

# FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm

Tanya Vladimirova and Hans Tiggele  
Surrey Space Centre  
University of Surrey, Guildford, Surrey, GU2 5XH  
Tel: +44(0) 1483 879278  
Fax: +44(0) 1483 876021

**Abstract:** The aim of this paper is to investigate CORDIC schemes for fast and silicon area efficient computation of the sine and cosine functions that are suitable for FPGA-based implementation. The results of theoretical investigation into redundant CORDIC is presented. Summary of CORDIC synthesis results based on Actel and XILINX FPGAs is given. Finally applications of CORDIC sine and cosine generators in small satellites are discussed.

**Keywords:** CORDIC algorithm, FPGA implementation, redundant signed-digit system, synthesis, sine, cosine

## 1. Introduction

The name CORDIC stands for Coordinate Rotation Digital Computer. The underlying method of computing the rotation of a vector in a Cartesian coordinate system and evaluating the length and angle of a vector was developed by Volder [Vold59]. The CORDIC method was later expanded for multiplication, division, logarithm, exponential and hyperbolic functions. The various function computations were summarised into a unified technique in [Walt71].

The resulting vector  $[x_n, y_n]^T$  of the rotation of a vector  $[x_0, y_0]^T$  by an angle  $\theta$  in Cartesian coordinates can be computed by the following matrix operation [Pirs98]:

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (1)$$

Using the identity

$$\cos \theta = \frac{1}{\sqrt{1 + \tan^2 \theta}} \quad (2)$$

and factoring out  $\cos \theta$  equation (1) can be modified to

$$\begin{bmatrix} x_n \\ y_n \end{bmatrix} = \frac{1}{\sqrt{1 + \tan^2 \theta}} \begin{bmatrix} 1 & -\tan \theta \\ \tan \theta & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \quad (3)$$

In the CORDIC method, the rotation by an angle  $\theta$  is implemented as several micro rotations by a given step angle  $\alpha_i$ . Any angle  $\theta$  can be represented to a certain accuracy by a set of  $n$  step angles  $\alpha_i$ . Specifying a direction of rotation or sign  $\sigma_i$ , the sum of the step angles  $\alpha_i$  approximates a given angle  $\theta$  as follows:

$$\theta = \sum_{i=0}^{n-1} \sigma_i \alpha_i, \quad \sigma_i \in \{-1, 1\} \quad (4)$$

The sign of the difference between the angle  $\theta$  and the partial sum of step angles  $\theta - \sum_{j=0}^{i-1} \sigma_j \alpha_j$

controls the sign of the step angles  $\sigma_i$ . To simplify the computation of the matrix product (3), the step angles  $\alpha_i$  are chosen such that  $\tan \alpha_i$  represents a series of powers of 2:

$$\tan \alpha_i = 2^{-i}, \quad i = 0, 1, \dots, n-1 \quad (5)$$

An auxiliary variable  $z_i$  is introduced that contains the accumulated partial sum of step angles and can be used to control the sign of the step angles.

The CORDIC method can be employed in two different modes, known as the “rotation” mode and the “vectoring” mode. In the rotation mode, the co-ordinate components of a vector and an angle of rotation are given and the co-ordinate components of the original vector, after rotation through a given angle, are computed. In the vectoring mode, the co-ordinate components of a vector are given and the magnitude and angular argument of the original vector are computed [Vold59].

