Parametrizable Architecture for Function Recursive Evaluation

Juan Manuel García Chamizo
Mª Teresa Signes Pont
Higinio Mora Mora
Gregorio de Miguel Casado

Departamento de Tecnología Informática y Computación-Universidad de Alicante
{juanma, teresa, hmora, demiguel}@dtic.ua.es

Abstract.- **This paper presents a function evaluation method developed under the scope of recursive expression of function convolution. This approach is based on a unique parametrizable formula capable of providing function points by successive iteration. The computational interest of the method is related to the compactness of the formula specially when calculating combinations of functions . When tackling design level, it also shows suitable for developing architectural schemes capable of dealing with different speed and precision issues.**

An architecture for reconfigurable FPGA based in distributed arithmetic implements the design for fast prototyping. The case of combined trigonometric functions involved in rotation is analyzed under this scope. Compared with others methods, our proposal offers a good balance between speed and precision.

Keywords.- **convolution, function evaluation, FPGA, serial-distributed arithmetic.**

I. INTRODUCTION

Research work in function computing has provided a set of well known approaches. That is the case of the CORDIC algorithm (COordinate Rotation DIgital Computer), wich provides function approximation using rotations in a three coordinate system: linear, circular and hyperbolic [1][2][3][4]. This algorithm only uses additions, subtractions and shift operations so that to perform the calculations. Several VLSI designs have been developed for CORDIC application to digital signal and image processing [5][6][7], matrix algebra and robotics [8][9][10][11]. Generally, high speed computation is provided by using pipelining and/or redundant arithmetic [12].

Regarding to the calculation of elementary functions, most methods use Look-Up Table (LUT) schemes. Newton-Raphson algorithm [13] provides a quadratic convergence iterative method that achieves a good approximation for inverse, square root and inverse square root, initializing computation with a seed [14][15]. Interpolation methods provide function approximation by using polynomial, rational or Taylor series, by means of coefficient computing. In polynomial interpolation [16], coefficient calculation is performed so that to minimize the relative error in the interval [17]. Tchebyshev's node calculation of coefficients performs interval partitioning in N parts, where N is determined by a previously obtained pattern. The computation of Stirling's coefficients is based on some function values that can be precalculated and stored in a table or just even calculated on the fly. For applications that require high speed and low-precision, LUT based schemes are often employed, with a great concern in the increasing size of the memory needed as the required precision grows. In general, multipartite LUT schemes [18][19] are considered a suitable approach because the amount of memory used becomes drastically reduced. However, a multi-operand adder is required in order to sum the table outputs.

This paper presents a recursive method that performs function evaluation by means of successive iteration on a parametrizable formula. Following the introduction, section 2 develops the basic concepts of the Convolution Based Recursive Method (CBRM). Section 3 is deals with the CBRM architecture design principles as well as implementation issues and cycle time analysis. Section 4 develops an evaluation of the architecture proposed. In section 5, an application for the calculation of the combined trigonometric functions involved in rotations (sine and cosine) is discussed. Section 6 deals with CBRM comparison with others approaches and section 7 summarizes results and presents the concluding remarks.

II. CONVOLUTION–BASED RECURSIVE METHOD (CBRM)

This section introduces some fundamental concepts of the Convolution-Based Recursive Method (CBRM) and outlines its particular features. Mathematical

demonstrations that are not under the scope of this paper will be omitted.

