ISSN 2047-3338

High Level FPGA Modeling for Image Processing Algorithms Using Xilinx System Generator

Alareqi Mohammed[1], Elgouri Rachid[1,2] and Hlou Laamari[1]

[1]Laboratory of Electrical Engineering and Energy System.Faculty of Sciences,Ibn Tofail University,Kenitra, Morocco
[2]National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco
alareqi_mohammed@yahoo.com, elgouri.rachid@yahoo.fr

***Abstract*— This paper presents concept of hardware software co-simulation for image processing using Xilinx System Generator (XSG). This technique, provides a set of Simulink blocks (models) for several hardware operations that could be implemented on various Xilinx FPGA. This paper presents an efficient architecture for various image processing algorithms for negatives, image enhancement, contrast stretching, for grayscale and color images by using fewest possible System Generator Blocks. Performances of theses architectures implemented in FPGA card XUPV5-LX110T prototyping Virtex5 were presented.**

***Index Terms*— Image Processing, Xilinx System Generator and Field Programmable Gate Array (FPGA)**

I. INTRODUCTION

FPGAs have demonstrated many benefits in applications involving large and independent data sets as wells as the acquisition of the digital data itself. One fundamental use of an FPGA is digital signal processing (DSP).Traditionally, DSPs are implemented using ASICs however, many are now being replaced by FPGA's due to the decreasing cost and re-configurability.

As FPGAs continue to increase in chip density in accordance with Moore's law, the potential use has also expanded. The latest boards are equipped with a variety of I/O ports, AD/DA converters as well as a soft or hard core microprocessor; this allows the device to act as a system on chip. Furthermore, software/hardware co-designs are becoming increasingly popular as a solution to some very complex and intensive computations using FPGAs.

FPGA's are widely used to design applications that require high speed parallel data processing, such as image processing. Image processing on a computer allows for the manipulation of pixels within a digital image and can be altered to serve several purposes.

Application areas of signal processing have grown dramatically in importance in recent times, in parallel with the growth of powerful and low-cost processing chips. This has led, in turn, to many new applications, including multimedia delivery and hand-held communications delivery. Image processing is one an important application among them, which has a strong mathematical basis.

The quality of image is enhanced by using the image processing technique which is widely used in many areas such as medical, video surveillance, target recognition and robotics application.

The need to process the image in real time, which is time consuming, leads to this implementation in hardware level, which offers parallelism, and thus significantly reduces the processing time. FPGAs are increasingly used in modern imaging applications image filtering [1], [2], medical imaging [3], image compression[4],wireless communication[5]. The drawbacks of most of the methods are that they use a high level language for coding. In this paper, we study digital images and its processing techniques, specifically point processing algorithms. Digital images are electronics snapshots taken of a scene or scanned from documents, such as photographs, manuscripts, printed texts, and artwork. The digital image is sampled and mapped as a grid of dots or picture elements (pixels). The digital image is picture information in digital form. The image can be filtered to remove noise and obtain enhancement [3]. It can also be transformed to extract features for pattern recognition. The image can be compressed for storage and retrieval, as well as transmitted via a computer network or a communication system. Digital image processing has found application in wide variety of fields of human endeavor.

There are number of well defined processes which go to make up a typical image application. Acquisition, Enhancement, Restoration, Segmentation and Analysis are the steps needed by just about every application which involves image processing [4]. Once images are inside the computer system, or more specifically, once they are read inside a program, the images are nothing but matrices Hence, all the operations that can be applied to matrices should theoretically be applicable to the images as well. Image arithmetic is the implementation of standard arithmetic operations, such as addition, subtraction, multiplication, and division for images.

Image arithmetic has many uses in image processing, both as a preliminary step in more complex operations and by itself [5]. DSP functions are implemented on two primary platforms such as Digital Signal Processors (DSPs) and FPGAs [6]. FPGA is a form of highly configurable hardware while DSPs are specialized form of microprocessors. Most engineers prefer FPGA over DSP because of massive parallel processing capabilities inherent to FPGA and time to market make it the better choice. Since FPGAs can be configured in hardware, FPGAs offer complete hardware Customization while implementing various DSP applications.

System Generator [7] is a DSP design tool from Xilinx that enables the use of the Mathworks model-based design environment Simulink for FPGA design. It is a system level modeling tool in which designs are captured in the DSP friendly Simulink modeling environment using Xilinx specific Blockset. All of the downstream FPGA implementation steps including synthesis and place and route are automatically performed to generate an FPGA programming file. System Generator provides many features such as System Resource Estimation to take full advantage of the FPGA resources, Hardware Co-Simulation [8] and accelerated simulation through hardware in the loop co-simulation; which give many orders of simulation performance increase [9]. This objective lead to the use of Xilinx System Generator (XSG), a tool with a high- level graphical interface under the Matlab, Simulink based blocks which makes it very easy to handle with respect to other software for hardware description [10].

