Novel HW Architecture Based on FPGAs Oriented to Solve the Eigen Problem

Ignacio Bravo, Manuel Mazo, José Luis Lázaro, Pedro Jiménez, Alfredo Gardel, and Marta Marrón

***Abstract*—A hardware solution is presented to obtain the eigenvalues and eigenvectors of a real and symmetrical matrix using field-programmable gate arrays (FPGAs). Currently, this system is used to compute the eigenvalues and eigenvectors in covariance matrices for applications in digital image processing that make use of the principal component analysis (PCA) technique. The proposed solution in this paper is based on the Jacobi method, but in comparison with other related works, it presents a different architecture that remarkably improves execution time, while reducing the number of consumed resources of the FPGA.**

***Index Terms*—CORDIC, eigenvalue, eigenvector, field-programmable gate array (FPGA).**

I. Introduction

The calculation of eigenvalues and eigenvectors is a problem addressed in numerous practical applications [9], [12], between that are those that uses techniques of principal component analysis (PCA) [6], [8].

One of the problems related to the application of PCA techniques is the necessary calculation of the transformation matrix, formed by the most significant eigenvectors of the covariance matrix. The difficulty in calculating this transformation matrix resides in the large size of the covariance matrix managed by many applications, for example, computer vision algorithms [4], [5], frequently requiring the online update of this transformation matrix. Some solutions for the calculation of eigenvalues and eigenvectors have been proposed [1], [2], [7].

In most of them, the calculation of eigenvalues and eigenvectors is done using digital processor-based platforms. Due to the features of its internal architecture, the calculation process is highly sequential, requiring a large amount of time to compute it. To improve the speed of the system, in some proposals, for example in [7], the trigonometric values necessary to make the calculation of eigenvalues and eigenvectors are stored in memory tables. In other proposals, as in [2], the objective is to accelerate execution by using multiprocessor platforms. Therefore, if the goal is the implementation of the calculation of eigenvalues and eigenvectors in a field-programmable gate array (FPGA), different solutions must be designed and executed concurrently within these devices while achieving high accuracy.

This paper has been divided into several parts. Section II describes the computation process for the matrix eigenvalues, analyzing the different alternatives that exist. Section III presents the new proposal for obtaining the eigenvalues and eigenvectors using a reconfigurable hardware. Finally, Sections IV and V discuss the empirical results and the conclusions obtained from this work, respectively.

Manuscript received May 25, 2007; revised September 15, 2007 and December 21, 2007. Current version published November 19, 2008. This work was supported by the Science and Education Ministry (MEC) under Projects RESELAI (REF-TIN2006-14896-C02-01) and SILPAR II (REF-DPI2006-05835).

The authors are with the Electronics Department, University of Alcala, 28871 Alcalá de Henares, Madrid, Spain (e-mail: ibravo@depeca.uah.es; ignacio.bravo@uah.es; mazo@depeca.uah.es; lazaro@depeca.uah.es; pjimenez@depeca.uah.es; alfredo@depeca.uah.es; marta@depeca.uah.es).

Digital Object Identifier 10.1109/TVLSI.2008.2001939

II. Eigenvalue and Eigenvector Computation

To obtain the eigenvalues of a matrix using specific hardware, diverse techniques have been proposed [7], all of them based on recurrent methods that look for matrix diagonalization. Within the different alternatives for the diagonalization process, the method proposed by Jacobi [11] is the more commonly used because it is easy to implement in terms of structure hardware of parallel processing.

The authors of this paper propose a no-systolic architecture, based on the method of Jacobi, which optimizes the resources consumed in the FPGA and at the same time maximizes speed of execution.