Convolution is an operation between two functions that is relevant to many different applications in digital signal and image processing [20], control engineering [21], mathematical morphology [22] or pattern analysis [23]. All these applications share as a common feature the calculation of mathematical spatial or temporary transforms that must be carried out by means of convolution. Generally, convolution is difficult to calculate. This drawback is faced by substituting the initial convolution expression of the functions by the product of these functions which have been previously transformed to frequential ones. However, CBRM is not concerned directly with this well-known duality: it rather exploits the primitive meaning of convolution. That is, the convolution between two functions is a mean to evaluate one of them using the other one as a unit [24]. This fundamental provides a powerful basic tool for function evaluation purposes. Equation (1) shows the expansion of convolution in the case of two discrete functions f and g in the interval $[0, +\infty)$. The result of the operation is also the function Y, which represents the evaluation of f by g (or g by f).

$$\begin{aligned} &\Psi(0) = f * g(0) = f(0) \cdot g(0) \\ &\Psi(1) = f * g(1) = f(0) \cdot g(1) + f(1) \cdot g(0) \\ &\dots \\ &\Psi(p) = f * g(p) = f(0) \cdot g(p) + \dots + f(i) \cdot g(0) \\ &\dots \end{aligned} \qquad (1)$$

In the case of discretized functions, a recursive formulation of the convolution can easily be achieved from (1) for function Ψ, providing a more compact and useful formula, see equation (2).

$$\Psi_{i+1} = \alpha \Psi_i + \beta G_i \qquad i \in [0, I] \subset \mathbb{N} \qquad (2)$$

Reciprocally, it can be demonstrated that any discrete recursive equation represented by a weighted sum of two terms like those of equation (2) can be split up into two convolving discrete functions (namely $f(i) = a^i$ and $g(i)=b\ G(i)$), if $\alpha \neq 0$. The equivalence between (1) and (2) provides an approach for discrete function evaluation suitable in many cases (trigonometric, hyperbolic, logarithmic, exponential, inverse, square-root, constant, lineal functions).
Particular features of CBRM are:

- CBRM is not concerned with the particular structural characteristics of the evaluated function because it only uses the Y value computed at i iteration as an explicit argument which allows the computation of the next value at iteration $i+1$.
- The algorithm runs under a unique compact recursive formula and suits in a great amount of cases.
- The evaluation of any function Y is carried out as a sum of two parts which have fixed contributions held by parameters a and b (see equation 2). G is an auxiliary function that its value can be provided by application parameters or evaluated in the same way as Y. It also can be a function of Y itself.
- Parameters a., b and function G characterizes the function Y regarding to behavioural aspects and iteration path. Attended that CBRM runs under a unique formula, the combinations of a., b values are crucial and little changes in any of these values affects dramatically the overall behaviour. Parameter space is tightly related with behavioural features.
- CBRM provides a new Y-value per iteration.
- The initial values of Y and G are fixed by the user specifications.

A *convolution table* can be built for providing the (a, b, G) associated to the calculation of $f*g$, but it must be noticed that in spite of the generic mathematical equivalence between (1) and (2), the computational usefulness of CBRM reduces when function G is not easy to carry out.

III. Architecture

The formalization of CBRM set by equation (2) leads to the functional architecture shown in figure 1, where each Y-value calculation involves two multiplications and one sum.

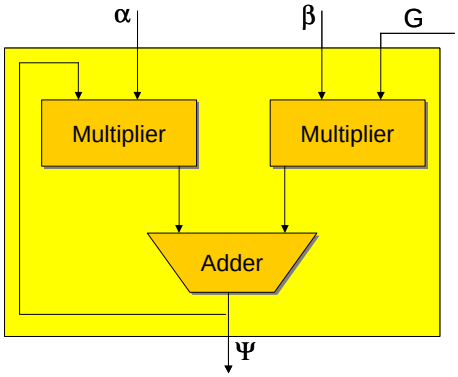

Figure 1: CBRM functional architecture

A. Design principles

The proposed design consists of using look-up tables (LUT) to store α and β constants and to make the effective calculation.

The implementation of the architecture using look-up tables provides a greater density of VLSI integration than in combinational methods. In this design, advances in technology play a determining role in performance improvement and in the reduction of the temporal delay [25]. The location of the look-up tables inside the arithmetic unit next to the rest of the operation's logic also reduces memory access time.

On the other hand, the use of look-up tables decrease hardware development costs, contributes flexibility and limits the number and variability of modules required. In addition, they can incorporate

elements of error detection and correction and, therefore, improvements in the results they produce.

