International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Research Gate Impact Factor (2018): 0.28 | SJIF (2019): 7.583

VLSI Architecture Design and Implementation of CANNY Edge Detection Subsystem

Ragi R G[1], Jayaraj U Kidav[2], Roshith K[3]

[1]Department of Electronics and Communication Engg, College of Engineering, Vadakara, Kerala, India

[2]National Institute of Electronics & Information Technology, (NIELIT) Calicut, Kerala, India

[3]Department of Electronics and Instrumentation Engg, College of Engineering, Vadakara, Kerala, India

Abstract: ***In Edge detection is one of the most fundamental algorithms in digital image processing. The Canny edge detector is the most implemented edge detection algorithm because of its ability to detect edges even in images that are intensely contaminated by noise. In this paper, a modified canny edge detector is designed implemented in MATLAB and implemented in FPGA. The mask for gradient calculation and in nonmaximal suppression bilinear interpolation of four pixels are considered. This edge detector is implemented as a preprocessing stage in iris detection subsystem. The motivation in designing the hardware modules of canny edge detector was to reduce its complexity, enhance its performance and to make it suitable development on a reconfigurable FPGA based platform for VLSI implementation.***

Keywords: Edge Detection, Canny Edge detector, FPGA

1. Introduction

Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction. Computer vision is a field of artificial intelligence that works on enabling computers to see, identify and process images in the same way that human vision does, and then provide the appropriate output within fraction of a second. Detection of edges in an image is a very important step towards understanding image features. Edges consist of meaningful features and contain significant information. It significantly reduces the image size and filters out information that may be regarded as less relevant, thus preserving the important structural properties of an image.Edge detection includes a variety of mathematical methods that aim at identifying points in a digital image at which the image brightness changes sharply or, more formally, has discontinuities. The points at which image brightness changes sharply are typically organized into a set of curved line segments termed edges.

In digital image processing, the term edge is a collection of the pixels, it refers to the part where the brightness of the image local area changes significantly in digital image processing. The general methods of edge detection are first order Derivative-gradient method, Second-Order Derivative and Optimal Edge Detection. A lot of edge detection algorithms, such as, Robert detector, Gauss-Laplace detector, Prewitt detector and Canny detector. Among the existing edge detection algorithms, the Canny edge detector has remained a standard for many years and has best performance. The primary disadvantage of using Canny edge detector is that it consumes a lot of time due to its complex computation andit is difficult to implement to reach the real-time response

FPGA, alternative to the custom ICs, can be used to implement an entire System On one Chip (SOC). The main advantage of FPGA is ability to reprogram. User can reprogram an FPGA to implement a design and this is done after the FPGA is manufactured. This brings the name "Field Programmable". FPGA are easy to implement within a short time with the help of CAD tools. FPGAs are some of the new trending areas of VLSI. The proposed system implemented an efficient programmable system with hardware to recognize human iris using MATLAB and FPGA.

2. Materials and Methods

2.1 Proposed method

The block diagram of the proposed system shown in Fig -1. The system is implemented with a canny edge detector with a slight modification in the original Canny algorithm using MATLAB and designed an efficient architecture for the algorithm for hardware implementation. The algorithm is implemented using a Hardware description language (here Verilog HDL) & Perform simulation and synthesize using Xilinx Design tools and verified the result. Human iris is used as the input parameter for edge detection and recognition.

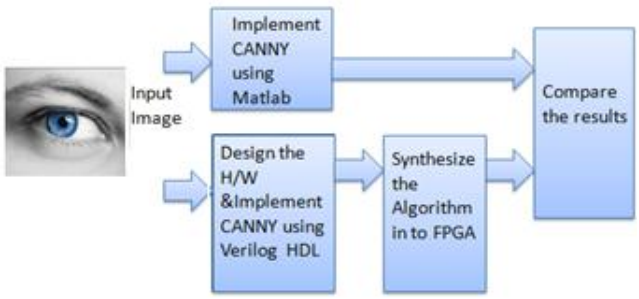

Figure 1: The flow diagram of proposed system

2.2 Feature Description

The Human Irisis used as the data for the proposed system. The iris is a thin circular diaphragm, which lies between the cornea and the lens of the human eye. A front-on view of the iris is shown in Fig-2. The iris is perforated close to its centre by a circular aperture known as the pupil.