II. XILINX SYSTEM GENERATOR

System Generator is a DSP design tool from Xilinx that enables the use of the MathWorks model-based Simulink design environment for FPGA design [11]. The design tools facilitate the design processes by obscuring the technical knowledge necessary for FPGA a Register Transfer Level (RTL) design. Instead, a design is modeled using the intuitive visual environment within Simulink that uses several specific block sets accelerate the development. Additionally, System Generator can perform the FPGA implementation steps: synthesis, mapping, and place and route to generate the FPGA executable file.

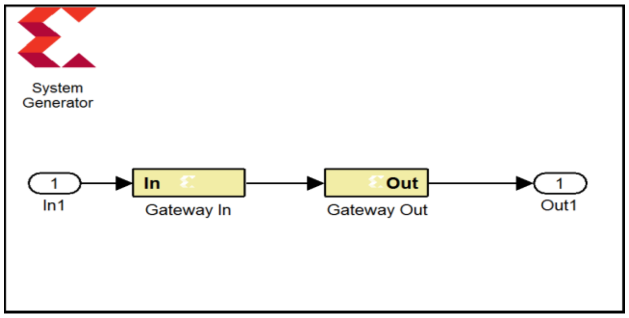


Figure 1: model system generator within Simulink

The Fig. 1 illustrates how a project could be modeled within Simulink. The Gateway In and Gateway Out blocks define the input and output from the FPGA respectively. Outside of this region, Simulink can be used to send customized signals and data into the simulated FPGA to observe the results. The System Generator block defines which type of FPGA board will be used, as well as provide several additional options for clock speed, compilation type and analysis.

With a library of over 90 DSP building blocks, System Generator allows for faster prototyping and design from a high-level programming stand point. Some blocks such as the M-code and Black box allow for direct programming in MATLAB M-code, C code, and Verilog to simplify integration with existing projects or customized block behavior. System Generator projects can also easily be placed directly onto the FPGA as an executable bitstream file as well as generating Verilog code for additional optimizations or integration with existing projects within the Xilinx ISE environment.

III. DESIGN FLOW FOR IMAGE PROCESSING WITH XILINX SYSTEM GENERATOR

For accomplishing Image processing task using Xilinx System Generator needs two Software tools to be installed. One is MATLAB Version R2011a or higher and Xilinx ISE 14.1. The System Generator token available along with Xilinx has to be configured to MATLAB. This result in addition of Xilinx Block set to the Matlab Simulink environment which can be directly utilized for building algorithmic model.

The algorithms are developed and models are built for image negative, enhancement etc… using library provided by Xilinx Blockset. The image pixels are provided to Xilinx models in the form of multidimensional image signal or R|G|B separate color signals in the form of vector in Xilinx fixed point format.

These models are simulated in Matlab Simulink environment with suitable simulation time and simulation mode and tested.

The reflected results can be seen on a video viewer. Once the expected results are obtained System Generator is configured for suitable FPGA board. FPGA board that used here is Virtex5. I/O planning and Clock planning is done and the model is implemented for JTAG hardware co-simulation. The System generator parameters are set and generated. On compilation the netlist is generated and a draft for the model and programming file in VHDL is created which can be accessed using Xilinx ISE. The module is checked for behavioral syntax check, synthesized and implemented on FPGA. The Xilinx System Generator itself has the feature of generating User constraints file (UCF), Test bench and Test vectors for testing architecture. Xilinx System Generator (XSG) has created primarily to deal with complex Digital signal processing (DSP) applications, but it has other application of this theme such as image processing also work with it. Bitstream compilation is done which is necessary to create an FPGA bit file which is suitable for FPGA input. The Fig. 2 shows the Design flow for Xilinx System Generator.

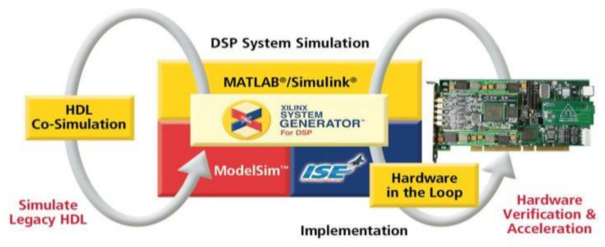


Figure 2: Design flow for Xilinx System Generator

IV. METHODOLOGY OF PROPOSED HARDWARE IMPLEMENTATION OF IMAGE PROCESSING METHOD

The image processing method need to be implemented in hardware in order to meet the real time applications. FPGA implementation can be performed using prototyping environment using Matlab/Simulink and Xilinx System Generator tool. The design flow of hardware implementation of image processing using XSG is given in Fig. 3.

