# IMAGE SEGMENTATION USING THRESHOLDING TECHNIQUE FPGA NEXYS A7 BOARD

A THESIS SUBMITTED TO
THE FACULTY OF ARCHITECTURE AND ENGINEERING
OF
EPOKA UNIVERSITY

BY

ARDIT DERVISHI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JULY, 2020

**THESIS TITLE: IMAGE SEGMENTATION USING THRESHOLDING TECHNIQUE FPGA NEXYS A7 BOARD**

submitted by Ardit Dervishi in partial fulfillment of the requirements for the degree of **Master of Science in Department of Computer Engineering, Epoka University** by,

Dr. Ali. O. Topal _____
Head of Department, **Computer Engineering, EPOKA University**

Prof. Dr. Betim Cico _____
Supervisor, **Dept., EPOKA University**

Assoc. Prof. Dr. Dimitrios Karras _____
**Dept., EPOKA University**

**Examining Committee Members:**

Dr. Ali Osman Topal, _____
Computer Engineering Dep, EPOKA University

Prof. Dr. Betim Cico _____
Computer Engineering Dept, EPOKA University

Assoc. Prof. Dr. Dimitrios Karras _____
Computer Engineering Dept, EPOKA University

**Date:** 24.07.2020

**I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.**

Name, Last name: Ardit Dervishi

Signature:

# ABSTRACT


**IMAGE SEGMENTATION USING THRESHOLDING TECHNIQUE FPGA NEXYS A7 BOARD**

Dervishi, Ardit

M.Sc., Department of Computer Engineering

Supervisor: Prof. Dr. Betim Cico

Nowadays, with the introduction of Graphics Processing Unit for general purpose issues, not only graphical ones, there has been an increasing attention towards exploiting GPU processing power for deep learning algorithms. In the world of technology as considered by many scientist and analysts everything is going very fast.In this thesis I will use FPGA boards rather than graphical card processing using like NVidia, as a case study in observation of behavior regarding with image segmentation.

Therea are several qualities that distinguish both processors, with classical graphical processing units being more flexible, not much complex and vice versa with FPGA processors offering programmability, reducing time latency, optimized energy in computation process. For while, it has been an enigma the comparison between two different mentalities ( software vs. hardware ) engineering mentalities will occur, thus the results will be compared to each-other like energy used, flip-flops, LUT-s etc. The whole system will be implemented in Xilinx Nexus A7 FPGA board and Vivado HLS 2018.2 software framework.



***Keywords:*** *Convolution Neural Network, Deep Learning, Image segmentation, Field programmable gate array, Xilinx, VHDL*

# ABSTRAKT

Në ditët e sotme, me prezantimin e Njësisë së Përpunimit të Grafikëve për çështje me qëllim të përgjithshëm, jo vetëm ato grafike, ka pasur një vëmendje në rritje drejt shfrytëzimit të fuqisë përpunuese GPU për algoritmet e të mësuarit të thellë. Në botën e teknologjisë, siç konsiderohet nga shumë shkencëtarë dhe analistë, gjithçka po shkon shumë shpejt. Në këtë tezë unë do të përdor bordet FPGA sesa përpunimin e kartave grafike duke përdorur si NVidia, si një rast studimi në vëzhgimin e sjelljes në lidhje me ndarjen e imazhit.

Ekzistojnë disa cilësi që i dallojnë të dy procesorët, me njësitë klasike të përpunimit grafik që janë më fleksibël, jo shumë kompleks dhe anasjelltas me procesorët FPGA që ofrojnë programueshmëri, zvogëlojnë vonesën në kohë, energjinë e optimizuar në procesin e llogaritjes. Përderisa, ka qenë një enigmë krahasimi midis dy mentaliteteve të ndryshme (softuer kundrejt harduerit) do të ndodhë mentaliteti inxhinierik, kështu që rezultatet do të krahasohen me njëri-tjetrin si energjia e përdorur, flip-flops, LUT-et etj. sistemi do të implementohet në bordin e softuerit Xilinx Nexus A7 FPGA dhe Vivado HLS 2018.2.

***Fjalët kyçe:*** *rrjeti i konvolucionit neural, deep learning, segmentim imazhi, FPGA, xilinx, vhdl ( gjuhe programimi )*

# ACKNOWLEDGEMENTS

I would like to express my special thanks to my supervisor Prof. Dr. Betim Çiço for his continuous guidance, encouragement, motivation and support during all the stages of my thesis. I sincerely appreciate the time and effort he has spent to improve my experience during my graduate years.

My acknowledgement goes to my thesis committee members, Dr. Ali Osman Topal Head of Computer Science Department and all the professors, for their comments, feedback and suggestion throughout entire thesis. Also, I note that I am always thankful to my parents, for their unconditional encouragement and support through entire my life.

# Table of Contents

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

## 1.1. Background of image segmentation

Image segmentation marks its origin in 70s and 80s during medical researches into bone tissues, bacterial cells, different viruses, later on it spread into X-ray, MRI images. Later on, its peak was achieved during the 90s, where a significant amount of paper research were published. Most of them were related with image reconstruction, for high-speed image processing, increasing the specificity of those images.

Ninety years after its invention, the Pap test continues to be the most used method for the early identification of cervical pre-cancerous lesions. In this test, the cytopathologists look for microscopic abnormalities in and around the cells, which is a time consuming and prone to human error task. This paper introduces computational tools for cytological analysis that in corporate cell segmentation deep learning techniques. These techniques are capable of processing both free-lying and clumps of abnormal cells with a high overlapping rate from digitized images of conventional Pap smears [1]

In order to transcend the insufficiency in computer resources, a significant amount of analysts and researches have conducted the results from the process of convolution neural network calculation, furthermore have produced different strategies ( methods ) for computation. In the figure below, the ‘strategies’ mainly focus in accelerating and optimizing, which will tremendously increase the performance by giving a better accuracy, less time processing, larger and maximized output result and higher efficiency in power producing. Many experiments are conducted in this section, using theses CNN industrial codes. The methodology in this strategy tends to label CNN architecture, which

is nonetheless but just the multiple convolution layers attached together as shown in the below figure [2]

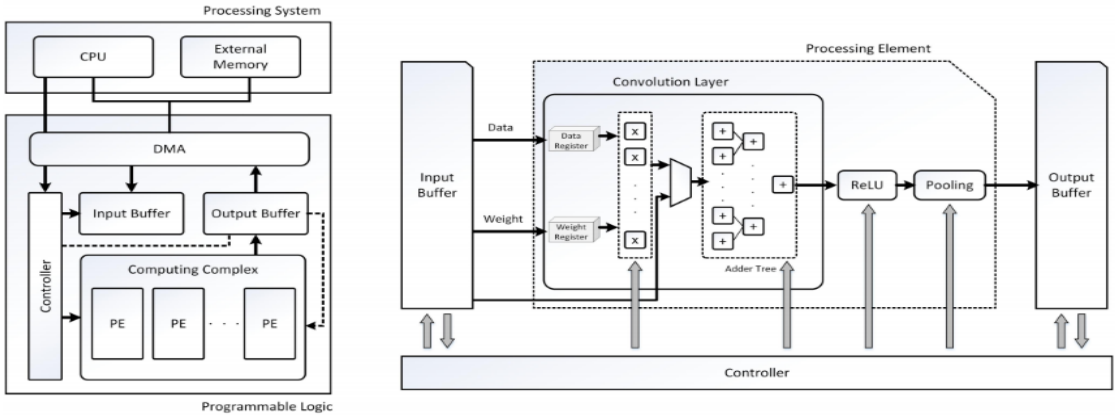



***Figure 1.*** CNN- based architecture for FPGA

'Researchers build what they call a 'body plan', adapted to segmentation and recognition in complex environments. They suggest that the body plan can be fixed or can be learned using statistical learning techniques'. In the methodology that we have been using, a pre-processing step is being utilized, where the images with abnormal cells that have a probability very low are rejected, with no preliminary division, moreover this methodology increases the efficiency, by performing faster than any other methodology. Furthermore, according to the probability of abnormal cells, the results vary to different possibilities. Through conventional Pap smears, a new methodology is being unveiled, that contains more than 108 field views, from real-world scenarios image database, where 86 cells are normal, 1 abnormal cell through millions of corresponding cells.

In the below figure, processing 'path' of convolution neural network where the basic data flows. Hierarchically, CNN contains multiplication and adder units connected together. Researchers and software developers have found that these units ( multiplication and adder units ) may decrease the processing speed, even though the number of layers is being optimized and reduced through the process of data reformation. Later on, these

researchers started to optimize the data access process, by creating new methodologies that enable and develop an improved floating point and increase the size of buffer.

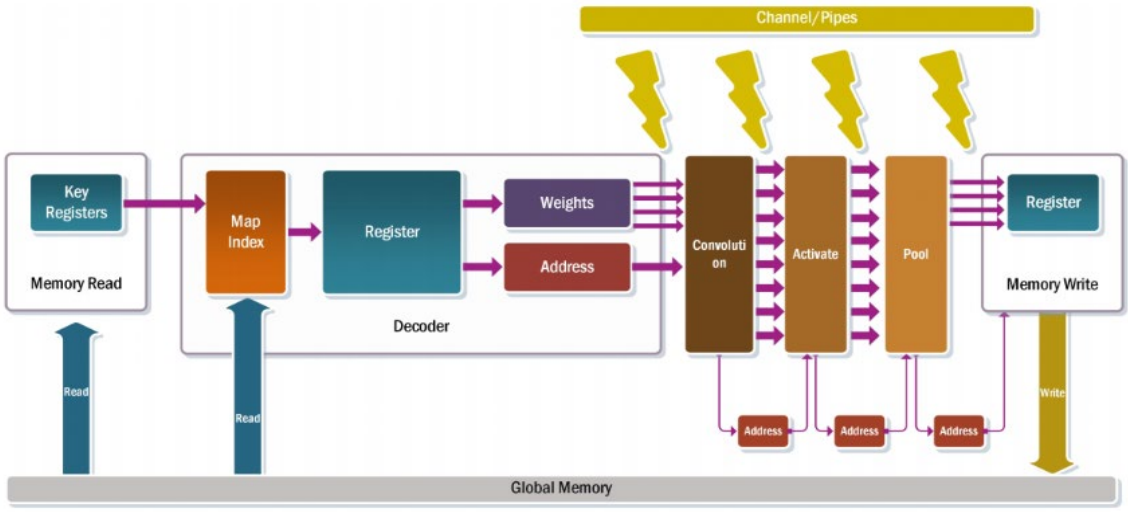



***Figure 2.*** Computation process overview of CNN

It is known that for the first time, Convolutional Neural Network (CNN) based deep learning model by Krizhevsky et al brought down the error rate on that task by half, beating traditional hand-engineered approaches [3] .He achieved that by simply performing an end-to-end data training, that does not need unsupervised pre-training like the traditional methods. During the upcoming years, the method used by Krizhevsky et al, deep neural network image classification, earned the status of one of the most important computer vision papers. It became quite vital the importance of convolution neural network in deep learning designs, especially the Krizhevsky et al trained network, which earned the name 'Alex Net'. This method become a fundamental strategy, used to solve different problems in computer vision.

## 1.2. Objective and aim

In this thesis I am going to stress out the necessity of FPGA board processors rather than traditional graphical processor units. Its vitality is related with ability to be programmable, large amount of data related to Deep Learning it is proportionally with the better results and accuracy. Moreover, in the results & discussion section 4.4, the outputs from two different "observing" views ( software & hardware ) mentality will be compared, and the reason why the hardware engineering mentality will provide us better throughput. Furthermore, there are some reasons which eager me toward this special topic:

- The latest paper researches through these years are in signal processing which makes the hottest area where there is still a lot of work to do in the future.

- Image processing applicability and usage in everyday life ( in my thesis mostly concentrated in image segmentation ), furthermore its field of study is related with segmentation of cancer cells, MRI images, etc.

## 1.3. Scope of work

The scope of signal processing ranges to multiple application starting with audio, speech, music, image and video processing. These are complemented by wireless communications and networking, and information security aspects [4]. Additionally several paper researches and science articles are oriented via this direction citing latest updating pointing them out not even practically but even theoretically. But in general words what is the definition of processing ? It is a represented way or method of how a given information can be more understandable , for instance, how discrete Fourier transforms are used to understand the frequency components of a signal. There are some

areas where signal processing takes place like : compression, communication and audio, image and Video, Bio-Sensors. Signal processing is a subcategory of lots disciplines like; electrical & electronic engineering, informatics, mathematics etc. that are related with signal modification, synthetization and analyzation, where the functions are defined as “information about the behavior or attributes of some phenomenon". These can rather be electronic measurements, images or even sounds. For instance, the necessity of signal processing is being noticed in properties like, increasing efficiency of storage, fidelity, increasing the signal transmission throughput and even better optimization in power used in blocks.

