IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 2, Issue 4 (May. – Jun. 2013), PP 26-36
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

FPGA Implementation for Image Processing Algorithms Using Xilinx System Generator

Neha. P. Raut[1] , Prof.A.V.Gokhale[2]
[1]P G Scholar VLSI, Department of Electronics Engineering,[1,2]Yeshwantrao Chavan College of Engineering Nagpur, Maharashtra, India

Abstract— *The paper presents information about FPGA implementation for various Image Processing Algorithms using the most efficient tool called Xilinx System Generator (XSG) for Matlab. System Generator is a DSP design tool from Xilinx that enables the use of the Math Works model-based Simulink design environment for FPGA design. In this paper various morphological and intensity image processing algorithms for negatives, image enhancement, threshold,, contrast stretching, Edge detection, boundary extraction for grayscale and color images are explored. Use of Xilinx system generator for image processing effectively reduces intricacy in structural design also provides additional feature for hardware co-simulation*

Index Terms*—FPGA Implementation, Xilinx System Generator, Matlab, Simulink, Co-simulation.*

I. INTRODUCTION

Over the past decade, the field of image analysis research has undergone a rapid evolution. Image processing Now-a-days have varied applications in the fields of medical imaging, whether meteorology, computer vision, digital photography, microscopy etc. Super-Resolution Imaging consolidates key recent research contributions from eminent scholars and practitioners in this area and serves as a starting point for exploration into the state of the art in the field. Recent advances in camera sensor technology have led to an increasingly larger number of pixels being crammed into ever-smaller spaces. This has resulted in an overall decline in the visual quality of recorded content, necessitating improvement of images through the use of post-processing.

This paper particularly features on developing suitable method for rapid and efficient way to perform hardware implementation for some of Basic crucial image processing algorithms that can be used in simple application specific devices. Image quality can be enhanced by some of basic morphological and intensity image transforms such as controlling it's illumination, contrast stretching, thresholding similarly some applications needs image segmentation via edge detection, boundary extraction, image negatives or extraction of positive from image negatives , image subtraction etc. these algorithms are focused in this paper.

This paper aims at (1) Developing algorithmic models in MATLAB using Xilinx Blockset for specific role. (2) Creating workspace in MATLAB to process image pixels in the form of multidimensional image signals for input and output images (3) Performing hardware implementation of given algorithms on FPGA.

II. XILINX SYSTEM GENERATOR

System Generator is part of the ISE® Design Suite and provides Xilinx DSP Blockset such as adders, multipliers, registers, filters and memories for application specific design. These blocks leverage the Xilinx IP core generators to deliver optimized results for the selected device. Previous experience with Xilinx FPGAs or RTL design methodologies is not required when using System Generator. Designs are captured in the DSP friendly Simulink modelling environment using a 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. Advantage of using Xilinx system generator for hardware implementation is that Xilinx Blockset provides close integration with MATLAB Simulink that helps in co-simulating the FPGA module with pixel vector provided by MATLAB Simulink Blocks[3].

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 R2010a. Or higher and Xilinx ISE 13.1. The System Generator token available along with Xilinx has to be configured to MATLAB. This results in addition of Xilinx Blockset 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 can be used here is Spartan6 xc6xls16-3csg324 or Virtex6. 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 Verilog HDL 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 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.1 shows the Design flow for Xilinx System Generator.

