IPHV8I3004X
International Journal Of Advanced Research and Innovation -Vol.8, Issue .III
ISSN Online: 2319 – 9253
Print: 2319 – 9245

AN IMPLEMENTATION OF EFFECTIVE IDEA, PLAN, AND USAGE OF RECONFIGURABLE CORDIC

[#1]KOMMA SHRUTHI, M.Tech Student,
[#2]V.SRIDHAR, Assistant Professor,
Dept of ECE,
MOTHER THERESSA COLLEGE OF ENGINEERING & TECHNOLOGY, KARIMNAGAR, TS, INDIA.

ABSTRACT: This short exhibits the key idea, outline technique, and usage of reconfigurable arrange turn advanced PC (cordic) designs that can be arranged to work either for roundabout or for hyperbolic directions in revolution and in addition vectoring-modes. It can, in this way, be utilized to play out every one of the elements of both round and hyperbolic cordic. We propose three reconfigurable cordic plans: 1) a reconfigurable revolution mode cordic that works either for round or for hyperbolic direction; 2) a reconfigurable vectoring-mode cordic for roundabout and hyperbolic directions; and 3) a summed up reconfigurable cordic that can work in any of the modes for both roundabout and hyperbolic directions. The reconfigurable cordic can play out the calculation of different trigonometric and exponential capacities, logarithms, square-root, et cetera of roundabout and hyperbolic cordic utilizing either pivot mode or vectoring-mode cordic in one single circuit. It can be utilized as a part of computerized synchronizers, designs processors, logical adding machines, et cetera. It offers significant sparing of region multifaceted nature over the traditional plan for reconfigurable applications.

Index Terms— *coordinate, rotation, digital, computer (cordic), fixed, point, floating, point.a, verilog, implementation, fixed, point, cordic, processor*

1. INTRODUCTION

The current research in the design of high speed VLSI architectures for real-time digital signal processing (DSP) algorithms has been directed by the advances in the VLSI technology, which have provided the designers with significant impetus for porting algorithm into architecture. Many of the algorithms used in DSP and matrix arithmetic require elementary functions such as trigonometric, inverse trigonometric, logarithm, exponential, multiplication, and division functions. The commonly used software solutions for the digital implementation of these functions are table lookup method and polynomial expansions, requiring number of multiplication and additions/subtractions. However, digit by-digit methods exist for the evaluation of these elementary functions, which compute faster than software solutions. Some of the digit-by-digit methods for the computation of the above mentioned elementary functions were described by Henry Briggs in 1624 in “Arithmetical Logarithmical” [1, 2]. These are iterative pseudo division and pseudo multiplication processes, which resemble repeated-addition multiplication and repeated-subtraction division. In 1959, Older has proposed a special purpose digital computing unit known as Coordinate Rotation Digital Computer (CORDIC), while building a real time navigational computer for use in an aircraft [3, 4]. This algorithm was initially developed for trigonometric functions which were expressed in terms of basic plane rotations. The conventional method of implementation of 2D vector rotation shown in Figure 1 using Givens rotation transform is represented by the equations

$$x_{\text{out}} = x_{\text{in}} \cos\theta - y_{\text{in}} \sin\theta,$$
$$y_{\text{out}} = x_{\text{in}} \sin\theta + y_{\text{in}} \cos\theta, \qquad (1)$$

