

## **SECURE BIOMETRICS AUTHORIZATION USING FPGA TECHNOLOGY**

**Dr VENKATESH BABU S<sup>1</sup>, RAHUL R<sup>2</sup>, SARMA SANDEEP JAYAKRISHNAN<sup>3</sup>,  
SREEJITH H<sup>4</sup>**

Associate Professor<sup>1</sup>, Assistant Professor<sup>2,3,4</sup>

Department Of Electronics and Communication Engineering,  
Indira Gandhi Institute of Engineering And Technology Nellikuzhi P.O, Kothamangalam,  
Ernakulam (Dist) Pincode 686691

### **Abstract**

Using a Field Programmable Gate Array (FPGA), the purpose of this study effort is to process the iris biometrics for the purpose of iris recognition, which is used for a secure authentication system. When it comes to identifying a person, the iris recognition system is the most trustworthy technology. However, it is a complicated procedure that requires a significant amount of computing costs. Patterns on the iris are characterised by a great degree of unpredictability and individuality. Utilising a technique known as edge detection in conjunction with several edge detection algorithms, such as Sobel, Prewitt, and Canny edge detection algorithms, our method is based on the segmentation of the iris of the eye. In this work, the iris image is preprocessed by utilising a MATLAB Simulink block configured to handle the edge detection operation. The edge detection block locates the edges in an input picture by estimating the gradient magnitude of the image. This is done in order to handle the edge detection operation. It is then subjected to post-processing in order to provide a normalised output picture. Because the processing of images in real time takes a substantial amount of time, we have decided to do it at the hardware level using FPGA, which provides parallelism and considerably reduces the amount of time required for processing these images. Additionally, the Spartan3E FPGA kit is used to implement the various edge detection methods that were discussed before. In the design that has been provided, the architecture of various edge detection methods is implemented by means of XSG, which stands for Xilinx System Generator Tool. The Xilinx ISE 10.2i development suite is used in order to synthesise the net lists that have been created. Through the use of hardware software co-simulation from XSG, the FPGA implementation is carried out. This allows for the utilisation of limited resources at an increased maximum operating frequency, along with a shorter processing time and a greater level of security compared to commercial systems.

**Keywords:** FPGA, Iris, Edge Detection, Thresholding, System Generator, Spartan3E

### **1. Introduction**

In today technologically dependent world, it is an imperative need to identify users of facilities and services. today is not only important for determining who accesses a system and/or service, but it is also important for determining the sorts of services that should be supplied to each individual user. In order to accomplish this identification, biometric is a relatively new technology that offers a better degree of security for access, in addition to being easy and pleasant for the one who is doing the identification. Currently, biometrics is being used in a wide variety of contexts, with iris biometrics recognition methods being among the most prevalent uses of this technology. This article provides a comprehensive discussion of the iris recognition

process that makes use of XSG and the processing of iris biometrics for the secure authentication system that makes use of FPGA. Both of these processes include image processing methods. The approach that we have presented manipulates the current picture by using an appropriate edge detection algorithm in order to highlight the features in the way that is wanted. In every picture, there are two-dimensional arrays of pixel values that make up the image.

In the field of image processing, the edge detection operation is a fundamental operation that is beneficial in the fields of feature identification and feature extraction processes. The limits of forms in an image are referred to as the edges, and each edge is defined as a single pixel that has a local discontinuity in its intensity. The method of determining the instantaneous change in the intensity of a picture by using a particular algorithm is being referred to as edge detection. There are a number of reasons that are responsible for these local discontinuities in pictures, which include but are not limited to light, shadows, and illumination. The ridge, step, ramp, and roof edges are the four distinct forms of edge creation that were generated as a result of these discontinuities on the surface. Edge detection draws attention to a discontinuity in the direction of the pixel intensity that is either abrupt or gradual. This results in a huge reduction in the quantity of data that has to be processed, and information that is not relevant is being filtered away. For the purpose of edge detection, there are a few different kinds of operators that may be used, and they are based on first and second order derivatives. The first order derivative process involves convolving the input picture with an appropriate mask in order to produce a gradient image. The most often used operators, such as Prewitt and Sobel, are the first order derivative operators, which are sometimes referred to as gradient operators. These operators locate edges by searching for the highest and lowest intensity values. After smoothing the picture to eliminate noise, the Canny operator is applied to the second order derivative. This operator is used after the smoothing process. With the integrated use of tools like MATLAB, Simulink, and XSG, it is possible to complete the work of image processing using FPGA to get the desired results. The model-based design environment known as Simulink is a component of MATLAB that is integrated with it.

