International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 06, June-2015

A Distinct Approach to Vein Pattern Recognition

An Effective Design for Secure Systems in Developing Nations

Muntaka Ahmed Chowdhury Chaity, Imtiaz Ahmed Khan, Iqbalur Rahman Rokon
Department of Electrical and Computer Engineering
North South University
Dhaka, Bangladesh

***Abstract*— Biometric-based identification has steadily gained immense popularity as the need for perfectly secure systems has become a top priority issue these days. Among biometric identifications such as fingerprint, vein pattern, iris, voice, face recognitions, vein pattern recognition stands out as it provides excellent security and further researches are still being conducted to improve the system. This paper introduces the skeleton of a vein pattern recognition system that we have designed using Verilog HDL (Hardware Description Language) keeping FPGA (Field Programmable Gate Array) implementation in mind. Developing nations can greatly benefit from using FPGA-based security systems as FPGAs are reconfigurable, faster but consume lower power and are thus more cost-effective than general purpose processors. We have used slightly modified existing algorithms to program most parts of our system. Our vein pattern recognition system ensures a far simpler but effective pattern extraction using Verilog HDL .**

Keywords—Vein pattern; Verilog HDL; FPGA

I. INTRODUCTION

Perfectly secure systems are very essential in today's world. Biometric identification has been used for authentication purposes for quite some time now. Biometrics refers to methods of recognizing a person based on a certain feature of his body like the voice, iris, fingerprint, retina etc [1]. Biometrics has numerous advantages over other means of authentication. Authentication using passwords and cards have numerous disadvantages such as the fact that passwords can be forgotten, and cards lost or stolen. Vein pattern recognition stands out among all other biometric identification, as vein patterns cannot be forged. Each person has a unique vein pattern. . Moreover, a person has to be alive with blood flowing through his veins for this system to work, making it impossible to breach.

Various vein pattern recognition systems have been designed and researched on in the current years. Most of these systems have been designed using software in computers, using languages such as C, Matlab. Examples of recent work involving hardware implementation of vein pattern recognition systems include FPGA-based systems with Nios 2 Linux Operating systems running at 50 MHz clock rate [2]. Inspired by such work, we have designed the framework of a pattern recognition system using Verilog HDL which would allow direct hardware implementation.

FPGA based systems have numerous advantages over the ones based on general purpose processors. There is a large computational speed-up compared to programs running on CPUs as FPGAs allow parallel processing [3]. Complex biological calculations, such as pattern recognition can take advantage of this as this allows a much faster evaluating of data. Getting the work done much faster at significantly fewer clock rates than general purpose processors allows the FPGA to consume much lower power. A FPGA-based Vein Pattern recognition system would thus be a very cost-effective, secure system. Developing countries where living standards demand products of lower cost could make great use of such a system for authentication. A vein pattern recognition system can be used in a lot of places such as hospitals, banks, passport offices, government offices, libraries, personal computers etc [1].

Throughout this paper, we have explained the structure and methods we have used to design a vein pattern recognition system using Verilog HDL. These steps have been used by existing systems, and the algorithms used have been modified to accommodate our design. A new approach to pattern matching for future work has also been introduced in the end.

II. GENERAL SYSTEM OVERVIEW

In this section, we have explained the basic steps that have been used to design typical vein pattern recognition systems so far. The four basic consecutive steps are image acquisition, preprocessing, image segmentation and postprocessing, features extraction and matching.

A. Image Acquisition

In a vein pattern recognition system, the first step is image acquisition. An image of the veins in a person's palm, finger or back of hand is captured, and stored in the system. Different methods of capturing an image of the vein patterns are used. These patterns cannot be captured in visible light as the veins are beneath the skin. But hemoglobin in human blood absorbs Infrared (IR) rays of 700nm-1000nm [4]. Thus, low cost modified webcams with attached IR filters can be used to capture these images, where vein appear as dark patterns against a lighter background [4]. This image is then cropped to get the Region of Interest (ROI).

B. Preprocessing

At this stage, the cropped image undergoes a few stages which improve the quality of the image [5]. These processes vary for different pattern recognition systems. The Gaussian low-pass filter and median filter can be used to reduce noises, and then Histogram stretching can be applied for contrast enhancing. [5]. Then the preprocessed image is sent for segmentation.

C. *Image Segmentation and Postprocessing*

This is very important step and several methods exist. Different pattern recognition systems have preferred different methods according to the need of their system. Segmentation methods include Repeated Line Tracking which traces lines along veins, Local Thresholding which binarizes the image, Laplacian of Gaussian which is used for edge detection and reduction of noise etc [5]. The segmented image is then postprocessed to reduce further noise and remove pixel blobs from image [5].

