Microelectronics Implementation of Directional Image-based Fuzzy Templates for Fingerprints

Rosario Arjona, Iluminada Baturone,
Depto. Electrónica y Electromagnetismo, Univ. de Sevilla
IMSE-CNM-CSIC, Seville, Spain
{arjona, lumi }@imse-cnm.csic.es

Santiago Sánchez-Solano
IMSE-CNM-CSIC, Seville, Spain
santiago@imse-cnm.csic.es

***Abstract*—Fingerprint orientation image, also called directional image, is a widely used method in fingerprint recognition. It helps in classification (accelerating fingerprint identification process) as well as in preprocessing or processing steps (such as fingerprint enhancement or minutiae extraction). Hence, efficient storage of directional image-based information is relevant to achieve low-cost templates not only for "match on card" but also for "authentication on card" solutions. This paper describes how to obtain a fuzzy model to describe the directional image of a fingerprint and how this model can be implemented in hardware efficiently. The CAD tools of the Xfuzzy 3 environment have been employed to accelerate the fuzzy modeling process as well as to implement the directional image-based template into both an FPGA from Xilinx and an ASIC.**

***Index Terms*— Fingerprint recognition, directional image, fuzzy modeling, biometric hardware, FPGAs, ASICs, CAD tools.**

I. Introduction

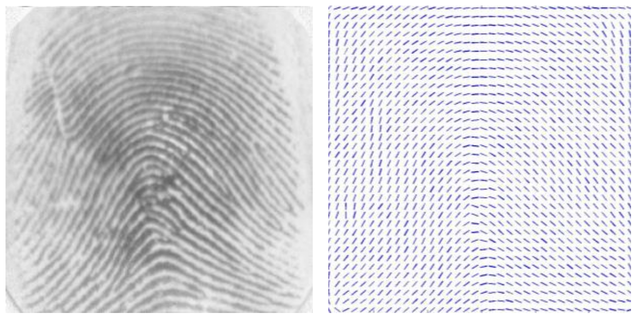
A fingerprint is the reproduction of the exterior appearance of the epidermis and its structural characteristics are *ridges* and *valleys*. In a fingerprint image (Fig. 1a), *ridges* are dark and *valleys* are bright. These structural characteristics can be used to obtain global descriptions, as those provided by directional image, which is a matrix whose elements encode local directions of ridges [2]. Fig. 1b shows the directional image of the fingerprint in Fig. 1a.

Directional image has been proven very useful to accelerate fingerprint identification, since it allows creating patterns to split the fingerprint database and reduce the number of comparisons to carry out between the query and the possible candidates. Directional image also offers suitable information for several preprocessing stages (such as segmentation and posterior enhancement of the fingerprint image), and it is helpful in processing stages to find singular points as well as minutiae. In addition, it provides relevant information to realize alignment, which is crucial in the matching stage [3].

Consequently, storing the directional image associated to each fingerprint can be very interesting as part of the template (information) stored during the enrollment of each user. Templates should be stored in a compact way: In the case of identification applications (find one user among many ones) because the number of templates (users) can be very high; In the case of authentication (one to one comparison) because many application scenarios involve embedded systems (such as smart cards) with constrained resources in terms of area and power. A directional image has a size equivalent to the total number of pixels in the fingerprint image, since each pixel has associated a direction. To reduce this size, the block directional image (a down sampling of the directional image) is normally used instead. However, this is more discontinuous and should be smoothed adequately [3].

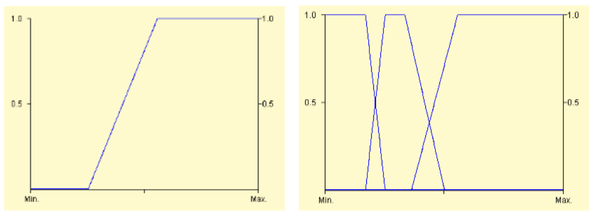
The alternative proposed in this paper is to consider a fuzzy rule base capable of inferring a smooth directional image from little information. Each rule describes the approximated direction dominating a fuzzy region of the fingerprint. Fuzzy modeling allows interpolating among a few rules instead of crisp descriptions that are costly (equivalent to one rule per pixel) or discontinuous (equivalent to one rule per block).

