HW\SW IMPLEMENTATION OF IRIS RECOGNITION ALGORITHM IN THE FPGA

Hentati Raida, YassineAoudni, Mohamed Abid

Research Laboratory CES, National school of Engineers of Sfax,3000, Tunisia

raida.hentati@enis.rnu.tn

Abstract :
Recent years have seen the growth of new pattern recognition applications which based on personal biometric characteristics. These applications are used for security, logical or physical access control, etc. Iris recognition has been successfully deployed in several large-scale public applications.
In this paper, three different methods are proposed to accelerate the simulation of biometric systems based on iris recognition with performance analysis. The whole iris recognition algorithm has been implemented in Cyclone II FPGA achieving significant reduction in execution time when compared with software implementation .The results show that with a clock speed of 50MHZ from image of 640*480, the gain in best method in the time execution is just 33% and the total time need by a software solution running on the same embedded microprocessor in the architecture.
Keywords: ***iris biometric, hardware/software implementation, FPGA***..

1. Introduction

Nowadays, the most of the applications that exploit biometrics-based personal recognition demand a fast response time. Therefore, this work focuses on the evaluation of the execution time performance of the proposed iris 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.
Mapping of C algorithm without modification in the software code on hardware, results may not be efficient or expected. Usually floating point operations are used in algorithms, but they are costly and complex in terms of hardware.
The paper describes the implementation of the real time iris recognition algorithm. Embedded iris identification system is the one of the iris identification applications.
Implementing such applications on a general purpose computer can be easier, but not very time efficient due to additional constraints on memory and other peripheral devices. Application specific hardware implementation offers much greater speed than a software implementation. With advances in the VLSI (Very Large Scale Integrated) technology hardware implementation has become an attractive alternative. Implementing complex computation tasks on hardware and by exploiting parallelism and pipelining in algorithms yield significant reduction in execution times.
There are many technologies available for hardware design. Application Specific Integrated Circuits (ASIC) and Digital signal processors (DSPs) and Field Programmable Gate Arrays (FPGA's) design offers highest performance, but the complexity and the cost associated with the design is very high. DSPs are specialized microprocessors, typically programmed in C, or with assembly code for improved performance. FPGA are reconfigurable devices. Hardware design techniques such as parallelism and pipelining techniques can be developed on a FPGA, which is not possible in dedicated DSP designs.
Our work consist on implementing biometric algorithms on reconfigurable hardware minimizes the time-to-market cost, enables rapid prototyping of complex algorithms and simplifies debugging and verification. Therefore, FPGAs are an ideal choice for implementation of real time iris recognition algorithms [22]. Our work aim to implement in HW /SW iris algorithm recognition. 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 and DSP, such as Ea, T. Valentian et al ,2005[19] ,Liu-Jimenez et al, 2009[7][8]; Xin Zhao 2009 [20] , Ryan N. Rakvic ,2010 [12],Lopez- et al, 2011[6]. In these works, an embedded systems base on iris personal recognition are proven to successfully address the demands of today's in terms of computational complexity, real-time performance, and cost.
The experimental results reported in this paper were obtained using DE2 cyclone II FPGA clocked at 50 MHZ.
This paper is organized in 6sections.. Section 2 describes target technology. Section 3 reviews briefly the basic principles underlying the iris recognition algorithm. Section 4 describes the adopted architecture. Section 5 presents the experimental results .Finally Section 6provides our conclusions.

2. FPGA Prototype Implementation

The prototyping platform was a Cyclone II FPGA with the Quartus II 9.0 synthesis software from Altera. The implementation language was VHDL. The NIOS II processor was built with the SOPC-Builder.
.The NIOS development board is shipped with a factory default 32 bits reference design. The core of the board is the ALTERA cycloneII 2C35 F672C6 FPGA [14]. Several peripheral devices and connectors (UART, VGA and Ethernet...) serve as interfaces between the cycloneII FPGA and the external environment. 8-MB DRAM, 512-KB SRAM, 4-MB Flash authorize implementation of complex applications. For the embedded system, we have used flash memory, SRAM, SDRAM, and timer . ALTERA introduces the SoPC builder tool , for quick creation and easy evaluation of embedded systems.
For SW implementation of our application, the use of a microprocessor is required. The use of HW accelerator, for optimization, affects the overall performance of the algorithm. For the highest degree of HW/SW integration, a softcore processor was used. The SoPC Builder MegaWizard is used to create NIOS embedded processor systems with 32-bits CPU core, built-in peripherals, on-and off-chip ROM and RAM support and bus support for external hardware modules. The ALTERA NIOS-II softcore processor (Fast version) is a 32-bits scalar RISC with Harvard architecture. The main feature of this softcore processor is its extensibility and adaptability. Furthermore, users can connect into the FPGA the on-chip processor and custom peripherals to a dedicated bus (Avalon Bus). Thus, users can define their instructions and processor peripherals to optimize the system for a specific application.

