www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3. Issue 10 October, 2014 Page No. 8704-8709

FPGA Implementation of Canny Edge Detection Algorithm

***Ms. P.H. Pawar*[1], *Prof. R. P. Patil*[2].**

[1]PG Student, SKNCOE, E & TC Department.
Vadgaon Bk, Pune, India
Poonampawar189@gmail.com

[2]Asst. Professor, SKNCOE, E & TC Department.
Vadgaon Bk, Pune, India
rohita_jagdale@yahoo.com

Abstract* - Edge detection is the first step in many computer vision applications. Edge detection of image significantly reduces the amount of data and filters out unwanted or insignificant information and gives the significant information in an image. This information is used in image processing to detect objects in which there are some problems like false edge detection, missing of low contrast boundaries, problems due to noise etc. In this paper Canny Edge Detection algorithm is implemented. Canny Edge Detection algorithm for stored image is implemented on Virtex 5 Field Programmable Gate Array (FPGA) board. Using Xilinx platform studio and Xilinx ISE output image displayed on Video Graphics Array (VGA) monitor which is interfaced with board by using DVI connector.

Keywords: - Edge detection, Canny Edge Detection, MATLAB/ Simulink, FPGA, VGA monitor.

I. Introduction

Edge detection is a low level operation used in image processing and computer vision applications. The main goal of edge detection is to locate and identify sharp discontinuities [1] from an image. These discontinuities are due to abrupt changes in pixel intensity which characterizes boundaries of objects in a scene as well as give boundaries between different regions in the image. Such boundaries are used to identify objects for segmentation and matching purpose. These object boundaries are the first step in many of computer vision algorithms like edge based obstacle detection, edge based face recognition, edge based target recognition, image compression etc. So the edge detectors are required for extracting the edges. There are many edge detection operators available. These operators identify vertical, horizontal, step and corner edges. The quality of detected edges by these operators is highly dependent on lighting conditions, noise, objects of same density and the intensities of edges in the scene. These troubles can be solved by adjusting different parameters in the edge detector and altering the value of threshold for what an edge is considered. These operators are awfully responsive to noise and edges that contain high frequency contents. So removal of noise may result in blurred and distorted edges. A large variety of operators are existing that can extract the edges from noisy image .But these edges are not as much of exact. That is due to the occurrence of noise that they take out false edges. They do not find the borders of object having tiny change in intensities values. That result in reduced localization of boundaries that means edges. So the operator is necessary that recognize such a regular change in intensities. So there are troubles of fake edge detection, problem because of noise, absent of low contrast boundaries, more computational time etc. Software and hardware implementation using MATLAB/ Simulink, Xilinx System Generator (XSG) is objective of paper. Canny edge detection algorithm is described. Canny is gradient based edge detection algorithm.

II. Literature Survey

There are various types of operators available for edge finding. In First order derivative input image is convolved by an adapted mask to generate a gradient image in which edges are detected by thresholding. The majority classical operators like Prewitt, Sobel, and Robert are the first order derivative operators [1]. These operators are also said as gradient operators. These gradient operators spot edges by looking for maximum and minimum intensity values. Also examine the allocation of intensity values in the neighborhood of a given pixel and find out if the pixel is to be classified as an edge. The gradient based detectors like Prewitt, Robert, Sobel, convolve the input image with their respective convolution mask to generate a gradient image. Threshold values are inured to detect edges. The output of these edge detectors is very much sensitive to the threshold. The Matlab realization of these operators uses an adaptive threshold. These operators are more sensitive to noise and having problem of false edge detection as well as missing edges. To overcome these disadvantages an optimal edge detection Canny algorithm [2] is present. Canny proposes a new approach to edge detection that is optimal for step edges contaminated by noise. The optimality of the detector is related to three criteria: Low error rate – edges should not be missed and there must not be spurious responses. Localization – distance between points marked by the detector and the actual centre of the edge should be less. Response– Only single response for single edge.

