RWTH AACHEN UNIVERSITY

INTERNSHIP REPORT

Industry internship at Enclustra FPGA Solutions, Räffelstrasse 28, CH-8045 Zürich

presented by
Arthur Ruder
Matr.-Nr. 310697

Supervised by Jelena Dragas

Aachen, 06.09.2019

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
API	Application Program Interface
APU	Application Processing Unit
ASIC	Application Specific Integrated Circuit
AXI	Advanced eXtensible Interface
CPU	Central Processing Unit
CSI2	Camera Serial Interface
CSI	Camera Serial Interface
DMA	Direct Memory Access
DNNDK	Deep Neural Network Development Kit
DPU	Deep Learning Processor Unit
DSP	Digital Signal Processor
FAE	Field Application Engineer
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
GUI	Graphic User Interface
HDL	Hardware Description Language
IP	Intellectual Property
MAC	Multiply and Accumulate
MIPI	Mobile Industry Processor Interface
ML	Machine Learning

MPSoC	Multiple Processor System on Chip
OS	Operating System
PCIe	Peripheral Component Interconnect Express
PL	Programmable Logic
PS	Processing System
PWM	Pulse Width Modulation
RPU	Real-time Processing Unit
SDK	Software Development Kit
SDR	Software Defined Radio
SDSoC	Software Development System on Chip
SoC	System on Chip
USB	Universal Serial Bus
VPU	Vector Processing Unit

1. Week

The first two days of the week were spent getting to know all colleagues and familiarize myself with internal processes and guidelines. Zurich is the headquarters of Enclustra GmbH and therefore the majority of hardware and software design is being done here. Around forty people, most of which are hardware and software engineers, work in the Zurich office. The company itself is divided into two areas, *Field Programmable Gate Array (FPGA)* Design Center and FPGA Solution Center. The former is offering customer-specific design services implementing applications on FPGAs and providing support and custom *Intellectual Property (IP)* components. Areas of expertise include wired networks and switching, wireless communications (*Software Defined Radio (SDR)*), smart cameras, embedded interfaces (*Peripheral Component Interconnect Express (PCIe)*, *Universal Serial Bus (USB)*, *Advanced eXtensible Interface (AXI)*, ethernet, etc.), test and measurement (sensors, data acquisition, *Digital Signal Processor (DSP)*) and drive/motion control. The latter designs custom FPGA/*System on Chip (SoC)* modules and IP solutions. Several base board families and FPGA module families are developed and supported which can be adapted to the needs of the application by offering different performance key points. Reference designs for each combination of base board and module are provided as a starting point for customers. Furthermore, my task was to do market research on artificial intelligence and artificial intelligence on FPGAs especially. The four key platforms for artificial neural network applications are shown in 1.1. A qualitative design trade-off is shown on the x- and y-axis in terms of power efficiency and performance versus flexibility and ease-of-use. As Enclustras focus is on the embedded market, the market survey has been mainly on *Graphics Processing Units (GPUs)*, FPGAs and *Application Specific Integrated Circuits (ASICs)* as full blown *Central Processing Units (CPUs)* are too inefficient for embedded applications. Possible competitors as well as toolkits provided by FPGA manufacturers such as Intel, Xilinx and Lattice have been evaluated. The results have been presented in a meeting in which a discussion has been held, where Enclustras products and services can fit. One of

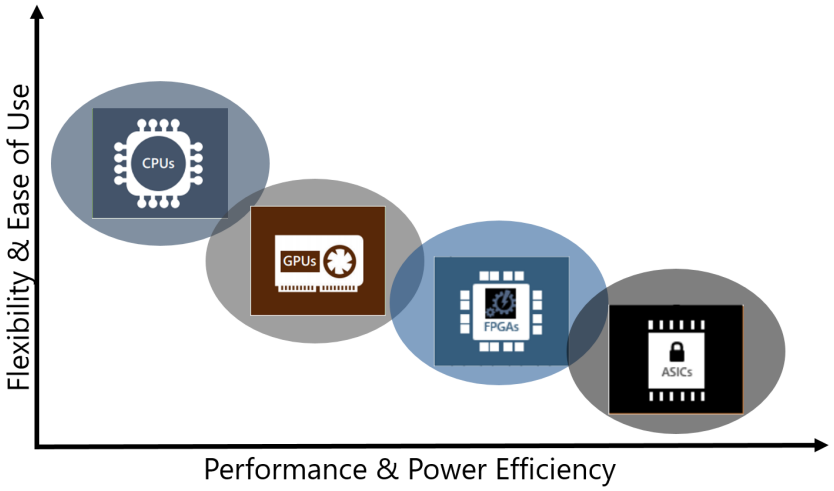

Figure 1.1.: Hardware platform overview

the results was to start planning an *Artificial Intelligence (AI)* demonstrator using Enclustra hardware. The purpose of this demonstrator was to showcase machine learning applications running on Enclustra hardware. As a preliminary step it was decided to check the Xilinx *Deep Neural Network Development Kit (DNNDK)* samples on an evaluation board, the ZCU 104.

