FILTER ARCHITECTURE FOR IMAGE PROCESSING USING NIOS II PROCESSOR AND FPGA

CARLOS C. ALMEIDA
ccaetanoa@gmail.com

VORNEI A. GRELLA
vagrella@gmail.com

ALEXANDRE T. OLIVEIRA
tomazati@gmail.com

EURÍPEDES G.O. NÓBREGA
egon@fem.unicamp.br

DENIS S. LOUBACH
dloubach@fem.unicamp.br

Advanced Computing, Control & Embedded Systems Laboratory / FEM
University of Campinas - UNICAMP
Campinas, São Paulo, Brazil

Abstract. *This research work introduces a digital signal processing architecture implemented through a field programmable hardware device using ConvNets, which are multiple layer networks containing convolution filter algorithms. Among several possible applications, 2D convolution network image processing may take advantage of the parallelism enabled by FPGA. In this sense, the purpose of this paper is to present a FPGA-based implementation of the ConvNet central part employing NIOS II Altera processor. Comparing the quality of the hardware-processed images with a MATLAB (software-processed), equivalent result motivates the discussion about the architecture adoption for the hardware image processing embedded system development.*

Keywords: *FPGA, Convolution Networks, Image Processing, NIOS II Processor.*

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering
P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

1. INTRODUCTION

The evolution of electronic devices has brought progress in the implementation of reconfigurable and programmable integrated circuits, such as FPGA (Field Programmable Gate Array). Basically, it is made of a number of programmable modules, and can be configured in the field, emulating the behavior of any digital circuit. These electronic devices are often used in embedded systems to implement software algorithms in hardware in order to execute considering a given parallelism level. This capacity has been explored targeting software solutions that demand high computational performance and/or have high level power consumption, thus making it feasible and conforming with real time performance and power constraints. Additionally, circuits may be prototyped and tested in advance. It means that using FPGA is an alternative to the Von Neumann's rule, which stands that each processor core only executes one instruction a time (SEBESTA, 2012).

Using FPGAs in embedded systems allows achieving both software flexibility and hardware speed. Nowadays, FPGA has posed itself as one of the main tools for the reconfigurable computing area. Currently advanced FPGA solution shave the possibility of implementing digital circuits using not only basic blocks, but also specialized ones like video controller circuits, PLLs, RAM and other possibilities (ALTERA®, 2011d, 2010a).

On the other hand, image processing and computer vision evolved in the last decades to the point that commercial applications are becoming economically attractive. However, the complexity of those applications like scene parsing has directed the research for efficient algorithm implementations. It means FPGA adoption in the pursuit of the parallelism performance or to use GPU (Graphic Processing Unit) (RAIZER ET AL, 2009).

One method that has driven attention lately is the CNN (Convolutional Neural Networks). CNN is also known as ConvNets. It may cover a range of applications like gene alteration (Ning et al, 2005) face detection and recognition (Garcia and Delakis, 2004; Lawrence et al, 1997), handwriting translation (LeCun et al, 1998), scene parsing (Farabet et al, 2012, 2013), and others. A number of methods using FPGA to implement computer vision applications can be found in the literature considering the last years, since the work of Farabet (2011).

In this context, this paper presents a first step regarding an architecture study for efficient hardware digital implementation of the ConvNets central part, the discrete convolution algorithm. A ConvNet is required to solve an image problem when a big image matrix has to be processed (512 x 512), usually also implying in an accordingly number of multiplications. Therefore, any time reduction for the convolution central processing unit, a multiplication and addition in a 2D or3D matrix configuration, will imply a significant reduction for the whole image processing time. Besides, the possibility of online reconfiguration of the hardware solution introduces new aspects of flexibility to ConvNet applications.

