AIVP ADVANCES IN IMAGE AND VIDEO PROCESSING
SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
VOLUME 2, ISSUE 5
ISSN: 2054 - 7412

Implementation of Image Processing Lab Using Xilinx System Generator

K. Anil Kumar and M. Vijay Kumar
Department of Electronics and Communication Engineering,
Gudlavalleru Engineering College, INDIA;
anilkumar.kanna3@gmail.com; vij4b7@gmail.com

ABSTRACT

The paper presents information on various image processing operations using Field Programmable Gate Arrays (FPGA). Processing of images on FPGA is complicated, since it needs separate architectures to process the image. To facilitate such operations, Matlab, Simulink and Xilinx system generator tools, which convert the image into suitable formats that are supported by FPGA, are used. XSG plays an instrumental role in generating VHDL/VERILOG code in tune with algorithms designed in Simulink. The generated code will be dumped into FPGA and then it performs operations on image. Use of XSG in image processing effectively reduces total design time of a system.

Keywords: Image, Simulink, FPGA.

1 Introduction

Over the past decades, the field of image processing has undergone a rapid evolution. Image processing has varied applications, computer vision, digital photography [8], Traffic load computation [7] etc. Current trends in digital camera technology have led to an increase in larger number of pixels being adjusted into smaller spaces. This result in an overall descends in the visual quality of images.

This paper particularly determinates on developing appropriate method to perform hardware implementation of various image processing algorithms that can be used in some applications. Image quality can be enhanced by creating image processing algorithms using Xilinx system generator such as, contrast stretching[8], edge detection[1],etc. these algorithms were mainly discussed in this paper. This paper aims at, (1) Implementing algorithms in MATLAB using Xilinx system generator (XSG) for specific role, (2) Generation of HDL code using Xilinx system generator token, (3) Hardware implementation of given algorithms on FPGA.

2 XSG Design Flow

For accomplishing Image processing task using FPGA, MATLAB [12], Simulink [6] and Xilinx system generator[13] tools are used. Simulink is a model based design environment integrated with Matlab. One of the block library provided by Simulink is Xilinx system generator. The System Generator token along with Xilinx has to be mapped to MATLAB. This agglomerate Xilinx Blockset to the Matlab, Simulink environment which can be directly used for building algorithms. The algorithms are developed for image negative, image enhancement, contrast stretching etc. using Xilinx Block set. These algorithms are simulated in Matlab, Simulink environment with appropriate simulation time. After obtaining the results, System Generator is configured for appropriate FPGA board. FPGA board used here is Virtex5 xc5vlx110t-3ff1136.

DOI: 10.14738/aivp.25.471
Publication Date: 25th September 2014
URL: http://dx.doi.org/10.14738/aivp.25.471

After compilation, programming file in VHDL has been created and can be accessed using Xilinx ISE. The module is checked for syntax check, synthesized and implemented on FPGA. The Xilinx System Generator can generate User constraints file (UCF) for testing architecture. Bit stream compilation is necessary to create an FPGA bit file which is inevitable for FPGA input. The Figure 1 depicts the system generator design flow.

