FPGA-Based Edge-Computing Acceleration

by

Saman Biookaghazadeh

A Dissertation Presented in Partial Fulfillment
of the Requirement for the Degree
Doctor of Philosophy

PREVIEW

Approved July 2021 by the
Graduate Supervisory Committee:

Ming Zhao, Co-Chair
Fengbo Ren, Co-Chair
Baoxin Li
Jae-Sun Seo

ARIZONA STATE UNIVERSITY

August 2021

ABSTRACT

The rapid growth of Internet-of-things (IoT) and artificial intelligence applications have called forth a new computing paradigm–edge computing. Edge computing applications, such as video surveillance, autonomous driving, and augmented reality, are highly computationally intensive and require real-time processing. Current edge systems are typically based on commodity general-purpose hardware such as Central Processing Units (CPUs) and Graphical Processing Units (GPUs) , which are mainly designed for large, non-time-sensitive jobs in the cloud and do not match the needs of the edge workloads. Also, these systems are usually power hungry and are not suitable for resource-constrained edge deployments. Such application-hardware mismatch calls forth a new computing backbone to support the high-bandwidth, low-latency, and energy-efficient requirements. Also, the new system should be able to support a variety of edge applications with different characteristics.

This thesis addresses the above challenges by studying the use of Field Programmable Gate Array (FPGA) -based computing systems for accelerating the edge workloads, from three critical angles. First, it investigates the feasibility of FPGAs for edge computing, in comparison to conventional CPUs and GPUs. Second, it studies the acceleration of common algorithmic characteristics, identified as loop patterns, using FPGAs, and develops a benchmark tool for analyzing the performance of these patterns on different accelerators. Third, it designs a new edge computing platform using multiple clustered FPGAs to provide high-bandwidth and low-latency acceleration of convolutional neural networks (CNNs) widely used in edge applications. Finally, it studies the acceleration of the emerging neural networks, randomly-wired neural networks, on the multi-FPGA platform.

The experimental results from this work show that the new generation of workloads requires rethinking the current edge-computing architecture. First, through the

PREVIEW

acceleration of common loops, it demonstrates that FPGAs can outperform GPUs in specific loops types up to 14 times. Second, it shows the linear scalability of multi-FPGA platforms in accelerating neural networks. Third, it demonstrates the superiority of the new scheduler to optimally place randomly-wired neural networks on multi-FPGA platforms with 81.1 times better throughput than the available scheduling mechanisms.

PREVIEW

This dissertation is dedicated to:

my parents, for they gave me everything unconditionally

my sister and step-brother, who has supported me throughout my long journey

and my true friends, whom always been there, when I needed them.

PREVIEW

ACKNOWLEDGEMENTS

I want to express my most profound appreciation to Dr. Ming Zhao for the guidance and help he generously provided me throughout these past few years. I am very thankful for the freedom I was given to explore various research directions and find what interests me most. From the long list of qualities I hope to have acquired from him, above all was his integrity, ethics, deep thinking, and unconditional support.

I want to thank Dr. Fenbo Ren for all his time and efforts throughout my research, especially helping me to understand hardware systems better. He generously assisted me with all the equipment and helped me learn the necessary knowledge. Without his support, I wouldn't have been able to start my dissertation smoothly.

I would also like to thank my committee members: Dr. Baoxin Li and Dr. Jae-Sun Seo, for their open-mindness, insightful bits of advice, and constructive feedback, which assisted me in the completion of this thesis. In addition, I give thanks to the exceptional one-on-one meetings with them, which improved the quality of my research leading to this point.

Furthermore, I would like to acknowledge Dulcardo Artega, Yiqi Xu, Wenji Li, Jorge Cabrera, Gregory Jean-Baptise, Douglas Otstott, Yitao Chen, Qirui Yang, Runyu Jin, and everyone from the VISA Lab. They all helped me learn about research and shape my ideas in the best fashion throughout my journey. I wish all these individuals the best of luck, and I hope our paths cross each other again soon.

Next, I must specially thank Dr. Raju Rangaswami. Throughout my time at Florida International University, Dr. Rangaswami allowed me to learn how academic ideas can transform into technology products. He taught me a lot about advanced storage systems topics throughout this journey and helped me with many new things.

Also, I must thank all my close friends who made my Ph.D. experience much more enjoyable. One great benefit of my Ph.D. experience was finding new friends from

PREVIEW

whom I learned a lot. I wish all of them the best in their future endeavors and thank their indirect but significant support for my work.