**Rotation mode:**

Inputs:  $x_0, y_0$ , angle  $z_0$

Iteration equations:

$$\begin{aligned} x_{i+1} &= x_i - y_i \sigma_i 2^{-i} \\ y_{i+1} &= y_i + x_i \sigma_i 2^{-i} \\ z_{i+1} &= z_i - \sigma_i \arctan 2^{-i} \end{aligned} \tag{6}$$

where  $i=0,1,2,\dots,n-1$ ,  $\sigma_i = \begin{cases} -1 & \text{if } z_i < 0 \\ +1 & \text{if } z_i \geq 0 \end{cases}$  (7)

Outputs:

$$\begin{aligned} x_n &= K_n (x_0 \cos z_0 - y_0 \sin z_0) \\ y_n &= K_n (y_0 \cos z_0 + x_0 \sin z_0) \\ z_n &= 0 \end{aligned}$$

where  $K_n = \prod_{i=0}^{n-1} \sqrt{1+2^{-2i}}$  (8)

is a scale factor, that represents the increase in magnitude of the vector during the rotation process since the rotation is not a pure rotation but a rotation-extension. When the number of iterations/micro-rotations is fixed the scale factor is a constant approaching the value of 1.647 as the number of iterations goes to infinity.

**Sine and Cosine Computation using the CORDIC Method**

The rotation mode of the CORDIC algorithm could be used to compute sine and cosine of an angle  $\theta$ . The computation of  $\sin \theta$  and  $\cos \theta$  is based on the rotation of an initial vector of unit length, that is aligned with the abscissa ( $x_0 = 1, y_0 = 0$ ).

Input values for  $n$  iterations:  $x_0 = 1, y_0 = 0, z_0 = \theta$

Outputs after  $n$  micro-rotations:

$$\begin{aligned} x_n &= K_n (x_0 \cos \theta - y_0 \sin \theta) = K_n \cos \theta \\ y_n &= K_n (y_0 \cos \theta + x_0 \sin \theta) = K_n \sin \theta \\ z_n &= 0 \end{aligned} \tag{9}$$

The magnitude of the initial vector increases by a factor  $K_n$  during the micro-rotations that constitute the rotation mode and an operation of division is required at the end of the rotation process in order to obtain the value of  $\sin \theta$  and  $\cos \theta$ . One simple way to avoid the operation of division is to compensate the scale factor by setting the initial value  $x_0 = 1/K_n$ , since the scale factor is a constant for a given number of iterations  $n$ .

In this paper we consider computation of sine and cosine of an angle  $\theta$  (rad), where  $\theta$  is an n-bit signed binary fraction and satisfies  $-\pi/2 \leq \theta \leq \pi/2$ . We compute  $\sin \theta$  and  $\cos \theta$  down to the  $n$ th binary position.

## 2. CORDIC Implementations

The CORDIC algorithm can be implemented as a sequential structure (unfolded in time), as a parallel structure (unfolded in space) or as a combination of the two (Figure 1). The sequential implementation, or also called “iterative” is based on three n-bit adders/subtractors and sign extending shifters, a look-up table (LUT) for the step angle constants, a finite state machine and assumes that one iteration per clock cycle is performed. The parallel implementation, or also called “cascaded”, is similar to an array multiplier structure, consisting of rows of adders/subtractors, with hardwired shifts and constants and can be implemented as a combinational logic for small designs or can be pipelined. The combined implementation, or also called “cascaded fusion”, is based on a sequential structure where the logic for several successive iterations is cascaded and is executed within one clock cycle [Wang95].

In sequential implementations bit-serial and binary adders have been used [Andr98], in cascaded designs the types of adders that have been used varies more widely – bit-serial adders, carry-save adders, binary adders, redundant adders, both binary and redundant adders [Andr98, Timm92]. The serial implementation with bit-serial adders yields the smallest area and lowest speed, the array structures with redundant adders yield fastest execution and largest area. A trade-off between area and speed would determine the right implementation approach for a given application.

