Journal of Information Technology Management

Print ISSN: 2008-5893

Online ISSN: 2423-5059

Implementation of Face Recognition Algorithm on Fields Programmable Gate Array Card

Fatima Zohra Allam

Signal and Communication Laboratory, Department of Electronics, National Polytechnic School, Algeria. ORCID: 0000-0002-2273-8685. E-mail: fatima_zohra.allam@g.enp.edu.dz.

Latifa Hamami-Mitiche

Signal and Communication Laboratory, Department of Electronics, National Polytechnic School, Algeria. ORCID: 0000-0002-4697-8288. E-mail: latifa.hamami@g.enp.edu.dz.

Hicham Bousbia-Salah

Signal and Communication Laboratory, Department of Electronics, National Polytechnic School, Algeria. ORCID: 0000-0002-6416-5491. E-mail: hicham.bousbia-salah@g.enp.edu.dz.

Abstract

The evolution of today's application technologies requires a certain level of robustness, reliability and ease of integration. We choose the Fields Programmable Gate Array (FPGA) hardware description language to implement the facial recognition algorithm based on "Eigen faces" using Principal Component Analysis. In this paper, we first present an overview of the PCA used for facial recognition, then use a VHSIC Hardware Description Language (VHDL) simulation and design platform, which is the ISE. We describe the operation of each block and implement, thereafter, the computation of the global centered images. This corresponds to the first step of the PCA algorithm to assess its performance. The comparison of the results of this implementation with that of MATLAB confirmed the operability and effectiveness of this method for centralizing images. We also implemented the last part of this algorithm which is the computation of the Manhattan distance. The tests have given very satisfactory results.

Keywords: Fields programmable gate array, VHSIC Hardware description language, Principal component analysis, Manhattan distance.

DOI: 10.22059/jitm.2020.75790

Introduction

The identification of individuals using their biometric characteristics extracted from the face arouses major interest for researchers in the development and improvement of biometric recognition systems. Nowadays, it is desirable to have a face identification system that meets a certain number of criteria related to robustness, autonomy, speed and precision. An FPGA can provide the resources necessary to achieve such performance in face identification.

A reconfigurable FPGA circuit enables this type of algorithm to be implemented so that it can be used in real time and using the processing and storage capacities available on the board used.

In this paper, we will present, in section 1, the FPGA board type SPARTAN 601 and the VHDL programming language. Thereafter, we will briefly present in section 2, the principle of PCA. Section 3 describes the algorithm for computing centered images. The simulation of the different modules of the FPGA board is explained in section 4 and the RTL analysis in section 5. The algorithm for computing average images is developed in section 6. While the behavioral simulation of the global scheme as well as the interpretation of the results were the subject of sections 7 and 8. The results obtained are given in section 9.

FPGA Circuits and VHDL Programming Language

FPGA (Fields Programmable Gate Array) are electronic components of the PLD (Programmable Logic Devices) family, fully reconfigurable. This allows them to be reprogrammed at will in order to significantly speed up certain computation phases. It is a set of elementary logic blocks that can be interconnected to perform various logic functions. The density of the doors is high and constantly changing (Masato-Inagi et al., 2010), (Monmasson & Cirstea, 2007), (Rodrequez-Andina et al., 2007) and (Skliarova & Ferrari, 2000).

The main advantage of an FPGA lies in the flexibility of its technology which allows it to be reused at will in different algorithms in a relatively short time (a few milliseconds) for different algorithms.

FPGA are made up of a network of logic blocks, memory blocks, dedicated blocks and input / output blocks. Logic blocks make it possible to perform operations with a few variables through LUTs (Look Up Table) (Monmasson & Cirstea, 2007). The RAM memory blocks make it possible to install addressable memories and FIFOs. The dedicated blocks make it possible to carry out numerous processing operations (DSP blocks), to manage the clock or communication interfaces. Input / output ports, on the other hand, allow the user to connect FPGA-based hardware platform devices (Clarke et al., 2006), (Deschamps & Bioul, 2006) and (Pistorius & Hutton, 2003).

FPGA programming is done via a hardware description programming language such as VHDL (Siguenza-Tortosa & Nurmi, 2002) and (Taraate, 2017). The VHDL is a development tool that transforms said description into a file that can be configured in four main steps (Betz & Rose, 1997), (Chen & Chang, 2017) and (Tang et al., 2013):

- Mapping: group the components obtained during the synthesis in specific blocks of the FPGA.
- Placement: choose specific locations on the FPGA where to place the blocks used and choose the pins of the FPGA corresponding to the input / output ports.
- Routing: establish electrical connections between the blocks used.
- Configuration: convert the information into a file that can be downloaded to the FPGA circuit to program it.

Procedure of Principal Component Analysis (PCA)

Principal Component Analysis, applied to facial recognition, can be broken down into two phases (Morizet, 2009), (Jian et al., 2004) and (Pentland et al., 1994):

a) Learning phase

The learning phase is an important step in the design of a classifier. The algorithm of the learning phase is illustrated in figure 1. The parts framed in red will be the object of the implementation on FPGA.

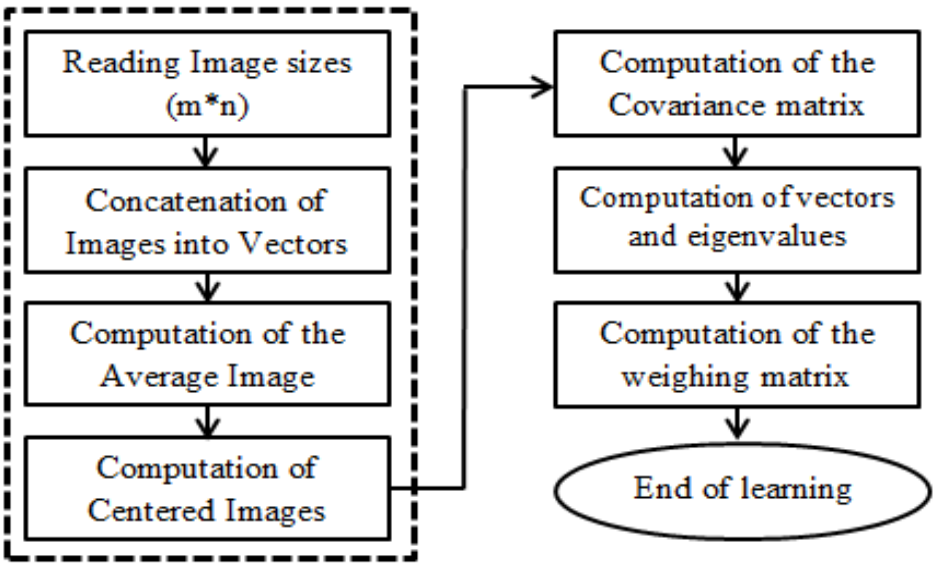


Figure 1. Algorithm of the learning phase

b) Decision phase

In this step, the test image will be compared to a set of images belonging to a database by computing the minimum distance between the eigenfaces. Figure 2 gives the algorithm of the decision phase.