2. Week

At the beginning of the week a task unrelated to AI was given to check upon internal documentation and customer support. Together with another recently hired employee, a day of out-of-box testing was scheduled. An Enclustra base board (Mercury+ PE1-400) together with a fitting FPGA module (Mercury+ XU1) featuring a Xilinx *Multiple Processor System on Chip (MPSoC)*. Some details will be given for this specific FPGA family as the Zynq-7000 SoC and the Zynq-MPSoC Xilinx product family are unique in the way dedicated ARM processors are combined with traditional FPGAs. A high level overview is shown in figure 2.1. It shows the difference between

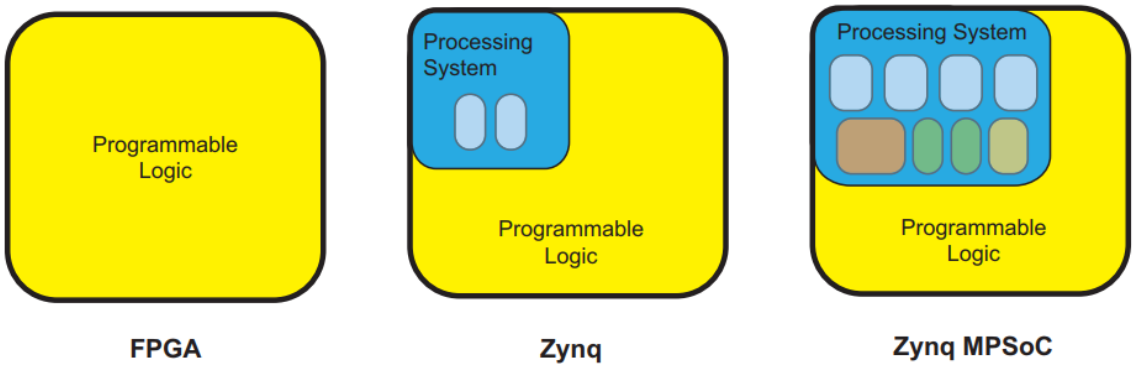

Figure 2.1.: High level ZYNQ family overview [1]

traditional FPGAs and the Zynq product family. The main benefit here is the division between *Processing System (PS)* and *Programmable Logic (PL)*. This allows to combine the benefits of traditional processing units with the flexibility of FPGAs. Custom logic and all peripheral devices can be implemented in the PL part, while system control and even complete *Operating Systems (OSs)* (such as embedded Linux) can be done in the PS. As this system is completely integrated in one package, the communication between the two fabrics is extremely fast and can be done using AXI interfaces. The MPSoC family even integrates several different types of processing units, *Application Processing Units (APUs)*, GPUs, *Vector Processing Units (VPUs)*

and *Real-time Processing Units (RPUs)*. The Enclustra module uses an MPSoC Xilinx FPGA and our task was to go through the whole process of bringing up the base board together with a corresponding module to test customer experience using the provided documentation, user manual and reference design. The goal was to find unclear instructions in the documentation and provide feedback as to the overall experience bringing up the hardware out-of-the-box. First, the hardware reference design was loaded in the Vivado Design Suite and the bit stream generated. After, the hardware description file was exported so it can be used using the Xilinx *Software Development Kit (SDK)*. This allows to create applications in C/C++ against the custom hardware design. All of the provided sample applications have been tested and verified. Some unclear instructions were identified and discussed with the employee in charge to improve customer experience. The rest of the week was spend updating the internal Wiki page for AI. Furthermore, the DNNDK sample applications were tested on the ZCU 104 evaluation board. The provided examples include several state-of-the-art neural networks demonstrating key applications for neural network inference, such as image classification, face detection, object detection and pose detection. As only the image classification example worked directly for this particular evaluation a fix needed to be found. Another task was to introduce the topic of AI to the whole company as *Artificial Neural Networks (ANNs)* was a completely new design field for a majority of the technical staff. Two PowerPoint presentations should be prepared, namely 'Introduction to AI' and 'Introduction to *Machine Learning (ML)* on FPGAs'. I started with the preparation of the first one in parallel with finding a bug fix for the other DNNDK sample applications, as these should be part of the second presentation.

3. Week

The main focus of this week was research and starting to layout the first presentation. It was assumed that the audience is technology savvy but has no particular background in AI. Thus, the presentation had to introduce the whole field and key concepts that enabled the rise of AI applications in recent years. The first draft of the presentation was discussed in a meeting and some changes were made to the overall structure, the content and the degree of complexity.The rough structure of the presentation is as follows:

- **Motivation**: To get the viewers interest it was shown that AI applications are already part of daily life for everyone. This was achieved by showing that all of the major companies such as Google, Apple, Facebook, Microsoft as well as Tesla, Netflix and Amazon use AI in their datacenters and products and allocate huge resources to AI research. The importance of AI was further enhanced by showing the rapid growth of annually published AI papers and startups developing AI systems. The trend from 1995 to 2015 resembles almost exponential growth in AI research and products.

- **Definition**: As AI has become such a buzz word in media a definition of the term was needed and what part of AI is actually used in all of the common applications. ANN that perform typical computer vision and language processing tasks are all part of ML, which is a subset of AI. ML itself can then be divided into further subsets using roughly three learning methods, supervised learning, unsupervised learning and reinforcement learning. As supervised learning is the most commonly used method, the presentation focused on this method used to train ANN. Furthermore, the different parts that comprise a ANN are introduced, namely the neuron and how neurons are formed into layers. These layers are then stacked together to form an ANN.

- **Key concepts**: An explanation of supervised learning was given with two distinct examples, linear regression and deep learning neural networks to

illustrate the idea behind supervised learning: predict a value y given an input x by deploying a function $f(x)$. This function $f(x)$ is acquired by deploying a learning algorithm and usage of a so called training set, consisting of input pairs x and y. The concept of inference and training were also explained with an emphasis on inference. Once a network is trained, only inference needs to be run, so this is the crucial part application wise.

