Slide 1 - Frontespizio
Good morning to everyone, I am Gabriele Cuni and I am going to introduce the thesis I developed under the supervision of the professor Andrea Calimera and the Doctor Roberto Giorgio RIzzo. The title of the thesis is DEPLOYING DEEP LEARNING ON FPGA: AN ASSESSMENT OF CONVNETS PERFORMANCE on Xilinx Zynq MPSoC using Vitis-AI development platform.

Slide 2 - Objective
As anticipated by the title, the aim of my thesis was the deployment of deep neural network algorithms on a Field-Programmable Gate Array also known as FPGA and the assessment of the performances in terms of latency and accuracy.

Slide 3 - Outline
Here you can see an outline of the principal assessment steps and components used, that I will introduce.
I will start with some introduction, than there is the deployment and assessment section where I will anayze the used state of the art CNN mobilenet, the tools and libraries used Vitis AI, the HW device and the deployment flow, finally there will be the experimental results

Slide 4 - Context
Starting from the research context.
Lately Deep neural networks are one of the most promising technologies in the IoT field driven by the quick advances in mobile computing.
Therefore there is an urgent need to push the AI frontiers to the network edge where the data is generated.
Edge Intelligence is a combination of Deep Neural Network algorithms and Edge Computing, which enables cutting-edge application such as Smart Cities and Internet of Vehicles

Slide 6 - Motivations
The reasons that make FPGAs an interesting HW for edge computing are the high energy efficiency in relation to GPUs that are direct competitors.

the high possibility of configuration and therefore flexibility thanks to the reprogrammable matrix of blocks and interconnections.

The low costs and flexibility compared to ASICs and

Slide 5 - Challenges
There are challenges to overcome to deploy CNN on FPGA that are:
the quantization techniques necessary to reduce the size of the models and run them even in the absence of the floating point unit.
(8-bit now) (16-bit future)
Quantization can lead to accuracy loss.

Furthermore, there is a lack of standardized tools to perform these operations and

This is a new research field that is starting to be explored in recent years for which not much literature is available on the subject.

Slide 7 - Mobilenets

The CNN model used was the MobileNet,
I choose it because it is a state of the art model and it is a benchmark for edge computing.
I also chose it, because it offers 2 knobs, the alpha and rho parameters used to change the topology of the model.
By changing these 2 knobs I made 16 different topologies to make an assessment of the performance with different neural networks.

Slide 8 - Xilinx Vitis AI

I used Xilinx Vitis AI tools and libraries to automate the CNN deployment flow.
Vitis AI accepts neural models coming from the major available framework as custom models or ready-to-use models from the Model Zoo.

Specifically the Vitis AI development kit provides the quantizer and compiler tools that are fondamenta steps in the deployment flow.
Quantizer does both post-training quantization and quantization aware-training.
The compiler converts the model into a format that is executable on the FPGA.

The configuration of the FPGA is done by means of the overlay.
The overlay is the HW representation of the CNN that in Xilinx is called Deep learning Processing Unit aka DPU.
The dpu is a set of **parameterizable hardware blocks** that are implemented on the FPGA and can be configured by choosing the number of resources that will be used as the number of Digital Signal Processing Unit, the number of Block RAM and LUT.

Slide 9 - Zynq and PYNQ

This is the Zynq Ultrascale + development kit used for the deployment of neural networks.
it is equipped with a Arm cortex A53 CPU on which the image pre-processing is executed, because there was no HW library for the computation of the preprocessing on the FPGA,
The feed forward pass of the neural model is performed on the FPGA.

The OS of the board is PYNQ that makes use of the linux kernel.
I chose to use PYNQ, because It allows the use of the Python language to access the FPGA HW resources.
It also exposes an easy to use interface thanks to Jupyter Notebook.

Slide 10 - FLOW

In this slide you can see the diagram of the deployment flow I have automated to test the libraries made available by Vitis AI and the performances that can be reached by CNN models on FPGA.

I created the building script, the testing script and
I chose how to configure the FPGA by choosing the max speed achievable by the configuration as you can see here B4096, than B3136, ecc ecc
and I chose the MobileNets topologies by choosing alpha and rho knobs.

The flow is divided into three main steps.

The first step is the Hardware Building Flow to make the FPGA HW configuration here called DPU.
I have made 8 different FPGA configurations by varying the number of operations per clock that any of them is able to execute.
The Hardware generation is then executed by the DPU-PYNQ and Xilinx Vivado applications which generates the DPU overlays and the fingerprint file for the Vitis AI compiler.
Varying the max speed knob of the DPU also the number of Digital Signal Processing block, Block RAM, Lookup Table and Flip-Flop are changed.

The second phase is the Model Building Flow which is a Python Script that I have made in order to quantize the MobileNets models, evaluate the quantized model and Compile the model.
The scripts make use of the Vitis AI libraries in order to execute the post-training quantization technique on the MobileNet weights and activations.
Then the compilation of the model is done, so the model is formatted such that it can be executed on the FPGA.
The fingerprint is used to compile the model for a specific DPU.
During the Model Building flow the quantized Mobilenets are also validated by measuring their accuracy on the imagenet validation dataset.

The third phase is the Deployment on the Zynq Ultrascale+ board.
Compiled model and FPGA configuration are uploaded on the Zynq device.
Here I have made the Testing script in python which carries out the throughput testing with 200 images
It computes the throughput for each MobileNet topology and for each hardware configuration of the FPGA.

Slide 11 - B4096

Here you can see the results of the 16 Mobilenet topologies on the B4096, the FPGA configuration with the most hardware resources.

The fastest MobileNet topology on the B4096 is capable of reaching a throughput of 61 FPS with an accuracy of 39.5%top-1 on the imagenet validation dataset.
This result takes into account the total time including pre-processing and inference.

The most accurate topology reaches 38 FPS with a Top-1 accuracy of 70.1%
This result guarantees the possibility of using the most accurate model even in real-time applications.

Analyzing the Pareto curve can be seen that the Rho knob is able to boost the throughput a lot without losing too much accuracy. Instead changing the Alpha knob drops the accuracy, but enables the fastest achievable throughput.

Slide 12 - B512

Here are the results obtained on the B512 FPGA configuration, that is the DPU with the fewest hardware resources.

The fastest MobileNet topology on the B512 architecture reaches a throughput of 60 FPS and a Top-1 Accuracy of 39.5%.

The more accurate one instead reaches a throughput of 26 FPS and an accuracy of 70.1%

Analyzing the Pareto curve can be seen, both Alpha and Rho knobs must be used to have an increment in FPS.
Which leads to a greater reduction in accuracy for having the same FPS as on the B4096

If the objective is the hw resources saving the B512 offers almost the same maximum FPS of the B4096, but it is not a good choice for the most accurate topologies on which FPS drops a lot.

Slide 13 - LUT

Here you can see the analysis of all the produced DPUs.
I have used the Lookup Table number in order to show the increment of throughput by incrementing the FPGA resources.
Switching between DPU architectures leads to a different increase, in terms of FPS, depending on the alpha parameter of the model. Lower is the alpha parameter, lower is the FPS increase by changing the architecture;
Higher is the alpha parameter, higher is the FPS increase by changing the architecture.

This way can be seen that topologies with alpha equal to 0.25 have enough resources even on the smallest DPU available, instead topologies with alpha equal to 1 need a bigger number of resources to fully show their capability

Slide 14 - Conclusion and future works

The accelerated implementation of the MobileNets on the FPGA has got excellent performance on all the DPU configurations.

The results have shown 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.