This paper is organized as follows. Section II summarizes how fuzzy modeling can be performed by the CAD tools of *Xfuzzy 3* [4], a design environment developed at the Microelectronics Institute of Seville and University of Seville. Section III presents the design methodology applied to implement in dedicated the template obtained with a fuzzy representation of the directional image. Two different design flows are illustrated: implementations based on Xilinx FPGAs and based on ASICs. Finally, Section IV shows conclusions of the work.



(a) (b)

Fig. 1. (a) Fingerprint image from FVC 2000 database [1]. (b) Directional image associated.



(a) (b)

Fig. 2. (a) An example of fuzzy set and its membership function. (b) Fuzzy sets for *x position*: *Left*, *Middle* and *Right*.

II. Fuzzy Modeling

A. *Fuzzy Systems*

A fuzzy system employs a knowledge base with *IF-THEN* rules that contain variables represented by fuzzy sets instead of crisp values. Fuzzy Logic allows translating knowledge expressed in a linguistic form and, vice versa, allows extracting linguistic knowledge from a rule base [5]. An example of a fuzzy rule describing part of the directional image in Fig. 1b is the following:

'IF the pixel is placed *in the middle* (regarding x position) AND *at the upper* part of the image (regarding y) THEN direction is *almost horizontal*'

Where *in the middle* and *at the upper* are fuzzy sets described by membership functions that assign a membership degree between 0 and 1 (Fig. 2) to locations (x, y) in the image. Meanwhile, *almost horizontal* is a fuzzy set describing a direction that is approximately zero.

Performance of fuzzy inference depends on the membership functions employed and the operators used for: (a) connecting antecedents (AND, in the example), (b) implication (of antecedents on consequents), and (c) aggregation (which combines implication results). The best inference mechanism for hardware implementation employs piecewise linear membership functions (triangular or trapezoidal) that overlap among them so that the sum of the membership degrees for each input is always one, what is known as a partition of unity (as in Fig. 2b). The antecedent connection as well as implication is represented by a product, and aggregation is done by a weighted average, for example Fuzzy Mean, where the weights are the activation degrees of the rules and the values weighted are the consequents of the rules (their most representative values, if they are fuzzy) [6]. Using partition of unity, division is not required, so that weighted average reduces to a weighted sum. In the case of modeling the directional image, overlapping between the input membership functions produces that 4 rules with different directions in their consequents can be active for each pixel location in the image. The fuzzy model provides a smooth interpolation among the 4 rules. The model can be finer or coarser depending on the number of membership functions covering x, y, and directions.

B. *Xfuzzy 3 Tools*

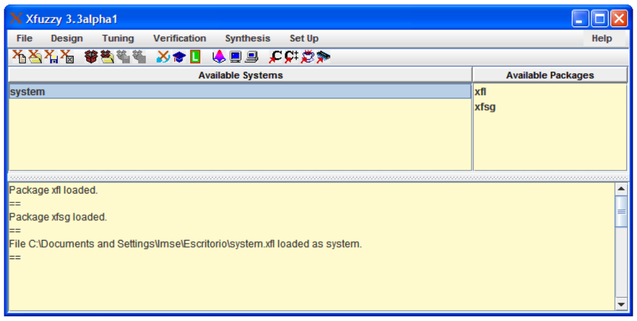
Xfuzzy 3 [4] is a design environment which includes CAD tools to cover the complete process of fuzzy logic design. From the main window of Xfuzzy 3 (Fig. 3), the user can select tools which allow automating description (in a specific language named *XFL3*), verification, tuning, identification, simplification and synthesis (in C, C++, or Java, to be included in software projects, or in VHDL, for hardware projects). These tools are the following:

1) For description:

- *xfedit*, which eases describing the logical structure of a fuzzy system, that is, its inputs, outputs, groups of membership functions for each variable, sets of operators for each rule base, rule bases (Fig. 4), and the system architecture (how rule bases are interconnected).
- *xfpkg*, which eases defining the function packages, that is, the code blocks describing the parameters, mathematical expressions and other features of membership functions, defuzzification methods, and unity and binary functions (related, respectively, to linguistic hedges and fuzzy connectives).