In order to make use of the Xilinx system generator (XSG), which is one of the block libraries that Simulink provides, it is essential to translate the System Generator token into MATLAB. Simulink also provides access to a variety of block libraries. All of the Xilinx block sets are stacked up and imported into the MATLAB and Simulink environment. Once they are imported, they are instantly available for use in the building of algorithms. Our research and development activities in the area of iris identification are primarily focused on the application of edge detection using a variety of operators and thresholding to edge detected pictures using Xilinx Block set. This is the core focus of our work related to iris identification. In order to mimic these approaches, the Xilinx system generator tool is used, and the related output image is analysed by using the video processing block that is established inside the XSG library tool box. Immediately after the gathering of the results, the System Generator is configured for the FPGA board that is pertinent to the situation. In this particular instance, the FPGA board that is being used is the Spartan XC3S500E-FG320 board. After the compilation procedure was finished, a programming file written in Verilog was produced after it was finished. It is possible to get access to this file by using the Integrated Development Tool (IDE) environment that is made available by Xilinx ISE10.2i. Synthesising the model with the necessary amount of simulation time enables the requisite HDL net list to be generated. This is made possible by the present circumstance. The code that was obtained is first subjected to a syntax check, after which it is synthesised in order to display the RTL Schematic of the design, and ultimately, it is implemented on a field-programmable gate array (FPGA). We were able to successfully create an

architecture, and the Xilinx System Generator has the capability to generate a User Constraints File (UCF) for the purpose of testing the architecture.

In order to programme the FPGA, bit stream creation is considered required. The Xilinx System Generator is a digital signal processing (DSP) design tool that was developed by Xilinx. It is designed to facilitate the use of the Math Works model-based Simulink design environment, which makes it relatively simple to operate in comparison to other tools for FPGA design. Due to the presence of these characteristics, we have decided to use Xilinx System Generator (XSG) in our work. It is primarily concerned with the processing of an image from pixel to pixel, as well as the change of the results of the pixel neighbourhood, which ultimately leads to the alteration of the actual or partial picture. The Xilinx system generator tool, which is integrated with the ISE 10.2 development suite and MATLAB 2007a, serves as the architectural foundation for the suggested architecture for the hardware implementation approach. With this design, the goal is to achieve overall high performance while simultaneously reducing development time and costs.

## **2. Related Work**

Image enhancement techniques make advantage of the retinal fundus image of the human eye. These procedures are carried out using intensity-based approaches such as brightness control, contrast stretching, and histogram equalisation. Simulink is used to process the design, and the Spartan3E development board was used to carry out the implementation for the design [1]. The hardware approach for processing iris biometrics, which involves applying the Gabor transform on the iris in order to extract features such as the iris signature from the angular area of the iris, as well as by using zero crossing representation. The matching of iris is accomplished by having a pattern recognition issue and making use of a measure known as mathematical distance [2]. A discussion is held on the detection of edges using a non-uniform assembly co-efficient for grayscale images, as well as the utilisation of multi-scalar structure elements to achieve equal merging weights for RGB images. In this case, the edges are created by using a variety of operators, such as the Sobel, Canny, and log operators [3]. Robustness in non-cooperative situations may be achieved by the use of the approach that is based on wild togetherness with clustering process. The original picture is employed to build the basic template for the correlation process, and the parameters that are used in this process include iris segmentation, filtering and edge extraction, and thresholding. For the purpose of localising the centre circle of the iris, the hysteresis and Hough transform is used [4].

