Cordic Core Specification

Author: Richard Herveille
Rherveille@opencores.org

Rev. 0.3
June 22, 2001

This page left intentionally blank

Revision History

Rev.	Date	Author	Description
0.1	14/01/01	Richard Herveille	First Draft
0.2	21/06/01	Richard Herveille	Fixed some minor issues. Improved readability.
0.3	22/06/01	Richard Herveille	Completely revised section 1.1

Contents

INTRODUCTION **1**

ARCHITECTURE **6**

POLAR TO RECTANGULAR CONVERSION **8**

SINE AND COSINE CALCULATIONS **9**

RECTANGULAR TO POLAR CONVERSION **12**

1

Introduction

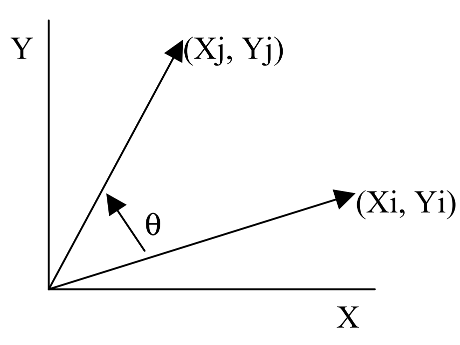
CORDIC (Coordinate Rotation Digital Computer) is a method for computing elementary functions using minimal hardware such as shifts, adds/subs and compares.

CORDIC works by rotating the coordinate system through constant angles until the angle is reduces to zero. The angle offsets are selected such that the operations on X and Y are only shifts and adds.

1.1 The numbers

This section describes the mathematics behind the CORDIC algorithm. Those not interested in the numbers can skip this section.

The CORDIC algorithm performs a planar rotation. Graphically, planar rotation means transforming a vector (Xi, Yi) into a new vector (Xj, Yj).



Using a matrix form, a planar rotation for a vector of (Xi, Yi) is defined as

$$\begin{bmatrix} X_j \\ Y_j \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{1}$$

The θ angle rotation can be executed in several steps, using an iterative process. Each step completes a small part of the rotation. Many steps will compose one planar rotation. A single step is defined by the following equation:

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} \cos\theta_n & -\sin\theta_n \\ \sin\theta_n & \cos\theta_n \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{2}$$

Equation 2 can be modified by eliminating the $\cos\theta_n$ factor.

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -\tan\theta_n \\ \tan\theta_n & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \tag{3}$$

Equation 3 requires three multiplies, compared to the four needed in equation 2.

Additional multipliers can be eliminated by selecting the angle steps such that the tangent of a step is a power of 2. Multiplying or dividing by a power of 2 can be implemented using a simple shift operation.

The angle for each step is given by

$$\theta_n = \arctan\left(\frac{1}{2^n}\right) \tag{4}$$

All iteration-angles summed must equal the rotation angle θ.

$$\sum_{n=0}^{\infty} S_n \theta_n = \theta \tag{5}$$

where

$$S_n = \{-1; +1\} \tag{6}$$

This results in the following equation for $\tan\theta_n$

$$\tan\theta_n = S_n 2^{-n} \tag{7}$$

Combining equation 3 and 7 results in

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \cos\theta_n \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \qquad (8)$$

Besides for the $\cos\theta_n$ coefficient, the algorithm has been reduced to a few simple shifts and additions. The coefficient can be eliminated by pre-computing the final result. The first step is to rewrite the coefficient.

$$\cos\theta_n = \cos\left(\arctan\left(\frac{1}{2^n}\right)\right) \qquad (9)$$

The second step is to compute equation 9 for all values of 'n' and multiplying the results, which we will refer to as K.

$$K = \frac{1}{P} = \prod_{n=0}^{\infty} \cos\left(\arctan\left(\frac{1}{2^n}\right)\right) \approx 0.607253 \qquad (10)$$

K is constant for all initial vectors and for all values of the rotation angle, it is normally referred to as the congregate constant. The derivative P (approx. 1.64676) is defined here because it is also commonly used.

