Hardware Implementation of a Biometric Recognition Algorithm based on In-Air Signature

Rosario Arjona, Rocío Romero-Moreno, Iluminada Baturone
Microelectronics Institute of Seville (IMSE-CNM)
Universidad de Sevilla - CSIC
Seville, Spain
{arjona, lumi}@imse-cnm.csic.es

***Abstract*—This paper presents the design of a prototype for a wearable device that implements a recognition system based on in-air signature into a FPGA that receives data from a 3-axis accelerometer. The Dynamic Time Warping (DTW) algorithm has been analyzed and simplified to reduce the complexity of the hardware architecture that implements the matching in the FPGA. Despite simplification, accuracy of the recognition is maintained and the Equal Error Rate, EER, is 4.21% considering a public database with 120 in-air signatures. A prototype based on a Spartan 6 LX9 microboard connected to an ultralow power ADXL345 accelerometer has been developed. Performance of the prototype working with in-air signatures has been verified with a script developed in *Matlab-Simulink*. The execution time for matching is 22 ms and the estimated average power consumption of the matching in the FPGA is 26 mW.**

Keywords—Biometrics; In-air signature; DTW; Hardware in FPGAs; CAD tools

I. Introduction

Biometrics is the science which studies measurements based on physical and behavioral attributes of a person to recognize individuals [1]. Several traits such as fingerprints, faces, voice, or iris, among others, have been widely studied and employed in biometric recognition systems. The application contexts range from forensic science, law enforcement, access control systems, and surveillance systems to the latest trends in wearable technology.

Wearable technology requires low-cost, small and lightweight devices, which imposes high constraints in terms of resources, real-time response and power consumption. The selection of a biometric trait suitable for wearable devices should consider small-size sensors to acquire the signals as well as algorithms of low complexity that maintain discrimination capability. The in-air signature satisfies these constraints. It is based on drawing a signature in the air (or any movement which is difficult to reproduce by an impostor) by employing a device which contains an accelerometer. In-air signature offers information from two points of view: physical and behavioral. On the one hand, it depends on the arm length and the hand size. On the other hand, it depends on the capability to move the wrist, the way to reproduce the signature (fast or slow movements) and the way to use the device [2].

Biometric recognition by in-air signature was proposed by the Group of Biometrics, Biosignals, and Security (GB2S), which belongs to the Centro de Domótica Integral (CeDInt), from Polytechnical University of Madrid. They implemented an in-air signature-based individual recognition system in a smartphone (iPhone 3G). Samples were acquired by the 3-axis accelerometer embedded in the smartphone [2]. The recognition algorithms were developed as an iOS application. Later, researchers from the National Chiao Tung University (NCTU) developed an application, which is currently available in Google Play [3]. Recently, in-air gestures have also been applied to handwriting recognition [4].

While the above commented solutions are software implementations, the focus of this work is the implementation of in-air signature-based recognition algorithms in dedicated hardware, meeting the requirements of wearable gadgets such as bracelets, car keys, etc., in terms of small size and low power consumption. The algorithm has been analyzed and simplified to achieve that the complete recognition process (enrollment and matching) can be included with low cost in the same device. In the enrollment phase, the in-air signature is acquired and stored in the device memory as a template to register an individual. In the matching phase, another in-air signature is acquired and matched against the template or the N templates stored in the memory. If one template is registered, the recognition process is known as authentication, while if N templates are registered, the process is known as identification. In any case, doing enrollment and matching within the same device is much more secure than doing them in separate devices, because the number of attacks that can be carried out decreases considerably.

The paper is structured as follows. Section II reviews the main algorithms applied to in-air signature recognition. Section III presents the analysis done to obtain the specifications for the hardware implementation of the system. Recognition results are shown for the selected specifications. Section IV describes the design of a prototype based on a FPGA and a 3-axis

This work has been partially supported by TEC2011-24319 and IPT-2012-0695-390000 projects from Ministerio de Economía y Competitividad of the Spanish Government (with support from the PO FEDER-FSE).

accelerometer as a first step towards the design of a wearable device. Finally, conclusions are given in Section V.

II. In-air Signature Recognition Algorithms

The in-air signature is composed of three sequences of acceleration values (*x*, *y*, and *z*) associated to the 3-axis accelerometer. Axis *x* represents the left-right direction, axis *y* represents the up-down direction, and axis *z* represents the front-back direction. The matching operation implies the comparison of the template to the input sequences, which results a score value. The comparison of the score value to a predefined threshold value results the recognition decision.

