

## Features

- Available for all Virtex<sup>™</sup>, Virtex-E, Virtex-II, Virtex-II Pro<sup>™</sup>, Spartan<sup>™</sup>-II, Spartan-IIe and Spartan-3 FPGA family members
- Functional configurations available:
  - Vector rotation (polar to rectangular)
  - Vector translation (rectangular to polar)
  - Sin and Cos
  - Sinh and Cosh
  - Atan and Atanh
  - Square root
- Optional coarse rotation and inverse coarse rotation modules to extend the range of CORDIC operation to the full 360 Degrees
- Optional amplitude compensation for CORDIC algorithm's amplitude scale factor
- Output rounding modes: Truncation, Round to Pos Infinity, Round to Pos / Neg Infinity and Round to Nearest Even
- Word Serial architectural configuration for small area
- Parallel architectural configuration for high throughput
- Control of the internal add-sub precision
- Control of the number of add-sub iterations
- Optional input and output registers
- Optional control signals: CE, ND, ACLR, SCLR, RFD and RDY
- X and Y data formats: Signed Fraction, Unsigned Fraction and Unsigned Integer
- Phase data formats: Radian, Pi Radian
- Fully optimized for speed and area
- Fully synchronous design using a single clock
- Incorporates Xilinx Smart-IP<sup>™</sup> technology for maximum performance
- To be used with v5.2i and later of the Xilinx CORE Generator<sup>™</sup>

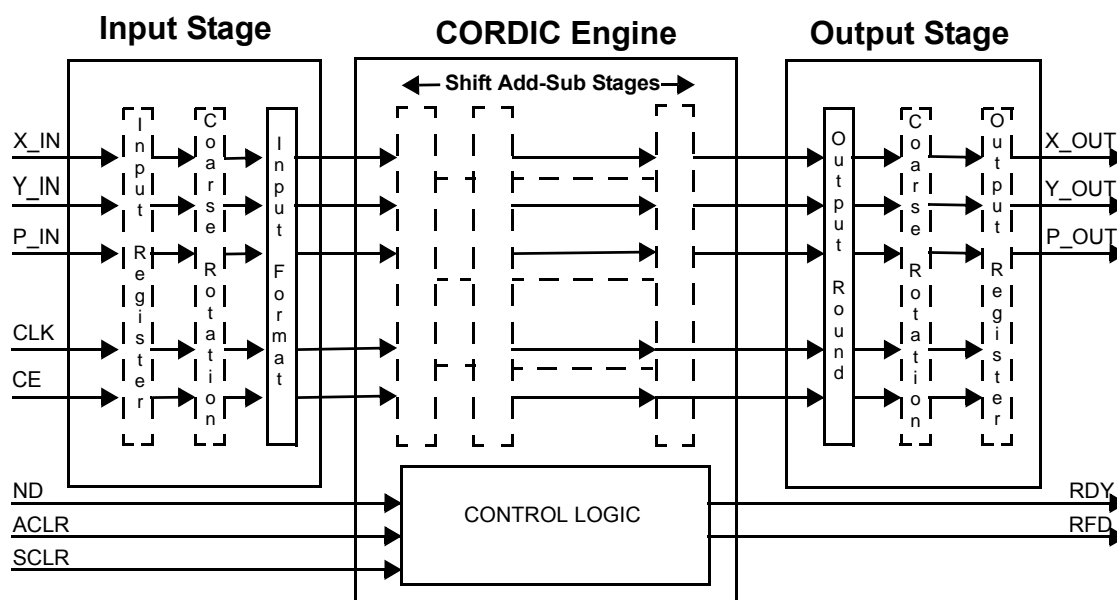


Figure 1: The CORDIC Core

## Functional Description

The CORDIC core implements a generalized “coordinate rotational digital computer” (CORDIC) algorithm. The CORDIC algorithm was initially developed by Volder [1] to iteratively solve trigonometric equations and was later generalized by Walther [2] to solve a broader range of equations, including the hyperbolic and square root equations. The CORDIC core implements: Rectangular <-> Polar Conversion, Trigonometric, Hyperbolic and Square Root Equations.

Two architectural configurations are available for the CORDIC core: a fully parallel configuration with single-cycle data throughput at the expense of silicon area and a word serial implementation with multiple-cycle throughput but occupying a small silicon area.

A coarse rotation is performed to rotate the input sample from the full circle into the first quadrant. (The coarse rotation stage is required as the CORDIC algorithm is only valid over the first quadrant). An inverse coarse rotation stage rotates the output sample into the correct quadrant.

The CORDIC algorithm introduces a scale factor to the amplitude of the result. The CORDIC core provides the option of automatically compensating for the CORDIC scale factor.

## The CORDIC Algorithm

The CORDIC algorithm was initially designed to perform a vector rotation, where the vector (X,Y) is rotated through the angle  $\theta$  yielding a new vector (X',Y').

### Equation 1: Vector Rotation Equation

- 1a)  $X' = (\cos(\theta) \times X - \sin(\theta) \times Y)$
- 1b)  $Y' = (\cos(\theta) \times Y + \sin(\theta) \times X)$
- 1c)  $\theta' = 0$

The CORDIC algorithm performs a vector rotation by breaking it down into a sequence of successively smaller rotations, each of angle  $\text{atan}(2^{-i})$ , known as “micro-rotations.”

### Equation 2: Vector rotation expressed as a series of ‘n’ micro-rotations.

- 2a)  $X' = \prod_{i=1}^n \cos(\text{atan}(2^{-i}))(X_i - \alpha_i Y_i 2^{-i})$
- 2b)  $Y' = \prod_{i=1}^n \cos(\text{atan}(2^{-i}))(Y_i + \alpha_i X_i 2^{-i})$
- 2c)  $\theta' = \sum_{i=1}^n \theta - (\alpha_i \cdot \text{atan}(2^{-i}))$   
 $\alpha_i = (+ \text{ or } -) 1.$

See [Vector Rotation](#) or [Vector Translation](#) for details on selecting  $\alpha_i$ .

Each micro-rotation stage can be expressed as a simple shift\_addsub operation.

### Equation 3: Expression for the $i^{\text{th}}$ microrotation.

- 3a)  $x_{i+1} = x_i - \alpha_i \cdot y_i \cdot 2^{-i}$
- 3b)  $y_{i+1} = y_i + \alpha_i \cdot x_i \cdot 2^{-i}$
- 3c)  $\theta_{i+1} = \theta_i + \alpha_i \cdot \text{atan}(2^{-i})$   
 $\alpha_i = (+ \text{ or } -) 1$

The CORDIC algorithm can be used to generate either a vector rotation or a vector translation.

### Vector Rotation

Vector rotation rotates the vector (X,Y) through the angle  $\theta$  to yield a new vector (X',Y'), as illustrated in [Figure 3](#).

Vector rotation is performed by selecting  $\alpha_i$ , such that  $\theta'$  converges towards zero. *I.e.*, when  $\theta_{i-1} \geq 0$ ,  $\alpha_i$  is set to -1 and when  $\theta_{i-1} < 0$ ,  $\alpha_i$  is set +1.

### Equation 4: Vector Rotation Equations

- 4a)  $X' = Z_i \times (\cos(\theta) \times X - \sin(\theta) \times Y)$
- 4b)  $Y' = Z_i \times (\cos(\theta) \times Y + \sin(\theta) \times X)$
- 4c)  $\theta' = 0$

$$Z_i = \frac{1}{\prod_{i=1}^n \cos(\text{atan}(2^{-i}))}$$

### Vector Translation

Vector translation rotates the vector (X,Y) around the circle until the Y component equals zero as illustrated in [Figure 5](#). The outputs from vector translation are the magnitude, X', and angle,  $\theta'$ , of the input vector (X,Y).

