Evaluating Shared Memory Heterogeneous Systems Using Traverse-compute Workloads

Yanwen Xu
yxu83@ucsc.edu
University of California, Santa Cruz

Ang Li
angl@princeton.edu
Princeton University

Tyler Sorensen
tyler.sorensen@ucsc.edu
University of California, Santa Cruz

1 PROBLEM/MOTIVATION

The end of Moore's law and Dennard's scaling has led to an explosion of specialized *processing units* (PUs); combining different PUs creates a *heterogeneous system*. However, these PUs often interact in a coarse-grained way, especially for accelerator-oriented systems, in which the trend is to offload nearly all computation to a discrete accelerator. However, many systems, especially SoCs found in mobile devices, are designed to be more balanced, featuring nearly equal CPU resources and accelerator resources [9]. In such cases, effectively utilizing both the CPU and the accelerator is essential. Moreover, the communication overhead associated with data transfer via PCIe and kernel launching makes the coarse-grained approaches less feasible for applications that require frequent communication between PUs. Recognizing these challenges and opportunities, academic researchers have increasingly explored novel heterogeneous systems. Numerous novel designs have emerged as open-source projects, promoting widespread access and collaboration. Many of these designs incorporate shared memory across PUs [4], enabling fine-grained interaction and minimizing communication overheads.

Many applications can benefit from fine-grained PU interaction by decomposing the workload into smaller tasks and mapping them according to the architectural strengths of each PU. In this work, we present one such application class, which we call *traverse-compute* [12]. These applications are a special type of tree applications that involve repetitive traversals of tree data structures, combined with performing computations on data located at the leaf node. Traverse-compute applications have widespread implications in scientific computing, statistical learning, and computer graphics.

In summary, this work will present how open-source hardware can be used to accelerate a pragmatic class of applications. Specifically, we show that the Duet [7] system can accelerate a suite of traverse-compute applications by up to 13.5× with a geomean of 6.43×. We will highlight the use of Grove [12]: an open-source benchmark suite of traverse-compute workloads that utilize fine-grained synchronization across PUs, and thus can provide a way for architecture researchers to evaluate their heterogeneous designs.

2 TRAVERSE-COMPUTE APPLICATIONS

This work examines traverse-compute workloads, a type of tree-based algorithm commonly found in domains such as n-body problems and k-nearest neighbor (KNN) applications. These problems are prevalent in scientific computing and statistics, and their naïve implementation involves computing pairwise interactions of points in a large dataset, leading to a runtime of $O(n^2)$.

One approach to accelerate n-body and KNN problems is by using spatial partitioning trees, such as quadtree, octree, and kd-tree [3]. These trees organize items in a dataset based on their spatial coordinates to speed up the comparison process. For example, the Barnes-Hut (BH) algorithm exhibits a complexity of $O(n \log n)$. In these algorithms, the tree is traversed repeatedly, and specific computations are applied to the visited nodes.

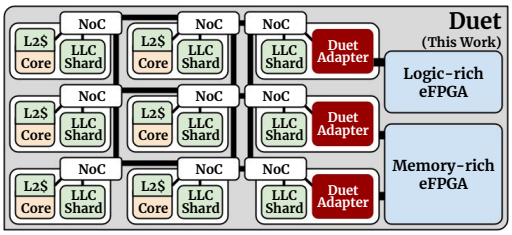

Figure 1: Duet Architecture

A key advantage of spatial partitioning trees is their ability to adjust leaf node sizes, which can be tuned to balance the number of regular computations and irregular memory accesses in the application. Increasing leaf node size leads to more regular computations at leaf nodes and fewer irregular memory accesses from tree traversals, whereas decreasing leaf node size has the opposite effect. This flexibility is particularly useful when offloading computations to accelerators with varying CPU-accelerator relative throughput.

