ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

FPGA Implementation of Point Processing Operation using Hardware Simulation

MS. DIPIKA S.WARKARI[1], DR.U.A.KSHIRSAGAR[2]

Department of EXTC, HVPM's, COET, Amravati, India[1]

HOD, Department of EXTC, HVPM's, COET, Amravati, India[2]

Abstract: 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. Specifically authors demonstrate point processes which use only the information in individual pixels to produce new images. Arithmetic operations, XOR operations, histograms, contrast stretching and intensity transformations are implemented using Xilinx System Generator (XSG). XSG is a useful tool to understand fundamental Digital Signal Processing (DSP) algorithms for Field Programmable Gate Array (FPGA) implementation. FPGAs provide a better platform for real-time algorithms on application-specific hardware with substantially greater performance than programmable DSPs. The study reveals that the hardware implementation results which are presented here will be extended to perform hardware-in-the-loop simulation.

Keywords: Image Processing, Point Processing, Xilinx System Generator, FPGA.

I. INTRODUCTION

In recent years, with the rapid development of economic, the process of town urbanization developed continuously, and land-use has becoming a community-wide hot issue. The construction land becomes increasingly tense and getting greater pressure. Therefore, extracting the current building Information from high resolution image and getting the current land use has provided a guarantee for rational planning and utilization of land resources in villages and towns. With the rapid development of remote sensing technology, we can choose the present application of the high resolution remote sensing image. The emergence of digital aerial camera solves the problem of the low productivity of traditional image and the poor real-time photographing [1].In this paper, we study digital images and its processing techniques, specifically point processing algorithms. FPGA [2] is a form of highly configurable hardware while DSPs are specialized form of microprocessors. System Generator [3] 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.

They operate on a pixel bases solely on that pixel's value. Although 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.

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 [4].

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 R2009a.or higher & Xilinx ISE 13.1. 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 Block set. 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

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

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 net list 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. Bit stream compilation is done which is necessary to create an FPGA bit file which is suitable for FPGA input. The Fig. shows the Design flow for Xilinx System Generator.

IV. PROPOSED WORK

The entire operation will propose using Simulink and Xilinx blocks goes through three phases & these are

1. Image pre-processing blocks.
2. Edge enhancement algorithm using XSG.
3. Image post-processing blocks

The design flow of hardware implementation of point processing operation using XSG is given in fig. 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 units are common for all the image processing applications which are designed using Simulink block sets.

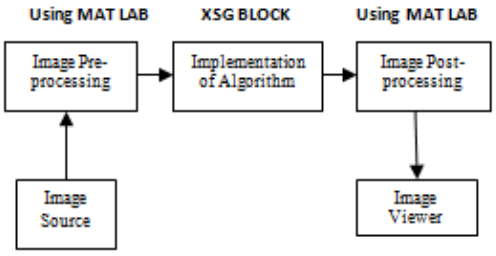


Fig: Design flow of hardware implementation of point processing operation

A. Image pre-processing blocks:

Image pre-processing in Mat lab helps to provide input to FPGA as specific test vector array which is suitable for FPGA Bit stream compilation using system generator.

1. Resize: Resize block set input dimensions for an image.
2. Convert 2-D to 1-D: image is converted into single array of pixels.
3. Frame conversion and buffer: It is used to setting sampling mode and buffering of data.

B. Image post processing blocks**:**

Image post processing helps to recreate image from 1D to 2D

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

This paper contain the fundamentals of all point processes in image processing and shows the simulation results using Xilinx System Generator for the models designed.

POINT PROCESSES

Point processes are the simplest and basic image processing operations. They operate on a pixel bases solely on that pixel's value. Although 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. This section contains an important point processing operations such as arithmetic operations, XOR operations, histograms with equalization, and contrast stretching and intensity transformations along with the implementations which are done using XSG.

1. Algorithm for 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 [5] Fig 2. Shows the XSG blocks involved while adding and subtracting 40 from the image. Fig 2. Shows the results of an image which has a size of 256 x 256.

