Research Report 234

A Redundant Arithmetic CORDIC System with a Unit Scale Factor

C T Clarke and G R Nudd

RR234

The CORDIC algorithm for the calculation of trigonometric functions has traditionally suffered from two problems; speed, and the necessity to pre-scale the inputs. The speed problem is overcome to a large extent by the introduction of redundant number systems which have been shown by others. Here we show a new CORDIC system which has a unit scale factor that can be ignored. The unit scale factor is achieved by rotating the vector in 3 dimensional space in a manner which scales it's projection onto the X-Y plane by the reciprocal of the overall scale factor. This new technique takes the same number of cycles as the standard CORDIC algorithm, with only marginally slower cycle times than the redundant system of Takagi [8]. The system is shown to be entirely compatible with redundant number system implementations of the CORDIC algorithm.

Department of Computer Science
University of Warwick
Coventry CV4 7AL
United Kingdom

December 1992

A Redundant Arithmetic CORDIC System With a Unit Scale Factor

C.T. Clarke, and G.R. Nudd

Department of Computer Science
University of Warwick
Coventry
CV4 7AL

December 1992

Abstract

The CORDIC algorithm for the calculation of trigonometric functions has traditionally suffered from two problems; speed, and the necessity to pre-scale the inputs. The speed problem is overcome to a large extent by the introduction of redundant number systems which have been shown by others. Here we show a new CORDIC system which has a unit scale factor that can be ignored. The unit scale factor is achieved by rotating the vector in 3 dimensional space in a manner which scales it's projection onto the X-Y plane by the reciprocal of the overall scale factor. This new technique takes the same number of cycles as the standard CORDIC algorithm, with only marginally slower cycle times than the redundant system of Takagi [8]. The system is shown to be entirely compatible with redundant number system implementations of the CORDIC algorithm.

1. Introduction

Trigonometric functions are widely used in many diverse fields. These functions are computationally difficult, and as a result, algorithms that use these functions heavily such as the Fourier transform were either significantly slowed or forced to use large look-up tables until the CORDIC algorithm was discovered. The CORDIC algorithm described by Volder in 1959 [9] is now well known and often used. Implementations are seen in areas such as calculators, arithmetic co-processors, DSP chips [7,3,1,4,5], and chirp-Z transform implementations [12]. The algorithm described by Volder can calculate the trigonometric functions, sine and cosine, and perform vector rotations. This was extended in 1971 by Walther [10], to a unified algorithm, that calculates many trigonometric functions, some directly, and others indirectly. The original algorithm required n iterations for an n bit result. Each iteration consisted of additions, shifts, and comparisons. The algorithm had a complexity of $O(n^2)$. The algorithm was recently modified by Takagi, and others[8,6] to incorporate redundant arithmetic . This resulted in an algorithm of complexity $O(n)$. However, up until 1974 the inputs or the outputs had to be scaled to take into account a vector extension which is a side effect of the CORDIC technique. In 1974 Despain [2] proposed a technique for making the scale factor unity. However it caused an increase in the number of cycles taken in the algorithm . In this paper we show that by generalising Volder's work to 3 dimensions we can make the 2 dimensional scale factor equal to 1, without needing any extra cycles. The scale factor can then be ignored. The hardware cost is not prohibitively high, and there is a speed advantage in not having to prescale the vector input. The technique for using redundant arithmetic systems devised by Takagi is then incorporated into the work to produce an algorithm which has a complexity of $O(n)$, and no pre- or post-scaling is required. The generalisation of the

CORDIC algorithm to 3 dimensions is in itself a useful result, but it suffers from limitations in the form shown here. The limitations do not affect the 2D CORDIC with unit scale factor. An application of the 3 dimensional CORDIC system is that of converting to a cartesian coordinate system from a spherical one.

2. CORDIC in 3 Dimensions

A vector R in 3 dimensional space is shown in figure 1. It has cartesian coordinates (X_i, Y_i, Z_i) and spherical coordinates (R_i, θ_i, ϕ_i).

