Model-based Design for Selecting Fingerprint Recognition Algorithms for Embedded Systems

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*—Most of contributions for biometric recognition solutions (and specifically for fingerprint recognition) are implemented in software on PC or similar platforms. However, the wide spread of embedded systems means that fingerprint embedded systems will be progressively demanded and, hence, hardware dedicated solutions are needed to satisfy their constraints. CAD tools from *Matlab-Simulink* ease hardware design for embedded systems because automatize the design process from high-level descriptions to device implementation. Verification of results is set at different abstraction levels (high-level description, hardware code simulation, and device implementation). This paper shows how a design flow based on models facilitates the selection of algorithms for fingerprint embedded systems. In particular, the search of a solution for directional image extraction suitable for its application to singular point extraction is detailed. Implementation results in terms of area occupation and timing are presented for different *Xilinx* FPGAs.**

Embedded Systems; Automated Hardware Design; Fingerprint Biometrics; Directional Image Extraction; Singular Point Extraction.

I. INTRODUCTION

Most of biometric solutions are implemented in software running on PC or similar platforms. However, the wide spread of embedded systems in current life style is requiring efficient implementations of biometric techniques in terms of area occupation, power consumption, and processing speed. There are several ways to implement biometric algorithms in embedded systems. The simplest one is to write the C code and compile it into a general-purpose processor included in the embedded system. However, biometric processing tasks, such as those employed for fingerprint recognition, usually require many operation cycles when they are performed sequentially. Hence, either the processing speed of the system is low or the cost is high (if high-speed power-hungry or complex processors are employed). Another solution is to use specialized blocks implemented in hardware to accelerate the tasks of the general-purpose processor. In most cases, these solutions usually resort to the use of hardware-software co-design. The solution that features the most reduced figures of area, power, and processing time is to implement the whole algorithm in hardware.

The drawback of hardware solutions, which should consider arithmetic and temporization issues not contemplated by software, is a higher design time. Generally, neither software (application) engineers take into account specific hardware features nor hardware engineers evaluate repercussion of their decisions on the application. On the one hand, software engineers develop complex biometric techniques which provide high accurate systems. On the other hand, hardware engineers find difficulties to translate these techniques to a hardware implementation and opt to approximate the solutions. This is one reason that increases the design cost of hardware solutions. Both worlds should be unified so as to obtain more efficient solutions and model-based techniques can help in this objective while maintaining the advantages, as will be shown in this work. A way to reduce the design cost is to employ CAD tools that automate and ease the hardware design from high-level software-like descriptions.

Nowadays, *Mathworks* is promoting *HDL Coder*, a tool which aims to facilitate hardware designs for any type of device (ASIC or FPGA) since it is possible to generate synthesizable HDL code from *Matlab* or *Simulink* and this code is independent on the device. Although the hardware synthesis is not completely optimized, the HDL code generated can be modified by hand if it is required. However, for most works these tools provide a good trade-off between low-cost hardware design and device implementation results in terms of area occupation and timing. In addition, *HDL coder* includes an interesting function named *FPGA-in-the-loop* that allows verifying high-level designs in hardware platforms. That is, the design at a model level in *Simulink* can be verified at a hardware level from the same *Simulink* environment. This accelerates the design process because hardware results are obtained straight off from the high-level description and it is verified that the design actually implements what has been simulated. If the results from the device are not correct for the application which is being developed, the designer can change easily the algorithms at a high level and starts again with the *FPGA-in-the-loop* process.

