A Digital Circuit for Extracting Singular Points from Fingerprint Images

Rosario Arjona and Iluminada Baturone
Electronics and Electromagnetism Department (University of Seville)
Microelectronics Institute of Seville (IMSE-CNM-CSIC)
Seville, Spain
{arjona, lumi}@imse-cnm.csic.es

***Abstract*—Since singular point extraction plays an important role in many fingerprint recognition systems, a digital circuit to implement such processing is presented herein. A novel algorithm that combines hardware efficiency with precision in the extraction of the points has been developed. The circuit architecture contains three main building blocks to carry out the three main stages of the algorithm: extraction of a partitioned directional image, smoothing, and searching for the patterns associated with singular points. The circuit processes the pixels in a serial way, following a pipeline scheme and executing in parallel several operations. The design flow employed has been supported by CAD tools. It starts with high-level descriptions and ends with the hardware prototyping into a FPGA from Xilinx.**

***Keywords*— Fingerprint recognition, singular point detection, biometric hardware, FPGAs, CAD tools.**

I. INTRODUCTION

A fingerprint is the reproduction of the exterior appearance of the epidermis [1]. Nowadays, most of fingerprint images are acquired by sensors that provide live scan to obtain directly fingerprint images digitalized by gray levels. Fig. 1(a) is an example of a fingerprint image from the FVC database [2] that has been acquired by an optical sensor. A fingerprint image contains structural characteristics known as *ridges* and *valleys*. In Fig. 1(a), *ridges* are dark and *valleys* are bright. Since the particular formation of ridges and valleys are unique for each fingerprint, they have been employed successfully for biometric recognition purposes. Different features can be extracted from ridge and valley configurations. The most extended local features are the *minutiae*, which are located where the *ridges* end or bifurcate. In the other side, global features give information of the whole fingerprint because they describe relationships between local values [1]. *Singular points* (*core* and *delta*) are examples of global features. They are depicted in Fig. 1(a) by a circle (in the case of the *core point*) and a triangle (the *delta point)*. Singular points have been applied in several stages of fingerprint recognition systems, as the following: (a) to split the fingerprint database into, usually, five classes (*arch*, *tended arch*, *left loop*, *right loop* and *whorl*) in order to reduce the number of comparisons to carry out between the query and the possible candidates [3]; (b) to determine fingerprint alignment by means of a reference system defined by singular points [1]; and (c) to define singular areas (located around singular points) that offer an important amount of distinctive information, as it has been proven in [4].

Most of the solutions reported in the literature to extract singular points are software algorithms implemented on general-purpose processors [5]-[8]. Other solutions, such as [4], translate directly the software algorithm into dedicated hardware. The approach presented herein is to develop a novel algorithm to extract singular points that is more suitable for a hardware implementation, maintaining accuracy in the results.

The paper is organized as follows. Section II reviews the main approaches for detecting singular point and describes the proposed hardware-friendly algorithm. Section III presents the digital circuit developed to implement such algorithm, explaining its architecture and the employed design flow, which is assisted by CAD tools from the high-level description (*Matlab-Simulink*) till device implementation (*Xilinx ISE*). Such methodology facilitates evaluating the accuracy of the circuit in its field of application as well as its performance in terms of area and processing speed. Finally, Section IV shows conclusions of the work.

II. A NOVEL HARDWARE-FRIENDLY ALGORITHM

A. Solutions Reported to Extract Singular Points

Contributions for singular point detection are based on searching for specific patterns of directions within *directional images.* The *directional image* (also called *orientation image, field* or *map, or directional field or map)* is a representation whose