2. CONVOLUTIONAL NETWORKS

A ConvNet is a neural network consisting of several stages that have repetitive processing modules involving basically two functions that are respectively a convolution and a subsampling. At each stage there are a filter bank module, a nonlinear function calculation

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

and a spatial pooling module (LeCun et al, 2011). The filter bank and nonlinear function correspond to the following transformation, which is similar to a traditional artificial neural network.

$$y_j = \varphi\left(b_j + \sum_{i=1}^{n} k_{i,j} * x_i\right) \quad (2.1)$$

Where each input x_i and output y_j are a 2D matrix of dimensions $n_1 \times n_2$ and $m_1 \times m_2$, including the filter and the nonlinear $\varphi(\cdot)$ function, and $k_{i,j}$ is the filter kernel used in the convolution operation represented by the "*". The input matrix corresponds in the first stage to the raw image matrix, and the output matrix is called feature map. This processing is a searching for some desired characteristic that may distributed occur in the image, and that reflects on the size and elements of the kernel $k_{i,j}$. In general, an important problem in object recognition is how to decide about a pixel taking into account a wide context. The pooling stage analyses a neighborhood for each element in each y_j matrix to continue the processing of each $m_1 \times m_2$ matrix under processing, configuring a reduced feature map, which corresponds to a subsampling. For each kernel $k_{i,j}$ there will be n_3 feature maps and also n_3 reduced feature maps, which are the number of different kernels to be used in the first processing of the raw image. After several different stages of layers of this same processing, however with different configurations for each layer, a classified image is the final result.

It may be realized that the convolution are the filter basic processing unit, responsible for the main computational cost of the whole process.

To get acquainted to a complete architecture for a ConvNet application, a programmable convolution network processor (CNP) developed by (Farabet et al, 2009) is presented in Figure 1. It can be analyzed as a SIMD (Single Instruction, Multiple Data)processor. It has a vector instruction containing elementary ConvNet operations.

Such instructions are hardware optimized. Then, the implementation of a CNP consists in reprogram the elements of the processor layers without the need to change the logical configuration of the FPGA. Figure 1 shows the modules that compose a CNP.

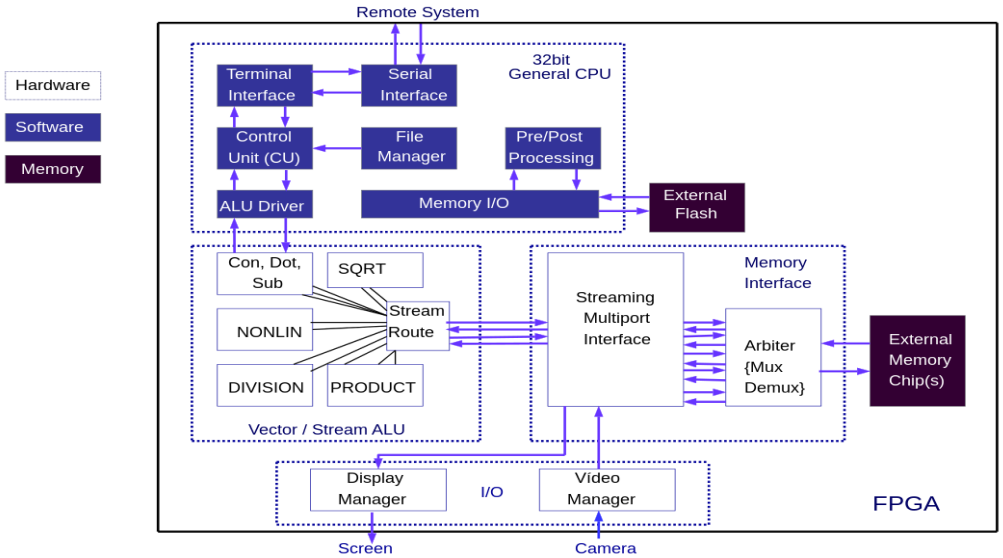

Figure 1. Architecture of a CNP - Adapted from (Farabet et al, 2009)

It may be noted in Figure 1 the "camera" as an input data and the "screen" as output. Access to external memory is made by an arbitrator, which selects the priority access for both reading and writing data in memory.

The driver Input/Output Memory provides a given latency access to the external memory, which helps CNP to have full access to read/write the same memory used by VALU (Video Arithmetic Logic Unit)interface.

The control unit is based on a state machine. A difference to a complex control unit is that the calculations are not performed by the CPU (Central Processing Unit), but by the VALU that can asynchronously read/write data from/to the interface .

The LeCun method (LeCun et al, 2011) works with the use of seven layers, three convolutions, three subsampling and two finishing. The operation is illustrated in Figure 2.

