ONNX-to-Hardware Design Flow for the Generation of Adaptive Neural-Network Accelerators on FPGAs

Federico Manca[2], Francesco Ratto[1]

[1]University of Cagliari (IT), [2]University of Sassari (IT)

Abstract

Neural Networks (NNs) provide a solid way of executing different types of applications speeding up onerous and long workloads. Their implementation at the **edge** entails many challenges, such as offering **diversity** and **flexibility**, while guaranteeing **sustainability**. That implies supporting evolving applications and algorithms in an **energy-efficient** manner. Using hardware or software **accelerators** can deliver fast and efficient computation of the NNs, while **adaptivity** can be exploited to support **long-term flexibility**. Handcrafting a NN for a specific device takes a lot of time and experience, and that's why **frameworks** for hardware accelerators are being developed. This work in progress is focused on combining the toolchain proposed by Ratto et al. [1], which has the characteristic capability of favoring adaptivity, with **Approximate Computing (AC)**.

Review of the State-of-art

We analyzed two possible solutions that exploit a similar **streaming architecture**: **FINN**, an experimental framework from **AMD Research Labs** based on **Theano**; **HLS4ML**, an open-source software designed to facilitate the deployment of **machine learning models** on **FPGAs**, targeting **low-latency** and **low-power edge applications**.

FINN

A given model is compiled by the **FINN compiler** which produces a synthesizable C++ description of a **heterogeneous streaming architecture**. All **QNN** parameters are kept stored in **the on-chip memory**. The computing engines communicate via the on-chip data stream. An ad-hoc topology is built for the network.

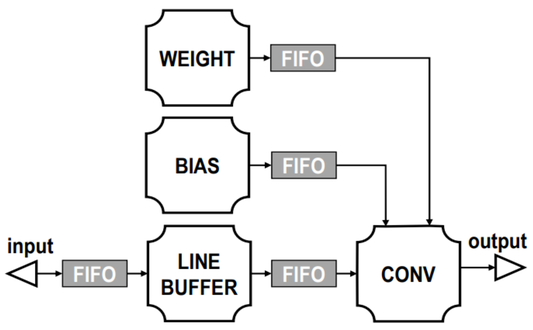
hls4ml

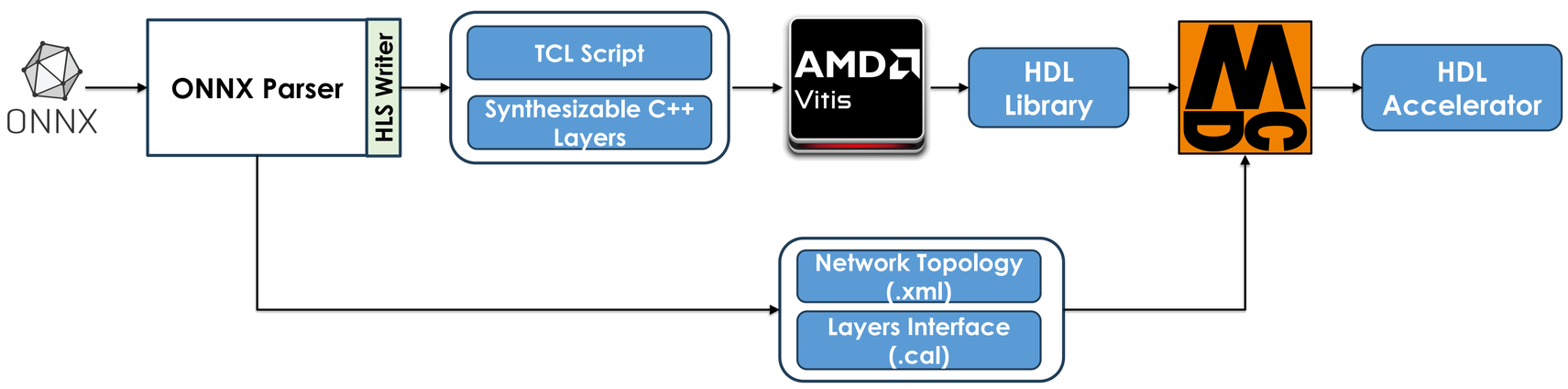
The main operation of the **HLS4ML library** is to translate the model of the network into an **HLS Project**, translating traditional open-source machine learning package models into HLS , configurable for ad-hoc cases.

Proposed Design Flow

TOOLS

- The **ONNXParser**, a Python application which parses the ONNX models and automatically create the code for a target device;
- The **Vitis HLS** tool, which synthesizes a C or C++ function into **RTL** code for implementation on **AMD FPGAs;**
- The **Multi-Dataflow Composer (MDC),** an open-source tool that can offer optional Coarse-Grained reconfigurability support for hardware acceleration.