Chandrashekar N.S., Dr. K.R. Nataraj implemented Canny algorithm [3] onto a Xilinx Virtex-4 FPGA platform and tested using ModelSim. The design development is done in VHDL and simulates the results in modelsim 6.3 using Xilinx 12.2. Tejaswini H.R. et.al implemented distributed canny edge detection algorithm [4] that gives cut-back in latency, optimal use of memory and increase in throughput with no defeat in edge detection performance as related to original edge detection algorithm on Xilinx's Nexys 3 FPGA. Chaithra.N.M, K.V. Ramana Reddy present canny edge detection algorithm implemented on Spartan 3E FPGA and developed VGA interfacing for displaying images on the screen. In this paper 128×128 Image is displayed on the monitor through FPGA. B.Muralikrishna, P.Sujitha, Habibulla Khan created IP Core for application i.e. edge Detection and this core is added to the Xilinx Library and integrated into design and added VHDL code according to functionality. Finally able built embedded system and added design to that system and executed the complete system on FPGA.

III. SYSTEM DESIGN

In this paper, image edge detection using Canny and Beamlet transform edge detection algorithm is described using MATLAB/Simulink. Canny is realized on Virtex 5 FPGA board. Edge is basic and important feature of an image. Edge is the boundary between two regions in image which relatively differentiate gray-level properties. In digital image edge is defined as a sharp change in intensity between neighbouring pixels.

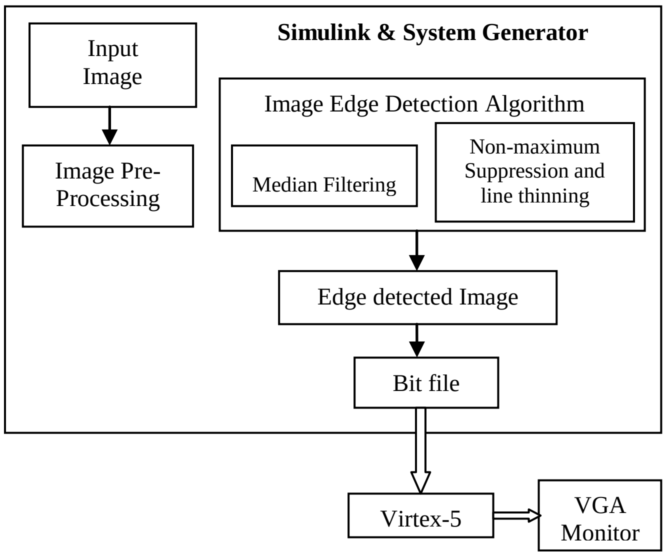
Simulink & System Generator
Input Image
Image Pre-Processing
Image Edge Detection Algorithm
Median Filtering
Non-maximum Suppression and line thinning
Edge detected Image
Bit file
Virtex-5
VGA Monitor

Fig. 1 System Block Diagram

Basically image is composed of object and background. Recognition of object shape from its background is important in some image processing applications where only shape of object matters and no any other information of image. This process is known as edge detection. A system block diagram explains basic idea about proposed hardware implementation on FPGA platform in Fig. 1. It consists of FPGA simulation board on which design of proposed algorithms for edge detection is going to dump. Basic idea about hardware implementation of canny edge detection algorithm on FPGA board is shown in Fig.3.1. Video Graphics Array (VGA) is to be interface with Virtex5 ML506 FPGA evaluation board
The overall system consists of following blocks image input, image pre-processing, image edge detection algorithm, FPGA board, Output on VGA monitor. The Simulink models are generated for system using Simulink block sets, which are present in MATLAB/Simulink. Input image used is image stored in memory. Then the resizing and preprocessing is done using the median filtering on the image to reduce noise which is present in the input image. After preprocessing the edge detection of the image is done and then the post processing is done using the non-maximum suppression and line thinning. The edge image is then detected with the help of Matlab/Simulink model. The Xilinx Platform Studio (XPS) is used to interface peripherals [4]. Then XPS along with xillinx ISE 14.1 generate a .bit file for the edge image and then downloaded to FPGA.

1) Input Image: Input image is a digital image which is of different formats such as .jpg, .bmp, .jpeg, .tiff, .gif, .png which are stored in memory. These images are colour as well as black and white images. These images are converted from RGB to gray colour using the MATLAB code. These images are selected in the Simulink model using the 'image from file/workspace' block. The size of the stored image is resized to 512X512 in the MATLAB code.

2) Image edge detection Algorithm: Here Canny edge detection algorithm is implemented on FPGA board, which is implemented in following steps: pre-processing, gradient calculation, non-maximum suppression, double thresholding, hysteresis. These steps are explained in detailed in next section.