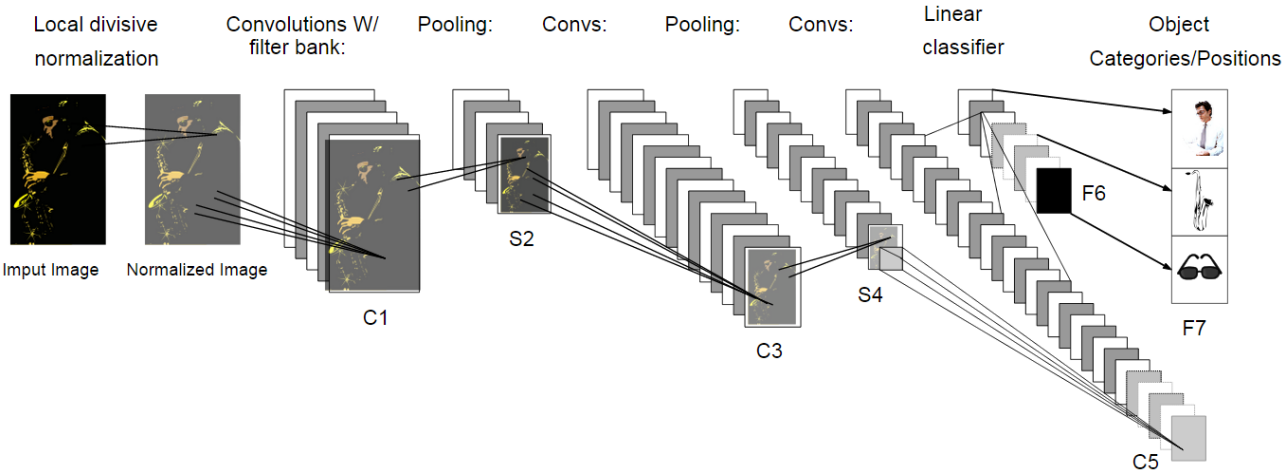

Figure 2. Example of ConvNet – Adapted from(LeCun et al, 2011)

As mentioned before, the ConvNet consists of multiple stages. The input and output of each stage are the feature maps, which consist of sets of arrays (LeCun et al, 2011). The ConvNet in Figure 2 is briefly explained in the sequence.

The first layer (C1) receives the image to be processed, following their normalization, and generates six desired convolution kernels using the input image, thereby producing six maps characteristics.

The second layer (S2) generates a grouping and subsampling of each map from the previous layer.

The third layer (C3) computes characteristics generated by high levels of convolution filters and results summation, thereby generating new maps.

The next layer (S4) is the same basic procedure for S2, but now for the generated maps C3.

The layer (C5) is the same C3 procedure, however using the maps generated by S4. In (F6), the generated map is a map of activation peaks for each category. The F7 layer simply combines the above categories into a single result.

The final results are several separated objects seen in the image, respectively attributed to a given class and positions.

3. APPLICATION AND MATHEMATICALFORMULATION

The use of the convolution is applied to image processing, considering the application of different filter kernels to the image data to extract specific characteristics, for example edge detection or increased sharpness.

The majority of the computational effort of a CNP machine is to calculate the 2D convolutions. Therefore, the convolution computation in parallel fashion was adopted, performing the respective sum of multiplications.

The basic mathematical formulation used for the digital convolution of the image filters are expressed as follows:

$$z_{ij} = y_{ij} + \sum_{m=0}^{K-1} \sum_{n=0}^{K-1} x_{i+m,j+n} w_{mn} \qquad (3.1)$$

where $x_{i,j}$ is the value of the position i,j of the 2D matrix concerning the input figure, W_{mn} is the value of the position mn matrix referring to the kernel filter K x K, $Y_{ij}$is the value in the plan which will be combined with the result and $Z_{ij}$is concerning the position of the array output plane.

Figure 3 illustrates the procedure for convolution, used as basis for the implementation of the corresponding algorithm.