Memory construction offers the architecture the possibilities of reliability and flexibility: the use of read only memories is a more robust option than combinational circuits and, alternatively, the use of read and write memories enables several different functions to be configured in the same chip. In addition, this option improves the maintenance and repair of the arithmetic unit [14], [26].

Finally, we stressed that the nature of the memories facilitates its reusability and provides a high degree of parallelism. Multiport memories with several access channels enable parallel results to be obtained in the same storage chip. In this way, design decisions for the arithmetic unit can be made: multiple access memories versus several similar single access memories.

B. Proposed implementation

The implementation proposed is concerned with the trade-off between time delay, memory and hardware resource saving.

The main idea consists of taking the operation performed by the function Ψ from a look-up table on every iteration. Due to the fact that α and β remain constant through the whole calculation, the following values of F and G will provide its result in the table. Figure 2 shows the Convolution-LUT which provides the result pursued.

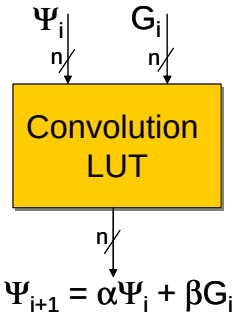

Figure 2: Convolution-LUT calculation

Memory size involves a limitation for the general case. Therefore, an operator fragmentation into blocks handling a set of digits capable of providing a suitable memory addressing.

Considering that the variables involved in the calculation are *n* digit length and that *k* stands for the number of digits of each set, the operators are divided into *t* parts: $t = n/k$. This way, obtaining the partial results for every block from the memory should be required. On every memory access the operator sign is also pointed out.

Convolution-LUT table memory access for each block is shown in Figure 3.

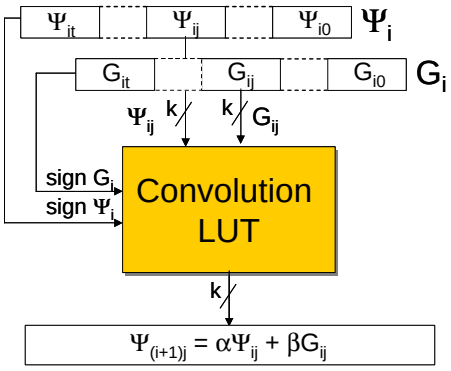

Figure 3: Convolution-LUT partial calculation

It should be pointed out that for *k=1*, the architecture proposed works in a serial way, because the calculation of the function Ψ performs so many LUT access as the number of digits of the numbers. In this particular case, the memory structure is shown in the following table.

TABLE I: CONVOLUTION-LUT STRUCTURE FOR K=1

ΨijGij	Y>0 G>0	Y>0 G<0	Y<0 G>0	Y<0 G<0
00	0	0	0	0
01	β	- β	β	- β
10	α	α	- α	- α
11	α + β	α - β	- α + β	- α - β

Finally, the *t* partial results shifted related to all the blocks considered have to be added in order to provide the following value of Ψ. We assume a complement number coding in order to allow add operations.

The addition sequence calculation establishes several design possibilities:

- Serial scheme for processing each add operands in a single iteration.

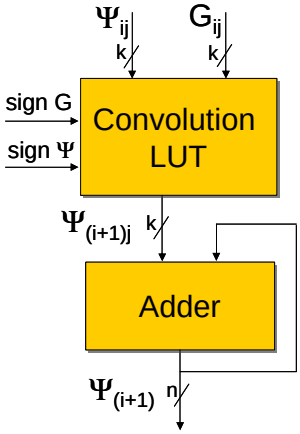

Figure 4: Serie structure add operands

- Add operand reduction scheme into two final ones so that to perform a final addition. This design is similar to that used in a multiplier for performing partial product reduction.