The improvement of brain pictures via the use of XSG is suggested in this article. The images are then introduced into an artificial neural network with the purpose of determining whether the brain is normal or disordered. The discrete wavelet transform is the procedure that is used in order to extract energy values from an improved picture by making use of filters for the purpose of providing proper hardware implementation [5]. The creation of an image segmentation framework that is based on the colour hue channel histogram of the HSV colour space is presented, along with the design of traditional edge detection algorithms like Sobel, Prewitt, and Robert, and it is implemented with the help of the Spartan3E kit [6, 7]. Before doing a comparison between the various edge detection operators, it is necessary to evaluate the peak signal to noise ratio (PSNR) and the mean squared error (MSE) for the edge detected picture that was produced. The fact that this is the case demonstrates the capacity of achieving accurate edge detection [8]. An

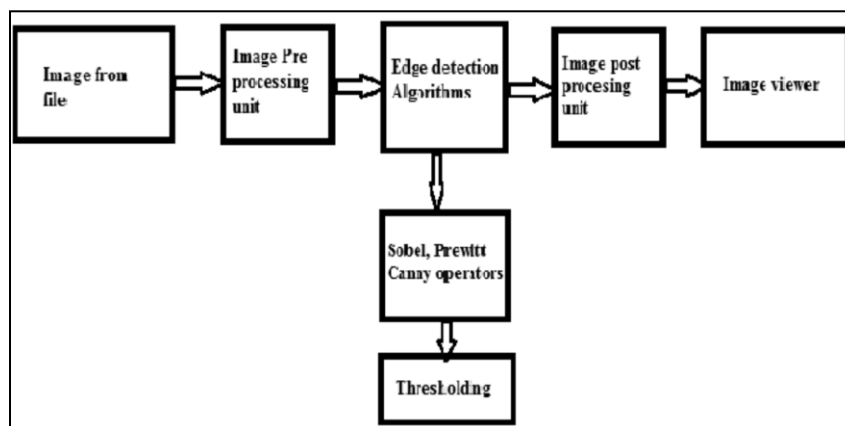
implementation of the Canny edge detection technique is carried out using a Virtex5 Field Programmable Gate Array (FPGA) board. via the use of Xilinx platform studio and Xilinx ISE, the output picture is presented on a Video Graphics Array (VGA) monitor, which is connected to the board via the utilisation of a DVI terminal [9]. The strategies for image recognition are built with the help of system Generator, and the method that is used here for detection is the Hough transform. This method is a highly effective method for locating parametric curves in an image, particularly lines. It has been shown that this idea can be effectively implemented in the Virtex 5 ML507 device (XC5VSXT platform) [10]. A pair of 3X3 convolution masks is used by the Sobel edge detector [11], which estimates the gradient in the x-direction and another gradient in the y-direction. This allows for the conversion of a two-dimensional pixel array into statistically uncorrelated data, which is necessary for the elimination of redundant data. Also, it illustrates how to construct a Sobel detector function in MATLAB that has a size of  $5 \times 5$  in order to be able to locate edges.

### 3. Proposed Design

Edge detection techniques that have been suggested include a number of transformations, including the following:

- A collection of blocks for image preprocessing
- Image post processing block set
- Edge detection methods that make use of XSG

The various edge detector operators, such as the Sobel, Prewitt, and Canny operators, which are used in the process of detecting the iris picture, are implemented using FPGA. Edge detection is a technique that aims to construct a line drawing of a scene from an image and to extract significant elements from the edges of an image, such as corners, lines, and curves. These characteristics may be used by higher-level computer vision algorithms, such as recognition and other similar algorithms. Both the pre-processing and post-processing of images are processes that are used by all image processing programmes that are developed using the Simulink block set. Figure 1 illustrates the real implementation of several edge detection techniques by making use of XSG.