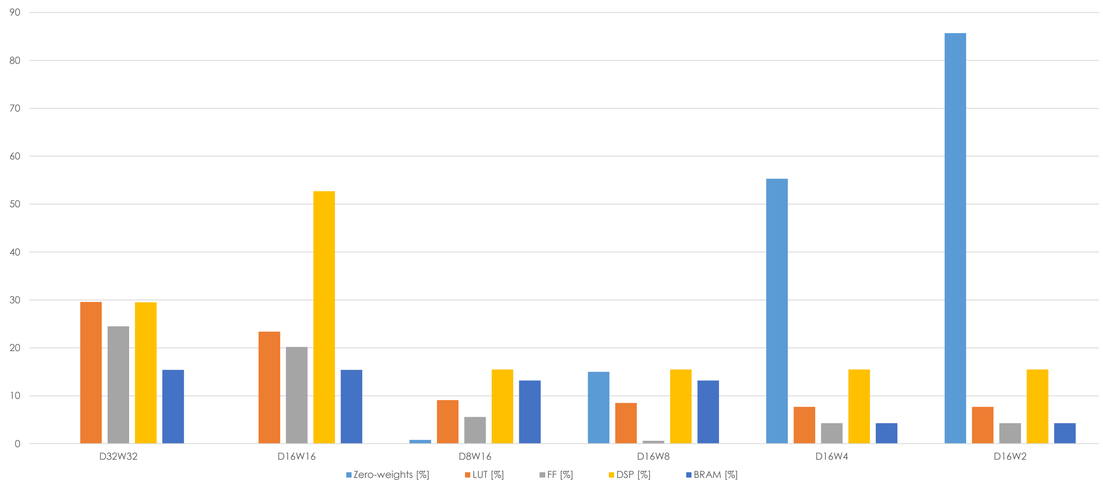


TOOLFLOW

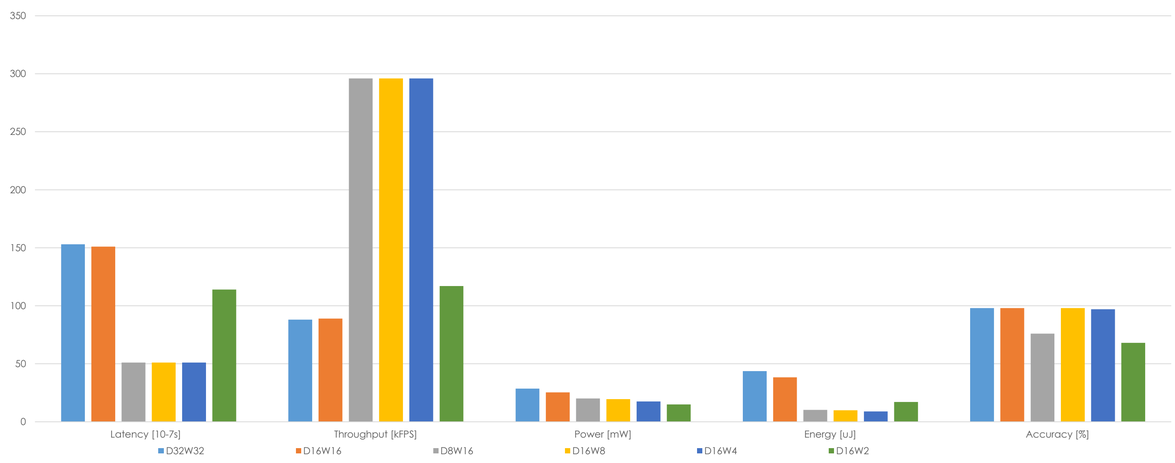
The model representation is fed to the **ONNX Parser**. The generated **C++ files**, based on an **ad-hoc template** that implement the layers and the **TCL** scripts, for the automatization of the synthesis, are given in input to **Vitis HLS**. The **HDL library** produced by Vitis HLS is given as input to the **Multi-Dataflow Composer**, together with the **XDF file** and the **CAL files**, which generates the HDL file of the complete dataflow.

Results

Results of exploration with **mixed precision** data on an accelerator made of two convolutional blocks (consisting of a **convolutional layer**, **max pooling**, **batch normalization**, and **ReLU activation layers**) followed by one **fully connected layer**. The accelerator classifies samples from the **MNIST dataset**. The model is quantized using **post-training quantization**. In the Datatype column, Dx-Wy denotes that x bits are used to represent **activations** and y bits are used to represent **parameters in fixed-point precision**. The reported results target a **Zynq7000**, comprising a ``xc7z020-1csg484ces'' chip, and have been retrieved through **post-synthesis simulations.**



Percentage utilization of LUTs, FFs, DSPs, BRAMs and Zero-Weights for every case studied.



Latency [10^{-7}s], Throughput [kFPS], Power [nW], Energy [uJ], and Accuracy [%] for each case.

It can be noticed that **accuracy** is not as affected by reducing **parameter precision** as it is by reducing **activations precision**. Moreover, reduced parameter precision leads to a **reduced memory footprint** (BRAM column) and a high percentage of **zero weights**. This latter can be exploited to combine quantization with **pruning**, which skips multiplications by zero. It is planned to carry out a broader comparison against state-of-the-art, based on significant on-board measurements and targeting more complex datasets. Nonetheless, it is worth recalling that state-of-the-art approaches are not conceived to support runtime adaptivity, which is motivating our research instead

Ongoing and Future work

- Our **ongoing work** intends to explore mixed precision in **adaptive** NN accelerators. NNs have demonstrated strong resilience to errors and can take great advantage of **Approximate Computing**.
- The ultimate goal will be the **efficient runtime management** of the system that implies, as a first step, the **combination** of the different **working points** over a reconfigurable substrate. This latter can be certainly achieved by leveraging on the whole set of functionality offered by the **MDC tool** to deploy and operate **reconfigurable** and **evolvable** NN **accelerators** for CPS, including the one presented in this study. Resulting accelerators will be able to **switch configurations at runtime** to **adapt** to the desired goal.

[1] Ratto, Francesco, et al. "An Automated Design Flow for Adaptive Neural Network Hardware Accelerators." Journal of Signal Processing Systems (2023): 1-23.

CPS SUMMER SCHOOL, 18-22 SEPTEMBER 2023, ALGHERO(IT)