2) For verification:

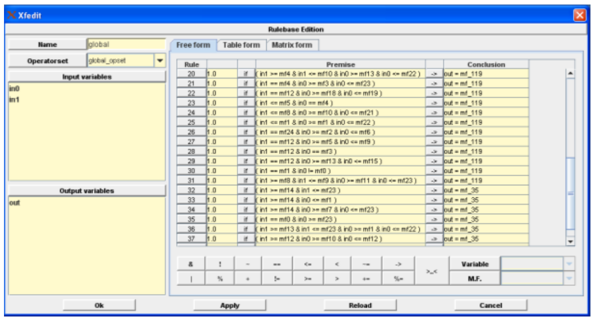
- *xplot*, to visualize graphically one of the outputs of the system against 1 (2D) or 2 (3D) of its inputs (Fig. 5).
- *xfmt*, to monitor how the output values are obtained by inferring from the input ones.
- *xfsim*, to simulate how the fuzzy system behaves within the application domain.



Available Systems	Available Packages
system	xfl
	xfsg

```
Package xfl loaded.
==
Package xfsg loaded.
==
File C:\Documents and Settings\Imse\Escritorio\system.xfl loaded as system.
==
```

Fig. 3. Main window of *Xfuzzy 3* environment.



Name: global
Operatorset: global_opset
Input variables: in0, in1
Output variables: out

Rule			Premise		Conclusion
20	1.0	if	(in1 >= mf4 & in1 <= mf10 & in0 >= mf13 & in0 <= mf22)	->	out = mf_119
21	1.0	if	(in1 == mf4 & in0 >= mf3 & in0 <= mf23)	->	out = mf_119
22	1.0	if	(in1 == mf12 & in0 >= mf18 & in0 <= mf19)	->	out = mf_119
23	1.0	if	(in1 <= mf5 & in0 == mf4)	->	out = mf_119
24	1.0	if	(in1 <= mf8 & in0 >= mf10 & in0 <= mf21)	->	out = mf_119
25	1.0	if	(in1 <= mf1 & in0 >= mf1 & in0 <= mf22)	->	out = mf_119
26	1.0	if	(in1 == mf24 & in0 >= mf2 & in0 <= mf6)	->	out = mf_119
27	1.0	if	(in1 == mf12 & in0 >= mf5 & in0 <= mf9)	->	out = mf_119
28	1.0	if	(in1 == mf12 & in0 == mf3)	->	out = mf_119
29	1.0	if	(in1 == mf12 & in0 >= mf13 & in0 <= mf15)	->	out = mf_119
30	1.0	if	(in1 == mf1 & in0 != mf0)	->	out = mf_119
31	1.0	if	(in1 >= mf8 & in1 <= mf9 & in0 >= mf11 & in0 <= mf23)	->	out = mf_119
32	1.0	if	(in1 >= mf14 & in1 <= mf23)	->	out = mf_35
33	1.0	if	(in1 >= mf14 & in0 <= mf1)	->	out = mf_35
34	1.0	if	(in1 >= mf14 & in0 >= mf7 & in0 <= mf23)	->	out = mf_35
35	1.0	if	(in1 == mf0 & in0 >= mf23)	->	out = mf_35
36	1.0	if	(in1 >= mf13 & in1 <= mf23 & in0 >= mf1 & in0 <= mf22)	->	out = mf_35
37	1.0	if	(in1 >= mf12 & in0 >= mf10 & in0 <= mf12)	->	out = mf_35

Fig. 4. Fuzzy rule base for a directional image-based template.