D. *Features Extraction and Matching*

The postprocessed image undergoes features extraction. Extraction can be done using a method which finds maximum curvature points in images [6]. Then Thinning is applied to the image by implementing Zhang-Suen's thinning algorithm [7], which makes the vein pattern one pixel thick. Finally the intersecting points of the veins can be found using algorithms which vary across different systems. Then matching can be done by using several methods, one of which being Modified Hausdorff distance formula. Hausdorff distance measures the distance between two sets of points, and ultimately measures accurately the difference between two images [5]. When a stored image in the database matches a newly entered image of vein pattern, access to the system is granted. Otherwise, access is denied.

III. OUR SYSTEM

Our pattern recognition system is mainly focused on features extraction and matching. The system was designed using Verilog HDL. We have processed a 'dummy' image that we made by placing black and white pixels in desired places to create a certain pattern. We worked on this 'dummy' image through the following steps: preprocessing, segmentation and postprocessing, and features extraction. The algorithms we have used, and the methods we have implemented have worked out successfully, and we can ascertain that this process of pattern extraction and matching will work on real vein patterns when a NIR image is taken from a camera, grey-scaled, and then processed through our suggested method.

We have 19 modules for synthesis in our Verilog-based design- one for each step in the process, a few MUXs and DeMUXs, two RAMs to store the image after each step, and one main module. We have also created 1 testbench module (stimulus) for verification of our design.

A. *Functions of modules in brief*

- top.v: This is the main module which controls the whole operation as all the other modules are instantiated here.
- image.v: This is a memory location from where images are called into the system to be processed, to check whether our system functions correctly or not. It is treated as an external storage.
- crop.v: This module crops a 30×30 image to a 20×20 image pixel by pixel at every positive clock edge.
- hextraction.v: This module applies horizontal extraction serially to rows of pixels in an image. It takes one row at a time, and using the curvature formula, identifies the maximum curvature points as veins, and sets these points to a pixel value of '1' while the other pixels remain '0'.
- vextraction.v: This module applies vertical extraction serially to columns of pixels in an image. It takes one column at a time, and using the curvature formula, identifies the maximum curvature points as veins, and sets these points to a pixel value of '1' while the other pixels remain '0'.
- thin1.v and thin2.v: These modules apply the Zhang Suen's thinning algorithm to the image being processed. It targets each pixel, scans the neighboring pixels, and by applying the algorithm, finally gives a pattern which is just one pixel thick.
- intersections.v: This module finds the intersecting points or cross pointa in the pattern by scanning the surrounding pixels and applying a suitable algorithm.
- coordinates.v: This module gives the coordinates of the intersecting points found by intersections.v.
- prepro.v: It is a RAM we have created which is to be used in the vein pattern recognition system. The image being processed is stored here pixel by pixel before going through each step in the process.
- postpro.v: It is a RAM we have created which is to be used in the vein pattern recognition system. The processed image is stored here after each step of image processing.
- load.v: It is used to load an image from any external device or storage into the RAM in our system, that is, prepro.v. It is used only once in the entire process just to bring an image into the system.
- reload.v: It is used to reload images from postpro.v to prepro.v after each step in the process reaches completion.
- clear.v: Clear changes all the pixel values of postpro.v to '1'.
- eraseto0.v: Eraseto0 changes all the pixel values of postpro.v to '0'.
- mux2×1.v: This MUX takes in 2 inputs and gives 1 output. The inputs and outputs are 32 bits each. This MUX is used to take input from load.v and reload.v and give out requested data to a specific functioning module.
- mux16×1.v: This MUX takes in 16 inputs and gives 1 output. The inputs and outputs are 32 bits each.
- demux1×16.v: This deMUX has 1 input and 16 outputs. The inputs and outputs are 32 bits each.
- demux 1×161.v: This deMUX has 1 input and 16 outputs. The inputs and outputs are 1 bit each. This deMux is used to drive clock signals into each block.