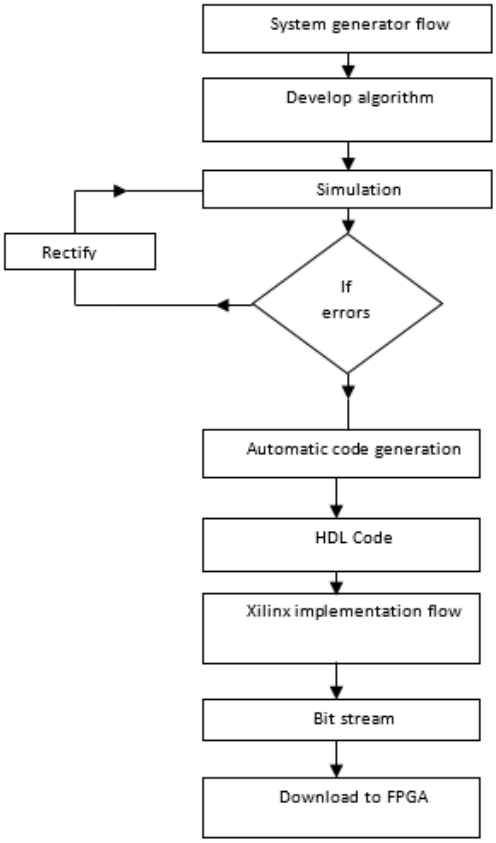


Figure 1: System Generator Design Flow

3 Design of Image Processing Lab on FPGA

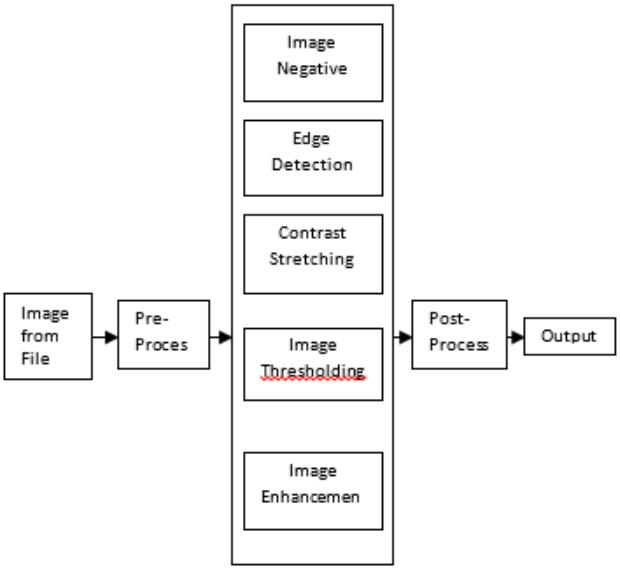


Figure 2: Design of Image Processing Lab on FPGA

Before processing the image on Xilinx system generator, it should be applied to preprocessing [6] block, which converts 2-D (frame) into 1-D (serial), as Xilinx system generator doesn't able to process the image in 2-D.Various operations such as edge detection, negative generation etc. can be performed on the arrived 1-D (serial) data stream using Xilinx system generator. To obtain the output image properly post processing [6] must be done, which converts 1-D into 2-D.

3.1 Image Pre-Processing

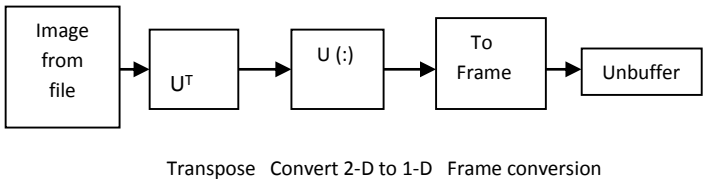


Figure 3: Pre-Processing Block Diagram

Image from file block reads the image from file. If the image is M-by-N array, the block outputs a binary or intensity image, where M and N are rows and columns. If the image is M-by-N-by-P array, the block outputs a color image, where M and N are rows and columns in each color plane, P. Then the transpose block transposes M-by-N matrix to N-by-M matrix. Convert 2-D to 1-D block converts the input data (2-D) to 1-D (serial) format. Later, Frame conversion block sets the output sampling mode to either frame based or sample based. Unbuffer block unbuffers an M-by-N input into a 1-by-N output. That is, inputs are unbuffered row wise.