## 1.4. Thesis structure

My thesis is divided into 5 chapters, where its organization is as below. It begins with Introduction into image processing, then literature review ( hypothesis and research questions ), FPGA and VHDL ( implementation and design of system), results and analysis and the last chapter conclusions and future work.

The first chapter begins with the abstract of work ( design ) that is going to be implemented associated with acknowledgment, introduction in image processing where a general overview of image segmentation, its applications. After that, I will state my problem and give my objectives, motivation.

Chapter 2 include papers that I have been analyzed, where I have identified the research gap. The paper researches that I have conducted are published during the last 3 years. Furthermore, I have stated the conclusions & future work of each paper. In this chapter, I am going to emphasize the hypotheses and research questions. Their aim is to represent thesis objectives, directivities and answer some important questions related with the topic.

This chapter will begin with the general idea of image segmentation and the methods used to implement it. Then I will focus on the methods that I am going to analyze using deep learning technique. What is deep learning and how it is going to affect to our results, what is the amount of data needed and how is the training process going to take place. I will introduce FPGA and its framework, the features and its main buttons, HDL programming language ( vhdl and Verilog programming languages ).

Chapter $4^{th}$ is considered to be one of the most important part in thesis, since all the experimental work is conducted. I will implement several blocks using Nexys A7 framework in order to gain a higher power efficiency, where the input data ( image ) is going to be converted into .hex file being programmable for our Nexys A7 board. There are going to be introduced some concepts like: reducing power consumption, reforming the code in order to optimize the number of flip – flops.

All the gathered information from experimental work will be used in order to compare with the current energy specification of the framework used even in the papers ( referred in literature review ). I will use graphical method of data representation to have better comprehensive results. In conclusions section all my results and work interpretation, is going to be included. I am going to state possible future work and references.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1. Introduction

The literature review has been based on numerous amount of papers related with the impact of image segmentation in signal processing, its correlation and increasing a fundamental problem. Mostly of the papers that were found by me and my mentor also gave me the understanding of having a good perspective in characterizing object boundaries and giving the details of the complex images for instance, detection of boundaries in tumor cells and being able to overcome with a solution, nowadays is a huge challenge. After narrowing my gap of research I went through a better analyze of where the papers came through, the years they were published which beside one paper it dated in 2012 the others where from the past 3 years.

From our systematic mapping study and also from other literature reviews in the same field, we can say that image analysis has been extensively studied [5]. The majority of papers dealt with image search purposes, most of the, based in deep learning and CNN as architecture, using GPU-accelaeration According to the sites where they were published: 75 % of paper researches were published in IEEE, 18 % in ResearchGate, 5 % in ACM and the rest from different articles, scientific magazines.

Deep learning is directly cited to be full group of information layers called learning techniques, where all these layers contain in decreasing order lots of processing phases, used to learn features, representations and extinguish models in different classifications. Common properties, features are inteconnected together through areas of optimism,

recognition model, graphic and signal processing inside neural network areas. Therefore, a group of neurons, which consists several processor name connects come together to create neural network. Through sensors, all these neurons become activate because of active neurons. Moreover, a slight effect is being noticed in the enviroment, because of action promoting neurons. Learning or credit assignment is about finding weights that make the NN exhibit desired behaviour. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages, where each stage transforms (often in a non-linear way) the aggregate activation of the network. Deep Learning is about accurately assigning credit across many such stages.

The aim of the thesis is to study the comparison between a traditional method such as Thresholding technique versus EM / MPM using FPGA, based on deep learning algorithms and convolution neural networks. The idea is to gain knowledge over state-of-art implementations in the field and to potentially distinguish future directions [6] .

Multimedia technology is popularized in consumer electronics, therefore we see increased use of image processing systems. New products require greater image capacity, higher image quality, and faster image processing speed. Many image processing systems are implemented on graphic controllers with Digital Signal Processors (DSP) or PC software. Additional effort is needed to control DSP work flow. For the segmentation algorithm, choosing FPGA hardware improves the speed [7].

In order to improve the speed for medical image for image segmentation systems, a high speed image processing system is presented from Xinxi Zhang, Yong Li, Jinyang Wang, Yulin Chen scientist, which underlines the necessity in integration of system for a vehicle-loaded computer to have high velocity in processing of image [8]. The system presented from the mentioned scientist requires FPGA hardware and software system in order to give better functionality in processing, preprocessing and image display. Better accuracy, high speed of the processing image is achieved by all hardware components of

FPGA like: programmable chips, different blocks, thus it gives stronger capability in real-time.

It is known the wide area of application of Image segmentation like robotic vision systems, medical purposes, signal processing and so on. There are different approaches proposed in this area like clustering, graph cutting, super-pixeling, SegNet, semantic image segmentation. There are 3 most known architectures, beginning with pipelined architecture, sequential architecture graph optimization which uses algorithms of segmentation called hybrid architecture. First one and third one, respectively, pipelined architecture is being related with hybrid designs, by replicating lots of elementary modules and re-organizing, scheduling those in parallel order. The second architecture design, sequential design, utilizes level control gates in the architecture, to produce a detailed control of the implementation occurring in CPU. It is worth knowing, that all these architecture example make possible conserving of dissipated power and having better segmentation in real-time.

Going thorough a detailed analysis through every research paper gave me the significance of the image processing. I observed each and every paper, and I identified all of their characteristics, what have been solved, conclusions and future work for every pattern. The first five are related with cell segmentation by using deep learning in order to train effective methods for strong and more improved image details and it has applications in diseases like *breast cancer, tumor, stained cells* etc. Using Deep Learning and Convolution Neural Network tool for object *detection framework and inverse imaging with different modifications to increase resolution* were identified in 4 paper researches. Problem solving again by using deep learning method with CNN tool in reliable reconstruction for nonlinear inverse imaging with non-linear Fourier data were conducted in 3 paper researches Regulating or restoring distortion of images while stretched horizontally or vertically by treating it as regression problem and using CNN tool. Creating a 3d images by optimizing their high speed segmentation applied in MRI images, bone cells images were evidenced in 3 research papers. In one of the papers it shows

another model in representing the shape in image segmentation where it takes the statistical prior information about the shape of the object to be segmented.

It is proposed a shape-based segmentation algorithm that utilizes convolutional neural networks to learn a posterior distribution of disjunction of conjunctions of half spaces to segment the object. This approach shows promising results on noisy and occluded data where it is able to accurately segment the objects. The proposed approach also benefits from fast inference via CNNs which is computationally more efficient that other methods such as density estimation and sampling [8].

Beside that, there were presented lots of fails, pragmatic thoughts in unsolved part which can be truly identified as a future work, where its purpose is giving answers and better objective understanding of different causes, events and unexplained features. Generally the unsolved part in most of the papers is related with the difficulties in adapting architectural hardware, difficulties in hyper selection and computational cost. There is an approach where to presume the pose and identify the action where convolution neural network is being noticed. For this kind of situation, there are methods which train multiple duties involving implementation of CNN.

## 2.2. Hypothesis and research questions

In order for me to deal with the problem, I have to divide it in subtopics by stating hypothesis and research questions which will guide me in my research (null hypothesis and zero hypothesis). I have proposed several research questions each of them posing different directions. We will try to give answers the questions like the advantage of FPGA board usage prior to classical graphical processor units and so on.

The first is based on the most efficient method by taking into the consideration the existing techniques. Another idea is to use another type of CNN architecture for the same task and see how it affects the solution. The last one is to user another type of

learning paradigm for image segmentation.

**H0:** We train a large amount of data for image segmentation, by using deep learning technique in order to have accurate and reliable results. This is a null hypothesis because it is true and proven theoretically by lots of paper researches.

- **How much accuracy will DL method provide us to have better results ?**

Deep learning method is a subset of machine learning, which is a subset of AI ( artificial intelligence ). It has a wide area like signal processing, computer vision, embedded systems, robotics etc. Differently from machine learning method more data are needed in order to be trained and compiled. This brings in better results. More computational costs brings the need of more "powerful" processors called FPGA ( field programmable gate array ).

- **Will image segmentation help us in realization of the determined hypothesis? If no, what will be the limitations ?**

I believe that image segmentation will help us a lot in achievement of our objectives. I think that both methods ( edge thresholding and EM/MPM ) will provide outstanding results which would help us in understanding of segmentation of tumor cells. Despite that, it offers limitations like requirement of enormous amount of data to be trained, high computational cost.

**$H_A$:** By analyzing pixels and their characteristics we will be able to detect damaged cells in several diseases. This is an alternate hypothesis because it need to be proven through research.

- **How efficient will these results be able to determine these damaged cells ?**

Different methods have different approaches and show different results. EM/MPM method classifies every pixel in an image by assigning a cost to an incorrect segmentation

based on the number of incorrectly classified pixels and iteratively finding the best probabilistic solution which fits the data. On the other hand edge thresholding methods lies in using multiple thresholds to find the edges, therefore once there is a pixel as a starting point, then we identify contours route through each and every pixel and mark that instantaneous route where we are below critical threshold value. Later on, we terminate it till that values is below the critical threshold value set up from us. This kind of treatment assumes that the route is in those continuous curves, by allowing us to follow the diminished sections that we have seen before, without noting every pixel is targeted as an edge. Even though, it has flaws regarded in finding the appropriate thresholding parameters, while lots of thresholding values can vary over the image.

- **Which is the most efficient method to deal with ?**

The most traditional and simplest method is thresholding compression based method, where it has an wide application computed tomography images. Basically, the key is to select the threshold value over and over until we have better results in edge detection. There are other methods like: clustering method or K-means algorithm is an iterative method which is based in partitioning the image in K-clusters, compression technique, fuzzy C-means clustering method, dual clustering method etc. Several paper researches imply the comparison between of two or more techniques where the most proficient method is EM/MPM algorithm where the processor uses portions of EM algorithm to perform MPM. The real question is how the output will result if we apply deep learning CNN in this algorithm? This is what I will try to evaluate in my thesis.

- **How will deep learning CNN indicate in image segmentation ?**

Deep learning is a subset of machine learning and it is quite vital in producing better results in image segmentation. Convolution Neural Network is set to be one of the most important and prominent architectures used in deep learning area. It consists of several layers, where these layers have filters, different sets of weight and they are not interconnected to each-other. A 2D portion of filter is named as Kernel. They are applied through convolution process.

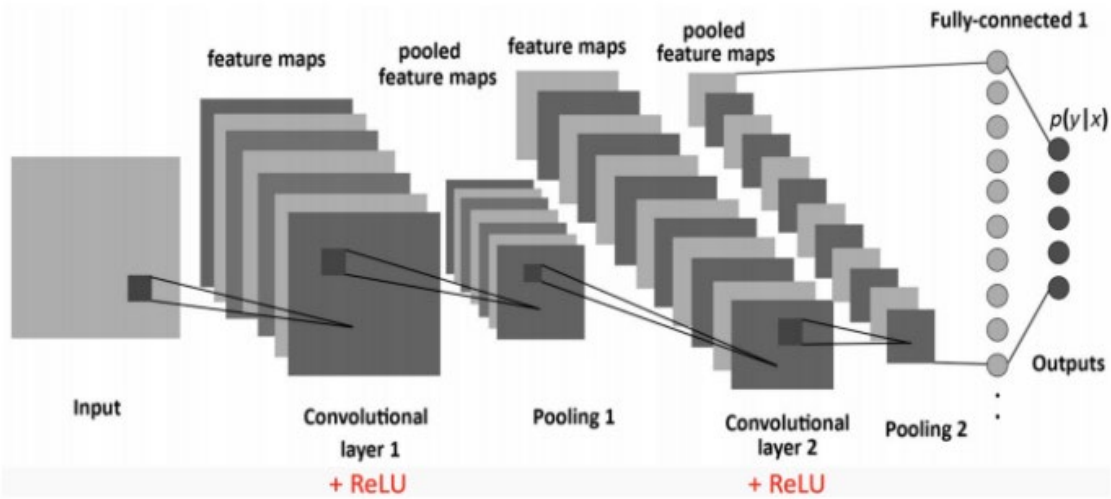



***Figure 3.*** CNN containing 2 convolution layers, 2 pooling layers, and a fully connected layer

- **What would the result be if we used Recurrent Neural Networks to deal with the problem of image segmentation?**

Recurrent Neural Networks are often used to pre-process data like: videos, simple text,different speeches, in that instantaneous position or time depending respectively in the prior data. At each time-stamp the model collects the input from the current time $X_i$ and the hidden state from the previous step $h_{i-1}$, and outputs a target value and a new hidden state. It exists a type of RNNs called LSTM ( long short term memory ) which avoids the issues such are gradient vanishing or exploding problems. LSTM architecture includes gates (input gate, output gate, forget gate), which regulate the flow of information into and out from a memory cell, which stores values over arbitrary time intervals [8].