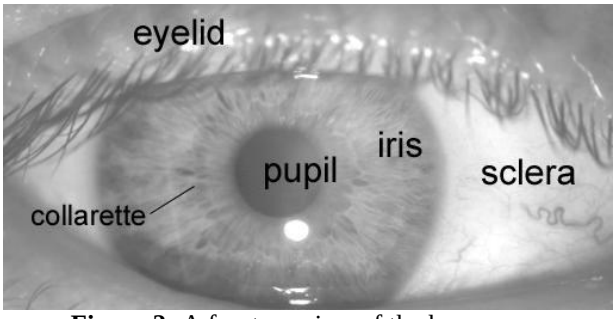

Figure 2: A front-on view of the human eye

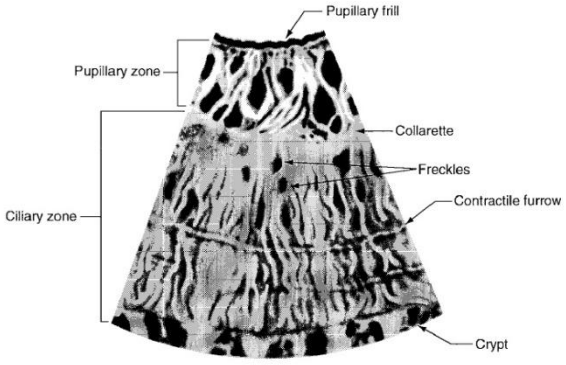

Figure 3: The structure of the iris seen in a frontal sector

The iris consists of a number of layers, the lowest is the epithelium layer, which contains dense pigmentation cells. The stromal layer lies above the epithelium layer, and contains blood vessels, pigment cells and the two iris muscles. The density of stromal pigmentation determines the colour of the iris. The externally visible surface of the multi-layered iris contains two zones, which often differ in colour. An outer ciliary zone and an inner pupillary zone, and these two zones are divided by the collarette – which appears as a zigzag pattern. The structure of the iris seen in frontal sector are shown in Fig.3 An iris recognition system is composed of three main stages. Pre-processing stage Feature extraction stage and Recognition stage.

2.3 Data Acquisition

Data set of iris comprised of images obtained from CASIA Iris Image Database V3.0 (or CASIA-Iris V3 for short)is used as the date set. CASIA-IrisV3 contains a total of 22,051 iris images from more than 700 subjects. All iris images are 8 bit gray-level JPEG files, collected under near infrared illumination.The data is pre-processed and used for edge detection, The pre-processing steps used are as follows

2.3.1Pre-processing Stage: This includes determining the boundary of the iris within the eye image, and extracts the iris portion from the image to facilitate its processing. The input to this stage is the eye image and the aim is to detect the iris portion which can be approximated by two circles, one is the iris/sclera (outer) boundary, and another interior to the first is the iris/pupil (inner) boundary.

2.3.2 Locating the Iris Region: The first step in the pre-processing stage is to apply one of the edge detection techniques to get an edge map of the iris image to enable determining all boundaries of the iris. After getting the edge map of the eye image a circular Hough transform is applied to detect the two circles of the iris/sclera (outer) and iris/pupil (inner) boundary. Then a linear Hough transform to detect the upper and lower eyelids if they are present in the image.

2.3.3 Iris Representation: Once the iris region is successfully detected the iris region is transformed so that it has fixed dimensions in order to allow comparisons. The dimensional inconsistencies between eye images are mainly due to the stretching of the iris caused by pupil dilation as a result of varying levels of illumination. Other sources of inconsistency include, varying imaging distance, rotation of the camera, head tilt, and rotation of the eye within the eye socket. A normalization process is needed to produce iris regions having the same dimensions, so that two photographs of the same iris under different conditions will have characteristic features at the same spatial location. This is realized by remapping each point within the iris region to a pair of polar coordinates (ρ, θ).

2.3.4 The Feature Extraction Stage: In order to provide accurate recognition of individuals, the most discriminating information present in the iris pattern must be extracted. Only the significant features of the iris must be encoded so that comparisons between templates can be made. The template that is generated in the feature encoding process will also need a corresponding matching metric, which gives a measure of similarity between two iris templates. This metric should give one range of values when comparing templates generated from the same eye, known as intra-class comparisons, and another range of values when comparing templates created from different irises, known as inter-class comparisons. These two cases should give distinct and separate values, so that a decision can be made with high confidence as to whether two templates are from the same iris, or from two different irises.

2.3.5 The Recognition Stage: In this stage the identification and verification of different iris is done by comparing the feature vector extracted from the iris with the other feature vectors to identify the person with this iris.