For the calculation of the eigenvalues of a Hermitian matrix $\mathbf{S} \in \Re^{M\times M}$, the algorithm proposed by Jacobi is used [3], [11]. This technique of diagonalization is based on the decomposition of the original matrix $\mathbf{S} \in \Re^{M\times M}$ in submatrices of 2×2 elements ($\mathbf{S}_{i,j} \in \Re^{2\times 2}$). These submatrices are defined in (1) and [3], where $s_{i,j}^{(k)}$ corresponds to elements set at zero in the matrix $\mathbf{S}^{(k)}$

$$\mathbf{S}_{ij}^{(k)} = \begin{bmatrix} s_{2i-1,2j-1}^{(k)} & s_{2i-1,2j}^{(k)} \\ s_{2i,2j-1}^{(k)} & s_{2i,2j}^{(k)} \end{bmatrix}, \qquad i,j = 1,2,\ldots,M/2. \quad (1)$$

The iterative process of diagonalization of matrix $\mathbf{S} \in \Re^{M\times M}$, proposed by Jacobi, meets the following condition: $s_{2i-1,2j}^{(k)} = s_{2i,2j-1}^{(k)} = 0; \forall i,j$. For it, the iterative process depicted in (2) is used

$$\mathbf{S}_{i,j}^{(k+1)} = \mathbf{R}\left(\alpha_{i,i}^{(k)}\right)^T \cdot \mathbf{S}_{i,j}^{(k)} \cdot \mathbf{R}\left(\alpha_{j,j}^{(k)}\right), \quad i,j = 1,2,\ldots,M/2 \quad k = 1,\ldots,M\log M. \quad (2)$$

where $\mathbf{S}_{i,j}^{(0)} = \mathbf{S}_{i,j}$. $\mathbf{R}(\alpha^{(k)})$ is a rotation matrix given by (3), and values $\alpha_{i,i}^{(k)}$ and $\alpha_{j,j}^{(k)}$ are the rotation angles given by (4)

$$\mathbf{R}(\alpha^{(k)}) = \begin{pmatrix} \cos\left(\alpha^{(k)}\right) & \sin\left(\alpha^{(k)}\right) \\ -\sin\left(\alpha^{(k)}\right) & \cos\left(\alpha^{(k)}\right) \end{pmatrix} \quad (3)$$

$$\tan\left(2\alpha_{i,i}^{(k)}\right) = \tan\left(2\alpha_{j,j}^{(k)}\right) = \frac{2s_{2i-1,2j}^{(k)}}{s_{2i,2j}^{(k)} - s_{2i-1,2j-1}^{(k)}} \quad i,j = 1,2,\ldots,M/2. \quad (4)$$

As observed in (4), the rotation angles are obtained from the diagonal submatrices ($\mathbf{S}_{i,i} \in \Re^{2\times 2}$). Therefore, it will be in the diagonal submatrices where diagonalization is obtained. The rotation angles of the diagonal submatrices $\mathbf{S}_{i,j} \in \Re^{2\times 2}, i \neq j$ are the calculated ones in the adjacent diagonal submatrices; these are $\mathbf{S}_{i,i} \in \Re^{2\times 2}$ and $\mathbf{S}_{j,j} \in \Re^{2\times 2}$.

After each iteration k (2), a phase of internal rearrangement of the elements of $\mathbf{S}_{i,j} \in \Re^{2\times 2}$ must be made and also a transference between submatrices of such form that locate in the diagonal submatrices the elements to be annulled [3].

After making $h = M\log M$ iterations, the first diagonal $\mathbf{S}^{(h)} \in \Re^{M\times M}$ corresponding to the original matrix $\mathbf{S} \in \Re^{M\times M}$ is obtained.