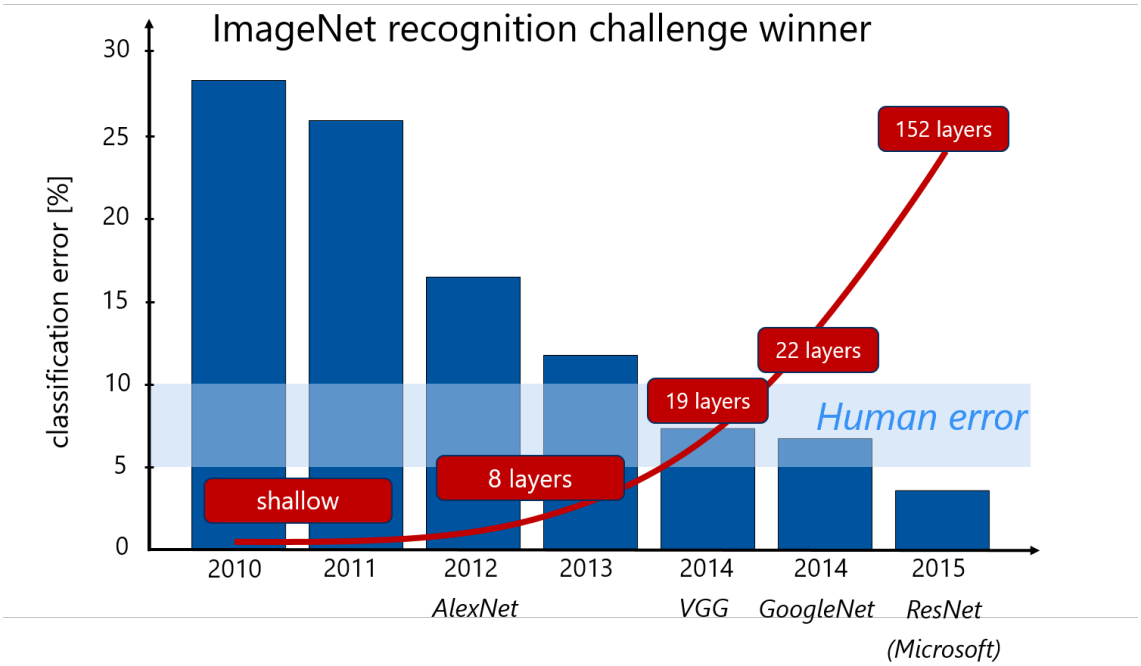

Figure 3.1.: ILSVRC winners

- **Deep Learning**: Figure 3.1 shows the winners of the state-of-the-art ImageNet Large Scale Visual Recognition Challenge. It illustrates that the breakthrough in performance came not only from more sophisticated networks, but mainly from stacking different kinds of layers deeply, hence the name. The presentation was ended with the question, what hardware is best suited for ANN applications. This question would be addressed in the second presentation 'Introduction to ML on FPGAs'.

Alongside preparing the presentation the error preventing the more sophisticated examples was identified as the board crashed while performing tasks related to video analysis (face detection, pose detection, etc.). At first, it was suspected there were

some problems with heat management and using the system monitor the temperature of the FPGA was investigated during operation. As this seemed to be well within allowed borders specified by the Xilinx data sheet, other causes had to be found.

4. Week

At the beginning of the week the first presentation 'Introduction to AI' was held before the technical stuff of the company. The general background and principles of ML have been introduced and an outlook given to the second presentation, which would go more into detail about the actual hardware realization. The rest of the week was spent going through various tutorials provided by Xilinx to familiarize myself with the workflow and the DNNDK toolkit. As the state of tools used for AI applications on FPGA is still in flux, several approaches needed to be evaluated:

- **DNNDK workflow**: Version 2.08 of the toolkit supported only the Caffe neural network training framework and needs a network description file and the trained weights as input. The key component here is the *Deep Learning Processor Unit (DPU)* IP core provided by the DNNDK toolkit. This core is integrated via Vivado into the block design of the hardware and can be configured and adjusted for several performance and power profiles.
- **DNNDK *Software Development System on Chip (SDSoC)***: Another option is to abstract away the whole Vivado block design process and use Xilinx SDSoC to implement the whole system in a higher programming language, C++. Supported functions can then be flagged as being executed in the PL part of the system. This approach makes using a traditional *Hardware Description Language (HDL)* obsolete and is deemed more accessible. This approach uses the established Xilinx reVision stack for development providing high level *Application Program Interfaces (APIs)* for computer vision.