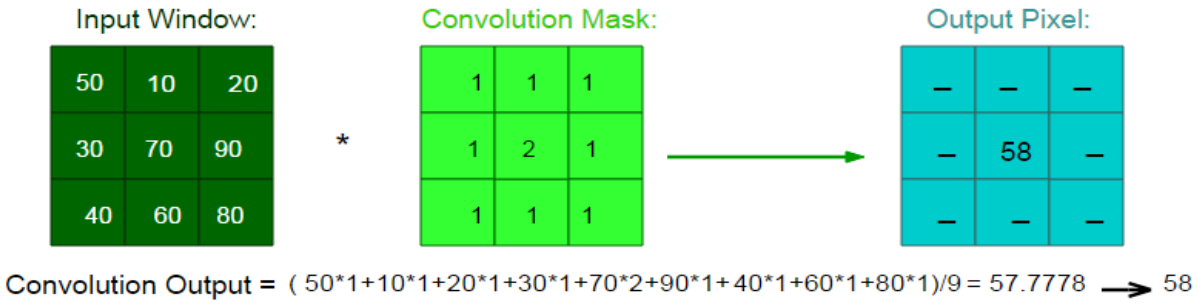

Figure 3. 2D Convolution

Still in figure 3, one can notice the convolution between an image sub-matrix and a given kernel. Considering that the kernel size is represented by a 3x3 matrix, a set of nine pixels, which is part of the image data matrix, will be multiplied by the corresponding kernel. The convolution is completed through processing repeatedly until the kernel has passed on all possible pixels of the source matrix. The final result of the convolution is a filtered version of the image matrix, according to the implicit filter transformation.

4. FIELD PROGRAMMABLE GATE ARRAY

FPGA is a programmable logic device that uses a network of logic gates and programmable keys. It allows the creation of a large number of digital circuits by connecting

these components together. The programming is performed by the customer and not by the manufacturer.

According to (Altera®, 2011d, 2010a), the modern FPGA permits the use of additional circuitry with specific functions that are commonly found in digital circuits. Examples are counter performance, data bus, JTAG, PLL (Phased Locked Loop) and memories.

Reconfigurable systems on a single chip are increasingly used in embedded form, due to the performance and speed that these devices present, and also because of the programmability through modern hardware description languages (PERRY, 2002; ORDONEZ ET AL., 2006).

Embedded processors in FPGA's can be used to prototype a specific application due to its customization and programming flexibility, yielding increased performance and significant cost savings, and permitting hardware consolidation as an ASIC production chip (ALMEIDA, 2015; RAIZER, 2010; TOMAZATI, 2009).

4.1 Development Tools

The research work described in this paper was developed using a DE2i-150 development kit based on the Cyclone IV FPGA from Altera. Figure 4 shows this kit. This board has multiple devices, peripherals and interfaces such as RS232, HDMI, Audio Input, SD, Memory, VGA, Ethernet (Altera®, 2011d, 2010a). Those peripherals and interfaces, present in a single board, facilitates the development of embedded systems.

Figure 4. Development Kit Altera Cyclone IV-150 DE2i

The following software tools from Altera were used to program the DE2i-150 kit:

- QUARTUS II v13.1 SP1: used to compile the HDL code and to synthesize the described hardware.
- QSYS (Altera®, 2011f): Used for the development and configuration of the NIOS II processors and its peripherals.
- NIOS II Software Build Tools for Eclipse: programming environment in C/C++, which allows to create software that will run in NIOS processors, compiled by QUARTUS II 13.1.

4.2 NIOS II processor

The developed system was designed to control all peripherals required by the present application, which has to be included in the basic core architecture. The adopted Altera NIOS II (Altera®, 2010b) is a 32-bit soft-core processor, which allows three different core version implementations: economic, standard and fast. This research works has used the fast version.

Using the QSYS tool, an extension of the QUARTUS II (Altera®, 2011f), the elements present in the processor (for example *sysid*, *jtag_uart*, *onchip_memory*, *sdram_controller*, *pll_DE2115*, and *performance_counter*) are instantiated one by one and connected coherently, as shown in Figure 5. This figure also illustrates the configuration process and presents all the devices connection with the NIOS core. It is allowed to the user to customize the complete processor.