The Virtex-5 ML506 evaluation board is user friendly which is used to realize this system. For this system awfully fewer hardware is required such as RS232, JTAG cable and VGA monitor. The downloading process of Virtex-5 is easy to understand. This FPGA board gives especially rapid results.

A. Software Design

In this paper, image edge detection algorithm is used for the detection of edges of object. The Fig. 2 shows the flow of the Canny edge detection algorithm.

1) Image pre-processing: Basically in many image processing applications input image is colour image. Here also input image is colour image. For edge detection purpose no any colour information is required so second step to minimize this unnecessary colour data. For this data reduction image is converted into grayscale image. This grayscale image containing some noise in it so filtering is applied. Median filter is preferred for this purpose.

2) Convolution with masks, Gradient and direction Calculation: Here, suppose $G(x, y)$ is a 2D Gaussian mask and $I(x, y)$ is the image, the first-order derivative of Gaussian is *gx*(*x*, *y)* and *gy*(*x*, *y*). Then the gradient of vertical direction *Ex*(*x*, *y)* and horizontal direction *Ey* (*x*, *y)* can be computed by the following equations:

$$G\,(x,y) = \frac{1}{2\pi\sigma^2}\{\,e^{(\frac{x^2+y^2}{2\sigma^2})}\} \tag{1}$$

$$gx(x,y) = \frac{\partial G}{\partial x} = \frac{1}{\pi\sigma^2}x\,\{\,e^{(-\frac{x^2+y^2}{2\sigma^2})}\} \tag{2}$$

$$gy(x,y) = \frac{\partial G}{\partial y} = \frac{1}{\pi\sigma^2}y\,\{\,e^{(-\frac{x^2+y^2}{2\sigma^2})}\} \tag{3}$$

$$Ex(x,y) = gx(x,y) * I(x,y) \tag{4}$$

$$Ey(x,y) = gy(x,y) * I(x,y) \tag{5}$$

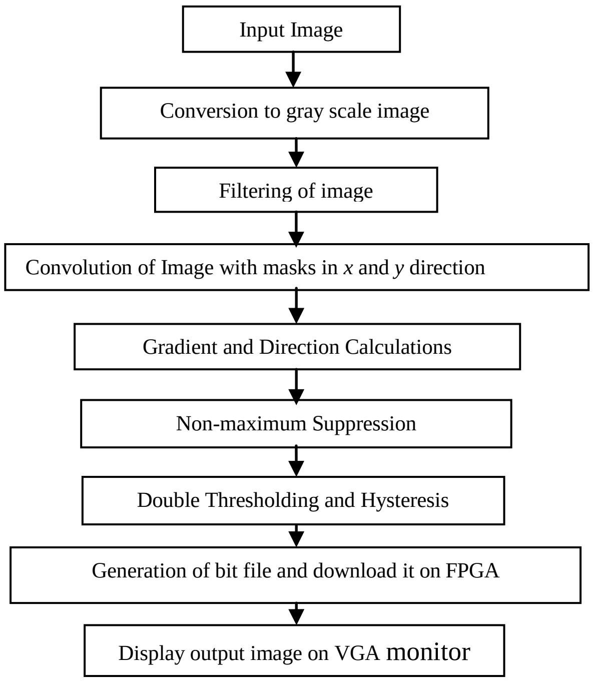
Input Image
Conversion to gray scale image
Filtering of image
Convolution of Image with masks in x and y direction
Gradient and Direction Calculations
Non-maximum Suppression
Double Thresholding and Hysteresis
Generation of bit file and download it on FPGA
Display output image on VGA monitor

Fig. 2 Flow of Canny Edge Detection Algorithm

Then, the gradient magnitude and angle can be calculated as follow:

$$Grad(x,y) = \sqrt{Ex^2(x,y) + Ey^2(x,y)} \tag{6}$$

$$\theta(x,y) = \tan^{-1}\left\{\frac{Ey(x,y)}{Ex(x,y)}\right\} \tag{7}$$

3) Non-maximum suppression: Non maximum suppression, which is used to reduce the edge thickness for improving localization. Once the direction of the gradient is known, the pixel which is not the local maximum is eliminated is referred as NMS [2]. The comparison is made between the current pixel and its 8-neighbors, along the direction of the gradient. A 3×3 window is adopted during the comparison.

