View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by idUS. Depósito de Investigación Universidad de Sevilla

Dedicated Hardware IP Module for Extracting Singular Points from Fingerprints

M.C. Martínez-Rodríguez, R. Arjona, P. Brox, I. Baturone
Instituto de Microelectrónica de Sevilla (IMSE-CNM),
Spanish National Research Council (CSIC) and University of Seville, Seville, Spain
Email: {macarena, arjona, brox, lumi}@imse-cnm.csic.es

***Abstract*—In this paper a new digital dedicated hardware IP module for extracting singular points from fingerprints is presented (in particular convex cores). This module comprises four main blocks that implement an image directional extraction, a smoothing process, singular point detection and finally, a post processing to obtain the exact location of the singular point. A Verilog HDL description has been developed for this solution. The description has been synthesized and implemented in FPGAs from Xilinx.**

I. Introduction

Biometric solutions are widely explored for several applications of individual's recognition. In particular, fingerprints solutions have been studied due to its characteristics of distinctiveness. Fingerprint recognition systems also offer usability and are widely accepted by the users [1].

Nowadays, many embedded systems require that the fingerprint solution meets several constraints in terms of area, power consumption and timing responses. For example, small devices such as car keys, tokens or smart cards require these constraints. Although most of the reported solutions are implemented in software, other solutions cover from hardware-software codesign [2] to dedicated hardware [3]. However, dedicated hardware solutions are more adequate to provide those advantages.

The recognition process is divided into two parts; extraction of features and matching. Feature extraction is computationally costly, so that several solutions implement this operation outside the embedded device. Since matching is generally much simpler, it is implemented in the embedded device. This is known as *Match on Card* [4]. A disadvantage of this solution is that presents security vulnerabilities caused by the communication between the devices [5]. This is why other solutions are explored to incorporate the extraction of the features in the embedded system devices, which is known as *Authentication on Card*. For this purpose, hardware IP modules are very interesting.

A fingerprint is the reproduction of the epidermis and the image captured is composed of ridges and valleys as shown in Fig. 1a. Fingerprint features are categorized into three levels. Level 1 identifies the whole fingerprint offering global characteristics. This level employs textures or geometric information. Level 2 identifies local characteristics. In this level minutiae's extraction is usually employed. Level 3 is based on a finer extraction of characteristics, i.e. pores or scars. A higher feature extraction level implies that the complexity of the algorithm is higher. Level 1 is generally applied to classification techniques, level 2 is used to identification/authentication methods, while level 3 is not widely extended yet.

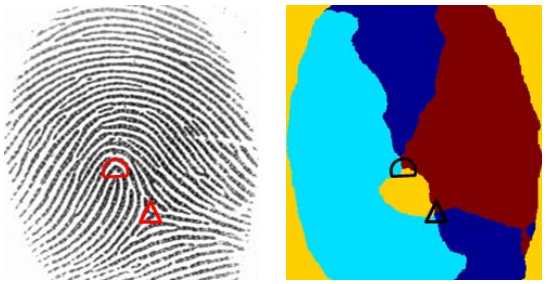
Directional image and singular points are among the level 1 features. Directional image is a matrix whose elements are the local ridge orientations at each pixel. Singular points are central features of fingerprint images. There are two types: core points, where ridge lines have a maximum curvature, and delta points, where ridges intersect. The singular points (in particular convex cores) are used to classify and align fingerprints [1], to define the singular area [6] and also to estimate the quality [7].

The dedicated hardware IP module described herein extracts the location of the convex core point from the fingerprint image. It has been hand written in Verilog language. The paper is organized as follows. In Section II, the architecture is exposed and all its blocks are briefly described. In Section III, the implementation results of the dedicated hardware IP module in Xilinx FPGAs are presented and compared to another implementation realized with Matlab-Simulink HDL Coder. Finally, conclusions are given in Section IV.