**Figure 1 Design Flow of Edge Detection Implementation in XSG**

The Spartan3E development board is used in order to carry out the hardware implementation of edge detection techniques. XSG, which stands for Xilinx system generator, is responsible for compiling all of the essential files.

**MATLAB's Image Pre Processing Unit** is a useful tool for delivering input to the Field Programmable Gate Array (FPGA). This supplies the essential test vector array that may be used for FPGA bit stream compilation via the utilisation of system generator. Images that are submitted as input to the File block are either colour or grayscale, depending on the user's preference. The RGB picture is changed to grayscale using a colour space conversion block, and the data that is now in two dimensions has to be converted to one dimension so that it may be processed further. The frame conversion block is responsible for converting the output signal into frame-based data and then supplying it to the un-buffer block. Within this block, the frame is converted into scalar samples produced at a higher sampling rate.

**Image Post Processing Unit:** The image post-processing blocks are used in order to transform the output of the picture back to the floating point type. It does this by using a Buffer block, which is responsible for converting scalar samples to frame output at a reduced sampling rate. Additionally, it generates an output with a different frame size by redistributing the data into each column of the input. Additionally, it travels through a signal block that converts the 1D format to the 2D (matrix) format, and a sub matrix block that extracts a continuous sub matrix from the M-by-N matrix. A sink is then used in order to show the output picture back in the monitor, and the Simulink block sets are utilised for this purpose.

#### 4. Methodology

**Edge Detection:** Edge detection is a key function that is used in the applications of computer vision and image processing. Edge detection is implemented with the primary purpose of locating and identifying abrupt discontinuities within the picture that is being provided. The sudden shift in pixel intensity that characterises the borders of objects and also helps to discover boundaries between distinct areas in the picture is the cause of these discontinuities. These boundaries are used for the purpose of identifying objects for the purposes of segmentation and matching. The technology that recognises edges not only has the ability to identify the grayscale value of an image but also to pinpoint its precise position.

**Calculating the gradient of picture intensity** at each pixel inside the image is another way that the Sobel Operator does its function. In addition to this, it determines the direction in which the intensity increases the most from light to dark as well as the pace of change in that direction. The outcome demonstrates the changes that occur in the picture at each pixel, and thus, the pixel representation of an edge comes into play. By convolving the picture with a pair of 3x3 filters, the Sobel operator is able to determine the estimated gradient of each pixel in the image. The magnitude of the gradient is just the sum of the two gradients that are estimated by these filters. The gradients are estimated in both the horizontal (x) and vertical (y) directions.

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

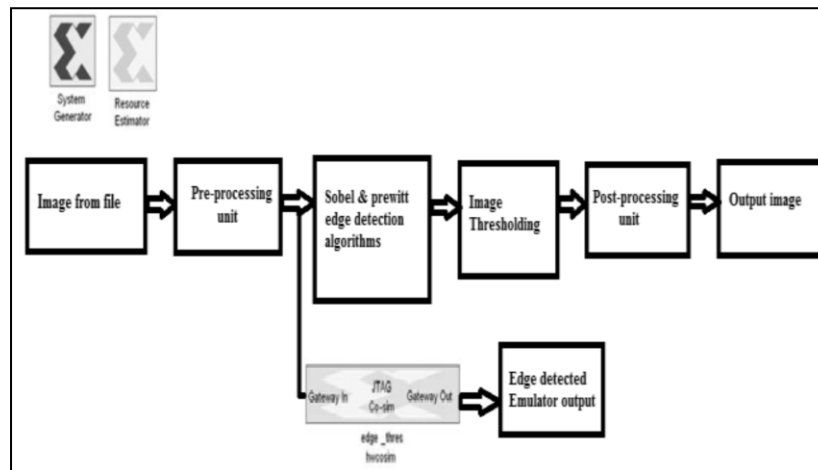
**Figure 2 Convolution Mask of Sobel Edge Detector**

