Handwritten Character Classification: Hardware-Software Co-Design Approach

António Moisés Dias
MEEC
FCT-UC
Coimbra, Portugal
2015232789

Afonso Viana de Lemos
MIEEC
FCT-UC
Coimbra, Portugal
2012169773

I. Introduction

In recent years there has been an increasing interest in Machine Learning (ML) and Artificial Intelligence (AI). The core feature of ML and AI projects/solutions that are Neural Networks (NN). One famous type of NN in Image Processing projects is the Convolutional Neural Network (CNN) due to its capacity of taking advantage of the data inherent properties. These CNN are widely used in a broad context of solutions like face detection, self driving cars, biometric authentication and handwritten character recognition.

CNNs can be extremely useful and are becoming essential to improve everyday life technologies, however, there are some constraints that can make CNNs unusable or not used to full potential. Three important ones are performance, cost and power.

Likewise, Hardware-Software Co-Design has been getting more attention, although it has been present for more than three decades [1]. The challenges of technology in our era demand constant creativity and innovation to improve solutions in various ways. May those be performance, power consumption, longevity, cost, to name a few examples. In this context come to play Hardware-Software Co-Design solutions.

This project aims to adapt a simple CNN into a Hardware-Software Co-Design solution, taking advantage of the customization of the Altera DE-115 Field Programmable Gate Array (FPGA) board and a Nios II embedded processor. The main objective is to classify handwritten digits using a CNN trained with the Modified National Institute of Standards and Technology (MNIST) dataset. The digits are captured using a TRDB-D5M CMOS Camera, connected to the FPGA board and then forwarded to the CNN, which runs in the Nios II processor side.

II. State of the Art

The state-of-the-art in Convolutional Neural Network (CNN) classifiers on Field Programmable Gate Arrays (FPGAs) involves exploring hardware and software-based approaches, as well as hardware/software co-design, to achieve high performance and low power consumption.

FPGAs have become a popular platform for accelerating CNNs due to their ability to perform computationally intensive tasks while also offering high level of flexibility.

Hardware-based approaches aim to design custom hardware accelerators specifically optimized for CNNs to achieve high performance and low power consumption. Through software optimizations and the use of high-level synthesis tools, software-based techniques take advantage of the parallelism present in the convolution operation to achieve great results.

Hardware/Software co-design approaches combine the strenghts of hardware and software approaches to achieve an optimal balance between performance and energy efficiency. This involves designing hardware accelerators that exploit parallelism and software optimizers that reduce memory accesses to improve the energy efficiency.

Some recent state-of-the-art works on Convolutional Neural Network classifiers on FPGAs are:

1) Hardware approaches:
   - Researchers proposed an accelerator for CNNs using a dataflow architecture with reconfigurable on-chip memory to achieve high-performance and low-power. The architecture includes an on-chip memory hierarchy and a flexible dataflow that enables efficient parallel processing.
   - Researchers presented a high-throughput and low-latency accelerator that exploits weight pruning, weight sharing and data reuse. The proposed accelerator uses a novel interleaved weight storage scheme that reduces the memory footprint and enables efficient parallel processing of multiple layers.
2) Software approaches:
   - Researchers proposed a software-based FPGA accelerator for CNNs that leverages layer-level parallelism to achieve high performance. The approach uses a layer-level parallelism framework that enables efficient parallel execution of multiple layers, and a layer-level pipelining scheme that reduces the latency.
   - Researchers presented a software implementation for CNNs on FPGAs that uses a high-level synthesis tool to automatically map the CNN to FPGA hardware. The proposed software implementation is based on the OpenCL framework and leverages the parallelism inherent in the convolution operation to

achieve high performance.

3) Harware/Software co-design approaches:
   - Researchers proposed a co-design approach that combines hardware and software optimizations to achieve high performance and low power consumption. The proposed co-design approach integrates hardware acceleration and software optimization to achieve an optimal balance between performance and energy efficiency.
   - Researchers presented a co-design approach that integrates hardware acceleration and software optimization to improve the performance and energy efficiency of CNNs on FPGAs. The proposed co-design approach includes a hardware accelerator that exploits parallelism and a software optimizer that reduces memory accesses to improve the energy efficiency.