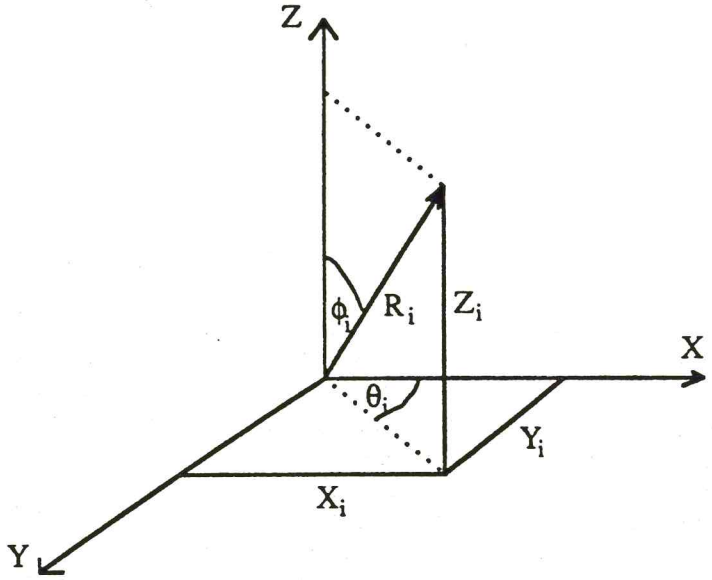

Figure 1. A vector in 3 dimensional space.

We can rotate this vector to become which has cartesian coordinates $(X_{i+1}, Y_{i+1}, Z_{i+1})$ and spherical coordinates $(R_i, \theta_i + \alpha_i, \phi_i + \beta_i)$. In this case, the relationship between the cartesian, and spherical coordinates of R, and S are shown in equations 1 to 6.

$$X_i = R_i \cos\theta_i \sin\phi_i \quad (1$$

$$Y_i = R_i \sin\theta_i \sin\phi_i \quad (2$$

$$Z_i = R_i \cos\phi_i \quad (3$$

$$X_{i+1} = R_i \cos(\theta_i + \alpha_i)\sin(\phi_i + \beta_i) \quad (4$$

$$Y_{i+1} = R_i \sin(\theta_i + \alpha_i)\sin(\phi_i + \beta_i) \quad (5$$

$$Z_{i+1} = R_i \cos(\phi_i + \beta_i) \quad (6$$

We define the following set of scalars:

$$U_i = R_i \cos\theta_i \cos\phi_i \quad (7$$

$$V_i = R_i \sin\theta_i \cos\phi_i \quad (8$$

$$W_i = R_i \sin\phi_i \quad (9$$

$$U_{i+1} = R_i \cos(\theta_i + \alpha_i)\cos(\phi_i + \beta_i) \quad (10$$

$$V_{i+1} = R_i \sin(\theta_i + \alpha_i)\cos(\phi_i + \beta_i) \quad (11$$

$$W_{i+1} = R_i \sin(\phi_i + \beta_i) \quad (12$$

If we expand out the left hand side of equations 4 to 6, and 10 to 12, and substitute from equations 1 to 3, and 7 to 9 wherever possible, we obtain:

$$U_{i+1} = U_i \cos\alpha_i \cos\beta_i - X_i \cos\alpha_i \sin\beta_i - V_i \sin\alpha_i \cos\beta_i + Y_i \sin\alpha_i \sin\beta_i \quad (13$$

$$V_{i+1} = V_i \cos\alpha_i \cos\beta_i - Y_i \cos\alpha_i \sin\beta_i + U_i \sin\alpha_i \cos\beta_i - X_i \sin\alpha_i \sin\beta_i \quad (14$$

$$W_{i+1} = W_i \cos\beta_i + Z_i \sin\beta_i \quad (15$$

$$Y_{i+1} = Y_i \cos\alpha_i \cos\beta_i + V_i \cos\alpha_i \sin\beta_i + X_i \sin\alpha_i \cos\beta_i + U_i \sin\alpha_i \sin\beta_i \quad (16$$

$$X_{i+1} = X_i \cos\alpha_i \cos\beta_i + U_i \cos\alpha_i \sin\beta_i - Y_i \sin\alpha_i \cos\beta_i - V_i \sin\alpha_i \sin\beta_i \quad (17$$

$$Z_{i+1} = Z_i \cos\beta_i - W_i \sin\beta_i \quad (18$$

This set of equations that splits any rotation into a set of smaller rotations in the same way as the 2 dimensional CORDIC algorithm. This is possible since we now have a definition of the rotated vector in terms of R, and the rotations α and β. We can therefore split any rotation into a set of smaller rotations. We give α_i, and β_i the same magnitude, but their signs may differ. This is simply for convenience. It also means that the accuracy of the algorithm is the same in all dimensions at a given iteration i. As with the 2 dimensional CORDIC algorithm, we choose $\alpha_i = a_i.\arctan 2^{-i}$, and $\beta_i = b_i.\arctan 2^{-i}$, where a_i, and are $\in \{-1,1\}$. From this, we can derive equations 19 to 22 for the sine, and cosine of α_i, and β_i. The a_i, and b_i factors disappear from the cosine equations because $\cos\delta = \cos -\delta$, whereas they are present in the sine equations because $\sin\delta = -\sin -\delta$.

