

# Parallel Processors Architecture in FPGA for the Solution of Linear Equations Systems

R. MARTINEZ, D. TORRES, M. MADRIGAL, S. MAXIMOV

Program of Graduated and Investigation in Electrical Engineering as the Morelia Institute Technology

Av. Tecnológico 1500, Col. Lomas de Santiaguito, Morelia, Michoacán  
MEXICO

ruben.martinez08@cfe.gob.mx <http://elec.itmorelia.edu.mx>

**Abstract:** - This paper presents a parallel array of processors implemented in a Field Programmable Gate Array (FPGA) for the solution of linear equations systems. The solution is performed using the division-free Gaussian elimination method. This algorithm was implemented in integrated processors in a FPGA Spartan 3 of Xilinx. A top-down design was used. The proposed architecture can handle IEEE 754 single and double precision floating-point data and the architecture is implemented in 240 identical processors. Also, an algorithmic complexity of  $O(n^2)$  was obtained using a  $n^2$  processors scheme that perform the solution of the linear equations. The parallel division-free Gaussian elimination method, the architecture's data distribution, the internal processor-element architecture and the communication scheme between processor elements (PE) are presented. Finally, this paper presents the obtained simulation results and synthesis of the modules designed in Very High Description Language (VHDL) using 40 and 100 Mhz frequencies.

**Key-Words:** - Field Programmable Gate Array (FPGA), Very High Description language (VHDL), Parallel Processing, Parallel Architectures, linear systems equations, Division Free Gaussian elimination Method.

## 1 Introduction

Linear systems are commonly found in many scientific fields, and they can range from a few to millions variables. Consequently, for large linear equation systems, the system solution involves long computational times. For this reason, parallel processing emerges as a good option for the solution of linear systems [1]. In the past years, many methods for the solution of linear system have been proposed. These methods present many advantages and drawbacks [2] and the selection of the method rests in the problem to be solved [2]. The technological development in parallel processing, computer systems and electronic digital devices have evolved engineering. The design of new parallel architectures for the solution of engineering problems is becoming popular due to the advantages of parallel processing [3]. For instance, repetitive process can be executed simultaneously in order to reduce computational efforts. Therefore, the application of parallel techniques in methods for linear systems solutions can improve significantly its performance.

Recently, parallel processing has been applied for the solution of problems in image processing, finite element, mathematical algorithms and power

electrical systems. It is important to say that due to the behavior and mathematical modeling of some physical systems, its solution of can be parallelized [5]-[7].

Section 2 presents the mathematical model of the one-step division-free Gaussian elimination method as well as its equations, conditions, algorithmic complexity and computational algorithm. Section 3 presents the proposed parallel architecture, named here Division-Free Parallel Architecture (DFPA). It also depicts the vertical and horizontal processing performed by PEs. The implementation in FPGA of the proposed architecture, the obtained simulations and synthesis of every module of the designed architecture is presented in Section 4. Section 5, shows the performed tests, obtained results and comparatives of the behavior of the proposed architecture against other architectures reported in the literature. Finally, Section 6 presents the conclusions.

## 2 One-Step Division-Free Bareiss Method

The characteristics of the one-step division-free method make it suitable for its implementation in an

array of processors. Let a linear system of equations be given by [1, 8].

$$\begin{aligned}
 A &= (a_{ij}), 1 \leq i, j \leq n, \\
 b &= (a_{ij}), 1 \leq i \leq n, n+1 \leq j \leq m \\
 x &= (a_{ij}), 1 \leq i \leq n, 1 \leq j \leq m-n.
 \end{aligned}$$

In general, the algorithm that reduces the matrix  $A$  to a diagonal form is more complex than the algorithm that reduces it to a triangular form. In this paper, the one-step division-free Gaussian elimination algorithm is used to reduce the matrix  $A$  to a diagonal form. The one-step division-free Gaussian elimination algorithm is described in (1).

$$\begin{aligned}
 a_{ij}^{(0)} &= a_{ij}, \quad 1 \leq i \leq n, 1 \leq j \leq m; \\
 1 \leq k \leq n, 1 \leq i \leq n, k \leq j \leq m; & \quad (1) \\
 a_{ij}^{(k)} &= \begin{cases} a_{ij}^{(k-1)}, & \text{if } i = k \\ \begin{bmatrix} a_{kk}^{(k-1)} & a_{kj}^{(k-1)} \\ a_{ik}^{(k-1)} & a_{ij}^{(k-1)} \end{bmatrix}, & \text{otherwise} \end{cases}
 \end{aligned}$$

