Image Processing using Xilinx System Generator (XSG) in FPGA

Ankita gupta[1], Himanshu Vaishnav[2], Himanshu Garg[3]

[1]Asst. professor, ECE, Poornima Group Of Institutions, jaipur, india
[2]Student, ECE, Poornima Group Of Institutions, Jaipur, India
[3]Student, ECE, Poornima Group Of Institutions, Jaipur, India

Abstract: **This paper presents the conceptual description of hardware & software simulation for image processing using Xilinx System Generator (XSG). This paper provides the theory and practical aspects of technique, which provide a set of Simulink model for several hardware operations using various Xilinx that could be implemented on various FPGA. This paper presents an efficient architecture for various image processing algorithms for image negatives, image enhancement, contrast stretching, Image Edge Detection, image Brightness Control, Range Highlighting Transformation, Parabola transformation 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.**

Key Word— Image Processing, Xilinx System Generator, Field Programmable Gate Array (FPGA), Simulink and transformation.

I. INTRODUCTION

In this paper, we study digital images and its processing techniques, specifically point processing algorithms. The handling of digital images is a subject of widespread interest. Image processing is used to modify pictures to improve them (enhancement, restoration), extract information (analysis, recognition) and change their structure (composition, image editing). FPGAs are increasingly used in modern imaging applications namely image filtering , medical imaging , image compression and wireless communication. The need to process the image in real time, leads to implement them in hardware, which offers parallelism, thus significantly reduces the processing time. The

drawback of most of the methods is that they use a high level language for coding, which requires thousands of coding lines for image processing applications which is inefficient as it takes much time. In order to solve this problem, a tool called Xilinx System Generator(XSG), with graphical interface under the MATLAB-Simulink is used which makes it very easy to handle with respect to other software for hardware description. FPGA is a form of highly configurable hardware while DSPs are specialized form of microprocessors. System Generator is the modeling tool in which designs are captured in the DSP friendly Simulink modeling environment using Xilinx specific Block set. Point processes are the simplest and basic image processing operations. point operations are the simplest, they contain some of the most powerful and widely used of all image processing operations. They are especially useful in image pre-processing, where an image is required to be modified before the man job is attempted. Important point Processing operations are arithmetic operations, XOR operations, histograms with equalization, and Contrast stretching and intensity transformations along with the implementations which are done using XSG.

The rest of the paper is organized as follows. Section -2 Discusses about Xilinx System Generator[10] and section-3 discusses about the Design Flow of Image Processing using the XSG[3]. Section-4 & 5 is all about the Image Processing block And Image Processing technique using XSG, Image Pre processing & post Processing technique[6] and other image Processing Techniques are Image Enhancement, Image negative[6], Image Edge Detection[12], image Brightness Control, Image Contrast Stretching[10], Range Highlighting Transformation, Parabola transformation[4]. Section-6 Discusses about Hardware Implementation. Section-7 Discusses about hardware Co-simulation which includes (a) A Compilation Target and (b) Clocking Tab,(c) Calling the Code generator[10] and last Section-8 Discusses the Conclusion.

II. XILINX SYSTEM GENERATOR

System Generator is part of the ISE® Design Suite and provides Xilinx DSP Block set 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 modeling environment using a Xilinx specific Block set. 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 Block set provides close integration with MATLAB Simulink that helps in co-simulating the FPGA module with pixel vector provided by MATLAB Simulink Blocks. 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 bit stream file as well as generating Verilog code for additional optimizations or integration with existing projects within the Xilinx ISE environment[10].

III. DESIGN FLOW FOR IMAGE PROCESSING WITH XILINX SYSTEM GENERATOR

System Generator works within the Simulink model-based design methodology. Often an executable spec is created using the standard Simulink block sets. This spec can be designed using floating-point numerical precision and without hardware detail. Once the functionality and basic dataflow issues have been defined, System Generator can be used to specify the hardware implementation details for the Xilinx devices. System Generator uses the Xilinx DSP blockset for Simulink and will automatically invoke Xilinx Core Generator™ to generate highly-optimized netlists for the DSP building blocks. System Generator can execute all the downstream implementation tools to product a bit stream for programming the FPGA. An optional test bench can be created using test vectors extracted from the Simulink environment for use with ModelSim or the Xilinx ISE® Simulator. The System Generator of DSP is shown in Fig.1