Vector translation is performed by selecting  $\alpha_i$  such that Y' converges towards zero. *I.e.*, when  $Y_{i-1} \geq 0$ ,  $\alpha_i$  is set to -1 and when  $Y_{i-1} < 0$ ,  $\alpha_i$  is set +1.

### Equation 5: Vector Rotation Equations

- 5a)  $X' = Z_i \times \sqrt{(X^2 + Y^2)}$
- 5b)  $Y' = 0$
- 5c)  $\theta' = \text{atan}(X / Y)$

$$Z_i = \frac{1}{\prod_{i=1}^n \cos(\text{atan}(2^{-i}))}$$

## The CORDIC Scale Factor

The outputs of the CORDIC algorithm, equations 4 and 5, are equivalent to a vector rotation or vector translation scaled by a constant  $Z_i$ . The constant  $Z_i$  is known as the CORDIC scale factor.

### Equation 6: The CORDIC Scale Factor

$$6a) \quad Z_i = \frac{1}{\prod_{i=1}^n \cos(\tan^{-1}(2^{-i}))}$$

The Taylor series expansion of  $\cos(\tan^{-1}(2^{-i}))$  is  $(1 + 2^{-2i})^{-1/2}$ . Hence, the constant  $Z_i$  can be expressed as

$$6b) \quad Z_i = \prod_{i=1}^n (1 + 2^{-2i})^{1/2}$$

The CORDIC scale factor is dependant on the number of iterations,  $n$ , but is constant with respect to  $\alpha_i$ . Only functional configurations: Rotate, Translate, Rectangular to Polar, and Polar to Rectangular are affected by the CORDIC scale factor. When these functional configurations are

selected, the CORDIC core provides the option of multiplying by  $1 / Z_i$  to cancel out the scaling factor. See section **Compensation Scaling** for further details.

## Configuration GUI

The CORDIC Core is configured using the four page GUI shown in **Figure 2**, **Figure 10**, **Figure 11**, and **Figure 12**. The parameters on the first page configure the functional selection and architectural configuration of the CORDIC core. The parameters on the second page configure the phase and magnitude data formats, optional control signals and synchronization. The parameters on the third page configure the rounding mode and input-outputs. The parameters on the fourth page configure the advanced configuration parameters: Coarse Rotation, Iterations, the internal Precision and optional Compensation Scaling. The fourth page also displays the latency and controls the inclusion of placement information.

The “Next” and “Back” buttons are used to move between pages on the GUI. Once all the parameters for the CORDIC core are configured, the “Generate” button is used to generate the core.

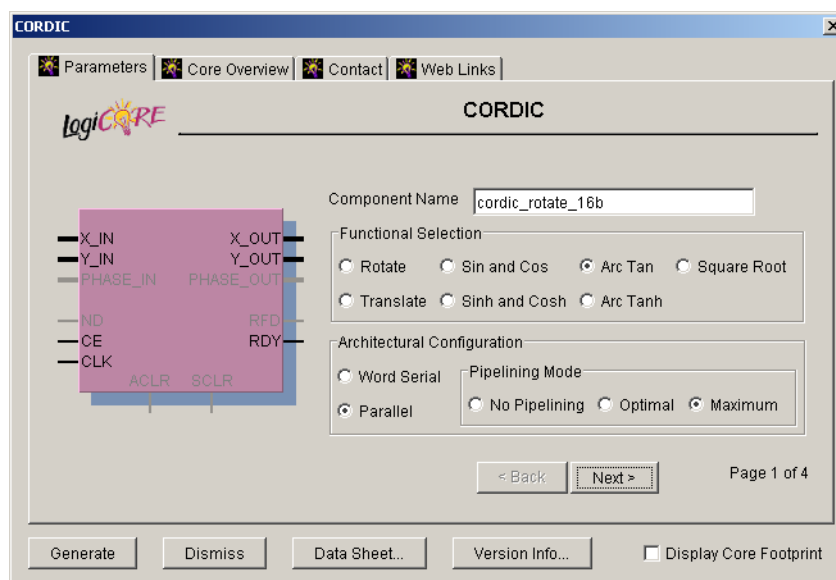


Figure 2: CORDIC V2\_0 Configuration GUI (Page 1)

### Component Name

The component name is used as the base name of the output files generated for the core. Names must begin with a letter and must be composed from the following characters: a to z, 0 to 9, and “\_”.

### Functional Selection

The CORDIC core can be used in a variety of functional configurations. The functional configurations are exclusive and are selected at instantiation time using the radio buttons on page 1 of the CORDIC GUI (**Figure 2**). Each functional configuration lists the input/output ports and the range of valid data.

In general, X\_IN, Y\_IN, X\_OUT and Y\_OUT express signed binary numbers of 1QN format and PHASE\_IN and PHASE\_OUT express signed binary numbers of 2QN format. When the square root functional configuration is selected, two new data formats are available, Unsigned Integer and Unsigned Fraction. See section [Data Input/Output Representation](#) for further details.

## Vector Rotation

### Polar to Rectangular Translation

When the vector rotation functional configuration is selected, the input to the CORDIC core is the input vector  $X, Y$  and the rotation angle  $\theta$ . The input vector  $X, Y$  is rotated by the angle  $\theta$  using the CORDIC algorithm. This generates the scaled output vector  $Z_1 \times (X', Y')$  as shown in [Figure 3](#).

An optional scaling module is provided to compensate for the CORDIC scale factor  $Z_1$ .

An optional coarse rotation and inverse rotation module is provided to extend the input/output ranges of  $X, Y$  and phase.

A Polar to Rectangular Translation can be implemented by selecting the vector rotation from function and setting the input vector to  $(Mag, 0)$  and rotation angle  $\theta$  as shown in [Figure 4](#).

The CORDIC vector rotation function preserves magnitude hence the user can scale the input/output range. Thus

if  $(X_{in}, Y_{in})$  rotated by angle  $\theta = (X_{out}, Y_{out})$  then

$K^*(X_{in}, Y_{in})$  rotated by angle  $\theta = K^*(X_{out}, Y_{out})$

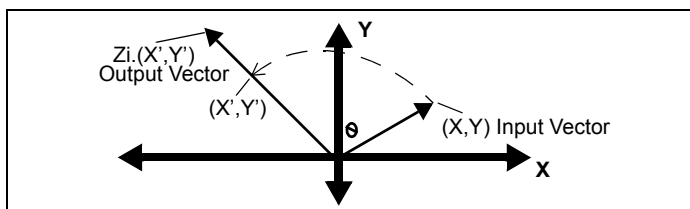


Figure 3: Vector Rotation

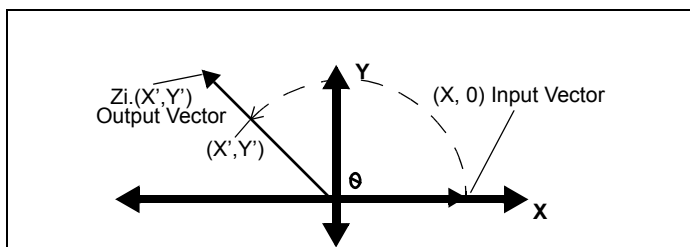


Figure 4: Polar to Rectangular Conversion

Table 1: Vector Rotation I/O

Signal	Description
X_IN	Input X Coordinate Range: $-1 \leq X\_IN \leq 1$
Y_IN	Input Y Coordinate Range: $-1 \leq Y\_IN \leq 1$
PHASE_IN	Input Rotation Angle Range: $-\pi \leq PHASE\_IN \leq \pi$
X_OUT	Output X Coordinate * Z Range: $-\sqrt{2} \leq X\_OUT \leq \sqrt{2}$
Y_OUT	Output Y Coordinate * Z Range: $-\sqrt{2} \leq Y\_OUT \leq \sqrt{2}$

### Example 1) Vector Rotation

The input vector,  $(X_{in}, Y_{in})$ , is expressed as a pair of signed 1QN format numbers. The rotation angle,  $\theta$  radians, is expressed as a signed 2QN number. The output vector,  $(X_{out}, Y_{out})$ , is expressed as a pair of signed 1QN format numbers. In this example, the input/output width is set to 10 bits and the output vector,  $(X_{out}, Y_{out})$ , is scaled to compensate for the CORDIC scale factor.