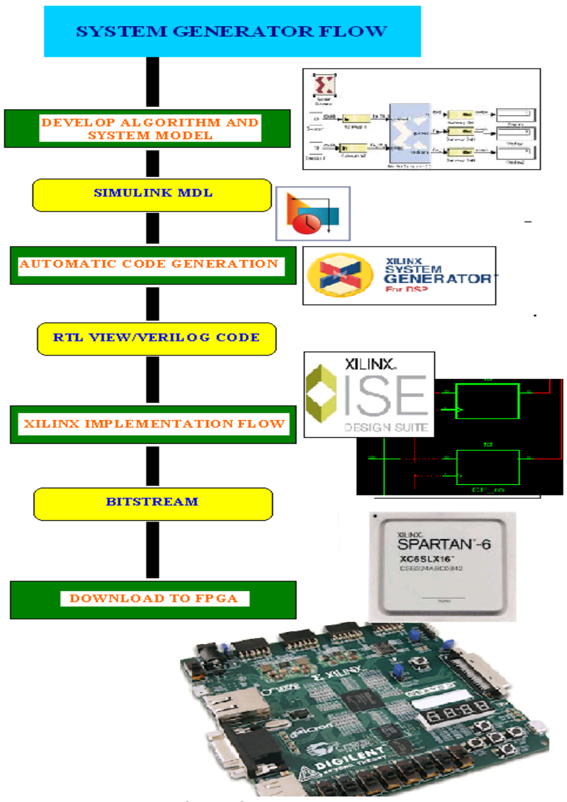
SYSTEM GENERATOR FLOW
DEVELOP ALGORITHM AND SYSTEM MODEL
SIMULINK MDL
AUTOMATIC CODE GENERATION
XILINX SYSTEM GENERATOR For DSP
RTL VIEW/VERILOG CODE
XILINX ISE DESIGN SUITE
XILINX IMPLEMENTATION FLOW
BITSTREAM
XILINX SPARTAN-6 XC6SLX16
DOWNLOAD TO FPGA

Figure 1: Design flow for Xilinx System Generator

IV. IMAGE PRE-PROCESSING AND POST-PROCESSING

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 Figure 2 the blocks utilized here are discussed. Input images which could be color or grayscale are provided as input to the File block.

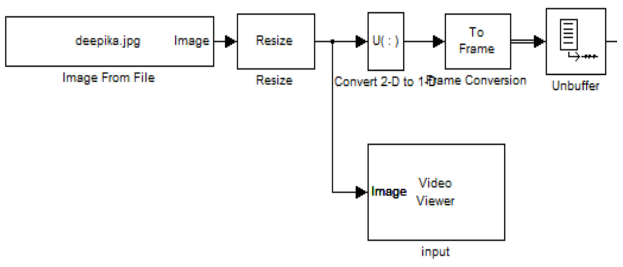

Figure 2: Image Pre-processing

In case of 5X5 Filter generation the system is clocked 1/5 times with normal clocking. The internal delay is occurred in this block which causes certain rows at the bottom of the image to be shifted to the top. To avoid this error the image is translated at the bottom for same number of rows. This helps in retrieving the entire image after filtering.

Image post processing helps recreating image from 1D array.
Post-processing uses-

- 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.
- Sink: It is used to display the output image back on the monitor.

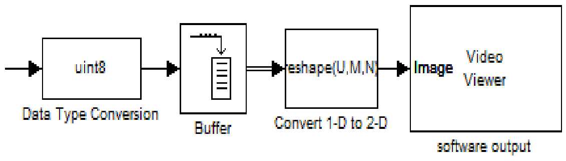

Figure 3: Image Post processing

V. XILINX MODELS FOR IMAGE PROCESSING ALGORITHMS

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. Standard Simulink blocks are not supported for use within the Gateway In/Gateway Out blocks.

A. Algorithm for Grayscale Image Negative

Inverting the sample values in image, produces the same image that would be found in a film negative. In Matlab this operation can be obtained by simple Inverter block or by Addsub block by subtracting one input by a constant 255.