4) Thresholding and and hysteresis: Since the output of the non maximum suppression stage contains some spurious edges resulted from noises, these responses which are called 'streaking' can be eliminated by the use of hysteresis thresholding. Next step in flow is generation of bit file to download it on FPGA board and display output image on VGA monitor.

B. Simulink model for Canny Edge Detection

The Simulink model shown in Fig. 3 is prepared in MTALAB with the help of Simulink library browser which contains the Image and Video processing block set. The software model for Canny Edge Detection consists of Image from file, Video Viewer, Embedded function block, resource estimator, virtex2 line buffer, median filter, constant, rational, mux blocks. Both input and output images are displayed using the video viewer block. While running the Simulink model each time different image dataset has to be selected.

1) Image from File Block: This block is basically used to access the input images. By double-clicking on this block, select the path where the input image is stored with size 512 × 512. Each time select the images from file to run the different datasets.

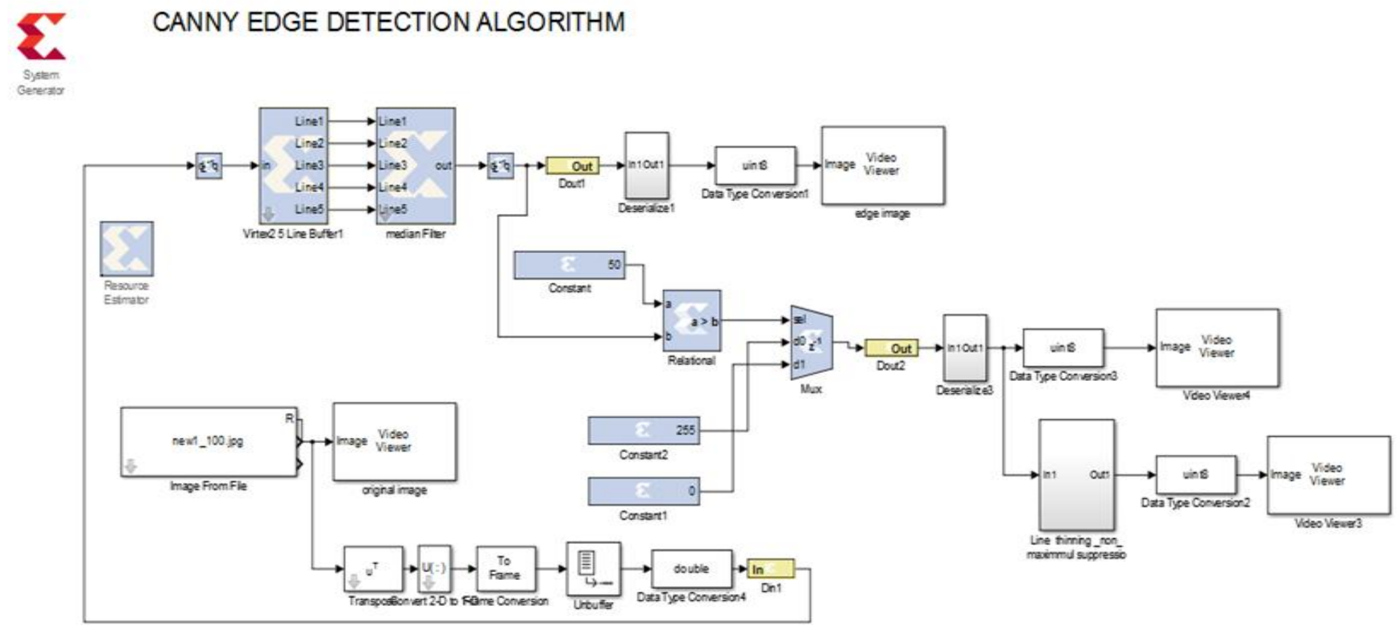

Fig. 4 Simulink Software for Canny Edge Detection

The use of Image from File block is to import an image from a supported image file also to read the image from file. If the image is a M-by-N array, the block output a intensity or binary image, where M and N are the digit of rows and columns in the image. If the image is a M-by-N-by-P array, the block outputs a color image, where M and N are the number of rows and columns in each color plane. It can provide the image signal selection for this model. The image signal is set to separate the color but for the system, gray scaled images are required. So color signals are terminated.

