\item Good morning.

\item My name is Carlos Cano, I'm a student at the University of Málaga, and I am going to present in the following a design focus on convolutional Neural Networks for the Classification of Patients with Saccadic Eye Movements. Implementation on a FPGA Device.

\item Basically, I'm going to start with a brief presentation of our motivation, Next i am going to focus on review the main FPGA function briefly, the device chosen, their justification and the main objectives in our design.

\item follow by short exposition of how we developed the network in the logic part then i will underline the main results obtained as far as resource usage and evaluation time is concerned.

\item Furthermore, I am going to explain the way of joining CNN design in the hardware part with the processor part and the results obtained.

\item In the end I am going to expose a demonstration of my design.

\end{itemize}

\item this project is framed in a larger project which is focused on the development of a low cost portable device meant to evaluate the acceptance degree in the Spino Cerebellar Ataxia disease Type 2(SCA 2).

\item This disease is associated with to aditional comorbidities. The first problem is represented by the fact that it has no cure, only rehabilitation being possible. Therefore it involves a medical monitoring. What is more, this illness causes immobility, making it very dificult for patients to travel to the ambulatory.

\item This two reasons require a low cost and portable solution. This type of solution allows the early diagnostis of the patient, and the reduction of the displacement of them. Moreover, when possible, it allows the establishment of the diagnostic in the patient’s house.

\item This establishing involves a classification of the patients in three groups: healthy, presynmptomatic and sick. Taking into account this clasification we focused on two different development lines. The first one was to find the best algorithm, and we considerer that it was a very hard task due to the difficulty involved in separating the presimpthomatic patients from the healthy ones, by using convultional networks as a last aproximation. The second line refers to the development of a low cost portable platform to help in doing the clasification mentioned above.

\item My personal contribution in this project is centered on this second line. I made a desing in a low-cost portable heterogenous device for doing the clasificacion, also i study the viability and the competitivnes of this implementation.

}

\item FPGA is a hardware platform focusing on exploiting the parallelism of an algorithm. FPGA is formed by a large amount of little logic cells called CLB, many resources for the interconnection and other types of resources such as DSPs or RAM blocks.

\item In the following i am going to refer to the hardware implementation in terms of highlight the main differences with a software solution in order to understand better we need to point out what software implementation mean involve in contrast with hardware implementation.

\itemI want to start with a brief example of our point of view. On the left figure, it can be seen a simple algorithm with three consecutive operation, for the software solution we need to decompose in a lot of operation the majority of them for loading and storing value, in total 12 operations for doing this simple task. on the other hand, Hardware solution has three different block that can be work in only two times and maybe if this is a repeated task, hardware solution can produce one output in every iteration.

It is a clear example of how hardware can suit better than software solution.

The main problems regarding the FPGA design were represented by the long period of time needed for development and the complete pure sequentially algorithm.

As far as the first problem was concerned, language has represented the main challenge because it is a hardware language in which the designer has to specify the logic associate to each signal. In order to avoid this problem a new language has been launched in recent years, namely HLS from XILINX.

On the other hand, many designs have a sequentially part, which does not improve their efficiency and it consumes the majority of the resources in a FPGA. Many manufactures are developing heterogenous architecture, to take advantage on every platforms of Xilinx marketed ZYNQ devices, which includes two ARM Cortex A9 processor and a Logic parts.

These facts fulfill our interest in making developments quickly with HLS and distribute the workload between Processor part and logic part.

\itemI In software, the solution is decompose in a sequence of rules, due to the fact that it only has one big general purposed hardware block that processes the code instruction by instruction, however we always this of a solution in software algorithmic term but the real problem does not have a step by step behavior.

As for hardware it makes it possible to create any general structure able to find the most suitable solution.

i want to resume the main algorithm differences focusing on three processor problem in our opinion.

The first issue is the data access.