A variety of hardware- and software-based strategies, as well as hardware/software co-designs, are used in the Convolutional Neural Network (CNN) classifiers using Field Programmable Gate Arrays (FPGAs), with the goal of obtaining high performance and low power consumption. In order to map CNNs to FPGA hardware, the emphasis is on taking use of parallelism, minimizing memory accesses, and adopting high-level synthesis tools.

Due to their capacity to carry out computationally hard tasks while providing a high level of flexibility and customization, FPGAs have generally become a popular platform for accelerating CNNs. Researchers are constantly looking for new and creative ways to enhance the performance, power efficiency, and flexibility of CNN classifiers on FPGAs as the state-of-the-art in this sector continues to advance.

III. Context

The baseline for this project are two different solutions presented below:

A. MNIST Classification - Hardware Solution

This solution discusses a low-power implementation of a convolutional neural network (CNN) on the Intel Cyclone IV FPGA to perform the task of handwritten digit classification. A sliding window buffer is implemented using the FPGA fabric and on-chip memory to minimize energy-expensive off-chip memory accesses, and a small, optimized GPU-like multiplier-accumulator (MAC) core computes all of the activation layers [2]. The handwritten digit is given by an image captured in real-time by a D8M Camera connected to the board (DE2-115 FPGA). The developed system is hardware only and designed in System Verilog HDL.

As expected, the results of this development are very good: low power, low cost and high efficiency. The implementation even uses less than 50% of the Cyclone IV processor [2]. However, there are some drawbacks. The complexity of the design, engineering effort to develop such a focused system and the lack of flexibility to adapt to different scenarios are the major ones. The project also takes a considerable amount of time to compile $\approx$ 40 minutes.

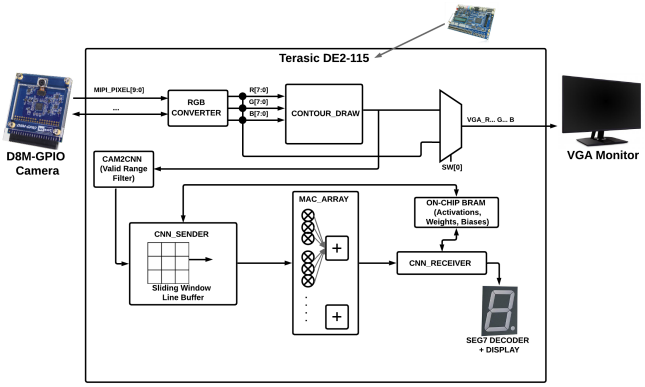

Fig. 1. MNIST Classification - System Architecture

B. MNIST Classification - Software Solution

Divergently, this second project implements a CNN only using the Nios II processor. The CNN is designed in C/C++ and the handwritten character is drawn by the user on a 800*480 LCD Touch Panel connected to the board [3]. This solution is also low cost, but easier to compile and to adapt to different circumstances. It is flexible in changing the CNN parameters, or even its pipeline.

There are also drawbacks in this implementation. It does not work in real-time, it is not so efficient, it is more resource demanding. The identification of an handwritten digit takes $\approx$ 20 seconds on average.

IV. Hardware-Software Co-Design

The main objective of this project is to implement a useful hardware-software co-design to take advantage of the qualities of these two frameworks. It is known that hardware is faster and low power but more complex to design. Software is more friendly but more power demanding and slower. Upon further discussion and investigation, the design of this project took the following architecture:

The pipeline is described below and illustrated in Fig. 3:

A. Hardware

a) : D5M CMOS Camera - The camera captures a 640x480 image. This resolution was configured to be enough to get a quality image without much processing.

b) : Greyscale - The greyscale module takes the RGB values of each pixel and polarizes into black or white according to a threshold.

c) : VGA Select - The VGA Select module allows the user to switch between an RGB image and a Greyscale one to be seen on the VGA screen.

