سومین کنفرانس ماشین بینایی
و پردازش تصویر ایران
تهران، ۵ و ۶ اسفند ماه ۱۳۸۳
The Third Conference on
Machine Vision, Image Processing
& Applications (MVIP 2005)
Tehran, Iran, Feb. 23-24, 2005

هفتادمین سال تاسیس دانشگاه تهران - ۱۳۸۳

Binarization and Thinning of Fingerprint Images by Pipelining

Shadrokh Samavi, Farshad Kheiri, Nader Karimi
Department of Electrical and Computer Engineering.
Isfahan Univ. of Technology, Isfahan, Iran 84156

Abstract : *Two critical steps in fingerprint recognition are binarization and thinning of the image. The need for real time processing motivates us to select local adaptive thresholding approach for the binarization step. We introduce a new hardware for this purpose based on pipeline architecture. We propose a formula for selecting an optimal block size for the thresholding purpose. We also present in this paper a new pipeline structure for implementing the thinning algorithm.*

Keywords: Fingerprint, Binarization, thinning, Pipeline Processing.

1 INTRODUCTION

In today's life style, security is one of the most important concerns. Technology could provide us with this purpose. Perhaps the first step is to identify a person. Identification cards or simple identification numbers are two examples of this system. By identifying a person more precisely, the system's security can improve.

One of the methods to identify a person is biometrics. It is defined as a science which studies human's behaviors and physical characteristics, to identify him/her [1]. This identification is done by recognition of face, hand, voice, retina, iris and fingerprint. Although iris recognition is one of the most precise methods, it is not acceptable by all the people as a non-invasive method. It seems that fingerprint recognition can be the next choice.

Fingerprint recognition was first employed by Scotland Yard in 1901 as an identification system for the first time [2]. They used Henry-Galton system. This system identifies five types of fingerprint which are shown in Figure 1. The categories that are shown in Figure 1 are arches, tented arches, left loops, right loops, and whorls.

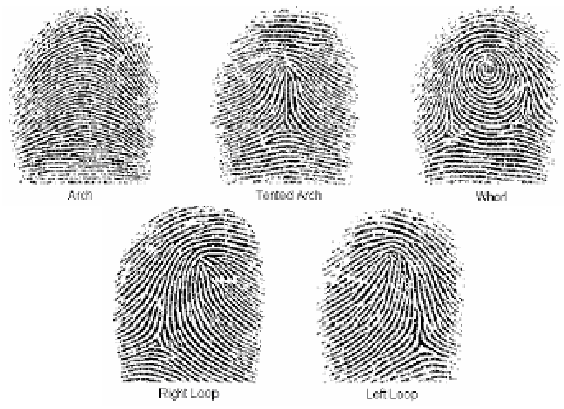


Figure.1: Five categories of fingerprints [2].

Later a more precise system, based on minutiae points was introduced. Minutiae points consist of two characteristics, ridge ending and ridge bifurcation.

Extracting minutiae points and matching them in two fingerprints are two main tasks in the recognition process. To extract minutiae, a fingerprint image should have 40 to 100 minutiae [2]. To match two fingerprints, between 10 to 16 minutiae should match. The variation is related to different human races [2].

For an input image the following processing stages which are shown in figure 2, can be used in minutiae extraction.

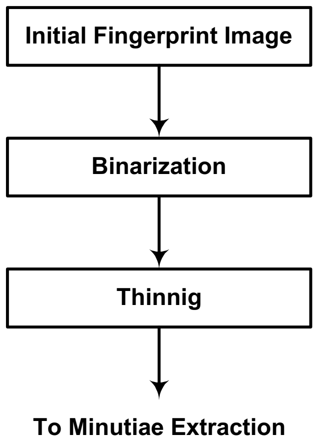


Figure 2: The common preprocessing stages used before minutiae extraction .

In binarization, a binary image will be obtained from a gray-scale image. Then the binarized image could be thinned to produce lines that are almost one pixel width. By thinning, minutiae extraction will be easier than before. It is obvious that applying a poor binarization method will affect all other stages adversely. Therefore it can be a critical step in fingerprint recognition.