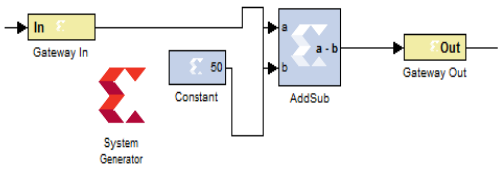


Fig 1: Algorithm for Brightness Control

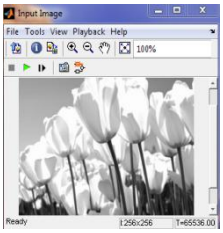


Fig 1: Original Image

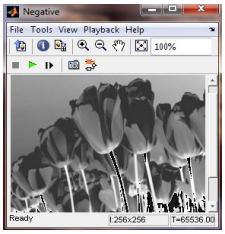


Fig 1: Output Image

DOI 10.17148/IJARCCE.2015.4420

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

2. Histogram Stretching (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 2.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 2 Shows the XSG blocks for the above contrast stretching to the finger print image and the results respectively.

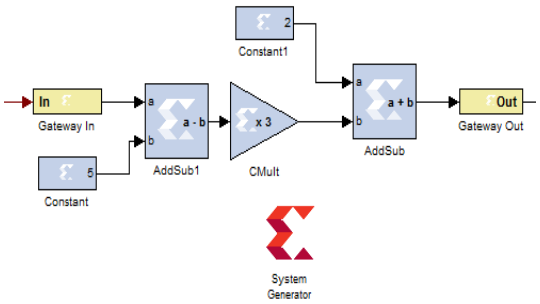


Fig 2: Algorithm for Contrast Stretching

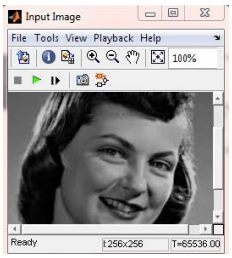


Fig 2: Original Image

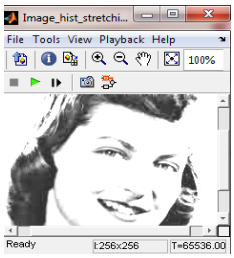


Fig 2: Output Image

3. Negative Transformation

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:

new pixel = 255 – old pixel (3)

Fig 3 shows the XSG blocks for the above negative transformation to the finger print image and the results respectively.

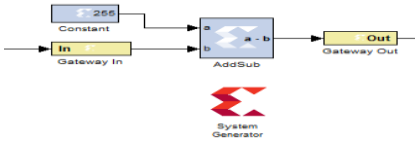


Fig 3: Algorithm for Negative Transformation

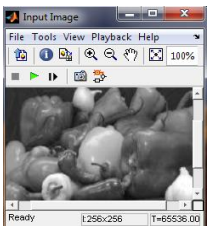


Fig 3: Original Image

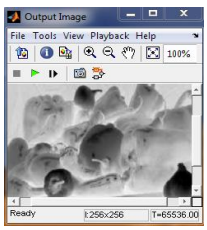


Fig 3: Output Image

4. Image Segmentation using Threshold

Image segmentation can be used to separate pixels associated with objects of interest from the image background. This is an important step in many imaging applications of automated analysis and robotics. We demonstrate segmentation on a simple pixel-by-pixel basis using threshold decisions. We use histogram of a image to determine the threshold value. Fig. shows the original cameraman image and histogram result.

Histogram shows that the object and background are well-separated. We use Mcode block and use 70 as a threshold in our demonstration. Fig. shows the MATLAB code for Mcode block and the image after segmentation results respectively.