Last but not least, I would not have been able to complete my dissertation without the help of many others here at ASU, some of whom are unknown to me. That is why I would like to extend my appreciation to the administrative staff, especially Monica Dugan, Pamela Dunn, Christina Sebring, and Jaya Krishnamurthy.

Thank you all!

PREVIEW

TABLE OF CONTENTS

PREVIEW

Page

PREVIEW

Page

PREVIEW

LIST OF TABLES

Table Page

PREVIEW

PREVIEW

LIST OF FIGURES

PREVIEW

Figure Page

PREVIEW

Figure Page

PREVIEW

PREVIEW

PREVIEW

Chapter 1

INTRODUCTION

The Internet-of-Things (IoT) will connect 50 billion devices and is expected to generate 400 Zetta Bytes of data per year by 2020. Even considering the fast-growing size of the cloud infrastructure, the cloud is projected to fall short by two orders of magnitude to either transfer, store, or process such vast amount of streaming data Fowers *et al.* (2012). Consequently, the consensus in the industry is to expand our computational infrastructure from data centers towards the edge. Existing edge servers on the market are simply a miniature version of cloud servers (cloudlet) which are primarily structured based on CPUs with tightly coupled co-processors (e.g., GPUs) HPE (2019b,a); Cisco (2019). However, CPUs and GPUs are optimized towards batch processing of in-memory data and can hardly provide consistent nor predictable performance for processing streaming data coming dynamically from I/O channels. Therefore, future edge servers call for a new general-purpose computing system stack tailored for processing streaming data from various I/O channels at low power consumption and high energy efficiency.

FPGAs are a great candidate to address the edge-computing challenges by harnessing the programmability and the ability to handle streaming data. FPGAs can be deployed alongside the conventional accelerators and enable a new generation of heterogeneity for accelerating the different type of IoT application. With the emergence of FPGAs, the benefits of heterogeneous systems become more significant as the FPGAs can handle a specific class of application, in both the cloud and the edge.

PREVIEW

While FPGAs are a great candidate for the next generation of the edge systems, there are several challenges the need to be addressed to make effective use of FPGA accelerators in the edge systems:

1.1 FPGA in the Edge

Over the next decade, a vast number of edge servers will be deployed to the proximity of IoT devices; a paradigm that is now referred to as fog/edge computing.

There are fundamental differences between traditional cloud and the emerging edge infrastructure. The cloud infrastructure is mainly designed for (1) fulfilling time-insensitive applications in a centralized environment; (2) serving interactive requests from end users; and (3) processing batches of static data loaded from memory/storage systems. Differently, the emerging edge infrastructure has distinct characteristics, as it keeps the promise for (1) servicing time-sensitive applications in a geographically distributed fashion; (2) mainly serving requests from IoT devices, and (3) processing streams of data from various input/output (I/O) channels. Existing IoT workloads often arrive with considerable variance in data size and require extensive computation, such as in the applications of artificial intelligence, machine learning, and natural language processing. Also, the service requests from IoT devices are usually latency-sensitive. Therefore, having a predictable performance to various workload sizes is critical for edge servers.

Existing edge servers on the market are simply a miniature version of cloud servers (cloudlet) which are primarily structured based on CPUs with tightly coupled co-processors (e.g., GPUs). However, CPUs and GPUs are optimized towards batch processing of in-memory data and can hardly provide consistent nor predictable performance for processing streaming data coming dynamically from I/O channels. Furthermore, CPUs and GPUs are power hungry and have limited energy efficiency [4],

PREVIEW

creating enormous difficulties for deploying them in energy- or thermal-constrained application scenarios. Therefore, future edge servers call for a new general-purpose computing system stack tailored for processing streaming data from various I/O channels at low power consumption and high energy efficiency.

OpenCL-based field-programmable gate array (FPGA) computing is a promising technology for addressing the aforementioned challenges. FPGAs are highly energy-efficient and adaptive to a variety of workloads. Additionally, the prevalence of high-level synthesis (HLS) has made them more accessible to existing computing infrastructures.

1.2 Loop Acceleration in Heterogeneous Systems

Many applications can benefit from computing on hardware accelerators, ranging from cloud computing to big-data and edge computing. Examples of these applications include (1) analysis of large quantity of data on big-data platforms, (2) training and running artificial intelligence (AI) and machine learning models in the cloud, (3) processing streams of requests and data from IoT devices, and (4) modeling and simulating the behaviors of scientific applications.