2) Video viewer: The Video Viewer block enables to view a binary, intensity, or RGB image or a video. The block gives simulation controls for pause, play, and step while running the model. The block also provide pixel section analysis tools. In code generation, Simulink Coder software does not generate code for this block. This block is used to display the images in MATLAB. The input image namely original image and output image namely video viewer 3 are displayed using this block.

3) Gateway In: This Xilinx block is the basic element of Xilinx Block set. This block is used to convert the Simulink input of type integer, single or double format to Xilinx data type. This Xilinx block is the basic element of Xilinx Block set. This block is listed in the following Xilinx block set libraries: Basic Elements, Data Types, Floating-Point and Index. The Xilinx Gateway In blocks are the inputs into the Xilinx portion of your Simulink design. It converts Simulink integer, fixed-point and double data types into the System Generator fixed-point type. Each block define a top-level input port in the HDL design generated by System Generator. This block is used to convert the Simulink input of type integer, single or double format to Xilinx data type.

4) Gateway Out: This block is listed in the following Xilinx block set libraries: Basic Elements, Data Types, Floating-Point and Index. Xilinx Gateway Out blocks are the outputs from the Xilinx portion of your Simulink design. This block convert the System Generator fixed-point or floating-point data type into a Simulink integer, single, double or fixed-point data type. According to its configuration, the Gateway Out block can either define an output port for the top level of the HDL design generated by System Generator, or be used simply as a test point that is trimmed from the hardware representation. This Xilinx block is used to convert the Xilinx data type of fixed-point or floating point into Simulink data type.

5) Serialize: This block is basically a sub-system which is used to convert the data into serial form, since the Xilinx system generator supports serial data transfer.

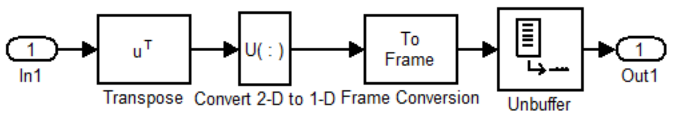

Fig. 5 Simulink Model for Serialization

Fig. 5 shows the different steps of serialize block, are their like transpose of frame and then conversion of 2D to 1D for serialization and finally un-buffering of the data to serially transfer it to FPGA. Transpose gives the data in form of [Mx1] vector. Then that vector is converted into 1-Dimension, and then un-buffers the data to serially transmit it.

6) Register: This block is listed in the following Xilinx Block set libraries: Basic Elements, Control Logic, Memory, Floating-Point and Index. The Xilinx Register block models a D flip-flop-based register, having latency of one sample period.

7) Data Type Conservation: Convert the input to the data type and scaling of the output. The conversion has two probable goals. One is to have the Real World Values of the input and the output is equal. The other goal is to have the Stored Integer Values of the input and the output is equal. Overflows and quantization errors can prevent the goal from being fully achieved.

8) Virtex2-5 Line Buffer: The Xilinx Virtex2-5 Line Buffer reference block buffers a sequential stream of pixels to construct 5 lines of output. Each line is delayed by N samples, where N is the length of the line. Line 1 is delayed 4*N samples, each of the following lines are delay by N fewer samples, and line 5 is a copy of the input. This block uses Virtex2 Line Buffer block which is located in the Imaging library of the Xilinx Reference Block set.

9) Median Filter: The Xilinx 5x5 Filter reference block is implemented using 5 n-taps MAC FIR Filters. The filters can be found in the DSP library of the Xilinx Reference Block set. Nine different 2-D filters have been provided to filter gray scale images. The filter can be selected by changing the mask parameter on 5x5 Filter block The 2-D filter coefficients are stored in a block RAM, and the model makes no specific optimizations for these coefficients.

10) Resource Estimator: This block is listed in the following Xilinx Block set libraries: Tools and Index. The Xilinx Resource Estimator block provides fast estimates of FPGA resources required to implement a System Generator subsystem. These estimates are calculated by invoking block-specific estimators for Xilinx blocks, and adding these values to obtain aggregated estimates of lookup tables (LUTs), flip-flops (FFs), block memories (BRAM), 18x18 multipliers, tri-state buffers, and I/Os.

10) De-serialize: This Xilinx block is again used to convert the data in parallel i.e. de-serialize form, since it is given as input to Simulink block. Fig. 6 shows de-serialize block which is the exactly inverse process of the serialize process. This will first buffer the data and then convert it into 2-Dimention form and then transpose is taken of the data.