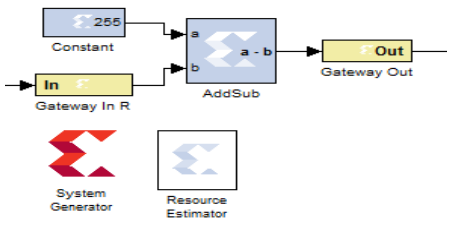

Figure 4: Algorithm for Image Negative using Addsub Block for Grayscale Images

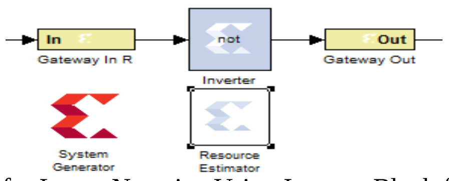

Figure 5: Algorithm for Image Negative Using Inverter Block for grayscale images

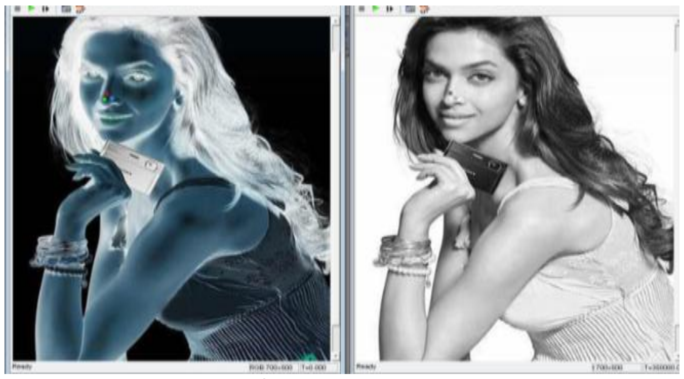

Figure 6: Results for Grayscale Image Negative

B. Algorithm for Color Image Negative

For color images the algorithm is similar but implemented for multidimensional R|G|B signals.

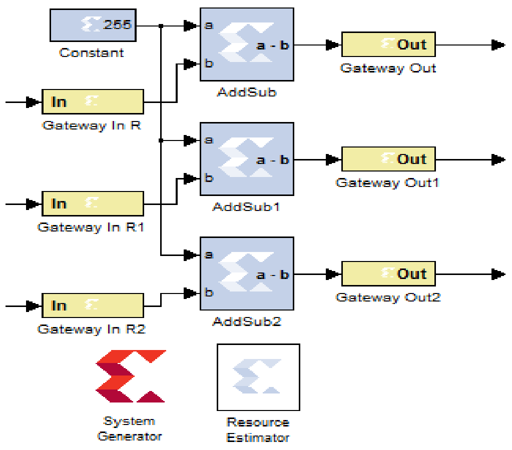

Figure 7: Algorithm for Color Image Negative using Addsub Block

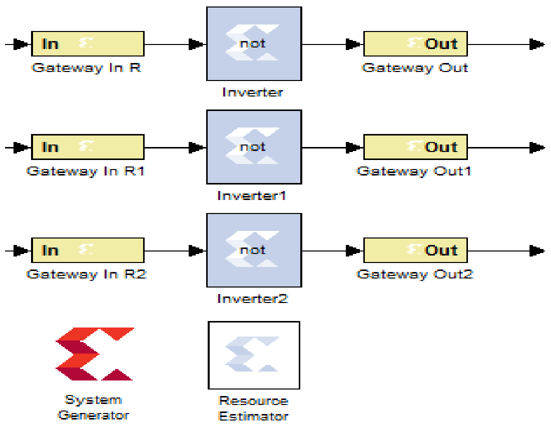

Figure 8: Algorithm for Color Image Negative using Inverter Block

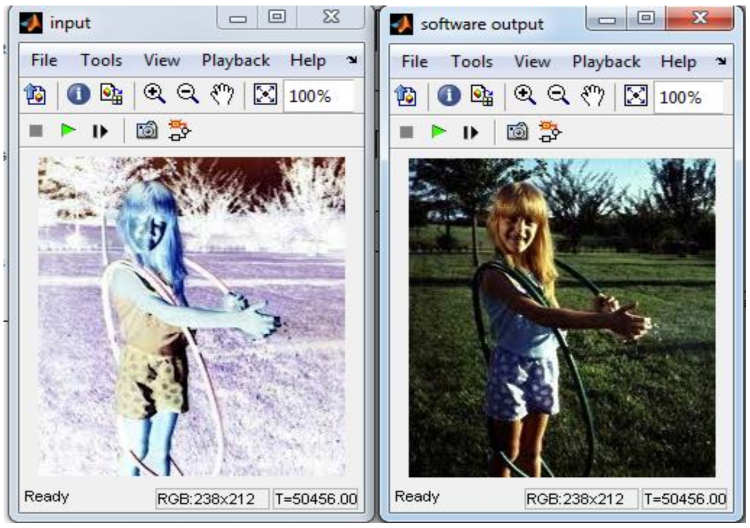

Figure 9: Results for Color Image Negative.

C. Algorithm for Image Enhancement.

Certain images can be enhanced for its perception by simply adding some illumination to it. For grayscale, single dimensional image signal is resolved and for color images multidimensional R|G|B image signals are resolved following a similar approach.