The acceleration values acquired can be employed directly (no normalization is applied), can be divided by the maximum acceleration acquired by the accelerometer (maximum normalization), can be divided by the maximum absolute value of the sequence (maximum on-sequence normalization), or can be subtracted from the average and divided by the maximum absolute value of the sequence (maximum-average normalization). The recognition results in [5] show that no normalization and maximum normalization offer a higher accuracy. Hence, this work focuses on both solutions, which are similar (since maximum normalization is just a scale adjustment) and are the least complex options from a hardware implementation point of view.

Two in-air signatures (i.e. the input signature to verify and the stored signature or template) cannot be compared directly because the signatures carried out by the same individual differ from one capture to another. For example, the individual can apply more or less speed and the signatures can start at different time instants. The sequences captured for each axis of the signatures should be aligned firstly (to correct the variations), and then processed to obtain a similarity value. Although Hidden Markov Models (HMMs) [6] and Neural Networks (NNs) have been employed for this purpose [7], dynamic programming algorithms are more effective to obtain the optimal alignment because HMMs and NNs require training stages. Two dynamic programming algorithms are employed in the literature: LCS (Longest Common Sequence) and DTW (Dynamic Time Warping). The LCS algorithm looks for the optimal alignment by maximizing the length of the common subsequence of two sequences, that is, it maximizes the similarities between two sequences. In contrast, the DTW algorithm looks for the optimal alignment by minimizing the Euclidean distance between two sequences, that is, it minimizes the differences between two sequences. Regarding complexity, LCS and DTW are similar since they are based on the same paradigm and employ similar operations. Regarding recognition performance, the work in [5] proves that DTW algorithms provide better results than LCS algorithms. Hence, this work focuses on DTW implementation.

In the DTW algorithm, the alignment of two sequences (*v* and *w*) is computed by using a cost matrix *S* whose elements, *S*(*i*, *j*), are obtained as follows:

0.116	0.002	0.126	0.053	0.057
0.000	0.102	0.000	0.020	0.018
0.014	0.165	0.010	0.052	0.048
0.016	0.171	0.012	0.055	0.052
0.012	0.158	0.008	0.048	0.045

Fig. 1. Example of matrix of Euclidean distances, *d(i,j)*.

$$S(i,j) = d(i,j) + \min\{S(i-1,j), S(i-1,j-1), S(i,j-1)\} \quad (1)$$

where *d*(*i*, *j*) is the difference between the points (v_i and w_j) of the sequences (*v* and *w*). Typically, *d* is computed by the Euclidean distance as follows:

$$d(i,j) = (v_i - w_j)^2 \quad (2)$$

The values of the first column and the first row of *S* are filled as follows:

$$\begin{aligned} S(i,1) &= d(i,1) + S(i-1,1) \\ S(1,j) &= d(1,j) + S(1,j-1) \end{aligned} \quad (3)$$

Let us illustrate how the DTW algorithm works with the following example:

v=*template* = [0.390, 0.679, 0.375, 0.500, 0.492]

w=*input* = [0.730, 0.359, 0.273, 0.265, 0.281]

where *template* is the sequence stored in the database at the enrollment phase and *input* is the sequence captured at the recognition phase.

The matrix of Euclidean distances for the example considered is shown in Fig. 1. Fig. 2 illustrates how these distances are employed to calculate the matrix *S*. The last element of the matrix *S* (within a circle in Fig. 2) is named the *matrix cost value* and will be referred to as C_x, C_y, and C_z, depending on the axis

0.116	0.118	0.244	0.297	0.354
0.116				
0.130				
0.146				
0.158				

$\longrightarrow$

0.116	0.118	0.244	0.297	0.354
0.116	0.218	0.118	0.138	0.156
0.130	0.281	0.128	0.170	0.186
0.146	0.301	0.140	0.183	0.222
0.158	0.304	0.148	0.188	$\bigcirc$ 0.228

Fig. 2. Construction of the cost matrix in the DTW algorithm by using the distances *d(i, j)* in Fig. 1.