Image source and image viewer are simulink block sets by using these blocks image can give as input and output image can be viewed on image viewer block set. Image pre-processing and image post-processing unite are common for all the image processing applications which are designed using Simulink blocksets.

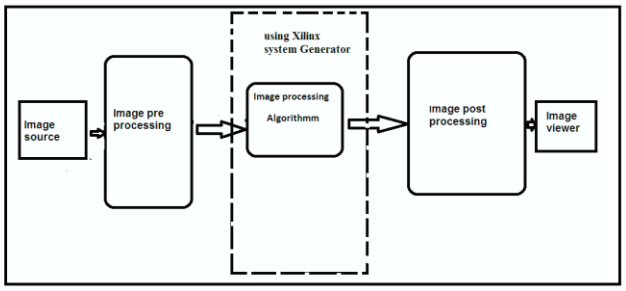


Figure 3: Design flow of hardware implementation of image processing

A. Image Pre-Processing Unit

Image preprocessing in Matlab helps in providing input to FPGA as specific test vector array which is suitable for FPGA Bitstream compilation using system generator.

- Resize: Set Input dimensions for an image and interpolation i.e. bicubic it helps in preserving fine detail in an image.
- Convert 2-D to 1-D: Converts the image into single array of pixels.
- Frame conversion and buffer: It helps in setting sampling mode and buffering of data.

The model based design used for image pre-processing is shown in Fig. 4. The blocks utilized here are discussed. Input images which could be color or grayscale are provided as input to the File block.

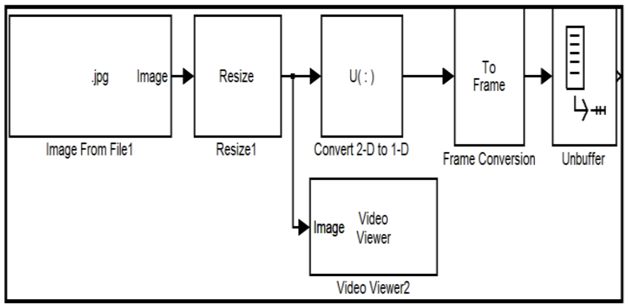


Figure 4: Image Pre-processing unit

B. Image Post-Processing Unit

Image post processing helps recreating image from 1D array.

Post-processing uses (Fig.5):

- Data type conversion: It converts image signal to unsigned integer format.
- Buffer : Converts scalar samples to frame output at lower sampling rate.
- Convert 1D to 2D: Convert 1D image signal to 2D image matrix.
- Video viewer: It is used to display the output image back on the monitor.

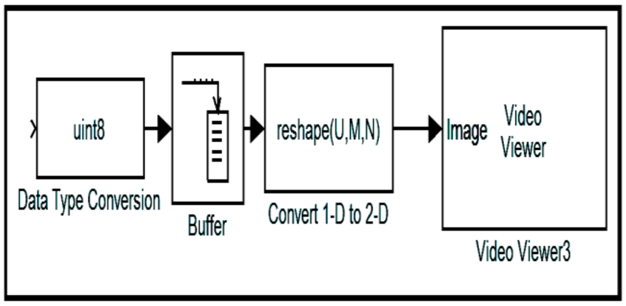


Figure 5: Image Post processing unit

V. XILINX MODELS FOR OPERATIONS OF IMAGE PROCESSING

Development of models is based on algorithms used for Image Processing. Some of Basic Algorithms that are mentioned above are described below. Once the FPGA boundaries have been established using the Gateway blocks, the DSP design can be constructed using blocks from the Xilinx DSP block set.

A. Algorithm for Negative Image

The negatives digitized images are useful in many applications, such as medical imaging and representation in photographs of a monochrome screen with films with the idea of using the resulting negative slides as normal.

Inverting the sample values in image produces the same image that would be found in a film negative (Fig. 6). In

Matlab this operation can be obtained by XOR function block or simple Inverter block or by Add sub block.

1. *Image Negative using XOR Operations*

The Exclusive OR function sets bits that are the same in each operand to 0 and bits that are different to 1. All pixels of a certain value can be found by applying XOR function. Almost a negative image produced by XOR with 255 to the image. Fig. 7 and Fig. 8, show the results of an image with XSG bloc.