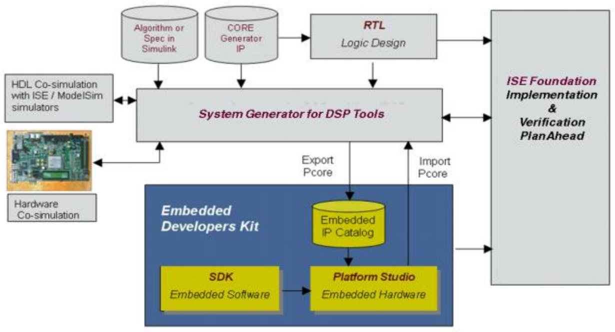

Algorithm or Spec in Simulink
CORE Generator IP
RTL Logic Design
HDL Co-simulation with ISE / ModelSim simulators
System Generator for DSP Tools
ISE Foundation Implementation & Verification PlanAhead
Export Pcore
Import Pcore
Hardware Co-simulation
Embedded Developers Kit
Embedded IP Catalog
SDK Embedded Software
Platform Studio Embedded Hardware

Fig.1: System Generator for DSP

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[3].

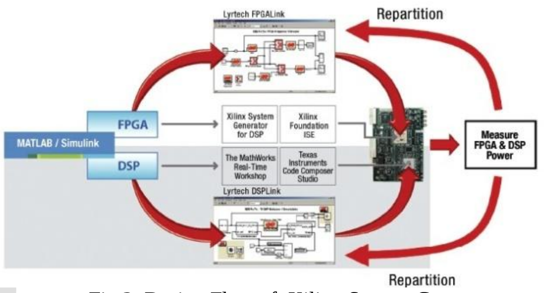


Fig.2: Design Flow of Xilinx System Generator

IV. SCHEMATIC OF IMAGE PROCESSING TECHNIQUE

The entire operation for any image processing technique using Simulink and Xilinx blocks mainly goes through three phases .

A. Image pre processing blocks:

As image is two dimensional (2D) arrangement, to meet the hardware requirement the image should be preprocessed and given as one dimensional (1D) vector. The model based design used for image pre processing is shown in Fig.3. To process 2D image it is converted into 1D by using convert 2D to 1D block. Frame conversion block sets output signal to frame based data and provided to unbuffer block which converts this frame to scalar samples at a higher sampling rate.

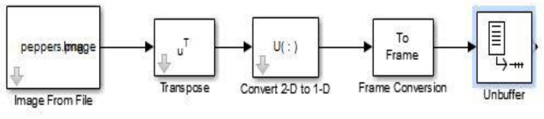


Fig.3: Image pre processing blocks

B. Image processing technique using XSG:

All Xilinx blocks should be connected between Gateway In and Gateway Out. Between those two blocks any technique can be designed. All Xilinx blocks work on fixed point but the real world signal (image, voice signal, etc.) are floating point so here the gateway in and gateway out blocks acts as translators for converting the real world signal into the desired form.

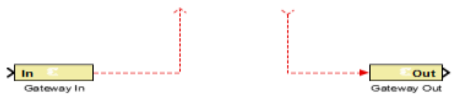


Fig.4: Xilinx Blocks

C. Image post processing blocks:

The image post processing blocks which are used to convert the image output back to floating point type are shown in Fig.5. For post processing it uses a buffer block which converts scalar samples to frame output at lower sampling rate, followed by a 1D to 2D format signal block[5].

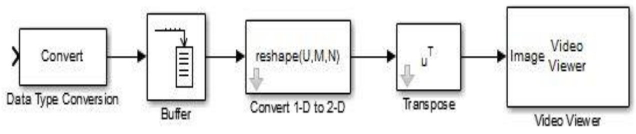


Fig.5: Image post processing blocks

V. IMAGE PROCESSING TECHNIQUES USING XILINX BLOCKS

In this section the basic image processing techniques namely image enhancement, color to gray scale conversion, image negative and image edge detection are implemented using Xilinx blocks and then they are implemented on VIRTEX5 FPGA

A. Image Enhancement

Image enhancement is basically improving the interpretability or perception of information in images for human viewers and providing better input for other automated image processing techniques. The image can be enhanced by adding a constant value (90) to the corresponding R, G and B components . If the input image is gray scale image only one component will be there instead of three components i.e.; R, G and B. 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. The Image enhancement algorithm shown in Fig.6 and Fig.7,respectively for gray and color image.