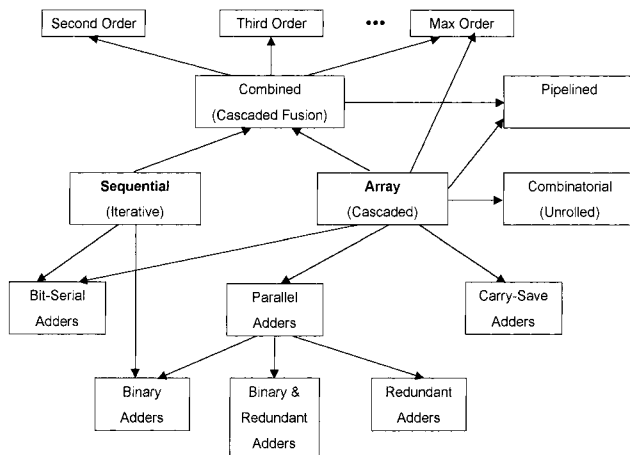


Figure 1. CORDIC Implementations

## 3. Summary of Fast CORDIC Methods and Structures

The parallel implementation is aimed at very fast designs and depends very much on the type of adders that are used. Adders based on the conventional two-digit binary system have time delay dependent on the bit length  $n$  and in the best case of fast hierarchical adder structures (as carry lookahead adders) the time delay for execution of one iteration is of

logarithmic order  $O(\log_2 n)$  [Pirs98]. The operation of addition can be made independent on the bit length by representing the operands in redundant signed digit (RSD) binary system, where  $0$  and  $1$ ,  $-1$  are used as binary digits. This system is called redundant because it allows several representations for a particular numerical value.

The introduction of the RSD system into the internal computation of the CORDIC method is considered one of the most effective ways to accelerate the algorithm [Erce87, Taka91, Timm92, Bake76]. It has exceeded the speed of CORDIC array implementations based on carry-save adders according to a comparative study of these methods in [Timm92]. In redundant CORDIC schemes for computation of sine and cosine  $x_i$ 's,  $y_i$ 's and  $z_i$ 's are represented by a redundant representation,  $\sigma_i$  is selected from  $\{\bar{1}, 0, 1\}$  by evaluating a few most significant digits of  $z_i$ . The application of RSD system to CORDIC gives rise to three problems, that compromise the efficiency as summarised below:

- The evaluation of the rotation operators  $\sigma_i$  in the redundant CORDIC algorithm is slow due to the fact that the evaluation of the sign of a redundant number requires detection of the sign of the most significant non-zero binary digit and needs inspection of all digits in the worst case:  
 Detection of the sign of a RSD number  $x$  by the most significant digit  $\text{MSD}(x)$   
 If  $\text{MSD}(x) = 1$ , then  $x > 0$   
 If  $\text{MSD}(x) = -1$ , then  $x < 0$   
 IF  $\text{MSD}(X) = 0$ , then  $x = 0$  or  $x > 0$ , or  $x < 0$

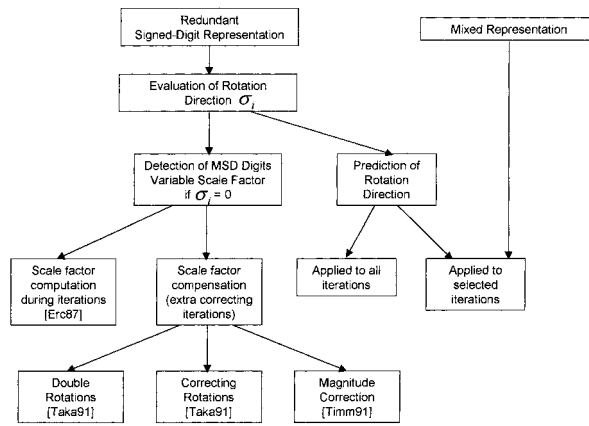
Another way to evaluate the rotation direction is to predict the values of  $\sigma_i$  by decomposing the angle in advance.

- $\sigma_i$  is allowed to take values from (1, 0, -1) and 0 is a valid choice. If  $\sigma_i = 0$  no rotation-extension takes place for some step angles, and the scale factor  $K_n$  becomes a variable dependent on the particular operand value. There are two ways to tackle this: either the scale factor is calculated during computation and the function values are corrected with it at the end of the rotation process [Erce87] or the scale factor is compensated during the iteration process via introduction of special iterations [Taka91, Timm92].
- Also converters from 2's complement representation to RSD and vice versa are required. The conversion from 2's complement to RSD is straightforward, however the conversion from RSD to 2's complement requires an extra addition operation over n-bit.