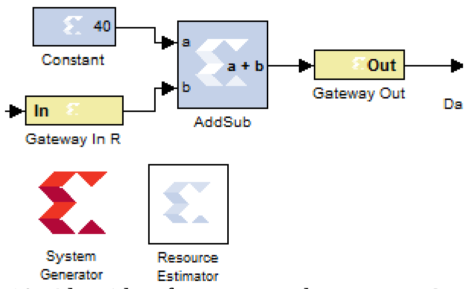

Figure 10: Algorithm for Image Enhancement Grayscale

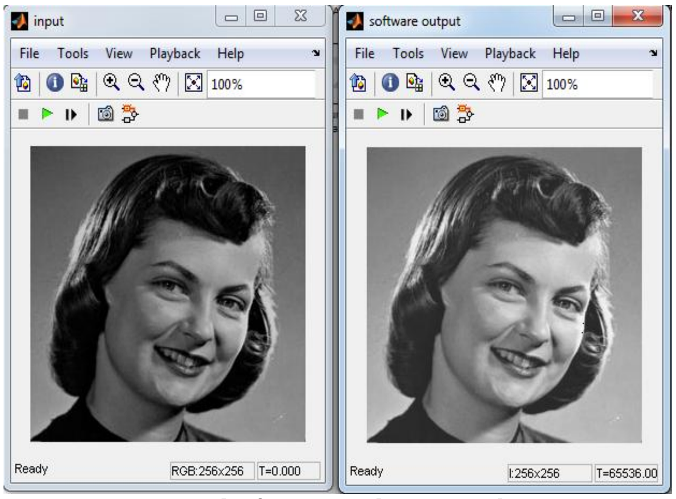

Figure 11: Results for Grayscale image enhancement.

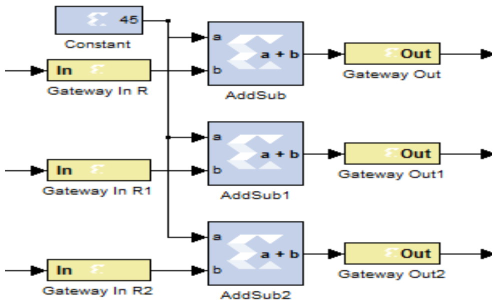

Figure 12: Algorithm for Color Image Enhancement

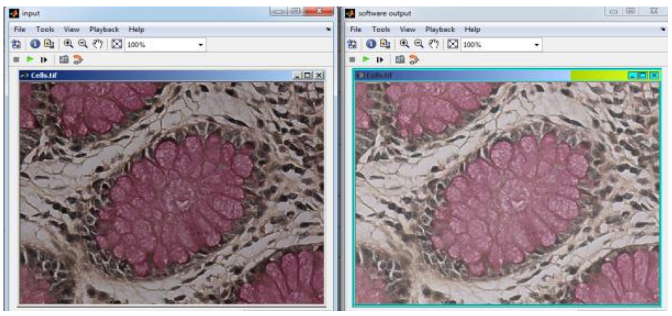

Figure 13: Results for Color Image Enhancement.

D. Algorithm for Contrast stretching

Changing the contrast of an image, change the range of luminance values present. When visualized in the histogram it is equivalent to expanding or compressing the histogram around the midpoint value. Mathematically it is expressed as:

$$new\_value = (old\_value - 0.5) \times contrast + 0.5$$
$$new\_value = (old\_value - 120) \times 3 + 127$$