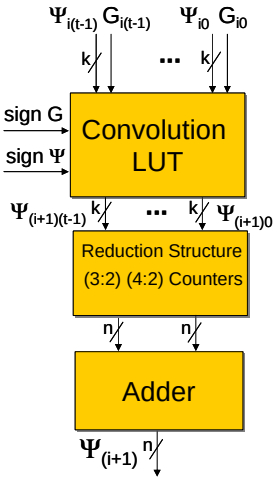

Figure 5: Reduction structure add operands

This approach provides the whole set of partial add operands in parallel in order to feed a reduction tree process. This way, in spite of a hardware complexity growth, a smaller operation delay is achieved.

For the calculation of a new value of Ψ it will be necessary to recode the result to the initial signed codification.

Cycle time

The delay related to every function value calculation depends on the particular implementation developed. According to the schemes analysed previously, these delays could be expressed in the following way, respectively:

- n · (Convolution-LUT + Adder)
- Convolution-LUT + ReductionStructure + Adder

So, the cycle delay depends on the particular implementation.

IV. Evaluation of the Proposed Architecture

In this section, we present estimates of the area costs and execution time of the architecture proposed in the previous section. The power consumption of the circuit is not important for this research and is therefore not dealt with in this paper. It will be studied in depth in the event that it can be implemented in a chip.

A. Area estimations

The model we use for the area estimations is taken from [10] and [27].The unit used is the size of a complex gate τ_a, since the area of the LUTs and other components are easily expressed in this unit.

- *Convolution-LUT*: Data storage imposes severe restrictions on k block size. As we can see in table 1, the area cost increases exponentially with the k value. Therefore, we have to achieve a balance between the memory required and the complexity of the circuit. The model assumes a 40 τ_a/Kbit rate for tables addressed by words of up to 6-bits long, a 35 τ_a/Kbit rate for 7-11 input bit tables, a 30 τ_a/Kbit rate for 12-13 input bit tables and 25 τ_a/Kbit one for 14-15 input bit tables. Following table shows the cost in terms of τ_a and bytes for the most common sizes and wordlength.

TABLE II: LUT MEMORY REQUIREMENTS

n	k	LUT memory requirements Bytes	complex gates
16	1	32 B	6 τ_a
	2	128 B	10 τ_a
	4	2 KB	60 τ_a
	8	512 KB	12800 τ_a
32	1	64 B	8 τ_a
	2	256 B	14 τ_a
	4	4 KB	120 τ_a
	8	1 MB	25600 τ_a

- *Adder*: A n-bit adder requires n τ_a. [27]
- *Reduction structure*: A 3:2 counter uses 2 τ_a, and a 4:2 counter uses 4 τ_a.

As shown in the previous tables, the main contributions to the area of the architecture come from the convolution-LUT. With the previous area cost for each component the total area for design alternatives can be calculated.

B. Delay estimations

Let τ_t be the delay of a complex gate, such as one full-adder. The delay estimations have been performed making the following assumptions:

- *Convolution-LUT*: According to [10], [27] analysis[1] we assume a delay of about $T_{LUT} = 3.5\tau_t$ for 8 input bit tables, $T_{LUT} = 5\tau_t$ for 12-13 input tables and $T_{LUT} = 7.0\tau_t$ for 16-18 input bit tables.
- *Adder*: The delay of the n-bit adder is lgn τ_t. [14].
- *Reduction structure:* This module has a delay of 2 τ_t for 3:2 counter stage and 3 τ_t for 4:2 counter stage. [14]

V. Application Example

In this section the case of rotation evaluation is presented as an example that outlines CBRM capabilities. Here, we are concerned with coordinates evaluation.