At each iteration, it is possible to compute all the  $2 \times 2$  determinants at the same time. Consequently, the algorithmic complexity for the computation of the determinants is given by  $nd = O(n)$  where  $nd$  is the number of determinants to be computed in one iteration,  $n$  is the number of rows and  $m$  is the number of columns of matrix  $A$ . Equation (2) shows the algorithm complexity.

$$O(n) = n \times m \quad (2)$$

where  $O(n)$  is the number of determinants to compute in each iteration,  $n$  is the number of rows,  $m$  is the number of columns, and  $m = n+1$ ; then:

$$O(n) = n^2 + n \quad (3)$$

Since the major exponent in (3) is  $n^2$ , the algorithmic complexity is  $O(n^2)$ . Data dependency exists in each iteration. Since iteration  $k+1$  requires the data computed in the preceding iteration, the iteration  $k$  has to be previously computed.

### 2.1 Proposed Architecture

The proposed architecture consists in an array of processors distributed in a matrix form, where the number of  $PEs$  is  $n^2$ . A master processor is in charge of the data distribution between  $PEs$ . The master processor distributes the data that will be used for each  $PE$  during the first iteration. The  $PE$  performs sums and multiplications to compute a  $2 \times 2$  determinant. The  $PEs$  require eight memory locations: the first four to store the four variables,

the fifth and sixth location to store the multiplication results, the seventh position to store the sum and the eighth location to store the  $PE$  identification number. Each processor can be uniquely identified by its position onto the processor grid.

---

#### Algorithm: One step division free method

---

```

for (k=0; k<n; k++)
  for (i=0; i<n; i++) // Row
    for (j=0; j<n+1; j++) // Column
      if (k==i)
        D[i][j]=C[k][j];
      else
        D[i][j]=(C[k][k]*C[i][j])-(C[i][k]*C[k][j]);
      end if
    end for
  end for
  for (p=0; p<n; p++) // matrix C
    for (q=0; q<n+1; q++)
      C[p][q]=D[p][q];
    end for
  end for
end for //end program

```

---

### 3 Parallel Architecture of the Processor

This section presents the proposed architecture DFPA, based in the Bareiss method, for the solution of linear equations. Also, the vertical and horizontal processing into the array of  $PEs$  is described. The proposed architecture is composed by an array of  $PEs$ , each  $PE$  compute the Bareiss determinant defined in (1). The processors are arranged as a grid. Fig. 1, depicts the processor grid.

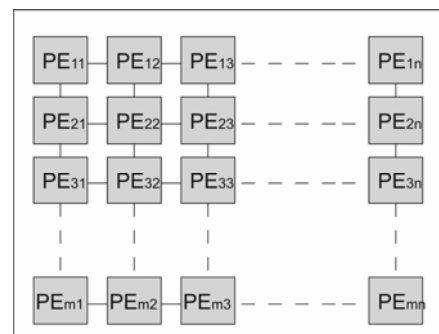


Fig. 1. Matrix form of the grid of  $PEs$ .

Every  $PE$  into the DFPA Architecture receives the Bareiss coefficients through the rows of the grid. The processor  $PE_{1n}$  receives the coefficient  $a_{1n}$  and successively for the rest of processors until the processor  $PE_{mn}$  receives the coefficient  $a_{mn}$ . These coefficients conforms the Bareiss determinant coefficients. Once the four coefficients are received in each  $PE$  the determinant is computed. This processing is performed in horizontal form as shown

in Fig. 2.

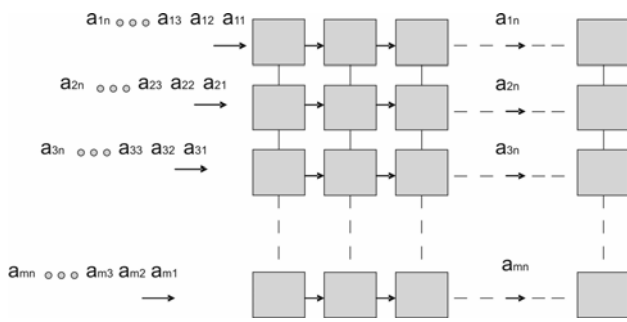


Fig. 2. Horizontal processing of the coefficients by rows.

The coefficient  $a_{11}$  is sent to every PE using the processor element  $EP_{11}$ . At first, this coefficient is sent in vertical form to every  $EP_{1n}$ , then each processor sends this coefficient in horizontal form to all the rows in the grid of PEs as shown in Fig. 3. The horizontal processing is used for the Bareiss determinant computation. After that, the result is sent to the others processors for the computation of the next iteration.