PREVIEW

By using accelerators, applications can achieve higher throughput Owens *et al.* (2008), lower response time Biookaghazadeh *et al.* (2018), and/or lower energy consumption Fowers *et al.* (2012).

A variety of accelerators are readily available for applications to choose for their computation needs in the cloud. Graphics Processing Units (GPUs) are the most widely used and can be easily found in many HPC and cloud systems. Other types of accelerators are also becoming increasingly available, e.g., Tensor Processing Units (TPUs) on the Google cloud and Field-Programmable Gate Arrays (FPGAs) on the Amazon cloud (F1 nodes). These accelerators come with different capabilities and limitations. For example, FPGAs can be reconfigured to run any applications but can provide only low clock frequency; GPUs can be programmed using high-level languages to accelerate highly parallel applications; and TPUs are specifically designed for deep learning workloads. Although a general understanding of different accelerators is available, choosing the right accelerators for applications in a heterogeneous computing system is still a difficult problem.

Several related works have studied the performance of common algorithms on accelerators. For example, Rodinia benchmark and its follow-up work Zohouri *et al.* (2016) are designed to benchmark heterogeneous platforms including CPUs, GPU, and FPGAs. These benchmarks usually provide insights on a macro level, for a complete algorithm on a hardware platform. However, they lack a thorough analysis of micro-level execution patterns that exist in different applications and the effectiveness of different hardware architectures in handling these patterns.

To address the above challenges, we study how the accelerators with different hardware architectures can accelerate different types of loops, which are the basic building blocks of almost every computationally intensive application. These applications typically consist of one or many nested and flattened loops. These loops can

PREVIEW

embody different patterns in terms of types and degrees of dependency and concurrency, and they can be found in many applications. For example, dynamic programming algorithms consist of one or more nested loops, where every iteration depends on another iteration that points diagonally in the iteration space. Therefore, abstracting the common loop patterns from applications and understanding how they perform on various hardware accelerators are essential steps towards optimally utilizing the accelerators for executing different applications. Although there is a great body of existing works on loop optimizations Wang *et al.* (2021); Juega *et al.* (2014); Konstantinidis *et al.* (2013); Baghdadi *et al.* (2019); Simbürger *et al.* (2013); Trifunovic *et al.* (2010); Grosser *et al.* (2012, 2011); Bastoul (2004); Loechner (1999); Ancourt and Irigoin (1991); Schreiber *et al.* (1990); Cousot and Halbwachs (1978); Lamport (1974), they cannot provide cross-accelerator comparisons that can help developers choose the right platform for their applications in a heterogeneous computing system.

To support the study of loop accelerations across different platforms, we developed *Loopy*, a collection of five fine-grained loop patterns that commonly exist in real-world applications such as linear algebra, optimization, and data analytics algorithms. Loopy parameterizes the key aspects of these loop patterns, including the type and degree of dependencies, data bit-precision, operational intensity, and size of the iteration spaces. It allows them to be flexibly tuned to model diverse loop characteristics. Loopy provides optimized OpenCL implementations of these loop patterns for both GPU and FPGA, the two most versatile and available accelerators. We focus on OpenCL because it is an important framework for the emerging heterogeneous computing paradigm.

Based on Loopy, we evaluated the performance of important loop patterns on several typical accelerators, including Intel A10 FPGAs and Nvidia T4 and RTX2080 GPUs. Our study made several key findings. First, for three out of five loop de-

PREVIEW

pendency patterns (intra-dimension dependency, conditional dependency, and half-parallelism half-dependency), FPGA has the potential to outperform GPU. For example, for the intra-dimension dependency pattern, the evaluated FPGA outperforms GPU by 17.5x. Second, for various computational intensities, FPGA can maintain an identical performance, whereas GPU performance is highly variable. For example, having eight conditional statements can degrade the GPU performance by up to 45%. Third, increasing the input data size can increase the performance difference between these two accelerators. For example, for the diagonal dependency loop pattern, the performance gap increases by 51%, while changing the input data size from 4MB to 256MB.