$$\cos\beta_i = \frac{1}{\sqrt{1+2^{-2i}}} \quad (19$$

$$\sin\beta_i = \frac{b_i 2^{-i}}{\sqrt{1+2^{-2i}}} \quad (20$$

$$\cos\alpha_i = \frac{1}{\sqrt{1+2^{-2i}}} \quad (21$$

$$\sin\alpha_i = \frac{a_i 2^{-i}}{\sqrt{1+2^{-2i}}} \quad (22$$

The divisor of all these equations is a constant for each iteration step. We will call this constant k_i. Substituting equations 19 to 22 into equations 13 to 18, we get:

$$U_{i+1} = \frac{1}{k_i^2}(U_i - X_i b_i 2^{-i} - V_i a_i 2^{-i} + Y_i a_i b_i 2^{-2i}) \quad (23$$

$$V_{i+1} = \frac{1}{k_i^2}(V_i - Y_i b_i 2^{-i} + U_i a_i 2^{-i} - X_i a_i b_i 2^{-2i}) \quad (24$$

$$W_{i+1} = \frac{1}{k_i}(W_i + Z_i b_i 2^{-i}) \quad (25$$

$$Y_{i+1} = \frac{1}{k_i^2}(Y_i + V_i b_i 2^{-i} + X_i a_i 2^{-i} + U_i a_i b_i 2^{-2i}) \quad (26$$

$$X_{i+1} = \frac{1}{k_i^2}(X_i + U_i b_i 2^{-i} - Y_i a_i 2^{-i} - V_i a_i b_i 2^{-2i}) \quad (27$$

$$Z_{i+1} = \frac{1}{k_i}(Z_i - W_i b_i 2^{-i}) \quad (28$$

The scale factor for Z_{i+1}, and W_{i+1} is different to that of U_{i+1}, V_{i+1}, X_{i+1}, and Y_{i+1}. This is not a problem since the two sets of variables do not interact in any way. These equations can be used to rotate a 3 dimensional vector. The equations 23 to 28 are iterated ignoring the k_i. We can prescale the inputs, or post scale the outputs by the overall scale factor K, for Z and W, and K^2 for U, V, X, and Y, where:

$$K = \prod_{i=1}^{n} \sqrt[2]{1+2^{-2i}} \quad (29$$

When we start, at $i = 1$, we need not only the cartesian coordinates X_1, Y_1, and Z_1, but also U_1, V_1, and W_1. These would need to be calculated, and hence any advantage would be lost. The only situation that it would be of use, is when the vector initially lay on one axis, as this would cause U_1, V_1, and W_1 to be trivial.

4. Application of 3D CORDIC to scaling

It can be seen from the above discussion, that this 3D CORDIC algorithm suffers from a problem at the first iteration. However, when we use this system as a unit scale factor 2D CORDIC, the effect disappears. We use the X-Y plane for our 2 dimensional rotation, and rotate the vector out of the plane to reduce it's projected length on the plane. The rotation out of the plane is chosen to have the opposite effect to that of the vector extension, and hence they cancel each other out, giving a unit scale factor. This is shown in figure 2. Assume a start vector A, this rotates in the 2D case to B which is extended. Instead, the vector is rotated to C. This vector is also extended, but the projection on the X-Y plane (D) is of the same length as A, and rotated by the correct angle in the X-Y plane.

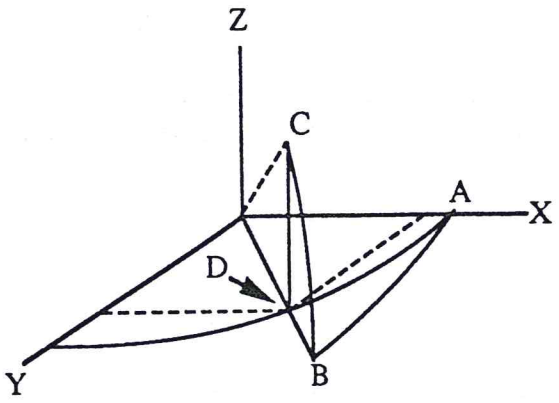

Figure 2. Rotating in 3D to remove the 2D scale factor.