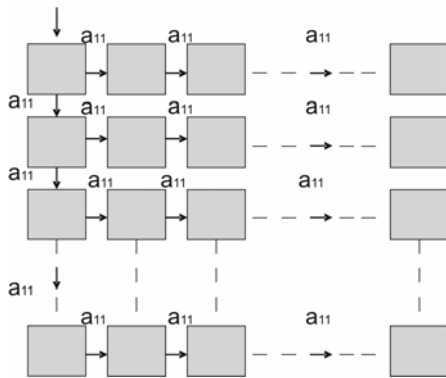


Fig. 3. Processing of element  $a_{11}$ .

Fig. 4 shows the vertical processing. Once the first determinant at each iteration is computed, the vertical processing is performed to compute and send the obtained results to the rest of processors in vertical form. The vertical processing comprises of sending the first row ( $a_{11}, a_{12}, a_{13}, \dots, a_{1n}$ ) to the PEs arranged in the following rows on the processor grid. This procedure permits the Bareiss determinant of the next iteration to be constructed.

A horizontal processing is performed to compute and send the first column coefficients ( $a_{11}, a_{21}, a_{31}, \dots, a_{m1}$ ) to the rest of the PEs arranged in subsequent columns on the grid as shown in Fig. 5. This procedure

completes the Bareiss determinant for iteration 1.

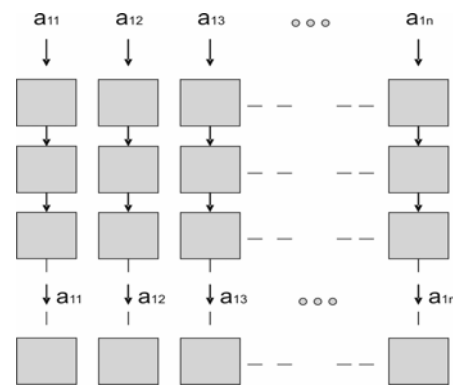


Fig. 4. Vertical processing of the coefficients in the first row of the proposed architecture.

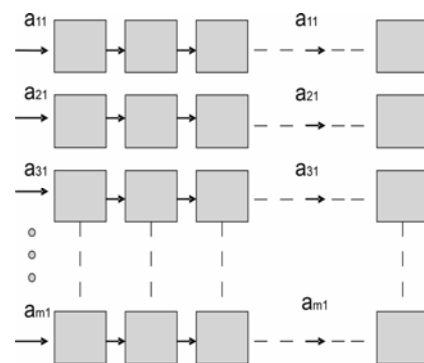


Fig. 5. Horizontal processing of the coefficients in the first column of the proposed architecture.

Fig. 6 depicts the allocation of the coefficients onto the PEs grid for any iteration. Also, it can be seen in the figure how the different determinants are conformed. A PE basically consists of a multiplier, an adder and an accumulator. The PE receives four 32-bits data to solve a Bareiss determinant performing two multiplications and a sum. Also, each received data is stored into a register in the internal memory of the processor.

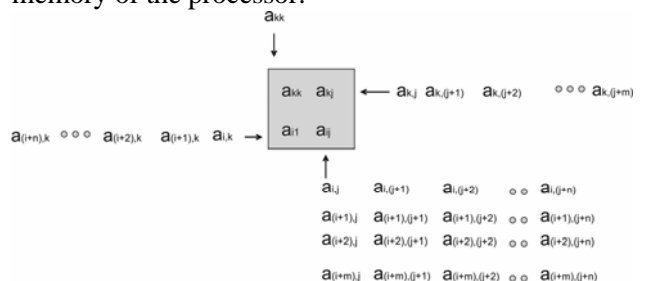


Fig. 6. EP processing for iteration  $n$ .

Similarly, the multiplication results are stored into two registers and the sum result in another register. Consequently, each PE is composed of eight

registers. In order to solve the linear equation system, the *PEs* need to communicate with their processor neighbors. Fig. 7 shows the internal blocks that compose a *PE*.

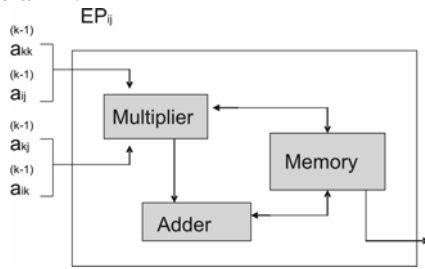


Fig. 7. Processor Element.

