HAL
open science

A new FPGA accelerator based on circular buffer unit per orientation for a fast and optimised GLCM and Texture feature computation

Mohamed Amin Ben Atitallah, Rostom Kachouri, Hassene Mnif

To cite this version:

Mohamed Amin Ben Atitallah, Rostom Kachouri, Hassene Mnif. A new FPGA accelerator based on circular buffer unit per orientation for a fast and optimised GLCM and Texture feature computation. IEEE international conference on Design & Test of integrated micro & nano-Systems (DTS), Apr 2019, Gammarth, Tunisia. hal-02172154

HAL Id: hal-02172154

https://hal.science/hal-02172154v1

Submitted on 3 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A new FPGA accelerator based on circular buffer unit per orientation for a fast and optimised GLCM and Texture feature computation

Mohamed Amin Ben Atitallah
LETI (E.N.I.S.), University of Sfax, TUNISIA
National Engineering School of Gabes (ENIG)
University of Gabes, TUNISIA
mohamed.amine@esiee.fr

Rostom Kachouri
Gaspard Monge Computer Science Laboratory
ESIEE-Paris
University Paris-Est Marne-la-Vallée, FRANCE
rostom.kachouri@esiee.fr

Hassene Mnif
LETI (E.N.I.S.), University of Sfax, TUNISIA
ENET'com Sfax, TNISIA
hassene.mnif@enetcom.usf.tn

***Abstract*— This paper presents an FPGA accelerator based on circular buffer unit per orientation for a fast and optimized Gray Level Co-occurrence Matrix (GLCM) and four Texture features computation. The Four texture features namely, contrast, energy, dissimilarity and correlation are computed using Xilinx FPGA. However, the computation of GLCM and four textures features are very complex and consume a lot of execution time. In this paper, an FPGA accelerator for fast computation of GLCM and four texture features are designed and implemented. This architecture was implemented on a Xilinx Zc-702 using Vivado HLS. The obtained results are then compared against other related works. The synthesis results on FPGA prove a significant gain (about 17%) in execution time compared to the previous work.**

Keywords—Image analysis applications; Parallel calculation; Haralick's texture feature; Hardware/Software Implementation; Circular buffer unit; FPGA; Vivado_HLS; Optimization; Execution time

I. INTRODUCTION

The texture features are used for image classification. These texture features capture information about the patterns that emerge in patterns of texture. The texture features are computed by construction a GLCM matrix that is computationally expensive. Once the GLCM matrix has been constructed, computations of the 14 texture features begin. Some of these Haralick texture features include contrast, energy, dissimilarity and correlation, as well as a variety of entropy measures. Due to the numerical nature of the calculation, this problem is the focus of our optimization.

The GLCM is an effective method of Haralick texture feature extraction, which is focus in texture analysis methods [1, 2], image retrieval [3], image classification [4], image segmentation [5], and image recognition [6].

However, the calculation of the GLCM and the features consume a lot of time. Therefore, many methods to speed up their calculations are highly desired. In order to improve the performance of the Gray Level Co-occurrence Matrix and the features algorithm, we propose a Hardware/Software (HW/SW) implementation on Xilinx FPGA. In the proposed design, the GLCM is calculated in parallel and four texture features are although calculated in parallel. We have implemented the proposed design on Zc_702 FPGA. The synthesis results on FPGA prove a significant gain (about 17%) in execution time compared to the previous work.

In this paper we will present, the GLCM matrices and **Haralick Texture Features** in Sect 2, some important related works in Sect 3, our HW/SW implementation in Sect 4, the experimental results in Sect 5 and finally, the conclusions in Sect 6.