We can now formulate the exact calculation the CORDIC performs.

$$\begin{cases} X_j = K(X_i \cos\theta - Y_i \sin\theta) \\ Y_j = K(Y_i \cos\theta + X_i \sin\theta) \end{cases} \qquad (11)$$

Because the coefficient K is pre-computed and taken into account at a later stage, equation 8 may be written as

$$\begin{bmatrix} X_{n+1} \\ Y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & -S_n 2^{-n} \\ S_n 2^{-n} & 1 \end{bmatrix} \begin{bmatrix} X_n \\ Y_n \end{bmatrix} \qquad (12)$$

or as

$$\begin{cases} X_{n+1} = X_n - S_n 2^{-2n} Y_n \\ Y_{n+1} = Y_n + S_n 2^{-2n} X_n \end{cases} \qquad (13)$$

At this point a new variable called 'Z' is introduced. Z represents the part of the angle θ which has not been rotated yet.

$$Z_{n+1} = \theta - \sum_{i=0}^{n} \theta_i \tag{14}$$

For every step of the rotation Sn is computed as a sign of Zn.

$$S_n = \begin{cases} -1 & if\ Z_n < 0 \\ +1 & if\ Z_n \geq 0 \end{cases} \tag{15}$$

Combining equations 5 and 15 results in a system which reduces the not rotated part of angle θ to zero.

Or in a program-like style:

```
For i=0 to n-1
        If (Z(n) >= 0) then
                Z(n + 1) := Z(n) – atan(1/2^i);
        Else
                Z(n + 1) := Z(n) + atan(1/2^i);
        End if;
End for;
```

The atan(1/2^i) is pre-calculated and stored in a table.

If we add the computation for X and Y we get the program-like style for the CORDIC core.

```
For i=0 to n-1
        If (Z(n) >= 0) then
                X(n + 1) := X(n) – (Yn/2^n);
                Y(n + 1) := Y(n) + (Xn/2^n);
                Z(n + 1) := Z(n) – atan(1/2^i);
        Else
                X(n + 1) := X(n) + (Yn/2^n);
                Y(n + 1) := Y(n) – (Xn/2^n);
                Z(n + 1) := Z(n) + atan(1/2^i);
        End if;
End for;
```

This algorithm is commonly referred to as driving Z to zero. The CORDIC core computes:

$$\lfloor X_j, Y_j, Z_j \rfloor = \left[P(X_i \cos(Z_i) - Y_i \sin(Z_i)),\ P(Y_i \cos(Z_i) + X_i \sin(Z_i)),\ 0\right]$$

There's a special case for driving Z to zero:

$$X_i = \frac{1}{P} = K \approx 0.60725$$

$$Y_i = 0$$

$$Z_i = \theta$$

$$\lfloor X_j, Y_j, Z_j \rfloor = [\cos\theta,\ \sin\theta,\ 0]$$

Another scheme which is possible is driving Y to zero. The CORDIC core then computes:

$$[X_j, Y_j, Z_j] = \left[P\sqrt{X_i^2 + Y_i^2},\ 0,\ Z_i + \arctan\left(\frac{Y_i}{X_i}\right)\right]$$

For this scheme there are two special cases:

1) $X_i = X$

$$Y_i = Y$$

$$Z_i = 0$$

$$[X_j, Y_j, Z_j] = \left[P\sqrt{X_i^2 + Y_i^2},\ 0,\ \arctan\left(\frac{Y_i}{X_i}\right)\right]$$

2) $X_i = 1$

$$Y_i = a$$

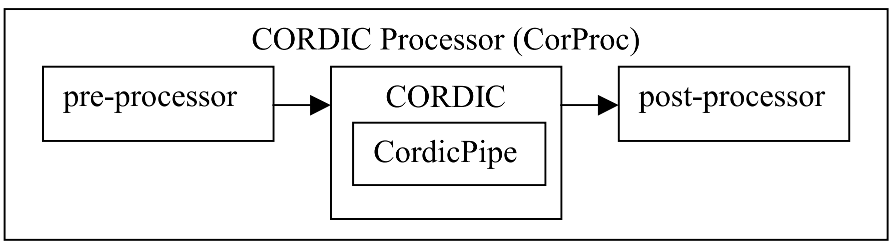
$$Z_i = 0$$

$$[X_j, Y_j, Z_j] = \left\lfloor P\sqrt{1 + a^2},\ 0,\ \arctan(a)\right\rfloor$$

2

Architecture

All CORDIC Processor cores are built around three fundamental blocks. The pre-processor, the post-processor and the actual CORDIC core. The CORDIC core is built using a pipeline of CordicPipe blocks. Each CordicPipe block represents a single step in the iteration processes.