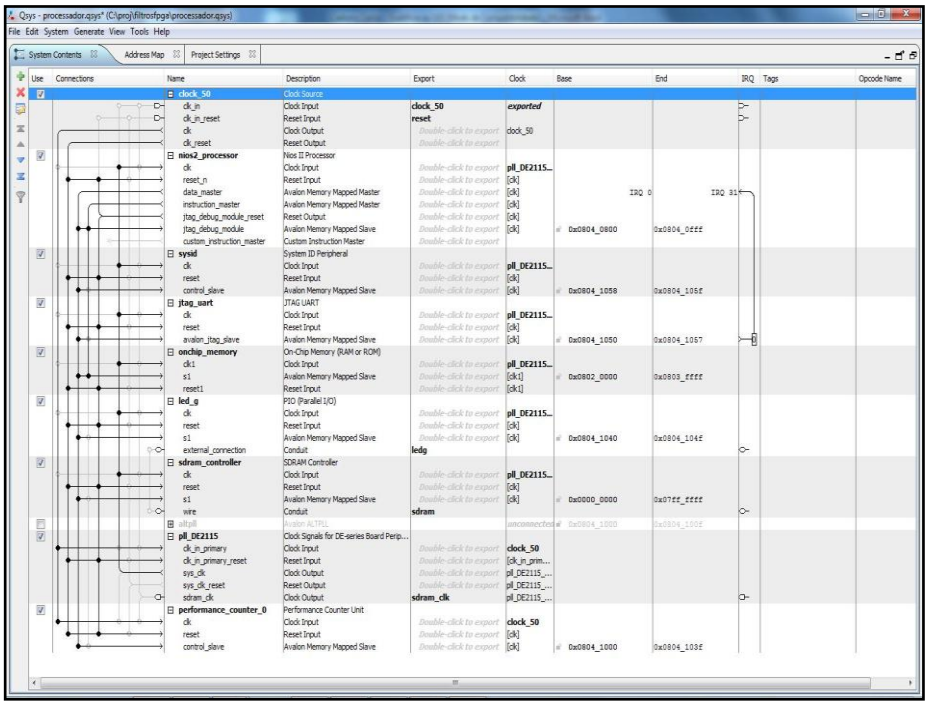

Figure 5. Processor Implementation in QSYS

It is necessary to get familiarized with the operation of all available devices to achieve a workable hardware configuration in QSYS, using the System Interconnect Fabric Avalon Bus (Altera®, 2011a).

5. ALGORITHM IMPLEMENTATION IN FPGA

The algorithm developed in FPGA works with a custom processor and performs convolution calculations required to obtain a Gaussian filter with the purpose of softening the image.

5.1 Dataflow and ACD convolution module

Firstly, the video signal or image is acquired and stored in the memory peripheral, reaching the CNP processing system where data will be processed by convolution network. After that, processed data are displayed.

The pipeline developed for CNP processing was performed using stages, representing the generic description of a set of logic elements and generic components. This hardware technique allows overlapping execution of multiple instructions, where some amount of buffer memory is used to store instructions before being executed.

Figure 6 shows the implementation of a custom module used to perform calculations of the convolution algorithm. The process ends only after the integration of the NIOS II processor through the creation of interfaces for the Avalon bus inside the module.