3.2 Image Post-processing

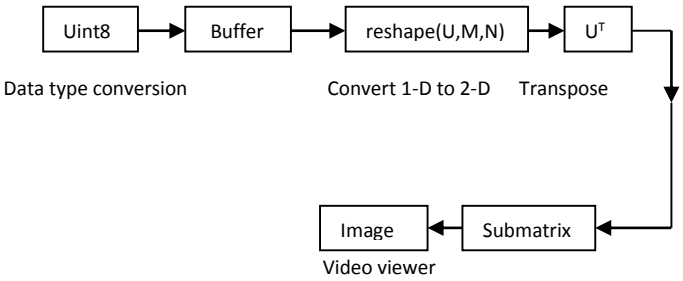


Figure 4: Post-Processing Block Diagram

Data type conversion block translates an input signal to the data type required for the Output data type parameter. The input can be any real or complex-valued signal. If the input is real, the output is real. If the input is complex, the output is complex. Buffer block performs frame-based processing. The block produces an output with a different frame size by redistributing the data in each column of the input. Buffering a signal to a larger frame size yields an output with a slower frame rate than the input. Convert 1-D to 2-D block converts the input data (1-D) to 2-D (frame) format. Sub matrix block extracts a contiguous sub matrix from the M-by-N input matrix.

4 Image Processing Algorithms

To implement image processing algorithms Xilinx system generator is used because of its ability to generate HDL code. Various algorithms like negative generation, image enhancement, contrast stretching, image thresholding, edge detection are implemented using XSG. Before going to implement algorithms, Gateway In and Gateway Out blocks should be connected in between pre-processing and post-processing blocks. These blocks act as input and output to the Xilinx portion of Simulink design.

4.1 Algorithm for Image Negative

The algorithm for gray scale image negative is given in Fig 5. Here, addsub block simply subtracts the constant 255 from pixel value to generate the negation of an input image. The output for this algorithm can be observed in Figure 6(b).

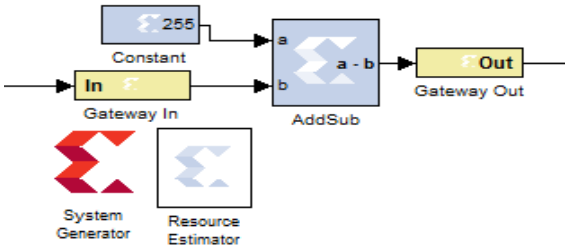


Figure 5: Algorithm for Gray Scale Image Negative

(a) Input Image (b) Output Image

Figure 6: Result for Gray Scale Image Negative

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

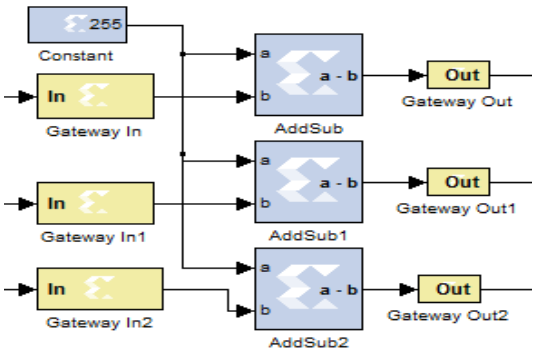


Figure 7: Algorithm for Color Image Negative

(a) Input Image (b) Output Image

Figure 8: Result for Color Image Negative

4.2 Algorithm for Image Edge Detection

Edge is a basic feature of an image. Edge detection produces an edge map that contains important information about the image. Edge detection is simply the masking operation with suitable filter mask. 5x5 Filter Mask provides coefficients for an Edge. 5x5 filters comprises of 5 n-tap MAC FIR filters, each MAC filter has separate architecture which consists of counter, addressable shift register, coefficient ROM, capture register, MAC engine. The absolute value of the FIR filters is computed and the data is narrowed to 8-bits. Sobel Y, Sobel X, Sobel X-Y, Smooth, Blur, 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 be clocked 5 times faster than the normal clock. Internal architecture for 5x5 filters and its internal diagram is given in Figure 9 and Fig. 10