2.1 Pre- and Post-Processors

Because of the arctan table used in the CORDIC algorithm, it only converges in the range of –1(rad) to +1(rad). To use the CORDIC algorithm over the entire 2π range the inputs need to be manipulated to fit in the –1 to +1 rad. range. This is handled by the pre-processor. The post-processor corrects this and places the CORDIC core's results in the correct quadrant. It also contains logic to correct the P-factor.

2.2 CORDIC

The CORDIC core is the heart of the CORDIC Processor Core. It performs the actual CORDIC algorithm. All iterations are performed in parallel, using a pipelined structure. Because of the pipelined structure the core can perform a CORDIC transformation each clock cycle. Thus ensuring the highest throughput possible.

2.3 CORDIC Pipeline

Each pipe or iteration step is performed by the CordicPipe core. It contains the atan table for each iteration and the logic needed to manipulate the X, Y and Z values.

3

Polar to Rectangular Conversion

Only CORDIC and CordicPipe are coded so far.

Coming soon.

4

Sine and Cosine calculations

Sine and Cosine can be calculated using the first CORDIC scheme which calculates:

$$\lfloor X_j, Y_j, Z_j \rfloor = [P(X_i \cos(Z_i) - Y_i \sin(Z_i)), P(Y_i \cos(Z_i) + X_i \sin(Z_i)), 0]$$

By using the following values as inputs

$$X_i = \frac{1}{P} = \frac{1}{1.6467} \approx 0.60725$$

$$Y_i = 0$$

$$Z_i = \theta$$

the core calculates:

$$\lfloor X_j, Y_j, Z_j \rfloor = [\cos\theta, \sin\theta, 0]$$

The input Z takes values from –180degrees to +180 degrees where:

0x8000 = –180degrees

0xEFFF = +80degrees

But the core only converges in the range –90degrees to +90degrees.

The other inputs and the outputs are all in the range of –1 to +1. The congregate constant P represented in this format results in:

$$Xi = 2^{15} \bullet P = 19898(dec) = 4DBA(hex)$$

Example:

Calculate sine and cosine of 30degrees.

First the angle has to be calculated:

$$360\deg \equiv 2^{16}$$

$$1\deg \equiv \frac{2^{16}}{360}$$

$$30\deg \equiv \frac{2^{16}}{360} \bullet 30 \approx 5461(dec) = 1555(hex)$$

The core calculates the following sine and cosine values for Zi=5461:

Sin : 16380(dec) = 3FFC(hex)

Cos : 28381(dec) = 6EDD(hex)

The outputs represent values in the –1 to +1 range. The results can be derived as follows:

$$2^{15} \equiv 1.0 \qquad\qquad 2^{15} \equiv 1.0$$

$$16380 \equiv \frac{1.0}{2^{15}} \bullet 16380 = 0.4999 \qquad\qquad 28381 \equiv \frac{1.0}{2^{15}} \bullet 28381 = 0.8661$$

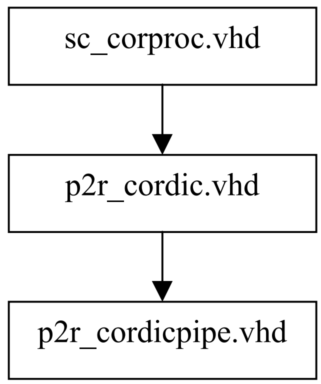
Whereas the result should have been 0.5 and 0.8660.