For fingerprint image processing, image production could be performed by line-scan procedure and on the other hand the need for a real-time process results in applying a pipeline structure. In this paper a new approach based on pipeline architecture is presented. This approach consists of thresholding and thinning. In part 2, binarization is explained in more details. Thinning algorithm is studied in part 3. The suggested pipeline structure is introduced at part 4 and then complexity analysis is discussed in part 5. Finally conclusions come in part 6.

2 BINARIZATION

For an input image, some processing stages should be used before extracting minutiae. One of these stages is binarization. In this stage the gray-scale image converts into a binary image. A binary image can be processed better than a gray-scale image [2]. As mentioned in [3], there are three main approaches in binarization:

1) Global Approaches,
2) Neighborhood-Based Approaches
3) Filter-Based Approaches.

In another point of view, approaches can be divided into two main categories:

1) Software routines
2) Hardware methods

Software approaches are computationally expensive, time-consuming, highly flexible, and non-real time. Filter-based approaches belong to this group. The other group which is of our interest can be implemented in real time systems. This group can be classified into global and local approaches. Global approach does not fulfill the requirements. As it is explained in [3], it can produce unfavorable results. These results can be the side effects of non-homogenous press of finger on the scanner. On the other hand, hardware implementation for this approach will be architecturally expensive.

Since we have limitation in the area of speed of processing, it seems that among the above approaches, local adaptive thresholding is a good choice. It has fast speed, low complexity, fair quality and good robustness [3].

The basic idea in thresholding is to select a threshold (T) to extract an object or several objects with the same value from background [6]. We limit our discussion to one-level thresholding for gray-scale images. Equation(1) can be used to binarize gray scale images:

$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \le T \end{cases} \qquad (1)$$

where f(x,y) is the value of a pixel in gray-scale image and g(x,y) is the binarized image. In adaptive thresholding, the image is divided into series of blocks. A threshold will be defined for each block. All pixels in the block will compare with this threshold. In this approach, we encounter with two main problems:

a) What is the optimal size of a block?
b) How is the threshold calculated for each block?

As mentioned in [2], it has been empirically proven that blocks with size 16*16 are the best fixed size for binarization of fingerprint images. To prove this subject, it is known from [6], to select a good threshold, the histogram peaks should be separated by deep valleys.

By inspection, it seems that variance is a good factor to select a fixed size block. Firstly, each image is divided into number of equal blocks. Next, the threshold value is calculated inside each block. Thirdly, the mean gray scale value of all blocks is calculated. Then variance will be defined from[4].

$$\sigma^2 = 1/n \sum_{1}^{n-1} (Ti - \mu)^2 \qquad (2)$$

T_i is the threshold value of each block, μ is the mean of all blocks, n is the number of blocks. To incorporate the value of block size into the formula, we multiply σ^2 by the forth root of block size. This formula satisfies the results found empirically. We, therefore, define the following factor for an N×N block:

$$\text{Block Factor} = \sigma^2 \sqrt[4]{N} \qquad (3)$$

We calculate this factor for various N's of size 4, 16, 64, and 256. These block sizes are for a 512×512 pixels image. We excluded 2, 8, 32, 128, and other block sizes since they did not produce good results. This factor results in 16*16 as being the choice. Results of applying different block sizes to an image are shown in Figure 3. By inspection, it seems that 16*16 is the best size. The mentioned factor produced good results for a large number of images.

