ISBN 978-93-84422-37-0
2015 International Conference on Advances in Software, Control and Mechanical Engineering (ICSCME'2015)
Antalya (Turkey) Sept. 7-8, 2015 pp. 1-8

Using an FPGA to Accelerate Iris Recognition

Safaa S Omran[1], Aqeel Al-Hilali[1]

[1] College of Elec. & Electronic engineering Techniques

Abstract: *:Iris recognition becomes one of the most accurate and secures biometric method used today. The execution time of the iris recognition algorithm on general purpose sequential system as central processing unit is too high, so it cannot work in the real time applications. In this paper, an enhancement for the iris recognition system was applied for each processing part to speed up the execution time and make the opportunity to work in real time applications.*

Two enhancements were made in this paper, the first one by using hardware implementation for all the iris recognition process which are: Segmentation, Normalization, Feature extraction, and Hamming distance using the FPGA. The second enhancement is by choosing a small part (quarter) from the iris region which contains sufficient features to make the recognition, hence reducing the processing time.

Keywords*: Iris recognition, Ridge Energy Direction, Hamming Distance, Segmentation, normalization, Circle Hough Transformer, Bresenham Circle Algorithm.*

1. Introduction

Iris recognition is one of the most accurate and high confidence for authentication methods that used today. The features inside the iris are unique from person to person, unchanged and cannot be manipulated with years therefore it was more accepted in our world for distinguished between users than an others biometric system. Recent years researchers were tried to develop a new algorithm for making the iris recognition system works in the real time applications. However making iris recognition in real time is quite a challenge especially iris recognition needs huge image processing and resources. Therefore, the researchers have tried to create iris recognition system with low-cost and works in real time applications. This was impossible to achieve in the past years with the sequential processor, but this become possible with advancement parallel processor like Field Programmable Gate Array (FPGA) technology. The goal of this research is to use high-performance FPGA technology to implement iris recognition in the parallel structure to get powerful and efficient for the iris recognition system.

2. Iris recognition system

Iris recognition system consist of five main stages: image captured, segmentation, normalization, features extraction, and matching. The first step of the iris recognition is acquisition image of eye with higher quality and clarity to avoid the process of removing noise from the captured image. This needs simple camera and stationary image of the users. Once the image acquisition various preprocessing steps will be performed on it. The prepressing includes segmentation (extracting the iris from the captured image), normalization (polar to rectangular conversion) and then template and mask generation by applying the RED [1] algorithm to the rectangular template. Then the template is matched with the database using hamming distance equation and the match identification is displayed. The flow of process is shown Figure 1. The CASIA V1 is used to capture the image.

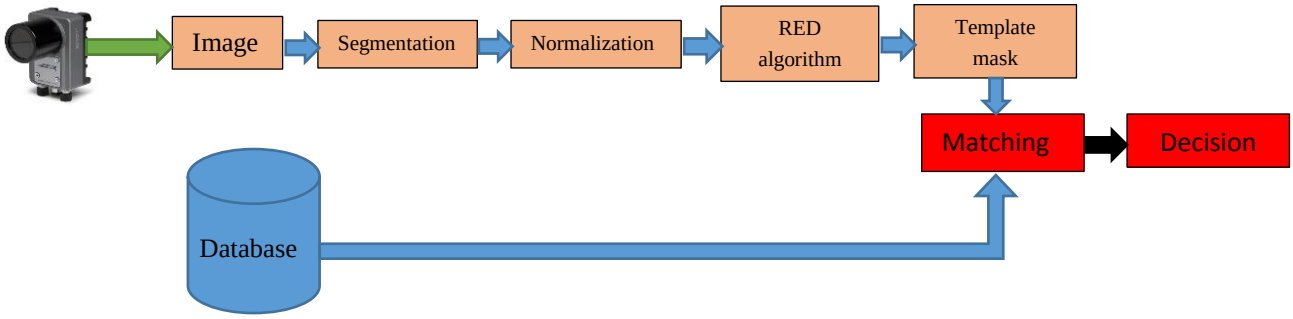


Fig.1: Iris recognition system

2. The Iris area

The numbers of pixels in the iris area are (90*480=43200) pixels. All researchers were applied their different methods for iris recognition on that iris area. Some researchers, used only the lower half part of the iris area in order to reduce the noise which comes from the eye lashes and eye brows, and to reduce the working time, since in this case half of the iris area will be taken [2] (90*240=21600). Other researchers used a small ring from the iris area (30*480=14400) pixels for iris recognition and a more reduction in the processing time they got [3].In previous paper we used only the half part of the ring from the iris area with size of (45*240=10800) pixels [4]. In this paper, all the hardware will be implemented only to the lower circular part of the iris area which is shown in fig.2.