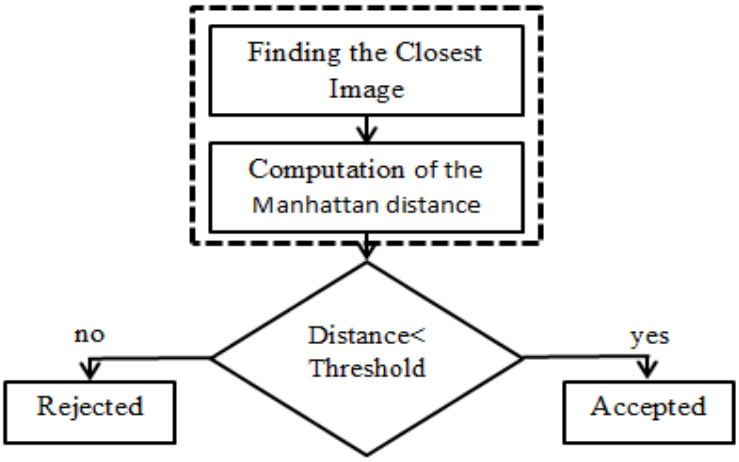


Figure 2. Algorithm of the decision phase

In what follows, we will explain the blocks that will be implemented on the FPGA card

Algorithm for Computing Centered Images

The centered images are computed from the database stored in the ROM. Figure 3 shows the overall block diagram. The parts framed in dotted lines will be implemented on the FPGA card.

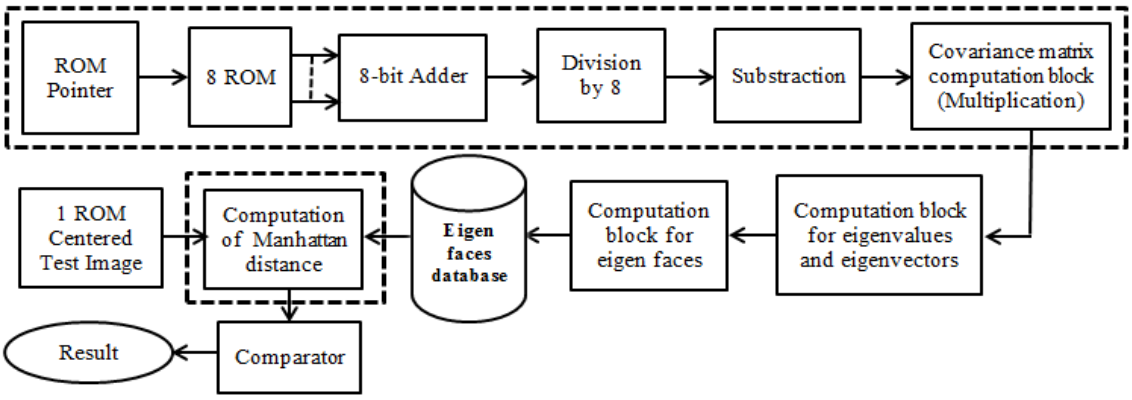


Figure 3. Diagram of computation of centered images

The implementation of the framed bets is illustrated in Figure 4:

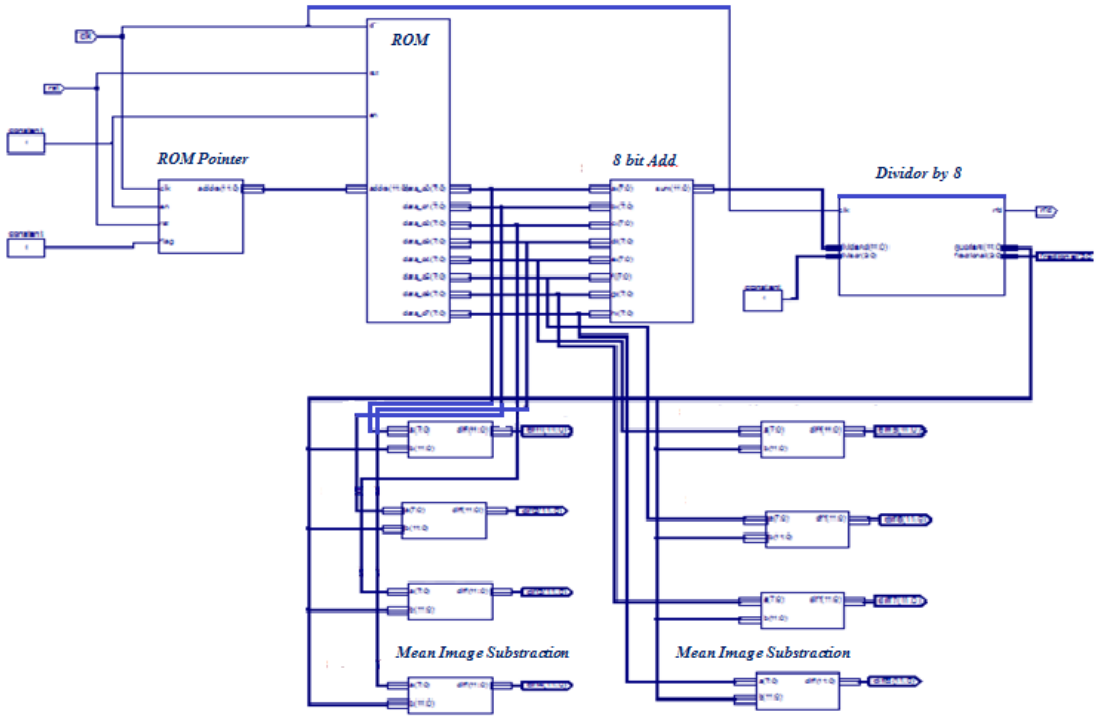


Figure 4. Algorithm for computing centered images on ISE

The circuit has four identical input signals and eight identical output signals:

- The "RESET" input resets the different modules.
- The "CLK" input ensures synchronization and control of the different modules.
- The “ENABLE” signal controls the simultaneous reading of all ROMs.
- The "FLAG" signal represents the end of reading signal.
- The "RFD" signal indicates the end of division of the nth value.
- The eight "DIFF" outputs represent the pixel values of the centered image.

Description and Simulation of the Different Modules

In this section, we will explain the methodology we adopted in our work. The figure 2 shows the overall architecture of our system.

Our work consists of programming an algorithm which computes the centered images starting from the images recorded in the database CASIA(Alaslani & Elrefaei, 2018), (Boulkenafet et al., 2016), (Feng et al., 2017) and (Komulainen et al., 2014). We have selected eight images from the ORL database. The size of a CASIA image is around 1728 × 2352, or 4064256 pixels for a single vector concatenated with a single image.

The increase in memory space leads to the increase in consumption of resources. We have reduced the dimensions in order to keep as much information as possible (without losing resolution) by using the MATLAB functionalities.

The prototyping board SPARTAN-6 SP601 (Alfke, 2009), (Challouf & Hicham, 2007) and (Jun & Wei, 2010) does not support more than 9011200 pixels per ROM (maximum memory space).

We will now describe and implement the most important modules:

a) Eight ROM memories

The capacity of the eight ROMs is 4096 words. Each word is coded on 8 bits because the intensities of the images vary between 0 and 255 (grayscale images).

b) ROM address pointer

It is a module which, at its output, delivers the address of the memory box located on ROMs. At each rising edge of the clock, a counter is incremented and assigns, at the output of the module, the address data which will point to a memory box in the ROM. The activation inputs "ENABLE" and "FLAG" must be set to "1".