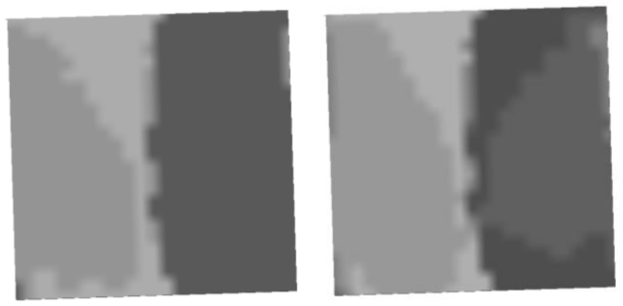


Fig. 5. Graphic representation provided by *xfplot* for two fuzzy models of the directional image in Fig. 1b. Each gray level represents a dominant direction.

3) For tuning:

- *xfsl*, which allows applying a wide set of supervised learning algorithms to adjust the parameters of the fuzzy system.

4) For identification:

- *xfdm* is a tool that extracts information from numerical data and creates a fuzzy system with algorithms based on grid or clustering partitions.

5) For simplification:

- *xfsp*, which applies simplification algorithms to either membership functions or rule bases of a fuzzy system.

6) For synthesis:

- *xfc*, *xfcc*, and *xfj*, which, respectively, translate the description of the system in *XFL3* to C, C++, and Java code.
- *xfsg* and *xfvhdl*, which, respectively, provides a FPGA implementation using Xilinx System Generator (SysGen) tool in Matlab-Simulink, and generates generic VHDL code.

III. Design flow for hardware implementation

A top-down design flow has been employed to implement a directional image-based template in hardware. It starts with a high-level description that creates the fuzzy model of the directional image and finishes with two types of devices: FPGAs and ASICs.

The first step towards creating the fuzzy model is to generate a training file containing the numerical data associated to the directional image. Data should be organized into three columns: x and y locations of the pixel, and the corresponding direction value. This file is generated with Matlab by applying a gradient-based algorithm that extracts directions in the interval [0º, 180º]. For the fingerprint image in Fig. 1a (300x300 pixels), the directional image contains 300x300 pixels.

The tool *xfdm* is employed to extract fuzzy rules from the numerical data in the training file. It is configured to use Wang-Mendel grid-based algorithm and to generate a 2-input, 1-output system, with triangular membership functions for each input, *product* as conjunction operator and *Fuzzy Mean* as defuzzification operator. For instance, with 25 functions for each input, the resulting fuzzy model is composed by 25x25 (625) rules and 625 consequents.

The next step is to employ the simplification tool *xfsp* to reduce, firstly, the number of consequents, and, secondly, the number of rules. The consequents are clustered into groups and the rules with the same consequent are grouped and merged (if possible) by applying a *Tabular Simplification* algorithm [7]. Finally, the parameters of the model are adjusted with the tool *xfsl*. As examples, Fig. 5 shows two fuzzy models whose consequents have been clustered into 3 and 5 groups, and the rules have been simplified from 625 to 39 (Fig. 5a) and to 77 (Fig. 5b). The 3 gray levels in Fig. 5a illustrate the areas of the directional image whose dominating direction is one of the following: 35º, 119º, 152º. The 5 gray levels in Fig. 5b illustrate the areas whose dominant direction is one of the following: 20º, 45º, 90º, 125º, 158º. The fuzzy model in Fig. 5b is finer because it uses more membership functions for the consequents. Fig. 4 shows part of the rule base generated for the simplest model. One of these rules (obtained by merging 225 rules) is the following:

‘IF the pixel is placed *on the right* (regarding x position) THEN direction is *35º approximately*’

A. FPGA-based Implementation

The tool *xfsg* allows communicating Xfuzzy with Matlab-Simulink, in particular with Xilinx System Generator (SysGen) tool. It uses a library of modules, named Xfuzzy Blockset or XfuzzyLib, which has been developed (Fig. 6) to implement each of the basic elements required in the active-rule driven architecture described in [8]. Using these modules, the tool *xfsg* generates a Simulink model of the fuzzy system. The parameters related to the bit size of the variables in the fuzzy system should be introduced by the user through the *xfsg* graphical user interface. Simulink results (taking into account hardware features) can be compared with the representation generated by Xfuzzy.