Considerable design effort has been dedicated to modifying the CORDIC algorithm with the aim to overcome the above problems (Figure 2).

Figure 2. Fast CORDIC schemes

We have considered four redundant CORDIC algorithms and have estimated their latency times according to the equations in Table 1 [Marx99]. The termination algorithm originally proposed by [Chen72] allows quitting the iteration process as early as possible, modified Booth encoding can be used for the same purpose [Timm92]. The following notation has been used in Table 2:  $\tau$  - delay of a full adder;  $\tau(\log_2 n)$  - the upper



bound of an n-bit non-redundant fast addition;  $\delta$  - delay of a redundant adder, independent of the bit-length;  $m$  - an arbitrary integer in the correcting method [Taka91] where a correction iteration is performed every  $m$ -th step.

Table 1

Name	Latency expression as a function of the bit length n
Non-redundant method	$n \cdot \tau \log_2 n$
Double rotation method [Taka91]	$n\tau + 2n\delta + \tau \log_2 n$
Correcting method [Taka91]	$(n - \lfloor (n+1)/m \rfloor)(\tau + \delta) + 2(\lfloor (n+1)/m \rfloor + \log_2 n)(\tau + \delta)$
Prediction method [Timm92]	$n\delta + \tau \lfloor \log_3 n - 1 \rfloor \log_2 n + \tau \log_2 n$
Prediction with termination method [Timm92]	$\delta(n+1)/2 + \tau \lfloor \log_3 ((n+1)/2 - 1) \rfloor \log_2 n + \tau \log_2 n + \delta \log(n/2)$

Figure 3 shows graphically the latency of the CORDIC implementations based on the expressions in Table 1 and estimated delays for XC4000XL using ratio  $r \equiv \delta/\tau = 2$ . It suggests that the prediction with termination method [Timm92] might lead to the fastest FPGA implementation.