2.4 The Canny Edge Detection Algorithm

The Process of Canny edge detection algorithm has following different steps:

2.4.1Smoothing using Gaussian filter: All images taken from a camera will contain some amount of noise, Since the Canny edge detector is susceptible to noise present in raw unprocessed image data the noise is to be removed. A filter based on a Gaussian (bell curve), where the raw image is convolved with a Gaussian filter is used to filter out the noise,. The kernel of a 5X5 Gaussian filter with a standard deviation of σ = 1.4 is shown in Fig- 4.

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}.$$

Figure 4: Gaussian filter

$$\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}$$

Figure 5: Mask to compute Gradients

2.4.2 Find the intensity gradient of the image: The Canny's algorithm marks boundaries at maxima in the magnitude of the image gradient in the direction of the gradient. These areas are found by determining gradients of the image. The original Canny's code use the mask shown in Fig-5 to compute X and Y components of the gradient.

An edge in an image may point in a variety of directions, so new implementation uses the mask [-1 0 1] to compute first differences in four directions: H(horizontal), V(vertical), D_1 and D_2 (diagonal).The X and Y components of the gradients are computed by projecting the diagonal differences onto the axes are$G_X = H + \frac{D_1+D_2}{2}$ and $G_Y = V + \frac{D_1-D_2}{2}$. The gradient magnitudes (also known as the edge strengths) can then be determined as an Euclidean distance measure by applying the law of Pythagoras equation $G = \sqrt{G_x^2 + G_y^2}$.. The direction of the edges is determined and stored to use for further processing as shown in Equation below.

$$\theta = \arctan(\frac{G_y}{G_x})$$

2.4.3 Non-maximum suppression: The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges. Basically this is done by preserving all local maxima in the gradient image, and deleting everything else. The algorithm is for each pixel in the gradient image. Given estimates of the image gradients, a search is then carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. The edge strengths are indicated both as colors and numbers, while the gradient directions are shown as arrows. Almost all pixels have gradient directions pointing north. They are therefore compared with the pixels above and below. The pixels that turn out to be maximal in this comparison are marked with white borders. All other pixels will be suppressed. The resulting edge pixels are marked with white borders.

The original canny algorithm uses a 9-pixel neighborhood. The normal to the edge direction (the gradient) is shown as an arrow, and it has components (u_x,u_y). To non-maximum suppress the gradient magnitude in this direction, there is only discrete values of the gradient at points $P_{i,j}$. Three points are required for non-maximum suppression, one of which will be $P_{x,y}$ and the other two should be estimates of the gradient magnitude at points displaced from $P_{x,y}$ by the vector **u.**

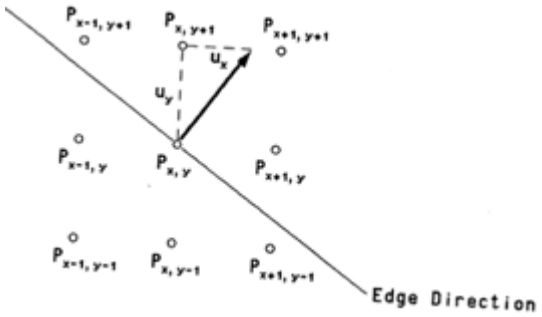

Figure 6: Support of the non-maximum suppression operator

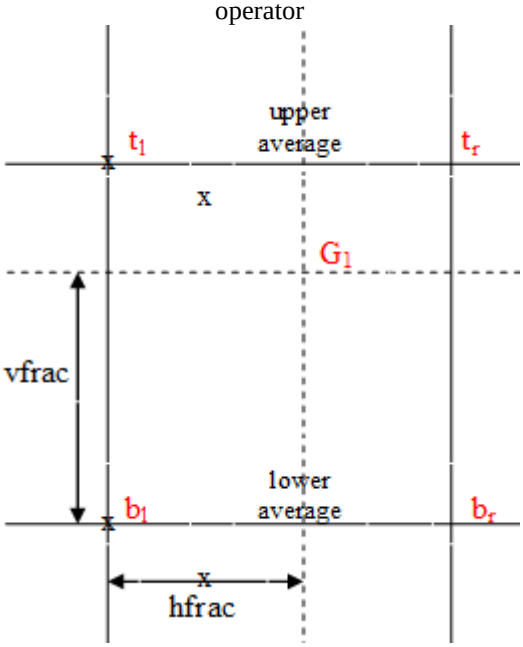

Figure 7: Bilinear interpolation

Now for any **u** consider the two points in the 8-pixel neighborhood of $P_{x,y}$ which lie closest to the line through $P_{x,y}$in direction **u**. The gradient magnitude at these two points together with the gradient at the point $P_{x,y}$ define a plane which cuts the gradient magnitude surface at these points. The plane is used locally approximate the surface, and to estimate the value at a point on the line. For example, in Fig -6 the value of a point in between $P_{x,y+1}$ and $P_{x+1,y+1}$ that lies on the line is estimated. The value of the interpolated gradient is