Therefore, hardware design avoids the multiple data access,, whereas software needs to write and than read every instruction processed, as you could be seen in the last slide. Those being the most recurrent but also the most longest operation. However in hardware we can connect block or data processor without any structure or resource that best fits with the solution, such as FIFO or stream Structure. get away all unnecessary operation like addressing.

\itemI The second problem is referring to the mathematical operation. Software needs a lot of time to process a group of basic operation because it has to do it one by one. So, if you have more processor you can do more operation but only because the number of processor is higher, they work in the same way. for example if we have to multiply two matrix of 30x30 with floating point representation, we have to do around 10.000 multiplications and 10.000 sums one after the other and also we have to fetch data and store. maybe around 40.000 clock cycles.

Apart from the comparison between software and hardware, there is another acquainted option, based on graphic card. However it has main three disadvantages, very expensive, high consume and large device.

\item in contrast, hardware solution has a lot of little mathematical processor, DSP, included for doing this type of operations, we can do in one iteration lot of the product and multiplication of two matrix of 30x30 can last around 100 clock cycles.

\begin{itemize}

\itemI Like I mentioned before, processors execute instruction one by one not making it possible to benefit of the parallelism of the real problem.

Hardware is able to implement different block that process the data in a concurrent form. Moreover each block is composed by many structures that execute the data in a parallel way too. For example we can split a big problem in six different parts and each part can be decompose in a set of different instruction that can be work at the same time. at the end of the design we can put a join block and all the design can be doing in a few cycles comparing to a software solution in which we have to decompose in sequential instructions.

This is for the main processor problem but i want to emphasize two of the big problems in FPGA design and how it has bee solved in recent years.

This problems regarding the FPGA design were represented by the long period of time needed for development and the complete pure sequentially algorithm.

As far as the first problem was concerned, language has represented the main challenge because it is a hardware language in which the designer has to specify the logic associate to each signal. In order to avoid this problem a new language has been launched in recent years, namely HLS from XILINX.

On the other hand, many designs have a sequentially part, which does not improve their efficiency and it consumes the majority of the resources in a FPGA. Many manufactures are developing heterogeneous architecture, to take advantage on every platforms of Xilinx marketed ZYNQ devices, which includes two ARM Cortex A9 processor and a Logic parts.

These facts fulfill our interest in making developments quickly with HLS and distribute the workload between Processor part and logic part.

\item Our project starts with a proposed CNN model. This architecture is formed by a convolutional layer with a ReLu activation function, followed by one max-pool function, connected to other convolutional layer, which has the same activation function and max-pool function. The output stage consists in two fully connected layers, the first with a ReLu activation function and the last layer with only 3 neurons and a SoftMax activation function. The next step is searching the best implementation for Keras.We choose Keras API because it offers us a tool for making both the description of the network and the training friendly, on a very high-level language. Due to its easy programing, we have been able to study different architectures with distinct numbers of layers and diverse types of training.

\item We need to check the functionality implemented in HLS, and we have to make a design layer by layer, with a simple operation. This fact is not possible with keras Therefore we consider MATLAB the best solution and every time we implement one layer we make a desing in HLS. trying different coding styles and applying HLS directives for optimizing the design.

In every iteration of our methodology we have to check the implementation.

\item The hardware optimization offers us three main optimizations in our view. The first optimization consists on suiting the data length depending on the necessities. Instead of software implementation, on Hardware we can use any length. For this purpose, we have considered to use fixed point with 8 bits for integer part and 10 bits for decimal part. This allows us to reduce the number of DSP uses and other important resources.

\item Other important optimization consists on removing memory access. this is a big problem in software solution due to the processor that needs a large amount of clock cycles to access to each data. In hardware design, we can connect different blocks with stream interface or we can only join the output one with the input, This is done by a dataflow directive in HLS.

\item The last main optimization is to benefit from the parallelism. The neural networks mainly the convolutional neural network has a strong parallelism behavior. The codding style has a big impact in this task.