Since System Generator supports hardware synthesis to ease the construction of DSP algorithms on FPGAs, the VHDL description of the Simulink model, including an associated

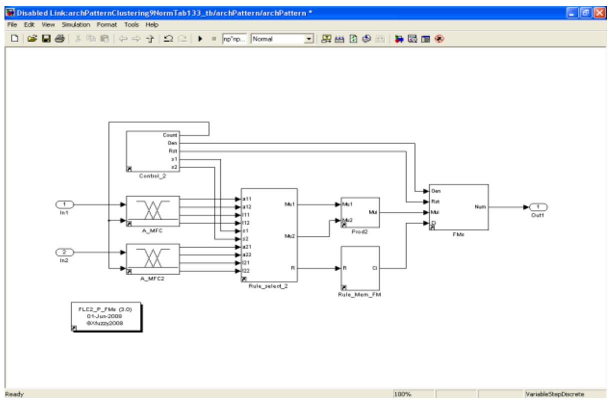


Fig. 6. *XfuzzyLib* module used by *xfsg*.

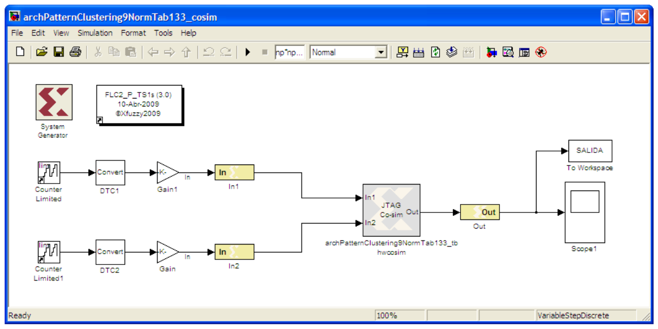


Fig. 7. Simulink model created with Co-simulation component for FPGA implementation.

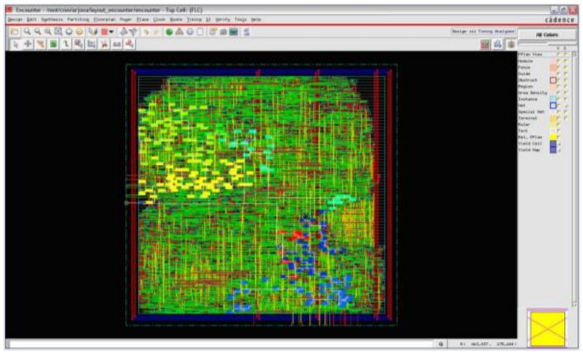


Fig. 9. Layout generated in *Cadence Encounter*.

testbench, are automatically generated. *Isim* simulator, from Xilinx *ISE* tools, can be used to simulate the circuit described in VHDL code. Using *XST* from Xilinx *ISE*, the VHDL code generated by System Generator (or, directly, the bitstream generated) can be implemented in a Xilinx FPGA, and obtain implementation details concerning resource utilization and timing. For example, a fuzzy model for the directional image in Fig. 1b, with 9 clusters for the consequents, 133 rules, 8 bits for inputs, 16 bits for output, and 12 bits to generate membership functions implemented in a Spartan 3A uses 4% of FPGA slices and 19.1 ns as minimum clock period.

Hardware/software Co-Simulation option from System Generator can be employed to verify the system in the FPGA and analyze its behaviour accordingly to software and modeling results. Fig. 7 shows an example of a Simulink model allowing co-simulation.

