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
Floating-point software-based implementations can be systems starts with high-level descriptions in graphical environment, is very useful to model, analyze and type of operations related. In addition, to work with matrices (images, in our case) and accelerates all very suitable tool for image processing because it is optimized are based on image processing. In this respect, 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.

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 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 of 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 manufactured. 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. Directional images 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:

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

(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, $3 \times 3$ 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.

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.

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Maximum frequency (MHz)</th>
<th>Slices (%)</th>
<th>Minimum execution time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>With CORDIC</td>
<td>582.8</td>
<td>17</td>
<td>1.27</td>
</tr>
<tr>
<td>With clustering</td>
<td>264.4</td>
<td>13</td>
<td>0.39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Maximum frequency (MHz)</th>
<th>Slices (%)</th>
<th>Minimum execution time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>582.8</td>
<td>17</td>
<td>0.25</td>
</tr>
<tr>
<td>Simplified</td>
<td>582.8</td>
<td>13</td>
<td>0.25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Proposal</th>
<th>Maximum frequency (MHz)</th>
<th>Slices (%)</th>
<th>Minimum execution time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>582.8</td>
<td>17</td>
<td>0.25</td>
</tr>
<tr>
<td>Simplified</td>
<td>582.8</td>
<td>13</td>
<td>0.25</td>
</tr>
</tbody>
</table>

(a) Comparison of results for directional image extraction

(b) Comparison of results for smoothing approaches

(c) Complete and simplified smoothing for different image sizes

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

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

Implementation performed into a Xilinx Virtex-5 FPGA

RxC: number of rows x number of columns in image

TABLE I
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).

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-INFOSS-248858 (www.mobydic-project.eu).

REFERENCES