To select an optimal thresholding value, we should minimize the erroneously classifying ridge as valley or vice versa. This is calculated in equation (4). By observing the histogram of a fingerprint in figure 4, and the resemblance between this histogram and the presented one in [5], ridge and valley can be replaced by object and background respectively. Under the same discussion as it has been done in [5].

$$T = (1/2)(\mu_1 + \mu_2) + (\sigma^2/(\mu_1 - \mu_2)) \,\mathrm{Ln}(P_1/P_2) \qquad (4)$$

μ_1 is the mean value of the object, μ_2 is the mean value of the background, σ^2 is the variance of the pixels. P_1 and P_2 are the probabilities of occurrence of the two classes of pixels. It seems to be legal to assume $P_1=P_2$ for fingerprint images if the region of interest in an image is only the fingerprint image, not background. Because it contains the same number of pixels of valley and ridge. So the above equation can be simplified as:

$$T = (\mu_1 + \mu_2)/2 \qquad (5)$$

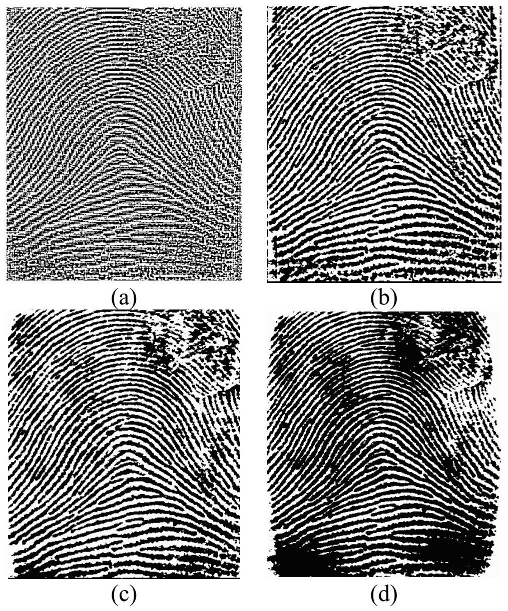


Figure 3: Binarized image with block Size (a) 4*4 (b) 16*16 (c) 64*64 (d) 256*256

The local adaptive thresholding is acted on an image in Figure 5. As it can be seen some minutiae points have been damaged. Hence, the binarized image needs an extra step to enhance the image. This stage is the thinning process.

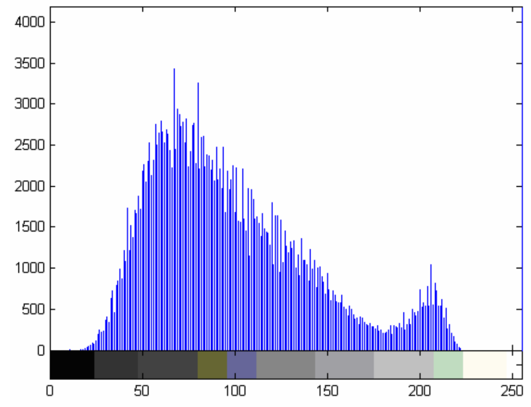


Figure 4: Histogram of a Typical Fingerprint.

3 THINNING PROCESS

From what is presented in [6], thinning is defined as a procedure to transform a digital binary pattern to a connected skeleton of unit width. Two basic implementations available for this approach are *sequential* and *parallel* methods.

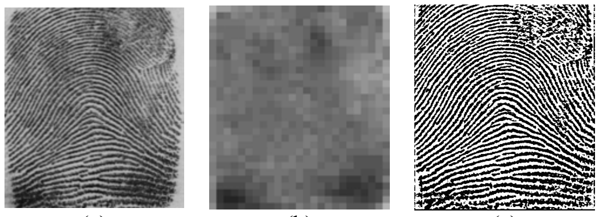


Figure 5: An Example of Local Adaptive Thresholding With Block size 16*16, (a) Fingerprint (b) Series of Blocks with Values of Thresholds to Compare with Initial Image, (c) Binarized Image.

In the sequential method, based on the results that are obtained, all pixels are examined and changed. The parallel method works on all pixels simultaneously. As discussed in [6] parallel thinning is substantially faster on pipeline computers. It is also mentioned in [7] that in all thinning algorithms only values of the closest neighbors are needed to remove a pixel. Therefore a window could be moved on the image in order to examine the center pixel based on its neighboring pixels.

The need for thinning process is discussed extensively in [8] and [9]. From what is indicated in [3], thinning procedure increases the speed of minutiae extraction.