Where (x_{in}, y_{in}) and (x_{out}, y_{out}) are the initial and final coordinates of the vector, respectively. The hardware realization of these equations requires four multiplications, two additions/subtractions and accessing the table stored in memory for trigonometric coefficients. The CORDIC algorithm computes 2D rotation using iterative equations employing shift and add operations. The versatility of CORDIC is enhanced by developing algorithms on the same basis to convert between binary to binary coded decimal (BCD) number representation by Daggett in 1959 [5]. These iterative methods were described using decimal radix for the design of powerful small machines by Meggitt in 1962 [6]. Subsequently, Walther in 1971 [7, 8] has proposed a unified algorithm to compute rotation in circular, linear, and hyperbolic coordinate systems using the same CORDIC algorithm, embedding coordinate systems as a parameter. During the last 50 years of the CORDIC algorithm a wide variety of applications have emerged. The CORDIC algorithm has received increased attention after an unified approach is proposed for its implementation [7]. Thereafter, CORDIC based computing has been the choice for scientific calculator applications and HP-2152A co-processor, HP-9100 desktop calculator, HP-35 calculator are a few such devices based on the CORDIC algorithm [1, 8]. The CORDIC arithmetic processor chip is designed and implemented to perform various functions possible in rotation and vectoring mode of circular, linear, and hyperbolic coordinate systems [9]. Since then, CORDIC technique has been used in many applications [10], such as single chip CORDIC processor for DSP applications [11–15], linear transformations [16–21], digital filters [17], [22–24], and matrix based signal processing algorithms [25, 26]. More recently, the advances in the VLSI technology and the advent of EDA tools have extended the application of CORDIC algorithm to the field of biomedical signal processing [27], neural networks [28], software defined radio [29], and MIMO systems [30] to mention a few. Although CORDIC may not be the fastest technique to perform these operations, it is attractive due to its potential for efficient and low cost implementation of a large class of applications. Several modifications have been proposed in the literature for the CORDIC algorithm during the last two decades to provide high performance and low cost hardware solutions for real time computation of a two dimensional vector rotation and transcendental functions. A new type of arithmetic operation called fast rotations or orthonormal μ-rotations over a set of fixed angles is proposed [31]. These orthonormal μ-rotations are based on the idea of CORDIC and share the property that performing the rotation requires a minimal number of shiftadd operations. These fast rotations methods form a viable low cost alternative to the CORDIC arithmetic for certain applications such as FIR filter banks for image processing, the generation of spherical sample rays in 3D graphics, and the computation of eigenvalue decomposition and singular value decomposition.

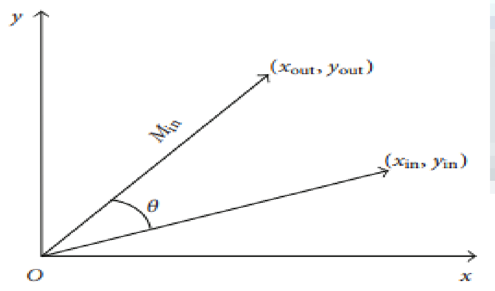

Figure 1: Two dimensional vector rotation.

We have carried out the critical study of different architectures proposed in the literature for 2D rotational CORDIC in circular coordinate system, to initiate the work for further latency reduction or throughput improvement. In this paper, we will review the architectures proposed for rotational CORDIC. Specifically, we focus on redundant unfolded architectures, employing techniques suitable to increase throughput and reduce latency. The rest of the paper is organized as follows. In Section 2, the basics of redundant arithmetic are presented. In Section 3, we present a review of generalized CORDIC algorithm, radix-2 and radix-4 CORDIC algorithms. In Section 4, general architectures being employed in literature for the implementation of the CORDIC algorithm are discussed. In Section 5, the complete taxonomy of rotational CORDIC algorithms is presented. Section 6 presents the low latency nonredundant CORDIC algorithm. Sections 7– 9 provide different redundant CORDIC algorithms along with the architectures being proposed in the literature for the rotational CORDIC, followed by the comparison of different methods in Section 10. Finally, conclusions are presented in Section 11.