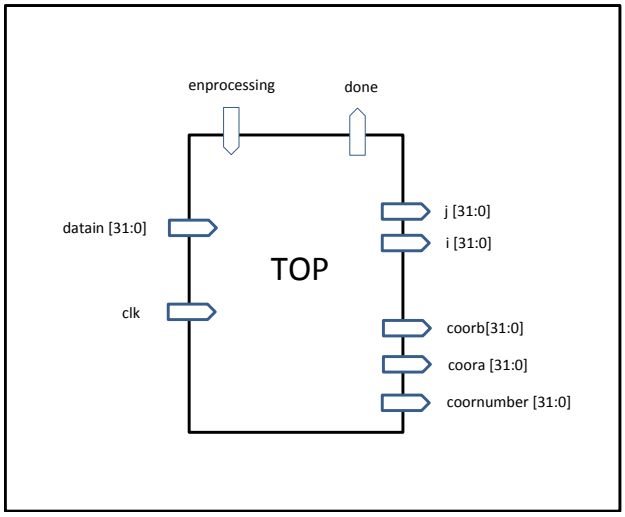

Fig. 1. Main module of the system

B. *Description of System*

The system is set to work when the 'enprocessing' signal is set to 1 from 0 which triggers the top module to start functioning. We have set apart a module named 'image.v' which would be treated as a storage which currently stores the image we would be working on. The stored images have a predefined dimension of 30×30. We have two RAMs, prepro.v and postpro.v, which would be frequently storing our image all throughout the processing steps. When the top module starts functioning, an image is transferred from image.v to prepro.v with the help of load.v which takes each pixel at every positive clock edge, and transfers it, pixel by pixel, to prepro. load.v has no more work after this step in the entire process. Then crop.v is enabled which calls the image, pixel by pixel, and crops the 30×30 image to 20×20. 'Pixel by pixel' here means each pixel is considered individually, processed, and sent to postpro. Once cropping is complete, the signal done is '1' from '0'. Then reload sets to work, transferring the cropped image from postpro to prepro. This action of reloading the processed image to prepro after each step is always controlled by reload.v. Next, eraseto0.v ensures that all the pixels in postpro are set to 0. This is the only time during the entire process when eraseto0.v functions. Then the system proceeds to the next step.

Next step is hextraction, which occurs when hextraction.v is enabled. It is enabled when cropping is over and the cropped image resides at prepro. Hextraction applies horizontal extraction to the cropped image, and each pixel of changed image is instantly uploaded to postpro. When hextraction is complete and done signal is '1', the image is now in postpro. But the image is not reloaded to prepro instantly like the previous step. Instead, vextraction is applied to it, and the combined hextracted and vextracted image is reloaded to prepro. Next, thinning is applied to the image by thin1.v and thin2.v in the same manner, and the one-pixel-thick image is reloaded to prepro. That's when clear.v sets to work for the first and only time. It clears the pixels of the entire postpro to '1'. Then intersections.v and finally coordinates.v perform their actions on the image. The coordinates.v is our final module, so when it signals 'done' to be '1', the coordniates of intersecting points of the pattern (image) are reloaded to prepro from postpro, and the entire process is complete.

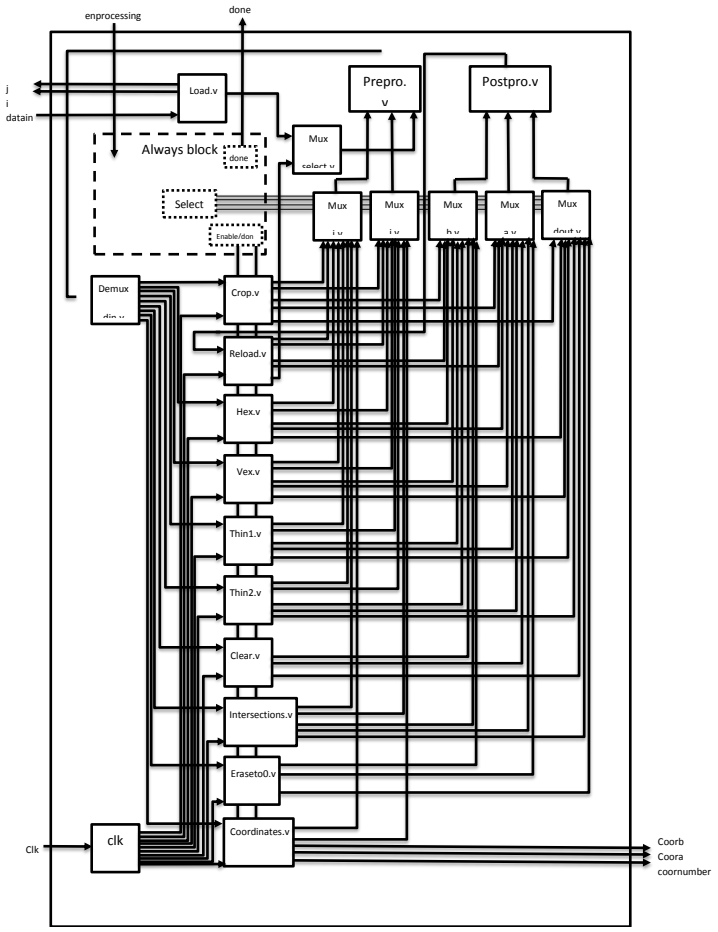

Fig. 2. Block diagram of the system

C. *Testing and Evaluation*

Our system was designed in ModelSim-Altera 6.5b and performance was tested using ISE Design Suite 14.5 (Xilinx). The block diagram of our entire system with all the modules is shown in Figure 2.