The eigenvectors for the matrix $\mathbf{S}$, which will be identified by $\mathbf{V} \in \Re^{M\times M}$, are obtained following the iterative process also proposed by Jacobi. Obtaining the matrix $\mathbf{V}$ is executed simultaneously with obtaining the eigenvalues. The process of obtaining $\mathbf{V}$ starts from an identity matrix $\mathbf{V}^{(0)} = \mathbf{I} \in \Re^{M\times M}$, which is also split in submatrices of elements $2\times 2 (\mathbf{V}_{i,j} \in \Re^{2\times 2})$ where, in each iteration, the value is updated according to the expression (5)

$$\mathbf{V}_{i,j}^{(k+1)} = \mathbf{V}_{i,j}^{(k)} \cdot \mathbf{R}\left(\alpha_{j,j}^{(k)}\right), \quad i,j = 1,2,\ldots,M/2; \quad k = 1,\ldots,M\log M \quad (5)$$

where $\mathbf{R}(\alpha^{(k)}) \in \Re^{2\times 2}$ is given by (3). After each iteration k (5), it is made similar to the calculation of eigenvalues, a process of rearrangement of the elements of each $\mathbf{V}_{i,j} \in \Re^{2\times 2}$ and achieving transference between adjacent submatrices identical to the eigenvalues

process [3]. After $h = M \log M$ iterations, the matrix of eigenvectors $\mathbf{V} = \mathbf{V}^{(h)} \in \Re^{M \times M}$ is obtained, where each column $v_j^{(h)}$ from $\mathbf{V}^{(h)}$ corresponds to the eigenvector associated with the eigenvalue $\lambda_{j,j}^{(h)}$ of $\mathbf{S}^{(h)} \in \Re^{M \times M}$.

III. New Parallel Architecture

The novel architecture shown in this work solves the problem of high number of internal resources consumed by the systolic approach [3] and implemented in [1], and it does not have high dependency with the size of the input matrix. As described before in Section II, a new architecture with an internal pipeline has been designed and implemented obtaining an excellent performance. From the point of view of the system speed, consumed resources, and accuracy, an excellent performance has been achieved, compared to other alternatives such us [1], [3].

The proposed algorithm is an iterative process with the following steps.

1) The first step for the calculation of eigenvalues is the storage of the matrix $\mathbf{S}_T$ in memory. Thanks to the condition of symmetry of $\mathbf{S} \in \Re^{M \times M}$, we can use its upper triangular matrix ($\mathbf{S}_T \in \Re^{M \times M}$) to obtain its eigenvalues and eigenvectors.
   At the time of handling the elements of $\mathbf{S}_T$, they must be grouped in 2×2 matrix size ($\mathbf{S}_{T_{i,j}} \in \Re^{2 \times 2}$), where the classification between diagonal ($\mathbf{S}_{T_{i,i}}$) and nondiagonal ($\mathbf{S}_{T_{i,j}}; i \neq j$) submatrices is made.
2) With respect to the calculation of eigenvectors, the initial matrix that is needed ($\mathbf{V}^{(0)}$) is an identity matrix $\mathbf{I} \in \Re^{M \times M}$ [3]. This matrix is stored in the same memory used for $\mathbf{S}_T$.
3) Once the memory contains all the initial data, the rotation angles $\alpha_{i,i}^{(k)}$ (4) are calculated using $\mathbf{S}_{T_{i,j}} \in \Re^{2 \times 2}$. The chosen method to solve (4) is CORDIC since obtaining the angles is one of the operations that this algorithm solves using a vectorization mode with circular coordinates [10]. The generated angles must be stored since they will be used in the process of calculating eigenvalues (2) and also extracting eigenvectors (5).
4) Once the rotation angles are obtained, the eigenvalue phase is started in the first term in order to optimize the global execution time of the whole algorithm, as it is justified later. Thus, each $\mathbf{S}_{T_{i,j}} \in \Re^{2 \times 2}$ must perform the indicated operation in (2). This operation includes two multiplications of submatrices of 2×2 elements, which are solved by using the CORDIC algorithm in a rotation mode for circular coordinates [10], since these multiplications correspond to the rotation of two vectors with an angle equal to $\alpha_{i,i}^{(k)}$.
   To solve (2) using CORDIC, the left product must be calculated first: $\mathbf{R}(\alpha_{i,i}^{(k)})^T \cdot \mathbf{S}_{T_{i,j}}^{(k)} = \mathbf{Q}_{i,j}^{(k)}$, where $\mathbf{Q}_{i,j}^{(k)} \in \Re^{2 \times 2}$; once obtained, the computation continues with the product of $\mathbf{Q}_{i,j}^{(k)} \cdot \mathbf{R}(\alpha_{j,j}^{(k)}) = \mathbf{S}_{T_{i,j}}^{(k+1)}$. If the number of stages of the CORDIC module, which coincides with the number of bits of the data (n) is such that $n > M$, when computing $\mathbf{Q}_{i,j}^{(k)}$ using CORDIC, an initial latency proportional to the number of stages of the internal architecture (n) is introduced. To optimize the use of CORDIC modules and reduce the execution time of calculating eigenvalues and eigenvectors, in practical applications in which $n > M$, the phase of calculation of eigenvectors (5) can be started using the same CORDIC module, during the latency of generation of the first element of $\mathbf{Q}_{i,j}^{(k)}$. The eigenvector computation phase is interrupted when this latency is surpassed and it is restarted when the operation $\mathbf{Q}_{i,j}^{(k)} \cdot \mathbf{R}(\alpha_{j,j}^{(k)}) = \mathbf{S}_{T_{i,j}}^{(k+1)}$ ends.
5) All the results of $\mathbf{S}_{Ti,j}^{(k)}$ and $\mathbf{V}_{i,j}^{(k)}$ must be stored in the same memory where the initial matrices had been stored. The form to store each $\mathbf{S}_{Ti,j}^{(k)}$ and each $\mathbf{V}_{i,j}^{(k)}$ must be in accordance to the process of rearrangement and transference of results [3].