II. RELATED WORK

Analysis of SQNR/MSE for LTI systems has been studied in [15,16]. Particularly, the approach in [23] focuses on the fixed-point accuracy of FFT units realized with integer multipliers. However, the results in the above references do not take into consideration the quantization error of constant coefficients leading to an unrealistic SQNR and MSE analysis, when error is defined as the difference between the fixed-point implementation and the reference floating-point model. Evaluating such an error is a crucial issue for verification purposes in scientific hardware computations [26]. Furthermore, the work in [19] has formally proved that if for the suitable low resolution variable or coefficient, its FB is increased by 1 bit only, it is possible to widely improve the hardware cost by reducing the bit-width of several other variables, which are reached through the next stages of logic. Hence, not having flexibility in controlling the bit-width of coefficients, as in [15,16,23], puts restrictions on the efficient design optimization. The analysis of MSE/SQNR for CORDIC units has been studied in [24] and [29]. In order to avoid the complex correlation between the quantization noise of intermediate variables and the value of input vectors, both approaches [24] and [29] ignore the quantization error of constant coefficients including the constant rotation angles as well as the constant coefficient K in the CORDIC algorithm. Although the MSE/SQNR analysis becomes straightforward, it is not safe, as it underestimates the exact error, defined as the difference between the fixed-point implementation and the reference floating-point model. Furthermore, the schemes in [24], [29] provide no flexibility in controlling bit-widths of coefficients, which may result in inefficient design optimizations. The approach in [27], which makes use of the error model in [28], focuses on the analysis of maximum mismatch for CORDIC multipliers required for FFT units. However, the SQNR analysis, still suffers from the same problems discussed for other conventional methods. The analysis of MSE/SQNR presented in this paper negligibly overestimates/underestimates the exact value of MSE/SQNR, which is robust and safe in contrast to the methods in [15,16,23,24], which underestimate/overestimate the exact MSE/SQNR. Furthermore, we present an analytical optimization technique, which provides flexibility in setting all the error sources independently.

III.CORDIC ALGORITHM

The CORDIC algorithm involves rotation of a vector v on the XY-plane in circular, linear and hyperbolic coordinate systems depending on the function to be evaluated. Trajectories for the vector vi for successive CORDIC iterations are shown in Figure 2. This is an iterative convergence algorithm that performs a rotation iteratively using a series of specific incremental rotation angles selected so that each iteration is performed by shift and add operation. The norm of a vector in these coordinate systems is defined as $\sqrt{x^2 + my^2}$, where $m \in \{1, 0, -1\}$ represents a circular, linear or hyperbolic coordinate system respectively. The norm preserving rotation trajectory is a circle defined by x2+ y2 = 1 in the circular coordinate system. Similarly, the norm preserving rotation trajectory in the hyperbolic and linear coordinate systems is defined by the function x2− y2 = 1 and x = 1, respectively. The CORDIC method can be employed in two different modes, namely, the rotation mode and the vectoring mode. The rotation mode is used to perform the general rotation by a given angle θ. The vectoring mode computes unknown angle θ of a vector by performing a finite number of micro rotations.

IV.CORDIC ARCHITECTURES

In this section, a few architectures for mapping the CORDIC algorithm into hardware are presented. In general, the architectures can be broadly classified as folded and unfolded as shown in Figure 4, based upon the realization of the three iterative equations (6). Folded architectures are obtained by duplicating each of the difference equations of the CORDIC algorithm into hardware and time multiplexing all the iterations into a single functional unit. Folding provides a means for trading area for time in signal processing architectures. The folded architectures can be categorized into bit-serial and word-serial architectures depending on whether the functional unit implements the logic for one bit or one word of each iteration of the CORDIC algorithm. The CORDIC algorithm has traditionally been implemented using bit serial architecture with all iterations executed in the same hardware [3]. This slows down the computational device and hence, is not suitable for high speed implementation. The word serial architecture [7, 48] is an iterative CORDIC architecture obtained by realizing the iteration equations (6). In this architecture, the shifters are modified in each iteration to cause the desired shift for the iteration. The appropriate elementary angles, αi are accessed from a lookup table. The most dominating speed factors during the iterations of word serial architecture are carry/borrow propagate addition/subtraction and variable shifting operations, rendering the conventional CORDIC [7] implementation slow for high speed applications. These drawbacks were overcome by unfolding the iteration process [41, 48], so that each of the processing elements always perform the same iteration as shown in Figure 5. The main advantage of the unfolded pipelined architecture compared to folded architecture is high throughput due to the hardwired shifts rather than time and area consuming barrel shifters and elimination of ROM. It may be noted that the pipelined architecture offers throughput improvement by a factor of n for n-bit precision at the expense of increasing the hardware by a factor less than n.