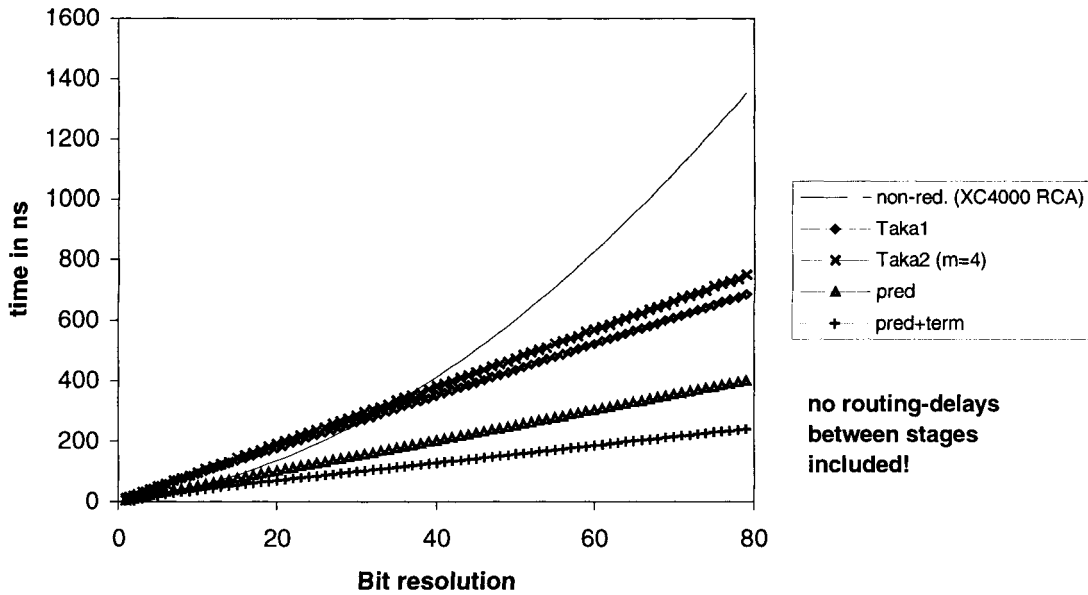


Figure 3. Estimated latency of CORDIC implementations in XC4000XL

#### 4. Redundant Adder Implementation in XILINX XC4000 Family

In RSD representation, number  $Y$  can be viewed as the difference between two positive binary numbers  $Y^*$  and  $Y^{**}$ . We have:

$$Y = \sum_{i=0}^n y_i \cdot 2^i = \sum_{i=0}^n (y_i^* - y_i^{**}) \cdot 2^i \text{ with } y_i^*, y_i^{**} \in \{1, 0\} \quad (12)$$

The conventional one-bit full adder assumes positive weights to all of its binary inputs and two outputs. Such adders can be generalised to four types of adder cells by imposing positive and negative weights to the binary input/output terminals [Hwan79]. Figure 4 shows the names and logic symbols of four types of generalised full adders. Each type of adder is named by the number of negatively weighted inputs contained in it.

Logic symbol				
Type	0	1	2	3
Function	$x+y+z = 2.c+s$	$-x+y+z = 2.c-s$	$-x-y+z = -2.c+s$	$-x-y-z = -2.c-s$

Figure 4 Generalised Full Adders [Vand90]

The addition of two signed digit (SD) numbers  $Y$  and  $Z$  can be performed by cascading two levels of generalised full adders of types 1 and 2. The logic circuit implementing the chosen function is sketched in Figure 5. The main drawback of this computation scheme with two numbers in redundant form is the amount of hardware, which is twice that in the carry-save case [Vand90].



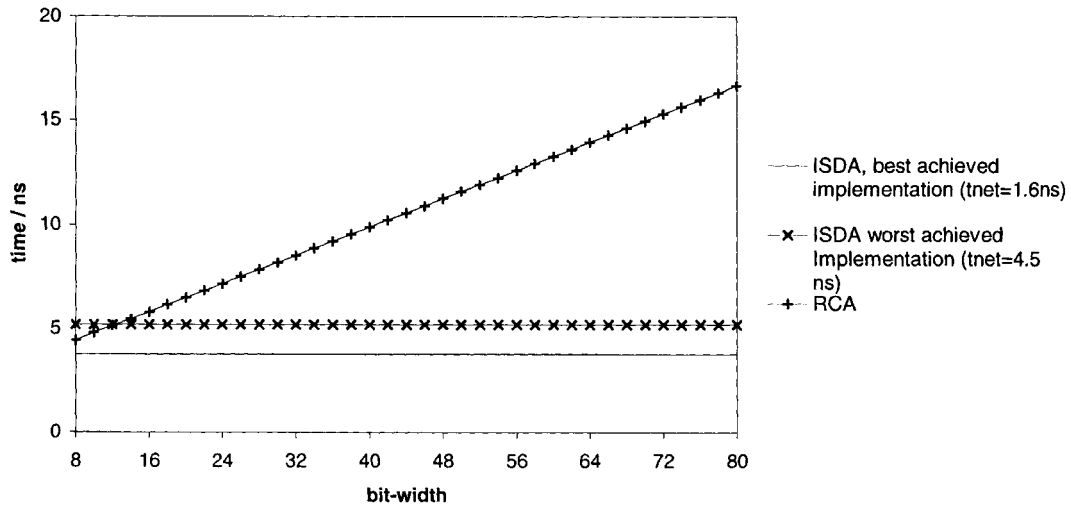


Figure 7

An iterative 12-bit to 32-bit CORDIC sine/cosine generator has been implemented in Actel using Synplify/Actmap and Leonardo Spectrum. Back annotated delay shows maximum speed of 2.7 MHz for 12 bits reduced to 400 KHz for 32-bit..

