Design and Implementation of Cosine Transforms Employing a CORDIC Processor

Sharaf El-Din El-Nahas, Ammar Mottie Al Hosainy, Magdy M. Saeb
Arab Academy for Science and Technology,
School of Engineering,
Alexandria, EGYPT

ABSTRACT

COordinate Rotation DIgital Computer (CORDIC) is widely used in recent DSP applications, due to its simple and well-designed approach that utilizes only add and shift operations instead of multipliers. In this work, we introduce a Field Programmable Gate Array (FPGA) implementation of a CORDIC processor that provides high performance and at the same time an efficient implementation area. This is achieved by increasing the number of segments of the signs vector curve. This has led to an increase in the number of comparators, however with small effect on the implementation area. As an application, we have developed a Discrete Cosine Transform (DCT) module that employs the CORDIC core to generate cosine terms.

I. INTRODUCTION

The Coordinate rotation algorithm, first introduced by Volder [1] for fast computations of trigonometric functions and their inverses, is a well-known and widely studied method for plane vector analyses. The CORDIC method can be used for multiplication and division, as well as for conversion between binary and mixed radix number systems. Walther [2] has demonstrated a "unified" CORDIC algorithm that can be used to calculate trigonometric, exponential and square root functions along with their inverses. Fundamentally, the CORDIC method evaluates elementary functions simply by lookup tables, shift and add operations instead of multiplication. A small number of the order of n, where n bits of precision is required in the evaluation of the functions of pre-calculated fixed constants is required to be stored in the look-up table.
The CORDIC algorithm has advantageous geometrical interpretations. This is clear with trigonometric and exponential functions that are evaluated via rotations in the circular, hyperbolic and linear coordinate systems, respectively. Their inverses can be implemented in a "vectoring" mode in the appropriate coordinate system [3].
The organization of this paper is as follows: section 2 explains the theory of CORDIC, section 3 illustrates CORDIC implementations, DCT architecture is explained in section 4, section 5 shows the simulation results, and finally we end this paper with our conclusions.

II. CORDIC THEORY

Rotating a vector in a cartesian plane by the angle Φ this can be arranged so that

$$x' = \cos\Phi\,[\,x - y\tan\Phi\,] \quad (1)$$
$$y' = \cos\Phi\,[\,y + x\tan\Phi\,] \quad (2)$$

If the rotation angles are restricted so that $\tan(\Phi) = \pm 2^{-i}$, the multiplication by the tangent term is reduced to a simple shift operation. Arbitrary angles of rotation are obtainable by performing a series of successively smaller elementary rotations. If the decision at each iteration i, is which direction to relate rather than whether or not to rotate, then the cos (Φ) term becomes a constant (because cos (-Φ) = cos (Φ)). The iterative rotation can now be expressed as:

$$x_{i+1} = K_i\,[\,x_i + m\,.\,d_i\,.\,2^{-i}\,.\,y_i\,] \quad (3)$$
$$y_{i+1} = K_i\,[\,y_i - d_i\,.\,2^{-i}\,.\,x_i\,] \quad (4)$$

Where $d_i = \pm 1$, K_i is a constant and m steers the choice of rectangular (m = 0), circular (m = 1), or hyperbolic (m = −1) coordinate systems. In this work, we are working in a circular coordinate system, so m will be always equal to one. However, the required micro-rotations are not perfect rotations; they increase the length of the vector, in order to maintain a constant vector length, the obtained results have to be scaled by a scaling factor K. Nevertheless, assuming consecutive rotations in positive and negative directions or both, the scaling factor is constant and can be precomputed according to the following equation:

$$K = \prod_{i=0}^{n-1} k_i = \prod_{i=0}^{n-1} \left(1 + 2^{-2i}\right)^{-1/2} \quad (5)$$