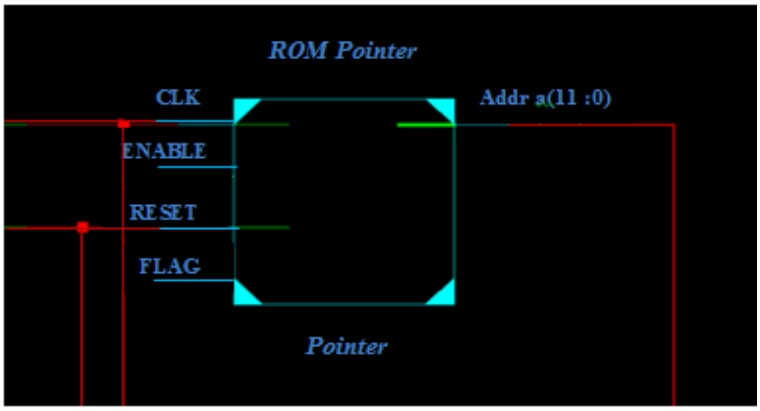


Figure 5. Schematic representation of the pointer under ISE

Figure 6 shows that the pointer works correctly. On each rising edge of the clock, the pointer increases by 1 and points to the next address.

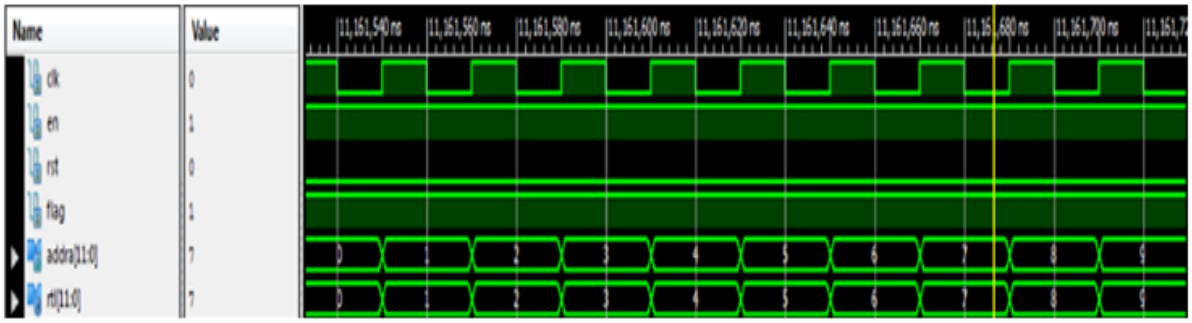


Figure 6. Pointer simulation results

c) ROM

On each rising edge of the clock, if the activation input "ENABLE" is set to "1", the "RESET" to "0" and a memory address is pointed through the address input (12 bits), the data located at the address pointed to is transmitted to the "Data" output.

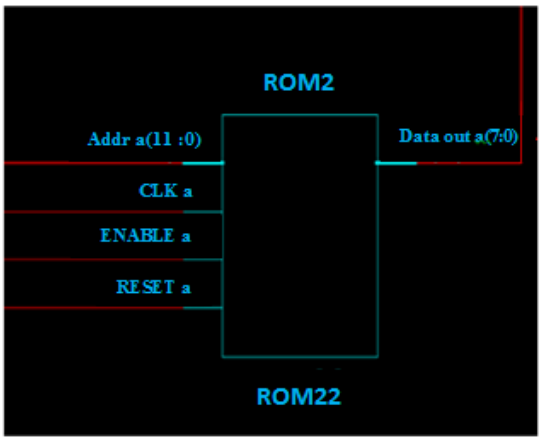


Figure 7. Structural description of ROM under ISE

The results of the behavioral simulation of each ROM were compared with the results computed by MATLAB. The ROM outputs correspond to the stored data and the order of each is maintained, which shows that the pointing is done correctly.

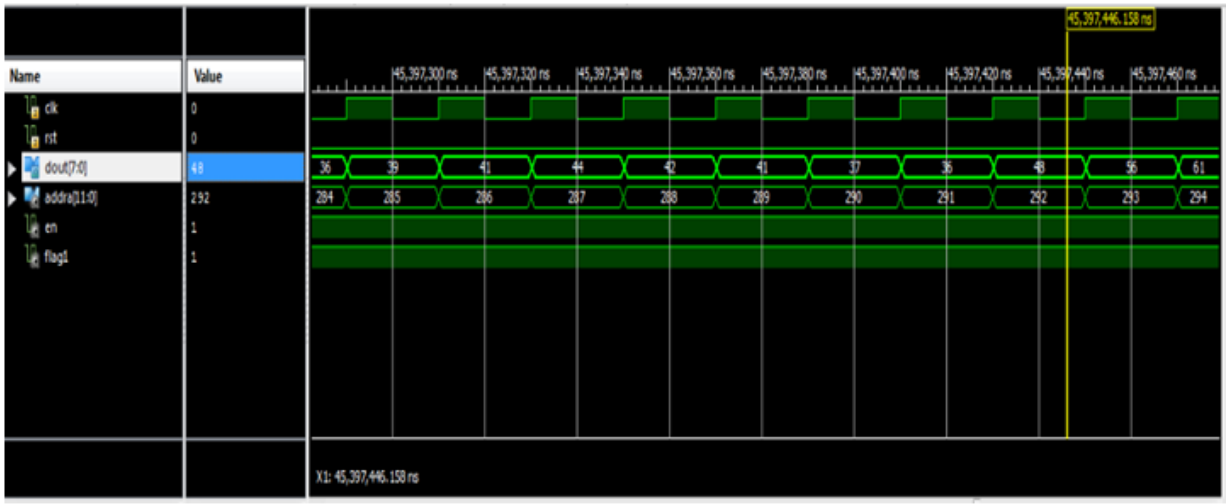


Figure 8. Simulation of ROM under ISE Xilinx

d) Eight bit adder

At each rising edge of the clock, the eight values of the 8 bits from the ROMs are added to the 8-bit adder. The output is on 12 bits to avoid any overshooting and thus avoid a loss of information.

By comparing with the results of MATLAB, we notice that the results are the same but with a slight shift of 12 positions. This lag has no influence on the rest of the treatment.

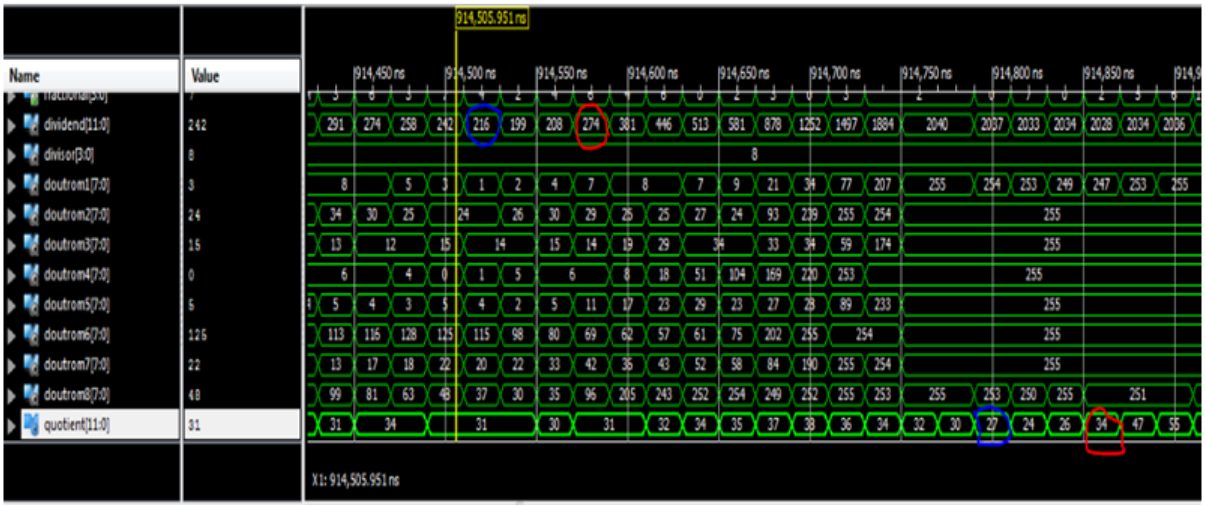


Figure 9. Behavioral simulation of the adder under ISE

e) Divider test

The division operation on VHDL using the arithmetic libraries is not synthesizable. We therefore opted for preprogrammed synthesizable VHDL modules (Bezerra & Lettnin, 2013) provided by ISE (Jaafar et al., 2017) to perform the division.