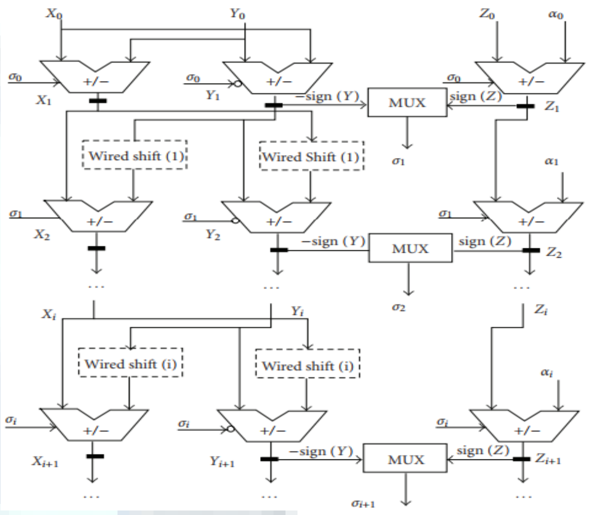

Figure 2: Unfolded pipelined CORDIC architecture.

V.FLOATING-POINT OPERATION ALGORITHM AND ANALYSIS

The algorithms of standard floating-point operators, adder/subtractor and multiplier, and non-standard floating-point operators, product-of-sum (PoS) operator and sum-ofproduct (SoP) are analysed and considered to increase computation performance. The algorithms can be applied for the single-and double-precision IEEE standard floatingpoint representations [53] as shown in Fig. 2.1. The single- and double-precision IEEE standard floating-point formats are binary computing formats that occupies 4 bytes (32 bits) and 8 bytes (64 bits). Both floating-point IEEE standard formants comprise three basic components, i.e. sign, exponent, and mantissa. The mantissa is composed of fraction and implicit leading digit. Tab. 2.1 shows the layout of the single- and double-precision IEEE standard floating-point formats, where a number of bits of each field are presented in square brackets.

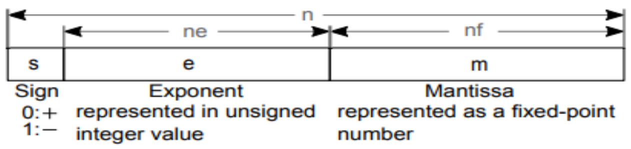
n
ne
nf
s
e
m
Sign
0:+
1:−
Exponent
represented in unsigned integer value
Mantissa
represented as a fixed-point number

Fig. 3: IEEE standard floating-point format

The layout of the single- and double-precision IEEE standards floating-point representations.

IEEE standard	*n*	Sign	Exponent (*ne*)	Mantissa (*nf*)
Single-precision	32	1[31]	8[30:23]	23[22:0]
Double-precision	64	1[63]	11[62:52]	52[51:0]

Common Functions of the common functions, i.e. Unpacking function, Comparison function, and Norm functions which are employed to perform the floating-point operational algorithms are examined. For the sake of simplification, all examples in this chapter are introduced in the single precision IEEE standard representation, where n=32, ne=8, and nf=23.

