13

Exploiting Run-Time Reconfigurable Hardware in the Development of Fingerprint-Based Personal Recognition Applications

Mariano Fons and Francisco Fons
Departament d'Enginyeria Electrònica, Elèctrica i Automàtica,
Univeristat Rovira i Virgili, Tarragona
Spain

1. Introduction

The current technological age brings the knowledge and the means to continuously improve the quality of life of human beings. One example can be seen in the recent advances done in the field of biometrics, where those physiological (fingerprints, iris, hand geometry, face, etc.) and/or behavioural (voice, gait, keystroke dynamics, signature, etc.) characteristics of human beings, unique and different to each individual, are used in order to either authenticate or identify individuals in a more reliable way, enhancing thus those existing personal recognition applications based on physical tokens (ID cards, keys, etc.), PINs or passwords. The deployment of automatic biometrics-based personal recognition systems and their acceptance by the society depends on several factors such as the ease of use, the non-intrusive methods of operation and their related privacy concerns; as well as their recognition accuracy, reliability and security levels, response time and system costs. All these factors will determine the successful spread of the biometric security in a wide range of daily use applications such as electronic payment, access systems, border control, health monitoring, etc. all over the world.
Among the different human traits analyzed in the field of biometrics, this work is focused on fingerprints. Fingerprints are the oldest and most deeply used signs of identity. Personal recognition based on fingerprints has been successfully deployed in law enforcement, government, and forensic applications for more than one century. The first recognition systems were based on human experts in charge of matching fingerprints. However, the current technological age demands the development of less expensive and fully automated fingerprint-based personal recognition systems, not only in the cited fields of application but also in many other daily use consumer applications (mobile phones, personal digital assistant devices, laptops, automatic teller machines, internet, e-commerce, etc.). Although big advances have been made in recent years, automatic and reliable biometric recognition is still an open research problem today. That ideal personal recognition algorithm able to unequivocally authenticate the identity of any user from his/her legitimate fingerprint features does not exist. The way to overcome the present limitations and improve the accuracy performance of current biometrics-based authentication systems consists of adding further processing stages into the recognition algorithms, which directly affects the

complexity, the processing power and the costs of the physical systems where to implement those applications.
This works focuses on the search of the proper system architecture able to face those demanding constraints for the application: a high computational power needed to achieve reliable recognition performances in terms of False Acceptance and False Rejection rates (FAR/FRR), a high security level in order to stand any kind of external attacks (cryptographic systems), real-time performance, and low cost. A novel approach of embedded system based on programmable logic devices such as field programmable gate arrays (FPGA), hardware-software co-design techniques, and the exploitation of run-time reconfigurable hardware is proven to successfully address the above requirements.
This chapter is split in nine sections and in each of the sections specific research topics are addressed. Section 2 provides a general overview of the proposed application to be dealt in this work: the development of an Automatic Fingerprint-based Authentication System (AFAS) in charge of verifying the identity of any individual based on the analysis of that distinctive information available in fingerprints. A description of the proposed personal recognition algorithm to be used as reference in this work and to be implemented under different processing platforms is presented. The accuracy performance achieved by the suggested algorithm when evaluated on a large database of fingerprints is addressed in Section 3. One public database composed of up to 800 fingerprint images corresponding to 100 different individuals is used for evaluation purposes. Impostor and Genuine distributions, as well as performance indicators such as FAR, FRR or EER (Equal Error Rate) are given in order to objectively compare the reached performance with the performance of other published algorithms evaluated with the same open database. After presenting the accuracy performance exhibited by the proposed recognition algorithm, Section 4 aims at defining the proper system requirements for the physical platform in charge of the authentication process. The main goal is to find a flexible and high-performance processing platform able to deploy the biometric security in a wide range of daily use applications at low cost, therefore an embedded system architecture is suggested. Two different implementations of the same recognition algorithm are carried out in this work. The first implementation, covered in Section 5, is based on purely software-based solutions. One high performance computing (HPC) platform under Windows operating system and three different embedded system platforms based on low-cost and mid-performance microprocessors are evaluated. The strengths and weaknesses of each of the architectures are pointed out, and based on that information, a different embedded system architecture is suggested in Section 6 to overcome the main limitations exhibited by the previous systems. An embedded system architecture based on a general-purpose microprocessor acting as application core processor, and a programmable and run-time reconfigurable logic region where to instantiate –multiplexed in time and under demand– application-specific hardware coprocessors in charge of the execution of those time-intensive tasks is proposed as alternative solution. Both the microprocessor unit and the hardware accelerators, together with memory blocks and other peripherals are all embedded under a System-on-Programmable-Chip (SoPC) device to provide a highly integrated and more reliable solution. The second implementation of the AFAS application under the proposed embedded system architecture is covered in Section 7. The performance achieved in this new scenario is compared against that of previous scenarios. An outstanding improvement in performance is achieved at a reasonable cost. The work ends with some concluding

remarks in Section 8, and the citation of some research references in Section 9. The reached results prove that the suggested system architecture based on hardware-software co-design techniques under run-time reconfigurable FPGA devices is a cost-effective alternative solution to those existing software-based processing platforms in the deployment of AFAS applications.

2. Fingerprint-based personal recognition algorithm

The personal recognition process is composed of two main phases, as depicted in Fig. 1: the enrolment phase and the authentication phase.

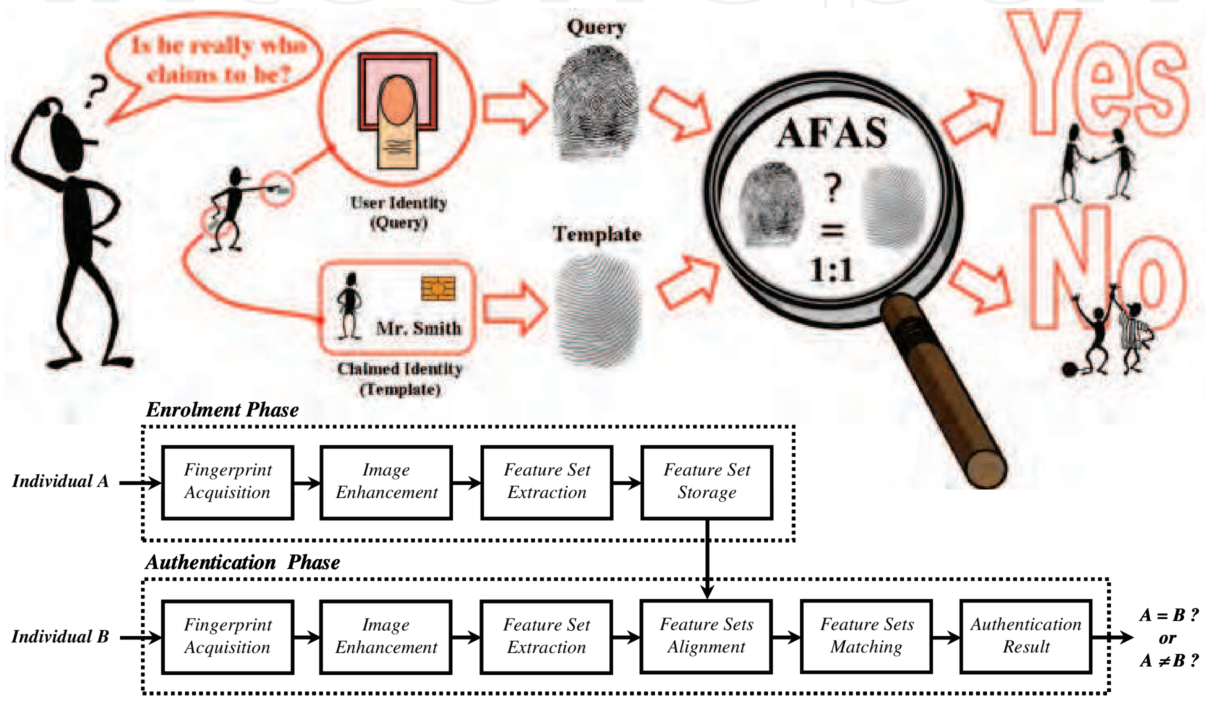

Fig. 1. Processing phases of an Automatic Fingerprint-based Authentication System

The enrolment phase is generally performed off-line, and consists in the registration of that set of biometric features extracted from the digital impression of the user's fingertip –known as *template*– together with any other relevant information of the user within the authentication system, either in a secure database or a personalized smart card. The authentication phase however is normally done on-line, and aims at validating the user's identity by comparing the set of on-line extracted biometric features –known as *query*– against those saved in the authentication system during the enrolment stage and linked to the legitimate individual claimed by the user –*template*–. The matching of both feature sets delivers a similarity score that is used to determine whether the user is really who claims to be, or on the contrary is an impostor who attempts to access the system fraudulently.
As it is indicated in Fig. 1, both phases –enrolment and authentication– are composed of a set of sequential stages. Each of the stages is, at the same time, split into smaller processing operations called tasks, and some of the stages/tasks carried out with the template and query fingerprints are common, as shown in Fig. 2. The aim of the authentication system is the execution of both phases of the processing; therefore the system has to be designed to afford any of the requested tasks along the application.