Furthermore, the IP core provided by the DNNDK toolkit was studied in more detail. A new base board was in the production state phase and the idea was to have a ML design ready to showcase the capabilities of the new board. Figure 4.1 shows an example hardware design with integrated DPU module. In this example a camera is connected via the *Mobile Industry Processor Interface (MIPI) Camera Serial Interface (CSI2)* interface to the PS. *Direct Memory Access (DMA)* is usually

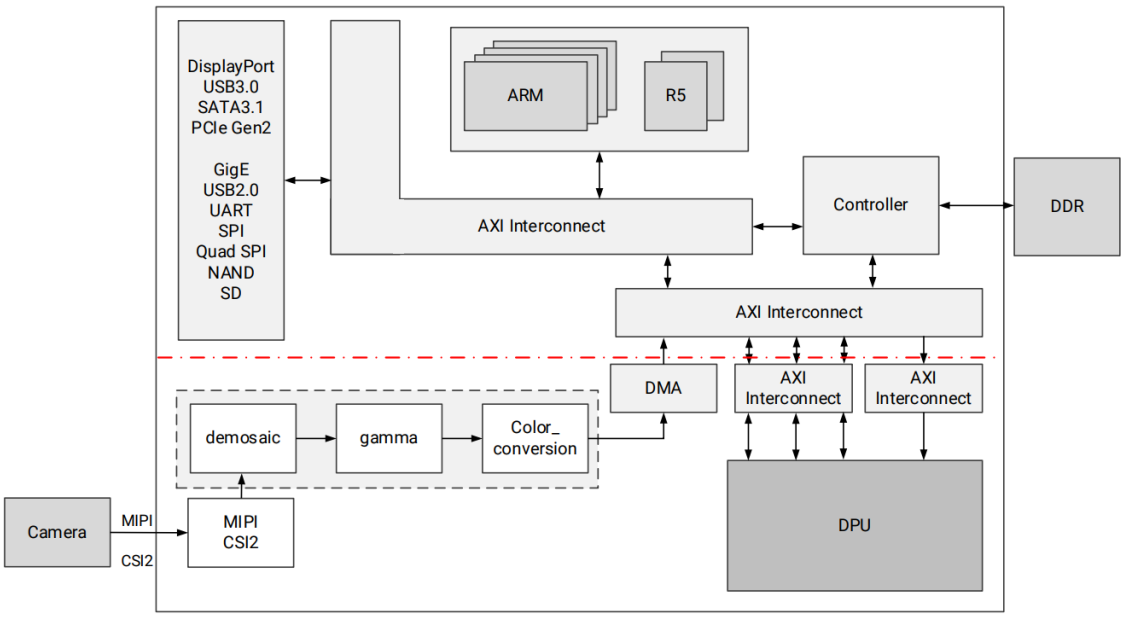

Figure 4.1.: Example system with integrated DPU [3, p. 8]

used in conjunction with an AXI interconnect to communicate with the PS. The captured images are used as the input to the neural network and the DPU itself can be viewed as a co-processor to the PS implemented in the PL fabric. The IP core itself is customizable and the number of DPU processor units, the size of the DPU and the usage of DSP blocks among other parameters are configurable. The decision on which size to use is based upon the performance demands of the application.

5. Week

In this week I started working on the second presentation 'Introduction to ML on FPGAs'. This time the focus should be on the hardware needed to handle typical ML workloads, namely inference and training along all major fields of applications where ANN are used. The main areas are: image classification, object detection, semantic segmentation, optical character recognition and speech recognition. The presentation structure is as follows:

- **ANN workload:** Using a state-of-the-art neural network (resnet50) the number of operations per image were illustrated to show the vast amount of compute and memory needed for a single image. This was done for inference and training respectively with the purpose of driving home the challenges involved in ML applications. Moreover, the majority of operations are costly *Multiply and Accumulate (MAC)* operations which take several clock cycles to complete.

- **Hardware for training:** The industry right now in terms of ANN training is dominated by NVIDIA and so the clear answer here was GPU. There are a number of start-ups developing alternative solutions to get into the ANN training market. The main advantage these start-ups have is that they can design from scratch and use an architecture tailored to the specific requirements of ANN training. The sheer number of floating point 32 operations and the requirements for memory are strictly not suited for FPGAs.

- **Hardware for inference:** The picture is different for inference. Here, a lot of research has been done in using quantization and pruning of ANN models without impeding the performance of these networks. The reason for this is, that neural networks are inherently over-parametrized and this is necessary for the training algorithms to work. Once a trained network is obtained however, the network can be compressed severely (up to 90 %) without network degradation. A qualitative comparison of the available platforms was made to

show the strengths and weaknesses of each platform. The flexibility of FPGAs make them a suitable platform for ANN inference.

- **FPGA architecture:** Figure 5.1 shows the two possible high-level architectures that are typically used in neural network implementations. On the left you have the streaming architecture where basically the structure of the neural network is mirrored by the hardware implementation. The main benefit is the efficiency and customizability as the hardware can be tailored specifically to each network. The other approach is shown on the right. This is a more general approach in that it has a single computation engine which breaks down the operations needed for neural network inference. These operations are controlled by a host and executed on demand. The main benefit here is the flexibility. As the types of layers in a neural network are fixed, efficient implementation of these different types of layers enables the deployment of arbitrary neural networks. The downside is that the implementation is limited by the architecture in terms of tailoring the hardware implementation to the neural network.