$$G_1 = \frac{u_x}{u_y}G(x+1, y+1) + \frac{u_y - u_x}{u_y}G(x, y+1)$$

Similarly the interpolated gradient at a point on the opposite side of $P_{x,y}$is

$$G_2 = \frac{u_x}{u_y}G(x-1, y-1) + \frac{u_y - u_x}{u_y}G(x, y-1)$$

The point $P_{x,y}$ is marked as a maximum if G(x,y) > G_1 and G(x,y) > G_2. The interpolation is similar for other gradient directions, and it will always involve one diagonal and one non-diagonal point. In practice, the divisions can be avoided by multiplying through by u_y. This scheme involves five multiplications per point, but this is not excessive, and it performs much better than a simpler scheme which compares the point $P_{x,y}$ with two of its neighbours. It also performs better than a scheme which used an averaged value for the gradient along the edge, rather than just the value at $P_{x,y}$.

In this work, the centre pixel $P_{x,y}$ is compared with two intensities G1 and G2,where G1 and G2 are the gradient obtained by the bilinear interpolation of four neighboring cells. The four cells are the four immediate pixels to the point where gradient orientation meet a circle of radius 2.

2.4.4 Bilinear interpolation: In computer vision and image processing, bilinear interpolation is one of the basic resampling technique. Bilinear interpolation uses only the 4 nearest pixel values which are located in diagonal directions from a given pixel in order to find the appropriate color intensity values of that pixel.

Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel's computed location. It then takes a weighted average of these 4 pixels to arrive at its final, interpolated value. The weight on each of the 4 pixel values is based on the computed pixel's distance (in 2D space) from each of the known points.

To calculate G1 the bilinear interpolation of four pixels t_l, t_r,b_l and b_r from Fig-7 the following equations can be used. In the Fig-7 hfrac and vfrac represents the fractional part of horizontal and vertical offset corresponding to the particular orientation ie fractional part of xoff and yoff. First the upper average and lower average are calculated using bilinear interpolation as

$$upper\ avg = (1 - h_{frac})t_l + h_{frac}\, t_r = t_l + (t_r - t_l)h_{frac}$$

$$lower\ avg = (1 - h_{frac})b_l + h_{frac}\, b_r = b_l + (b_r - b_l)h_{frac}$$

Then using the upper avg and lower avg by using bilinear interpolation G1 can be calculated as

$$G_1 = (1 - v_{frac})upper\ avg + v_{frac}\, lower\ avg = upper\ avg + (lower\ avg - upper\ avg)h_{frac}$$

Fig-6shows the effect of non-maximal suppression on the test image.From this stage referred to as non-maximum suppression, a set of edge points, in the form of a binary image, is obtained. These are sometimes referred to as "thin edges".

2.4.5 Edge tracking by hysteresis thresholding

Large intensity gradients are more likely to correspond to edges than small intensity gradients. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis. The edge-pixels remaining after the non-maximum suppression step are (still) marked with their strength pixel-by-pixel. Many of these will probably be true edges in the image, but some may be caused by noise or colour variations for instance due to rough surfaces. The simplest way to discern between these would be to use a threshold, so that only edges stronger that a certain value would be preserved. The Canny edge detection algorithm uses double thresholding. Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

Strong edges are interpreted as “certain edges”, and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels). Thus strong edges will (almost) only be due to true edges in the original image. The weak edges can either be due to true edges or noise/colour variations. The latter type will probably be distributed independently of edges on the entire image, and thus only a small amount will be located adjacent to strong edges. Weak edges due to true edges are much more likely to be connected directly to strong edges.

Once this process is complete the result is a binary image where each pixel is marked as either an edge pixel or a non-edge pixel. From complementary output from the edge tracing step, the binary edge map obtained in this way can also be treated as a set of edge curves, which after further processing can be represented as polygons in the image domain.

2.5 MATLAB Implementation

MATLAB is a software package for computation in engineering, science, and applied mathematics. It offers a powerful programming language, excellent graphics, and a wide range of expert knowledge. MATLAB is published by and a trademark of The Math Works, Inc. The canny edge detection is implemented in MATLAB with a slight modification in the gradient calculation. . MATLAB now optionally supports parallel computing. Still, MATLAB is usually not the tool of choice for maximum-performance computing.