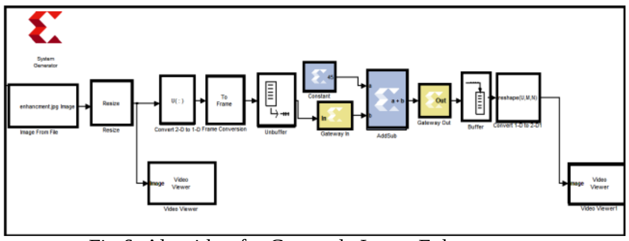


Fig.6: Algorithm for Grayscale Image Enhancement

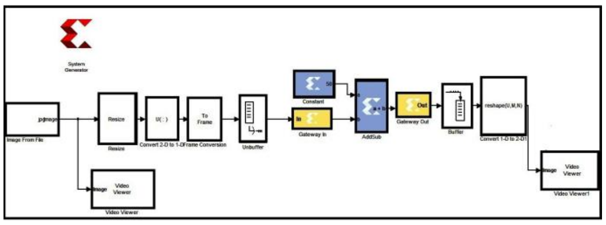
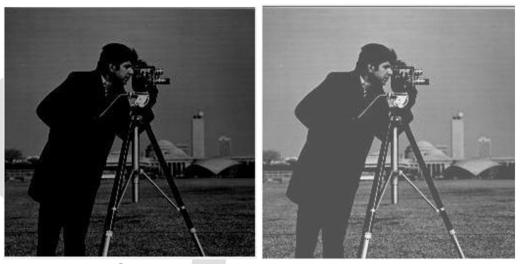


Fig.7: Algorithm for Color Image Enhancement



Original Image Output Image

B. Image Negative

A negative image is a total inversion, in which light areas appear dark and vice versa. A negative color image is additionally color-reversed, with red areas appearing cyan, greens appearing magenta and blues appearing yellow. Reversing the intensity levels of an image produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size. Image negative can be implemented by simply subtracting the R, G, B components from the constant value 255 as shown in Fig.8. Similarly for a gray scale image only one component will be there instead of R, G and B.

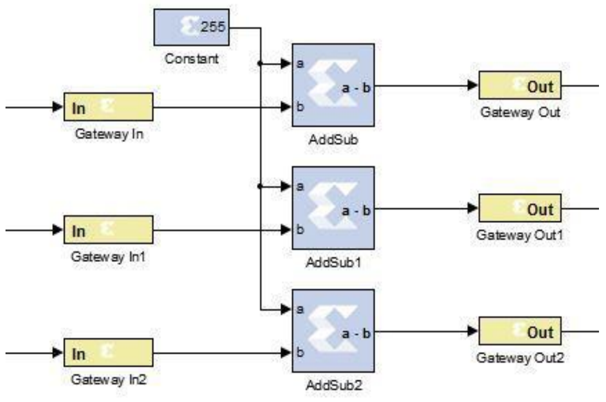


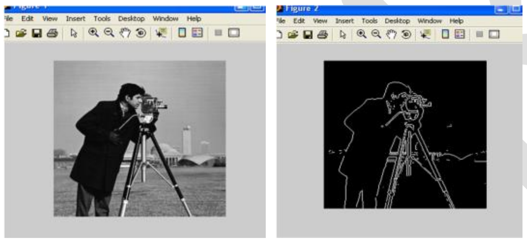
Fig.8: Image negative

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

Original Image Output Image

C. Image Edge Detection

Edge detection is one of the most commonly used operations in image analysis and there are probably more algorithms in literature for enhancing and detecting edges. An edge is point of sharp change in an image, a region where pixel locations have abrupt luminance change i.e. a discontinuity in gray level values. There are different edge detector operator masks to detect edges. They are Ordinary operator, Roberts's operator, 4-Neighbour operator, Prewitt operator and Sobel operator. To perform the edge detection a convolution operation of the input image with any of the above mentioned filter masks to be performed. The design flow of edge detection using Xilinx System Generator is shown in Fig. 9(SED) and Fig.10(CED).