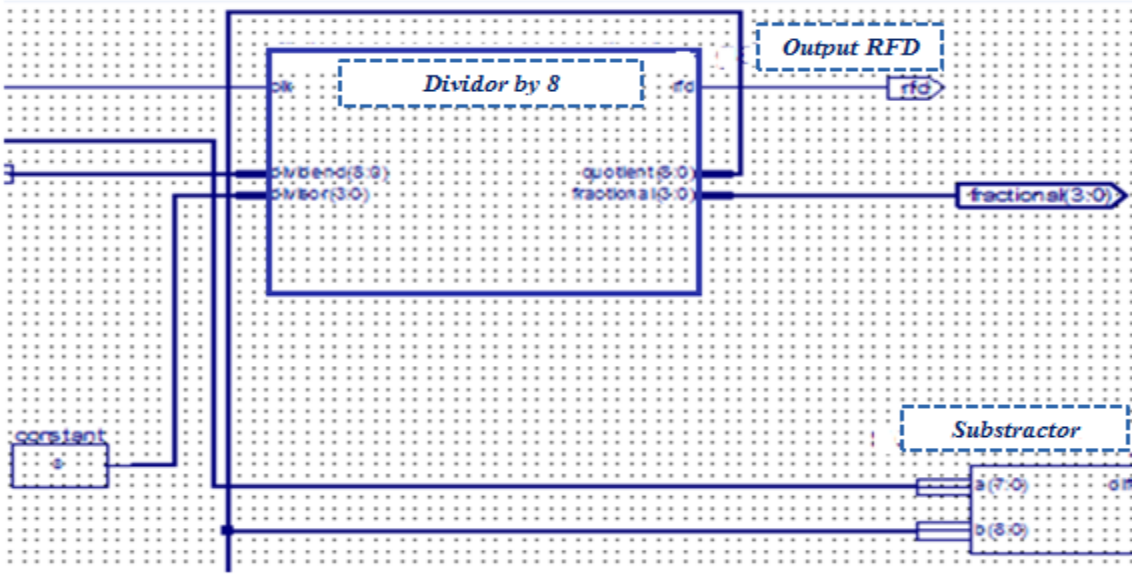


Figure 10. Schematic description of the divider

A slight lag is present, however no influence on the results is noted.

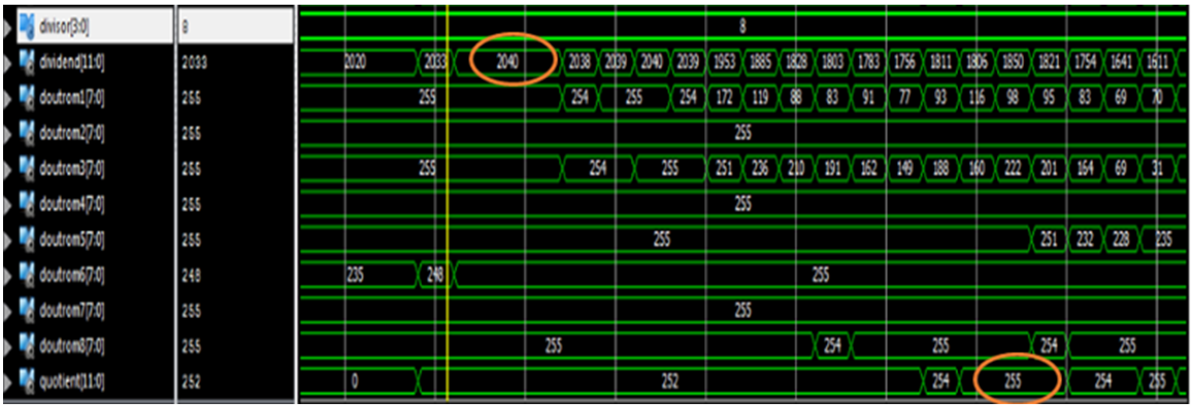


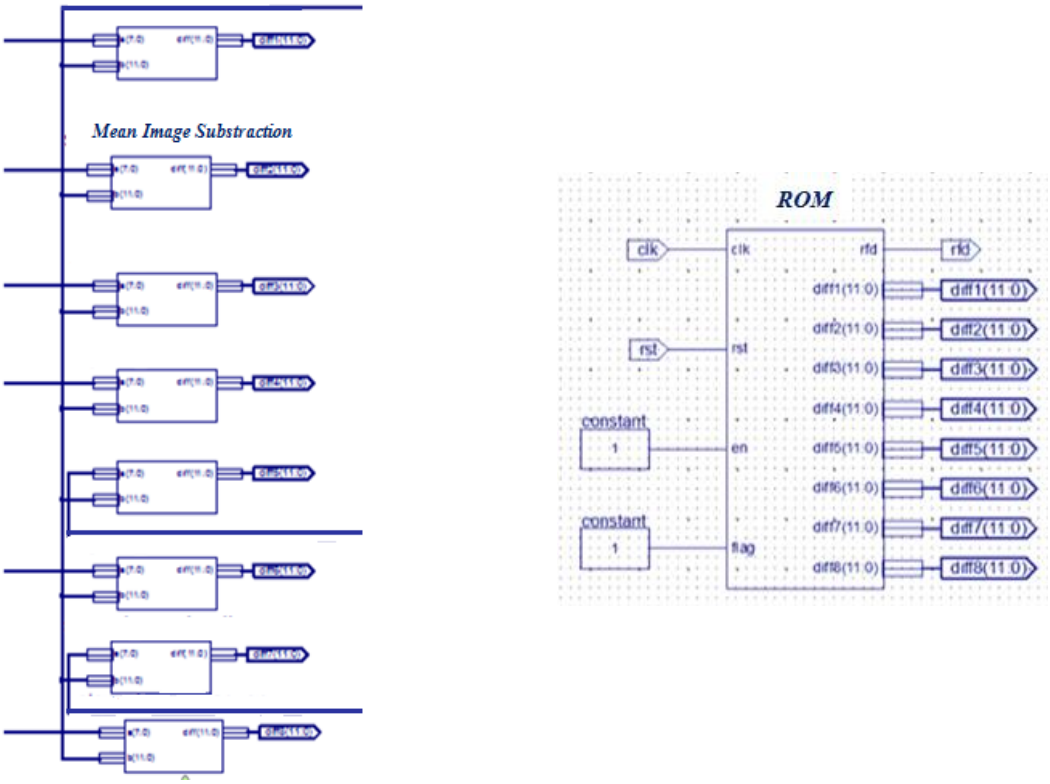
Figure 11. Results of the behavioral simulation of the divisor

f) Subtractor

There are three types of description in VHDL (Salcic, 2001) and (Taraate, 2017):

- The behavioral description.
- The structural description.
- The description of data flow.

In this approach, we used the three types of description.