The steps for implementation in MATLAB are

- Read the image
- Define different parameters: scaling, sigma, hi_thres, lo_thres, vert.horz,gamma,radius
- Function Gradient: Perform Gaussian filtering for noise removal and calculate magnitude and orientation of intensity gradient at each pixel
- Function Adjgamma: Adjusts image gamma. Gamma values in the range 0-1 enhance contrast of bright regions, values > 1 enhance contrast in dark regions.
- Function Non max sup: Gradient magnitude is non maxima suppressed in the gradient direction
- Function Hysthresh: Function performs hysteresis thresholding of an image.

2.6. Introduction to FPGA

FPGA contains a two dimensional array of logic blocks and interconnections between logic blocks. Both the logic blocks and interconnections are programmable. Logic blocks are programmed to implement a desired function and the interconnects are programmed using the switch boxes to connect the logic blocks.To get our desired design, all the sub functions implemented in logic blocks must be connected and this is done by programming. Xilinx logic block consists of one Look up Table (LUT) and one Flip-flop. An LUT is used to implement number of different functionality. The output of the LUT gives the result of the

logic function that it implements and the output of logic block is registered or unregistered output from the LUT. SRAM is used to implement a LUT.A k-input logic function is implemented using 2^kX 1 size SRAM. Advantage of such an architecture is that it supports implementation of so many logic functions, however the disadvantage is unusually large number of memory cells required to implement such a logic block in case number of inputs is large.

LUT based design provides for better logic block utilization. A k-input LUT based logic block can be implemented in number of different ways with tradeoff between performance and logic density. An n-LUT can be shown as a direct implementation of a function truth table. Each of the latch hold's the value of the function corresponding to one input combination. A wire segment can be described as two end points of interconnect with no programmable switch between them. A sequence of one or more wire segments in an FPGA can be termed as a track. Typically, an FPGA has logic blocks, interconnects and switch blocks (Input/output blocks). Switch blocks lie in the periphery of logic blocks and interconnect. Wire segments are connected to logic blocks through switch blocks. Depending on the required design, one logic block is connected to another and so on.

2.7 FPGA design flow

A simplified version of design flow is given in the Fig -8. There are different techniques for design entry. Schematic based, Hardware Description Language and combination of both etc. Selection of a method depends on the design and designer. If the designer wants to deal more with Hardware, then Schematic entry is the better choice. When the design is complex or the designer thinks the design in an algorithmic way then HDL is the better choice. Language based entry is faster but lag in performance and density.

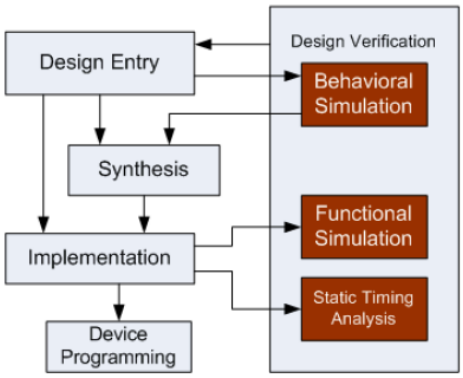

Figure 8: FPGA Design Cycle

HDLs represent a level of abstraction that can isolate the designers from the details of the hardware implementation. Schematic based entry gives designers much more visibility into the hardware. It is the better choice for those who are hardware oriented. Another method but rarely used is state-machines. It is the better choice for the designers who think the design as a series of states. But the tools for state machine entry are limited.

Synthesis: The process which translates Verilog code into a device netlist format, i.e. a complete circuit with logical elements (gates, flip flops, etc…) for the design. If the design contains more than one sub designs, ex. to implement a processor, we need a CPU as one design element and RAM as another and so on, then the synthesis process generates netlist for each design element.adc= Synthesis process will check code syntax and analyze the hierarchy of the design which ensures that the design is optimized for the design architecture, the designer has selected. The resulting netlist(s) is saved to an NGC (Native Generic Circuit) file (for Xilinx® Synthesis Technology (XST)). In this thesis the proposed design is implemented on FPGA. The design is verified by simulation and compared with the MATLAB simulation results. The design is successfully implemented in VIRTEX 6 (XC6VLX240T) FPGA and the estimated frequency of operation is 100MHz.

2.8 The Canny Edge Detection System Architecture

Fig-9shows the hardware architecture for the canny edge detection system.It consists of different processing blocks,input, output and intermediate memories and a controller to control the data transfer between different processing blocks and memories.