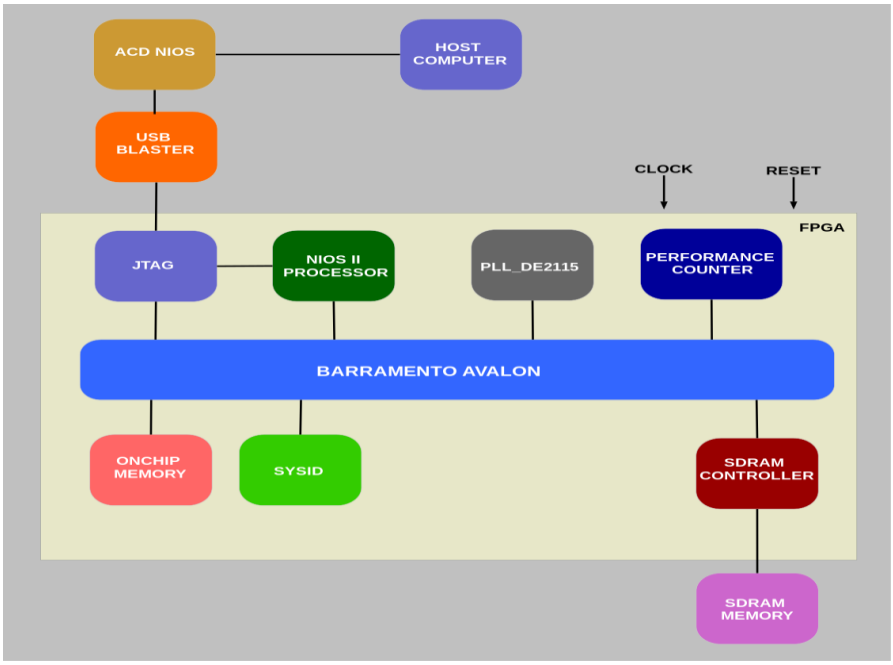

Figure 6. Dataflow with Processor, ACD module implemented in NIOS II and Avalon Bus

During the circuit development, some peripherals were used. The Avalon bus is responsible for all communication between the processor and other devices.

In Figure 6, it is illustrated the implementation breakdown of system blocks CNP module hardware (in VHDL) running and their data streams.

Following, it is presented a brief description of each peripheral processor used.

ACD NIOS is the Discrete Convolution Algorithm (ACD) circuit responsible for processing the acquired image, making the image convolution in a given kernel.

JTAG is a peripheral used to allow debugging and downloading the developed software into the processor memory.

NIOS II Processor is used to perform the execution of each instruction. Fast core version was used for maximum performance efficiency, as mentioned before.

SDRAM Controller is responsible to control the dataflow between the SDRAM memory and the peripherals. It covers 32 MB of memory.

SysID has the purpose of identifying the processor and others components of the system. The identifier can be generated automatically or manually.

Performance Counter is a peripheral used to measure time between two or more events.

PLL_DE2-115is an external component developed in QSYS. It is responsible for deriving the 50 MHz clock signal, then generating other clock signals used in the system.

On-Chip Memory is a memory module used to store data and instructions used by the processor. It is located inside the FPGA.

SDRAM (Synchronous Dynamic Random Access Memory) is an external memory module. It is a dynamic random access memory (DRAM), and it is synchronized with the system bus.

USB-Blaster is responsible for data transfer between the PC (Personal Computer) and Altera devices via USB.

Data, such as image or video, is passed to a *Shift Register*. This information is simultaneously stored in an image matrix and into a RAM, then allowing parallel processing.

The image matrix is convolved with the kernel matrix, which has the function of a particular desired filter. The image matrix has the input image data, displacement of this information and the convolution process occurs simultaneously. It is possible due to the parallelism enabled by the FPGA-based implementation, making the processing faster, unlike what happens if this entire process took place in a sequential fashion. In that way, the processor performs a reading module output, which already contains the result based on multiplications and additions in hardware (PEDRONI, 2004; PAGANO, 2012; ALMEIDA, 2015).

6. RESULTS

To compare the results, the same convolution performed by the ACD module in a NIOS II processor was performed in MATLAB. After a series of tests with different images, it was obtained the pictures shown in Figure 7.

Figure 7. Comparison of Lena images performed by MATLAB and NIOS II FPGA

That figure presents the original image Lena (Source Image) followed by the smoothed filtered image by a Gaussian filter implemented in MATLAB and the smoothed image by a Gaussian filter implemented in NIOS II.

The image generated by the NIOS II has an error of 0.3831% compared to the image obtained in MATLAB.

7. CONCLUSIONS

By comparing those implementations in NIOS II FPGA (hardware) with MATLAB (software), it is possible to note the potential for a FPGA-based embedded system.

The circuit developed in NIOS II obtained substantially identical images (error ~0.3831%) processed by MATLAB. The result of the convolution is the same for all cases, with the advantage that it can be used in the image acquisition in real time for use in several engineering applications.

The developed hardware also has the benefit of being easily reprogrammed for any filtering function, for example the Sobel edge detection, the Gaussian smoothing of the images, the Bartlett filter, in the processing of images, and also allows to creating new flexible topologies for convolution network.