(a) Structural description (b) Behavioral description

Figure 12. Schematic representation of a centered image computation block

RTL Analysis

RTL analysis transforms VHDL language into logic gates (Dossis, 2011) and (Lyalin, 2007). This step makes it possible to determine certain syntax errors such as bad connections between the modules, undeclared or non-coherent signals, etc. Errors and warnings appear in the "Message" tab. It is not necessary to do the RTL analysis before the synthesis. However, it is essential to correct these errors and warnings before proceeding to the next steps.

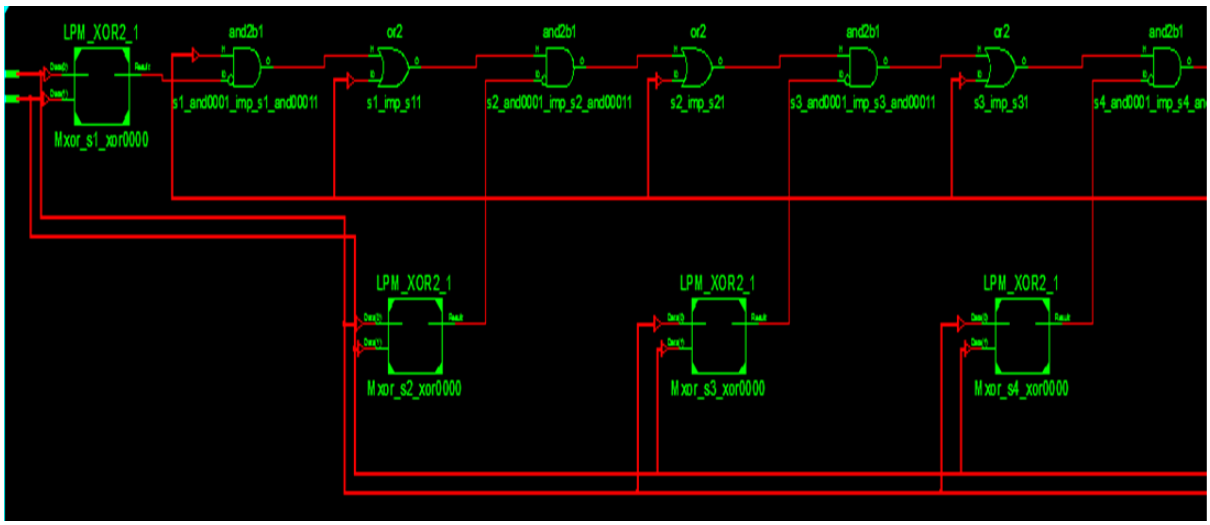


Figure 13. Schematic representation in RTL relating to the computation of centralized images

Algorithm for Computing Average Images

As mentioned above, the "RESET" button, set to the high state, allows to initialize the whole system. On each rising edge of the clock, if the "RESET" is set low and the activation signals "ENABLE" and "FLAG" high, then the pointer will point to the addresses of the eight ROMs simultaneously.

The ROM data is read and assigned to the inputs of the adder. The latter will compute the sum of the eight words coming from the eight ROMs. At the output of the adder, the sum is introduced at the input of the divider in order to compute the average by dividing by 8 the sum from the adder.

The result obtained is introduced into a subtractor that will compute the difference between the values (pixels) of each ROM (image) and the output of the divider. The result of this process represents the centered images. Processing at the level of all ROMs takes place in parallel.

After this description, we will present the results of the simulation of the global scheme of our algorithm under ISIM.

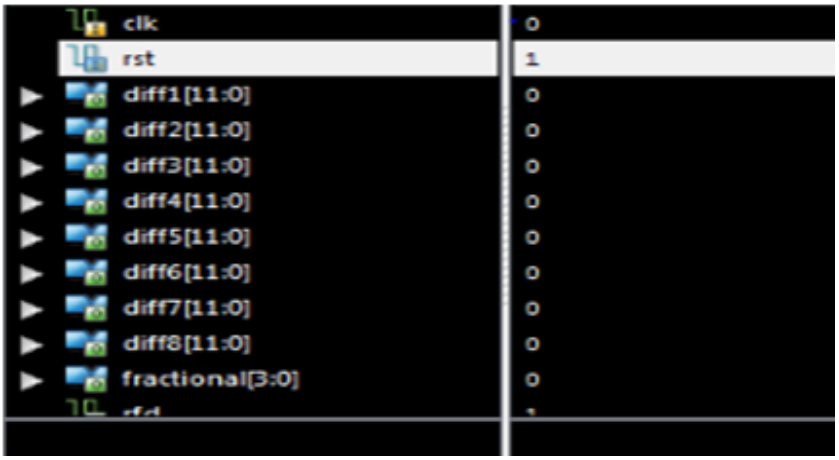


Figure 14. Illustration of the RESET (High State RESET)

Behavioral Simulation of the Global Scheme

The implementation diagram includes two inputs "ENABLE" and "FLAG" to start the computations (activation signals), a clock input "CLK" for synchronization, an asynchronous "RESET" input for reset and eight outputs for the difference. We previously performed the subtraction computations using MATLAB. The results obtained coincide.

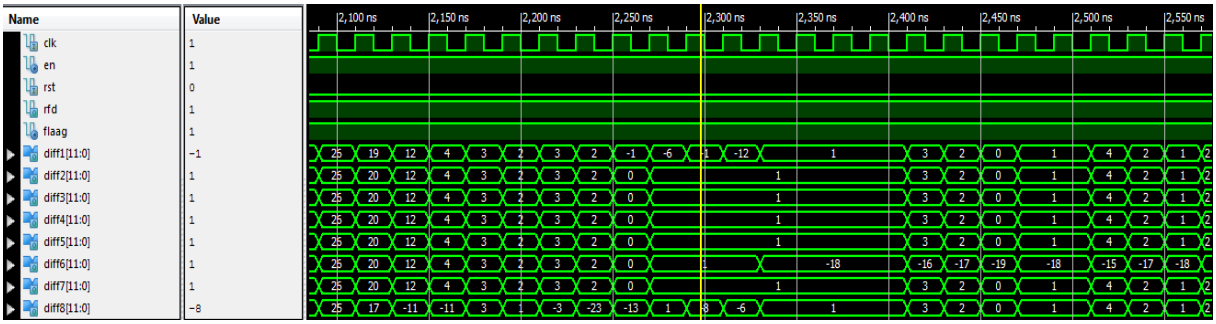


Figure 15. Behavioral simulation of the global scheme (Low State RESET)

Interpretation of Simulation Results

a) Real-time simulation

This part consists of loading the algorithm programmed using the ISE platform on the FPGA board and checking its functioning. After configuring and connecting the FPGA board, we proceed to load the algorithm. We observe the results using the chip scope.

In the implementation phase of the program, we encountered problems with mapping. The displayed error message is as follows:

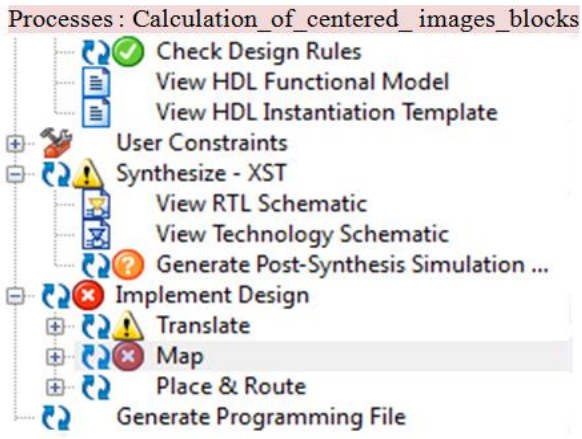


Figure 16. Mapping error

This is why we decided to test the functionality of our program with only two ROMs (Case of two images). We found that the FPGA card could not support the weight of our program from the storage point of view, which is why it was necessary to temporarily reduce the memory space as much as possible.