TABLE I: Normalization approaches for the score values:

Score normalized by the maximum length of the template and input sequence	Score normalized by the maximum length of the templates	Score normalized by the length of the aligned sequences for each axis
$\frac{E_x + E_y + E_z}{\max(L_{template}, L_{input})}$	$\frac{E_x + E_y + E_z}{\max L}$	$\frac{E_x}{L'_x} + \frac{E_y}{L'_y} + \frac{E_z}{L'_z}$

considered. If L_v is the length of the sequence v and L_w is the length of the sequence w, the last element will be referred to as $S(L_v, L_w)$.

Although the matrix cost value offers information about the similarity of two sequences, the sequences are firstly aligned to determine a similarity value by means of the Euclidean distance. The optimal path for alignment is obtained from the element $S(L_v,\ L_w)$ to the element $S(1,1)$ by the type of movement performed in the computation of the matrix S: vertical (i-1,j), diagonal (i-1,j-1), or horizontal (i,j-1), which is given by the value considered in the minimum operation in (1). For the example considered, the optimal path obtained is shown in gray in Fig. 3. The path indicates the indices of the aligned elements from the original sequences (the column indices of the elements in the path indicate the elements of the template and the row indices, the elements of the input sequence) as follows:

templateIndices = [1, 2, 3, 3, 4, 5]

inputIndices = [1 ,1, 2, 3, 4, 5]

Hence, the horizontal or vertical movements imply to insert elements in the sequences to align them. The aligned sequences obtained in the example are the following:

template' = [0.390, 0.679, 0.375, 0.375, 0.500, 0.492]

input' = [0.730, 0.730, 0.359, 0.273, 0.265, 0.281]

Once the sequences are aligned (v', w'), their length, L', is the same. The Euclidean distance of the aligned sequences is calculated as follows:

$$E = \sum_{k=1}^{L'} (v'_k - w'_k)^2 \qquad (4)$$

For each axis, a Euclidean distance value is obtained (E_x, E_y, and E_z), so that he final score is calculated by the sum of the Euclidean distance values:

$$score = E_x + E_y + E_z \qquad (5)$$

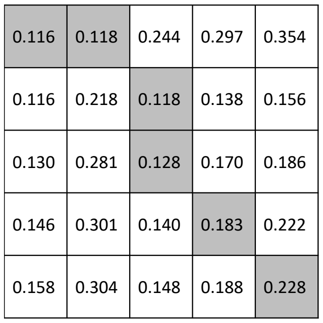
III. Algorithm simplification for Hardware Implementation

Hardware implementation oriented to wearable devices requires selecting algorithms of low complexity to satisfy the constraints imposed by the application context. At the same time, recognition performance should be considered to achieve a tradeoff between simplicity and accuracy. Recognition accuracy is evaluated in terms of EER (Equal Error Rate). The EER is the value where False Match Rate (FMR) and False Non-Match Rate (FNMR) coincide (FMR=FNMR). FMR and FNMR represent the two types of recognition errors at matching. If the biometric samples from two different individuals match, it is a false match, and if the two biometric samples from the same individual do not match, it is a false non-match. The number of false matches for the impostor distribution determines the performance indicator named as False Match Rate (FMR). The number of false non-matches for the genuine distribution determines the performance indicator named as False Non-Match Rate (FNMR).

The comparison of the score value to a predefined threshold value, t, results the match or non-match decision. Therefore, FMR and FNMR are functions of the threshold value selected, t, so they can be expressed as FMR(t) and FNMR(t). There is a tradeoff between FMR and FNMR in every biometric recognition system. The value predefined for the threshold t depends on the final application. The evaluation of a biometric recognition system in a generic way requires considering all possible values for t to compute FMR and FNMR values over the genuine and impostor distributions.

In order to analyze how algorithm simplification for hardware implementation influences recognition performance, the in-air signature database in [8] has been considered (in particular, the database GB2SDB1). It is composed of 30 individuals and 4 in-air signature captures for each individual.

The score value used in the hardware realization should be normalized; so that it does not depend on the time taken by the individual in drawing the signature (this time is usually different for the same individual and the same signature). In order to normalize the score values, the approaches shown in Table I have been considered. For the normalization based on the division of the score value by the maximum length of the template and input



0.116	0.118	0.244	0.297	0.354
0.116	0.218	0.118	0.138	0.156
0.130	0.281	0.128	0.170	0.186
0.146	0.301	0.140	0.183	0.222
0.158	0.304	0.148	0.188	0.228

Fig. 3. Path which describes the alignment of sequences.