	0 deg	**30 deg**	**45 deg**	**60 deg**	**90 deg**
Sin	0x01CC	0x3FFC	0x5A82	0x6EDC	0x8000
Cos	0x8000	0x6EDD	0x5A83	0x4000	0x01CC
Sin	0.01403	0.49998	0.70709	0.86609	1.00000
Cos	1.00000	0.86612	0.70712	0.50000	0.01403

Table 1: Sin/Cos outputs for some common angles

Although the core is very accurate small errors can be introduced by the algorithm (see example and results table). This should be only a problem when using the core over the entire output range, because the difference between +1 (0x7FFF) and –1 (0x8000) is only 1bit.

4.1 Core structure



4.2 IO Ports

Port	Width	Direction	Description
CLK	1	Input	System Clock
ENA	1	Input	Clock enable signal
Ain	16	Input	Angel input
Sin	16	Output	Sine output
Cos	16	Output	Cosine output

Table 2: List of IO Ports for Sine/Cosine CORDIC Core

5.3 Synthesis Results

Vendor	Family	Device	Resource usage	Max. Clock speed
Xilinx	Spartan-II	XC2S100-6	387slices	116MHz

Table 3: Synthesis results for Rectangular to Polar CORDIC Core

5

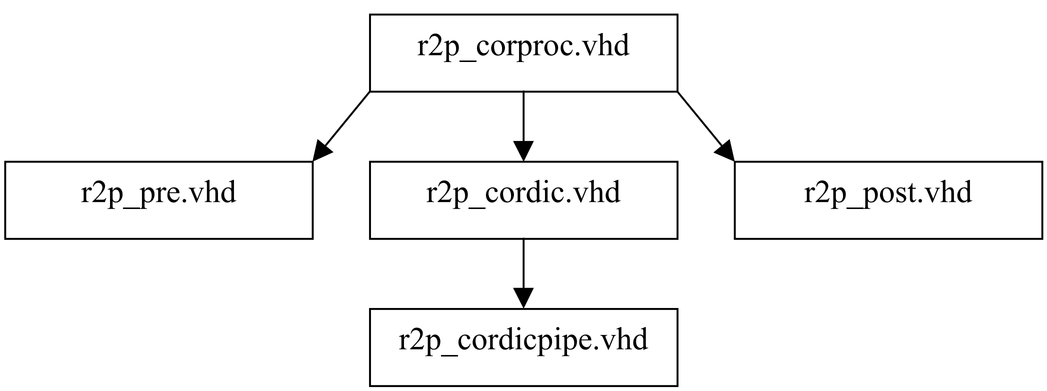
Rectangular to Polar Conversion

The rectangular to polar coordinate processor is built around the second CORDIC scheme which calculates:

$$[X_j, Y_j, Z_j] = \left[P\sqrt{1+a^2},\ 0,\ \arctan(a)\right]$$

It takes two 16bit signed words as inputs (Xin, Yin), which are the rectangular coordinates of a point in a 2-dimensional space. The core returns the equivalent Polar coordinates where Rout is the radius and Aout the angle or θ.

5.1 Core structure



5.2 IO Ports

Port	Width	Direction	Description
CLK	1	Input	System Clock
ENA	1	Input	Clock enable signal
Xin	16	Input	X-coordinate input. Signed value
Yin	16	Input	Y-coordinate input. Signed value
Rout	20	Output	Radius output. Unsigned value.
Aout	20	Output	Angle (θ) output. Singed/Unsigned value.

Table 4: List of IO Ports for Rectangular to Polar CORDIC Core

The outputs are in a fractional format. The upper 16bits represent the decimal value and the lower 4bits represent the fractional value.

The angle output can be used signed and unsigned, because it represents a circle; a -180 degree angle equals a +180 degrees angle, and a –45 degrees angle equals a +315 degrees angle.

5.3 Synthesis Results

The table below shows some synthesis results using a pipeline of 15 stages.

Vendor	Family	Device	Resource usage	Max. Clock speed
Altera	ACEX	EP1K50-1	2190lcells	68MHz
Xilinx	Spartan-II	XC2S100-6	704slices	93MHz

Table 5: Synthesis results for Rectangular to Polar CORDIC Core