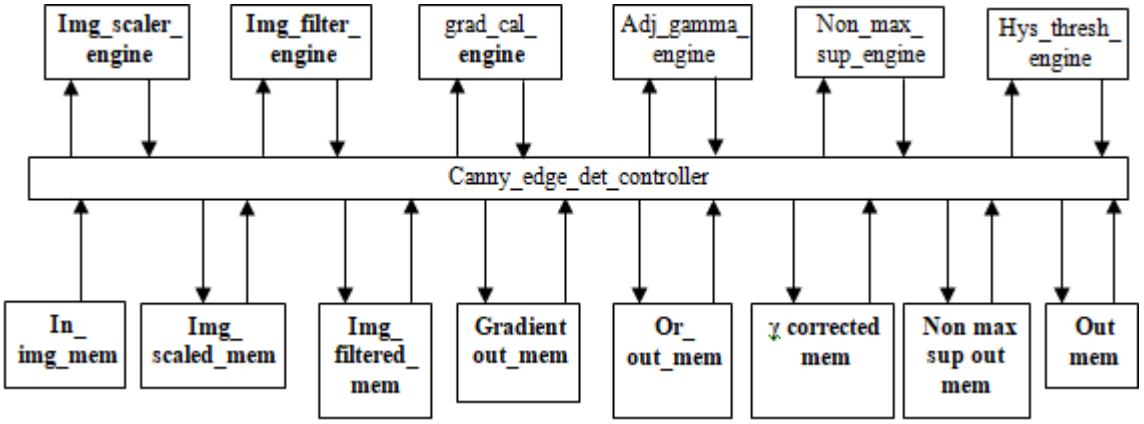

Figure 9: The system architecture of canny edge detector

3. Results and Discussions

3.1 MATLAB Simulation Results

a)To locate the iris boundary the edge map of original input eyeimage is created and output is obtained as in Fig 10.

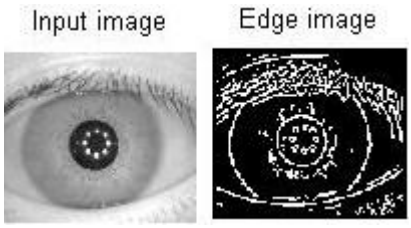

Figure 10: MATLAB simulation result showing complete edge map

3.2 Implementation result and discussion

Here ISim Simulator tool was used in order to simulate the design and check the functionality of the design. Once the functional verification is done, the design was synthesized from the XILINX ISE Design Suite 14.3 tool. The appropriate tests cases have been identified in order to test this modeled architecture. Based on identified values, the simulation results which describe the operation of process has been achieved. This proves that the modeled design work's properly as per its functionality.

3.3 Simulation result

The test bench is developed in order to test the modeled design. This developed test bench will automatically force the inputs and will make the operations of algorithm to perform. Simulation results are shown in Fig-11.

3.3.1 Behavioral simulation (RTL simulation) : This is first of all simulation steps those are encountered throughout the hierarchy of the design flow. This simulation is performed before synthesis process to verify RTL (behavioral) code and to confirm that design is functioning as intended. Behavioral simulation can be performed on Verilog designs. In this process, signals and variables are observed, procedures and functions are traced and breakpoints are set. This is very fast simulation and so allows the designs to change the HDL code of the required functionality is not met within a short time period. Similar to the MATLAB model, given 84 X 96 scaled eye image is given as the input to the algorithm and tested and verified the algorithm for each block separately.