$X_{in}$  : "0010110101"  $\Rightarrow$  00.10110101  $\Rightarrow$  0.707

$Y_{in}$  : "0001000000"  $\Rightarrow$  00.01000000  $\Rightarrow$  0.25

$Pin$  : "1100110111"  $\Rightarrow$  110.0110111  $\Rightarrow$   $-\pi/2$

$X_{out}$  : "0001000001"  $\Rightarrow$  00.01000001  $\Rightarrow$  0.25

$Y_{out}$  : "1101001100"  $\Rightarrow$  11.01001100  $\Rightarrow$  -0.707

## Vector Translation

### Rectangular to Polar Translation

When vector translational functional configuration is selected, the input to the CORDIC core is a vector  $X, Y$ . The input vector is rotated using the CORDIC algorithm until the Y component of the internal vector is zero. This generates a scaled output magnitude,  $Z_1 \cdot Mag$  and output phase  $\theta$  as shown in [Figure 5](#).

An optional scaling module is provided to compensate for the CORDIC scale factor  $Z_1$ .

An optional coarse rotation and inverse rotation module is provided to extend the input/output ranges of  $X, Y$  and phase.

The rectangular to polar translation can be implemented by selecting the vector translational function and setting the input vector  $X, Y$  as shown in [Figure 5](#).

The CORDIC vector translation function preserves magnitude; hence, the user can scale the input/output range.

Thus, if vector  $(X_{in}, Y_{in})$  is translated to  $(X_{out}, \theta)$ , then Vector  $K \cdot (X_{in}, Y_{in})$  is translated to  $(K \cdot X_{out}, \theta)$ .

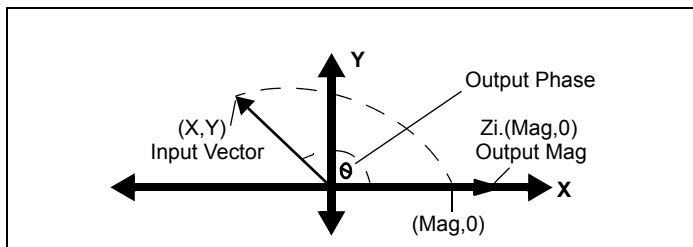


Figure 5: Vector Translation (Polar to Rectangular)

Table 2: Vector Translation I/O

Signal	Description
X_IN	Input X Coordinate Range: $-1 \leq X_{IN} \leq 1$
Y_IN	Input Y Coordinate Range: $-1 \leq Y_{IN} \leq 1$
X_OUT	Output Magnitude * Z Range: $0 \leq X_{OUT} \leq \sqrt{2}$
PHASE_OUT	Output Phase Range: $-\pi \leq \text{Phase Out} \leq \pi$

### Example 2) Vector Translation

The input vector,  $(X_{in}, Y_{in})$ , is expressed as a pair of signed 1QN format numbers. The output magnitude  $(X_{out})$ , is expressed as a signed 1QN format number. The output angle,  $\theta$  radians, is expressed as a signed 2QN number. In this example the input/output width is set to 10 bits and the output  $(X_{out})$ , is scaled to compensate for the CORDIC scale factor.

Xin : "0010110101" => 00.10110101 => 0.707

Yin : "0001000000" => 00.01000000 => 0.25

Xout : "0011000000" => 00.11000000 => 0.75

Pout : "0000101011" => 00.00101011 => -0.168

## Sin and Cos

When the Sin Cos functional configuration is selected, the CORDIC algorithm is used to rotate the unit vector  $\theta$  radians around the circle. The input to the SinCos core is the rotation angle, Phase\_in. The output from the SinCos core is the vector  $(X_{out}, Y_{out})$ .

No output scaling is required as the unit vector is internally prescaled to compensate the CORDIC scale factor.

An optional coarse rotation and inverse rotation module is provided to extend the input/output ranges of X, Y and phase.

Table 3: Sin and Cos

Signal	Description
PHASE_IN	Input Angle $\theta$ Range: $-\pi \leq \text{PHASE\_IN} \leq \pi$
X_OUT	Output $\cos(\theta)$ Range: $-1 \leq X_{OUT} \leq 1$
Y_OUT	Output $\sin(\theta)$ Range: $-1 \leq Y_{OUT} \leq 1$

### Example 3) Sin and Cos

The input (Phase\_in) is expressed as a signed 2QN number. The output vector  $(X_{out}, Y_{out})$  is expressed as a pair of signed 1QN numbers. In this example the input/output width is set to 10 bits.

PHASE\_IN : "0001100100" => 000.1100100 => 0.781

X\_OUT : "0010110101" => 00.10110101 => 0.707

Y\_OUT : "0010110101" => 00.10110100 => 0.707

## Sinh and Cosh

When the SinhCosh functional configuration is selected, the CORDIC algorithm is used to move the unit vector through hyperbolic "angle," Phase\_in, along the hyperbolic curve as shown in Figure 6. The input to the SinhCosh core is the hyperbolic "angle," Phase\_in. The hyperbolic "angle" represents the log of the area under the vector  $(X_{out}, Y_{out})$  and is unrelated to a trigonometric angle. The output from the SinhCosh core is the vector  $(X_{out}, Y_{out})$ .

No output compensation scaling is required.

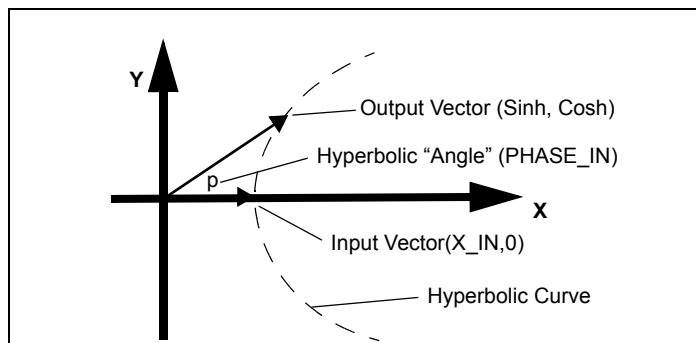


Figure 6: Hyperbolic Sinh Cosh

Table 4: Sinh and Cosh

Signal	Description
PHASE_IN	Input Hyperbolic Angle Range: $-\pi/4 \leq \text{PHASE\_IN} \leq \pi/4$
X_OUT	Output Cosh Range: $1 \leq \text{X\_OUT} < 2$
Y_OUT	Output Sinh Range: $-2 \leq \text{Y\_OUT} < 2$

**Example 4) Sinh and Cosh**

The input (Phase\_in), is expressed as a signed 2QN number. The output vector (Xout, Yout) is expressed as a pair of signed 1QN Numbers. In this example the input/output width is set to 10 bits.

PHASE\_IN : "0001001110" => 000.1001110 => 0.781

X\_OUT : "0100110001" => 01.00110001 => 1.191

Y\_OUT : "0010100110" => 00.10100110 => 0.648

**ArcTan**

When the Arc Tan functional configuration is selected, the CORDIC algorithm is used to rotate the input vector, (Xin,Yin), along the circle to (Xout,0). The output of the ArcTan core is the angle,  $\text{atan}(\text{Yin}/\text{Xin})$ , through which input vector was rotated.