3 DUET ARCHITECTURE

Duet [7] is a novel, tightly-integrated, cache-coherent CPU-FPGA architecture (Fig. 1). Multiple embedded FPGAs (eFPGA) are attached onto the on-chip network of a chip multi-processor through the Duet Adapters. Hardware accelerators emulated with the eFPGAs share the memory system with the processors using the same memory access mechanisms, e.g., cache hierarchy, address translation, and atomic operations. Duet enables fine-grained, transparent data sharing between the processors and the eFPGA-emulated accelerators, simplifies the use of on-chip hardware accelerators, and improves performance through careful design of the memory system and coherence protocol.

Two open-source frameworks have stemmed from the Duet project, namely Dolly [6] and Gem5-Duet [5]. Dolly is a silicon-proven, register-transfer-level (RTL) model (Verilog/SystemVerilog) of Duet based on an array of open-source projects, including but not limited to OpenPiton [1] and PRGA [8]. Gem5-Duet is a Gem5-based [2], cycle-level model of Duet. Both can be used to simulate and evaluate applications run on the Duet architecture. In this work, we use Gem5-Duet because it enables the exploration of a much larger design space and faster simulation in the user space.

4 HETEROGENEOUS DECOMPOSITION

This work will present a recently published framework called Redwood [12], which provides a high-level API and a runtime that

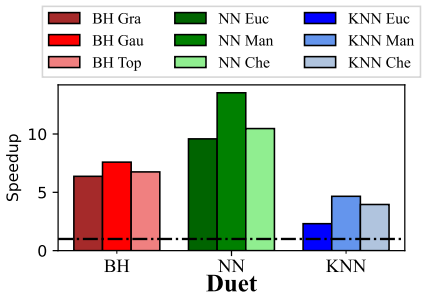

Figure 2: Speedups of the optimal heterogeneous configuration vs. the optimal homogeneous configuration of Grove.

	Leaf Size CPU	Leaf Size w/ Duet	Ratio	Avg Speedup
BH	3.33	512	153.6	6.9x
NN	26.67	426.67	16	11.2x
KNN	26.67	128	4.8	3.64x
Average	19	355	18.8	6.43x

Table 1: Optimal leaf node sizes for Duet implementation and CPU-only implementation for each application class. We report the average across each distance metric.

decomposes traverse-compute applications into traversal and reduction phases. In the traverse phase, CPU threads can access the spatial tree structure, check traversal conditions, and perform traversals. The computation phase, which happens at leaf nodes, can be efficiently computed on the eFPGA in parallel.

Different systems might have different CPU/FPGA relative throughput; we configured Redwood to target Duet, by finding the optimal leaf node size for CPU-only and heterogeneous configurations. For FPGA code, we implemented our computation kernels in using Algorithmic C [10], synthesized them with Catapult HLS [11], and applied the post-HLS timing annotation in the simulator's configuration script. We are working on future work to target a version of the chip that was taped out.

5 RESULTS

We evaluated Duet with Grove [12], a recently published open-source benchmark suite implemented with Redwood, containing nine pragmatic tree traversal applications, including algorithms like BH, nearest neighbor (NN), and KNN. Each algorithm has different computation patterns: accumulation, min reduction, and sorting. Each application has various distance metrics, such as Euclidean, Manhattan, and Chebyshev, or interaction kernels like Gravity, Gaussian, or Top Hat, These applications are used in different domains, ranging from astrophysics to statistical learning.

For the Duet implementation of Redwood, we utilize an optimized sorting network for KNN, and we applied the timing from the Spiral Project: Sorting Network IP Generator [13], which generates customized sorting networks in synthesizable RTL Verilog. To represent realistic offload overhead, the simulation models multi-stage asynchronous FIFOs, and CPU/FPGA clock penalties are also modeled with 1.5GHz/333MHz.

Our baselines are CPU sequential implementations of each application. We swept through the leaf node sizes to find the optimal configuration that yield the best performance, as shown in Table 1. We observe that, on average, the heterogeneous implementation is optimal with a leaf node size of 18× larger than the CPU, thus highlighting the ability for accelerators to have a higher throughput on regular computations. The overall speedups of running the nine applications are shown in Figure 2. Duet achieved a 13.53× highest speedup in the NN and a 6.43× geomean speedup.