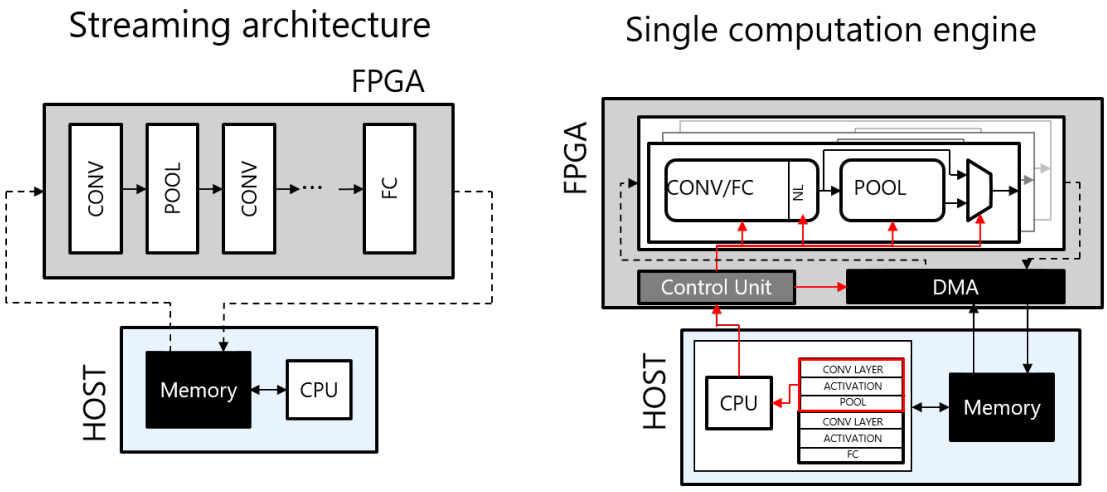

Figure 5.1.: FPGA architecture overview

- **DNNDK work flow:** Lastly, the DNNDK work flow is introduced with a focus on adjusting the DPU IP core to custom boards. Some demonstrations implemented on the ZCU 104 evaluation board were used to finish the presentation and show the employees, what is possible with the tools available.

In this week on Thursday there was a Xilinx ML seminar in Munich, where I went to with my supervisor to get more information about Xilinx AI solutions. This was a whole day event with several segments, showing off the capabilities and the work flow of Xilinx cloud and edge AI tools. This was also a great opportunity for networking and speaking in person to top Xilinx *Field Application Engineer (FAE)* engineers.

6. Week

All of the new information gathered at the Xilinx seminar last week needed to be transferred to the internal Wiki and properly documented. Three main tasks were worked upon in this week, namely preparing the second presentation 'Introduction to ML on FPGA', getting all of the DNNDK sample applications to work on the ZCU 104 evaluation board and evaluating a possible collaboration with an ETH start-up called Synthara.

- **Presentation:** Extensive market research has been conducted to find resources and ideas on how to present the different hardware platforms. The difficulty lies therein, that there are no standardized performance metrics for neural networks. Performance is strongly dependent on the network used and the individual use case. This leads to a lot of unfair comparisons, both in research and industry when numbers are shown. Therefore, a qualitative approach was chosen as doing all of these comparisons would have taken an extreme amount of work time and effort, requiring special hardware as well.

- **Bugfix for ZCU 104 evaluation board:** After reading through a vast amount of documentation and employing the help of online resources, mainly the Xilinx official forums, a solution was found to the problem. It turned out to be a specific problem of the ZCU 104 evaluation board which made it hard to track down. The solution to this was provided by an unofficial patch by on e of the Xilinx employees online. The board has power issues when running at full load resulting in the already mentioned problem of freezing the board in the middle of running the sample applications. After applying the patch all of the sample applications worked. These included image classification with resnet50 and inception-v1 as well as real time face detection, object detection and pose detection using other popular ANN

- **Synthara collaboration:** During research for neural network accelerator implementations I read about an ETH start-up providing this service in the form

of an ASIC. However, their prototypes as a proof-of-concept are implemented on FPGAs. Thus, we reached out to them and scheduled a meeting. During this meeting we discussed the possibility of a cooperation. The idea was to implement a demonstrator for the Embedded World 2020 conference showing off Enclustra hardware and using the Synthara neural network accelerator. The demo is a game of rock-paper-scissors played by a human player against a robotic hand. The setup can be seen in figure 6.1. The robot hand is controlled via USB using *Pulse Width Modulation (PWM)* to control each finger individually. The human players movement is captured by a camera connected via MIPI to the FPGA board. The FPGA handles all image preprocessing and is running a custom neural network capable of detecting hand gestures. The control signals for the robot hand are also given by the FPGA.