**Prewitt operator** - It is a differentiation operator that computes an estimate of the gradient of the image intensity function. The Prewitt operator is also known as the Prewitt differentiation operator. The Prewitt operator is based on the process of convolving the picture in both the horizontal and vertical directions using a d-filter that is tiny, differentiable, and has an integer value. Similar to other operators, it is quite cheap in terms of the calculations that it does. Image detection is accomplished by the use of the edge model, with the output of the operator being the greatest value of the model operator that is most comparable to the area that has been identified. The differential and filtering procedures are implemented in the same way by both the Prewitt operator and the Sobel operator.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & -1 & 1 \\ 0 & 0 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

**Figure 3 Convolution Mask of Prewitt Edge Detector**

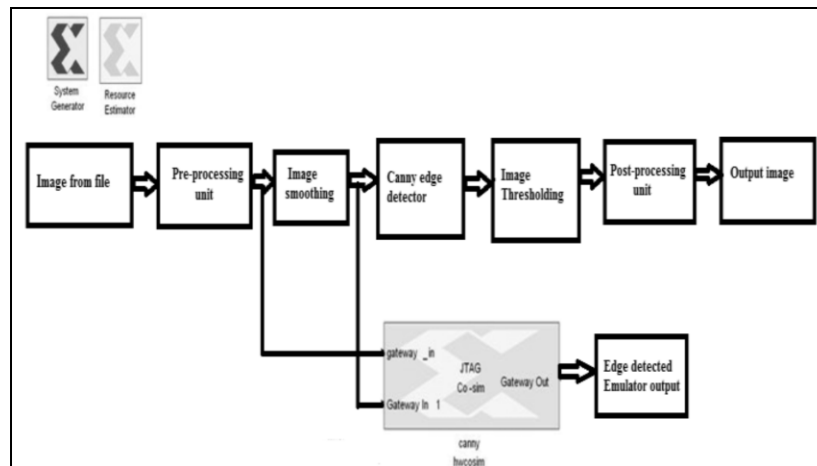
The function of the Prewitt edge detection operator is virtually identical to that of the Sobel edge detection operator; however, the kernels used by Prewitt are different. In order to identify edges in eight different directions, the Prewitt edge detection operators contain a pair of 3x3 convolution kernels. An illustration of the XSG model block diagram of the edge detection algorithms developed by Sobel and Prewitt is as follows:



**Figure 4 XSG Model Design of Edge Detection**

**Canny edge detector** an edge recognition operator known as the Canny edge detector employs a multi-stage algorithm to recognise a broad variety of edges in pictures. This operator is also known as the Canny edge detector. Find out the best possible edge is the primary emphasis of our endeavour. This approach represents the true edges in the picture to the greatest extent feasible, which is what is meant by the term "optimal edge detector." Additionally, it offers excellent localization, which means that the edges that are marked should be as near as possible to the edges that are present in the actual picture. A number of

processing steps are required in order to put the canny edge detector method into action. The first thing that has to be done before performing edge detection is to get rid of any noise components that are present in a picture. Image enhancement is essentially the act of increasing the interpretability or perception of information contained within pictures for human viewers, as well as providing improved input for other automated image processing methods. In addition, image filtering is accomplished via the use of model-based design, which allows for the determination of various filtering architectures and the creation of Xilinx blocks. To illustrate the XSG model block diagram of the canny edge detection technique, the following is shown:



**Figure 5 XSG Model Design of Canny Edge Detection**

## **5. Hardware Software Co-Simulation and Results**

In order to ensure that the design is implemented in the Spartan3E development kit, the hardware board is installed with the help of the XSG (Xilinx System Generator) Board Description Builder. The clock is provided with a frequency of 100 MHz, and it is configured to operate as a single-step clock for the hardware while the software simulation is engaged in lock step. In order to produce the code automatically, one must first click the generate button inside the system generator block dialogue box. For the design, the code generator generates an FPGA configuration bit stream file that is appropriate for hardware co-simulation. Additionally, the code generator includes extra interfacing logic that allows for communication with the design via the use of a physical interface that operates between the FPGA platform and the personal computer. The creation of a new JTAG Co-simulation block occurs after the compilation of the design is finished. All of the Xilinx blocks that are included inside the gateway in and out blocks of the initial system are included in this block collection. Converting the Simulink input type, such as integer, single, or double format, to the Xilinx data type is accomplished by the use of the Gateway in and Gateway out blocks. Additionally, the output signal that is generated by the FPGA hardware is shown via the Co-simulation process. In order to synthesise the design, the XST tool was used as the synthesis tool. Within the context of this model, the hardware description language is set to verilog. The picture that is shown here is of a size of 250 by 250 pixels, and it serves as an input to the design for edge detection and thresholding operations. The results that were produced by using effective

algorithms such as Sobel and Prewitt are displayed below, and the outputs of the associated FPGA and hardware are also stated here.

The outputs of the Sobel edge detection scheme are shown in Figure 6, and the output of the Prewitt edge detection scheme employing XSG is displayed in Figure 7. In addition, the input picture that is sent to the canny algorithms is an iris image with dimensions of 250 by 250 pixels. The optimised edge detection output is achieved for both the XSG model-based design and the hardware software co-simulation using the Spartan3E development board. Figure 8 displays the findings that correspond to these two methods. The HDL files that are created are synthesised with the help of Xilinx. This information will be provided by the synthesis of the code, which will include the errors, warnings, RTL schematic of that code, device utilisation summary, and other relevant details. Through the process of synthesis, an abstract form of intended circuit behaviour, also known as register transfer level (RTL), is transformed into a design implementation that is expressed in terms of logic gates. The Xilinx Synthesis Technology (XST) tool, which is a component of the Xilinx ISE 10.2i software, has been responsible for the synthesis of verilog code for the Sobel and Prewitt algorithms.



**Figure 6a) Original image    b) Simulink software output    c) Emulator FPGA output**

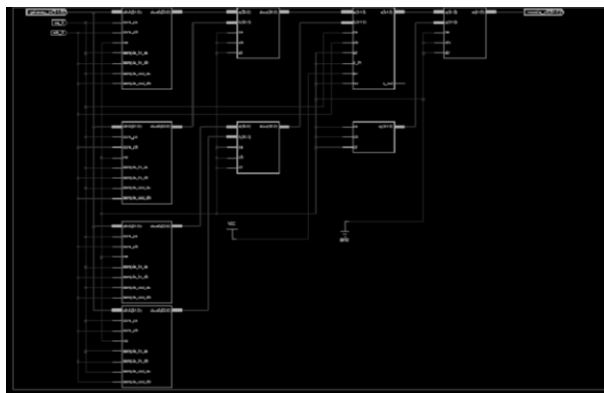


**Figure 7 a) Original image    b) Simulink software output    c) Emulator FPGA output**

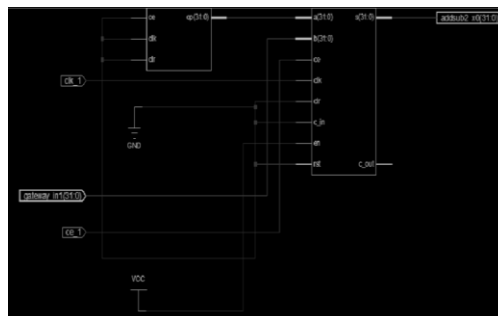


**Figure 8a) Original image    b)Simulink software output    c) Emulator FPGA output**

As can be seen in Figure 10, the top-level RTL schematic of the proposed canny architecture is shown here. This is a schematic depiction of the pre-optimized design that is displayed at the Register Transfer Level (RTL). Adders, multipliers, AND gates, and OR gates are some examples of the generic symbols that are used in the RTL Schematic representation of this architecture. This representation is created after the HDL synthesis step has been completed. The Spartan3E starting board is the target for the system blocks that have been developed. As part of the design summary report, the synthesised design also includes information on the utilisation of the target device for the suggested algorithms inside the design.