Removing the scaling constant from the iterative equations yields a shift-add algorithm for vector rotation. The product of the K can be applied elsewhere in the system or treated as part of a system processing gain or by initiating the rotating vector by the reciprocal of the gain of a certain number of iterations. The angle of a composite rotation is uniquely defined by the sequence of the directions of the elementary rotations. That sequence can be represented by a decision vector. The set of all possible decision vectors is an angular measurement system based on binary arctangents. A better conversion method uses an additional adder-subtractor that accumulates the elementary rotation angles at each single iteration. The elementary angles can be expressed in any convenient angular unit. Those angular values are supplied by a small lookup Table (one entry per iteration) or are hardwired, depending on the implementation. The angle accumulator adds a third difference equation to the CORDIC algorithm

$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (6)$$

The CORDIC rotator is normally operated in one of two modes, the rotation mode and the vectoring mode[5].
In the rotation mode, a vector (x, y) is rotated by an angle θ. The angle accumulator is initialized with the desired rotation angle θ. The rotation decision per iteration is made to diminish the magnitude of the residual angle in the angle accumulator. The decision per is therefore based on the sign of the residual angle after each step. Naturally, if the input angle is already expressed in the binary arctangent base, the angle accumulator may be eliminated.
For rotation mode the CORDIC equations are

$$x_{i+1} = x_i + y_i \cdot d_i \cdot 2^{-i} \quad (7)$$
$$y_{i+1} = y_i - x_i \cdot d_i \cdot 2^{-i} \quad (8)$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (9)$$

Where $d_i = -1$ *if* $z_i < 0$, $+1$ otherwise

This provides the following results:

$$x_n = A_n [\, x_o \cos z_o - y_o \sin z_o \,] \quad (10)$$
$$y_n = A_n [\, y_o \cos z_o + x_o \sin z_o \,] \quad (11)$$
$$z_n = 0 \quad (12)$$
$$A_n = \prod_n \left(1 + 2^{-2i}\right)^{1/2} \quad (13)$$

In vectoring mode, length R and the angle towards the x-axis α of a vector (x, y) are computed. the CORDIC rotator rotates the input vector through whatever angle is necessary to align the result vector with the x-axis The result of the vectoring operation is a rotation angle and the scaled magnitude of the original vector (the x component of the result) The vectoring function works by seeking to minimize the y component of the residual vector at each rotation. The sign of the residual y component is used to determine which direction to rotate next. If the angle accumulator is initialized with zero, it will contain the traversed angle at the end of the iterations.
In vectoring mode the CORDIC equations are:

$$x_{i+1} = x_i - y_i \cdot d_i \cdot 2^{-i} \quad (14)$$
$$y_{i+1} = y_i + x_i \cdot d_i \cdot 2^{-i} \quad (15)$$
$$z_{i+1} = z_i - d_i \cdot \tan^{-1}(2^{-i}) \quad (16)$$

Where $d_i = +1$ *if* $y_i < 0$, -1 otherwise

This provides the following results:

$$x_n = A_n\sqrt{x_o^2 + y_0^2} \quad (17)$$

$$y_n = 0 \quad (18)$$

$$z_n = z_o + tan^{-1}(y_o / x_o) \quad (19)$$

$$A_n = \prod_n \left(1 + 2^{-2i}\right)^{1/2} \quad (20)$$

This set of equations will be used to design the micro-architecture shown in the following section.

III. IMPLEMENTATION OF THE CORDIC

We have used the unfolded structure of the CORDIC [5] to implement the three main equations 7,8 and 9 or 14,15 and 16. Figure 1 shows that at each iteration we need three adder-subtractors and a ROM to store the arctan value, the shift operation will be hardwired in the system. Based on an innovative method to implement the CORDIC [4] we can be able to replace the z accumulation path that is used to make the decision of the rotation direction, with a block of comparators, adder and a register to store the decision vector, or the signs vector. The micro-architecture is shown in Figure 2.