Here we get the foreground portion as a white image. The Mcode block description is as follows:

```
function z = new pixel(x, y)
if x>y
z = x;
else
z = 1;
end
```

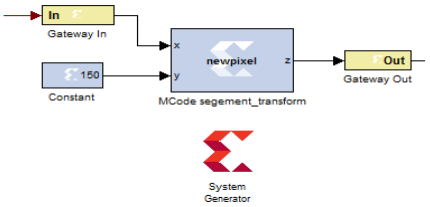


Fig 4: Algorithm for Segmentation Threshold

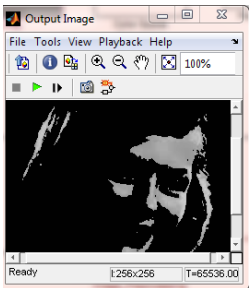


Fig 4: Original Image

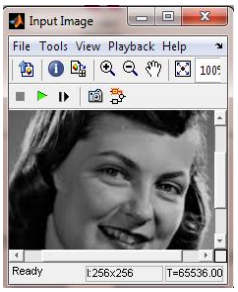


Fig 4: Output Image

5. Range highlighting Transformation

An intensity transform can also highlight a range of pixels while keeping others constant. Fig 5 shows the Xilinx blocks implementation and the resulting image.

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

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

z = x;
else
z = 1;

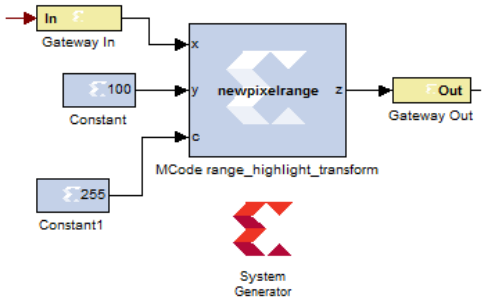


Fig 5: Algorithm for Range Highlighting Transformation

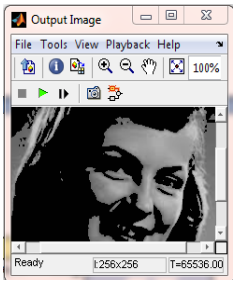


Fig 5: Original Image

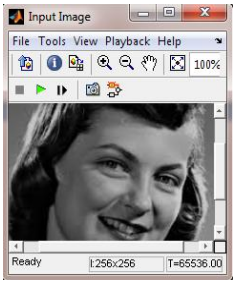


Fig 5: Output Image

5. 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 [6].

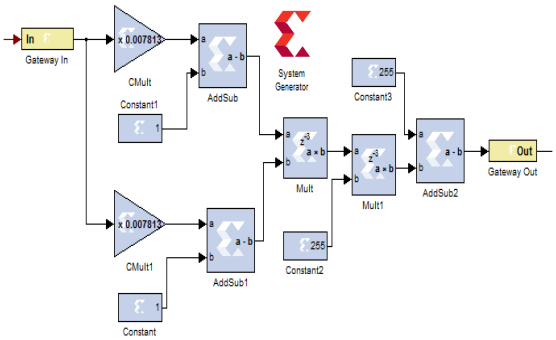


Fig 6: Algorithm for Parabola Transformation

Fig 6: Original Image

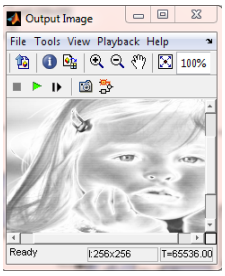


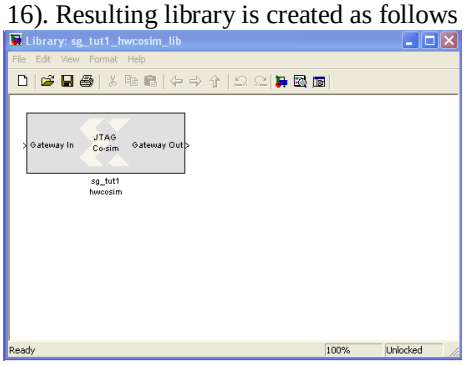
Fig 6: Output Image

V. 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.[7][8] Steps Followed in Hardware Co-simulation System generator is configured as:

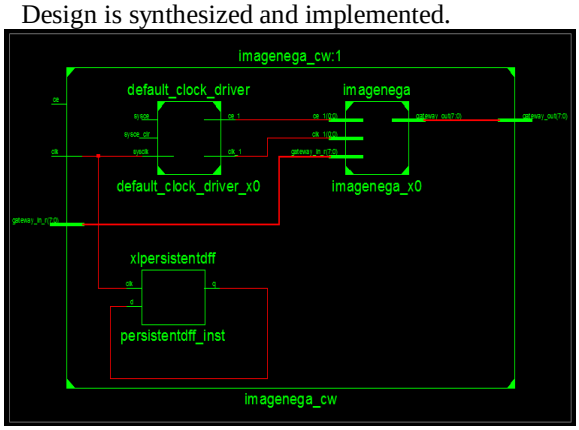
A. Choosing a Compilation Target

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



Synthesis tool: Specifies the tool to be used to synthesize the design.

- Hardware Description Language: Specifies the HDL language to be used for compilation (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.

Table showing analysis for resource estimation and simulation time for nexys3 or spartan6 FPGA

ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 4, April 2015

Algorithm/ Resources	Slice	Flip-flops	IOBs	CLKs	PSNR
Brightness Control	1	1	17	1	7.921
Contract Sketching	1	1	27	1	11.48
Negative Transformation	1	1	17	1	8.312
Range Highlight Transformation	7	1	17	1	10
Segmentation Threshold	6	1	17	1	21.89
Parabola Transformation	1	1	68	1	8.879

VI. CONCLUSION

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.

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

ACKNOWLEDGMENT

It is our almost duty and desire to express acknowledgement to the various torchbearers, who have rendered valuable guidance during the preparation of project. First of all, I extend my deepest gratitude to our guide **Dr.U.A.Kshirsagar** for providing immense support and guidance. She guide throughout never leaving me unanswerered for any of doubts. It is her constant persuasion, encouragement, inspiration and able guidance that help me in completing project successfully.

REFERENCES

[1] X.L. Liu, F.Z. Duan, and H.L. Gong, "Achievements and Prospects of Aerial Image Technology," Frontier Science, 2007(03):10~14.
[2] Ownby, M., Mahmoud,W.H., "A design methodology for implementing DSP with Xilinx System Generator for Matlab",IEEE International Symposium on System Theory, pp.404-408, 2003
[3] Xilinx Inc, "System Generator for Digital Signal Processing" http://www.xilinx.com / tools / dsp.htm.
[4] R. Harinarayan, R. Pannerselvam, M. Mubarak Ali, Dhirendra Kumar Tripathi "Feature Extraction of Digital Aerial Images by FPGA based implementation of edge detection algorithms" 978-1-4244-7926-9/11/$26.00 ©2011 IEEE.
[5] Neha. P. Raut, Prof.A.V.Gokhale " FPGA Implementation for Image Processing Algorithms Using Xilinx System Generator" 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
[6] V.Elamaran, G.Rajkumar "FPGA implementation of point processes using Xilinx system generator" Journal of Theoretical and Applied Information Technology31st July 2012. Vol.No.41
[7] Alba M.Sanchez G, Richardo Alvarez G, Sully Sanchez G, FCC and FCE BUAP, "Architecture for filtering images using Xilinx System Generator," International Journal on Mathematics and Computers in Simulation, vol.1(2), pp.101-107, May 2007.
[8] C.H.Chen, Uvais Quidwai, "Digital Image Processing: An Algorithmic Approach with MATLAB", CRC Press, 2009.
[9] DSP System Generator User guide release 12 .1.
[10] Xilinx System Generator User's Guide,.
[11] Matlab Website: http:/www.mathworks.com
[12] Mn-Ta Lee; Shih-Syong Chen; "Image copyright protection scheme using Sobel technology and genetic algorithm", 2010 International Symposium on Computer Communication Control & Automation (3CA), Volume: 1 Publication Year: 2010.
[13] Ana Toledo Moreo, Pedro Navarro Lorente, F.Soto Valles, Juan Suardiaz Muro, Carlos Fernandez Andres, "Experiences on developing computer vision hardware algorithms using Xilinx system generator," Microprocessors and Microsystems, vol. 29(8-9), pp.411-419, Nov 2005.