II. GLCM AND FOUR HARALICK TEXTURE FEATURES

In image processing domain, a texture is a field of the image that appears as a coherent and homogeneous domain, that mean forming everything for an observer. Texture analysis is a very important area in treatment of images, among the main elements of interpretation of the visual message. As well as the filtering is to improve the visual quality of the image or to extract attributes of the image, by modifying the value of gray level of a pixel according to the value of its neighbors. Among the different approaches from the texture analysis we mention the co-occurrences matrix, the texture spectrum, the wavelets and the mathematical morphology. In this paper, we are interested in the co-occurrence matrices. The co-occurrence matrices are analogous to two-dimensional histograms. They represent the number of occurrences of particular pairs of pixel in the image. The elements of the GLCM represent the probabilities of occurrences of the pair of gray levels (i,j) separated by a distance d and oriented by an angle θ. The direction measures are selected as (1), (2), (3), and (4), which corresponds to (θ=0◦, 45◦, 90◦, 135◦) in Figure 1, respectively.

The co-occurrence matrices contain the first-order space averages. Several clues have been proposed by Haralick that correspond to descriptive 14 texture features which can be calculated from these matrices. Among the 14 texture features, we are interested just for four types of statistics (Table 1). Eenergy or Angular second moment (ASM) expresses the regular character of the texture. In general, a high energy is observed when the image is very regular, that is to say when the high values of the GLCM are concentrated in some places of the matrix. Contrast (CON) more the texture is contrasted, more the term is greater. Correlation (COR) can be likened to a measure of the linear dependence of gray levels in the image. Dissimilarity (DISS) is used to measure the dissimilarity between two gray levels *i* and *j*. These four selected texture features are better for texture feature extraction [9].

In fact, the arithmetic operations of texture features in HW have almost the same order of complexity in implementation.

978-1-7281-1129-2/19/$31.00 ©2019 IEEE

$$P(i,j,\mathrm{d},0) = \left|\left\{\left((k,l),(m,n)\right) \in (N*M)^2 \; while \; \left(k-m=0, |l-n| = \mathrm{d}, \mathrm{I}_{\mathrm{k,l}} = i, \mathrm{I}_{\mathrm{m,n}} = j\right)\right\}\right| \quad (1)$$

$$P(i,j,\mathrm{d},45) = \left|\left\{\begin{matrix} \left((k,l),(m,n)\right) \in (N*M)^2 while \\ (k-m=\mathrm{d}, \mathrm{l}-\mathrm{n}=-\mathrm{d}) \vee (k-m=-\mathrm{d}, \mathrm{l}-\mathrm{n}=\mathrm{d}), \mathrm{I}_{\mathrm{k,l}} = i, \mathrm{I}_{\mathrm{m,n}} = j \end{matrix}\right\}\right| \quad (2)$$

$$P(i,j,\mathrm{d},90) = \left|\left\{\left((k,l),(m,n)\right) \in (N*M)^2 \; while\left(|k-m| = \mathrm{d}, l-n=0, \mathrm{I}_{\mathrm{k,l}} = i, \mathrm{I}_{\mathrm{m,n}} = j\right)\right\}\right| \quad (3)$$

$$P(i,j,\mathrm{d},135) = \left|\left\{\begin{matrix} \left((k,l),(m,n)\right) \in (N*M)^2 \; while \\ (k-m=\mathrm{d}, \mathrm{l}-\mathrm{n}=\mathrm{d}) \vee (k-m=-\mathrm{d}, \mathrm{l}-\mathrm{n}=-\mathrm{d}), \mathrm{I}_{\mathrm{k,l}} = i, \mathrm{I}_{\mathrm{m,n}} = j \end{matrix}\right\}\right| \quad (4)$$

With:
• (k, l), are the coordinates of a graylevel pixel i ∈ [0, nmax-1].
• (m, n), those of the graylevel pixel j ∈ [0, nmax-1].