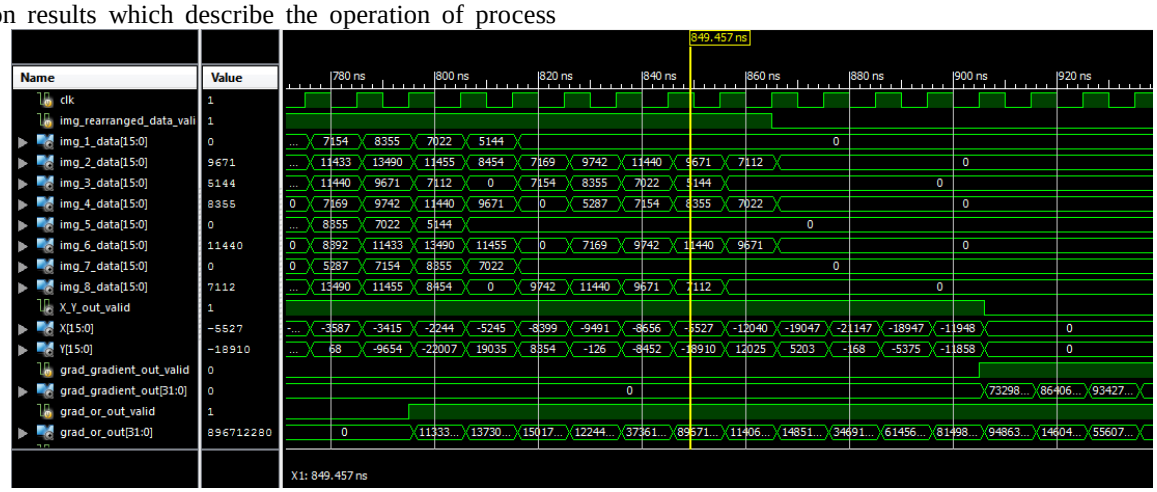

Figure 11: The simulation result

3.3.2 Synthesis results

The developed design was simulated and verified their functionality. Once the functional verification are done, the RTL model was taken to the synthesis process using the Xilinx ISE tool. In synthesis process, the RTL model will be converted to the gate level net list mapped to a specific technology library.The Gradient calculation block implemented in verilog and coding was done for ML605

Evaluation Kit. The ML605 Evaluation Kit is based on the XC6VLX240T-1FFG1156 Virtex-6 FPGA.The input to gradient block ie the original eye image is loaded initially into a block RAM using a .coe file. Since the original eye image size from CASIA database is 280 X 320, that means the image contain 89600 pixels. The gradient calculation on this original image will results in shortage of hardware resources. So the original image is scaled by a factor of 0.3 to get a smaller image of size 84 X 96, ie total 8064 pixels. Then using an image rearrange block, the image is rearranged to 8 different memories in 8 different ways. For that 8 synchronous dual port RAMs are used, in which through one port data will be written into RAM and through the other port data is read from memory.

Then from these eight rearranged images, the XY calculation block subtract one image from the other in a specific way to get horizontal(h), vertical(v), diagonal(d1) and off-diagonal(d2) components of gradient. Then the X and Y components of gradient is calculated from the previously described equation. Then the gradient calculation block calculate the gradient and orientation of the gradient at each pixel. For gradient calculation to calculate square root the CORDIC IP core is utilized. For orientation calculation block to evaluate tangent CORDIC IP core is utilized.

The inputs and outputs viewed through ChipScope Pro Analyzer is as shown in Fig 12.