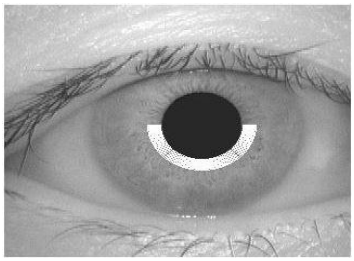


Fig. 1: Quarter iris region that used in implement process

3. Implementation of FPGA

Field Programmable Gate Arrays (FPGAs) are inherently parallel structures, have large number of registers and embedded memory blocks, and high-speed memory and storage interfaces have provided a suitable solution to facilitate a complete system-on-chip design. FPGA are reconfigurable after programmed with a specific design. FPGA allows the designer to create a design with parallel function and model, simulate and editing that design without costly of going to manufactured and adding a new circuit to the design every time change something in the design. VHDL (Very high Hardware Description Language) which is a common language that used to programmed FPGA. VHDL statements are essentially parallel, not sequential. VHDL permit to the programmer facilities to dictate the type of hardware that is integrated on an FPGA.

4. Segmentation

Segmentation is the process of extraction iris from the captured image. The segmentation process consist of two main tasks which they are edges detector to detect the iris and pupil boundary and Circle Hough Transformer (CHT) to find iris and pupil parameter that will mark their locations in the captured image [5]. The first stage of the segmentation is detection of edges of pupil and iris. Many algorithms have been used to detect edges. In this paper the canny edge algorithm was chosen over various edge algorithms to detect the iris and pupil edges. The canny edge algorithm was chosen over various edges algorithm to detect the iris and pupil boundary. The canny edge detection consist of five main process, Gaussian smoothing process, Sobel gradient calculation process, non-maximum suppression process, double threshold process and hysteresis process, as

shown in figure 3. [6] Gaussian smoothing process is filtering the captured image by mask to create an image with low noise. 5*5 Gaussian filters mask was chosen in our designed that will implemented in FPGA. Therefore, Gaussian filters needs 25 pixels involved from the captured image to calculate the value of single pixel in the captured image. Therefore, to calculate the value of one-pixel in captured image by applying Gaussian filter requires going 25 times to memory to load 25 data value that involved in finding the value of that pixel. Since it loads 25 data from the memory, so it's requires 25 clock cycles in sequential process only to find value of single pixel. This is huge time consumed if we have image with size 320*280 so it will required 2240000 clocks to perform the 25 Gaussian smoothing filter process on the captured image in sequential process. To parallel Gaussian-smoothing process in FPGA, this will done by design the memory in parallel structure that can load the entire involved pixel from the captured image only in single cycle. Therefore this designed faster the operation of performing Gaussian smoothing process 25 times than the sequential process. The next step in the design of canny edge is Sobel gradient calculation process. Sobel gradient calculation process is the operation of detection the direction and strength of possible edge pixel. Sobel gradient consist of two filter one estimate the gradient in x-axis and the other estimate the gradient in y-axis. Sobel gradient requires 9 pixel to involve to find possible edge pixel In each direction. This mean it is requires 18 pixel to involve to find possible edge pixel in each direction (x-axis and y-axis). Also same parallel memory structure designed and implemented in FPGA to detection the gradient direction only in single clocks. Both gradient (x-axis and y-axis) are calculated in parallel structure as well. The parallel memory in Sobel gradient, loads 9 data in single clock. The direction of the gradient is determined by using the fixed point arithmetic unit and by designed multiplication with shifts and addition/subtraction to increase speed. Then results are stored in a memory to be used as an input for the next stage. The third process of the canny edge is non-maximum suppression. Non-maximum suppression process is used to minimize the edge thickness to improve localization. The Non-Maximum Suppression also needs an 8-pixel to be involved to determine each pixel's value. Therefore, 16-pixel are determined simultaneously by using special design in FPGA. The four process in the canny edge is Double threshold process. This process is required only comparator non memory required. The process is executed by multiplies comparator. The data that result from the process consist of 2 bits for single pixel to represent three different values in the memory. The last process is hysteresis. Hysteresis process is comparing the pixels that results from Double threshold process based on two threshold T_{high} and T_{low} .