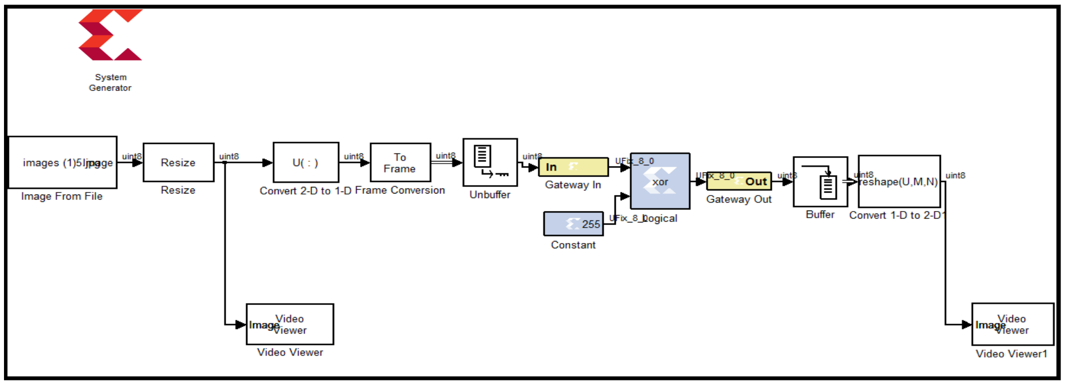


Figure 6: Algorithm for Image Negative using XOR Block XSG

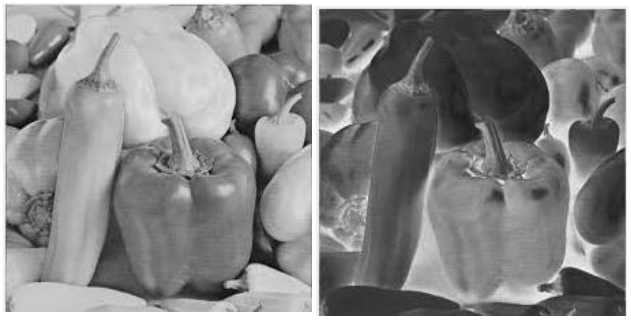


Figure 7: Results for Grayscale Image Negative

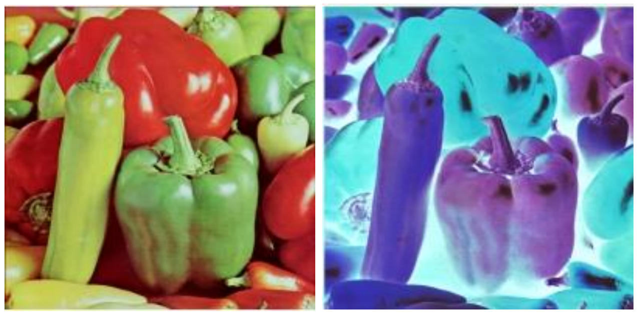


Figure 8: Results for color Image Negative

2. *Image Negative using NOT Operations*

In order to see the negative of the image, we need to change the values in the image matrix to double precision (invert all pixel of matrix). This is done to invert the color matrix, so we can achieve this by using the gate NOT which available in Xilinx system generator. Fig.10 and Fig. 11 show the results of an image with XSG blocks.