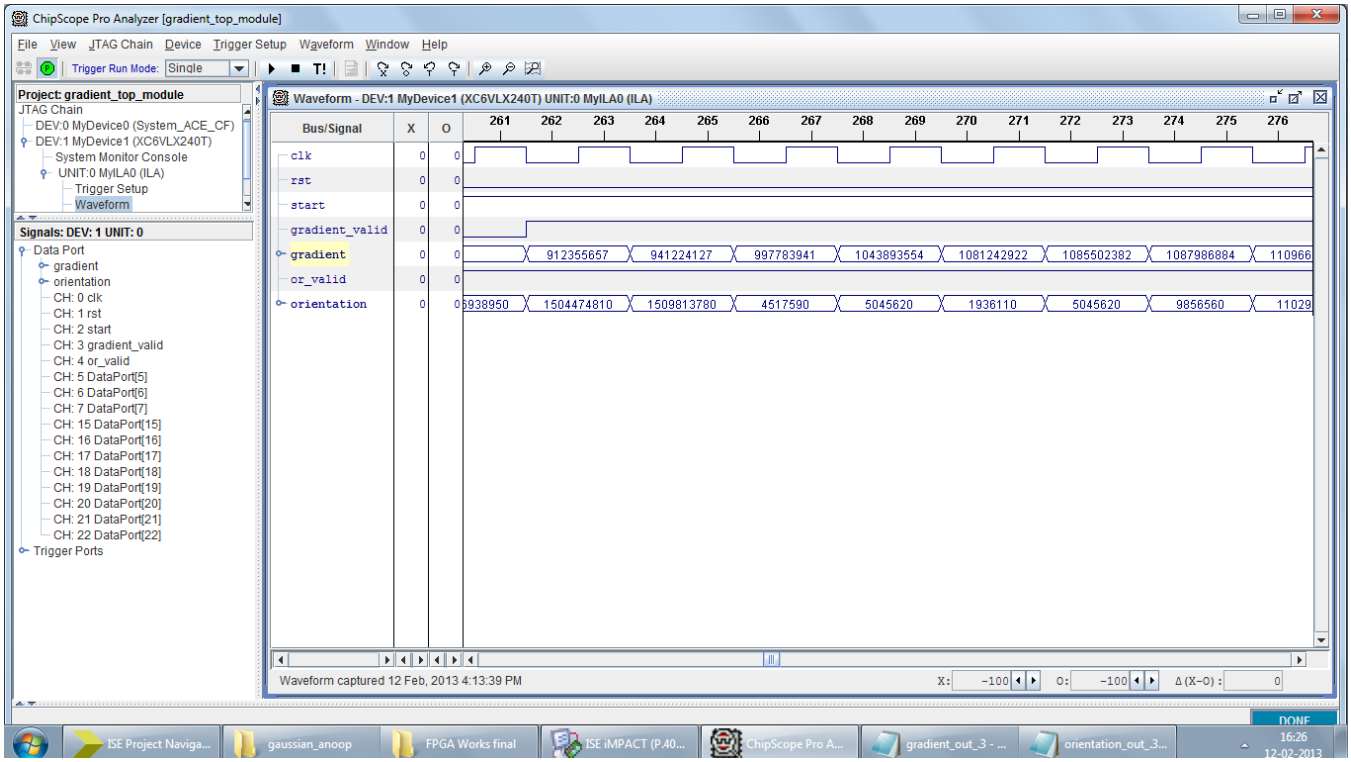

Figure 12: Chipscope output on ML 605 Evaluation kit

The input eye image and the gradient output displayed on the VGA screen is as in Fig 13.

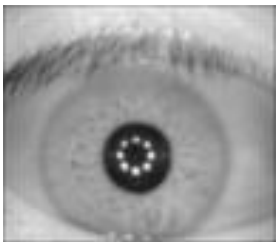

Figure 13 (a): Input eye image

Figure 13 (b): output displayed on VGA display

4. Conclusions

The canny edge detection subsystem with slight modification in both gradient mask and non-maximal suppression which is as a pre-processing part for iris detection system, is implemented in MATLAB for output verification. This edge map is used to detect inner and outer iris boundary and eyelid boundary in iris if any. The next step is VLSI implementation. The programming is done in Verilog and the functionalities are verified using XILINX ISE tools. This project has an efficient VLSI architecture design and implementation on Xilinx FPGA. Here the system performance in terms of speed and hardware utilization FPGA system is very efficient.

The hardware architecture of gradient processor using 84 X 96 pixels block of 8-bit image information as input has been presented. The entire system was simulated using MATLAB for a84X96 eye image. Apart from the MATLAB software model, in order to compare the performance of the system the algorithm was developed using Verilog and simulated on

ISim simulation tool and analyzed the performance. Once the functional verification sare done, all modules in the architecture were synthesized by using Xilinx ISE design tools.

Advancements in FPGA, flash memory, and capacitor technologies have enabled a lower power, nonvolatile memory backup solution that supports a battery-free environment and the benefits. The biometric security recognition system is profoundly scalable, secure, rapid, assessable and accurate. This module of image processing proposes an efficient section for any biometric device utilized for a real-time identification process with higher performance and greater accuracy.

References

[1] E. Wolff. Anatomy of the Eye and Orbit. 7th edition. H. K. Lewis & Co. LTD, 1976
[2] J. F. Canny, "Finding edges and lines in images," M.S. thesis, Mass.Inst. Technol.; AI Lab. TR-720, 1983.
[3] John Canny. “A computational approach to edge detection”. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-8(6):679–698, Nov. 1986.
[4] L. Masek, "Recognition of Human Iris Patterns For Biometric identification", Thesis Report, The University of Western Australia, 2003.
[5] P. C. Kronfeld, “The gross anatomy and embryology of the eye,” in The Eye, vol. 1, H. Davson, Ed. London: Academic,1968, pp. 1–66.
[6] F. H. Adler, “Physiology of the Eye”. St. Louis, MO: Mosby, 1965.
[7] D. Ziou, S. Tabbone, “Edge Detection Techniques – An Overview”
[8] D. Marr and E. C. Hildreth. Theory of edge detection. Proceedings of the Royal Society, London B, 207:187-217, 1980
[9] Jean Ponce, “Lecture 26: Edge Detection II”, 12/2/2004.
[10] S. Price, "Edges: The Canny Edge Detector", July 4, 1996.
[11] Margaret M Fleck “Some defects in finite difference edge finders” IEEE PAMI No. 3, Vol. 14. March 1992. pp 337-345
[12] Chinese Academy of Science - Institute of Automation (CASIA).Database of the Eye Grayscale Images.http:llwww.sinobiomtrics.com

.