Original Image Output Image

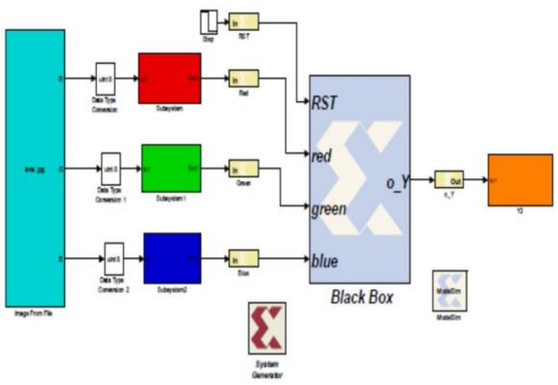


Fig.9: Sobel edge detection

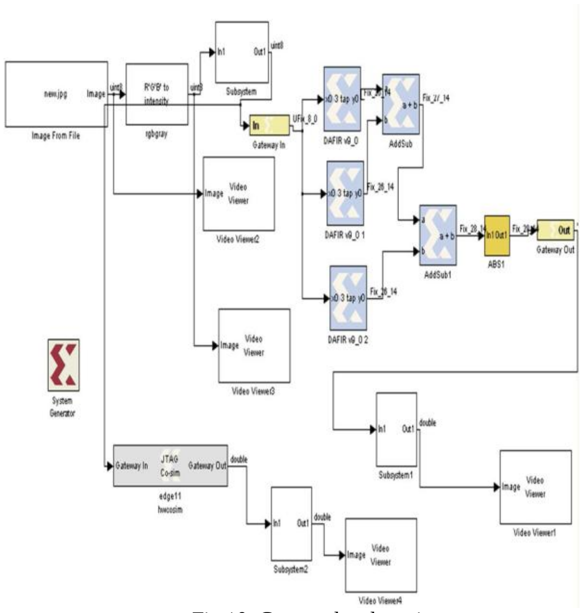


Fig.10: Canny edge detection

D. Image Brightness Control

The arithmetic operations include adding, subtracting, dividing, and multiplying pixels by a constant value. Addition and subtraction can adjust the brightness of the image . Shows the XSG blocks involved while adding and subtracting 40 from the image. The algorithm for the image brightness control is shown in Fig.11

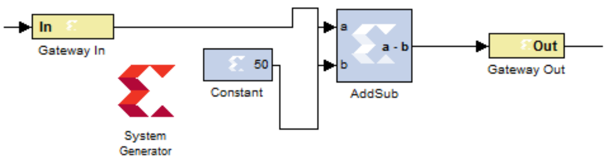
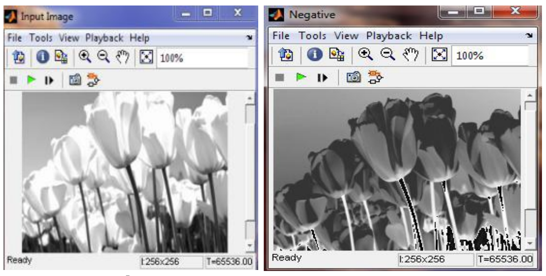


Fig.11: Algorithm for Brightness Control



Original Image Output Image

E. 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 = (12/4) (old pixel-5) + 2 .... (1)

where new pixel is its result after the transformation. Fig.12 shows the XSG blocks for the above contrast stretching to the finger print image and the results respectively we demonstrate another piecewise linear function which is as follows:

j = ((255-193) / (255-160)) (i-160) + 192 ..... (2)

Where i is the original gray level and j is its result after the transformation. Fig. Shows the XSG blocks for the above contrast stretching to the finger print image and the results respectively.

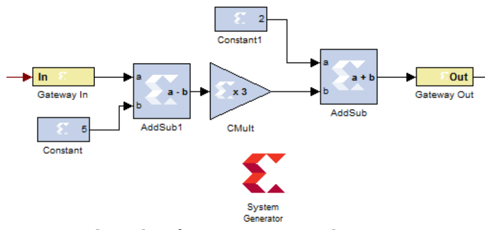
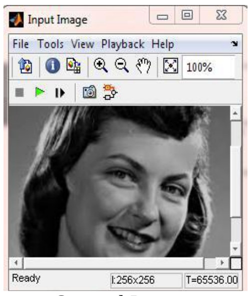
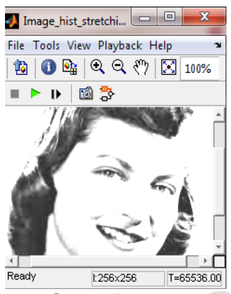


Fig 12: Algorithm for Contrast Stretching



Original Image



Output Image

F. Range highlighting Transformation

An intensity transform can also highlight a range of pixels while keeping others constant. Fig. shows the Xilinx blocks implementation and the resulting image.

```
function z = new pixel one(x, y, c)
        if (x > y) & (x < c)
            z = x;
      else
                z = 1;
```