**Figure 9 RTL Schematic of Sobel & Prewitt Edge Detection Algorithm**



**Figure 10 RTL Schematic of Canny Edge Detector**

**Summary of Device Utilisation** The summary of device utilisation displays the information that is associated with the analysis of device utilisation. In the report that was prepared, the comprehensive report on the design that was being processed is shown. The outputs of the Slice Logic Distribution and Synthesis offer an estimate of how the design will be packed and placed into the target architecture. Additionally, the map provides information on utilisation after the packing and placement processes have been completed. It is stated that the device utilisation summary that was obtained for the Sobel, Prewitt, and canny architecture is as follows: with the comparison results of the edge detection algorithms that were discussed earlier, the canny edge detector operator possesses a greater set of results, which includes a 76% reduction in area and a 78% increase in its speed of operation in comparison to the Sobel and Prewitt operators.

**Table 1 Device Utilization Summary**

| Parameters                  | Sobel Operator | Prewitt Operator | Canny Operator |
|-----------------------------|----------------|------------------|----------------|
| Slices                      | 75             | 75               | 12             |
| LUT                         | 132            | 132              | 32             |
| IOB                         | 66             | 68               | 64             |
| Combinational path delay    | 9.595 ns       | 9.595 ns         | 5.363 ns       |
| Maximum Operating Frequency | 104.22 MHz     | 104.22 MHz       | 186.46 MHz     |

**Evaluation Parameter:** The performance assessment of edge detection algorithms is achieved by detecting actual edges, processing time, error ratio, noise level, and other parameters, among others. The Mean Squared Error (MSE) and the Peak Signal to Noise Ratio (PSNR) are the two metric standards that are commonly followed when it comes to noise concentration. In the process of image compression, the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR) are factors that are considered. However, in this case, it is used to evaluate the quality of picture edge detection. It is possible to do a comparison between edge detection operators and a common picture. The operator that has a strong capacity for edge detection is the one that produces an image with a lower PSNR and a higher MSE.

**MSE** stands for mean squared error, and it is a measure that takes into account the degradation function as well as the statistical properties of noise in the edge detected picture. The difference between the estimator and the parameter is measured by calculating the average squared difference. An increase in the MSE in terms of picture edge identification might be achieved by ensuring that there are a greater number of edge points on the image. Additionally, it is able to identify weak edge points. As a result, there is a more significant disparity between the original picture and the processed version.

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M,N}$$

**The Peak Signal-to-Noise Ratio**, also known as the PSNR, is often stated using the decibel (dB) scale. The power spectral noise ratio (PSNR) (PSNR) is an approximate evaluation of the quality of the reconstruction. If you want to get accurate results from edge detection, the PSNR should be lower. Calculated based on the MSE, the PSNR was determined by

$$\text{PSNR} = 10 \log_{10} \left( \frac{R^2}{\text{MSE}} \right)$$

The mean squared error (MSE) and peak signal-to-noise ratio (PSNR) for the edge detection algorithms Sobel, Prewitt, and Canny appear below.

**Table 2 Results of MSE & PSNR**

| Edge detector operators used | MSE   | PSNR  |
|------------------------------|-------|-------|
| Sobel                        | 39.52 | 32.20 |
| Prewitt                      | 37.87 | 32.38 |
| Canny                        | 42.48 | 31.79 |

According to estimates, the value of MSE for Canny is ten percent more than that of the Prewitt operator and six percent higher than that of the Sobel operator. In addition, the PSNR value that was acquired indicates that the Canny operator is 2% lower than the Prewitt operator and 0.5% lower than the Sobel operator.