In what follows, we will apply the Principal Component Analysis on two images only in order to verify the possibility of implementing the Principal Component Analysis in real time, hoping to integrate external storage elements later (using the resources of the board) to store the entire database.

The diagram on ISE of the block (top module) for computing centered images is illustrated in Figure 17.

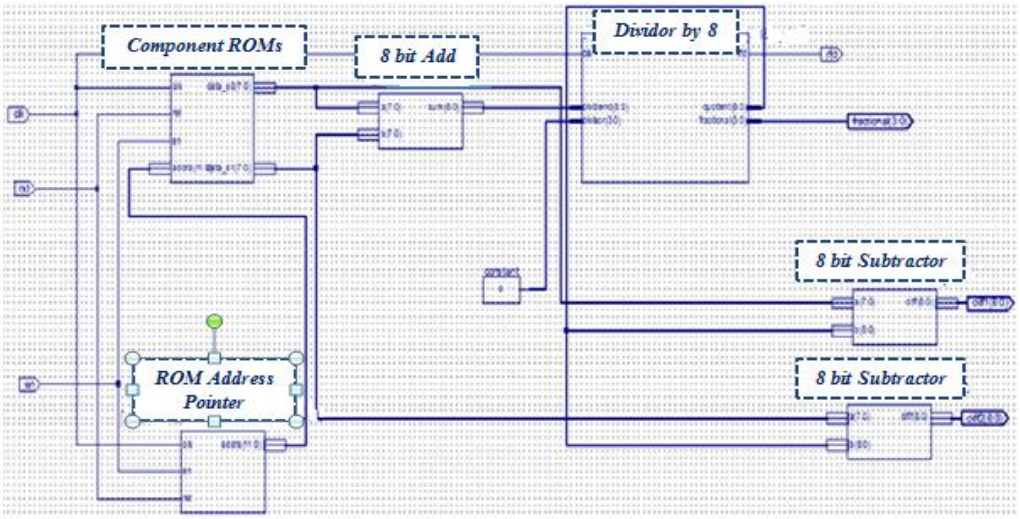


Figure 17. Schematization on ISE of the computation block for centered images

Figure 18 shows the simulation of the total block using two images:

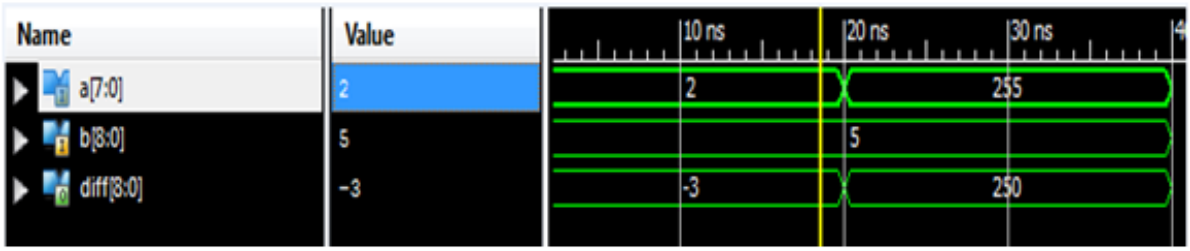


Figure 18. Simulation of the total block (case of two images)

Table 1 illustrates the results of the implementation:

Table 1. Implementation results

Average_Block_of_Two_Images Project Status (05/05/2019 - 09:30:50)			
Project File :	Average_Two_Images.xise	Parser Errors :	No error
Module Name :	Average_Block_of_Two_Images	Implementation State :	Programming File Generated
Target Device :	xc6slx 13-2csg324	- Errors :	No Errors
Product Version :	ISE 13.4	- Warnings :	4 Warnings (3 new, 0 filtered)
Design Goal :	Balanced	- Routing Result :	All signals Completely Routed
Design Strategy :	Xilinx Default (unlocked)	- Timing Constraiints :	All Constraints Met
Environment :	System Settings	- Final Timing Score :	0 (Timing Report)

Figure 19 represents the validation of all the implementation phases of our algorithm:

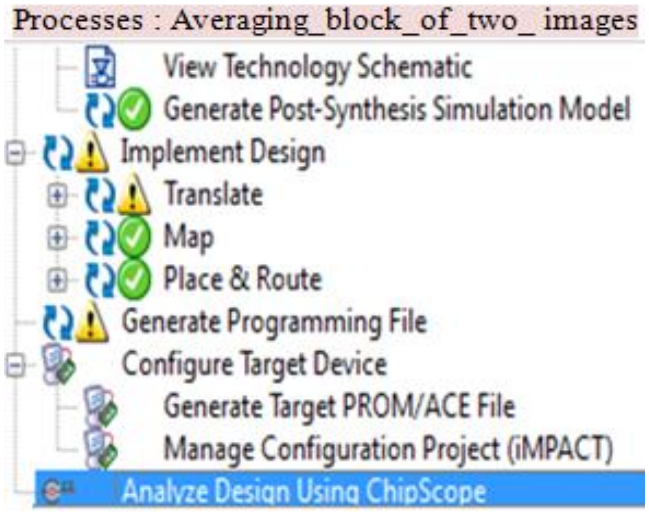


Figure 19. Validation of the algorithm implementation steps

The resources consumed by the algorithm are summarized in Table 2:

Table 2. Resources consumed by the algorithm

Type of Logic		Use	Available	Use in %
Number of slices		482	18224	2
Number of slices	Flip flops	65	65	100
	LUT	379	9112	4
Number of LUT used as	Shift registers	0	28800	0
	Memory	29	7200	1
Number of inputs / outputs used		56	440	12
Maximum Frequency (MHz)		1.995		

From Table 2, we note a reliability of overall consumption of resources of the PCA algorithm tuned with an unimportant maximum frequency. The results of the implementation on the FPGA card correspond perfectly to the results of the simulation on ISE.

b) Multiplication

Figures 20 and 21 represent the schematic description under ISE and the simulation under ISIM of the multiplication block, respectively:

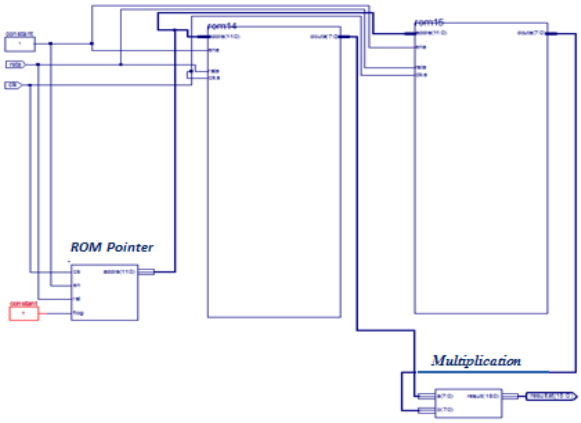
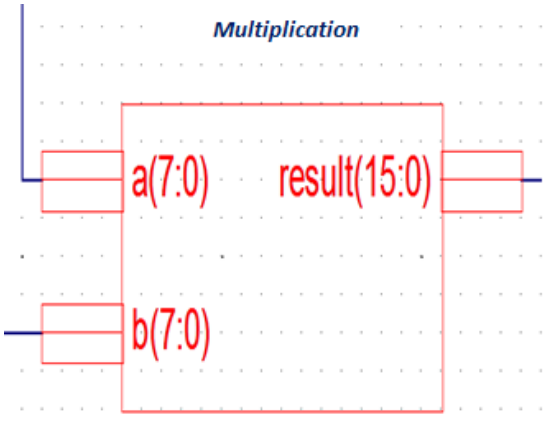