Model-based techniques are being widely applied in the field of control domain, and they are also attracting interest in the field of signal processing. This work shows how they can be very useful to implement fingerprint-based biometric applications. The paper is organized as follows. Section II

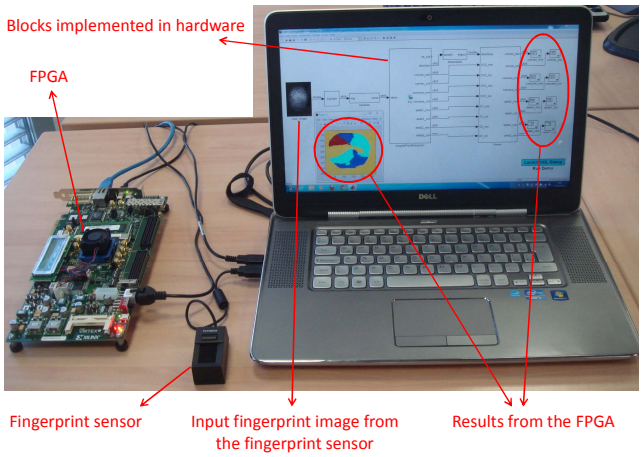


Fig. 1: An example of *FPGA-in-the-loop* functionality.

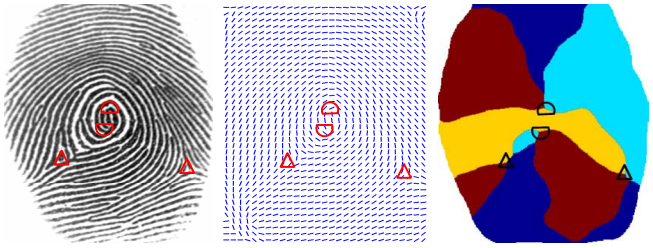
describes the CAD tools employed for automated hardware design from high-level descriptions to device implementation defining a hardware design flow. Two case studies are analyzed in Section III to illustrate how fingerprint processing algorithms can be selected taking into account software accuracy and hardware constraints at the same time. Firstly, different proposals for directional image extraction are evaluated from the most accurate solution to a low-cost approach. Then, a second example shows the influence of the previous solutions in the extraction of singular points (which in turn depends on the directional image extraction). The complete design flow employed is briefly described and it is illustrated how implementation results in terms of area occupation and execution time can be easily obtained. Conclusions and future work lines are given in Section IV.

II. A HARDWARE DESIGN FLOW USING CAD TOOLS

As commented in Introduction, CAD tools facilitate the hardware design for biometric solutions. We refer to automated design processes which create top-down design flows (from high-level descriptions to device implementations). Basically, the designer only develops the high-level description and is in charge of checking that the behavior results coincide for the software and hardware implementations. Those stages related to hardware functions (HDL code generation, hardware code simulation, hardware synthesis, device implementation, and implementation verification at device level) are performed in an automatic way by using the corresponding tools.

Biometric applications for fingerprint recognition employ fingerprint images as individual characteristics and thus they are based on image processing. In this respect, *Matlab* is a very suitable tool for image processing because it is optimized to work with matrices (images, in our case) and accelerates all type of operations related. In addition, *Simulink*, as interactive graphical environment, is very useful to model, analyze and simulate systems that work with fingerprint images.

The first step to develop fingerprint recognition embedded systems starts with high-level descriptions in *Matlab-Simulink*. Floating-point software-based implementations can be



(a) (b) (c)

Fig. 2: (a) Fingerprint image from [1], (b) Directional image associated, and directional image segmented using four homogeneous direction regions. All representations have singular points depicted.

compared to fixed-point hardware-based implementations. Thus, the influence of hardware features in the system performance can be evaluated.

Once high-level descriptions are completed, it is necessary to obtain the associated HDL code. In the context of *Matlab-Simulink*, this can be performed in an automatic way using *HDL Coder*, which lets generate VDHL or *Verilog* code from *Matlab* or *Simulink* descriptions. The resulting synthesizable code is independent on the device because it can be mapped for ASICs or FPGAs. Also, it is possible to generate testbenches which can be employed to simulate and verify the circuit at hardware code level. *ISE Isim* (for FPGAs) and *Mentor Graphics ModelSim* (for FPGAs and ASICs) provide support for simulations. This constitutes another verification point whose results can be compared to the results from the high-level descriptions.