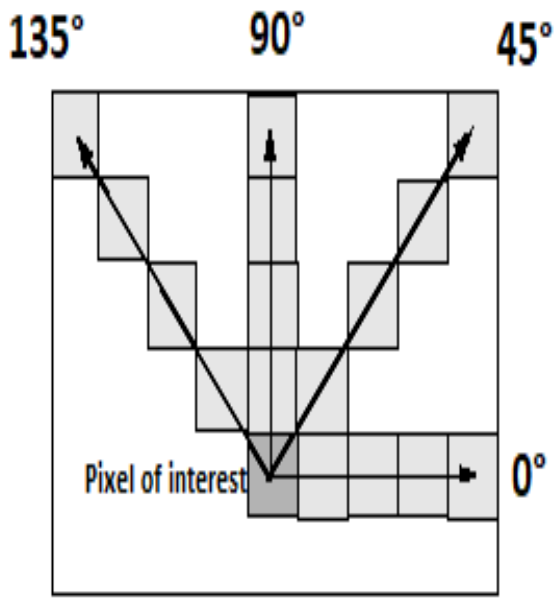

Fig. 1. Direction meseare

TABLE I. TEXTURE FEATURES

Features	Formula
Energy or ASM	$f_1 = \sum_{i=1}^{Ng}\sum_{j=1}^{Ng} p(i,j)^2$
Contrast	$f_2 = \sum_{n=0}^{Ng-1} n^2 \sum_{i=1}^{Ng}\sum_{j=1,\lvert i-j\rvert=n}^{Ng} p(i,j)$
Correlation	$f_3 = \frac{\sum_i\sum_j (i,j)p(i,j) - u_x u_y}{\sigma_x \sigma_y}$
Dissimilarity	$f_4 = \sum_{i=1}^{N}\sum_{j=1}^{N} \lvert i-j\rvert * p(i,j)$

III. RELATED WORKS

Many works have been focused on speeding up the process of the GLCM and the texture features calculation algorithms on FPGAs.

Girisha and al [10] proposed a HW implementation to calculate the GLCM matrices for single direction (θ=0°) and a distance (d=1). The proposed architecture is implemented on a Xilinx FPGA. In fact, the size of the input image is 8x8 pixels, each pixel is represented on (n=4) bits and (Ng=8). In fact, the input image of this architecture is stored in two memory blocks to determine the number of occurrences of particular pairs of pixel in the image.

This proposed architecture can compute GLCM matrices for just a single angle.

Ali Reza and al [11] proposed a Hardware implementation to compute the GLCM for four angles (θ= 0 °, 45 °, 90 °, 135 °) in parallel for an image size 128X128 pixels. Each pixel of the image is represented on (n = 8). The HW implementation of GLCM was done on the a Xilinx Virtex 5 platform. In fact, the size of the matrix GLCM is NgxNg (256x256) since each pixel is presented on (n = 8) bits and with distance (d = 1). This HW implementation can compute the GLCM matrices in parallel and stored them in memory blocks RAM to determine the calculations of the texture features. The computation of the texture features are done on integer format of 16 bits.

The output results of this proposed architecture are done in an integer format, which generates inaccurate results. In addition, the size of the input image is fixed.

Amin and al [12] proposed a HW / SW implementation to calculate four GLCM matrices (0 °, 45 °, 90 ° and 135 °) in parallel. The HW implementation of GLCM was done on the Zedboard platform based on the Zynq circuit. In fact, the size of the matrix GLCM is NgxNg (256x256) since each pixel is presented on (n = 8) bits with (d = 1). The coefficients of the matrix are presented on (m = 16) bits.

This architecture has as disadvantage that the size of image is limited.

Dimitris and al [13] presented a HW implementation to compute 16 GLCMs matrix and four feature vectors using a single core. They chose Ng = 64, the number of gray level for different image sizes from (512 X 512) to (2048 X 2048). They used fixed-point operations instead of floating-point operations to compute four texture features namely, Angular Second Moment (ASM), correlation (Cor), Inverse Difference Moment (IDM) and entropy. Their architecture has 16 computation units of GLCM matrices. The size of the GLCM matrices is NgxNg (64X64) since each pixel is presented on (n = 6) bits and with distance (d = 1). This HW implementation can compute the GLCM matrices of 16 image blocks in parallel

and stored them in two RAM blocks to determine the calculations of the texture features. This architecture takes into account the overlap between the windows but it doesn't take into account the overlap between the different image blocks, which generates not accurate results.

The Dimitris hardware implementation method is the best solution compared to the existing since the input image size is unlimited compared to the other architectures.