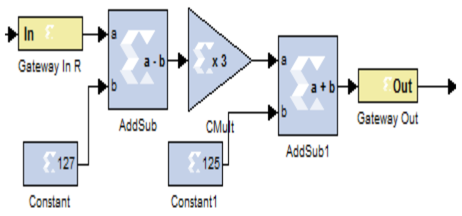

Figure 14: Algorithm for Contrast stretching grayscale

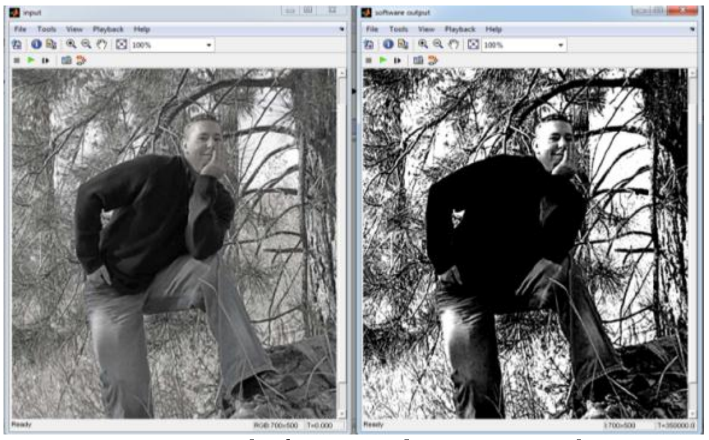

Figure 15: Results for Grayscale contrast stretching

A similar approach can be followed for developing color contrast images. It can be shown in Figure 17. The contrast stretching for color images stretches intensity values, hue values and saturation values for each pixel in an image for each R, G, and B signals.

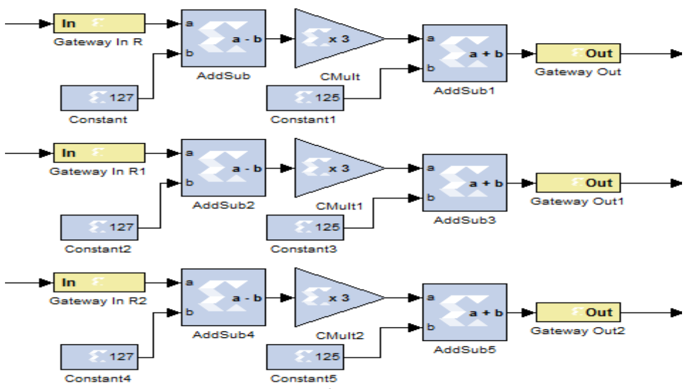

Figure 16: Algorithm for Color Contrast Stretching

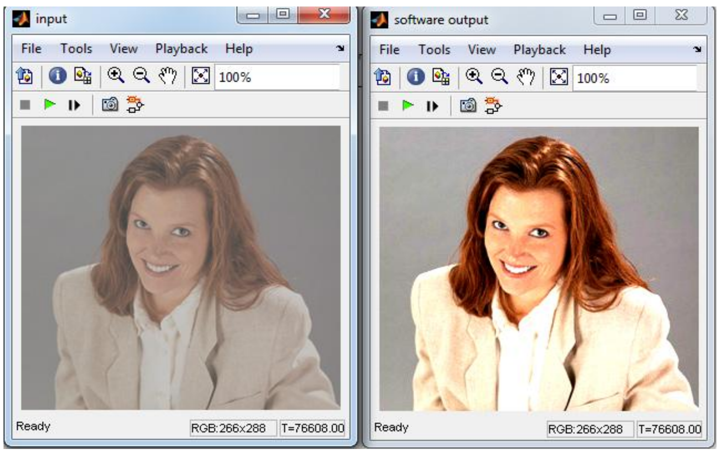

Figure 17: Results for color Contrast Stretching

E. *Algorithm for Image Thresholding*

Thresholding an image is the process of making all pixels above a certain threshold level white while others black. For implementing the algorithm a suitable constant is taken e.g. 55, a Mux is used for replacing the thresholds by white.