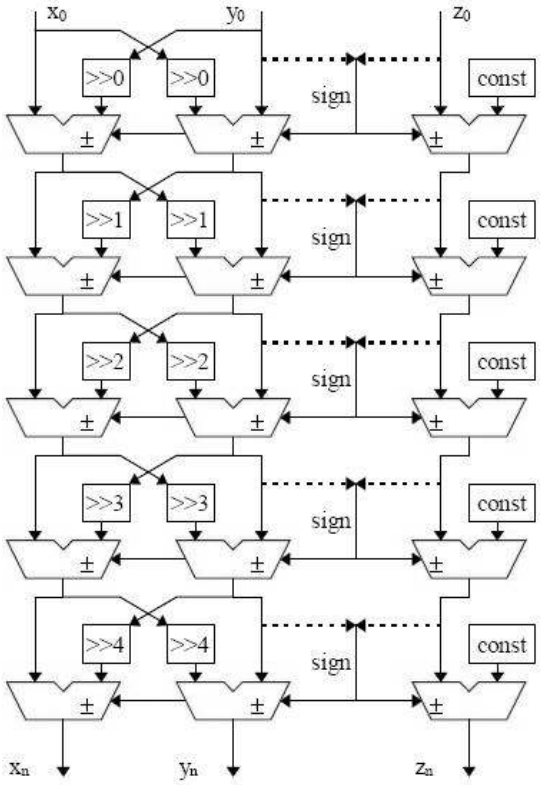

Figure 1: CORDIC Implementation

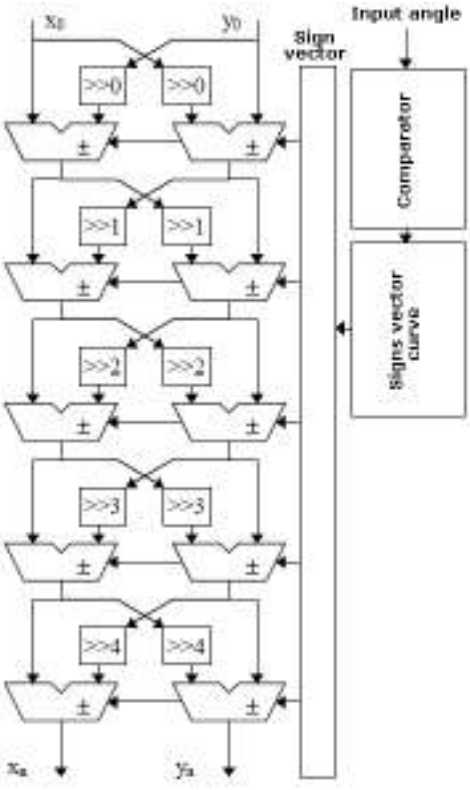

Figure 2: New architecture

The signs vector is generated from a linear equation that depends only on the input angle, Figure 3.

$$d = \theta + \begin{bmatrix} -C_0 \\ -C_0 + C_1 \\ -C_0 + C_1 + C_2 \\ -C_0 + C_1 + C_2 + C_3 \end{bmatrix} \quad (21)$$

All these constants depend only on the number of iterations n, and can be calculated using a Matlab model for the CORDIC. The most proper only way to increase the accuracy of the signs vector is to use another curve that has more segments, but this will increase the number of comparators. Accordingly, a higher real estate area for implementation is required.

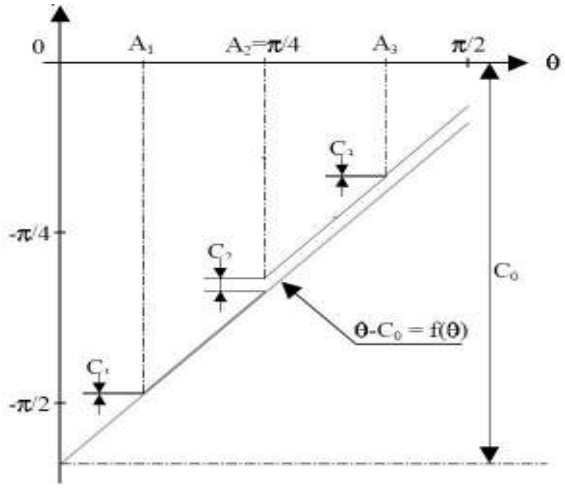

Figure 3: A four-segment curve

IV. DCT MODULE USING THE CORDIC

According to the equation of DCT

$$C(u) = \alpha(u) \sum_{x=0}^{N-1} f(x) \cos\left[\frac{\pi(2x+1)u}{2N}\right], \text{ for } u = 0, 1, 2, \ldots, N-1 \qquad (22)$$

