A Fast CORDIC Co-Processor Architecture for Digital Signal Processing Applications

Javier O. Giacomantone, Horacio Villagarcía Wanza, Oscar N. Bria
CeTAD[H] – Fac. de Ingeniería – UNLP
hvw@info.unlp.edu.ar

Abstract

The coordinate rotational digital computer (CORDIC) is an arithmetic algorithm, which has been used for arithmetic units in the fast computing of elementary functions and for special purpose hardware in programmable logic devices. This paper describes a classification method that can be used for the possible applications of the algorithm and the architecture that is required for fast hardware computing of the algorithm.

Keywords:

Computer Architectures, CORDIC, Computer Arithmetic, Hardware Algorithms, Digital Signal Processing Applications.

[H] Centro de Técnicas Analógico-Digitales. Director: Ing. Antonio A. Quijano.

A Fast CORDIC Co-Processor Architecture for Digital Signal Processing Applications

Javier O. Giacomantone, Horacio Villagarcía Wanza, Oscar N. Bria
CeTAD[H] – Fac. de Ingeniería – UNLP
hvw@info.unlp.edu.ar

Abstract

The coordinate rotational digital computer (CORDIC) is an arithmetic algorithm, which has been used for arithmetic units in the fast computing of elementary functions and for special purpose hardware in programmable logic devices. This paper describes a classification method that can be used for the possible applications of the algorithm and the architecture that is required for fast hardware computing of the algorithm.

Keywords: Computer Architectures, CORDIC, Computer Arithmetic, Hardware Algorithms, Digital Signal Processing Applications.

I. Introduction

The Coordinate Rotation Digital Computer (CORDIC) is an arithmetic technique, which makes it possible to perform two dimensional rotations using simple hardware components. The algorithm can be used to evaluate elementary functions such as cosine, sine, arctangent, sinh, cosh, tanh, ln and exp. CORDIC algorithm appears in many applications because it uses only primitive operations like shifts and additions to implement more complex functions. This is why the development of special purpose hardware architectures, sought to be considered.

This paper is organised as follows: Section II reviews the CORDIC algorithm. Section III presents a classification of the possible applications of the (2D) algorithm. The fast architecture solution, which is geared towards programmable devices, is given in section IV.

II. The CORDIC algorithm

The algorithm was introduced by J. Volder [Vol59] as a special purpose digital computer for real time navigation problems.

Similar algorithms were presented by J. Meggit [Meg62] and Linhart and Miller [Mill69] but it was Walther [Wal71] who formally introduced this theory, in 1971. This was the generalised algorithm that allowed the computation of many elementary functions including multiplication, division, sine, cosine, arctan, sinh, cosh, tanh, arctanh, ln, exp, and square root.

[H] Centro de Técnicas Analógico-Digitales. Director: Ing. Antonio A. Quijano

All the evaluation procedures in CORDIC are computed as a rotation of a vector in three different coordinate systems with an iterative unified formulation.

The rotation angle θ is approximated by the sum of $\alpha_{m,i}$ which are the partial step angles.

$$\theta = \sum_{i=0}^{n-1} c_i \alpha_{m,i} \qquad (1)$$
$$c_i \in \{-1,1\}$$

where $\alpha_{m,i}$ is defined by:

$$\alpha_{m,i} = \frac{1}{\sqrt{m}} \tan^{-1}\left[\sqrt{m}\ 2^{-S(m,i)}\right] \qquad (2)$$

The coordinate system of operation is determined by m as follows

$m=1$	circular coordinate system
$m=-1$	hyperbolic coordinate system
$m \to 0$	linear coordinate system

replacing in (2)

$$m \to 0 \quad \Rightarrow \quad \alpha_{0,i} = 2^{-S(0,i)}$$
$$m = 1 \quad \Rightarrow \quad \alpha_{1,i} = \tan^{-1} 2^{-S(1,i)}$$
$$m = -1 \quad \Rightarrow \quad \alpha_{-1,i} = \tanh^{-1} 2^{-S(-1,i)}$$

$S(m,i)$ is an integer shift sequence satisfying:

$$S(m,i) \le S(m,i+1) \le S(m,i)+1$$

The unified CORDIC algorithm can be written as:

1) Read $x(0), y(0), z(0)$

2) For i = 0 to n-1 the CORDIC iteration equation gives the new coordinates after each pseudorotation

$$\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -mc_i\, 2^{-S(m,i)} \\ c_i\, 2^{-S(m,i)} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix} \qquad (3)$$

3) The angle must be updated, giving:

$$z(i+1) = z(i) - c_i \alpha_{m,i} \qquad (4)$$

Actually equation (3) is called a pseudorotation because the m-norm of the vector $[x\ y]^T$, which is defined as $\sqrt{x^2+y^2 m}$, is not the same after each step.

