FPGA Implementation of a Minutiae Extraction Fingerprint Algorithm

Mariano López[1] and Enrique Cantó[2]
[1] Technical University of Catalonia, Avda. Victor Balaguer s/n, 08800 Vilanova i la Geltru, Spain, lopezg@eel.upc.edu
[2] Rovira Virgili University, Avda. Països Catalans 26, 43007 Tarragona, Spain, ecanto@etse.urv.es

Abstract- **Fingerprint recognition is one of the most common techniques used for biometric identification. Currently fingerprint technology is suitable to recognize users with high accuracy and low execution times using microprocessors able to solve algorithms with high-computational cost. However, the microprocessor's cost could make the use of fingerprint biometric conditional on specific applications. This paper presents the implementation of a whole minutiae extraction fingerprint algorithm using a Spartan-3 FPGA, as an appropriate solution for portable devices and for the low-cost consumer market. The internal architecture of the proposed embedded system is based on a soft-core microprocessor and several dedicated coprocessors designed in order to accelerate the resolution of the algorithm. Experimental results show as minutiae of fingerprint are obtained in 988 ms when an image of 256x256 pixels is analyzed.**

I. INTRODUCTION

Security is becoming an important challenge for usual activities that require high confidence levels such as access control, cash terminal or internet banking among others. The security of these systems is traditionally based on guarantying the user's identity by using identification cards or passwords, prior to give access to confidential information, relevant places or restricted resources. However, this identification method presents several disadvantages, basically due to its inherent risk of loss or robbery. Identification systems based on biometric features lack of these problems, since the user's identity is determined based on physiological or behavioral characteristics unique for each person. Fingerprint is one of the most widespread identification techniques allowing high-medium confidence rates. Additionally, the small-size and low-cost of sensors used in capture devices has contributed to increase its commercial use [1][2].

A fingerprint can be seen as a set of interleaved ridges and valleys on the surface of the finger. As Fig. 1 shows, the capture device returns an image, usually with 256 grey-levels, which consists of dark (ridges) and bright (valleys) lines. The most widespread fingerprint matching approach relies on the fact that the uniqueness of a fingerprint can be determined detecting prominent singular points known as minutiae, which are represented either by bifurcation or termination of ridges.

Nowadays the research effort in fingerprint algorithms is focused on improving their performances, basically increasing the reliability and reducing the error rates. Usually the implementation is based on a high-performance microprocessor, such as a desk computer, able to work at frequencies in the GHz range [3]. The algorithm runs on a microprocessor that sequentially executes the routines involved in the fingerprint processing. Recent advances in the field of microelectronics have improved the microprocessor computational power which allows algorithms to run with high accuracy without increasing the execution times. However, in applications related to the low-cost consumer market the device cost could determine the viability of the final product. Low-cost microprocessors, more suitable in the consumer's device market, are generally too slow for applications requiring intensive computations such as those involved in minutiae fingerprint extraction.

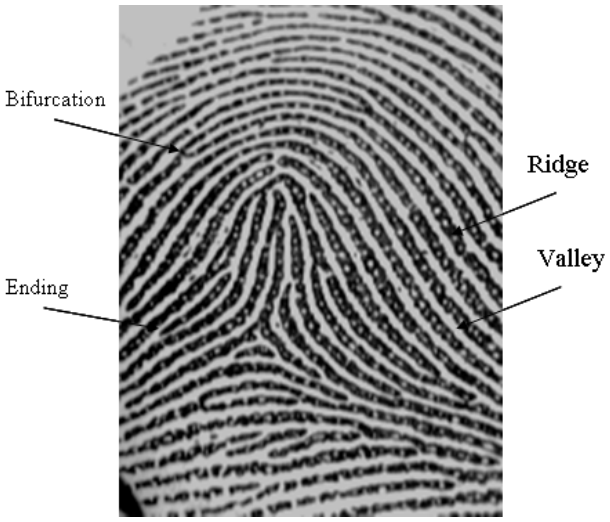

Fig. 1. Acquired fingerprint image.

A dedicated coprocessor allows designers to accelerate the processing time in those applications whose implementation can be carried out using high-speed parallel processing or several pipeline stages. There are several examples that show as dedicated coprocessor performances can overcome those offered by microprocessors in fingerprint processing [4]-[7]. However, coprocessors require a major design effort, especially for algorithms based on floating point computations or when sequential operations hamper the application of parallelism. In these cases a software implementation on microprocessor is likely better than in dedicated hardware.