Let *P(xi,yi)* be a point of a circumference with radius *R* and q_i the angle with the horizontal axis

$$x_i = R \cos \theta_I$$
$$y_i = R \operatorname{sen} \theta_i \qquad (3)$$

[1] Implementation using a family of standard gates from the AMS 0.35 µm CMOS library

Let's have arbitrary rotation increment *Dq* of the angle in each iteration:

$\forall$ iteration i, $q_i = q_{i-1} + Dq$,

so,

$$\begin{aligned} cos\ q_i &= cos\ (q_{i-1} + Dq) \\ sen\ q_i &= sen\ (q_{i-1} + Dq) \end{aligned} \qquad (4)$$

Setting: $cos\ Dq = a$, $sen\ Dq = b$

Expanding and substituting (4) in (3) and grouping terms:

$$\begin{aligned} x_i &= a\ x_{i-1} + (-b)\ y_{i-1} \\ y_i &= a\ y_{i-1} + b\ x_{i-1} \end{aligned} \qquad (5)$$

A crossed evaluation of the coordinates x_i and y_i can be carried out by CBRM, through the computation of a pair of functions Ψ and G if identifying

$\Psi_i \equiv x_i$ $\qquad$ $G_i \equiv y_i$

so,

$$\begin{aligned} \Psi_{i+1} &= \alpha\ \Psi_i + \beta\ G_i \\ G_{i+1} &= \alpha\ G_i + \beta\ \Psi_i \end{aligned} \qquad (6)$$

With additional restriction $\alpha^2 + \beta^2 = 1$

CBRM provides a mean to calculate the rotation coordinates with fixed iteration path held by parameters *a* , *b*. The number of iterations depends on the whole interval angle to rotate and on the increment *Dq* (i.e. on *a* , *b.)*.

We proposed the following architecture to resolve the expression (6), see figure 7. We assume n=32 bits, k=8, t=4.

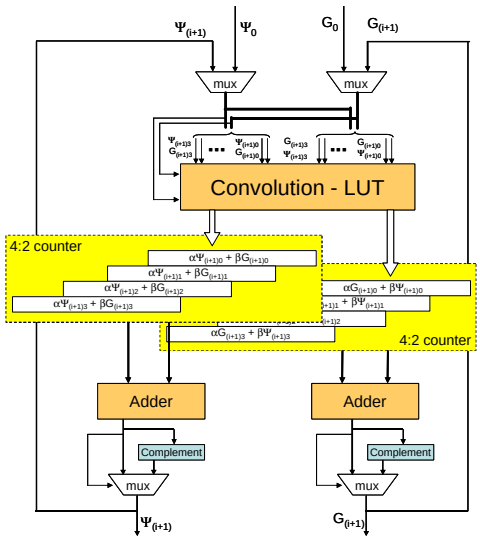

Figure 6: Block diagram of the proponed architecture for n= 32 bits

According to the previous evaluation of the propose architecture, this implementation requires approximately 1 MB of memory. That is ≈ 25600 complex gates of area. Nevertheless, the time delay for the calculation of a new value of the functions Ψ and G is about 15 complex gates.

In order to verify the operation of the new calculation method a series of simulation tests has been made in reconfigurable hardware platform.

Hardware design and simulation techniques and their subsequent implementation in reconfigurable systems such as FPGAs, enable valid designs to be made and a high productivity in development to be obtained. The results of this study are focused on designs implemented on standard FPGAs, specifically on the Xilinx family of FPGAs. These devices have had a wide acceptance and are used by the scientific community as a whole for processing results.

In spite of the fact that the speed of these devices is not very high (up to 50 MHz), by choosing them we aim to establish a homogeneous basis for comparing the addition algorithms and draw conclusions from this comparison. Using one of these devices, the simulation and testing of the proposed design have been carried out with definition and simulation free software *Xilinx WebPack 3.2XE*n[2].

Table III presents the path delay of the CBRM iteration for n = 32 and n= 64 bits of precision.

TABLE III: PATH DELAY OF THE CBRM ITERATION (NS)

n	Convolution LUT	Reduction Structure	Adder	Total
32	6.5	1.7	7.3	15.5
64	6.5	3.5	13.1	23.1

The following table shows to the deviation produced in the calculation for several $\Delta\theta$ and number of values to process. The error that shows the table is referred on the last value.