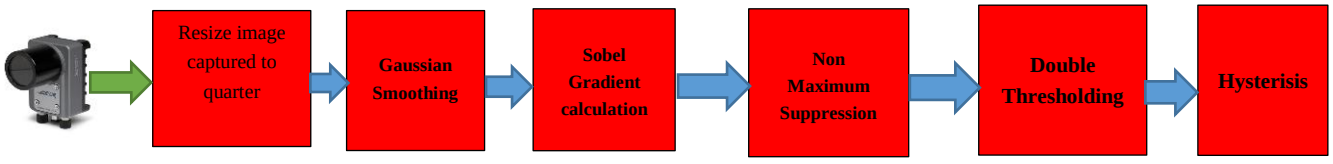


Fig. 2: Canny edge system

The next stage in the iris segmentation is parallelized circle search, the CHT is the method used to detect uncompleted circles. CHT is used to find circle parameter that defines the location of the iris and pupil in the image captured. CHT is then applied to select the best-fit circle to mark iris boundary location. CHT consist of two stages, the first stage is used to generate circle point, Bresenham Circle Algorithm (BCA) is used for that purpose and second stage is to find the best-fit circle that describes the iris and pupil boundaries via value of maximum accumulator [7].

BCA is used generate address of the circle point in the canny edge's image. For each edge pixel in the edge image, a circle is generated with a predetermined centered and radius at that edge pixel. Bresenham algorithm are generated the coordinates label (x_p,y_p) of the circle points. BCA consist of five variable, x, y, r, z and iz. Depending on these variable (x_p,y_p) of the next point will be determined. The q , z and iz are condition that controls the movement of x and y. depending on these conditions the x or y will be move one step in x-axis or in y-axis or in both axis's. 8 circle point will generate every time BCA generate single (x_p,y_p) point, due to the fact of circle's symmetry, the (x_p,y_p) of the other point will simple computed by simple process unit. Calculate one (x_p,y_p) coordinate, the other 7 coordinate will generate by relocating the x and y point and negative their values. After generating the address of 8 coordinate points, these points will be the input value to the parallel accumulator. The accumulator will parallel summation eight value of address that feed form BCA with the variable that contains the value of the previously summation (variable will initially zero every time new circle generating). Once the BCA generate all the address point of the circle, the accumulator will summation all value

in that circle and store the results in the register to be compared with accumulator of the next circle. The next circle point will generated by changing the value of either the x or y or both or changing the radii. Figure 4 illustrates the architecture of the BCA & CHT.

