Research Statement

Shashank Obla
(412) 708-7719
sobla@andrew.cmu.edu

Overview

We have entered an era where general-purpose compute is no longer sufficient to meet the demands of datacenter-scale applications. While custom accelerators are the clear path forward, their traditionally rigid nature makes high-performance economically infeasible for all but the largest applications with static, predictable workloads. *Democratizing access to next-generation performance requires harnessing the programmability of existing reconfigurable computing platforms and developing novel heterogeneous architectures that balance flexibility with efficiency. Towards this goal, my primary research interests lie in designing sustainable and programmable accelerators for datacenter-scale computing, alongside the abstractions necessary to model and optimize applications running on them.*

Current Research

My current research investigates the use of reconfigurable hardware, especially FPGAs, *to deliver adaptable, custom solutions using highly parameterizable hardware accelerators that can be rapidly specialized for various workloads and domains*. Taking inspiration from software specialization and system modeling frameworks such as queueing theory, I have developed performance abstractions and architectural techniques that utilize FPGA programmability for rapid tuning and optimization.

Tuning Input-Dependent Streaming Pipelines

FPGAs offer the programmability necessary to tune designs for specific use-cases and deployments, which is critical for streaming pipelines with data-dependent dataflows, such as deep-packet inspection or database analytics pipelines. *Unlike regular workloads like matrix multiplication, the optimal configuration for input-dependent applications varies not just across resource constraints or performance targets but also with the data being processed.* Since static analysis cannot accurately predict the performance of these pipelines, runtime or simulation-based approaches are used to drive design-space exploration (DSE) and optimization. However, these methods fail to scale for large, real-world designs with long workloads.

To address this challenge, we proposed RapidQ, a performance model that allows designers to capture the dataflow of their streaming pipeline as a queueing system, enabling fast case-by-case re-tuning for resource efficiency. Unburdened by the functional details of the design, our trace model drives a fast queueing simulation that can predict the performance of the pipeline without repeating full functional simulations. Our simulator is over 7x faster than the state-of-the-art while maintaining sufficient accuracy for rapid exploration. The RapidQ tuning flow yields up to 42% resource savings for real-world workloads compared to tuning only for buffer sizes or module throughputs. *By streamlining the modeling and rapid customization of streaming pipelines, RapidQ significantly lowers the barrier for customized deployments of input-dependent applications on FPGAs.*

Common-Case Optimized Reconfigurable Accelerators

Traditional static approaches to accelerator design severely limit tuning and optimization workflows, making custom accelerators infeasible for the fluctuating demands on modern datacenter applications. Existing solutions either fail when an application's performance or functional requirements exceed their original design, or are fundamentally inefficient during normal operation because they are over-provisioned for the worst-case.

We proposed a heterogeneous approach utilizing hardware kernels specialized for high performance on common-

case inputs, backed by software to provide necessary coverage, prevent failures, and handle momentary load bursts. Leveraging reconfigurable hardware, the accelerator design can adapt to changes in the common-case by exposing the design space using highly parameterizable High-Level Synthesis (HLS) based implementations. *We showcased our approach on two real-world applications: Log Monitoring and Database Group-By Aggregation.* For log monitoring, inspired by the existing Pigasus Network Intrusion Prevention System, we built a string-matching pipeline in HLS and customized it to achieve processing rates over 200Gbps on a single FPGA + CPU server for the new domain's constraints. Parameterizations allow the design to sustain throughput across varying rules and log compositions. For database analytics, our approach enables previously infeasible aggregation acceleration to scale to full database workloads. By specializing near-storage FPGAs to handle high-frequency keys, we reduced the network bandwidth needed between storage and a user VM by orders of magnitude, while software provides elastic capacity as needed. Additionally, the FPGA design implemented in HLS allows customization to specific database parameters, improving efficiency over a wide class of database queries.