No output scaling is required.

An optional coarse rotation and inverse rotation module is provided to extend the input/output ranges of X, Y and Phase.

Table 5: ArcTan

Signal	Description
X_IN	Input X Coordinate Range: $-1 \leq \text{X\_IN} \leq 1$
Y_IN	Input Y Coordinate Range: $-1 \leq \text{Y\_IN} \leq 1$
PHASE_OUT	Output Angle Range: $-\pi \leq \text{Phase Out} \leq \pi$

**Example 5) Arc Tan**

The input vector (Xin, Yin) is expressed as a pair of signed 1QN numbers. The output  $\theta$  radians, is expressed as a signed 2QN number. In this example, the input/output width is set to 10 bits.

Xin : "0010100000" => 00.10100000 => 0.625

Yin : "0010000000" => 00.10000000 => 0.500

Pout : "0010001100" => 001.0001100 => 1.094

**ArcTanh**

When the ArcTanh functional configuration is selected, the CORDIC algorithm is used to move the input vector, (Xin,Yin), along the hyperbolic curve to (Xout,0), as shown in **Figure 7**. The output of the ArcTanh core is the hyperbolic "angle,"  $\text{atanh}(\text{Yin}/\text{Xin})$ . The hyperbolic "angle" represents the log of the area under the vector (Xin,Yin) and is unrelated to a trigonometric angle.

The Yin component of the input vector, (Xin,Yin), must be less than or equal to  $4/5$  \* the Xin component, or the CORDIC approximation will not converge.

No output scaling is required.

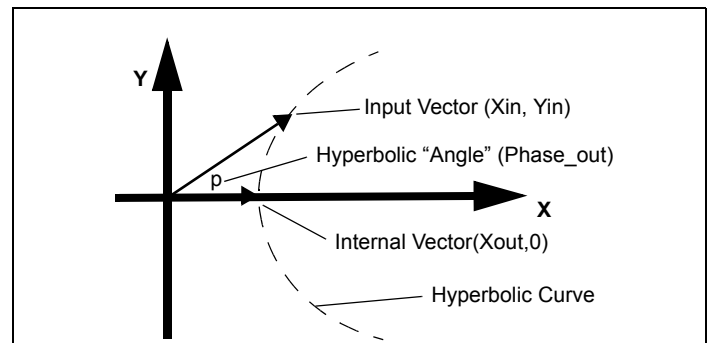


Figure 7: Hyperbolic ArcTanh

Table 6: ArcTanh

Signal	Description
X_IN	Input X Coordinate Range: $0 < \text{X\_IN} < 2$
Y_IN	Input Y Coordinate Range: $-2 \leq \text{Y\_IN} \leq 2$ $-\text{X\_IN} * 4/5 \leq \text{Y\_IN} \leq \text{X\_IN} * 4/5$
PHASE_OUT	Output Angle Range: $-\pi/2 \leq \text{Phase Out} \leq \pi/2$



### Example 6) Arc Tanh

The input vector (Xin, Yin) is expressed as a pair of signed 1QN numbers. The output,  $\theta$  radians, is expressed as a signed 2QN number. In this example, the input/output width is set to 10 bits.

Xin : "0001100101" => 00.01100101 => 0.395

Yin : "0001100101" => 00.01100101 => 0.395

Pout : "0001110001" => 000.1110001 => 0.883

### Square Root

When the square root functional configuration is positive number, Xin, and the output is a positive number, Xout.

Two data formats are available, "Unsigned Fraction" and "Unsigned Integer." A simplified CORDIC algorithm is used to generate the square root. CORDIC algorithm-related output scaling is not required.

Table 7: Square Root

Signal	Description
X_IN	Input X Value Range: $0 \leq X\_IN$
X_OUT	Output Square Root Range: $0 \leq X\_OUT$

### Example 7a): Square Root - Unsigned Fraction

The input (Xin), is expressed as a unsigned 1QN format fraction. The output scalar, (Xout), is expressed as an unsigned 1QN format fraction. In this example, the input/output width is set to 10 bits.

Xin : "0000100000" => 0.000100000 => 1/16

Xout : "0010000000" => 0.010000000 => 1/4

### Example 7b): Square Root - Unsigned Integer

The input (Xin), is expressed as an unsigned integer. The output scalar, (Xout), is expressed as an unsigned integer. In this example, the input width is set to 10 bits. The output width must be set to 6 bits.

Xin : "0000100000" => 32

Xout : "000110" => 6

### Architectural Configuration

Two architectural configurations are available for the CORDIC core: Parallel, with single-cycle data throughput and large silicon area, and Word Serial, with multiple-cycle throughput and a smaller silicon area.

### Word Serial Architectural Configuration

The CORDIC algorithm requires approximately one shift-addsub operation for each bit of accuracy. A CORDIC core implemented with the word serial architectural configuration, implements these shift-addsub operations serially, using a single shift-addsub stage and feeding back the output.

A word serial CORDIC core with N bit output width has a latency of N cycles and produces a new output every N cycles. The implementation size of a word serial CORDIC core is directly proportional to the Precision.

### Parallel Architectural Configuration

The CORDIC algorithm requires approximately one shift-addsub operation for each bit of accuracy. A CORDIC core with a parallel architectural configuration implements these shift-addsub operations in parallel using an array of shift-addsub stages.

A parallel CORDIC core with N bit output width has a latency of N cycles and produces a new output every cycle. The implementation size of a parallel CORDIC core is directly proportional to the Precision \* the Iterations.

