Deploying Deep Learning on FPGA: an assessment of ConvNets performance on Xilinx Zynq MPSoC using Vitis-AI development platform

Gabriele Cuni
Politecnico di Torino

***Abstract*—Deep neural networks are one of the most promising technologies in the IoT field, nevertheless they require a high number of operations to be executed. IoT application development is often subject to strict limitations in terms of hardware resources, which makes it complex to use deep machine learning techniques on edge devices. Additionally, edge computing often requires low execution times, in order to be suitable for real-time applications. The above contrasting requirements place a challenging technology problem, which can be addressed by deploying deep neural networks on an efficient and optimised hardware. Field programmable gate array (FPGA) can be a viable alternative to GPUs to accelerate deep neural network inference, even on edge devices. In this thesis, we propose an assessment of ConvNets performance, achieved through Vitis-AI on a Zynq UltraScale+ MPSoC. The assessment is done by testing a set of Mobilenets, which has been obtained by varying the width multiplier and the input resolution, on different FPGA configurations, that are obtained by varying the allocated resources to the deep learning processing unit. On one hand the results show a high throughput, which is compatible with real-time application, even on the smallest available FPGA architecture. On the other hand, all models suffer a large accuracy reduction, that is due to the need to use the post-training quantization technique. Although, taking into account, as an upper-bound, the achievable accuracy using a quantization aware-training technique, the results show that the deployment of deep neural networks on FPGA is a viable option.**

***Index Terms*—FPGA, Xilinx Vitis AI, Deep learning, Mobilenet**

I. Introduction

FPGAs are an efficient alternative to GPUs, but at the moment there are not standardized development flows for the deployment of deep neural networks on them. Nevertheless, thanks to their high parallelism, FPGAs are an excellent hardware option for the implementation of deep neural networks, because they are more energy efficient with respect to GPUs and have a high flexibility.

This thesis work proposes a deployment flow that accepts as input a state-of-the-art convolutional neural network pre-trained on the Imagenet dataset and reproduces as output an executable model which can be deployed on a Xilinx FPGA, the Zynq UltraScale+ MPSoC. The proposed flow is made over the Xilinx Vitis AI environment and the Tensorflow framework and it deals with all the aspects necessary for the treatment of the models. During the thesis eight different FPGA hardware configurations are made and evaluated over the Zynq UltraScale+ MPSoC. The proposed architectures are generated by selecting different resource utilization on the FPGA. Finally, thanks to the proposed development flow sixteen different Mobilenets are tested for each FPGA hardware configuration. The Mobilenets topologies are obtained by varying the width multiplier and the resolution multiplier hyper-parameters. The testing phase had the objective to evaluate the performance obtained in terms of inference speed and the accuracy by the models once run on the board.

II. Background & Assessment

A. Zynq UltraScale+ MPSoC

The Zynq UltraScale+ MPSoC ZCU104 is an evaluation board whose purpose is to enable the design and testing of embedded machine learning vision tasks and applications. It is composed of a Processing System (PS) and a Programmable Logic (PL) which is the FPGA. The processing system is equipped three processors: an Arm Cortex-A53 (CPU), a Real-time Processing Unit and a GPU. Instead, the programmable logic is a 16nm FinFET+ FPGA that is the home of the Deep learning Processing Unit (DPU).

The Deep learning Processing Unit is a set of parameterizable IP cores that are implemented on the programmable logic of the Xilinx board in order to accelerate the software application for computer vision as deep neural networks. It is the hardware configuration of the FPGA and in this thesis 8 different DPUs are made with the DPU-PYNQ software by varying the number per clock operation that the DPU is able to achieve.

B. Xilinx Vitis AI

The Vitis AI development environment is the set of tools and libraries given by Xilinx in order to deploy deep neural networks on the Xilinx boards. It can be used to deploy a neural model taken from the Vitis AI Zoo which contains a selection of pre-optimized models ready to be deployed, otherwise it can be used to deploy a custom model. The Vitis AI utilities used in this project are the quantizer and the compiler:

- AI Quantizer: It transforms a floating point model into a fixed point 8-bit integer model and it supports the post-training quantization and the quantization aware-training. The quantizer requires a representative calibration dataset, which is used to calibrate the model weights

and activations. The dataset is a subset of the validation Imagenet dataset.

- AI Compiler: It is the utility which transforms the quantized model in a format that is compatible with the Deep learning Processing Unit (DPU) of the FPGA. The compiler requires as input the quantized model by the Vitis AI compiler and fingerprint file of the target DPU.