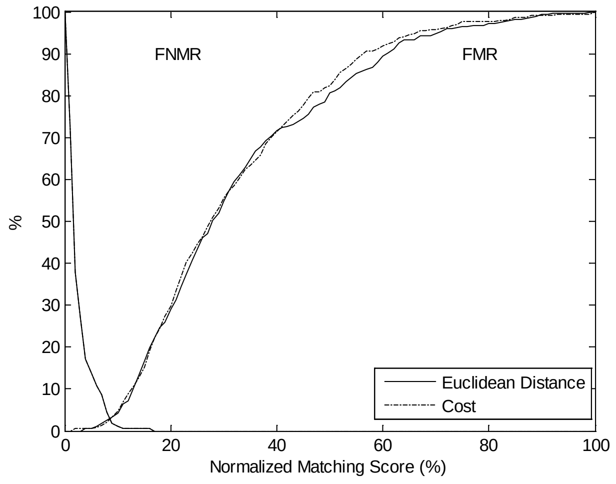


Fig. 4. Comparison of FNMR and FMR results when considering Euclidean distances and matrix cost values.

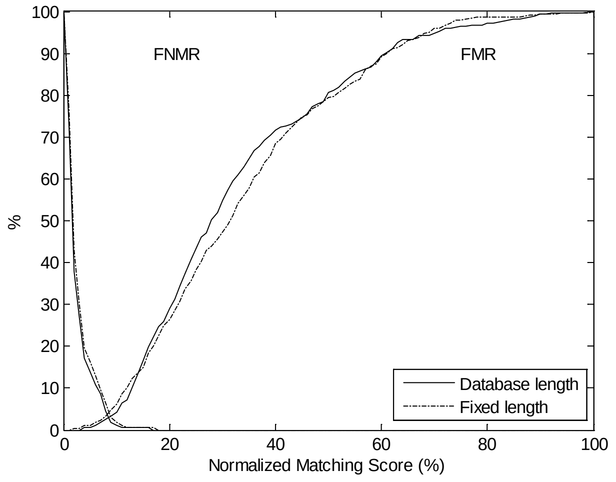


Fig. 5. Comparison of FNMR and FMR results when considering sequences with variable and fixed lengths.

sequences (first column of Table I) the EER value is 3.69%. The normalization based on the division of the score value by the maximum length of the stored templates, as in the second column of Table I, gives an EER value of 4.90%. The division of the Euclidean distances by the length of the aligned sequences for each axis, as in the third column of Table I, gives an EER value of 5.48%. Therefore, the normalization selected for the hardware implementation is the first approach because it offers the highest accuracy.

The above commented results are obtained from the computation of the Euclidean distances in (4). As summarized in Section II, such distances for each axis are computed after the alignment of the sequences once the optimal path is found from the cost matrix processing. Another study carried out has been to use as score metric the matrix cost value, which is obtained directly from the matrix computation, instead of using the Euclidean distances in (4). This implies an important reduction of complexity because it is not necessary to process the matrix to obtain the aligned sequences and then to compute the Euclidean distance. Regarding accuracy, performance is maintained because recognition results are similar for the two approaches. In the example illustrated in Section II, it can be seen that the matrix cost value C and the Euclidean distance value E are the same for the two sequences (both results are 0.228). Fig. 4 illustrates the FNMR-FMR curves obtained when this simplification is applied to the database GB2SDB1 (compared to applying the algorithm without simplification).

Another simplification considered for hardware implementation is to take into account fixed-length instead of variable-length sequences because the creation of the cost matrix is less complex if the sequence sizes do not change. Let us define L_{fixed} as the length value predefined for the sequences. For the experimental database analyzed herein, this value has been fixed to 600. The captured sequences whose length was higher than L_{fixed} were down-sampled. In contrast, padding was needed for the sequences whose length was lower than L_{fixed}. Fig. 5 illustrates how the influence of such simplification on the FNMR-FMR curves of the database GB2SDB1 is small. The resulting EER is 4.21%, which is very similar to the EER without simplification.

In summary, the expression of the normalized matching score selected to be computed in hardware is the following:

$$normScore = \frac{C_x + C_y + C_z}{L_{fixed}} \quad (6)$$

where C_x, C_y, and C_z are the matrix cost values resulting from the DTW matrices computed for the x-, y-, and z-axis sequences from the template and input captures, respectively.