Out of all thinning algorithms, what are presented in [10] and [11], are selected for our purpose. Simplicity and fixed-size window (3×3) of this algorithm motivated us to choose this method. There are algorithms that use different windows [6]. Of course we modified the algorithm to fulfill the needs of our application.

To implement the thinning algorithm, based on what is presented in [11], four conditions in each sub-iteration should be provided. There are two common conditions in each sub-iteration which are:

a) $3 \leq B(P_1) \leq 6$ b) $A(P_1)=1$

The number of 01 patterns in the ordered set $P_2, P_3, \ldots, P_8, P_9$ is $A(P_1)$ and the sum of $P_2, P_3, \ldots, P_8, P_9$ is $B(P_1)$. The locations of $P_2, P_3, \ldots, P_8, P_9$ in relation to P_1 are shown in Table 1.

There are 512 possible combinations that these one bit pixels could have. P_1 should be 1, therefore the number of cases reduce to 256. Based on the first condition, it is concluded that the number of 1's in P_2 to P_9 should be 3, 4, 5 or 6. If we consider the neighboring pixels of P_1 as forming a ring then the next condition indicates that the 1's in this ring should follow each other. It means that there should be no gap between 1's in the string. The first member of this group of 1's can be located in eight different positions.

Table 1.: Position of pixels in window.

P_9 **(i-1,j-1)**	**P_2** **(i-1,j)**	**P_3** **(i-1,j+1)**
P_8 **(I,j-1)**	**P_1** **(i,j)**	**P_4** **(i,j+1)**
P_7 **(i+1,j-1)**	**P_6** **(i+1,j)**	**P_5** **(i+1,j+1)**

The number of possible combinations will reduce to 32 terms; there are 8 permutations for each sum. The other two conditions that should be satisfied in each sub-iteration are as following.

c) $P_2*P_4*P_6=0$ d) $P_4*P_6*P_8=0$

Or

c') $P_2*P_4*P_8=0$ d') $P_2*P_6*P_8=0$

Therefore, there are two sub-iterations. Sub-iteration I should satisfy conditions a, b, c and d. Then sub-iteration II should satisfy conditions a, b, c', and d'. Conditions (c) and (d) cause combinations shown in table 2 to be omitted.

Table 2 : Combinations omitted by conditions (c) and (d).

P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	0
1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	0
0	0	1	1	1	1	1	1
1	1	1	1	1	0	0	1

No combinations should omit from 3 or 4 ones, because from P_2 to P_6 and from P_4 to P_8 there are 5 gaps, therefore these two conditions will be satisfied for all 3 or 4 connected-ones. Finally 26 combinations will remain. Each combination is considered as a minterm and its logic function is realized. The *thinning processor circuit* (TPC) is implemented based on these 26 combinations.

The number of repetition of the iterations should be discussed as another topic. It is indicated in [12] that the value of the spatial frequency of the ridges and valleys in a local neighborhood lies in a certain range for 500 dpi image. This range is [1/3, 1/25]. Half of the inverse of this value determines the thickness of ridges. Therefore, at most, six iterations are needed to be performed. Each time two pixels are omitted after applying the thinning algorithm. This has been proved experimentally. For a group of fingerprint images, a number of iterations are applied and then the initial binarized image is subtracted from this image for a region of interest. The total number of changed pixels for each iteration is then normalized. Figure 6 shows that after six iterations the thinning process is exhausted.

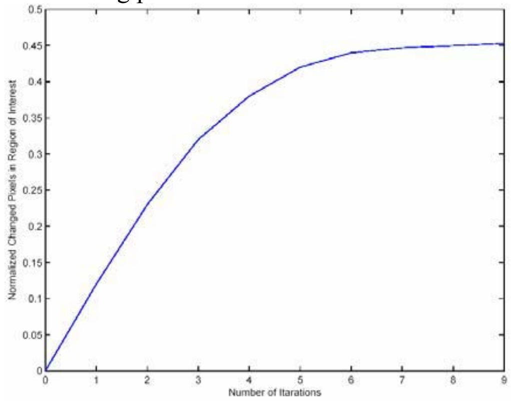


Figure 6 Variation of pixels after iterations for a fingerprint image.