The goal of our research project is to solve the problems of the proposed architecture of Dimitris.

IV. Hardware Architecture of GLCM and Texture Features

FPGAs, or reconfigurable chips, have not stopped evolving since their creation and now they are used in complete systems (Xilinx Zynq or Altera Stratix). Nevertheless, there are still many fields application fields which they are absent, and wrongly [14]. For that, many programmers used the High Level Synthesis (HLS) tools to increase the productivity of FPGA. The HLS tools make it possible to pass from the high-level behavioural description of a system, written in C / C ++ code, System C ..., to a synthesizable RTL code consisting of a data path and a control logic [15].

A. Architectural design for GLCM and texture features

The architecture of the implemented hardware is shown in Figure 3. Our architecture was developed by a c code compatible with the HLS tool. The SW part iteratively feeds the Xilinx platform with four vectors. Each pixel of the vectors is presented by (n=8) bits so (N_g=256). Our proposed architecture reads each vector's pixels, computes the GLCM of each vector and their respective texture features for the four directions and for a distance d=1, and store them into bank memory.

Once the matrices are computed, we can precede the computation of the four texture features in parallel. The output of our system includes two outputs, one presents the texture features results and the other to detect the end of sending data. The SW part is used for setting and sending data via the DMA. Also, perform the division operation to maintain the accuracy of the floating values of contrast, correlation, dissimilarity and ASM. The arrangement of the four vectors for the four angles (θ =0°, 45°, 90° and 135°) is done in the software part. In fact, in our architecture we use a Direct Access Memory in order to accelerate the Data transfer.

B. Circular buffer unit

In order to avoid the saturation of memory, four circular buffers units have been used to reduce the external bandwidth requirements of our architecture. The Figure 2 shows the pixels of the input block. The squares in the grid represent the pixels in the block. The colored pixels are stored in the circular buffers. Two outputs are presented at the end of each circular buffer, namely the central pixel (black background) and its neighboring pixel for the four directions (gray background). As shown in Figure 2, in every clock cycle the last pixel is removed from each circular buffer, the neighborhood is shifted by 1 pixel to the right and a new pixel is inserted into the buffer.

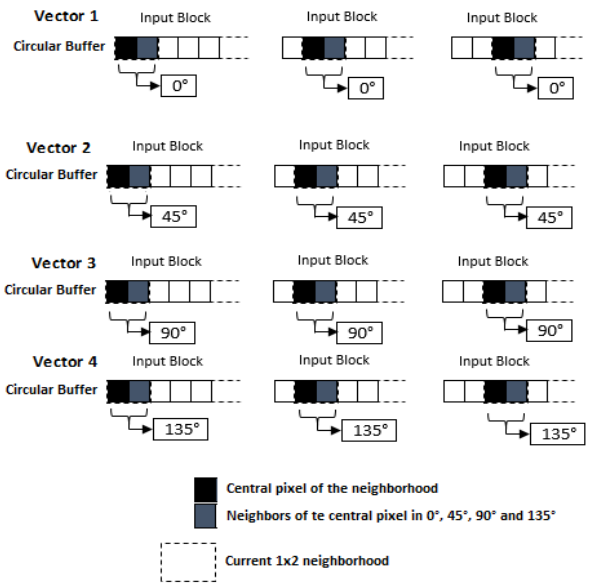

Fig. 2. Computation of GLCM matrix using four circular buffers units

C. RTL synthesis and optimization

We want to implement the GLCM matrices and four texture features with a configuration based on Xilinx FPGA of the Zynq XC7z020 family with a maximum frequency of 100 MHz. After analyzing the results, we can create a new solution for the project with different constraints and optimization guidelines and synthesize the new solution. We can also repeat this process until we achieve the desired performance. It should be noted that the use of multiple solutions allows further development while maintaining the previous results. With Vivadeo HLS, we can apply various guidelines to optimize HW / SW design. In fact, we have developed different solutions steps for image size 176X144, which are illustrated in Table 2.