3. Our application: Iris recognition

Our application is an iris recognition biometric system. Iris recognition is a relatively new biometric technology, as great advantages, such as variability, stability and security, thus is the most promising for high security environment. It has many potential applications such as access control, network security, etc. We are interested in the iris biometric authentication. We notice that the iris is the round, pigmented tissue that lies behind the cornea [21]. The patterns within the iris are very unique to each person, and even the left eye is unique in comparison to the right eye. And the probability of having same iris is 10^{-72} [12] .

3.1 Image acquisition

This step is one of the most important and deciding factors for obtaining a good result. A good and clear image eliminates the process of noise removal and also helps in avoiding errors in calculation. This work uses the image provided by the National Institute of Standards and Technology (NIST) the database ICE 2005 [13]. This database consists of 2953 grayscale eye images of 132 people. They are acquired with a dedicated LG2200 camera. Each image captures one eye and has a size of 640x480 pixels.

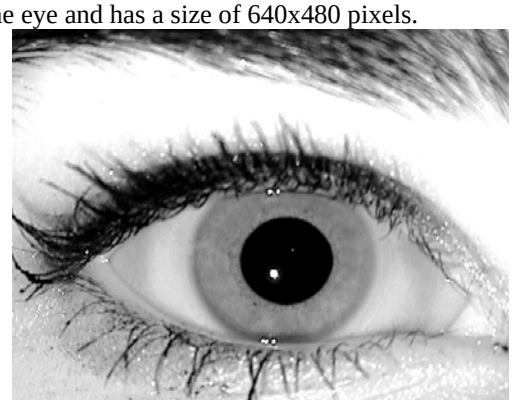

Figure 1: Example of iris images in the ICE 2005 database

3.2 Image preprocessing

First, we applied the morphological function in the image of eye. Then the image was filtered using the Gaussian filter, which blurs the image and reduces effects due to noise. Next the edge detection is followed by finding the boundaries of the iris and the pupil, we use Canny filter.

The Hough transform is another way of detecting the parameters of geometric objects. In this case, has been used to find the circles in the edge image. For every edge pixel, the points on the circles surrounding it at different radius are taken, and their weights are increased if they are edge points too, and these weights are added to the accumulator array. Thus, after all radii and edge pixels have been searched, the maximum from the accumulator array is used to find the center of the circle and its radius. The Hough transform is performed for the iris outer boundary using the whole image, and then is performed for the pupil only, instead of the whole eye, because the pupil is always inside the iris.

3.3Iris Localization

The iris part lies between the scalera and the pupil. Hence in this step is separating the iris part from the eye image. The iris inner and outer boundaries are located by finding the edge image using the Canny edge detector. The use of Hough transform, localize the iris. The eyelashes were separated by thresholding, and those pixels were marked as noisy pixels, since they do not include in the iris code.

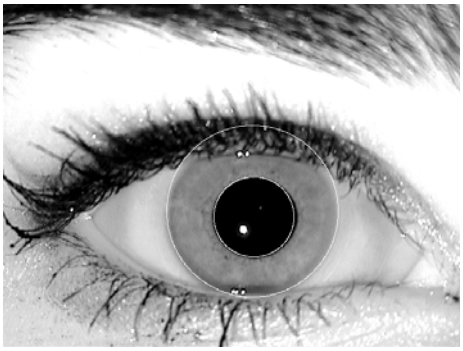

Figure 2 :Image with boundaries

3.4 Normalization

For every pixel in the iris, an equivalent position is found out on polar axes. Normalization process involves unwrapping the iris and converting it into its polar equivalent. It is done using Daugman's Rubber sheet model [16].