Taking N equals to 16 points, positive or negative number, there will be 16 cosine function in the transformation, but in the first cosine function, at u = 0, the angle will be always equal zero, so its value will be 1. The input points to be transformed will be entered sequentially, so at each clock tick we have a new point ready to be transformed, because of this we use 15 CORDIC modules to calculate the DCT of 16 points, this helps to increase throughput, and decrease latency to 32 clock cycles. Each CORDIC module calculates cosine functions of a single point in the frequency domain as shown in Figure 4.

The cosine argument is given by:

$$\theta = \pi(2x+1)u/2N$$

It contains two variables x & u, both of them assume take the value from 0 to 15. Since we do not have multipliers in our design, we have to put these arguments in a Look Up Tables (LUTs). Accordingly, the CORDIC module accepts input angle only in the range from 0 to 90 degrees. Therefore, we have to replace θ with another angle in the range from 0 to 90 degrees. The value of the cosine component of this angle equals to the absolute value of the cosine component of the original angle. Moreover, we need one bit more for each new angle to indicate half of the angle. Hence, the cosine component will be positive in the right hand side and negative in the left hand side. The half bits will be stored in a LUT and connected to the accumulator to decide the next operation will be adding or subtracting, as shown in figure 4. We can generate all angles using counters, but there is an advantage in using LUT instead of counters, because by using LUT we can store Φ instead of the original angle θ, its cosine component is free from the CORDIC gain and also multiplied by $\sqrt{2}$, moreover we can adjust this angle to keep the CORDIC error to minimum value.

$$\Phi = \cos^{-1}(\sqrt{2} / 1.64676025 * \cos\theta) \qquad (23)$$

To save time and area, we can store the signs vector in LUTs instead of the angle. In this case, we do not need any comparator and we can use a signs vector curve that gives very accurate results. Using the LUT of the half bits to control the accumulator operations saves the area and speed of the pre and post-processors that determine the quarter of the input angle and put the output of the CORDIC in the right quarter.