TABLE II. Synthesis Results with Vivado_HLS

Solutions	FPGA	DSP %	FF %	LUT %	BRAM %	Latency number
#Solution 1	Zc-102	38	2	5	91	2557658
#Solution 2	Zc-102	38	3	7	91	1019949
#Solution 3	Zc-102	38	3	7	91	1170725
#Solution 4	Zc-102	38	3	9	91	**232504**

In order to improve the architectural performances, we have exploited the optimization levels favored by Vivado HLS. Firstable, in the "Solution 2", we applied the RESOURCE directive at the level of GLCM matrices to implement them as BRAMs. After that, the ALLOCATION directive is applied to the texture features operations, thus making it possible to share the resources used by the hardware block. In the "Solution 3", we applied the UNROLL directive in the level of texture features. Finally, in the "Solution 4", we sought to improve the maximum throughput completed by hardware IP. For this reason, the PIPELINE optimization option is applied at the level of GLCM and texture features. It should be noted that the

synthesis of the "Solution 1" is carried out without any optimization criteria. From the results shown in Table 2, we note that the Solution 4 has the minimum of latency number (or cycles number) over the other solutions. Indeed, the solution 4 allows a significant contribution about 56% in cycles number compared to the solution 1. The next analyzes will be developed based on the "Solution 4" since it satisfies a compromise of the cycles number.