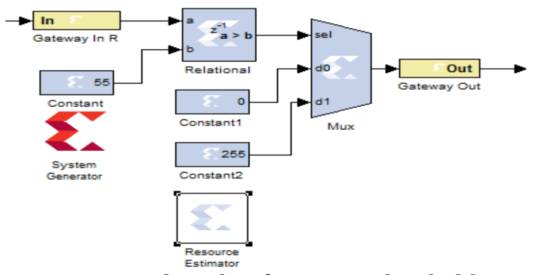

Figure 18: Algorithm for image Thresholding

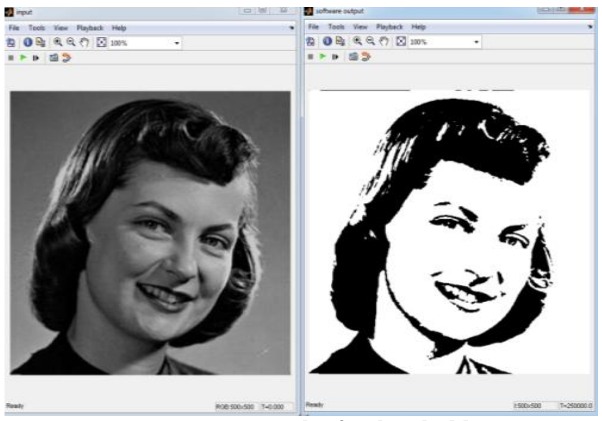

Figure 19: Results for thresholding

F. *Algorithm for Edge detection*

Edge detection is simply the filtering and masking operation with suitable filter mask. 5x5 Filter Mask provides coefficients for an Edge, Sobel X, Sobel Y, Sobel X-Y, Blur, Smooth, Sharpen, Gaussian, or Identity filtering can be accessed. For Filtering operation the delay is created by 5x5 filter block to avoid this error the system generator has to clocked 5 times faster than the normal clock.

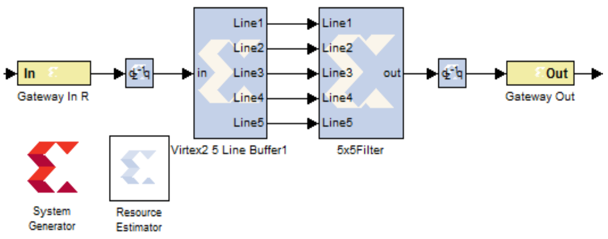

Figure 20: Algorithm for Edge detection for Grayscale Images

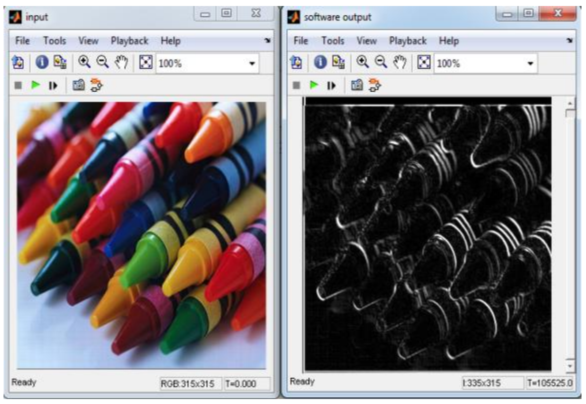

Figure 21: Results for Sobel X-Y detection Grayscale.

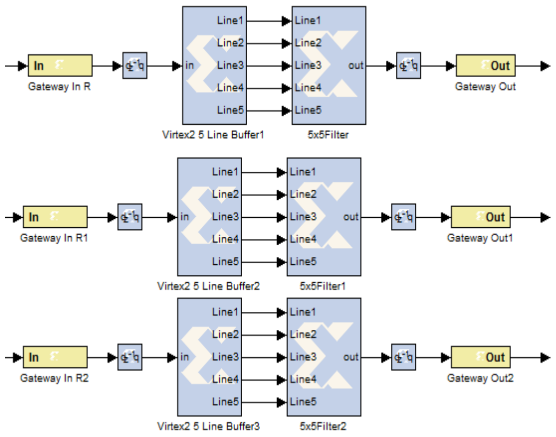

Figure 22: Algorithm for Color Edge Detection

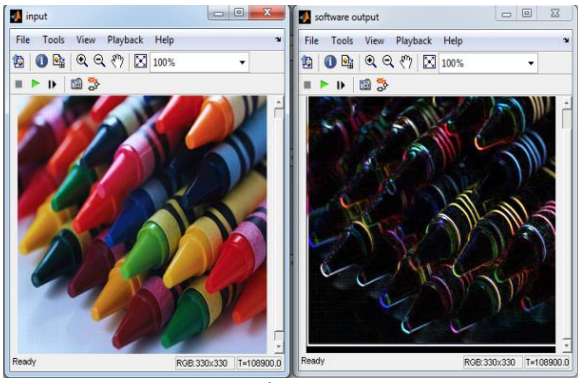

Figure 23: Results for Color Edge detection

G. *Algorithm for Boundary Extraction*

For Boundary Extraction the image is first filtered using Sobel X-Y Mask. These masks have better noise suppression or smoothing. This can be achieved by 5X5 filter block. Then the image is filtered using Gaussian mask. This mask resembles the log Fourier spectrum. Using this, the pixel with brightest value are particularly highlighted. Image can then be threshold for complete boundary extraction