Screenshots of the waveforms of simulation of the design using a particular image (Figure 3) as an input is shown in Figure 4. The image we have used to show the waveforms is a pattern which has two crossing points. Our system makes the image undergo the series of processing steps described earlier and gives the number of intersecting points, and their coordinates as outputs, which can be seen in the waveforms. Once the scanning for intersecting or cross points is over, the done signal, which was '0' until now, changes to '1'.

The RTL (Register Transfer Level) schematic, which verified that our design is perfectly synthesizable and can be easily implemented in hardware, is shown in Figure 5.

Fig. 3. One of the 'dummy' images

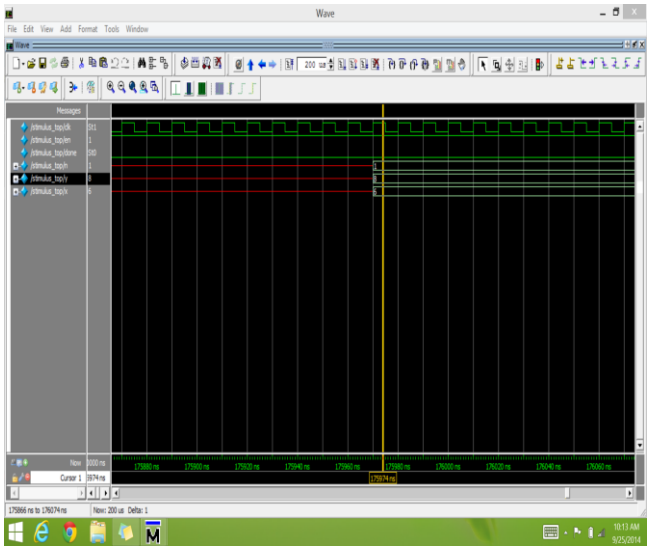

Fig. 4. Waveforms showing coordinates of intersections and number of points

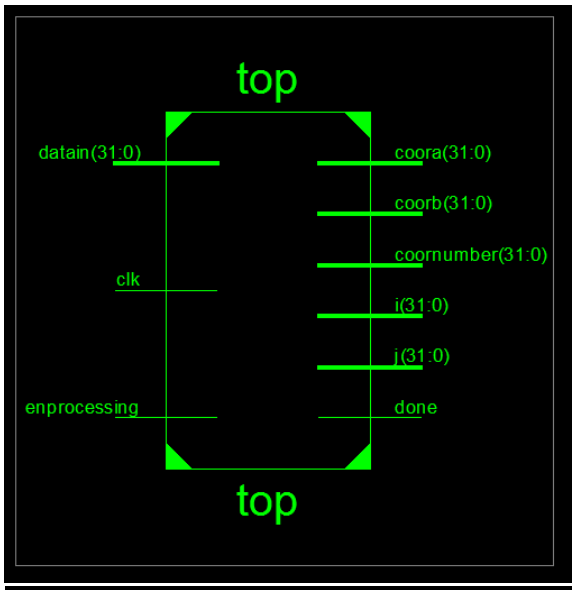

Fig. 5. RTL Schematic – Main module 'top.v'

IV. ALGORITHMS USED IN OUR SYSTEM

A. *Hextraction and Vextraction*

In our system, for horizontal extraction and vertical extraction processes, we have used a slightly modified version of finding maximum curvature points of image profiles, which finds the centerlines of veins using evaluating the maximum curvature points [6]. The algorithm has 3 steps consecutively: extracting position of center of veins, connecting center positions, and labeling images [6]. In our system, the cross-sectional profile of the pattern is checked which appear as dents because the pattern is darker then the background. These curves have large curvature, so center positions of the pattern are found by calculating maximum curvature of cross-sections. [6]. Then a filtering operation is done to connect the centerlines of the pattern and remove noises. Then labeling of image is done by considering pixel values smaller then a particular threshold as part of background, while the equal or larger values are part of the pattern [6]. This process is applied horizontally to rows and vertically to columns of pixels and finally extraction is complete.