It can be observed that after applying iterations six times, variations in values of pixels will be negligible. For thinner ridges this can take place sooner.

4 PIPELINE STRUCTURE

To binarize a gray-scale image based on adaptive thresholding approach, firstly the image is divided into series of 16×16 blocks. Then the mean gray scale value of each block is calculated as the threshold, T, for that block. By comparing each pixel in a block with this mean value binarization is performed.

We need to add the pixels of a 16×16 blocks in a circuit. The number of inputs for such circuit is large. To reduce the number of inputs a pipeline is devised which loads only one row of pixels of block at each clock pulse. This means that sixteen 8-bit values are loaded at each clock pulse. Therefore, for every sixteen pixels there is a *mean value calculator unit (MVCU)*.

To calculate the mean value of 16×16 block all pixels are added and the sum is divided by 256. In a pipeline structure to implement the MVCU a seventeen 8-bit inputs adder is needed. This adder is to take 16 pixels of each line of a block and add them with the results obtained from previous line which is another 8-bit input. This last input is a feedback from the low part of the output of the adder. To divide the total result into 256 an 8-bit right shift is needed. Figure 7 shows the structure of a MVCU.

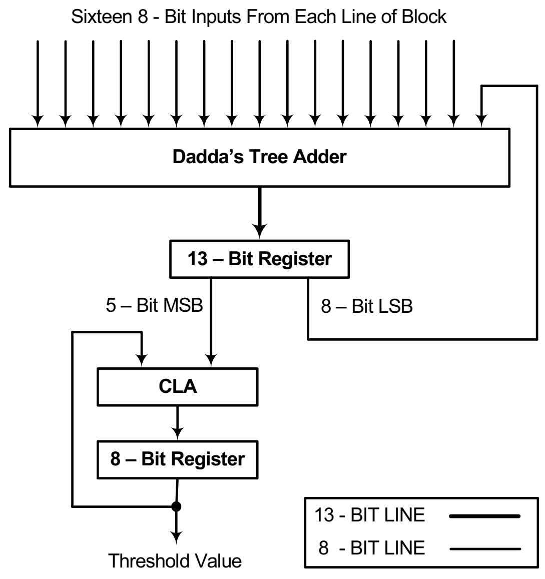
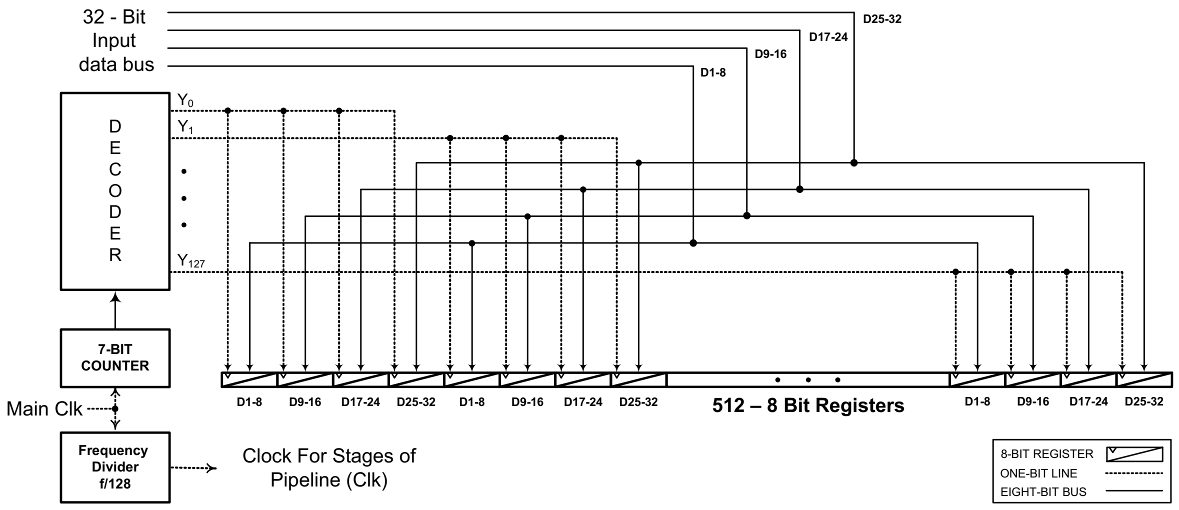