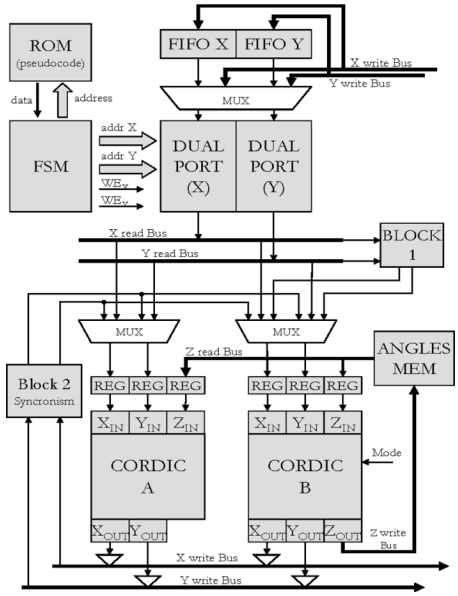

Fig. 1. New parallel architecture developed for the computation of eigenvalues and eigenvectors computation.

Doing the previous steps, the first iteration is finalized. Then, the system repeats steps 3)–5) during h iterations more.

For the implementation in FPGAs of the proposed algorithm the architecture shown in Fig. 1 has been developed. It may be noted that this new alternative is formed by only two CORDIC modules, memories to store the temporary and final results, registers, and multiplexers. The architecture proposed presents a temporal behavior and a processing time very similar to those given by a systolic array, while the number of resources used is fewer compared with those of other alternatives [1]. Another remarkable aspect of the developed architecture is the minimum increase in number of internal resources when the dimension of the input matrix is enlarged; only the size of Dual-Port and ROM memories must be modified. The elements forming the architecture shown in Fig. 1 are as follows.

