Hardware/Software Co-Design of Edge DNN Accelerators with TFLite

Jude Haris[*], Perry Gibson[*], José Cano[*],
Nicolas Bohm Agostini[†], David Kaeli[†]

[*] *University of Glasgow, United Kingdom*
[†] *Northeastern University, United States*

ABSTRACT

In this work we discuss SECDA-TFLite, a open-source toolkit for developing DNN hardware accelerators, integrated within the TFLite DNN inference framework. The toolkit leverages the principles of SECDA, a hardware/software co-design methodology which reduces the design time of optimized DNN inference accelerators on edge devices with FPGAs. Utilizing SECDA-TFLite, we further reduce the initial setup costs associated with integrating a new accelerator design within a target DNN framework, allowing developers to focus on the design. SECDA-TFLite also includes modules for cost-effective SystemC simulation, profiling, and AXI-based data communication. Additionally, we briefly cover our case study, where we use SECDA-TFLite to efficiently develop three different DNN accelerator designs on a PYNQ-Z1 board. We evaluate the three accelerator designs across five common CNN models and two BERT-based models, achieving an average performance speedup across models of up to 2.9$\times$ for the CNN models and an average speedup of up to 2.5$\times$ for the BERT-based models.

KEYWORDS: DNN accelerator design; Design methodology; Hardware-software co-design; SystemC; Simulation; HLS

1 Introduction

Deep Neural Networks (DNNs) have demonstrated high accuracy in learning tasks, such as image classification, speech recognition and many more. However, current solutions that attempt to deploy DNNs on low-power and resource-constrained edge devices (e.g., smartphones, tablets, and wearables) are inefficient, presenting challenges that can span several levels of the hardware/software stack to run efficiently [TCR$^+$18]. To address these inefficiencies, custom hardware accelerators for FPGAs have been developed to reduce the DNN inference cost.

Given the resource-constrained nature of edge FPGAs combined with the large memory footprint and computational demands of DNN models, a given DNN is unlikely to fully fit on an accelerator. Thus for DNN inference, the accelerator must operate in close communication with the CPU, which requires careful co-design with the host CPU code to ensure that data is managed efficiently.

Therefore, an effective design methodology for a DNN accelerator design should, for a given set of hardware resource constraints, produce performant accelerators that effectively leverage available resources and can respond quickly to changing workload requirements (e.g., introduction of new types of layers or operations).

To address these challenges, developers can utilize the SECDA methodology [HGC+21] (*SystemC Enabled Co-design of DNN Accelerators*). A key part of SECDA is that most hardware design is performed in simulation, and can be easily synthesized on real hardware (i.e., an FPGA) for more robust testing. However, the first step in any instantiation of a SECDA-based workflow is the initial integration with the target Application Framework. This instantiation includes sub-steps such as setting up the simulation environment and providing a path to offload host-side computations to the accelerator designer. This first step requires great developer effort when creating new projects which can be considered as redundant as the first step is similar for new projects within the same Application Framework. Therefore we highlight the tailored solution for this problem, SECDA-TFLite, an open-source toolkit which extends the TFLite DNN framework, such that it can be more easily used to develop new DNN hardware accelerators using the SECDA design methodology.

The SECDA-TFLite toolkit leverages the TFLite delegate system to provide a robust and extensible set of utilities for integrating DNN accelerators for any DNN operation supported by TFLite.

2 Background

SECDA (*SystemC Enabled Co-design of DNN Accelerators*) is a hardware/software co-design methodology to efficiently produce optimized DNN inference accelerators for edge devices using FPGAs. SECDA uses SystemC as an accelerator simulation framework, which also allows candidate designs to be iterated upon efficiently.

Using SystemC High-Level Synthesis (HLS), we can produce a synthesizable design from the same accelerator definition. The co-design of a software accelerator driver and a hardware accelerator is achieved by integrating SystemC's simulation features within the target edge-based DNN framework. Embedding the simulation environment and the hardware accelerator into the same software environment reduces the costs of exploring hardware/software co-design trade-offs via simulation, relative to synthesizing the design on an FPGA with every change.

3 SECDA-TFLite

SECDA is a generic methodology that can be applied to a variety of application frameworks (i.e., DNN inference frameworks), with the first step being to provide integration with the target framework. However, our observation is that there is a limited number of popular DNN inference frameworks, and thus this initial setup step can be reused between designs defined on the same framework, further reducing the barriers to developing new accelerators.

SECDA-TFLite is a TFLite-specific toolkit that provides the initial development environment when following the SECDA methodology within TFLite, and a set of utilities to aid development. This enables the developer to begin prototyping and integrating their new design with significantly reduced initial setup costs. While the original SECDA case study was embedded within TFLite [HGC+21], the integration was ad-hoc since it was focused on