C. Deep neural networks deployment flow

In this chapter the proposed deployment flow will be analyzed. By observing the figure 1 it can be immediately understood that the flow is divided into three main phases: the model building, the hardware building and the deployment phases.

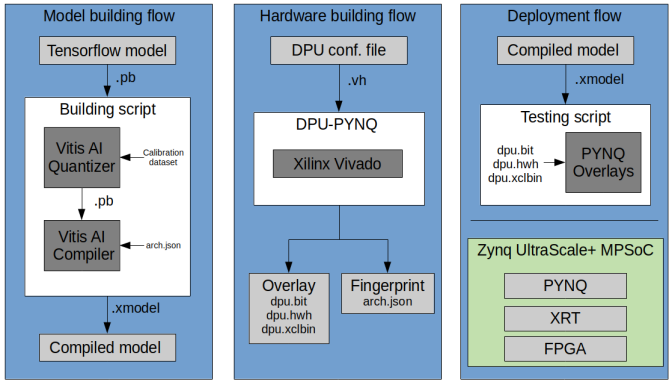

Fig. 1. The proposed deployment flow

The hardware building phase is done on the host computer. The objective is the creation of different DPU overlays in order to program the hardware configuration of the FPGA. These overlays are made with the DPU-PYNQ environment, which creates the DPU specifically for the PYNQ environment that is installed on the Zynq board. The two main results of this flow are the overlay files and the fingerprint file. This phase is the first one that must be executed, because the resulting files are mandatory to execute the other two flows.

The second phase is the model building one, which is also executed on the host computer. The main operations are the post-training quantization and the compilation that are paramount to have a deployable deep neural network graph. Instead, a secondary operation performed here is the accuracy measurement of the models after quantization. These steps are done by the proposed Python building script, thanks to the Vitis AI library. It is important to notice that the compiler requires the fingerprint file created in the previous flow.

The last flow is the deployment on the Zynq Ultrascale+ MPSoC, obviously this is the last phase to be done as it requires the compiled model from the model building flow and the DPU overlay files from the hardware building flow. During this phase is executed the testing script in Python that evaluates the inference throughput of the compiled model on the FPGA.

D. Deployment on the Zynq UltraScale+ MPSoC

The deployment of the neural networks on the board starts by uploading the base overlay, which is the reference design of the board made by the Xilinx Vitis AI platform. The base overlay is made up of three files: the bitstream file, the Vivado design file and the bit file that provides the python wrappers. They are made during the DPU generation and there are a specific trio of files for any DPUs. Other than configuration files, also the neural networks obtained from the compilation process must also be uploaded on the Zynq board. The neural models are in the Xilinx format after the compilation, which is a single graph file with extension xmodel. The Zynq operating system and the PYNQ application expose a Jupyter Notebook environment thanks to which it is possible to control the device and execute bash commands via a linux shell.

On board of the Zynq UltraScale+ MPSoC device a Python script was executed in order to test all the mobilenets with all the DPU architectures. The software we created accepts as input the hyper parameters of the mobilenet model that must be tested and the name of the DPU on which you want to test. Then the script performs the latency test by submitting 200 images of the imagenet validation dataset to the neural network and calculates the preprocessing time, the inference time and the total time of both the operations.

It is important to note that not all the code executed on the Zynq device is executed on the FPGA areas. The image preprocessing and the throughput measurement are executed on the quad-core Arm Cortex-A53, instead the feed forward pass of the Mobilenet is executed on the FPGA specially configured by the selected DPU.

III. Experimental Results

Here will be analyzed the results obtained on the fastest and the slowest DPU architectures. The number of frames per second indicated in the graphs is calculated from the total of the preprocessing and inference times. Moreover, it is important to notice that the throughput results expressed in FPS are taken from the measurements done on the Zynq UltraScale+ MPSoC within the testing script proposed by us. Instead the showed accuracy results are an achievable upper-bound obtained by Tensorflow developers thanks to the quantization aware-training technique on which the Vitis AI quantizer is based.

A. B4096 Architecture

Figure 2 shows the results obtained with the best performing DPU architecture in terms of number of operations per clock. As you can see from the graph, the four neural networks with the alpha parameter equal to 1 are the choice to be made if you want to maximize accuracy. It is interesting to note that with the performances offered by this DPU, even the most accurate network (70.1% top-1 and 88.9% top-5), and therefore also the most demanding of resources, is able to exceed the threshold of 35 FPS considering both preprocessing and inference. This result guarantees the possibility of using