- *ROM Memory*: It stores an ordered set of pseudocodes that are partitioned and decoded by the finite state machine (FSM). These pseudocodes correspond to the activation/deactivation of each control signal from Fig. 1 according to this algorithm-flow.
- *Dual-Port Memory (DP)*: It stores $\mathbf{S}_T$ and the identity matrix. Therefore, its size will be $(F + M^2)/2$ data with $2n$ bits for each one where F (6) is the number of data of the matrix $\mathbf{S}_T$. Also, this memory is used to store temporary data in each iteration and the final result of eigenvectors and eigenvalues. This memory type has been chosen to be able to make readings/writings simultaneously for coordinates x and y, for each of the vectors handled by CORDIC modules

$$F = M \cdot (M + 1)/2. \tag{6}$$

- *Finite-State Machine (FSM)*: It decodes the information of the ROM memory and activates the necessary control signals as well as the address for DP memory.
- *First-Input–First-Output (FIFO) Memories*: They temporarily store the eigenvectors in each iteration when the data bus of the DP memory is occupied by some phase of the eigenvalues computation.
- *CORDIC Modules*: Two CORDIC-A and CORDIC-B modules for circular coordinates have been used, whose internal structures are identical. The difference is that module A works only in rotation mode, whereas module B can execute rotation and vectorization

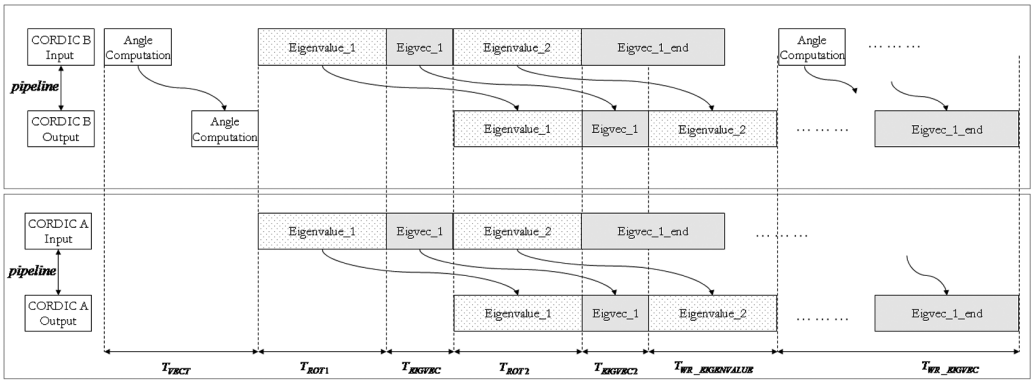

Fig. 2. Sequence of operations of CORDIC modules.

modes. This is possible if $n > M/2$, only one module working in vectorization mode is enough to obtain $M/2$ rotation angles $(\alpha_{i,i}^{(k)})$ with maximum efficiency. The internal structure of a developed CORDIC module corresponds to a parallel architecture of n stages. The maximum size of the input data, n, is not limited. Due to the existence of hardware multipliers in the output stage of each CORDIC module, in order to apply the correction factor of the CORDIC algorithm [10] the value of n will be limited to 18 bits.

- *Block 1*: This block is made up of registers, multiplexers and one adder/substractor. Its function is to generate the input data for CORDIC_B module while computing the rotation angles according to (4).
- *Block 2*: This block implements the feedback and interchange of values for the first and second rotation of the eigenvalue phase. It avoids storing these data in DP memory while making the transition between the first and second rotation.
- *ANGLES Memory*: In this one the rotation angles generated by the CORDIC_B are stored. The system addresses an angle based on the calculation phase in which the FSM is operating.

To further describe the operation of the system, note that the ROM memory of Fig. 1 contains the ordered sequence of each iteration: angle computation, double rotation for eigenvalues, and rotation for eigenvectors. The operation of this new proposal is described next, and its temporary sequence is shown in Fig. 2.