II. Architecture

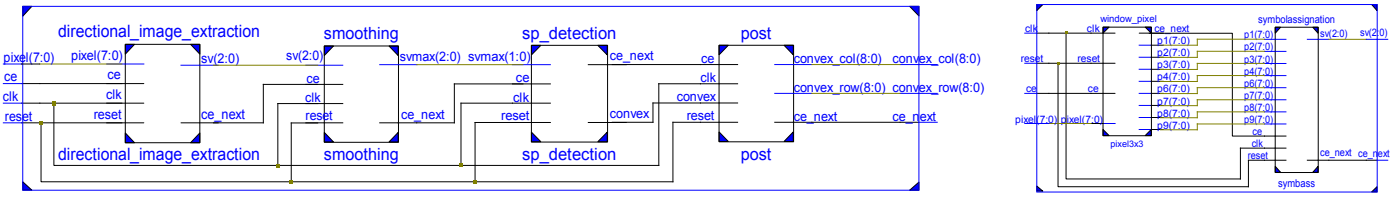
The fingerprint image is swept from the first pixel of the fingerprint, column by column and row by row. Therefore, several stages are carried out serially to each pixel.

The first stage generates a directional image assigning a symbol to each pixel related to the direction of the fingerprint



(a) (b)

Figure 1. (a) Fingerprint image and (b) 4-symbol directional image with convex core and delta points marked



(a) (b)

Figure 2. (a) RTL scheme of the architecture and (b) the directional image extraction block

ridges in that pixel. The second stage applies a smoothing process to the directional image. The result is that a new symbol is assigned to each pixel, which is the most predominant symbol in a window of 27x27 pixels around the current pixel. After that, singular points are detected. The smoothed directional image is explored through windows of 9x9 pixels looking for patterns that identify the singular point. The last step applies a post-processing in order to detect false positives and provides the location of the true singular point.

The architecture exposed here is applied to standard 8-bit grayscale fingerprint images. The size of the images considered is 440 rows and 300 columns. The architecture comprised four main blocks. Fig. 2a illustrates the main blocks and its interconnection.

A. Directional image extraction

The directional image is an image of symbols that represent the direction of the ridges at each pixel. This block classifies the ridge direction at each pixel in one out of a discrete set of possible directions. Depending on the number of symbols selected to represent the directional image, the range of directions associated to each symbol varies. Table I shows the range of directions corresponding to each symbol when 8 symbols (left side) and 4 symbols (right side) are selected.

The ridge direction at a pixel can be computed precisely by the following formula

$$dir(pixel) = \frac{\pi}{2} + \arctan\left(\frac{G_y(pixel)}{G_x(pixel)}\right) \qquad (1)$$

being Gx and Gy the horizontal and vertical gradients respectively for each pixel.

To calculate the gradients, a 3x3 window around the current pixel is processed in parallel and a Sobel filter is applied. Two line buffers with a depth of 300 words of 8 bits each and two 8-bit delay registers for each row are necessary to obtain all the pixels of the 3x3 window in parallel.

With the 3x3 pixels available in parallel, the gradients are calculated with full precision (11 bits), and then the direction and consequently the symbol is decided following a classification that it is established through the relation between the values of the gradients. The classification avoids computing division and trigonometric operations and only applies multiplications by constants and logical operators. Since 8 symbols are selected, the output of this block has a width of 3 bits. Fig. 2b shows the RTL scheme of the directional image extraction block.

Table I
8 AND 4 SYMBOLS ASSIGNATION IN THE DIRECTIONAL IMAGE

8 symbols			4 symbols		
	Bits	Range		Bits	Range
S1	000	$[0°, 22.5°)$	S1	00	$[0°, 45°) \cup [157.5°, 180°)$
S2	001	$[157.5°, 180°)$			
S3	010	$[22.5°, 45°)$	S2	01	$[22.5°, 67.5°)$
S4	011	$[45°, 67.5°)$			
S5	100	$[67.5°, 90°)$	S3	10	$[67.5°, 112.5°)$
S6	101	$[90°, 112.5°)$			
S7	110	$[112.5°, 135°)$	S4	11	$[112.5°, 157.5°)$
S8	111	$[135°, 157.5°)$			

B. Smoothing process

Smoothing process is applied to the directional image in order to obtain a homogenous directional image. After this process, some isolated and noisy symbol values in the directional image, which can appear due to irregularities in the fingerprint image, are removed.