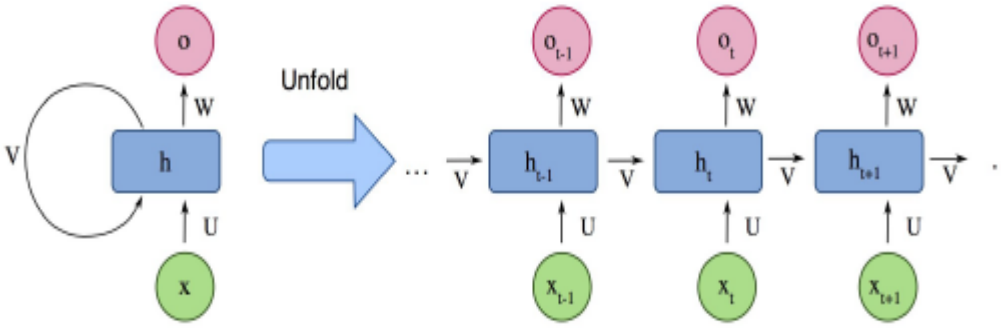



***Figure 4.*** Architecture of RNN ( recurrent neural network )

Furthermore, Recurrent Neural Network can pattern a sequence of data, where every template its is related with the prior model. The perfect process sustainable for the RNN is by convolving every layer, thus increasing effectiveness of the upcoming pixel.

- **Graphical processing units disadvantages comparing to FPGA processors ?**

Even though GPUs are more flexible than FPGAs, they lack in quality while FPGAs offering a better system throughput hardware algorithm representation. For sure, hardware is faster and better than software. In the prior, FPGAs offer lower energy consumption, higher agility, lower latency, higher execution speed etc. All these qualitities insight the importance of FPGAs in nowadays application.

# CHAPTER 3

# METHODOLOGY

In my thesis I am going to work from theoretically perspective in accordance with the stated hypothesis. The system will be implemented in Nexys A7 board FPGA processor, where I am going to take binary image from MATLAB R2018a framework. The output .mif image from MATLAB will be processed with BRAM blocks of Xilinx VIVADO 2018.2 framework, and according to the methods implemented, I will state the results. These results with be compared and analyzed with the previous results of literature review papers. The most flexible method is Quantitative research because it deals with experimental study. Quantitative methods emphasize objective measurements and the statistical, mathematical, of the throughput taken from simulation test benches. In the end, I will try to train these data through deep learning method and Keras framework, in order to achieve the specificity of the segmented image we want.

## 3.1. FPGA

Digital hardware has been greatly improved and expanded in last 40 years . Since the invention of MOSFET transistor, the number of transistors in a chip has only grown exponentially approximately accordingly with the Moore law and today into a silicone chip they are integrated hundreds of millions of transistors. In the past, most applications of digital systems were oriented towards computational systems. The great development of integrated circuit technology has made that most of the electronic systems such as telecommunication systems, control systems , power electronic systems and systems found nowadays in IoT to be based on the digital technology. FPGA stands for Field Programmable Gate Array meaning they are nothing more than reconfigurable logic blocks (logic gates, memory elements, DSP components, etc.) and interconnects.

Nowadays there are several types of FPGA boards like : Altera Cyclone board, SainSmart EP4CE6, FPGA development board EP4CE40, Altera Cyclone II mini board, Basys-3 board, Virtex family boards very expensive and adaptive to higher complexity computation processes. I have used a medium-type FPGA board Nexys A7, which was the best processing board at our university, therefore I have challenged all the limitation that Nexys A7 board has provided to me. FPGA is very different from other chips that can be bought in the market is that FPGA doesn't do anything , it has no intended function unlike microcontrollers which is a digital system well-built, it acts like a computer because it has all its component hardwired and it can do something useful if it is being programmed [10]. At beginning, when we start routing at FPGA-s boards, there are being noticed lots of wires connected with each-other by means of switches. Every block in FPGA is related with two components : wire segments and their length. The ratio between used block logics, routing area against wire segments is inversely proportional. They have a greater advantage in design FPGA system by giving a trade-off in time optimization and productivity amount of money used. FPGA on the other side acts like a 'stupid' circuit because they do not have a digital system built-in, but they make up for it by giving the user extreme flexibility allowing him to do anything imaginable in the digital domain.The two main components of a FPGA are CLB (configurable logic blocks) and switch matrix interconnections or routing channels and input/output block (IOB) [11]. A schematic of FPGA is given in the figure below.

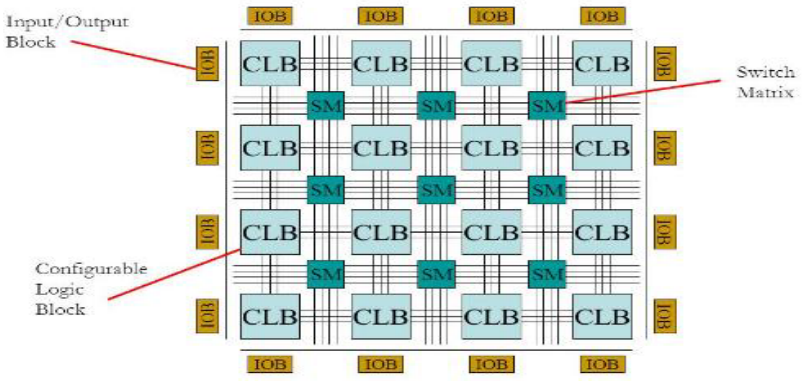



***Figure 5.*** FPGA structure

**Configurable logic blocks :** Is the basic logic unit of an FPGA which can be used to build combinatorioal or sequential circuits.The number and properties vary from one device to another but each CLB consists of 4 or 6 input look-up tables which have 1 or 2 outputs respectively, multiplexers, carry and control and flip flops.

- **LUT** - Look-Up table are the primary elements forthe application of logic. A LUT can accomplish any 4 input Boolean function. Except Boolean functions a LUT can accomplish synchronuous RAM and a 16 bit shift register also.

- **Carry and control** - This component includes the logic of fast math, multiplier logic, multiplexer logic. Each CLB holds special logic and guidance for fast signal generation and fast signal proccessing .This leads to increased efficiency and performance at adders, subtractors, accumulators, comparators and counters.

- **Memory element** - It can be a Flip Flop or a Latch equipped with the Set and Reset inputs. Entries can be also inverted. Such an element may apply synchronous or asynchronous logic.A schematic of a CLB (configurable logic block) is shown in the figure below.

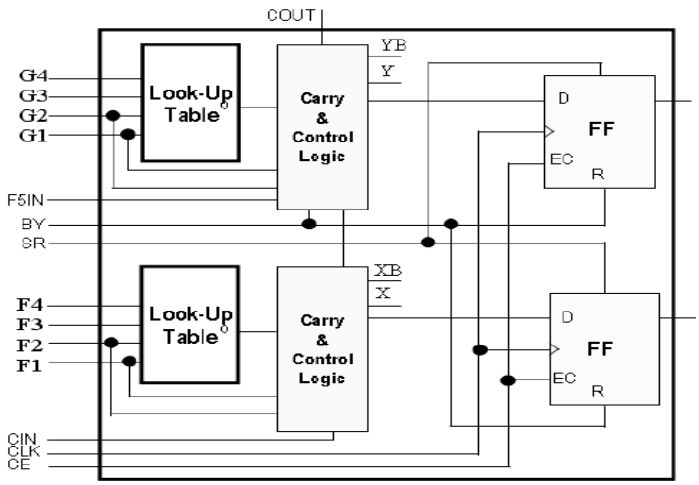



***Figure 6.*** Structure of CLB

**Switch matrix interconnection or routing channel :** While a CLB provides the basic logic, a flexible connection gives connections for signals between CLBs and between CLBs and IOB. Routing takes many forms, starting with the one designed for connecting CLBs to the lines long and fast vertical and horizontal interfaces that interface with routing for the system clock and other global signals.

**Input / Output blocks IOB:** I/O's on FPGAs are grouped into banks where each bank is able to uphold different input/output standards. Today FPGAs offer dozens of I/O bands to provide I/O flexibility. IOBs are interfaces between packet pins and CLBs.

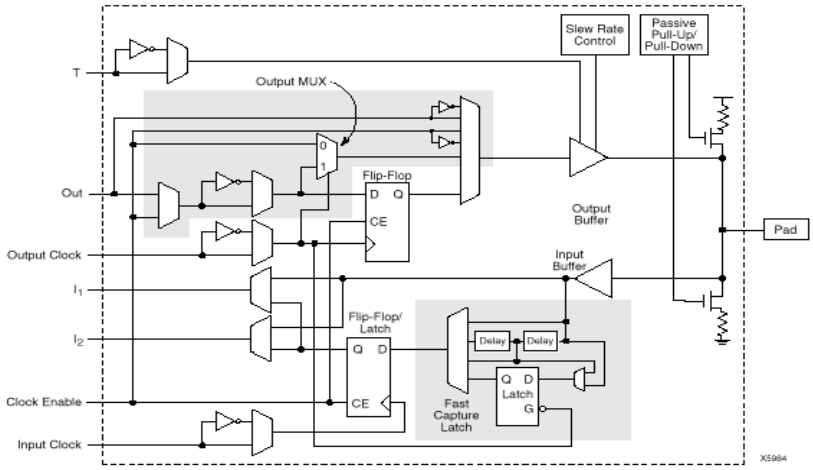



***Figure 7.*** IOB schematic

## 3.2. NEXYS 4 DDR BOARD

The Nexys4 DDR board is a complete , ready-to-use digital development platform. The FPGA in this digital platform is the latest Artix-7TM designed by Xilinx (Xilinx part number XC7A100T) which is extremely powerful FPGA and very new in the market. The Nexys4 DDR is an update of the previous Nexys4 Board. One of the major improvements is 16 MiB Cellular RAM with a 128 MiB DDR2 SDRAM memory. The Nexys4 DDR has a high capacity FPGA , generous external memories , Ethernet, Collection of USB and

other ports giving it the possibility to design digital systems from introductory combinatorial circuits to powerful embedded processors. Nexys4 DDR have several improved built-in peripherals including temperature sensor, accelerometer, MEM digital microphone 16 user switches and LEDs , two tricolor LEDS, micro SD card connector, PDM microphone , four Pmod ports , serial flash , USB-UART bridge, 128 MiB DDR2, Diligent USB-JTAG , 10/100 Ethernet PHY USB HID host for mice, Pmod for XADC signals, PWM audio output , PDM microphone and two 4-digit 7 segment displays.

My board Artix -7 FGPA 100T includes.

- 15850 logic slices, each with four 6-input LUTs and 8 flip flops
- 240 DSP slices
- 4860 Kbits of fast block RAM
- Six clock management tiles
- On chip analog-to-digital converter (XADC)
- Internal clock speeds exceeding 450 MHz [9]

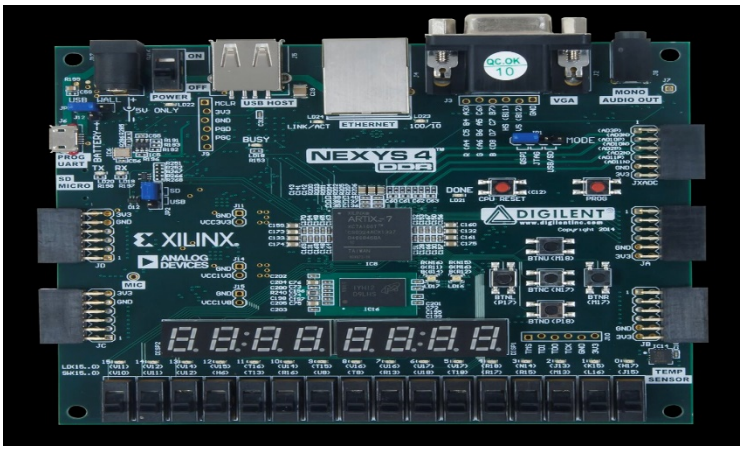



***Figure 8.*** Nexys 4 DDR FPGA board

## 3.3 HDL program and its design

A digital system can be described at different levels of abstraction and from different perspectives. With the advancement of the design process level of abstraction and perspective have changed, both from human designers and software tools [13]. It is desirable to have a common framework for sharing information between designers and various software tools [10]. For this purpose serve the hardware description languages (HDL). It is rather a language in which it shows the relation between the behaviors and structure of electronic and digital circuits. It assembles a lot with C , C++ programming and its structure ( HDL syntax and semantic ) it includes notations in which different parts of code are executed out-of-order without affecting the final throughput. Moreover, it include notion of time which is quite an important hardware attribute. HDL is the program where all the elements of the programming language disciplines like C/ C++ provide the designer all the features for a perfect hardware development.