TABLE III: ABSOLUTE ERROR

	number of calculated points (I)	
	I= 12	I= 36
$\Delta\theta = \pi/4$ rad	$3.52\ 10^{-6}$	$9.41\ 10^{-6}$
$\Delta\theta = \pi/72$ rad	$1.17 \cdot 10^{-5}$	$6.17 \cdot 10^{-5}$
$\Delta\theta = \pi/360$ rad	$5.92 \cdot 10^{-5}$	$9.51 \cdot 10^{-5}$

It is observed that error increases with the amount of points to calculate. On the other hand, also it is observed that whichever minor is the increase of the angle more iterations will be necessary to reach the final value.

VI. COMPARISON WITH OTHERS APPROACHES

In order to obtain conclusions in terms of precision and error the comparison of the CBRM has been performed with CORDIC, which stands for one of the most extensively used algorithms for performing

[2] http://www.xilinx.com

function calculation with an iterative scheme [1][2][3][4].

CORDIC algorithm motivates a great amount of FPGA implementations [12]. Here a rotation mode bit-serial design is considered. This implementation uses three shift-registers, three serial adder-subtractors, and one serial ROM. All this configurates a circuit divided in three parallel subcircuits. The following tables show the results.

TABLE IV: PATH DELAY OF THE CORDIC ITERATION (NS)

n	Total
32	404.928
64	1720.704

TABLE V: ABSOLUTE ERROR CORDIC

	number of calculated points (I)	
	I= 12	**I= 36**
$\Delta\theta = \pi/4$ rad	10^{-6}	10^{-6}
$\Delta\theta = \pi/72$ rad	10^{-6}	10^{-6}
$\Delta\theta = \pi/360$ rad	10^{-6}	10^{-6}

It is observed that the obtained error is similar to the CBRM, whereas time delay is much greater.

VII. CONCLUSIONS

The main purpose of this paper is to present a generic method for function evaluation that achieves good performances in speed and a good trade-off between error and precision. Fundamentals of this method claime that convolution is a means to evaluate functions. Design has been implemented in reconfigurable FPGA based hardware. Comparison with an important alternative approach computation scheme outlines good time performances for CBRM. The basic case of combined trigonometric functions involved in rotation that is analyzed in this paper offers good issues for signal and image processing transforms calculation. Fourier and Hough transforms can follow CBRM calculation scheme [28] with good results. These encouraging partial conclusions make quite reasonable to study in depth the capabilities of convolution to provide a more complete set of recursive evaluating patterns in order to extend the CBRM.

REFERENCES

[1] Volder, J.E. "The CORDIC trigonometric computing technique". IRE Trans. Elect. Comput., vol EC- 8 pp. 330-334. Sept. 1959.

[2] Walther, J.S. "A unified algorithm for elementary functions", Proc. Spring. Joint. Comput. Conf., pp379-385, 1971

[3] Haviland, G.L. and Tuszynski, A.A. "A CORDIC arithmetic processor chip", IEEE Trans. on Computers., vol C-29 nº2 pp. 68-79- 1980.

[4] Nakayama et al. " A 6.7 MFLOPS floatig-point coprocessor with vector/matrix instructions". IEEE Journal on Solid-State Circuits, vol 24 nº 5 pp. 1324-1330 -1989.

[5] Ahmed, H.M. " Directions in DSP Processors", IEEE Journal on Selected Areas i Communications , vol. 8 nº8 pp. 1420-1427- 1990.

[6] de Lange, A.A et al. "Real time applicactions of the floating-point pipeline CORDIC processor in massive-parallel pipelined DSP algorithms". Proc. ICASSP-90 pp. 1013-1016-1990.

[7] Hu, Y.H. "CORDIC-based VLSI architectures for Digital Signal Processing", IEEE Signal Processing Magazine, nº 7 pp. 16-35- July 1992.