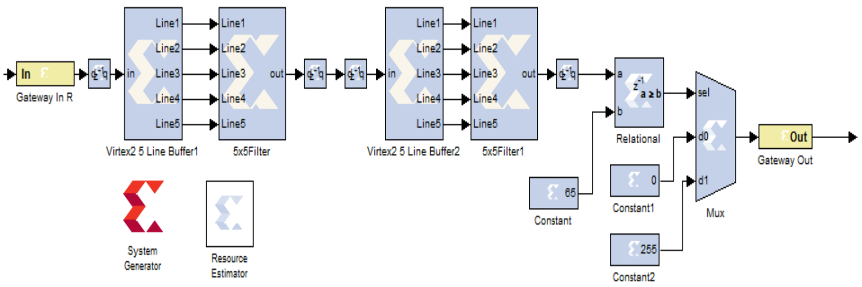

Figure 24: Algorithm for Boundary Extraction

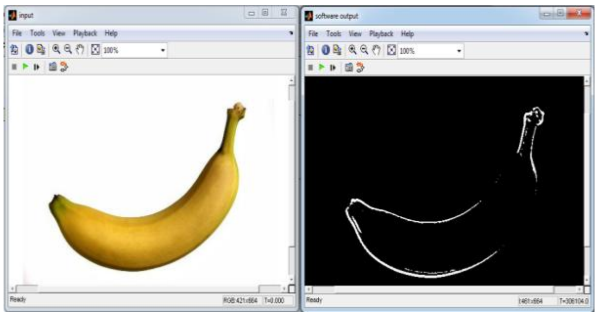

Figure 25: Results for boundary Extraction

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.[3][6]
Steps Followed in Hardware Co-simulation

System generator is configured as:

A. *Choosing a Compilation Target*

- Part: Defines the FPGA part to be used (nexys 3 spartan6 –xc6slx 16). Resulting library is created as follows

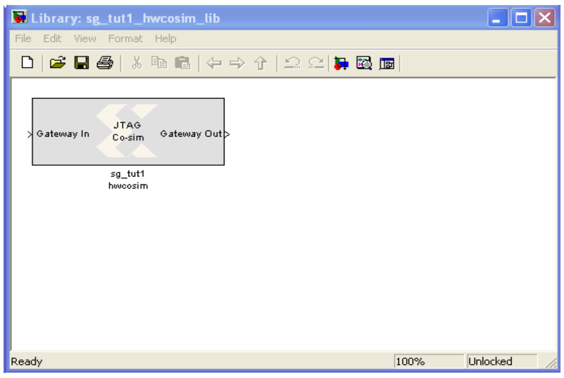

Figure 26: Hardware co-simulation block

- 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 testbench: This instructs System Generator to create a HDL testbench .
- Design is synthesized and Implemented.

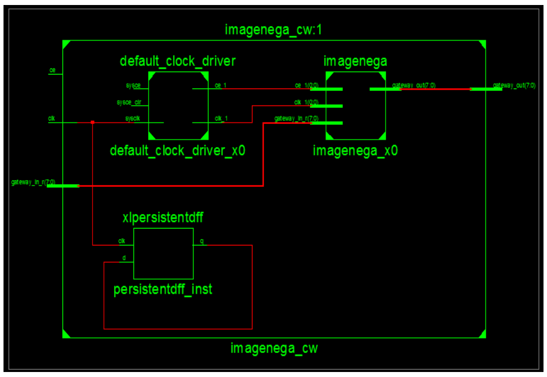

Figure 27: RTL Schematic for Image Negative

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.