### 3.1. Communication Scheme between processor elements

The communication between processors is granted by the use of read/write communication channels. These channels transmit the coefficients from a *PE* to another. Vertical channels are used for the columns onto the processor grid whereas horizontal channels are used for rows. Fig. 8, presents the interconnections of communication channels between *PEs*. One the communication channels between processor are defined, it is necessary to define the events that take place in the proposed processor array in every clock cycle; this is, to define the information flow inside the processor array for each clock cycle. For instance, consider that we are in the iteration 1 and all *PEs* already have stored the coefficients needed for the computation of the iteration 2.

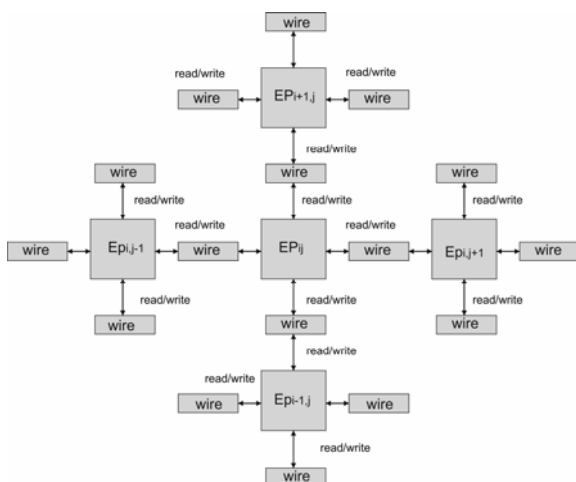


Fig. 8. Communication channels between PEs.

## 4. FPGA implementation

A FPGA device, is used for the processing phase. The FPGA's consist in a bidimensional matrix

composed of configurable block that can be connected by general resources of interconnection [7]. The synthesis of the modules presented in this paper was developed in the ModelSIM SE 6.3 f software. This software permits a top-down design. The higher level of abstraction employs VHDL, whereas the lower level is represented by logic gates. In this section, the synthesis of the modules of proposed parallel architecture, based in the parallelization of the one-step division free Gaussian elimination, is presented. The synthesis of the architecture was developed in the Xilinx ISE 8.1i software for FPGAs of the Spartan 3 family of Xilinx, to obtain the number of Gates, IOBs, CLBs, Slices, Shift, Flip Flops, and LUTs. The connection diagrams between digital components with different abstraction levels were also obtained. The proposed architecture is composed by the following components or modules: An adder-subtractor; a multiplier; a serial data input/output; memory registers and control registers. All the modules were programmed in VHDL language and it includes standard IEEE libraries Fig. 9 Shows all modules designed. In the simulations of the designed VHDL modules presented in this paper, a pulse clock of 100 ns was considered.

Table 1  
Times Measurements.

Description	Operations	Time per operation (sec)	Total time per operation (sec)
Adder-subtractor	1	$4000 \times 10^{-9}$	$4000 \times 10^{-9}$
Multiplier	2	$1500 \times 10^{-9}$	$3000 \times 10^{-9}$
Memory registers	6	$1500 \times 10^{-9}$	$9000 \times 10^{-9}$
Data input/output	4	$1325 \times 10^{-9}$	$5300 \times 10^{-9}$
Control	4	$100 \times 10^{-9}$	$400 \times 10^{-9}$
Total Time			$21700 \times 10^{-9}$

The synthesis of the modules describes in this section are shown in Table 2.

Table 2  
Number of components per Processor

Module	Total Gates	CLB's	IOB's	Shift	Mult 18x18	FF	Lut 4 input
Adder-subtractor	765	32	193			64	64
Multiplier	17829	56	129		4	64	111
Memory registers	515		129			64	
Data input/output	267	2	4	4		1	
Control	756	42	12			33	47
Total Components	20132	132	467	4	4	226	222