Angle Computation: Initially, after storing the values of the input matrix in DP memory, the first step is to calculate the rotation angle (4) ($\mathrm{T_{VECT}}$ from Fig. 2). If the input matrix has $M \times M$ elements in each iteration, calculate $M/2$ different rotation angles. The FSM of Fig. 1 is in charge of decoding the first pseudocodes of the ROM of this figure; these contain the information associated with the addresses of the DP that has the necessary operands for angle computation. Looking at expression (4), it can be seen that the denominator presents a subtraction operation. To make this operation, Block 1 is used; the result is sent to the input registers of the x, y coordinates of the CORDIC_B module. For the calculation of each rotation angle $\alpha_{i,i}^{(k)}$, according to (4), the following data $s_{2i-1,2j}, s_{2i,2j}, s_{2i-1,2j-1}$ are needed.

The process of angle computation is sequential, first take from the DP the three necessary operands for the first angle; next, the associated ones to the second angle; and so on. When the data of the last rotation angle to be calculated are introduced in CORDIC, the input of new data is interrupted until the latency of the CORDIC is surpassed.

At this time, no other calculation starts inasmuch as the phase of calculation of eigenvalues and that of eigenvectors both depend on these angles. When the angles are generated, they are stored in ANGLES memory of Fig. 1 to be read later in the phase of calculating eigenvalues and eigenvectors.

1) First Rotation of Eigenvalues (Eigenvalue_1): Once the last rotation angle is generated, the phase of eigenvalue computation is initiated. This phase is divided into two stages: first rotation $\mathbf{R}(\alpha_{i,i}^{(k)})^T \cdot \mathbf{S}_{T_{i,j}}^{(k)} = \mathbf{Q}_{i,j}^{(k)}$ ($\mathrm{T_{ROT1}}$ from Fig. 2) and second rotation $\mathbf{Q}_{i,j}^{(k)} \cdot \mathbf{R}(\alpha_{j,j}^{(k)}) = \mathbf{S}_{T_{i,j}}^{(k+1)}$ ($\mathrm{T_{ROT2}}$ from Fig. 2).

In both rotations, the necessary data $\mathbf{S}_{T_{i,j}}^{(k)}$ are stored in DP. This memory has a clock frequency ($\mathrm{T_{CLK2X}}$) which doubles the frequency for the rest of the sequential elements of the design proposal ($\mathrm{T_{CLK}}$). This is why four data can be extracted simultaneously from the DP in one $\mathrm{T_{CLK}}$, thus allowing the introduction of a data vector (x, y) for each CORDIC in the same clock cycle.

For the first iteration, the eigenvalue phase needs the input matrix $\mathbf{S}_T^{(0)} = \mathbf{S}_T$ while the eigenvector computation starts with an identity matrix $\mathbf{I} \in \Re^{M \times M}$. In the rest of the iterations, the results obtained from the previous iteration are used as input data.

2) First Part of Eigenvector Rotation (Eigvec_1): At the same time that the last data of the first rotation process of the eigenvalue calculation is introduced, and because of the latency of the CORDIC modules, the first elements $\mathbf{Q}_{i,j}^{(k)}$ may have not been obtained yet. Therefore, in that interval of time delay, the first data associated with eigenvector calculation are introduced. The time consumed in this phase is denominated $\mathrm{T_{EIGVEC}}$ in Fig. 2.

3) Second Part of the Eigenvalue Rotation (Eigenvalue_2): From the moment the first results of the first eigenvalue rotation are generated, the data input for the eigenvector computation is interrupted and the second rotation for the eigenvalue calculation begins ($\mathrm{T_{ROT2}}$ in Fig. 2). The results of the first rotation, are introduced into the CORDIC modules along with their generation. Thus, the second rotation can be initiated and again, the ANGLES memory must be addressed to introduce in both CORDIC modules the necessary angles for this second rotation.

While the new data for the second rotation are introduced, at the output of the CORDIC modules, the results of the eigenvectors introduced previously are obtained. These results are put on queue through FIFO memories and orderly stored in the DP.

4) End of Eigenvector Rotation (Eigvec_End): When the introduction of new input data for eigenvalue computation is finished, the remaining eigenvector data are extracted from the DP and they start to be inserted in the CORDIC modules ($\mathrm{T_{EIGVEC2}}$ from Fig. 2). The results of eigenvectors from *Eigenvec{_}1* phase that are continuously generated are stored in FIFO memories and will be loaded from these into the DP when this one becomes free.