The properties of a digital circuit are based on several concepts like connection, concurrency, timing and entity. The entity is a basic building block, that models a part of a real circuit. It is self-sustaining and independent, and it does not have any information about other components. The connection models the connecting cable between different parts of the digital system. The entities cooperate with each-other while lots of entities can be active at the same time by performing many actions in parallel. This can be described as concurrency, while timing is correlated to concurrency and it expresses the start and end of each action and builds a schedule and a queue of numerous other actions.

The purpose of an HDL is to design and describe a digital system in a complete, accurate and reliable manner. To achieve this, the foundation of language must be based on the hardware action model, and its semantics must be able to capture the main features of the circuit. A digital system can be described in four different levels of abstraction and from three different perspectives. Although these descriptions have similar basic features, detailed layout and their models vary widely.

| | |
|---|---|
| **The entity** : | describes the external interface of a circuit where they are included circuit name, basic names and characteristics of input and output gates. |
| **The architecture** | specifies the inner workings or organization of a circuit. In VHDL several different architectural bodies can be developed for the same entity declaration but then we only have to select one of them to associate with the entity to be simulated and synthesized. |
| **Package declaration** | A VHDL package usually contains a set of cunstructs that are generally used and needed by many VHDL programs such as data types, subprograms and components. |
| **The package body** | contains the implementation and code of the subprograms and other components. |
| **A configuration** | specifies which architecture to connect to entity declaration . This is because some architecture can be connected to the same entity declartion in the VHDL. |

***Figure 9***. Explanation table of HDL features

Ideally, it is attempted to develop a single HDL language for it to cover all levels and all perspectives. However this is very difficult to accomplish because of the many variations between levels of abstractions and perspectives would make the hardware description language very complex. Modern HDL languages cover descriptions of behavioral and structural model of the circuit but not in physical terms. They provide constructs to support circuit modeling at gate level and register transfer level (RTL) and in a limited degree of abstraction at the processor and transistor levels. Two of the most important HDL hardware description language are Verilog and VHDL. In my thesis I will use VHDL language for some of the reasons that I am going to explain below.

## 3.4 VHDL ASSEMBLY LANGUAGE

FPGA board can be easily programmable with HDL programing language either VHDL or Verilog. They serve for the same purpose but, also have differences between

them. VHDL can be used to describe models, while it originates from government program to elaborate complex integrated systems.

VHDL and Verilog are the most popular HDL languages. Although the syntax and the presentation of these two languages are very different, their abilities and goals are very similar, not to say the same. Both of these languages are standard and are supported by most software. We can say that VHDL has one better support for a parameterized design. In low level modeling Verilog is simpler than VHDL. Verilog is way simple for a new hardware programmer due to its resemblance to C programming language.

Verilog is supporting User-Defined Primitives (UDP).This feature is very popular for ASIC designers. VHDL and Verilog implement register-transfer-level (RTL) abstractions. Since, they were firstly presented in the late 80s, immediately all engineers were eager to use these programming languages, at a higher level of abstraction register-transfer-level based simulators. One of the strongest features of VHDL, is its richness, deterministic and more voluble than Verilog. In the other hand, Verilog syntax is more like C/ C++ programming language, that's the reason why engineers designing in VHDL programming language need extra coding. Another advantage in VHDL language, is error-catching ability. The highest level is a system level of abstraction that features constructs intended for system-level design applications.

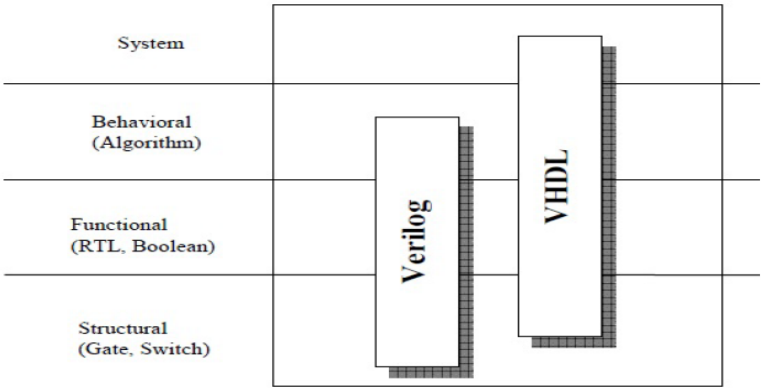



***Figure 10.*** Levels of abstraction

The derived VHDL model will consist of a combination of behavioral, RTL and structural definitions mapped directly from the *Simulink* model. The library in VHDL is called the place where project units are stored. Usually associated with one directory in the memory space of the computer. The program determines the link between the name of the VHDL library and its physical directory.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity div10 is
    Port ( rst :      in    STD_LOGIC;
           clkin :    in    STD_LOGIC;
           clkout :   inout STD_LOGIC);
end div10;

architecture Behavioral of div10 is
   signal clkdiv : unsigned(2 downto 0);

begin
   process(clkin, rst)
   begin
      if rst = '0' then
         clkdiv <= "100";
         clkout <= '0';
      elsif clkin = '0' and clkin'event then
         if clkdiv = "000" then
            clkdiv <= "100";
            clkout <= not(clkout);
         else
            clkdiv <= clkdiv - 1;
         end if;
      end if;
   end process;
end Behavioral;
```

***Figure 11.*** VHDL sample code

### 3.4.1 History of VHDL

VHDL was sponsored by the US Department of Defense as a standard for hardware documentation in early 1980s and then transferred to IEEE 27. IEEE ratified it as standard 1076 at 1987, which is today referred to as VHDL-87. Every IEEE standard is reviewed within a few years. The IEEE revised the VHDL standard in 1993, which we refer to as the VHDL-93 and then in 2001 made some modifications and fixed some bugs which were annoying the designers a lot. The latest version is referred to as VHDL-2001. Like in every programming language, firstly we need to learn its syntax and language construction. There are several differences between VHDL and Verilog, they differ as below:

- **Compilation :** In VHDL entity/architecture pairs, can be compiled each on its own, by giving the ability for each design unit to keep its files while in Verilog compilation is achieved by speeding up the simulation, so the original language is unchanged. As a result care must be taken with both the compilation order of code written in a single file and the compilation order of multiple files.

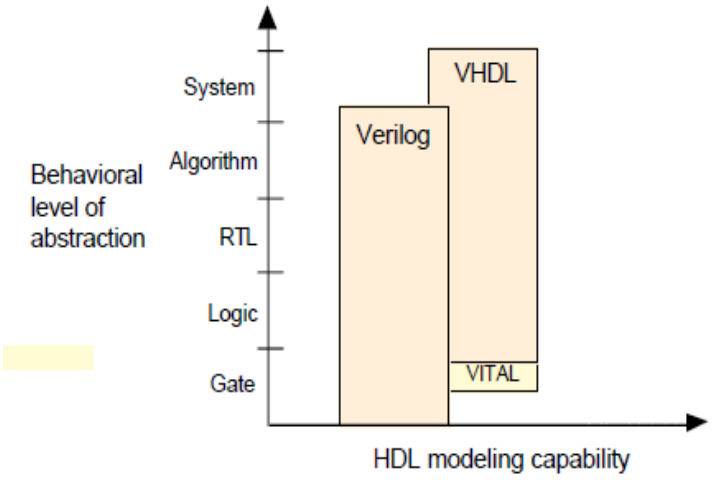



***Figure 12***. HDL modeling capability

- **Data types :** Dedicated functions for converting objects, it allows a multitude language which results in make the models easier to be written and read, while in Verilog all the data types are defined from the Verilog and not by the user as in VHDL. The data types
  are much more easier to use, and the vast simplicity that offers gives better advantage than VHDL.

- **Libraries :** it serves like a storehouse, which saves all the packages, configurations and architecture. Moreover, it manages design projects, but in Verilog such concept does not exist.
- **Managing large designs :** Configuration, generate, generic and package statements all help manage large design structures while in Verilog nothing can be done to manage large designs [11].
- **Language Extensions :** its ability is mainly related with allowing new architectures to be modeled. It gives higher advantage to VHDL language, since Verilog is not 'able' to use those tools.
- **Procedures and tasks :** it is a feature of VHDL language, Verilog does not possess this ability.
- **Readability :** Verilog contains more C/ C++ syntax resemblance, differently from VHDL. This is a strong reason why software developers most likely, would be comfortable with Verilog rather than VHDL programming language.
- **Verboseness :** VHDL is a strong language, where it needs precision in coding. Signals representing objects of different bits widths may be assigned to each other. The signal representing the smaller number of bits is automatically padded out to that of larger number of bits, and is independent of whether it is the assigned signal or not [12].

### 3.4.2 VHDL design tools

- **Entity** – which is used to define the external view of a symbol/object.
- **Architecture** – it is a design unit which defines the function of a model/schematic.
- **Configuration** – it is a design unit which is used to associate an entity with the architecture.
- **Package** – it is a design unit which is an array of the information that can be referred from the VHDL models and nonetheless consists of package body and package declaration.

A VHDL program is usually processed in four stages:

*1. VHDL functional simulation*

*2. Synthesis*

*3. PAR (Place and Route)*

*4. Bit-Stream Generation*

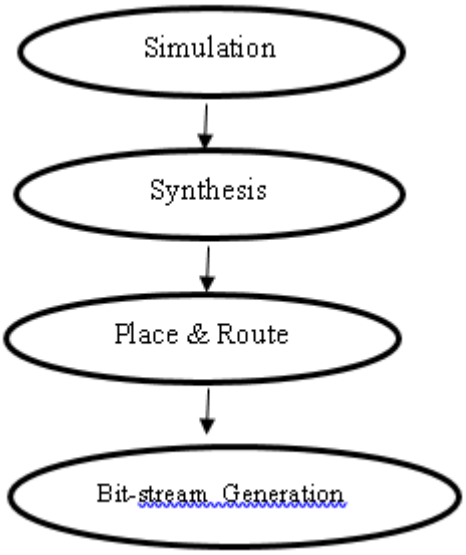
1st Stage the source code of VHDL is checked, It is like a functional simulation where the syntax is checked just as the traditional programming languages. VHDL simulator take places in order to do the functional simulation to check the behavioral simulation of the circuit and to check circuit functionality without building the digital system physically.

2nd Stage, here the logic simulation takes place, where its task is to present the structured description of the synthesized circuit in accordance with the FPGA used.

3rd Stage, where Place and Route happens, the structured description is planned to be place in the FPGA. The right CLB , interconnection and IOB are chosen for the given

FPGA in order to have the most optimized circuit. In this stage also the timing analysis and timing simulation are done and the I/O pins constraints are made.

4th Stage, in bit stream generation, the FPGA program is conveyed into 1s and 0s. After this final step the program is in the FPGA and designed digital system can be tested physically.




## 3.5 XILINX VIVADO SOFTWARE

Vivado is the newest Design Suite of the Xilinx company. It has a lot of improvement from its previous ISE Design Suite. The only disadvantage is that it takes a large storage capacity approximately 20 Gigabyte. Vivado supply design teams with tools and methodology needed to leverage C-based design and optimized reuse , integration automation and accelerated design closure . The improvements that are made in Vivado help a lot to accelerate the BitStream generation of large HDL programs. Three main units in the HDL programming that are accelerated are given below:

- ✓ Accelerating High Level Design
- ✓ Accelerating Verification
- ✓ Accelerating Implementation

The Software-defined Ip generation, blocked-based IP integration and model-based Design integration help in the accelerating high level design. New equation methods in the logic simulation, integrated mixed language simulator, verification IP and the new Vivado HLS helps in the Verification acceleration. Vivado has four times faster implementation than ISE Design Suite, it uses 20 % less Design density and optimize the digital system in order for the FPGA to use less power. Power consumption advantage can go up to 35 %.

Firstly, Vivado was released at 2012 but it have gone down a long road of improvements to reach at the newest version Vivado 2019.1.3. At the newest version of Vivado partial configuration is included with no additional cost in order to help the designers. There are 2 key criteria in our model design like bus interface and memory. EoBM ( External-on Board Memory ), DDR3 SDRAM ( double-data-rate three synchronous dynamic random access ) its function is mainly related with saving image segment output from MPM and data image from host computer. In our framework, XILINX Vivado it occurs by means of PCI express bus the data transfer between PC and DDR3. Serial Rapid I/O, 10 Gigabit Ethernet and Peripheral Component Interconnect Express are just a few bus technologies that have been proposed and applied by the scientist for high speed of data transmission.

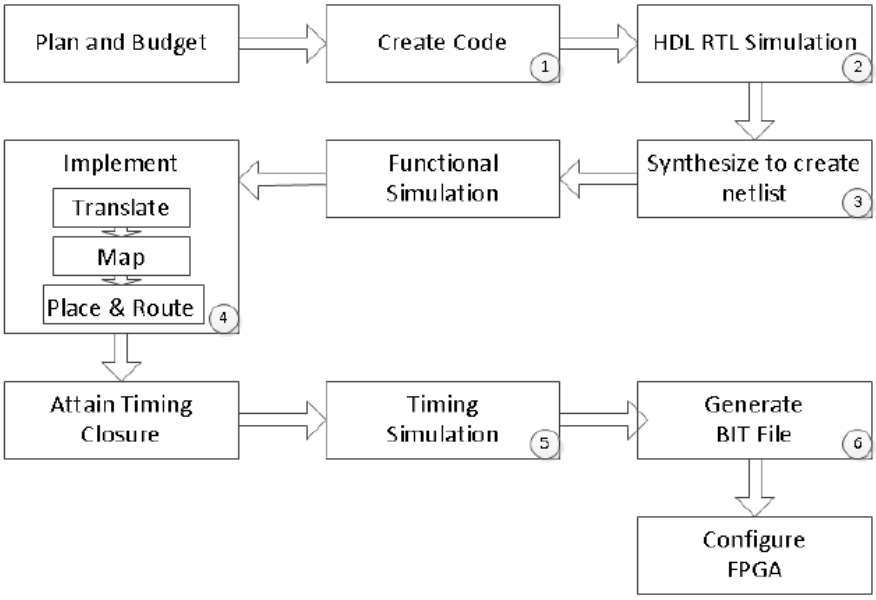



***Figure 13***. Artix-7 based Basys3 scheme

Additionally, Direct Memory Access (DMA) controller has been applied in our system which provides DMA services to devices on the Processor Local Bus (PLB).Vivado does not need another program like ISE Design Suite which need Digilent Adept in order to download the bitstream into the FPGA. It has the tool Hardware manager which make possible downloading the bitstream. Vivado also have another constraint file to used to map the VHDL programming variables into the FPGA. ISE Design suite constraint file extension is UCF and one example of UCF file is given below :