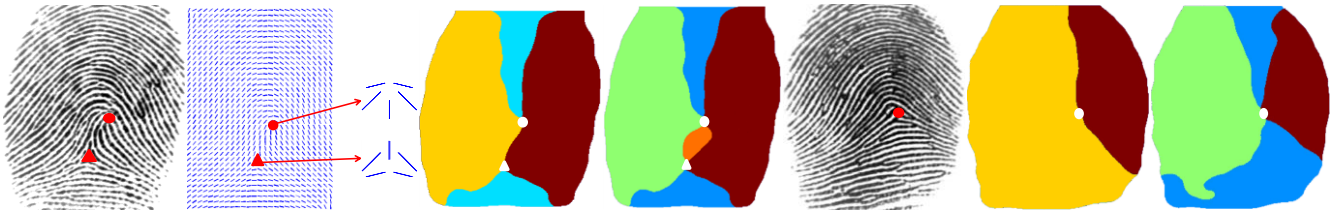

Figure 1. (a) A *right loop* fingerprint [2] with singular points depicted (*core* as a circle and *delta* as a triangle), (b) directional image, (c) patterns of directions for *core* and *delta* points, and directional images partitioned with (d) 3 and (e) 4 homogeneous regions. (f) *Arch* fingerprint [2] with *core point* depicted, and (g) directional images partitioned with (g) 2 and (h) 4 homogeneous regions (in this case, vertical direction does not appear).

elements encode the local directions of the ridges. Fig 1(b) shows the directional image extracted from the fingerprint in Fig. 1(a) by using the algorithm reported in [9]. The approaches reported for computing the directional image are gradient- or mask-based algorithms [10]. In any case, since the resulting directional image contains incorrect directions due to noise or the low definition of ridges in the fingerprint images captured by sensors, a smoothing process is necessary to create homogeneous and continuous regions of directions.

The basic idea to extract singular points is to examine the direction changes in the directional image so as to find direction patterns as illustrated in Fig. 1(c). The reported approaches follow mainly two techniques: (a) computation of indexes that determine discontinuities in the directional image, and (b) techniques based on partitioning the directional image into regions with homogeneous direction values. Examples of the first technique are: (a.1) *Poincaré index* [11], (a.2) *curvature* measurement [5], and (a.3) local histogram of the directional image [6]. They evaluate the change of directions around singular points where there is not a dominant direction. The second technique analyzes the behavior around intersections of homogeneous regions: if direction regions converge, then there is a *core point*; if direction regions diverge, then there is a *delta point* [7]. Since it is possible to find false and isolated singular points, some techniques require an iterative process to ensure the decision. A solution is to employ hybrid approaches that make a first decision by using the information provided by the intersections of homogeneous regions and confirm it with another technique, such as *Poincaré index* [8].

Among fingerprints, *arch* fingerprints do not have any *delta point* and its *core point* is located where the curvature is not high (Fig. 1(f)). For this type of fingerprints, techniques such as *Poincaré index* offer poor results. This is why some solutions distinguish *arch* fingerprints and assign them a special treatment. In fact, the proposal in [8] employs 2 regions for *core point* detection in *arch* classes and 3 regions for the rest of fingerprint classes: singular points in *non-arch* fingerprints are located where the three areas intersect (Fig. 1(d)), while in *arch* fingerprints the points situated along the separation line between the two areas are analyzed to determine which one supposes the maximum change in the direction values (Fig. 1(g)).

B. Proposed solution

A new algorithm more suitable for hardware implementation has been developed. Instead of applying a special procedure for *arch* fingerprints like in [8], a common technique is applied to all types of fingerprints. The first step of the proposal is to generate four homogeneous regions in the directional image because they allow extracting *core points* for all the fingerprint classes (including *arch* classes, as can be seen in Fig. 1(h)). The second step analyzes where the four regions intersect so as to find the particular patterns of directions that show each type of singular point, as depicted in Fig. 1(c).

Partitions in fingerprints are created by distinguishing four clusters in the values of the directions: those corresponding to 0º or 180º, 45º, 90º, and 135º (let us identify them as *c0*, *c1*, *c2* and *c3* clusters, respectively). The values of the directions can be evaluated by computing gradients or using masks. In the case of gradients, the direction value D for a pixel (i, j) is computed as:

$$D(i,j) = \frac{\pi}{2} + \tan^{-1}\left(\frac{G_y(i,j)}{G_x(i,j)}\right) \qquad (1)$$

where G_x and G_y are the horizontal and vertical gradients at each pixel, which can be evaluated by using different filters or operators.