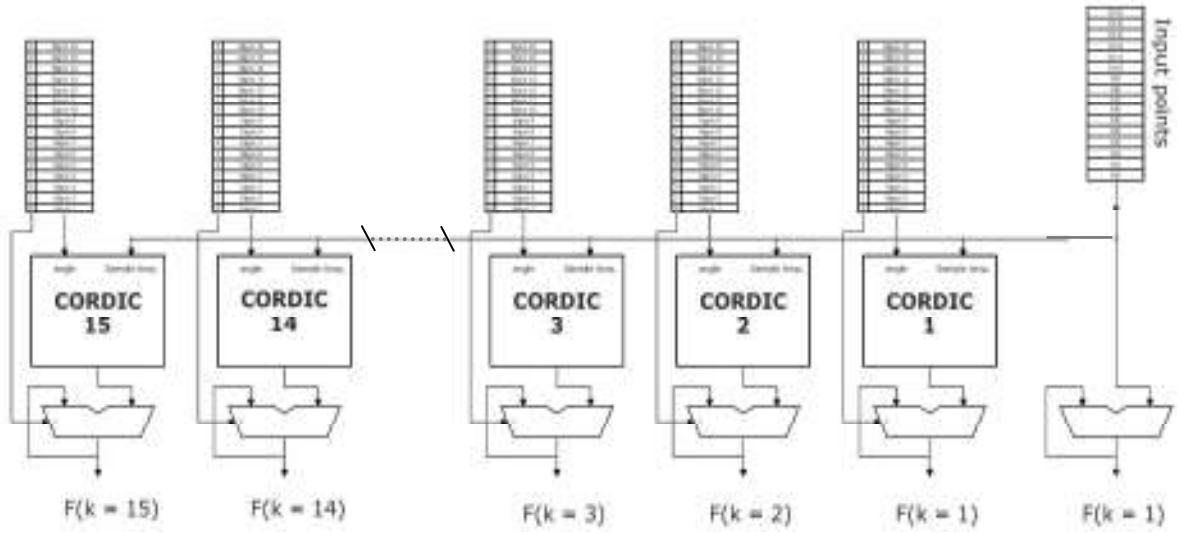

Figure 4: The DCT architecture

V. EXPERIMENTAL RESULTS

The design is implemented using Xilinx ISE 8.1 tools for synthesis in a Vertix2Pro and ModelSim SE 6.1 for Simulation. The simulation of the 16-point DCT is shown in figure 5. The sixteen input points are entered sequentially, one every clock cycle. Each point is inserted to all 15 CORDIC modules but with different angles from the LUTs, it takes 15 clock cycles to finish the CORDIC iterations and enters the accumulator, after 32 clock cycles the results of accumulators are passed to the output ports all in the same time and scaled by the factor 2^{15} . If the step between angles is 0.5 degree, then the maximum error in the cosine values of the CORIDC module is $1.831054*10^{-4}$ due to angle quantization. The step between any two angles must be more than 0.447624 degree. This error is acceptable in our application because in the 16-point DCT the step between the angles equals 5.625 degree. The maximum error in DCT values is 0.002746.

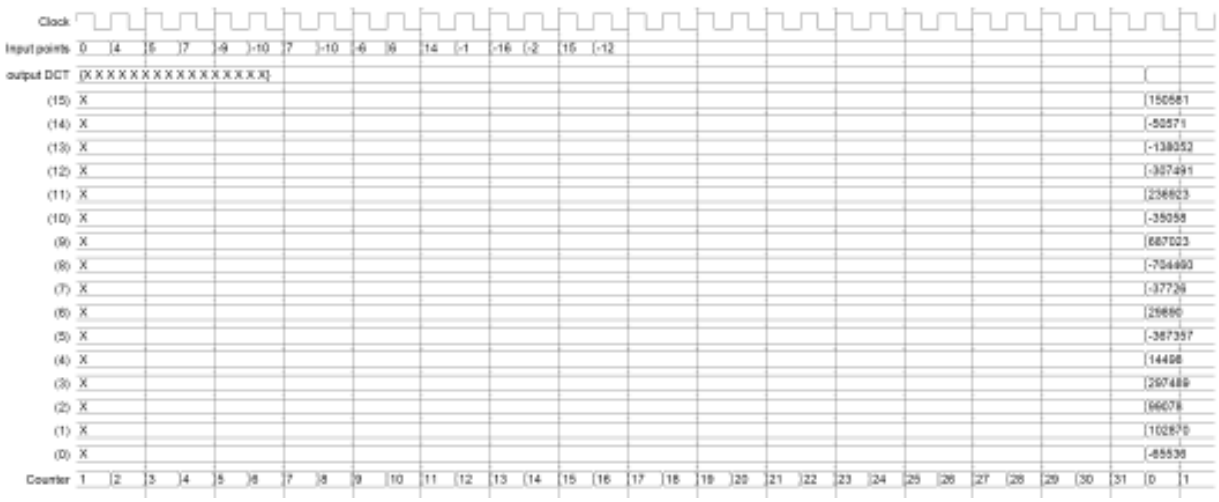

Figure 5: Simulation of the 16-point DCT

The maximum frequency that should be used is 266.951 MHz, the total number of slice registers is 10,327 out of 27,392 (37%), the total equivalent gate count for the design is 189,090, the floor plan is shown in figure 6. This floor plan provides the required verification of a completed implementation.