## 6. Conclusion

Through the use of Xilinx System Generator, the Sobel, Prewitt, and Canny edge detection algorithms are devised and implemented on the Spartan3E FPGA kit for the purpose of iris identification. As a result of the fact that it is less difficult to produce a stream of bit files as opposed to writing thousands of code lines in order to execute image processing algorithms on FPGA, we have decided to use XSG for our design. As a consequence of the implementation findings that were acquired, the performance of various edge detector methods was reviewed, and it was determined that canny has a 76% decrease in area utilisation compared to Sobel and Prewitt operators, as well as a 78% rise in its maximum operating frequency. In addition to having a design that has been optimised, the Canny architecture is also less likely to produce noise. This resulted in the canny operator being superior to the other algorithms that were already in existence. This is due to the fact that it has the highest MSE and PSNR values, which are parameters that are used to evaluate the effectiveness of algorithms in terms of creating correct edges to define object boundaries.

## 7. References

- [1]. Galbally, J., Fierrez, J., Alonso-Fernandez, F. and Martinez-Diaz, M. Evaluation of direct attacks to fingerprint verification systems, *Journal of Telecommunication Systems*, 2010, 47(3-4), 243-254.
- [2]. Baldisserra, D., Franco, A., Maio, D. and Maltoni, D., Fake fingerprint detection by odor analysis, *International Conference on Biometric Authentication*, 2005, 3832, 265-272.
- [3]. Poh, N. et al. Benchmarking quality-dependent and cost-sensitive score-level multimodal biometric fusion algorithms, *IEEE Transactions on Information Forensics and Security*, 2009, 4(4), 849-866.
- [4]. Nagar, A., Nandakumar, K. and Jain, A. K. A hybrid biometric cryptosystem for securing fingerprint minutiae templates, *Pattern Recognition Letters*, 2010, 31, 733-741.
- [5]. Ross, A., Jain, A. and Reisman, J. A hybrid fingerprint matcher, *Pattern Recognition*, 2010, 36(7), 1661-1673.

- [6]. Jain, A. K., Feng, J. and Nandakumar, K., Fingerprint matching, IEEE Computer, 2010, 201, 36-44.
- [7]. Pan, S.B., Moon, D., Gil, Y., Ahn, D. and Chung, Y. An ultra-low memory fingerprint matching algorithm and its implementation on a 32-bit smart card, IEEE Transactions on Consumer Electronics, 2003, 49(2), 453-459.
- [8]. Karimian N, Guo Z, Tehranipoor F, Woodard D, Tehranipoor M, Forte D. Secure and reliable biometric access control for resource-constrained systems and IoT. arXiv preprint arXiv:1803.09710. 2018 Mar 26.
- [9]. Gupta, P., Ravi, S., Raghunathan, A. and Jha, N. K., Efficient fingerprint-based user authentication for embedded systems, Design Automation Conference, 2005, 244-247.
- [10]. Lindoso, A., Entrena, L., López-Ongil, C. and Liu, J., Correlation-based fingerprint matching using FPGAs, IEEE International Conference on Field-Programmable Technology, 2005, 87-94.
- [11]. Schaumont, P., Hwang, D. and Verbauwhede, I., Platform-based design for an embedded fingerprint authentication device, IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 2005, 24(12), 1929-1936.
- [12]. Rui Z, Yan Z. A survey on biometric authentication: Toward secure and privacy-preserving identification. IEEE access. 2018 Dec 27;7:5994-6009.
- [13]. Zhang J, Qu G. Recent attacks and defenses on FPGA-based systems. ACM Transactions on Reconfigurable Technology and Systems (TRETS). 2019 Aug 21;12(3):1-24.
- [14]. Danese, G., Giachero, M., Leporati, F., Matrone, G. and Nazzicari, N. An FPGA-based embedded system for fingerprint matching using phase-only correlation algorithm, IEEE Euromicro Conference on Digital System Design: Architectures, Methods and Tools, 2009, 672-679.
- [15]. Tan TN, Lee H. High-secure fingerprint authentication system using ring-LWE cryptography. IEEE Access. 2019 Feb 14;7:23379-87.