The underlying concept of a hardware-software co-design is to take advantage from both hardware and software implementations. The system architecture consists in a low-cost microprocessor, which executes sequential or simple operations, and several hardware coprocessors that speed-up

978-1-4244-1666-0/08/$25.00 ©2008 IEEE

the tasks responsible of the high execution times. Moreover, these architectures can be implemented in low-cost devices like FPGAs (Field Programmable Gate Arrays) with the additional benefits related to this technology such as short time-to-market and NRE (Non-Recurring Engineering) cost.

There are several publications related to FPGA implementations of fingerprint processing. Reference [4] was one of the first published works dealing with this issue. The authors presented a special-purpose hardware accelerator based on FPGA technology able to reach a matching speed of the order of 10^5 matches per second. The paper explores the advantages of parallelism showing as it is possible to obtain performances close to ASIC at affordable price. Subsequently, other papers were published using more advanced FPGA. For example in [5], it is presented a Hw/Sw architecture implemented on Virtex II and consists of a general purpose embedded 32-bit microprocessor and hardware accelerators The minutiae extraction algorithm proposed reaches a 55% execution time reduction when compared to a traditional software implementation. However, the total execution time is about 4 s. More recently, in [6] was published a hardware accelerator for fingerprint analysis based on Maio's algorithm, obtaining a significant improvement against its software implementation based on ARM microprocessor. In [7], only part of a whole fingerprint algorithm, devoted to improve the quality of fingerprint images, was implemented in an embedded system consists in a microprocessor and a hardware accelerator that allows a resolution of the algorithm four times faster than the only software execution.

This paper presents a hardware-software co-design of a whole fingerprint algorithm implemented in a Xilinx low-cost Spartan 3 FPGA. The main contribution of this paper is that the proposed system is able to process a fingerprint image obtaining a minutiae pattern in less than 1 s. The architecture is based on Microblaze soft-core microprocessor and three hardware accelerators designed to speed-up the fingerprint processing time.

II. MINUTIAE EXTRACTION ALGORITHM

The minutiae extraction algorithm processes the fingerprint image in several stages in order to find the singular points related to bifurcation and termination of ridges. The number of stages and the processing involved in each one differs slightly depending on the algorithm employed, being in our case six stages that are briefly described in this section.

A. Normalization

The first step in fingerprint analysis is to standardize the pixel intensity by adjusting the range of grey-level values to a determined mean and variance [8]. This step is important since facilitates the processing of subsequent stages where a typical threshold that depends on the intensity and contrast of the image is used. Let I(x,y) the grey-level value at pixel (x,y), and M_o and V_o the desired mean and variance. The normalized image $I_N(x,y)$ is obtained as follows:

$$I_N(x,y) = \begin{cases} M_o + \sqrt{\dfrac{V_o \cdot (I(x,y) - M)^2}{V}} & \text{if } I(x,y) \geq M \\ M_o - \sqrt{\dfrac{V_o \cdot (I(x,y) - M)^2}{V}} & \text{if } I(x,y) < M \end{cases} \quad (1)$$

where M and V are the estimated mean and variance of I(x,y).

B. Segmentation

The aim of segmentation is to separate the foreground from the background areas. The foreground is associated with the region that contains information of interest with ridges and valleys. The background area does not contain valid information and it corresponds to the region outside the borders of the fingerprint. As it can be seen in Fig. 2.a the background presents a very low grey-scale variance, whereas due to the presence of ridges and valleys the foreground exhibits a high variance. The method based on variance thresholding is employed to perform the segmentation [9]. The variance V of a pixel (x,y) is defined as:

$$V(x,y) = \frac{1}{M \cdot N} \sum_{i=-M/2}^{M/2} \sum_{j=-N/2}^{N/2} \left(I(x+i, y+j) - E_{xy}\right)^2 \quad (2)$$

being I(x,y) the pixel intensity, M and N the size of the block used to calculated the variance, and E_{xy} the average mean of the intensity associated with this block:

$$E_{xy} = \frac{1}{M \cdot N} \sum_{i=-M/2}^{M/2} \sum_{j=-N/2}^{N/2} I(x+i, y+j) \quad (3)$$

If the variance result of a block is lower than a threshold, the pixel and its surrounding neighbors in a window of size W x L are segmented. The segmentation of the surrounded area is carried out assuming that grey-level of pixels belonging to a small window W x L have a similar intensity. Using this simplification the computational cost is substantially reduced without affecting the accuracy of subsequent stages.