Figure 20. Schematic of the multiplication block



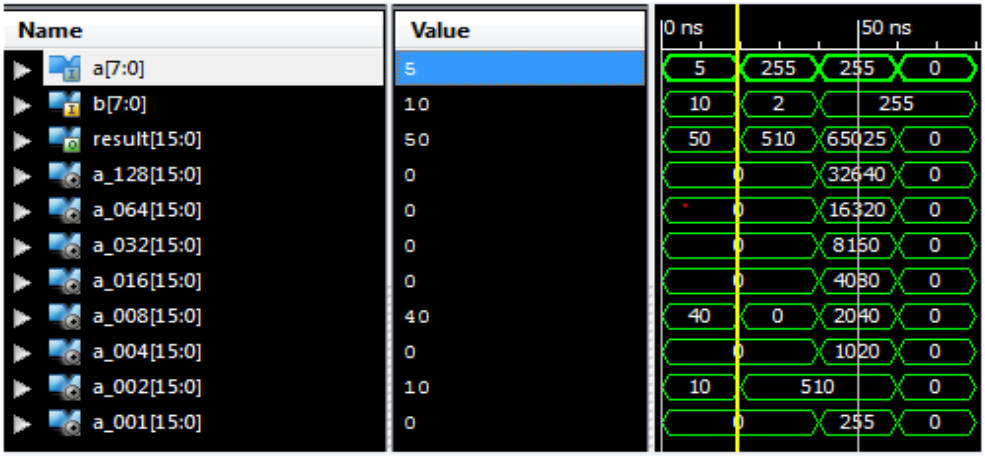


Figure 21. Simulation of the multiplication block under ISIM

The last step of the PCA technique represents the computation of the distance between the eigenfaces previously computed and the centered test image in order to identify the latter as a real client or impostor.

Principle for Computing the Manhattan Distance

For the computation of the distance between the eigenfaces of the database and that of the test image, several methods are proposed, among them, the Euclidean distance and the Manhattan distance. In our case, we choose to use the Manhattan distance, despite the fact that the Euclidean distance is much more natural but it is generally used to perform spatial calculation (Barnouti et al., 2016), (Das & Chatterji, 1990) and (Melter, 1991) and (Yuhui et al., 2015).

In addition, the Euclidean distance, given by the relation (2), contains the square root, so it will be more difficult to program. On the other hand, the Manhattan distance, given by the relation (1), often brings good results and it is easier to program. Which justifies our choice.

$$Manhattan\ distance: D_M = \sum_{i=1}^{n} |x_i - y_i| \quad (1)$$

$$Euclidean\ Distance: D_E = \sqrt[2]{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (2)$$

Figure 22 represents the schematic of the Manhattan distance calculator and figure 23 represents RTL architecture of the Manhattan distance computation block:

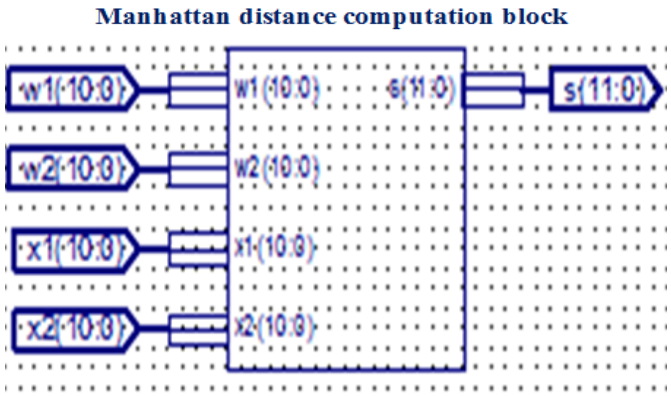


Figure 22. Schematic description of the Manhattan distance calculator

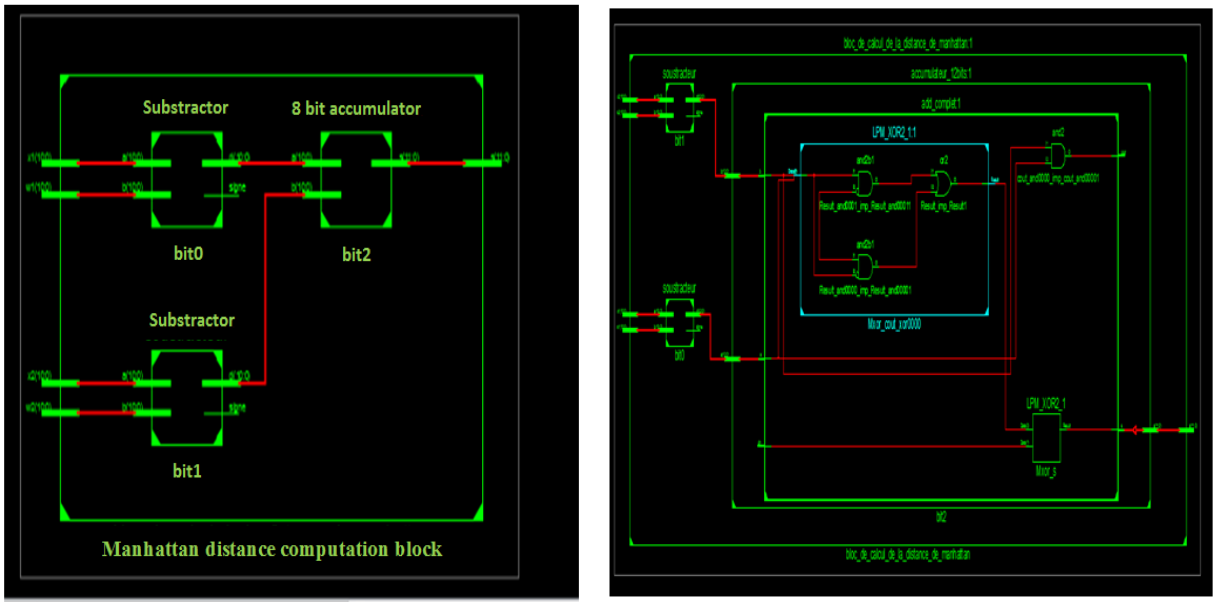


Figure 23. RTL architecture of the Manhattan distance computation block

Table 3 shows the RTL architecture of the Manhattan distance computation block:

Table 3. RTL architecture of the Manhattan distance computation block

Manhattan_distance_calculation_block Project Status (06/10/2019 - 09:30:50)			
Project File :	manhattan_dist_xise	Parser Errors :	No error
Module Name :	Manhattan distance calculation block	Implementation State :	Placed and Routed
Target Device :	xc5vlx 50-1ff676	- Errors :	No Errors
Product Version :	ISE 12.1	- Warnings :	2 Warnings (2 new)
Design Goal :	Balanced	- Routing Result :	All signals Completely Routed
Design Strategy :	Xilinx Default (unlocked)	- Timing Constraiints :	
Environment :	System Settings	- Final Timing Score :	0 (Timing Report)

Table 4 summarizes the results of the simulation, under ISE, of the Manhattan distance computation block:

Table 4. Simulation results for the Manhattan distance computation

Type of Logic		Use	Available	Use in %
Number of MUX used		236	4556	5
Number of slices	Unused flip flops	120	565	21
	LUT	379	9112	2
	Logiques	258	2176	2
Number of LUT used as	Shift registers	482	12224	2
	Memory	83	2176	3
Maximum Frequency (MHz)		1.137		

Conclusion

In this paper, we presented the results of our contribution to the FPGA implementation of the PCA algorithm for facial recognition. We started by describing the functioning of some modules that make up our program and we presented the results of the behavioral simulation relating to each. We were able to implement two essential parts of the PCA classification technique on FPGA, we can say that this method can be synthesized and implemented on circuit.