Summary of experimental results is shown in Table 3 and Table 4: Table 5 gives LUT synthesis results for comparison purposes.

Table 3. Summary of CORDIC synthesis results based on ACTEL FPGAs

Designs	Length	Actmap <sup>1</sup> 3.5.04	Synplify <sup>1</sup> 5.1.4	Spectrum <sup>1</sup> 5.6.9	Speed <sup>2</sup>	Frequency A54SX16-2
	bits	Area/Delay <sup>4</sup>	Area/Delay <sup>4</sup>	Area/Delay <sup>4</sup>	ns	MHz (Fclk) <sup>5</sup>
Iterative	12	420/574	307/334	347/424	169.5	5.9 (71.4) <sup>5</sup>
Iterative	14	538/784	399/414	428/536	192.3	5.2 (72.5) <sup>5</sup>
Iterative	16	674/958	424/462	501/633	232.5	4.3 (68.5) <sup>5</sup>
Iterative	24	1170/----	694/727	995/1248	357.2	2.8 (66.6) <sup>6</sup>
Iterative	32	1963/----	887/1000	1419/1710	526.3	1.9 (62.5) <sup>6</sup>
Cascaded	12	----/----	862/888	1326/1378	44.8	22.3 <sup>6</sup>
Cascaded	14	----/----	1970/----	2164/2164	192.3	5.2 <sup>3</sup>
Cascaded	16	----/----	2853/----	2941/3718	222.2	4.5 <sup>3</sup>

Table 4. Summary of CORDIC synthesis results based on XILINX FPGAs

Design	Length	Foundation <sup>7</sup> Express 1.5	Target Device	Speed <sup>2</sup>	Frequency
	bits	Area/Delay		ns	MHz (Fclk)
Iterative	12	106/139	XC4010XL-09	370.3	2.7 (32.1)
Iterative	14	133/145	XC4010XL-09	526.3	1.9 (27.5)
Iterative	16	162/178	XC4010XL-09	588.2	1.7 (27.2)
Iterative	24	317/376	XC4062XL-09	1643.8	0.6 (14.6)
Iterative	32	506/626	XC4062XL-09	2480.6	0.4 (12.9)
Cascaded	12	210/210	XC4010XL-09	187.6	5.3
Cascaded	14	288/288	XC4010XL-09	192.9	5.2
Cascaded	16	378/378	XC4062XL-09	330	3.1

Table 5. LUT synthesis results

Design	Length	Actmap <sup>1</sup>	Synplify <sup>1</sup>	Spectrum <sup>1</sup>	Speed <sup>2</sup>	Frequency
A54SX16-3		3.5.04	5.1.4	5.69		
	bits	Area/Delay <sup>4</sup>	Area/Delay <sup>4</sup>	Area/Delay <sup>4</sup>	ns	MHz (Fclk)
LUT	12	513/859	384/453		43.66	22.1
LUT	16	1899/---- <sup>8</sup>	946/---- <sup>8</sup>		84.03	11.9

Note 1 All synthesis tools operated in a "push-button" fashion with maximum optimisation enabled were available.

Note 2 Speed estimate based on Vital simulation using typical operating conditions.

Note 3 Estimate frequency given by Synplify

Note 4 All module count given by Designer Place and Route software.

Note 5 Actel Netlist Selected

Note 6 Synplify Netlist Selected

Note 7 Foundation Express build 3.1.140

Note 8 ---- Synthesis not performed

## 6. Application

### Legendre polynomials - the first step of the IGRF model

The Earth's Magnetic Field computation as part of satellite attitude determination system is a very computationally intensive procedure and is mainly done in software. A structure based on CORDIC modules has been proposed [Vlac99] for the calculation of Legendre polynomials - the first step of the International Geomagnetic Reference Field (IGRF) model [Wert85]. It consists of four blocks comprising CORDIC modules for sine/cosine as well as other functions and a control block. The estimated delay based on a 32-bit iterative CORDIC module implemented in XC4085XL was compared with the delay of the software implementation running on a Pentium 333 MHz computer for five different values of the constants  $m, l$ . The improvement in speed was 44% for  $m = l = 10$ , 37% for  $m = l = 15$ , 32% for  $m = l = 20$ , 28% for  $m = l = 25$  and 23% for  $m = l = 36$ .