```
NET "sw<0>" LOC = "G18"; # Bank = 1, Pin name = IP, Type = INPUT, Sch name = SW0
NET "sw<1>" LOC = "H18"; # Bank = 1, Pin name = IP/VREF_1, Type = VREF, Sch name = SW1
```

where the variable after net is VHDL variable and after LOC is the circuit variable found in the manual or seen directly at the FPGA, where between the parenthesis of get ports the VHDL variable is written and the location of the circuit can be seen at the

variable after the word PIN. Vivado constraint file extension is XDC and one example of a XDC file is given below :

```
#set_property -dict { PACKAGE_PIN J15 IOSTANDARD LVCMOS33 }
[get_ports { SW[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
#set_property -dict { PACKAGE_PIN L16 IOSTANDARD LVCMOS33 }
[get_ports { SW[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
```

## 3.6 DEEP LEARNING AND ITS APPLICABILITY

Deep learning is the foundation of what we call today AI ( artificial intelligence ). Its application lies in computer vision, speech recognition, robotics, signal processing and even *detecting cancer*. It is well known that deep learning network offers better improvement in accuracy for many AI assignment, but also it gives a trade-off regarding its cost in design complexity. There are techniques that enable processing efficiency in DNN by giving better accuracy and throughput, but in the other hand it can give a hard time in engineers while applying these methods widely in AI systems.

As mentioned above since the specificity and accuracy lies with the high cost of computational complexity, while traditional GPU-s have failed in providing the required acceleration of data analyzing, here the necessity of FPGA-s ( programmable more advanced GPU-s ) becomes substantial.

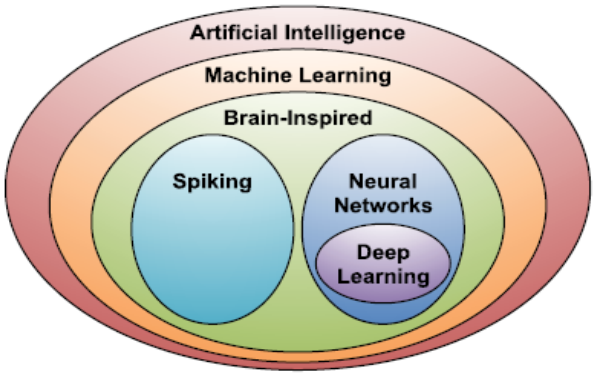



***Figure 14.*** Relation of deep learning with other methods

The first computational element of the brain is the neuron. Therefore, neurons are being connected to a vast numbers of dendrites and the output is going to be an axon. All the signals coming into neuron are computed and the output signal is generated into axon, these are called activations. The signal $x_i$ is multiplied with weight $w_i$ and the sum of all signals as expressed in the figure below gives the output activation, while f($\bullet$) is a non-linear function. Inside neural network an area deep learning is located, consisted of more than three-layers. Nowadays, the average amount of network layers utilized from deep learning method varies from four up to a thousand. Their ability is related of learning and constructing designs with great complexity compared to 'traditional' neural networks. For instance, is processing visual data through deep learning, where the pixels are assigned into the first layers of deep neural network and the results are represented as a low- level features, as contours or lines.

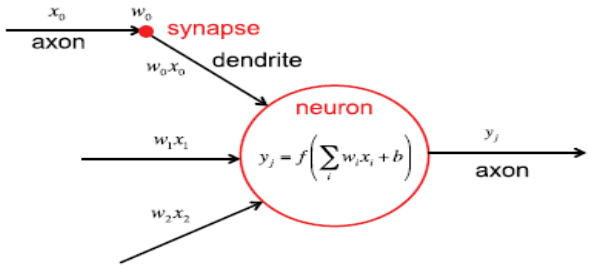



***Figure 15***. Connections to a neuron in the brain

### 3.6.1 History of DNN

Deep Neural Networks began as initial proposal nearly in 40s, where the first application assignments started not before than in late 70s, early 80s with Le-Net network for digit recognition. Later on, in 2010s an exploit in DNN application were noticed with Microsoft speech recognition, and Alex-Net design system in 2012 for image recognition. Below is the DNN breakthrough evolution through the years.

**DNN Timeline**

- 1940s – Neural network was proposed
- 1960s – Deep neural network was proposed
- 1989 – Neural networks for recognizing digits ( Le-Net )
- 1990s – Hardware for shallow neural nets ( Intel ETANN )
- 2011 – Breakthrough DNN-based speech recognition ( Microsoft )
- 2012 – DNNs for vision start supplanting hand-crafted approaches ( Alex Net )
- 2014 – now … - Rise of DNN accelerator research ( New-flow )

How deep learning neural network came to the phase as we know now? Lots of computational data, which led to the development of open source frameworks and combining this with the evaluation of the algorithmic techniques which highly improved application accuracy and significantly broadened the domains of DNNs area of field.

### 3.6.2 CNN based in image segmentation

The comprised methods of CNN in image segmentation, may alter from each-other in spite of dimensions size input, network depth, filter size, input size etc. Several methods were proposed for image segmentation like: Deep-Medic, FCN-8 and all of these methods had the same root architecture U-Net. It has been proved that CPU core can compute around 6 billion floating points operations. Compared to an average human brain this

amount of computation is likely unimaginable to be processed, even though there are no records of 100% power exploiting of human brain. Unlikely to traditional CPU-s human brain is able of computing lots of tasks per fraction of time for instance classification of images. That is the reason why even in the beginning of DNN development in 40s, researchers tried to imitate human brain, where this concept was called as Artificial Neural Network.

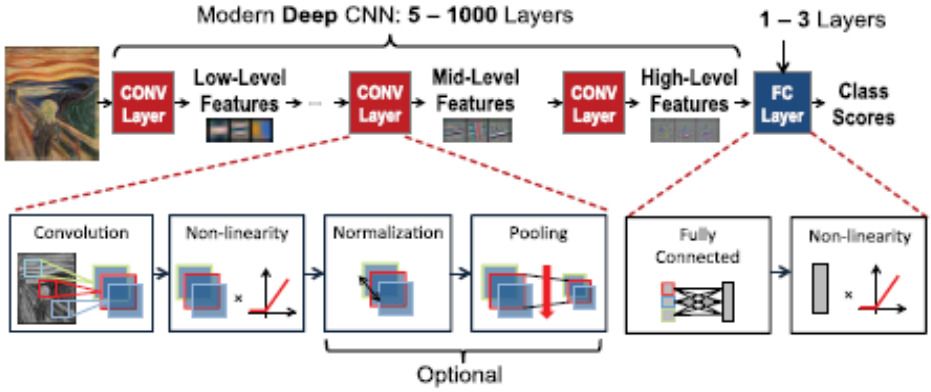



***Figure 16.*** CNN ( convolution neural network )

Each of the convolution layers of the CNNs produce a high level of abstraction name f-map ( feature map ), which conserves essential information. Nowadays, CNNs are able to perform in high level, while introducing a hierarchy layer. After the convolution of the CNN layers the input activators are structure in 2 dimension feature maps, called channel. Every channel is comprised of filter sets, unique for each and every channel, where many times this filter set is denoted as a 3-D filter. Therefore, the convolution products for every point are added together, where the result of the computation output is nonetheless but activation output, named output feature map. Moreover, all input feature maps are processed together as a batch, resulting in improvement of filter weights.

Additionally, there are other optional layers, as observed in the figure above like, nonlinearity ( generally it can evaluate the maximum value of two intersecting function ), pooling ( it makes the network to withstand to any invariance or distortion ) and

normalization which is nonetheless but, controlling the input distribution through the layers. It formula is as below :

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \varepsilon}}\gamma + \beta$$

$\gamma, \beta$ : parameters

$\varepsilon$: small constant

In convolution neural network the processes occur as below :

Firstly when the images arrive the computer is to much literal and unable to decide, therefore ConvNets matches all the pieces of the image, then filtering happens later on pooling ( max pooling ), normalization, ReLu, fully connected layer then learning. Below all the steps will be explained.

**FILTERING**

- All the features are lined up within the image patch.
- Each image pixel is being multiplied by the corresponding feature pixel.
- Later on they are all added up and then divided by the total number of pixels in the feature.

Then convolution happens which is the repeated application of this feature several times. The output is a map across the image where the feature occurs, and one image becomes a stack of filtered images.

**POOLING**

- It shrinks the image stack by picking a window size and a stride ( generally by 2 pixels).
- Use the window by going across the filtered images.
- Later on, from every window the maximum values is taken. ( Max Pooling )
- Perform "*max pooling*" within the stack.

**NORMALIZATION**

- If the pixel is a positive number it is left as it is, otherwise we set it 0. ( ReLu process )

**RELU ( rectified linear unit )**

- A stack of images becomes a stack of images with no negative values. After that *deep stacking* happens, where layers are repeated several time. Final layer is *fully connected layer* , the matrix 2x2 is rearranged and put in a single list, being easier to visualize. Therefore fully connected layer chooses the number of intermediate neurons.

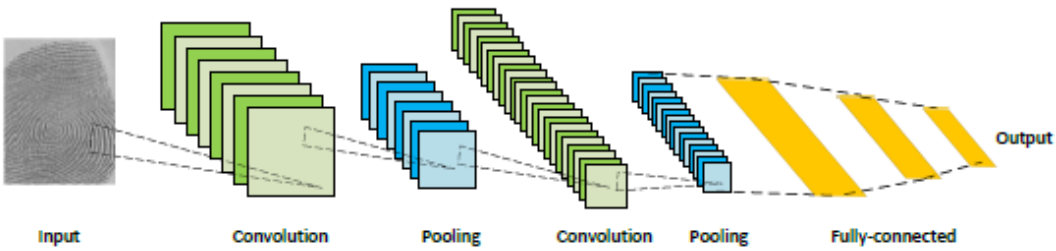



***Figure 17.*** Architecture of convolutional neural network

# CHAPTER 4

# DESIGN AND IMPLEMENTATION

## 4.1 SYSTEM ARCHITECTURE

It is essential in the system architecture that before setting up system implementation, we need to state and design blocks of the architecture. In the scope of work, we mentioned that we need two RAM-s, one as input and the next one as output, another block of random-access memory ( BRAM ) and video graphics-array ( VGA ), in order to display the resulting image. While designing in HDL in Xilinx Vivado 2018.2, when we create the module there is a set of scripts generated:

Different components of the routing architecture consists on different VHDL models. TCL scripts and pin location constraint files for VHDL code synthetization can reassure the tile-ability of the layout.

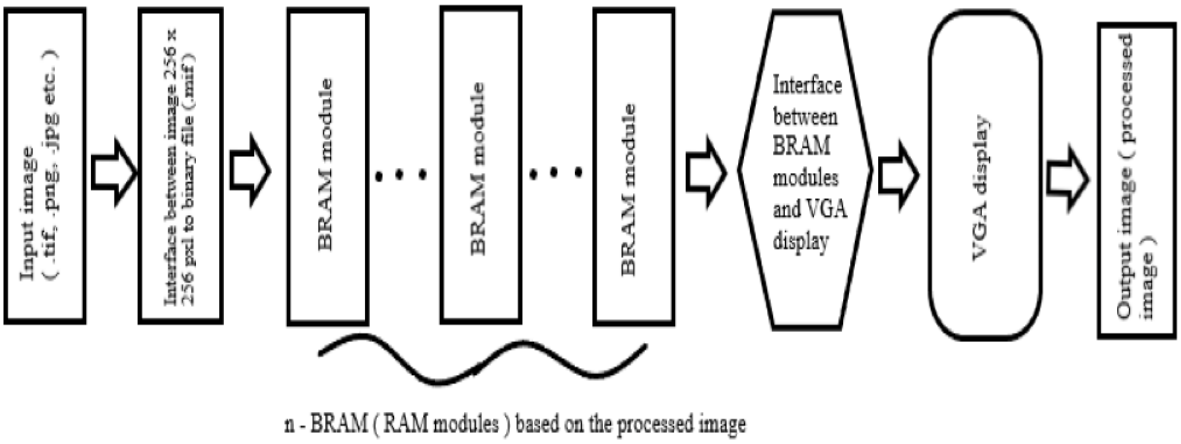



***Figure 18.*** General architecture overview

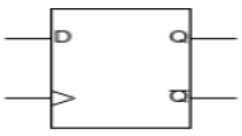
In our project Artix-7 Nexys 4 DDR FPGA board, we are going to implement an image processing of image size 256 x 256 pixels. The reason why we picked the image size in

256x256 pxl is because of the original image size. Its height and width varies from 480 – 540 pixels so in order to maintain a square size processed image we cropped it to the 256x256 pixel image Before stating the methodology, that we are going to imply, worth knowing that it wont be efficient toward large size processing images in real-time, we will introduce the cache design, cache coding, FIFO, kernel convolution.

In the above figure, all the stages where the image is processed are shown perfectly in order. It begins with the raw image ( .tif, .png, .jpeg ), then converted image to binary file ( readable for Vivado HLS 2018.2 framework, by means of converting interface to .hex file. After that, the converted image file is processed through n-BRAM modules, later on through interface between BRAM module as output and VGA display. Last but not least, the processed image goes into VGA display giving as output $H_{sync}$, $V_{sync}$, $h_{count}$, $v_{count}$, video_on. Between BRAM modules we will use 3 stages of Convolution Neural Network to train the image. For instance, from a set of images, the method is going train, process, identify and result an output image that is going to be an approximate resemblance of the original image.

### 4.1.1 Cache Design

Through implementation, the assignment of accessing 259 pixels of image subject has been applied. The assignment is divided in 2 subsets of operation: 1st one data fetching from the memory block and 2nd one saving that data fetched into memory units. In digital electronics, D-flip flop are substantial in memory block building. Its operation is quite simple, observing D-data input and at different clock cycle ( falling and raising ) the data output Q is shifted. Also, Q remains the same, for the data at same clock cycle shifting from input to output.



| Clock | D | $Q_{next}$ |
|---|---|---|
| Rising edge | 0 | 0 |
| Rising edge | 1 | 1 |
| Non-Rising | X | Q |

The structure of cache requires an 8-bit push pixel by pixel, for a serial to parallel design, and while all 259 are being pushed in the memory pipeline, then 9 pixels are pulled-out to the mask position. Input data is pushed from the first flip-flop FF1 assuming that all the flip-flops are synchronous sharing the same clock signal. Below figure, shows perfectly the concept of cache structure.

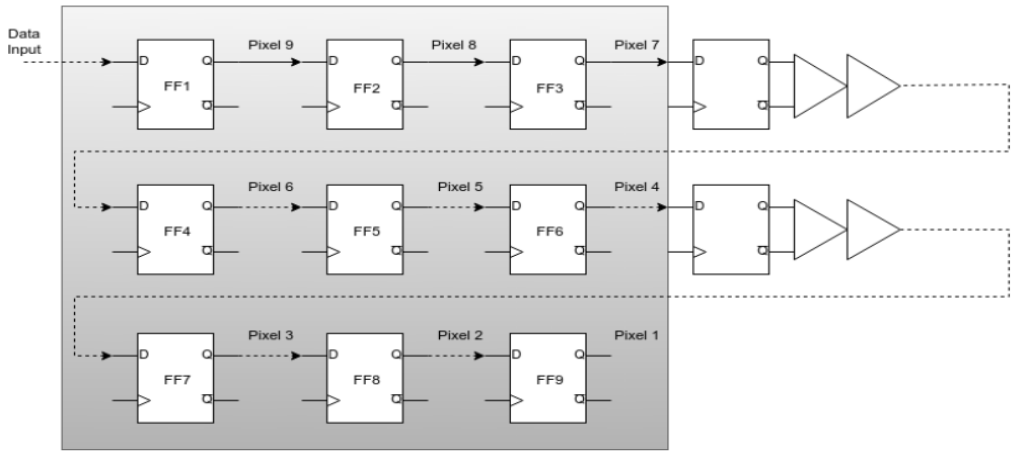



***Figure 19.*** Concept of cache structure

Even though, this methodology is quite relevant still it lacks in its convenience in saving all the pixels in flip-flop memory units, when the system is related to mask. Additionally, a more innovative method is using FIFO ( first-in first-out ), where all the pixels are saved in each row, with exception of pixels correlated to kernel_width.

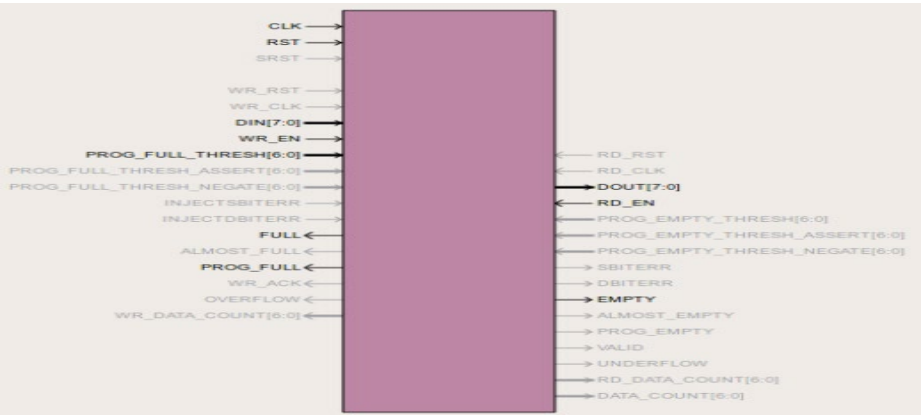



***Figure 20.*** FIFO schematic design

In the FIFO method there is a full threshold single programmable input port prog_full, which functions to shown when the byte exceed the value of threshold. As it is shown in the figure rd_en is being connected with prog_full by reading the data from port called 'dout'. Then the timing of byte is being synchronized with $128^{th}$ byte shifted in FF1. The 'prog_full_thresh' is remaining of width_img with 'kernel_width'. Generally, this is what happens inside FIFO structure method.

## 4.2 EXPERIMENTAL WORK

The goal is to design a FPGA based system, able to perform image processing assignments faster than traditional processors. The images which are in ( .tif, .jpg, .png file ) are converted by Matlab means in binary file, and the edges are detected using traditional method thresholding technique and later compared to EM / MPM based technique. For this assignment we are going to need : Nexys 4 DDR FPGA board, Display monitor, captured images, Xilinx Vivado software design

### 4.2.1 MATLAB

MatLab is a well known program among engineers and scientist. Million of them uses and trust Matlab everyday. Matlab combines a desktop environment tuned for iterative analysis and design processes with a programming language that expresses matrix and array mathematics directly. In Matlab, Live Editor can be used in order to create scripts that combine code , formatted text and output. MatLab is built professionally and all its toolboxes are rigorously tested. Creating a new algorithm in MatLab is very convenient because it let you see directly how your data are affected by the algorithms you are using. If the result that the engineer wants are not achieved directly iterative work

is done until the right result are achieved and after that MatLab generates directly a program to reproduce and automate all the work.

```
IP = imresize(I,[256 256]);  % Resizing the image to 256x256:
figure;  imshow (IP);
for i = 1:3 % 24-bit RGB image: we will convert it to a 12-bit RGB image:
   IN(:,:,i) = IP(:,:,i)/16; % every plane converted to 4 bits. right shift
end; figure; imshow(IN*16); % This is just so that 'im-show' can display the image
```

The image is being resized into 256 x 256 pixels, later on the 24-bit of RGB is converted into 12-bit RGB.

```
q = quantizer ('ufixed', 'round', 'saturate', [4 0]);
textfile = 'myimg.txt';
fid = fopen (textfile, 'wt'); % generates text file in write mode
```

After that, above is a small piece of code where the conversion of image, is being done in *mytext.file.*

### 4.2.2 Image read in VHDL

In signal processing assignments, it is substantial to load binary images in VHD implementation for different simulations. As mentioned above VHDL programming language is a technical language, and is not able to read jpg, tif, bmp files [13]. Therefore, the captured images, are required to be conveyed into binary text files using *use std.textio.all* library. The following function shows how to read the images [14].