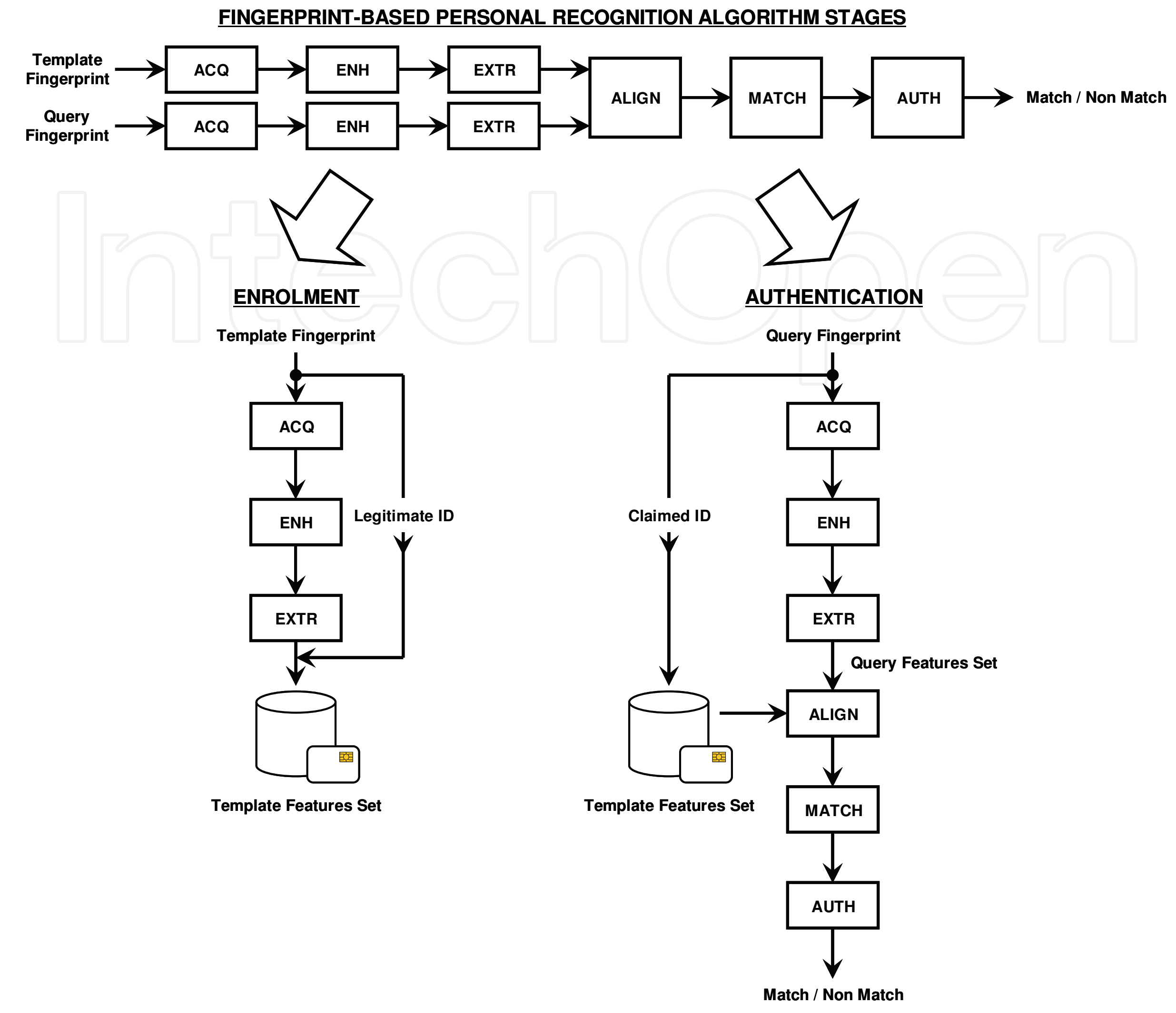

Fig. 2. Enrolment and authentication stages decomposition

The proposed recognition algorithm in charge of the enrolment and the authentication processes is not developed from scratch but based on some existing reference biometric algorithms and known techniques well described in the scientist literature. Specific image processing operations like convolutions, filters, etc. and other signal computations in the field of trigonometrics, statistics, etc. are performed on the acquired images in order to deduce that distinctive information available in the fingerprints. For a better understanding of the involved computational tasks refer to the authors' work (Fons et al., 2010). Fig. 3 shows the different processing steps that take place in the suggested fingerprint-based personal verification flow. A hybrid fingerprint matching algorithm that relies on the field orientation map and the set of minutia points extracted from the fingerprints is proposed for its physical implementation. Those classical biometric traits are considered as the genuine marks of identity of any individual. The computational load of the suggested algorithm is equivalent to those other similar or dissimilar algorithms that define the state of the art in fingerprint personal recognition today (Maltoni et al., 2009; Nanni & Lumini, 2009; Yang & Park, 2008).

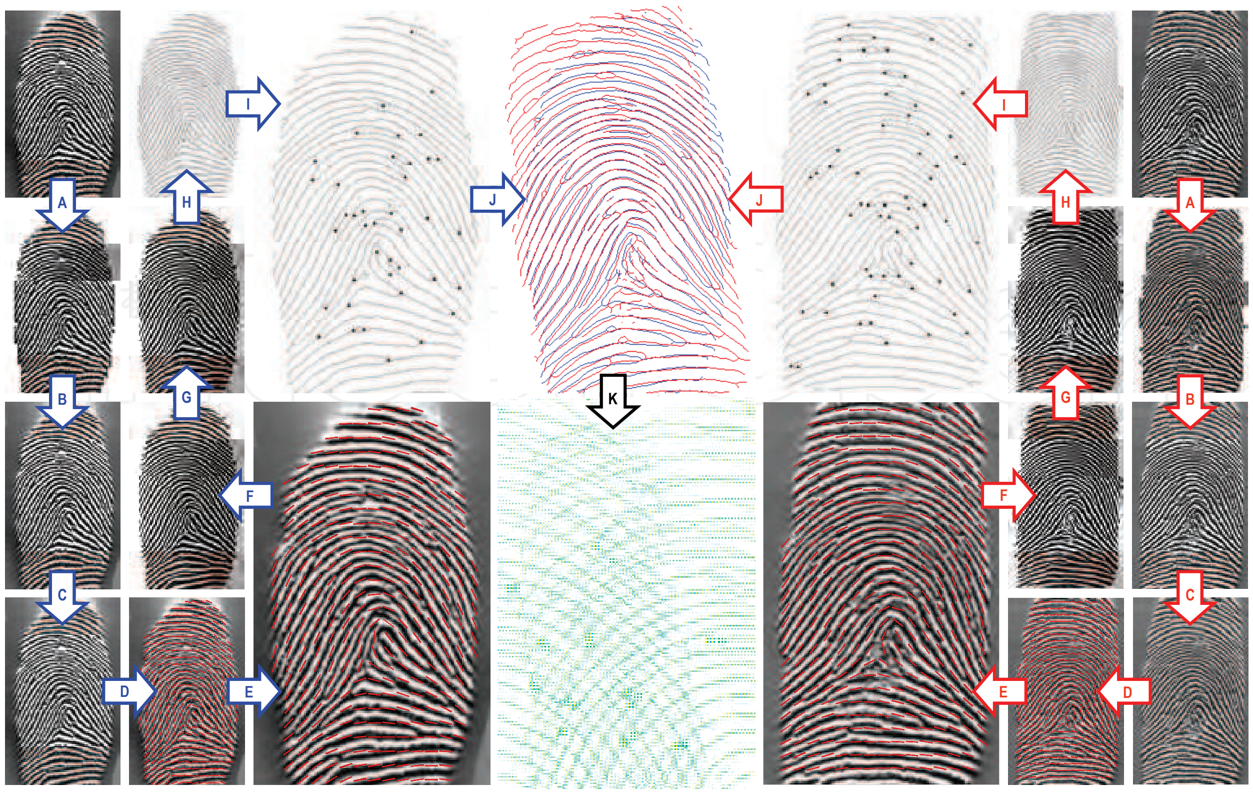

Fig. 3. Intermediate results in the processing of template (left side) and query (right side) fingerprints

A summary of the processing stages involved in the suggested personal recognition algorithm can be deduced from Fig. 3 when authenticating one query fingerprint (right side, red arrows) against one previously enrolled template fingerprint (left side, blue arrows). Up to 11 different tasks (A-K) are carried out along the processing, covering the image enhancement stage (tasks A-G), the feature sets extraction stage (tasks H-I), the feature sets alignment (task J) and the feature sets matching (task K) stages:

- Task A refers to the image segmentation process, which takes as input the acquired fingerprint impression and aims at isolating the valid fingerprint area, also known as foreground, from the rest of the image, also known as background.
- Task B refers to the image normalization process, which aims at adapting the variation of grey level intensities along ridges and valleys in the different regions of the fingerprint.
- Task C refers to the isotropic filtering of the image, which aims at removing some of the hazard noise that could be present in the fingerprint impression.
- Task D refers to the field orientation map computation, which consists in the calculation of the dominant direction of ridges and valleys in each local region of the fingerprint.
- Task E refers to the filtered field orientation map computation, which pursues the enhancement of the previously computed field orientation map.
- Task F refers to the image binarization process, which aims at discriminating ridges and valleys based on the directional filtering of the image according to the enhanced field orientation map.
- Task G refers to the image smoothing process, which aims at enhancing the black and white representation of the image by removing some of the noise that could be present in the binary version of the fingerprint image.

- Task H refers to the image thinning process, which aims at progressively removing the ridge pixels of the image preserving the geometric topology of the ridge-valley pattern till obtaining one skeleton of one single pixel wide to make easy the subsequent identification of minutia points.
- Task I refers to the minutia extraction process, which aims at deducing those salient features spatially distributed along the ridge-valley pattern such as the ridge endings and the ridge bifurcations. Those features will be used as discriminatory information of the fingerprint, together with the filtered field orientation map.
- Task J refers to the image alignment process, which aims at looking for any spatial correspondence between both template and query images based on the extracted feature sets. In case of positive alignment, the overlapped area between both fingerprint impressions is deduced. The overlapped area becomes the region of interest for comparison of template and query prints in the next stage.
- Task K refers to the image matching process and the authentication result (match/non-match) computation based on the comparison of the feature sets (field orientation maps and minutia points) previously aligned.

Most of the cited tasks deal with fingerprint images and/or big amounts of data so a high computational demand is expected for the physical platform in charge of the processing. Although a first implementation of the recognition algorithm under a personal computer platform has been developed in order to validate the accuracy performance reached by the suggested algorithm, more cost-effective system solutions have also been evaluated in this work in order to make easy the spread of those fingerprint-based biometric applications in the consumer arena, accessible to whomever, wherever and whenever.

3. Recognition accuracy performance

In order to prove the validity of the suggested fingerprint recognition algorithm it is needed to proceed with the evaluation of its accuracy performance when submitted to test under a large fingerprint database. The fingerprint recognition algorithm needs to be properly tuned to the environment conditions (fingerprint sensor, sensing technique, attended/unattended acquisition method, etc.) of the real application. The selected database corresponds to the database DB3 of the Fingerprint Verification Competition FVC2004 contest (Maio et al., 2004). This public database is 110 fingers wide, and 8 samples per finger in depth, which results in a total of 880 fingerprint images. All the images were collected by using a thermal sweeping sensor. The complete database is split in two subsets A and B. The subset A is composed of 100 fingers (800 images) and the subset B is composed of 10 fingers (80 images). The subset B is firstly used in order to adjust some of the parameters of the algorithm to the properties of the fingerprint images acquired with the selected sensor, and once the algorithm is properly tuned, the subset A is used in order to verify the real performance of the application. The performance evaluation procedure follows the same criteria than in FVC contests:

i. In order to get the impostor distribution, one sample of each finger in the subset A is collected. A total of 100 images are used, and each of the images is matched against the others to compute the False Match Rate –FMR– or False Acceptance Rate –FAR– distribution. If the matching of *g* against *h* is performed, the symmetric one (i.e., *h* against *g*) is not executed in order to avoid correlation. A total of 4950 matches are carried out.

ii. In order to deduce the genuine distribution, each of the samples corresponding to one finger is matched against the other samples of the same finger. Similarly to the impostor distribution procedure, if the matching of *g* against *h* is performed, the symmetric one (i.e., *h* against *g*) is not executed in order to avoid correlation. The total number of genuine tests results in 2800, and from them it is possible to compute the False Non-Match Rate –FNMR– or False Rejection Rate –FRR– distribution.