TABLE SHOWING ANALYSIS FOR RESOURCE ESTIMATION AND SIMULATION TIME FOR NEXYS3 OR SPARTAN6 FPGA

	Slices	F/F's	LUT	IOBs	Mults	Image size	Simulation time (MATLAB/FPGA)	Synthesis time (FPGA)
Image negative (Addsub)	4	0	8	16	0	700*500 pixels	3.5 sec	5.7 sec
Image negative (Inverter)	0	0	0	16	0	700*500 pixels	3.5 sec	6.11 sec
Color Image negative (Addsub)	4696	0	24	48	0	238*212 pixels	0.50 sec	5.41 sec
Color Image negative (Inverter)	0	0	0	48	0	430*600 pixels	2.58 sec	5.85 sec
Image Enhancement	2	0	8	16	0	256*256 pixels	0.65 sec	5.46 sec
Color Image enhancement	6	0	24	48	0	417*497 pixels	2.07 sec	5.51 sec
Contrast stretching	12	2	36	26	0	700*500 pixels	3.5 sec	5.51 sec
Color contrast stretching	37	6	108	78	0	266*288 pixels	0.76 sec	6.19 sec
Thresholding	1	1	1	16	0	500*500 pixels	2.5 sec	6.78 sec
Edge Detection	163	469	337	16	5	335*315 pixels	1.05 sec	8.22 sec
Color edge detection	487	1407	1039	48	15	330*330 pixels	1.08 sec	9.31 sec
Boundary extraction	332	1127	896	16	0	461*664 pixels	3.06 sec	9.68 sec

VII. CONCLUSIONS

We conclude from this paper that Xilinx System Generator is a versatile tool to perform software and hardware image processing task. It provides rapid means to do hardware implementation of complex techniques used for processing images with minimum resource and minimum delay. The need of rapid prototyping tools such as MATLAB Simulink and Xilinx System Generator are increasingly important in recent times because of time-to-market constraints. It provides simplicity and ease for Hardware implementation.

The features of Simulink/Xilinx System Generator-to-FPGA flow can be discussed as follows:

• Fast time-to-market for DSP development: With the assistance of specified DSP blocks for FPGA, the Simulink/Xilinx System generator-to-FPGA flow can greatly shorten the development cycle from algorithm to hardware.

• Friendly graphics interface: Although the schematic entry is also a GUI interface, the Simulink is easier to organize input data and much convenient to observed output in many ways.

• Flexible modelling and simulation: The design can be well organized into hierarchical modules and easy to be combined with other entry method for design decision and convenient to debug and simulation.

• Supports variety in software simulation: This tool support software simulation, but most importantly it generates necessary files for implementation in all Xilinx FPGA'S, with the parallelism, robust, speed and automatic area minimization. These features are essentials in real time image processing.

REFERENCES

Bibliographical References

[1] R.C. Gonzalez, R.E. Woods, "Digital Image Processing".New Jersey: Prentice-Hall 2008 v.
[2] DSP System Generator User guide release 12 .1.
[3] Xilinx System Generator User's Guide,www.Xilinx.com , www.Xilinxforum.
[4] Matlab Website : ,http:// www.mathworks.com.
[5] White paper: Using System Generator for SystematicHDL Design, Verification, and Validation WP283(v1.0) January 17, 2008
[6] Tutorial - Using Xilinx System Generator 13.2 for Co-Simulation on Digilent NEXYS3 (Spartan-6) BoardAhmed Elhossini February 22, 2013
[7] International journal of mathematical models and methods in applied sciences ' Architecture for filtering images using Xilinx System Generator 'Alba M. Sánchez G., Ricardo Alvarez G., Sully Sánchez G.; FCC and FCE BUAP
[8] International Journal of VLSI design & Communication Systems (VLSICS) Vol.2, No.4, December 2011 DOI : 10.5121/vlsic.2011.2409 95 , An Efficient Fpga Implementation Of Mri Image Filtering And Tumour Characterization Using Xilinx System Generator By Mrs.S.Allin Christe, Mr.M.Vignesh, Dr.A.Kandaswamy
[9] IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 2, March 2012 ISSN (Online): 1694-0814 www.IJCSI.org Hardware Software co-simulation for Image Processing Applications, A.C.Suthar, Mohammed Vayada,C.B.Patel,G.R.Kulkarni.