The next stage of the design flow is the device implementation. For implementations based on *Xilinx* FPGAs, tools from *Xilinx* ISE environment complete the process.

Latest versions of *Matlab-Simulink* include workflows to carry out the complete design process from the same environment. That means that high-level descriptions, HDL code generation, behavior simulation and implementation into a FPGA device can be performed from the same *Matlab-Simulink* environment because the corresponding tools communicate to *Matlab-Simulink*. Moreover, *FPGA-in-the-loop* functionality allows viewing results directly from the device. It is another verification level in addition to behavior simulations from floating-point, fixed-point and testbench descriptions. This is an important issue in embedded system design because functional results from the FPGA device implementation are verified before the final system is manufactured.

In this work, we perform the complete design process from high-level descriptions (where the most suitable algorithm for a hardware implementation is selected) to device implementation and verification (using the *FPGA-in-the-loop* functionality). Fingerprint images from standard and large databases [1] are considered. Also, a fingerprint sensor is employed and performance of the system is evaluated by using fingerprint images captured in live from the individual. Fig. 1 shows an example of *FPGA-in-the-loop* functionality to extract singular points, as will be described in Section III.B. In this case, fingerprints are captured by the FS90 optical sensor

from *Futronic* [2] and results are obtained from a *Xilinx Virtex-6 FPGA ML605* board.

III. CASE STUDIES

A. Evaluation of algorithms for directional image extraction

Ridges in fingerprint images (depicted with dark color in Fig. 2(a)) are structural characteristics whose local directions compose a matrix named *Directional Image* (*Orientation Image, Field* or *Map, or Directional Field or Map*). Fig. 2(b) is the directional image extracted from fingerprint in Fig. 2(a). Directional image gives global information of a fingerprint and plays an important role in fingerprint recognition. It is used in several authentication stages, from fingerprint acquisition to matching stage. Direction values offer information for enhancement and segmentation of fingerprint images [3] before feature extraction; singular points extraction [4], which are significant points in fingerprints; alignment process [5] necessary to correct placement when fingerprint is captured with different translation and rotation; and matching stage using directional features such as *Fingercodes* [6]. Depending on the application, the algorithm to extract the directional image has to be more or less accurate.

Most popular approaches for computation of directional image are *gradient-based* or *mask-based algorithms* [7]. Since gradients are the most natural and accurate technique, let us take into account gradient-based algorithms. Gradient operators (*Gaussian*, *Sobel*, or *Prewitt*) are usually employed to compute gradients. Once convolutions with the operator windows are applied, horizontal and vertical gradients (G_x and G_y) are obtained. The ridge edge for the pixel *(i, j)* is orthogonal to the gradient values and the direction value *D* for a pixel (*i, j*) is computed as:

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

Drawbacks of this technique are non-linearity, discontinuities and sensitivity to the noise in fingerprint images [8]. To overcome these problems, the fingerprint image is enhanced, and gradient values are combined, averaged, and smoothed. A software implementation in *Matlab* for the complete directional image is available in [9].

The required accuracy in directional image depends on the final application. To create a suitable hardware implementation which satisfies constraints in embedded systems, simplifications have been considered. The first decision is not to implement enhancement and smoothing processes because these operations can be performed by previous or subsequent stages if the recognition application requires them. In addition, 3x3 *Sobel* operators are employed since they are ones of the simplest operators for edge detection (in this case for ridge detection in fingerprints) carrying out convolutions with integer values. Another way to reduce the complexity is to approximate the arctangent function by means of CORDIC (*Coordinate Rotation Digital Computer*) method, which is an efficient method to compute trigonometric functions in hardware.