[8] Cavallaro, J.R Luk, F.T. "CORDIC arithmetic for SVD processor". Journal of Parallel and Distributed Computing , nº5, pp.271-290, 1988

[9] Cavallaro, J.R Lester, A.C. "CORDIC processor array for the SVD of a complex matrix". SVD and Signal processing , II, Algorithms, Analysis and Applications, R.J. Vaccado (editor), Elsevier Science Publishers, pp. 227-239, 1991

[10] Ercegovac, M. Y lang, T. "Division and Square root: Digit-Recurrence, Algorithms and Implementations", Kluwer Academic Pub., 1994

[11] Villalba, J., "Diseño de Arquitecturas CORDIC multidimensionales" Tesis Doctoral Dept. de Arquitectura de Computadores. Universidad de Málaga . Nov.1995

[12] Andraka, Ray. "A survey of CORDIC algorithms for FPGA". FPGA's 98.Proceedings of the 1998 ACM/SIGDA sixth international symposium of Field Programmable Gate Arrays-22-24 de Febrero de 1998. Monterrey, CA.pp.191-200

[13] Schulte, M.J. and Schwartzlander.E. "Hardware Designs for Exactly Rounded Elementary Functions. IEEE Transactions on Computers, vol 43, nº 8, August 1994.

[14] Wong W,F. Goto,E."Fast Evaluation of Elementary Functions in Single Precision".IEEE Transactions on Computers, vol.C-44, pp.453-457, 1995

[15] Ito, M. Naofumi, T. "Efficient Inicial Approximation for Multiplicative Division and Square Root by a Multiplication with operand Modification". IEEE Transactions on Computers, vol 46, nº4. Abril 1997

[16] Koren,I."Computer Arithmetic Algorithms". Prentice Hall, Englewood Cliffs, NJ, 1993

[17] Cao, Jun. Wei, Belle, W.Y. "High performance of the 13th symp. on Computer Arithmetic (ARITH'97)

[18] Schulte, M.J. and Stine, J.E. "Accurate function approximations by symmetric table look-up and addition". 11th International Conference on application-specific, systems, architecture and processors, 1997

[19] Hassler H. and Takagi, N. "Function Evaluation by Table Look-up and Addition " . Proceedings of the 12th Symposium o Computer Arithmetic, pp.10-16, 1995.

[20] González,R.C; Woods, R.E. "Tratamiento digital de imágenes". Addison-Wesley. Iberamericana. S.A. 1996

[21] Katsuhiko O."Ingeniería de control moderna". Prentice-Hall Hispanoamericana. 1993

[22] Serra, J. "Image Analysis and Mathematical Morphology". Academic Press, London, 1982.

[23] Kittler, J. and Alkoot, F.M.: "Sum versus Vote Fusion in Multiple Classifier Systems," IEE Transactions on Pattern Analysis and Machine Intelligence, vol. 25, nº 1, January 2003.

[24] Weyl, H: Meter, F. "Die Vollstaändigkeit der primitiven Darstellungen einer geschlossen kontinuierlichen Gruppen". Math. Ann. tXCVII. Pp.737-755. (1927)

[25] B. Parhami, "Computer Arithmetic: Algorithms and Hardware Designs", *Oxford University Press*, 2000.

[26] P.T.P.Tang, "Table look-up Algorithms for Elementary Functions and Their Error Analisys"", *Proc. Symp. Computer Arithmetic*, pp. 232-236, 1991.

[27] J-A.. Piñeiro, J.D. Bruguera, J.M. Muller. "Faithful powering computation using table look-up and a fused accumulation tree". *Proceedings of the 15th International Symposium of Computer Arithmetic (ARITH'15)*, 2001.

[28] García Chamizo, J.M.; Signes Pont, M.T.; Mora Mora, H.; de Miguel Casado, G.: "Parametrized Architecture for Hough Transform Recursive Evaluation", Proc. SMMSP 2003, Barcelona, Spain, 2003.