Accelerating Conv2D and DepthwiseConv2D Tensor Operations for TensorFlow Lite on Embedded FPGA with Hybrid Custom Floating-Point Approximation

1st Yarib Nevarez
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

2nd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

3rd Given Name Surname
dept. name of organization (of Aff.)
name of organization (of Aff.)
City, Country
email address or ORCID

Abstract—**Convolutional neural networks (CNNs) have become ubiquitous in the field of image processing, computer vision, and artificial intelligence (AI). Given the high computational demands of CNNs, dedicated hardware accelerators have been implemented to improve compute efficiency in FPGAs and ASICs. However, most commercial general-purpose deep learning processing units (DPUs) struggle with support for low-power, resource-limited devices. In this publication, we present a dedicated hardware accelerator for TensorFlow (TF) Lite on embedded FPGA emulating an Edge TPU coprocessor to delegate Conv2D and DepthwiseConv2D tensor operations. The hardware design is implemented with high-level synthesis (HLS). This accelerator incorporates the support for TF Lite quantization for fixed-point and floating-point. The proposed optimization decomposes floating-point calculation for the dot-product. This approach accelerates computation, reduces energy consumption and resource utilization. To demonstrate the potential of the proposed accelerator, we address a design exploration with custom-built CNNs covering fixed-point quantization, floating-point single precision, half-precision, brain floating-point, TensorFloat, and custom reduced formats for approximate processing, including logarithmic computation. A single accelerator running at 150 MHz on a Xilinx Zynq-7020 achieves 45X runtime acceleration on Conv2D tensor operation compared with ARM Cortex-A9 at 666MHz, and 5X compared with the equivalent standard floating-point implementation in HLS with Xilinx LogiCORE IP. This accelerator yields a peak performance of 1.6 TFLOPS/watt and 152 MFLOP/s. The entire hardware design and the implemented TF Lite software extensions are available as an open-source project.**

Index Terms—**Artificial intelligence, convolutional neural networks, depthwise separable convolution, hardware accelerator, TensorFlow Lite, embedded systems, FPGA, custom floating-point, logarithmic computation, approximate computing**

I. INTRODUCTION

THE constant research and the rapid evolution of artificial neural network (ANN) architectures are driving the transition to smarter and more powerful AI applications, where CNN-based models represent the essential building blocks of deep learning algorithms in computer vision tasks [1]. Applications such as smart surveillance, medical imaging, natural language processing, robotics, and autonomous navigation have been powered by CNN-based models in industry and academia [2]. Nonetheless, dedicated hardware is often a requirement to accelerate execution due to the high computational demands
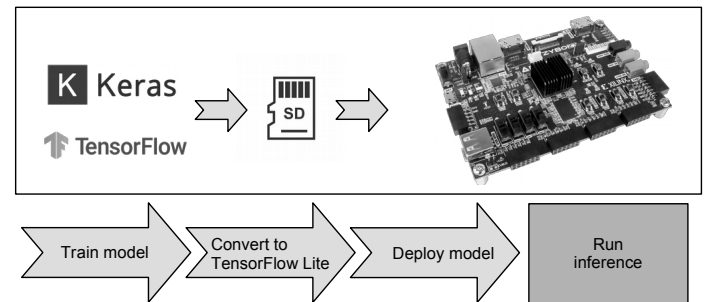

Fig. 1. Deployment workflow.

of CNNs. In terms of pure computational throughput, graphics processing units (GPUs) offer the best performance [3]. In terms of power consumption, FPGA solutions are well known to be more energy efficient (vs GPUs). As a result, numerous FPGA accelerators have been proposed, targeting both high performance computing (HPC) for data-centers and embedded systems applications [4]. However, most commercial deep learning processing units (DPUs) are not designed for low-power, resource-limited embedded FPGAs.

In this paper, we present a hardware/software co-design framework to accelerate tensor operators for TF Lite Micro on resource-constrained and low-power embedded FPGAs. Based on their high computational cost, in this paper, we accelerate Conv2D and DepthwiseConv2D operators. The proposed architecture supports TF Lite quantized models with int8 and standard float32. To enhance resource utilization and energy consumption, the tensor operators are designed as separate hardware engines, where they are optionally instantiated in the FPGA fabric. To accelerate floating-point computation, we propose a decomposed floating-point calculation for the vector dot-product. To enhance on-chip memory usage and for quality reconfigurability, the proposed design employs hybrid custom floating-point formats, where filter and bias tensors are quantized and stored with a reduced format on the hardware accelerator. The hardware design is implemented with high-level synthesis (HLS).
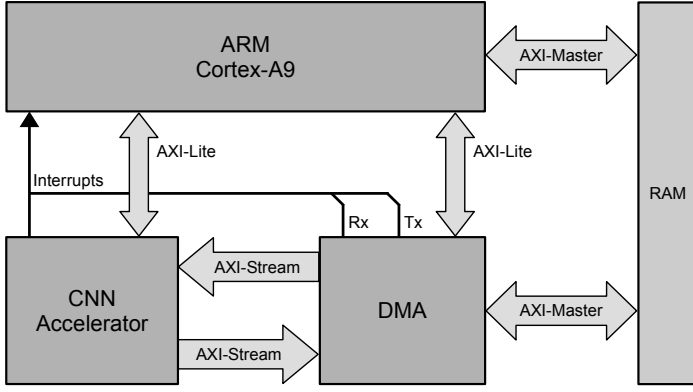
Fig. 2. System-level architecture of the proposed embedded platform.