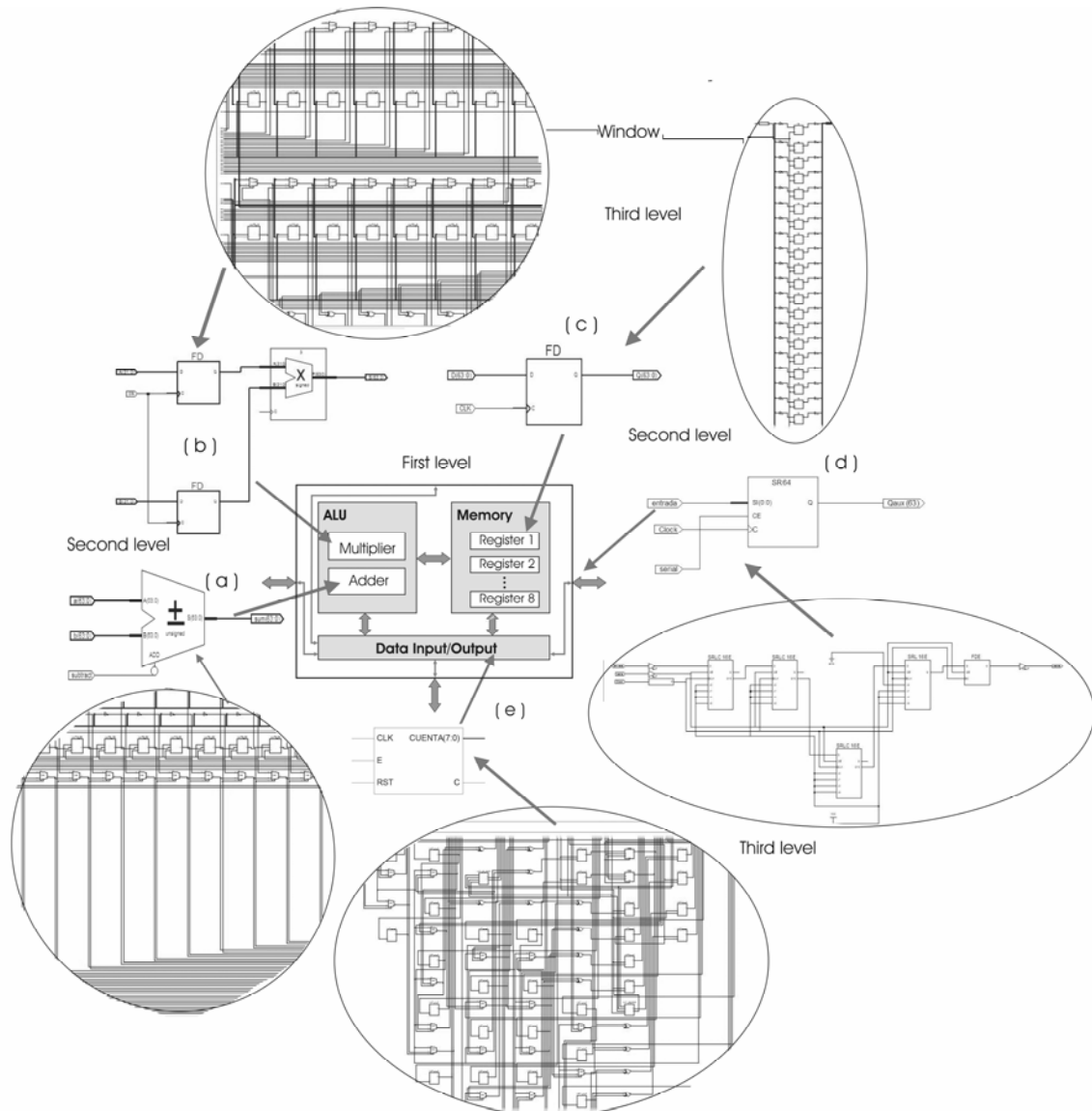


Fig. 9. (a) Adder-subtractor block, (b) Multiplier block, (c) Memory register block, (d) Data input/output block, (e) Data sending controller block.

## 5 Experimental results

In this paper, a parallel architecture, named here Division-Free Parallel Architecture (DFPA), was designed. A comparison of the obtained results against the results reported in [9] is presented. In [9], a parallel architecture for linear equation solution using the LU algorithm, named here “Parallel LU” is described. This architecture uses a 40 Mhz frequency for different matrix sizes.

Table 3 shows a comparison of the obtained times of the DFPA against Parallel LU. An improvement factor is also presented. If this factor is greater than one, then the DFPA is better than the Parallel LU.