The smoothing filter selected applies the maximum operator, that is, the predominant symbol value in a window centered in the current pixel is selected for the symbol value of the pixel in the new homogenous directional image. The size of the window depends on the sensor employed. Herein, the size selected is a 27x27 window that provides a good performance in optical, capacitive and thermal sweeping sensors.

Since the simultaneous access to 729 (27x27) pixels is very costly. The 27x27 smoothing is factorized in (3x3x3)x(3x3x3) smoothing to obtain hardware reduction, as follows:

1) smooth a 3x3 window.
2) smooth a (3x3)x(3x3) window by smoothing the previous 3x3 smoothed symbols.
3) smooth a (3x3x3)x(3x3x3) window by applying again the operation to the (3x3)x(3x3) previous smoothing symbols.

Fig. 3 illustrates the factorization of the 27x27 window (left) in the 9x9 window(middle) and 3x3 window (right) for the smoothing process.

To address the factorization, the process is comprised by three concatenated blocks that carried out each step above.

1) 3x3 Smoothing : A 3x3 window around the current symbol value provides 9 symbol values in parallel. To implement it, two line buffers and two delay registers for each row are

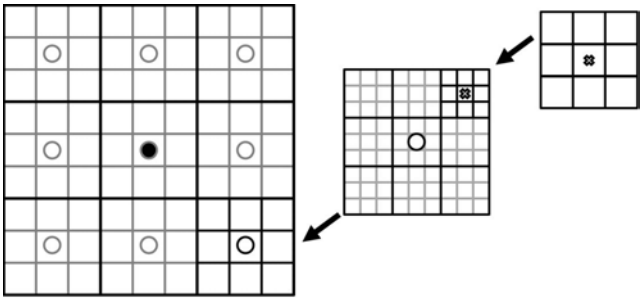


Figure 3. Factorization of a 27x27 window (left) in the 9x9 window (middle) and 3x3 window (right) for the smoothing process

necessary. Each line buffer has a depth of 300 bits and words of 3 bits, and each register is a 3-bit delay register.

Once the 9 symbol values are provided in parallel, they are counted and the output is the number of occurrences of each symbol value. Since there are 8 symbols and the count of each symbol occurrence needs 4 bits for the 9 values in a 3x3 window, the output has a total size of 32 bits.

2) 9x9 Smoothing : To compute the 9x9 smoothing, a 9x9 window around the current pixel should be considered. However, only 9 symbols which accumulate the results of 9 3x3 smoothings are required to be available in parallel. To obtain those values, two buffers with a size of two lines, 600 bits, and words of 32 bits (the size of the output of the previous stage), as well as, six 32-bit delay registers are necessary.

Once the 9 3x3 smoothing outputs are provided in parallel, they are added and the output of this block is the number of occurrences of each symbol value in a 9x9 window. The count of each symbol occurrence needs 7 bits. Therefore, the 9x9 smoothing output has a size of 56 bits.

3) 27x27 Smoothing : Similarly to the previous block, the 27x27 smoothing applies a 27x27 window around the pixel. However, only 9 symbols which accumulate the results of 9 9x9 smoothings are required. To obtain those values, two 56-bit buffers with a size of three lines, and 18 56-bit delay registers are necessary.

Once the 9 9x9 smoothing outputs are provided in parallel, they are counted, adding all the occurrences of each symbol in each window. The output of this block is the symbol value with the highest number of occurrences in the 27x27 window. Hence, the size of the output of this block is 3 bits.