When the first data of eigenvalue computation inside the *Eigenvalue_2* phase are generated, these are accumulated in an orderly fashion in DP ($\mathrm{T_{WR\_EIGENVALUE}}$). At this moment, the iteration has still not been finalized since they have left results for eigenvectors that have still not been generated (*Eigenvec_1_end* phase).

Nevertheless, to accelerate the operation of the system, it is exactly at this moment when the next iteration begins. After a certain number of iterations, h, the process of eigenvalue and eigenvector calculation can be considered final.

IV. Results

This section describes the temporary results, errors, and resources consumed using a Xilinx XC2VP7 FPGA obtained from the calculation of eigenvalues and eigenvectors for input symmetrical matrices.

The results obtained after the Place&Route process show that our new architecture performs efficiently from the point of view of precision. To obtain maximum precision in calculating eigenvalues and eigenvectors, it is important to fix the optimal number of the necessary h iterations. As indicated previously, the empirical form $h = M\log(M)$ can be a possible solution [3]; nevertheless, depending on the precision desired for obtaining the eigenvalues and consequently the eigenvectors, it is necessary to look for the optimal value of h. The number of optimal iterations given in [3] has been obtained experimentally for a systolic architecture, thus in our case the optimal value of h has also been found empirically. After several simulation-tests for different n-sizes and M-sizes, it can be concluded that, due to the internal structure of the proposed architecture in this work, the optimal value is $h = 6 + M\log(M)$, for all M value.

Another important aspect that determines the final precision of the results is the different rounding elements that take place inside the CORDIC modules due to the internal arithmetical operations. Particularly, there are three internal rounding blocks to be considered: 1) on the shift registers; 2) maximum input size of the multiplying hardware used in the correction factor; and 3) output size of the hardware multipliers. Thus, the effect of the three previous situations will be considered, comparing the corresponding errors with the one obtained using truncation. After different simulations using rounding instead truncation in the internal CORDIC elements, the eigenvector average quadratic error is improved at least by 85%.

The temporary study of the proposal made in this work to calculate eigenvalues and eigenvectors, and calling $\mathrm{T_{EIG}}$ the time consumed in a complete iteration of the system described in Fig. 2, the total time necessary to obtain the eigenvalues and eigenvectors will be $\mathrm{T_{TOTAL}}$

$$T_{\mathrm{TOTAL}} = L_i + h \cdot T_{\mathrm{EIG}} + L_f \tag{7}$$

where Li corresponds to an initial latency associated to the load of the input matrix $(\mathbf{s}_T)$ and the identity matrix on the DP memory, considering that initial matrices are stored in external memory.

On the other hand, $\mathrm{T_{EIG}}$ is the time consumed in each iteration of the process of eigenvalue and eigenvector computation. This time is decomposed into the sum of the vectorization time $(\mathrm{T_{VEC}})$ (time of generation for the rotation angles) and the time for the eigenvalue computation $(\mathrm{T_{EV}})$. Again, it is divided into a series of processing times: $\mathrm{T_{RD\_DP}}$ (time needed to read from the DP the data needed for the angle calculation), $\mathrm{T_{REG}}$ (time associated to the load of the three inputs of CORDIC B), and $\mathrm{T_{CORDIC\_VEC}}$ (time that takes CORDIC B to calculate all angles necessary to make the rotations). Thus, the value for $\mathrm{T_{VEC}}$ is obtained by expression

$$\begin{aligned} T_{\mathrm{VEC}} &= T_{\mathrm{RD\_DP}} + T_{\mathrm{REG}} + T_{\mathrm{CORDIC\_VEC}} \\ &= (1 + n + M/2)T_{\mathrm{CLK}}. \end{aligned} \tag{8}$$

Once the vectorization stage is finalized, the phase of calculation of eigenvalues and eigenvectors in the next iteration begins. As described in Section III and as shown in Fig. 2, the implementation of a pipeline that optimizes the speed of the system has been chosen.