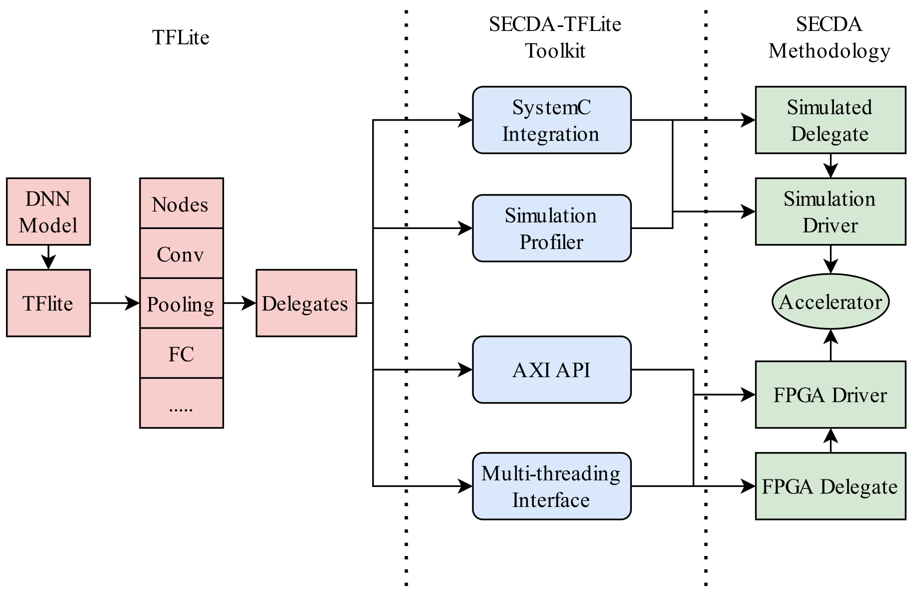

Figure 1: Overview of the SECDA-TFLite toolkit and how it is used within the SECDA design methodology for TFLite.

providing support for a single accelerator, rather than a generic integration for future developers. SECDA-TFLite aims to be a robust open-source toolkit for anyone who wants to develop new DNN accelerators within TFLite. Figure 1 shows an overview of the key aspects SECDA-TFLite's integration in TFLite. Highlighting the four key components of the toolkit: SystemC Integration, Simulation Profiling, Data Communication, and Multi-threading libraries. These provide the essential utilities which enable developers to start developing their designs.

4 Case Studies & Evaluation

To demonstrate the value of the SECDA-TFLite toolkit and how it provides a foundation for efficiently developing DNN accelerators within TFLite using the SECDA methodology, we developed three different FPGA-based DNN accelerator designs targeting Convolutional and BERT based DNN models.

We develop the designs for resource-constrained edge devices such as our target device, the PYNQ-Z1 board. For Convolutional Neural Network (CNN) models we port, improve, and integrate the Vector MAC (VM) and Systolic Array (SA) based designs, which were previously defined within the original SECDA case study [HGC$^+$21], using the SECDA-TFLite toolkit. We accelerate convolutional layers, which in TFLite are implemented using the *GEMM convolution* algorithm. Thus, we develop the aforementioned custom accelerators and their respective drivers to reduce the inference time of the model. For models in the BERT family, we note that they contain high-level DNN layer structures commonly referred to as *transformer layers*. However, these transformer layers can be decomposed to several Matrix Multiplication operations, which are represented as Fully Connected (FC) layers within TFLite models. Thus for BERT-based models, we develop a new FC-GEMM accelerator design targeted to accelerate the FC layers within TFLite.

For CNN accelerator evaluation we benchmark five widely used CNN models quantized to signed 8 bits integers: MobileNetV1, MobileNetV2, InceptionV1, ResNet18, and ResNet50, all defined on the ImageNet dataset. Similarly we benchmark 8 bit quantized MobileBert and Albert to evaluate the FC-GEMM accelerator.

For the VM accelerator, we observe an average speedup across models of 2.5× and 1.5× and an average energy saving of 2.3× and 1.5× for one and two threads respectively, in each case when compared to CPU-only inference.

Similarly, for the SA accelerator, we observe an average speedup across models of 2.9× and 1.7× and an average energy saving of 2.5× and 1.6× for one and two threads, respectively, in each case when compared to CPU-only inference.

Finally, for our FC-GEMM accelerator, we observe an average speedup across models of 2.5× and 1.6× and an average energy saving of 2.4× and 1.6× for one and two threads respectively, in each case when compared to CPU-only inference.

5 Related Works

While there are a range of design tools for designing DNN accelerators, there are fewer tools available to enable the developer to easily integrate their design workflow and design with a pre-existing DNN framework. SMAUG [XYB+19] provides a simulation-based design methodology that uses gem5-Aladdin to perform full system simulation of the host system, the off-chip memory accesses, and the accelerator design itself. However, SMUAG does not does not have a direct path to map candidate designs to hardware and run hardware evaluation on target FPGA devices with the chosen DNN framework.

6 Conclusion

The SECDA-TFLite toolkit enables tight integration of accelerator designs with TFLite while also enabling the developer to easily follow the SECDA design loop within TFLite, thus improving opportunities for co-design of the accelerator delegate and driver. We provide utilities for SystemC interfacing, simulation profiling, data communication, and multi-threading for the driver.

References

[HGC+21] Jude Haris, Perry Gibson, José Cano, Nicolas Bohm Agostini, and David Kaeli. SECDA: Efficient Hardware/Software Co-Design of FPGA-based DNN Accelerators for Edge Inference. In *SBAC-PAD*, pages 33–43, 2021.

[TCR+18] J. Turner, J. Cano, V. Radu, E. J. Crowley, M. O'Boyle, and A. Storkey. Characterising Across-Stack Optimisations for Deep Convolutional Neural Networks. In *IISWC*, pages 101–110, 2018.

[XYB+19] Sam Likun Xi, Yuan Yao, Kshitij Bhardwaj, Paul Whatmough, Gu-Yeon Wei, and David Brooks. SMAUG: End-to-End Full-Stack Simulation Infrastructure for Deep Learning Workloads. *arXiv*, 2019.