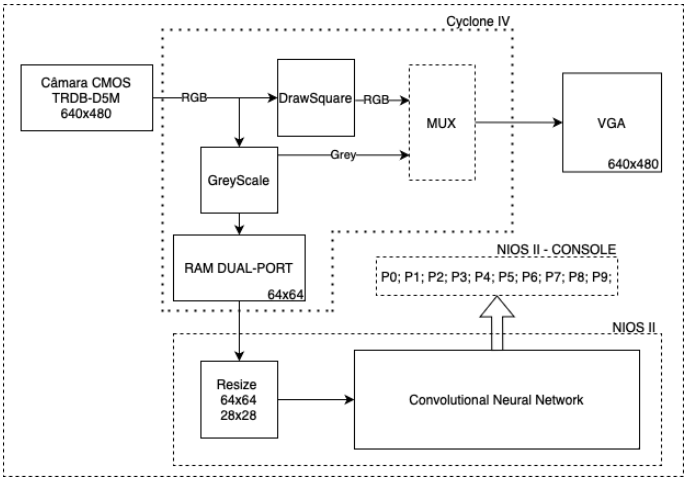

Fig. 2. Architecture of Hardware-Software Co-Design

d) : Dual-Port RAM - The DP RAM was design to save a 64x64 greyscale image which corresponds to the area of interest (image to be classified in the CNN). The memory addresses are being controlled by the Pixel Row(y) and Pixel Column(x) values given by the Camera module. Since the RAM is linear and the image is not, the columns and rows are combined as $address = y[5..0]x[5..0]$.

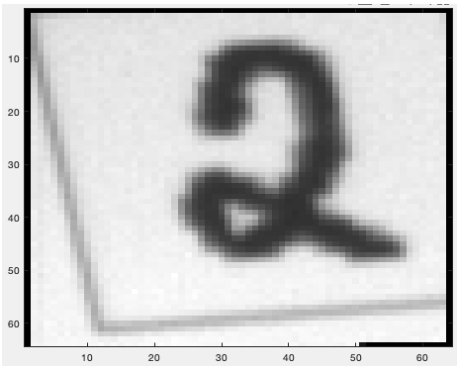

Fig. 3. 64x64 Matrix of the Input Image

B. Software

a) : Resize - In the C/C++ script the image is read from the Dual Port RAM and resized into a 28x28 image which is the input of the CNN.

b) : The CNN we use here is designed for handwritten character recognition, using the MNIST dataset as the training data. The network consists of multiple layers that process the input image in a hierarchical fashion to extract meaningful features and make a prediction about the class of the input image, from 0 to 9;

The first two layers of the network are convolution layers that use filters to extract features from the input image. These features are then down-sampled using max-pooling layers to reduce the size of the feature maps and reduce computation costs. The output of the pooling layers is then flattened and fed into two fully-connected (dense) layers that make the final prediction about the class of the input image.

The activation functions used in the network (relu()) introduce non-linearity into the model, allowing it to learn complex relationships between the input image and the target classes. The use of multiple convolution and fully-connected layers allows the network to learn increasingly complex representations of the input image, enabling it to make more accurate predictions.

The architecture we used was the following:

- Input image: The input image is a 64x64 greyscale image that will be processed by the CNN.
- Convolution Layer: This layer performs a convolution operation on the input image. It uses 3x3 filters with 1 input channel and 8 output channels. The convolution operation extracts features from the input image and applies the filters to each local region of the image. The relu() activation function is applied to the result of the convolution operation.
- Max-pooling Layer: This layer performs a max-pooling operation on the output of the previous convolution layer. The max-pooling operation down-samples the feature maps by taking the maximum value in each 2x2 region of the feature maps. This helps to reduce the size of the feature maps and reduce computation costs.
- Convolution Layer: This layer performs another convolution operation, similar to the first convolution layer, with 3x3 filters, 8 input channels and 4 output channels. The relu() activation function is applied to the result of the convolution operation.
- Max-pooling Layer: This layer performs another max-pooling operation, similar to the first max-pooling layer, to down-sample the feature maps.
- Flatten (reshape): The output of the previous max-pooling layer is flattened (reshaped) into a 1-dimensional tensor to prepare it for the fully-connected layers.
- Fully-connected Layer: This layer is a dense layer that performs a matrix multiplication between the input and a set of weights, followed by a bias addition and relu() activation function. The number of neurons in this layer is 32.
- Fully-connected Layer: This layer is the final layer in the CNN and performs a matrix multiplication between the input from the previous layer and a set of weights, followed by a bias addition. The number of neurons in this layer is 10, which corresponds to the number of classes in the MNIST dataset (0-9). The output of this layer represents the predicted class of the input image.