B. *ASIC-based Implementation*

The tool *xfvhdl* provides automatic translation from XFL3 descriptions to generic VHDL code, which can be used to synthesize the system regardless of device. The tool, which also follows the active-rule driven architecture employed by *xfsg*, generates a set of VHDL files and a testbench, which eases hardware simulation (for example, with Mentor ModelSim).

The same fuzzy model implemented in the FPGA, as commented above, has been implemented in an ASIC, selecting the (same) configuration parameters in the graphical user interface of *xfvhdl*: 8 bits for inputs and 12 bits for membership functions. Synthesis has been done with Synopsys Design Compiler following a semi-custom design flow with UMC CMOS 180 nm technology. Synopsys Design Compiler selects wire-load models that depend on the number of cells and applies a pessimistic estimation according to the design complexity. Hence, timing and area reports should be analyzed to select the adequate solution. In the example considered herein, the clock signal was set to 5 ns because this value provided the best trade-off between area and time. Moreover, area optimizations were applied. The design reports were 34880 μm^2 for area and 7.44 ns for minimum clock period. Fig. 8 shows schematic view for the design in Synopsys Design Compiler.

Cadence Encounter has been employed to carry out the layout. Using netlist and restrictions files generated by Synopsys Design Compiler, the design flow includes floorplan, placement, timing optimization, clock design and routing stages. The layout, created without pads, is shown in Fig. 9.

IV. CONCLUSIONS

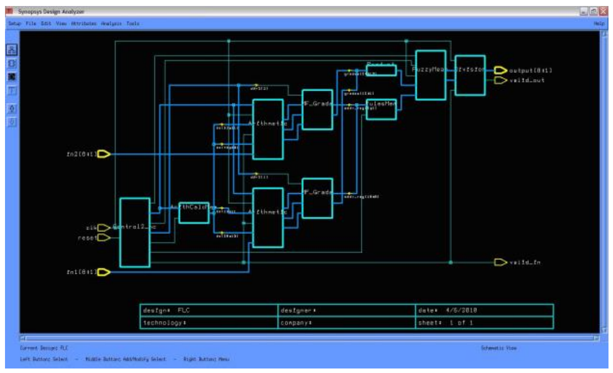


Fig. 8. Schematic view for the design in *Synopsis Design Compiler* with UMC CMOS 180 nm technology.

Fuzzy models can describe fingerprint directional images in a compact and smooth way. A low number of parameters associated with the fuzzy rules should be stored instead of the whole directional image. The complete process of obtaining the fuzzy model and implementing it in an FPGA or ASIC can be automated with the CAD tools of Xfuzzy environment, interacting with Matlab-Simulink, Xilinx ISE, Synopsys and Cadence. The results obtained in both target devices are efficient in terms of area and speed.

REFERENCES

[1] http://bias.csr.unibo.it/fvc2000/

[2] A. Grasselli, “On the automatic classification of fingerprint-Some consideration of the linguistic interpretation of pictures,” in *Methodologies of Pattern Recognition*, S. Watanabe, Ed. Academic Press, 1969, pp. 253-273.

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

[4] http://www.imse-cnm.csic.es/Xfuzzy/

[5] L. A. Zadeh, "Fuzzy Sets," Inf. Contr., 1965.

[6] I. Baturone, A. Barriga, S. Sánchez-Solano, C. J. Jiménez-Fernández, D. R. López, *Microelectronic design of fuzzy logic-based systems*, CRC Press, 2000.

[7] I. Baturone, F. J. Moreno-Velo, A. A. Gersnoviez, "A CAD Approach to Simplify Fuzzy System Descriptions", Proc. FUZZ-IEEE'2006, pp. 2392-2399, Vancouver (Canada), July 2006.

[8] S. Sánchez-Solano, A. Cabrera, I. Baturone, F. J. Moreno-Velo, and M. Brox, "FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications". IEEE Trans. on Industrial Electronics, vol. 54, n. 4, pp. 1937-1945. Aug. 2007.