IV. Design of a Wearable Prototype

As a first approach for a wearable device, the Spartan-6 LX9 microboard has been selected to evaluate the implementation of

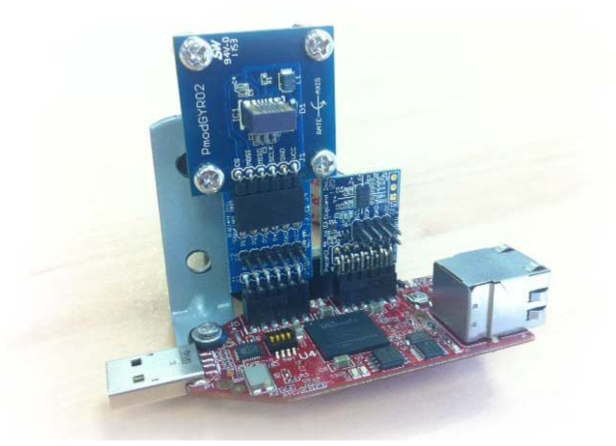
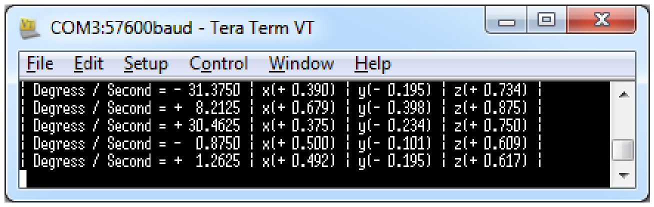


Fig. 6. Spartan 6 LX9 microboard connected to a 3-axis accelerometer and a gyroscope through Pmod interfaces.


COM3:57600baud - Tera Term VT
File Edit Setup Control Window Help

```
¦ Degress / Second = - 31.3750 ¦ x(+ 0.390) ¦ y(- 0.195) ¦ z(+ 0.734) ¦
¦ Degress / Second = +  8.2125 ¦ x(+ 0.679) ¦ y(- 0.398) ¦ z(+ 0.875) ¦
¦ Degress / Second = + 30.4625 ¦ x(+ 0.375) ¦ y(- 0.234) ¦ z(+ 0.750) ¦
¦ Degress / Second = -  0.8750 ¦ x(+ 0.500) ¦ y(- 0.101) ¦ z(+ 0.609) ¦
¦ Degress / Second = +  1.2625 ¦ x(+ 0.492) ¦ y(- 0.195) ¦ z(+ 0.617) ¦
```

Fig. 7. Examples of acceleration and orientation values acquired by the prototype.

the acquisition and the matching stages as dedicated hardware in a low-cost FPGA (Spartan-6 LX9 CSG324-2 from Xilinx).

The in-air signature acquisition is performed by the small, thin and ultralow power ADXL345 3-axis accelerometer. It uses a standard 12-pin Pmod connector that is inserted in the LX9 microboard. Additionally, a gyroscope (the ADXRS453) which captures information of movements in different directions and orientations has also been connected to the microboard using another Pmod interface. A future study to carry out is how the fusion of different information (accelerations and orientations) can improve recognition performance. Fig. 6 shows the Spartan 6 LX9 microboard connected to both types of acquisition devices. The gyroscope in particular has been fixed adequately to the microboard to avoid undesirable vibrations of the sensor when performing the in-air signatures.

The acquisition stage (needed by enrollment and matching phases) is in charge of capturing the acceleration values provided by the accelerometer. The code provided by Analog Devices in [9] has been reused to program in the FPGA a system based on the processor MicroBlaze that communicates with the accelerometer via the SPI interface.

A. *Hardware Design Methodology*

A hardware design methodology based on CAD tools has been employed. It follows a top-down design flow that starts with the high-level description of the system in *Matlab-Simulink*. The software reported in [10] has been employed to describe the complete alignment of sequences by means of the DTW algorithm, which is in charge of the matching operation. The code programmed in *Matlab* uses floating-point arithmetic. This Matlab code was translated to a *Simulink* model which takes into account the simplifications explained in Section III and hardware considerations such as delays, buffers, memory, and fixed-point data. The performance of both implementations was compared.

A script has been developed to capture the acceleration data into the *Matlab* workspace by a serial communication. In this way, the matching stage that is performed inside the prototype as dedicated hardware can be compared to the matching stage performed in *Matlab-Simulink* with the same data. Fig. 7 shows examples of accelerations and orientations captured by the prototype with this script.