### Direct Digital Synthesis

Suitable for more than 14 bits, no need of in-phase / Quadrature components

## 7. Conclusions

The analysis of the adder delay have shown that starting from 32 bits the redundant SD adder would have a smaller delay than an RC adder implemented using the XILINX fast carry chain, however it will require a CLB consumption equal to four times that of the RC adder.

## 8. References

[Andr98] R.Andraka. A Survey of CORDIC Algorithms for FPGA Based Computers – Proc. Of the 1998 CM/SIGDA Sixth International Symposium on FPGAs, 11bruary,1998, Monterey, CA, pp.191-200.

[Bake76] P.W.Baker. Suggestion for a Binary Cosine Generator, IEEE Transactions on Comput., February, 1975, pp. 1134-1136.

[Chen72] T.C.Chen. Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots, IBM J. Res.Development, July, 19972, pp.380-388.

- [Erce87] M.D.Ercegovac, T.Lang. Fast Cosine/Sine Implementation Using CORDIC Iterations, IEEE Trans. On Comput., vol.40, n 9, 1987, pp. 222-226
- [Marx99] M.Marx. FPGA Implementation of  $\sin(x)$  and  $\cos(x)$  Generators Using the CORDIC Algorithm, Final Year Project Report, School of Electronic Engineering, University of Surrey, Guildford, UK, 1999.
- [Pirs98] P.Pirsch.Architectures for Digital Signal Processing, John Wiley & Sons, 1998.
- [Taka91] N.Takagi. Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation, IEEE Trans. On Comput., vol. 40, n 9, 1991, pp. 989-994.
- [Timm92] D.Timmerman, H.Hahn, B.J.Hosticka. Low Latency Time CORDIC Algorithms, IEEE Transactions on Comput., vol.41, n 8, 1992, pp.1010-1014.
- [Timm91] D.Timmerman, H.Hahn, B.J.Hosticka, B.Rix. A New Addition Scheme and Fast Scaling Factor Compensation Methods for CORDIC algorithms, Integration – the VLSI Journal, vol. 11, n 1, 1991, pp. 85-100.
- [Vand90] A. Vandemeulebroecke, E.Vanzieledhem, et al. A New Carry-Free Division Algorithm and its Application to a Single Chip 1024-b RSA Processor”, IEEE Journal of Solid-State Circuits, vol.25, n 3, 1990, pp.748-755.
- [Vank96] J.Vankka. Methods of Mapping from Phase to Sine Amplitude in Direct Digital Synthesis, Proc of the 1996 IEEE International Frequency Control Symposium, 1996, pp. 942 –950.
- [Vlac99] A. Vlachos. Design and Implementation of CORDIC Modules for ADCS, MSc Project Report, School of Electronic Engineering, University of Surrey, Guildford, UK, 1999.
- [Vold59] J.Volder. The CORDIC Computing Technique, IRE Trans. Comput., Sept. 1959, pp.330-334.
- [Walt71] J.S. Walther. A Unified Algorithm for Elementary Functions, Proc. AFIPS Spring Joint Computer Conference, pp.379-385, 1971.
- [Wang96] A Unified View of CORDIC Processor Design, in Application Specific Processors, Ed. By Earl E. Swatzlander, Jr., Kluwer Academic Press, 1996, pp.121-160.
- [Wert85] J. Wertz. Spacecraft Attitude Determination and Control, D.Ridel Publishing Company, London, 1985.