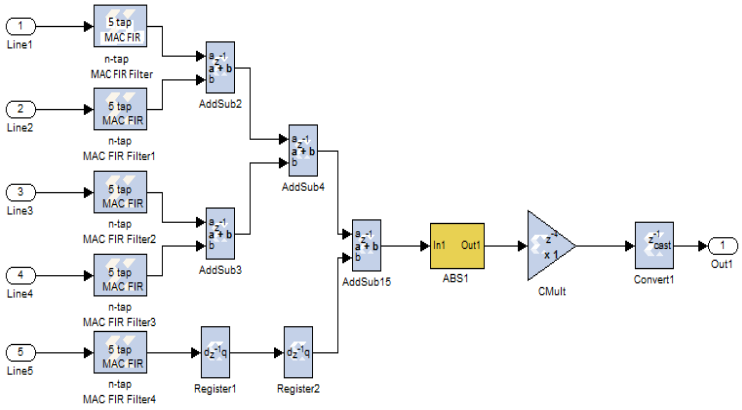


Figure 9: Internal Architecture for 5x5 Filter

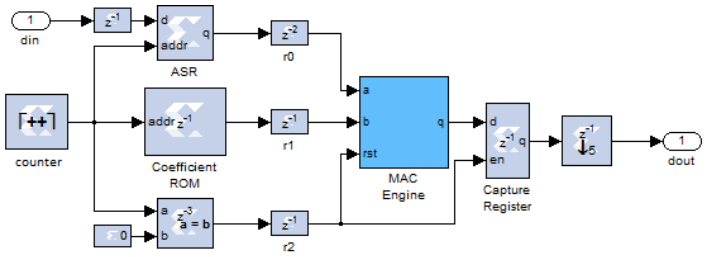


Figure 10: Internal Architecture for 5 tap MAC Filter

Algorithm and result for gray scale image edge detection is shown in Fig 11 and Fig 12(b)

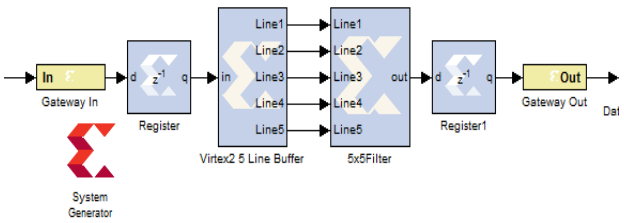
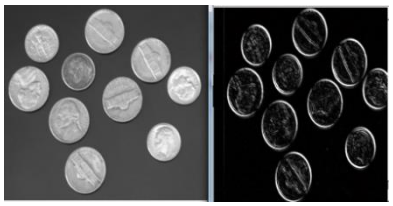


Figure 11: Algorithm for Gray Scale Image Edge Detection



(a) Input Image **(b) Output Image**

Figure 12: Result for Gray Scale Image Edge Detection

Algorithm for color image edge detection is given below. It is similar to Gray scale image edge detection but is implemented for R, G, B signals.

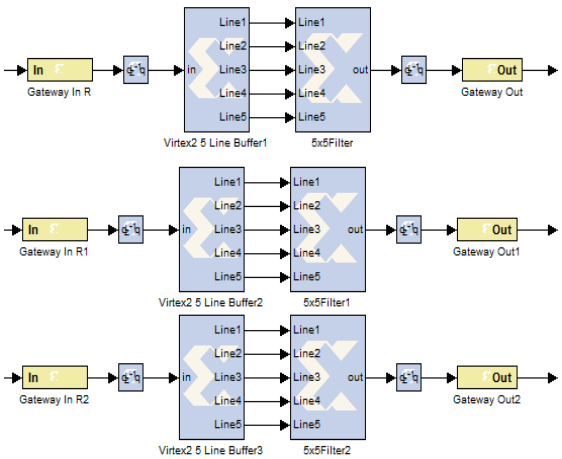


Figure 13: Algorithm for Color Image Edge Detection

(a) Input Image **(b) Output Image**

Figure 14: Result for Color Image Edge Detection

4.3 Algorithm for Image Thresholding

Thresholding an image is the method of making all pixels are white above a certain threshold value 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. Mathematically it is expressed in equation (1) and (2).