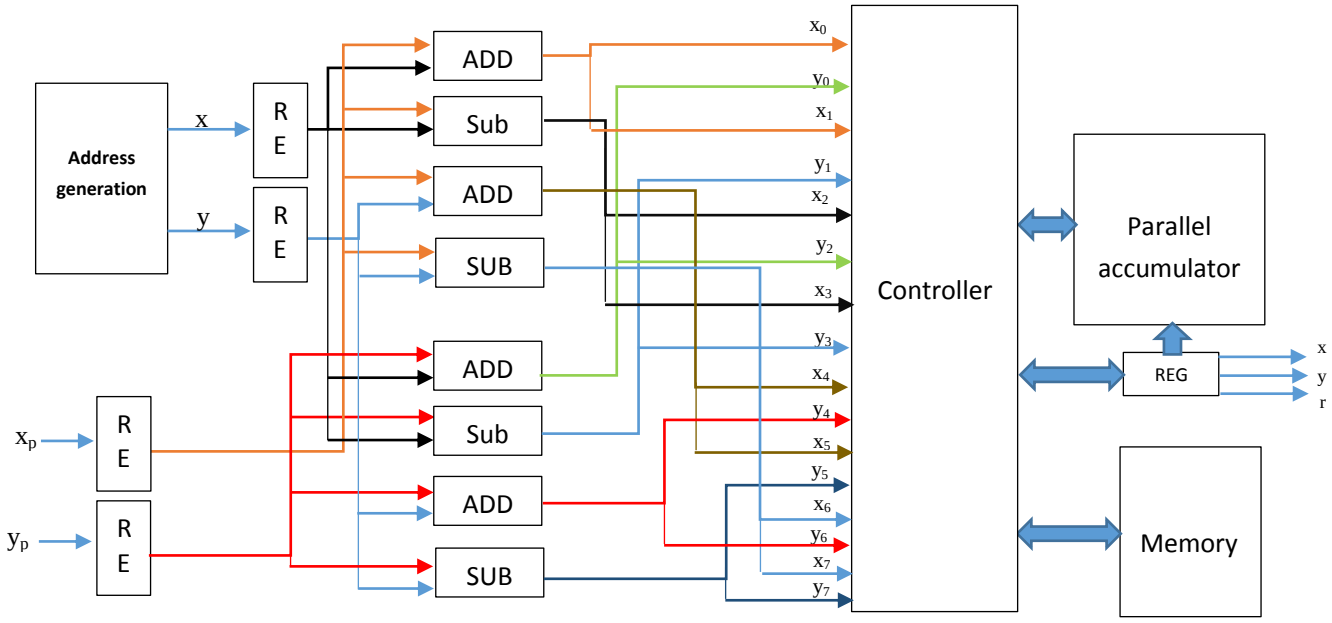
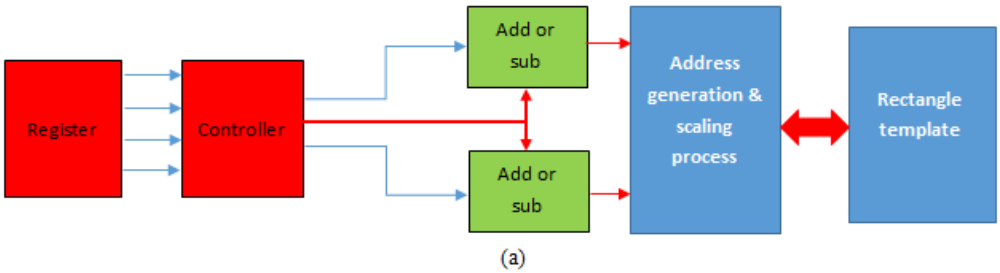


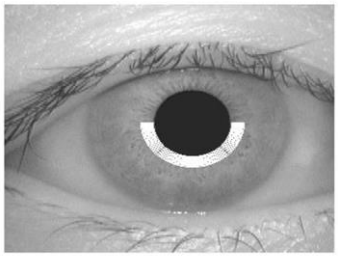
Fig. 3: Architecture of Circle Hough Transformer using BCA in FPGA

5. Normalization

The normalization process is a way to converting segmented iris from its polar coordinate form to rectangle coordinate form. This section will focus on designed and implemented polar to rectangle converting in FPGA. BCA was chosen to be designed and implemented in FPGA over the other algorithms. Once the segmentation process is completed. It's generating 4 parameters, which they are the center point of the iris and pupil and the radius of the iris and pupil. The result of segmentation will be inputted to the BCA. As stated previously, BCA needs three input which they the x and y point and r that will get these inputted from the segmentation process. The BCA consist of 4 main processes. These processes are the registers which store the parameters value that results from the segmentation process. The second process is the controller which control the address generation of (x_p,y_p) based on three conditions. These three conditions will select the address of the desired pixel in the captured image. The three conditions selected the address of the desired pixel by increasing or decreasing one step in x point or y point or both point. Once the three condition calculated the address of pixel is read from the captured image and copy to iris rectangle template. Once complete calculated the addresses of all pixels in first circle and stored in iris rectangle template, the next circle will generate by increasing the radius by one and same process will applied as the first circle. The radius will keep increasing until it reaches the radius value of iris boundary. the number of selected desired pixel will be different form circle to circle basing on value of the radius, the higher the value of the radius the higher the number of selected pixel in circle and vice versa. Therefore, BCA generate unequal rectangle template as shown in figure 5a.



(a)



(b)

(C)

(d)

Fig. 4 : a) Architecture of normalization using BCA. b) Quarter iris region. c) Rectangle template that generated using BCA before scaling process. d) Rectangular Iris generated after scaling process with height 45 pixel and width 240 pixel of the quarter iris region.

Scale process is use to equal the iris rectangle template that has been generated from BCA. BCA generate known number of pixel for each circle depending on the value of radius. This will help in scaling process since the number of pixel is constant for each radius this can yield constant number to scale for each circle. After the scaling process is completed the rectangle template is generated (fig.5c and d).