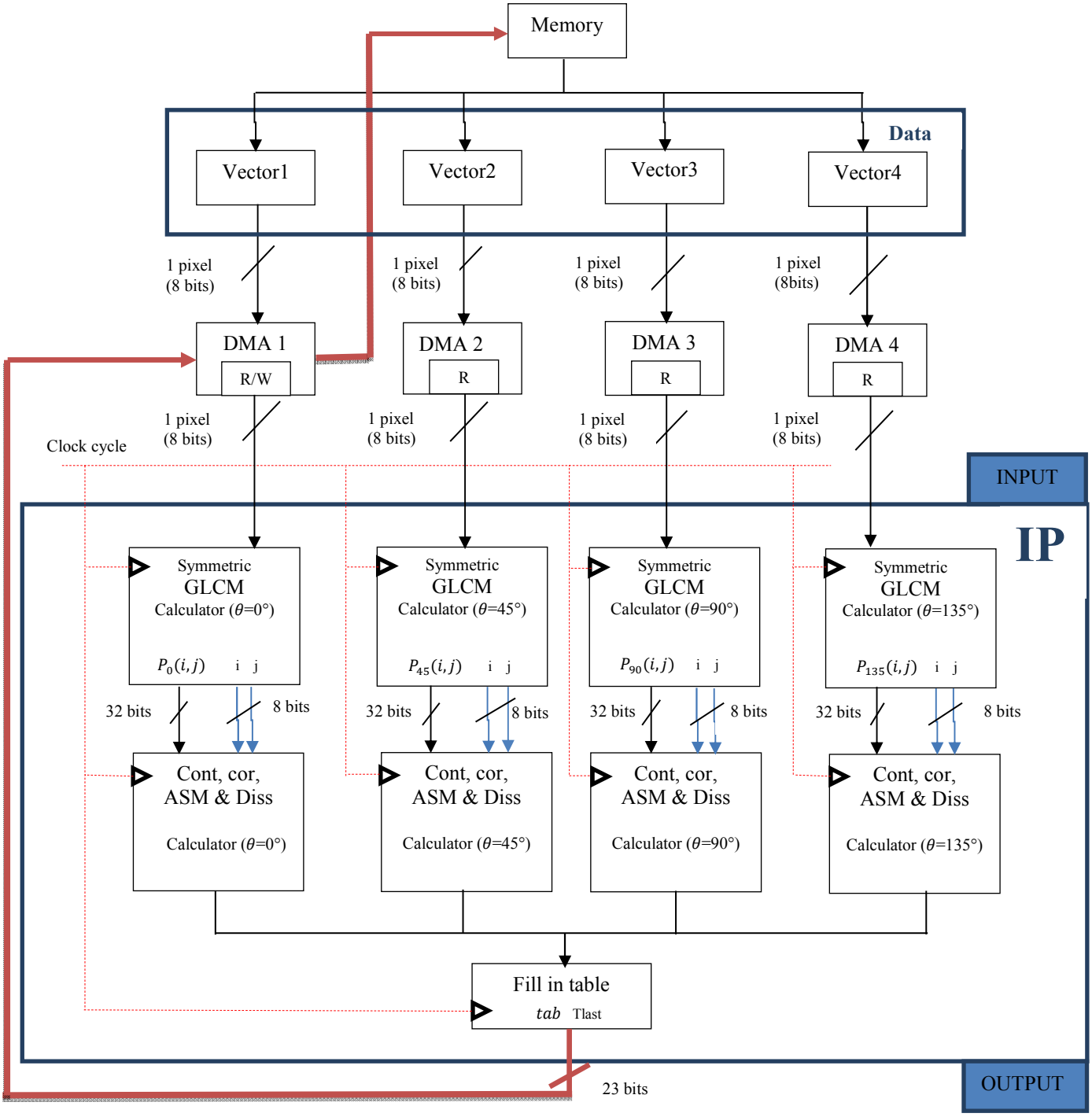

Fig. 3. Proposed HW/SW architecture of GLCM and texture feature computation

V. EXPERIMENTAL RESULTS

In this section, our goal is to valid the performance of our proposed design in Hardware/Software codesign.

The implementation of our architecture has been done on Zc-702 FPGA device using Vivado HLS. The Zynq FPGA family contains in addition to the programmable logic (PL) a dual core ARM microprocessor system with its memory controllers and external peripherals as shown in Figure 4. It is a hardcore processor that can operate with a maximum frequency of 700 MHz.

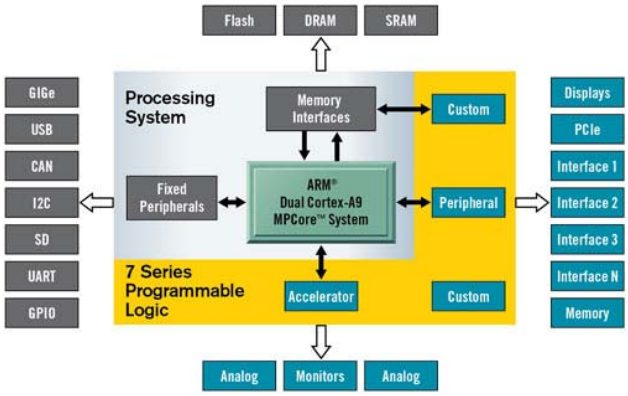

Fig. 4. Architecture of Zc-702 FPGA

The HLS tool in its internal operation combines three main steps (Figure 5). The first is the compilation. It is used to transform the input code of the HLS tool into an Intermediate Representation (IR) adapted to the needs of the tool. In the second step, the circuit in its intermediate representation undergoes transformations. Finally, we generate the bitstream to configure the HW part using Vivado IDE.

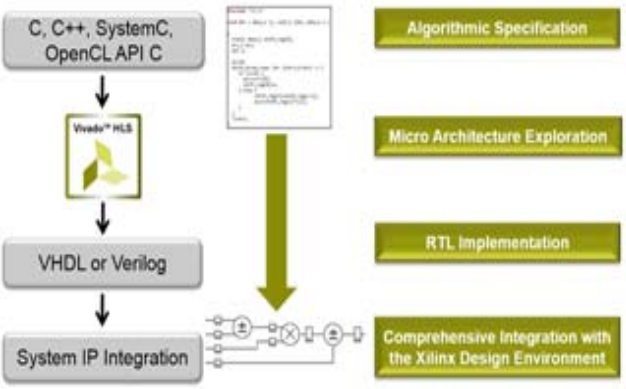

Fig. 5. System design flow

In the context of SoC development, Xilinx provides the ZC 702 FPGA card. As shown in the Figure 6, it consists of a PS (Processing System) part based on an ARM Dual Cortex A9 processor that communicate with a PL (Programmable Logic) part through internal communication bus. In order to improve the architectural performance of the GLCM design, four vectors are grouped together in the same hardware as the parallel Design IP by exploiting four DMA ports in parallel. The first DMA is configured in read /write mode at the same time while the others DMA are used in only write mode.