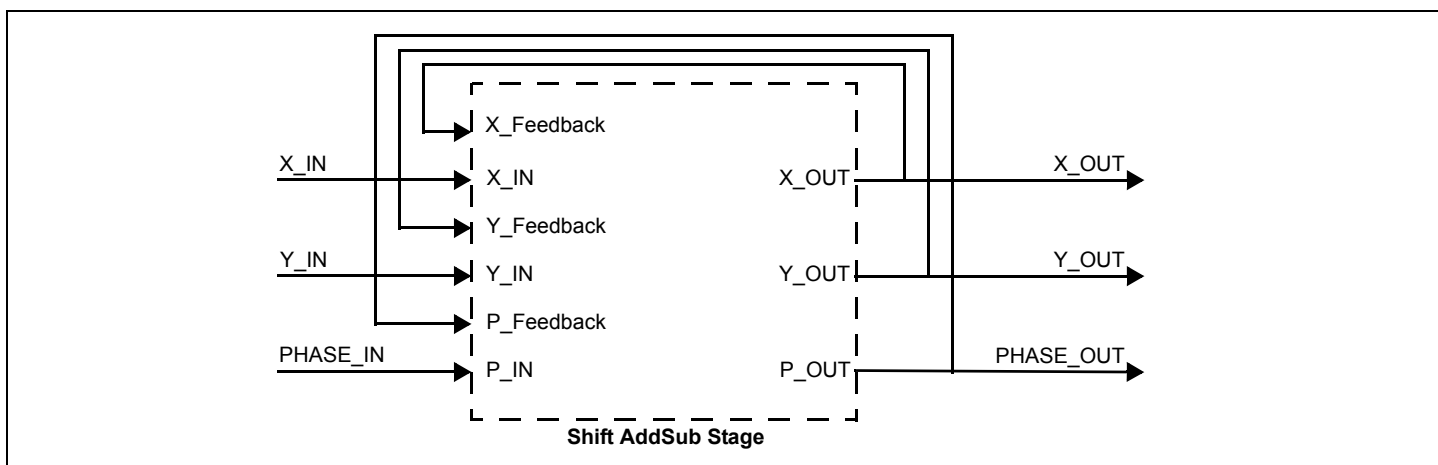


Figure 8: Word Serial Architectural Configuration

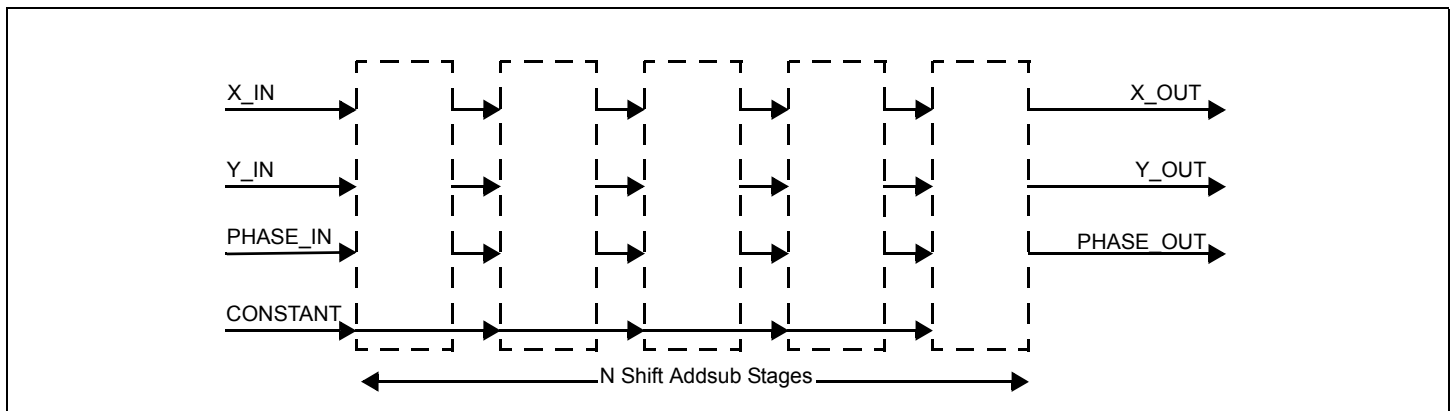


Figure 9: Parallel Architectural Configuration

## Pipelining Mode

The CORDIC core has three pipelining modes available, “None,” “Optimal,” and “Maximum.” The choice of pipelining mode is restricted according to the selection of Functional Configuration and Architectural Configuration. Unavailable pipelining modes are greyed out.

### Pipeline Mode: None

The CORDIC core is implemented without pipelining.

### Pipeline Mode: Optimal

The CORDIC core is implemented with as many stages of pipelining as possible without using any additional LUTs.

### Pipeline Mode: Maximum

The CORDIC core is implemented with a pipeline after every shift-addsub stage.

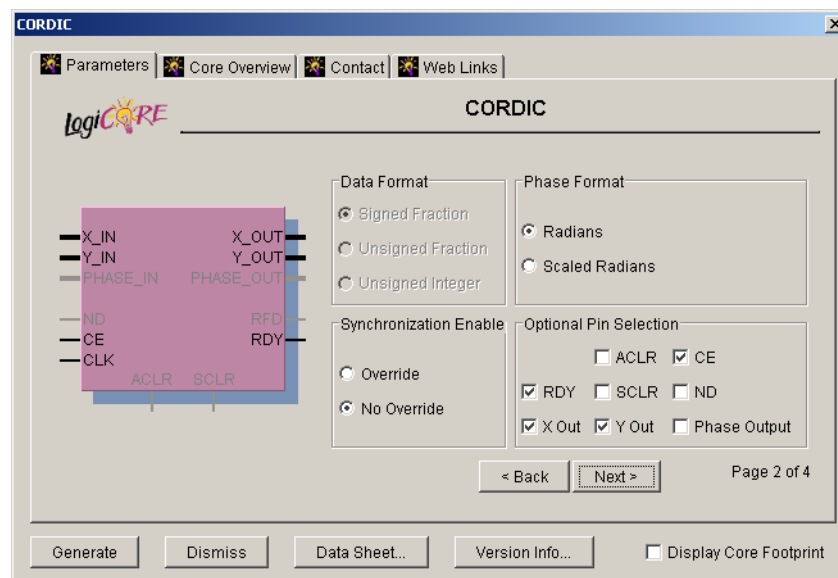


Figure 10: CORDIC V2\_0 Configuration GUI (Page 2)

## Data Format

The CORDIC core has three data formats available to express the X and Y components of data samples, “Signed Fraction,” “Unsigned Fraction,” and “Unsigned Integer.”

In general the data format is always set to “Signed Fraction.” For the “Square Root” functional configuration data formats,

“Unsigned Fraction” and “Unsigned Integer” are available instead of “Signed Fraction.”

### Signed Fraction

The x and y inputs and outputs express signed fractions of Q1 format.

Example: “11100000” represents the value - 0.5.



## Unsigned Fraction

The x and y inputs and outputs express unsigned fractions of Q1 format.

Example: "11100000" represents the value +1.75.

## Unsigned Integer

The x and y inputs and outputs express unsigned integers.

Example: "11100000" represents the value +224.

## Phase Format

The CORDIC core has two formats available to express phase, "Radians" and "Scaled Radians." The phase format is selected using radio buttons.

### Radians

The phase input and output express signed fractions of Q2 format in radians units.

Example: "01100000" represents the value 3.0 Radians.

### Scaled Radians

The phase input and output express signed fractions of Q2 format with pi-radian units. One scaled-radian is equal to  $\pi * 1$  radians.

Example: "11110000" represents the value  $-0.5 * \pi$  radians.

## Synchronization Enable

The Sync Enable parameter has two states, "Override" if CE overrides the SCLR signal, and "No Override" if CE does not override the SCLR signal.

If Sync Enable = "Override" then SCLR is ignored when CE is deasserted.

"No Override" is selected by default as this results in a more efficient implementation.

## Pin Selection

### Control Signals

The following control signals are optional: ND, RDY, ACLR, SCLR, and CE.

The presence of the control signals, CLK and RFD are determined automatically based on the Architectural Configuration, Pipelining Mode, Register Inputs and Register Outputs.

### Output Ports

The following output signals: X\_OUT, Y\_OUT and PHASE\_OUT, are optional. The default states for the corresponding check boxes are determined automatically based on the functional configuration.

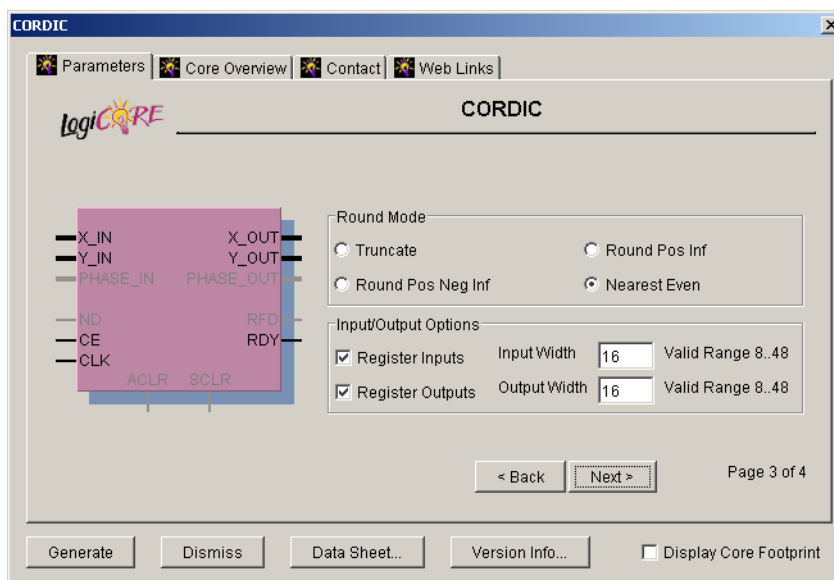


Figure 11: CORDIC V2\_0 Configuration GUI (Page 3)

## Round Mode

The CORDIC core has four output rounding modes available: "Truncate," "Positive Infinity," "Pos Neg Infinity," and "Nearest Even."