```
-- by FPGA4student.com
impure function init_mem(mif_file_name : in string) return mem_type is
    file mif_file : text open read_mode is mif_file_name;
    variable mif_line : line;
    variable temp_bv : bit_vector(DATA_WIDTH-1 downto 0);
    variable temp_mem : mem_type;
begin
    for i in mem_type'range loop
        readline(mif_file, mif_line);
        read(mif_line, temp_bv);
        temp_mem(i) := to_stdlogicvector(temp_bv);
    end loop;
    return temp_mem;
end function;
```

We have modified the code where in the Nexys A7 FPGA board we have declared several components as input and as output : resetn ( input ), Switches ( 0 – 11 ) input, R-G-B ( 4-bit grayscale per each, total 12 ) output, and Hsync, Vsync as output. According to the architectural block schemes, there is BRAM as input and as output. BRAM is designed based on IP core function of Xilinx, and later on the images are introduced to be shown in VGA_display. There are different types of mode for VGA, and the one that we are going to use is 640x480, refresh frequency of 60Hz, 25Mhz frequency.

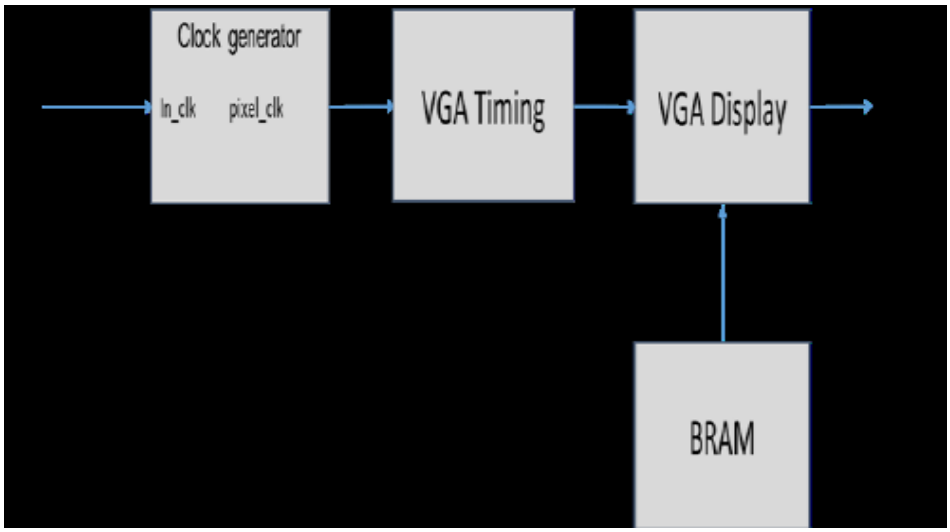



***Figure 21.*** Block Diagram of VGA display

VGA ( video- graphic- array ) its function is controlling the monitor, its standard is 15-pin connector, used to manipulate video devices. Its interface shows the way the information is sent through VGA device to our FPGA board. One of the most basic protocols, VGA is designed to be used with AC cathode ray tube. The electron beam transaction toward the screen left-right direction, for a certain refresh rate, for example 60Hz ( horizontally synchronization ) and also moves from bottom-up ( vertically synchronization ). We can also alter RGB ( red-green-blue colors ) on VGA interface according to what we wat to apply:

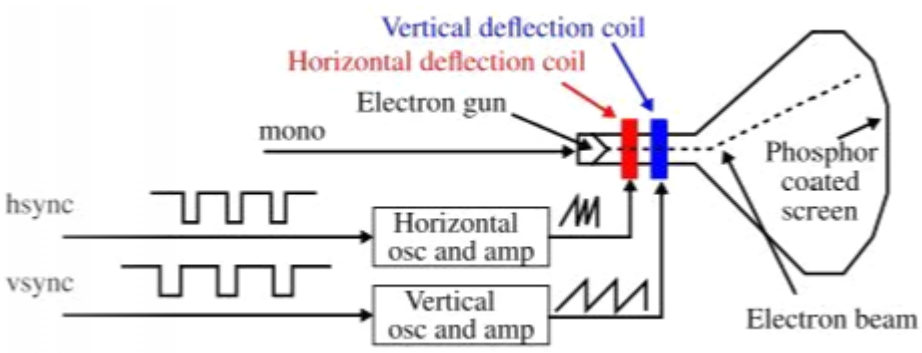



***Figure 22.*** VGA interface

VGA is connected to Nexys A7 diligent board through Xilinx Vivado with each and every signal of RGB. The composition of these three main colors give us all the colors used in everyday life. According to the below table, we are going to refer it in our design, changing background screen color with the colors we are interested to use:

In VGA, the video controller gives us output data and sync signals in serial way. Video_on ability is to display and image enable While implementing it in VHDL, we are going to use two types of counter 800-module, 500-module counter respectively.

| VGA_RED | VGA_GREEN | VGA_BLUE | Resulting Color |
|---|---|---|---|
| 0 | 0 | 0 | Black |
| 0 | 0 | 1 | Blue |
| 0 | 1 | 0 | Green |
| 0 | 1 | 1 | Cyan |
| 1 | 0 | 0 | Red |
| 1 | 0 | 1 | Magenta |
| 1 | 1 | 0 | Yellow |
| 1 | 1 | 1 | White |

***Figure 23.*** Colors expressed with VGA interface

```
constant bits_vga_tick: integer:= ceil_log2(clock_pixel_ratio);
constant HD: integer:=640; -- horizontal display area
constant HF: integer:=16; -- horizontal front porch
constant HB: integer:=48; -- horizontal back porch
constant HR: integer:=96; -- horizontal retrace
constant VD: integer:=480; -- vertical display area
constant VF: integer:= 10; -- vertical front porch
constant VB: integer:= 33; -- vertical back porch
constant VR: integer:=2; -- vertical retrace
```

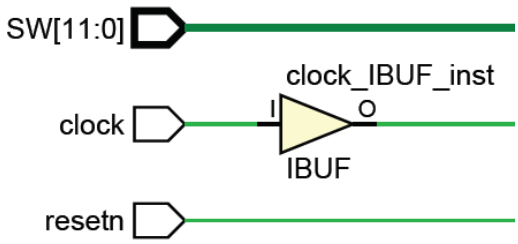
Later on, in the 'primarymodule.vhdl' code we have the combination of the read_file saved as *myimg.txt.*

## 4.3 Top module and blocks

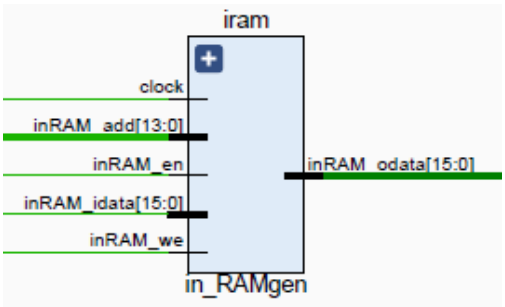
During our simulation the FPGA Nexys A7 board acts as "central unit", where all the operations regarding with displaying of "epoka.jpg" image varying different colors in background, sobel edge detection of image, segmentation and so forth occur.

***Figure 24.*** FPGA Nexys A7 board connected through VGA & USB cable with monitor and PC respectively

After we run the codes, in fact the synthesis, simulation takes a couple of hours, we are able to projectile the schematic design where we are going through each and every block:

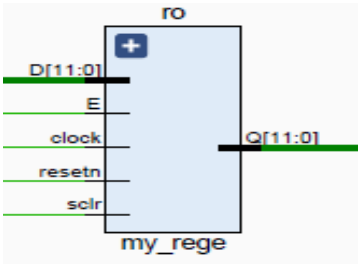


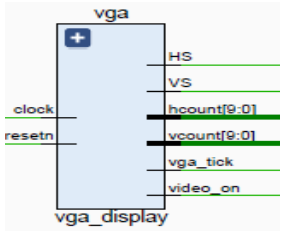

These are the declared input as in the primary_module consisting on : 12-switches, clock, resetn button. The reason behind 12-switches is 4-bit for each color ( since there are three RGB in total ) each pin in the constraint_file assigns to each color bit.



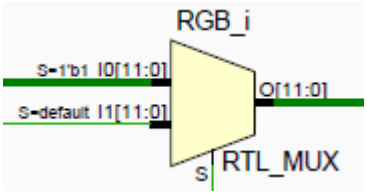
It is the main block ( BRAM ), where the important operations occur like : clock, RAM write, RAM read, RAM input enable. BRAM gives as output data stream of 16-bit data, going into the MUX ( multiplexer block ). The latter one works as a 'switch' for RGB, triggered by an AND logic circuit, giving the access in RGB display for the image, either away from image contours displaying the color of background from 12-bit switches.

After that we have another RAM block in output, functioning as regenerator and VGA display consisting in clock, resetn (input), HS, VS, hcount[9,0], vcount[9,0], vga_tick, video_on (output).

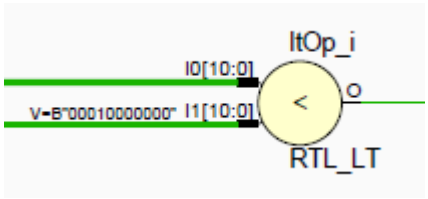



The last block module is what we call RGB_Mux, where takes input from the output BRAM ( shown above ) and ground ( reset ) purpose, enabled by video_on ( VGA display output ). In the output we are going to have Hsync, Vsync, 3-sets of array R[3,0], G[3,0], B[3,0].




The process of bit bit_regulator is being done from two comparator blocks as shown below. Its duty is enabling or resetting D flip-flop modules, and transmitting the bit array in BRAM input, and setting horizontal_counter and vertical_counter in gateway of VGA display block.




After simulation process is finished, our challenge lies in several factors like ; for lower power consumption in design circuit how it can be done, or reducing LUT, FF flip-flops for better efficiency, programming through combination of processes not from usual codes etc. These questions I will try to solve them later on.

## 4.4 RESULTS AND DISCUSSION

Our designed system enables all the features that a random image can generate. Below the original image of a flower is represented. Due, to specifics given to the code, every switch creates different visual image.

***Figure 25.*** Original image processed in Vivado IDE, hardware manager

After synthetization and implementation of the code, we obtain different images where each of them shows a property such are : sobel edge detection, color-image inverting, grayscale image, etc. Switch [0] enables the grayscale image , while switch [1] gives a better quality in grayscaled image ( first image up-left ). In the 2nd image ( up-right ) the features of edges are stressed even better by giving a higher specificity in the image. The 3rd image ( down-left ) is an image created based on sobel edge detector generated from switch [0], switch [1], switch [3] pressed on at the same time in the FPGA board. The sobel image is just an operator 3x3 kernel, where a convolution process occurs, and has horizontal & vertical approximation derivatives. A is the source image, $G_x$ and $G_y$ are the derivative approximations:

$$G_X = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \qquad G_Y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

While in the 4th image ( down-right ) generated by enabling switch [0], switch [1], switch [2], switch [4], where the stress is given into blue-color inverting image, generally the foreground color as pink is inverted into blue-color. It is worth mentioning that in all these processed images, convolution operations occur. The convolution is a mathematical operation where multiplication and addition of two functions produces a third modified function. As mentioned, all the 4 images below are the representation of convolution operations indicating different features that we want to stress out from the original image.

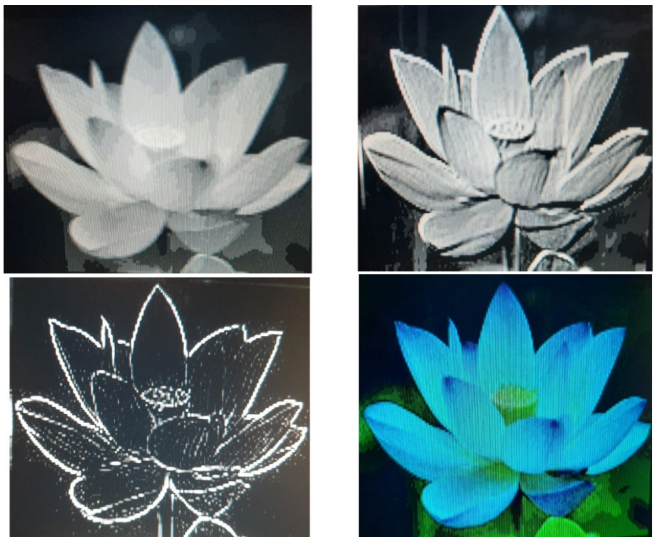


***Figure 26.*** Grayscale image, edge detector image, sobel edge detection, blue-color inverting image respectively

We intended to provide hardware engineering mentality while designing the system rather than the common used software engineering mentality. Therefore, while displaying the results it is obvious that when designing the system from hardware perspective wehave better energy preservation, fewer D-flip flops used, fewer LUTs and less BRAM blocks.