To operate the proposed accelerator framework, we train a CNN model on TensorFlow or Keras, then this is converted into a TensorFlow Lite model with either float32 or int8 quantization, then the model is stored in a micro SD card along with the software and the FPGA configuration bitstream for deployment. See **Fig.** 1.

To demonstrate the potential of the proposed framework, we address a design exploration with two custom-built CNN architectures for CIFAR-10 image classification. We demonstrate the performance of the Conv2D and DepthwiseConv2D operator engines with fixed-pint quantization, floating-point with Xilinx LogiCORE IP, decomposed floating-point with single precision, half-precision, brain floating-point, TensorFloat, and custom reduced formats for approximate processing, including logarithmic computation. We present a detailed runtime computation schedule and an exhaustive accuracy performance using custom reduced floating-point formats.

A single accelerator running at 150 MHz on a Xilinx Zynq-7020 achieves 45X runtime acceleration on Conv2D tensor operation compared with ARM Cortex-A9 at 666MHz, and 5X compared with the equivalent standard floating-point implementation in HLS with Xilinx LogiCORE IP. This accelerator yields a peak performance of 1.6 TFLOPS/watt and 152 MFLOP/s. The entire hardware design and the implemented TF Lite software extensions are available as an open-source project.

II. Related work

III. Background

IV. System Design

V. Experimental results
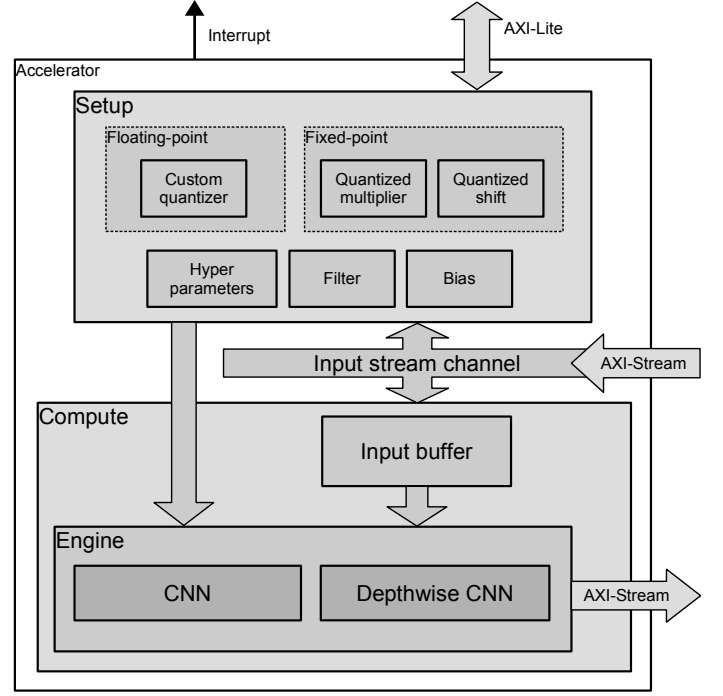
VI. Conclusions

Acknowledgments

Fig. 3. Hardware architecture of the proposed accelerator.

References

[1] M. Hassaballah and A. I. Awad, *Deep learning in computer vision: principles and applications*. CRC Press, 2020.

[2] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," *Progress in Artificial Intelligence*, vol. 9, no. 2, pp. 85–112, 2020.

[3] E. Nurvitadhi, G. Venkatesh, J. Sim, D. Marr, R. Huang, J. Ong Gee Hock, Y. T. Liew, K. Srivatsan, D. Moss, S. Subhaschandra *et al.*, "Can fpgas beat gpus in accelerating next-generation deep neural networks?" in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 5–14.