Since gradients provide more accurate results for the direction values than masks, the proposed algorithm employs gradients because the coarse estimation of directions given by masks combined with clustering provide worse results. Instead of using filters to compute gradients, *Sobel* operators have been employed because they are one of the simplest operators for edge detection (in this case, for ridge detection in fingerprints) and their coefficients are integers. Horizontal and vertical gradients (G_x and G_y) are obtained after the convolution of windows centered at each pixel of the image with the horizontal

TABLE I
Processing to obtain clusters from gradient values

IF	Gradient values	Previous quadrant	THEN
1)	G_x=0 OR G_y/G_x>=2.413	-	cluster *c0*
2)	G_y=0 OR G_y/G_x<0.414	-	cluster *c2*
3)	G_x=G_y OR	1st and 3rd	cluster *c1*
4)	(G_y/G_x>0.414 AND G_y/G_x<2.413)	2nd and 4th	cluster *c3*

Firstly, G_x and G_y are converted to the first quadrant

and vertical *Sobel* matrices. Instead of calculating the direction values (using inverse trigonometric functions as in (1)) and subsequently applying a clustering (replacing each value of the directional image by the most similar cluster value), the proposed algorithm simplifies this processing to just evaluating the relation between the values of G_x and G_y to identify the cluster which the pixel belongs to. Once G_x and G_y are converted to the first quadrant, the processing to implement is detailed in Table I, which is very much simpler than computing the arctangent function. In fact, this processing can be seen as an already clustered computation of the arctangent function.

Smoothing is applied after directional image is partitioned into clusters to obtain homogeneous regions. A simple smoothing process considers the neighboring pixels inside a window centered at the analyzed pixel and assigns to the pixel the cluster value with the highest number of occurrences inside the window. Finally, the last step of the algorithm is to evaluate if the pattern of a singular point is found at a window centered at the analyzed pixel.

The algorithm has been described in *Matlab* language and has been verified with public and large databases of the Fingerprint Verification Competition (in particular, with FVC 2002 DB1 [2] and FVC 2006 DB3 [12]).

III. DIGITAL IMPLEMENTATION

A. *Architecture of the Processing Circuit*

The developed implementation processes the image in a serial way. Once fingerprint image is captured and stored in a memory, a serialize block reads the data and provides pixels one by one to the next block. The architecture contains three main blocks with the following functionality: extraction of the clustered directional image, smoothing of the clustered directional image, and singular point detection (SP detection). A pipeline scheme is employed where each functional block processes (mainly in parallel) the pixels of the window centered at each analyzed pixel.

The block in charge of computing the clustered directional image applies *Sobel* masks for a 3x3 window. Convolution involving the 9 pixels is performed in parallel once the 9 pixels are available. Two line buffers and two delays for each line of the image are considered for this purpose. The cluster value is selected between the four representative directions accordingly to a combinatorial circuitry that implements the processing shown in Table I.

The smoothing block processes a 9x9 window. For each pixel, it computes the cluster value with the highest number of occurrences between the 81 values inside the window. In order to ease hardware implementations, a 3x3 smoothing is firstly performed in parallel because a 3x3 window is the window size also processed in parallel by the previous block. It returns the winner direction value in a 3x3 window and its number of occurrences. Then, a 9x9 smoothing is an extension composed by nine 3x3 windows, where the winner direction value is computed from the nine previous results (again, the current window is computed in a parallel way). Two line buffers and two delays are required for the 3x3 smoothing, and two 3-line buffers and two delays of three values for each line are required for the 9x9 smoothing.

The block in charge of singular point detection compares the values from smoothed clustered directional image within a 3x3 window according to the defined patterns for each type of singular point. Its hardware implementation is similar to the previous blocks because it employs two line buffers and two delays for each line to implement the comparison of the 9 values in parallel.

The functionality of the complete architecture has been verified at the application level with *Matlab-Simulink*. A dynamic model that includes the functional blocks defined above is shown in Fig. 2. It considers hardware parameters such as the number of bits (data are processed in fixed point), delays, and the line buffers employed by each block. The performance of each block as well as the whole system can be evaluated and compared with the software result for each fingerprint image considered.