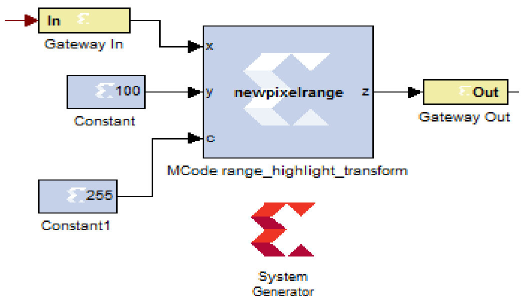
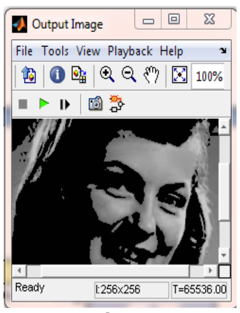
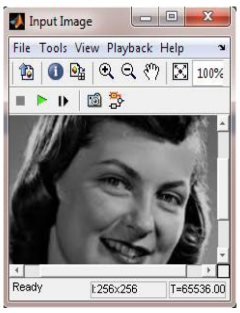


Fig.13:Algorithm for Range Highlighting Transformation



Original Image



Output Image

G. Parabola Transformation

The two formulas for the parabola transformation are as follows:

new pixel = 255 – 255 ((old pixel/128) -1)2.... (4) and

new pixel = 255 ((old pixel/128) -1)2 .....(5)

Xilinx blocks are connected for the above equations and displayed in Fig. Both the results are observed and produced .Similarly, polarize transformation, iso-intensity contouring transformation and bit-clipping transformation can also be implemented. The Algorithm for parabola transformation s shown in Fig.14

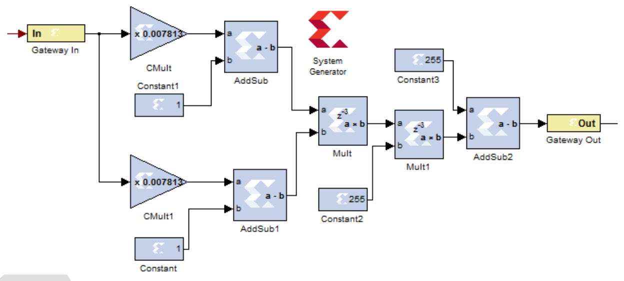
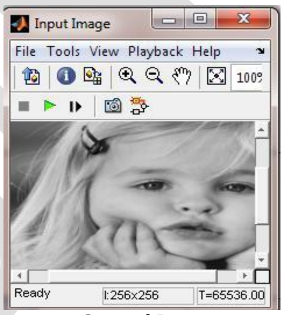
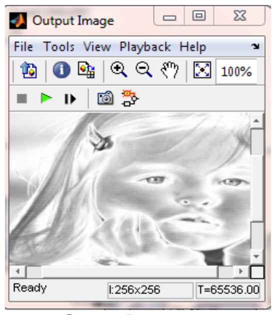


Fig.14: Algorithm for Parabola Transformation



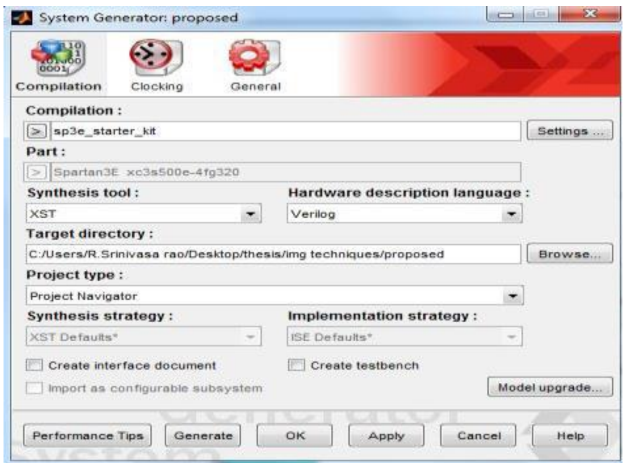
Original Image



Output Image

VI. HARDWARE IMPLEMENTATION

[5]The architectures explained above deal only with software simulation level. For implementing this design in a FPGA board the entire module should be converted to FPGA synthesizable one. For that purpose main module for any image processing is converted for JTAG hardware co-simulation, this is done with the help of System generator token. By clicking the system generator token a new window will open as shown in Fig.15. This block is configured according to the target platform and a bit stream (*.bit) file is generated. After the bit stream file is generated, hardware co-simulation target is selected and in this project, Spartan 3E starter kit (XC3S500E-FG320) is used for board level implementation. After clicking the generate button in the System generator block a hardware co-simulation block will be generated. To perform the hardware software co-simulation, the hardware co-simulation block added in the design and thereby we can see FPGA and XSG/software output at a time. The entire architecture with the hardware and software co-simulation design for the edge detection is shown in Fig.16.