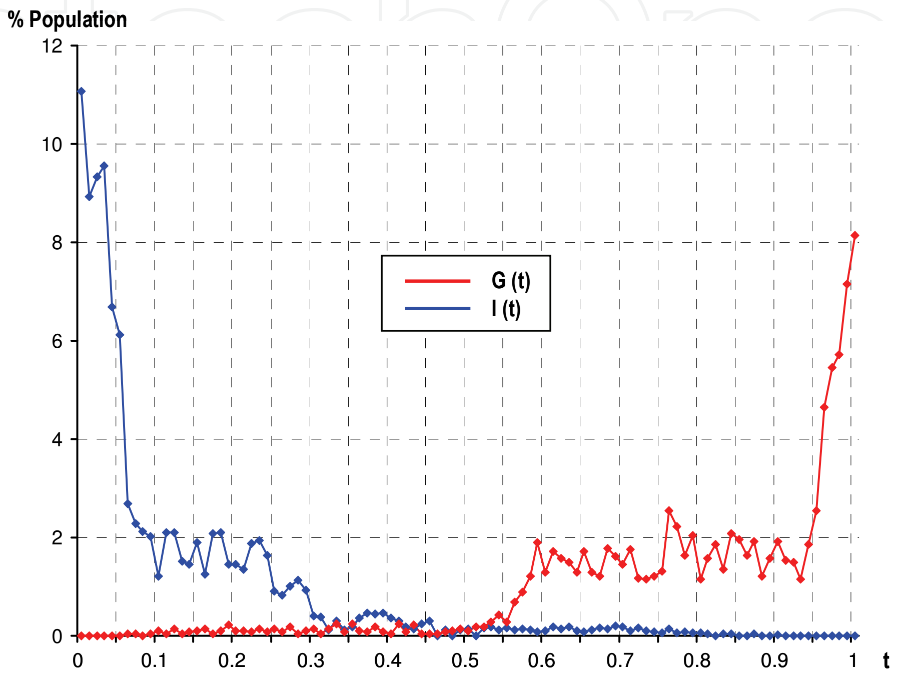

Fig. 4. Genuine and Impostor distributions

Given one template and one query fingerprints, the recognition algorithm provides a similarity score between both images within [0,1]. Similar images, understood as images belonging to the same finger, will have scores close to 1, while dissimilar images, understood as images from different fingers, will present scores close to 0. After performance evaluation with the subset A, the algorithm features an Equal Error Rate EER=4.162%. The Genuine and Impostor distributions –I(t) and G(t)–, the representations of the performance indicator rates FMR and FNMR as a function of the similarity threshold score t –FMR(t) and FNMR(t)–, and the Receiver Operating Characteristic (ROC) curve of the tested algorithm are shown in Figs. 4, 5 and 6 respectively.
The parameter EER is the main indicator used to evaluate the performance of the recognition algorithms in FVC contests. If comparing the performance of the proposed algorithm against those presented in FVC2004 with the same database, the proposed algorithm would be ranked in 17th position from a total of 41 participants in the open category (executed by one personal computer platform without resources constraints), where the winner algorithm presented an EER=1.18% and the last classified algorithm an EER=43.95%; or ranked in 5th position from a total of 26 participants in the light category (executed by a personal computer platform with restrictions on the execution time and the memory resources), where the winner algorithm presented an EER=2.92% and the last classified algorithm an EER=54.28%.

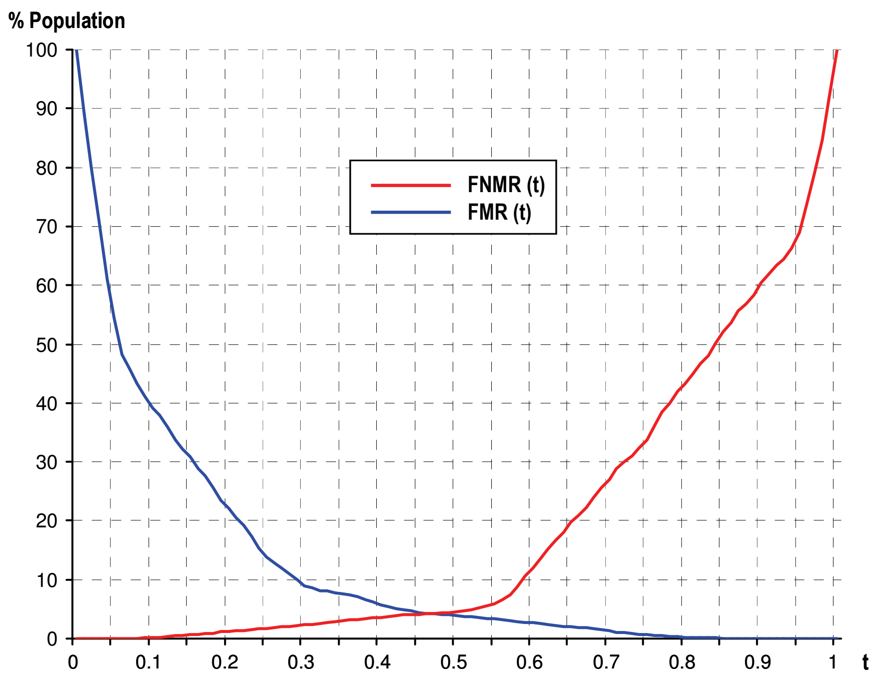

Fig. 5. False Match and False Non-Match distributions

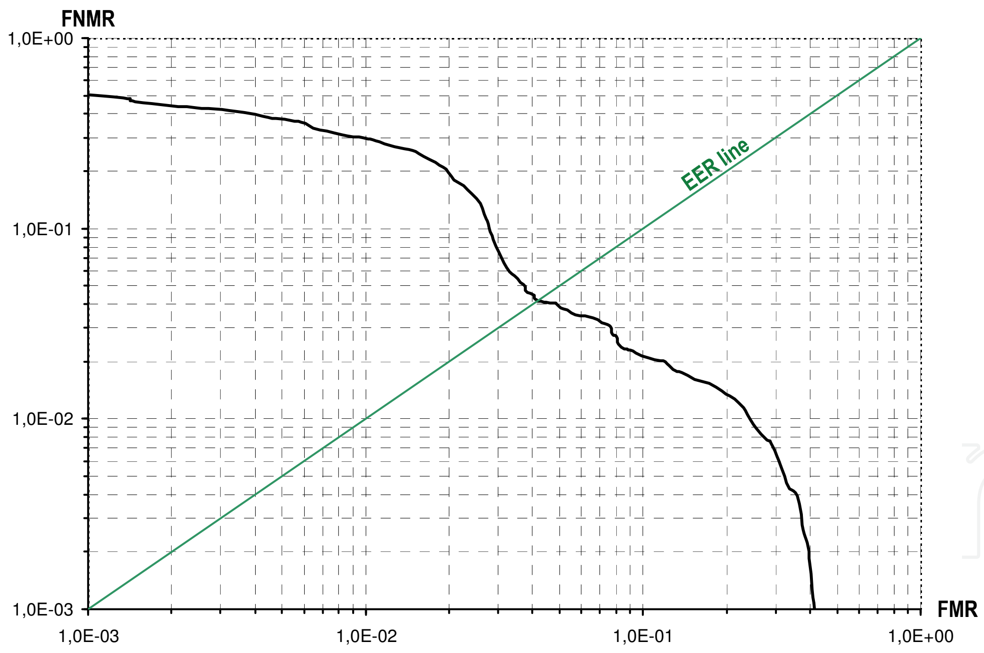

Fig. 6. Receiver Operating Characteristic curve

The first implementation of the recognition algorithm is carried out under a personal computer platform and uses floating point operations in order to be as much accurate as possible in the different computations (statistical analysis parameters like standard deviation, square root calculation, trigonometric computing, etc.) carried out along the recognition process. After proving the validity of the proposed algorithm, a new version of the algorithm is developed by replacing those floating point operations by fixed point operations in order to reduce the complexity of the processing and the computational demands of the physical platforms where to implement the AFAS application. A new evaluation performance loop of the modified version of the algorithm is performed with very similar results –the EER evolves from 4.162% to 4.242%–. Therefore the new version of the algorithm is also accepted and used as reference to be implemented under low-cost and low-performance microprocessors without floating point units (FPU) on embedded system platforms in the next stage.

4. Application execution time requirements definition

Nowadays most of the applications that exploit biometrics-based personal recognition demand a fast response time to the physical systems in charge of the processing. In case of fingerprint-based authentication systems, soft real-time performance is normally required. In this specific context, soft real-time is understood as providing the proper recognition response within a reaction time short enough to be unnoticed by the user. This reaction time covers the interval elapsed since the user presents his identity credentials to the system and puts his finger on the sensing surface of the capture device till the moment when the automatic authentication system provides the result of the verification process. Reaction times in the range between 1.5s and 3.5s are usually accepted as normal and valid authentication response times for any AFAS application. Therefore, this work focuses on the evaluation of the execution time performance of the proposed fingerprint recognition algorithm when implemented on different computational platforms in order to determine those efficient architectures able to meet the execution time requirements at the lowest possible cost.

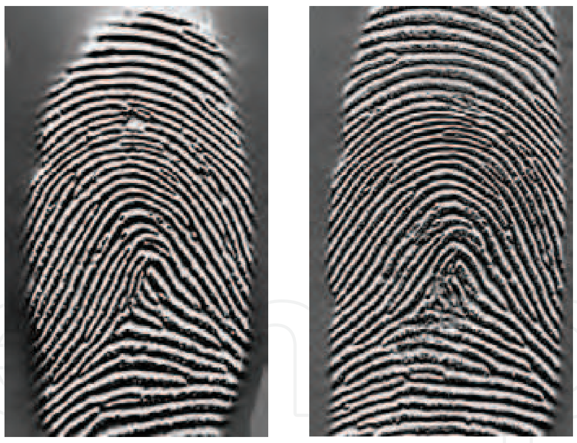
Fig. 7. Template and Query fingerprints used in the evaluation process

In order to perform a fair comparison between platforms, the same template and query fingerprints have to be used in all scenarios. Among the different images of FVC2004 DB3 database, two fingerprint impressions taken from the same finger have been selected as template and query fingerprints respectively thus it is possible to build some representative enrolment and authentication processes to be used as reference for evaluation purposes. The two greyscale images depicted in Fig. 7, of size 268x460 pixels and with a resolution of 8 bits and 500 dpi, are used as reference in order to properly compare the same processing effort in all scenarios.

5. Proof of concept I: software-only implementation

Different computational platforms addressing the execution of software-based applications have been selected for processing speed evaluation purposes. The scope covers from high-cost and high-performance personal computer platforms to low-cost and mid-performance embedded system platforms based on general-purpose hard-core or soft-core processors. One personal computer and three embedded system platforms have been evaluated, as indicated in Table 1. The evaluation procedure permits to point out in an easy way which advantages and disadvantages in performance are featured by each of the suggested architectures.

Technical Features	Personal Computer Platform	Embedded System Platform 1	Embedded System Platform 2	Embedded System Platform 3
Platform	Acer Aspire 9420	Altera Excalibur EPXA10	Xilinx Spartan 3AN	Xilinx Virtex4 ML401
Family	MPU Intel Core 2 Duo	SoPC EPXA10F1020C1	FPGA XC3S700AN	FPGA XC4VLX25
Processor	Intel Core 2 Duo T5600	ARM922T	MicroBlaze	MicroBlaze
Processor data bus	64 bits	32 bits	32 bits	32 bits
Number of cores	2	1	1	1
Type of core	Hard-core	Hard-core	Soft-core	Soft-core
Technology	65 nm	180 nm	90 nm	90 nm
Clock speed	1.83 GHz	200 MHz	66.667 MHz	100 MHz
Bus speed	667 MHz	200/100 MHz	133.3/66.6 MHz	200/100 MHz
Cache	2 MB L2	8 KB Inst. Cache	8 KB Inst. Cache 8 KB Data Cache	32 KB Inst. Cache 64 KB Data Cache
Operating system	Windows XP	–	–	–
AFAS program code	DDR2 SDRAM (2 GB)	SoPC SRAM (256 KB)	DDR2 SDRAM (64 MB)	DDR SDRAM (64 MB)
AFAS application data	DDR2 SDRAM (2 GB)	DDR SDRAM (128 MB)	DDR2 SDRAM (64 MB)	DDR SDRAM (64 MB)
SDRAM/SRAM data bus	64 bits	32 bits	16 bits	32 bits
SDRAM frequency	≥ 200MHz	125 MHz	133.333 MHz	100 MHz

Table 1. Computational platforms used in the execution time performance evaluation process

The execution time performance reached in each of the platforms, in both enrolment and authentication stages, is presented in Tables 2 and 3 respectively. The enrolment process of the template fingerprint and the authentication process of the query fingerprint with the enrolled template are evaluated. The authentication execution times are obviously longer than the enrolment times. Special attention needs to be done to the authentication stage since, unlike the enrolment stage, the authentication process is normally carried out on-line in the real application so real-time response is usually requested. The enrolment stage tends to be less critical since it is normally carried out off-line –under the supervision of application staff to guarantee the reliable enrolment of the user in the system– so no real-time performance is usually demanded.