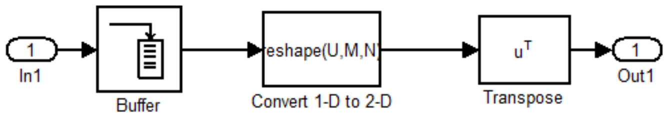

Fig. 6 Simulink Model for De-serialization

11) Embedded MATLAB function block: This block is also called as S-function block; in this block code for the designed algorithm is embedded. This block is programmed for line thinning and non-maximum suppression.

12) Xilinx System Generator Token: Xilinx System Generator token available in Xilinx block set is basically used for FPGA compatibility i.e. HDL code generation of the designed Simulink model. Using both Simulink and Xilinx blocks in the model helps for Xilinx System Generator simulation, code generation, and synthesis. Double clicking on the Xilinx system generator will ask for the path to store the generated code, for which device to generate, the code and in which language like HDL, VHDL. For this system the Virtex-5 board is used and VHDL code is generated. VGA interfacing is done using Xilinx Platform Studio (XPS). And programming FPGA through XPS output image displayed on VGA monitor.

IV. IMPLEMENTATION AND RESULTS

Image Edge detection algorithm is designed in MATLAB/Simulink and it implemented on Virtex-5 FPGA as shown in the following Fig. 7.

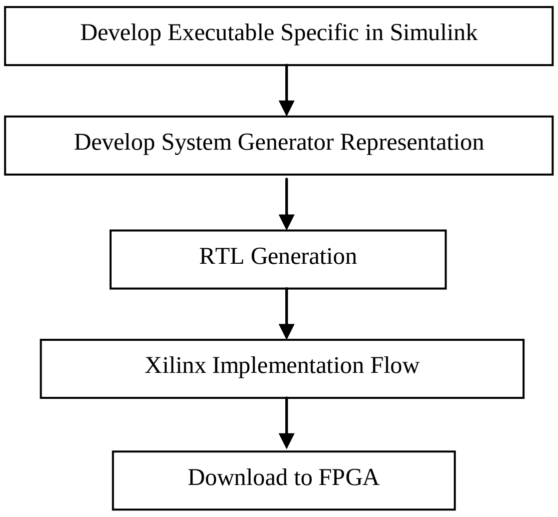
Develop Executable Specific in Simulink
Develop System Generator Representation
RTL Generation
Xilinx Implementation Flow
Download to FPGA

Fig. 7 Design and Implementation Flow

Develop Executable Specific in Simulink: In this step generate the Simulink model for the system using Simulink block sets in MATLAB.

Develop System Generator Representation: Taking the reference of Simulink model generate another system generator model for the same system using Xilinx block sets for the compatibility of FPGA implementation. Then the VHDL code is generated using System generator token. The VHDL code which is generated using the system generator is open in the Xilinx ISE 14.1. Then synthesis of the code will give the information about the errors, warnings, RTL schematic of that code, device utilization summary etc. VGA interfacing via XPS is done which gives easy way of hardware interfacing with FPGA boards.

RTL schematic of the Simulink model: Synthesis is a process by which an abstract form of designed circuit behavior or register transfer level (RTL) has been converted into a design implementation i.e., in terms of logic gates. The synthesis of VHDL code has been carried out by Xilinx Synthesis Technology (XST) tool, which is a part of Xilinx ISE 14.1 software. The Top-level RTL schematic for the Background Subtraction Algorithm developed and implemented on FPGA is shown in Fig. 8. This is a schematic representation of the pre-optimized design shown at the Register Transfer Level (RTL). This representation is in terms of generic symbols, such as adders, multipliers, counters, AND gates, OR gates and is generated after the HDL synthesis phase of the synthesis process. This system blocks are designed for the Virtex-5 ML506 board.