8. ACKNOWLEDGEMENTS

The authors would like to thank CAPES (Coordination for the Improvement of Higher Education Personnel) for the financial support through research grant and UNICAMP by supporting infrastructure development of this work.

9. REFERENCES

Almeida, C. C.; "Arquitetura do Módulo de Convolução para Visão Computacional Baseada em FPGA". Dissertação (Mestrado) *UNICAMP, Universidade Estadual de Campinas, Campinas, Brasil, 2015*.

Altera®. NIOS® II Embedded Evaluation Kit, Processor reference Handbook, Cyclone III Edition User Guide, 2010a, 2010b.

Altera®. NIOS® II Embedded Evaluation Kit, Avalon Interface SpecificationsCyclone III edition, 2011a, 2011d, 2011f.

Farabet, C.; Poulet, C.; Han, J.Y.; LeCun, Y.; "An FPGA-Based Processor for Convolutional Networks" *978-1-4244-3892-1/09 IEE, 2009E.*

Farabet, C.; Poulet, C.; LeCun, Y.; "An FPGA-Based Stream Processor for Embedded Real-Time Vision with Convolutional Networks" *IEEE Transactions on Pattern Analysis and Machine Intelligence 2011*.

Farabet, C.; Martini, B.; Akselrod, P.; Talay, S.; LeCun, Y., Culurciello, E.; "Hardware Accelerated Convolutional Neural Networks for Synthetic Vision Systems". *IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012.*

Farabet, C.; Couprie, C.; Najman, L.; LeCun, Y.; "Learning Hierarchical Features for Scene Labeling", *IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 35, NO. 8, 2013.*

Garcia, C. and Delakis, M.; "Convolutional face finder: A neural architecture for fast and robust face detection". *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(11):1408–1423,* 2004.

Lawrence, S.; Lee Giles, C.; Chung Tsoi, A.; Back, A. "Face recognition: a convolutional neural network approach," *IEEE Trans. Neural Netw., vol. 8, no. 1, pp. 98–113, Feb. 1997.*

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. "Gradient-based Learning Applied to Document Recognition," *Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.*

LeCun, Y.; Kavukcuoglu, K.; Farabet, C.; "Convolutional Networks and Applications in Vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence ISCAS 2010: 253-256, 2011*.

Ning, F.; Delhomme, D.; LeCun, Y.; Piano, F.; Bottou, L.; Barbano, P. "Toward Automatic Phenotyping of Developing Embryos from Videos," *IEEE Transactions on Image Processing, special issue on Molecular and Cellular Bioimaging, 2005.*

Ordonez, E.D.M.; Penteado, C.G.; da Silva, A.C.R.; "Microcontroladores e FPGAs: Aplicações em Automação", *volume I. Novatec, 2006.*

Pagano, D. M.; "Proposta de uma arquitetura de processamento de sinais utilizando FPGA". Dissertação (Mestrado) UNICAMP, *Universidade Estadual de Campinas, Campinas, Brasil, 2012*.

Pedroni, V.; "Circuit design with VHDL". *The MIT Press, 2004*.

Perry, D. L. ;"VHDL Programing by Example". [S.l.]: *McGraw-Hill, 2002*.

Raizer, K.; Idagawa, H. S.; Nóbrega, E.G.O.; Ferreira, L.O.S; "Training and applying a feedforward multilayer neural network in GPU", 11/2009, *Científico Internacional, Iberian-Latin-American Congress on Computational Methods in Engineering (CILAMCE), Vol. cd, pp.1-6, Buzios, RJ, BRASIL, 2009*

Raizer, K.; "Análise de Sinais de ECG com o uso de Wavelets e Redes Neurais em FPGA". Dissertação (Mestrado) *UNICAMP, Universidade Estadual de Campinas, Campinas, Brasil,2010*.

Sebesta, R.W.; "Concepts of Programming Languages". *10th Edition, Bookman, 2012*.

Tomazati, A. O.; "Detecção do Complexo QRS em Sinais Cardíacos utilizando FPGA". Dissertação (Mestrado) *UNICAMP, Universidade Estadual de Campinas, Campinas, Brasil, 2009*.