Table 3  
Comparison DFPA vs Parallel LU

Matrix	DFPA (sec)	Parallel LU (sec)	Improve
24	1.36E-04	6.84E-04	5.02
30	1.70E-04	1.41E-03	8.26
36	2.04E-04	1.17E-03	5.73
42	2.38E-04	1.66E-03	6.96
48	2.72E-04	3.00E-03	11.02
54	3.06E-04	4.89E-03	15.94
96	5.45E-04	1.67E-02	30.58

Moreover, the architecture presented in [10], named here “Pipeline LU”, was compared against the proposed architecture. Also, a comparison with the architecture Pipeline LU against a 1.6 Ghz Pentium M processor is presented in [10] and used in this paper for comparison purposes. Table 4 presents the comparison of computation time for matrices between 100 and 1000 equations. It can be seen in

the table that the DFPA and Pipeline LU architectures use a 100 Mhz frequency.

Table 4  
Comparison DFPA vs Pipelien LU (time ms)

Matrix	DFPA 100 Mhz	Pipeline LU 100 Mhz	Pentium M (1.6 GHz)	Improve vs Pentium	Improve vs Pipeline LU
100	0.227	0.46	9.11	40.13	2.03
300	0.681	8.76	134.20	197.06	12.86
500	1.14	40.50	661.00	579.82	35.53
800	1.82	167.60	2984.50	1639.84	92.09
1000	2.27	328.40	7871.50	3467.62	144.67

## 6 Conclusion

During the revision of mathematical algorithms for the solution of linear equation systems, the methods that use division require a major processing time and its implementation in hardware produces complex architectures. However, the division free method proposed by Bareiss [8] presented many advantages for parallelization. For this reason, this method was selected and it is the base of the proposed parallel architecture in a FPGA.

The parallelization of the division free Gaussian elimination methods produces a simple independent process that can be implemented in identical processors and its hardware implementation is easily constructed by using basic algebraic operations. The obtained algorithmic complexity is  $O(n^2)$  under a scheme of  $n^2$  processors that solve a linear equation system of  $n$  order.

The performed simulations of the modules that compose a processor show a low time results in nano-seconds for this kind of computations.

Finally, the construction of VHDL modules for digital systems for the simulation and synthesis of a matricial processor is possible using the ModelSIM software and Xilinx ISE software.

### References

[1] Shietung Peng, Stanislav Sedukhin, Parallel Algorithm and Architecture for Two-step Division-free Gaussian Elimination, *IEEE Transactions*, 0-7803-4229-1/97, 1997, pp. 489-502,  
 [2] Fernando Pardo Carpio, *Arquitecturas Avanzadas, Universidad de Valencia, España, Enero 2002.*  
 [3] M. J. Beauchamp, Scott Hauck, Keith D. Underwood and Scott Hemment, Architectural

Modifications to Enhance the Floating-Point Performance of FPGA's, *IEEE Transactions on VLSI Systems*, Vol. 16 No. 2, February 2008, pp. 177-187.

[4] D. Torres, Herve Mathias, Hassan Rabah, and Serge Weber, SIMD/restricted MIMD Parallel Architecture for Image Processing Based on a New Design of a Multi-mode Access Memory, presented at *International Conference on Parallel and Distributed Processing Techniques and Applications PDPTA'99*, Las Vegas Nevada USA, 1999.  
 [5] Rubén Martínez Alonso, Domingo Torres Lucio, Paralelización del Algoritmo del Método de Bareiss Libre de División para Solución de Sistemas de Ecuaciones Lineales de Ingeniería Eléctrica, *4th International Congress and 2nd National Congress of Numerical Methods in Engineering and Applied Sciences* ISBN 978-84-96736-08-5, Morelia, Michoacán, México, del 17 al 19 de Enero de 2007.  
 [6] Ronald Scrofano, Ling Zhuo, Vicktor K. Prasanna, Area-Efficient Arithmetic Expression Evaluation Using deeply Pipelined Floating-Point Cores, *IEEE Transactions on VLSI Systems*, Vol. 16 No. 2, February 2008, pp. 167-176.  
 [7] Xilinx, DS099. Spartan-3 Family, complete data sheet, <http://www.xilinx.com>, product specification.  
 [8] Bareiss E. H. Sylvester's Identity and Multistep Integer-Preserving Gaussian Elimination, *Mathematics of Computation*, 22, 1968, pp. 565-578.  
 [9] Xiaofang Wang and Sotirios G. Ziavras, Parallel Direct Solution of Linear Equations on FPGA-Based Machines, *IEEE Proceedings Symposium, (IPDPS 2003) Parallel and distributed Processing*, 2003.  
 [10] Vikash Daga, Gokul Govindu, Viktor Prasanna, Efficient Floating-Point based Block LU Decomposition on FPGAs, *ERSA 2005*, pp.137-148. Las Vegas Nevada, USA, pp. 137-148, Jun, 21-24.2004.