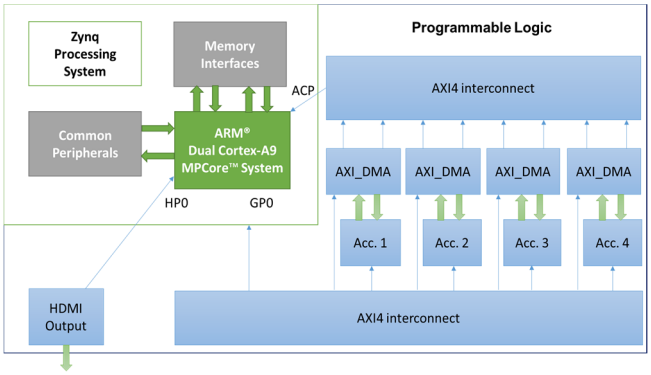

Fig. 6. Heterogeneous SoC design

In the table 2, a comparative study was realized between our proposed architecture and previous work.

In fact, according to the Figure 7, we notice a minimization about 17% of the execution time of our HW/SW solution compared to the Dimitris[13] solution.

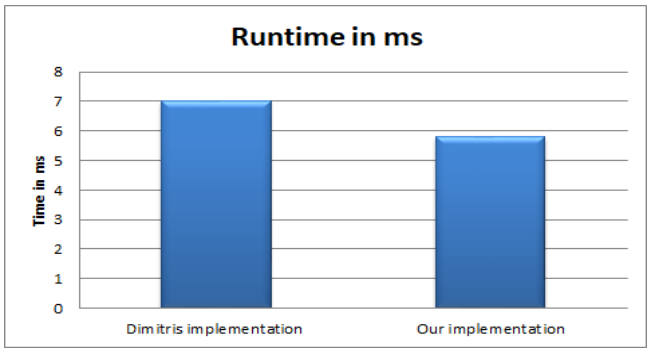

Fig. 7. Comparison of execution time of our proposed HW / SW solution and the other work

As shown in the Figure 8, we notice a minimization about 94% of the execution time of the HW / SW solution compared to the SW solution. As a result, we can conclude that the HW / SW implementation is more efficient than the SW part.

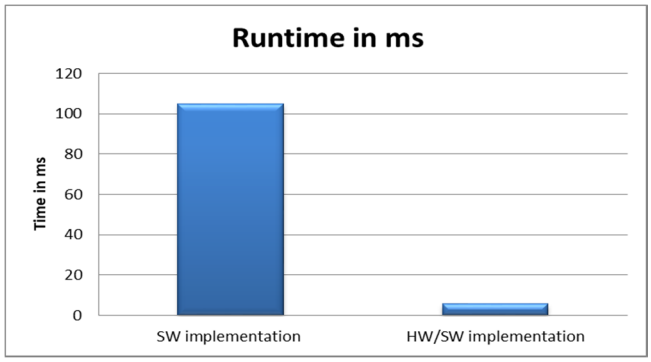

Fig. 8. Comparison of the execution time of the SW and our proposed HW / SW solution

From these analyzes, it is obvious that the HW / SW solution allows a compromise between the SW flexibility and the HW performance.

TABLE III. VALIDATION RESULTS OF PROPOSED METHODE

Proposed architectures	FPGA	Pixels number	GLCM size	Image size	HW/SW calculations	Freq (Mhz)	Execution time
Dimitris et al [13]	XCV2000E -6	6	64x64	176X144	GLCM+Cor+ ASM+IDM+ Entropy	100	7 ms
Our architecture	Zc-702	6	64x64	176X144	GLCM+Cont+ ASM+Cor+ DISS	100	**3.9 ms**
Our architecture	Zc-702	8	256x256	176X144	GLCM+Cont+ ASM+Cor+ DISS	100	**5.8 ms**

VI. CONCLUSION

In conclusion, in this paper we proposed an FPGA accelerator based on circular buffer unit per orientation for a fast and optimized GLCM (θ= 0 °, 45 °, 90 °, 135 °) and four Texture features computation. The synthesis results on FPGA prove a significant gain (about 17%) in execution time compared to previous work. To highlight the performance of our proposed architectures, we adopted a joint HW/SW design to estimate the execution time. These analyzes proved a considerable gain (about 94%) of the HW/SW solution compared to the SW in execution time.