B. *Thinning*

After applying horizontal and vertical extraction, we used Zhang-Suen's thinning algorithm for making the pattern just one pixel thick [7]. The pixel values of the pattern are considered '1's with the background pixels beings '0's. The Zhang-Suen's algorithm is applied in two steps to pixels with values '1's whose surrounding 8 neighboring pixels are observed. In the first step, a certain contour point X is considered for deletion if the following conditions are met [7]:

1. 2<= N(X1) <= 6
2. S (X1) = 1
3. X2*X4*X6
4. X4*X6*X8

Here N(X1) is the number of neighboring pixels of X1 which are non-zero, and S(X1) is the number of 0 to 1 transitions from X2 to X9. In the second step, conditions (1) and (2) are still the same, but (3) and (4) become - (3) X2*X4*X8=0 and (4) X2*X6*X8=0 respectively [7]. If one or more conditions from (1) to (4) are not met, then the value of the point being considered is not changed [7]. Otherwise, it is changed to '0' from '1'. In this way, the entire pattern is thinned to a structure which is one pixel thick.

C. *Finding end and cross-points*

From the thinned image, we found the crossing points and end points in the pattern using a simple process. We inspected every black pixel's 3×3 local neighbors [5] to see if the black pixel is a cross point or an end point. If it has one black neighbor only or two black neighboring pixels together, then this black pixel is an end point. If the chosen pixel has three black neighbors, or three black neighbors separated at least by one white pixel, then it is a cross point [5]. We successfully found the coordinates of the cross points and the number of cross-points in our pattern.

V. CONCLUSION AND FUTURE SCOPES

A vein pattern recognition system is very essential for today's world as the need for a perfectly secure system grows every day. A person trying to use the system has his vein patter's NIR image captured, processed, and matched to the images stored in the database. The system, if implemented in hardware, preferably FPGA, would be very advantageous and cost-effective as FPGAs are reconfigurable, faster than general processors due to parallel processing, and consume less power. We have successfully designed a pattern recognition system using Verilog HDL for hardware implementation.

Our design is programmed to currently process an image and give the coordinates of the intersections of veins. The process currently ends here, but it makes way for an efficient method that we would be using in future for matching an image with a stored image in the database. The cross points that we have found are certain distances away from each other. If we calculate the maximum and minimum distances, and then total distance from one cross point to another, and store these three values in a database, the same image can be matched with an exactly same image afterwards, because the maximum, minimum and total distance values will match. This would also ensure that system still functions accurately even if the hand is displaced slightly, because the distances between cross points will remain the same even if the coordinates of these points change. Since each human has a unique vein pattern, the values of distances will never be the same for a different pattern, thus ensuring a perfectly secure vein pattern recognition system.

.

REFERENCES

[1] M. A. Ahmed, H. M. Ebied, E. M. El-Horbaty, and A. M. Salem, "Analysis of palm vein pattern recognition algorithms and systems," Int. J. Bio-Med. Informatics. E-Health, vol. 1, pp. 10-14, June-July 2013.

[2] P. C. Eng and M. Khalil-Hani, "FPGA-based embedded hand vein biometric authentication system," in *Proc.TENCON 2009-2009 IEEE Region 10 Conf.*, Singapore, 23-26 Jan, pp.1-5.

[3] D. Popig, D. Ryle, D. Pellerin, and E. Stahlberg, "Applying field programmable logic arrays to biological problems,". [Online].

[4] M. Khalil-Hani and P. C. Eng, "Personal verification using finger vein biometrics in FPGA-based system-on-chip," in *Conf. ELECO 2011 7th Intl. Conf. Elect. Electron. Eng.,* Bursa, Turkey, 1-4 Dec, pp. 151-156.

[5] C. Petitimbert, M. Distler, M. G. Myrtue, S. N. Jensen, T. B. Moeslund, K. Nasrollahi, "Biometric identification using hand vein patterns," P6 Student Project, Dept. Electron. Syst., Aalborg Univ., Aalborg, Denmark, 2011.

[6] N. Miura, A. Nagasaka, and T. Miyatake, "Extraction of finger-vein patterns using maximum curvature points in image profiles ," *IAPR Conf. MVA 2005*, Tsukuba Sci. City, Jpn., May 16-18, pp. 347-350.

[7] A. S. Karne and S. S. Navalgund, "Implementation of an image thinning algorithm using verilog and MATLAB," Proc. Nat. Conf. Women Sci. Eng. (NCWSE 2013), Dharwad, India, pp. 333-337.