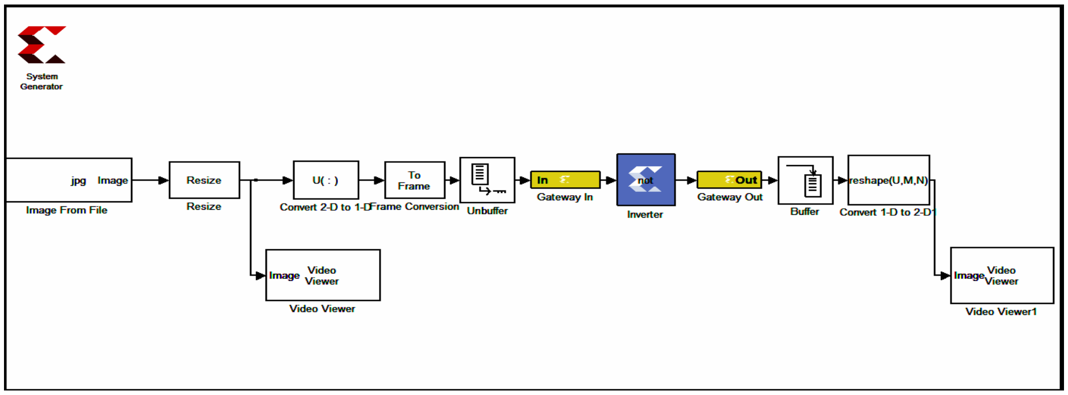


Figure 9: Algorithm for Image Negative using NOT Block XSG

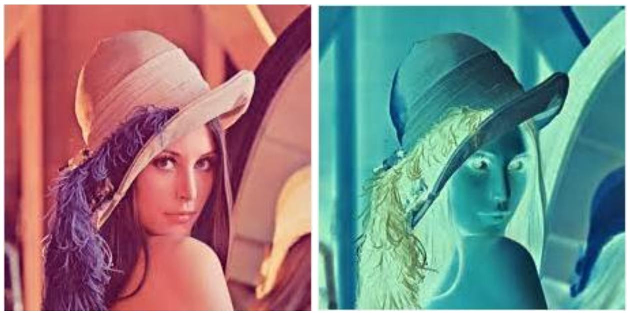

Input Image Output Image

Figure 10: Results for color Image Negative

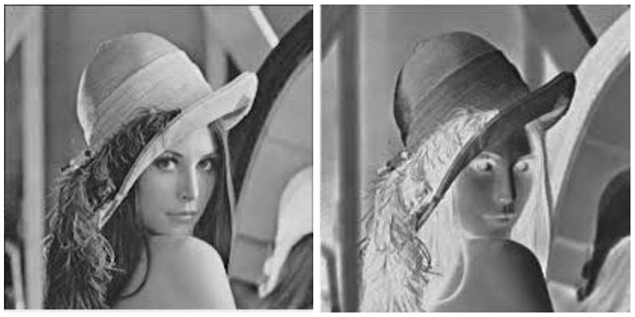

Input Image Output Image

Figure 11: Results for Grayscale Image Negative

3. Image Negative using Add sub Block

The negative transform exchanges dark values for light values and vice versa. This is the complement of a grayscale image like a photographic negative.

The equation is as follows:

$$Image\ Neg.\ (I, J) = max\_gray - image\ (I, J) \qquad (1)$$

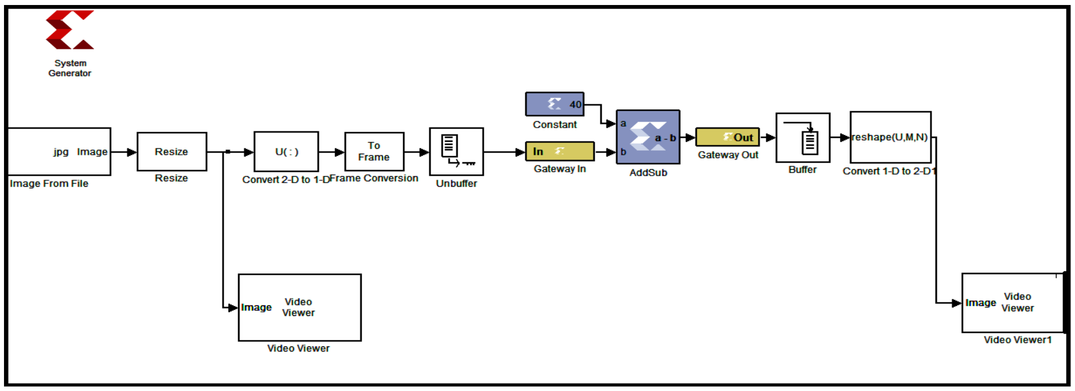


Figure 12: Algorithm for Image Negative using Add sub Block

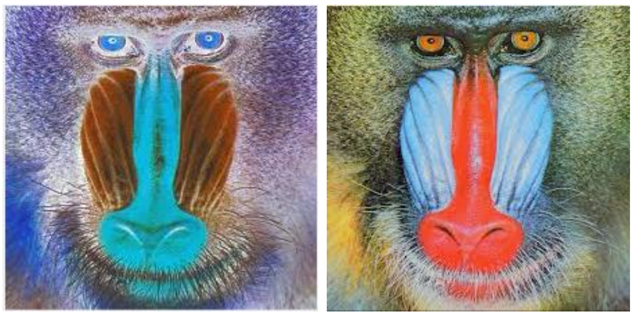

Input Image Output Image

Figure 13: Results for Color Image Negative

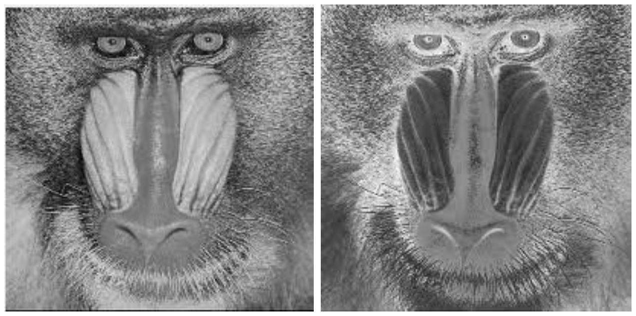

Input Image Output Image

Figure 14: Results for Grayscale Image Negative

B. Algorithm for Image Enhancement

In this we shows that how image can be enhanced by adding a constant to each pixel values. Image filtering can also be done using model based design different filtering architecture can be defined and Xilinx block can be created. Figure 15 to Figure 18 show the simulation model with its result.

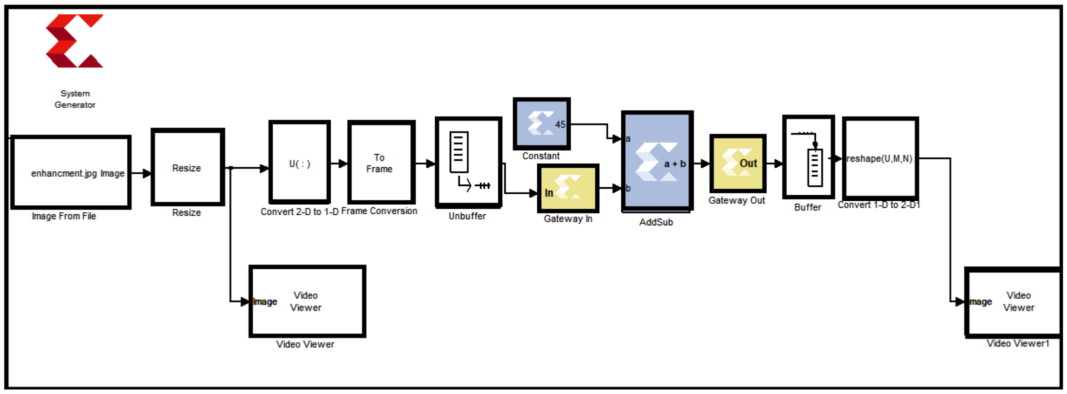
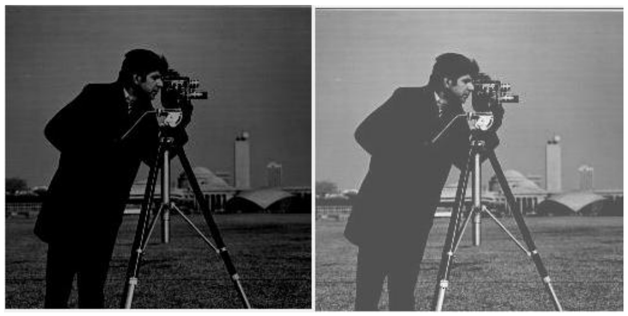


Figure 15: Algorithm for Grayscale Image Enhancement



Input Image Output Image

Figure 16: Results for Grayscale Image Enhancement

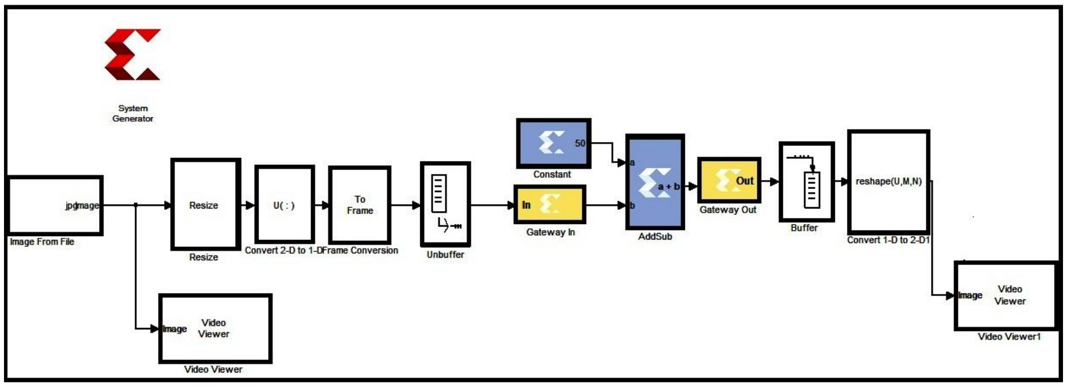
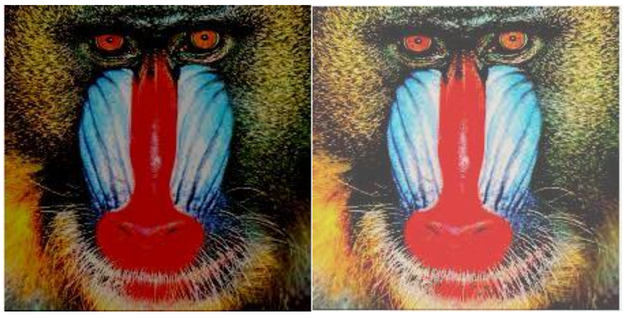