In order to maintain the m-norm constant a scaling operation is needed.

$$\begin{bmatrix} x'(n) \\ y'(n) \end{bmatrix} = \frac{1}{K_m(n)} \begin{bmatrix} x(n) \\ y(n) \end{bmatrix} = \frac{1}{\prod_{i=0}^{n-1} \sqrt{1+mc_i^2 2^{-2S(m,i)}}} \begin{bmatrix} x(n) \\ y(n) \end{bmatrix} \qquad (5)$$

Computing Modes

In CORDIC there are two fundamental modes of operation defined by Volder [Vol59] as rotation and vectoring modes. When the algorithm works in the rotation mode, the data that is used are the coordinate components of a vector and the desired angle of rotation, after n iterations the algorithm approximates the final coordinates of the rotated vector.

In the vectoring mode the coordinates components after rotation are given and the algorithm calculates the angle of rotation.

The set of $\{c_i, i = 1\ to\ n-1\}$ determine the direction of rotation of each step, which is known as the forward rotation mode [Hu88]:

$$z(0) = \theta$$
$$z(0) - z(n) = \theta - z(n) = \sum_{i=0}^{n-1} c_i \alpha_{m,i}$$

To satisfy the above equation: $c_i = sign\ of\ z(i)$.

In the vectoring mode or backward rotation mode [Hu90]:

$$z(0) = 0$$
$$c_i = -sign\ of\ x(i)y(i)$$

Shift sequence and convergence

The shift sequence determines the convergence and the scaling factor. The selection of the adequate shift sequence must be done to fulfil the following error criteria:

The angle approximation error due to the finite set of angles $\{\alpha_{m,i}, i=0, n-1\}$ is :

$$\varepsilon = \theta - \sum_{i=0}^{n-1} c_i\ \alpha_{m,i}$$

It is desired that for any given rotation

$$|\varepsilon| \le \alpha_{m,n-1}$$

In order to satisfy the above condition a partial angle in iteration i must be compensated except for the error by all the following partial angles

$$\alpha_{m,i} \le \sum_{j=i+1}^{n-1} \alpha_{m,j} + \alpha_{m,n-1}$$

According to the above criteria the maximum angle is:

$$|\theta| \le \sum_{i=0}^{n-1} \alpha_{m,i} + \alpha_{m,n-1} \equiv \max.angle$$

The accuracy, approximation error and rounding error have been analysed [Wal71] [Hu92] [Cav93].

The following values for the shift sequence were presented by Walther [Wal71], satisfying the convergence criteria

m	S(m,i)	Max. angle
1	0,1,2,3,4,5,….,i,…	1.743287
0	1,2,3,4,5,6,…,i+1,..	1.000000
-1	1,2,3,4,4,5,…,12,13,13,14..	1.118173

III. Applications of the algorithm

The CORDIC algorithm can perform elementary functions efficiently as an alternative method to using tables or polynomial approximation but the real power of the technique resides on the fact that provides solution to a broad class of problems with the same iterative algorithm. It is the aim of this section to classify these applications.

The applications of the algorithm can be considered in three main groups:

- The first group includes only the simple original task, but it is very important as it allows the vector to be rotated in a plane and the new coordinates of the vector or rotated angle to be determined. This first group also includes the first direct applications like conversions between decimal and binary systems and polar into Cartesian coordinates.

- The second part of the classification involves the elementary functions. These are determined by the proper setting of the initial values of the algorithm or by the double application of it. The arithmetic operations and functions that we can generate with the appropriate use of CORDIC are multiplication, division, sine, cosine, tan, arctan, sinh, cosh, tanh, arctanh, ln, exp, and square root.

- The third group is definitely the one that we can call general application for image processing, pattern recognition and digital signal processing in general. For the sake of simplicity this group is presented as three subgroups:

    - Transformation: Cosine Discrete Transform, Discrete Fourier Transform (FFT), Chirp Z Transform, Hartley Transform, and Hough Transform

    - Digital Filters: Orthogonal Digital Filters and Adaptive Lattice Filters

    - Matrix Algorithms: QR factorization (Kalman Filters, Linear System solvers), Toeplitz and covariance system solvers, least square deconvolution and eigenvalue, and SVD with application to array processing.

