor architecture, program behavior, and desired metrics.

However, to our knowledge, no prior research has numerically evaluated these tradeoffs, leaving architects to either try to intuitively predict the optimal implementation or spend a large amount of time to design and test each potential implementation.

For these reasons, our research involves creating novel analysis of:

This project is funded in part by NSF grant CCF-1615014.

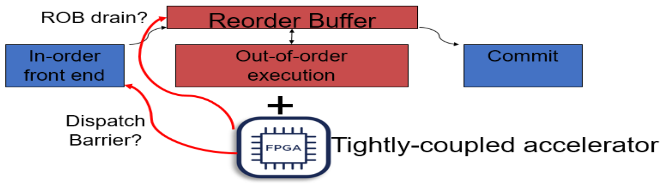

Fig. 1. High-level system diagram. Our studies evaluate impacts of integrating tightly-coupled accelerators into OoO Cores, and how design choices requiring dispatch barriers or ROB drains can have significant performance impacts.

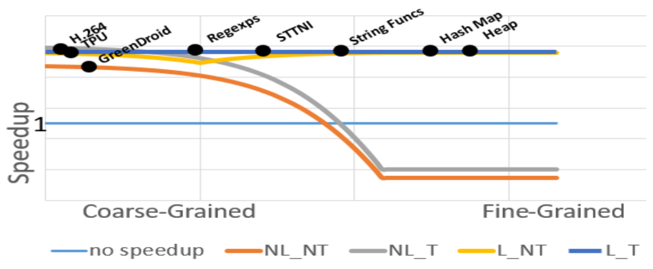

Fig. 2. Analytical performance model of different accelerator implementations over different granularities. Estimated granularities of H.264 [1], Google's TPU [6], GreenDroid [7], Speech Recognition using STTNI [8], Heap management [4] [3], regular expression [4], string function [4], and hash map [4] acceleration. Baseline of an ARM A72 processor, 20% of code acceleratable with 10x speedup. Note that fine-grained acceleration exacerbates the difference in implementation speedup.

1) Thorough evaluation of the more intricate performance penalties that arise from tightly-coupled accelerators.

2) The previously non-modeled design-space considerations for building tightly-coupled accelerators.

3) How to use (1) and (2) to create domain-specific programmable and/or configurable tightly-coupled hardware accelerators.

III. Proposed Innovations

To address (1) and (2), we create a tightly-coupled accelerator performance model. This first-order modeling takes into account processor characteristics (ROB size, issue and commit rates, misprediction penalty, acceleration factor), program behavior (% acceleratable code, invocation frequency), and values dependent on both (IPC, average ROB_occupancy, and branch MPKI). Note that there may be ways to predict rough estimates for the last category as a function of statically known values. We hope to gain valuable insights to determine what

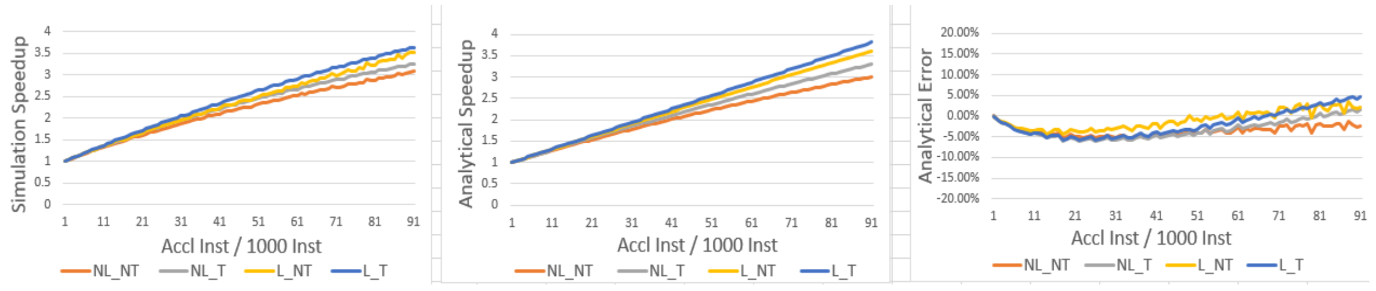

Fig. 3. Analytical Speedup vs Simulations with varying the number of accelerator instructions (increasing invocation freq and % acceleratable code).