After this processing the resulting image (in our case images of 256x256 pixels) usually presents several insolated clusters of pixels which segmentation needs a specific processing. The elimination of these clusters, in order to obtain a more compact image, can be achieved by applying two morphological transforms known in computer vision as opening and closing, leading to the final result shown in Fig 2.b.

C. Ridge extraction

The outcome of this stage is a binary image where the value of each pixel could be 0 or 1. Pixel set to 0 corresponds to a foreground ridge, whereas pixel set to 1 is associated with a background valley. The first step consists of an enhancement of the image by obtaining a better definition between the ridges and valleys. For that each pixel of the image is convolved with a Gabor filter. Gabor filters are employed because they have frequency and orientation selective properties. A two-dimensional Gabor filter consists of a sinusoidal wave with a

particular orientation and frequency modulated by a Gaussian envelope:

$$G(x, y, \phi, f) = \cos\left(2\pi f x_{\phi}\right)\exp\left(\frac{-1}{2}\left[\frac{x_{\phi}^2}{\sigma_x^2}+\frac{y_{\phi}^2}{\sigma_y^2}\right]\right) \quad (4)$$

$$x_{\phi} = x\cos\phi + y\sin\phi \quad (5)$$

$$y_{\phi} = -x\sin\phi + y\cos\phi \quad (6)$$

By selecting a Gabor filter tuned at the spatial frequency of the ridges and a properly orientation its response can be optimized in order to maximize the ridge-valley structure by reducing the noise of the image. Before filtering, it is necessary to estimate the orientation ϕ of the ridges contained in the fingerprint. The orientation image is calculated with the improved Rao algorithm method described in [8]. The calculation of angle ϕ is carried out in a window of size N x M centered at pixel (i,j). For each pixel of the window the gradients $\partial_x(i, j)$ and $\partial_y(i, j)$ are calculated (in our case using the Sobel operator) and its orientation is found by applying the following expression:

$$\phi(i, j) = \frac{1}{2}\operatorname{arctg}\left(\frac{V_y(i, j)}{V_x(i, j)}\right) \quad (7)$$

being

$$V_x(i, j) = \sum_{u=i-\frac{N}{2}}^{u=i+\frac{N}{2}}\sum_{v=j-\frac{M}{2}}^{v=j+\frac{M}{2}} 2\partial_x(u, v)\partial_y(u, v) \quad (8)$$

$$V_y(i, j) = \sum_{u=i-\frac{N}{2}}^{u=i+\frac{N}{2}}\sum_{v=j-\frac{M}{2}}^{v=j+\frac{M}{2}} \partial_x^2(u, v)\partial_y^2(u, v) \quad (9)$$

In order to speed-up the processing time the orientation of the central and surrounding pixels in a block W x L are considered identical. The result of the orientation is approximated by one of a set of 16 discrete angle values between -90º and 90º. After the filtering, the image is binarized by comparing the resulting grey-level value with a threshold (see Fig 2.c).