### Truncate

The outputs, X\_OUT, Y\_OUT, PHASE\_OUT, are truncated.

### Round Positive Infinity (Round Up)

The outputs, X\_OUT, Y\_OUT, PHASE\_OUT, are rounded (1/2 rounded up).

### Round Pos Neg Infinity (Round Away From Zero)

The outputs, X\_OUT, Y\_OUT, PHASE\_OUT, are rounded (1/2 rounded up, -1/2 rounded down).

### Round To Nearest Even

The outputs, X\_OUT, Y\_OUT, PHASE\_OUT, are rounded towards the nearest even number (1/2 rounded down and 3/2 is rounded up).

Table 8: Rounding Modes

	Truncate	Round+/-	Round +	Even
1.50	1	2	2	2
1.00	1	1	1	1
0.50	0	1	1	0
0.25	0	0	0	0
0.00	0	0	0	0
- 0.25	-1	0	0	0
- 0.50	-1	-1	0	-1
- 0.75	-1	-1	-1	-1

## Input Options

### Register Inputs

When the Register inputs box is checked, the input signals, X\_IN, Y\_IN, PHASE\_IN and ND are registered.

### Register Outputs

When the Register outputs box is checked, the output signals, X\_OUT, Y\_OUT, PHASE\_OUT, RFD and RDY are registered.

### Input Width

Input Width controls the widths of the input ports, X\_IN, Y\_IN and PHASE\_IN. The Input Width can be configured in the range 8 to 48 bits.

### Output Width

Output Width controls the widths of the output ports, X\_OUT, Y\_OUT, PHASE\_OUT. The Output Width can be configured in the range 8 to 48 bits.

## Advanced Configuration Parameters

### Iterations

Iterations configures the number of internal add-sub iterations to perform.

If Iterations is set to zero the number of iterations to be performed is determined automatically based on the required accuracy of the output.

By default Iterations is always set to zero thus the number of iterations is automatically determined.

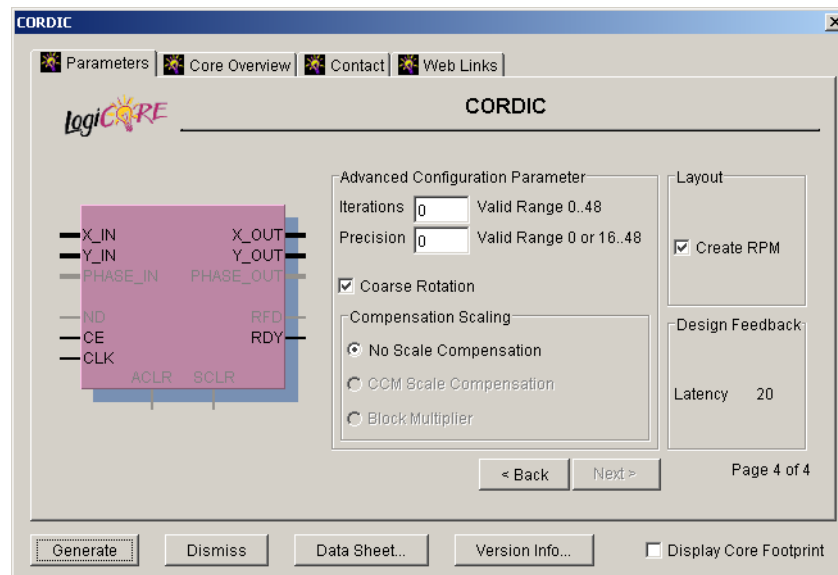


Figure 12: CORDIC V2\_0 Configuration GUI (Page 4)

## Precision

Precision configures the internal precision of the internal add-sub iterations.

If Precision is set to zero the internal precision is determined automatically based on the required accuracy of the output and the number of internal iterations.

By default Precision is always set to zero thus the internal precision is automatically determined.

## Coarse Rotation

Coarse Rotation controls the instantiation of the coarse rotation module. The coarse rotation module rotates the input sample into the first quadrant and inverse rotates the output into the correct quadrant. Coarse Rotation extends the CORDIC operational range to the full circle.

Instantiation of the coarse rotation module is the default for the following Functional Configurations: Vector rotation, Vector translation, Sin and Cos, and Arc Tan. If Coarse Rotation is turned off for these functions the input/output range is limited to the first quadrant ( $-\pi/4$  to  $+\pi/4$ )

Coarse Rotation is not required for the following functional configurations: Sinh and Cosh, Arctanh and Square Root.

## Compensation Scaling

The output values generated using the CORDIC core, Xout and Yout, are scaled by the CORDIC scale factor,  $Z_i$  as a side effect of the CORDIC algorithm. (For an explanation of this see [The CORDIC Scale Factor](#)).

## Functional Configurations affected by CORDIC Magnitude Scaling

Functional Configurations Vector Rotation and Vector Translation are affected by CORDIC magnitude scaling.

If CORDIC Compensation Scaling is set to "No Scale Compensation" then the magnitude of output values, Xout and Yout, shall be scaled by a ratio of approximately 1.165. The exact ratio of CORDIC magnitude scaling depends on the number of iterations performed by the CORDIC algorithm.

If CORDIC compensation scaling is set to "CCM Scale Compensation," the magnitude compensation shall be performed using a LUT based Constant Coefficient Multiplier.

If CORDIC compensation scaling is set to "Block Multiplier" the magnitude compensation shall be performed using the built-in Block Multiplier. Note "Block Multiplier" mode is only available when the project's target architecture has built-in multipliers (Virtex-II, Virtex-II Pro and Spartan-3)

## Functional Configurations not affected by CORDIC Magnitude Scaling

CORDIC functional configurations: SinCos, SinhCosh, ArcTan, ArcTanh and Square Root, are not affected by CORDIC magnitude scaling.

For these functional configurations CORDIC compensation scaling is always set to "No Scale Compensation."

## Layout

### Create RPM

When the Create RPM box is checked, the core will be generated with relative location attributes attached.

Note that when a core is created as an RPM (relationally placed macro) it is possible that one or more of the core dimensions may exceed those of the device being targeted. If this is the case, mapping errors will occur and the compilation process will fail. In this case, the core should be regenerated with the Create RPM checkbox unchecked.

## Design Feedback

### Latency

Latency configures the latency of the design with the currently selected parameters. Latency is defined as the number of clock cycles between the input being sampled and the corresponding sample appearing on the outputs.

## GUI Control

### Generate

Generates the CORDIC core with the currently configured parameters.

### Dismiss

Cancels generation of the CORDIC and returns the user to the main page of the Core generator tool.

### Datasheet

Brings up a pdf document of the CORDIC data sheet.

### Version Info

Brings up a Popup window with the New Features and Bug Fixes for the Cordic V2.0.

### Display Core Footprint

When the core is generated if the "Display Core Footprint" box is checked the Footprint View popup-window appears. This popup-window contains a graphical representation of the placement of the generated core. Note if the Create RPM is deselected then this popup-window will be empty.