VI.CONCLUSION

In this paper, we have surveyed the algorithms for unfolded implementation of 2D rotational CORDIC algorithms. Special attention has been devoted to the systematic and comprehensive classification of solutions proposed in the literature. In addition to the pipelined implementation of non redundant radix-2 CORDIC algorithm that has received wide attention in the past, we have discussed the importance of redundant and higher radix algorithms. We have also stressed the importance of prediction algorithms to precompute the directions of rotations and parallelization of x/y path. It is worth noting that the considered algorithms should not be implemented as alternatives over the others, rather they should be integrated depending on the design constraints of a specific application. We can draw final conclusions about the different algorithms to achieve efficient implementation of application specific rotational CORDIC algorithm. As far as the application of redundant arithmetic to the pipelined implementation of the conventional radix-2 CORDIC algorithm is concerned, area is doubled with reduction in the adder delay of each stage from (log2n)tFA to 2tFA. Similarly, the hardware and iteration delay of redundant radix-2 CORDIC can be reduced by employing prediction technique for the pre computation of direction of rotations. Further, the latency reduction of this can be achieved by integrating the prediction technique with the redundant radix-4 arithmetic trading the area for variable scale factor computation. Another important observation about the solutions proposed with fully parallelization of x/y path is that it affects the modularity and regularity of the architecture leading to a poor scalable implementation. Finally, we conclude that the solution which can allow the design of scalable architecture, employing prediction and x/y path parallelization techniques to redundant CORDIC algorithm can achieve both latency reduction and throughput improvement.

REFERENCES

[1] D. S. Cochran, “Algorithms and accuracy in the HP-35,” Hewlett-Packard Journal, vol. 23, no. 10, 1972.
[2] J.-M. Muller, Elementary Functions: Algorithms and Implementation, Birkhauser, Boston, Mass, USA, 2004. ¨
[3] J. E. Volder, “The CORDIC trigonometric computing technique,” IRE Transactions on Electronic Computers, vol. 8, no. 3, pp. 330–334, 1959.
[4] J. E. Volder, “The birth of CORDIC,” Journal of VLSI Signal Processing, vol. 25, no. 2, pp. 101–105, 2000.
[5] D. H. Daggett, “Decimal-binary conversions in CORDIC,” IRE Transactions on Electronic Computers, vol. 8, pp. 335–339, 1959.
[6] J. E. Meggitt, “Pseudo division and pseudo multiplication processes,” IBM Journal, vol. 6, no. 2, pp. 210–226, 1962.
[7] J. S. Walther, “A unified algorithm for elementary functions,” in Proceedings of the AFIPS Spring Joint Computer Conference, pp. 379–385, May 1971.
[8] J. S. Walther, “The story of Unified CORDIC,” Journal of VLSI Signal Processing, vol. 25, no. 2, pp. 107–112, 2000.
[9] G. L. Haviland and A. A. Tuszynski, “A CORDIC arithmetic processor chip,” IEEE Journal of Solid-State Circuits, vol. 15, no. 1, pp. 4–15, 1980.
[10] Y. H. Hu, “CORDIC-based VLSI architectures for digital signal processing,” IEEE Signal Processing Magazine, vol. 9, no. 3, pp. 16–35, 1992.
[11] A. A. J. de Lange, A. J. van der Hoeven, E. F. Deprettere, and J. Bu, “Optimal floating-point pipeline CMOS CORDIC processor,” in Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS ’88), vol. 3, pp. 2043–2047, June 1988.
[12] A. A. J. de Lange, A. J. van der Hoeven, E. F. Deprettere, and P. Dewilde, “An application specific IC for digital signal processing: the floating point pipeline CORDIC processor,” in Proceedings of the European Conference on ASIC Design (ASIC ’90), pp. 62–67, May 1990.
[13] D. E. Metafas and C. E. Goutis, “A DSP processor with a powerful set of elementary arithmetic operations based on cordic and CCM algorithms,” Microprocessing and Microprogramming, vol. 30, no. 1–5, pp. 51–57, 1990.
[14] D. Timmermann, H. Hahn, B. J. Hosticka, and G. Schmidt, “A programmable CORDIC chip for digital signal processing applications,” IEEE Journal of Solid-State Circuits, vol. 26, no. 9, pp. 1317–1321, 1991.
[15] A. A. J. de Lange and E. F. Deprettere, “Design and implementation of a floating-point quasi-systolic general purpose CORDIC rotator for high-rate parallel data and signal processing,” in Proceedings of the 10th IEEE Symposium on Computer Arithmetic, pp. 272–281, June 1991.