Figure 7: Internal structure of a MVCU

There are two adders in the circuit of Figure 7. The adder with 17 inputs is more time demanding and requires special attention. Therefore, after applying different methods it was concluded that Dadda's method which uses Carry Save Adders (CSA) produces least delay [13]. Since there are seventeen 8-bit inputs, based on the Dadda's design an initial layer is required to reduce the number of inputs to thirteen. Four CSA's are employed in the first layer and then five layers of CSA are used to reduce the thirteen inputs to two outputs. At a final stage a carry look ahead adder (CLA) is used to add the two final outputs.

The first stage of the pipeline is shown in Figure 8. This stage receives the image through a 32-bit bus and distributes it on 512 registers. The next 15 stages of the pipeline, which are shown in Figure 9, perform the binarization process. Every row of the image that comes into the pipeline is divided among 32 MVCU's. By the time that 16 rows of the image are loaded into the pipeline each MVCU has calculated the mean gray scale value of a 16×16 block. The output of a MVCU is latched into an 8-bit register. This register is clocked once every 16 clock pulse of the regular pipeline stage. At this time the pixels of each of these 16 rows are compared with the corresponding latched outputs of MVCU's to produce a row of a binary image.



. Figure 8. First of the pipeline responsible for receiving the image through a 32-bit bus.