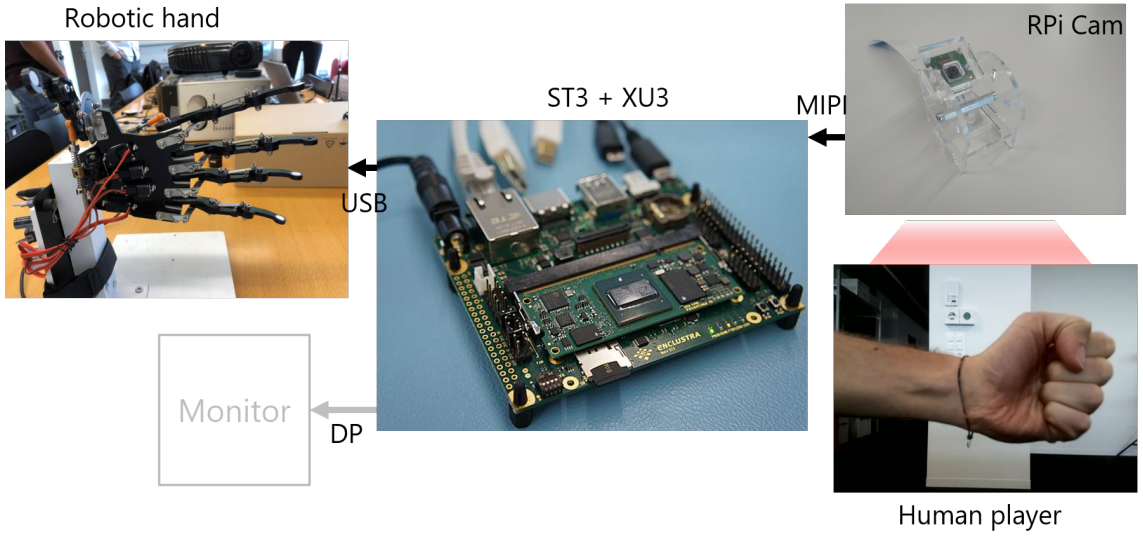

Figure 6.1.: Rock-paper-scissors demonstrator setup

7. Week

At the beginning of the week I held the second presentation 'Introduction to ML on FPGAs' in front of the assembled engineering employees. At the end of the presentation a live demonstrator was shown at my work place. The ZCU 104 was used as the platform to show off all of the ML examples provided by the DNNDK and upcoming questions answered. The rest of the week was spent doing in depth research on integrating MIPI into a Vivado block design. The reason for this is that Enclustra developed a new base board, the Mars ST3. This board features a MIPI *Camera Serial Interface (CSI)* connector allowing high-speed video streaming with up to four lanes and a bit rate of 2.5 Gbit/s per lane (or 2.9 Gbit/s depending on the chosen clock rate). The camera chosen was the Raspberry Pi camera with a SONY image sensor. As MIPI is not an open standard, research has been conducted into open source implementations of interfacing with the MIPI protocol. After discussing the open source alternatives with more experienced employees, those solutions were deemed unsuitable for the task at hand. The alternative solution is to use a Xilinx IP core which is available as a time limited evaluation license. The starting point was the ZCU 104 reference design, which included the whole MIPI IP core design. It consists of two main parts, the MIPI D-PHY and the MIPI Tx/Rx subsystem A high-level view of the Xilinx MIPI D-PHY IP core system is shown in figure 7.2. The communication takes place between a Master and Slave with one clock lane and up to 4 data lanes. This IP core allows proper communication on this high-speed I/O interface standard. The complete receiver subsystem is shown in figure **??**. The D-PHY IP core is part of this subsystem and in combination with the rest of the Rx subsystem allows the integration of a MIPI based image sensor and an image sensor pipe. The captured images can then be accessed via AXI interfaces. This part of the system needed to be integrated into the complete hardware design in Vivado consisting of the ZYNQ MPSoC, the DPU IP core and the usual peripheral interfaces (USB, DisplayPort, HDMI, etc.). On top of that, an embedded Linux OS needed to be built to control the applications and provide a working demonstration

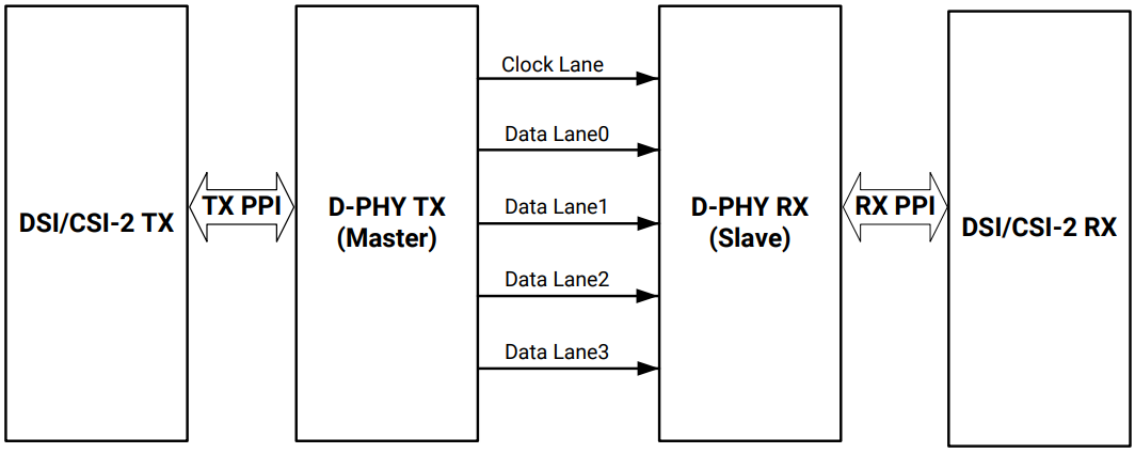

Figure 7.1.: D-PHY MIPI IP core overview [4]