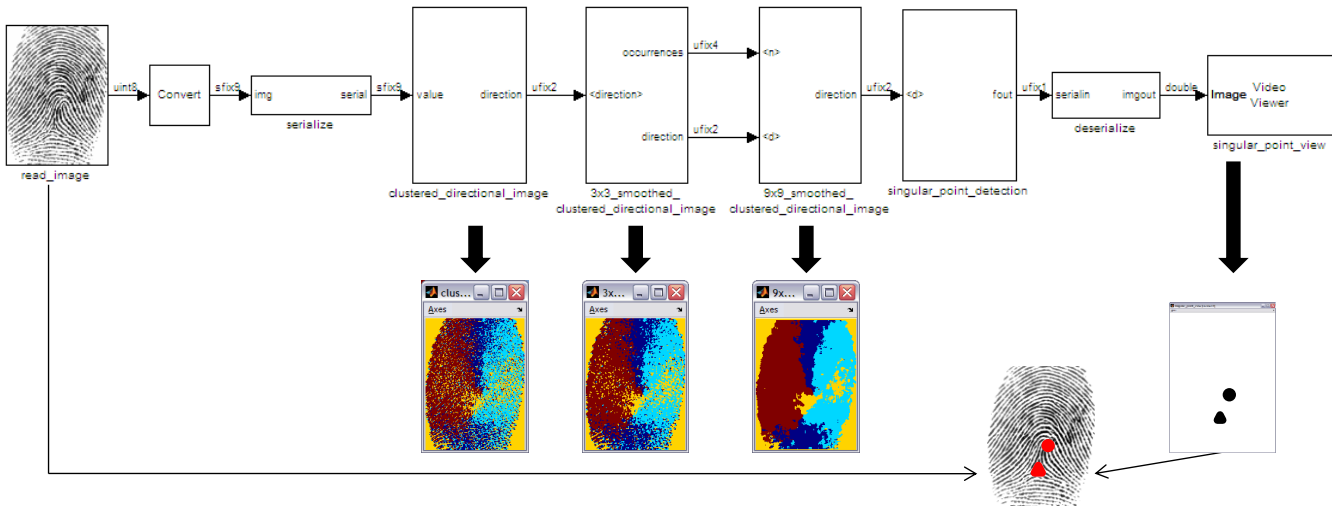

Figure 2. Hardware architecture evaluated in *Matlab- Simulink*.

TABLE II

(a) Number of bits and FPGA occupation for each block

Block	Number of bits	Area (% slices)
Clustering	14 (for gradients) and 2 (for directions)	7
Smoothing 3x3	4 (for the counter) and 2 (for directions)	4
Smoothing 9x9	7 (for the counter) and 2 (for directions)	18
SP detection	2 (for directions) and 1 (for decision)	3

Implementation performed into a Spartan 3A (5888 slices in total)

(b) Timing implementation results for the proposed design

Latency (clock cycle)	Throughput (clock cycle)	Maximum frequency (MHz)
9*C+8	1	264.4

C: number of cols of the fingerprint image

B. FPGA-based Implementation

A top-down design flow has been employed to implement this proposal for singular point detection in hardware. CAD tools from *Matlab-Simulink* and *Xilinx ISE* facilitates the design of a complete FPGA-based hardware prototype. Implementation has been performed in an automatic way because *Simulink HDL Coder* supports HDL code generation, including an associated testbench. *HDL Coder* eases the construction of DSP algorithms on FPGAs or ASICs and reduces the complexity of the design process. The generated *VHDL* or *Verilog* description is independent on the device. In this work, VHDL description has been employed and a *Xilinx Spartan* 3A FPGA has been selected as target device. The CAD tools from *Xilinx ISE* provide implementation details concerning resource utilization and timing. In particular, *Isim* simulator has been used to simulate the circuit described in VHDL code at a hardware level.

Table II(a) shows the number of bits employed by each block and its occupation in percentage of slices. Serialize and deserialize blocks are not synthesized because they are considered as signal conditioning blocks. Although many operations are performed in parallel, only 32 per cent of slices are occupied by the whole circuit. Timing results are viewed in Table II(b). *Latency*, which indicates when the first valid value is obtained, depends on the 9x9 smoothing, which is the slowest operation. *Throughput* is one clock cycle because operations within each block are performed in parallel.

The simplicity of our proposal to calculate the clustered directional image is shown in the first row of Table III(a). If the arctangent function is implemented (using a CORDIC module) and a posterior clustering is applied, the implementation is worse in terms of area and speed (second row of Table III(a)). The proposed design invests 0.39 ms to process a 374x276 fingerprint image (working at the maximum frequency of 264.4 MHz). The complete singular point detection algorithm implemented in [4] invests a time of 31.20 ms to process an image of 560x296 pixels, using a *Xilinx Virtex II* FPGA working at 25 MHz. Table III(b) shows timing and area results for different implementations of directional image extraction (without considering clustering). These solutions employ HW/SW co-design and coprocessors while the proposed implementation focuses on dedicated hardware. From the results in Table III, it can be seen that the proposed solution provides a considerable reduction of complexity together with a high speed.