If we start with $Z_1 = 0$, then U_1, and V_1 will also be 0 since ϕ_1 must be $\frac{\pi}{2}$. Also, to calculate X_n, and Y_n, we do not need to calculate W_n or Z_n, and hence we do not need W_1 or Z_1. All we need do is set X_1, and Y_1 to the initial vector and, U_1 and V_1 to 0. To perform a rotation with no scale factor we then rotate θ by the required vector rotation, and rotate ϕ by the angle $\arccos\frac{1}{K^2}$. The overall effect of the two rotations is a single unscaled rotation.

5. Combination of Scaleless CORDIC, and Redundant Number Systems

We can incorporate redundant number arithmetic into this algorithm without difficulty. The rotation of ϕ is known at the design stage, and so we do not need to use the double rotation method suggested by Takagi [8] for this rotation. If we use the double rotation method for the rotation of θ, then we get equations 30 to 35, where c_i, and e_i are $\in \{1,-1\}$, each denoting the sign of a half rotation in the X-Y plane. These two half rotations combine to form a positive, a negative, or no rotation. When there is no rotation there is still an extension of the vector.

$$U_{i+1} = U_i S_1 - X_i S_1 b_i 2^{-i} - V_i S_2 2^{-i} + Y_i S_2 b_i 2^{-2i} \quad (30$$

$$V_{i+1} = V_i S_1 - Y_i S_1 b_i 2^{-i} + U_i S_2 2^{-i} - Y_i S_2 b_i 2^{-2i} \quad (31$$

$$Y_{i+1} = Y_i S_1 + V_i S_1 b_i 2^{-i} + X_i S_2 2^{-i} + U_i S_2 b_i 2^{-2i} \quad (32$$

$$X_{i+1} = X_i S_1 + U_i S_1 b_i 2^{-i} - Y_i S_2 2^{-i} - V_i S_2 b_i 2^{-2i} \quad (33$$

where:

$$S_1 = 1 - c_i e_i 2^{-2i-2} \quad (34$$

and:

$$S_2 = \frac{c_i + e_i}{2} \quad (35$$

The scale factor still needs to be calculated, to find the correcting rotation of the vector off the X-Y plane. The new scaling factor L in this case is:

$$L = \prod_{i=1}^{n} \left(1 + 2^{-2i-2}\right) \cdot \sqrt{1 + 2^{-2i}} \qquad (36$$

The rotation of ϕ will be $\arccos\frac{1}{L}$. Note that this scale factor is different to that in the previous section.

So far we have not addressed the issue of making the decision of whether to rotate in a positive or negative direction at any particular stage. The case of rotations of ϕ, is simple because we know the angle through which we are going to rotate B. Hence we can pre-calculate the set of b_i so that :

$$B = \sum_{i=1}^{n} b_i \arctan 2^{-i} \qquad (37$$

Once calculated, b_i can be stored in a very small ROM (n bits). The calculation of the rotations in the X-Y plane is done as Takagi suggested. We refer to the angle through which we rotate the vector as T_1. The calculation of c_i, and e_i is made by taking the most significant three digits of T_i at stage i, and c_i, and e_i are defined as follows:

$$c_i e_i = \begin{Bmatrix} -1 & 1 & \text{if } \left[t_{i-1}^{i-1} t_{i-1}^{i} t_{i-1}^{i+1}\right] < 0 \\ 1 & -1 & \text{if } \left[t_{i-1}^{i-1} t_{i-1}^{i} t_{i-1}^{i+1}\right] = 0 \\ 1 & 1 & \text{if } \left[t_{i-1}^{i-1} t_{i-1}^{i} t_{i-1}^{i+1}\right] > 0 \end{Bmatrix} \qquad (38$$

Note that c_i indicates the direction of rotation, and e_i represents whether the second rotation is in the same direction as the first (+1), or not (-1). We can find the new value T_{i+1} from equation 39.