IV. CORDIC Architectures

There are multiple hardware structures that can be used to implement a CORDIC processor. The interaction between the three most important structures, iterative, serial iterative and unrolled (parallel implementation), and the basic arithmetic circuits to implement them have already been classified [Vlad99].

The primary target application of the architectural design presented in this paper is a CORDIC co-processor for high throughput digital signal processing tasks, where the CORDIC co-processor is intend to be implemented in a programmable device, interacting with a digital signal processor.

A design for a programmable device can impose severe restrictions. It is important to realise that consideration must be given for the trade-off factors associated with the area consumption, accuracy and throughput must be considered.

A survey of algorithms for FPGA have been carried out [Andra98], where the principal features of each structures have been presented.

The following section presents a particular design based on the unrolled CORDIC processor [Andra98].

A Fast CORDIC Design

The following CORDIC module deals with the performance of the algorithm in both modes of operation and in the three coordinate systems.

A simple unit is used called CORDIC element (Fig.1). The main components are three adders/subtractors, two shift registers and the necessary combinatorial hardware to control the adders and select the operation mode.

The C.E. element is the kernel of the CORDIC processor and it’s primary function is to perform eq.(3) and eq.(4).

It is clear from eq.(3) that it can be implemented basically with adders and appropriate shifters but without using multipliers.

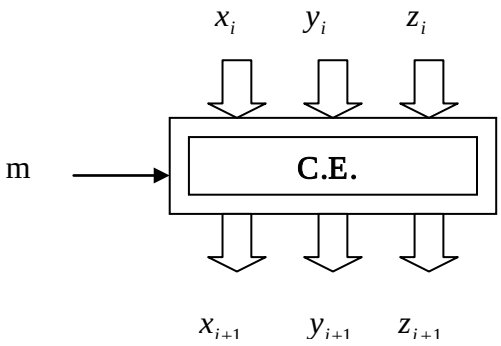

Fig.1

The basic operation that the CORDIC element performs is not more than a crossaddition [Vol59], to sum or substruct a shifted value of x_i to y_i to obtain y_{i+1} or a shifted value of y_i to x_i to obtain x_{i+1} (Fig.2).

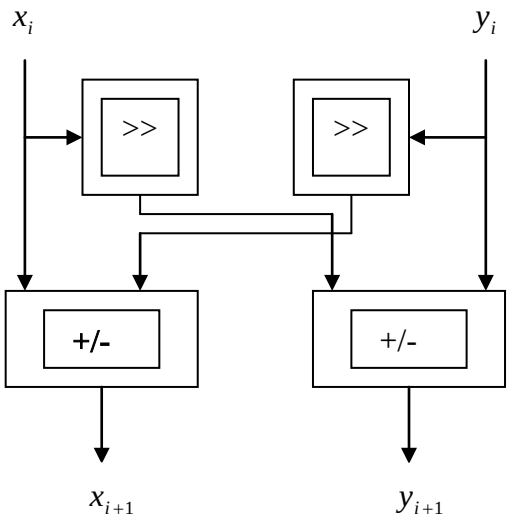

Fig.2

A parallel pipelined structure is used to implement the system consisting in an array of C.E., each of them performing a computation in parallel with the other and separated by registers to form a pipeline structure (Fig.3).

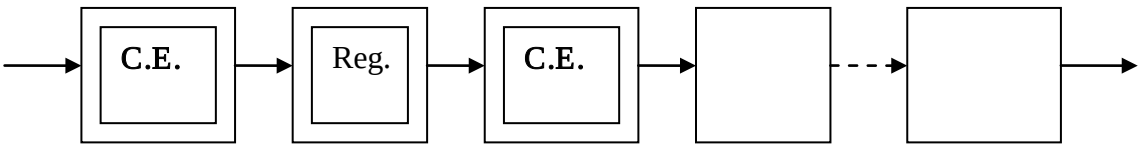

Fig.3

Any CORDIC processor should contain a proper module to perform four basic functions:

1) The basic iterations of eq.(3)
2) The scaling of the vector module
3) The angle update iteration
4) The storage of the arc tangent radix constants ATR

Fig.4 shows the whole design which basically consist of four modules, the pseudorotation (P) and the scaling factor (S) blocks are very fast parallel, pipeline structures, of which the first implements the basic iterations and the angle updating, while the second fits the vector to the correct trajectory.