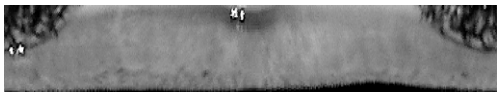

Figure 3 :Unwrapping the iris result

3.5 Encoding

The final process is the generation of the iris code. The Gabor filters are used in this step. The iriscode is formed by assigning 2 elements for each pixel of the image. Each element contains a value 1 or 0 depending on the sign + or – of the real and imaginary part respectively. Noise bits are assigned to those elements whose magnitude is very small and combined with the noisy part obtained from normalization.

$$h_{\{Re,Im\}} = \mathrm{sgn}_{\{Re,Im\}} \int_{\rho}\int_{\phi} I(\rho,\phi) e^{-i\omega(\theta_0-\phi)} \cdot e^{-(r_0-\rho)^2/\alpha^2} e^{-(\theta_0-\phi)^2/\beta^2} \rho d\rho d\phi \quad (1)$$

Where, $h_{\{Re,\,Im\}}$ has the real and imaginary part, each having the value 1 or 0, depending on which quadrant it lies.

3.6 Code Matching

Comparison of the bit patterns generated is done to check if the two irises belong to the same person. Calculation of Hamming Distance (HD) is done for this comparison. HD is a fractional measure of the number of bits disagreeing between two binary patterns. Since this code comparison uses the iriscode data . We note For a recognition algorithm, the equal error rate (EER) is a good indicator of its recognition performance. We get an EER equal to 7.14% .

4. Adopted approach

The process of hardware/software partionning starts with a modular system description, which hardware and software modules are described in VHDL and C respectively. For software modules, we use NIOSII to compile, the C description of each system module. For hardware modules, the computing time and the area are

determined using the synthesis and simulation results of the VHDL description. Quartus and ModelSim SE/EE tools are used for synthesis and simulation respectively.

The deduction of execution time is based on test bench file that generates vectors test for hardware and software modules. The software module of our hardware/software SoC is NIOSII processor. The Quartus processor is a synthesisable VHDL model of a 32-bit processor. The hardware module of our hardware/software SoC is an accelerator. This module is described with VHDL code. the figure4 shown the adopted architecture

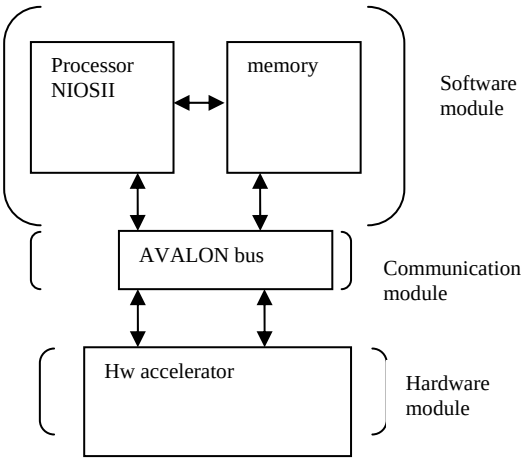

Figure 4 : adopted HW/SW architecture.

4.1 Profiling and hardware–software partitioning

Partitioning an embedded application into hardware and software parts is the first step of the implementation in the FPGA. The software parts are usually executed by an embedded processor, while the hardware parts are implemented as accelerator.Typically, tasks will not need the same amount of processing time. Thus, a computational workload analysis is considered. For this purpose, a parallel computational "Gprof" GNU profiling is performed.

gprof [2] is a Lunix instruction, that can give performance statistics for a program written in C /C++,let as to know where the program is spending most of its time, how often each function was called, and the total amount of time that each function required. The obtained results are reported in figure 5 in terms of the CPU time percentage spent in the process execution.