White (255) = old value > 55 (1)

Black (0) = old value < 55 (2)

A binary decision can be carried out for each pixel using the rule

$$F(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T_{xy} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Where T_{xy} is the threshold assigned to location (x, y) in the image

Algorithm and result for image thresholding can be observed in Figure 15 and 16(b)

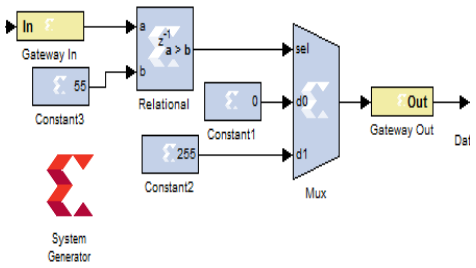


Figure 15: Algorithm for Image Thresholding

(a) Input Image (b) Output Image

Figure 16: Result for Image Thresholding

4.4 Algorithm for Contrast Stretching

Contrast stretching means changing the contrast or luminescence values of an image. It can be achieved by using constant multiplier and add sub blocks and basic algorithm is given below. Mathematically it is expressed in equation (4).

New value = {(old value-240)*3} + 238 (4)

Algorithm and result for contrast stretching can be observed in Figure 17 and 18(b)

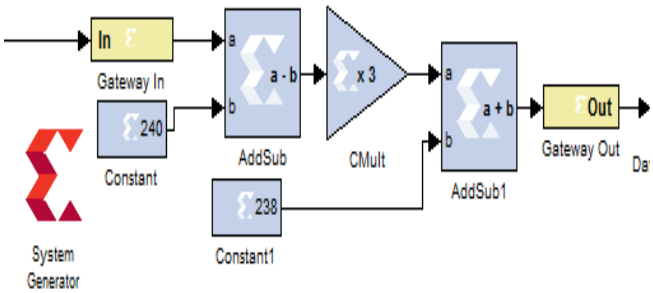


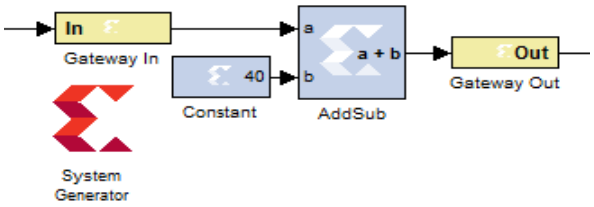
Figure 17: Algorithm for Contrast Stretching

(a) Input Image (b) Output Image

Figure 18: Result for Contrast Stretching

4.5 Algorithm for Image Enhancement

Images can be enhanced by adding some constant values to it. The basic algorithm and its result were shown in Figure 19 and Figure 20(b).



(a) Input Image (b) Output Image

Figure 19: Algorithm for Image Enhancement

Figure 20: Result for Image Enhancement

4.6 Implementation of Image Processing Lab

In this section, grouping of various image processing algorithms is done simply by using multiplexer. It gives the output for any of these algorithms connected to (d0-d4) lines of multiplexer by giving some constant value as shown in Figure 21. For example 2 is given as constant, it gives the image thresholding output shown in Figure 22(b).