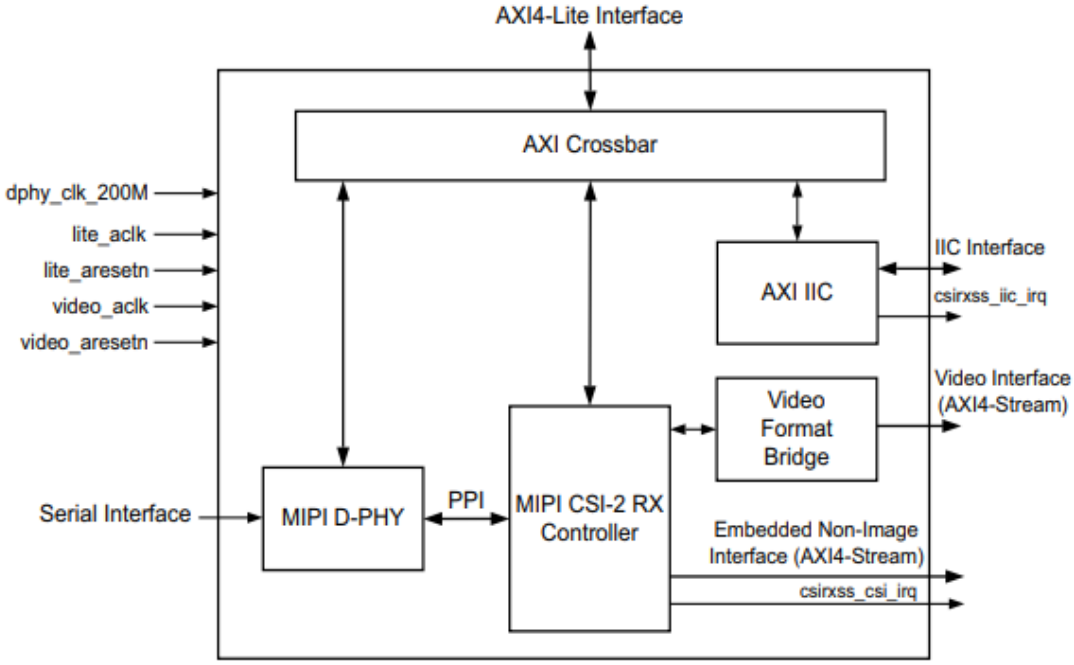

Figure 7.2.: Receiver MIPI IP core subsystem [5]

environment. The chosen Linux distribution for this task was the Xilinx supported Petalinux, which is itself based on Yocto.

8. Week

Figure 8.1 shows the reference design provided by Xilinx for the ZCU 104 board showing the complete hierarchy of the MIPI CSI block design with all additional blocks. These blocks are needed for further image processing. The data coming from the image sensor is in an unprocessed format called RAW. Therefore, an image processing pipeline needs to be integrated to convert this RAW image format into useable data in RGB format for example. The following Xilinx IPs are used to accomplish this task: 'Sensor Demosaic', 'Gamma LUT', 'Video Processing subsystem' and 'Video Frame Buffer Write'. The data transfer is handled by AXI streaming interfaces. Using the reference design as a starting point, the MIPI subsystem was integrated into the hardware design together with the DPU block. Afterward, Petalinux had to be configured and built. This enabled control of the whole system via an embedded Linux OS host system running on the ARM cores. Several steps are necessary for setting up a Petalinux environment:

- **Vivado hardware design:** First of all a working hardware design needed to be created in Vivado and successfully synthesized. This hardware design then needs to be exported in .hdf file format. This allows importing the hardware design as a template for the Petalinux system generation.

- **Creation of Petalinux project:** Petalinux is a command line tool for a Linux OS which abstracts away some of the details of building an embedded Linux OS. During generation of a new project, the previously used hardware design file is imported so that the system can access all of the implemented features.

- **Configuration:** In the next steps, necessary packages, user written apps, file system packages and custom modules can be added to the Petalinux project via console commands and a *Graphic User Interface (GUI)* environment simplifying interaction with all of the possible options.