The implementation in the device of the matching algorithm as dedicated hardware has been done with *HDL Coder* (included in *Matlab-Simulink*), which aims to facilitate hardware designs for any type of device (ASIC or FPGA), since it is possible to generate synthesizable HDL code (VDHL or Verilog). *HDL Coder* has also been used to generate the testbenches employed to simulate and verify the circuit at hardware code level. ISE *Isim* simulator has been employed for these simulations. This constitutes another verification point whose results can be compared to the results from the high-level descriptions.

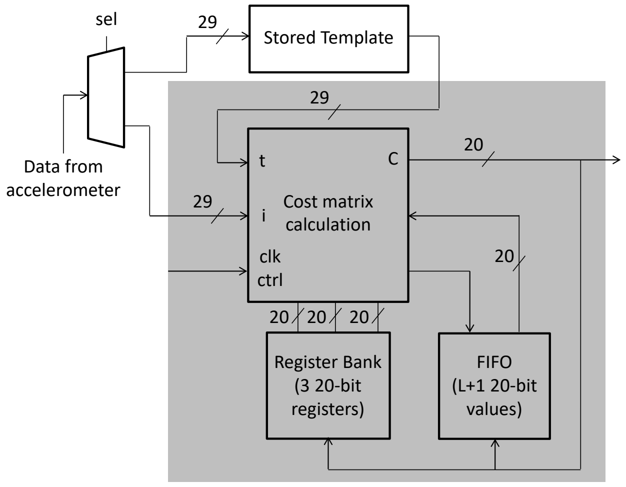


Fig. 8. Structure of the block *Matching*.

The final stage of the design flow is the device implementation. Since the prototype is based on a *Xilinx* FPGA, the tools from *Xilinx* ISE environment have been used to complete the process.

B. *Hardware Implementation of the Matching Algorithm*

Hardware implementations of the DTW algorithm in FPGAs can be seen in [11], where a hardware coprocessor is described in the context of a HW/SW system. The implementation described in the following for the block *matching* is focused on dedicated hardware to achieve more reduced figures of area, power, and processing time. Data are processed in a serial way, that is, the acceleration values captured by the accelerometer are transmitted one by one to the block *matching*, which carries out the DTW algorithm.

The inputs to the block *matching* are the template stored in a BRAM inside the FPGA in the enrollment stage and the acceleration values captured in the verification stage. The acceleration values are coded with 29 bits (1 bit for the sign, 16 bits for the integer part, and 12 bits for the fractional part). The output is the last element of the DTW cost matrix, $S(L_{fixed}, L_{fixed})$, for each axis, coded with 20 bits (16 bits for the integer part and 4 bits for the fractional part). Fig. 8 shows the structure of this block. It receives the template and input acceleration values for each axis and returns the matrix cost value associated to that axis.

For each clock cycle, the operations expressed in (1), (2) and (3) are computed in parallel. Hence, the process finishes after

TABLE II: Implementation results for *matching* block

Device	Slices (%)	Max. frequency (MHz)	Block RAMs (%)	Power (mW)
Spartan 6	27	49	3	26

1430 slices and 32 18-Kb block RAMs in total for Spartan-6 LX9 CSG324-2.

L_{fixed} x L_{fixed} clock cycles. Instead of storing all the elements of the DTW cost matrix, only L_{fixed}+1 elements of the matrix associated to the currently processed element are stored. A FIFO memory (which stores L_{fixed}+1 elements) and registers are employed for this purpose as illustrated in Fig. 9 for the example analyzed in Section II.

The block has been implemented in the Spartan-6 LX9 CSG324-2 FPGA of the Avnet LX9 microboard, employed for the design of the prototype. The schematic of the block is illustrated in Fig. 10. The occupation and timing results are included in Table II. The occupation percentage of Spartan-6 FPGA allows the inclusion of other biometric algorithms to implement multi-biometric fusion (for example, processing the gyroscope data, as commented above). Since each acceleration value is processed in one clock cycle, the execution time for aligning two 600-element sequences is 7.35 ms (600x600/49 MHz). Such results meet real-time requirements and improve considerably the time required by a software implementation (410 ms running on a 3.2-GHz Intel Core i7 CPU with 6-GB RAM). Regarding power consumption, the implementation offers a low value (in addition, the device can be powered down when no in-air signature is drawn).