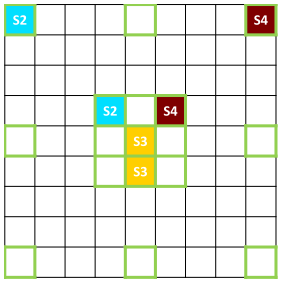


Figure 4. Pattern of a convex core point

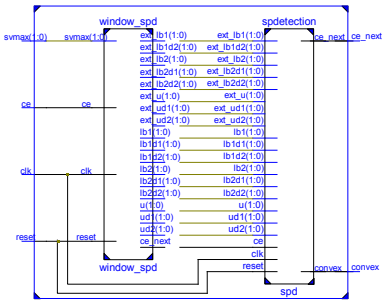


Figure 5. Singular point detection RTL scheme

C. Singular point detection

The singular points are located where the regions of a 4-symbol representation of the directional image intersect, as illustrated in Fig. 1b for the case of a convex core [7]. To detect singular points different patterns are searched in a 9x9 window. The translation of the smoothed directional image coming from the last stage (with 8 symbols) into a directional image with 4 symbols is as simple as the elimination of one of the bits, in particular, the last bit. So, the pixels of the new smoothed directional image are represented by only 2 bits.

Besides, only 17 of the 9x9 symbols have to be checked to identify the patterns. Fig. 4 illustrates the pattern of a convex core point. To allow processing the 17 symbols in parallel, 4 2-bit buffers of two lines, 600 words, and several 2-bit delay registers are necessary. After that, only some comparisons have to be carried out to detect the patterns. The RTL scheme of this block is shown in Fig. 5.

D. Post-processing of singular point detection

Post-processing is performed to detect false positives and thus determine the exact location of true singular points. This is done by evaluating several heuristic rules.

The output of this block is the row and column where the singular point is detected. Therefore, two counters (one for counting the rows and another for the columns) have also been included. The RTL scheme of this block is shown in Fig. 6.

III. FPGA IMPLEMENTATION

The architecture described above can be implemented in ASICs or FPGAs. Here, implementation results on FPGAs from Xilinx are shown. The ISE Project Navigator tool from Xilinx has been used. The architecture has been described in Verilog HDL language. Each RAM memory used as a buffer has been implemented with the CORE Generator tool.

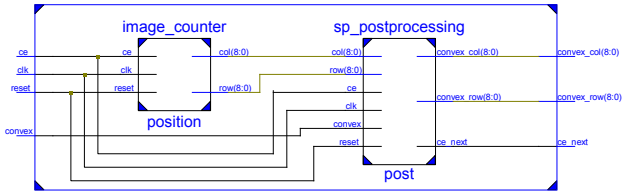


Figure 6. RTL scheme of the post processing block

Table II
IMPLEMENTATION RESULTS VIRTEX 5-XC5VLX50T (SLICES: 7200, SLICE REGISTERS: 28800 AND SLICE LUTS: 28800)

	Simulink HDL Coder implementation				Dedicated hardware IP module			
	Occupied slices	Slice registers	Slice LUTs	RAM Kbits	Occupied slices	Slice registers	Slice LUTs	RAM Kbits
Directional image	2.83 %	1.32 %	1.47 %	0	0.97 %	0.51 %	0.67 %	36
3x3 Smoothing	2.21 %	1.1 %	0.59 %	0	1.87 %	0.31 %	0.95 %	36
9x9 Smoothing	8.06 %	4.34 %	4.01 %	0	4.18 %	1.52 %	1.52 %	72
27x27 Smoothing	28.82 %	10.88 %	18.85 %	0	5.51 %	1.16 %	3.5 %	576
SP detection	4.03 %	3.30 %	0.76 %	0	1.19 %	0.54 %	0.44 %	72
post-SP	2.67 %	0.38 %	1.28 %	0	0.62 %	0.25 %	0.38 %	0