## Physical Description

### Pinout

Table 9: Pin Descriptions

Port Name	Port Width	Direction	Has Pin	Description
X_IN	Input_Width	IN	Dependent	X component of input sample
Y_IN	Input_Width	IN	Dependent	Y component of input sample
PHASE_IN	Input_Width	IN	Dependent	Phase component of input sample
X_OUT	Output_Width	OUT	Optional	X component of output sample
Y_OUT	Output_Width	OUT	Optional	Y component of output sample
PHASE_OUT	Output_Width	OUT	Optional	Phase component of output sample
ND	1	IN	Dependent	New sample on input ports
RFD	1	OUT	Dependent	Ready for new data sample
RDY	1	OUT	Dependent	New output data is ready
CLK	1	IN	Dependent	Clock
CE	1	IN	Optional	Clock enable
ACLR	1	IN	Optional	Asynchronous clear
SCLR	1	IN	Optional	Synchronous clear

### CORDIC Data Inputs

X\_IN, Y\_IN and PHASE\_IN are the data input ports for the CORDIC Core. All data input ports are read simultaneously to form a single input sample. The width of the data input ports is configured using the parameter "Input Width." The data input ports are optionally registered. The data input ports required for a particular Functional Configuration are automatically determined by the GUI as shown in Table 10.

#### "X\_IN" Data Input Bus

The X\_IN data input bus contains the X component of the input data sample.

#### "Y\_IN" Data Input Bus

The Y\_IN data input bus contains the Y component of the input data sample.

#### "PHASE\_IN" Data Input Bus

The PHASE\_IN data input bus contains the phase component of the input data sample.

### CORDIC Data Outputs

X\_OUT, Y\_OUT and P\_OUT are the data output ports for the CORDIC core. The width of the CORDIC data output ports is set using the parameter "Output Width." The data output ports are optionally registered. The default settings for data input ports required for a particular Functional Configuration are automatically determined by the GUI as

shown in Table 10. The output data ports are optional and may be modified from the default settings.

#### X\_OUT Data Output Bus

The X\_OUT data output bus contains the X component of the output data sample.

#### Y\_OUT Data Output Bus

The Y\_OUT data output bus contains the Y component of the output data sample.

#### P\_OUT Data Output Bus

The P\_Out data output bus contains the phase component of the output data sample.

Table 10: Input/Output Pins vs. Functional Configuration

	XIN	YIN	PIN	XOUT	YOUT	POUT
Rotate	1	1	1	1	1	0
Translate	1	1	0	1	0	1
Sin Cos	0	0	1	1	1	0
Arc Tan	1	1	0	0	0	1
Sinh Cosh	0	0	1	1	1	0
Arc Tanh	1	1	0	0	0	1
Sq. Root	1	0	0	1	0	0

## Data Input/Output Representation

Data Signals X\_IN, Y\_IN, X\_OUT and Y\_OUT are expressed in 1QN format.

Data Signals PHASE\_IN and PHASE\_OUT are expressed in 2QN format.

### Q Format Unsigned Numbers

A QN format number is an N bit 2's complement binary number; a sign bit followed by an N bit mantissa (fraction). QN format can be used to express numbers in the range -1 to  $(1 - 2^{-n})$ .

An XQN format number is a QN format number left shifted by X bits. XQN format can be used to express numbers in the

range  $2^X \times (-1 \text{ to } (1 - 2^{-n}))$  or  $-2^X \text{ to } (2^X - 2^{X-n})$ .

Examples of XQN Format Numbers

Table 11: 1QN Format Data

	SB	D8	D7	D6	D5	D4	D3	D2	D1
+1	0	1	0	0	0	0	0	0	0
-1	1	1	0	0	0	0	0	0	0
+Pi/4	0	0	1	1	0	0	1	0	0
-Pi/4	1	1	0	0	1	1	0	1	1
^ <---Binary Point									

Table 12: 2QN Format Data

	SB	D8	D7	D6	D5	D4	D3	D2	D1
+1	0	0	1	0	0	0	0	0	0
-1	1	1	1	0	0	0	0	0	0
+Pi	0	1	1	0	0	1	0	0	1
-Pi	1	0	0	1	1	0	1	1	1
^ <---Binary Point									

## Control Signals

The following section describes the control signals used by the CORDIC core. All control signals are synchronous to the rising edge of CLK, except ACLR.

A timing diagram for a CORDIC core with a Word Serial Architectural Configuration is shown in Figure 13.

A timing diagram for a CORDIC core with a Parallel Architectural Configuration is shown in Figure 14.

## Optional Control Pins

### CLK

CLK (Clock). All core operation is synchronous with the rising edge of the CLK input, except the optional ACLR input.

The CLK signal is mandatory when ((Pipeline Mode != Pipeline None) or (Register Inputs = True) or (Register Outputs = True)). Otherwise, the CLK signal is not present.

### ND

When the ND (New Data) input is high, the input data is sampled on the same rising clock edge. ND is ignored if CE is low or if RFD is low.

The ND signal is mandatory when (Architectural Configuration = Word Serial). Otherwise, the ND signal is optional.

### RFD

RFD (Ready for Data) indicates that the core is ready to sample new input data. The RFD signal is set high upon startup or during reset.

The RFD signal is mandatory when (Architectural Configuration = Word Serial). Otherwise, the RFD signal is not present.

### RDY

The RDY (Ready) output signals that a new valid data sample is present on the Data Output Ports. RDY is pulsed high on the first clock cycle of valid data at the output. The RDY signal is set low upon startup or during reset.

The RDY signal is mandatory when (Architectural Configuration = Word Serial). Otherwise, the RDY signal is optional.

### ACLR

All control signals are synchronous to the rising edge of CLK except ACLR. When ACLR is asserted (High), all the core flip-flops are asynchronously initialized. The core will remain in this state until ACLR is deasserted.

The ACLR signal is optional.

### SCLR

When SCLR is asserted (High), all the core flip-flops are synchronously initialized. The core will remain in this state until SCLR is deasserted.

The SCLR signal is optional.

### CE

When CE (Clock Enable) is Low, all the synchronous inputs are ignored and the core remains in its current state.

The CE signal is optional.

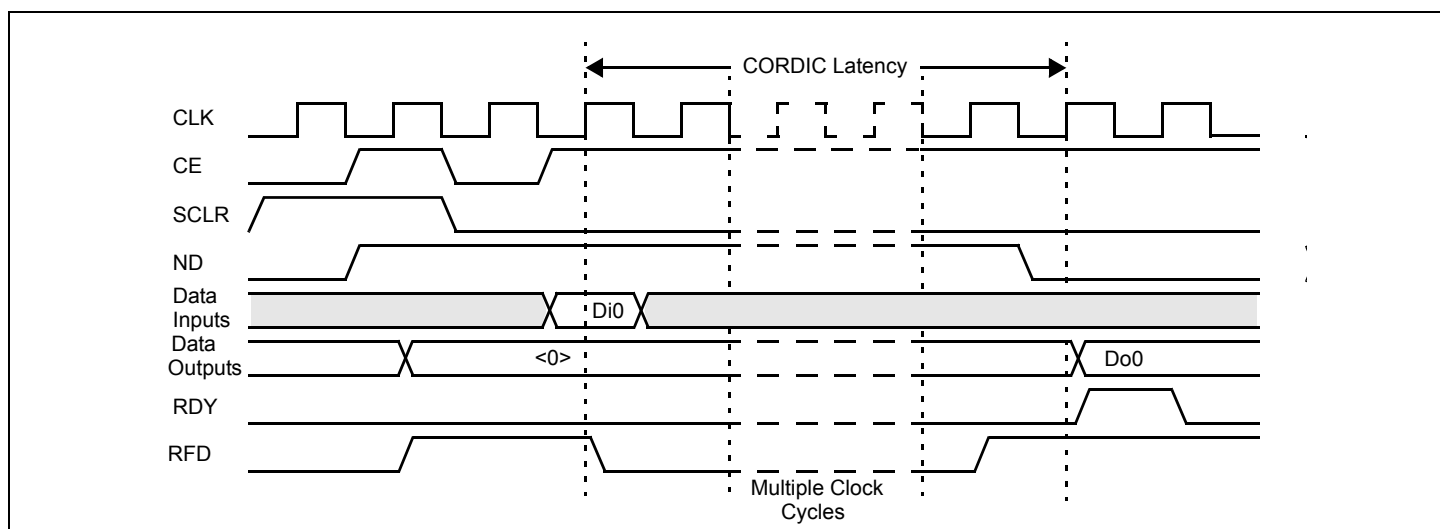


Figure 13: Control Signal Timing Diagram (Word Serial Architecture)