Overall, this CNN architecture is a simple yet effective approach for handwritten character recognition, and its abbility to learn meaningful features from the input image makes it a powerfu,l tool for image classification tasks.

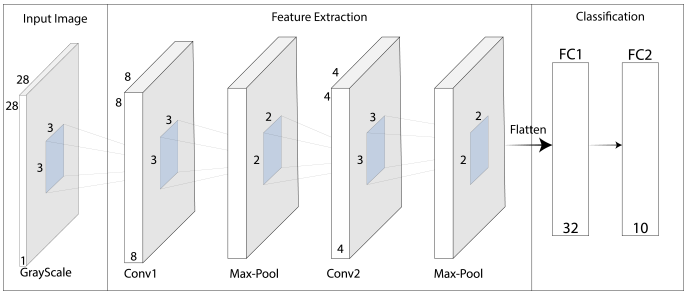

Fig. 4. CNN Architecture

V. Results

The approach to this project tried to take advantage of the best qualities of both the Hardware and the Software solutions overviewed in chapter III. Table I displays different metrics about some characteristics of all three approaches. This metrics were all tested in the DE2-115 FPGA.

Design	Compilation Time	Complexity	Process Time
Hardware only	40 min	HIGH	1 s
Software only	5 min	LOW	20 s
Hardware/Software	7 min	MEDIUM	12 s

TABLE I
METRICS COMPARISON

All three designs have both attractive and unattractive metrics. The hardware only solution is the best for real-time handwritten digit recognition, however the complexity during development is high and each iteration of development takes quite some time to compile. It is also not suitable to change, for example, the size of the image to be collected. Each change to the project brings long time and effort to implement, due to the low-level description of functions and features.

The Software design is clearly easier to develop and has more flexibility to change. It is mainly programmed in C/C++, which is widely used in various implementations on different machines and systems. Because this is a high-level programming language, it also uses a lot of resources to perform tasks, even the simpler ones. The execution of the code is also sequetial, so there is no parallelism happening during runtime.

The Co-Design approach deals with image processing in Hardware and the CNN in Software. It has better results than the hardware only project in terms of complexity and effort and compilation time and is also a bit more complicated than the software only approach. It trades off complexity for process and compilation times. This design could be optimized for better process time results, there is room for improvement.

VI. Conclusion

The results of the comparison show that each approach has its own advantages and disadvantages. The software-based approach is the easiest to code, but takes the longest to process an image. On the other hand, the hardware-based approach is difficult to code and has the longest compilation time, but has the shortest processing time.

The hardware/software approach is a compromise between the two, with medium coding difficulty and an average compilation time and an itermediate processing time. This approach may be suitable for applications that requeire a balance between processing speed and development time.

In the end, the method of approach will be determined by the demands and limitations of the application. The hardware-based solution can be the best option if processing speed is the most crucial component. However, the software-based method might be preferable if development time and coding simplicity are more crucial. For applications that need to strike a balance between processing speed and development time, the hardwware/software method might be a useful option.

References

[1] Jürgen Teich. Hardware/software codesign: The past, the present, and predicting the future. volume 100, pages 1411–1430. Institute of Electrical and Electronics Engineers Inc., 5 2012.

[2] Grant Yu. MNIST_Classification_FPGA. https://github.com/grant4001/MNIST_Classification_FPGA, 2020. [Online; accessed 2022/23].

[3] Song Jiahao. CNN_Nios. https://github.com/RunTimeError2/CNN_Nios, 2018. [Online; accessed 2022/23].