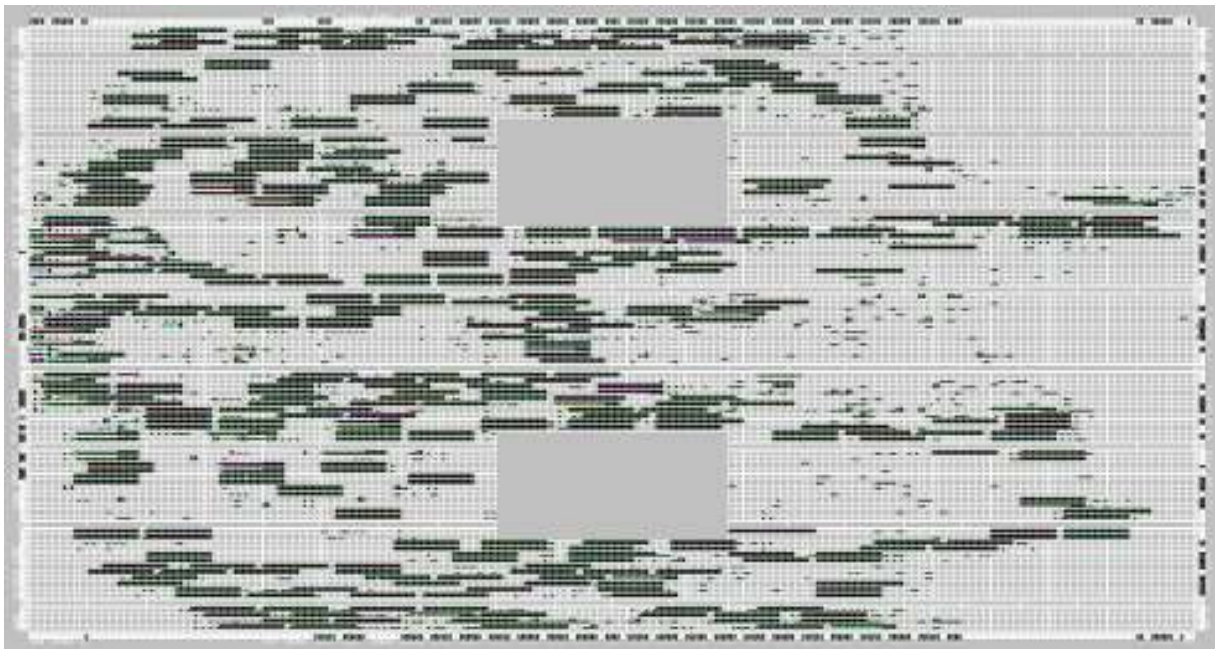

Figure 6: Floor plan of the 16-point DCT in Vertix2Pro

CONCLUSIONS

CORDIC-based designs have several advantages when building micro-architectures with no multipliers. This principle is used to save the implementation area measured by the gate count of the FPGA material. We can reduce the area, and at the same time keep a relatively high performance adopting the CORDIC design principle. Moreover, to improve our design, we use a linear correlation between the rotation angle θ and the corresponding direction of all micro-rotations and increasing the number of segments to improve the accuracy of the results. This improvement of the accuracy will be on the expense of a relatively small increase in implementation area. In the DCT module, we have made full use of this approach by storing the angles in a LUT to produce results already computed by the DCT factor. The reciprocal of the CORDIC gain is concurrently computed without using any additional multipliers.

REFERENCES

[1] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. on Electronic Computers*, vol. EC–8, pp. 330–334, Sep. 1959.

[2] J. S. Walther, "A unified Algorithm for Elementary Functions," in *Proceedings of the 38th Spring Joint Computer Conference*, Atlantic City, New Jersey, pp. 379–385, 1971.

[3] E. Grass, B. Sarker, and K. Maharatna "A Dual-Mode Synchronous / Asynchronous CORDIC Processor," *Proceedings of the 8th IEEE International Symposium on Asynchronous Circuits and Systems*, Manchester, UK, 2002.

[4] M.W. Kharrat, M. Loulou, and N. Masmoudi, "A new method to implement CORDIC algorithm," *in Proc. IEEE Int. Conf. Electronics, Circuits and Systems,* Malta, vol. 2, pp. 715–718, Sept. 2001.

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