[4] K. Abdelouahab, M. Pelcat, J. Serot, and F. Berry, "Accelerating cnn inference on fpgas: A survey," *arXiv preprint arXiv:1806.01683*, 2018.
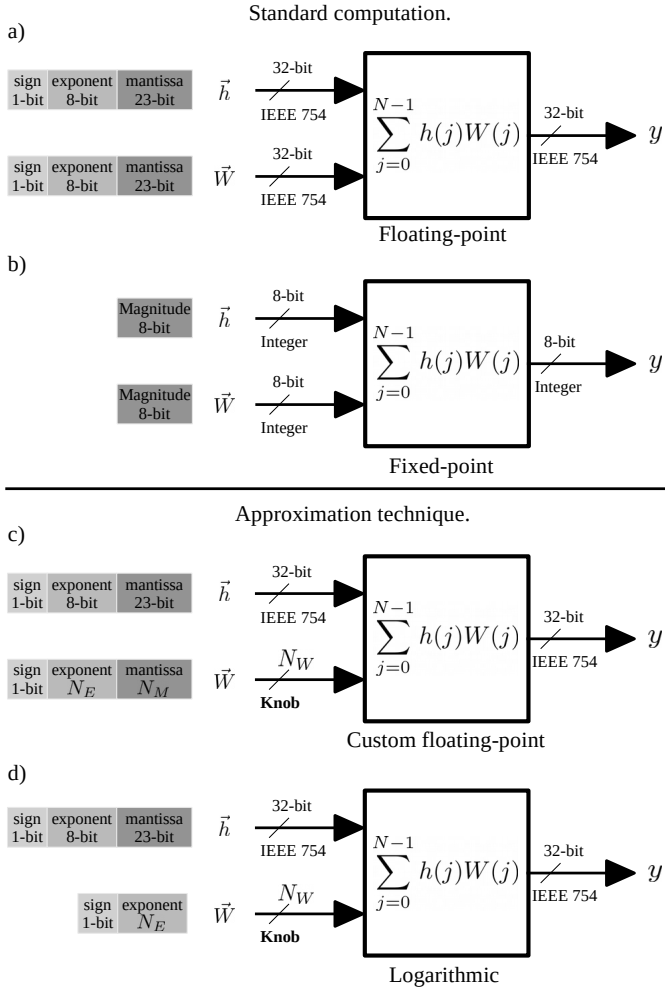
a)

Standard computation.

| sign 1-bit | exponent 8-bit | mantissa 23-bit | $\vec{h}$ |

$\frac{32\text{-bit}}{\text{IEEE 754}}$

$\sum_{j=0}^{N-1} h(j)W(j)$

$\frac{32\text{-bit}}{\text{IEEE 754}}$ y

| sign 1-bit | exponent 8-bit | mantissa 23-bit | $\vec{W}$ |

$\frac{32\text{-bit}}{\text{IEEE 754}}$

Floating-point

b)

| Magnitude 8-bit | $\vec{h}$ |

$\frac{8\text{-bit}}{\text{Integer}}$

$\sum_{j=0}^{N-1} h(j)W(j)$

$\frac{8\text{-bit}}{\text{Integer}}$ y

| Magnitude 8-bit | $\vec{W}$ |

$\frac{8\text{-bit}}{\text{Integer}}$

Fixed-point

Approximation technique.

c)

| sign 1-bit | exponent 8-bit | mantissa 23-bit | $\vec{h}$ |

$\frac{32\text{-bit}}{\text{IEEE 754}}$

$\sum_{j=0}^{N-1} h(j)W(j)$

$\frac{32\text{-bit}}{\text{IEEE 754}}$ y

| sign 1-bit | exponent N_E | mantissa N_M | $\vec{W}$ |

$\frac{N_W}{\textbf{Knob}}$

Custom floating-point

d)

| sign 1-bit | exponent 8-bit | mantissa 23-bit | $\vec{h}$ |

$\frac{32\text{-bit}}{\text{IEEE 754}}$

$\sum_{j=0}^{N-1} h(j)W(j)$

$\frac{32\text{-bit}}{\text{IEEE 754}}$ y

| sign 1-bit | exponent N_E | $\vec{W}$ |

$\frac{N_W}{\textbf{Knob}}$

Logarithmic

Fig. 4. Proposed hardware modules for vector dot-product.

Application

Machine learning library

TensorFlow Lite core

TensorFlow Lite delegate

Hardware abstraction layer

Accelerator driver

Board support package

Hardware platform (Zynq-7000 SoC)

Programmable Logic (PL)
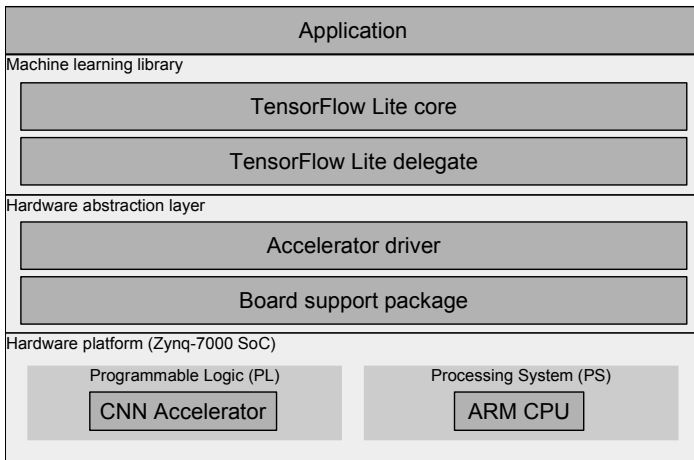
CNN Accelerator

Processing System (PS)

ARM CPU

Fig. 5. System-level overview of the embedded software architecture.