| Name | Constraints | Status | WNS | TNS | WHS | THS | TPWS | Total Power | Failed Routes | LUT | FF | BRAMs | URAM | DSP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ synth_1 | constrs_1 | synth_design Complete! | | | | | | | | 253 | 70 | 0.00 | 0 | 0 |
| ✓ impl_1 | constrs_1 | route_design Complete! | NA | NA | NA | NA | NA | 1.657 | 0 | 253 | 71 | 0.00 | 0 | 0 |

| Name | Constraints | Status | WNS | TNS | WHS | THS | TPWS | Total Power | Failed Routes | LUT | FF | BRAMs | URAM | DSP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ✓ synth_1 | constrs_1 | synth_design Complete! | | | | | | | | 28 | 31 | 0.00 | 0 | 0 |
| ✓ impl_1 | constrs_1 | route_design Complete! | NA | NA | NA | NA | NA | 1.026 | 0 | 28 | 31 | 0.00 | 0 | 0 |

First photo shows the total power, LUT, flip-flops from hardware perspective, the second photo is from software perspective. Beside that, we were able to utilize the difference in power efficiency, blocks, flip-flops etc. while designing two systems, for example: an “epoka.jpg” background altered in different colors, in RGB, corresponding to different switches, while the second image was a random image whose pixels assign white and black colors. The results indicate that we have a better energy conservation, fewer amount of blocks, LUTs and flip-flops used in image segmentation of images with different colors rather than white and black, lower energy change between successive pixels during 60Hz VGA refresh rate. In Sobel edge detection we used Sobel filter operator which uses two 3x3 kernels, which are convolved with the original image to calculate the approximations of the derivatives, horizontally and vertically change.

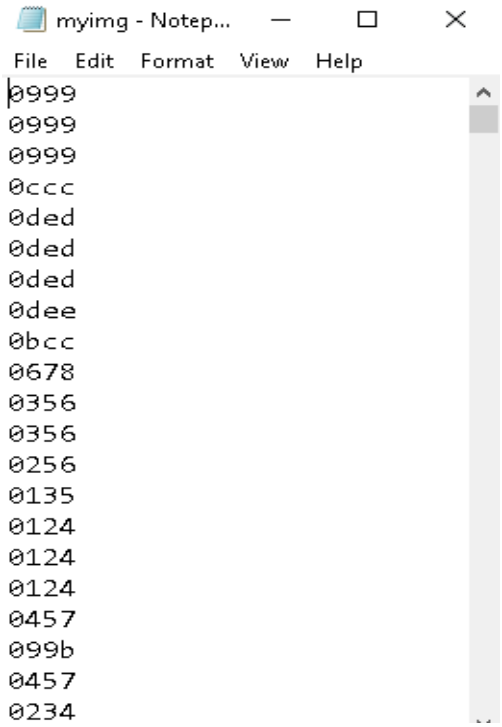



***Figure 27.*** Hex File

According to our results, the effectives of “hardware” is higher than that of “software” since it requires less utilization energy, fewer flip-flops, blocks, etc.

# CHAPTER 5

# CONCLUSIONS AND FUTURE RECOMMENDATION

## 5.1 CONCLUSIONS

Image processing is a hot topic nowadays, with a variety of application in everyday topic like : Computer Vision, Nanotechnology, IoT ( internet of things ), Medical image processing etc. Our aim, is for the captured images, we had to increase the specificity of contours, power consumption optimization for a higher efficiency and decreasing as many as possible the number of LUT-s, D flip-flops blocks.

In our thesis we went in a hardware implementation using Diligent Nexys A7 FPGA board through Xilinx Vivado 2018.2 software platform version. We compared two different mentalities hardware engineering mentality versus software engineering mentality, where we observed that “hardware” is faster, better, more optimized than “software”. I have analyzed the results coming from the comparison of two images segmentented. The first image had different intensity of colors, while the second image had gray color. From the results we acquired that greater amount of energy, higher blocks used, more flip-flops were required in the first image rather than the second one. It was associated with the greater change in intensity values of two corresponding pixels, thus requiring higher energy in performing image segmentation. In the first Chapter, I have started with the abstract of work ( design ) that is going to be implemented associated with acknowledgment, introduction in image processing where a general overview of image segmentation, its applications. After that, I will state my problem and give my objectives, motivation. Chapter 2 include papers that I have been analyzed, where I have identified the research gap. The paper researches that I have conducted are published during the last 3 years. Furthermore, I have stated the conclusions & future work of each paper. In this chapter, I am going to emphasize the hypotheses and research questions. Their aim is to

represent thesis objectives, directivities and answer some important questions related with the topic. This chapter begun with the general idea of image segmentation and the methods used to implement it. Then the focus was on the methodology used. I introduced FPGA and its framework, the features and its main buttons, HDL programming language ( vhdl and Verilog programming languages ). Chapter $4^{th}$ is considered to be one of the most important part in thesis, since all the experimental work is conducted. I successfully implemented several blocks using Nexys A7 framework in order to gain a higher power efficiency, where the input data ( image ) is converted into .hex file being programmable for our Nexys A7 board. Concepts like: reducing power consumption, reforming the code in order to optimize the number of flip – flops were introduced. As a result, from the data processed it is shown the necessity of FPGA processor nowadays, its trade-off related with time execution with greater optimization and better accuracy, and being less flexible, more complex.

We were able to identify and process images through Hardware Engineering mentality differently from software engineering mentality. During our methodology, we analyzed the total amount of LUT-s, power efficiency, time delay, D-flip flops, were nearly 9-times, 70%, ½-times respectively were lower during hardware-implementation rather than software-implementation.

## 5.2 FUTURE WORK & RECOMMENDATION

Future work lies in using multiple of CNN to affect the training of a larger amount of data. For sure, it will demand greater computational cost, higher GPU processor performance to withstand the operation. New GUI interface also will be designed to replace ImageJ tasks and a software interface which is used to connect with common medical image programs like Slicer or Visualization Toolkit (VTK) will be designed [14].

Moreover, better FPGA boards with greater abilities are required like FPGA Virtex V, VII boards etc. These boards are more adaptive related with usage of more layers convolution neural networks in making possible gathering larger amount of data trainable for higher and better image accuracy.

# REFERENCES


[1] P. Pooyoi, "Snow scene segmentation using cnn-based approach with transfer learning.," *2019 16th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON).,* pp. 50-62, 2019.

[2] F. H. S. R. R. U. D. M. R. M. T. C. C. M. B. A. G. C. &. M. Araújo, "Deep learning for cell image segmentation and ranking.," *Computerized Medical Imaging and Graphics,* pp. 13-21, 2019.

[3] "frontiers," [Online].

[4] "springeropen," [Online].

[5] "Frontier," [Online]. Available: https://www.frontiersin.org/.

[6] E. &. Ç. B. Nishani, "Computer vision approaches based on deep learning and neural networks: Deep neural networks for video analysis of human pose estimation.," *2017 6th Mediterranean Conference on Embedded Computing (MECO),* pp. 1-4, June, 2017.

[7] "Scholar Work," [Online].

[8] Y. S. L. C. Chao Liu, "3D EM/MPM Image Segmentation Using an FPGA," *Journal of,* 2015.

[9] O. Mazhar, "Real-time Image Processing using FPGA,," pp. 1-5, 15th December 2015.

[10] P. M. Aiken Pang, "Beginning FPGA:Programming metal," *Springer Science and Business Media LLC,* 2017.

[11] P. P. Chu, "FPGA Prototyping by VHDL Examples," 2008.

[12] "Xilinx," [Online].

[13] L. L. E. L. A. S.-V. S. Edwards, "Design of embedded systems: formal models, validation, and synthesis," 1997.

[14] P. P. Chu, "RTL hardware design using VHDL: coding for efficiency, portability, and scalability.," 2006.

[15] D. J. Smith, "VHDL and Verilog compared and contrasted-plus modeled example written in VHDL, Verilog and C.," *33rd Design Automation Conference Proceedings, 1996,* pp. 771-776.

[16] "Very Large Scale Integration (VLSI)," [Online].

[17] "fpga4student," [Online].

[18] S. Mittal, "Learning to Combine Top-Down and Bottom-Up Signals in Recurrent Neural Networks with Attention over Modules.," 2020.

[19] S. B. Y. P. F. P. A. K. N. &. T. D. Minaee, "Image Segmentation Using Deep Learning: A Survey," *arXiv preprint arXiv:2001.05566.,* 2011.

# APPENDIX A

## Matlab Code for converting image into .hex file