Task ID	Processing Stage	Personal Computer Platform	Embedded System Platform 1	Embedded System Platform 2	Embedded System Platform 3
Task 1	Image segmentation	2.810 ms	1083.219 ms	299.578 ms	227.035 ms
Task 2	Image normalization	0.470 ms	178.940 ms	46.960 ms	32.772 ms
Task 3	Image isotropic filtering	7.030 ms	5304.010 ms	719.703 ms	467.329 ms
Task 4	Field orientation	2.190 ms	834.062 ms	344.651 ms	244.916 ms
Task 5	Filtered field orientation	0.620 ms	97.061 ms	26.646 ms	17.294 ms
Task 6	Image directional filtering and binarization	13.440 ms	3792.712 ms	860.133 ms	609.518 ms
Task 7	Image smoothing	12.350 ms	1536.114 ms	360.012 ms	229.732 ms
Task 8	Image thinning	1.250 ms	1695.930 ms	547.847 ms	404.085 ms
Task 9	Minutiae extraction and minutiae filtering	0.630 ms	76.626 ms	35.404 ms	23.982 ms
Total Execution Time:		**40.790 ms**	**14598.674 ms**	**3240.934 ms**	**2256.663 ms**

Table 2. Enrolment process execution time performance

As it can be deduced from the tables, the real-time performance requested to the application is not achieved in all the scenarios. The personal computer platform is able to meet the requested performance, but those other scenarios based on low-cost and mid-performance embedded processors running at low operation frequencies are far away from the requested timing performance. The big latency exhibited by the embedded system platform 1 with regard to the other two embedded system platforms is justified by the fact that no data cache is enabled in that scenario, which severely affects the final performance of the application.

On the one hand, although the powerful processor embedded in the personal computer platform is able to reach the requested performance, its cost is excessive for those low-cost consumer applications demanding biometric recognition. On the other hand, although the embedded system platforms tested in this work are able to meet the system cost requirements of the consumer applications arena, the exhibited execution time performances are clearly insufficient. Therefore, it is needed to find alternative system architectures able to meet both key requirements: high performance and low cost.

Task ID	Processing Stage	Personal Computer Platform	Embedded System Platform 1	Embedded System Platform 2	Embedded System Platform 3
Task 1	Image segmentation	2.810 ms	1083.219 ms	299.578 ms	227.035 ms
Task 2	Image normalization	0.470 ms	178.940 ms	46.960 ms	32.772 ms
Task 3	Image isotropic filtering	7.030 ms	5304.010 ms	719.703 ms	467.329 ms
Task 4	Field orientation	2.500 ms	987.089 ms	407.445 ms	289.661 ms
Task 5	Filtered field orientation	0.620 ms	113.959 ms	30.987 ms	20.171 ms
Task 6	Image directional filtering and binarization	15.940 ms	4460.569 ms	1014.939 ms	720.095 ms
Task 7	Image smoothing	14.220 ms	1752.322 ms	412.503 ms	261.745 ms
Task 8	Image thinning	1.410 ms	1767.383 ms	552.091 ms	402.946 ms
Task 9	Minutiae extraction and minutiae filtering	0.630 ms	93.783 ms	45.002 ms	29.487 ms
Task A	Field orientation maps alignment	3224.530 ms	279636.069 ms	210269.854 ms	138208.006 ms
Task B	Minutiae alignment, feature sets matching and authentication decision	4.220 ms	370.712 ms	161.973 ms	107.972 ms
Total Execution Time:		**3274.380 ms**	**295748.055 ms**	**213961.035 ms**	**140767.219 ms**

Table 3. Authentication process execution time performance

6. Run-time reconfigurable embedded system design

There exist in the market many automatic biometrics-based personal authentication systems implemented on high performance computer platforms –HPCs, PCs, etc.– (One Touch SDK, n.d.; Verifinger SDK, n.d.), embedded general-purpose or application-specific processors –MPUs, MCUs, GPUs, ASSPs– (FxIntegrator, n.d.; plusID, n.d.; SDA, n.d.), embedded digital signal processors –DSPs– (MV1210 and MV1250, n.d.; SFM, n.d.; TMS320, n.d.), or embedded systems based on central processing units –CPUs– plus application-specific hardware accelerators –ASICs– off-chip or on-chip (FPC2020 and FPC-AM3, n.d.; ML67Q5250, n.d.; SecurASIC, n.d.; TCD50D, n.d.). Furthermore, many research articles have been published dealing with the acceleration of some of the stages that take place in one personal recognition algorithm by means of field programmable logic –FPGAs, SoPCs– (Liu-Jimenez et al, 2006; Lopez-Ongil et al, 2004; Pavan Kumar et al, 2007; Yang et al, 2006). However, to the best of the authors' knowledge, up to date there is no work that takes advantage and exploits the dynamic reconfigurability performance of FPGAs (Becker et al, 2007) in the physical implementation of a complete personal recognition application based on biometrics.

Time-to-market pressures and cost constraints are pushing embedded systems to new levels of flexibility and system integration. In this work, a novel embedded system architecture is proven to successfully address the demands of today's biometrics-based personal recognition systems in terms of computational complexity, real-time performance, development cycles and cost. The proposed embedded system architecture is based on five key factors to afford the challenging demands:

a. General-purpose microprocessor system.

As in most of the embedded systems in the market today, the usage of low-cost and mid-performance microprocessors (of 16-bits or 32-bits, running at operating frequencies of up to 200-600MHz) provides certain flexibility required in any application. Software-based solutions have additional advantages such as the rapid development of the application by making use of a set of libraries with application-specific functions, which avoids writing the software application from scratch, and provides a cost-effective solution. However, in those applications demanding a high computational power and real-time performance, certain limitations exist when trying to develop the entire application with purely software platforms based on either one single processor (MPU, MCU, DSP, etc.) or multicore/multiprocessor systems due to the inherent limitations in working frequency, restricted data path, shared resources, sequential workflow execution, and reduced parallelism characteristics featured by those standard products.

b. Programmable logic device embedded in the system.

When purely software-based systems are not enough to meet the expected real-time performances of one real-world application, the usage of hardware-based accelerator devices as complementary processing units has been proven to be an efficient solution. Programmable logic devices such as FPGAs are much more flexible than semi-custom or custom devices like ASSPs or ASICs. ASSPs and ASICs have a fixed peripheral set that limits the number of applications that they can be efficiently used in; but FPGAs allow implementing custom peripherals and made-to-measure glue logic tailored to the requirements of any application. Over recent years, FPGA devices have gained an enormous amount of processing power and functionality thanks to the continuous advances in silicon technologies. The current FPGAs are able to embed much more memory and logical resources, as well as many DSP blocks, multiple clock management units and big amounts of high-speed transceivers for fast communication purposes in one single device. The technology has evolved till the point that the size of today's FPGAs is several orders of magnitude higher than the first FPGAs, reaching values above two millions of flip-flops and LUTs. The programmability performance of FPGAs make them unique in the market and the continuous improvements in the semiconductors field permits reducing the costs of FPGA devices, making them more and more competitive. The flexibility of FPGAs eliminates the long design cycle associated with ASICs, and the usage of IP libraries written in standard hardware description languages and automated design/verification tools reduce the development cycles of those applications based on programmable logic devices.

c. Hardware-software co-design techniques.

The usage of one general-purpose MPU and one FPGA as a companion chip offers a much greater degree of flexibility and allows the development of any application by means of hardware-software co-design techniques. The exposed system architecture approach gives flexibility at two levels: at software level, with the MPU-based application management; and at hardware level, with the design of modular cores synthesized in the FPGA.

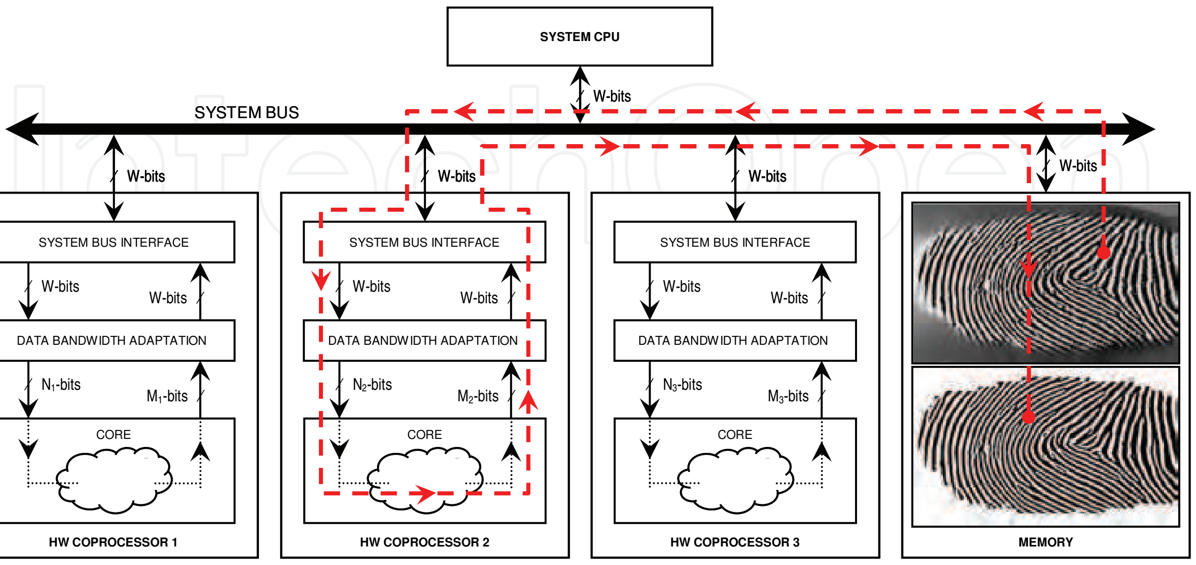

Fig. 8. Physical implementation of one computational platform based on a general-purpose MPU (system CPU), several hardware cores (HW coprocessors) and one memory block. Process execution flow example of one image processing task carried out by one of the application-specific hardware coprocessors instantiated in the system

The FPGA is introduced in the system as a general-purpose device where to instantiate those application-specific hardware coprocessors required to speed up those critical tasks of the application. It permits to design an adaptive and highly-integrated multiprocessor system oriented to the development of real-time applications. Apart from the inherent flexibility featured by the microprocessor, the programmable logic device provides additional flexibility and a high degree of parallelism in the implementation of functional circuits. In the FPGA it is possible to instantiate either additional microprocessors (e.g. VHDL instances of soft-core processors) or made-to-measure VLSI hardware accelerators in charge of specific tasks aiming at offloading those MPU algorithm-intensive operations, as shown in Fig. 8. With an improved bandwidth among the MPU –system CPU-, the FPGA, the memory resources and the rest of peripherals available in the embedded system, soft and hard real-time applications can be successfully developed through this approach.

d. Run-time reconfigurable FPGAs.