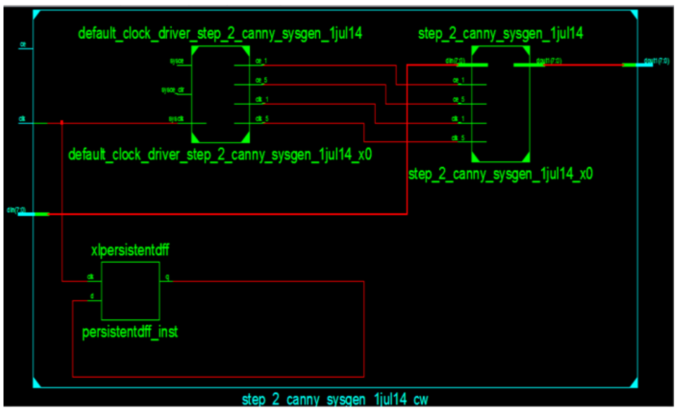
default_clock_driver_step_2_canny_sysgen_1jul14
step_2_canny_sysgen_1jul14
default_clock_driver_step_2_canny_sysgen_1jul14_x0
step_2_canny_sysgen_1jul14_x0
xlpersistentdff
persistentdff_inst
step_2_canny_sysgen_1jul14_cw

Fig. 8 Top-level RTL Schematic

Device Utilization Summary: Input image is of size 512 X 512 and edge detected image is of size 200 X 200 is done using Xilinx 14.1, on Virtex-5 evaluation board. Image is resized to 200 X200.

TABLE I. Device Utilization Summary

Device Utilization Summary			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	452	32640	1%
Number of Slice LUTs	205	32640	0%
Number of fully used LUT-FF pairs	197	460	42%
Number of bonded IOBs	17	480	3%
Number of BUFG/BUFGCTRLs	1	32	3%

Device utilization summary shows information related to device utilization analysis. Detailed reports provide access to reports that are generated as the design is processed. Secondary reports - provides access to the secondary reports of users choice. Table I shows the device utilization summary for the system. Slice registers available are 32640, out of which 451 are used for implementation which is equal to 1% of available resources. Number of slice LUTs used is 0% which gives 205 usages out of 32640. Usage of LUT-FF pairs is 197 out of 460 which is equal to 42%. Utilization of Bonded IOBs is 17 out of 480 which 3% of available IOBs. Numbers of BUFG used are 1 out of 32 which is 3% of available. These summaries tell that the use of the device or hardware for system is very less.

The system is implemented using Xilinx 14.1 and evaluation board of Virtex-5 ML506. Connection is made using JTAG cable for downloading the program to FPGA, DVI connector for VGA interface.

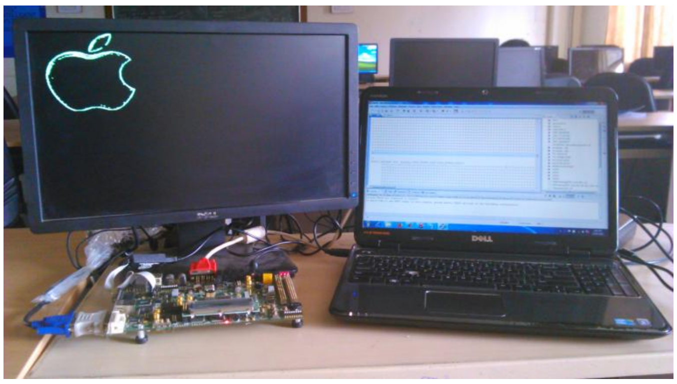

Fig. 9 Experimental Setup for System

Fig. 9 shows the experimental setup for Virtex-5 board. On laptop the .bit file and Simulink models in MATLAB are generated. And using Xilinx 14.1 version Xilinx platform studio the .bit file is synthesis and downloaded to Virtex-5 board. Output is displayed on VGA monitor for this DVI connector is used to interface the VGA monitor to Virtex-5 board. RS232 serial cable is used to transmit the image serially to the board. After the synthesis of the VHDL code the synthesis report is generated in which the device utilization summary is given.

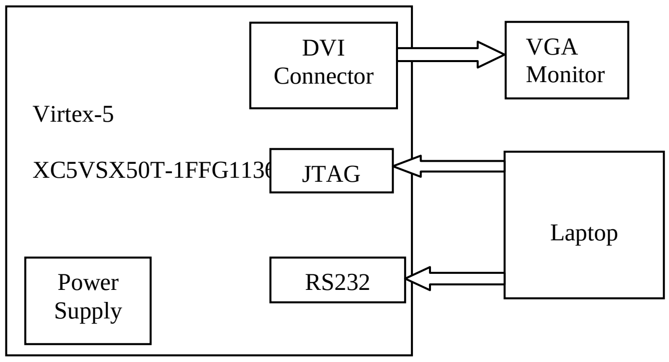

Fig. 10 Interfacing of Virtex-5 with VGA

The above Fig. 10 showes interfacing of Virtex-5 with VGA monitor for display the edge detected image. VGA monitor is interfaced with Virtex-5 using DVI connector which is avilable on the board. The JTAG cable is connected to laptop and board is used to download the program to FPGA. The RS232 cable is used to transfer the images from laptop to board for the processing, it transfer the data serially.