Each block has been synthesized and implemented separately to verify them individually in a Virtex 5-xc5vlx50t-3ff1136. The occupation results for each block are presented in Table II. They are compared with the results of the same implementation described in high level code and translated to HDL language with the HDL Coder tool integrated into Matlab&Simulink as presented in [7]. The HDL Coder tool was selected due to its versatility in the selection of the target device, FPGA or ASIC (System Generator tool, for example, is adequate only for FPGAs from Xilinx). The main difference of both designs is that the dedicated hardware IP module makes use of dedicated RAMs available in the FPGAs. Hence the percentage of occupied slices, slice flip-flops and slice LUTs is reduced considerably in all the blocks. The synthesis of the complete proposed module uses only 12.6 % of the occupied slices, 5.48 % of slice registers, 6.83 % of slice LUTs and 792 Kbits of RAM. This makes it possible implementations of the proposed IP module in simpler FPGAs such as the Spartan 6-sp6lx25-2csg324, as shown in Table III. In this platform, the Simulink HDL Coder design has not slices enough to be implemented. The maximum frequency of operation is limited by the access time to the memory. The bottleneck is the Smoothing 27x27 block, since the maximum frequency of this block is the maximum frequency of the system, which is over 118 MHz in both FPGAs. Nevertheless, such frequency allows processing 118 megapixels per second, which meets the constraints for real-time operation. For example, the time to process a whole fingerprint image of 440x300 pixels is 1.12 ms at 118 MHz and 2.64 ms at 50 MHz. In the latter case, and considering the Spartan-6 FPGA, the total power consumption estimated by XPower is 77 mW.

Table III
IMPLEMENTATION RESULTS SPARTAN 6-SP6LX25 (SLICES: 3758, SLICE REGISTERS: 30064 AND SLICE LUTS: 15032)

	Dedicated hardware IP module			
	Occupied slices	Slice registers	Slice LUTs	RAM Kbits
Directional image	1.92 %	0.49 %	1.08 %	36
3x3 Smoothing	2.26 %	0.33 %	1.46 %	36
9x9 Smoothing	6.23 %	2.57 %	2.5 %	72
27x27 Smoothing	8.49 %	1.52 %	5.54 %	576
SP detection	1.62 %	0.44 %	0.59 %	72
post-SP	0.53 %	0.12 %	0.44 %	0
Total	22.19 %	5.6 %	10.62 %	792

IV. CONCLUSIONS

A dedicated hardware IP module for extracting singular points of fingerprints at real-time has been developed. The solution is based on four main steps that are translated into four hardware blocks in the architecture. The architecture has been described in Verilog HDL language instead of using high-level synthesis tools and implemented in several FPGAs from Xilinx. Since the design takes advantage of dedicated RAMs available in FPGAs, the number of slices required is small, compared to an implementation performed by using the high level synthesis tool HDL Coder. The resources consumption is low, which is very interesting to implement fingerprint recognition algorithms in hardware devices such as FPGAs or ASICs.

ACKNOWLEGMENT

This work was partially supported by TEC2011-24319 and INNPACTO IPT-2012-0695-390000 projects from the Spanish Government (with support from the PO FEDER-FSE). M.C. Martínez-Rodríguez is supported by FPI fellowship program for Ph.D. Students from Spanish Government. P. Brox is supported by 'V Plan Propio de Investigación' from the University of Seville.

REFERENCES

[1] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. London, United Kingdom: Springer-Verlag, 2009.

[2] M. Fons, F. Fons, E. Cantó, and M. López, "FPGA-based personal authentication using fingerprints," *Journal of Signal Processing Systems*, vol. 66, no. 2, pp. 153–189, 2012.

[3] C. Militello, V. Conti, J. Sorbello, and S. Vitabile, "A novel embedded fingerprint authentication system based on singularity points," *Int. Conf. on Complex, Intelligent and Software Intensive Systems (CISIS)*, pp. 72–78, 2008.

[4] P. Grother, W. Salamon, C. Watson, M. Indovina, and P. Flanagan, "MINEX II – Performance of Fingerprint Match-on-Card Algorithms Phase II Report," National Institute of Standards and Technology (NIST), Tech. Rep. 7477, Feb. 2008.

[5] N. Ratha, J. Connell, and R. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Systems Journal*, vol. 40, no. 3, pp. 614 – 634, 2001.

[6] R. Arjona, A. Gersnoviez, and I. Baturone, "Fuzzy models for fingerprint description," in *Fuzzy Logic and Applications*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6857, pp. 228 – 235.

[7] R. Arjona and I. Baturone, "A hardware solution for real-time intelligent fingerprint acquisition," *Journal of Real-Time Image Processing*, vol. 9, no. 1, pp. 95 – 109, 2014.