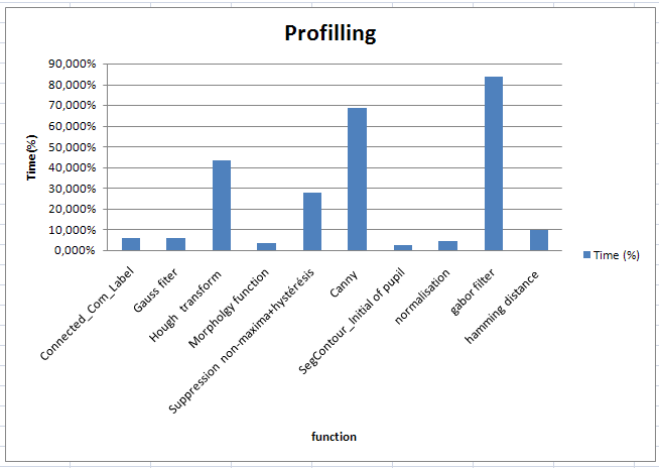

Figure 5 Profiling result of the iris recognition

We note that computational workload of this application is too much unbalanced. Some processes have negligible complexity; others especially .

The "Gabor filter" is still the most computational-expensive task with more than 40% of the total computing time complexity. This is because it continent many arithmetic operations .Therefore, we can identify the most

computational-expensive tasks are “Gabor filter”, ”Gaussian masque”, “canny”, and “hough transform”. To accelerate the biometric recognition algorithm, we used a hardware accelerator.

4.2 *C2H* compiler

The first method base on C2H compiler.We implement the Hough transform in hardware using the C2H.The compiler generate the HDL module and the interface into the system through the Avalon bus .We have adopted the architecture formed by the processor, the memory and the accelerator. After creating the hardware accelerator by the C2H, it is added to all components of the architecture and its connections with these components are established in the SOPC builder, over a VHDL file on this accelerator is added to all project files.
Table 1 shows the results of execution time and the occupation

Table 1 the Execution time and the occupation surface results of Hough transform with HW accelerator

Function	Execution times before acceleration (ms)	Execution time after acceleration (ms)	Gain in%	Total logic element	Total registers
Hough transform	10654	6392	28 ,97	13,449 (55%)	9335

This solution offers two advantages: reducing the time of the executive function and time to market, but it has a major inconvenient; it is not suitable for other functions in the code because the compiler does not accept real numbers. The second solution is described in VHDL code manually . Then the accelerators is created and added it in SPOC Buider from Altera.

4.3 HW/SW implementations

First,we create the project. Then, we add the hardware IP. The added IP presents the function described in hardware language (VHDL). It communicates with the processor through Avalon bus. After that we create the software projects, we write in language C the driver and we build the software project. This file will be loaded in the FPGA. The image is sent to SRAM memory by the DE2 Control Panel tool. The NIOS II processor communicates with HW accelerator through the Avalon bus. The accelerator treats the data while the Avalon Bus supports the control and runs the data tradeoff between blocks and memories. The accelerator can’t access to any memory only it should receive data from NIOS II and wait the processor to recuperate results.
The table 2 shows the results of execution time and the occupation surface of our system.

Table 2 Execution time and Occupation surface results

architectures	Execution times before acceleration (ms)	Execution time after acceleration (ms)	Gain in%	Total logic elements
With one accelerator(morphologies operations)	1768 ,80	1574	11.4	7.948 (19%)
With two accelerators(gauss filter+ morphologies operations)	3347	2242	33,4	33.004 (32%)

We notice that using architecture with two accelerators is more efficient than architecture with one accelerator. But it requires more occupation surface in the FPGA. As shown in table 2, the execution time of the iris recognition depends on the arithmetic operation of the HW accelerators functions.
The design properties of the presented design methods are concluded in table 3 resting on four aspects: processing rate (area), performance, power consumption and design time

Table3 Properties of design Methods

Methods	area	performance	design time	Power consumption
Using C2H compiler	+	-	++	++
Using one accelerator	+	+	-	+
Using two accelerators	-	++	-	-

We can see in table3, that design using C2H compiler provides the lowest configuration time overhead because the generate automatically the VHDL code . However, it has less performance because it isn't accepting the real number. Using the architecture with two accelerators ensured more performance than the author architecture.

5 Conclusion

Our application is a system of iris recognition biometric. Our algorithm consists of five main parts, which are image preprocessing, iris segmentation, normalization, encoding and matching. Three design methods are implemented and analyzed through comparing its on-chip area occupation, and time execution performance. The implementation results demonstrate the hardware accelerator is an effective way to increase the processing rate of time execution. However, the number of logic element is more important when using the architecture of tow accelerators. Besides, the use of tool generator as C2H can improving design time. But these tool present some weakness such as not accept the reel number.