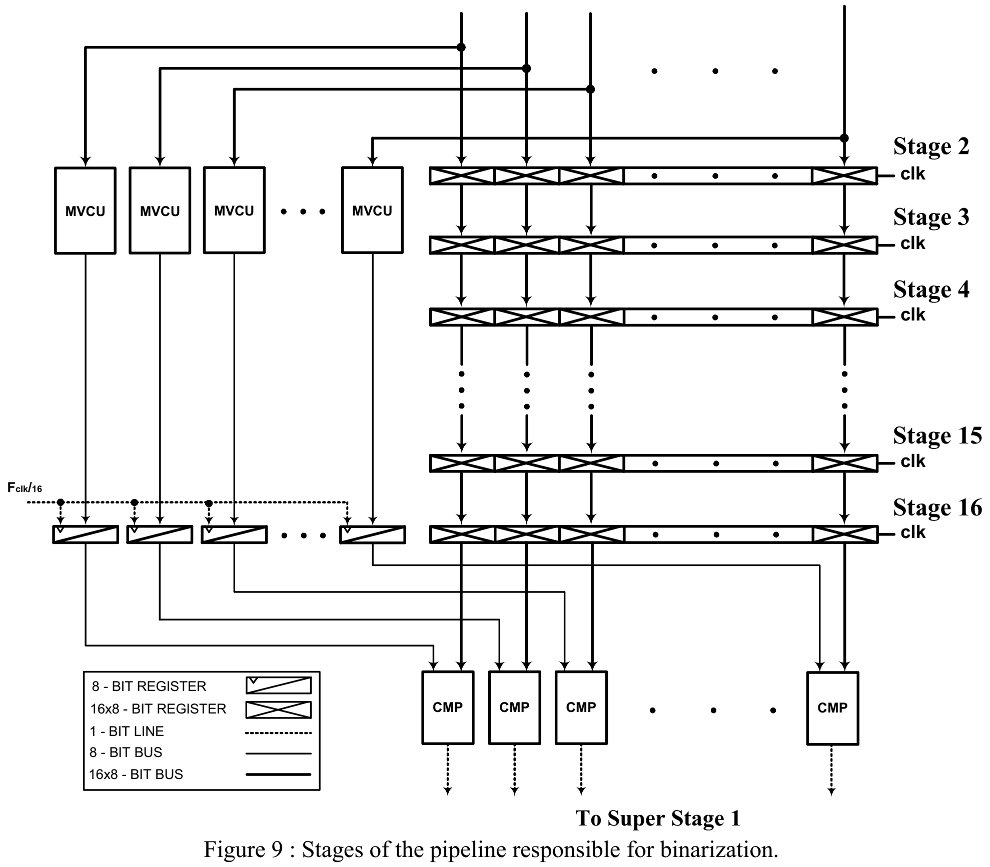


Figure 9 : Stages of the pipeline responsible for binarization.

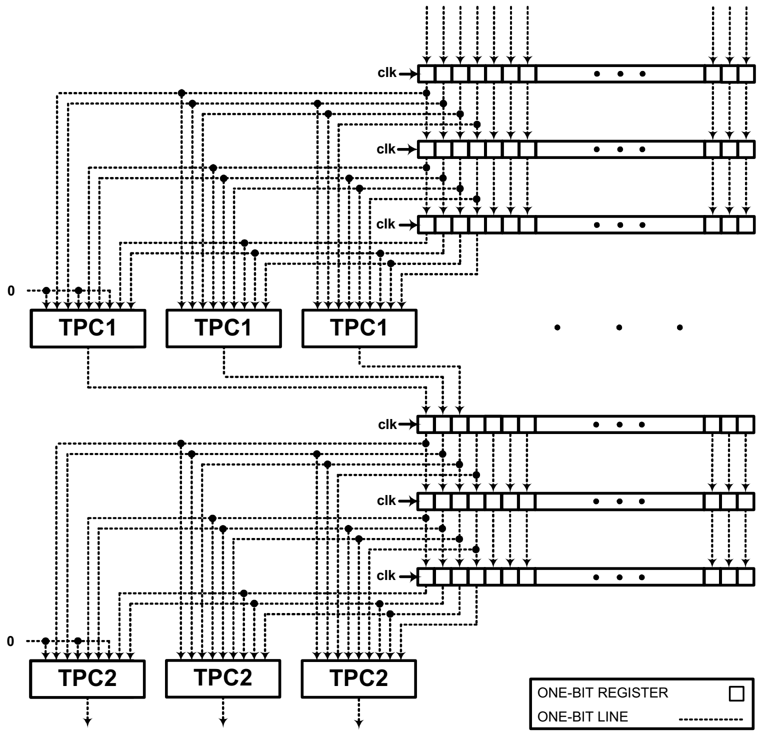


Figure 10: The pipeline structure of thinning algorithm.

The hardware design that we propose for the thinning process has advantages over the methods presented in [6] and [14]. One of these advantages is the application of a fixed size window which results in easier implementation. Also in comparison with [15] our design is advantageous. We implement the modified thinning algorithm by changing the first condition mentioned in section 3. We also simplified the implementation of conditions, whereas in [15] a direct method is applied. Furthermore, in [15] the number of iterations that are required is not mentioned. Our method uses a fixed number of iterations.

Figure 10 shows the hardware required for one iteration of the thinning process. The thinning mask that is used is a 3×3 window. Since two sets of conditions should be checked six pipeline stages are required to implement one iteration of the thinning algorithm. The binarized image is loaded into three stages. The nine pixels of the window are loaded into a *thinning processor circuit* (TPC1) which implements the first thinning condition. Then the outputs of TCP1's are loaded into the next three stages of the pipeline. The second set of conditions will be checked by a TCP2.

Based on results that are presented in Figure ??? these conditions should be examined six times. We refer to the six stages of Figure 10 as one super stage. Therefore six super-stages are required to complete the thinning process

5 EXPERIMENTAL RESULTS

The suggested structures were implemented using Vertex II FPGAs from Xilinx Corporation.. Table 3 shows the implementation results and worst case delay in pipeline structure. The simulation results show that the longest delay takes place in the first stage of pipeline and it equals to 2.56 microseconds. Therefore the processing time for a 512*512 image is about 1.45 milliseconds.