This common-case optimized design pattern enables the use of previously underutilized compute accelerators positioned along the communication channel, including SmartNICs and FPGAs. By offloading processing from CPUs and fundamentally reducing the net volume of data communicated, this approach can alleviate communication as a major source of inefficiency in the datacenter.

Future Research Directions

Building upon my foundation of harnessing FPGAs' under-tapped programmability to design and tune common-case specialized reconfigurable accelerators, my future research goals focus on expanding to novel applications of Partial Reconfiguration (PR) on FPGAs and heterogeneous Systems-on-Chip (SoCs) that place reconfigurable components alongside traditional compute devices and ASIC accelerators. I am particularly excited to explore the following research questions:

- **Novel Uses for FPGA Reconfiguration:** Current research often views the 100+ms FPGA reconfiguration time as a limitation, expecting new architectural interventions to make software-like program-switching feasible. However, these solutions incur significant resource costs since the program state for an FPGA (the bitstream) is orders of magnitude larger than standard CPU or GPU binaries. *How can we exploit existing FPGA reconfigurability to enable multi-tenant, shareable FPGA solutions ranging from virtualization infrastructure to workload auto-scaling?* Answering this requires re-evaluating conventional wisdom by tailoring solutions to the underlying FPGA micro-architecture rather than in spite of it.

  For instance, consider the role of a communication network in a multi-tile shared FPGA. Unlike multi-core CPUs, where threads or processes might use complex collective communication patterns or shared memory to move data, hardware designs employ much more deliberate data movement patterns (such as streaming) to achieve the desired efficiency gains. *By reflecting this use case and leveraging the circuit-switched nature of programmable routing in an FPGA, a custom network architecture can achieve greater resource efficiency and provide bandwidth guarantees far better than soft, packet-switched approaches.*

- **Abstractions for Heterogeneous SoC Design:** The architectures of reconfigurable, general-purpose, and custom-compute accelerators have been converging over the past decade. Processors are incorporating specialized hardened units, while communication-focused ASICs (like Intel IPUs and Nvidia DPUs) are integrating general-purpose compute to improve coverage and future-proof the SoC. *How do we reason about and design such heterogeneous systems for performance, resource and energy efficiency, as well as sustainability?*

  Currently, heterogeneous systems dedicate specific components to distinct applications or varying phases or functionality within a single application. However, the tradeoff spectrum between highly efficient hardened accelerators and flexible software-programmable compute remains largely unexplored for the same functionality

with varying workload demands. Much like a power grid balances varying loads using baseload sources (analogous to ASICs) and flexible peaking plants (analogous to processors), *we can develop abstractions to model the interaction between application demands and the programmability and processing power of heterogeneous components, achieving a system more efficient than relying on a single kind of compute.* Beyond architectural exploration, addressing the broader question will require us to establish application benchmarks, develop abstractions that effectively map applications to various hardware structures, and design new programming models for dynamic data and compute orchestration.

- **Smart Communication Fabrics:** As application data and compute footprints constantly rise, they demand scale-out solutions across multiple chips and even entire server racks. While custom interconnects or specific solutions like NVLink offer immense chip-to-chip bandwidth, and collective communication techniques optimize data movement, the additional compute required to prepare the data for communication (including filtering, reduction or repackaging) and to coordinate data movement can introduce significant, non-deterministic overhead. *Can we design smarter communication fabrics capable of efficient, on-the-fly compute that allow for highly adaptable, extremely low-latency, and high-throughput communication?*

  Many applications, from database analytics to broader domains such as scientific computing, can benefit from computing over "data-on-the-move". Doing so not only reduces bandwidth requirements but also offloads operations with low arithmetic intensity to spatial architectures better suited to handle them. Analyzing the common operations within these domains can guide the design of custom reconfigurable accelerators, or even chiplet-style ASIC integrations, that alleviate pressure on communication infrastructure and free up expensive CPU cycles for core application logic. *Developing methodologies to identify these compute pushdown opportunities and accurately predicting their impact on end-to-end application performance will be instrumental in showcasing the feasibility of smart communication fabrics.*