As a perspective, we propose to study the parallelism of any parts of the algorithm to accelerate the processing. We plan also to extend our implementation in multiprocessor architecture.

References

[1] P. Wildes, S. C. Hsu R. J. Kolczynski J. R. Matey J. C. Asmuth and S. E. McBride. "Automated, noninvasive iris recognition system and method". U.S. Patent, No. 5,572,596.

[2] Susan L. Graham, Peter B. Kessler; and Marshall K. McKusick, Gprof: A Call Graph Execution Profiler, in: Proc. 1982 the SIGPLAN '82 Symposium on Compiler Construction. http://www.gnu.org/software/binutils/manual/gprof-2.9.1/

[3] C2H user guide www.altera.com.

[4] Aoudni Yassine ,Abid M, Gogniat ,PhillippeJ " custom Instruction Integration Method witnin Reconfigurable SOC and FPGA Devices"IEEE ICM, Dhahran Arabie Saudite December 2006..

[5] J. G. Daugman. Biometric personal identification system based on iris analysis. U.S.Patents No;5,291,5,560,1994.

[6] M. Lopez, J. Daugman ,E. Canto´ "Hardware recognition algorithm" –software co-design of an iris", Information Security, IET March 2011

[7] Liu-Jimenez, J.; Sanchez-Reillo, R.; Miguel-Hurtado, O.; « Improving security in ID tokens through HW/SW co-design » Security Technology (ICCST), 2010 IEEE International Carnahan Conference on Spain ,December 2010.

[8] C. Sanchez-Avila and R. Sanchez-Reillo, "Two different approaches for iris recognition using gabor filters and multiscale zero-crossing" Pattern Recognition, no. 38, vol. 2, pp: 231-240, 2005.

[9] Daugman, J. G.: "Probing the uniqueness and randomness of IrisCodes: Results from 200 billion iris pair comparisons," Proceedings of the IEEE, vol. 94, no. 11, Nov. 2006, pp 1927-1935

[10] Daugman, J. G.: "How iris recognition works," IEEE Trans. Circuits Syst. Video Technology, vol. 14, no. 1, pp. 21-30, Jan. 2004.

[11] A.K.Jain, R.M.Bolle and S.Pankanti, Eds., Biometrics: Personal Identification in a Networked Society. Norwell, MA: Kluwer,1999.

[12] Ryan N. Rakvic,1 Hau Ngo,1 Randy P. Broussard,2 and RobertW. Ives1 « Comparing an FPGA to a Cell for an Image Processing Application »EURASIP Journal on Advances in Signal Processing Volume 2010, Article ID 764838, 7 pages

[13] http://share.int-evry.fr/svnview-eph/.

[14] http://www.altera.com/literature/lit-index.html.

[15] R.Hentati ,B.Dorrizi and M.Abid "Software Implementation Of The OSIRIS Iris Recognition Algorithm In FPGA" 23rd International Conference On Microelectronics (ICM),2011

[16] J. Daugman. "Statistical richness of visual phase information : Update on recognizing *persons by iris patterns."* International Journal of Computer Vision, 2001.

[17] Rizkalla, m. E., palaniswamy, k., sinha, a. S. C., elsharkawy, m., salama, p., lyshevski, s., gundrum,h. Asic memory design of 2-d median filters. In proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems, 2000.

[18] SOPC Builder Applications ALTERA 2006, http://www.altera.com/products/software/products/sopc/sop-index.html

[19] Ea, T. Valentian, A. Rossant, F. Amiel, F. Amara, A. Algorithm implementation for iris identification. Circuits and Systems, 2005. 48th Midwest Symposium on 1207- 1210 Vol. 2 7-10 Aug. 2005

[20] ZHOU Hu-Lin, XIE Mei "Iris Biometic Processor Enhanced Module FPGA-based Design" 2010 Second International Conference on Computer Modeling and Simulation

[21] U. von Seelen, "IrisCode template compression and its effects on authentication performance," Presentation at the Biometrics Consortium Conference 2004,

[22] Aoudni Y., Abid M,., Gogniat G., Philippe J. Custom Instruction Integration Method within Reconfigurable SoC and FPGA Devices, Custom Instruction Integration Method within Reconfigurable SoC and FPGA Devices.