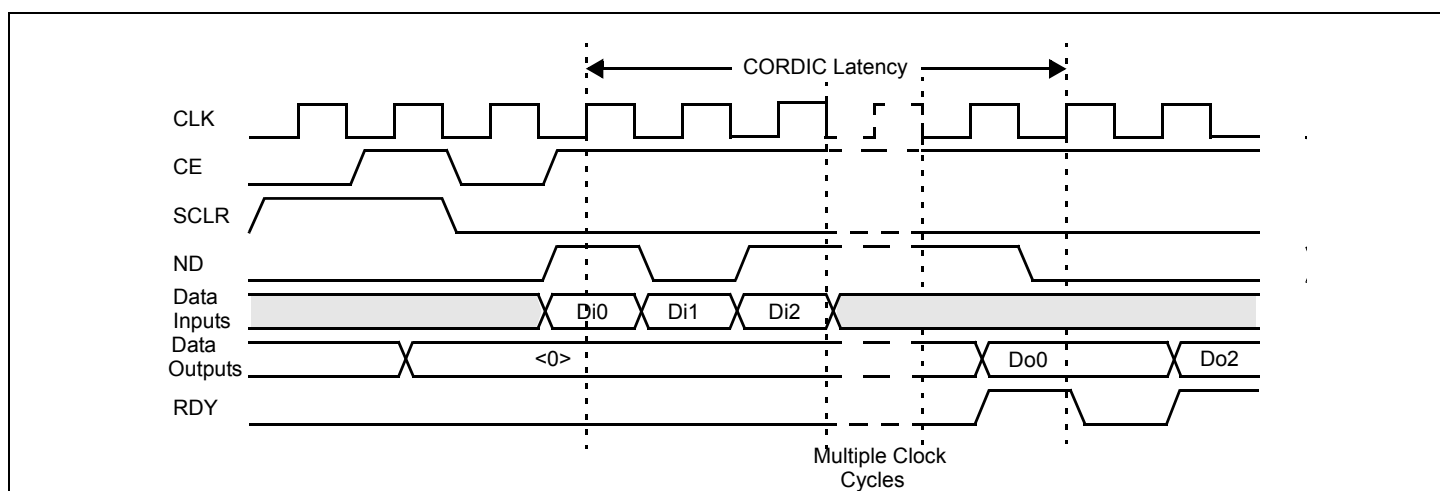


Figure 14: Control Signal Timing Diagram (Parallel Architecture)

## Characterization Data

The Characterization Data in [Table 13](#) and [Table 14](#) is was generated for CORDIC Cores with 16 bit Input and Output widths, automatically determined Iterations and Precision, Coarse Rotation, no Compensation Scaling and Maximum Pipelining.

Table 13: Characterization Data - Parallel Architecture

Parallel	Rotate	Translate	Sin Cos	ArcTan	SinhCosh	Arctanh	Sqrt Frac	Sqrt Int
Size	680	613	613	571	660	625	235	103
Speed in MHz	170	164	164	165	168	160	207	227

Table 14: Characterization Data - Word Serial Architecture

Word Serial	Rotate	Translate	Sin Cos	ArcTan	SinhCosh	Arctanh	Sqrt Frac	Sqrt Int
Size	366	300	299	279	273	279		
Speed	133	134	134	134	114	116		



## Parameter Values in the XCO File

Names of the XCO parameters and their parameter values correspond to the names and values shown in the GUI. Underscore characters (\_) are used instead of spaces.

The text in an XCO file is case insensitive. shows the XCO file parameters and values, and summarizes the GUI defaults. **Figure 15** is an example of the CSET parameters in an XCO file.

```
CSET synchronization_enable = No_Override
CSET create_rpm = true
CSET register_inputs = true
CSET sclr = false
CSET architectural_configuration = Parallel
CSET phase_format = Radians
CSET nd = false
CSET data_format = SignedFraction
CSET register_outputs = true
CSET rdy = true
CSET aclr = false
CSET ce = true
CSET y_out = true
CSET component_name = cordic_rotate
CSET pipelining_mode = Maximum
CSET compensation_scaling = No_Scale_Compensation
CSET input_width = 16
CSET round_mode = Round_Nearest_Even
CSET iterations = 0
CSET precision = 0
CSET functional_selection = Rotate
CSET output_width = 16
CSET phase_output = false
CSET x_out = true
CSET coarse_rotation = true
```

*Figure 15: Example CSET Parameters From XCO File*

**Table 15: Parameter File Information**

Parameter Name	XCO Filename Values	Default GUI Setting
aclr	True, False	False
architectural_configuration	Word_Serial, Parallel	Parallel
ce	True, False	True
compensation_scaling	no_scale_compensation, ccm_scale_compensation, block_scale_compensation	no_scale_compensation
component_name	ASCII text starting with a letter and based upon the following character set: a..z, 0..9 and _	blank

Table 15: Parameter File Information (Continued)

Parameter Name	XCO Filename Values	Default GUI Setting
coarse_rotation	True, False	True
create_rpm	True, False	True
data_format	SignedFraction, UnsignedFraction, UnsignedInteger	SignedFraction
functional_selection	Rotate, Translate, Sin_and_Cos, Sinh_and_Cosh, Atan, Atanh, Square_Root	Rotate
input_width	8 to 48	16
iterations	0 to 48	0
nd	True, False	False
output_width	8 to 48	16
phase_format	Radians, Scaled_Radians	Radians
phase_out	True, False	False
pipeline_mode	minimum, optimum, maximum	maximum
precision	0 or output_width to 48	0
rdy	True, False	True
register_inputs	True, False	True
register_outputs	True, False	True
rfd	True, False	False
round_mode	Round_Truncate, Round_Pos_Inf, Round_Pos_Neg_Inf, Round_Nearest_Even	Round_Pos_Neg_Inf
sclr	true, false	false
synchronization_enable	No_Override, Override	No_Override
x_out	True, False	True
y_out	True, False	True

## Background Information

### References

1. Volder, J., "The CORDIC Trigonometric Computing Technique" IRE Trans. Electronic Computing, Vol. EC-8, Sept. 1959, pp330-334
2. Walther, J.S., "A Unified Algorithm for Elementary Functions," Spring Joint computer conf., 1971, proc., pp379-385

### Ordering Information

This core may be downloaded from the Xilinx [IP Center](#) for use with the Xilinx CORE Generator System v5.2i and later. The Xilinx CORE Generator System tool is bundled with all Alliance Series Software packages, at no additional charge.

To order Xilinx software, please visit the Xilinx [Silicon Xpresso Cafe](#) or contact your local Xilinx [sales representative](#).

Information on additional Xilinx LogiCORE modules is available on the Xilinx [IP Center](#).

---

---

## Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/28/03	1.0	Revision History added to document.
03/28/03	1.1	Fixed Hyperbolic Transformations. Fixed PiRadian Format
03/28/03	2.0	Improved parameterization. New rounding modes. New Data Formats.