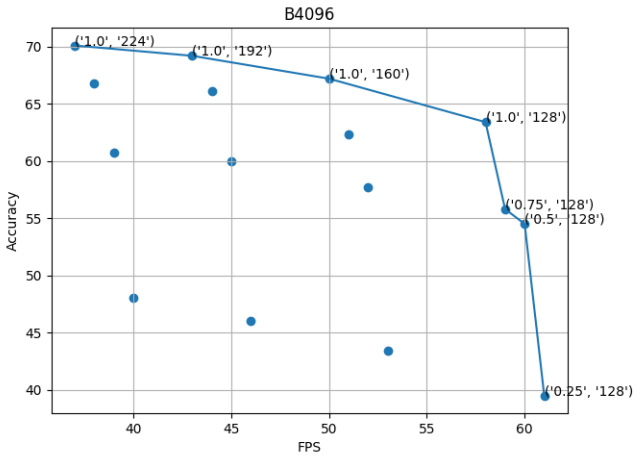

Fig. 2. B4096 is the best performing DPU architecture

this architecture with the most accurate model even in a real-time environment. In addition, it should be noted that the fastest network on each architecture, which is Mobilenet 0.25 128, on the B4096 is capable of reaching a throughput of 61 FPS. This is the maximum result obtained in terms of speed, taking into account the total time including preprocessing and inference.

B. B512 Architecture

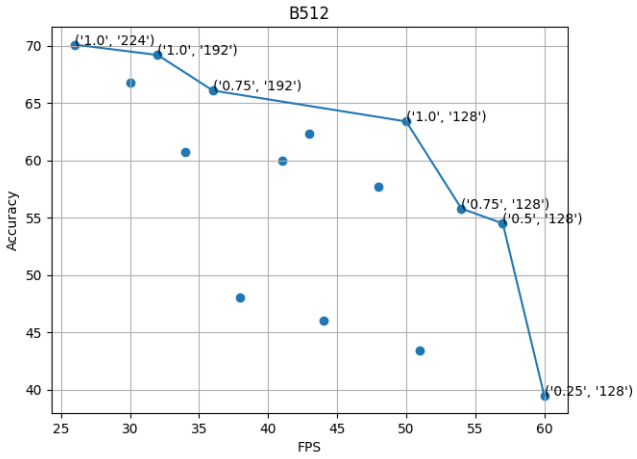

Fig. 3. B512 is the DPU architecture with less resources

After the best performing DPU it is interesting to analyze the smallest one, the so-called B512 architecture. As can be seen in figure 3, the fastest model and the more accurate one remains the same as on the B4096 architecture. With the B512 architecture, however, the most accurate network does not allow to obtain a minimum of 30 FPS, therefore to exceed this threshold you have to sacrifice a bit of accuracy by reducing the size of the input image. It is interesting to notice the best throughput, that is equal to 60 FPS, because it is not far from the result obtained on the B4096; therefore if the hardware resources optimization is taken into account, the best choice is the B512 architecture.

C. Rules of thumb

Here are some general considerations regarding the outcomes deriving from the analysis of all the graphs and numerical outputs:

- If the goal is to have the highest accuracy, it is necessary to keep the width multiplier equal to one and select how many FPS to obtain thanks to the size of the input image. By varying the image size parameter, a very high increase in inference speed is obtained at the expense of a small loss of accuracy.
- If the aim is to exceed the threshold of 50 FPS it is necessary to lower the width multiplier, thanks to which it is possible to exceed the threshold of 60 FPS. However, this increase in preprocessing and inference speed leads to a notable decrease in accuracy.
- Speaking of architectures, choosing a DPU with a high number of resources is advantageous only if you want to use a high width multiplier parameter, instead with a neuron number decreasing there is no saving by using a DPU with a high number of resources, in these conditions it is better to opt for the architecture that uses the least number of possible resources.

IV. Conclusions

In the research carried out during this thesis, we implemented and tested a deployment flow for hardware accelerated deep neural networks on FPGA. The proposed development flow that deals with model preparation, programmable logic configuration and testing, allowed us to execute different Mobilenets topologies on the FPGA on board of the Zynq UltraScale+ MPSoC.

The accelerated implementation of the Mobilenets on the FPGA has got excellent performance on all the DPU configurations. The fastest model has a throughput of 61 FPS on the best performing DPU and 60 FPS on the smallest DPU configuration, with a top-1 accuracy of 39.5%. Instead, the more accurate model, which is also the slowest, has a throughput of 38 FPS on the fastest DPU and 26 FPS on the smallest one, with a top-1 accuracy of 70.1%. The results show a high throughput, which is compatible with real-time edge applications, even on the smallest available FPGA architecture. Therefore, it can be said that the deployment of deep neural networks on FPGA is an excellent design choice, although it is necessary to be very careful in the quantization phase to avoid excessive accuracy reduction of the models.