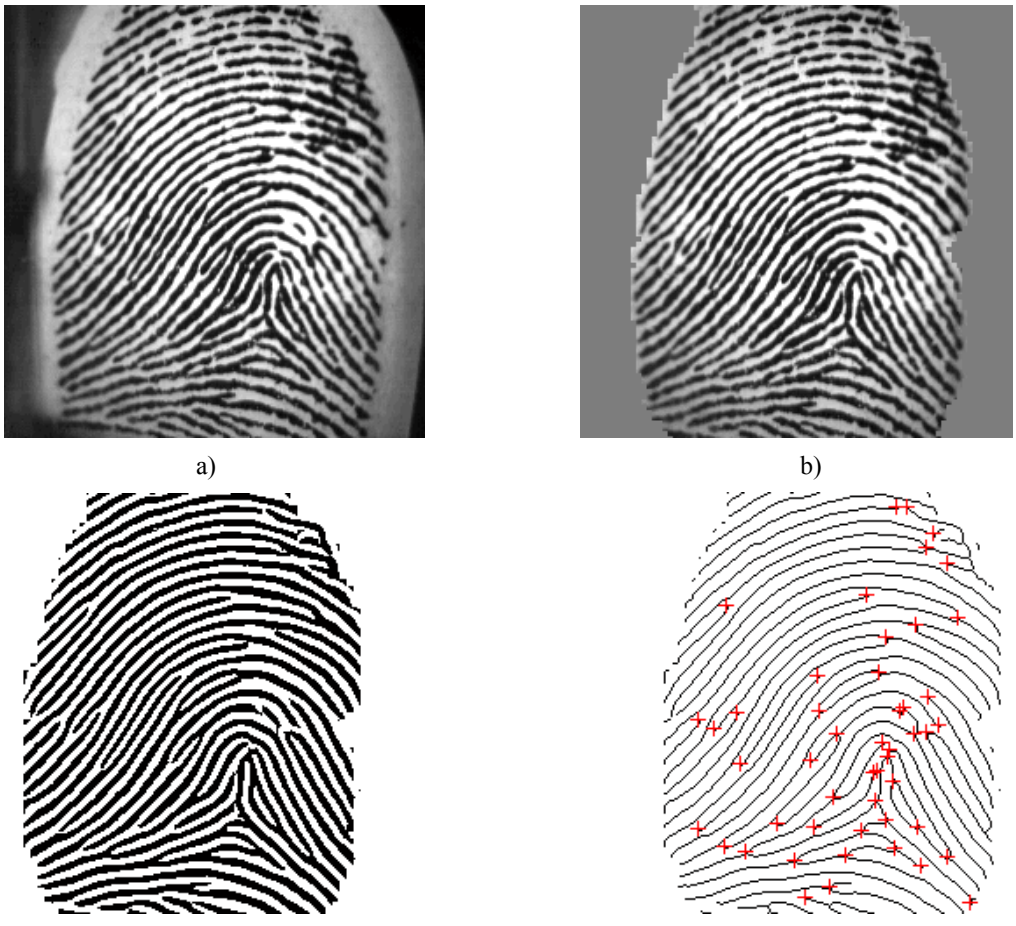

Fig. 2. Stages involved in the fingerprint signature extraction: a) Original fingerprint, b) Segmented fingerprint, c) Ridge extraction and d) Thinning and minutiae extraction

D. Thinning

Thinning is performed prior to minutiae extraction. The thinning process consists of a set of iterative morphological operations that reduces the width of ridges until they are one pixel wide. The Zhang-Suen was the parallel thinning algorithm used to perform this stage. The processing is based on the evaluation of the 8 adjacent neighbors to a central pixel that determines whether this central pixel is deleted. However, after applying this algorithm the thinned image presents some diagonal lines that have a wide pixel higher than 1, producing a visual aspect similar to a stair. This effect can be eliminated by applying a second algorithm that offers a better definition that facilitates the extraction of minutiae (see [11] for a wider explanation).

E. Minutiae Extraction

The minutiae extraction is a very simple task that can be carried out by examining the connectivity of the pixels in the thinned image. A pixel with a connectivity of 1 corresponds to an ending minutiae; a connectivity of 3 leads to a bifurcation, whereas otherwise is not a significant pixel.

F. Post-processing stage

After running the minutiae extraction it is necessary a post-processing stage that eliminates false minutiae. The presence in the thinned image of undesired spikes and breaks leads to many spurious minutiae that are detected and removed based on heuristics methods [8].

III. ALGORITHM PROFILING AND HARDWARE-SOFTWARE PARTITIONING

The whole minutiae extraction algorithm described in section II was programmed and tested using a data base of 168 fingerprints images related to 21 users (8 samples for each user). The algorithm was executed on an Intel Centrino 1.7 GHz high-performance microprocessor obtaining the average execution times for a single fingerprint image shown in table I. The table also shows the execution times on Microblaze, since this is the microprocessor used in the proposed embedded architecture.

TABLE I
EXECUTION TIMES OF MINUTIAE EXTRACTION ALGORITHM

Stage	Execution Time on 1.7GHz Intel Centrino	Execution time on Microblaze at 40MHz
Normalization	4 ms	306 ms
Segmentation	168 ms	1535 ms
Ridge extraction	300 ms	12022 ms
Thinning	186 ms	1281 ms
Minutiae extraction	9.5 ms	124 ms
Post-processing	0.5 ms	64 ms
Total Execution Time	668 ms	15336 ms

TABLE II
PARAMETERS AND VALUES USED IN THE ALGORITHM

Stage	Parameters
Normalization	Mean=100, Variance=125
Segmentation	M=8, N=8, W=3, L=3
Ridge extraction	M=9, N=9, W=4, L=4

The size of blocks and the main values for parameters used in the calculations of each stage are shown in table II. On the other hand, the root square of (1) has been substituted by a faster linear expression, obtained from the Taylor series of the same function truncated to the first term of the series $\left(\sqrt{x} \cong \sqrt{a} + (x-a)/2\sqrt{a}\right)$.