Finally, our next work are considering to extract the texture features from a video frames.

ACKNOWLEDGMENT

We thank our colleagues from ESIEE Paris, who provided insight and expertise that greatly assisted the research and0improved the manuscript.

REFERENCES

[1] Theodoridis, S., & Koutroumbas, K. (2000) "Pattern recognition," San Diego: Academic.

[2] Haralick, R.M.; Shanmugam, K.; Dinstein, I. "Textural Features for Image Classification," IEEE Trans. Syst. Man Cybern. 1973, SCM-3, 610–621.

[3] De Almeida, C.W.D.; De Souza, R.M.C.R.; Candeias, A.L.B. "Texture Classification Based on Co-Occurrence Matrix and Self-Organizing Map," In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Istanbul, Turkey, 10–13 October 2010; pp. 2487–2491.

[4] Kekre, H.; Thepade, S.D.; Sarode, T.K.; Suryawanshi, V. "Image retrieval using texture features extracted from GLCM," LBG and KPE. Int. J. Comput. Theory Eng. 2010, 2, 695–700.

[5] Srinivasan, G.N.; Shobha, G. "Segmrntation Techniques for ATDR," NAUN Int. J. Comput. 2008, 2, 165–171.

[6] Tuceryan, M. "Moment-based texture segmentation," Pattern Recognit. Lett. 1994, 15, 659–668.

[7] Sahoo, M. "Biomedical Image Fusion and Segmentation using GLCM," In Proceedings of the 2nd National Conference-Computing, Communication and Sensor Network (CCSN), Orissa, India, 29–30 October 2011; pp. 34–39.

[8] Zhang, Y. "Optimisation of building detection in satellite images by combining multispectral classification and texture filtering," ISPRS J. Photogramm. Remote Sens. **1999**, 54, 50–60.

[9] Zheng, S.; Zheng, J.; Shi, M.; Guo, B.; Sen, B.; Sun, Z.; Jia, X.; Li, X. "Classification of cultivated Chinese medicinal plants based on fractal theory and gray level co-occurrence matrix textures," J. Remote Sens. 2014, 18, 868–886.

[10] Girisha A B, M C Chandrashekhar, Dr.. M Z kurian, "FPGA implementation of GLCM," International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering Vol. 2, Issue 6, June 2013.

[11] Ali Reza, Asadollah, Babak , "High performance implementation of texture features extraction algorithms using FPGA architecture," J Real-Time Image Proc (2014) 9:141–157.

[12] M.A. Ben Atitallah, R. Kachouri , M.Kammoun, H.Mnif "An efficient implementation of GLCM algorithm in FPGA," IINTEC (2018).

[13] Dimitris Maroulis & Dimitris K. Iakovidis & Dimitris Bariamis "FPGA-based System for Real-Time Video Texture Analysis," 2008 Springer Science + Business Media, LLC. Manufactured in The United States.

[14] I. Gonzalez, E. El-Araby, P. Saha, T. El-Ghazawi, H. Simmler, S. G. Merchant, B. M. Holland, C. Reardon, A. D. George, H. Lam, G. Stitt, N. Alam, and M. C. Smith, "Classi_cation of Application Development for FPGA-Based Systems,"□ Aerospace and Electronics Conference, 2008. NAECON 2008.

[15] W. Meeus, K. Van Beeck, T. Goedeme, J. Meel, and D. Stroobandt, "An Overview of Today's High-Level Synthesis Tools," Design Automation for Embedded Systems (2012).