The Fig. 11(a) shows the input image which is of size 225 x 225 Fig. 11(b) shows the edge detected image is of size 200 X 200 which is displayed on VGA monitor as shown in Fig. 9. In output image edge pixels are having white colour and non edge pixels are of black colour.

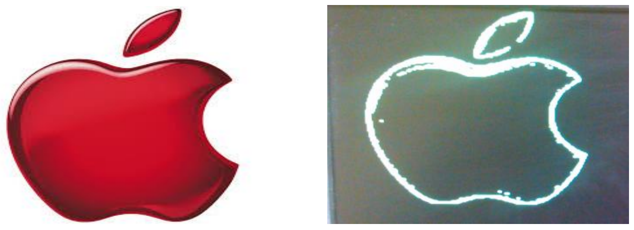

Fig. 11(a) Input image (b) Output image

Fig. 12(a) shows another image of flower of size 217X230 which is inputted to system generator model of Canny and output image of 200 X 200 is shown in Fig.12(b).

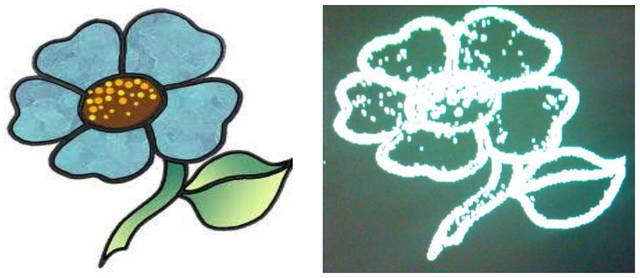

Fig.12(a) Input Image (b) Output Image

VI. CONCLUSION

The implementation of Canny Edge Detection Algorithm on FPGA is described, and it is carried out successfully. The hardware and software implementation of system is found to be working properly. The MATLAB/Simulink models are used for the compatibility of implementing the image processing system on FPGA. Because it has direct functions related to the image processing and it also provides the Xilinx block sets for the FPGA implementation. The Virtex-5 ML506 evaluation board is user friendly which is used to implement this system. For this system very less hardware is required such as JTAG cable, RS232 and VGA monitor. Resources used are also very less for implementation. The result of Canny Edge Detection algorithm detects more edges whit less missing edges. It is not noise sensitive. It is useful in those digital image applications where only shape and size of image is required.

REFERENCES

[1] Raman Maini, Dr. Himanshu Aggarwal "*Study and Comparison of Various Image Edge Detection Techniques*" IJIP, Volume (3): Issue (1).

[2] Xiaoyang Li, Jie Jiang, Qiaoyun Fan "*An Improved Real-time Hardware Architecture for Canny Edge Detection Based on FPGA*" Third International Conference on Intelligent Control and Information Processing July 15-17, 2012 - Dalian, China.

[3] Chandrashekar N.S., Dr. K.R. Nataraj "*Design and Implementation Of A Modified Canny Edge Detector Based On FPGA*" International Journal of Advanced Electrical and Electronics Engineering, (IJAEEE), ISSN (Print): 2278-8948, Volume-2, Issue-1, 2013

[4] Tejaswini H. R, Vidhya N, Swathi R Varma, Santhosh B "*An Implementation Of Real Time Optimal Edge Detector And VLSI Architecture*" International Conference on Electronics and Communication Engineering, 28th April-2013

[5] B.Muralikrishna, P.Sujitha, Habibulla Khan "Real Time Edge Detection Modelling With FPGA" International Journal of VLSI and Embedded Systems-IJVES, Vol 04, Article 06103; June 2013

[6] Chaithra.N.M., K.V. Ramana Reddy "*Implementation of Canny Edge Detection Algorithm on FPGA and displaying Image through VGA Interface*" International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-6, August 2013