TABLE I

(a) Comparison of results for directional image extraction

Proposal	Maximum frequency (MHz)	Slices (%)	Minimum execution time (ms)
With CORDIC	81.3	15	1.27
With clustering	264.4	8	0.39

Implementation performed into a Xilinx Spartan-3A FPGA for a fingerprint image with 374x276 pixels

(b) Comparison of results for smoothing approaches

Proposal	Maximum frequency (MHz)	Slices (%)	Minimum execution time (ms)
Complete	582.8	50	0.25
Simplified	582.8	32	0.25

Implementation performed into a Xilinx Virtex-5 FPGA for a fingerprint image with 374x388 pixels

(c) Complete and simplified smoothing for different image sizes

Proposal (RxC)	Complete (96x96)	Simplif. (96x96)	Complete (236x192)	Simplif. (236x192)	Complete (374x388)	Simplif. (374x388)
Slices (%)	17	13	27	19	50	32

Implementation performed into a Xilinx Virtex-5 FPGA
RxC: number of rows x number of columns in image

This description is performed as a *Simulink* model in floating-point data. After the verification of this behavior model, a hardware model (in fixed-point data) is developed. Data are computed in a serial way to simulate how pixels are received from a fingerprint sensor and the convolutions of *Sobel* operators are based on buffers and delays to consider previous values in the sequence of pixels. Conversions of data types and rates are required by the inputs of a *CORDIC block*, whose function is to receive G_x and G_y values and return direction values. It is a predefined *Simulink* block which implements *arctangent* function using *phase* output. Results from both models can be compared and hardware considerations can be evaluated.

Once simulated at high level, *Simulink HDL Coder* tool is used to generate HDL code from the *Simulink* model. At the end of this stage, we have a hardware description of the directional image computation and its associated testbench for simulations.

The target platform selected is a *Spartan-3A* FPGA from *Xilinx* and the implementation process is completed by ISE environment. The automatic testbench generated is simulated by *Isim* tool which is another verification point for the hardware description. Implementation results are shown in the first row of Table I(a).

B. Evaluation of algorithms for singular point extraction

Previous section evaluates an alternative to extract the directional image. However, selecting a proposal is dependent on the application. As mentioned, directional image can be employed for singular point extraction. *Singular points* are central features of fingerprint images created by the ridge lines [8]. There are two types of singular points: *core points* (located where ridge lines have maximum curvature), and *delta points* (situated where three ridge lines intersect). Core points can be convex or concave, depending on the orientation of the ridges. In Fig. 2 *convex core points* are depicted as semi-circles facing upwards, *concave core points* as semi-circles facing downwards, and *delta points* as triangles.

The techniques reported in the literature to detect singular points are devoted to search abrupt changes of direction values in directional images. Let us focus on the contribution in [10], which proposes an approach for singular point extraction that is suitable for hardware implementations because it is based on coarse representations of the directional image. Directional image is clustered in homogeneous direction regions and singular points are located where regions intersect. Proposal in [10] concludes that four direction values (0º, 45º, 90º and 135º) are necessary to generate an associated representation for all fingerprint classes for the purpose of singular point extraction. The representations for directional image discussed in the previous subsection were composed by continuous values (from 0º to 180º). Instead of applying arctangent function (or its approximation with CORDIC function) and subsequently applying a clustering (replacing each value of the directional image by the most similar cluster value), the proposed algorithm simplifies the processing to just evaluating the relation between the values of G_x and G_y to identify the cluster which the pixel belongs to. That is, once G_x and G_y are calculated, several logical conditions are considered to assign a cluster value. Thus, hardware description for clustering is done as described in the previous subsection but without considering CORDIC block. Reduction of complexity in the computation of arctangent function is translated to a simple hardware, as shown in the second row of Table I(a).