The FPGA device embedded in the system allows exploiting the parallelism and acceleration features inherent to the programmable logic design, so it is possible to meet real-time performance by spreading the functionality across the different core resources (MPU and FPGA) available in the system. However, the resources available in the FPGA are not unlimited, and the cost of those resources increases exponentially when the size of the FGPA increases. Therefore, it is convenient to reduce the size of the FPGA in the design to reach affordable costs for the complete system. In this direction, and owing to the fact that the proposed biometrics-based personal recognition applications feature a sequential nature (the personal recognition algorithm consists of a set of mutually exclusive image processing tasks executed one after the other), it is possible to exploit the reconfigurability performance featured by some FPGA devices in order to minimize the system hardware needs.

Dynamic partial reconfigurability performance of some existing FPGAs refers to the ability of modifying the functional content of one portion of the FPGA –reconfigurable region– on-the-fly while keeping the rest of the FPGA –static region– fully operative without interruption. The main benefit of doing so is the optimization in the functional density of the device: the same hardware resources available in the reconfigurable region of the FPGA can be time-multiplexed in order to allocate different functionalities (FPGA contexts) along the application execution time. Therefore the amount of needed resources in any application can be minimized, and the total size of the FPGA can be reduced in comparison to the static implementation of all the functionalities instantiated permanently in a bigger FPGA. The main constraint in the usage of run-time reconfigurable FPGAs is the reconfiguration overhead: the time needed in order to modify the functional content of the reconfigurable region in the different contexts. Therefore the minimization of the reconfiguration latencies plays an important role in those systems. Fig. 9 shows the comparison between static and dynamic FPGA-based design concepts.

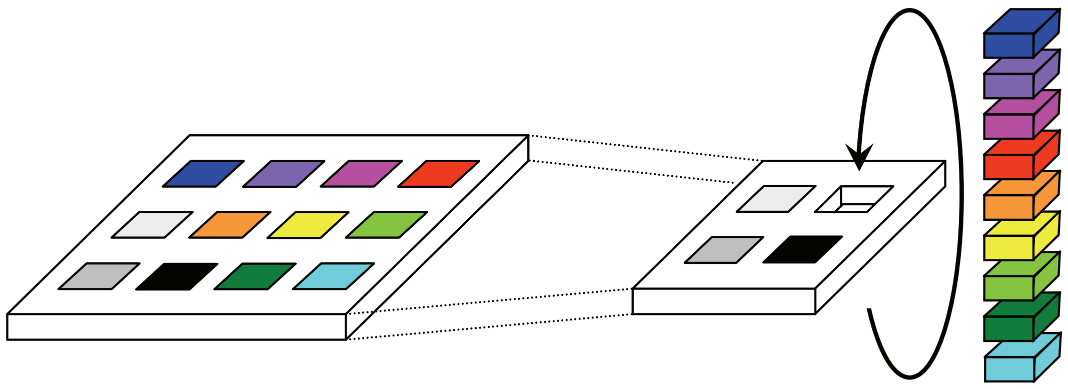

Fig. 9. Comparison between static FPGA-based design concept (left side) and run-time reconfigurable-FPGA-based design concept (right side). The coloured boxes represent each of the different functional blocks in which the application is partitioned

Any application that can be structured as a sequence of mutually exclusive tasks can be proposed to be implemented by means of run-time reconfigurable FPGAs. Fig. 10 shows the scheduling of one application into a sequence of mutually exclusive stages, and the partitioning of each of the processing stages present in the chain into either series or parallel tasks. Each of the tasks can be executed by hardware or by software. Those critical tasks are implemented by hardware to take advantage of higher processing bandwidths and acceleration data path architectures. In this direction, it is possible to make the process truly parallel and at the same time to free some master CPU resources. The rest of less expensive tasks remain as software tasks to be handled by the master CPU of the system. The final partitioning of the application into software tasks, static hardware tasks and dynamically reconfigurable hardware tasks mainly depends on the cost (resources availability, power consumption, etc.) and timing (real-time performance) constraints demanded to the system.

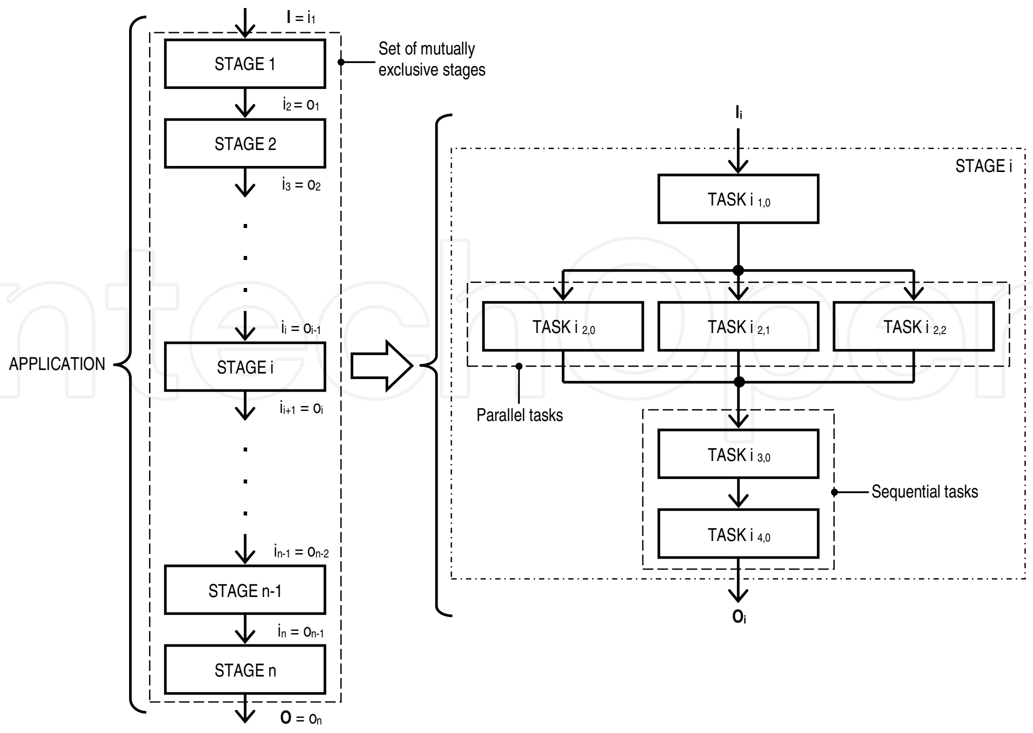

Fig. 10. Deployment of one application as a set of mutually exclusive stages that can be implemented through dynamic reconfigurable embedded systems. Partitioning of each of the stages into hardware and software tasks executed either sequentially or in parallel taking advantage of programmable logic

e. System-on-programmable chip platform.

The usage of a general-purpose MPU together with programmable and reconfigurable logic gives a high level of flexibility to the system and provides the mechanisms to achieve real-time performance. However, higher integration means lower costs. Therefore, the integration of those main resources and other key peripherals such as memory, timers, interrupt controllers, etc. on a single chip provides an efficient way of optimizing the whole system cost. Embedded biometric recognition is therefore possible by making use of highly integrated platforms. Additional benefits of the system integration are the improvements in reliability and security. It is possible to embed most of the processing in a single SoPC device well-protected against external attacks by means of security protocols and cryptographic processors dealing with the exchange of information between the SoPC device and the external world. For this reason, the usage of SoPC or system-on-chip devices that embed one FPGA is especially encouraged in the experimental tests carried out in this work.

The suggested system architecture is depicted in Fig. 11. At least one run-time reconfigurable region is present in the programmable logic device to synthesize those flexible application-specific hardware coprocessors that can be dynamically instantiated on demand along the application execution time. One specific reconfiguration controller is in charge of the reconfiguration task, supervised by the master processing unit. The AFAS application is connected to the external world by means of a series or parallel communication link with a Host. All or some of the functional blocks depicted in Fig. 11 are embedded in the same chip.

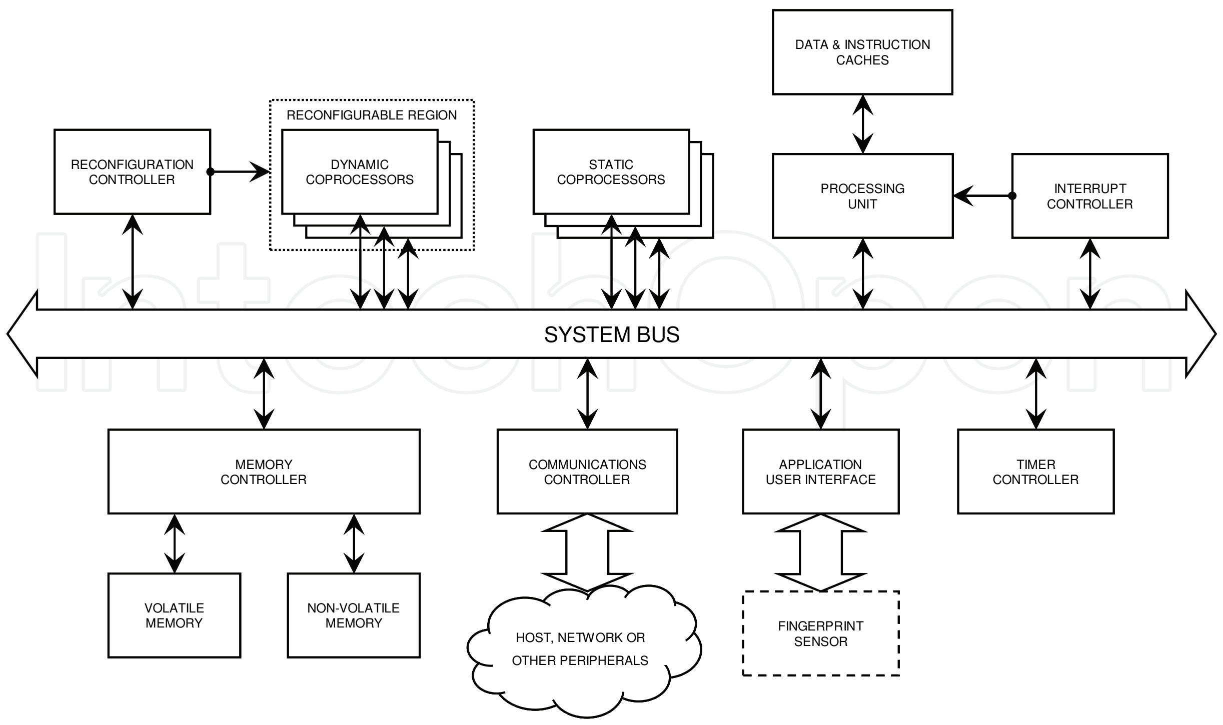

Fig. 11. Run-time reconfigurable embedded system architecture

7. Proof of concept II: hardware-software implementation

A run-time reconfigurable embedded system is presented in this section as general-purpose processing platform where to implement the AFAS application by means of hardware-software co-design techniques. A commercial development board ML401 based on the system-on-programmable-chip device Virtex-4 XC4VLX25 from Xilinx Inc. is used to verify the validity of the proposed system architecture. Additionally to the highly-integrated ML401 development platform, a fingerprint sensor has been connected to the I/O expansion ports of the evaluation board in order to make possible the acquisition of fingerprints in the application, and one RS-232 link has been established between the evaluation board and a personal computer platform in order to simulate the interface between the recognition module and the host or high-level application that makes use of the personal recognition result, as shown in Fig. 12.