As it can be seen segmentation, ridge extraction and thinning are by far the stages with the higher execution time requiring about the 97% of the processing time when executed on Intel or Microblaze. Moreover, the computations involved in these stages can be accelerated using parallel processing and pipeline schemes. Thus, taking into account such as considerations the embedded system is composed of a Microblaze processor and three coprocessors to speed-up the three higher time-consumption stages. The rest of the stages, normalization, minutiae extraction and post-processing, have a very low computational cost being executed by software on the microprocessor.

IV. COPROCESSORS ARCHITECTURE

The coprocessors have been designed in order to work with an external SRAM memory. This SRAM is necessary, since the size of the internal FPGA memory is too small to allocate the original and the successive processed fingerprint images obtained when algorithm is executed.

Microblaze is based on a Harvard architecture that consists of two different buses. The LMB (Local Machine Bus) is the faster bus, which connects the microprocessor to an internal on-chip memory usually used to targeted program instructions. The OPB (On chip Peripheral Bus) is a slower bus, normally employed to access input/output peripherals such as SRAM memory controller [12].

The internal hardware structure of the whole system is depicted in Fig. 3. As figure shows, both microprocessor and hardware coprocessors share the external SRAM through a SRAM memory controller whose input signals are managed by a multiplex that drives these signals to the coprocessors or the microprocessor. The CS signal is activated when an address belonging to the memory map of the SRAM memory is presented on the OPB bus. In such case, depending on the two least significant bits of the OPB address, the control of the lines is transferred to coprocessors. Otherwise, CS is set to 0 and the input signals of the memory controller are directly connected to OPB bus.

TABLE III
AREA AND CRITICAL PATH FOR EACH COPROCESSOR

Coprocessor	Area (CLB's slices)	Maximum clock speed
Segmentation	1274	63 MHz
Ridge extraction	1964	52 MHz
Thinning	196	101 MHz
SRAM controller	73	147 MHz

Table III shows the coprocessors and SRAM memory controller synthesis results using Leonardo Spectrum for a Spartan 3 FPGA.

V. RESULTS

The coprocessors have been designed using VHDL hardware description language. The implementation of the whole system was carried out employing the development tools of Xilinx and the architecture were synthesized for a Spartan 3 XC3S2000 with a clock frequency of 40MHz. The external SRAM memory used to store data and images were the asynchronous CY7C1062AV33-12BGC.

Table IV shows the execution times for the stages solved by software and those implemented in dedicated hardware. The execution times of the stages running on Microblaze are identical to those shown in Table I. Approximately hardware and software take the 50% of the total time leading to an overall execution time of 988 ms. Other remarkable result is that thinning and segmentation coprocessors are faster than their software implementations on Intel, even working at a clock frequency almost 43 times slower. In these two cases the internal structure of these algorithms is very suitable to obtain an optimized hardware implementation.

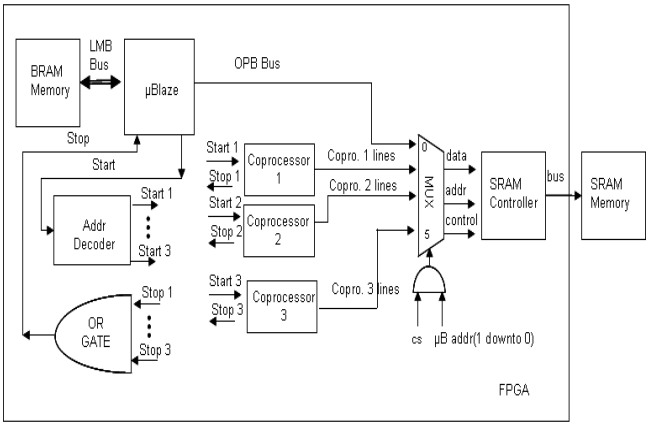

Fig. 3. Scheme for the embedded architecture.