TABLE III

(a) Comparative of computations for clustered directional image

Proposal	Maximum frequency (MHz)	Number of slices	Minimum execution time (ms)
This proposal	264.4	454	0.39
With CORDIC	81.3	854	1.27

(b) Comparative of directional image implementations

Proposal	Image size (pixels)	Frequency (MHz)	Number of slices	Execution time (ms)
Spartan 3 [13]	512x280	-	2468	12
Altera EPXA10 [14]	516x280	50	9123	25
Spartan 3 [15]	256x256	100	575	262
Altera ByteBlaster [16]	288x224	33	-	1.2

IV. CONCLUSIONS

A hardware implementation for singular point detection has been developed based on a novel and simple algorithm. It has been designed by following a low-cost design flow supported by CAD tools from *Matlab-Simulink* and *Xilinx ISE*, which allows verifying the functionality of the circuitry at the level of application. Its features concerning area occupation and processing speed improve the proposals previously reported in the literature, making it very suitable for embedded and real-time systems. A future work line is to include this circuit in a fingerprint classification and/or verification system.

REFERENCES

[1] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, Handbook of Fingerprint Recognition, 2nd ed., Springer, 2009.

[2] FVC 2002 DB1a, http://bias.csr.unibo.it/fvc2002/

[3] R. Arjona, A. Gersnoviez, and I. Baturone, "Fuzzy Models for Fingerprint Description," 9th International Workshop on Fuzzy Logic and Applications, in press.

[4] C. Militello, V. Conti, F. Sorbello, and S. Vitabile, "A Novel Embedded Fingerprint Authentication System based on Singularity Points", International Conference on Complex, Intelligent and Software Intensive Systems, pp. 72–78, 2008.

[5] S. Mohammadi, and A. Farajzadeh, "Fingerprint Reference Point Detection Using Orientation Field and Curvature Measurements," IEEE International Conference on Intelligent Computing and Intelligent Systems, pp. 25–29, 2009.

[6] V. S. Srinivasan, and N. N. Murthy, "Detection of Singular Points in Fingerprint Images," Pattern Recognition, Volume 25, Issue 2, pp. 139–153, 1992.

[7] T. Ohtsuka, and T. Takahashi, "A New Detection Approach for the Fingerprint Core Location Using Extended Relation Graph," IEICE Transactions on Information and Systems, Volume 88, Issue 10, pp. 2308–2312, 2005.

[8] H. K. Lam, Z. Hou, W. Y. Yau, T. P. Chen, and J. Li, "A Systematic Topological Method for Fingerprint Singular Point Detection," 10th International Conference on Control, Automation, Robotics and Vision, pp. 967–972, 2008.

[9] Directional image algorithm, http://www.csse.uwa.edu.au/~pk/research/matlabfns/

[10] D. Chen, X. Ji, F. Fan, J. Zhang, L. Guo, and W. Meng, "Comparative Analysis of Fingerprint Orientation Field Algorithms," Proceedings of the 5th International Conference on Image and Graphics, pp. 796–801, 2009.

[11] M. Kawagoe, and A. Tojo, "Fingerprint Pattern Classification," Pattern Recognition, Volume 17, Issue 3, 1984.

[12] FVC 2006 DB3a, http://bias.csr.unibo.it/fvc2006/

[13] E. Cantó, M. Fons, M. Lopez, and R. Ramos, "Acceleration of Complex Algorithms on a Fast Reconfigurable Embedded System on Spartan-3," International Conference on Field Programmable Logic and Applications, pp. 429–434, 2009.

[14] F. Fons, M. Fons, E. Cantó, and M. López, "Flexible hardware for fingerprint Image Processing," Research in Microelectronics and Electronics Conference, PRIME, pp. 169–172, 2007.

[15] M. L. Garcia, and E. F. C. Navarro, "FPGA Implementation of a Ridge Extraction Fingerprint Algorithm Based on Microblaze and Hardware Coprocessor," International Conference on Field Programmable Logic and Applications, pp. 1–5, 2006.

[16] G. Chao, S. Lee, H. Lai, and S. Hornq, "Embedded Fingerprint Verification System," Proceedings of the 11th International Conference on Parallel and Distributed Systems, pp. 52–57, 2005.