The redwood paper evaluated proprietary shared memory SoCs like Nvidia Tegra and Intel SoCs alongside Duet. One key advantage of the Duet over these CPU-GPU systems is that the eFPGA-emulated accelerators can be invoked with only one or very few memory-mapped control register accesses, which reduces the overhead associated with launching kernels as had in these CPU-GPU systems. In addition, the Duet system implements bi-directional cache coherence between the CPUs and the eFPGAs, which means that data can be shared implicitly between the CPUs and eFPGA-emulated accelerators without the need for explicit data transfer operations. This further reduces the overhead associated with data movement and improves overall system performance.

In summary, we present a new open-source heterogeneous hardware design evaluated using a recently published open-source benchmarks suite containing pragmatic applications that utilize fine-grained heterogeneous interactions.

REFERENCES

[1] Jonathan Balkind, Michael McKeown, Yaosheng Fu, Tri Nguyen, Yanqi Zhou, Alexey Lavrov, Mohammad Shahrad, Adi Fuchs, Samuel Payne, Xiaohua Liang, Matthew Matl, and David Wentzlaff. [n. d.]. OpenPiton: An Open Source Manycore Research Framework. In *ASPLOS 2016*.

[2] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. 2011. The gem5 simulator. *ACM SIGARCH* 39, 2 (2011), 1–7.

[3] Nikhil Hegde, Jianqiao Liu, Kirshanthan Sundararajah, and Milind Kulkarni. 2017. Treelogy: A benchmark suite for tree traversals. In *2017 IEEE ISPASS*. IEEE.

[4] Tianyu Jia, Paolo Mantovani, Maico Cassel Dos Santos, Davide Giri, Joseph Zuckerman, Erik Jens Loscalzo, Martin Cochet, Karthik Swaminathan, Gabriele Tombesi, Jeff Jun Zhang, et al. 2022. A 12nm Agile-Designed SoC for Swarm-Based Perception with Heterogeneous IP Blocks, a Reconfigurable Memory Hierarchy, and an 800MHz Multi-Plane NoC. In *ESSCIRC*. IEEE, 269–272.

[5] Ang Li. 2022. Gem5-Duet: A Gem5-based Simulator for Tightly-Integrated CPU-FPGA Systems. https://github.com/angl-dev/gem5-duet.

[6] Ang Li, August Ning, and David Wentzlaff. 2022. Duet-Dolly (OpenPiton x PRGA) Research Platform. https://github.com/PrincetonUniversity/Duet.

[7] Ang Li, August Ning, and David Wentzlaff. 2023. Duet: Creating Harmony between Processors and Embedded FPGAs. In *HPCA*. 745–758. https://doi.org/10.1109/HPCA56546.2023.10070989

[8] Ang Li and David Wentzlaff. 2021. PRGA: An Open-Source FPGA Research and Prototyping Framework. In *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Virtual Event, USA) *(FPGA '21)*. Association for Computing Machinery, New York, NY, USA, 127–137. https://doi.org/10.1145/3431920.3439294

[9] Yakun Sophia Shao, Brandon Reagen, Gu-Yeon Wei, and David Brooks. 2015. The aladdin approach to accelerator design and modeling. *IEEE Micro* 35, 3 (2015).

[10] Siemens. [n. d.]. Algorithmic C. https://github.com/hlslibs/.

[11] Siemens. [n. d.]. Catapult High-Level Synthesis and Verification. https://eda.sw.siemens.com/en-US/ic/catapult-high-level-synthesis/.

[12] Yanwen Xu, Ang Li, and Tyler Sorensen. 2023. Redwood: Flexible and Portable Heterogeneous Tree Traversal Workloads. https://github.com/xuyanwen2012/redwood-rt. In *ISPASS*.

[13] Marcela Zuluaga, Peter Milder, and Markus Püschel. 2016. Streaming Sorting Networks. *ACM Trans. Des. Autom. Electron. Syst.* 21, 4, Article 55 (may 2016), 30 pages. https://doi.org/10.1145/2854150