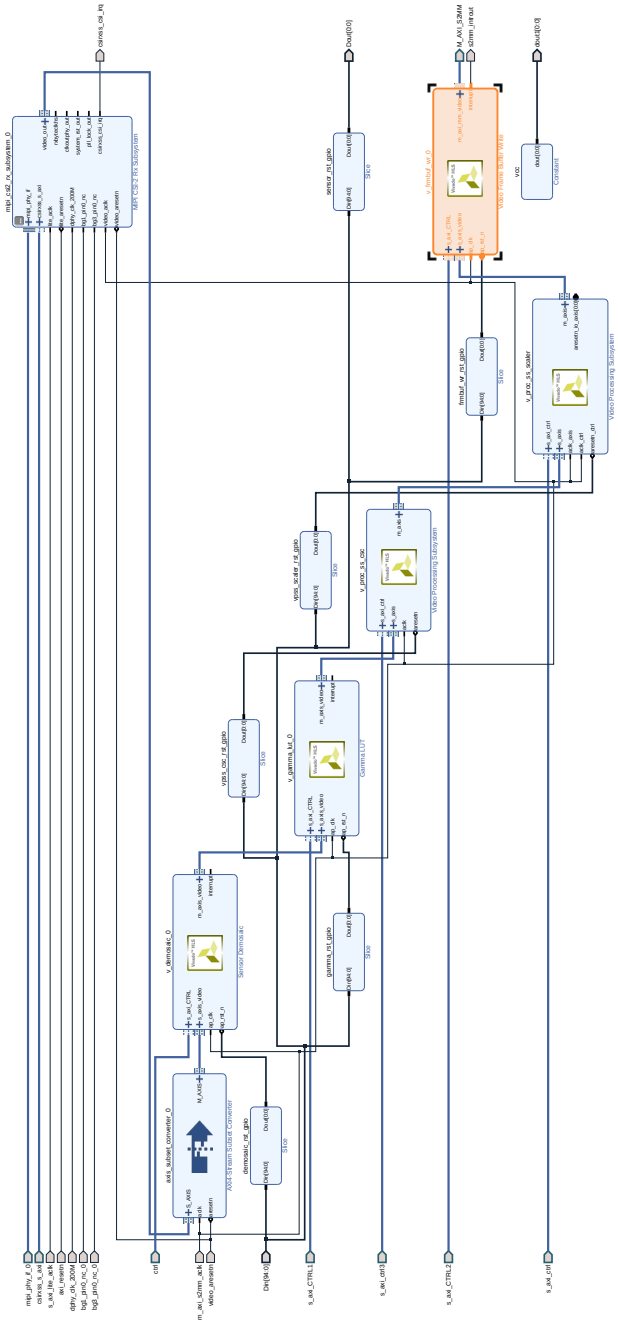

Figure 8.1.: MIPI CSI block design

- **Building the system:** After all of the system is configured, the necessary packages and files need to be downloaded and a root file system and kernel image constructed. This can also be done via console commands. After successfully building the whole system the necessary files need to be generated for the system to boot. This includes a BOOT.bin file, an image.ub file and the root file system. These files and directories are the minimum necessities for the Petalinux OS

To create a bootable image, an SD card is used and properly formatted. The SD card needs to partitioned into two primary partitions, BOOT formatted as FAT32 and ROOTFS, formatted as ext4. The Petalinux files are then copied over into the respective directories and the SD card can be used as the boot image for the FPGA board. Another Synthara conference call was due to further discuss details about the Embedded World 2019 demonstrator and a visit was organized to the company in order to create a schedule for the collaboration and labor division between Enclustra and Synthara.

9. Week

In this week a more closer evaluation of available neural network inference tools by Intel was done. The reason for this is the dependency solely on one FPGA supplier is not ideal. In order to be flexible with product design and familiarity with all of the tools on the market for neural network inference, it was decided that some time should be spend on evaluating alternatives to Xilinx. Figure 9.1 shows an overview

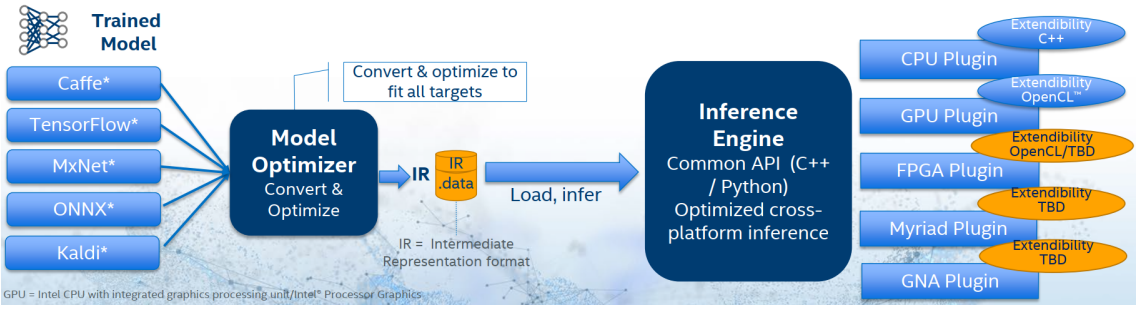

Figure 9.1.: OpenVINO toolkit overview [2]

over Intels OpenVINO toolkit. The workflow itself is similar to Xilinx DNNDK. The starting point is a trained model using popular neural network training frameworks (Caffe, Tensorflow, etc.). The trained network is then passed to a model optimizer which performs tasks such as quantization, stripping away layers only needed for training and other tasks. This operation is hardware independent. An intermediate representation is generated and passed on to the inference engine. This is a high level API allowing the implementation of neural networks on the target hardware. The main difference is its universal approach compared to the Xilinx DNNDK. Intel acquired the FPGA company Altera and took over their FPGA modules. Therefore, the OpenVINO toolkit does not only support FPGAs but also all of other product families Intel offers, such as CPUs, GPUs as well as other more specialized hardware. One of the main problems with Intels offering is its support of only one FPGA familiy, namely the Arria 10 FPGAs. Moreover, the model optimizer is not as powerful as

the Xilinx DNNDK one. No pruning is taking place and as of this date, there is no support for INT8 precision, only reduced precision floating point, which is not ideal. Therefore, the Xilinx approach is deemed superior. Integration and building of a custom Petalinux distribution continued this week with familiarizing myself with the overall workflow and possible debugging features. The DPU IP core was successfully integrated into the ZCU 104 reference design and synthesized. As the DNNDK tool is still in a beta phase, there are frequent updates. A major update was released this week. This updated version was investigated and documented in the internal Enclustra Wiki.

Bibliography

[1] Loise Crocket et al. *Exploring Zynq MPSoC: With PYNQ and Machine Learning Applications*. Strathclyde Academic Media, 2019.

[2] Intel. "AI on Intel - From data center to the edge - AI solutions using Intel architecture". 2019.

[3] Xilinx. *DPU for Convolutional Neural Network v1.2*. Ed. by Xilinx. Mar. 26, 2019. URL: https://www.xilinx.com/support/documentation/ip_documentation/dpu/v1_2/pg338-dpu.pdf.

[4] Xilinx. *PG202 MIPI D-PHY*. Ed. by Xilinx. July 2, 2019. URL: https://www.xilinx.com/support/documentation/ip_documentation/mipi_dphy/v4_1/pg202-mipi-dphy.pdf.

[5] Xilinx. *PG232 MIPI RX subsystem*. Ed. by Xliinx. Apr. 4, 2018. URL: https://www.xilinx.com/support/documentation/ip_documentation/mipi_csi2_rx_subsystem/v3_0/pg232-mipi-csi2-rx.pdf.

A. Workday reports