After clustering, a simple smoothing process is necessary to group direction values into homogeneous regions. Direction clusters from the neighboring pixels are considered centering a window at the analyzed pixel and assigning to it the cluster value with the highest number of occurrences inside the window. A 27x27 smoothing window is suitable for the most of fingerprint images. To ease hardware implementations, a 3x3 smoothing is firstly performed in parallel. It returns the number of occurrences for each cluster value in a 3x3 window. Then, a 9x9 smoothing is an extension composed by nine 3x3 windows which sums the number of occurrences for each cluster value computed from the nine previous results (again, the current window is processed in parallel). And, finally, a 27x27 smoothing is the sum in parallel of results from an extension composed by nine 9x9 windows and the selection of the cluster value with the highest number of occurrences. A possible simplification to this approach (with the consequent reduction of complexity) is to consider the winner direction value and its number of occurrences within each window (simplified smoothing), instead of computing the number of occurrences for all the four cluster values (complete smoothing). Implementation results for each smoothing approach using the design flow presented and employing a *Xilinx Virtex-5* FPGA are displayed in Table I(b). Since the models are parameterized, implementation can be evaluated for different image sizes as illustrated in Table I(c). Area occupation results demonstrate the complexity of the complete smoothing with respect to the simplified smoothing.

Finally, the last step of the algorithm consists of locating where direction regions intersect and determine the type of singular point. The design methodology using *FPGA-in-the-loop* allows evaluating the different solutions in an easy way. An example of the obtained results is given in Table II. Such results can be obtained and evaluated automatically for a wide set of fingerprints to conclude if the accuracy in directional image extraction or smoothing is enough for singular point extraction, which allows increasing considerably the hardware performance (in terms of resource consumption and speed).

TABLE II
Singular points extracted for the fingerprint in Fig. 2(a) using different approaches

PROPOSAL	CVC	CCC	D1	D2
I	148,142	178,127	283,49	302,221
II	148,140	177,128	284,41	301,221
III	151,141	177,127	285,47	300,220

Proposal I: [9] with clustering; Proposal II: arctangent approximated with clustering and smoothing considering number of occurrences for all cluster values; Proposal III: arctangent approximated with clustering and smoothing considering number of occurrences for only the winner direction.

The results named as CVC, CCC, D1, and D2 are the singular point locations in fingerprint images (expressed in rows and columns) that have been extracted for convex core, concave core, delta1 and delta2 points, respectively.

IV. Conclusions

Model-based techniques using *Matlab-Simulink* CAD tools allow evaluating different approaches for fingerprint algorithms in an easy way. Such evaluation is basic to take good design decisions that result in hardware implementations with better features of area, power or speed. This has been illustrated with the implementation of algorithms for directional image extraction from fingerprint images and its subsequent application to singular point extraction.

Acknowledgment

This work was partially funded by Spanish Ministerio de Economía y Competitividad under the Project TEC2011-24319 and Junta de Andalucía under the Project P08-TIC-03674 (both with support from FEDER), and by the European Community through the MOBY-DIC Project FP7-INFSO-ICT-248858 (www.mobydic-project.eu).

References

[1] FVC fingerprint database, http://bias.csr.unibo.it/fvc2002/

[2] Fingerprint sensor, http://www.futronic-tech.com/product_fs90.html

[3] L. Hong, Y. Wan, A. Jain, "Fingerprint Image Enhancement: Algorithm and Performance Evaluation," IEEE Transactions on Pattern Analysis and Machine Intelligence, 20, 8, pp. 777-789, 1998.

[4] M. Kawagoe, A. Tojo, "Fingerprint Pattern Classification," Pattern Recognition, 17, 3, pp. 295-303, 1984.

[5] N. Yager, A. Amin, "Evaluation of Fingerprint Orientation Field Registration Algorithms," Proceedings of the 17th International Conference on Pattern Recognition, 4, pp. 641-644, 2004.

[6] S. Prabhakar, "Fingerprint Classification and Matching using a Filterbank," Thesis, 2000.

[7] D. Chen, X. Ji, F. Fan, J. Zhang, L. Guo, W. Meng, "Comparative Analysis of Fingerprint Orientation Field Algorithms," Fifth International Conference on Image and Graphics, pp. 796-801, 2009.

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

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

[10] R. Arjona, I. Baturone, "A Digital Circuit for Extracting Singular Points from Fingerprint Images," Proceedings of the IEEE 18th International Conference on Electronics, Circuits, and Systems (ICECS'2011), pp. 627-630, 2011.