1.3 Multi-FPGA Acceleration of AI on the Edge

In recent years, FPGAs have received tremendous attention in the world of neural network acceleration. FPGAs can provide unique benefits to accelerate Convolutional Neural Networks (CNNs). First, FPGAs can guarantee tight latency bounds for incoming requests. Conventional CNN accelerators, i.e., GPUs, have shown the ability for the acceleration of a batch of requests, by leveraging their farm of processing cores. Unfortunately, they lack the potential to guarantee low-latency services for individual requests Zhang *et al.* (2016, 2018). In contrast to GPUs, FPGAs can leverage their reconfigurable deep pipeline to service the requests in a streaming fashion and provide a predictable low latency. Second, conventional processors are usually power-hungry, which makes them challenging to deploy in power- or energy-constrained environments. Differently, FPGAs are highly power-efficient due to their low operational clock frequency. In conclusion, FPGAs are considered as an excellent platform for accelerating CNNs for deployment.

PREVIEW

The ever-increasing complexity of emerging CNNs requires FPGAs with a higher amount of resources, such as memory bandwidth and logical units, to achieve low-latency and high-throughput inferences. Even high-end FPGA chip technologies can host only a small section of a whole CNN model. For example, the Intel Stratix 10 FPGA can perform only 5000 multiply-accumulation (MAC) operations per clock cycle, which is even less than the total number of operations for a single layer of a typical CNN, such as VGG-16 or ResNet. As a result, they fall short in handling heavier CNNs for ultra-low latency (less than ten milliseconds), and high-throughput (more than 60 images/frames per second). Such a problem is even more significant for accelerating more computationally intensive operations, for example, three-dimensional (3D) convolutions, which show great potentials in video processing applications. This challenge can be potentially addressed by utilizing a cluster of FPGAs, connected through a high-bandwidth communication infrastructure.

Achieving linear speedup using a multi-FPGA solution is not straightforward. First, we need to have an efficient design on a single FPGA and achieve state-of-the-art performance. Such performance benefits should be reflected in the acceleration of various CNN operations. Second, the pipeline of multiple FPGAs should be correctly managed to ensure that all FPGAs are doing useful works to handle incoming requests. Third, CNN partitioning, which is the process of mapping different parts of the model onto different FPGAs, should be done intelligently to make sure the workload is balanced across the FPGAs.

Related works Zhang *et al.* (2016); Jiang *et al.* (2019) have studied the multi-FPGA acceleration of neural networks. These works come with several limitations. First, they do not provide a general architecture to accelerate various types of CNNs. For example, they are only able to accelerate either two-dimensional (2D) or 3D convolutions, but not both. Second, they do not optimally exploit the FPGA acceler-

PREVIEW

ation resources, which leads to sub-optimal performance, compared to the maximum theoretical performance of an FPGA. Third, they are designed and developed, using low-level hardware programming languages (Jiang et al. Jiang *et al.* (2019) used Xilinx HLS), which makes it difficult to extend and support by the widely-used deep learning frameworks, such as Tensorflow Abadi *et al.* (2016) and Caffe Jia *et al.* (2014).

In this thesis, we present a novel multi-FPGA CNN accelerator that can leverage a deep pipeline of FPGAs, connected through a high-performance I/O channel. First, we adopted the Intel Deep Learning Accelerator (DLA) Aydonat *et al.* (2017) architecture and applied various optimizations to achieve an efficient design on a single FPGA. Using a novel systolic array design, our architecture has reduced the total resource consumption of the DLA by up to 25% and increased the overall performance by 24%. We developed this design using OpenCL, which enables convenient integration with widely-used deep learning frameworks. Also, it enables the integration of the accelerator in a heterogeneous environment, where the same OpenCL code can run across different processors. Second, we extended the design to support data communication with other FPGAs in the pipeline, using a 40Gb/s QSFP+ I/O channel. Using a network of connected FPGAs enables temporal (distributing the layers onto different FPGAs) and spatial (splitting a single layer and mapping it onto multiple FPGAs) parallelization of the layers. Using this configuration, a user can allocate a set of FPGAs in a network, with no prior information about the network architecture. The user can interact with these FPGAs as a single FPGA with a large number of resources. Further, she/he can select a neural network model and deploy it on these FPGAs. The framework can automatically split the model into several sub-models, and deploy each sub-model onto an FPGA. This cluster of FPGAs can provide the same or better latency and energy-efficiency, compared to the available CPU or GPU solutions. Third, we extended the design to support 3D convolutions, on top of 2D

PREVIEW

Reproduced with permission of copyright owner. Further reproduction prohibited without permission.