The selected SoPC/FPGA device is partitioned in two regions in the biometric application: one static region and one partially reconfigurable region (PRR). In the static region, different components that will be permanently present along the application execution time are instantiated such as one 32-bit MicroBlaze soft-core processor (CPU), data and instruction caches, local memory, one memory management unit (MMU) and other memory controllers to access on-chip and off-chip memory blocks, one dedicated reconfiguration controller in charge of the dynamic reconfiguration of the device, other standard peripherals such as interrupt controller, timer, UART, general-purpose input/output ports, etc. and one specific interface between the static region and the reconfigurable region based on FIFO memories and dedicated 32-bit registers. In the reconfigurable region, application-specific hardware coprocessors will be instantiated under demand along the application execution time in order to perform those image and signal processing tasks required by the AFAS application. Table 4 shows the amount or resources available in the proposed system-on-programmable-chip and the partitioning of the device into the static and the reconfigurable regions.

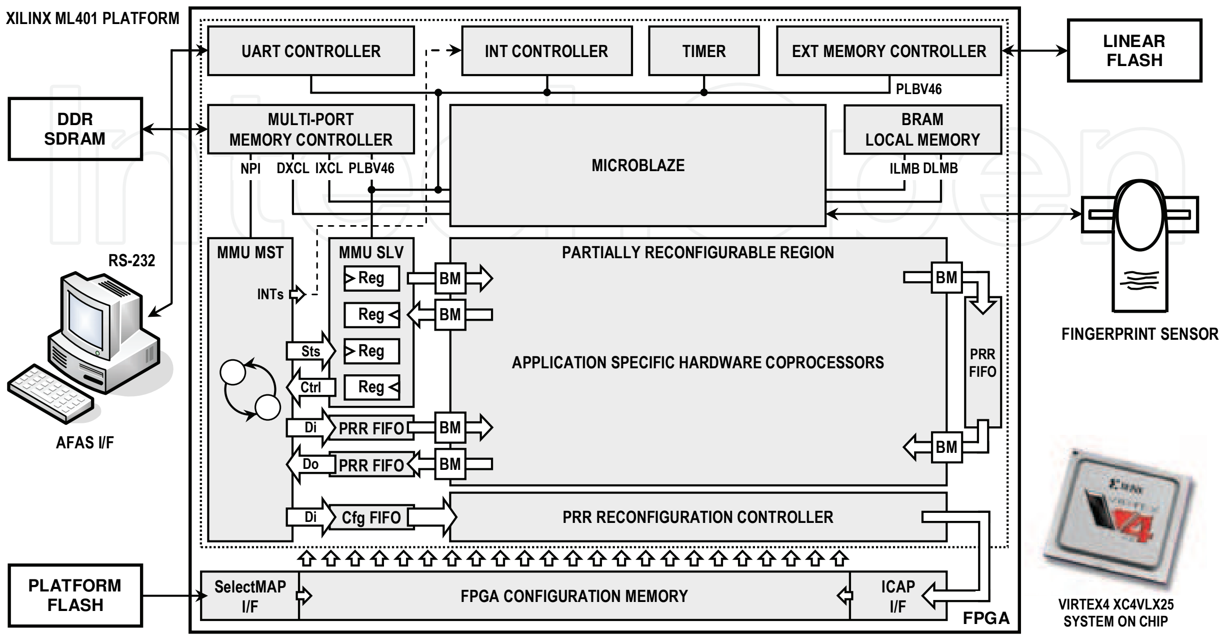

Fig. 12. Run-time reconfigurable embedded system architecture proposed in the physical implementation of an Automatic Fingerprint-based Authentication System

Resources	Xilinx XC4VLX25	Spatial Partitioning	
		Static Region	Reconfigurable Region
1-bit Flip Flop	21504	10240	11264
4-input LUT	21504	10240	11264
1-bit RAM	1327104	921600	405504
DSP Block	48	4	44

Table 4. Spatial partitioning of the programmable logic device into one static region and one reconfigurable region

The proposed system-on-programmable-chip is a SRAM-based device. Only volatile memory is embedded on the chip. Additionally to the on-chip volatile memory, the suggested platform is provided with off-chip volatile and non-volatile memory ICs, as it is shown in Fig. 12. Two different types of off-chip non-volatile memories are used:

- The Platform FLASH memory block (4 Mbytes) stores the initial bitstream that defines the configuration of the FPGA upon power up. This initial configuration is composed of the hardware content of the static region (master CPU, memory controllers and other peripherals), and one bootloader application which is executed by the master CPU and is in charge of initializing the system. The initial content of the reconfigurable region of the FPGA remains blank after power up. The transfer of the initial bitstream from the platform FLASH to the internal configuration memory of the

FPGA is automatically done during power up through a dedicated SelectMAP interface present in the FPGA.

- The Linear FLASH memory block (8 Mbytes) contains the definition of those reconfigurable hardware coprocessors to be instantiated in the reconfigurable region of the FPGA along the application execution time, as well as the AFAS program code to be executed by the master CPU. Moreover, the linear FLASH is used in the AFAS application as storage memory where to save the templates of those genuine users registered into the system in the enrolment stage. The reconfiguration process of the PRR is done by means of the dedicated hardware reconfiguration controller instantiated in the static region and the ICAP controller inherent to the device.

Apart from the off-chip FLASH memories, one off-chip SDRAM memory block is also present in the system. During the power up sequence, the bootloader is in charge of initializing the different controllers instantiated in the static region of the FPGA and transferring to the SDRAM memory block (64 Mbytes) the content of the linear FLASH, that is, the AFAS program code and the partial bitstreams that define each of the contexts in which the reconfigurable region is time-multiplexed along the AFAS application. In this way, the off-chip SDRAM memory acts as program and data memory in the application and can be accessed by either the CPU through the PLB bus or the MMU master controller through a dedicated NPI bus. Once all the information is properly transferred to the SDRAM memory, the bootloader gives the control to the AFAS application, and the AFAS application starts.

A multi-bus system architecture permits the interconnection between the different processing blocks. Two specific made-to-measure memory management units –MMU master and slave in Fig. 12– are instantiated in the static region, which aim at interfacing the master CPU and the rest of controllers provided in the static region with those reconfigurable coprocessors instantiated in the reconfigurable region. The interface between the static and reconfigurable regions is built through specific Bus Macros (BM) and some bidirectional FIFO memories intended for a fast exchange of big amounts of data. Moreover, some 32-bit registers are instantiated in the static region in order the master CPU to configure the static and reconfigurable hardware coprocessors, and to control and monitor the application processing flow. The interface between the MMU master and the PRR reconfiguration controller present in the static region is also implemented through a dedicated FIFO memory, as depicted in Fig. 12. The reconfiguration controller is in charge of reading the partial bitstreams previously saved in the SDRAM memory block during power up, and transferring them to the ICAP, which configures the reconfigurable region of the FPGA with the new functional content defined by each bistream. Another FIFO memory block is instantiated in the static region, which acts as a temporary buffer of that information that needs to be shared between different contexts of the PRR region. Before reconfiguring a new context in the PRR region, those parameters that have to be used in the next contexts are saved in that FIFO. After the reconfiguration process, the content of that dedicated FIFO is transferred again to the reconfigurable region in order the new reconfigurable coprocessors instantiated in the PRR to make use of such information.

The interface between the master CPU and those application-specific hardware coprocessors instantiated in the FPGA, either in the static or reconfigurable regions, is provided with some interrupt lines in order any of those hardware coprocessors to be able to notify to the master CPU about the end of the processing task that is being executed by hardware. Furthermore, in order to reduce the reconfiguration time of the PRR, the size of the reconfigurable region has been minimized as much as possible. A specific reconfiguration controller is instantiated in the static region of the FPGA in order to allow fast reconfiguration without impacting on CPU load. The CPU is only responsible for indicating to the reconfiguration controller the specific partial bitstream that has to be downloaded in the PRR at any time, and once this is defined, the reconfiguration controller is in charge of the reconfiguration process without the need of any further action by the master CPU. Once the reconfiguration is done, the reconfiguration controller notifies the end of the task to the CPU, and the master CPU continues driving the AFAS application program flow. The soft-core processor (master CPU) has been configured to operate at a maximum frequency of 100MHz, and the hardware coprocessors instantiated in the FPGA are designed to operate at either 100MHz or 50MHz depending on the specific task.

The required skills to develop any design based on FPGAs or SoPCs are more demanding than those needed to develop purely software applications. Some background on electronic circuits and programmable logic design, as well as the knowledge of one hardware description language like Verilog or VHDL is required to develop applications based on such kind of architectures. Similarly to what happens with software programming languages and their libraries of functions, some libraries of Intellectual Property descriptions (IPs) of certain functionalities are available to speed up the development of designs based on programmable logic. Moreover, specific EDA tools dependent on the device vendor are normally available to reduce the development cycles when designing with FPGA devices, and the designer needs to get familiar with the processing flow of each automated tool.

Although commercial non-volatile FPGAs have enjoyed great success as development, rapid-prototyping and testing platforms, their use in certain embedded applications has been limited due to their relative high cost in comparison with other solutions. At this level (using the FPGA to implement a static design which keeps invariant during all its execution), the design flow and development tools have been successfully deployed by many vendors (Altera, Actel, Atmel, Lattice, Xilinx, etc.) since decades. However, if the FPGA resources become static after configuration, the device turns into an expensive, power-hungry, low-performance on-field programmable ASIC solution. For FPGAs to become more practical as end-use devices it has been promoted their dynamic reconfiguration capability, i.e., once powered up, the FPGA can be partially reconfigured at run-time, while other part of the FPGA continues operating uninterrupted and automatically maintaining state information between two consecutive reconfigured contexts. In this way, the functions processed in the FPGA can be sequentially swapped in a similar way to the program flow of a CPU-based software application. For this more flexible FPGA conception, however, the designer needs to possess some specific background in those techniques linked to the exploitation of dynamic partial reconfiguration. Moreover, the development tools that automate the new design flow for those applications based on run-time reconfigurable hardware have been an open issue since a long time ago. Recently, however, this landscape experienced a great and definitive change. Xilinx Inc. pushed a definitive impulse to that long-time open issue related to the software tools needed in the PR design flow. Just in 2006, Xilinx presented the new PR design flow fully supported in

Virtex-4 devices. The new top-down design flow eliminated the weakest points highlighted in the previous flows. While still unreleased to the general public, these tools are nowadays presented in the way of an early access version restricted to a limited number of qualified partners who deploy them and contribute feedback to their improvement. This research work is focused on Virtex-4, the first device equipped with a level of PR performance (both technological aspects and supported development tools) acceptable for commercial perspectives. Once finished all the development of our proof-of-concept application, authors think that the current PR flow is today an accepted practice for expert developers with a deep knowledge of the FPGA low-level configuration architecture and, then, it is ready for industrial use. The current Xilinx toolset available in the Xilinx Early Access Partial Reconfiguration (EAPR) lounge and used in this work made possible to automate all the PR design methodology and finish all the phases of the design flow at a reasonable time with no concerns. The toolset used in this work is composed of EDK 9.2.02i to build the PLBv46 bus processor system, PlanAhead 9.2.7 to constrain the floorplan in a friendly graphical way, ISE 9.2.04i_PR12 to generate the bitstreams, as well as ChipScope Pro 9.2i to facilitate the system debugging. In Fig. 13 it is shown all the process to generate both partial and full bitstreams to be downloaded at run-time into the FPGA.