6. Feature extraction

The Ridge Energy Direction (RED) algorithm is used to extract features from the rectangle template. Usually, the features extraction is converting rectangle iris template to binary representation [1]. The features extraction of RED is based on the direction of the ridges that appear on rectangle iris template. The RED algorithm applies two directional filters on the rectangle iris template (vertical and horizontal). Applying these two directional filters on rectangle iris template will create two outputs for every pixel in rectangle template. Depending on the output value both filters will tell the appearance of a strong ridge and is encoded with a single bit to indicate the ridge direction in binary template. If the output value of the vertical filters is higher than the output value of horizontal filter then 1 is store in location of center filter else 0 will store in location. A rectangle binary template will generate after complete the RED algorithm contains the value that resulted from comparing of two directional filters of the RED algorithm. This rectangle binary template will be used in the matching process. Also, the rectangle binary template will masked with another template have the same size as rectangle binary template. The mask template contains 1 which indicates the appearance of the ridge and 0 absences. This step has already been built and tested with simulations showing a higher execution time, speedup efficiency achieved by an FPGA compared to sequential process.

7. Hamming Distance

The template matching is process of comparing the current template with template that has already been store in database until find one that matches in the database. Hamming distance (HD) is used to measure of how close the two template to each other. The more the HD closes to zero the more the close of two templates to each other. Highest closeness between matched templates is 0.32 as indicated by Daugman [8]

$$HD = \frac{\|(Template\ A \otimes Template\ B) \cap MaskA \cap Mask\ B\|}{\|Mask\ A \cap Mask\ B\|} \quad (1)$$

Where templates A is the Iris template captured image and Template B is the iris template from the database and $\otimes$ symbol indicates the binary exclusive-or operator to detect disagreement between the bits that represent the directions in the two templates, $\cap$ is the binary AND function, $\|\bullet\|$ is a summation, and mask A is associated binary mask for captured image template and also mask B is associated binary mask for database. The denominator ensures that only required valid bits are included in a calculation.

8. Results

The proposed algorithm of the iris recognition designed and implemented on FPGA and experimented on various iris database (CASIA V1 & CASIA Interval). The algorithm is experiment on two type of process, central processing unit (CPU) and parallel processing unit (PPU) and comparing the performance of the

algorithm between them. The CPU that has been used to test iris algorithm is Intel(R) Core(TM) i5. The processor is consist of two cores with 4 logic processor, 2.60GHz clock and 3230 MB cache. The full iris recognition algorithm was performed under Windows 7 using the MATLAB 2013a software. While the PPU that used to test the iris algorithm, was execute on Spartan 3AN boards. The Spartan 3AN board includes a XC3S700AN FPGA chip with 50 MHz clock. The full iris recognition algorithm is designed and implemented on the Spartan 3AN. The full algorithm is programmed on FPGA using VHDL. The Xilinx ISE suite 14.1 was used to implemented our VHDL program. The Xilinx ISE suite 14.1, involved synthesis, simulation, and programming environments. The execution time results and overall performance of iris recognition between the CPU and FPGA shown in table 1. Xilinx ISE suit 14.1 for implementation of our VHDL program. The ISE suit 14.1 includes synthesis, simulation, and programming environments. The results of the execution time and overall performance of iris recognition between the CPU an FPGA is shown in table 1. Table (1) shows the execution time results of the iris recognition on two processor type (CPU and FPGA). It is clear in the table (1) that the segmentation is most consumer time in CPU since segmentation was used the CHT, which is a technique that consumed time to locate pupil and iris boundaries since CHT is brute force tries many pixel to locate the iris in the captured image. The table (1) illustrates the acceleration performance and time excitation that achieved in FPGA compared to the CPU. The results show that FPGA is much faster than the CPU. For example, the optimized MATLAB version 2013a takes 6.51977 sec as average to complete segment iris while the FPGA takes 3.890 ms as average for segment iris. The main result of this research is to speed up of iris recognition and implemented on a modest sized FPGA and gets higher speed results. In this paper we implemented two type of iris recognition (segmentation and normalization) in FPGA and the other from iris recognition (features extraction and matching) was implemented in different paper. The results in the table (1) show that approximately 1676 and 1463 faster than the CPU process in performing the segmentation and normalization. The architecture designed in fig (6, 7) in FPGA has major effect speed up the overall processes of iris. the architecture in fig(6) shows that it generated 8 address location, loads these address ,perform parallel accumulated and finally comparing with registers to get the results only one clock cycle while it needs many clock cycle to perform these sequential processor. The other architecture in fig (7) shows how parallelized normalization can loads the desired pixel form the captured image and stored in rectangle template only in one clock cycle and then perfuming the scaling function to rectangle template to get fix size rectangle template. the parallelized architecture of system in FPGA improved execution of normalization compared in sequential processor as shown in table(1).