In order to compute the score value for the complete recognition system as in (6), it is necessary to calculate a matrix cost value for each axis (*x*, *y*, and *z*). This can be performed in a serial or parallel way. In the serial approach, less resources but more time is required, and, in the parallel approach, more resources but less time is required. Taking into account the results shown in Table II, the serial approach has been selected in this prototype. Hence, the total execution time for aligning two 600-element in-air signatures (with three sequences each signature) is 22.05 ms.

The block *matching* works with fixed-point data (in contrast to the software implementation also analyzed in *Matlab-Simulink*, which works with floating-point data). The selection of the fixed-point accuracy in hardware has been done by comparing the resulting cost metric values from the software implementation SW_{cost} to the resulting cost metric values from the hardware implementation HW_{cost} for all the in-air signatures of the public database considered. The error measured as in (7) gives an acceptable value of 0.21 for the fixed-point data selected. The total number of comparisons *N* is determined by the genuine and impostor distributions by removing symmetric comparisons and avoiding correlations: each sample is matched against the remaining samples of the same in-air signature for the genuine distribution, and the first sample of each person is matched against the first sample of the remaining people for the impostor distribution.

$$error = \frac{1}{N}\sum_{k=1}^{N}\left| SW_{COST\,k} - HW_{COST\,k} \right| \qquad (7)$$

V. CONCLUSIONS

This paper describes the hardware implementation of a biometric recognition system based on in-air signature. The recognition algorithm has been analyzed to obtain specifications that are a good trade-off between hardware simplification and recognition accuracy. A public database of signatures has been employed for this analysis. A low-cost design flow supported by CAD tools from *Matlab-Simulink* and *Xilinx* ISE has allowed designing a prototype based on the Spartan-6 LX9 microboard connected to an ADXL345 3-axis accelerometer. Its functionality has been verified at different abstraction levels (from software to hardware descriptions). Preliminary biometric recognition results obtained with the prototype show that the calibration of the sensors is crucial to reduce dispersion of the genuine population. In future work, systematic analysis of the recognition errors will be carried out as well as multi-biometric fusion will be considered to combine in-air signature with other biometric traits, such as fingerprints or voice.

REFERENCES

[1] A. K. Jain, P. Flynn, and A. Ross, Handbook of Biometrics, Springer, 2008.

[2] J. Guerra-Casanova, C. Sánchez-Ávila, G. Bailador, and A. de Santos-Sierra, “Authentication in mobile devices through hand gesture recognition”, Int. J. Inf. Secur. Vol. 11 (2), pp. 65-83, April 2012.

[3] In-air signature for mobile security, http://spectrum.ieee.org/consumer-electronics/portable-devices/inair-signature-gives-mobile-security-to-the-passwordchallenged

[4] Y. Hsu, C. Chu, Y. Tsai, and J. Wang, “An inertial pen with dynamic time warping recognizer for handwriting and gesture recognition”, DOI 10.1109/JSEN.2014.2339843, IEEE Sensors Journal.

[5] J. Guerra-Casanova, C. Sánchez-Ávila, G. Bailador, and A. de Santos-Sierra, “Time series distances measures to analyze in-air signatures to authenticate users on mobile phones,” IEEE Int. Carnahan Conf. on Security Technology (ICCST), pp. 1-7, October 2011.

[6] J. S. Wang, and F. C. Chuang, “An accelerometer-based digital pen with a trajectory recognition algorithm for handwritten digit and gesture recognition, “ IEEE Trans. on Industrial Electronics. Vol. 59 (7), pp. 2998-3007, 2012.

[7] L. Tong, Q. Song, Y. Ge, and M. Liu, “HMM-based human fall detection and prediction method using tri-axial accelerometer,” IEEE Sensors Journal. Vol. 13 (5), pp. 1849-1856, 2013.

[8] Gesture Database, https://sites.google.com/site/engb2s/gb2sgesturedb1

[9] ADXL345 pmod Xilinx FPGA Reference Design, http://wiki.analog.com/resources/fpga/xilinx/pmod/adxl345

[10] Software implementation of a Dynamic Time Warping Algorithm, http://www.mathworks.com/matlabcentral/fileexchange/6516-dynamic-time-warping

[11] H. Liu, and N. Bergmann, “An FPGA softcore based implementation of a bird call recognition system for sensor networks,” Proc. of the Design and Architectures for Signal and Image Processing. DASIP, pp. 1-6, October 2010.