The application is split into a set of sequential stages, and each stage is partitioned into hardware and software tasks. Only those tasks demanding a high computational power or those time-critical tasks that would take too much time if executed by the system CPU are ported to hardware. Specific hardware coprocessors are instantiated in the reconfigurable region of the device to execute such tasks meanwhile the remaining and computationally less expensive tasks are assigned to the system CPU, which furthermore acts as the master processor in charge of driving the application, scheduling the tasks, monitoring the execution flow, and handling the reconfiguration of the PRR when needed along the authentication process. Those partially or fully pipelined hardware coprocessors instantiated in the dynamically reconfigurable region of the FPGA play the role of slave processors in charge of executing those tasks commanded by the master CPU. The dynamic hardware coprocessors are present only when they are needed, thus the same hardware resources available in the reconfigurable region are reused to instantiate different circuits in the application. In Fig. 14 it is shown how the different coprocessors are downloaded into the FPGA to reach a time-multiplexing of the resources placed in the defined PR region of the FPGA. This work results one of the first contributions in the scientific literature that exploits the Xilinx Early Access Partial Reconfiguration electronic design automation tools.

In Section 5 the algorithm has been ported to the presented embedded system (referenced as Embedded System Platform 3 in Tables 1, 2 and 3) and executed purely by software by its MicroBlaze core processor alone. No dedicated hardware was implemented in that scenario, and the application was not able to meet the demanded real-time performance. However, as a result of that implementation under a purely software-based embedded platform, it has been possible to identify those time-expensive computational tasks that constrain the real-time performance of the application in the embedded system. Those time-critical tasks identified in Section 5 are now transferred to hardware to speed up the processing. Owing to the limited resources available in the programmable logic device, up to 9 different reconfigurable contexts have been needed in order to instantiate all the hardware coprocessors along the execution time. Outstanding real-time performances are achieved.

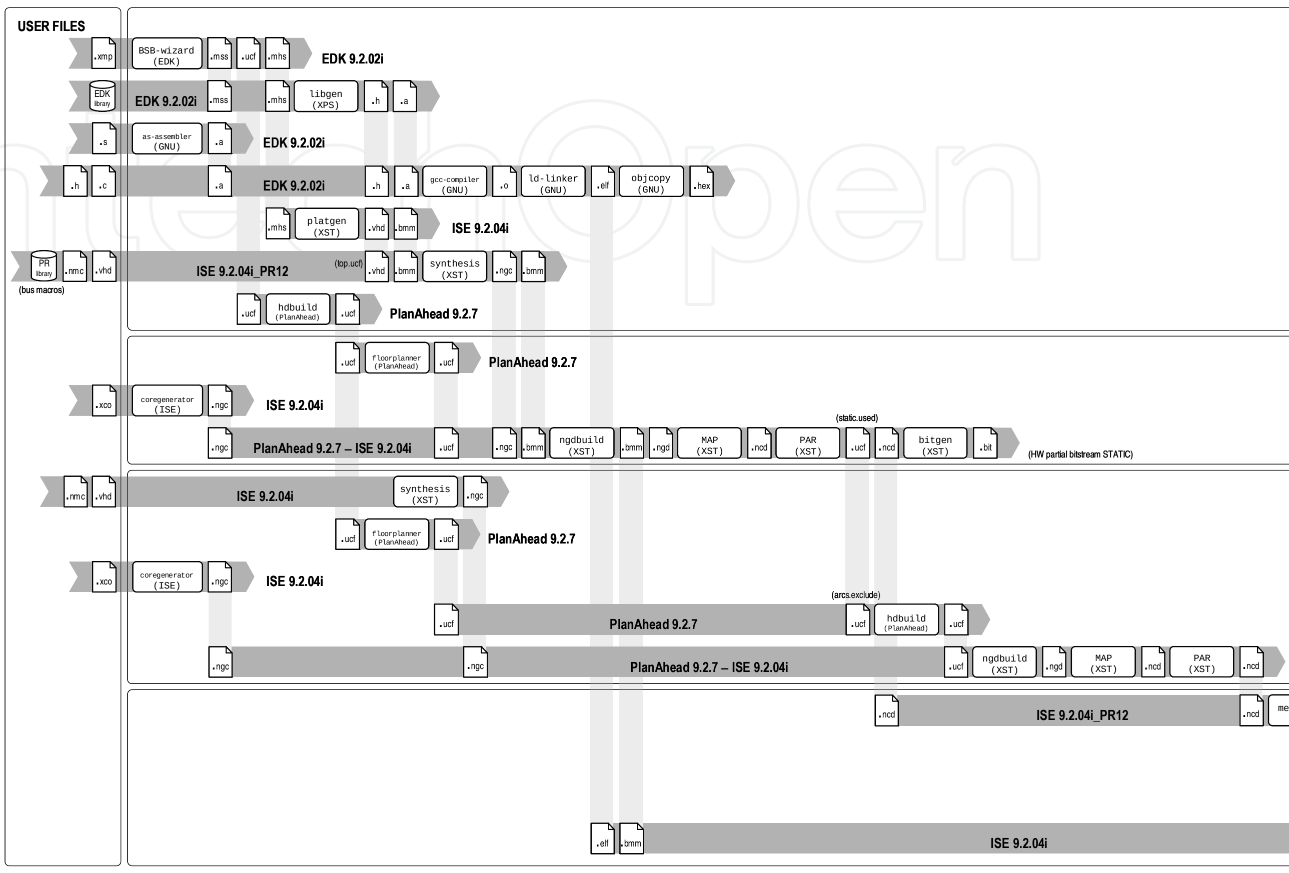

Fig. 13. PR design flow (EDA tools, source code files and resultant bitstreams)

Tables 5 and 6 provide the execution time performance of the application in both enrolment and authentication stages in two different scenarios: (i) when the application is executed purely by software under the system CPU alone, and (ii) when the application is implemented by means of hardware-software co-design techniques making use of the dynamic reconfigurability performance of the suggested FPGA. The final partitioning of the application into hardware and software tasks is also detailed. In the second scenario all tasks are ported to hardware except the fingerprint acquisition process, which is kept as software task under the action of the system CPU. The FPGA resource usage in both the static and reconfigurable regions is shown in Table 7.

Task ID	**Processing Stage**	**Software-only Implementation** (1)	**Hardware-Software Implementation**	
			Sw-only Task	**Hw-Sw Task**
Task 0	Fingerprint acquisition	500.000 ms	500.000 ms	
Task 1	Image segmentation	232.046 ms		0.672 ms
Task 2	Reconfiguration 1 → 2			0.841 ms
	Image normalization	33.087 ms		0.850 ms
Task 3	Reconfiguration 2 → 3			1.045 ms
	Image isotropic filtering	512.171 ms		2.563 ms
Task 4	Reconfiguration 3 → 4			1.025 ms
	Field orientation	285.485 ms		0.669 ms
Task 5	Reconfiguration 4 → 5			1.046 ms
	Filtered field orientation	19.143 ms		0.419 ms
Task 6	Reconfiguration 5 → 6			1.107 ms
	Image directional filtering and binarization	656.043 ms		2.465 ms
Task 7	Reconfiguration 6 → 7			1.045 ms
	Image smoothing	253.553 ms		0.447 ms
Task 8	Reconfiguration 7 → 8			0.974 ms
	Image thinning	416.316 ms		0.902 ms
Task 9	Reconfiguration 8 → 9			0.943 ms
	Minutiae extraction and minutiae filtering	25.699 ms		4.919 ms
Total Execution Time (2)**:**		**2433.543 ms**	**21.932 ms**	

(1) : The software-only execution times are slightly higher than in the Embedded System 3 scenario of Table 2 because of the reduction of the cache memory size in this new scenario in order to allocate additional memories in the hardware coprocessors (only 8KB of Instruction and Data caches are instantiated in MicroBlaze interface instead of the initial 32KB Instruction cache and 64KB Data cache).
(2) : Task 0 is not included in the computation of the total execution time.

Table 5. Execution time performance reached in the enrolment stage: SW-only versus HW-SW implementations

Task ID	Processing Stage	Software-only Implementation [(1)]	Hardware-Software Implementation	
			Sw-only Task	Hw-Sw Task
Task 0	Fingerprint acquisition	500.000 ms	500.000 ms	
Task 1	Image segmentation	232.046 ms		0.672 ms
Task 2	Reconfiguration 1 → 2			0.841 ms
	Image normalization	33.087 ms		0.850 ms
Task 3	Reconfiguration 2 → 3			1.045 ms
	Image isotropic filtering	512.171 ms		2.563 ms
Task 4	Reconfiguration 3 → 4			1.025 ms
	Field orientation	337.419 ms		0.669 ms
Task 5	Reconfiguration 4 → 5			1.046 ms
	Filtered field orientation	22.178 ms		0.419 ms
Task 6	Reconfiguration 5 → 6			1.107 ms
	Image directional filtering and binarization	774.750 ms		2.465 ms
Task 7	Reconfiguration 6 → 7			1.045 ms
	Image smoothing	287.507 ms		0.447 ms
Task 8	Reconfiguration 7 → 8			0.974 ms
	Image thinning	417.350 ms		0.820 ms
Task 9	Reconfiguration 8 → 9			0.943 ms
	Minutiae extraction and minutiae filtering	32.497 ms		7.606 ms
Task A	Reconfiguration 9 → A			1.045 ms
	Field orientation maps alignment	139935.838 ms		157.671 ms
Task B	Reconfiguration A → B			1.035 ms
	Minutiae alignment, feature sets matching and authentication decision	108.608 ms		20.737 ms
Total Execution Time [(2)]:		**142693.451 ms**	**205.025 ms**	

[(1)] : The software-only execution times are slightly higher than in the Embedded System 3 scenario of Table 3 because of the reduction of the cache memory size in this new scenario in order to allocate additional memories in the hardware coprocessors (only 8KB of Instruction and Data caches are instantiated in MicroBlaze interface instead of the initial 32KB Instruction cache and 64KB Data cache).
[(2)] : Task 0 is not included in the computation of the total execution time.