TABLE IV
HARDWARE-SOFTWARE EXECUTION TIMES FOR MINUTIAE EXTRACTION ALGORITHM

Stage	Execution time on Microblaze at 40MHz	Coprocessors resolution time	% acceleration Hw versus Sw execution
Normalization	306 ms	--	--
Segmentation	1535 ms	130,1 ms	91.5 %
Ridge extraction	12022 ms	327,4 ms	97.2 %
Thinning	1281 ms	36,3 ms	97.1 %
Minutiae extraction	124 ms	--	--
Post-processing	64 ms	--	--
Execution time on Sw or Hw	494 ms	493,8 ms	--
Total Execution Time	**987,8 ms**		

As it was mentioned on section III, segmentation, ridge extraction and thinning represent the 97% of the total execution time on Microblaze. So it's clear that these stages are suitable candidates to be implemented in hardware due to their computational cost. However, depending on the design criterion it is possible as well to design in hardware additional stages. Under a point of view of acceleration the most efficient solution is a hardware implementation of all stages. In contrast, this solution entails a higher design effort along with increasing the area occupied by the embedded system. The design effort can be relief purchasing a set of cores provided by various vendors. The coprocessor can be carried out as a top design unit built by convenient instantiation of these basic cores. However this time design reduction involves increasing the final price. Thus, the decision about the optimal partitioning depends on a set of design constraints that take into account hardware and software implementations trade-off. In our particular case, the main goal was to reduce the total execution time below 1 second with the minimum number of hardware coprocessors and with maximum reduction of the design effort.

VI. CONCLUSIONS

A hardware-software co-design of a fingerprint minutiae extraction algorithm was presented. The architecture of the presented system is based on an embedded low-cost microprocessor and several coprocessors units that speed-up the execution time of the whole algorithm. The results show as the proposed system is able to solve all the stages involved in a fingerprint algorithm in 988 ms, presenting similar performances to those offered by high-performance and cost microprocessors. These performances can be

obtained if the system is implemented with internal dedicated coprocessors specifically designed to solve those stages that present the higher-computational cost.

ACKNOWLEDGMENT

Authors thank financial support from Ministerio de Educación y Ciencia de España, under grant TEC2006-12365-C02-02.

REFERENCES

[1] D. Maltoni, D. Maio, A.K. Jain and S Prabhakar, *Handbook of fingerprint Recognition,* Springer Verlag, 2003

[2] A. K. Jain, R. Bolle and S. Pankanti, Biometrics: Personal identification in networked society, Kluwer Academic publishers, 1999.

[3] Fingerprint Verification Competition 2006 (FVC2006). Available: http://bias.csr.unibo.it/fvc

[4] N. Ratha, D. Rover and A.K. Jain, "Fingerprint Matching on Splash 2," FPGAs in a Custom Computing Machine, D. Buell, J. Arnold and W Kleinfolder (eds.) IEEE Computer Society Press, 1996, pp. 117-140.

[5] S. Yang, K. Sakiyama and I. Verbauwhebe, "A Compact and Efficient Fingerprint Verification System for Secure Embedded Devices," IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2005), March 2005, pp. 609-612.

[6] E. Canto, N. Canyelles, M. López, M. Fons and F. Fons, "Coprocessor of the ridge line following fingerprint algorithm", XIX Conference on Design of Circuits Integrated Systems, Bordeaux (France), 2004., pp. 139-143 .

[7] M. Lopez, E. Cantó and M. Fons, "Hardware-software co-design of a fingerprint image enhancement algorithm," 32nd Annual Conference of the IEEE Industrial Electronics, Paris, France, Nov. 2006.

[8] L. Hong, Automatic personal identification using fingerprints, Ph. Dissertation, Michigan State University, June 1998.

[9] L. Hong, Y. Wan and A. K. Jain, "Fingerprint image enhancement", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, nº 8, 1998, pp. 777-789.

[10] T. Y. Zhang, C.Y. Suen, "A Fast Parallel Algorithm for Thinning Digital Patterns", Communications of the ACM, pp.236-239, Vol. 27, Num. 3, March 1984

[11] C.M. Holt, A. Stewart,M. Clint, R.H. Perrot, "An Improved Fast Parallel Thinning Algorithm", Communications of the ACM, pp.156-160, Vol. 30 Iss. 2, 1987.

[12] Xilinx Inc., "Microblaze processor reference guide", June, 2004.