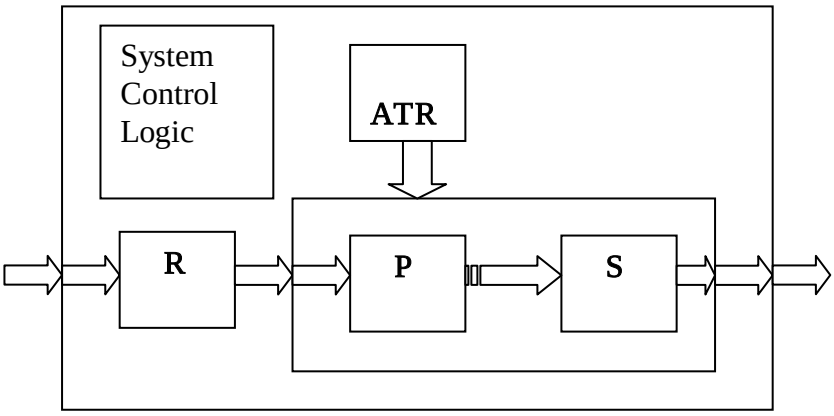

Fig.4

The ATR block only contains the angle steps for each iteration.The rotation module (R) performs a real rotation of 90 or –90 degrees, in order to extend the angle range.

The design behaviour was first simulated in C/C++, then described in VHDL and in then tested in the adequate VHDL testbench.

V. Conclusions

The pipeline CORDIC architecture design, presented is appropriate for solving trigonometric relations at high speed. It can be used in real time applications, but the fundamental advantage is the possibility of generating different elementary functions, with the same unit.

The algorithm needs adders and multipliers but using specific sets of ATR constants the hardware implementation is simplified (eg. shifters as multipliers). The separate distribution of the ATR constants to each adder permits a hardwire solution instead of using a specific ROM.

The shifters need not to be programmable, which is a beneficial asset. Nevertheless, a trade-off

between area and accuracy, needs to be made, so that the utility of the design incorporates the considerations required of accuracy bits needed for each arithmetic task and the hardware resources available.

Acknowledgement

The authors are grateful to Griselda Lyn for providing comments and suggestions that greatly improved this paper.

References

[Vol59] Volder, J., "The CORDIC Trigonometric Computing Technique", IRE Trans. Electronic Computing, Vol. EC-8, Sept 1959, pp. 330-334.

[Wal71] Walther, J. S., "A unified algorithm for elementary functions", Spring Joint Computer Conf., 1971, Proc., pp. 379-385.

[Hu88] Hu, Y. H. and Sung, T. Y. "Efficient Implementation of the Chirp Z-Transform using a CORDIC processor", IEEE Trans. ASSP, Vol. 38, No. 2, Feb. 1990, pp. 352-354.

[Hu92] Hu, Y. H., "The quantization effects of the CORDIC algorithm", IEEE Trans. on Signal Processing, Vol. 40, No. 4, April 1992, pp. 834-844.

[Mill69] Linhardt, R. J. and Miller, H. S., "Digit-by-Digit Transcendental-Function Computation", RCA, Rev. 30 (1969), pp. 209-247.

[Andra98] Andraka, R. "A survey of CORDIC Algorithms for FPGA Based Computers", Proc. of ACM/SIGDA Sixth International Symposium on FPGAs, Feb. 1998, Monterrey, CA, pp. 191-200.

[Vlad99] Vladimirova, T. and Tiggler, H. "FPGA Implementation of Sine and Cosine Generators Using the CORDIC Algorithm", Proc. of Military and Aerospace Application of Programmable Devices and Technologies Conference (MAPLD 99), Sep. 1999, Laurel, MA, A-2, pp. 28-30.

[Meg62] Meggitt, J.E. "Pseudo division and pseudo multiplication processes", IBM Journal of Research and Development, 1962, No. 6, pp. 210-226.

[Cav93] Kota, K. and Cavallaro, J.R. "Numerical accuracy and hardware tradeoffs for CORDIC arithmetic for special-purpose processors", IEEE Trans. on Computers, Vol. 42, No. 7, July 1993, pp.769-779.