\item I want to show a example of this optimization , in the second layer we have two loop, The outer loop processes one input in each iteration, this is done in a sequentially way in order to reduce the usage of resources. The inner loop is made by a pipeline role. Inside it there is a function split in 11 stages with one clock cycle in every stage. Another important optimization is represented by the time reduction multiplication function. Thus, we reduce 90 operation, in just 11 clock cycles. So the total time reduction is 115 instead of around 37000.

\item For the results, There are two main aspects that we worry about. The first one is represented by the resource usage

\item This table refers to the first aspect resources usage, we can see the usage of LUT, the most important resource, we have use only 61$\%$. This is a good mark because we are exploiting a large amount but we need another resources for putting CNN design in connection with the processor part and other resources. The other important mark is the usage of DSP, we use 81 per cent. This is a good result because we are very close to the total.

\item As for time efficiency, we compare the results with other portable solutions. We use two personal computers.In the table there are represented the features of each computer. We reach a reduction of time between 8 and 12 times. These are hopeful results for continuing with our purpose.

\item The first step is to export the HLS design to VIVADO project. HLS is only for creating IP block, but is necessary to export this IP block to VIVADO for connecting with other modules such as clock or reset signals or for connect with the io interface. In out design we implemented the weight in a separated memory ram for facilitating the parallelism, the input port and output port have been designed with AXI PROTOCOL, this is a standard bus for micro-controller and is the bus that xilinx chosen for communicate between block or with the processor. We have implemented in the same bus the control functionality for starting, configuration and checking the status of the ip block.

For making the design more optimize we add additional functionalists to the CNN basic implementation. We put a controller that it can be do the batch operation. we configure the IP block with lot of inputs, maximum 200, and then start the block, at the end we have only to take the outputs value when the block reports that the operations has concluded.

\note{

\begin{itemize}

\item The design diagram can be seen in this picture, In which it can observed the same IP block than in slide before with the number 5. The block with the number one is necessary for all design and it is for controlling the clock and reset signal, with the number two, the block is the processor, this is the only block that it wont be synthesized because it has already existed only we have to instantiate for put all the connection around it. number three is for multiplexing AXI BUS Interface, number 4 is the AXI BUS Controller RAM interface, we have to write weight on the RAM memory, and we use AXI interface too. at last, with the number six are the ram memories.

we could have put another type of connection for writing weights in FPGA but we consider this option one of the faster solution.

\item In this slide it can be seen the three main result of the implementation, in the up left there is the usage utilization, in this case we have use all BRAM block, it is not a bad point, because we have development all the desing and the tool has considered put in BRAM for improving the performance, and we use 85$\%$ of the total DSP, this is a good point also. For power consuming, the reports are normal, there are nothing to emphasize, all parameters are in the normal range. in the figure down, there is the time report, all parameters are in blue, not in red so this values are good enough for our design. however if there were red values there would not be a problem, only we would have to find the location and consider if there was a real problem.

\item After Synthesis we have to do the implementation. Implementation is the process in which Vivado transform the hardware specification in a real map of resources. In the figure can be show a real implementation of our design. In blue are all the logic resources usage. in orange is the processor part.

After this Vivado generate a bitstream file, for configure the FPGA. We can configure it with this program or as in our case, we can export this file, and generate another files and create the code in SDK.

\item Vivado SDK is derived from eclipse IDE and have the majority of the option, and another for put the fpga bitstream in different place, such us SD CARD, also have a terminal for communicating with the processor, we can use this easily.

\item Our code is formed by a few function, we configure the IP block through the AXI interface, then we put all inputs in the ram memory at the same way that put the weight and the configured bytes, the rest of the code is for obtained the intermediate results and the final results. In the next slide the real working of this code can be seen.

The intermediate outputs are needed in order to make the training in the next part of this project.

\item All the results match with the matlab implementation. However i am going to explain with more details in the demonstration part.