It has been proven that the novel proposal developed in this work improves maximum work frequency by 23% and reduces considerably (a factor of 16 times less) the number of consumed resources. If the proposed architecture in this work is compared with the classic systolic architecture of [3], the saving of internal resources of the FPGA is quite significant, without penalizing the run time of the proposal with respect to the classic one.

V. Conclusion

In this work, a new no-systolic proposal for the calculation of eigenvectors and eigenvalues is implemented completely in an FPGA. This new architecture is based on the Jacobi method for matrices of 2×2 elements. The newly developed proposal improves indices as frequency or resources consumed with respect to other similar works implemented in FPGAs. In addition, one of the main contributions of the proposed alternative is the use of a novel architecture, which occupies a reduced number of internal resources of the FPGA compared with the classic systolic architecture.

Using this novel architecture with different matrix sizes, the number of additional consumed internal resources is practically negligible, as it is only necessary to modify the size of the ROM and DP memories. On the other hand, an increase in data size (n) implies an increase in the number of stages of CORDIC modules. Nevertheless, as was reflected in the results section, working only with two modules has resulted in an increase that is remarkably inferior to the classic systolic architecture.

References

[1] A. Ahmedsaid, A. Amira, and A. Bouridane, "Accelerating MUSIC method on reconfigurable hardware for source localization," in *Proc. Int. Symp. Circuits Syst. (ISCAS)*, 2004, vol. 3, pp. 369–372.

[2] C. Bodda and N. Steenbock, "Singular value decomposition on distributed reconfigurable systems," in *Proc. 12th Int. Workshop Rapid Syst. Prototyp.*, 2001, pp. 38–43.

[3] R. P. Brent, F. T. Luk, and C. Van Loan, "Computation of the singular value decomposition using mesh-connected processors," *J. VLSI Comput. Syst.*, vol. 1, no. 3, pp. 242–270, 1983.

[4] M. Cavadini, M. Wosnitza, and G. Troster, "Multiprocessor system for high resolution image correlation in real time," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 3, pp. 439–449, Jun. 2001.

[5] A. El-Amawy, "A systolic architecture for fast dense matrix inversions," *IEEE Trans. Comput.*, vol. 38, no. 3, pp. 449–455, Mar. 1989.

[6] W. Haiqing, S. Zhihuan, and L. Ping, "Improved PCA with optimized sensor locations for process monitoring and fault diagnosis," in *Proc. 39th IEEE Conf. Dec. Control*, 2000, vol. 5, no. 5, pp. 4353–4358.

[7] T. J. Herron, K. M. Reddy, R. Garg, and K. Devanahalli, "Eigen decompositions of covariance matrices on a fixed point DSP," presented at the Nat. Conf. Commun., Indian Inst. Technol., Madras, India, 2003.

[8] J. A. Jimenez, M. Mazo, J. Urena, A. Hernandez, F. Alvarez, J. J. Garcia, and E. Santiso, "Using PCA in time-of-flight vectors for reflector recognition and 3-D localization," *IEEE Trans. Robot.*, vol. 51, no. 5, pp. 909–924, Oct. 2005.

[9] C. M. Liaw, "Optimal controller with prescribed dominant energy eigenvalues," *IEE Proc. D Control Theory Appl.*, vol. 138, no. 4, pp. 405–409, 1991.

[10] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. AFIPS Spring Joint Comput. Conf.*, 1971, pp. 379–385.

[11] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. Oxford, U.K.: Oxford Science, 1999.

[12] D. Yang and V. Ajjarapu, "Critical eigenvalues tracing for power system analysis via continuation of invariant subspaces and projected Arnoldi method.," *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 324–332, Jan. 2007.