```
clear all; close all; clc;
I = imread ('epoka.jpg'); % RGB image
figure; imshow(I);
% Resizing the image to 256x256:
IP = imresize(I,[256 256]);
figure; imshow (IP);
% 24-bit RGB image: we will convert it to a 12-bit RGB image:
for i = 1:3
   IN(:,:,i) = IP(:,:,i)/16; % every plane converted to 4 bits. right shift
end
figure; imshow(IN*16); % This is just so that 'imshow' can display the image properly
% Converting to text file. Format: 0|R|G|B in hexadecimal
q = quantizer ('ufixed', 'round', 'saturate', [4 0]);
textfile = 'myimg.txt';
fid = fopen (textfile, 'wt'); % generates text file in write mode
for i = 1:256
   for j = 1:256
      R = IN(i,j,1); G = IN(i,j,2); B = IN(i,j,3);
      Rh = num2hex(q, double(R)); Gh = num2hex(q, double(G)); Bh = num2hex(q, double(B));
      fprintf(fid, '0%s%s%s\n',Rh, Gh, Bh);
   end
end
```

```
fclose (fid);
```

## ANEX VHDL code – Primary module

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library UNISIM;
use UNISIM.vcomponents.all;
library work;
use work.pack_xtras.all;
--use work.lpm_components.all;
-- Nexys-4 Board:
-- 12-bit color: R (4 bits), G (4 bits), B (4 bits)
-- 4-bit grayscale images: R=G=B
entity vga_ctrl_12b is
	generic (TYPE_CTRL: string:= "MEMORY"; -- "12BASIC" (2048 colors), "BASIC" (8 colors), MEMORY
					clock_pixel_ratio : integer:= 4); -- (Available clock)/25MHz (pixel clock)
							-- Only two available
	port ( clock: in std_logic;
			resetn: in std_logic; -- low level reset
			SW: in std_logic_vector (11 downto 0);
			R, G, B: out std_logic_vector (3 downto 0);
			HS, VS: out std_logic);
			--vga_clk: out std_logic;
```

```
                    -- debug signals
                    --video_on: out std_logic;
                    --hcount, vcount: out std_logic_vector (9 downto 0));
end vga_ctrl_12b;

architecture structure of vga_ctrl_12b is


    component vga_ctrl_ram
        generic (clock_pixel_ratio : integer:= 2;
                                    NPIXELS: INTEGER:= 256; -- Picture:
NPIXELSxNPIXELS. Max for Nexys-4: 525x525 -#-256

        -- So far, we only work with square images
                                    FILE_IMG: STRING:= "myimg.txt"; -- text file
containing the image
                                    nbits: integer:= 12);  -- number of bits for each pixel.
Example: 3, 8, 12, 15
        port ( clock: in std_logic;
                        resetn: in std_logic;
                        SW: in std_logic_vector (nbits-1 downto 0);
                        RGB: out std_logic_vector (nbits-1 downto 0);
                        HS, VS: out std_logic;
                        vga_clk: out std_logic;
                        -- debug signals
                        video_on: out std_logic;
                        hcount, vcount: out std_logic_vector (9 downto 0));
    end component;
```

```
  signal RGB: std_logic_vector (11 downto 0);
        signal vga_clk: std_logic;
        -- debug signals
        signal video_on: std_logic;
        signal hcount, vcount: std_logic_vector (9 downto 0);
begin
-- tested for 16, 32, 64,
gbc: if (TYPE_CTRL = "MEMORY") generate
                ramd: vga_ctrl_ram generic map (clock_pixel_ratio => clock_pixel_ratio,
NPIXELS => 256, FILE_IMG => "myimg.txt", nbits => 12) -- -#-256
                --ramd: vga_ctrl_ram generic map (clock_pixel_ratio => clock_pixel_ratio,
NPIXELS => 256 FILE_IMG => "myimgt.txt", nbits => 12) -- 128x128
                                port map (clock, resetn, SW, RGB, HS,VS, vga_clk, video_on,
hcount, vcount);
                        R <= RGB(11 downto 8); G <= RGB (7 downto 4); B <= RGB (3
downto 0);
         end generate;
end structure;
```

## VGA_generator VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
use ieee.math_real.log2;
use ieee.math_real.ceil;

entity my_genpulse_sclr is
```

```
	--generic (COUNT: INTEGER:= (10**8)/2); -- (10**8)/2 cycles of T = 10 ns --> 0.5 s
	generic (COUNT: INTEGER:= (10**2)/2); -- (10**2)/2 cycles of T = 10 ns --> 0.5us
	port (clock, resetn, E, sclr: in std_logic;
		Q: out std_logic_vector ( integer(ceil(log2(real(COUNT)))) - 1 downto
0);
		z: out std_logic);
end my_genpulse_sclr;
architecture Behavioral of my_genpulse_sclr is
	constant nbits: INTEGER:= integer(ceil(log2(real(COUNT))));
	signal Qt: std_logic_vector (nbits -1 downto 0);
begin
	process (resetn, clock)
	begin
		if resetn = '0' then
			Qt <= (others => '0');
		elsif (clock'event and clock = '1') then
			if E = '1' then
				if sclr = '1' then
					Qt <= (others => '0');
				else
					if Qt = conv_std_logic_vector (COUNT-1,nbits) then
						Qt <= (others => '0');
					else
						Qt <= Qt + conv_std_logic_vector (1,nbits);
					end if;
				end if;
			end if;
```

```
            end if;
        end process;
        z <= '1' when Qt = conv_std_logic_vector (COUNT-1,nbits) else '0';
Q <= Qt; end Behavioral
```