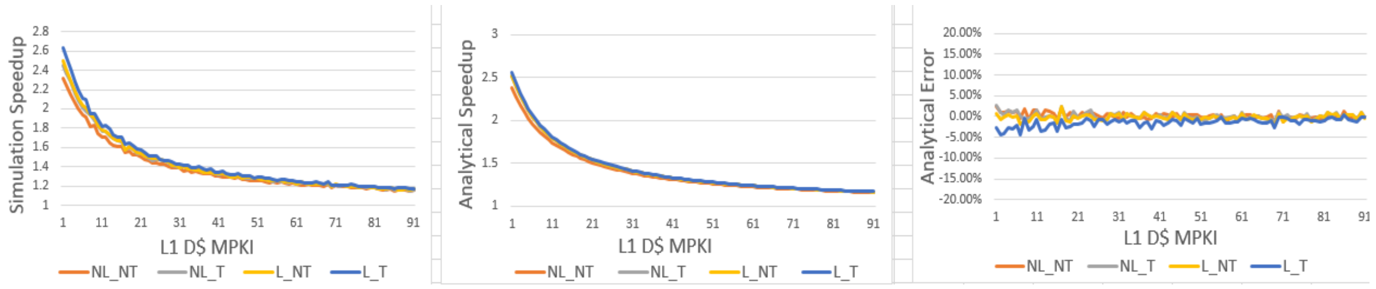

Fig. 4. Analytical Speedup vs Simulations with varying the number of memory L1 D$ misses.

application and architecture pairs are most suitable for fine-grained, tightly-coupled acceleration.

To address (3), we will determine the most suitable architecture for programmable fine-grained acceleration based on our models. We expect the outcome to resemble something like a tightly-coupled configurable FPGA-style fabric to accelerate fine-grained tasks. These accelerators could be reconfigured for the given workload without losing much performance by generalizing the hardware.

Once the FPGA has been integrated into the CPU **once**, all future reconfigurations of the FPGA will not require re-integrating each accelerator into the core.

IV. Preliminary Results and Future Work

We have started validation through our adaptive microbenchmark that varies program and accelerator parameters. We have already run several tests on the gem5 simulator to match our analytical model to simulation results. We have run our microbenchmark with hundreds of different parameter values (see Figures 3&4), and we get our estimated speedups to almost always be within 5% error. We see this as a promising lead that our model can predict general trends, and will show this analysis on the poster.

We are also evaluating large matrix multiplication through accelerated sub-matrix operations of various sizes (for example 2x2 or 4x4 tiles) and hope to have finalized results for the poster. In the future, we would also like to test the heap manager designs in [3] and [4] for the benchmarks they used to show how implementation choice impacts performance in proposed academic accelerator designs.

At the same time, we will start generating a reconfigurable accelerator hardware framework that can be configured for each of the accelerator designs. We will make sure that our framework is flexible for a large variety of tightly-coupled accelerators across different domains. In the future we would also like to create a hardware cost model to do a design-space exploration of performance vs. hardware cost. Much later work could involve designing a physical chip with embedded FPGA fabric on an SOC.

References

[1] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, "Analysis, fast algorithm, and vlsi architecture design for h. 264/avc intra frame coder," *IEEE Transactions on Circuits and systems for Video Technology*, vol. 15, no. 3, pp. 378–401, 2005.

[2] Y.-T. Chen, J. Cong, M. A. Ghodrat, M. Huang, C. Liu, B. Xiao, and Y. Zou, "Accelerator-rich cmps: From concept to real hardware," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, pp. 169–176, IEEE, 2013.

[3] S. Kanev, S. L. Xi, G.-Y. Wei, and D. Brooks, "Mallacc: Accelerating memory allocation," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 33–45, ACM, 2017.

[4] D. Gope, D. J. Schlais, and M. H. Lipasti, "Architectural support for server-side php processing," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 507–520, ACM, 2017.

[5] M. S. B. Altaf and D. A. Wood, "Logca: a performance model for hardware accelerators," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 132–135, 2015.

[6] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," *arXiv preprint arXiv:1704.04760*, 2017.

[7] N. Goulding-Hotta, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, P.-C. Huang, M. Arora, S. Nath, V. Bhatt, J. Babb, *et al.*, "The greendroid mobile application processor: An architecture for silicon's dark future," *IEEE Micro*, vol. 31, no. 2, pp. 86–95, 2011.

[8] G. Shi, M. Li, and M. Lipasti, "Accelerating search and recognition workloads with sse 4.2 string and text processing instructions," in *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*, pp. 145–153, IEEE, 2011.