TABLE II: The execution time of iris recognition on CPU and FPGA.

Iris recognition system	Optimized Matlab code on Intel ® CPU (ns)	Spartan 3 AN XC3S700 (50 MHz)
Segmentation	6.51977 sec	3.890 m sec
Normalization	360.105 ms	246 micro sec
RED Algorithm (Digital filter)	64.011 ms	144 microsec
Hamming distance	7.7 ms	20 ns

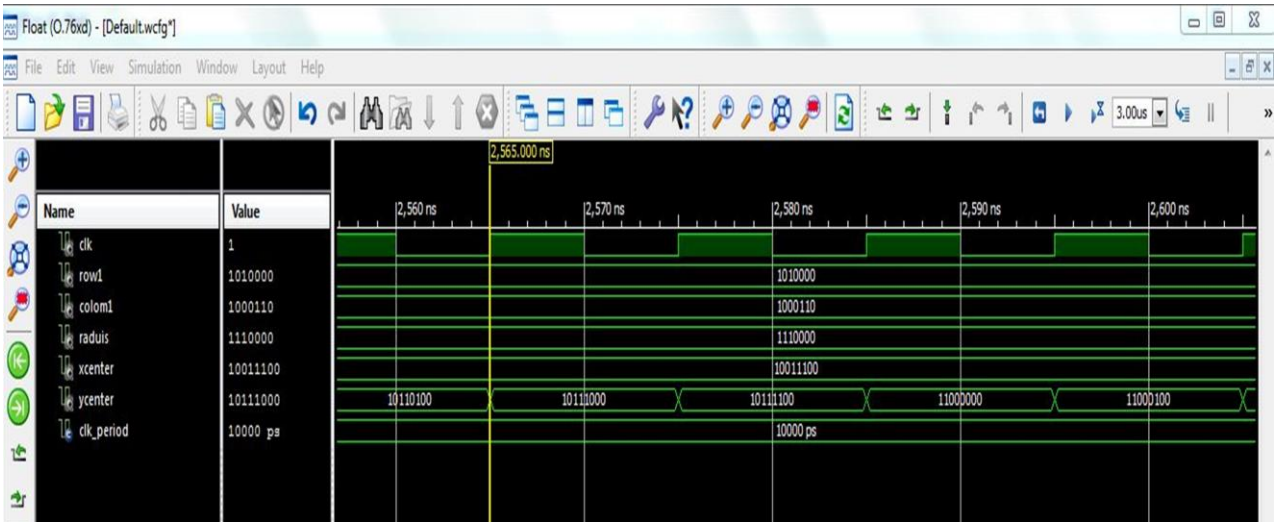


Fig. 5: Simulation of segmentation using CHT on FPGA

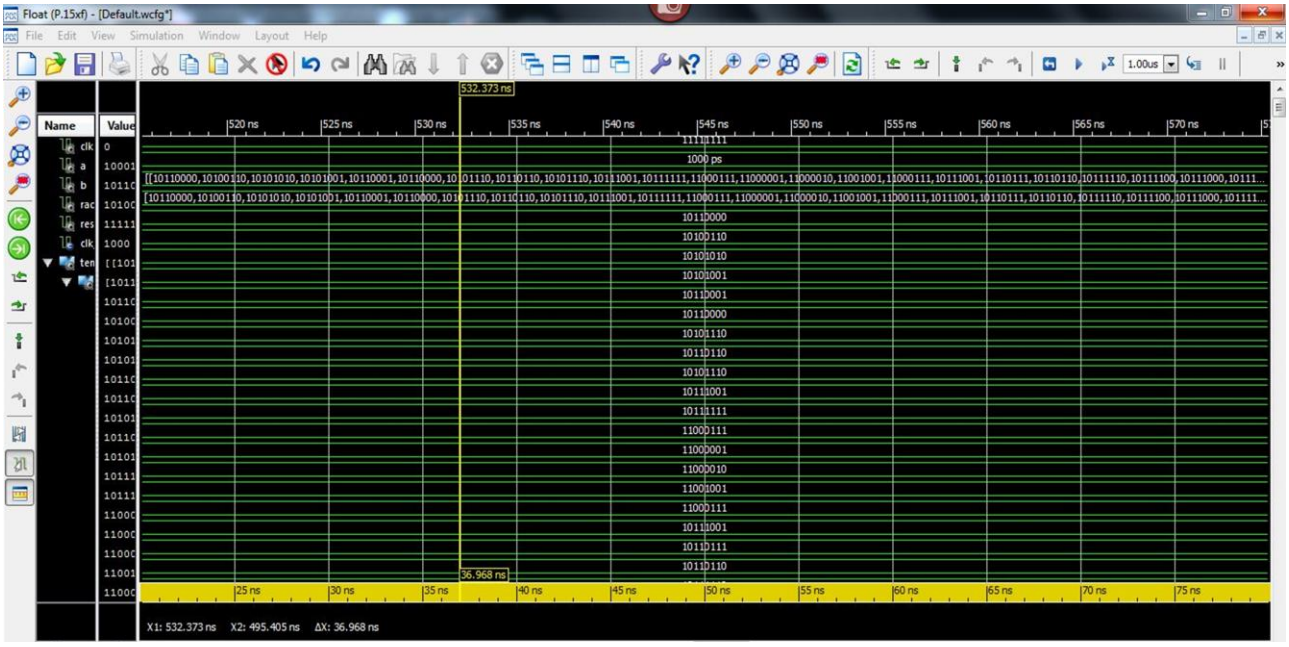


Fig. 6: Simulation of normalizations using BCA on FPGA

9. Conclusion

This paper shows parallelized of two part of iris recognition algorithm on FPGA. The parallelized in FPGA for segmentation and normalization help to speed up the execution of iris recognition by approximately 1676 and 1463 faster than executing CPU process for segmentation and normalization respectively. The resized of captured image helps to speed up the execution at least to quarter the time of execution and maintained the same accuracy as well in segmentation process. Quarter of the iris is sufficient to identify between human. FPGA allows making iris recognition system works in real time with low cost and higher speed and performance.

10. Reference

[1] R. W. Ives, R. P. Broussard, L. R. Kennell, R. N. Rakvic and D. M. Etter, "Iris Recognition using the Ridge Energy Direction (RED) Algorithm," 42nd Asilomar Conference in Signals, Systems and Computers, 2008, Pacific Grove, CA, 2008.
http://dx.doi.org/10.1109/ACSSC.2008.5074610