System Generator: proposed

Compilation : sp3e_starter_kit

Part : Spartan3E xc3s500e-4fg320

Synthesis tool : XST

Hardware description language : Verilog

Target directory : C:/Users/R.Srinivasa rao/Desktop/thesis/img techniques/proposed

Project type : Project Navigator

Synthesis strategy : XST Defaults*

Implementation strategy : ISE Defaults*

Create interface document

Create testbench

Import as configurable subsystem

Fig15: System generator token

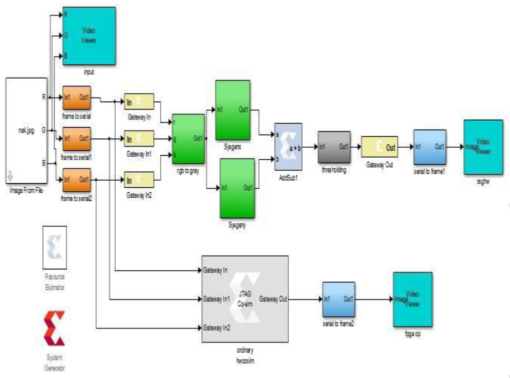


Fig.16: Hardware software Co-simulation for the edge detection

VII. 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. Compilation Target

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

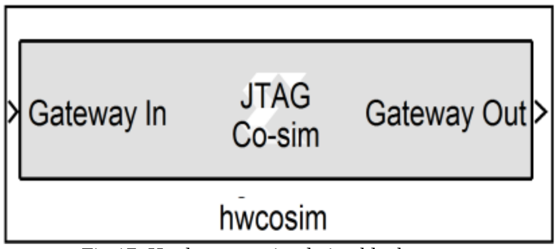


Fig.17: Hardware co-simulation block

- 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. Calling the Code Generator

- The code generator is invoked by pressing the Generate button in the System Generator token dialog box[10].

VIII. CONCLUSION

We conclude from this paper that this paper is a Xilinx System Generator is a good platform to perform the Image Processing in Software and Hardware manner. It is a Versatile tool to perform Image Processing and it provide rapid means to do hardware implementation of complex technique used for processing images with minimum resource and minimum delay. It provides simplicity and ease for Hardware implementation. 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. This paper implemented for high speed image enhancement applications using FPGA. The image enhancement techniques such as brightness and contrast adjustment are important factors in medical images. This paper explains implementation of Image Enhancement, Image negative, Image Edge Detection, image Brightness Control, Image Contrast Stretching, Range Highlighting Transformation, Parabola transformation.

REFERENCES

[1]. Xilinx System Generator User's Guide.

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

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

[4]. Kalyani A. Dakre and Prof. P. N. Pusdekar" "Image Enhancement using Hardware co- simulation for Biomedical", *International Journal on Recent and Innovation Trends in Computing and Communication Volume: 3 Issue: 2.*

[5]. R.Srinivasa Rao and R. Nakkeeran , "High level abstraction method for implementing Image Processing Techniques on FPGA", International Conference on Knowledge Collaboration in Engineering March 27- 28, 2015

[6]. Alareqi Mohammed, Elgouri Rachid and Hlou Laamari, " High Level FPGA Modeling for Image Processing Algorithms Using Xilinx System Generator ". International

Journal of **C**omputer **S**cience and **T**elecommunications [Volume 5, Issue 6, June 2014] .

[7]. 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

[8]. 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.

[9]. 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.

[10]. MS.DIPIKAS.WARKAR, DR.U.A.KSHIRSAGAR "FPGA Implementation of Point Processing Operation using Hardware Simulation", *International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 4, April 2015.*

[11]. R. Gonzalez, R. Woods, "Digital Image Processing" Third edition: Prentice-Hall 2008.

[12]. Ravi.s, Abdul Rahim.B, Fahimuddin shaik," FPGA Based Design and Implementation of Image Edge Detection Using Xilinx System Generator", *International Journal of Engineering Trends and Technology (IJETT – Volume 4 Issue 10 - Oct 2013.*