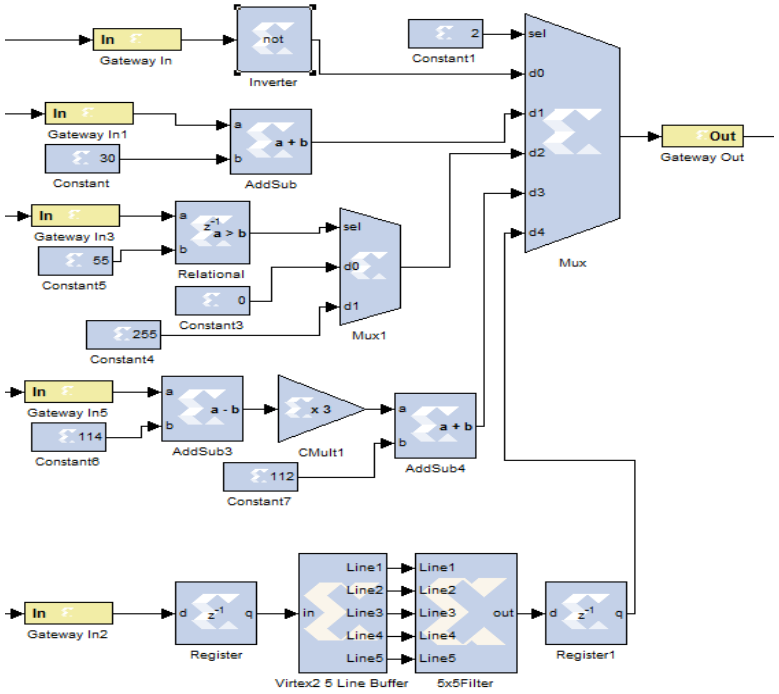


Figure 21: Implementation of Image Processing Lab

(a) Input Image (b) Output Image

Figure 22: Result for Image Processing Lab for Constant 2

5 Implementation using Xilinx

Before processing the images on FPGA, HDL code should be generated by using system generator token. The system generator token is used to generate the code. Every Simulink model must contain system generator token to generate the code for algorithms designed in Simulink. Once a system generator token is added to a model, it is easy to generate code for the designed model. To generate the code, one should follow the procedure. First step in the procedure is opening of system generator token and selection of, compilation type which specifies the type of compilation that should be produced when the code generator is invoked. HDL net list compilation type is used in every model. Part, which the specific parts to be used on FPGA. Virtex5 xc5vlx100t-3ff1136 FPGA is used in every model. Synthesis tool specifies the tool used to synthesize the design. XST tool was used as a synthesis tool for the models described in the section. Hardware Description Language, Specifies the HDL language used for compilation of the design. In every model, VHDL is set for hardware description language. Create test bench, this intimate's system generator to create a HDL test bench.

Second step in the procedure is clocking tab. The parameters for the clocking tab are as follows.

- *FPGA Clock period; defines the period in nanoseconds of the system clock.*
- *Clock pin location; defines the pin location for the hardware clock.*

After setting the above parameters, press generate button in Xilinx system generator token. Then it automatically generates the code in specified path in Target directory. Open ISE navigator window and open the code generated files and simulate the code. Automatically it generates the RTL and Technology schematic diagrams, synthesis reports and timing diagrams. RTL schematic and Timing diagrams for image processing lab are shown in Figure 23 and Figure 24.