Table 3. Implementation results

FPGA type	Employed CLB Slices	Worst Case Delay
2V8000bf975	42452	2.56 μ s

Figure 11 (a) shows a typical fingerprint image that is processed by the proposed hardware. Figure 11 (b) depicts a binarized image produced by the binarization section of the pipeline. Figure 11 (c) is the final thinned image that the proposed pipeline has produced.

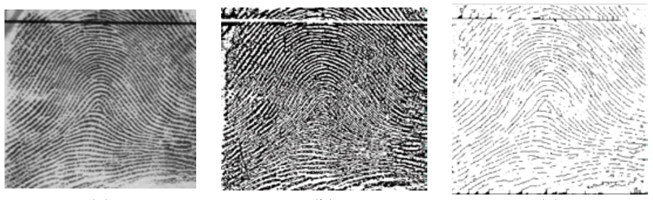

(a) (b) (c)

Figure 11: (a) A typical fingerprint image (b) The binarized image (c) The thinned image.

6 CONCLUSION

This paper presents an improved binarization and thinning algorithm. We offered a novel architecture for implementing these algorithms. The proposed method, which is based on pipeline structures, is well suited for real time processing of fingerprint images when line scanners are used. We were able to achieve execution times of less than 1.5 ms. Numerous images have been processed by the proposed hardware which proved its correct functionality and performance. Furthermore the structure has highly regular and expandable.

References

[1] ***http://www.eb.uah.edu/ece/biometric.htm June,2004.***

[2] M. Eriksson, "Biometrics Fingerprint Based Identity Verification", UMEA UNIVERSITY, Department of Computing Science, August 2001.

[3] P. Meenen, R. Adhami, "Approaches to Image Binarization in Current Automated Fingerprint Identification Systems", Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, AL 35899 USA.

[4] A. Papoulis, A.U. Pillai, *Probability, Random Variables and Stochastic Processes*, McGraw-Hill, 2002.

[5] R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Prentice Hall, New Jersey, 2002.

[6] B. K. Jang, R. T. Chin, "One-Pass Parallel Thinning : Analysis, Properties and Quantitative Evaluation", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 14, No. 11, pp. 1129-1140, 1992.

[7] T.M. Bernard, A. Manzanera, "Improved Low Complexity Fully Parallel Thinning Algorithm", *International Conference on Image Analysis and Processing,* Venice, Italy ,pp. 215, 1999.

[8] E. V. Duro, "Fingerprints Thinning Algorithm", *IEEE Aerospace and Electronic Systems Magazine,* Vol. 18, Issue: 9, pp 28-30, 2003.

[9] E. V. Duro, "Mathematical morphology Approaches for Fingerprints Thinning", The *36th Annual International Proceedings on Security Technology,* pp. 43-45, 2002.

[10] T. Y. Zhang, C.Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns", *Image Processing and Computer Vision, ACM*, Vol. 27, No. 3, pp. 236-239, 1984.

[11] H. E. Lu, P.S.P. Wang, "A Comment on A Fast Parallel Algorithm for Thinning Digital Patterns", *ImageProcessing and Computer Vision, ACM*, Vol. 29, No. 3, pp 239-242, 1986.

[12] A. Jain, L. Hong, Y. Wan, "Fingerprint Image Enhancement: Algorithm and Performance Evaluation", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 20, No. 8, 1998.

[13] B. Parhami, *Computer Arithmetic*, Oxford University Press, 2000.

[14] L. Huang, G. Wan, C. Liu, "An Improved Parallel Thinning Algorithm", *Proceeding of Seventh International Conference on Document Analysis and Recognition,* pp.780–783, 2003.

[15] P.Y. Hsiao, C. H. Hua, C.C. Liu, "A novel FPGA Architectural Implementation of Pipeline Thinning Algorithm", *Proceedings of the International Symposiumon Circuits and Systems.*, pp 593-596, Vol. 2, 2004.