The simulations carried out using the ISE platform on the various modules, as well as that carried out on the overall diagram, showed that the technique worked well.

The results of the real-time simulation were conclusive, they agree perfectly with the simulation results, which proves that the program is functional on the chosen card, namely the SPARTAN 6-SP601 for the implementation of the centered images and the Virtex 4 for the implementation of the Manhattan distance.

References

Barnouti, N.H., Al-Dabbagh, S.S.M, Matti, W.E, & Naser, M.A.S. (2016). Face detection and recognition using Viola-Jones with PCA-LDA and square euclidean distance. International Journal of Advanced Computer Science and Applications, 7(5), (pp. 371-377)

Betz, V. & Rose, J. (1997). VPR: A new packing, placement and routing tool for FPGA. *In International Workshop on Field Programmable Logic and Applications. London,* (pp. 213-222).

Chen, S.C. & Chang, Y.W. (2017). FPGA placement and routing. *IEEE ACM International Conference on Computer-Aided Design (ICCAD).*

Clarke J.A., Gaffar, A.A, Constantinides, G.A., & Cheung, P.Y.K. (2006). Fast word-level power models for synthesis of FPGA-based arithmetic. *IEEE International Symposium on Circuits and Systems.*

Das P., Chatterji B. (1990). Orthogonal distances for digital pictures. *Information Sciences, 50,* (pp. 123-150).

Deschamps, J.P, Bioul, G. (2006). Synthesis of arithmetic circuits FPGA, ASIC, and embedded systems. *John WILEY & SONS*.

Dossis, M. (2011). Formal generation of synthesizable RTL from regular programs. *6th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS).*

Feng, Q., Yuan, C., Pan, J.S., Yang, J.F., Chou, Y.T., & Zhou, Y. (2017, February). Superimposed Sparse Parameter Classifiers for Face Recognition. *IEEE Transactions on Cybernetics. 47(2),* (pp. 378-390)

Jaafar, A. Soin, N. and Hatta, N. (2017). An educational FPGA design process flow using Xilinx ISE 13.3 project navigator for students. *IEEE 13th International Colloquium on Signal Processing & its Applications (CSPA)*

Lyalin, A. (2007). Formal methods of algorithm analysis for decreasing RTL verification complexity. *9th International Conference - The Experience of Designing and Applications of CAD Systems in Microelectronics.*

Melter R. (1991). A survey of digital metrics. *Vision Geometry Contemporary Mathematics*. *Contemporary Mathematics, 119,* (pp. 95-106).

Masato-Inagi, M., Takashima, Y., Nakamura, Y. (2010). Globally optimal time-multiplexing of inter-FPGA connections for multi-FPGA prototyping systems. *IPSJ Transactions on System LSI Design Methodology*.

Pistorius, J. & Hutton. M. (2003). Placement rent exponent calculation methods, temporal behaviour and fpga architecture evaluation. *SLIP'03: Procedings of the 2003 International Worhshop on System Level Interconnect Prediction,* (pp. 31-38).

Rodrequez-Andina, J.J., Moure, M.J., & Valdes, M.D. (2007). Features, design tools, and application domains of FPGA. *IEEE Transactions on Industrial Electronics, 54(4).*

Salcic, Z. (2001). Synthesizing Logic From VHDL Description. *VHDL and FPLDs in Digital Systems Design, Prototyping and Customization.*

Siguenza-Tortosa, D. and Nurmi, J. (2002). VHDL Based simulation environment for proteo NoC. *Seventh IEEE International. High-Level Design Validation and Test Workshop, 1(6),* (pp. 27-29).

Skliarova, I. & Ferrari, A.B. (2000). Exploiting FPGA-based architectures and design tools for problems of reconfigurable computations. *Proceedings 13th Symposium on Integrated Circuits and Systems Design. Computer Science.*

Taraate, V. (2017). Design and simulation using VHDL constructs. *PLD Based Design with VHDL – RTL Design, Synthesis and Implementation. Springer Edition.*

Tang, Q. Mehrez, H. and Tuna, M. (2013). Routing algorithm for Multi-FPGA based systems using multi-point physical tracks. *International Symposium on Rapid System Prototyping (RSP).*

Yuhui, Z., Byeungwoo, J., Danhua, X., Jonathan, W.Q.M., & Hui, Z. (2015). Image segmentation by generalized hierarchical fuzzy C-means algorithm. *Journal of Intelligent & Fuzzy Systems, 28(2),* (pp. 961-973).

Pentland, A., Moghaddam, B. and Starner, T. (1994, Jun). View-based and modular eigenspaces for face recognition. *Computer Vision and Pattern Recongnition. Proceedings CVPR'94, IEEE Computer Society Conference,* (pp. 84-91).

Jian, Y., Zhang, D., Frangi, A. and Y. Yang, J. (2004, January). Two-Dimensional PCA: A new approach to appearance-based face representation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(1),* (pp. 131–137).

Monmasson, E., & Cirstea, M. N. (2007, August). FPGA design methodology for industriel control systems. *A review IEEE Transactions on Industrial Electronics, 54(4).*

Morizet, N. (2009, March). Biometric Recognition by Multimodal Fusion of Face and Iris. *PhD thesis, specialty: Signal and Images. National School of Telecommunications. Paris, France.*

Alfke, P. (2009, August). Xilinx Virtex-6 and SPARTAN-6 FPGA families. *IEEE Hot Chips 21 Symposium (HCS).*

Li Jun, L. & Wei, W. (2010, November). PCI express interface design and verification based on SPARTAN-6 FPGA. *IEEE 12th International Conference on Communication Technology.*

Bezerra, E. & Lettnin, D. (2013, October). Writing Synthesizable VHDL Code for FPGA. *Synthesizable VHDL Design for FPGA.*

Komulainen, J., Hadid, A., & Pietikäinen, M. (2014, January). Context based face anti-spoofing. *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS), Arlington, VA, USA.*

Boulkenafet, Z., Komulainen, J., & Hadid, A. (2016, August). Face Spoofing Detection Using Colour Texture Analysis. *IEEE Transactions on Information Forensics and Security, IEEE Biometrics Compendium, 11(8),* (pp. 1818-1830)

Alaslani, M.G., & Elrefaei, L.A. (2018, April). Convolutional neural network based feature extraction for iris recognition. *International Journal of Computer Science and Information Technology (IJCSIT), 10(2),* (pp. 65-78).

Challouf, M. and Hicham, M. (2007). Tutorial Xilinx ISE 9.1. *INSAT Tunis.*

Bibliographic information of this paper for citing:

Allam, Z. F., Hamami-Mitiche, L., & Bousbia-Salah, H. (2020). Implementation of Face Recognition Algorithm on Fields Programmable Gate Array Card. *Journal of Information Technology Management*, 12(2), 40-58.