Figure 17: Algorithm for Color Image Enhancement



Input Image Output Image

Figure 18: Results for color Image Enhancement

C. Algorithm for Image Contrast stretching

The contrast of an image is its distribution of light and dark pixels. To stretch a histogram, contrast stretching is applied to an image to fill the full dynamic range of the image. We can stretch out the gray levels in the center of the range by applying piecewise linear function according to the equation.

$$New\ pixel = 3\ (old\ pixel\text{-}5) + 2 \qquad (2)$$

where new pixel is its result after the transformation. Fig.19, 20 and 21, shows the XSG blocks for the above contrast stretching to the image and the results respectively.

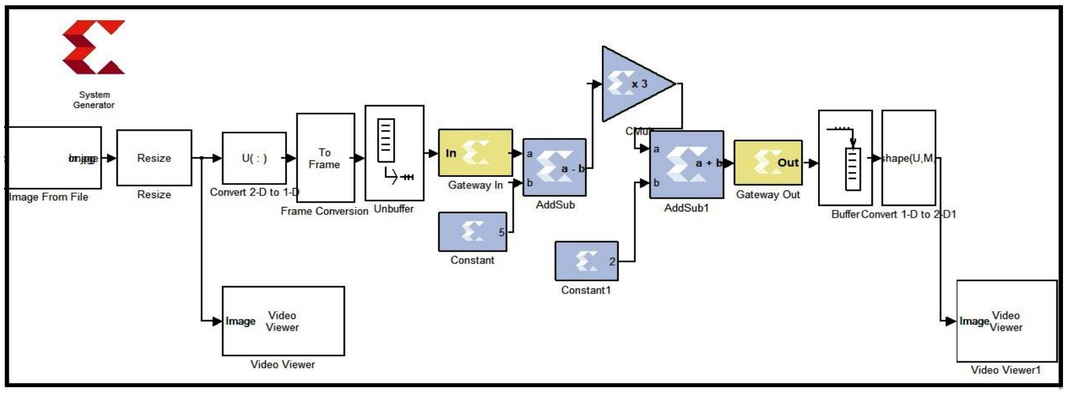


Figure 19: Algorithm for image Contrast stretching

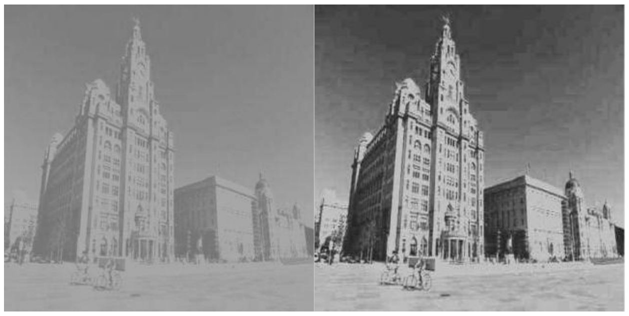


Figure 20: Results for Grayscale Image Contrast stretching

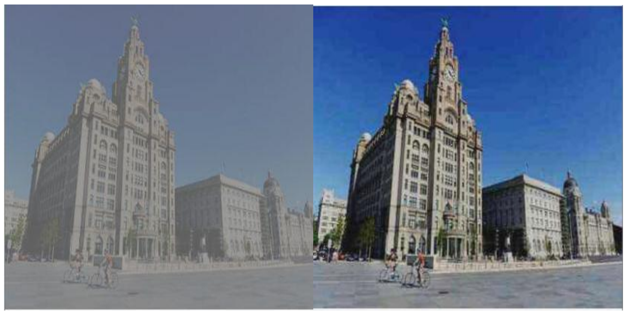


Figure 21: Results for color Image Contrast stretching

VI. HARDWARE CO-SIMULATION

Once your hardware board is installed, the starting point for hardware co-simulation is the System Generator model or subsystem you would like to run in hardware. A model can be co-simulated, provided it meets the requirements of the underlying hardware board. This model must include a System Generator token; this block defines how the model should be compiled into hardware. The first step in the flow is to open the System Generator token dialog box and select a compilation type under Compilation. Steps Followed in Hardware Co-simulation System generator is configured as:

A. Choosing a Compilation Target

- Part: Defines the FPGA part to be used (Virtex5 XUPV5-LX110T). Resulting library is created as follows:

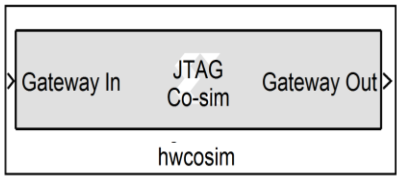


Figure 22: Hardware co-simulation block

- Synthesis tool: Specifies the tool to be used to synthesize the design.
- Hardware Description Language: Specifies the HDL language to be used for compilation i. e Verilog.
- Create test bench: This instructs System Generator to create a HDL test bench.
- Design is synthesized and implemented.

B. Clocking Tab

- FPGA clock period(ns): Defines the period in nanoseconds of the system clock
- Clock pin location: Defines the pin location for the hardware clock.

C. Invoking the Code Generator

- The code generator is invoked by pressing the Generate button in the System Generator token dialog box.

VII. CONCLUSION

The Xilinx System Generator tool is a new application in image processing and offers a friendly environment design for the processing, because processing units are designed by blocks.

This tool support software simulation, but the most important is that can synthesize in FPGAs hardware, with the parallelism, robust and speed, this features are essentials in image processing.

In this paper, a real-time image processing algorithms are implemented on FPGA. Implementation of these algorithms on a FPGA is having advantage of using large memory and embedded multipliers. Advances in FPGA technology with the development of sophisticated and efficient tools for modeling, simulation and synthesis have made FPGA a highly useful platform.

The design architecture used in this paper can be used for all Xilinx FPGA Kit with proper user configuration in system generator block and could be extended to real time image processing.

REFERENCES

[1]. S. Hasan, A. Yakovlev, and S. Boussakta, "Performance efficient FPGA implementation of parallel 2-D MRI image filtering algorithms using Xilinx system generator" *IEEE International Conference on Communication Systems Networks and Digital Signal Processing (CSNDSP)*, pp. 765–769, July2010.

[2]. R. Harinarayan, R. Pannerselvam, M. Mubarak Ali, and D. Kumar Tripathi, "Feature extraction of Digital Aerial Images by FPGA based implementation of edge detection algorithms" *IEEE International Conference on Emerging Trends in Electrical and Computer Technology* (ICETECT), pp.631 – 635, March 2011.

[3]. C.John Moses, D. Selvathi, S.Sajitha Rani, "FPGA Implementation of an Efficient Partial Volume Interpolation for Medical Image Registration" *IEEE International Conference on Communication Control and Computing Technologies*(ICCCCT-10), pp.132–137,Oct.2010.

[4]. M.F. Bin Othman , N. Abdullah, and N.A. Bin Ahmad Rusli ,"An Overview of MRI Brain Classification using FPGA Implementation" *IEEE Symposium on Industrial electronics & Applications* (ISIEA),pp.623 – 628, Oct. 2010.

[5]. H. Taha, A.N.Sazish, A.Ahmad, M. S.Sharif, and A. Amira, "Efficient FPGA Implementation of a Wireless Communication System Using Bluetooth Connectivity" *IEEE International Symposium Circuits and Systems* (ISCAS), pp.1767-1770, June 2010.

[6]. M.Ownby and W.H.Mahmoud,"A design methodology for implementing DSP with Xilinx System Generator for Matlab," *IEEE International Symposium on System Theory*, pp.404-408, March 2003.

[7]. Xilinx Inc., "System Generator for Digital Signal Processing: http://www.xilinx.com / tools / dsp.htm.

[8]. T. Saidani, D. Dia, W. Elhamzi, M. Atri and R.Tourki, "Hardware Co-simulation for Video Processing Using Xilinx System Generator," *Proceedings of the World Congress on Engineering*, vol.1, Jun 2009. London, U.K

[9]. P.Karthigaikumar, K.Jaraline Kirubavathy and K.Baskaran, "FPGA based audio watermarking–Covert communication," *Microelectronics Journal*, vol.42, pp.778-784, Feb 2011.

[10]. A. T. Moreo, P. N. Lorente, F. S. Valles, J. S. Muro and C. F. Andres, «Experiences on developing computer vision hardware algorithms using Xilinx system generator" *Microprocessors and Microsystems,* Vol. 29, pp.411-419 November 2005.

[11]. Xilinx System Generator User's Guide, www.Xilinx.com.