Table 6. Execution time performance reached in the authentication stage: SW-only versus HW-SW implementations

Task ID	Processing Stage	Hardware Resources			
		1-bit Flip Flop	4-input LUT	1-bit RAM	DSP Block
–	Application flow (static design)	7005	8888	755712	4
Task 0	Fingerprint acquisition	–	–	–	–
Task 1	Image segmentation	4978	4612	147456	20
Task 2	Image normalization	371	334	0	8
Task 3	Image isotropic filtering	5275	5831	92160	28
Task 4	Field orientation	3339	3166	92160	8
Task 5	Filtered field orientation	2857	2983	129024	0
Task 6	Image directional filtering and binarization	5462	4166	313344	29
Task 7	Image smoothing	4892	3265	147456	0
Task 8	Image thinning	1013	2821	239616	0
Task 9	Minutiae extraction and minutiae filtering	487	3379	55296	0
Task A	Field orientation maps alignment	2632	8943	387072	0
Task B	Minutiae alignment, feature sets matching and authentication decision	642	4379	258048	5
Total Design Resources:		**38953**	**52767**	**2617344**	**102**
Total Device Resources:		**21504**	**21504**	**1327104**	**48**

Table 7. FPGA resources usage in each of the application contexts

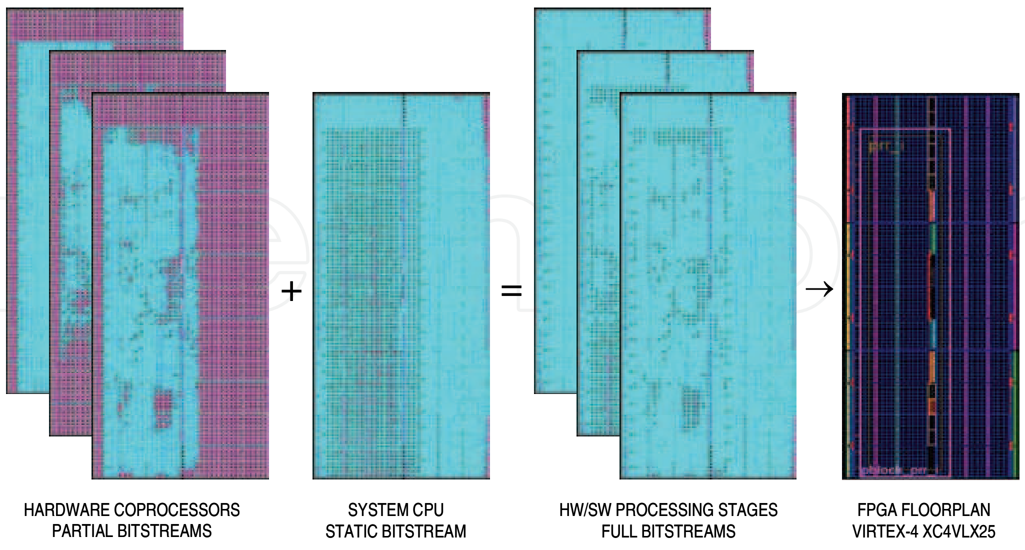

Fig. 14. Temporal partitioning of the application in sequential tasks running in the PRR of a FPGA. The bitstream gets composed of a static region and a reconfigurable region (left). Spatial partitioning of the application floorplanned in both static and reconfigurable regions on the Xilinx Virtex-4 XC4VLX25 device (right)

The physical resources needed to implement each of the hardware coprocessors are detailed. From the resources usage shown in Table 7 it can be deduced that the reconfigurability performance of the FPGA permits a notorious reduction of the amount of resources needed in the programmable logic device in comparison with the amount of resources that would be needed in case of using a non-reconfigurable FPGA, where all coprocessors would be instantiated permanently in a static way. Thanks to the reconfigurability performance exhibited by the suggested device and the hardware-software partitioning of the application it has been possible to develop one application that demands 38953 1-bit flip-flops, 52767 4-bit LUTs, 2617344 1-bit RAM cells and 102 DSP blocks with one device that features 21504 1-bit flip-flops, 21504 4-bit LUTs, 1327104 1-bit RAM cells and 48 DSP blocks. The reuse of the hardware resources allows reducing the amount of resources at the expense of the reconfiguration overhead, which is also minimized by the design of an efficient reconfiguration controller. The amount of needed resources and the reached performances exhibited by the suggested run-time reconfigurable embedded system clearly outperform those featured by one PC platform. The total authentication execution time results in 205.025 ms, which leads to a speed up of ×686.58 (or ×695.980 depending on the used cache) when compared against the purely software implementation of the recognition algorithm under the same embedded system platform, and a speed up of ×15.97 with regard to the application execution time featured by the PC platform presented in Section 5.

8. Conclusion

The successful spread of products and services that exploit the advantages provided by fingerprint biometrics in both public and private sectors depend on several factors today. Although the universality, distinctiveness and permanence characteristics of human fingerprints are proven facts that make them reliable signs of identity, the acceptance of automated fingerprint-based personal recognition systems, focused on either identification or authentication purposes, is constrained by social and technical factors. Among the social factors, the most important ones refer to the security and privacy concerns related to the protection of the user's information integrity; and among the technical factors, the most limiting ones refer to the accuracy of the recognition system, the authentication response time and the cost of the whole application. All they are barriers to the broad adoption of that kind of systems worldwide. If fingerprint recognition technology continues to mature and efficient and reliable systems able to overcome all those barriers are designed, automated fingerprint-based recognition can have a profound influence on the way we conduct our daily business in the near future.

As far as authors know at the moment of publication of the present work, there does not exist in the market any AFAS application based on dynamically reconfigurable hardware. Flexible and dynamically reconfigurable hardware allows a more efficient usage of the system resources by having hardware present in the FPGA device only when it is in use. Thus given a fixed size for the FPGA, it is possible to instantiate specific coprocessors at a given time, and to eliminate them after they have been used in order to allow further coprocessors to be instantiated making use of the same FPGA resources in the following stages of the application. This technique allows reducing the overall hardware system size at nearly null cost –FPGA reconfiguration overhead–.

The results presented in this work prove that the suggested system architecture can be an efficient alternative to those existing AFAS based on either expensive personal computer platforms or embedded systems that make use of MPUs, GPUs, DSPs, ASSPs or ASICs. This novel approach, focused on the exploitation of run-time reconfigurable FPGA devices and hardware-software co-design techniques, pursues two main objectives: (i) to meet the required expectations for the application, which means to fulfil the functionality demands (accurate FAR/FRR personal recognition rates) with the proper response time (real-time) and reliability levels (protection against fraudulent attacks); and (ii) to meet those requirements with the minimum possible cost for the system, and with the proper flexibility to allow future changes/improvements in the personal recognition algorithm (added-value). There are endless uses for embedded systems based on SoPC or FPGA devices in the consumer, military, aerospace, automotive, communications, and industrial markets worldwide. In this direction, the proposed embedded system architecture, based on run-time reconfigurable hardware, is proven to be a valid and cost-effective solution that encourages the reduction of system resources in the physical implementation of those complex computational applications demanding high processing power and real-time performances such as the ones resulting from the biometrics field. As computer technology continues to advance and economies of scale reduce costs, fingerprint biometric systems based on the suggested topology can become a more efficient and cost-effective means for personal verification in both public and private sectors. The proposed system architecture can thus help in paving the way for the exploitation of biometric systems all over the world.

9. References

Becker, J.; Hübner, M.; Hettich, G.; Constapel, R.; Eisenmann, J. & Luka, J. (2007). Dynamic and Partial FPGA Exploitation. *Proceedings of the IEEE*, Vol. 95, No. 2, (February 2007), pp. 438-452, ISSN 0018-9219

Fons, M.; Fons, F. & Cantó, E. (2010). Fingerprint Image Processing Acceleration through Run-Time Reconfigurable Hardware. *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 57, No. 12, (December 2010), pp. 991-995, ISSN 1549-7747

FPC2020 ASIC Fingerprint Processor and FPC-AM3 Biometric Module, (n.d.), 2011, Available from http://www.fingerprints.com

FxIntegrator Fingerprint Recognition Module, (n.d.), 2011, Available from http://www.biometrika.it

Liu-Jimenez, J.; Sanchez-Reillo, R. ; Lindoso, A. & Miguel-Hurtado, O. (2006). FPGA Implementation for an Iris Biometric Processor, *Proceedings of IEEE International Conference on Field Programmable Technology*, pp. 265-268, ISBN 0-7803-9729-0, Bangkok, Thailand, December 13-15, 2006

Lopez-Ongil, C.; Sanchez-Reillo, R. ; Liu-Jimenez, J.; Casado, F.; Sánchez, L. & Entrena, L. (2004). FPGA Implementation of Biometric Authentication System Based on Hand Geometry, *Proceedings of International Conference on Field Programmable Logic and Appplication*, pp. 43-53, LNCS 3203, Antwerp, Belgium, August 30 – September 1, 2004

Maio D. ; Maltoni, D. ; Cappelli, R. ; Wayman, J.L. & Jain, A.K. (2004). FVC2004 : Third Fingerprint Verification Competition, *Proceedings of International Conference on Biometric Authentication*, pp. 1-7, LNCS 3072, Hong Kong, China, July 15-17, 2004

Maltoni, D.; Maio, D.; Jain, A.K. & Prabhakar, S. (2009). *Handbook of Fingerprint Recognition, Second Edition,* Springer-Verlag, ISBN 978-1-84882-253-5, London, England

ML67Q5250 Fingerprint Authentication MCU, (n.d.), 2011, Available from http://www.okisemi.com

MV1210 and MV1250 Bioscrypt Fingerprint Modules, (n.d.), 2011, Available from http://www.l1id.com

Nanni, L. & Lumini, A. (2009). Descriptors for Image-based Fingerprint Matchers. *Experts Systems with Applications*, Vol. 36, No. 10, (December 2009), pp. 12414-12422, ISSN 0957-4174

One Touch SDK. (n.d.), 2011, Available from http://www.digitalpersona.com

Pavan Kumar, A.; Kamakoti, V. & Das, S. (2007). System-on-Programmable-Chip Implementation for On-Line Face Recognition. *Pattern Recognition Letters*, Vol. 28, No. 3, (February 2007), pp. 342-349, ISSN 0167-8655

plusID Universal Biometric Devices (n.d.), 2011, Available from http://www.privaris.com

SDA Stand-Alone Fingerprint Recognition Modules (n.d.), 2011, Available from http://www.secugen.com

SecurASIC Chip, (n.d.), 2011, Available from http://www.cogentsystems.com

SFM Series Fingerprint Modules, (n.d.), 2011, Available from http://www.supremainc.com

TCD50D Digital ID Hardware Engine, (n.d.), 2011, Available from http://www.upek.com

TMS320 Texas Instruments DSP Platforms, (n.d.), 2011, Available from http://www.ti.com

Verifinger SDK. (n.d.), 2011, Available from http://www.neurotechnology.com

Yang, S.; Sakiyama, K. & Verbauwhede, I. (2006). Efficient and Secure Fingerprint Verification for Embedded Devices. *EURASIP Journal on Applied Signal Processing*, Vol. 2006, No. 3, (January 2006), pp. 1-11

Yang, J.C. & Park, D.S. (2008). A Fingerprint Verification Algorithm Using Tessellated Invariant Moment Features. *Neurocomputing*, Vol. 71, No. 10-12, (June 2008), pp. 1939-1946, ISSN 0925-2312