[2] S. S. Omran and A. A. Al-Hilali, "Half Iris Matching Based On Red Algorithm," in First International Engineering Conference (IEC2014), Erbil, 2014.

[3] S. S. Omran and A. A. Al-Hilali, "Half Iris versus Circular Iris Matching," in Proceedings of 2015 International Conference on Image Processing, Production and Computer Science, Istanbul, 2015.

[4] S. S. Omran and A. A. Al-Hilali, "Quarter of Iris Region Recognition Using the RED Algorithm," in 17th UKSIM-AMSS International Conference on Modelling and Simulation, Cambridge, 2015.

[5] J. Daugman, "How iris recognition works," IEEE Transactions on Circuits and Systems for Video Technology, vol. 14, no. 1, pp. 21-30, 2004.
http://dx.doi.org/10.1109/TCSVT.2003.818350

[6] J. Canny, "A Computational Approach to Edge Detection," IEEE Transactions Pattern Analysis and Machine Intelligence, Vols. PAMI-8, no. 6, pp. 679 - 698, 1986.
http://dx.doi.org/10.1109/TPAMI.1986.4767851

[7] H. T. Ngo, R. N. Rakvic, R. P. Broussard and R. W. Ives, "Resource-Aware Architecture Design and Implementation of Hough Transform for a Real-time Iris Boundary Detection System," , IEEE Transactions on Consumer Electronics, vol. 60, no. 3, pp. 485 - 492, 2014.
http://dx.doi.org/10.1109/TCE.2014.6937334

[8] J. Daugman, "High confidence visual recognition of person by a set of statistical independence," IEEE Transaction on Pattern Analysis and Machine Intelligence,, vol. 15, no. 11, pp. 1148-1161, November 1993.
http://dx.doi.org/10.1109/34.244676