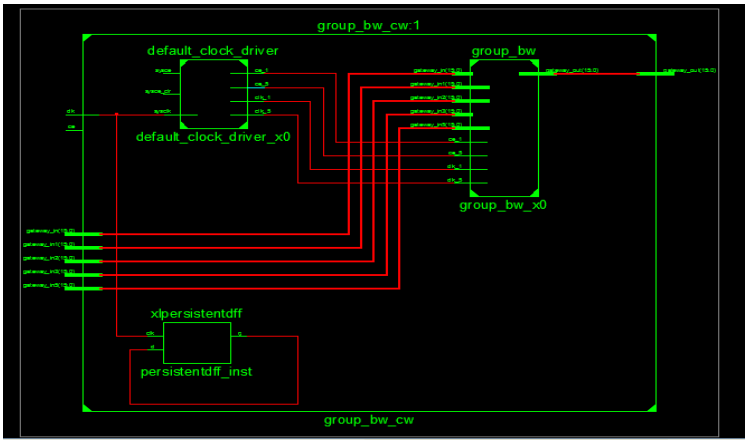


Figure 23: RTL Schematic for Image Processing Lab

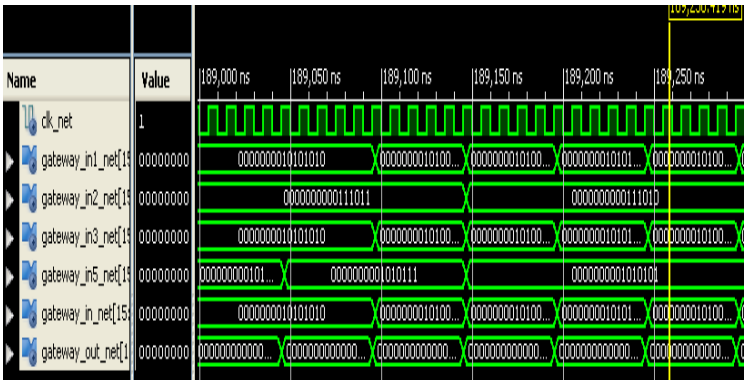


Figure 24: Timing Diagram for Image Processing Lab

6 Conclusion

Xilinx system generator is very helpful tool for software and hardware image processing tasks. It provides means to do hardware implementation of image processing algorithms with minimum resource and minimum delay. Thus, Matlab, Simulink and Xilinx system generator tools are extremely important in today's world as technology grows rapidly, that is, large number of pixels is being crammed into ever-smaller spaces. It provides easy hardware implementation.

7 Future Scope

The same concept can be extended to various fields like video processing, computer graphics, animations etc. The paper describes simulation part and suitable results are observed. The same results can also be verified on FPGA by dumping the code generated by system generator token.

REFERENCES

[1]. D.Selvathi, J.Dharani,"*Realization of Beamlet Transform Edge Detection Algorithm Using FPGA*", International Conference on Signal Processing, Image Processing and Pattern Recognition (ICSIPR), PP.1-5, 2013.

[2]. Jinbo Wu, Zhouping Yin, and Youlun Xiong, "*The Fast Multilevel Fuzzy Edge Detection of Blurry Images*" IEEE Signal Processing Letters, vol.14, No.5, may 2007

[3]. C.Sujatha, D.selvathi,"*Hardware Implementation of image edge detection using XSG*", Asian Journal on Scientific Research, 2014.

[4]. Ravi.s, Abdul Rahim,"*FPGA based design and implementation of image edge detection using XSG*", international journal of engineering trends and technology (IJETT), vol.4, issue.10, PP.4657-4660, oct 2013.

[5]. Obili ramesh, P.V.Krishna Mohan Gupta, "*A Real Time Hardware And Software Co-Simulation of Edge Detecetion For Image Processing System*", International Journal on Engineering and Research Trends(IJERT), vol.2, issue.8, PP.1695-1701, aug-2013.

[6]. Simulink user's guide, 2010.

[7]. "*Hardware software co-simulation for traffic load computation using matlab and simulink model blockset*", International Journal of Computational Science and Information Technology (IJCSITY), Vol.1, IssueNo.2, May 2013.

[8]. *"FPGA Based Design and Implementation of Image Architecture using XILINX System Generator* ", IJCAE, Vol. No. 3 Issue 1, July 2012, pp132- 138.

[9]. "*Accelerated Image Processing on FPGAs*", IEEE Transactions on Image Processing , Vol. 12, issue 12, Dec. 2003,pp1543-1551.

[10]. "*Architecture for filtering images using Xilinx system generator*", INTERNATIONAL JOURNAL of MATHEMATICS AND COMPUTERS IN SIMULATION, Issue 2, Volume 1, 2007,pp 101-106.

[11]. "*Performance Efficient FPGA Implementation of 2D image filtering using XSG*", IJECT Vol. 3, Issue 4, Oct - Dec 2012,pp 374-377.

[12]. Matlab hdl coder, http:// www.mathworks.in/products/hdl-coder.

[13]. Xilinx System Generator User's Guide, http://www.Xilinx.com/support/sw_ manuals/sysgen_user.pdf.

[14]. Digital Image Processing text book by Rafael C.Gonzalez, Richard