$$T_{i+1} = T_i - c_i (e_i + 1) \arctan 2^{-i} \qquad (39$$

6. Impact of the new algorithm

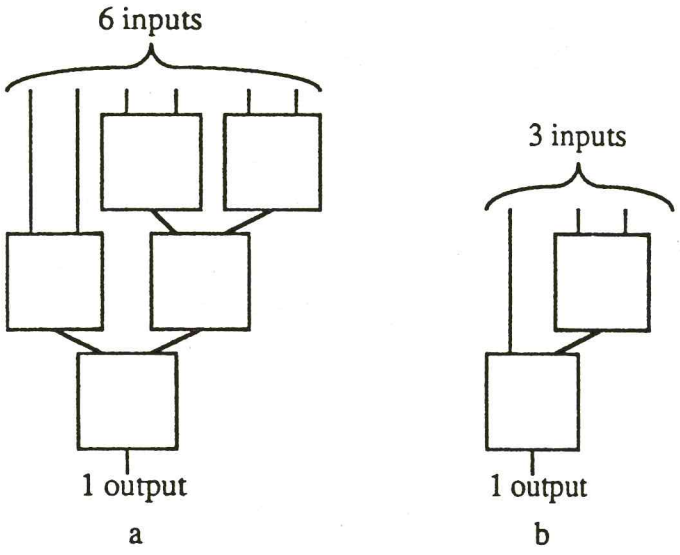

Figure 3. Example bit slice of redundant adder layouts for a) A unit scale factor system b) Takagi's suggestion.

The equations for the 'scale factor'less redundant arithmetic CORDIC system are more complex than the equations suggested by Takagi [8]. The number of terms is doubled. This would, however only lead to a 50% longer delay in the adder stage due to the increased number of inputs from 3 to 6, as shown in figure 3.

As well as the increased need for addition, there would need to be two more registers for U and V, and multiple shifters. The addition of more shifters and registers should not greatly affect the clock period of a hardware implementation of the new algorithm because the operations would be performed in parallel. We could use the redundant method suggested by Duprat [6], and this would result in an adder delay equal to that of the constant scale factor technique given by Takagi. It would also be possible to incorporate this new system into the unified algorithm of J. S. Walther [10]. It should be noted however, that when using the unified algorithm, the scale factor is greater than 1 in the circular rotation case, equal to 1 in the linear case, and less than 1 in the hyperbolic case. This means that we need a different correction for each type of rotation. The obvious solution is to arrange for the two rotations to be of the same type. The sets of additions and subtractions necessary to compensate for the scale factor will be different in each case, and trivial for the linear case.

7. Conclusions

We have presented an extension to the CORDIC algorithm [9] which allows us to remove the need for scaling. The extension also has use in the conversion of spherical polar coordinates into cartesian coordinates. The removal of the need for scaling significantly improves the CORDIC algorithm and increases its potential throughput. This algorithm is entirely compatible with Walther's unified algorithm [10], and the various redundant arithmetic schemes [8,6] for the CORDIC algorithm.

References

1. Considine, V., CORDIC Trigonometric Function Generator For DSP, In Proc ICASSP89, IEEE, 1989, pp. 2381-2384.
2. Despain, A.M., Fourier Transform Computers Using CORDIC Iterations, IEEE transactions on computers, October 1974; c-23 (10) : pp. 993-1001.
3. Dixon, G., An Array Processor Implementation of The CORDIC Algorithm, IEE colloquium on VLSI signal processing architectures, May 1990 : pp. 5/1-5/8.
4. H. M. Ahmed, and Fu, K.H., A VLSI Array CORDIC Architecture, In Proc. ICASSP89, IEEE, 1989, pp. 2385-2388.
5. J.R. Cavallaro, and Luk, F.T., CORDIC arithmetic for an SVD processor, In Proc. 8th Symposium on Computer Arithmetic, IEEE Computer Society Press, 1987, pp. 113-119.
6. J. Duprat, and Muller, J.M., Fast VLSI Implementation of CORDIC Using Redundancy, In Algorithms and Parallel Architectures, 1991.
7. Timmermann D., Hahn H., Hostika B.J., and Schmidt G., A Programmable CORDIC Chip for Digital Signal Processing Applications, IEEE journal of solid-state circuits, September 1991; 26 (9) : pp. 1317-1321.
8. Takagi, N., Redundant CORDIC methods with a constant scale factor for sine and cosine computation, IEEE transactions on computers, September 1991; 40 (9) : pp. 989-995.
9. Volder, J.E. The CORDIC Trigonometric Computation Technique. In *Computer Arithmetic*. IEEE Computer Society Press, Swartzlander, E.E., pp. 226-230, 1990.
10. Walther, J.S., A unified algorithm for elementary functions, IEEE Computer Society Press, Vol. 1 1990, pp. 272-278.

11. X. Hu, R. G. Harber, and Bass, S.C., Expanding the Range of Convergence of the CORDIC Algorithm, IEEE transactions on computers, January 1991; 40 (1) : pp. 13-21.

12. Y. H. Hu, and Naganathan, S., A Novel Implementation of a Chirp Z-Transform Using a CORDIC processor, IEEE transactions on acoustics, speech, and signal processing, February 1990; 38 (2) : pp. 352-354.